ESPSS

European Space Propulsion System Simulation

EcosimPro Libraries User Manual

(VOLUME 1)

Doc.: 4000103800/11/NL/CP – TN4130	
ESPSS Version: 3.0	Date: 30-11-2013

Client :	ESA
Contract:	ESPSS-3: 4000103800/11/NL/CP
Main Contractor	Empresarios Agrupados
Contributors	Astrium / Cenaero / Kopoos / VKI

ABSTRACT:

The European Space Propulsion System Simulation (ESPSS) is an ESA initiative that aims to create a European simulation platform for spacecraft and launch vehicle propulsion systems under transient and steady (design & off-design) conditions.

This document contains the User Manuals for the ESPSS-3 Libraries (FLUID_PROPERTIES, FLUID_FLOW_1D, TANKS, TURBO_MACHINARY, COMB_CHAMBERS, *STEADY,* COMP_DATABASE, SATELLITE and ELECTRICAL PROPULSION) provided with ***EcosimPro 5.0 and 5.2.***

Separate chapters are dedicated to each Library and contain an overview (recommendations and tips), a detailed description of the components and the most important aspects of the physical modeling for each of the components. Chapter 9 is devoted to demonstrating ESPSS capabilities when coupled to external software. Two applications are selected: zooming and system optimization.

ESPSS-3 includes new features such as the simulation of solid/hybrid combustors, ramjets, scramjets and the convection/mixing of combusted gases downstream of a chamber. It also includes new libraries for steady/quasi-steady models: the STEADY Library for designing rocket engine cycles, the SATELLITE Library for orbital and attitude motion and the ELECTRICAL_PROPULSION Library. New useful tips and application examples have been added to help the user define new models.

Revision table

Version	Modifications	Paragraphs
V1.0 (31-01-08)	*First Version*	
V1.0.1 (19-03-08)	○ Added Real fluids MMH, N2O4 and Hydrazine ○ Added an EXCEL tool for plotting the fluid properties ○ New comments on the FLUID_FLOW_1D and COMB_CHAMBERS libraries ○ Subsonic velocities in a combustor are now calculated using adjacent junction densities ○ Updated paragraph 9.3.3, Table 9.3 and Fig. 9.4	
V1.4.1 (29-05-09)	Added the new Chapter "How to build a model" and multiple comments and tips in the library's Overview Sections	§3, …
	THERMO_TABLE_INTERP FORTRAN library:	
	○ Using the CEA enthalpy reference at combustor components level	§4.4.6
	○ The behavior of simplified liquids under two-phase conditions (zero pressure) has been made more robust	--
	○ Improving the extrapolation techniques under two-phase flow at P<Ptriple	§4.4.5.5
	○ 2D properties table interpolation: Search algorithms are improved. Extrapolation under solid conditions has been improved eliminating bugs near the triple point where the curves have only one point	--
	FLUID_PROPERTIES library:	
	○ Storing the CEA coefficients as constant data outside the EL functions.	--
	○ CEA coefficients for the carbon atom element have been replaced. Before C_gr was considered. The temperature limits for the CEA coefficients of the H, O, OH atoms have been corrected	(software bug fixes)
	○ New constant Cp and formation enthalpy (extracted from the real properties files) have been included for the MMH/UDMH vapors	--
	○ Homogeneous equilibrium model: Programming is rearranged speeding up the iterative processes. With non-condensable gases Tsat is assumed to be zero because two phase correlations do not work	--
	○ The expression evaluating the sub cooled critical mass flow has been extended to Perfect Liquids, supercritical pressures and liquid / non-condensable mixtures and protected under perfect gas conditions.	A1
	○ New friction (Friedel) correlations are used in case of two-phase flow	A2.3
	○ Density derivatives (drho_dp) are approximated by the gas phase values in case of two-phase two fluid mixtures.	--
	FLUID_FLOW_1D library:	
	○ New internal flags are used to initialize the fluid capacities in the DISCRETE block, so that the RESTORE_STATE function is available	--
	○ The viscosity value has been added to the port variables allowing upgrading of the filter component and future improvements	§5.2.2.1
	○ The AbstractJunctionLoss component is rearranged to allow inheritance of a fluid separator component and to speed up the calculation	--
	○ Improved Roe scheme performances (Pipe components). Ghost values calculated in the Ports_Handling routine are made equal to the nearest cell values	Appendices A9-10
	○ The work of the gravity forces is included in the conservation equations	§5.3.8.2

	○ Lateral Tee branches with direction angle can be simulated	§5.3.2, 5.4.14
	○ Using the centred scheme, the new input data *CAV_DAMP* allows better simulation of a liquid front travelling along the pipe	§5.1.3.6
	○ New Cold Thruster component based on the resistive pipe including adapted equations for subsonic/supersonic exit conditions	§5.4.10
	○ New Tube_HP component inherited from the ABS_tube where the wet perimeter and the cross section are input	§5.4.15
	○ Pressure regulator components have been upgraded, taking into account the stroke characteristics.	§5.4.18 §5.4.19
	○ Added diffusive and conductive flows to the junction equations preventing numerical instabilities at zero mass flow	§5.3.1.1
	TANKS library:	
	○ New messages concerning inappropriate inputs in junction elevations connected to the Tanks, or when the liquid volume becomes gasified …	(software changes)
	○ Corrected a bug in the 2D wall thermal model (in case of constant conductivity, k variable was confused with an index)	"
	○ Tank_CylDomesSph component: the thermal connection of the Semi-sphere with the lower dome is reversed (reversed concavity)	"
	○ Simplified Tanks can work under negative gravities.	§6.1.2
	○ The total mass inside the Tanks is now calculated explicitly	--
	○ The heat exchange correlations are upgraded at the gas/liquid interface	--
	○ Boiling/condensation mass flows equations are upgraded. The old formulation has been extended to any discretized tank fluid volume. Condensation mass flows are switched off for the moment	§6.2.2.1
	○ The liquid side can be initialized in gas conditions to allow simulation of filling processes	§6.1.2.2
	○ A new component (CheckValve_FlSep) has been added allowing a common pressurizing circuit of several tanks with different liquids	§6.3.5
	TURBO_MACHINERY library:	
	○ Performance maps: The spline interpolation is changed to linear interpolation to prevent incorrect values at the corners of the maps	(software changes)
	○ New flag added to calculate the sound speed at the inlet or at the outlet of compressors and turbines	§7.2.1.4, etc.
	○ New Compressor_gen component working with generalized (built-in) non dimensional maps	§7.2.1
	COMB_CHAMBERS library:	
	○ dH_inj, dH_inj_rate functions are simplified. CombProper_PH is replaced by the CombProper_eq functions where the vapor and the non-condensable mass fractions are used as inputs	(software changes)
	○ Combustor_eq component. New conservation equations for the vapor mixture have been implemented allowing robust and simple simulation of liquid injections.	§8.1.2, §8.2.3
	○ *Combustor_rate* component: A new model for liquid droplet evaporation has been implemented with automatic limitation of the vaporization factors when the liquid temperature is low	§8.1.2, §8.2.4
	○ Coherent simulation of the solid propellant gases of the starter (assuming they have the same composition as the non-condensable gases). Non-condensable gases have a new mixture equation	§8.2.3.4
	○ *Preburners*: The outlet mass flow is now solved by taking the formulation of Junction components (physical inertia)	§8.2.5.1
	○ *Nozzle* component: Equilibrium or frozen condition is now an option for the	§8.2.8

	supersonic sections of a nozzle. The nozzle mass flow is calculated without dependency on the Nozzle option (ideal or non-isentropic).	
	○ CoolingJacket component: Wall ports are now capacitive, thus eliminating some algebraic loops using Bartz correlations.	(software changes)
	○ The nonlinear boxes (pressure calculation) of the combustor component inside the "CombProper_eq" functions are hidden.	"
	○ The size of the 1D mesh can now be introduced directly in the combustor, nozzle or CR components	§8.1.2.2
	○ Improved initializations of the state variables in accordance with the selected fluids for the injectors	(software changes)
V2.0.0 (01-02-10)	*This Version corresponds to the final delivery of ESPSS-II and Version 4.6 of EcosimPro*	
	THERMO_TABLE_INTERP FORTRAN library:	
	○ The behavior of simplified liquids under two-phase conditions (zero pressure) has been made more robust by adding a very low (but not null) density to the vapors	(software changes) --
	○ The external environment variable PROP_TABLES is no longer needed (the argument list of FORTRAN functions has changed)	§4.1.3
	○ CEA coefficients are no longer stored in the sources: They will be read from the CEA properties files (thermo.lib, transport.lib) at execution time	§4.1.3
	○ CEA properties interpolation functions have been moved to FORTRAN using those of CEA program	(software changes)
	FLUID_PROPERTIES library:	
	○ To facilitate the inclusion of new fluids, properties files and fluid names are included as global variables to be transferred by argument to the FORTRAN routines	§4.1.3 and §4.3.4
	○ Under two-phase, two-fluid flow and *P lower than Ptriple*, the vapor pressure is limited to positive values; vapor density is extrapolated at T=Ttriple assuming ideal gas; saturated properties (except energy) are fixed at Ttriple	(software changes)
	○ New EXCEL tool generating files of Peng-Robinson real fluid properties. Hydrazine and MMH fluids have been upgraded using this tool.	§4.3.3.1.2
	○ Two-phase sound speed is calculated approximately from the one-phase values assuring continuity at phase change using the Walls correlation.	§4.4.5.4
	FLUID_FLOW_1D Library:	
	○ Laminar pressure drop is calculated from a laminar Reynolds number: Re_lam replace dp_lam input data	§5.3.2.2
	○ Critical mass flow (sub cooled, two-phase or gas) can now be calculated exactly for pure fluids	§5.3.4 and A1
	○ Pressure drop coefficient can be calculated automatically from the orifice area and the adjacent connected flow areas	§5.4.6.5
	○ The NonAdiabaticVol abstract component is restored. New cavity components (inherited from a NonAdiabaticVol) with 1 or 3 fluid ports are now available	§5.4.2
	○ New output variables containing the first and the last node values of significant variables have been added. Non useful or "confusing" variables used for programming reasons are now hidden to the user (PRIVATE specification using EcosimPro 4.6 language)	§3.4
	TANKS library:	
	○ A delay to the vaporized mass flow to improve robustness has been added.	§6.2.3.2

	○ A new component to simulate ground & flight heat fluxes to the walls of a Tank has been added	§6.3.6
	○ The component CheckValve_FlSep has been upgraded allowing real gas properties in the pressurizing circuit and in the HP reservoir	§6.3.5.3
	COMB_CHAMBERS library:	
	○ New input with the molar fractions (H2O, CO, CO2, N2, H2, He) of the starter solid propellant gases	§8.1.2.3 and §8.2.3.4
	○ Combustor efficiency: the conservation equations of combustor models are upgraded with this new capability	§8.2.3.4
	○ New Multiplexer/Demultiplexer components to allow splitting a chamber thermal port into two different CoolingJackets and vice versa	§8.1.2.2 and §8.3.1
	○ A thermal port to the Combustor components allowing heat transfer between the injector plate and the exterior has been added	§8.3.3.3 and §8.3.5.3
	○ Thermal barrier coating: New set of chamber components allowing simulation of a thermal barrier coating	§8.3.2
	○ Injector torus: new CoolingJacket component including two inlet-outlet toruses with two fluid ports at each torus	§8.3.8
	○ Film cooling: new Nozzle component allowing to simulate a film cooling injection with or without mixing equations with the main flow	§8.2.9 and §8.3.4
	○ Nozzle components: improving the Mach number calculation to avoid subsonic solutions	(software changes)
V2.0.1 (15-04-11)	Bugs related to extrapolations within the propellant database of *perfect gases* were reported and fixed.	(software changes)
	This Version updates the properties files for the following *simplified liquids*:	§4.3.2
	○ MMH, UDMH & N2H4, according to reference RD-12 (Schmidt)	
	○ IPA, according to Aiche Data Compilation and http://www.rshydro.co.uk/sound-speeds.shtml	
	○ MON1, MON3 & N2O4, according to RD-14 (USAF Propellant Handbooks) and RD-47 (Rockwell)	
V2.2.0 (01-09-11)	*This version accounts for the particular upgrades and experience acquired since Version 2.0 was released. New tips and comments have been added*	
	THERMO_TABLE_INTERP FORTRAN library:	
	○ The Fortran library is upgraded redirecting all the output messages to unit 6 (for Matlab applications). Also, the RESET_VARIABLES EcosimPro command can work properly without having to reload the properties files	(software changes)
	○ N2, He,O2, H2, CH4 real properties files are rebuilt with more points	
	○ Extrapolating in real properties at P>Pmax, T<Tmin or under two-phase flow is better detected and handled	
	FLUID_PROPERTIES library:	
	○ FluidName argument is replaced by the FluidKeys more general code. The two phase sound speed is more coherently limited to 0.01 m/s	
	○ New FL_prop_vs_rp function. Gamma property in FL_prop_vs_xy functions added	
	○ Non-condensable gases mixture calculation is more robust at very low non-condensable mass fractions.	§4.5.3.1
	FLUID_FLOW_1D Library:	
	○ The initialization blocks are upgraded to force dependence on the	(software

	combusted gases properties. More than 2 pre-burners are allowed	changes)
	○ The wall compressibility correction (source term in the mass equation) is upgraded with the corresponding energy source term	§5.3.8.1 and §5.3.8.2
	○ Pressure Regulators have been made more robust cancelling some input data (Lstroke) with no relevant importance	§5.4.18 and §5.4.19
	○ New input with the friction pipe calculation option (one or two-phase). Default option is one-phase, as for the heat exchange calculation option	§5.4.8.4 and §5.4.15.4
	○ Junction components: Gas diffusion and conductivity terms at Junction components have been cancelled	§5.3.1
	○ Junction components: Supersonic conditions are allowed as an option, and can be detected automatically	§5.3.2.4
	○ The Heat Exchanger is upgraded for cross flow arrangements accounting for several baffles and tube passes	§5.4.7
	○ New, more robust version of the Cold Thruster component using supersonic junctions (non adapted conditions can be calculated including shock inside the nozzle)	§5.4.10
	○ New resistive components for subsonic or supersonic boundary conditions imposing static pressure and temperature at Pipe exits and also Mach/velocity at Pipe inlets (intakes)	§5.4.21
	○ Valves have been upgraded to consider EqualPercentage, Linear, QuickOpening or user-defined laws	§5.4.16
	○ Fluid Cavities (not the simple Volumes) account for the volume expansion due to wall compressibility	§5.4.2
	○ The output variables P1, Pn are redefined so that the damping term qn (artificial viscosity) does not affect them.	§3.4
	○ Fluid port is upgraded with the static temperature, calculated at the connected capacitive component	§5.2.2.1
	TANKS Library:	
	○ Added new 1D tanks with 4 ports and ability to simulate two or more different inlet/outlet currents entering the tank at the same time	§6.1.1
	○ Bug in the Tank_CylDomesSph component fixed (the thermal connection of the lower dome and the semi sphere was wrong)	(software changes)
	○ Single tanks can simulate horizontal cylindrical tanks with level calculation	§6.3.3
	○ Overflowing of a tank outlet is upgraded to properly account for inlet/outlets flows of non-condensable gases	§6.2.1.4
	○ 1D Tank can work with simplified fluids and under supercritical conditions. New option with no mass exchange at the gas/liquid interface is available	§6.2.3.2
	○ Surface tension effects in 1D tanks are cancelled because the corresponding functions belong to Astrium	
	○ Condensation (raining) flow is now calculated at 1D tanks if the ullage volume is under two-phase flow	§6.2.2
	○ Tank models account for the volume expansion due to wall compressibility and dispose of a new port with the signal level	§6.3.4.4
	TURBOMACHINNERY Library:	
	○ Rearranging the calculation of generic maps with better protection under null speed or flows	(software changes)
	○ Adding the isentropic efficiency calculation for user defined compressors and turbines	"eta" out var
	○ Added new inputs to the generic turbine, compressor and pumps permitting design conditions and constant rpm option	§7.2.1.4 and §7.2.5.4

	○ Pumps: tables of loss of head vs. NPSH and compressibility effects due to pump cavitation have been included	§7.2.3.4 and §7.2.4.4
	COMB_CHAMBERS Library:	
	○ Chambers can be initialized under burning condition at a given MR and pressure	§8.3.3.4 and §8.3.5.4
	○ Bug in calculating the CF coefficient in case of ideal nozzles corrected. Ideal nozzles are now calculated at constant gamma in accordance with the ideal isentropic formulation	(software changes)
	○ Combustors: No reduction in injected cold vapors is made. Nevertheless, liquid vaporization is less if Tgas nears Ttriple	§8.2.3.3
	○ Nozzle & Combustion chambers: New input allowing modification of the aspect ratio of a given profile without changing the relative shape	§8.1.2.2,
	○ CombChambers: The external injector thermal port is connected to the thermal capacity of the injectors, not to the chamber	§8.3.5.5 and §8.3.3.5
	○ New steady combustor component calculating the gas properties and mass flow vs. propellant fluids, MR, solid propellant gases and chamber pressure	§8.3.11
	○ Reconstruction of nozzle and chamber diameters is included in the continuous block allowing geometry redefinition during a simulation	(software changes)
V2.4.2 (30-04-12)	*Version 2.4 fixes bugs found by the users and upgrades the existing libraries with new components and options. New tips and comments have been added in the UM to help the user build models*	
	THERMO_TABLE_INTERP FORTRAN library:	
	○ The Fortran library is upgraded to return entropy values of a simplified liquid. This upgrade was necessary for the calculation of the Pump efficiency in the STEADY library using simplified liquids	§4.4.4.1
	○ Saturated vapor properties are correctly extrapolated if P< Ptriple and T=Ttriple. (Cpg value was not extrapolated in V2.2)	§4.4.5.5
	○ Pressure curves (normally only one) crossing the critical one are eliminated from the properties files. The NTO file is rebuilt to improve rho-u transitions from subcritical to supercritical temperatures at P > Pcrit	§4.3.3.2
	FLUID_PROPERTIES library	
	○ New "HT_critCond"option, so the heat exchange coefficient is calculated using Pr number =1 (no dependence on cp value)	A3
	○ New "HT_boiling"option for pipes using the same correlation as in Tanks	A3 & §5.4.8.4
	○ Function FL_state_vs_ph added to allow calculating the thermodynamic state as a function of pressure and enthalpy	§4.2.1
	FLUID_FLOW_1D Library:	
	○ Junctions consider greater internal inertia(3*Diam). The ZONE statement for check-valves is replaced by an IF statement. New ZONE statement to avoid calculation of flow in closed valves.	(software changes)
	○ A bug (wrong index) defining the outlet mass flow in resistive pipes is fixed.	
	○ Tee volume is changed according to a right Tee. New input data for Y and User defined Tee types added.	5.4.14.4
	○ Re_lam is no longer a global parameter, but a specific datum for Junctions and valves	5.4.6.4
	○ CAV_DAMP is no longer an input for Pipes because it is not necessary	(software changes)
	○ Most of the port variables are made PRIVATE because they are redundant	

	with component local variables ○ New input with the nodal pressure drop coefficient for not distributed losses in Pipes	§5.4.8.4 and 5.4.15.4
	TANKS Library: ○ Film boiling bubble formulation included (as a new option) and better simulation of the generalized boiling process	§6.2.2.1
	○ Improved mass exchange options at the gas/liquid IF are renamed with more logical names (advanced →Diffusion; default → EvapPool) (Diffusion approach is not limited giving negative evaporating flows)	§6.2.3.2 and §6.3.4.4
	○ It will not be supposed that the liquid can be stuck to the top of the Tank under negative gravities.	§6.1.2.2
	○ Condensation (raining) flow is now optional: droplets can remain as a fog in the ullage volume or can drop into the liquid volume at a given speed	§6.2.2.1 and §6.3.4.4
	TURBOMACHINNERY library: ○ New "NPSH_dyn" input in Pumps: if FALSE, NPSH is calculated considering the static pressure upstream the inducer instead of the total pressure	§7.2.3.9
	○ New Pump option (tdh_correction = TRUE) so that the pump head is modified to account for highly compressible liquids	§7.2.3.8 and §7.2.3.9
	COMB_CHAMBERS library: ○ Bug fixing in mesh_size2 function: cooling-jacket axial distances are better calculated and different from the axial lengths.	(software changes)
	○ Liquid injection in cold chambers is greatly reduced. Reverse flow in pre-burners is forbidden	
	○ Default value of vaporization factors f_v[] in combustor_rate models is set to 10, which is more realistic	§8.3.3.4 and 8.3.5.4
	○ New components (for pipes and Tees downstream of a pre-burner) are available to compute the delay in the transport of the combustion products; this also permits the mixing of combusted gases and pure fluids (chamber with more than 2 injectors)	§8.3.10
V2.6.0 (30-07-12)	*This development version includes a new library for direct STEADY calculations designing cycles and accounts for new features included in the libraries COMP_DATABASE, SATELLITE and EP (Electrical Propulsion).*	
	FLUID_FLOW_1D Library: ○ The *"Gcr_exact"* option have been changed to *"Gcr_ideal"*	(software changes)
	STEADY & STEADY_EXAMPLES libraries contain a complete set of components able to calculate the performances of a cycle under design and off-design conditions. Main changes from ESA prototype (Version 1.9) are*:* ○ Design/OffDesign options have been redefined with simpler and more flexible options. Code specifications to force appropriate closing equations solving non-linear equation system added ○ Combustor formulation has been upgraded allowing the simulation of staggered cycles. Chemical equilibrium accounts for wall heat exchange. ○ Orifices: sonic flow limitation included. ○ Pumps and Turbines: off-design behavior assuming generalized maps included. Efficiency equations are rearranged in a more simple way. ○ Pipes/Tubes: new code to include off-design calculation and other improvements. ○ CoolingJacket: centred scheme in channels accounting for channel geometry and pressure losses. New counterflow cooling jacket.	New chapter §11

	○ Turbines: Dependence on the combusted gases properties (Cp=f(T)) included. ○ Components initialization is highly improved: experiments do not need particular initialisation in design mode in most of the cases. ○ OffDesign models allow generating a design partition so that only one model can be used to design and analysis modes. ○ Compressor and Intake components added.	
	COMP_DATABASE Library: ○ Creation of an electronic component database compatible with ESPSS, for characterizing existing hardware components (e.g. valves, filters, pressure regulators….).	New §11
	SATELLITE Library: ○ Calculation the orbital and attitude motion of a satellite, and to study the effect of the acceleration of the satellite on the tanks.	New §12
	EP Library: ○ Transient simulation of Electric Propulsion systems.	New §13
	Application Examples: ○ Transient simulation of Electric Propulsion systems.	New §14
V3.0.beta (29-09-13)	*This version is a preliminary release of the ESPSS project, Phase 3. It includes new features such as the simulation of solid/hybrid combustors, ramjets, scramjets and the convection/mixing of combusted gases downstream of a chamber, etc.* *The main upgrades are listed below:*	
	FLUID_PROPERTIES Library: ○ Molar fractions changed to mass fractions arguments in thermo routines. ○ A complete set of Van der Waals properties functions added. ○ Properties functions explicitly depend on the burned gases chemical compositions. Perfect gas or Van der Walls state of equation is optional. ○ ESPSS components must be initialized with the new parameter *burnerGasesOption* depending on whether combusted gases are present or not. ○ Typical solid propellant constituents and their products are included in the list of chemicals and in the combusted gases properties calculation.	Table 4-4 New §4.4.2 (software changes) §5.1.2 & input data descr. §4.1.2 and §8.3.6.4
	FLUID_FLOW_1D Library: ○ Diluted non-condensable gases are transported in volumes and Pipes. ○ Convection and mixing of the chemical constituents are now calculated by the transport equations in any component downstream of a combustor. ○ Initialization of the dynamic state variables is simpler, with no dependence on the order of the components instance. ○ New Intake component behaving like the VolPsTsVs_TMD component but including the standard atmosphere relationships and using as boundaries the Mach and altitude flight conditions. ○ Any component working with combusted gases (placed downstream of a combustor) will be automatically initialized with non-condensable gases. ○ The Roe scheme in Pipe components has been upgraded to work with supersonic flows transitions in capacitive Pipes.	§5.3.1.1 §5.1.3.6,§5.3.1.1, §5.3.3.1 (software changes) §5.4.22 §5.1.3.3 §5.4.8.4

	○ Including of absorption - desorption effects in Pipes is still under development.	
	COMB_CHAMBERS Library:	
	○ Combusted gases properties functions are re-built with a simpler and faster code. The *setPfGasConst* function (assigning current combustor gases properties to any downstream component) is NO Longer used.	(software changes).
	○ Initialization of the state variables is simpler and more powerful with no dependence on the order of the combustor instance.	' '
	○ *Combustor* components are upgraded with new options: *GasLiqOption* concerning the liquid propellant vaporization models and *rateOption* to choose equilibrium or reaction delay method.	§8.3.3.2 and §8.3.5.2
	○ *Combustor* components also have the option *liquidExitAllowed* to include or not the amount of liquid exiting a chamber in the gas mixture passing by turbines and other FLUID_FLOW components.	§8.1.2.5 and §8.2.5
	○ New solid/hybrid combustor components including a 2D evaporation model for the solid (and liquids) propellants together with a 1D reaction delay model for the combustor core.	§8.2.6, §8.3.6
	○ New ramjet/scramjet combustor components including a supersonic intake, shock capture, a 1D model for the liquid fuel evaporation together with a 1D reaction delay model.	§8.2.7, §8.3.7
	○ Default value of vaporization factors f_v[] in combustor models is set back to 1.	§8.3.3.4 and 8.3.5.4
	○ *starter_y* input data contain the mass fractions of the starter gases, not the molar fractions.	§8.3.3.4 and 8.3.5.4
	○ The ProdMixer_Tee component has been rebuilt to model a mixture of a pure fluid with combusted gases. Mixture of two combusted gases is now simulated by a standard Tee component.	§8.3.10
	○ Carbon graphite atom element (C_gr) has been set back to (C) because solid chemicals currently have problems reading CEA properties.	§4.1.2, §8.1.2.7
	○ *"xxx_Rate"* and *"xxx_eq"* combustor types have become deprecated components because their characteristics are now included in new components without extension names	§8.1.1
	TANKS Library:	
	○ Inclusing of absorption - desorption effects in Tanks. This option is still under development.	§6.3.4.2
	○ A bug calculating the heat fluxes in "theta" direction (component ABS_Dome_ins) has been fixed.	§6.3.1.5
	○ Cancelling any dz[] derivative in the TANK walls components, as it is considered a fixed grid in the wall.	(software changes)
	STEADY Library:	
	○ Better explanations for building steady models (design vs. off-design)	§11.1.2
	○ The automatic initialization of ALG variables is more complete with no dependence on the combustor instance orders. Junctions & Valves can be initialized in mass flows and temperature.	(software changes), §11.3.1.4
	○ New design option "known_pressures" for Turbines.	§11.2.2
	○ Modification of the calculation of dh_ise in pumps by an approximate expression depending on the inlet density and deltaP.	§11.3.6.5
	○ Orifices flow area in design calculations accounts for critical flow	§11.3.3.3

	conditions. ○ Input data "eta" (nominal efficiency) of Turbines and Pumps components is changed to "eta_o". Input data "Tw_throat" (design wall temperature) of CoolingJackets components is changed to "Tw_design".	See comp. input data
V3.0 (30-11-13)	*This version stabilizes ESPSS 3.0.beta and provides a first operational version for solid, hybrid and ramjets components, and for the simulation of absorption / desorption in Pipes and Tanks.*	
	FLUID_PROPERTIES Library: ○ New SET_OF of Chemicals for solid/hybrid combustors. ○ Carbon graphite atom element (C) has been set back to (C_gr). ○ Passage across the critical point in two phase, two fluid mixtures is more robust.	(software changes)
	FLUID_FLOW_1D Library: ○ New "Tube_res" and "VolPsTsVsCap_TMD" components for the simulation of resistive Pipe components. ○ Absorption/Desorption of non-condensable gases is now operational in Pipes.	§5.4.15, §5.4.20 §5.3.8.2, 5.4.8, 5.4.15
	TANKS Library: ○ Fixing a bug concerning the lower dome wall where the thermal port temperatures were calculated in reverse order. ○ Absorption/Desorption of non-condensable gases is now operational in Tanks.	(software changes) §6.2.2.2, 6.3.4.2
	COMB_CHAMBERS Library: ○ Solid/Hybrid combustors have new input data for the simulation of more specific rubber materials. ○ Using donor_cel function at the exit of a preburner improving reverse flow calculations.	§8.3.6 (software changes)

INDEX

1. INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

The European Space Propulsion System Simulation (ESPSS) is an ESA initiative that aims to create a European simulation platform for spacecraft and launch vehicle propulsion systems under transient and steady (design & off-design) conditions.

This document contains the User Manuals for the ESPSS 3.0 Libraries provided with EcosimPro5.2 [RD-1]. It is assumed that the reader is familiar with the EcosimPro interface, as well as with the laws of fluid mechanics governing the physics of propulsion systems. Chapter 2 presents the global ESPSS overview related to the high level objectives of the libraries. Chapter 3 is devoted to illustrating with a simple example how a user can easily build a model using the ESPSS libraries.

Chapter 4 contains a description of the FLUID_PROPERTIES library. There are no components, but it does provide a large collection of thermodynamic functions extensively used in the ESPSS libraries. Chapters 5 to 8 are dedicated to the transient libraries (FLUID_FLOW_1D, TANKS, TURBO_MACHINARY and COMB_CHAMBERS) and contain an overview (recommendations and tips) and a detailed description of their components. In order to provide a direct access to the Physical Modeling expressed in reference AD-4, a special section describing its formulation is included for each library. This chapter is structured according to the inheritance level of the components. Special care has been taken in the inheritance hierarchy to avoid repeated formulation in the components. Thus, the most 'dependent' components (those operational) inherit the formulation of their parents. Normally, a reduced group of "Abstract" (parent) components is sufficient to contain the main formulation.

Chapter 9 is devoted to demonstrating the capabilities of ESPSS when coupled to external software. Two applications are selected: a) Zooming: EcosimPro can be coupled to 3D CFD software acting as a simple component, and b) System optimization: EcosimPro can also be coupled to an external optimizer whose purpose is to find the best combination of parameters leading to the objectives imposed by the user.

Interesting application examples for getting started with the use of the ESPSS libraries can be found in Chapter **10** and in reference RD-5. The corresponding simulated systems are templates for more complicated models.

Chapter 11 describes the STEADY library for the direct calculation of the steady performances of liquid rocket engine cycles. Useful examples and comments have been included to help the user design new cycle models for launchers and spacecraft vehicles.

Chapter 12 describes the library COMP_DATABASE, which can be used as a template for the creation of an electronic component database compatible with ESPSS, for characterizing existing hardware components (e.g. valves, filters, pressure regulators….). Chapter 13 describes the library SATELLITE, which allows computing the orbital and attitude motion of a satellite, and studying the effect of the acceleration of the satellite on the tanks. Chapter14 focuses on the library EP, for the transient simulation of Electric Propulsion systems (simple models for Thrusters, Xenon flow controller (XFC) and Power Processing Unit (PPU).

1.2 VERSION 2.0 UPGRADES

Version 2.0 accounts for the upgrades and experience acquired under the Industrial Evaluation phase [RD-7] (liquid injection in combustion chambers, *experimental* validation of priming cases, two-phase tanks filling, ESCA engine pressurization systems, etc) and for the new capabilities developed for the Phase II of the ESPSS, see revision table, ESPSS V2.0.

Multiple software upgrades were needed to validate the code against experimental data (priming cases and ESCA engine). Most of them were collected in the intermediate Version 1.4.1 (see revision table). Version 2.0 incorporates specific new user requirements and stabilizes preliminary versions of the code. The main improvements of Version 2.0 with respect to version 1.0 are:

- Simulations of pipe networks are more robust and faster, specially dealing with two-phase (priming), two-fluid systems.
- Models of tanks (boiling process) and pressurisation systems (real gas properties for the pressuring gas) are improved with faster simulations.
- Combustor models are upgraded in many respects permitting more successful simulations during the ignition and shut down processes.

Version 2.0 is non-compatible with Version 1.0 in the following respects:

- Pressure regulators have a new control port that must be connected to a point of the circuit using a pressure sensor component.
- Global variable Dp_lam (present in any experiment) must be replaced by Re_lam (see §5.3.2.1).
- Some components (junction, valves and combustors) are provided with new input data. Default values (automatically loaded editing the schematic file) would normally reproduce Version 1.0 results.

With Version 2.0 the external environment variable PROP_TABLES is no longer needed. PROP_TABLES folder (containing the fluid properties files) must be under the FLUID_PROPERTIES library folder. CEA coefficients are no longer stored in the sources: They will be read from the CEA files "thermo.lib" and "transport.lib" at execution time. These files (as for the properties files) are provided with the PROP_TABLES folder.

ESPSS Libraries are adapted to work with EcosimPro 4.6, which includes an updated version of the THERMAL library and other useful upgrades. Some non-useful output variables of the ESPSS libraries (used for programming reasons) are now hidden from the user (new PRIVATE specification using EcosimPro 4.6 language), so the plot variables selecting menus are simpler and faster.

1.3 VERSION 3.0 UPGRADES

1.3.1 Version 2.2

Version 2.2 takes into account the experience acquired since Version 2.0 was released, mainly derived from user feedback. The main improvements of Version 2.2 with respect to Version 2.0 are:

- The real properties files and the thermodynamic functions are upgraded gaining in robustness near the critical point and the triple point temperature.
- The Cold Thruster component is upgraded, now being able to simulate supersonic/subsonic transitions inside the nozzle.
- Valves are provided with user-defined characteristics.
- The Heat Exchanger component is upgraded to include Cross Flow arrangements.
- Fluid Cavities and Tanks account for the volume expansion due to wall compressibility.
- Tables of loss of head vs. NPSH and compressibility effects due to pump cavitation have been included in Pumps.
- 1D Tanks can work with fluids in supercritical conditions and simplified fluids, and they provide more robust simulations due to the use of better protected correlations concerning the gas/liquid interface.
- Combustors can be initialized under ignited conditions. Additionally, transitions from cold liquid injection conditions to burned gases conditions in the chamber are more robust than in Version 2.0.
- New components are available for Ps/Ts/Mach boundary conditions (intakes).
- Geometrical calculations are included in the continuous block of the components, so parametric studies can be done in a geometrical design without having to restart the simulation.

1.3.2 Version 2.4

Version 2.4 improves the simulation models in the following points:

- TANKS library accounts for film boiling phenomena and better simulation of the generalized boiling process. Moreover:
- The Tee volume calculation is changed according to a right Tee. A new input data for Y and User defined Tee types is added.
- New components (for Pipes and Tees down-stream a pre-burner) are available to compute the delay in the transport of the combustion products. They also permit the simulation of a mixture of combusted gases and pure fluids (chamber with more than 2 injectors).
- New option in Pumps (*tdh_correction* = TRUE) so that the pump head is modified to account for highly compressible liquids.

1.3.3 Version 2.6

Version 2.6 includes a new library for *direct STEADY calculations designing cycles* and takes into account new features included in the Libraries COMP_DATABASE, SATELLITE and ELECTRICAL_PROPULSION:

- The *STEADY* library contains a complete set of components (combustor, nozzle, turbines, pumps and valves) able to calculate the performances of any cycle type under design and off-design conditions.
- The STEADY_EXAMPLES library contains a set of cycle types to help the user build models.
- The COMP_DATABASE library can be used as a template for the creation of an particular component compatible with ESPSS, characterizing existing hardware (e.g. valves, filters, pressure regulators….)
- The SATELLITE library allows computing the orbital and attitude motion of a satellite, and studying the effect of the acceleration of the satellite on the tanks.
- The EP library, for the transient simulation of Electric Propulsion systems (simple models for Thrusters, Xenon flow controller (XFC) and Power Processing Unit (PPU).

1.3.4 Version 3.0

This version 3.0 *gains in calculation speed, accounts for important upgrades in the calculation of the combustor gas properties, includes new options and components (scramjet & solid/hybrid combustors) and upgrades the ESPSS libraries with the absorption/desorption of gases. The STEADY library has gained in robustness after months of tests*:

- FLUID_PROPERTIES Library changes:
  - a/ Molar fractions arguments changed to mass fractions in CEA thermo routines.
  - b/ A complete set of Van der Waals properties functions has been added.
  - c/ Properties functions explicitly depend on the burned gases chemical compositions. Perfect gas or Van der Walls state of equation is optional.
  - d/ Typical solid propellants constituents and their products are included in the list of chemicals and in the combusted gases properties calculation.
- FLUID_FLOW_1D Library changes:
  - a/ Diluted non-condensable gases are transported in volumes and Pipes.
  - b/ Convection and mixing of the chemical constituents are calculated dynamically (using the transport equations) for any component downstream of a combustor. These components will calculate properties *in accordance with the current chemical compositions, not with the combustor properties*. Thus, ESPSS components must be initialized with the **burnerGasesOption** parameter, which has to options:
    - *burnerGasesOption = FLUID_PROPERTIES.Chemicals* if the component is placed downstream of a combustor. The orking fluid is treated as a mixture of variable chemical composition calculated with the Perfect gas or the Van der Walls state of equation.

- *burnerGasesOption = FLUID_PROPERTIES.noBurnGases* if the component is not downstream of a combustor. Mixture of a (real) fluid with a non-condensable gas.

- c/ Initialization of the dynamic state variables is simpler, with no dependence on the order of the components instance and no need to use DISCRETE blocks.
- d/ The Roe scheme in Pipe components has been upgraded to work with supersonic flows transitions in capacitive Pipes.
- e/ Including non-condensable gas absorption - desorption effects.

- TANKS Library:
  - a/ Including non-condensable gas absorption - desorption effects.
- COMB_CHAMBERS Library changes:
  - a/ Combusted gases properties functions are re-built with a simpler and faster code. *Perfect gas or Van der Walls state of equation is optional*.
  - b/ The *ProdMixer_Tee* component has been rebuilt to model a mixture of a pure fluid with combusted gases. Mixture of two combusted gases is now simulated by a standard Tee component. *ProdMixerPipe* does not longer exists because a simple Pipe replace it.
  - c/ *Combustor* components are upgraded with new options: **GasLiqOption** concerning the liquid propellant vaporization models and **rateOption** to choose equilibrium or reaction delay method:
    - *GasLiqOption = Advanced*: convection of liquid is calculated assuming the same speed as in the gas side. Liquid droplets size is fine-tuned in time and position through the "f_v[]" boundary variables multiplying a reference size. Vaporization is modeled depending on combusted gas conditions, latent heat, etc.
    - *GasLiqOption = UserDefined*: no convection of liquid is calculated. Vaporization is assumed to be within a delay time just after the injection plate if the ignition order is given (boundary *IgnitFlag*=1), or null if no ignition order is given.
    - *rateOption = TRUE*: this is a first approach of a non-equilibrium chamber based on a time-delay between the equilibrium and the actual burned gases composition.
    - *rateOption = FALSE*: all the vapors will react instantaneously.
  - d/ Combustor components have the option **liquidExitAllowed** to include (or not) the amount of liquid exiting a chamber in the gas mixture passing through turbines and other FLUID_FLOW components.
  - e/ New solid/hybrid combustor components including an "evaporation" model for the solid (and liquid) propellants together with a 1D reaction delay model for the combustor core.
  - f/ New ramjet/scramjet combustor components including an intake, shock capture, a 1D model for the liquid fuel evaporation together with a 1D reaction delay model.
- *STEADY* library changes:
  - a/ The automatic initialization of ALG variables is more complete with no dependence on the combustors instance order and no need of DISCRETE blocks. Improving the initialization of enthalpy and mass flows unknowns.
  - b/ New design option "known_pressures" for Turbines.
  - c/ Combusted gases properties function are re-built with a simpler and faster code.
  - d/ The calculation of dh_ise in pumps is modified by an approximate expression depending on the inlet density and deltaP.
  - e/ Orifices flow area in design calculations takes into account critical flow conditions.
- *NEW APPROACH INITIALIZING STATE VARIABLES:* A new powerful approach initializing state (dynamic) variables has been included. Previous versions did not assure proper initial values for more

than 1 pre-burner, and could fail depending on the components instance order in the schematic of a model.

The new initialization (also for the STEADY library) is done thanks to the following changes:

- Better use of the PRIORITY 200 directive in the INIT block of the WorkingFluid, ProdPF_Mixer and pre-burners to be sure that any new fluid is defined (and the properties file read) before any other component.
- Components working with combusted gases must be initialized with a non-condensable gas (which is on the other hand more physical) because it would be practically impossible for the user to define all the combusted gases compositions at the initial time.
- No need to transfer the pre-burners gases constants (R and Cp) to the downstream components because these properties are calculated in any ESPSS component at the current P/T/chemicals mass fractions conditions, thanks to another important upgrade in Version 3 including convection equations for chemicals.

This new initialization was impossible in previous versions because the chemical composition of the burner gases was not properly "transported" downstream of the combustor. It was assumed to be the same as in the combustor.

1.4 NON-COMPATIBILITY ISSUES W.R.T. VERSION 2.0

With respect to Version 2.0, most of the components (Pipes, Junctions, Valves, Pressure Regulators, Pumps, Heat Exchangers, Tanks, CoolingJackets and Combustors) are provided with new input data and calculation options. *It must be considered that:*

- Default values (automatically loaded editing the schematic file) would normally reproduce Version 2.0, 2.4 results. *The following exceptions are noted*:

  FLUID_FLOW_1D Library:

  - Re_lam is no longer a global parameter, but a particular input data for junctions and valves. Re_lam must be deleted from the experiment files boundary variables.
  - CAV_DAMP is no longer an input for Pipes because it slows down the simulation with little improvement of the results.
  - Any component working with combusted gases (therefore, placed downstream of a combustor) will be automatically initialized with non-condensable gases at the initial values of Po/To.
  - The numerical parameters using Roe scheme have to be defined by the user. Using the Roe scheme, the "Case" input data has been suppressed and a new input data, "Isent_Correl", allows supersonic flow transitions.

  COMB_CHAMBERS Library:

  - *"xxx_Rate"* and *"xxx_eq"* combustor types have become deprecated components because their characteristics are now included in new components without extension names.

    Models using the old *xxx_Rate"* and *"xxx_eq"* combustor types will use automatically the formulation contained in the new combustor types without extension name, but the user should change them. *To do this, the user can edit (outside EcosimPro) the corresponding *.eds files and to replace the old type with the new one without extension _rate or _eq.*
  - *starter_y* input data contain the *mass* fractions of the starter gases, not the *molar* fractions.
  - The default value of vaporization factors f_v[] in combustor models is set back to 1.
  - The ProdMixer_Tee component has been rebuilt to model a mixture of a pure fluid with combusted gases. The mixture of two combusted gases is now simulated by a standard Tee component.

  TANKS Library:

- Tank options for the gas/liquid interface are renamed with more logical names (advanced → Diffusion; default → EvapPool). These new options should be loaded manually in the attributor editor of the schematic view for any model containing 1D tanks.
- Currently, the spin and lateral acceleration effects in TANKS are deactivated because the corresponding Astrium FORTRAN function does not work with the GCC compiler.

STEADY Library (Changes from preliminary version 2.6):

- The input data "*eta*" (nominal efficiency) of Turbines and Pumps components has been changed by "*eta_d*". Similarly, the input data "*Tw_throat*" (design wall temperature) of CoolingJackets components of the STEADY library have been changed by "*Tw_design*".
- Junctions & Valves can be initialized in mass flows and temperature in case of collection of flows upstream the valve.

- Most of the port variables are made HIDDEN because they are redundant with the local variables of the components. Old plot configurations should probably be modified choosing component variables instead of port variables, see §3.4.
- *The lists of arguments of some properties functions have changed to allow the calculation of both pure fluid and combusted gases properties, see §4.2.1. Combustion functions and components no longer work with molar fractions but with mass fractions*.

1.5 APPLICABLE DOCUMENTS

AD-1 086-038-XY-L-X1820 "ESPSS. Management Proposal"

AD-2 086-038-XY-L-X1819 "ESPSS. Technical Proposal"

AD-3 086–031–LRU–L–0001 "ESPSS User Requirements Document"

AD-4 086–031–NT–L–0001 "ESPSS Physical Model Specification Document"

1.6 REFERENCE DOCUMENTS

RD-1 EcosimPro Dynamic Simulation Tool. User Manual. EA International. 1992-2013

RD-2 EcosimPro CONTROL library

RD-3 EcosimPro THERMAL library

RD-4 EcosimPro MECHANICAL library

RD-5 4000103800/11/NL/CP – TN-4120. ESPSS-3. Software Verification and Validation Plan

RD-6 4000103800/11/NL/CP – TN-4110. ESPSS-3. Software Transfer Document

RD-7 12205/07/NL/CP – TN-2110. ESPSS-2. Feasibility Demonstration of an European Space Propulsion System Simulation Tool. Industrial Evaluation

RD-8 REFPROP V7.0. Reference Fluid Thermodynamic & Transport Properties. National Institute of Standards & Technology. (NIST)

RD-9 Gordon, S., and B. J. McBride: Computer Program for the Calculation of Complex Chemical Equilibrium Compositions with Applications. NASA Reference Publication 1311 (1994)

RD-10 Reid, R.; Prausnitz,J.M; Poling, B.; The properties of Gases & Liquids, 4th Edition, 1987

RD-11 C.-A. Schley & J. Kollien. Algorithms for efficient computation of Propulsion Fluid Properties. AIAA 96-0235, January 15-18, 1996 / Reno, NV

RD-12 E. W. Schmidt, Hydrazine and its Derivatives - Preparation, Properties, Applications, Second Edition, Vol. 1, J. Wiley & Sons, 2001

RD-13 Marco De Rosa, Fluid properties of N2O4 taking into account decomposition, TEC-MP/2007/385/MDR, 11-04-2007. Draft Technical Note. ESA/ESTEC

RD-14 Alfred C. Wright. USAF propellant handbooks – nitric acid/nitrogen tetroxide oxidizers. Technical Report AFRPL-TR-76-76, Martin Marietta Corp., 1977.

RD-15 Roger A. Svehla and Richard S. Brokaw. Thermodynamic and transport properties for the N2O4 –2NO2 –2NO + O2 system. Technical Report TN D-3327, NASA, 1966

RD-16 Kenneth P. Coffin and Cleveland O'Neal, Jr. Experimental thermal conductivities of the N2O4 - 2NO2 system. Technical Report TN-4209, NASA, 1958

RD-17 Kuentzmann P.; Lecourt R.; Levy F.; Prud'homme R.; Degtiarev G.L.; Naoumov V.I. Rapport d'expertise COMPERE (tome 2). ONERA RT 75/6116 DEFA/Y TOME 2. 1999

RD-18 Relap5 Manual Volume 4: Models and Correlations, Appendix 7-A presents the derivation of the Homogeneous Equilibrium Sound Speed and the Frozen Sound Speed of two phase mixtures

RD-19 Fluid Mechanics, Landau and Lifshitz, Pergamon Press, (Chapter VIII Sound, Section 63, Problem 1: Determine the velocity of sound in a nearly homogeneous two-phase system

RD-20 Transfer Processes. An introduction to Diffusion, Convection and Radiation. D.K. Edwards. V.E. Denny. A.F. Mills. University of California. Los Angeles. Holt, Rinehart and Winston, Inc. 1973. Pag.210

RD-21 ESPSS-RIBRE-TN-0001. Mathematical Model Formulations for Propellant Tanks. Astrium - EADS

RD-22 ESPSS-RIBRE-TN-0007. Functions for 1D Propellant Tank Geometries. Astrium – EADS

RD-23 Bartz, D.R., Turbulent Boundary-Layer Heat Transfer from Rapidly Accelerating Flow of Rocket Combustion Gases and of Heated Air, Advances in Heat Transfer, pp. 2-108, 1965

RD-24 Ascher H. Shapiro. The Dynamics and Thermodynamics of Compressible fluid flow

RD-25 ESPSS-RIBRE-TN-0010. Mathematical Model Formulations for Engine Injector Head, Combustion Chamber, convergent-divergent Nozzle and Cooling Jacket

RD-26 Fluent Inc. Fluent 6.3 User's guide. Lebanon, NH, 2006

RD-27 Fluent Inc. Fluent 6.3 UDF Manual. Lebanon, NH, 2006

RD-28 D E. Goldberg. Genetic Algorithms. Addison Wesley, 1994

RD-29 G. Holt and S. Bassler. Preliminary design of axial compressors using artificial intelligence and numerical optimization techniques. ASME Paper 91-GT-334, 1991

RD-30 S. Pierret, H. Kato, R. Filomeno Coelho, and A. Merchant. Aeromechanical optimization method with direct cad access - application to counter rotating fan design. ASME paper GT2006-90505, 2006

RD-31 I. De Falco. An Introduction to Evolutionary Algorithms and their Application to the Aerofoil Design Problem. In Von Karman Institute for Fluid Dynamics : Lecture Series 1997-05 : Inverse Design and Optimization Methods, 1997

RD-32 D. Quagliarella. Genetic algorithm applications in computational fluid dynamics. In Winter Periaux Galan Cuesta, editor, Genetic Algorithms in Engineering and Computer Science, pages 417–442. John Wiley & Sons, 1995

RD-33 J. Periaux, M. Sefrioui, B. Stoufflet, B. Mantel, and E. Laporte. Robust genetic algorithms for optimization problems in aerodynamic design. In Winter Periaux Galan Cuesta, editor, Genetic Algorithms in Engineering and Computer Science, pages 371–396. John Wiley & Sons, 1995

RD-34 M A. Trigg, G R. Tubby, and A G. Sheard. Automatic genetic optimization approach to two-dimensional blade profile design for steam turbines. Journal of Turbomachinery, pages 11–17, 1999

RD-35 S. Obayashi and S. Takanashi. Genetic Optimization of Target pressure distributions for inverse design methods. AIAA-95-1649-CP, pages 33–42, 1995

RD-36 V. Valentin. Three-dimensional optimization of a compressor blade at design and off-design operations. VKI PR 1998-26, 1998

RD-37 C. Poloni. Hybrid GA for multi objective aerodynamic shape optimization. In Winter Periaux Galan Cuesta, editor, Genetic Algorithms in Engineering and Computer Science, pages 397–415. John Wiley & Sons, 1995

RD-38 S. Obayashi. Aerodynamic optimization with evolutionary algorithms. In Von Karman Institute for Fluid Dynamics : Lecture Series 1997-05 : Inverse Design and Optimization Methods, 1997

RD-39 C.M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995

RD-40 Z. Michalewicz, D. Dasgupta, R.G. Le Riche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. Computers and Industrial Engineering Journal, 30(2):851–870, 1996

RD-41 C A C. Coello. Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: a Survey of the State of the Art. Computer Methods in Applied Mechanics and Engineering, 191:1245–1287, 2002

RD-42 C.R. Houck J.A. Joines. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with gas. In First IEEE International Conference on Evolutionary Computation, pages 579–584. IEEE Press, 1994

RD-43 K. Deb. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 186:311–338, 2000

RD-44 George P. Sutton. Rocket Propulsion Elements. An introduction to the engineering of Rockets. Sixth Edition. John Wiley & Sons, Inc

RD-45 Idelchik, I.E.; Handbook of Hydraulic Resistance, 3rd edition, Research Institute for Gas Purification, Moscow, Russia, 1994

RD-46 E. Benjamin Wylie & Victor Streeter; Fluid Transients in Systems. Prentice Hall, Inc. 1993

RD-47 Mitchell, R. C. et al., Engineering Properties of Rocket Propellants. Final Report, AFRPL-TR-73-105, Rockwell, 1973.

RD-48 USER Manual TriaX OrbitaL tool available at www.kopoos.com

RD-49 Cours de technologie spatiale: - Techniques et technologies des véhicules spatiaux T 4, CNES, 1993.

RD-50 Michael Paluszek et al., Spacecraft attitude and orbit control, Princeton Satellite systems inc., 2009 .

RD-51 R. Ouziaux, J. Perrier, Mécanique des fluides appliquée, tome 1 Fluides incompressibles, 1966.

RD-52 EcosimPro FLIGHT_SIM Library

RD-53 P. Duchon et al., STABILISATION DES SATELLITES, SupAero, 1983.

RD-54 R. Guiziou., DESS AIR & ESPACE, systèmes de contrôle d'attitude et d'orbite, Uni. Aix-Marseille III, 2001.

RD-55 S.Bouiges, Calculs astronomiques pour amateurs, Masson, 1992.

RD-56 Vernon Chi. Quaternions and Rotations in 3-Space, 25 September 1998 ; Leandra Vicci, 27 April 2001

RD-57 Vladimir A. Chobotov, Orbital Mechanics course.

RD-58 Eai-Kci-Me-09 TN3610PhysicalModelSpec Satellite Library06 Technical note TN-3610 Physical Model Specification

RD-59 Chang, E. T., and N. A. Gokcen. "Solubilities of gases in simple and complex propellants." *Journal of Spacecraft and Rockets* (Journal of Spacecraft and Rockets), 1969

RD-60 Jokela, K., Ferran Valenci i Bel, and I. Kälsch. "Dynamic solubility of presurised Helium gas in liquid propoellant in closed storage tanks." *4th International Spacecraft Propulsion Conference.* Cagliari: ESA, 2004

RD-61 Knowles, P. J. "Helium Absorption into Nitrogen Tetroxide (NTO) and Aerozine-50(A-50)." *Journal of Spacecrafts and Rockets* 9, no. 9 (1972)

RD-62 Singhal, A. K., H. Y. Li, M. M. Athavale, and Y. Jiang. "Mathematical basis and validation of the full cavitation model." *ASME Journal of Fluids Engineering*, 2002: 617-624

RD-63 Valencia i Bel, Ferran. *Dynamic Solubility of a gas into the propellant.* EWP-2252, ESA, 2004

RD-64 Yang, H. Q., A. K. Singhal, and M. Megahed. "The full cavitation model." In *Industrial two phase flow CFD*. Rhode-Saint-Genese: von Karman Institute, 2005

2. ESPSS OVERVIEW

2.1 PROJECT AREAS

ESPSS Software can be used for system concept definition, mission analysis, impact studies, investigation of anomalies and optimization, testing and pressuring/propellant loading. It is structured in different project areas (libraries):

- The Fluid Properties library allows, among other capabilities, calculating real properties of the typical working fluids in propulsion systems as done by the well-known NIST code [RD-8].
- The Fluid Flow library simulates complex pipe two-phase, two fluid systems improving the numerical schemes classically used in other codes, dealing in particular with two-phase discontinuities and shock waves (Roe scheme).
- The Combustion Chamber library simulates, among many other capabilities, the chemical equilibrium of an arbitrary mixture of chemicals as the well-known CEA code [RD-9] does, but in transient and non-adiabatic conditions where ignition and global reaction times are also considered.
- The Tanks and Turbo-Machinery Libraries will allow integrating the state of the art of these rocket subsystems.
- The *STEADY* library contains a complete set of components (combustors, cooling-circuits, nozzles, turbines, pumps and valves) for calculating the performances of any cycle type under design and off-design conditions. This means that the "sizing" of the cycle is calculated under steady conditions by the code giving some design conditions such as the chamber pressures, mixture ratio, efficiencies, etc.
- Other advanced capabilities, like:
  - Improved algorithms for fast transient 1D Fluid Flow, performing better tracking of the pressure and phase discontinuities.
  - An Optimization Module for Design and Test Fitting.
  - Standard Interfaces with other software tools used for the analysis of propulsion systems, like CFD codes and thermal analyzers (Zooming).

2.2 ECOSIMPRO ENVIRONMENT

The library's components have been defined using the ECOSIMPRO software [RD-1], the simulation tool recommended by ESA for ECLS. It has been used successfully in the European activities of the ISS program. It incorporates the following main features:

- A user-friendly Graphic User Interface (GUI).
- Encapsulation (object-oriented programming), which enables the abstraction of a component's behavior (equations and data) into abstract components.
- Inheritance, which enables joining the behavior of the parent components into the inherited components. The formulation can be 'partitioned' in sub-components in such a way that a modification in an inner component will be automatically taken into account by the inherited components.
- Aggregation, which permits the creation of complex components (models) thanks to the connection of multiple instances of single components. Again, a modification in the formulation of the aggregated components will be automatically taken into account by the complete model.
- State of the art solvers.
- The possibility of exchanging and sharing models or model data.

- To enable the users to add their own models (using an Object Oriented Language) and to protect these models when they are sent to another party.

The ESPSS libraries provide palettes of components[1] (represented by symbols) that can be used to build graphically complex systems. This document aims to provide the information needed to understand and use the library components, and to provide the required knowledge of the physical behavior. The Standard EcosimPro libraries (Math, Control, Electrical, Mechanical and Thermal), often the bases of the main ESPSS libraries, are documented in references RD-2 to RD-4.

2.3 PROJECT ORGANIZATION

The development of a Software Tool for the simulation of Space Propulsion Systems is an ambitious project requiring the coordination of multiple disciplines. To this effect, Empresarios Agrupados Internacional (EAI) has sought the collaboration of leading companies specialized in some of the key areas of the project.

The combined efforts of an expert in the design and production of simulation software (Empresarios Agrupados Internacional), the primary industrial architect and stage integrator of the Ariane launchers (EADS Astrium – Space Transportation), a company at the forefront of CFD and Optimization methods (CENAERO), a firm with broad experience in chemical- and electrical-type Satellite Propulsion Systems (KopooS), along with the collaboration of the Von Karman Institute (VKI), guarantee the successful development of the project.

1 A component is the basic simulation unit generated by Ecosimpro Language representing some physical/logical behaviour. Ecosimpro uses these components as symbols that can be dragged & pasted graphically to build more complex models.

3. HOW TO BUILD and EXPLOIT an ESPSS MODEL

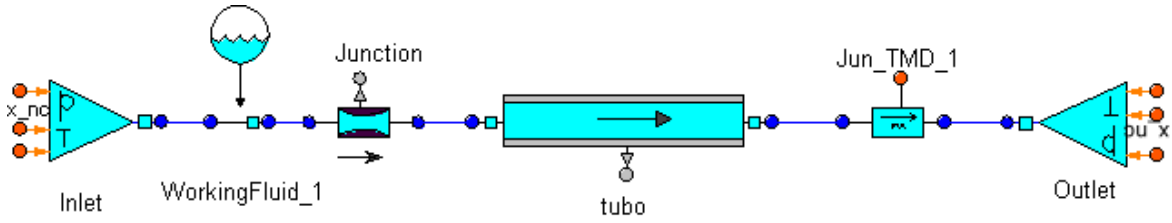
ESPSS libraries are open to a large variety of simulation cases and geometries. The intrinsic difficulties of simulating such fluid system models are not avoided in the ESPSS libraries. Even though messages are issued to prevent erroneous conditions, the user should carefully set the input data and numerical options. ESPSS users should be aware that, without proper geometrical and boundary data, it may be impossible to achieve a stable simulation, as would be the case in reality.

The main procedures for using the ECOSIMPRO tools are explained in this chapter. The other chapters of this Manual are dedicated to each of the ESPSS Libraries, providing an overview of them (see the "Building a Model" sub-chapters for particular tips and advice) and a detailed physical and mathematical description of their components.

The application examples contained in the FLUID_FLOW_1D_EXAMPLES and ROCKETS_EXAMPLES libraries are templates for more complicated models. Chapter 10 and Reference RD-5 describe these interesting models. It is recommended to start by looking at these examples and reading the User Manual.

3.1 DESCRIPTION OF THE EXAMPLE

A simple example (see "Test_pipe" model of the FLUID_FLOW_1D_EXAMPLES library) is described here for *illustrative* purposes. It shows the steps to be followed when building a model using the ESPSS libraries.



The model represents a pipe between two boundaries: The left one imposes P-T. The right one forces the mass flow circulating through the pipe. The objective is to simulate the water-hammer response when suddenly stopping the outlet mass flow.

3.2 USING THE ECOSIMPRO GUI

To create this model select the schematic view of EcosimPro; next, follow the steps indicated below:

- Create a new schematic using the button  or the File->New-> Schematics command. Save it in a previously created user library (File->New->Library) and choose a name (test, for example).
- From the icon explorer tab of the appropriate ESPSS libraries, select one by one the different components shown in the figure above and drag them into the schematic window
- Arrange and name (alias) the components in the schematic as in the figure above:
  - Use the Rotate buttons if necessary.
  - To change the size of a component, select the component, click the right button, select "component shape option" and change the size by dragging the symbol's corners.
  - To change the position of the component's name press the SHIFT key, move the mouse pointer over the labels at the same time, left-click and drag it.
- Draw connectors between the components like in the figure above. A tooltip will appear whenever the mouse moves over a port, displaying the information of that port:
  - Select the connection button on the right-hand toolbar  , or press the SHIFT key and move the mouse pointer over a port at the same time. Left-click on the port to be connected

- Click the points of the schematic drawing where the connector is required to run (if any)
- Left-click the target port, which must be of the same type as the origin port
- To connect different components with right angles, there are two options: 1) select right angle connections mode by pressing  button and connect components as explained before. 2) Create a connection straight line between two components, press SHIFT and left click over the line. A point will be created to divide the line into two different segments. Drag the point to the desired place and use  button to force right angles.
- To delete extra points of a connector line, just press SHIFT over the point and click on the point to be deleted

- Use the "Edit source code" button  to add global data to the model if necessary. The use of global data variables (pressure and temperature for example) is useful to initialize the same conditions in different components. In the example below, three lines (here marked in bold type) have been added to the default source code shown when clicking the "Edit source code" button:

```
USE FLUID_FLOW_1D
USE FLUID_PROPERTIES
USE THERMAL

COMPONENT Test_Pipe

DATA
REAL P_ini = 1e5
 REAL T_ini = 300

END COMPONENT
```

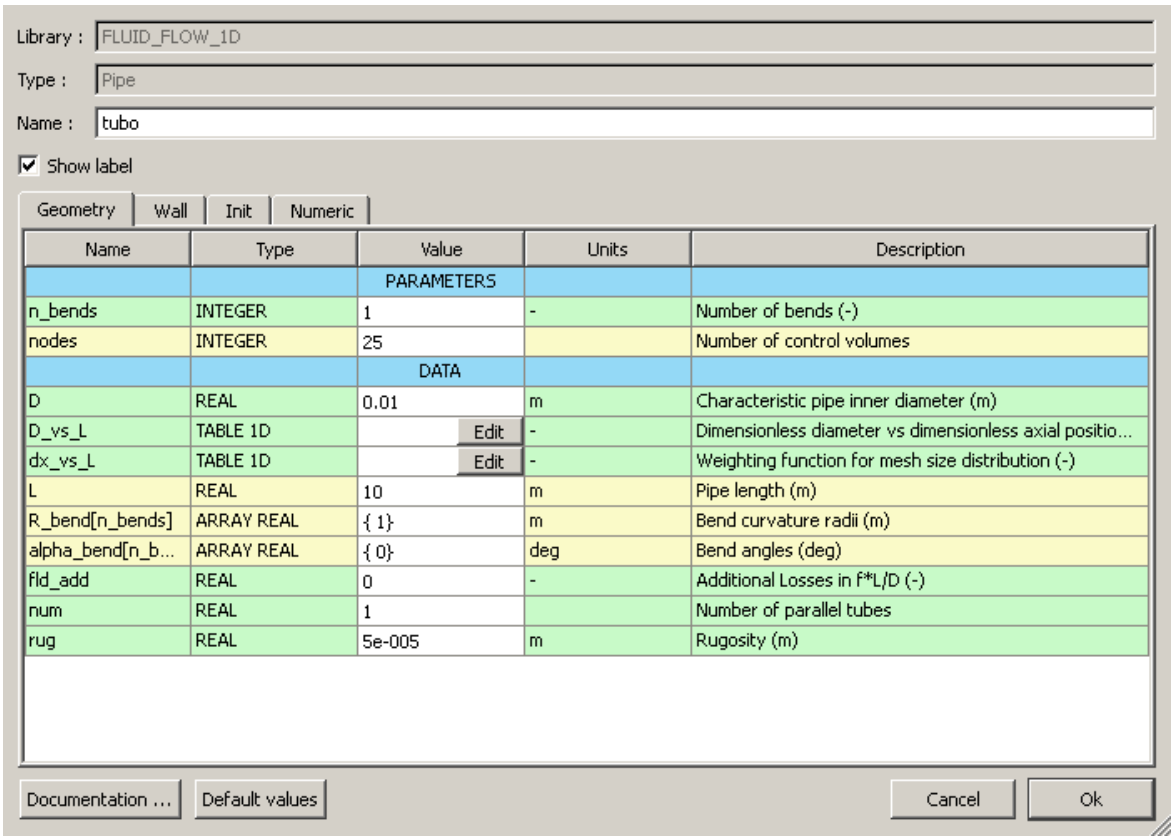
  Then, variables *P_ini* and *T_ini* can now be used in any component entering the input data

- Set the data of the components. It is possible now to customize the component data. The table below (automatically build by EcosimPro when generating documentation) collects all the model data. Many input data (some of them not shown in table below) have the default value.

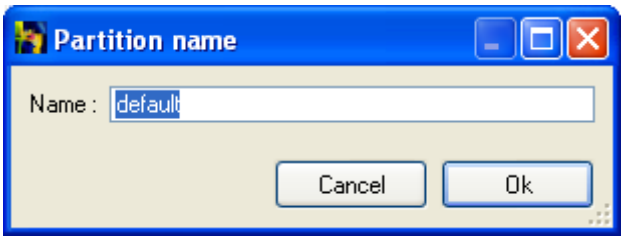
DATUM	VALUE	DESCRIPTION
Jun_TMD_1.tao	0.0001	Time constant of mass flow filter (s)
Junction.Ao	0.7*0.01**2	Junction area when fully open (m^2)
Junction.Gcr_ideal	0	TRUE, exact critical flow calculation, FALSE, approximated
Junction.m_o	1	Initial mass flow (kg/s)
Junction.zetab	1	Backward loss coefficient (-)
Junction.zetaf	1	Forward loss coefficient (-)
WorkingFluid_1.fluid	Real_H2O	Working fluid
WorkingFluid_1.fluid_nc	NoFluid	Non-condensable working fluid
tubo.D	0.01	Characteristic pipe inner diameter (m)
tubo.E_wall	1e11.	Wall elasticity modulus , if mat=None (Pa)
tubo.L	10	Pipe length (m)
tubo.P_o	**P_ini**	Initial pressure – only for INIT_PX, INIT_PT or INIT_RPT options (Pa)
tubo.R_bend[n_bends]	{ 1}	Bend curvature radii (m)
tubo.T_o	**T_ini**	Initial temperature – only for INIT_TX, INIT_PT or INIT_RPT options (K)
tubo.T_outside	300	Environment temperature for the pipe (K)
tubo.alpha_bend	{ 0}	Bend angles (deg)
tubo.dx_vs_L		Weighting function for mesh size distribution (-)
tubo.e_wall	0.001	Wall thickness (m)

tubo.fld_add	0	Pressure drop coefficient for additional Losses
tubo.h_outside	0	Heat transfer coefficient with the environment (W/m^2*K)
tubo.hc_dat	1	Heat transfer coefficient – only if ht_option = Ht_constant
tubo.ht_option	1	Wall-fluid heat transfer option
tubo.init_option	1	Option to specify the initial thermodynamic state
tubo.mat	SS_310	Wall material (-)
tubo.n_bends	1	Number of bends (-)
tubo.nodes	25	Number of control volumes
tubo.num	1	Number of parallel tubes
tubo.rho_o	1	Initial density – only for INIT_RT or INIT_RPT options
tubo.rug	5e-005	Rugosity (m)
tubo.v_wall	0.3	Wall Poisson coefficient, if mat=None (-)
tubo.x_nco	0	Initial non-condensable mass fraction – only for INIT_PT or INIT_RT options (-)
tubo.x_o	0	Initial quality – only for INIT_PX or INIT_TX options (-)

- To change data of a component double-click on the corresponding symbol to open the Attributes Editor. For example, for the Pipe component, data values should look like what is shown in figure below.



- Finally, generate the EcosimPro model using the "Generate model" button  . The following window will be displayed. Click OK and the building of the model will have finished. It can be simulated as described in the following section.

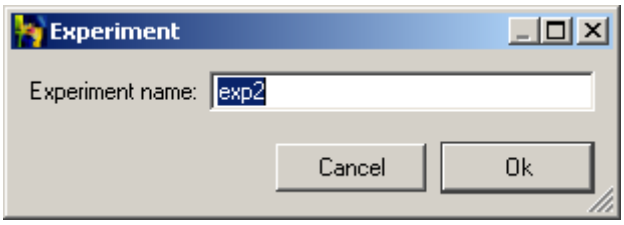


The "Generate model" button of the EcosimPro GUI compiles the graphical model and generates the default partition (mathematical ordering). It should be pointed out that at this moment EcosimPro will internally find an automatic ordering of the equations of the whole model, searching for the appropriate boundaries and algebraic variables (if any) to be solved.

3.3 SIMULATE THE MODEL (EXPERIMENT FILES)

It is now ready for the simulation.

- Select "test.default" (if the name of the partition created is "default") from the library where the model was saved, right-click, select option "New experiment". The following window will be displayed. Type in a name for the experiment:



- A default experiment should automatically appear in the editing window containing all the boundaries needed to run the model. Change the TSTOP, CINT (Communications Interval) and other values as indicated in the experiment here below:

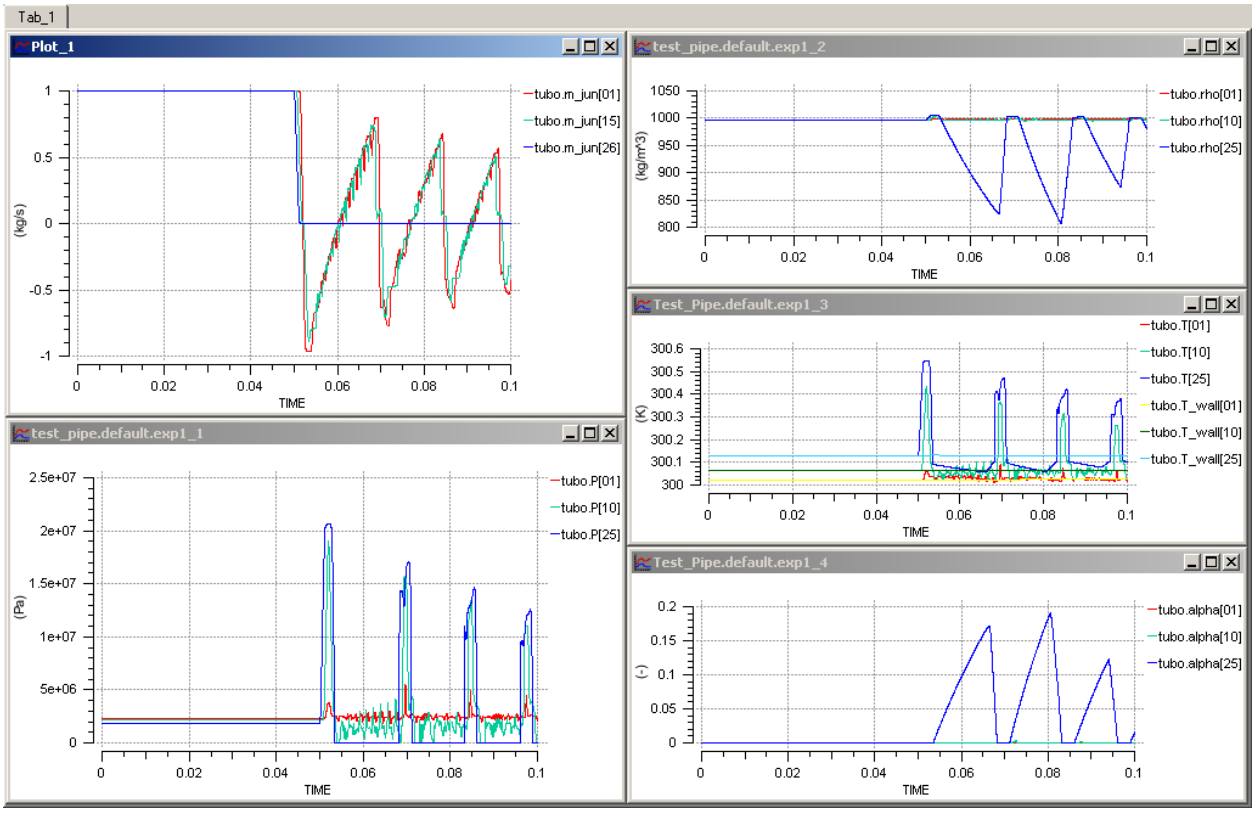
```
EXPERIMENT exp2 ON Test_Pipe.default
DECLS
  TABLE_1D wl = {{0, 0.05, 0.051, 10}, {1, 1, 0, 0}}

INIT    -- set initial values for variables if not defined in the model

BOUNDS    -- set expressions for boundary variables: v = f(t,...)
  Damp = 0.2
  Jun_TMD_1.s_massflow.signal[1] = timeTableInterp(TIME,wl)

Inlet.s_pres.signal[1] = P_ini
Inlet.s_temp.signal[1] = T_ini
Inlet.s_xNonCond.signal[1] = 0.00

  Outlet.s_pres.signal[1] = 1e5
Outlet.s_temp.signal[1] = T_ini
Outlet.s_xNonCond.signal[1] = 0.
BODY
--- These values will overwrite the ones used in the schematic model
T_ini = 300
  P_ini = 25e5
  WorkingFluid_1.fluid = Real_H2O  -- CryoProp_MMH – Real_R134A  -- Real_MMH  -- PfLiq_H2O  --
WorkingFluid_1.fluid_nc = NoFluid – PfGas_Air – PfGas_Usr1 –
tubo.x_nco = 0.00
  tubo.L = 2
  tubo.D = 0.01
```

```
tubo.m_o = 1

--Following lines should always be added in the experiments
IMETHOD = DASSL_SPARSE
  REPORT_MODE=IS_STEP
REL_ERROR = 1e-4--1e-5
  ABS_ERROR = 1e-4--1e-5

-- Select convenient values for CINT, TSTOP, etc.
TIME = 0
  TSTOP = 0.1
  CINT = 0.01
  INTEG()
END EXPERIMENT
```

- Note here some of the multiple possibilities for coding an experiment:
  - The experiment file is needed for controlling integration steps, boundary conditions (see the inlet/outlet imposed conditions), redefinitions of DATA values (see P_ini, T_ini variables initializing Pipe conditions), etc. Double dash (--) means a comment.
  - Note the use of the *timeTableInterp* function. The use of tables (or other functions available in EcosimPro) allows the definition of the boundary conditions as a function of TIME
  - "IMETHOD = DASSL_SPARSE" means that the DASSL (Differential Algebraic System Solver) sparse integration method will be employed (See the EcosimPro User Manual).
  - *"REPORT_MODE = IS_STEP" means that an output (in the plots and in the reports) will be produced at each calculation time step (given by the integration method). This is very useful because outputs will be produced when necessary independently of the communication interval CINT that can be set to high values.*
  - The absolute and relative errors permitted in the integration (REL_ERROR and ABS_ERROR experiment variables) should be between 1e-4 and 1e-6, see §3.5
  - Other options can be found in §3.6, §3.7 and §3.8

- Save the experiment. The experiment name will appear in the Workspace area
- Now the experiment can be simulated using the Monitor. Right click over the experiment name in the Workspace area, and select "Simulate in Monitor". Clicking on the  button will produce same action
- The Experiments monitor comes up. Add the necessary plots to visualize the results.
  - To add a plot, click on "Tab_1" and then click "New Plot"  . A window will appear with the "drawable / plotable" variables of the experiment.
  - Apply a filter to help with selecting variables. Tick the boxes alongside the selected variables:
- Just as you may wish to plot the evolution of one variable compared with another, you may also wish to track the numerical values of the variables. The main task of the Watch is to display the variable values of the experiment, enabling them to be modified wherever possible.
  - The first step is to add variables by right clicking in the Watch Area and selecting the option "Edit Watch".
  - A menu very similar to that of the plots will appear, enabling you to add or delete variables.
- To simulate the experiment click "Simulate"  . By clicking "New Integration"  on the toolbar, the simulation can be extended or started at new times.
  - Use the mouse to perform interactive zooms on the plots.
  - To resize the scale, you only need to click on the plot area and press an "r".

Below is a print-out of the monitor tool containing some plots obtained with this model:



The following phenomena can be shown:

- Pressure rises (ρcv) are due to the wave trips caused by a sudden stop of the fluid. This wave is reflected in the open end as a negative flow.
- When this backflow is stopped again at the closed end, the "negative" pressures that should be created are limited to the vapor pressure.
- Then, the corresponding vapor bubble formation takes place, water column separation is produced, and the column enters the tank.
- The bubble collapse begins when this liquid column is stopped by the tank pressure and begins to enter the pipe, collapsing the vapor.
- A new cycle starts when the liquid column is stopped at the closed end when the vapor is eliminated.

3.4 OUTPUT VARIABLE NAMES

The most important variables (sensitive case) to exploit in a simulation run are detailed below. Different filters can be applied in the Monitor tool to select variables in a plot or watch, or using REPORTS:

VARIABLES	Monitor Filter	Component Type	VARIABLES	Monitor Filter	Component Type
Pressure (Pa)	*.P	Volume, Tee, Cavities	**Heat exch. Coeff. (W/m2/K)**	*.hc	Volume, Tee, Cavities
	*.P[*] dP_loss	Pipe, Tanks & Comb.		*.hc[*]	Pipe, 1D Tanks & Comb. Chambers
Temperature (K)	*.T	Volume …	**Density (kg/m3)**	*.rho	Volume …
	*.T[*]	Pipe, Tanks & Comb.		*.rho[*]	Pipe, 1D Tanks & Comb. Chambers
Total & Non-	*.m, *.m_nc	Valve/Junction	**Heat flux (W)**	*.q	Volume, Tee,

VARIABLES	Monitor Filter	Component Type	VARIABLES	Monitor Filter	Component Type
condens. mass flow (kg/s)					Cavities
	*.m_jun[*]	Pipe, Tanks & Comb.		*.q[*]	Pipe, 1D Tanks & Comb. Chambers
Pressure drop (Pa)	*.dP_loss	Junctions &Pipe (see note)	**Friction coeff. (-)**	*.fr[]	Pipes
$(\rho v)_{lam}$ **(ρv)**	*.Glam *.G	Valves, Junctions	**$(\rho v)_{crit}$**	*.Gcr	Valves, Junctions
Stem pos. (0-1)	*.pos	Valves			
Non-condens. mass fraction	*.x_nc	Volumes, …	**Vapour mass fraction (-)**	*.x	Volumes, …
	.x_nc[*]	Pipe & Tanks		*.x[*]	Pipe & Tanks
Diluted gas mass fraction	*.xd_nc	Volumes, …	**Void Fraction (-)**	*.alpha	Volumes, …
	.xd_nc[*]	Pipe & Tanks		*.alpha[*]	Pipe & Tanks
Velocity (m/s)	*.vel	Volume, …			Pipe, 1D Tanks & Comb. Chambers
	*.vel[*]	Pipes, 1Dtanks	**Reynolds no. (-)**	*.Re[*]	
Mach number	*.Mach	Volumes, …	**Mach number**	.Mach[*]	
Level	*.level *.Lh	Tank single 1D tank	**Evap. mass flow** **Boil mass flow** **Abs. mass blow**	m_evap m_boi m_abs	1D tank
Power	*.power	Turbo-machinery	**rad/s**	*.omega	Turbo-machinery
Torque	*sh*.T		**rpm**	*sh*.n	
Chemicals mass fractions	bu_eq[*].x[*] "" "" x_chem[*]	Combustors Pipes Volumes	**Chemicals mass fractions (dyn)**	bu_dy[*.x[*]	Combustors
	noz_conc[*].x[*]	Nozzles	**Temp., Pres. Mach**	*_noz[*]	Nozzle variables
Liquid mass fractions	*.x_lf[*], *.x_lo[*]	Combustors			
Pipe end vars	*1, *n	See notes	**Inertance**	*node	Inertance, dL/A

Notes:

- [*] means node variables: 1D Pipes, Combustors and Tanks.
- The Pipe variables that do not depend on the number of nodes are: *T1, Tn, P1, Pn, alpha1, alphan, vsound1, vsoundn, rho1, rhon, vel1, veln* and *arhovel1, arhoveln* (= vsound*rho*vel, the Joukowsky formula) containing the first and the last node static values of the respective variables.
- *dP_loss* accounts for the static pressures losses between the inlet and the outlet of a pipe due to friction, gravity, area change and inertia terms, including the first and the last half node length.
- *Vapor mass fractions (quality, "x" variable) includes the non-condensable gases and the vapor masses*
- The "*bu_eq[*].x[*]*" arrays contain the **mass** fractions of each one of chemicals (x[*]) for each one of the nodes (bu_eq[*]). For example, bu_eq[2].y[CH4] is the equilibrium mass fraction of chemical CH4 at chamber node 2.
- *Units are S.I. (m, rad, Pa, K, kg, kg/s, J, W, W/m^2, etc.)*

Other more specific variables are explained in the libraries documentation (html files, /doc directories). *It should be noted that most of the fluid port variables are not shown to the user to avoid redundancy with the internal variables defined in the components.*

3.5 GLOBAL SIMULATION PARAMETERS

Global simulation parameters are library variables or solver parameters that can be normally defined in the BOUND or BODY blocks of the experiment to control the integration. The most important of these are:

- The Gravity vector components (GRAVX, GRAVY, GRAV) relative to a body axis system are global boundary variables to be defined in the experiment file. In this way, gravity forces are taken into account.
- Gravity accelerations are assumed to be the same for any discretized volume of the pipe component. Centrifugal or Coriolis body forces (as a function of a global rotation speed and the volume position) are not included yet. For lumped volumes, only the vertical vector GRAV is active because the calculated volume height is assumed to be in the vertical direction.
- The library variable "Damp" (the artificial viscosity parameter) is an input to be defined in the experiment for every model. Values from 0.1 to 0.4 provide good smoothing of numerical pressure oscillations. In normal cases, Damp = 0.4-0.6 slightly smooth water-hammer pressure peaks, but saves CPU time. Use Damp = 1 or greater in priming cases or when acoustic effects are not important.
- Mixture of chemicals (in Combustors and any other component placed downstream of a combustor) will be calculated with the Perfect gas or the Van der Walls state of equation depending on the boundary global variable **VDW_option** = 0 or 1 respectively applying to all the components of a model
- The absolute and relative error permitted in the integration (REL_ERROR and ABS_ERROR experiment variables) should be between 1e-4 and 1e-6. New models should use REL_ERROR and ABS_ERROR = 1e-5. Then, the user can try other smaller or greater values depending on the stability of the model. In normal cases, set these variables to 1e-4 to speed up the calculation. In priming cases with non-condensable gases, lower values are normally used in the experiment.

3.6 USING WRITE / FOR COMMANDS IN THE EXPERIMENT

The statements WRITE and FOR can be used in the experiment file to perform sensitivity studies. For example:

In a transient case:

```
WRITEF ("Results.res","\n MR    dP   A1   A2   A3")

WHILE (INTEG_CINT() != INTEG_END )
WRITEF("Results.res","\n%g ",PreBurner.Combustor.MR)
END WHILE
```

In a steady case (using the STEADY library in a particular case):

```
WRITEF ("Results.res","\n MR dP A1 A2 A3")
FOR(i IN 1,5)
MR = 0.8 + (i-1)/10
PreBurner.MR_o = MR
   NEW_BRANCH("MR=$MR")
FOR(j IN 1,5)
dP = 80e5 + (j-1)*10e5
     J1.dP_design = dP
J2.dP_design = dP
STEADY()
```

```
        -- this WRITE allows to generate an ECEL file to be post processed later
    WRITEF ("Results.res","\n%g %g %g %g",MR,dP,J1.A,J2.A,J3.A)
    END FOR
    END FOR
```

3.7 SAVING / RESTORING STATE

It is possible to save /restore the state of a model at a defined time by means of the EcosimPro related commands (SAVE_STATE/ RESTORE_STATE) in the experiment file.

For example, the user may want to save all the variables of a model after a "difficult" execution where a steady state is reached. This will make it possible to restart another execution with the same conditions as those of the previous run but changing some boundaries or data. See the example below:

First execution:

```
...
BODY
...
  TSTOP = 5
  INTEG()
-- save state at Time = 5
 SAVE_STATE("mySteady")
END EXPERIMENT
```

Second execution:

```
BOUNDS    -- set expressions for boundary variables: v = f(t,...)
  -- Boundaries can be changed
...
BODY
-- restore a previous state
  SET_INIT_ACTIVE(FALSE)      -- deactivate INIT blocks
  RESTORE_STATE("mySteady")
...
-- changing some input data if necessary
  Chamber.A_inj_oxy = 0.0005 -- 0.0005 --
  Chamber.A_inj_red = 0.0005 -- 0.0005 --

-- new integration from 5 to 15 seconds
  CINT = 0.1
  TSTOP = 15
  INTEG()
END EXPERIMENT
```

3.8 PARAMETRIC STUDIES

ESPSS can properly use the command RESET_VARALBLES so there is no need to reload the properties files. For example, in making parametric studies, it is possible to perform several different runs (INTEG) starting from the same initial conditions but changing some of the input data. See for example:

```
-- First calculation reaching some steady conditions
IMETHOD = DASSL_SPARSE
REPORT_MODE=IS_STEP
REL_ERROR = 1e-4 --1e-5
ABS_ERROR = 1e-4 --1e-5
TIME = 0
CINT = 0.01
TSTOP = 0.1
...
Pipe.D = 0.10
INTEG()-- The first call to INTEG() internally calls the EXEC_INIT() command !!

-- New calculations changing some data
```

```
FOR (i IN 1,10)
    RESET_VARIABLES()
    Pipe.D = 0.15+ (i-5)/100 --New pipe input data value
    …
    EXEC_INIT()

    TIME = 0
    INTEG()
END FOR
```

3.9 SELECTING MATERIAL PROPERTIES

Wall materials (Pipe, Tank and Combustion Chamber components) are selected via the ENUMERATIVE variable "mat" in which a list of materials is provided. Material properties are included in the THERMAL library [RD-3]:

The properties are interpolated from external data contained in files (1D temperature tables) in the THERMAL/TABLES directory. The use of external data files has important advantages:

- The properties can be easily changed or completed by just modifying the external data file. There is no need to modify the EcosimPro library.
- Coding of the component is far less dependent on the type of the material

The users can build their own properties files. They must be named as follows: MatUsr1.txt, MatUsr2.txt or MatUsr1.txt, and added to the THERMAL/TABLES directory. At EcosimPro level, it is enough to indicate the same name for "mat" in the attributes editor of a component using material properties.

The format of these properties files is as follows. For example, for the AL2219:

Register no.	Values	Comments
1	AL2219	Material name. (Same name as the file name)
2	3 2 34	Number of constant properties, number of variable properties and number of points for variable properties
3	2800	Numerical value of the first constant property (density in this case)
4	70.E9	Numerical value of the second constant property (E modulus in this case)
5	0.30	Numerical value of the third constant property (Poisson Modulus in this case)
6	20 30	Minimum allowed temperatures for variable properties (one column per variable)
7	600 600	Maximum allowed temperatures for variable properties (one column per variable)
8	20 23.98 22	T1 Var1(T1) Var2(T1). In this case: T1 Cp k
…	…	…
8+no of temper.	600 1000.0 155	Tn Var1(Tn) Var2(Tn).

Currently available materials are listed in the following table:

MATERIALS				
Aluminium	AL2219	Al5083	AL3003_F	AL6061_T6
AL7020	Carbon	Carbon Steel	Copper	GCF or Epoxy fiberglass
Cryosof	H920A	Dacron filled with helium,	VIP (Polystyrene foam)	Dacron filled with nitrogen
H920A	SS_304	SS_304L	SS_310	SS_316
SS_321	Titanium	MatUsr1	MatUsr2	MatUsr3

where SS is Stainless Steel, AL is Aluminium alloys and MatUsrX is user customized material..

4. FLUID_PROPERTIES LIBRARY

4.1 OVERVIEW

4.1.1 Purpose of the library

FLUID_PROPERTIES is an ECOSIMPRO library entrusted with calculating fluid properties. The functions available in this library are mainly used by the FLUID_FLOW_1D library to simulate fluid systems. The most important features are summarized below:

- Most of the fluids used for rocket applications are available
- Fluids are supported in different categories depending on the type used, see §5.1.2:
    1. Perfect gases (transport and heat capacity properties obtained from CEA coefficients (temperature dependent)).
    2. Van der Waals gases (transport and heat capacity properties obtained from CEA coefficients (temperature dependent)).
    3. Perfect gases (transport and heat capacity properties interpolated in files (temperature dependent)).
    4. Simplified liquids interpolated in tables (properties not pressure dependent)
    5. Real fluids interpolated in tables considering either liquid, superheated, supercritical or two-phase flow (temperature and pressure dependent).
- Mixtures of a real fluid with a non-condensable gas are allowed. The homogeneous equilibrium model is used to calculate the properties (quality, void fraction, etc.) in case of two-phase flow. Mixtures of two real fluids (fractional distillation) are not allowed.
- User-defined fluids are available. The properties must be defined in external data files and can be of any of the last three types previously mentioned.

The FLUID_PROPERTIES library does not contain any components. It only provides a large collection of functions that return the value of a fluid property (or the complete thermodynamic state) by introducing relevant parameters (i.e. temperature, internal energy, pressure, density, etc.).

4.1.2 Library items

The different enumeration types define series of labels for which the corresponding properties will be predefined or calculated using thermodynamic functions.

Table 4-1 – Fluid keys and Library Items

FluidKeys (Possible fluids grouped by categories)	{PfGas_Air, PfGas_N2, PfGas_O2, PfGas_CO2, PfGas_CO, PfGas_CH4, PfGas_H2, PfGas_He, PfGas_Ar, NoFluid, PfGas_Usr1, PfGas_Usr2, PfGas_Usr3, PfGas_Usr4, PfGas_Usr5, PfLiq_H2O, PfLiq_IPA, PfLiq_MMH, PfLiq_MON1, PfLiq_MON3, PfLiq_N2H4, PfLiq_N2O4, PfLiq_O2, PfLiq_H2, PfLiq_UDMh, PfLiq_Oil, PfLiq_Kerox, PfLiq_Usr1, PfLiq_Usr2, PfLiq_Usr3, PfLiq_Usr4, PfLiq_Usr5, Real_N2, Real_CO2, Real_H2O, Real_He, Real_Xe, Real_Ar, Real_Air, Real_O2, Real_N2O, Real_NTO, Real_H2n, Real_H2p, Real_MMH, Real_hydrazine, Real_CH4, Real_C2H6, Real_C3H8, Real_C4H10, Real_C4H10_i, Real_C12H26, Real_ethylene, Real_propylene, Real_ethanol, Real_ammonia, Real_R12, Real_R22, Real_R134A, Real_R141B, Real_Usr1, Real_Usr2 , Real_Usr3 , Real_Usr4 , Real_Usr5,}

Chemicals (Possible chemicals constituents according to CEA)	{ H2, O2, H2O, H, O, OH, H2SO4, SO2, SO, N2, NO, CO, CO2, NH3, NO2, N2O, H2O2, HO2, HNO, HNO2, COOH, HCO, HCOOH, Ar, CH4, C3H6, C_gr, MMH, N2O4, HCN, C2H4, C2H6, Jet_A, HCHO, O3, Air, He , N, Al2O, N2H4, C2H2, C3H8, C4H10, UDMH, CH3OH, C2H5OH,C4H10i, NH4NO3_IV, NH4CLO4_I,S2, K, KNO3_a, KNO2, K2SO4, Al2O3, Cl, HCl, AlCl, AlOCl, AlO, S_a, AlS, H2S, SH, Al2O2, Al_cr, AlOH, AlO3H3 }
FluidProps	{fprop_pressure, fprop_temperature, fprop_density, fprop_volume, fprop_energy, fprop_enthalpy, fprop_entropy, fprop_viscosity, fprop_conductivity, fprop_beta, fprop_kappa, fprop_vsound, fprop_cp, fprop_gamma, fprop_sigma, fprop_void_fr, fprop_quality, fprop_phi}
INIT_OPTION	{INIT_PT, INIT_PX, INIT_TX, INIT_RT, INIT_RPT}
Phase	{vapor, liquid, two_phase, supercritical}

The variable **FluidKeys** identifies the fluid and its category (Perfect gas, Perfect liquid or Real fluid). A property file is available for each one of the fluids not defined by the user.

The variable **Chemicals** identifies the different chemical constituents whose properties can be calculated in the same way as in the CEA code [RD-9]. Typical solid propellants are included.

ESPSS libraries will use the INIT_OPTION flag to be able to initialize a fluid volume in different ways (see *Init_Vol* function in Table 4-4).

4.1.3 Property files

Properties (except for those related to the state equations of a perfect gas or liquid) are calculated based on the interpolation in external data contained in files (1D or 2D data tables). The use of external data files has important advantages with respect to applying complex state equations:

- The properties can easily be changed or completed just modifying the external data file. No (or very little) modification will be needed in the Libraries
- The calculation speed is higher because the interpolation algorithms make use of the last entry search in to the 2D tables and do not need to evaluate complex (iterative) state equations
- Coding of the component is far less dependent on the type of state equation used

The users may build their own property files. They must be named as follows: *PfGasUsr1.dat …, PfLiqUsr1.dat* … or *RealUsr1.dat* … (depending on the type) … and added to the PROP_TABLES directory. At EcosimPro level, it is enough to indicate the corresponding name (PfGas_Usr1, … PfLiq_Usr1 … Real_Usr1, …) in the "WorkingFluid" component (see §5.4.24).

Chapter 4.3 gives the formats that must be respected when building property files. Pre-established user-defined fluids (*PfGasUsr1.dat …, PfLiqUsr1.dat* … or *RealUsr1.dat*) do not require any modification of the code to be used, but they cannot be used as propellants because the code does not know what the chemical composition of the new fluid is. *Chapter 4.3.4 describes the modifications to be made in the code by adding new fluids (with real or simplified property files) linked to a specific CEA chemical*.

The PROP_TABLES folder, which must be under the FLUID_PROPERTIES folder, is delivered with ESPSS. The CEA property files (thermo.lib, transport.lib) containing CEA coefficients (see §4.4.1) will also be read in this folder at execution time. The user is responsible for any updates to these files. The tables below summarize the applicability ranges in terms of P/T for the available fluids:

Table 4-2 – Real fluid ranges in terms of P/T

Real Fluids	LiquidRange		GasRange	
	Pres. (bar) (Psat – Pcr)	Temp.(K) (Tsat – Tcrit)	Pres. (bar)	Temp. (K) (Tmin – Tmax)
ParaHydrogen	Psat – 12.84	Tsat – 32.94	0. – 1200	13.8 – 400
NormalHydrogen	Psat – 13.03	Tsat – 33.24	0. – 500	18.0 – 700
Dodecane	Psat – 18.17	Tsat – 658.1	0.– 3500	263.6 – 1000
Methane (CH_4)	Psat – 45.99	Tsat – 190.6	0 – 10000	90.7 – 625
Propane (C_3H_8)	Psat – 42.47	Tsat – 369.8	0 – 1030	85.5 – 623
Butane (C4H10)	Psat – 37.96	Tsat – 425.1	0. – 690	134.9 – 637
MMH	Psat – 80.64	Tsat – 567.	0 – 350	220. – 620
Hydrazine	Psat – 149.4	Tsat – 653.5	0 – 400	182.3 – 710
Oxygen (O_2)	Psat – 50.43	Tsat – 154.6	0. – 820	54.4 – 1000
N_2O	Psat – 72.45	Tsat – 309.5	0 – 500	182.3 – 525
N_2O_4	Psat – 101.3	Tsat – 431.15	0 – 300	261.9 – 600
Ethanol	Psat – 61.48	Tsat – 513.9	0. – 2800	250 – 770
Ethane	Psat – 48.72	Tsat – 305.3	0. – 700	90.3 – 625
Ethylene	Psat – 50.42	Tsat – 382.3	0. – 3000	122.0 – 450
Propylene	Psat – 46.65	Tsat – 365.6	0. – 2000	100.0 – 600
Helium (He)	Psat – 2.27	Tsat – 5.19	0. – 1000	0.8 – 1500
Nitrogen (N_2)	Psat – 33.96	Tsat – 126.2	0. – 22000	63.15 – 2000
Argon (Ar)	Psat – 48.63	Tsat – 150.7	0. – 10000	83.8 – 700
Xenon (Xe)	Psat – 58.42	Tsat – 289.7	0. – 7000	161.4 – 750
Air	Psat – 37.85	Tsat – 132.6	0. – 700	64.0 – 1000
Water (H_2O)	Psat – 220.6	Tsat – 647.1	0. – 10000	273.2 – 1275
Ammonia (NH_3)	Psat – 113.3	Tsat – 405.4	0. – 10000	195.5 – 700
R12	Psat – 41.36	Tsat – 385.1	0. – 2000	116.1 – 578
R22	Psat – 49.90	Tsat – 369.3	0. – 600	115.7 – 554
R134A	Psat – 40.59	Tsat – 374.2	0. – 700	169.8 – 561
R141B	Psat – 42.12	Tsat – 477.5	0. – 4000	169.7 – 716

Table 4-3 – Ideal fluid temperature ranges

Perfect Liquids	Temp. (K) Range	**Perfect Gases**	Temp. (K) Range
Water	273– 523	Air	200 – 1000
IPA	183 – 333	Argon	200 – 1000
Keroxene (to be revised by the user)	273 – 600	CH4	200 – 1000
MMH	218 – 333	CO2	200 – 1000
MON1	258 – 333	CO	200 – 1000
MON3	258 – 333	H2	200 – 1000
NTO	258 – 333	He	2 – 1500
N2H4	273 – 333	O2	200 – 1000
O2	57 – 213		
Oil(to be revised by the user)	273 – 600		
UDMH	218 – 333		
Water (H_2O)	273.2 – 1275		
Ammonia (NH_3)	195.5 – 700		
Ethanol	250 – 770		

Notes:
1 Perfect gas and simplified liquid property files should be enlarged by the user if greater ranges or more accurate data are required.

4.2 CLASSIFICATIONOF FUNCTIONS

4.2.1 Main level. EL functions

The functions listed in Table 4-4 are used to calculate a given property, or a complete state of a mixture of fluids. The argument list is in alphabetical order.

Table 4-4 – Main level of thermo-dynamic functions

NAME	Argument list	TYPE	DESCRIPTION
Chemicals functions according to CEA			
PgasEnergy (Returns Cp, H or S according to CEA depending on ical option)	T	REAL	Temperature
	chem	ENUM Chemical	Chemical name of the substance
	ical	INTEGER	1, Cp calculation; 2 enthalpy; 3 entropy
PfGas_cond_vs_T, PfGas_visc_vs_T (Return conductivity, viscosity according to CEA)	chem	ENUM Chemical	Chemical name of the substance
	T	REAL	Temperature
Cp_mix, H_mix S_mix (Return Cp, H or S of a mixture according to CEA)	mix	SET_OF(Chemicals)	Set of Chemicals present in the mixture
	x[mix]	REAL vector	Mass fraction of every chemical in the mixture
	T	REAL	Temperature
T_mix_vs_H T_mix_vs_S (The reverse function of H_mix and S_mix)	mix	SET_OF(Chemicals)	Set of Chemicals present in the mixture
	x[mix]	REAL vector	Mass fraction of every chemical in the mixture
	H (or S)	REAL	Enthalpy or entropy
Trans_mix (Transport properties of a mixture according to CEA)	mix	SET_OF(Chemicals)	Set of Chemicals present in the mixture
	x_eq[mix]	REAL vector	Mass fraction of every chemical in the mixture
	T	REAL	Temperature
	eta_mix	REAL	Viscosity
	lambda _mix	REAL	Conductivity
Fluid functions			
FL_CritProp (Critical mass flow calc.)	fluid	ENUM FluidKeys	Main working fluid
	fprop	ENUM FluidKeys	Wanted fluid property
	ier	OUT INTEGER	Error code
FL_Gcrit_fun (Returns the critical flow per unit of area)	P	IN REAL	Pressure
	T	IN REAL	Inlet temperature
	P_nc	IN REAL	Non-condensable partial pressure
	fluid	IN ENUM	Main working fluid
	ier	OUT INTEGER	Error code
	rho	IN REAL	Density
	v	IN REAL	Velocity
	vsound	IN REAL	Speed of sound
FL_Tsat_vs_p (Saturation temperature vs pressure)	fluid	IN ENUM	Main working fluid
	hf	OUT REAL	Enthalpy of saturated liquid
	hg	OUT REAL	Enthalpy of saturated gas
	ier	OUT INTEGER	Error code
	p	IN REAL	Pressure
	rhof	OUT REAL	Density of saturated liquid
	rhog	OUT REAL	Density of saturated gas
	jy	OUT INTEGER	Pointer into tables

NAME	Argument list	TYPE	DESCRIPTION
FL_psat_vs_T (Saturation pressure vs temperature)	fluid	IN ENUM	Main working fluid
	ier	OUT INTEGER	Error code
	jy	OUT INTEGER	Pointer into tables
	T	IN REAL	Temperature
FL_state_vs_ru **(Complete state vs. density and energy)** **&** **FL_state_vs_ph** **(Complete state vs. pressure and enthalpy)**	mix	SET_OF(Chemicals)	Set of Chemicals present in the mixture
	x_mix	IN REAL	Mass fractions composition
	P_nc	OUT REAL	Non-condensable partial pressure
	T	OUT REAL	Temperature
	Tsat	OUT REAL	Saturation temperature
	alpha	OUT REAL	Void fraction
	cond	OUT REAL	Thermal conductivity of the mixture
	condf	OUT REAL	Thermal conductivity of saturated liquid
	condg	OUT REAL	Thermal conductivity of saturated vapor
	cp	OUT REAL	Cp of the mixture
	cpf	OUT REAL	Cp of saturated liquid
	cpg	OUT REAL	Cp of saturated vapor
	drho_dh	OUT REAL	drho/dh at constant p
	drho_dp	OUT REAL	drho/dp at constant h
	fluid	IN ENUM	Working fluid
	fluid_nc	IN ENUM	Non-condensable working fluid
	hf	OUT REAL	specific enthalpy of saturated liquid
	h	IN REAL	Mixture specific enthalpy (only in **FL_state_vs_ph)**
	hg	OUT REAL	specific enthalpy of saturated vapor
	ipx	OUT INTEGER	Last index for x interpolation
	ipy	OUT INTEGER	Last index for y interpolation
	ier	OUT INTEGER	Error flag
	p	REAL	Pressure (outlet in **FL_state_vs_ru)** **(**inlet in **FL_state_vs_ph)**
	phase	OUT ENUM	Phase
	x	OUT REAL	Quality –gas mass fraction- including vapour and non-condensable gases
	rho	IN REAL	Gas/liquid mixture density
	rhof	OUT REAL	Density of saturated liquid
	rhog	OUT REAL	Density of saturated vapor
	sigma	OUT REAL	Surface tension
	u	REAL	Mixture specific energy (inlet in **FL_state_vs_ru)** **(**outlet in **FL_state_vs_ph)**
	visc	OUT REAL	Viscosity of the mixture
	viscf	OUT REAL	Viscosity of saturated liquid
	viscg	OUT REAL	Viscosity of saturated vapor
	vsound	OUT REAL	Speed of sound of the mixture
	x_nc	IN REAL	Non-condensable mass franction
FL_prop_vs_pT **FL_prop_vs_rhou** **(return a property)**	mix	SET_OF(Chemicals)	Subset of species
	x[mix]	REAL vector	Mass composition of chemicals
	P	IN REAL	Pressure
	…	IN REAL	Second input variable (T, h, rho, x, etc.)
	fluid	IN ENUM	Main working fluid
	fprop	IN ENUM	Wanted fluid property
	ier	OUT INTEGER	Error code
Init_Vol	mix	SET_OF(Chemicals)	Subset of species
	xp[mix]	REAL vector	Fuel mass composition
	P_o	IN REAL	Initial Pressure (if it is used)
	P	OUT REAL	Effective Pressure
	P_nc	OUT REAL	Non-condensable pressure
	T	OUT REAL	Effective Temperature
	T_o	IN REAL	Initial temperature (if it is used)
	alpha	OUT REAL	Void fraction
	c	OUT REAL	Speed of sound
	fluid	IN ENUM	Main working fluid

NAME	Argument list	TYPE	DESCRIPTION
(To initialize a fluid capacity)	fluid_nc	IN ENUM	Non-condensable working fluid
	visc	OUT REAL	Viscosity of mixture
	cond	OUT REAL	Thermal conductivity of mixture
	jx	OUT INTEGER	Pointer into table
	jy	OUT INTEGER	Pointer into table
	ier	OUT INTEGER	Error flag
	init_flag	IN ENUM	Init option
	x_o	IN REAL	Initial quality (if it is used)
	qu	OUT REAL	Quality –gas mass fraction- including vapour and non-condensable gases
	rho_o	IN REAL	Initial density (if it is used)
	rho	OUT REAL	Gas/liquid mixture density
	u	OUT REAL	Mixture specific energy
	x_nco	IN REAL	Initial non-condensable mass fraction (if it is used)
	x_nc	OUT REAL	Non-condensable mass fraction

Two main separate groups of functions are provided in Table 4-4:

- The first set of functions (*PfGas_..., Cp_mix, H_mix, T_Mix_vs... and Trans_mix*) refers to the functions that calculate properties according to CEA code.
- The remaining functions, compiled in a FORTRAN library called "*thermo_table_interp.lib*", are based on external data file interpolation. These FORTRAN functions will provide a real fluid behavior (*using 2D property tables*) or an ideal one, the main level of the functions splitting the calculation into several categories of functions depending on the fluid type.

The table below shows the main functions of the FORTRAN library

Table 4-5 – Main FORTRAN functions to calculate the properties of the fluids

Function name	Function value	Arguments
thermo_prop	(subroutine)	ig: Fluid key identifying fluid & type (real of simple) j1, v1: Nist number (see Table 4-6) and value of the first input j2, v2: Nist number (see Table 4-6) and value of the second input iwant: Output key: If the 1's digit > 0: out(1- 7,1) = State Variables If the 10's digit > 0 out(11-17,1) = Derivatives If the 100's digit > 0:out(19-20,1) = Cond. And Visc. Out(20,3): outlet (see below) x: Quality (0=liq, 1=gas, gas mass fraction) ier: Error code (see Figure 4-2): nor: Interpolation order (1 lineal, 2 parabolic, etc.) jt: Index used to speed up the searching (temperature variable) in thermodynamic tables (I/O). jp: Index used to speed up the searching (pressure variable) in thermodynamic tables (I/O).

Function name	Function value	Arguments
tsat_vs_p	Tsat	ig, P, n_or, r_l, r_g, h_l, h_g, iprp, ier See thermo_prop r_l, h_l : ρ, h of saturated liquid (outlet) r_g, h_g : ρ, h of saturated gas (outlet)
psat_vs_t	Psat	ig, T, n_or, iprp, ier See thermo_prop

The main input arguments of the "themo_prop" function are: j1-v1 and j2-v2.

- If the state variables are RU, then j1=3, v1=ρ and j2=7, v2=energy.
- If the state variables are RT, then j1=3, v1=ρ and j2=2, v2=T, etc.

The main output arguments of the "themo_prop" function are:

- The quality
- "out" matrix with all the properties according to the table below:

The properties returned by the "thermo_prop" function are:

Table 4-6 – Properties returned by the "thermo_prop" function

out(1,1)	P	Pressure (Pa)
out(2,1)	T	Temperature (K)
out(3,1)	ρ	Density (kg/m3)
out(4,1)	v	Specific volume (m3/kg)
out(5,1)	h	Enthalpy (J/kg)
out(6,1)	s	Entropy (J/kg/K)
out(7,1)	u	Energy (J/kg)
out(11,1)	cp	Specific heat at constant pressure (J/kg/K)
out(12,1)	cv	Specific heat at constant volume (J/kg/K)
out(13,1)	γ	Isentropic exponent (-)
out(14,1)	β	Volumetric expansively (1/K)
out(16,1)	κ	Isothermal compressibility (1/Pa)
out(17,1)	vsound	Sound speed (m/s)
out(19,1)	λ	Conductivity (W/m/K)
out(20,1)	μ	Viscosity (Pa.s)

Note 1: The first index of the "out" matrix is the variable identification according to the numbering indicated in Table 4-6. The second index is used for the saturation properties, which are also returned:

- out(i,1) contains all variables for the actual (sub-cooled, re-heated or two-phase) conditions
- out(i,2) contains all variables for the liquid saturation conditions at current pressure.
- out(i,3) contains all variables for the vapor saturation conditions at current pressure.

Note 2: The "*out*" FORTRAN indexes are reversed calling this routine from a C++ or EL function.

4.2.2 Software breakdown of the FORTRAN routines

Three main levels of calculation in the routines exist:

- User interface (*thermo_prop, p_vs_rhou, u_vs_pt* routines (see above)
- Searching procedures in diagrams H-S, P-T, … (*y_f_px* and *p_f_xy* routines).
- 1-D, 2-D interpolation routines, including triangular zones and extrapolation control.

Table 4-7 – "Searching procedures" routines

Name of the function	Inputs	Function's value	Argument list
lee_tab (Reading of the property tables)	-	None	*ig, ier* Note: "*lee_tab*" routine is automatically called by the main level routines only when the properties file has not been read yet
y_f_px	Pressure, X : t, ρ, v, h, s, or u	Y (a property)	Similar to the previous function but P,x are the inputs in this case, instead of y,x.
p_f_xy	X : ρ or s Y : t, h, or u	Pressure	*ig*: ID number of the fluid. *Ix*: ID of the x-variable according to NIST *iy*: ID of the y-variable *x* Value's variable type ix. *Y* Value's variable type iy *n_or*: Degree of interpolation *qu*: Quality *his_t*: Same meaning as in "thermo_prop", Table 4-5 *his_p*: " " *ier:* Error code

Table 4-8 – Interpolation routines

Name	Description	Argument's list
int_3p	Reverse interpolation in 2 dimensions : p = f(x,y) Error codes (see Figure 4-2): ➢ 0, no error ➢ 1,extrap. In the left-down hand side of the table ➢ 2,extrap.centre-down hand side ➢ 3,extrap. Right-down hand side ➢ 4,extrap. Right-centre hand side ➢ 5,extrap. Up-right hand side ➢ 6,extrap. Up-centre hand side ➢ 7,extrap. Up-left hand side ➢ 8,extrap. Left-centre hand side ➢ 10, between curves ➢ 99, Insufficient points	*x* 1D vector of the x-coordinates. If the first curve has 5 points, x(6) will be the first x of the second curve, etc. *y* Similarly as for x for the y coordinates *p* Vector with the characteristic parameters of the curves *nx* Vector with the number of points of each curve *np* Number of curves *ngx* Degree of interpolation in X (1, linear: 2, parabolic, etc.) *ngp* Degree of interpolation in Y (1, linear: 2, parabolic, etc.) *xbar* Value of X (input) *ybar* Values of the function (input) *pbar* Value of the outlet in P *ier* Error code. See description *his_t*: See Table 4-5 *his_p*: See Table 4-5
int_3r	Direct interpolation in 2 dimensions: y = f(p,x)	Same list as before but this time instead of x,y, the values of the input variables are P,x

Name	Description	Argument's list
int_2	Interpolation in 1D Error codes : ➢ 0, no error ➢ -1, extrap. From left ➢ 1, extrap. From right ➢ 3, Insufficient points	*xbar* Input value in X *x* Vector with x-coordinates of the curve. *Y* Vector with y-coordinates of the curve *nx* Vector with the number of points of each curve *ng* Degree of interpolation in X (1, linear: 2, parabolic, etc.) *n* number of points of the curve *ybar* Y-value interpolated *ier* Error code. See description *his_t*: See Table 4-5
find_cell	Find the number of the point in the x-vector previous to x-bar	*xbar* Input value in X *x* Vector with x-coordinates of the curve *nx* Vector with the number of points of each curve *n* number of points of the curve *ier* Error code. See before *prev*: Number of the point in the x-vector previous to xbar
zreal,	Computation of x that makes of a non-linear equation function f(x) equal to 0. The secant method is used.	*External f* : Name of the function that is required to cancel *eps*: Maximum absolute error allowed. *Nsig*: Number of significant figures of the x-computation. *Xl, xr*: Solution limits *itmax*: Maximum number of iterations (input) and iterations carried out (output) *ier* Error code

Notes:

- The 2D interpolation functions can work with a different number of points in each curve. However, the curves cannot cross each other.
- The 2D functions allow the interpolation within the triangles formed by the edge points of the curves. The bounds created by the first or the last points have to be monotonous in x.
- The interpolation function in 1D must be monotonous in x.

4.2.3 Example of use of thermo functions calls

Below some examples of use of thermo functions calls that will help you build your own applications.

Calling pure fluid property functions:

```
FUNCTION NO_TYPE example_1(SET_OF(Chemicals) mix, REAL x[mix],
       ENUM FluidKeys fluid, REAL P1, REAL h1,
       OUT REAL T1, OUT REAL s, INTEGER ier)

BODY
       -- Properties dependence on the gases composition or pure fluid:
       -- fluid = Comb_Gas1, then the function FL_prop_vs_... will use
       --          the chemicals compos, x[]
       T1 = FL_prop_vs_ph(Chemicals, x, fluid, P1, h1, fprop_temperature,T1, ier)
       -- Note that T1 is also used as input for iteration purposes
       s  = FL_prop_vs_ph(Chemicals, x, fluid, P1, h1, fprop_entropy, T1, ier)
       …
       -- Other functions as FL_prop_vs_rhoT, FL_prop_vs_Px, FL_prop_vs_Tx
       -- do not need Chemicals, x, as inputs
       …
END FUNCTION
```

If *fluid* is always a pure fluid, "chemicals" and mass fractions inputs can be avoided:

```
FUNCTION NO_TYPE example_2(SET_OF(ENUM FluidKeys fluid, REAL P1, REAL h1,
      OUT REAL T1, OUT REAL s, INTEGER ier)

DECLS
      REAL dummy1[noBurnGases]  -- noBurnGases is a FLUID_PROPERTIES variable type
BODY
      -- In this case noBurnGases, dummy1: arguments are not used, just declared:
      T1 = FL_prop_vs_ph(noBurnGases,dummy1, fluid, P1, h1, fprop_temperature,T1, ier)
      s  = FL_prop_vs_ph(noBurnGases,dummy1, P1, h1, fprop_entropy, T1, ier)
…
END FUNCTION
```

Other more complicated examples of thermo functions calls can be found in the ESPSS code, for example initializing state variables in the INIT block of the Capacity abstract component.

4.3 PROPERTY FILES DESCRIPTION AND GENERATION

4.3.1 Perfect Gases

For perfect gases, the following properties (columns) will be read in the corresponding file (see §4.1.3 and 2), one row per temperature. For example, for Air, this is the listing of the property file:

PfGas_Air 28.958538

Temp	Cp	visc	cond
200	1004.37722	1.43692141e-005	0.0199273382
208	1004.37722	1.46537806e-005	0.0203219773
…			
1000	1141.5086	4.17737195e-005	0.0672072485

The numeric value of the first line is the molecular weight. The format is free. New or different values can be added to the file, so that the fluid properties of the corresponding fluid will be changed. The user may build his own property files as explained in §4.1.3.

4.3.2 Simplified Liquids

For simplified liquids, the following properties (columns) will be read in the corresponding file (see §4.1.3), one row per temperature. For example, below is the listing of the property file for the N2O4:

PfLiqN2O4 92.011

Temp	dens	Cp	vsound	cond	visc	Pvap
258.15	1522.9	1438.65	1163.69125	0.1445	0.0006173	15280
261.9	1514.6	1458.22	1147.0975	0.1439	0.0005931	19018
…						
333.15	1353.1	1733.03	831.81625	0.1069	0.00026	506700

The column Pvap is not used at the moment, perfect liquids assume zero vapor pressure. The numeric value of the first line is the molecular weight. The format is free. New or different values can be added to the file, so that the fluid properties of the corresponding fluid will be changed. You can build your own property files as explained in §4.1.3.

The property files for simplified liquids have been taken from the following references:

- MMH, UDMH, N2H4 according to reference RD-12 (Schmidt)
- IPA: Aiche Data Compilation and http://www.rshydro.co.uk/sound-speeds.shtml
- MON1, MON3 & N2O4: RD-14 (USAF Propellant Handbooks) and RD-47 (Rockwell) for MON densities
- Other fluids from RD-8 (NIST data base)

4.3.3 Real fluids

4.3.3.1 Property files generation

4.3.3.1.1 NIST Properties

In this case, the data tables have been generated running the **REFPROP** (NIST) program [RD-8] and saving the results in files. Property table generation was an independent task carried out prior to the development of the EcosimPro thermodynamic routines.

Pressure is used as a parameter and temperature as the first independent variable. The tables cover the liquid, vapor and super-critical zones. Tables are generated to guarantee continuity between the two-phase, vapor, liquid and super critical zones.

Most of the real property files have been generated with the REFPROP code.

4.3.3.1.2 EXCEL Tool generating Peng-Robinson's state equation Properties

Some fluids (Hydrazine, MMH, and UDMH) are not available using the NIST program. For these cases, the following methods (see references RD-10, RD-11 and RD-12) have been implemented in an EXCEL tool building the property files for a variable range of temperatures and pressures:

- A/ *Density (= 1/v) calculation according to the Peng-Robinson equation of state*:

$$P = \frac{R_g T}{v-b} - \frac{a\alpha}{v^2 + ubv + wb^2}; \quad \alpha = (1 + m(1 - T_r^{0.5}))^2; \qquad T_r = T / T_{cr}$$

$$a = \frac{0.45724 R_g^2 T_{cr}^2}{P_{cr}}; \qquad b = \frac{0.07780 R_g T_{cr}}{P_{cr}}$$

$$m = 0.37464 + 1.54226\omega - 0.26992\omega^2; \qquad \omega = Ac.factor$$

- B/ *Enthalpy and entropy properties based on the calculation of the non-ideal term* ($T\left(\partial P / \partial T\right)_V$ -P) to be added to the Cp(T) polynomial adjustment coefficients calculated at zero pressure (ideal gas):
  - $\Delta h = \int_{To}^{T} C_v(T)dT + \int_{\infty}^{v} (T\left(\frac{\partial P}{\partial T}\right)_V - P)dV + PV$
  - $\Delta s = \int_{To}^{T} \frac{C_v(T)}{T} dT + R\ln\frac{V}{Vo} + \int_{\infty}^{v} (\left(\frac{\partial P}{\partial T}\right)_V - R/V)dV$

  Assuming Cv(T) =A-Rg+BT+CT^2+DT^3, the previous integrals result in the following expressions:

$$h = h_0 + f(1+m)\sqrt{\alpha} + p \cdot v + (A - R_g)T + BT^2/2 + CT^3/3 + DT^4/4$$

$$s = s_0 + f \cdot m \cdot \sqrt{\alpha}\sqrt{T_r}/T + R_g Ln(v-b) + (A - R_g)Ln(T) + BT + CT^2/2 + DT^3/3$$

where

$$f = \frac{a}{b \cdot fic} Ln\left(\frac{2v + b(u - fic)}{2v + b(u + fic)}\right); \qquad fic = \sqrt{u^2 - 4w}$$

- C/ Saturation pressure is assumed to be a polynomial or exponential adjustment of temperature.
- D/ Transport properties are assumed to be a polynomial adjustment of temperature only.

The *gen_prop.xls* EXCEL tool consists of two main sheets:

- "*MainData*" sheet: This sheet contains all the data needed to run the macros generating the properties file.
- "*MainPlots*" sheet: This sheet contains the "skeleton" of the plots of the main properties.

Clicking on the Command Button "Generate ESPSS propFile" of the "*MainData*" sheet, the corresponding Visual Basic routines are executed. As a result, the "*MainPlots*" sheet will be filled with the actual properties and two new sheets will be generated:

- "PropSat" sheet with the saturation line properties. The Clausius Clapeyron equation has been added to test the error committed by comparing the (Hgas-Hliq) value derived from the integration of the state equation with the value obtained from the Clausius Clapeyron equation.
- "PropFile" sheet with the complete set of properties according to the real property ESPSS file. This sheet is ready to be exported to the PROP_TABLES directory as a new ESPSS real property file.

The Command Button "Plot particular Graphs" is available in the "MainPlots" sheet to draw any property as a function of another one.

The EXCEL macros are completely automatic and will use the variables defined as "names" in the "*MainData*" sheet. The corresponding VB routines are simple (even though the complex formulation solving cubic state equations) and they are programmed in a straightforward way to allow you to identify the different input data and to modify correlations if necessary.

The "MainData" sheet is shown in Figure 4-1. The user should enter each piece of data for the particular fluid he wants to generate. If no correlations are available, at least a constant value (A coefficient) should be entered.

The "*nsat*" variable defines the number of points of the saturation curve. This should be about 50. These data will also define the number of pressures between the triple point and the critical point, so the size of the properties file will be quadratic with nsat. Temperatures are evenly spaced on the saturation line. For each saturation pressure, a gas curve (from Tsat to Tmax) and a liquid (from Ttriple to Tsat) curve will be generated. Supercritical curves will be generated from Ttriple to Tmax if Pmax > Pcrit according to a geometric progression in pressure.

With these methods, liquid properties (obtained with the appropriate solution of the Peng-Robinson equation) can be generated in the same way as gas properties, so all the properties are consistent and evidence good transitions between subcritical to supercritical pressures and temperatures.

It should be noted that the user is responsible for the properties generated with this tool. Indeed, obtaining appropriate fluid properties is a matter of vital importance that requires much effort. In our case, we use this tool to update the real properties of Hydrazine, UDMH and MMH avoiding the problems in ESPSS version 1.0, where there were strong discontinuities near Tcrit for pressures above the critical point (specific correlations for the liquid density and sound speed used in the previous version no longer required using the Peng-Robinson equation). For these two fluids, the calculated liquid densities have errors of less than 2 %. The saturation pressure for Hydrazine is taken from RD-12. Most data is taken from RD-11.

The Peng-Robinson equation adapts poorly to the NTO properties because this fluid is reactive and a variable concentration of nitrogen oxides must be considered in the gas phase. See §4.3.3.1.3.

Global parameters to be defined (white cells):

Fluid name:		T [K]	p [Pa]
Hydrazine	Triple point	274.82	4.37E+02
	Critical point	653.00	1.47E+07
	Max. Cond.	900	5.00E+07

Peng-Robinson state equation

Rg [J/kg/K]	Ac. f. (ω)	u	w
259.5	0.328	2	-1

Cond. [W/m/K] = A + B*T + C*T^2

A	B	C	
7.8890E-02	*-5.2200E-04*	*1.0280E-06*	**GAS**
8.82E-02	2.74E-03	-4.68E-06	**LIQ**

Cp_gas [J/kg/K] = A+B*T+C*T^2+D*T^3

A	B	C	D
-1.6496E+02	*8.8590E+00*	*-1.1000E-02*	*6.0000E-06*

H_ref (J/kg)	S_ref (J/kg/K)
Ho	So
2.2161E+06	*5.1424E+03*

Visc. [Pa s]

A	B	C	D	
-2.1200E-06	*4.0250E-08*	*-3.2000E-11*	*2.9100E-14*	**GAS= A + B*T + C*T^2 + D*T^3**
-25.959	3047.77	3.86E-02	-3.17E-05	**LIQ = exp(A + B/T + C*T + D*T^2)**

Sat. Pres [Bar] = exp(A + B/T + D*T + C*Ln(T))

A	B	C	D
5.8758E+01	*-7.0700E+03*	*-7.0880E+00*	*4.5700E-03*

Points for the sat line

nsat
50

Surf. T [N*m] = A + B*T + C*T^2

A	B	C
1.246E-01	-1.917E-04	0.000E+00

Derived constants (Not to be defined)

m	a	b
8.51E-01	8.94E+02	8.97E-04

Generate ESPSS propFile

Peng-Robinson (u=2; w=-1) equation:

$$P = \frac{R_g T}{v-b} - \frac{a\alpha}{v^2 + ubv + wb^2}$$

where

$$a = \frac{0.45724 R_g^2 T_{cr}^2}{P_{cr}}; \quad b = \frac{0.07780 R_g T_{cr}}{P_{cr}}$$

$$\alpha = \left[1 + m(1 - T_r^{0.5})\right]^2; \quad T_r = T / T_{cr}$$

$$m = 0.37464 + 1.54226\omega - 0.26992\omega^2$$

$$\omega = Ac.factor$$

References

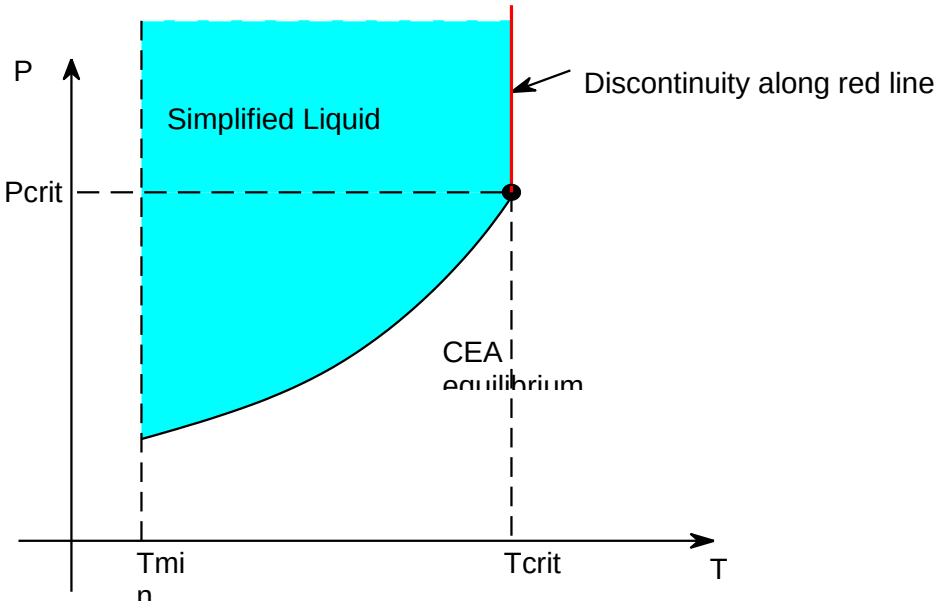
1 C.-A. Scheley & J. Kollien. Algorithms for efficient computation of Propulsion Fluid Properties. AIAA 96-0235, January 15-18, 1996 / Reno, NV

2 R.C. Reid, J.M. Prausnitz & B.E. Poling. The properties of Gases & Liquids. Four Edition. McGraw Hill, Inc.

Figure 4-1 Main sheet of the Gen_Prop EXCEL tool

4.3.3.1.3 Equilibrium state equation properties (N2O4)

For N2O4, the gas properties have been calculated using the CEA equilibrium properties. Liquid properties have been calculated using specific correlations giving the liquid sound speed and density. This approach generates the properties in the region (Pressure > Psat & Temperature < Critical Temperature) using a semi-perfect liquid approach. Of course, the liquid table would be relatively good when temperature is considerably below critical, and it would have very large errors when the critical temperature is approached:



Reference RD-13 provides the information, methods and data used for building the properties. This report documents the fluid properties table generated for EcosimPro for Nitrogen Tetroxide (N2O4) and for the two standard Mixed Oxides of Nitrogen (MON-1 and MON-3). Properties have been calculated from the triple point up to pressure pmax = 30 Mpa, and temperature Tmax = 600 K.

Two main problems were identified during the trade-off between different correlations and data sources. First of all, it is very difficult to find reliable correlations for all properties of liquid N2O4 which are directly linked to experimental results. The sources found are the USAF propellant handbook [RD-14], dated 1977 and an ONERA report [RD-17]. These publications have been taken as references, and their correlations are used to generate the liquid property tables, unless more recent and reliable data are found.

Data are even scarcer for MON-1 and MON-3, as detailed later in this report. The only liquid properties available for these two oxidizers are saturation pressure and liquid density. All other liquid properties have been assumed equal to the N2O4 properties. The general properties such as triple point, critical point and saturation curve, etc. are taken from RD-14 and RD-16.

The second problem is the spontaneous decomposition that takes place as soon as N2O4 is in gaseous state. According to RD-15, Nitrogen Tetroxide is dissociated into NO2, the dissociation being complete at 400 K (at ambient pressure). At higher temperatures, a second dissociation reaction (4 species at equilibrium, N2O4, NO2, NO, O2), much slower than the first, starts to take place, creating NO and O2 and taking into account this phenomenon involves calculating a chemical equilibrium for every pressure and temperature of interest in the vapor and supercritical phase. This task is accomplished using the CEA code. Reaction contribution to Cp and thermal conductivity are taken into account.

4.3.3.2 Real properties file format

In addition to the methods previously described, the user may build its own property's files (see §4.1.3), as long as they respect (as for the other real fluids) the format restraints described in Table 4-9 below:

Table 4-9 – Registers contained in the real property files

Register type	Register description
1	A text line **("Gas and supercritical Properties"** for example**)**
2	One line with the following values (free format): Molecular weight, (gr/mole), Ref. pressure, (Pa), Ref. temperature, (K), Ref. entropy, (J/kg.K), Ref. enthalpy (J/kg.), Minimum value of pressure (Pa), Maximum value of pressure (Pa), Minimum value of temperature (K), Maximum value of temperature (K)
3 to 5	A line (free format) with the properties of Table 4-6 for the triple point (liquid) Another for the triple point (vapor) Another for the critical point**.**
6	One line with the number of pressures (np_g) in gas conditions (*including reheated vapor, and supercritical conditions for all range of temperatures*).
7	One line with the first pressure value and the number of temperatures (nt_{g1}) considered for this pressure.
8 (nt_g lines)	For each temperature, one line with the properties of Table 4-6starting with temperature and in the same order (see note 1). The temperature range must go from the saturation point (or the melting point if P >= Pcrit) to the maximum temperature.
	Repeat registers type 7 and 8 for each pressure table. Pressures must go from one below or equal to the triple point to the critical one, and then till the maximum pressure. The triple and the critical pressures must be included. For pressures lower than the triple one, temperatures must go from the minimum one (melting) to the maximum temperature. For pressures lower than the critical one, temperatures must go from the *exact* saturation temperature to the maximum one. For pressures equal to or higher than the critical pressure, temperatures must go from the minimum (melting) one to the maximum temperature (*see note 2*).
9	A text line **("Liquid properties"** for example**).**
10	One line with the number of pressures (np_l) in liquid conditions.
11	One line with the first pressure value (the triple point), the number of temperatures (nt_{l1}) considered for this pressure and the *surface tension* σ (N/m) for the saturation temperature of the current pressure.
12 (nt_l lines)	For each temperature, one line with same properties as for the gas tables. The temperature range must go from the melting point to the saturation temperature *exactly*.
	Repeat registers type 11 and 12 for each pressure table. Pressures must go from the triple point to the critical point, and must have the same values as for the gas tables. The triple pressures and the critical pressures must be included, the latter having the same values as on the gas side and only up to the critical temperature. The technique used in generating a liquid table is as follows: for the triple pressure, only one point is considered; two temperatures (melting and boiling point) for the second pressure; one more point (that of the new saturation temperature) for the next pressure, etc.

Notes:

- 1: The properties stored (columns) are those given in Table 4-6 beginning with the temperature and in the same order. The pressure is only stored at the beginning of each temperature table.
- 2: The temperatures considered for a given pressure can be different from the ones considered for another pressure. The interpolation procedures are not restricted to rectangular tables.
- 3: *The curves* $\rho(P,u)_{P=cte}$ *(for liquid and gas sides) must not cross each other*. See §5.3.3.2 and 5.3.8.4.

The critical point must be included exactly in the tables. This point is in the corner of the three property tables (liquid, vapor and supercritical zones). Special treatments are used for interpolating near this point: Indeed, some derivative properties have an infinite value (Cp, transport properties …) in the critical point. To work around this problem, the *derivatives* of the critical point are not directly used when reading the real properties, but they will be interpolated with a second order adjustment using the two points previous to Tcrit and the first one after Tcrit.

For a better understanding of the real fluid properties format, an example with comments is shown in Table 4-10.

4.3.3.3 EXCEL tool plotting fluid properties

An EXCEL tool (*plot_prop.xls* file) with macros activated by buttons is provided with the property files. This tool allows one-touch plotting of the main properties (H-S, T-S Rho-T diagrams or any particular variable you choose) of a real fluid respecting the format of Table 4-9. Some plots are given below as examples.

4.3.4 Adding new fluids to the ESPSS libraries

New fluids and their property files can be stored under the FLUID_PROPERTIES/PROP_TABLES folder, *so they will be ready to be used in the ESPSS libraries*. There are two ways to add new fluids:

- A) Using pre-defined (*PfGasUsr1.dat …, PfLiqUsr1.dat … or RealUsr1.dat*) fluids; no code modifications are necessary, see §**4.1.3**, but these fluids can not be used as propellants.
- B) Using specific fluids with specific names linked to a particular CEA chemical; the following simple EL manipulations must be made in the FLUID_PROPERTIES library, "FluidProps_common.el" file:

  1. Add the specific fluid and file names for the fluid to be added (indicated here in bold) at the end of the following lists:

     ```
     --Logiccal names of the fluids:
     ENUM FluidKeys = {
     …
             Comb_Gas,  myFluidName }

     --Names of the corresponding properties data files:
     STRING FileName[FluidKeys] = {
     …
          "RealUsr1","RealUsr2","RealUsr3","RealUsr4","RealUsr5","CombGas"," myFluidName"}
     ```

     The file **myFluidName.dat** (according to the formats explained in previous paragraphs) must be present in the FLUID_PROPERTIES/PROP_TABLES folder.

  2. Then, add the integer which will correspond to the chemical position of the new fluid (see ENUM Chemicals). For example if the new fluid is really H2SO4, this chemical being in the 7th position of the Chemical list, then add a 7 at the end of the following list:

     ```
     CONST INTEGER ChemCode[FluidKeys] = \
     …
               0, 0, 0, 0,   0, 0, 0, 0,   0, 0, 0, 0, 0,  -1,  7}
     ```

  3. The next step is to add a value (zero if not known) to the properties dimensioned to FluidKeys (eps_11[FluidKeys], sig_11[FluidKeys]):

     ```
     -- Lennard-Jones parameters
     CONST REAL eps_11[FluidKeys] = \
     …
               0,   0,    0,    0,   0,     0,0,0,0,0,0,0,0,0 , 0}

     CONST REAL sig_11[FluidKeys] = \
     …
                0,   0,    0,    0,   0,    0,0,0,0,0,0,0,0,0 , 0}
     ```

4. Add the new fluid name to the end of the different sub-sets of fluids:

   SET_OF (FluidKeys) *FluidName* = {
   …
   Real_Usr1 , Real_Usr2 , Real_Usr3 , Real_Usr4 , Real_Usr5, **myFluidName** }

5. If the new fluid is according to a real property file, then add the new name to the list of the real fluids:

   SET_OF (FluidKeys) *Real_Fluids* = \
   …
   Real_Usr1 , Real_Usr2 , Real_Usr3 , Real_Usr4 , Real_Usr5 , **myFluidName**}

6. If the new fluid is according to a simplified liquid property file, then add the new name to the list of the PerfectLiq_Fluids:

   SET_OF (FluidKeys) *PerfectLiq_Fluids*= \
   …
   PfLiq_Usr1, PfLiq_Usr2, PfLiq_Usr3, PfLiq_Usr4, PfLiq_Usr5, **myFluidName**}

7. If the new fluid is according to a simplified gas property file, then add the new name to the list of the PerfectGas_Fluids:

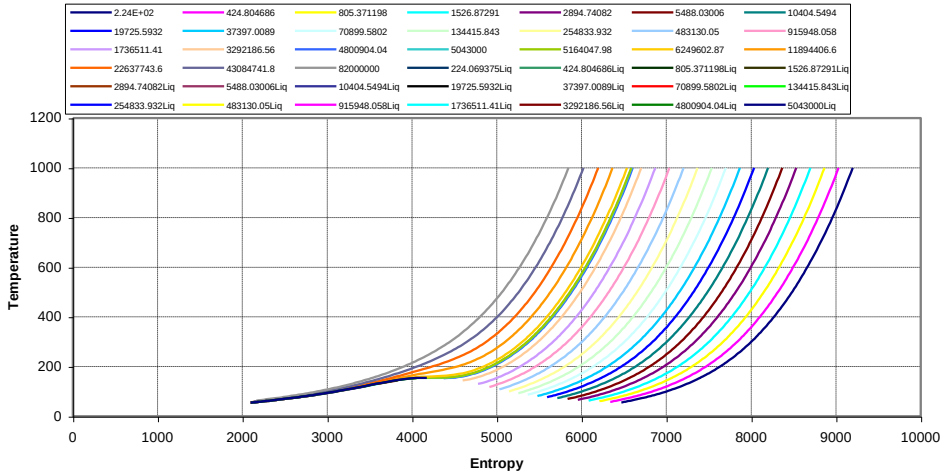
   SET_OF (FluidKeys) *PerfectGas_Fluids*= \
   …
   PfGas_Usr1 , PfGas_Usr2 , PfGas_Usr3 , PfGas_Usr4 , PfGas_Usr5, **myFluidName** }

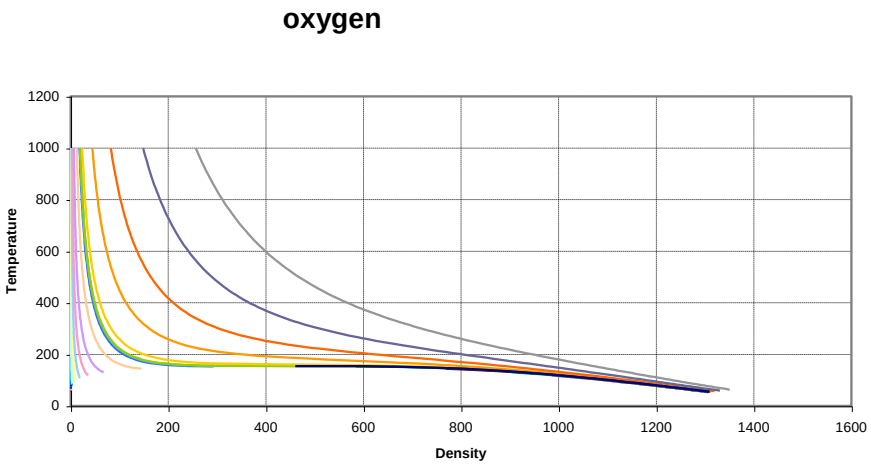
After completing these steps, the ESPSS libraries must be compiled following the installation procedure (see RD-6), and the new fluid, **myFluidName**, will appear in the list of fluids to be selected in the working fluid component, see §5.4.24.

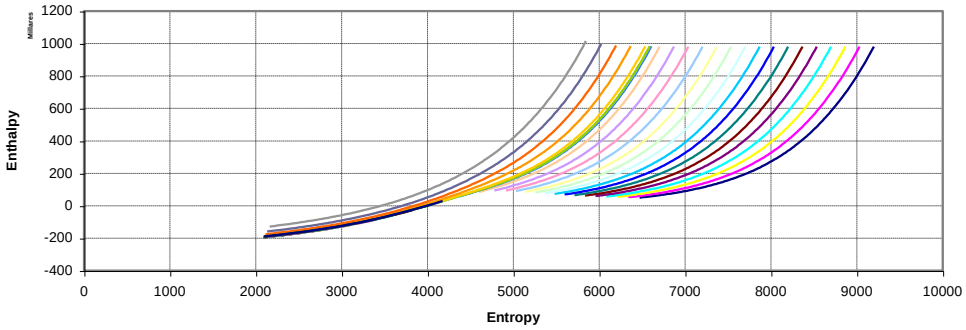
Table 4-10 – Example of a real property file

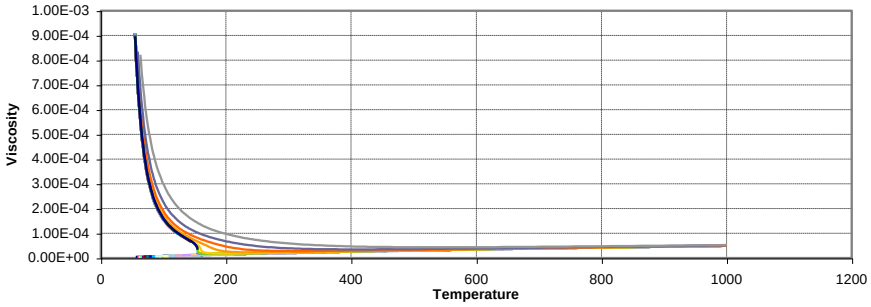
Registers	Values (*Blue colors are comments, not present in the file*)
First line	Vapor properties (Text comments)
Second line	Mol. Weight, Ref. pres., Ref. temp, Ref. entropy Ref. enth., Min. pres., Max. pres., Min. temp., Max. temp. 16.0428000 101325.000 298.149994 6674.46060 909952.093 11696.0641 0.100000E+10 90.6941000 625.000000
3 lines with the properties of Table 4-6 for the triple point (liquid and gas cond.) and for the critical pressure	Pressure Temp. Density Spec. Vol Enth Entropy Energy Cp Cv gamma beta kappa v_sound Cond Viscosity 11696.06 90.69 451.475 0.221E-02 -71820. -709.945 …... (TP liquid) 11696.06 90.69 0.25074 3.9881310 472442.2 5291.13 …... (TP 64ás) 4599200. 190.56 161.384 0.6196^{E}-02 416933.9 2569.49 …… (Crit Point)
Number of pressures in **gas side**	81
First pressure properties table	11696.0 130 (Pressure value and number of temperature points) Temp Density Spec. Vol Enth Entropy Energy Cp Cv gamma beta kappa v_sound Cond Viscosity 90.69 0.25074 3.9881310 472442 5291.13 425796.8 2110.03 1573.50 1.33055 …… (First temperature set of values. If P < Pcrit, T value must be the Tsat) 91.90 0.24736 4.0425352 474988 5319.02 427706.3 2107.69 1572.16 1.33066 …… (Second temperature set of values.) … 625. 0.361E-01 27.692 1814801. 9790.22 1490907. 3362.88 2844 …… (Last temperature set of values)
…	Same as before for each one of the pressure tables in the gas side. The critical pressure must be included exactly from the minimum to the maximum temperature Super-critical pressure tables must go from the minimum to the maximum temperature
Number of pressures in **liquid side**	43
First pressure liquid properties table. *This table corresponds with the triple point: Only one point*	11696.0 1 0.187607086E-01 (Pressure value, number of temperature points and surface tension value) Temp. Density Spec. Vol Enth Entropy Energy Cp Cv gamma beta kappa v_sound Cond Viscosity 90.69 451.475 0.22149E-02 -71820.5 -709.945 -71846.4 …(First temperature set of values. This temp. should be the triple point)
Second pressure properties table. *This table should have two points*	11696.0 2 0.187607086E-01 (Pressure value, number of temperature points and surface tension value) Temp. Density Spec. Vol Enth Entropy Energy Cp Cv gamma beta kappa v_sound Cond Viscosity 90.69 451.475 0.2214E-02 -71820.5 -709.945 -71846.4 …(First temperature set of values. This temperature should be the melting point at the current pressure) … 91.90 449.866 0.2222E-02 -67749.3 -665.399 -67779.6 …(Last temperature set of values. This temperature must be the saturation one at the current pressure)
---	Same as before for each one of the pressure tables in the liquid side. The last table must be the critical pressure, having same values as in the gas side and only till the critical temperature Liquid pressures must go from the triple point to the critical point, and must have the same values as in the gas tables.

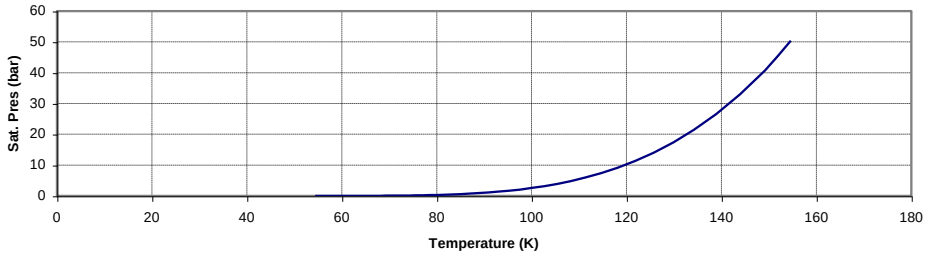
Note: See Table 4-6 with the Units (SI) in which the properties must be stored

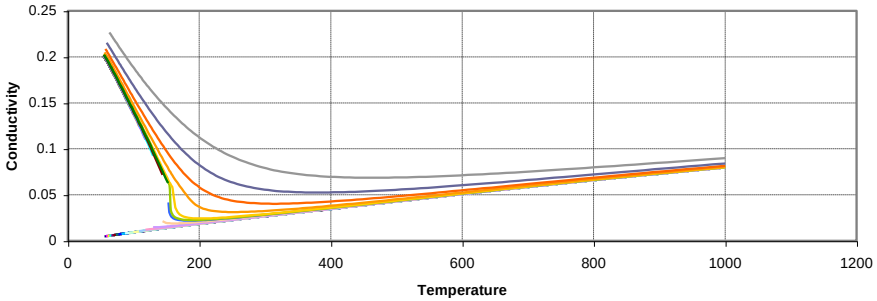

2.24E+02
424.804686
805.371198
1526.87291
2894.74082
5488.03006
10404.5494
19725.5932
37397.0089
70899.5802
134415.843
254833.932
483130.05
915948.058
1736511.41
3292186.56
4800904.04
5043000
5164047.98
6249602.87
11894406.6
22637743.6
43084741.8
82000000
224.069375Liq
424.804686Liq
805.371198Liq
1526.87291Liq
2894.74082Liq
5488.03006Liq
10404.5494Liq
19725.5932Liq
37397.0089Liq
70899.5802Liq
134415.843Liq
254833.932Liq
483130.05Liq
915948.058Liq
1736511.41Liq
3292186.56Liq
4800904.04Liq
5043000Liq
Temperature
Entropy

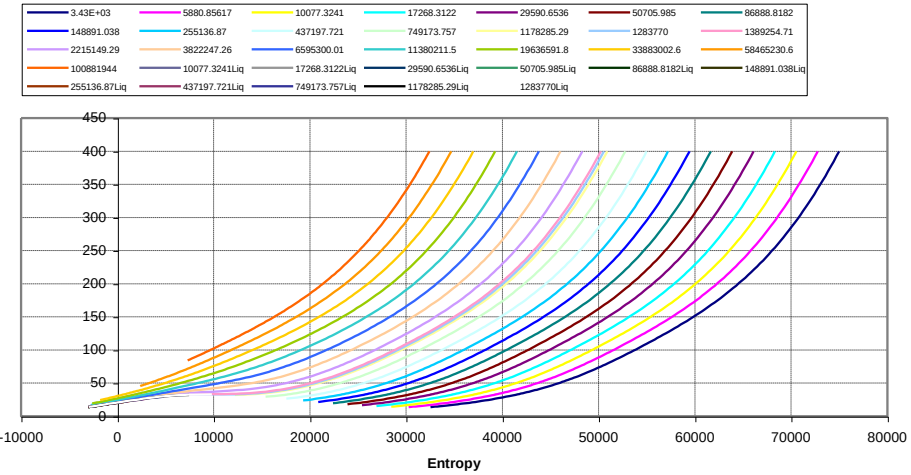

oxygen
Temperature
Density


Millares
Enthalpy
Entropy

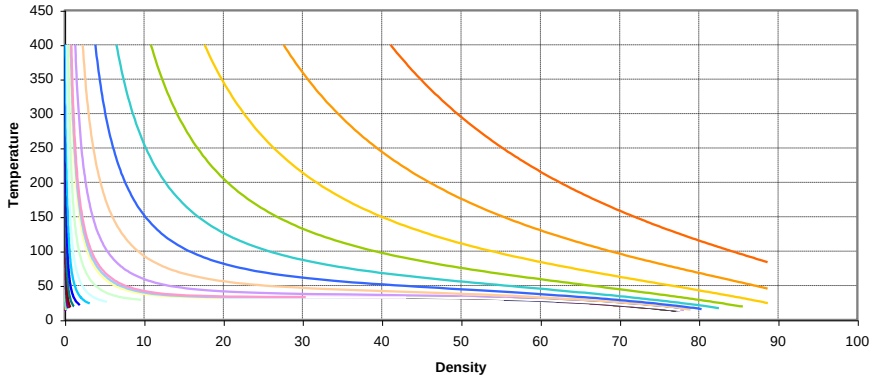

Viscosity
Temperature

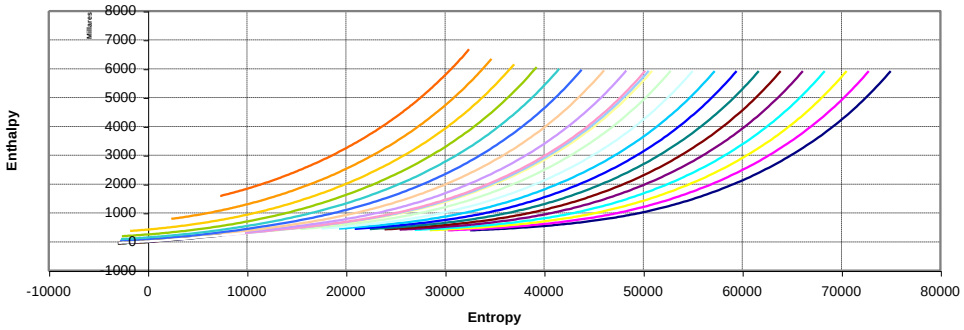

Sat. Pres (bar)
Temperature (K)

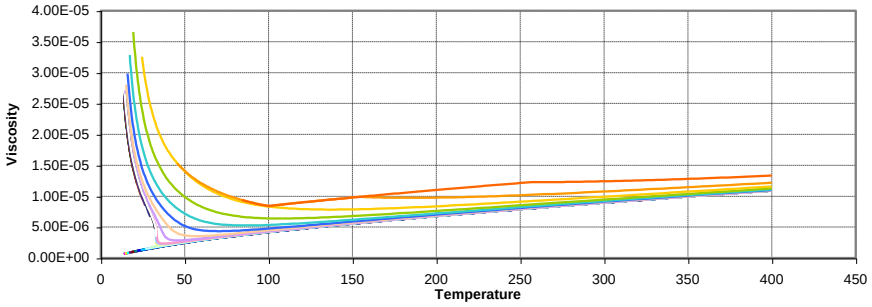

Conductivity
Temperature

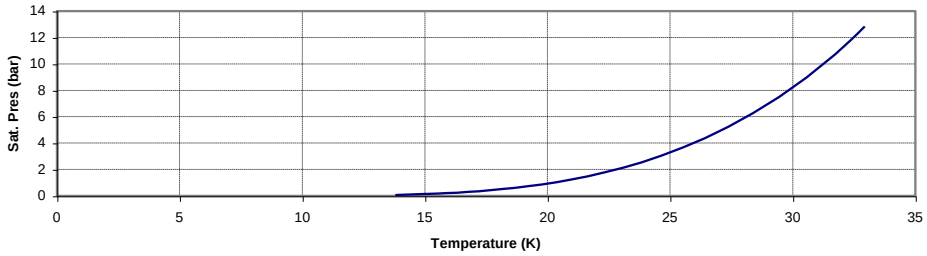

3.43E+03
5880.85617
10077.3241
17268.3122
29590.6536
50705.985
86888.8182
148891.038
255136.87
437197.721
749173.757
1178285.29
1283770
1389254.71
2215149.29
3822247.26
6595300.01
11380211.5
19636591.8
33883002.6
58465230.6
100881944
10077.3241Liq
17268.3122Liq
29590.6536Liq
50705.985Liq
86888.8182Liq
148891.038Liq
255136.87Liq
437197.721Liq
749173.757Liq
1178285.29Liq
1283770Liq
Temperature
Entropy

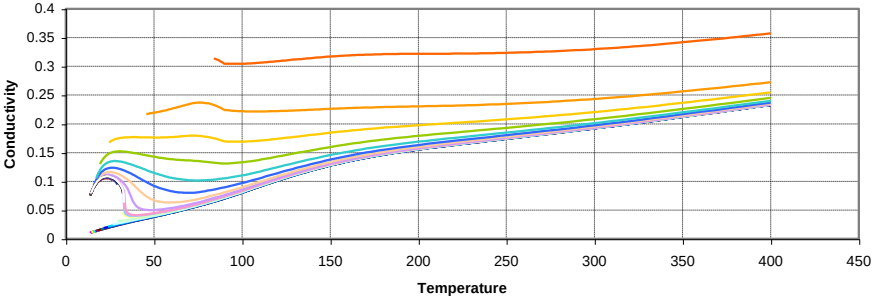
parahyd


Temperature
Density


Millares
Enthalpy
Entropy


Viscosity
Temperature


Sat. Pres (bar)
Temperature (K)


Conductivity
Temperature

4.4 PROPERTIES FORMULATION FOR PURE FLUIDS

4.4.1 Perfect Gas properties according to CEA

4.4.1.1 Equation of State

The formulation that calculates the thermodynamic state as a function of Rho-T is presented below. The programming is based on the perfect gas state equation. The expressions used are summarized as follows:

$$P = \rho \cdot \frac{R \cdot T}{MW}$$

$$Z = 1; \quad \beta = 1/T; \qquad \kappa = 1/P$$

The expressions used for calculating the energy are based on the computation of the specific heat at constant pressure for ideal gases (Cpº) as a function of temperature only (by means of polynomial expressions). The expression proposed was obtained from a very large database providing data for a very wide temperature range (between 200 and 20000 K) and can be summarized as follows:

$$\frac{Cp}{R} = a_1 \cdot T^{-2} + a_2 \cdot T^{-1} + a_3 + a_4 \cdot T + a_5 \cdot T^2 + a_6 \cdot T^3 + a_7 \cdot T^4$$

$$H = \mathrm{H}(T_0) + \int_{T_0}^{T} C_p(T)dT =$$

$$R \cdot T \cdot \left(-a_1 \cdot T^{-2} + a_2 \cdot T^{-1} \ln T + a_3 + a_4 \cdot \frac{T}{2} + a_5 \cdot \frac{T^2}{3} + a_6 \cdot \frac{T^3}{4} + a_7 \cdot \frac{T^4}{5} + \frac{b_1}{T} \right)$$

$$S = S(T_0, P_0) + \int_{T_0}^{T} \frac{C_p(T)}{T} dT - \frac{R}{MW} \log(P/P_0) = -\frac{R}{MW} \log(P/P_0$$

$$R \cdot \left(-a_1 \cdot T^{-2} - a_2 \cdot T^{-1} + a_3 \cdot \ln T + a_4 \cdot T + a_5 \cdot \frac{T^2}{2} + a_6 \cdot \frac{T^3}{3} + a_7 \cdot \frac{T^4}{4} + \frac{b_2}{T} \right)$$

4.4.1.2 Transport properties

The functions giving the viscosity and the thermal conductivity if data is available (mainly from NIST species database, in this case the NIST database is not as extensive as for the thermodynamic properties) are also based on polynomial expressions.

The viscosity and thermal conductivity functions take the following form, respectively:

$$ln\eta = A_1 \cdot lnT + \frac{B_1}{T} + \frac{C_1}{T^2} + D_1; \qquad ln\lambda = A_2 \cdot lnT + \frac{B_2}{T} + \frac{C_2}{T^2} + D_2$$

Otherwise, these properties are estimated as follows:

4.4.1.2.1 Viscosity

There are different estimation methods available. The approach used is subject to the availability of property data: critical properties and the dipole moment. In this case the expression used [RD-10] is:

$$\eta = \frac{\left[0.807 \cdot T_r^{0.618} - 0.357 \cdot exp(-0.449 \cdot T_r) + 0.340 \cdot exp(-4.058 \cdot T_r) + 0.018\right] \cdot F^o{}_P \cdot F^o{}_Q}{\zeta \cdot 10^7}$$

where T_r is the reduced temperature computed as follows $T_r = T/T_c$, F^oP and F^oQ are correction factors and ζ is the reduced inverse viscosity.

The reduced inverse viscosity, ζ, is calculated as follows:

$$\varsigma = 0.176 \cdot \left(\frac{T_c}{MW^3 \cdot P_c^4} \right)^{1/6}$$

F^o_P is a correction factor that mainly depends on the polarity of the molecule. It is computed as follows:

$$F^o_P = 1 \qquad 0 \le \mu_r \le 0.022$$
$$F^o_P = 1 + 30.55 \cdot (0.292 - Z_c)^{1.72} \qquad 0.022 \le \mu_r \le 0.075$$
$$F^o_P = 1 + 30.55 \cdot (0.292 - Z_c)^{1.72} \cdot |0.96 + 0.1 \cdot (T_r - 0.7)| \qquad 0.075 \le \mu_r$$

F^o_Q is a correction factor used only in quantum gases. In the case before us, its value is 1. μ_r is the relative dipole moment computed as follows:

$$\mu_r = 52.46 \cdot \frac{\mu^2 \cdot P_c}{T_c^2}$$ where μ is the dipole moment in debyes

If critical properties and dipole moment are not available, other estimations must be done based on quantum formulation. In these cases, the following expression must be used:

$$\eta_k = \frac{\eta_{ns} \cdot \sqrt{MW_k \cdot T}}{\Omega_k \cdot 10^7}$$

where η_{ns} is a constant (26.6958 in S.I.) and Ω_k

$$\Omega_k = ln\left(\frac{50 \cdot MW_k^{4.6}}{T^{1.4}} \right)$$

4.4.1.2.2 Thermal Conductivity

Similarly to viscosity, the thermal conductivities that are not available are estimated. The approach used is summarized as follows:

$$\lambda_k = \frac{\eta_k \cdot R \cdot \left(3.75 + 1.32 \cdot \left(C_{p,k} / R - 2.5\right)\right)}{MW_k}$$

4.4.2 Van der Waals functions

4.4.2.1 Thermodynamic Properties

The formulation calculating the thermodynamic state as a function of Rho-T is present below. The equation of state of Van der Waals is:

$$P = \frac{R \cdot T}{V - b} - \frac{a}{V^2}$$

where a and b are characteristic parameters computed as follows:

$$a = \frac{27}{64} \cdot \frac{R^2 T_c^2}{P_c} \qquad b = \frac{R \cdot T_c}{8 \cdot P_c}$$

For the computation of the volumetric expansion (β) and the isothermal compressibility (κ) coefficients, the following formulation has been used:

$$\beta = \frac{1}{V}\left(\frac{\partial V}{\partial T}\right)_{P=ct} = \frac{R \cdot V^2}{PV^3 + 2 \cdot a \cdot b - a \cdot V}$$

$$\kappa = -\frac{1}{V}\left(\frac{\partial V}{\partial P}\right)_{T=ct} = \frac{-\left[V \cdot (V-b)\right]^2}{2 \cdot a \cdot (V-b)^2 - R \cdot T \cdot V^3}$$

The specific heat has been computed as follows:

$$Cv = Cp_{ideal}(T) - R \cdot \frac{V+b}{V-b} \qquad Cp = Cv + \frac{\beta^2 \cdot V \cdot T}{\kappa}$$

For the internal energy, enthalpy and entropy the expressions utilized are summarized as follows:

$$H = H_{ideal}(T_0) + \int_{T_0}^{T} Cp_{ideal}(T)dT + R \cdot T \cdot \left[\frac{2 \cdot a}{R \cdot T \cdot V} - \frac{b}{V-b}\right]$$

$$U = H_{ideal}(T_0) + \int_{T_0}^{T} Cp_{ideal}(T)dT + \frac{3 \cdot a}{V} - R \cdot T \cdot \frac{V+b}{V-b}$$

$$S = S_{ideal}(T_0, P_0) + \int_{T_0}^{T} \frac{Cp_{ideal}(T)}{T}dT - \frac{R}{MW} log(P/P_0) - R \cdot ln\left[Z \cdot \left(1 - \frac{b}{V}\right)\right]$$

$$sound\ speed = \sqrt{\frac{c_p}{c_v \cdot \rho \cdot \kappa}}$$

The thermo derivatives $\partial\rho/\partial h_{P=cte}$, $\partial\rho/\partial P_{h=cte}$ are computed based on the values of β and κ. The expressions were already shown in §4.4.1.

4.4.2.2 Transport Properties

Transport Properties for Van der Waals gases are computed as they were Perfect Gases; therefore, they are calculated as explained in §4.4.1.2.

4.4.3 Interpolated Perfect Gas properties

4.4.3.1 Equation of State

For the state equation, the same expressions as in§4.4.1.1 are used. The expressions used for calculating the energy are based on the table interpolation of the specific heat at constant pressure (Cp^o) as a function of temperature only:

$$\frac{C_p}{R} = Table\_Interp(T)$$

$$H = \mathrm{H}(T_0) + \int_{T_0}^{T} C_p(T)dT$$

$$S = S(T_0, P_0) + \int_{T_0}^{T} \frac{C_p(T)}{T}dT - \frac{R}{MW}\log(P/P_0)$$

4.4.3.2 Transport Properties

The functions giving the viscosity and the thermal conductivity are interpolated from the external user-defined property file as a function of temperature:

$$viscosity = f_v(T)$$
$$conductivity = f_c(T)$$

4.4.4 Simplified Liquid interpolated properties

For simplified liquids, the formulation proposed for defining the thermodynamic state is presented below.

4.4.4.1 Equation of State

Based on the tables read from the property file, density, sound speed and specific heat are interpolated as functions of the temperature. Thus, the volumetric expansively can be obtained as follows:

$$\beta = \frac{-1}{\rho}\frac{d\rho}{dT}\bigg|_{P=cte}$$

This derivative is calculated numerically. Then, the following other thermodynamic derivatives can be calculated:

$$Cv = \frac{Cp}{1 + T \cdot \beta^2 \cdot v_{sound}^2 / Cp}$$
$$\kappa = \frac{1}{\rho}\frac{d\rho}{dP}\bigg|_{T=cte} = \frac{Cp}{\rho \cdot v_{sound}^2 \cdot Cv}$$

Once these properties have been interpolated, the equation of state can be applied assuming constant compressibility with pressure:

$$\rho(P,T) = \rho(T)\left[1 + \kappa(P - P_{ref})\right]$$

The enthalpy is calculated by numerically integrating the Cp:

$$H = \mathrm{H}_{\mathrm{ideal}}(T_0) + \int_{T_0}^{T} Cp_{ideal}(T)dT$$

The entropy is calculated by numerically integrating Cp and β:

$$s = \int_{T_0}^{T} \frac{Cp_{ideal}(T)}{T} dT - \int_{P_0}^{P} \beta \frac{dP}{\rho}$$

4.4.4.2 Transport properties

Viscosity and thermal conductivity are interpolated from the external data of the property file as a function of temperature:

$$viscosity = v(T)$$
$$conductivity = \lambda(T)$$

4.4.5 Real Fluid interpolated properties

For real fluids, the mentioned FORTRAN functions (see 4.2.2) will perform special searching techniques in 2D property tables to interpolate the desired property in the nearest cell of the data tables. A single reading will be done the first time that a function's call of any property is made.

The functions will identify whether a certain fluid is liquid, vapor or two-phase flow by giving a pair of variables that can be *rho-T, rho-U, s-H, P-H, P-T*, etc.

4.4.5.1 Equation of State

With the exception of two-phase flow or in the case of table extrapolation (see below), no special hypothesis has been made concerning the Equation of State and the properties: all the properties are interpolated using the data tables of the properties file.

4.4.5.2 Interpolation procedures

The functions consider two forms of interpolation:

- Direct interpolation using the couples P-S, P-H, etc., and returning a property.
- Reverse interpolation using the couples H-S or Rho-U and returning pressure.

The figure below represents an H-S diagram and shows the thermodynamic zones automatically treated by the functions. The extrapolation error number is indicated.

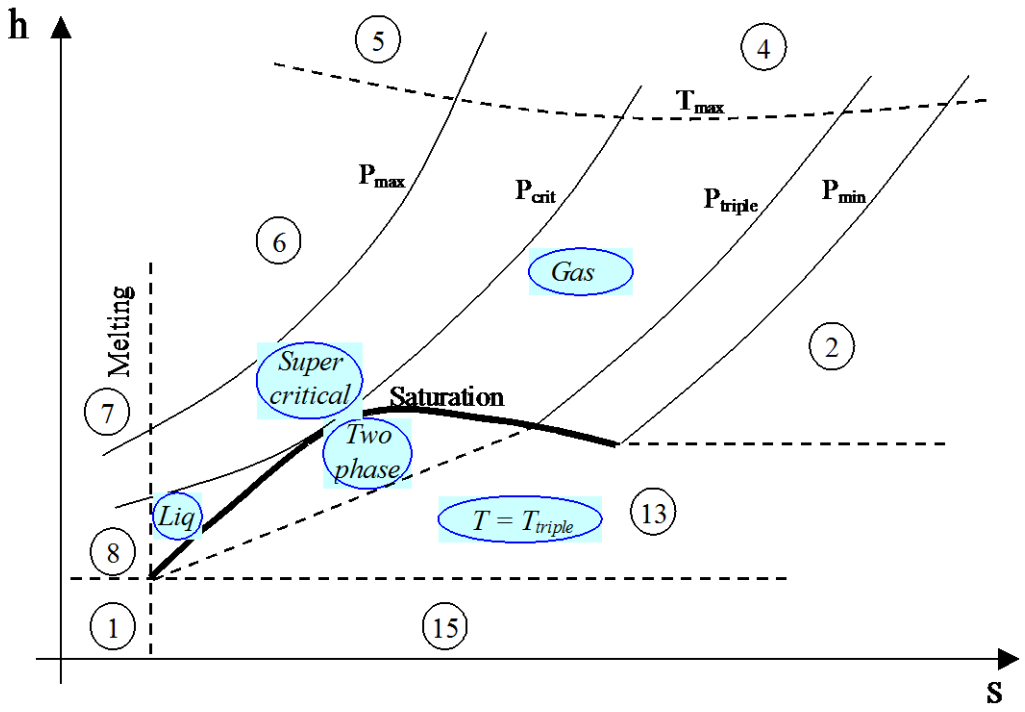


Figure 4-2 Typical H-S diagram

Direct interpolation, y=f(P,x), where "*x*" can be h (enthalpy), u (energy), T (temperature), ρ (density) or s (entropy). Using P-T, no two-phase conditions can be calculated. The logic taken by the thermodynamic function determining the fluid zone is:

If P > Pcrit ➔ 2D interpolation in the *super critical* data tables using the "y=*Int_3r(P,x)*" function. Return

Otherwise, a vapor two-phase limit (x_g) at the current pressure shall be interpolated. The x_g value is the *x* saturated vapor property:

1. If x > x_g ➔2D interpolation in the *super-heated vapor* data tables using the "y=*Int_3r(P,x)*" function. Return
2. Otherwise, a liquid two-phase limit (x_l) at the current pressure shall be interpolated. The x_l value is the *x* saturated liquid property:

- If x < x_l ➔2D interpolation in the *liquid* data tables using the "y=*Int_3r(P,x)"* function. Return
- Otherwise, the quality of the *two-phase* conditions shall be determined according to the formulation given in §4.4.5.3 and 4.4.5.4

Reverse interpolation, P=f(x,y), where "*x"* can be ρ or s and "*y"* h, u or T. The logic taken by the thermodynamic functions to determine the fluid zone is:

A critical limit (y_crit) at the current "x" value shall be interpolated in the critical pressure curve:

1. If y > y_crit ➔2D interpolation in the *super critical* data tables using the "P=*Int_3p(x,y)"* function. Return
2. Otherwise, a two-phase limit (y_g or y_l depending on the x position regarding the critical point) shall be interpolated using the current "x" value:
   - If x > y_g AND x > x_crit ➔2D interpolation in the *super-heated vapor* data tables using the "P=*Int_3p(x,y)"* function. Return
   - If x > y_l AND x < x_crit ➔2D interpolation in the *liquid* data tables using the "P=*Int_3p(x, y)"* function. Return
   - Otherwise, the quality of the *two-phase* conditions shall be determined according to the formulation given in §4.4.5.3 and 4.4.5.4

Note 1: The relations "x > x_g" are reversed if x== density.

Note 2: At P<Ptriple and two-phase flow, the vapor phase is extrapolated at T=Ttriple assuming ideal gas or using the real gas properties at the current pressure. Liquid phase is assumed to be frozen at Ttriple.

4.4.5.3 Quality and Void fraction calculations

Under two phase conditions the quality of the mixture is calculated from the saturation properties of the liquid and steam phases.

Knowing the mixture (vapor/liquid) density and energy, the following two equations are used:

$$quality = x = \frac{u - u_{liq}}{u_{vap} - u_{liq}}$$

$$1/\rho = 1/\rho_{liq} + x(1/\rho_{vap} - 1/\rho_{liq})$$

An iterative process in pressure is needed. For each iteration, the saturation conditions will be calculated and the pressure fulfilling the two previous equations can be found.

The void fraction is calculated as follows:

$$\alpha = void\ \ fraction = V_{vap}/(V_{vap} + V_{liq}) = (\rho_{liq} - \rho)/(\rho_{liq} - \rho_{vap})$$

The transport properties and the heat capacity in two-phase conditions are calculated simply:

$$\mu = x\mu_{vap} + (1-x)\mu_{liq}$$

$$\lambda = x\lambda_{vap} + (1-x)\lambda_{liq}$$

$$c_p = xc_{p,vap} + (1-x)c_{p,liq}$$

Nevertheless, the liquid and vapor saturation values of the transport properties, together with those of the heat capacities, densities and enthalpies (latent heat), will be returned for the calculation of other two-phase properties such as the sound speed and the film coefficient.

4.4.5.4 Sound speed calculation

Under one-phase conditions the sound speed is directly given by the FORTRAN function interpolations. The returned value is equivalent to the following expression that uses other properties (Cp, β, κ) also returned by the properties function:

$$v_{sound} = \sqrt{\frac{c_p}{\rho\kappa c_p - \beta^2 T}}$$

Under two-phase conditions the sound speed must be calculated. The equilibrium sound speed presents discontinuities at phase changes. In order maintain robustness of the system the following approach is given by Wallis (1969):

$$1/v^2{}_{sound} = (\alpha\rho_{vap} + (1-\alpha)\rho_{liq})(\alpha/\rho_{vap}/v^2{}_{sound,vap} + (1-\alpha)/\rho_{liq}/v^2{}_{sound,liq})$$

Another "frozen" [RD-18, RD-19] sound speed expression can be used instead, which is also continuous with the sound speed at phase changes:

$$dP/dT = \frac{xc_{p,v} + (1-x)c_{p,l}}{T(x\beta_v v_v + (1-x)\beta_l v_l)}; \qquad v: specific\ volume; \qquad x: quality$$

$$v_{sound}^2 = \frac{T\left(dP/dT\, v\right)^2}{x\left(\varepsilon \cdot c_{p,v} - T\, dP/dT\, v_v((1+\varepsilon)\beta_v - \kappa_v\, dP/dT)\right) + (1-x)\left(\varepsilon \cdot c_{p,l} - T\, dP/dT\, v_l((1+\varepsilon)\beta_l - \kappa_l\, dP/dT)\right)}$$

where:

$$\varepsilon = 0 \quad \Rightarrow \text{"frozen" sound speed}$$
$$\varepsilon = 1 \quad \Rightarrow \text{"equilibrium" sound speed}.$$

Sub-indexes "v" and "l" indicate vapor and liquid saturated conditions.

4.4.5.5 Extrapolation techniques

There may be situations where the required property is out of the tables. In these circumstances, the interpolation function will perform the following extrapolation techniques:

- If T>Tmax or P>Pmax ➔ the perfect gas equations are used based on the compressibility (Z) and heat capacity (Cp) of the nearest table point to the required one.

  The transport properties are extrapolated as follows

  $$\mu = \mu_{nearest}(T/T_{nearest})^{0.66}; \qquad \lambda = \lambda_{nearest}(T/T_{nearest})^{0.66}$$

  Concerning the Cp, the specific energy and the entropy function S, the following **blending** procedures with CEA properties are performed if T>Tmax:

  A/ Transition zone between the Cp_NIST at Tmax and Cp_CEA at Tmax+100K

  $$Cp_{blended} = Cp_{NIST,\max} + (Cp_{CEA}(T) - Cp_{NIST,\max})\tanh\left(\frac{T - T_{\max}}{20}\right)$$

  $$H = \mathrm{H}_{\mathrm{NIST,\,max}} + \int_{T_{\max}}^{T} Cp_{blended}(T)dT; \qquad S = S_{\mathrm{NIST,\,max}} + \int_{T_{\max}}^{T} Cp_{blended}(T)dT/T$$

  where the Cp_CEA is calculated according to paragraph 4.4.1

B/ Extrapolation zone using Cp_CEA at T > Tmax+100K

$$Cp = Cp_{CEA}(T);$$

$$H = H_{CEA}(T) - H_{CEA}(T_{\max+100}) + H_{NIST,\max}(T_{\max}) + \int_{T_{\max}}^{T_{\max+100}} Cp_{blended}(T)dT$$

$$S = S_{CEA}(T) - S_{CEA}(T_{\max+100}) + S_{NIST,\max}(T_{\max}) + \int_{T_{\max}}^{T_{\max+100}} Cp_{blended}(T)dT/T$$

It should be noted that this blending procedure needs to iterate if the thermodynamic input is other than the temperature. For each iteration, a numerical integration must be done calculating the dH (or dS) in the transition zone

- If P<Pmin AND phase = vapor ➔ Same as before, perfect gas equations.
- If P<Ptriple AND phase = two-phase ➔ same calculations as in §4.4.5.3 but using the extrapolated saturation properties for pressures lesser than the triple one: Vapor phase is extrapolated at T=Ttriple assuming ideal gas or using the real gas properties at the current pressure. Liquid phase is assumed fixed (frozen) at Ttriple.
- If T<Ttriple AND phase = liquid ➔ It is assumed that the fluid remains liquid. Transport properties and thermodynamic derivatives are that of the last table point. Energy, enthalpy and entropy will be extrapolated linearly.
- If T<Ttriple and P<Pmin AND phase = liquid ➔ No action is taken.

4.4.6 CEA enthalpy reference

Independently of the enthalpy reference with which the property files were stored, the enthalpy and energy properties of *combustor* components will be shifted to always have the CEA reference at the EcosimPro libraries level.

The CEA enthalpy reference, ($H_{CEA}(To)$, see §4.4.1), i.e. the *formation* enthalpy, will be used inside combustors for any kind of fluid (real or perfect). The following shift will be applied to the enthalpy and energy values read in the property tables:

$$H_{shift} = H_{CEA}(To) - H_{table}(To, Po)$$

Where Po=0 and To = 298.15 K. Because the CEA properties do not consider pressure influence, a pressure reference of zero was considered calculating the real enthalpy shift. Indeed, for real fluids, only at P=0 the pressure influence can be ignored.

4.5 MIXTURE OF FLUIDS

4.5.1 Perfect gas mixture calculation

4.5.1.1 Thermodynamic properties

Perfect gas mixtures are calculated with linear mixing rules assuming the same temperature for all the constituents.

$$x_k = \frac{\rho_k}{\sum_{k=1}^{nchem}\rho_k}; \quad P_k = \rho_k \frac{R \cdot T}{MW_k}; \qquad P = \sum_{k=1}^{nchem} P_k$$

Where:

R is the gas constant = 8314.4 J/kmol K

MW_k is the molecular weight of the chemical constituent k
T is the mixture temperature K
ρ_k is the density of the chemical constituent k
x_k is the mass fraction of the chemical constituent k

Previous equations are explicit if the densities ρ and the temperature of the mixture are known. If the internal energy of the mixture "u" is known (state_RU instead of state_RT), a Newton iteration in temperature must be done for this to be true:

$$0 = U - \sum_{k=1}^{nchem} x_k U_k(T)$$

where $U_k(T)$ is a pure fluid function giving the specific energy of the chemical k. The molecular weight of the mixture MW_mix and the molar fractions y_k are calculated as follows:

$$\frac{1}{MW_{mix}} = \sum_{k \in nchem} \frac{x_k}{MW_k}; \qquad y_k = molar\ \ fraction = \frac{x_k \cdot MW_{mix}}{MW_k}$$

The energy properties are computed as follows:

$$c_p = \sum_{i=1}^{nchem} x_k \cdot c_{p,k}(T); \quad c_v = \sum_{i=1}^{nchem} x_k \cdot c_{v,k}(T)$$

$$H = \sum_{i=1}^{nchem} x_k \cdot H_k(T); \quad S = \sum_{i=1}^{nchem} x_k \cdot S_k(T) + \sum_{i=1}^{nchem} y_k \cdot \ln y_k$$

The sound speed is calculated as follows:

$$\gamma = \frac{c_p}{c_p - R/MW_{mix}}$$

$$v_{sound} = \sqrt{\gamma RT / MW_{mix}}$$

4.5.1.2 Transport Properties

Regarding the transport properties, the computation of the mixture viscosity is computed as follows [RD-20]:

$$\eta_{mix} = \sum_{i=1}^{nchem} \frac{y_i \cdot \eta_i}{y_i + \sum_{\substack{j=1 \\ j \neq i}}^{nchem} y_j \cdot \phi_{ij}}$$

where ϕ_{ij} is the interaction parameter computed depending on the availability of data. If the data are available in the database it is computed by means of the following mathematical correlation:

$$\ln \eta_{ij} = A \cdot \ln T + \frac{B}{T} + \frac{C}{T^2} + D$$

If the data are not available, the interaction parameter is estimated with the following formulation:

$$\phi_{ij} = \frac{1}{4} \cdot \left[1 + \left(\frac{\eta_i}{\eta_j} \right)^{0.5} \left(\frac{MW_i}{MW_j} \right)^{0.25} \right]^2 \left(\frac{2 \cdot MW_j}{MW_i + MW_j} \right)^{0.5}$$

Similarly to viscosity, the thermal conductivity for mixtures is computed as follows:

$$\lambda_{mix} = \sum_{i=1}^{nchem} \frac{y_i \cdot \lambda_i}{y_i + \sum_{\substack{j=1 \\ j \neq i}}^{nchem} y_j \cdot \psi_{ij}}$$

If there is data available, a similar expression is used

$$ln\psi_{ij} = A \cdot lnT + \frac{B}{T} + \frac{C}{T^2} + D$$

Otherwise the interaction parameter for the thermal conductivity is based on the one computed for viscosity. The expression is as follows:

$$\psi_{ij} = \phi_{ij} \cdot \left[1 + \frac{2.41 \cdot \left(MW_i - MW_j\right) \cdot \left(MW_i - 0.142 \cdot MW_j\right)}{\left(MW_i + MW_j\right)^2} \right]$$

4.5.2 Van der Waals gas mixture calculation

4.5.2.1 Thermodynamic Properties

Van der Waals gas mixtures are calculated assuming a perfect mixture but individually each constituent is considered a real gas according to Van der Waals equation of state. The mixing equations utilized are summarized as follows:

$$x_k = \frac{\rho_k}{\sum_{k=1}^{nchem} \rho_k}; \qquad P_k = \frac{R \cdot \rho_k \cdot T}{MW_k \cdot \left(1 - b_k \cdot \rho_k\right)} - a_k \cdot \rho_k^2; \qquad P = \sum_{k=1}^{nchem} P_k$$

where,

R is the gas constant = 8314.4 J/kmol K
MWk is the molecular weight of the chemical constituent k
T is the mixture temperature K
ρk is the density of the chemical constituent k
xk, is the mass fraction of the chemical constituent k
ak & bk are Van der Waals characteristic parameters of the chemical constituent k

As previously described these equations are explicit if the densities ρk and the temperature T are known. If the internal energy of the mixture "u" is known instead of the temperature T, a Newton-Raphson's iteration in temperature must be done.

$$0 = U - \sum_{k=1}^{nchem} x_k U_k(T)$$

where Uk(T) is a pure fluid function giving the internal energy of the chemical k. These values are computed as explained in 4.4.2. The molecular weight of the mixture MWmix and the molar fractions yk are calculated as for Perfect Gases. It is summarized as follows:

$$1/MW_{mix} = \sum_{k=1}^{nchem} \frac{x_k}{MW_k}$$

$$y_k = molar\ \ fraction = \frac{x_k \cdot MW_{mix}}{MW_k}$$

The energy equations are computed as follows:

$$c_p = \sum_{i=1}^{nchem} x_k \cdot c_{p,k}(T); \quad c_v = \sum_{i=1}^{nchem} x_k \cdot c_{v,k}(T)$$

$$H = \sum_{i=1}^{nchem} x_k \cdot H_k(T); \quad S = \sum_{i=1}^{nchem} x_k \cdot S_k(T) + \sum_{i=1}^{nchem} y_k \cdot \ln y_k$$

cp,k cv,k are the specific heats at constant pressure and volume respectively for a pure and real Van der Waals gas, Hk and Sk are the specific enthalpy and entropy for a pure and real Van der Waals gas. These parameters are computed as a pure fluid (using the formulation previously explained).

Based on that γ and vsound can be computed as follows:

$$v_{sound} = \sqrt{\frac{c_p}{c_v \cdot \rho \cdot \kappa}}$$

Regarding the volumetric expansion (β) and the isothermal compressibility (κ) coefficients, they are computed differently. For the computation of the Van der Waals's characteristic parameters, a and b, mixinjg rules have been used. They are summarized as follows:

$$a_{mix} = \left(\sum_{k=1}^{nchem} y_k \cdot a_k^{0.5} \right)^2 \qquad b_{mix} = \sum_{k=1}^{nchem} y_k \cdot b_k$$

β and κ are computed as follows:

$$\beta = \frac{1}{V}\left(\frac{\partial V}{\partial T}\right)_{P=ct} = \frac{R \cdot V^2}{PV^3 + 2 \cdot a_{mix} \cdot b_{mix} - a_{mix} \cdot V}$$

$$\kappa = -\frac{1}{V}\left(\frac{\partial V}{\partial P}\right)_{T=ct} = \frac{-\left[V \cdot (V - b_{mix})\right]^2}{2 \cdot a_{mix} \cdot (V - b_{mix})^2 - R \cdot T \cdot V^3}$$

4.5.2.2 Transport Properties

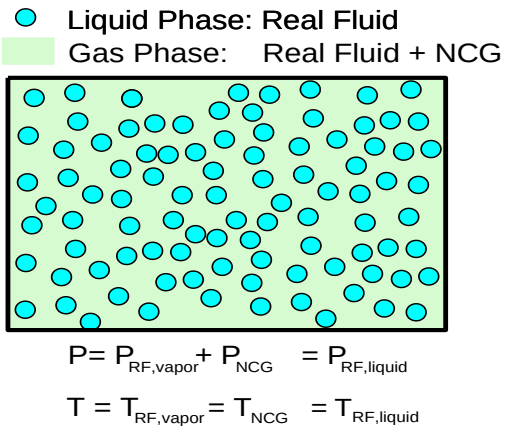
Regarding the transport properties, they have been computed similarly as in perfect gas mixtures.

4.5.3 Real Fluid – Perfect gas mixture calculation

4.5.3.1 The homogeneous equilibrium model (HEM)

The homogeneous equilibrium model of a mixture of two fluids is calculated.

One fluid must be real or a simplified liquid, and the other a perfect non-condensable gas (NCG)



Assuming that:

- The Real Fluid in possibly sub cooled liquid, saturated or superheated vapor state and the NCG form a homogeneous mixture with a uniform temperature.
- The Real Fluid, if present, occupies the entire volume. NCG, if present, occupies the same volume as the Real Fluid vapor according to the Gibbs Dalton Law.
- If NCG and Real Fluid Liquid are present, the Real Fluid vapor is saturated (relative humidity is equal to 1)
- If NCG is present, the Real Fluid liquid conditions are the sub cooled conditions corresponding to P and T. Liquid phase pressure = P_{vap} + P_{NCG}),
- The NCG gas is insoluble in the Liquid Phase of the Real Fluid. There can be no NCG if the volume is filled with the liquid phase of the Real Fluid

The following state equations are involved in presence of liquid:

$$\rho_{liq}, u_{liq} = f_{state}(fluid, P, T) \quad where \quad P = P_{nc} + P_{vap} \quad (sub\,cooled\,conditions)$$
$$\rho_{vap}, u_{vap} = f_{sat}(fluid, P_{vap}) \quad where \quad P_{vap} = f_{sat}(fluid, T) \quad (saturated\,cond.)$$

"u" is the internal energy. Subscript "nc" denotes the non-condensable fluid. "f" denote the corresponding pure fluid functions. In this system of equations, Pnc and T are unknowns.

Assuming that the volume density, ρ, the non-condensable mass fraction, x_{nc}, and the mixture energy, u, are known (dynamic variables in the capacitive fluid components), the following closing equations allow the calculation of the homogeneous temperature and the non-condensable pressure:

$$\rho_{nc}\alpha = \rho\, x_{nc} \quad where \quad \rho_{nc} = f_{state}(fluid_{nc}, P_{nc}, T)$$
$$u = (1 - x_{nc})(x\, u_{vap} + (1 - x)u_{liq}) + x_{nc} u_{nc}$$

Applying the definitions of the void fraction (α=gas_volume / total_volume) and quality (x = vapor_mass / (vapor_mass+ liquid_mass)), it is possible to find an expression for the quality appearing in the equation above as a function of densities:

$$\alpha = (\rho_{liq} - \rho_{cond})/(\rho_{liq} - \rho_{vap})$$
$$x = \alpha\, \rho_{vap}/(\rho_{liq} - \alpha(\rho_{liq} - \rho_{vap}))$$

The variable $\rho_{cond} = (1 - x_{nc})\rho$ refers to the condensable mass (liquid & vapor) divided by the total volume. The non-condensable density ρ_{nc} and the vapor density refer to the gas volume and not to the total volume, so $\rho_{nc} \neq x_{nc}\,\rho$. For the sake of clarity, it can be seen that the last two equations are identities introducing the definition of:

- gas_volume = vap_volume = non_condensable_volume
- total_volume = gas_volume + liquid_volume
- α = gas_volume / total_volume,
- quality = vapor_mass / (vapor_mass+ liquid_mass).
- ρ = (nc_mass + liquid_mass + vapor_mass) / total_volume
- ρcond = (vapor_mass + liquid_mass) / total_volume
- ρnc = nc_mass / gas_volume
- ρliq = liquid_mass / liquid_volume
- ρvap = vapor_mass / gas_volume

A double iteration by Newton-Raphson in T and Pnc is required to solve the equations from above. In general, the Newton-Raphson iteration in T and Pnc converges very well unless the non-condensable mass fraction in a liquid medium is very small. To protect these situations, the minimum mixture

temperature is limited to 0.1*Ttriple. Furthermore, a warning message will be issued if the converged quality is lower than zero.

In case of ρ(1-xnc) < ρvap or T>Tcrit, the equation system is simpler: If no liquid is present, none of the previous will apply. The conditions of reheated fluid will be used instead for the vapor:

$$\rho_{vap} = \rho(1-x_{nc})$$

$$P_{vap} = f_{state}(fluid, T, \rho_{vap}); \qquad \alpha=1$$

4.5.3.2 Void fraction, transport properties and speed of sound calculation

The heat capacity, viscosity and thermal conductivity are calculated as a mixture of a liquid and a composed gas (the vapor and the non-condensable gas). The mixture properties are calculated simply (weighing the pure fluid properties with the mass fractions):

$$c_p = x_{mix} c_{p,gas} + (1-x_{mix}) c_{p,liq}$$
$$\mu = x_{mix}\mu_{gas} + (1-x_{mix})\mu_{liq}$$
$$\lambda = x_{mix}\lambda_{gas} + (1-x_{mix})\lambda_{liq}$$

where the mixture quality (x_{mix}) is defined as the mass ratio of gas (vapor + non condensable)

$$x_{mix} = \alpha\,\rho_{gas}/(\rho_{liq} - \alpha(\rho_{liq} - \rho_{gas}))$$

The void fraction α has the same meaning than in a pure two-phase fluid, i.e. the gas volume divided by the total fluid volume.

The gas mixture properties are calculated as follows:

$$\rho_{gas} = \rho_{vap} + \rho_{nc}$$
$$c_{p,gas} = (\rho_{vap} c_{p,vap} + \rho_{nc} c_{p,nc})/\rho_{gas}$$

Similarly, the gas mixture transport properties and sound speed are calculated as follows:

$$\mu_{gas} = (\rho_{vap}\mu_{vap} + \rho_{nc}\mu_{nc})/\rho_{gas}$$
$$\lambda_{gas} = (\rho_{vap}\lambda_{vap} + \rho_{nc}\lambda_{nc})/\rho_{gas}$$
$$v^2{}_{sound,gas} = \rho_{gas}/(\rho_{vap}/v^2{}_{sound,vap} + \rho_{nc}/v^2{}_{sound,nc})$$

All individual properties have been computed with pure fluid functions.

The sound speed is approximated as an equivalent two-phase mixture where the vapor phase is in fact a mixture of a non-condensable fluid with 100% humidity:

$$1/v^2{}_{sound} = (\alpha\rho_{gas} + (1-\alpha)\rho_{liq})(\alpha/\rho_{gas}/v^2{}_{sound,gas} + (1-\alpha)/\rho_{liq}/v^2{}_{sound,liq})$$

5. FLUID_FLOW_1D LIBRARY

5.1 OVERVIEW

FLUID_FLOW_1D is an ECOSIMPRO library for 1D *transient* simulation of two-fluid, two-phase systems. Below are its most important features:

- The conservation equations include gas, liquid and two-phase flow regimes for ideal or real fluids. The working fluid(s) can be easily selected from a large collection of fluids included in the FLUID_PROPERTIES library.
- The fluid phase will be automatically calculated. The homogeneous equilibrium model is used to calculate a real fluid under two phase conditions with or without a non-condensable gas mixture.
- Absorption/desorption effects of non-condensable gases are optionally included in the formulation.
- Flow inversion, inertia, gravity forces and high speed phenomena are considered in pipes, volumes and junctions, the pipes also incorporating an area-varying non-uniform mesh 1-D spatial discretization into n (input data) volumes.
- Calculation of concentrated (valves) and distributed (pipes) load losses including two-phase wall friction correlations.
- Heat transfer between the walls and the fluid. Multiple thermo-hydraulic correlations and initialization options are included.
- Other special components such as check valves, pressure regulators, heat-exchangers and Tees (for convergent and divergent flows) are available.
- 1D Pipe flows can be simulated using some of the most up-to-date, robust and accurate CFD techniques upwind (Roe) or centred schemes.

Hydraulic or pneumatic systems where the heat transfer or system controls are coupled will be easily evaluated with the FLUID_FLOW_1D library. Cavitation and priming phenomena under two-phase flow (with or without a non-condensable gas traveling in a liquid) will be calculated in pipes or other components. Moreover, FLUID_FLOW_1D can be used in making detailed analyses of transient aspects due to inertia (water-hammer) and bubble collapse (priming).

The FLUID_FLOW_1D library provides a large palette of components (represented by icons) to be inserted (click and drag) in a model. Other components that you might require can be easily built by means of inheritance and aggregation.

5.1.1 Component classification

The components of the FLUID_FLOW_1D Library are listed in Figure 5-1 below, which shows the inheritance hierarchy.

In an ESPSS fluid network, every component is either a resistive component or a capacitive component. A resistive component receives the state variables (pressure, density, velocity, chemical composition and enthalpy) as input and gives back the flow variables (volumetric, mass and enthalpy flows) as output. A capacitive component receives the flow variables as input and gives back the state variables at output. To build a fluid network, the user must connect resistive components to capacitive ones, alternatively. So, from a computational point of view, components are divided into two classes:

- C (capacitive) elements, integrating the mass and the energy conservation equations. The thermodynamic functions will be used to calculate the complete thermodynamic state
- M (momentum) elements, calculating explicitly (inertia terms) the mass flows between C elements. Reverse flow is allowed.

This computational scheme prevents the appearance of algebraic loops and high index DAEs (Differential Algebraic Equations) in the mathematical model of the pipe network.

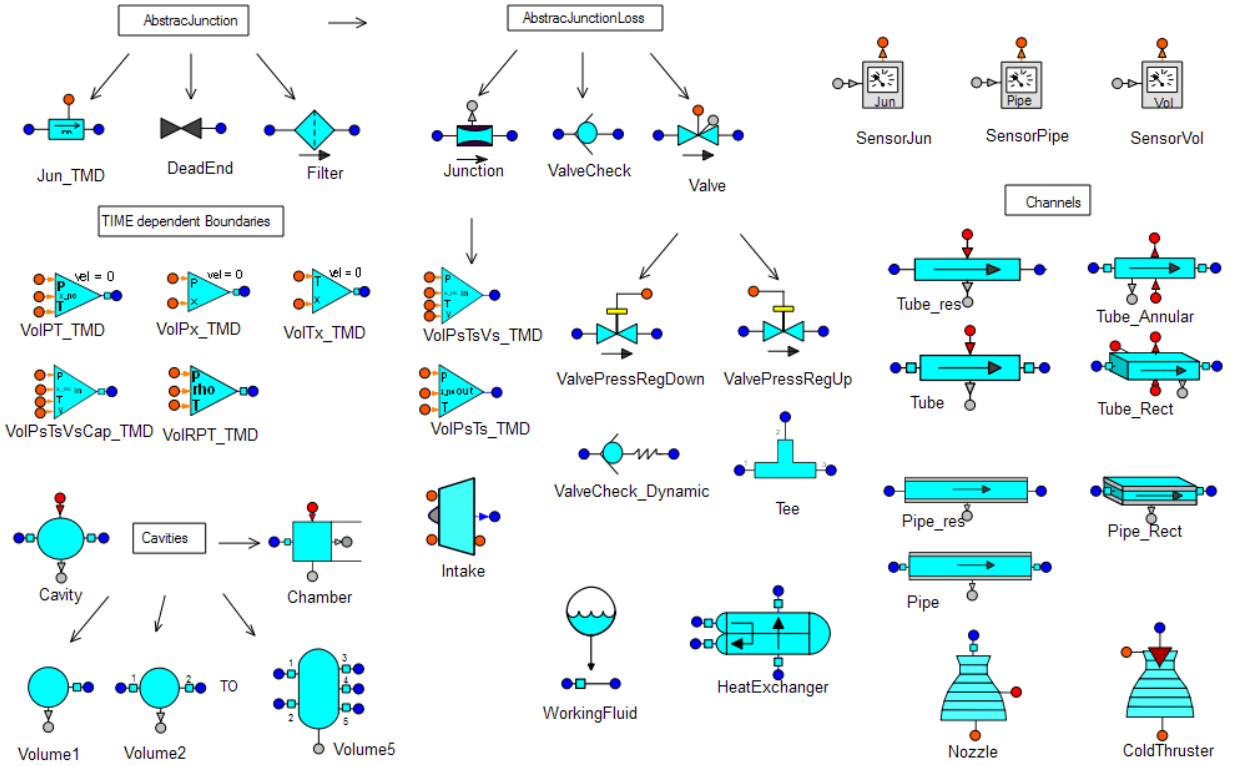


Figure 5-1 FLUID_FLOW_1D palette of symbols

The existing components of the FLUID_FLOW_1D library have the following types:

- Volumes and Heat-exchangers components are C elements.
- Junctions, Valves, Filter and Jun_TMD components are M elements.
- Bound components (VolPT_TMD, VolPx_TMD, etc.) are also C elements. (Here, TMD means time dependent). VolPsTsVs_TMD and VolPsTs_TMD are M elements because they calculate mass flow. In TMD elements, state variables are imposed in the experiment file (fixed or depending on the time) or by means of CONTROL library components connected to its CONTROL ports.
- The Intake component behaves as the VolPsTsVs_TMD component but including the standard atmosphere relationships
- Pipe, Tube, Tube_Rect, Tube_Annular, HeatExchanger and Nozzle components are a chain of alternating C and M elements beginning and ending with a C element.
- Pipe_res, Tube_res and ColdThruster components are a chain of alternating C and M elements beginning and ending with an M element. The ColdThruster incorporates an internal valve component.
- Others topological components such as the Tee component are M elements because they internally finish in junctions, even if they have some internal C components.

The graphical symbols of the components provide information about the kind of computational element to which each port is connected. Ports belonging to a C element have a small dot in the middle of the arrow while ports belonging to an M element are just represented by the arrow (see Figure 5-1).

It is noted that the Tube and Pipe components can simulate an *area-varying non-uniform* 1-D *mesh,* as is the case for the *ColdThruster* and *Nozzle* components, even though, for simplicity, the symbol graphical representation does not indicate this capability on Pipes and Tubes components.

5.1.2 Thermodynamic Property Functions

Beside the resistive/capacitive computational scheme, the use of thermodynamic functions is another of the basic techniques programming ESPSS components. Depending on which type of fluid has been chosen, the same function will provide either a real fluid behavior (including homogeneous two-phase mixtures) or an ideal one. All the thermodynamic properties are provided by these functions, so no major modifications will be needed in components if new improvements are made in fluid properties.

Thermodynamic functions are collected in a special library called FLUID_PROPERTIES (see §4.1). Components are designed to deal with two types of mixtures of fluids depending on the choice for **burnerGasesOption** (a parameter for most of the ESPSS components):

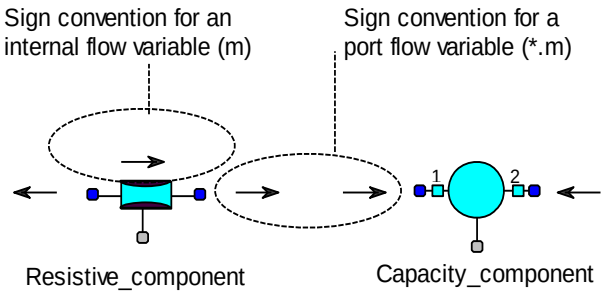
- *burnerGasesOption = FLUID_PROPERTIES.noBurnGases* (default value) if a component is not downstream of a combustor: an homogeneous equilibrium model of a **pure fluid** (in one or two-phase conditions) with a perfect gas will be applied. The corresponding thermodynamic functions will automatically detect when the vapor begins to condensate or vice versa (cf. §4.5.3).Three categories of state of equation are foreseen for *pure* fluids:
  - Perfect gases. Cp and transport properties are read from files as tables of temperature, cf. §4.3.1 and §4.4.3.
  - Simplified liquids. Cp, sound speed, density and transport properties are read from files as tables of temperature, cf. §4.3.2 and §4.4.4.
  - Real fluids "read" from files that consider all possible zones of operation (liquid, superheated, supercritical and two-phase flow), cf. §4.3.3, §4.4.5.
- *burnerGasesOption = FLUID_PROPERTIES.Chemicals* if a component is placed downstream of a combustor: a **chemicals' mixture** of variable composition is calculated with the Perfect gas or the Van der Walls state of equation (boundary global variable *VDW_option* = 0 or 1 respectively). Properties follow CEA code, but in particular EcosimPro Language functions, cf. §4.4.1, §4.5.1.

5.1.3 Building a Model

5.1.3.1 Model arrangement and mass flow sign criteria

Models of arbitrary piping networks are built by an alternating (C – M) arrangement of the two types of FLUID_FLOW_1D components. It is useful to think of the C elements as physical volumes and the M elements as the interface, junctions, between adjacent C elements. The Pipe component behaves as a chain of C-M-C-M …-C elements. Special components such as Tees behave as volumes with each one of its ports ending with M elements.

Reverse flow is allowed, even if the connection of a component is flipped. Capacitive (C) components have IN type fluid ports, and the opposite for resistive (M) components.



The sign convention is: Mass flow at ports (**.m* variable) is positive if it enters into a capacitive component or if it exits from a resistive component. Otherwise, the symbol of pipes and junctions will show an arrow indicating the flow direction with positive values of the internal variable "m" (junctions) or "m_jun[]" (pipes).

The FLUID_FLOW_1D library models high speed phenomena in pipes. Among the calculated variables are the static pressures and temperatures, and the Mach numbers.

5.1.3.2 Defining the working fluids

- FLUID_FLOW_1D models must use a special component called "*WorkingFluid*" (see §**5.4.24**) that defines the working fluids of a loop *not connected to any combustor***.** It can be inserted anywhere in the loop. All interconnected components will have the same working fluids. Every time there is a disconnected loop in a model, the insertion of this component will be required.

  For the first (main) fluid, any fluid category described previously is possible. For the second fluid (non-condensable gas), only perfect gases can be chosen. The homogeneous equilibrium model of a real fluid with a non-condensable fluid is *implicitly* calculated. This makes the CPU time increase when a non-condensable fluid is "traveling" along a liquid system. The available fluids are described in §4.1.2.

  Simplified liquids are assumed to have a null constant vapor pressure. Then, using this kind of fluid, bubble formation should take place when the pressure reaches a negative value (water-hammer). Under these circumstances, capacitive components (pipe and volumes) will calculate the corresponding void fraction and the pressure peaks due to possible bubble collapse much as if a real fluid has been selected. The difference is that a real fluid would take into account the real vapor pressure, density and latent heat (assumed to be null for a simplified liquid).

- Components connected to a Combustor of the COMB_CHAMBERS library (cf. §**8**) do not need a WorkingFluid component because the combustor itself will define dynamically the composition of the burned gases. See §5.1.2.

5.1.3.3 Initializing fluid conditions

Capacitive components (Volumes and Pipes) can be initialized in different ways according to the INIT_OPTION data. Not all the initialization inputs (P_o, T_o, rho_o, x_o, x_nco) are necessary:

- INIT_PT. In this case, the initial temperature and pressure (T_o and P_o data) will be imposed. "x_nco" data will impose the initial non-condensable mass fraction.

  INIT_RT. The same as before, but using density (r_o) instead of pressure. This option can be useful to initialize a tank with a known total mass. Using a perfect gas plus a liquid, "r_o" refers only to the *condensable* fluid (assumed to be liquid).

  INIT_TX. Only used in saturation conditions. The initial temperature and quality (T_o and *x_o* data) will be imposed. This option is forbidden with non-condensable gases (x_nc=0).

  INIT_PX. Same as before, but using saturated pressure (P_o) as input instead of temperature.

  INIT_RPT. This option is only valid using a perfect gas plus a liquid. In this case, initial temperature, pressure and *condensable mass* per unit of total volume are imposed; the initial non-condensable mass fraction being calculated, not imposed.

- Initializing a model and choosing the appropriate boundaries are very important in view of the many possible solutions in fluid mechanic: in particular, be careful imposing mass flows not compatible with the sonic limitations, or initializing inappropriate P/T conditions with the required fluid conditions.

  As a general rule, it is recommended to initialize a circuit with the same conditions as in the nearest boundary. The use of global data variables (pressure and temperature) is useful to initialize the same conditions in different components. In normal situations it is advisable to initialize with no mass flow (m_o = 0). A non-zero initial mass flow can force high initial accelerations. See §3.2 and §3.3 for more details.

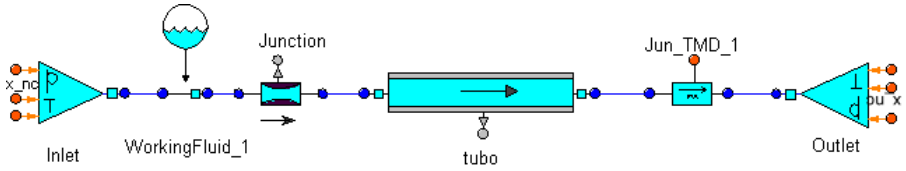
- Every capacitive component can be initialized with a certain non-condensable mass or void fraction, so that a gas mixture or gas sweeping processes can be easily studied. If one part of the circuit is filled with non-condensable gas and other parts with a liquid (main fluid), make sure that the initial non-condensable mass fraction is 1 on the gas sides and 0 on the liquid sides.

- *Be aware that any component working with combusted gases (therefore, placed downstream of a combustor) will be automatically initialized with non-condensable gases at the initial values of P/T because it is not possible, a priori, to know the chemical composition that would be produced in the respective upstream combustors.*

5.1.3.4 Boundary conditions

- Boundary components (see §5.4.5 and §5.4.20) will determine the P/T/Rho/mass fraction at the model interfaces. The boundary values can be time dependent. The boundary conditions should be modeled with the appropriate component: (Volxx_TMD, …). Do not use the volume component for a boundary. For example, if only liquid is present in the volume at time = 0, a small outgoing flow will force the liquid to turn into vapor.

  Boundary values can be defined with CONTROL components connected to the boundary ports. If the CONTROL ports of these Voxx_TMD components are disconnected, the boundary conditions will be automatically inserted in the experiment file. (See the explanations on the experiment printout of the application example contained in Section 3.3):



Note in the previous figure how a boundary in inlet or outlet mass flow (Jun_TMD component) needs to be connected to another VolPT_TMD component (alias Outlet in this case) so that the pressure and temperature would be known in case the forced mass flow were going into the system.

```
DECLS
  REAL Tini = 300
  TABLE_1D wl =  {{0, 0.05, 0.051, 10}, {1, 1, 0, 0}}
…
BOUNDS    -- set expressions for boundary variables: v = f(t,…)
  Jun_TMD_1.s_massflow.signal[1] = timeTableInterp(TIME,wl)

Inlet.s_pres.signal[1] = 25e5
  Inlet.s_temp.signal[1] = Tini
  Inlet.s_xNonCond.signal[1] = 0.001

  Outlet.s_pres.signal[1] = 1e5
  Outlet.s_temp.signal[1] = Tini
  Outlet.s_xNonCond.signal[1] = 0.
…
```

5.1.3.5 Recommendations and tips

- Use the "*Volume*" component to simulate an adiabatic fluid capacity under liquid, vapor or two-phase conditions. This component is the simplest model of a tank for calculating the pressure and temperature evolution according to the inlet/outlet mass flows. For more complex reservoirs, use the TANKS library components. Volume components are also representative of fluid capacities mixing or splitting flows depending on the conditions of connected components.

  The *Cavity* component, besides the previous capabilities, simulates non adiabatic fluid capacities disposing of a thermal port.

  The pressure of a Volume (*defined at the top of the volume*) should not be confused with the pressure at an outlet of the volume: The latter takes into account the gravitational effects depending on the actual liquid level and on the junction heights.

- The boundary conditions should be modeled with the appropriate component: (Volxx_TMD, …): VolPx_TMD, VolTx_TMD and VolPT_TMD simulate tank conditions at zero speed. VolPsTsVs_TMD impose *static* conditions and speed at Pipe inlets and VolPsTs_TMD static conditions at pipe exits (for subsonic or supersonic conditions). Use VolPsTsVsCap_TMD to simulate static conditions at the boundaries of *resistive* Pipes (for subsonic or supersonic conditions).

- Use the "*Pipe_...*" or "*Tube_...*" type components to simulate a 1D line discretized in several nodes. Both types of components include internal heat exchange and two-phase wall friction correlations. *Pipe* components include the simulation of the wall assuming one node per fluid node, but they do

not have an external thermal port to be linked to another thermal component (the Pipe is assumed to exchange heat with the exterior through a constant external heat exchange coefficient). In contrast, the "*Tube*" component that includes a thermal port (but not the thermal wall nodes) must be used with other thermal components simulating the wall (a "THERMAL.Cylinder" component for example).

As a general rule, we recommend using few (1-to 5) nodes per pipe if upi want *only* to simulate the *global* pipe inertia and pressure drop effects. You can add more nodes if you want to obtain particular results at a *given* distance inside the Pipe. *The use of very small pipes can give rise to numerical problems (time step proportional to the smallest length of the Pipe)*. You can avoid this by using alternative components such as a volume or, indeed, just by including these small Pipes in other closer lines or Junctions.

In Priming cases, the "Inertance Rule", where the number of nodes is chosen to have similar values of (dL/A) for all tubes and pipes of the system, gives good results [RD-7]. *A complementary rule in priming cases is to take (as maximum) one node per diameter length*.

- Use the "*Tee*" component for single pipe bifurcation or collection of flows. For connections involving more than three pipes, use the Volume components and put a junction (or a control valve) between the volume and the pipes.
- Junctions, Valves, and Filters components allow simulating concentrated pressure drops connecting capacitive elements. They should be placed in the model just where the physical fitting component is found.
- Use the "*DeadEnd*" component to close any volume or tank fluid ports that will not be used.
- System coordinates relative to a body axis are given in the junction type components. The inlet – outlet pipe position will be known through its connected junctions. This prevents a junction having two different elevations, those of the connected pipes, but assumes *that the Pipe is in a straight line*. When calculating the hydrostatic pressure contribution, the Δz data prevails in case of non-coherent data between the total pipe length input (L) and the x, y, z coordinates of the pipe ends; nevertheless, the total length input data (L) will be considered for the pressure drops and inertia effects. The volumes include the hydrostatic height effects, so an error message will be issued if the junctions connected to the Volume are beyond the volume's height.

5.1.3.6 Global simulation parameters

Global simulation parameters are library variables or solver parameters *that automatically appear in the models' experiments to control the integration*. The most important of these are:

- The Gravity vector components (GRAVX, GRAVY, GRAV) relative to a body axis system are global boundary variables to be defined in the experiment file.

  Gravity accelerations are assumed to be the same for any discretized volume of the pipe component. Centrifugal or Coriolis body forces (as a function of a global rotation speed and the volume position) are not included yet. For lumped volumes, only the vertical vector GRAV is active because the calculated volume height is assumed to be in the vertical direction.
- The library variable "*Damp*" (the artificial viscosity parameter) is an input to be defined in the experiment for every model. Values from 0.1 to 0.5 provide good smoothing of numerical pressure oscillations. In normal cases, Damp = 0.5 – 1.0 slightly smoothes water-hammer pressure peaks, but saves CPU time. Use Damp = 1 in priming cases or when acoustic effects are not important.
- The absolute and relative error permitted in the integration *(REL_ERROR and ABS_ERROR experiment variables) should be between 1e-4 and 1e-6*. New models should use REL_ERROR and ABS_ERROR = 1e-5. Then, you can try other smaller or greater values depending on the stability of the model. In normal cases, the calculation can be speeded up by setting these variables to 1e-4. In priming cases with non-condensable gases, lower REL_ERROR and ABS_ERROR values are normally used in the experiment.
- Mixture of chemicals (in Combustors and any other component placed downstream of a combustor) will be calculated with the Perfect gas or the Van der Walls state of equation depending on the

boundary global variable *VDW_option* = 0 or 1, respectively, applying to all the components of a model.

5.1.3.7 Numerical limitations

From a numerical point of view, the experience achieved from many tests is:

- All the *fluid* ports of a model should be connected. Otherwise, the default partition (mathematical arrangement of equations of the complete model, which is made automatically by ECOSIM) may have strange variables (elevations, enthalpy, etc.) as boundary conditions.
- Direct steady conditions are hard to obtain with the FLUID_FLOW_1D library due to the great quantity of dynamic variables programmed in the components' equations. In these situations, it is better to let the DASSL solver obtain the steady conditions after some integration time *or use alternative models built with the STEADY library [§***11***].*
- The calculation of the heat exchange coefficient in pipes takes into account two-phase regimes. There may be a problem when passing from supercritical to two-phase flow, especially near the critical point. Indeed, at the critical point some property derivatives are not defined and they do not have finite values. Then, some numerical correlations (the heat exchange coefficients, for example) using Cp risk having discontinuities. If this is the case (the simulation becomes slow), it is advisable to use use- ht_option = HT_tube (one phase correlations) or HT_critCond (so the heat exchange coefficient is calculated at Pr=1, without dependence on the cp value), see appendix A3, at least while this situation lasts (changing input data for a particular time window with the monitor tool).
- The Jun_TMD (imposed mass flow) component should only be used as an exit or inlet boundary condition. Placing it in the middle of a circuit can lead to numerical problems because it assumes continuity in enthalpy, while in fact this component would replace a pump where there is no enthalpy conservation.

  Time dependent boundary conditions should not force instantaneous mass flow changes because of the infinite fluid acceleration this implies. With respect to the valves, a characteristic actuator time of 0 (input data "tao"=0) can produce high "unphysical" pressure peaks.
- Library components have been designed to deal with *homogenous* two-phase, two-fluid conditions. In the extreme situation of a high pressure liquid front traveling along a pipe filled with gas at low pressure (priming cases), the pipe discretization can produce numerical pressure peaks. Nevertheless, the main value of the pressure surge and its reflections are well calculated with relatively few nodes.

  The simulation of a mixture process between the liquid and the gas depends somewhat on the number of nodes, especially in the situation of pipes filled with a mixture of liquid and non-condensable gas enclosed between two closed valves. Then, the flow in these pipes will no longer be 1D, but like in a *sloshing* volume, with low speed and high pressure fluctuation due to the abrupt stops of the sloshing flows against the closed ends.

  Under vacuum conditions, circuits can be initialized even with lower pressures than the saturation one (*a minimum of 10-100 Pa is recommended*). Try to reduce ABS_ERROR and REL_ERROR integration parameters in case of non-convergence (or when the simulation becomes blocked), although not to values lower than 1e-6.
- Sudden expansion of a liquid under the saturation pressure may produce very low temperatures, even under the triple point (snow formation). This dangerous situation is now better supported by assuming the vapor phase is extrapolated as a real gas at P<Ptriple, T=Ttriple and that the liquid behaves as a saturated liquid at Ttriple.
- The numerical scheme of the 1D mesh of the Pipes can be: *centred*, *Roe_1st* or *Roe_hr_lim*. *Centred* is the default scheme, *Roe_1st* is the upwind Roe scheme, and *Roe_hr_lim* is the Roe scheme with a high resolution reconstruction with limiters. *The "Centred" scheme behaves more robustly and is less time consuming than the Roe scheme but is less precise capturing shock discontinuities. Use ROE scheme only if necessary and for shock capture. The resistive Pipe (component "Pipe_res") can have numerical troubles in some particular cases, see RD-5, paragraph 2.1.6*.
- Activation of absorption -desorption effects increases the CPU time and the risk of instabilities.

5.2 LIBRARY ITEMS

5.2.1 Library Variables

The following variables are visible in every FLUID_FLOW_1D component and can be redefined in the experiment file (BOUND specification), see *§5.1.3.6*.

NAME	INITIAL	DESCRIPTION	UNITS
Damp	0.2	Artificial Viscosity Parameter	(-)
GRAV	REAL	9.806	
GRAVx	REAL	0	
GRAVy	REAL	0	

5.2.2 Port Types

5.2.2.1 *"fluid"* port

This port has the following variables:

NAME	DESCRIPTION	UNITS
burnerGasesOption	Type of mixture of fluids. Select chemicals only for components downstream of a burner (Port parameter)	-
A	Connected volume Area	m^2
Gcrit	Critical mass flow per unit area -set by volume	kg/m^2 s
I	Half inertia of the connected volume	m^-1
P	Static pressure - set by volume	Pa
Q	Volumetric flow - set by junction	m3/s
fluid	Working fluid name	
fluid_nc	Non-condensable working fluid name	
h	Total enthalpy (specific enthalpy+specific kinetic energy ½ V²)	J/Kg
m	Total mass flow - set by junction	Kg/s
m_nc	Non-condensable mass flow	Kg/s
md_nc	Non-condensable mass flow diluted in the liquid phase	Kg/s
mh	Total enthalpy * flow - set by junction	W
m_nx[burnerGasesOption]	Massflow of the chemical constituents. Only defined for the components placed downstream of a combustion chamber	-
x_eq[burnerGasesOption]	Mass fraction of the chemical constituents. Only defined for the components placed downstream of a combustion chamber	-
n_fluid	Number of times the working fluid is defined (*always 1 if model is correct*)	-
rho	Static density - set by volume	kg/m3
T	Static temperature - set by volume	K
v	Velocity - set by volume	m/s
x_nc	Non-condensable mass fraction set by volume	Kg/s
xd_nc	Non-condensable mass fraction diluted in the liquid phase set by volume	Kg/s
x_jun	Junction OX coordinate- set by junction	m

NAME	DESCRIPTION	UNITS
y_jun	Junction OY coordinate- set by junction	m
z_jun	Junction elevation - set by junction	m
visc	Viscosity set by volume	Kg/m s

Note: The pressure is corrected by a numeric term (artificial viscosity see §A9.2) necessary to smooth *numeric* pressure oscillations. This term is null under steady conditions.

5.2.2.2 "vol_measure, jun_measure and pipe_measure" ports

These ports have different types to prevent bad connections between the components and the corresponding sensors.

The ports have only an array variable to save several magnitudes calculated inside the components:

NAME	TYPE	DESCRIPTION
n	CONST INTEGER	Number of output signals (Port parameter)
signal[n]	EQUAL OUT	Real variable array containing measures

Depending whether this port belongs to a capacitive, resistive or pipe type component, the list of magnitudes saved are:

-- Fluid volumes output signals (14 signals):
meas_out.signal[1] = Pressure (Pa)
meas_out.signal[2] = Temperature (K)
meas_out.signal[3] = Fluid mass (kg)
meas_out.signal[4] = Level (m)
meas_out.signal[5] = Internal wall heat exchange coefficient (J/s/m2)
meas_out.signal[6] = dV (m3/s)
meas_out.signal[7] = Volume (m3)
meas_out.signal[8] = Void fraction, (-)
meas_out.signal[9] = Sound speed (m/s)
meas_out.signal[10]= Enthalpy shift respect to CEA, only for combustor cavities, (J/kg)
meas_out.signal[11]= Non-condensable partial pressure, (Pa)
meas_out.signal[12] = Vapor quality, (-)
meas_out.signal[13] = Gas enthalpy according to CEA, only for combustor cavities, (J/kg)
meas_out.signal[14] = Liq. enthalpy according CEA, only for combustor cavities, (J/kg)

-- Resistive output signals: (2 signals)
meas_out.signal[1] = Mass flow, kg/s)
meas_out.signal[2] = Enthalpy flow, J/s)

-- Pipe output signals (dimensioned to 3 times the number of nodes):
meas_out.signal[1+3*(i-1)] = P[i] (Pressures, Pa)
meas_out.signal[2+3*(i-1)] = T[i] (Temperatures, K)
meas_out.signal[3+3*(i-1)] = m_jun[i+1] (mass flows, kg/s)

It is important to note that these port variables can be used in the CONTINUOUS block of topological components to implement particular formulations involving different components. An example is in the "CombustChamber_ramjet" component using in this case the port signals generated in the Intake component, see §8.3.7.5.

5.3 ABSTRACT COMPONENTS. MAIN FORMULATION

Abstract components describe some physical behavior that does not represent a complete component, but can be used as a base for other components. These components contain the main formulation later used in the inherited components. (EcosimPro inheritance property)

5.3.1 AbstractJunction

This component represents the basic abstract junction where no mass accumulation is considered.

5.3.1.1 Mass and Energy Conservation Equations

The mass and enthalpy flows of the inlet and outlet ports are equal (no mass accumulation in junctions):

$$m_1 = m_2 = m$$
$$m\_nc_1 = m\_nc_2 = m\_nc$$
$$md\_nc_1 = md\_nc_2 = md\_nc$$
$$mh_1 = mh_2 = mh$$
$$m\_nx_1[i] = m\_nx_2[i] = m\_nx[i]$$

where indexes 1 and 2 refer to the connected fluid ports. For each port, "m", "m_nc", "md_nc", "mh", "m_nx[i]" are the total mass flow, non-condensable mass flow, non-condensable mass flow diluted in the liquid phase, enthalpy and chemicals mass flows. These flows are calculated taking into account the flow direction (use of the function "donor_cell"), so different temperatures, mass fractions and densities may exist at both sides of a junction:

```
mh     = m * donor_cell(m, h1, h2) + mcond
m_nc   = m * donor_cell(m, x_nc1, x_nc2)
md_nc  = m * donor_cell(m, xd_nc1, xd_nc2)
Q =      m / donor_cell(m, rho1, rho2)
m_nx[i] = m * donor_cell(m, x_eq1[i], x_eq2[i])
```

where "h", "rho", "x_nc", "xd_nc" and "x_eq" are the enthalpy, density, non-condensable mass, non-condensable mass diluted in the liquid phase and chemical species mass fractions, all of them calculated by the connected volumes (capacities, see §5.3.3). The last equation to calculate the mass fraction of the chemical species is only used if the port parameter *burnerGasesOption = Chemicals*; if *burnerGasesOption = noBurnGases* then the mass fractions of the chemical species are equaled to 0.

The function "donor_cell" determines an abrupt change (but continuous) in an upstream property depending on the mass flow direction. The property returned (function value) will be calculated as a function of the flow (first argument) and the upstream and downstream properties values (second and third arguments):

$$\text{Property} = 0.5\left[(1+c)\cdot Upstream\_prop + (1-c)\cdot Downstream\_prop\right]$$
$$c = flow/(abs(flow)+1e\text{-}10)$$

The junction mass flow, m, will be calculated in the junction type inherited components according to their nature, see below.

5.3.2 AbstractJunctionLoss

Inherited from an "AbstractJunction", this component represents a concentrated load loss with sonic flow limitation. It is the basic *resistive* abstract component containing the momentum conservation equation for this type of component.

5.3.2.1 Momentum Conservation Equation

The following momentum balance equation dynamically calculates the mass flow per unit of area:

$$(I_1 + I_2)\,(A \cdot \frac{dG}{dt} + G \cdot \frac{dA}{dt}) + \mathrm{lv} * \frac{dG}{dt} = (P + 0.5\rho v^2)_1 - (P + 0.5\rho v^2)_2 - 0.5(\zeta + \zeta_{crit})\frac{G|G|}{\rho_{up}}$$

where,

P_1 , P_2 = static pressure at port 1 and 2, calculated by the connected volumes

$(0.5\rho v^2)_{1\text{-}2}$ = dynamic pressure at port 1 and 2, calculated by the connected volumes. ρ and "v" are the mean density and speed at the connected volumes

I_1 , I_2= half inertia of the connected pipe ends 1 and 2, respectively

A = actual valve area

Lv = sqrt(Aref). This term (valve inertia) is very small but makes the equation not singular if A=0

G = mass flow per unit of area

ζ= pressure drop coefficient (-)

ρ_{up}= upstream gas/liquid mixture density

The mass flow will be calculated as $= G\,A$. The use of G (mass flow per unit or area) instead of m (mass flow) allows a complete closing (A = 0) of the valve without making singular the system of equations.

Dynamic pressures at port 1 and 2 are calculated at the connected volumes using their mean velocities v1 and v2. These velocities are multiplied by the cosine of the port angles (see §5.3.3.3).

5.3.2.2 Laminar-turbulent regimes

The pressure drop contribution is quadratic with mass flow: $0.5(\zeta + \zeta_{crit})G|G| / \rho_{up}$.

This term would make the momentum equation singular at zero flow because very small perturbations in pressure lead to non-negligible variations in mass flow ($\partial G / \partial P \to \infty$). But physically, what is really happening is that pressure losses are linear with mass flow for laminar regimes. To account for this, the quadratic term $G|G|$ is linearized for G<G_{lam} using the "fpow" function included in the MATH library:

$$G|G| = \begin{Bmatrix} k(G)G & G < G_{lam} \\ G^2 sign(G) & G > G_{lam} \end{Bmatrix}$$

$$G_{lam} = \eta\,\mathrm{Re}_{lam}\sqrt{\mathrm{A}}; \qquad k(G)_{G\to 0} \to G_{lam}$$

k(G) is a smoothing factor to assure continuous transitions between laminar to turbulent regimes. η is the upstream viscosity calculated by the connected volumes, Re_{lam} is input data for Junctions and Valves. Its default value is 2000. This value should be reduced to about 50 for sharp small orifices at low pressure, see Appendix A11.

5.3.2.3 Sonic flow limitation

The sonic flow limitation is taken into account by applying a correction factor (as an adder) to the pressure loss coefficient, ξcrit, which limits the mass flow per unit area to be less than or equal to (≤) the critical flow per unit area. First, the flow under steady state conditions is obtained by cancelling the derivatives in the previous momentum equation:

$$G_{st} = \sqrt{2\rho_{up}((P + 0.5\rho v^2)_1 - (P + 0.5\rho v^2)_2)}$$

The adder factor is calculated in such a way that if the flow attempts to be greater than critical flow, the following extra term will limit the flow to the critical value:

$$\zeta_{crit} = \max\,((G_{st} / G_{crit})^2 - \zeta, 0)$$

where "Gcrit" is the critical (sonic) flow per unit of area $(\rho c)_{crit}$ calculated by the capacity components (see §5.3.4) connected by the Junction. The critical mass flow can be calculated in two ways: computing

an ideal expansion until sonic conditions (only for pure fluids) or from an *empirical* correlation based on real gamma values, see Appendix A1.

5.3.2.4 Subsonic/supersonic transitions

If the input data "choked_option" = FALSE, the limitation of sonic speed is cancelled. Supersonic conditions are automatically detected and decoupled from the current downstream conditions:

Subsonic conditions will be activated if:

```
WHEN(supers_out AND (dP < 0 OR f1.rho*f1.v/Gcr < 0.99)) THEN
        supers_out = FALSE
END WHEN
```

Supersonic outlet begins when P>Pout and G> Gcrit

```
WHEN(f1.P > f2.P AND f1.rho*f1.v/Gcr > 1.01) THEN
        supers_out = TRUE
END WHEN
```

where $dP = (P + 0.5\rho v^2)_{up} - (P + 0.5\rho v^2)_{down}$ and G is the mass flow per unit of area.

The momentum equation, see §5.3.2.1, will be modified under supersonic conditions (supers_out = TRUE): Instead of using the pressure drop balance, the upwind supersonic flow is maintained:

$$(I_1 + I_2)\,(A \cdot \frac{dG}{dt} + G \cdot \frac{dA}{dt}) + \mathrm{lv} * \frac{dG}{dt} = 1000(G_{up} - G)vel_{up}$$

5.3.3 Capacity

The "Capacity" component simulates a volume with several fluid ports named *f[j]*. It is the basic *capacitive* abstract component containing the mass and energy conservation equations.

5.3.3.1 Conservation Equations

Here below are the general equations for a non-adiabatic variable volume. It is assumed that all the mixture (non-condensable plus main fluid in liquid, gas or two phase conditions) is at only one temperature.

Mass conservation equation:

$$\frac{d\rho}{dt}V + \rho\frac{dV}{dt} = \sum_{j \in Ports} m_j$$

Non condensable mass fraction (x_nc) conservation equation:

$$\rho V \frac{dx_{nc}}{dt} + x_{nc}(\frac{d\rho}{dt}V + \rho\frac{dV}{dt}) = \sum_{j \in Ports}\left(m_{nc\,j} - md_{nc\,j}\right)$$

Non condensable mass fraction diluted in the liquid phase (xd_nc) conservation equation:

$$\rho V \frac{dxd_{nc}}{dt} + xd_{nc}(\frac{d\rho}{dt}V + \rho\frac{dV}{dt}) = \sum_{j \in Ports} md_{nc\,j}$$

Chemicals mass fractions (x_chem) calculation according to an homogeneous mixture (only when burnerGasesOption = Chemicals ; otherwise x'chem[i]= 0):

$$\rho V \frac{dx_{chem}[i]}{dt} + x_{chem}[i](\frac{d\rho}{dt}V + \rho\frac{dV}{dt}) = \sum_{j \in Ports} m_{nx\,j}[i]$$

Energy conservation equation:

$$\frac{d\rho}{dt}V\,u+\rho\frac{dV}{dt}u\;+\;\rho Vdu=\sum_{j\in Ports}(mh)_j\;+q_{in}-PdV$$

$$u=total\ specific\ energy=u_{st}+v^2/2$$

where ρ, x_nc, xd_nc, x_chem and u are the fluid mixture density (including two phase flow), the different mass fractions and the total energy respectively in the center of the volume; m_j, m_nc_j, md_nc_j, m_nx_j and mh_j are the respective mass flows and the total enthalpy flow at port number j calculated at the connected resistive type components (see §5.3.1). V is the volume, which can change with time.

5.3.3.2 Pressure, temperature and quality calculation

Assuming that the volume V and its rate of change are known, previous conservation equations can be used to calculate the derivatives of the mixture density, mixture energy and non-condensable mass fraction. These variables can be integrated, so they are known at any time.

Assuming thermodynamic equilibrium, the conservation equations are always valid even if the fluid conditions are liquid, vapor or *homogeneous* two-phase flow. Then, the complete thermodynamic state (partial pressures, temperature, *quality* …) can be calculated using the thermodynamic routines. (See §5.1.2):

```
FL_state_vs_ru(burnerGasesOption, x_chem, fluid,fluid_nc, rho,u-0.5*vel**2, x_nc,
        P_nc, phase, rhof, rhog, P, T, Tsat, hf, hg,
        x, alpha, cp, cpf, cpg, drho_dp, drho_dh, vsound,
        visc, viscf, viscg, cond, condf, condg, sigma, ipx, ipy, ier)
```

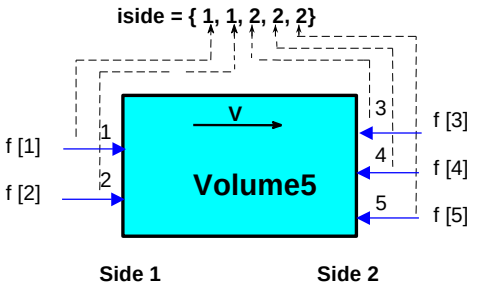
Inputs are: x_chem, chemicals mass fractions; *fluid* and *fluid_nc* with the fluid names and type; *rho, x_nc* and *u* are the mixture density, the non-condensable mass fraction and the mixture energy (dynamic variables). All the arguments from *P_nc*, *phase* (liquid, vapor or two-phase) to *ier* (the error code) are outputs: *x, alpha* are the quality and the void fraction respectively; *drho_dp, drho_dh* are thermo derivatives;

- Both actual and saturated (liquid and vapor) properties are returned. So, *cp, cpf, cpg* are the mixture, saturated liquid and saturated vapor heat capacities respectively; *hf, hg* are the saturated liquid and vapor enthalpies, *rhof, rhog* the saturated liquid and vapor densities, etc.
- With a non-condensable gas in the mixture, the saturated vapor density and enthalpy will contain the gas mixture (vapor + non-condensable) values. See §4.5.3.2 and §4.2.1 (FL_state_vs_ru function).

5.3.3.3 Volume-Average Velocities

Volume average velocities are required for the total energy conservation equation, the evaluation of the wall frictional forces and the evaluation of the wall heat transfer.

For the calculation of the average velocity, volumes are considered to have two sides, side 1 and side 2. The side of each fluid port is specified by means of the "iside" array. For example, a volume with 5 fluid ports is shown in next figure, where each blue arrow represents an *inlet* fluid port. Fluid ports numbers 1 and 2 are on side 1, and fluid ports 3, 4 and 5 are on side 2, and the iside values to specify such port locations are shown on top of the volume.



The total mass flow rate entering the volume at side 1 and 2 is

$$m_{1,in} = \sum_{\forall\ ports\ in\ side1} m_{j,in};\quad m_{2,in} = \sum_{\forall\ ports\ in\ side2} m_{j,in}$$

Port mass flows can be positive or negative. It is defined as positive when entering the volume. The average velocity in the volume is defined as

$$v = \frac{(m/\rho)_{in,1} - (m/\rho)_{in,2}}{2A}$$

where ρ is the upstream density of the corresponding inlets, and A is the cross area of the volume.

This average fluid velocity is transmitted to the ports to be used in the junction equations, see §5.3.2.1, after being multiplied by the cosines of the port angle α (input data) because the lateral velocities do not compute in the total pressure:

$$v(j) = v\ \cos(\alpha_j)$$

5.3.4 VolumeVariable

Inherited from a "Capacity", this component represents a generic variable volume with several fluid ports named *f[i].* No liquid separation is foreseen in case of two-phase flow conditions, so all the fluid ports will have the same density, quality and void fraction as the lumped volume.

The critical flow (compressible and sub-cooled choking) per unit of area $(\rho c)_{crit}$ at each port number j is calculated as a function of the pressure, density and the sound speed of the lumped volume. Two options are available: exact or approximate calculation, see Annex A1. This critical flow will be used in the connected junction type components to take into account the sonic flow limitation.

The pressure elevation at port number j produced by gravity forces is evaluated as follows:

$$\Delta P(j) = \rho\ g\ (z_{top} - z_{jun,j})$$

5.3.5 VolumeConstant

Inherited from a "VolumeVariable", this component serves to derive all the components that are adiabatic volumes with more than one branch. In this case, the volume variation term of the continuity equation is zero:

$$dV/dt = 0\ .$$

The q_{in} term appearing in the energy equation is also fixed to zero (*adiabatic process*).

5.3.6 NonAdiabaticVol

Inherited from a "VolumeVariable", this component represents a generic non adiabatic Volume. Mass and energy conservation equations are inherited from the Capacity component.

The term q_{in} appearing in the energy conservation equation of the parent component (Capacity) permits the exchange of heat through a THERMAL port.

The walls (which can be represented by THERMAL components) are not included in this component:

$$Q_{in} = h_{film} A_{wall} (tp.T(1) - T)$$

where *tp* is the name of the thermal port (with one node) connected to the Volume. The heat exchanged with the fluid is transmitted through this port:

$$tp.q(1) = Q_{in}\ .$$

tp.T(1) behaves as the *internal* wall temperature, to be determined in the connected THERMAL component. The film coefficient is calculated using empirical correlations (see Appendix A3).

$$h_{film} = htc\_fun(hf_{dat}, D_h, Cond, \mathrm{Re}, \mathrm{Pr}, ...)$$

5.3.7 Vol_TMD

This component represents time dependent (TMD) boundary conditions. The purpose of this component is to establish the complete thermodynamic state in a fluid port "f" by imposing two variables (P-T, P-x, T-x …) and the non-condensable mass fractions (x_nc). The following thermodynamic functions will be used in the inherited components:

Init_Vol(f.fluid, f.fluid_nc, INIT_ OPTION, 0, P_o, T_o, x_o, rho_o, P, P_nc, f.x_nc, u, T, rho, vsound, x, alpha, ier)

f.Gcrit = FL_Gcrit_fun(f.fluid, P, rho, T, alpha, vel, vsound, ier)

Subscript "_o" denotes the imposed variables. These functions will return all the unknown properties (T, partial pressures, and densities …, etc.) by calling the appropriate thermodynamic functions depending on the value of "INIT_OPTION" to be specified in the inherited components.

5.3.8 ABS_Tube, ABS_Tube_res

These components simulate *area-varying non-uniform mesh high resolved 1D fluid veins* that exchange heat with a 1D thermal port. The *ABS_Tube* abstract component has capacitive ports (P/H port variables are output) while the *ABS_Tube_res* component has resistive ports (calculating port mass flow).

Both incorporate the 1D mass, energy and momentum equations in transient conditions. The number of volumes in which the pipe is discretized will be a parameter.

5.3.8.1 Governing equations

All kinds of flows (compressible or nearly incompressible flows, single component or two-component flows, single phase or two-phase flows) can be simulated by using the following system of governing equations, here in area-scaled matrix conservation form:

$$\frac{\partial \omega}{\partial t} + \frac{\partial\, f(\omega)}{\partial x} = \Omega(\omega)$$

where:

$$\omega = A\begin{pmatrix} \rho \\ \rho x^{nc} \\ \rho x_d^{\,nc} \\ \rho x^{chem}[i] \\ \rho v \\ \rho u \end{pmatrix};\quad f(\omega) = A\begin{pmatrix} \rho v \\ \rho v x^{nc} \\ \rho v x_d^{\,nc} - D\dfrac{\partial \rho x_d^{\,nc}}{\partial x} \\ \rho v x^{chem}[i] \\ \rho v^2 + P + qn \\ \rho v(u + P/\rho \end{pmatrix};\quad \Omega(\omega) = \begin{pmatrix} -\rho A\, k_{wall}(\partial P/\partial t) \\ -\rho x^{nc} A k_{wall}(\partial P/\partial t) + V(R_d - R_a) \\ -\rho x_d^{\,nc} A k_{wall}(\partial P/\partial t) + V(R_a - R_d) \\ -\rho x^{chem}[i] A k_{wall}(\partial P/\partial t) \\ -0.5(\Delta\xi/\Delta x)\rho v|v|A + \rho g A + P(dA/dx) \\ -\rho u A\, k_{wall}(\partial P/\partial t) + \Delta Q/\Delta x + \rho g v A \end{pmatrix}$$

where ρ, x^{nc}, x_d^{nc}, $x^{chem}[i]$, P, u are the gas/liquid mixture density, the non-condensable mass fraction, the non-condensable mass fraction diluted in the liquid phase, the chemicals mass fraction, the pressure and the total energy respectively. "*qn*" is the artificial dissipation term, see below. A is the variable flow area and v the velocity.

This system of equations represents the general case of a mixture of chemicals. The different source terms are the following:

- In the first equation governing the mixture mass conservation, a source term responsible for the wall compressibility effect of the mixture, determinant in water hammer simulations, is included
- In the second equation governing the non-condensable mass conservation (if any), a similar source term is included
- In the third equation governing the non-condensable mass conservation diluted in the liquid phase (if any), a similar source term is included
- In the fourth equation governing the chemicals mass conservation (if any), a similar source term is included
- In the fifth equation governing the mixture momentum conservation, a source term represents the friction ($\Delta\xi$, proportional to Δx, is the pressure drop coefficient derived by empirical correlations, see below), another one takes into account the gravity, and the last one is responsible for the area variation
- In the last equation governing the mixture energy conservation, a source term, ΔQ (proportional to Δx, see below), takes into account the heat transfer with the wall; another source term, ρgvA, takes into account the gravity work

The implementation of this set of equations must take into account various parameters, i.e. the geometry (flow area and *dx* can vary along the pipe, see the input data at the operational Pipe components), *the numerical scheme*, the boundary conditions, and the flow thermodynamics (state law, composition, etc.).

5.3.8.2 Source terms

- The equivalent distributed friction, $\Delta\xi(i)$ is calculated as follows:

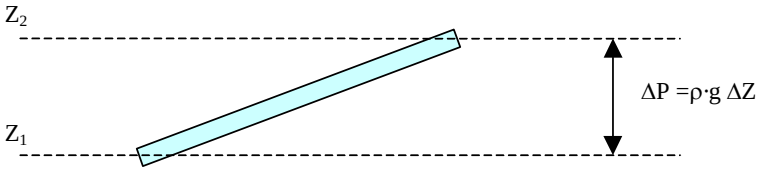
$$\Delta\xi_i = k\_add/n + \sum_{bend,j} hdc\_bend(\alpha_{bend,j}, R_{bend,j}, D_i, Rug)/n + \frac{\Delta x_i}{D_i}\cdot hdc\_fric(D_i, Rug, \mathrm{Re}_i)$$

K_add is an input data representing concentrated load losses to be distributed along the pipe.

Function "*hdc_bend*" calculates the bend pressure drop coefficient (see AppendixA2-1).

Function "*hdc_fric*" calculates the friction factor including laminar ($f \cong k/\mathrm{Re}$) and turbulent regimes (see AppendixA2-2).

- *g* represents the gravitational acceleration, if any. It is computed as the scalar product of the gravity vector (gx, gy, gz) with the direction of the pipe in the global axis system (Δx, Δy, Δz), where Δx, … are the difference of position of the tube tips.



- K_{wall} is the wall compressibility. Assuming linear elasticity for the pipe wall material:

Pipe anchored with expansion joints throughout $\kappa_{wall} = D_{in}/t/M_E$

Pipe anchored at its upstream end only $\kappa_{wall} = \frac{D_{in}}{tM_E}\cdot\left(\frac{5}{4}-\mu\right)$

Pipe fully anchored $\kappa_{wall} = \frac{D_{in}}{tM_E}\cdot\left(1-\mu^2\right)$

M_E is Young's modulus of elasticity, μ is Poisson's ratio and t is the wall thickness. The wall compressibility shall be multiplied by P' to account for the volume change. For this purpose P' *is calculated from the current state variables (density and energy) and the thermodynamic derivatives:*

$$P' = (\rho' - \partial\rho / \partial h_p * (u' + P\rho' / \rho^2)) / (\partial\rho / \partial P_h - \partial\rho / \partial h_p / \rho)$$

- The term ΔQ(i) gives the exchange of heat with the walls:

$$\Delta Q_i = h_{film,i} \Delta x_i \{P_{inner}(tp\_in.T(i) - T_i) + P_{outer}(tp\_out.T(i) - T_i)\},$$

where P_{inner} and P_{outer} are the wet perimeters; *tp_in, tp_out* are the names of the thermal ports connected to the tube with the same number of nodes as the fluid vein.

The port temperatures *tp_...T(i)* behave as the wall *internal* temperatures, to be determined in the connected THERMAL components.

The film coefficient is calculated by using empirical two-phase correlations; see Appendix A3:

$$h_{film,i} = htc\_sp(x_i, D_i, \lambda_i, \mathrm{Re}_i, \mathrm{Pr}_i, ...)$$

$$\mathrm{Re}_i = 0.5(m_{jun,i} + m_{jun,i+1}) D_i / (\mu_i \ A_i)$$

- The term qn(i) accounts for pressure losses due to turbulence terms not directly calculated with a 1D approach. This term numerically stabilizes the system of equations. It is calculated as follows:

$$qn(i) = -Damp \frac{m_{jun,i+1} - m_{jun,i}}{A} V_{sound,i}$$

Damp is a boundary to be defined in the experiment file. Values from 0.3 to 1 provide good smoothing of numerical pressure oscillations without hiding water-hammer pressure peaks.

- The absorption/desorption terms [RD-59 to RD-64] inside the pipe component are modeled adding one more advection diffusion-equation for the absorbed gas and modifying the existing one for the NCG. The main difference with respect to the standard NCG equation is the appearance of diffusive fluxes and source terms. Even though the diffusive fluxes are represented together with the usual convective counterpart, their discretization is substantially different. This is due to the feature of the related operator, which shows an elliptic nature. They can therefore be described by centred differences.

According to a Godunov's discretization the time variation of the conserved variable at cell i could be computed by corresponding fluxes at the face interfaces i+1/2 and i-1/2. Considering the diffusive flux as governed by Fick's law a single flux could be described by:

$$f_{abs,i-1/2} = D_{i-1/2} A_{i-1/2} (\partial \rho x_d^{nc} / \partial x)\big|_{i-1/2}$$

The description of the derivative of the density and the mass fraction could be described with a simple difference of the east and west cells. By calling ΔL_i the size of the *i*-th cell, the first the term on the right hand side of the previous equation is:

$$(\partial \Xi / \partial \eta)\big|_{i-1/2} = \frac{\frac{1}{2}(\Xi_i - \Xi_{i-1})}{\Delta\eta} = \frac{\frac{1}{2}(\Xi_i - \Xi_{i-1})}{\frac{1}{2}\left(\frac{\Delta x_{i-1}}{2} + \frac{\Delta x_i}{2}\right)} = \frac{\Xi_i - \Xi_{i-1}}{\frac{\Delta x_{i-1}}{2} + \frac{\Delta x_i}{2}},$$

The diffusion coefficient $D_{i-1/2}$ and the cross section area of the pipe $A_{i-1/2}$ in a specific cell are computed with a linear interpolation between the adjacent cell nodes:

$$D_{i-1/2} = D_{i-1} + \frac{D_i - D_{i-1}}{\frac{\Delta L_{i-1}}{2} + \frac{\Delta L_i}{2}} \left(\frac{\Delta x_{i-1}}{2}\right)$$

The diffusion coefficient is computed after the Stoke-Einstein relation: $D_{AB} = \frac{kT}{6\pi R_A \mu_B}$

where k is the Boltzmann constant, μ is the viscosity of the liquid and R is the radius of the gas molecule. This formula is an approximation, so if better data are available they should be used. An easy way to adjust the diffusion coefficient is to manipulate the diffusion turbulence factor that could account for both the coefficient inaccuracy and the absorption enhancement caused by turbulence.

The source terms for desorption and absorption, respectively R_d and R_a are written as

$$R_d = C_d \rho_g \left(P_{equil} - P_g\right)\left(1 - x^{nc}\right)x_d^{nc}$$

$$R_a = C_a \rho_g \left(P_g - P_{equil}\right)\left(x_{abs\,\lim} - x_d^{nc}\right)x^{nc}$$

where C_d and C_a are empirical coefficients (by default set as $C_d = 2.0$ and $C_a = 0.1$) and $x_{abs\,\lim}$ represents the maximum quantity of absorbed gas into the liquid. The equilibrium pressure P_{equil} is computed as $P_{equil} = P_{sat} + 0.39\rho_l k$ where k is the turbulence kinetic energy ($\left[m^2 s^{-2}\right]$) and P_{sat} is the saturation of the gas for that specific conditions. If the pressure differential $\left(P_{equil} - P_g\right)$ is positive thean R_d is active while $R_a = 0$, otherwise is R_a is active and $R_d = 0$.

The maximum quantity of absorbed gas represented by the term $x_{abs\,\lim}$ is computed by means of Henry's constant with the equation $x_{abs\,\lim} = K_{Henry} p_{NCG}$ (the pressure is considered in Pascal).

The Henry's coefficient is given in the form:

$$K_{Henry} = 10^{A/T+B} 10^{-11} / 1.013$$

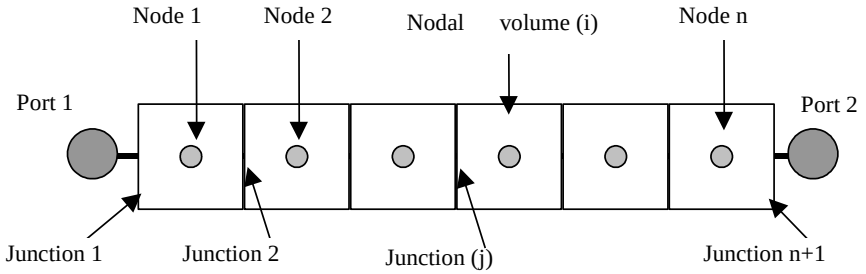
At the moment it is computed exactly only for the combination of between gases He, N_2, O_2, Ar, N_2O_3, and liquids N_2O_4, N_2H_4, MMH. The data are taken from RD-59.

5.3.8.3 Numerical schemes

In Appendices A9 and A10 two available numerical schemes are described: centred and upwind (Roe). Pressures and temperatures are associated with the n nodes. The dynamic mass flows are calculated either at the internal junctions (*centred scheme*, where each junction is associated with two half volume inertias), or at the nodes as for the other magnitudes (*upwind scheme*).

For the *ABS_tube* component, the mass flows at the first and last junctions (1 and n+1) will be given by the Junction type components connected to the pipe. Note that the first and last half-nodal inertias are included in the junction component equations (see §5.3.2).

For the *ABS_tube_res* component, the port mass flows will be calculated by the 1D set of equations making use of the P/H variables calculated at the capacitive components connected to the resistive Pipe.



5.3.8.4 Pressure, temperature and quality calculation

At each discretized volume, the non-derivative state variables (pressures, qualities and temperatures) will be calculated using the state functions. See §5.3.3.2. If a non-condensable gas plus a real fluid is used, the homogeneous equilibrium model will be also applied. See §4.5.3.1.

For the *ABS_tube* component, the sonic flow per unit of area $(\rho c)_{crit}$ is calculated at each pipe port as a function of the corresponding node pressure, density and the sound speed. These critical flows will be used at the connected junction type components to take into account sonic flow limitation.

5.3.9 ABS_Pipe, ABS_Pipe_res

These components simulate *area-varying non-uniform mesh 1D* pipes with one wall thermal node per fluid node. They are inherited from the "ABS_Tube" components, where the main formulation was established

These components have closed the thermal port inherited from the ABS_Tube component. In exchange, the following equation calculates the thermal wall temperatures:

$$T'_{wall,i} = \frac{-\Delta Q_i - h_{out} A_{wall,ext} (T_{wall,i} - T_{out})}{Cp_{wall} \cdot m_{wall,i}}$$

h_{out} and T_{out} are input data allowing the exchange of heat with the exterior. $\Delta Q(i)$, the internal wall exchange, was calculated by the "ABS_Tube" component.

5.3.10 Function donor_cell

This function belongs to the MATH Library. It determines an abrupt (but continuous) change in an upstream property depending on the mass flow direction:

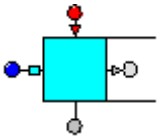
$$\text{Property} = 0.5\left[(1+c)\cdot Upstream\_prop + (1-c)\cdot Downstream\_prop\right]$$
$$c = m/(|m| + 1e-10)$$

5.4 OPERATIONAL COMPONENTS

5.4.1 Chamber

5.4.1.1 Description

Inherited from a "NonAdiabaticVol", this component represents a Chamber of an actuator.



5.4.1.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.1.3 Ports

Name	Type	Parameters	Direction	Description
f[nports]	Fluid	burnerGasesOption	IN	Inlet/Outlet fluid ports
tp_in	THERMAL	n=1	IN	Thermal port connected to the wall
m_out	MECHANICAL		OUT	Driving mechanical port
meas_out	vol_measure	See §5.2.2.2	OUT	Volume outlet signals

5.4.1.4 Data

Name	Type	Description	Units
A_pis	REAL	Piston area	m2
L	REAL	Volume length	m
Pw	REAL	Wet perimeter. (0, circular cross area)	m
Vo	REAL	Initial fluid volume	m^3
init_option	ENUM	Option to specify the initial thermodynamic state	
P_o	REAL See §5.1.3.3	Initial Pressure	Pa
T_o		Initial temperature	K
rho_o		Initial density	kg/m^3
x_o		Initial quality	-
x_nco		Initial non-condensable mass fraction	-
xd_nco		Initial gas mass fraction diluted in the liquid phase	-
side	INTEGER	Chamber side: 1 left; -1 right side	-
z_bottom	REAL	Bottom elevation relative to a z fixed axis	m
ht_option	ENUM	Wall-fluid heat transfer option (see A3)	
hc_dat	REAL	Heat transfer coefficient if ht_option = Ht_constant	W/m^2*K

The fluid volume (Vo) is the only geometric data that is always needed. If the other two geometric data are left as zero, the volume geometry is considered to be a sphere. See §5.4.23.4.

5.4.1.5 Formulation

Main conservation equations: See §5.3.6 (NonAdiabatic volume).

It is assumed that the volume variation owes to a piston displacement with a given area:

$$\frac{dVolume}{dt} = A_{piston} \cdot \left(\frac{dx}{dt}\right)_{piston}$$

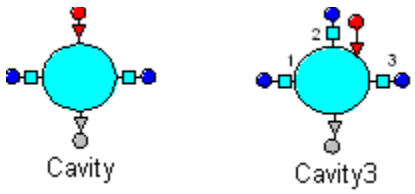
The volume rate of change will be considered by the general volume equations inherited in this component (see §*5.3.3.*) The velocity of the piston is obtained from the mechanical port and the force (pressure times the piston area) will be transmitted to that port.

5.4.2 Cavity, Cavity3

5.4.2.1 Description

Inherited from a "NonAdiabaticVol", these components represent *non adiabatic* Volumes with a given number of fluid ports (one or three) provided with a thermal port.

This fluid capacity accounts for geometrical volume changes under pressure variations due to wall compressibility.



5.4.2.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.2.3 Ports

Name	Type	Parameters	Direction	Description
f[nports]	Fluid	burnerGasesOption	IN	Inlet/Outlet fluid ports
tp_in	THERMAL	n=1	IN	Thermal port connected to the wall
meas_out	vol_measure	See §5.2.2.2	OUT	Volume outlet signals

5.4.2.4 Data

Name	Type	Description	Units
L	REAL	Volume length. L=0, a sphere. L<0, horizontal cylinder	m
Pw	REAL	Wet perimeter. (0, circular cross area)	m
Vo	REAL	Fluid volume	m^3
kappa_wall	REAL	Compressibility due to wall expansibility, dV_VdP	1/Pa
init_option	ENUM	Option to specify the initial thermodynamic state	
P_o	REAL See §5.1.3.3	Initial Pressure	Pa
T_o		Initial temperature	K
rho_o		Initial density	kg/m^3
x_o		Initial quality	-
x_nco		Initial non-condensable mass fraction	-
xd_nco		Initial gas mass fraction diluted in the liquid phase	
iside[10]	INTEGER	Port sides (1 inlet, 2 outlet)	
iangle[nports]	REAL	Port angles: 0, frontal inlet/outlet; 90, lateral inlet/outlet	deg
z_bottom	REAL	Bottom elevation relative to a z fixed axis	m
ht_option	ENUM	Wall-fluid heat transfer option (see A3)	
hc_dat	REAL	Heat transfer coefficient if ht_option = Ht_constant	W/m^2*K

The Fluid volume (Vo) is the only geometric data which is always needed. If the other two geometric data are left as zero, the volume geometry is considered to be a sphere. See §5.4.23.4.

If necessary, use *ht_option = HT_critCond* when passing near the critical point; the Prandtl number will then be assumed to be 1 (no dependence on cp value).

5.4.2.5 Formulation

Main conservation equations: See §5.3.6 (NonAdiabatic volume).

It is assumed that the volume variation is due to wall compressibility:

$$\frac{dV}{dt} = V\, k_{wall} P' \;;\; P' = (\rho' - \partial\rho/\partial h_p * (u' + P\rho'/\rho^2)) / (\partial\rho/\partial P_h - \partial\rho/\partial h_p/\rho)$$

P' is calculated from the current state variables (density and energy) and the thermodynamic derivatives.

5.4.3 DeadEnd

5.4.3.1 Description

This component represents a permanently closed orifice. It serves to close an open fluid port.

DeadEnd

5.4.3.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.3.3 Ports

Name	Type	Parameters	Direction	Description
f_out	fluid	burnerGasesOption	OUT	Closed fluid port

5.4.3.4 Data

Name	Type	Description	Units
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m

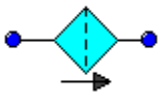
5.4.3.5 Formulation

This component sets the mass and enthalpy flows of the connected fluid port to zero.

5.4.4 Filter

5.4.4.1 Description

Inherited from an "AbstractJunction", this component represents a filter.



5.4.4.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.4.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet / Outlet fluid ports
f2	fluid	burnerGasesOption	OUT	

5.4.4.4 Data

Name	Type	Description	Units
fluid	ENUM	Fluid for which the reference conditions were calculated	-
P_ref	REAL	Pressure for reference conditions	Pa
T_ref	REAL	Temperature for reference conditions	K
dP_ref	REAL	Pressure loss at reference conditions	Pa
m_ref	REAL	Mass flow at reference conditions	kg/s
n	REAL	Exponent of the mass flow in pressure loss equation	
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m

5.4.4.5 Formulation

Mass and Energy Conservation Equations: *See* §*5.3.1.1.*

Momentum equation*:* $(I_1 + I_2)\cdot\frac{dm}{dt} = P_1 - P_2 - \Delta P_{REF}\left(\frac{m}{m_{REF}}\right)^n \frac{\rho_{REF}}{\rho}\left(\frac{\mu}{\mu_{REF}}\right)^{2-n}$

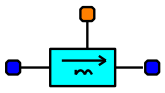
where,

n is the exponent of the mass flow in the pressure loss equation
I1, I2 = half inertia of the connected pipe ends 1 and 2, respectively
m = mass flow that circulates through the sensor
mref = reference mass flow
ΔPref = reference pressure loss
ρref, μref = reference fluid density and viscosity
P1, P2 = pressure at ports
ρ = fluid density

5.4.5 Jun_TMD

5.4.5.1 Description

Inherited from an "AbstractJunction", this component represents an orifice with imposed mass flow.



Jun_TMD

5.4.5.2 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	noBurnGases	OUT	Inlet / Outlet fluid ports
f2	fluid	noBurnGases	OUT	
s_massflow	CONTROL.analog_signal	(n = 1)	IN	Imposed mass flow (kg/s)

5.4.5.3 Data

Name	Type	Description	Units
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m
tao	REAL	Time constant of mass flow filter	s

5.4.5.4 Formulation

Mass and Energy Conservation Equations*: see* §*5.3.1.1.* Momentum Equations*:* the mass flow is imposed through a delay equation:

$$\frac{dm}{dt} = \frac{m_{controlled} - m}{tao}; \quad tao = \text{Time delay (s)}$$

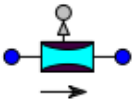
m = Actual mass flow

$m_{controlled}$ = Controlled mass flow = s_massflow.signal[1]

5.4.6 Junction

5.4.6.1 Description

Inherited from an "AbstractJunctionLoss", this component represents a concentrated load loss with constant throat area and sonic flow limitation.



5.4.6.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids
choked_option	BOOLEAN	FALSE, Supersonic conditions allowed (no choked limitation)

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

The *"choked_option" = TRUE* calculates supersonic conditions at a junction, for example simulating a supersonic intake. See §5.3.2.4.

5.4.6.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet / Outlet fluid ports
f2	fluid	burnerGasesOption	OUT	
meas_out	jun_measure	See §5.2.2.2	OUT	Junction outlet signals

5.4.6.4 Data

Name	Type	Description	Units
Ao	REAL	Junction area when fully open	m^2
m_o	REAL	Initial mass flow	Kg/s
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m
Gcr_ideal	REAL	TRUE, ideal critical flow expansion, FALSE, empirical correlation	-
zetab	REAL	Backward loss coefficient	-
zetaf	REAL	Forward loss coefficient	-
Re_lam	REAL	Laminar Reynolds number, see note 3	-

Note 1: *zetaf*, *zetab* = 0 means automatic calculation of the pressure drop coefficients from the orifice area and the adjacent connected flow areas.

Note 2: Gcr_ideal = TRUE means ideal calculation of the critical mass flow, see Appendix A1.2. This option is valid in case of only one fluid.Gcr_ideal = FALSE gives more realistic values, especially in flushing liquids.

Note 3: If the local Re > Re_lam, pressure losses are quadratic. A transition zone is considered near Re_lam, see §5.3.2.2. Re_lam value should be reduced to about 50 for sharp small orifices at low pressure (see Appendix A11).

5.4.6.5 Formulation

Mass and Energy Conservation Equations*: See §5.3.1.1.*

Momentum Equations*: See §5.3.2.1.*

The throat area is a constant input data: A = Ao. If zeta coefficients (input data) are null, the pressure drop coefficients will be automatically calculated from the orifice area and the adjacent connected flow areas [RD-45] assuming abrupt area change:

*zetaf = 0.02 + 0.5(1 - Ao/A1) + (1 - Ao/A2)**2*

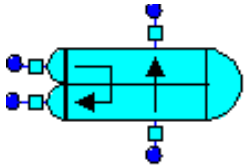
*zetab = 0.02 + 0.5(1 - At/A2) + (1 - Ao/A1)**2*

where A1, A2 are the flow area of the connected volumes transmitted by the inlet/outlet fluid ports.

5.4.7 HeatExchanger

5.4.7.1 Description

This component represents a 1D shell-and-tube heat exchanger working with two separated fluids, with or without phase change inside it. Parallel, Counter Flow or Cross Flow dispositions are modeled. *Heat exchanged, mass storage and pressure losses are physically modelled*, the conductance being calculated as a function of the geometry and the transient boundary conditions.



5.4.7.2 Construction Parameters

Name	Type	Description
burnerGasesOption_t	SET_OF(Chemicals)	Type of mixture of fluids
burnerGasesOption_s	SET_OF(Chemicals)	Type of mixture of fluids
type	ENUM	Heat exchange type (Counterflow, ParallelFlow or CrossFlow)
nt	CONST INTEGER	Number of shell-side passes (baffles+1) for crossFlow disposition or tubes nodes number for parallel/counter flow
n_pass	CONST INTEGER	Number of tube-side passes; for U-tubes, n_pass = 2

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.7.3 Ports

Name	Type	Parameters	Direction	Description
fs_in	fluid	burnerGasesOption_s	IN	Shell Inlet port
fs_out	fluid	burnerGasesOption_s	IN	Shell Outlet port
ft_in	fluid	burnerGasesOption_t	IN	Tube Inlet port
ft_out	fluid	burnerGasesOption_t	IN	Tube Outlet port

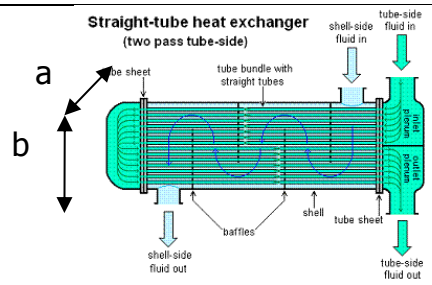
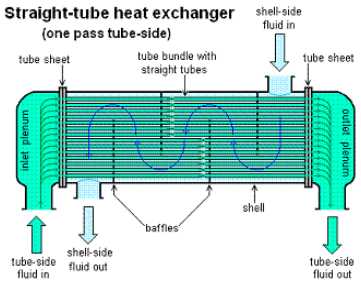
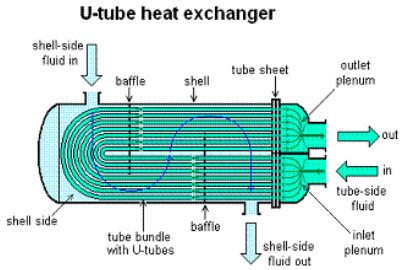
5.4.7.4 Data

Name	Type	Description	Units
n_ch	INTEGER	Number of parallel tubes	-
in_disp	ENUM	Disposition of Shell/Tube inlets.	-
L_t	REAL	Tube length per pass	m
D_t	REAL	Internal tube diameter	m
th_t	REAL	Tube wall thickness	m
rug	REAL	Tube rugosity	m
kf_t	REAL	Multiplier of the tube friction factor	
th_f	REAL	Fin thickness	m
A_f	REAL	Fin **total** exchange surface (both sides)	m2
a_sh	REAL	Shell width and height (See note 1)	m
b_sh	REAL		m
L_sh	REAL	Shell path length; 0, calculated	m
A_sh	REAL	Effective shell flow area - 0, automatic calculation	m^2
th_c	REAL	Case wall thickness	m
kf_sh	REAL	Multiplier of the shell friction factor	
fr_option	ENUM	Tubes-internal fluid friction option (See §A2)	
ht_option	ENUM	Tubes-internal fluid heat transfer option (see §A3)	
hc_dat	REAL	Tubes-internal fluid heat transfer coef.	W/m^2*K
P_o	Initial data	Initial Tube-pressure	Pa
P_sh_o		Initial Shell-pressure	Pa
T_o		Initial temperature	K
x_nco		Initial non-condensable mass fraction at tubes side	-
xd_nco		Initial gas mass fraction diluted in the liquid phase	-
x_sh_nco		Initial non-condensable mass fraction at shell side	-
T_outside	REAL	Environment temperature for the pipe	K
h_outside	REAL	Heat transfer coefficient with the environment	K
mat_c	ENUM	Case wall Material	-
rho_c	REAL	Case wall density if Case_mat=None	Kg/m3
cp_c	REAL	Case wall specific heat if Tube_mat=None	J/kg/K
mat_t	ENUM	Tube & Fin Material	-
rho_t	REAL	Tube & Fin wall density if Case_mat=None	Kg/m3
cp_t	REAL	Tube & Fin wall specific heat if Tube_mat=None	J/kg/K

Notes:

1. For parallel/counter flow disposition: "a_sh, b_sh" are the dimensions of the shell flow area. For crossFlow disposition, "a_sh, b_sh" are the dimensions of the shell perpendicular to the tubes, see figure below.
2. "A_sh" data has a great influence on the heat exchange efficacy. Give a known datum, if possible. Default calculated values are explained in the formulation paragraph, §5.4.7.6.
3. Wall materials are selected via the ENUMERATIVE variable "mat" in which a list of materials is provided (see §3.9).

The following table summarizes some typical cross flow dispositions and the associated parameters:

Type	in_disp	nt	N_pass
	same_corner	5	2
	baffle_opposed	5	1
	tube_opposed	3	2

5.4.7.5 Topology

The "HeatExch" component is built topologically with two fluid veins ("Tube_Annular" component, one for the tubes, one for the shell), and two arrays of Dnode THERMAL nodes with the same dimension as the number of tube/shell nodes: one simulating the fins and tube walls, and the other for the case.

The number of nodes to be calculated in any of the fluid or wall sides will be nt* n_pass. For example, a heat exchanger with two tube passes and two baffles forcing the fluid in the shell to cross the tubes three times will have in total 3*2 thermal nodes on the tube side, on the shell side and in the fins.

Heat fluxes between the fluid veins (tubes and shell) and the walls are calculated inside the "Tube_Annular" components through the calculation of the fluid temperatures, heat exchange areas, heat

exchange coefficients (depending on the Reynolds number and hence on the geometry) and on the fluid transport properties (see §5.3.8.2 and A3). Note that the tube wall and fin diffusive nodes will receive the heat fluxes of both fluid veins by the corresponding wetted areas.

5.4.7.6 Formulation

Diffusive node Equations:

The wall thermal nodes (case and tube wall including the fins) are simulated by the corresponding "Dnode" component of the THERMAL library, whose thermal capacities are calculated from the geometrical data (thicknesses, lengths and surfaces) and from the material properties:

$$C_{fins} = cp_{fins}\rho_{fins}(\text{th\_f} \cdot \text{A\_f}/2 + \text{th\_t} \cdot \pi \cdot \text{D\_t} \cdot \text{L\_t} \cdot \text{n\_ch} * \text{n\_pass})$$

$$C_{case} = cp_{shell}\rho_{shell}(\text{th\_c} \cdot \text{L\_sh} \cdot 2(\text{a\_sh} + \text{b\_sh}))$$

Fluid Vein Geometry:

Each fluid vein is simulated by the corresponding "Tube_Annular" component. The flow and wet areas on the tube side are multiplied by the number of tubes, so that only one tube component will simulate the tubes. The flow and wet areas on the shell side are calculated as follows (*the shell flow area and shell length will be calculated only if no input data is given for these variables*):

```
IF(type == CounterFlow) THEN
      L_sh = n_pass*L_t
      A_sh = a_sh*b_sh/n_pass - 0.25*PI*D_t**2*n_ch - th_f*A_f/2/L_sh
ELSEIF(type == ParallelFlow) THEN
      L_sh = n_pass*L_t
      A_sh = a_sh*b_sh/n_pass - 0.25*PI*D_t**2*n_ch - th_f*A_f/2/L_sh
ELSE -- CrossFlow
      L_sh = b_sh*nt + L_t*(nt-1)/nt
      A_sh = (a_sh - D_t*n_ch)*(L_t/nt) - th_f*A_f/2/L_sh
      A_sh=(a_sh-D_t*sqrt(n_ch*n_pass*a_sh/b_sh))*(L_t/nt)-
th_f*A_f/2/L_sh_eff
END IF

P_tu = (A_f + PI*(D_t+2*th_t)*L_t*n_ch*n_pass) /L_sh
P_ca = 2*(a_sh+b_sh)
```

where P_ca is the shell/case contact perimeter and P_tu is the shell/tube contact perimeter. The following assumptions have been made:

Shell flow area, A_sh:

1. For cross flow disposition, the obstacle area due to the tubes is proportional to the tube diameter, tube length and to a number of tubes assuming a regular tube disposition along the "a_sh * b_sh" section `(sqrt(n_ch*n_pass*a_sh/b_sh))`.
2. For parallel/counter flow disposition, the obstacle area due to the tubes is proportional to the tube section and to the number of tubes.
3. The obstacle area due to the fins is proportional to its thickness and to the equivalent fin height.

Shell path length, L_sh:

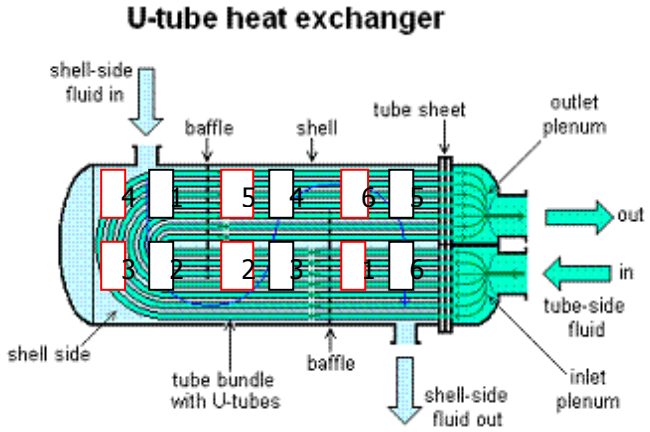
1. For cross flow disposition, it is proportional to shell height, b_sh, multiplied by the number of shell passes. An additional term is considered due to the bends between shell passes.
2. For parallel/counter flow disposition, the shell path length is proportional to the tube length and to the number of tube passes.

Shell wetted perimeters, P_ca, P_tu:

1. The fin/tube contact perimeter, P_tu, is calculated as the total exchange area divided by the shell path length.
2. The case contact perimeter, P_ca, is calculated from the characteristic shell dimensions.

Shell nodes array (n_sh) calculation in correspondence with the tube nodes:

The following scheme for a typical cross flow disposition summarizes the calculation of the correspondence matrix between the shell and tube fluid nodes depending on the inlet/outlet disposition:



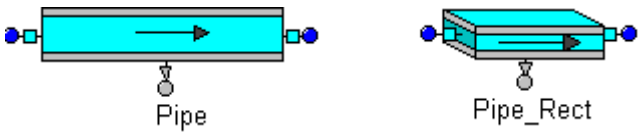
The tube node numbers (red ones, i) will be in contact with the shell node numbers (black ones, n_sh[i]). n_sh matrix is automatically calculated by the code. Once this matrix is calculated, the heat fluxes going to the diffusive nodes are:

```
FOR (i IN 1, nt*n_pass)
      IF(type == CounterFlow) THEN
            Fins_q[i] = Tubes_q[i] - Shell_q[nt-i+1]
      -- ParallelFlow
      ELSEIF(type == ParallelFlow) THEN
            Fins_q[i] = Tubes_q[i] - Shell_q[i]
      -- CrossFlow
      ELSE
            Fins_q[i] = Tubes_q[i] - Shell_q[n_sh[i]]
      END IF
END FOR
```

5.4.8 Pipe and Pipe_Rect

5.4.8.1 Description

Inherited from the "ABS_Pipe", these components simulate a cylindrical or rectangular *area-varying non-uniform mesh 1D* pipe with one wall thermal node per fluid node.



5.4.8.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids
n_bends	CONST INTEGER	Number of bends
nodes	CONST INTEGER	Number of control volumes
scheme	ENUM	Numerical scheme used
AbsorOption	ENUM	Activate/deactivate absorption -desorption option. *Activation can increase a lot the CPU time*

- Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.
- The numerical scheme can be: *centred*, *Roe_1st* or *Roe_hr_lim*. *Centred* is the default scheme, *Roe_1st* is the upwind Roe scheme, and *Roe_hr_lim* is the Roe scheme with a high resolution reconstruction with limiters. *The "Centred" scheme behaves more robustly and less time consuming than the Roe scheme but is less precise.*

5.4.8.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	IN	Inlet/Outlet fluid port
f2	fluid	burnerGasesOption	IN	Inlet/Outlet fluid port
meas_out	pipe_measure	See §5.2.2.2	OUT	Pipe outlet signals

5.4.8.4 Data

Name		Type	Description	Units
num		REAL	Number of parallel identical tubes *(See note 4)*	-
L		REAL	Pipe length	m
dx_vs_L		TABLE	Mesh size distribution vs non-dimensional axial position*(See note 1)*	-
D	For circular pipes	REAL	Nominal pipe inner diameter	m
D_vs_L	For circular pipes	TABLE	Non-dimensional diameters vs non-dimensional axial position	-
a	For rectangular pipes	REAL	Nominal cross section width	m
b	For rectangular pipes	REAL	Nominal cross section height	m
a_vs_L	For rectangular pipes	TABLE	Non-dimensional cross section width &height vs non-dimensional axial position	-
b_vs_L	For rectangular pipes	TABLE	Non-dimensional cross section width &height vs non-dimensional axial position	
SR_bend	For rectangular pipes	REAL[n_bend]	Side ratio of bend cross sections, a/b	-
R_bend		REAL[n_bend]	Ratio of curvature of bends	m
alpha_bend		REAL[n_bend]	Bend angles	degrees
rug		REAL	Roughness	m
fld_add *(See note 6)*		REAL	Pressure drop coefficient for distributed losses	-
fld_nod *(See note 6)*		REAL[nodes]	Pressure drop coefficient for *non* distributed losses	-
k_f		REAL	Multiplier of the friction factor	-
k_d		REAL	Multiplier of the numerical dissipation	-
init_option		ENUM	Option to specify the initial thermodynamic state	
P_o		REAL See §5.1.3.3	Initial pressure	Pa
T_o		REAL See §5.1.3.3	Initial temperature	K
rho_o		REAL See §5.1.3.3	Initial density	kg/m^3
x_o		REAL See §5.1.3.3	Initial quality	-
x_nco		REAL See §5.1.3.3	Initial non-condensable mass fraction	-
m_o		REAL	Initial mass flow	kg/s
anchor		ENUM	Pipeline anchor type	
mat *(See note 2)*		ENUM	Material	-
E_wall		REAL	Wall Elasticity Modulus , if mat=None	Pa
v_wall		REAL	Wall Poisson Modulus , if mat=None	-
cp_wall		REAL	Wall specific heat , if mat=None	J/kg/K
rho_wall		REAL	Wall density, if mat=None	Kg/m3
e_wall		REAL	Wall thickness	m
fr_option *(See note 7)*		ENUM	Wall-internal fluid friction option (see §A2)	
ht_option		ENUM	Wall-fluid heat transfer option (see §A3)	
hc_dat		REAL	Heat transfer coefficient if ht_option = Ht_constant	W/m^2*K
h_outside *(See note 3)*		REAL	Heat transfer coefficient with the environment	K

Name	Type	Description	Units
T_outside	REAL	Ambient temperature for the pipe	K
UserDefSolubData	Absorption/ desorption data	TRUE : user input data	
A_coef_sol		Binary solubility coefficients if UserDefSolubData = TRUE	
B_coef_sol			
TI		Turbulence intensity, free stream speed	-
Ca		Multiplier of absorption mass flow	
Cd		Multiplier of desorption mass flow	
Diff_Turb_Factor		Factor weighting diffusion of diluted gases	-
xd_nco		Initial gas mass fraction diluted in the liquid phase	-
entropy_fix	Numerical ROE scheme parameters, see Note 7	Entropy fix used	
entropy_fix_multiplier		Multiplier in the entropy fix	-
integration_rule		Numerical integration rule for pressure derivatives	
dp_correction		Correction on pressure derivative linearizations	
limiter		Limiter used	
preconditioner		Preconditioner used	
reconstructed_variables		Reconstructed variables	
central_reconstruction		Reconstruction on central part of numerical flux	
source_upwind_smoothng		Source term upwinding smoothing parameter	m/s
Isent_Correl		If TRUE, supersonic flow is allowed. *In normal cases put FALSE to avoid high CPU consumption (implicit equations)*.	

Note 1: To facilitate the data insertion of diameters and grid sizes as a function of the axial position, the following tables were defined at the input data interface:

- "D_vs_L": Non-dimensional diameters vs. normalized axial position. "Non-dimensional" means that the "y" values of the diameters table are divided by the nominal diameter. "x" values are axial lengths divided by the total length, *and must be between 0 and 1*
- "dx_vs_L": Mesh size distribution vs. normalized axial position. "x" values are axial lengths divided by the total length, *and must be between 0 and 1.* "y" values are the **relative** mesh size depending on the axial position (x): The lower y value the lower mesh size. For example, a "y" value of 0.5 at x=0 and a "y" value of 2 at x=1 means that meshes at inlet (x=0) are four times smaller than at the exit. The same effect is obtained if the "y" value is 1 at x=0 and 4 at x=1.

  Tables "D_vs_L" and "dx_vs_L" will not be affected by a change in the number of nodes, in the length of the pipe or in the nominal diameter (scalars). Default values are: {{0,1},{1,1}}, so constant diameter and uniform grid size distribution will be used. For rectangular pipes, the same criteria than in D_vs_L table will apply for the *a_vs_L, b_vs_L* tables.

Note 2: Wall materials are selected via the ENUMERATIVE variable "mat" in which a list of materials is provided (see §3.9).

Note 3: *h_outside, T_outside* are simple constant parameters to simulate the heat exchange with the exterior. For other more complex exchanges, use the "*Tube*" components.

Note 4: The "*num*" data serves to simulate several identical parallel tubes with one only component.

Note 5: "*fld_add" (*a pressure drop coef. multiplying the dynamic pressure) accounts for additional pressure losses different than friction or elbows (bends), or for elbow losses not included in the elbow data. Use "*fld_nod[i]*" to define particular local node pressure losses.

Note 6: If *fr_option*= FR_tube, then two phase correlations will be used calculating the wall friction coefficient (this can lead to discontinuities passing from sub critical to supercritical conditions. If *fr_option*= FR_tube_1ph, only one phase correlations will be used considering the mean mixture values of the fluid properties.

Note 7: Using the *Roe* scheme the user must define the values of the respective numerical parameters, the default values normally working well. The experienced user is free to hardcode himself some other values for these numerical parameters according to the following considerations:

- Although implemented, the preconditioning *preconditioner* = *turkel* is not always efficient as it should be coupled with some time integration restrictions (explicit/implicit, CFL); the alternative to this preconditioning [13], set by *preconditioner* = *altern*, has not been fully studied yet.
- Many tests on gas and liquid flows showed the VanAlbada *limiter* was the best option with regard to precision and CPU time for gas flows, and similarly VanLeer, for liquid flows. Anyway, some limiters can introduce discontinuities in the system of equations and considerably slow down the computation.
- *central_reconstruction* is always set to TRUE, as many EcosimPro tests have shown the importance to the result precision of reconstructing not only the upwind part of the flux, but also the central part. However, reconstructed values can sometimes lead to unphysical extrapolated properties;
- *reconstructed_variables* (var_rec) is always set to *primitive*, as a full study of the influence of the var_rec choice on the results is not yet performed, and var_rec = prim_rec gives correct results;
- For the test cases considered thus far, setting *integration_rule* = *midpoint* seems to be a good choice as far as precision and quality of the results are concerned, as it is the faster approach and the difference in the results is barely noticeable;
- The influence of *dp_correction* was only studied in some cases, *dp_correction* =TRUE is safe from incorrect behaviors. More time should be dedicated to a complete study on this parameter;
- The *entropy_fix* = *entropy_pres_max* and *entropy_fix_multiplier* =4 is generally the best option. Anyway, you can hardcode another value to meet personal requirements (comparison with other codes, etc.).
- *source_upwind_smoothing* (eps_lin)=0 may be necessary to avoid unphysical behavior due to the discontinuities handling of DASSL. This happens mainly with liquid flows.
- *Isent_Correl*. If TRUE, supersonic flow is then allowed. In normal cases use FALSE to avoid high CPU consumption (implicit equations calculating isentropic expansions).

Several application test cases have been studied. The table below shows some recommended values:

- Single fluid gas flow in a pipe. (case I);
- Two fluid gas flow in a pipe. (case II);
- Gas flow in a nozzle. (case III);
- Liquid flow in a pipe. (case IV);
- Liquid flow in a nozzle. (case V).

Case	I	II	III	IV	V
precond	*unprecond*	*unprecond*	*unprecond*	*unprecond*	*unprecond*
lim	*VanAlbada*	*VanAlbada*	*VanAlbada*	*VanLeer*	*VanLeer*
central	TRUE	TRUE	TRUE	TRUE	TRUE
var_rec	*prim_rec*	*prim_rec*	*prim_rec*	*prim_rec*	*prim_rec*
rule	*midpoint*	*midpoint*	*midpoint*	*midpoint*	*midpoint*
dp_cor	TRUE	FALSE	TRUE	FALSE	FALSE
fix	*entropy_pres_max*	*no_fix*	*entropy_pres_max*	*no_fix*	*no_fix*
mult	4	4	4	4	4
eps_lin	0	0	20	20	20

These application test-cases from I to V have been kept very general, and therefore, you cannot always benefit from all the numerical developments.

5.4.8.5 Formulation

This component incorporates the mass, energy and momentum equations in transient operating conditions according to an *area-varying non-uniform mesh high resolved* 1D spatial discretization. *See*

§5.3.8 and *§5.3.9.*The hydraulic diameters, wet areas and the mesh size for each node are interpolated along the axial position from the non-dimensional geometry tables:

$$x_i = x_{i-1} + \Delta x_{mean} Interp(x_{i-1/2} / L, \quad dx\_vs\_L)$$
$$D_i = D\ Interp(x_{i-1/2} / L, D\_vs\_L); \quad A_{wet,i} = \pi D_i\ dx_i$$

"*Interp*" is an interpolation function. First argument is the x entry value. The second argument is the table to be interpolated. "Δx_{mean}" is calculated iterating in the condition:

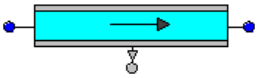
$$L = \Delta x_{mean} \sum Interp(x_{i-1/2} / L, \quad dx\_vs\_L)$$

For a rectangular pipe, the hydraulic diameters and wet areas are similarly calculated but using the interpolated ai, bi values as follows:

$$D_i = 2a_i b_i / (a_i + b_i); \quad A_{wet,i} = 2(a_i + b_i)\Delta x_i$$

5.4.9 Component Pipe_res

Like the Pipe, this component simulate a cylindrical *area-varying non-uniform mesh 1D* pipe with one wall thermal node per fluid node. The difference is that now the fluid ports are resistive, i.e., calculating mass flow. It has the same interface (data & ports) as in the Pipe component.

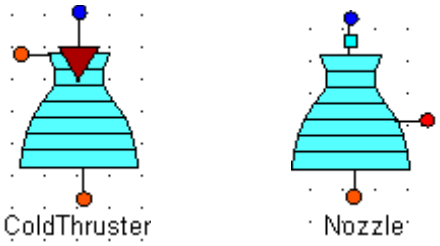


The Pipe_res can be connected directly to capacitive components (volumes or capacitive Pipes) instead of to a resistive valve or junction. The input data, ports and parameters are the same as in the Pipe component. The only difference is that the fluid ports are OUT type.

5.4.10 ColdThruster

5.4.10.1 Description

The Nozzle and the ColdThruster components simulate a cylindrical *area-varying non-uniform mesh 1D* nozzle with one wall thermal node per fluid node. Transitions from subsonic/supersonic (throat) and from supersonic to subsonic (shock wave in a non-adapted nozzle) are included as well.



5.4.10.2 Ports, Parameters &Data

The input data interface is as in the Pipe component.

5.4.10.3 Topology

- The Nozzle component has a capacitive inlet port that is internally connected to a variable area Pipe type component.
- The ColdThruster component has a resistive inlet port that is internally connected to a Valve type component, and this Valve to a variable area Pipe type component as before. Then, the difference is that the ColdThruster can control the inlet area.

In both cases, the outlet fluid port of the Pipe is connected to a Junction type component with *no* sonic flow limitation. See§5.3.2.4. The outlet of this Junction is mathematically closed by assigning the outlet

pressure signal (control variable) to the fluid port pressure. The outlet temperature is assumed to be constant and equal to the initial temperature.

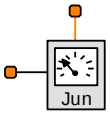
5.4.10.4 Formulation

This component directly uses the formulation of the Junction & Pipe components from which it has been topologically built. *See §5.3.8*and *§5.3.9*. Both Centred and Roe schemes are able to capture a shock wave in the divergent part of the nozzle under non-adapted conditions. The "Centred" scheme behaves more robustly than the Roe scheme but is less precise.

5.4.11 SensorJun

5.4.11.1 Description

This component represents a Sensor that measures Junction variables. Thus, the measures can be used as an input to a control block.



SensorJun

5.4.11.2 Ports

Name	Type	Parameters	Direction	Description
meas_in	jun_measure	(n = 2)	IN	Junction signals
s_out	CONTROL.analog_signal	(n = 1)	OUT	Outlet signal ()

5.4.11.3 Data

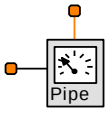
Name	Type	Description	Units
Measure	ENUM Jun_Measure	Required measure type	m
bias	REAL	Bias	-
gain	REAL	Gain	-

The value of the required measure, taken from the "jun_measure" port, is transmitted to the "s_out" CONTROL port. Possible measures are: {MassFlow, EnthalpyFlow}.

5.4.12 SensorPipe

5.4.12.1 Description

This component represents a Sensor that measures Pipe node variables. In this way the measures can be used as an input to a control block.



SensorPipe

5.4.12.2 Construction Parameters

Name	Type	Description
in	CONST INTEGER	Number of Pipe node where the Measure is taken
nodes	CONST INTEGER	Number of control volumes of the connected pipe

5.4.12.3 Ports

Name	Type	Parameters	Direction	Description
meas_in	pipe_measure	(n = 3*nodes)	IN	Pipe signals
s_out	CONTROL.analog_signal	(n = n_out=1)	OUT	Outlet signal ()

5.4.12.4 Data

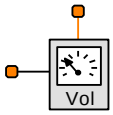
Name	Type	Description	Units
Measure	ENUM Pipe_Measure	Required measure type	m
bias	REAL	Bias	-
gain	REAL	Gain	-

The value of the required measure, taken from the "pipe_measure" port is transmitted to the "s_out" CONTROL port. Possible measures are: {NodePressure, NodeTemperature, and JunMassFlow}.

5.4.13 SensorVol

5.4.13.1 Description

This component represents a Sensor that measures Volume variables. This way the measures can be used as an input to a control block.



SensorVol

5.4.13.2 Ports

Name	Type	Parameters	Direction	Description
meas_in	vol_measure	(n = 14)	IN	Volume signals
s_out	CONTROL.analog_signal	(n = n_out=1	OUT	Outlet signal ()

5.4.13.3 Data

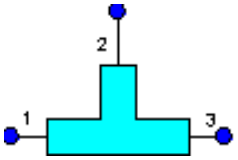
Name	Type	Description	Units
Measure	ENUM Vol_Measure	Required measure type	m
bias	REAL	Bias	-
gain	REAL	Gain	-

The value of the required measure, taken from the "vol_measure" port is transmitted to the "s_out" CONTROL port. Possible measures are: {Pressure, Temperature, TotalMass, LiquidHeight, etc}.

5.4.14 Tee

5.4.14.1 Description

This component represents an adiabatic Tee to simulate a joint or a split of two branches, including orifice restrictions at each port.



5.4.14.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.14.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet / Outlet fluid ports
f2		burnerGasesOption	OUT	
f3		burnerGasesOption	OUT	

5.4.14.4 Data

Name	Type	Description	Units
D1	REAL	Diameter of run connection 1	m
D2	REAL	Diameter of branch connection 2	m
D3	REAL	Diameter of run connection 3	m
Gcr_ideal	REAL	TRUE, ideal critical flow expansion, FALSE, empirical correlation	-
init_option	ENUM	Option to specify the initial thermodynamic state	
P_o	REAL See §5.1.3.3	Initial Pressure	Pa
T_o		Initial temperature	K
rho_o		Initial density	kg/m^3
x_o		Initial quality	-
x_nco		Initial non-condensable mass fraction	-
xd_nco		Initial gas mass fraction diluted in the liquid phase	
x	REAL	X coordinate relative to a body axis system	m
y	REAL	Y coordinate relative to a body axis system	m
z	REAL	Elevation relative to a body axis system	m
zeta1	REAL	Loss coefficient of run connection 1	-
zeta2	REAL	Loss coefficient of branch connection 2	-
zeta3	REAL	Loss coefficient of run connection 3	-
Re_lam	REAL	Laminar Reynolds number, for mass flow calc. (see App. A11)	-
Tee_type	ENUM	right_Tee, Y_Tee or UsrDef_Tee	
Vo	REAL	Tee fluid volume for UsrDef_Tees	m^3
L	REAL	Tee characteristic length for UsrDef_Tees	m
iangle[3]	REAL	Port angles for UsrDef_Tees, see note 2	degrees

Note1: Not all the initialization inputs are necessary. See §5.1.3.3 about possible initializations.

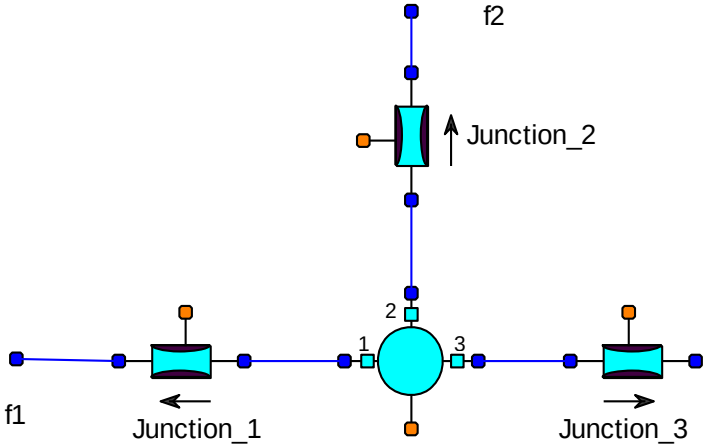
Note2: Put iangle=90" for lateral branches in case of UsrDef Tees.

5.4.14.5 Formulation

The Tee component is equivalent to a Volume3 with 3 Junctions. The volume is assumed to be adiabatic. The calculation of the geometrical volume and equivalent length depends on the Tee option:

```
IF(Tee_type == right_Tee) THEN
    Loo = D1 + D2 + D3
    Voo = PI / 8 * (D1 ** 2 + D3 ** 2) * (D1 + D2 + D3)
ELSEIF(Tee_type == Y_Tee) THEN
    Loo = 0.5*(D1**3+D2**3+D3**3) / (D1**2+D2**2+D3**2)
    Voo = PI/4 *(D1**3+D2**3+D3**3)/2
ELSE
    Loo = L  -- input data
    Voo = Vo-- input data
```

END IF

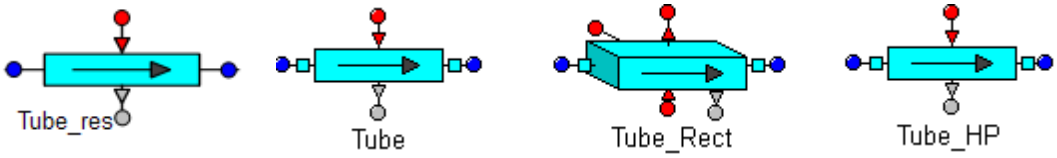


Volume

5.4.15 Tube, Tube_res, Tube_ Rect and Tube_HP

5.4.15.1 Description

Inherited from the "ABS_Tube"Component, these components simulate a cylindrical, rectangular or given (Tube_HP) *area-varying non-uniform mesh 1D* fluid vein that exchanges heat with a thermal port. The tube wall should be simulated by a THERMAL component like a "Cylinder" or a "Dnode" component.



For the *Tube_HP* component, the cross-section flow area and the wet perimeter are constant input data. The *Tube_res*s can be connected directly to capacitive components (volumes or capacitive Pipes) instead of to a resistive valve or junction.

5.4.15.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids
n_bends	CONST INTEGER	Number of bends
nodes	CONST INTEGER	Number of control volumes
scheme	ENUM Scheme	Numerical scheme used
AbsorOption	ENUM	Activate/deactivate absorption -desorption option.

Notes:

- Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.
- The numerical scheme can be: *centred*, *Roe_1st* or *Roe_hr_lim*. *Centred* is the default scheme, *Roe_1st* is the upwind Roe scheme, and *Roe_hr_lim* is the Roe scheme with a high resolution reconstruction with limiters. *The "Centred" scheme behaves more robustly and less time consuming than the Roe scheme but is less precise.*

5.4.15.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	IN	Inlet / outlet fluid ports
f2	fluid	burnerGasesOption	IN	

Name	Type	Parameters	Direction	Description	
tp_in	THERMAL.thermal	(n = nodes)	IN	Thermal port	
tp_lat	THERMAL.thermal	(n = nodes)	IN	Lateral thermal port	Only for rectangular pipes
tp_out	THERMAL.thermal	(n = nodes)	OUT	Outlet thermal port	
meas_out	pipe_measure	See §5.2.2.2	OUT	Pipe outlet signals	

For a rectangular Tube, the ports "tp_in, tp_lat, tp_out" are connected to the inner, lateral and outer perimeters respectively. *The lateral thermal port includes both sides.* Cylindrical tubes have only one thermal port, "tp_in".

5.4.15.4 Data

Name		Type	Description	Units
num		REAL	Number of parallel identical tubes *(See note 2)*	-
L		REAL	Pipe length	m
dx_vs_L		TABLE	Mesh size distribution vs non-dimensional axial position*(See note 1)*	-
D	Circular pipes	REAL	Nominal pipe inner diameter	m
D_vs_L	Circular pipes	TABLE	Non-dimensional diameters vs non-dimensional axial position	-
A_cr	For HP pipes	REAL	Cross-section flow area	m2
P_wet	For HP pipes	TABLE	Inner wet perimeter	m
a	Rectangular pipes	REAL	Nominal cross section width	M
b	Rectangular pipes	REAL	Nominal cross section height	m
a_vs_L	Rectangular pipes	TABLE	Non-dimensional cross section width &height vs non-dimensional axial position	-
b_vs_L	Rectangular pipes	TABLE	Non-dimensional cross section width &height vs non-dimensional axial position	
SR_bend	Rectangular pipes	REAL [n_bends]	Side ratio of bend cross sections, a/b	-
R_bend		REAL [n_bends]	Ratio of curvature of bends	m
alpha_bend		REAL [n_bends]	Bend angles (0, no bend)	degrees
rug		REAL	Roughness	m
fld_add *(See note 4)*		REAL	Pressure drop coefficient for additional losses	-
fld_nod *(See note 4)*		REAL[nodes]	Pressure drop coef. for *non*-distributed losses	-
k_f		REAL	Multiplier of the friction factor	-
k_d		REAL	Multiplier of the numerical dissipation	-
init_option		ENUM	Option to specify the initial thermodynamic state.	
P_o		REAL See §5.1.3.3	Initial Pressure	Pa
T_o		REAL See §5.1.3.3	Initial temperature	K
rho_o		REAL See §5.1.3.3	Initial density	kg/m^3
x_o		REAL See §5.1.3.3	Initial quality	-
x_nco		REAL See §5.1.3.3	Initial non-condensable mass fraction	-
m_o		REAL	Initial mass flow	kg/s
fr_option *(See note 5)*		ENUM	Wall-internal fluid friction option (See §A2)	
ht_option		ENUM	Wall-fluid heat transfer option (see §A3)	
hc_dat		REAL	Heat transfer coef. if ht_option = constant	W/m^2*K
UserDefSolubData		Absorption/ desorption data	TRUE : user input data	
A_coef_sol		Absorption/ desorption data	Binary solubility coefficients if UserDefSolubData = TRUE	
B_coef_sol		Absorption/ desorption data	Binary solubility coefficients if UserDefSolubData = TRUE	
TI		Absorption/ desorption data	Turbulence intensity, free stream speed	
Ca		Absorption/ desorption data	Multiplier of absorption mass flow	
Cd		Absorption/ desorption data	Multiplier of desorption mass flow	
Diff_Turb_Factor		Absorption/ desorption data	Factor weighting diffusion of diluted gases	-
xd_nco		Absorption/ desorption data	Initial gas mass fraction diluted in the liquid	-
entropy_fix		Numerical ROE scheme	Entropy fix used	
entropy_fix_multiplier		Numerical ROE scheme	Multiplier in the entropy fix	-

Name	Type	Description	Units
integration_rule	parameters, see *§5.4.8.4*	Numerical rule for pressure derivatives	
dp_correction		Correction on pressure derivative linearizations	
limiter		Limiter used	
preconditioner		Preconditioner used	
reconstructed_variables		Reconstructed variables	
central_reconstruction		Reconstruction on central part of numerical flux	
source_upwind_smoothing		Source term upwinding smoothing parameter	m/s
Isent_Correl		If TRUE, supersonic flow is allowed. *In normal cases put FALSE to avoid high CPU consumption*	

Note 1: To facilitate the data insertion of diameters and grid sizes as a function of the axial position, the following tables were defined at the input data interface:

- "D_vs_L": Non-dimensional diameters vs. normalized axial position. "Non-dimensional" means that the "y" values of the diameters table are divided by the nominal diameter. "x" values are axial lengths divided by the total length, *and must be between 0 and 1.*
- "dx_vs_L": Mesh size distribution vs. normalized axial position. "x" values are axial lengths divided by the total length, *and must be between 0 and 1.* "y" values are the **relative** mesh size depending on the axial position (x): the lower the y value, the lower the mesh size. For example, a "y" value of 0.5 at x=0 and a "y" value of 2 at x=1 means that the mesh at inlet (x=0) are four times smaller than at the exit (x=1). The same effect is obtained if the "y" value is 1 at x=0 and 4 at x=1.

  Tables "D_vs_L" and "dx_vs_L" will not be affected by a change in the number of nodes, in the length of the pipe or in the nominal diameter (scalars). Default values are: {{0,1},{1,1}}, so constant diameter and uniform grid size distribution will be used. For rectangular pipes, the same criteria as in D_vs_L table will apply for the *a_vs_L, b_vs_L* tables.

Note 2: The "*num*" data serves to simulate several identical parallel tubes with only one component.

Note 3: "*fld_add" (*a pressure drop coef. multiplying the dynamic pressure) accounts for additional pressure losses different than friction or elbows (bends), or for elbow losses not included in the elbow data. Use "*fld_nod[i]*" to define particular local node pressure losses.

Note 4: If *fr_option* = FR_tube, then two-phase correlations will be used calculating the wall friction coefficient (this can lead to discontinuities passing from subcritical to supercritical conditions. If *fr_option*= FR_tube_1ph, only one phase correlation will be used considering the mean mixture values of the fluid properties.

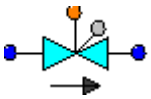
5.4.15.5 Formulation

This component incorporates the mass, energy and momentum equations in transient conditions according to an *area-varying non-uniform mesh high resolved* 1D spatial discretization. *See §5.3.8.* The hydraulic diameters, wet areas and the mesh size for each node are interpolated along the axial position from the non-dimensional geometry tables. See the Pipe component formulation.

5.4.16 Valve

5.4.16.1 Description

Inherited from an "AbstractJunctionLoss", this component represents an orifice with variable throat area.



5.4.16.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.16.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet / outlet fluid ports
f2	fluid	burnerGasesOption	OUT	
meas_out	jun_measure	See §5.2.2.2	OUT	Junction outlet signals
s_pos	CONTROL.analog_signal	(n=1)	IN	Valve position

5.4.16.4 Data

Name	Type	Description	Units
Ao	REAL	Junction area when fully open	m^2
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m
tao	REAL	Time constant of area change filter	s
zetab	REAL	Backward loss coefficient	-
zetaf	REAL	Forward loss coefficient	-
valve_char	ENUM	Valve type (EqualPercentage, Linear, QuickOpening or Userdefined (See note 3)	
zeta_vs_pos	TABLE	Pressure drop coef vs. non dimensional stroke position	
Arel_vs_pos	TABLE	Non dimensional valve flow area vs. non dimensional stroke position	
Gcr_ideal	REAL	TRUE, ideal critical flow expansion, FALSE, empirical correlation	-
Re_lam	REAL	Laminar Reynolds number (see note 4)	-
m_o	REAL	Initial mass flow	Kg/s

Note 1: *zetaf*, *zetab* = 0 means automatic calculation of the pressure drop coefficients from the orifice area and the adjacent connected flow areas. See §5.4.6.5.

Note 2: Gcr_ideal = TRUE means ideal calculation of the critical mass flow, see Appendix A1.2. Gcr_ideal = FALSE gives more realistic values, especially in flushing liquids.

Note 3: valve_char account for different valve types: EqualPercentage, Linear, QuickOpening or Userdefined, see next paragraph.

Note 4: If the local Re > Re_lam, pressure losses are quadratic. A transition zone is considered near Re_lam, see §5.3.2.2. Re_lam value should be reduced to about 50 for sharp small orifices.

5.4.16.5 Formulation

Mass and Energy Conservation Equations*:* See §*5.3.1.1.*

Momentum Equations*:* the derivative of valve area (see §*5.3.2.1*) is calculated as follows:

The value of the non-dimensional valve position (pos_com) will be an inlet through the control port to be specified in the experiment or connecting a CONTROL library component to the Valve control port. Then, depending on the valve type, the commanded position will be transformed accordingly:

```
pos_mod =    ZONE(valve_char == EqualPercentage) pos_com**3
             ZONE(valve_char == Linear) pos_com
             ZONE(valve_char == Userdefined) linearInterp1D(Arel_vs_pos, pos_com)
             OTHERS (1 - exp(-10*pos_com))
```

The valve actuator is modeled with a simple delay in the commanded position:

$$\frac{d(pos)}{dt} = \frac{pos\_\mathrm{mod} - pos}{tao}$$

Then, the derivative of valve area closing the formulation of the "`AbstractJunctionLoss`" component (see §*5.3.2.1*) is calculated as follows:

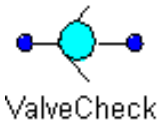
$$A = \mathrm{Ao}\ \ \mathrm{pos}; \quad \frac{dA}{dt} = A_0 \frac{d(pos)}{dt}$$

If the valve is User-defined, the pressure drop will be a function of the valve position (zetaf, zetab will not be used). The actual zeta value will be interpolated.

5.4.17 ValveCheck

5.4.17.1 Description

This component represents a non-return Valve. It is inherited from the "Valve" component.



5.4.17.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.17.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet fluid port
f2	fluid	burnerGasesOption	OUT	Outlet (back) fluid port

5.4.17.4 Data

Name	Type	Description	Units
Ao	REAL	Junction area when fully open	m^2
dpmin	REAL	DP for complete opening	Pa
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m
tao	REAL	Time constant of area change filter	s
zetaf	REAL	Forward loss coefficient	-
valve_char	ENUM	Valve type (EqualPercentage, Linear, QuickOpening or Userdefined (See note 3)	
zeta_vs_pos	TABLE	Pressure drop coef vs. non dimensional stroke position	
Arel_vs_pos	TABLE	Non dimensional valve flow area vs. non dimensional stroke position	
Gcr_ideal	REAL	TRUE, ideal critical flow expansion, FALSE, empirical correlation	-
Re_lam	REAL	Laminar Reynolds number	-
m_o	REAL	Initial mass flow	kg/s

Note 1: *zetaf* , *zetab* = 0 means automatic calculation of the pressure drop coefficients from the orifice area and the adjacent connected flow areas. See §5.4.6.5.

Note 2: Gcr_ideal = TRUE means ideal calculation of the critical mass flow, see Appendix A1.2. Gcr_ideal = FALSE gives more realistic values, especially in flushing liquids.

5.4.17.5 Formulation

Mass Momentum and Energy Conservation Equations*:* same as in the Valve component. The value of the signal controlling the valve area is a function of the pressure difference:

$$s\_pos.signal = \begin{cases} 1 & (if\ \ P_{up} > P_{back} + \Delta P\min) \\ \dfrac{P_{up} - P_{back}}{\Delta P\min} & (if\ \ P_{up} > P_{back}) \\ 0 & (others,\ no\ back\ \ flow) \end{cases}$$

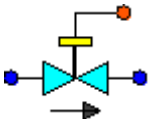
where Pup, Pdown are the inlet outlet port pressures. ΔPmin is an input data.

Once the commanded position is known, the Valve formulation is applied.

5.4.18 ValvePressRegDown

5.4.18.1 Description

This component represents a pressure regulator controlling the flow area as a function of the downstream pressure. It is inherited from the "`AbstractJunctionLoss`" component.



5.4.18.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.18.3 Ports

Name	Type	Parameters	Direction	Description
f1	Fluid	burnerGasesOption	OUT	Inlet / outlet fluid ports
f2	Fluid	burnerGasesOption	OUT	
s_pres	Analog_signal	1	IN	Pressure signal

5.4.18.4 Data

Name	Type	Description	Units
Ao	REAL	Junction area when fully open	m^2
P_close	REAL	Minimum downstream pressure required to open the valve	Pa
P_open	REAL	Downstream pressure that totally closes the valve	Pa
Re_lam	REAL	Laminar Reynolds number	-
Kg_vs_pos	TABLE_1D	Av correction vs non-dimensional stroke position	-
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m
tao	REAL	Time constant of area change filter	s
zetab	REAL	Backward loss coefficient	-
zetaf	REAL	Forward loss coefficient	-
Gcr_ideal	REAL	TRUE, ideal critical flow expansion, FALSE, empirical correlation	-
m_o	REAL	Initial mass flow	Kg/s

Note 1: *zetaf*, *zetab* = 0 means automatic calculation of the pressure drop coefficients from the orifice area and the adjacent connected flow areas. See §5.4.6.5.

Note 2: Gcr_ideal = TRUE means ideal calculation of the critical mass flow, see Appendix A1.2. Gcr_ideal = FALSE gives more realistic values, especially in flushing liquids.

5.4.18.5 Formulation

The value of the non-dimensional valve position (pos_com) controlling the area is a function of the downstream pressure P:

$$pos\_com = \begin{cases} 0 & (if\ \ P > P_{close}) \\ \dfrac{P_{close} - P}{P_{close} - P_{open}} & (if\ \ P_{close} > P > P_{open}) \\ 1 & (if\ \ P < P_{open}) \end{cases}$$

where P is the regulated outlet pressure, Popen, Pclose are inputs specifying the regulation zone of the valve. Popen must be smaller than Pclose. The actuator is modeled with a simple delay in the commanded position, without limiting the stroke speed:

$$\frac{d(pos)}{dt} = \frac{\text{linearInte rp1D(Kg\_vs\_pos, pos\_com)} - pos}{tao}$$

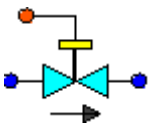
Then, the derivative of the valve area closing the formulation of the "`AbstractJunctionLoss`" component (see §*5.3.2.1*) is calculated as follows:

$$A = \text{Ao pos;} \quad \frac{dA}{dt} = A_0 \frac{d(pos)}{dt}$$

5.4.19 ValvePressRegUp

5.4.19.1 Description

This component represents a Relief Valve controlling the flow area as a function of the upstream pressure. It is inherited from the "`AbstractJunctionLoss`" component.



5.4.19.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.19.3 Ports

Name	Type	Parameters	Direction	Description
f1	Fluid	burnerGasesOption	OUT	Inlet / outlet fluid ports
f2	Fluid	burnerGasesOption	OUT	
s_pres	Analog_signal	1	IN	Pressure signal

5.4.19.4 Data

Name	Type	Description	Units
Ao	REAL	Valve area when fully open	m^2
P_close	REAL	Minimum upstream pressure required to open the valve	Pa
P_open	REAL	Upstream pressure that totally opens the valve	Pa
Re_lam	REAL	Laminar Reynolds number	-
Kg_vs_pos	TABLE_1D	Av correction vs non-dimensional stroke position	-
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m
tao	REAL	Time constant of area change filter	s
zetab	REAL	Backward loss coefficient	-
zetaf	REAL	Forward loss coefficient	-
Gcr_ideal	REAL	TRUE, ideal critical flow expansion, FALSE, empirical correlation	-
m_o	REAL	Initial mass flow	Kg/s

Note 1: *zetaf*, *zetab* = 0 means automatic calculation of the pressure drop coefficients from the orifice area and the adjacent connected flow areas. See §5.4.6.5.

Note 2: Gcr_ideal = TRUE means ideal calculation of the critical mass flow, see Appendix A1.2. Gcr_ideal = FALSE gives more realistic values, especially in flushing liquids.

5.4.19.5 Formulation

The value of the non-dimensional valve position (pos_com) controlling the area is a function of the upstream pressure P:

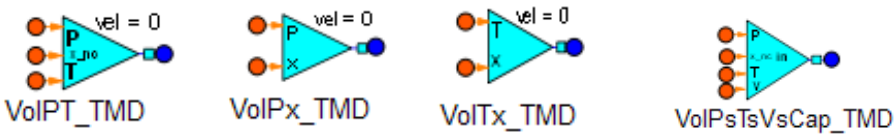
$$pos\_com = \begin{cases} 0 & (if\ \ P < P_{close}) \\ \dfrac{P - P_{close}}{P_{open} - P_{close}} & (if\ \ P_{close} < P < P_{open}) \\ 1 & (if\ \ P > P_{open}) \end{cases}$$

where P is the regulated inlet pressure, Popen, Pclose are inputs specifying the regulation zone of the valve. Pclose must be smaller than Popen. The actuator is modeled as for the ValvePressRegDown component, see §5.4.18.5.

5.4.20 VolPT_TMD, VolPx_TMD, Voltx_TMD and VolPsTsVsCap_TMD

5.4.20.1 Description

These components represent *capacitive* time dependent (TMD) *static* boundary conditions in P-T, P-qual and T-qual respectively. The external fluid velocity is assumed to be 0 as in a Tank, or input data for the VolPsTsVsCap_TMD component.



5.4.20.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

All these components have xd_nc as input data to specify the diluted gas mass fraction in liquid entries. VolPsTsVsCap_TMD also dispose of A_o input data to specify the junction area.

5.4.20.3 Ports

Name		Type	Parameters	Direction	Description
f		fluid	burnerGasesOption	IN	Inlet/Outlet fluid port
s_pres		analog_signal	(n = 1)	IN	Imposed variable 2 (-)
s_temp			(n = 1)	IN	Imposed variable 1 (-)
s_xNonCond	For VolPT type		(n = 1)	IN	Imposed non condensable mass fraction (-)
s_quality	For VolTx, VolPx		(n = 1)	IN	Imposed quality (-)

5.4.20.4 Formulation

See §5.3.7 in which the Init_Vol function is explained. Depending on the component type, the following inputs will be taken from the port signals and redirected to the Init_Vol function:

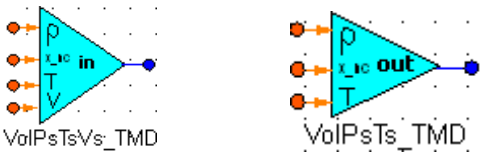
- VolPT_TMD: x_nco (the non-condensable mass fraction), P_o and T_o, vel =0.
- VolPx_TMD: *P_o* and *x_o*. vel =0.For pure fluids only
- VolTx_TMD: T_*o* and *x_o*. vel =0. For pure fluids only

The port signals can be defined with CONTROL components connected to its ports. If the signal ports are disconnected, the signal values will be boundary conditions to be defined in the experiment file.

5.4.21 VolPsTs_TMD and VolPsTsVs_TMD

5.4.21.1 Description

These components represent *resistive* time dependent boundary condition imposing *static* pressure, speed and temperature at Pipe inlets or exits (for subsonic or supersonic conditions).



5.4.21.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Only in VolPsTs_TMD. Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.21.3 Ports

Name		Type	Parameters	Direction	Description
f		fluid	burnerGasesOption	OUT	Inlet/Outlet fluid port
s_pres		analog_ signal	(n = 1)	IN	Imposed variable 2 (-)
s_temp			(n = 1)	IN	Imposed variable 1 (-)
s_xNonCond			(n = 1)	IN	Imposed non condensable mass fraction (-)
s_vel			(n = 1)	IN	Imposed speed or Mach number

Note: Input «s_vel» will be considered as speed (m/s) if it is positive. *Negative values are assumed to be Mach numbers.*

5.4.21.4 Data

Name	Type	Description	Units
Ao	REAL	Junction area. If 0, the connected cross area will be used	m^2
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m

5.4.21.5 Formulation

VolPsTs_TMD: Non-condensable mass fraction, static P and T are inputs. This component should be used to simulate Pipe exit conditions.

- It is built topologically by inserting a `Junction` and a `VolPT_TMD` component to which the non-condensable mass fraction, pressure and temperature signals (control variables) are assigned.

VolPsTsVs_TMD: Non-condensable mass fraction, static P, T and speed are inputs. This component should be used when simulating Pipe inlet conditions.

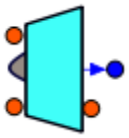
- It is coded similarly to a Junction component where the non-condensable mass fraction, pressure and temperature signals (control variables) are assigned to the upstream conditions of the fluid port. Instead of using a pressure drop balance, the upwind imposed speed is used to calculate the mass flow using the density calculated from the P/T inputs.

The port signals can be defined with CONTROL components connected to its ports. If the signal ports are disconnected, the signal values will be boundary conditions to be defined in the experiment file.

5.4.22 Intake

5.4.22.1 Description

The Intake component represents a non-ideal intake. It uses the standard atmosphere relationships and the Mach and altitude flight conditions as boundaries at the inlet. The outlet conditions are calculated by iterating in the outlet velocity to fulfill a pressure recovery value function of the inlet Mach number.



5.4.22.2 Ports

Name		Type	Parameters	Direction	Description
f		fluid	burnerGasesOption	IN	Inlet/Outlet fluid port
s_alt		analog_signal	(n = 1)	IN	Imposed altitude (-)
s_mach			(n = 1)	IN	Imposed Mach number (-)
meas_out			(n = 1)	OUT	Outlet signals, see note

Note:

- meas_signal[1]: Enthalpy shift CEA - Real properties (J/kg/K)
- meas_signal[2]: Static ambient pressure (Pa)
- meas_signal[3]: Static ambient temperature (K)
- meas_signal[4]: Flightspeed (m/s)
- meas_signal[5]: Inlet mass flow

Inlet port signals "s_mach" and "s_alt" give variable flight conditions to be imposed at the inlet of the intake. These port signals can be defined with CONTROL components connected to its ports.

If the signal ports are disconnected, the signal values will be boundary conditions to be defined in the experiment file.

5.4.22.3 Data

Name	Type	Description	Units
TPR_table	TABLE	Intake Total Pressure Recovery vs. flight Mach number	-
Ao	REAL	Junction area. If 0, the connected cross area will be used	m^2
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m

5.4.22.4 Formulation

This component behaves similarly as the VolPsTsVs_TMD component. The inlet ambient conditions (Po, To) are calculated through the "**atm_cond**" function, which uses the following atmosphere formulation:

- First, the geopotential altitude alt_g is calculated (altitude correction due to g=g(alt)): $\mathrm{alt_g} = \frac{\mathrm{alt}\cdot \mathrm{R_{earth}}}{\mathrm{alt}+\mathrm{R_{earth}}}$
- A base altitude alt_b is obtained by interpolating with alt_g in the geopotential altitude table (h_tab). This base altitude is the lower of the current altitude stretch of the table.
- The temperature gradient gr_b, the pressure ratio Pr_b and the base temperature T_b for each altitude stretch are obtained by entering with the base altitude in the corresponding table (gr_tab, Pr_tab, T_tab).

Then, the static temperature and pressure T_o and P_o are calculated:

$$T_o = T_b + gr_b \cdot \frac{(alt_g - alt_b)}{10^3}$$

$$P_o = \begin{cases} P0 \cdot Pr_b \cdot e^{-g_o \cdot \frac{Mmol}{RGAS} \cdot \frac{(alt_g - alt_b)}{T_b}} & (if\ gr_b == 0) \\ P0 \cdot Pr_b \cdot \left(\frac{T_b}{T_o}\right)^{g_o \cdot \frac{Mmol \cdot 10^3}{RGAS \cdot gr_b}} & (if\ gr_b \neq 0) \end{cases}$$

where *P0, R_earth, g_o* and *Mmol* are respectively the pressure at sea level, the earth's radius, the gravity at the earth's surface and the molecular weight, all of them known constants.

Once the inlet conditions are determined as a function of the flight altitude, the inlet velocity is calculated with the imposed Mach number:

$$vel = M_{in} \cdot \sqrt{\gamma \cdot RGAS/Mmol \cdot T_o}$$

The outlet conditions of the Intake are calculated by iterating in the outlet velocity to fulfill a total pressure recovery value (input data, TPR_table), function of the inlet Mach number:

$$P_{tot} = P_o \cdot \left(1 + \frac{\gamma - 1}{2} \cdot (vel/v_{sound})^2\right)^{\frac{\gamma}{\gamma-1}}$$

$$TPR = \frac{P_{out} \cdot \left(1 + \frac{\gamma-1}{2} \cdot (v_{out}/v_{sound})^2\right)^{\frac{\gamma}{\gamma-1}}}{P_{tot}}$$

where *M_in, P_o*, are the flight Mach number and the static inlet pressure, *P_out* is the outlet static pressure, *γ* is the isentropic constant of the air and *v_out* is the velocity at the intake exit considering the losses (or the shock) corresponding to the TPR (total pressure recovery), a priori determined as a function of the imposed Mach number by interpolating in the TPR table.

Note that *P_out* is in fact the static pressure (port variable) calculated by the connected capacitive component downstream of the intake, that becomes implicit together the previous equations. *A typical example is done in the high level "CombustChamber_ramjet" component, see §8.3.7.6, where an intake is connected to a supersonic combustor. The inlet pressure of the combustor will be coupled to the intake equations and to the fixed discharge pressure of the combustor, P_o, the ambient pressure*.

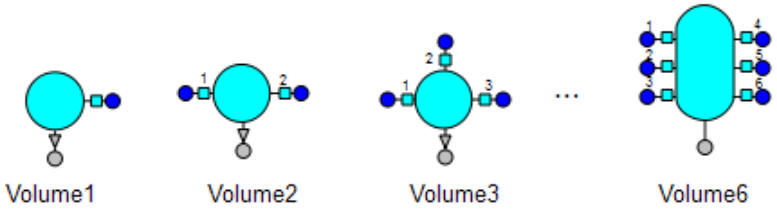
Also note that the default TPR table (input data) yields very important losses at high Mach numbers. The user is responsible of providing appropriate TPR tables according to the design of the intake.

5.4.23 Volume1 to Volume6

5.4.23.1 Description

Inherited from a "VolumeConstant", these components represent adiabatic Volume (simplified reservoir) with a given number of fluid ports (one to six depending on the component).

These components are *also* representative of a fluid capacity mixing or splitting flows depending on the conditions of connected components.



5.4.23.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

5.4.23.3 Ports

Name	Type	Parameters	Direction	Description
f[nports]	fluid	burnerGasesOption	IN	Inlet/Outlet fluid ports
meas_out	vol_measure	(n = 14)	OUT	Volume outlet signals

5.4.23.4 Data

Name	Type	Description	Units
init_option	ENUM	Option to specify the initial thermodynamic state	
P_o	REAL See §5.1.3.3	Initial Pressure	Pa
T_o		Initial temperature	K
rho_o		Initial density	kg/m^3
x_o		Initial quality	-
x_nco		Initial non-condensable mass fraction	-
xd_nco		Initial gas mass fraction diluted in the liquid phase	-
Vo	REAL	Initial fluid volume	m^3
L	REAL	Volume length - if zero volume is assumed to be a sphere	m
Pw	REAL	Wet perimeter. (0, circular cross area)	m
z_bottom	REAL	Elevation at the bottom of the volume relative to a z fixed axis	m
iside[nports]	INTEGER	Port sides (1 inlet, 2 outlet)	
iangle[nports]	REAL	0, frontal inlet/outlet; 90, lateral inlet/outlet (degrees)	degrees

Note: Not all the initialization inputs are necessary. See §5.1.3.3 about possible initializations

Use "iangle=90" for lateral branches where only the static pressure will be seen by the connected pipe

The "*iside(i)*" data (see §5.3.3.3) determines if the port "*i*" is on the inlet or on the outlet side. If the port configuration is as in the symbol, you do not need to modify the default values of *iside*. For example, if there is one port on the left side and 3 ports on the right of a Volume4 component, you should introduce *iside* ={1,2,2,2} instead of {1,1,2,2} ,which are the default values (two inlets- two outlets in this case).

The fluid volume (Vo) is the only geometric data that is always needed. If the other two geometric data are left as zero, the volume geometry is considered to be a sphere.

L	Pw	Geometry
0	0	Sphere, with radius calculated from the total volume
>0	0	Circular cylinder, with radius calculated from the cross area
>0	>0	Non Circular cylinder
0	>0	Circular cylinder, with radius calculated from the wet perimeter

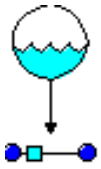
5.4.23.5 Formulation

See paragraph 5.3.5. Typical calculated volume variables are transmitted to the "meas_out" port.

5.4.24 WorkingFluid

5.4.24.1 Description

This component specifies the working fluids in a loop of a model: one main fluid with real properties and another as a perfect gas. It must be placed once in every loop of the model. All components connected to the same loop will have the same working fluids that can be mixed one with the other.



Another more specific component (CheckValve_FlSep) for the definition of *two* real working fluids in a circuit can be found in §6.3.5. The CheckValve_FlSep is recommended if a check valve is physically placed in a circuit preventing the vapors to be mixed with another fluid. It can be also used if no check valve is placed but no appreciable back flow of a main fluid would pass upstream. In this way, the mathematical resolution can be simplified considerably (no need to calculate mixture of fluids upstream of this component) and both upstream and downstream fluids can use real properties.

5.4.24.2 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	noBurnGases	IN	Inlet /outlet ports
f2	fluid	noBurnGases	OUT	

5.4.24.3 Data

Name	Type	Description	Units
fluid	ENUM	Working fluid	
fluid_nc	ENUM	Non-condensable working fluid	

Notes:

1 The available fluids are described in §4.1.2 and §4.1.3. In case of *no* need for a Non-condensable fluid, choose fluid_nc=NoFluid. Then, pure fluid properties will be assumed. The non-condensable mass fraction will be zero regardless of their initial value.

2 Some of the possible fluids listed in the pop-up menu selecting the working fluid (IPA, H2O2) do not have available property files yet. Methanol, acetone, benzene and toluene property files have no transport properties.

The fluid properties (except for those related to the state equations of a perfect gas or liquid) are based on the interpolation in external data contained in files (1D or 2D data tables).

The users may build their own property files. The files must be named as follows: *PfGasUsr1.dat …, PfLiqUsr1.dat* … or *RealUsr1.dat* … (depending on the type) … and added to the PROP_TABLES directory. The format must follow the specifications described in §4.3.3.2. For ESPSS models, it is enough to indicate the corresponding fluid name in this component.

Moreover, specific fluids with specific names linked to a particular CEA chemical can also be added, see §4.3.4

5.4.24.4 Formulation

There is no specific formulation on this component. The selected fluid names will be transmitted to the ports. All the port variable values are passed from one port to the other.

6. TANKS LIBRARY

6.1 OVERVIEW

TANKS is an ECOSIMPRO library for the transient simulation of rocket engines and spacecraft tanks. The most important features are the following:

- Gas, liquid and two-phase flow regimes are modeled for real fluids contained in Tanks. Mixtures of a real fluid (two phase conditions) with or without a non-condensable gas are included in the formulation.
- Absorption/desorption effects of non-condensable gases are optionally included in the formulation.
- Multiple correlations to calculate the heat transfer between the tank walls and the fluid cavities are included. The correlations work under boiling conditions.
- Most of the typical shapes of the walls (2D spatial discretization) have been included as separate components to model heat conductivity in walls and insulations.
- Single Tank models working with a pressuring gas and a liquid (both exchanging heat with the walls) are available. It includes the liquid level calculation.
- Bladder Tank component. It also includes the bladder motion calculation.
- 1D Tank components. These models consider a 1D moving spatial discretization for the liquid and gas sides, both exchanging heat with the walls. The heat and mass exchange at the liquid/gas interface is modeled.
- 1D Tank components also include the film boiling & bubbles rising formulation.

The TANKS components can be connected to the FLUID_FLOW_1D components to simulate a complete rocket engine cycle. Rocket engine and spacecraft pressurization systems where the Tanks play an important role are easy to evaluate using this Library.

6.1.1 Components classification

The components of the TANKS Library are listed below. All tank type components behave externally as capacitive elements (see §5.1.1):

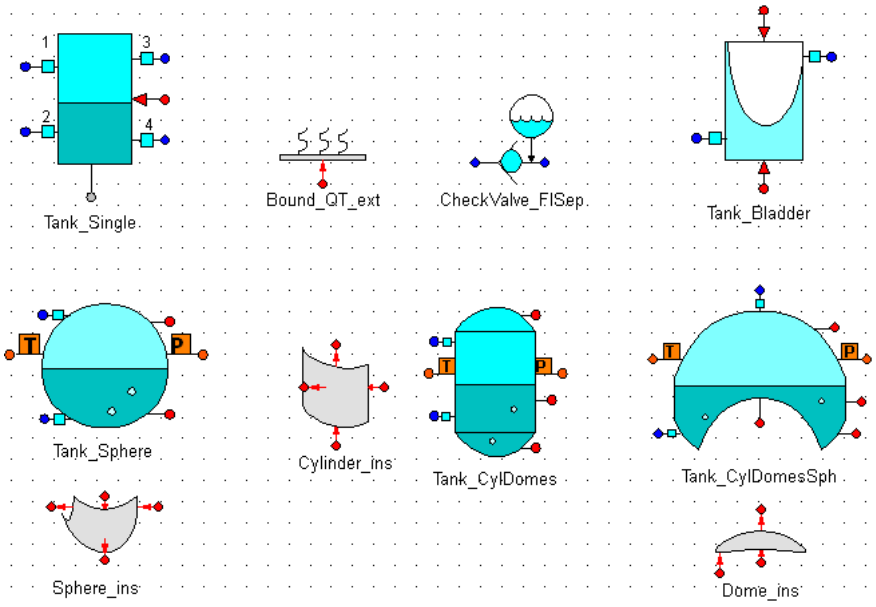


Figure 6-1 TANKS palette of symbols

1D Tanks (Tank_Sphere, Tank_CylDomes and Tank_CylDomesSph) and insulation components (shown in grey) include wall thermal models with a 2D nodal discretization:

- First direction is throughout the thickness,
- Second one along the height (vertical axis) with equidistance slabs.

The *CheckValve_FlSep* component represents a fluid separator. It should be used if a common circuit pressurizes different tanks with different liquids or for the definition of two "separated" **real** fluids working in the same loop.

6.1.2 Building a model

TANKS library can be considered as an important complement of the FLUID_FLOW_1D library for the simulation of rocket engine and spacecraft tanks. The tips of the fluid library will apply here.

Walls materials are selected via the ENUMERATIVE variable "mat" of the corresponding component in which a list of materials is provided (see §3.9).

6.1.2.1 Choosing the appropriate Tank Model

- The *Tank_Single* component allows calculating the liquid level evolution assuming that the gas mixture (vapor plus a non-condensable) is separated from the liquid. Nevertheless, liquid and gas phases are at the same equilibrium temperature that they would reach in a homogeneous mixture. One phase - one fluid state (sub cooled or superheated) is also included in the formulation of this component.
- The *Tank_Bladder* component assumes two different fluid temperatures and pressures, one for the liquid one for the gas mixture. It considers a bladder separating the liquid from the gas, in general two differenr fluids. It can be used as an accumulator or a pressurized tank.
- The *1D Tank* models calculate the temperature variation (wall, liquid and gas sides) along a "vertical" axis. The heat and mass transfer at the liquid/gas interface, as well as the heat transfer at walls/fluid sides (including the film boiling & bubbles rising formulation) are calculated.

  Use the *"Sphere_ins", "Cylinder_ins" and "Dome_ins"* components if a layer of insulation is covering the 1D Tanks. These components can be connected to the respective thermal ports of the Tank. In this way, not only will the heat conduction through the walls *(already included in the Tank components*) be modeled, but also through the insulation layers according to their wall properties.

6.1.2.2 Specific tips (Gravity & Port heights)

- Single and 1D Tank models can simulate the liquid level passing through a fluid port in such a way the exiting flow can be liquid or gas depending on the liquid level. Then, a "liquid" outlet will be automatically connected to the gas part of the tank if the remaining liquid in the tank is below the level of such fluid port, which is an input data through the connected junction, and vice versa, a "gas" outlet will be automatically connected to the liquid part of the tank if the remaining liquid in the tank is over the level of such fluid port.
- It is assumed that gravity acts in only one direction, i.e. top or bottom (*positive through the bottom)*. Only the vertical vector (GRAV variable, to be defined in the experiment file, see §5.1.3.6) is active because the *lateral accelerations are not included in the formulation*. Simplified Tanks can work under negative gravities. 1D Tanks (using special heat transfer between the liquid and the gas) do not allow negative gravity.

  The liquid height of a Tank is taken into account to calculate the liquid column pressure. An error message will be issued if the Tank ports are beyond the Tank height. Note that any Tank port (junction or valve components) has an input (z_level) with the junction level.

6.1.2.3 Numerical limitations

From a numerical point of view, the following recommendations and limitations should be considered:

- The wall and liquid volume discretization can have non-negligible influence on the 1D tank model results: Indeed, the local wall temperature seen by a liquid volume will depend on its size, the wall temperatures being constant inside each wall slab at a given time. The consequence is that the heating and possible evaporation of the liquid can be affected by the discretization.
- While pressurizing a tank with an inflow jet, the *vaporization rate could be better calculated with only one gas volume where a complete mixture of the non-condensable gases and vapors is calculated*. The vaporization rate is highly dependent on the vapor partial pressure that can be overestimated at the gas volume nearest to the liquid using several gas volumes. On the other hand, if the gas side contains two-phase flow (fog), numerical instabilities can be found with more than 1 elements.
- Use the absorption/desorption of non-condensable gases (NCG) only for particular studies, see §6.2.2.2. The number of liquid volumes has great influence on the diffusion of the diluted gases, but not on the equilibrium value of the absorbed/desorbed gas from/to the gas side. In some rare cases, numerical oscillations can be found calculating the film boiling in the liquid part of the Tank if more than one discretized liquid volume are used.
- The 1D Tank formulation will not be valid when the gas or the liquid side becomes empty (this normally happens if the outlets are near the bottom and a continuous outgoing mass flow is imposed). An appropriate message will stop the simulation in these cases.

  It is assumed that the incoming / outcoming flows are connected to the lowest liquid volume or to the upper gas volume. Then, 1D Tank formulation is not valid if the gas side has more than one discretized volume and a gas port is placed between the liquid level and the top of the tank, or when the liquid side has more than one discretized volume and a liquid port is placed between the bottom and the liquid level. An appropriate message will stop the simulation. This problem does not appear when there is only one discretized volume.
- The "heatExch" option = Diffusion at the gas/liquid interface (see §6.2.3.2) seems to be more sensitive to internal tank conditions (non-condensable mass fraction). It is recommended to use first "heatExch" option = EvapPool. *"heatExch = noExchange" will deactivate mass and heat exchange (in case of supercritical tanks, for example*).
- 1D Tanks must work with a non-condensable gas different than the real fluid filling the liquid part. The non- condensable gas will never condensate, even if it is the same fluid than the real fluid.

6.2 ABSTRACT COMPONENTS. MAIN FORMULATION

6.2.1 Abs_Tank

Inherited from a FLUID_FLOW_1D "Capacity", this component represents a *0D tank volume* calculating the liquid surface motion. It is the simplest model of a tank. Liquid, vapor and a possible non-condensable gas are assumed to be at the same equilibrium temperature. One phase state (sub cooled or superheated fluids) is included in the formulation.

6.2.1.1 Conservation and state Equations

Mass and energy conservation equations are inherited from the Capacity component (see §5.3.3). It is assumed that the volume variation is due to wall compressibility:

$$\frac{dV}{dt} = V\, k_{wall} P' \; ; \; P' = (\rho' - \partial\rho/\partial h_p * (u' + P\rho'/\rho^2)) / (\partial\rho/\partial P_h - \partial\rho/\partial h_p/\rho)$$

P' is calculated from the current state variables (density and energy) and the thermodynamic derivatives.

6.2.1.2 Heat Exchange with Walls and External Heating:

The term q_{in} appearing in the energy conservation equation of the parent component (Capacity) permits the exchange of heat through a THERMAL port.

$$q_{in} = h_{film} A_{wall} (T_{wall} - T)$$

The following equation calculates the thermal wall temperature:

$$T'_{wall} = (q_{port} - q_{in}) / (cp_{wall} \cdot m_{wall})$$

where q_{port} is the heat flux given by the connected thermal port. The film coefficient is calculated according to A3.2.

6.2.1.3 Calculation of the Liquid Surface Elevation.

Under two-phase flow conditions, with gravity forces, and assuming a flat vapor/liquid surface area, the liquid surface elevation z_{level} is calculated as follows:

$$z_{level} = z_{bottom} + (1-\alpha)(z_{top} - z_{bottom})$$

z_{top} and z_{bottom} are the top and bottom elevations of the volume.

The void fraction α is calculated according to the HEM method (see §4.5.3.1). The pressure elevation at port number *j* produced by the liquid column under gravity forces is:

$$\Delta P(j) = \begin{cases} g(\rho_g(z_{top} - \max(z_{jun,j}, z_{level})) + \rho_f(\max(z_{level}, z_{jun,j}) - z_{jun,j})) & (two\ phase) \\ \rho\ g\ (z_{top} - z_{jun,j}) & (otherwise) \end{cases}$$

where *g* is the actual gravity acceleration.

6.2.1.4 Over-flowing an outlet

The Tank model takes into account that the liquid surface elevation can move through the outlets. The quality, x, and the void fraction, α, at the outlet port number j are calculated as a smoothing between 0 and 1 when the liquid level passes near the junction elevation:

$$x_j = 0.5(1 - \tanh((z_{level} - z_{jun,i}) / D_{tank} / \text{overfl\_f}))$$
$$\alpha_j = x_j / (x_j + (1-x_j)\rho_g / \rho_l)$$

ρ_g and ρ_f are the gas and liquid densities calculated by the state functions. *overfl_f* is the ratio of the characteristic outlet flanges diameter to tank diameter. The enthalpy and density at the outlets will be calculated as a homogeneous two-phase flow with the previously calculated local void fraction α_j :

$$\rho_j = \alpha_j \rho_g + (1-\alpha_j)\rho_l; \qquad x_{nc,j} = x_{nc} \rho \cdot \alpha_j / (\rho_j \alpha)$$
$$h_j = x_j h_g + (1-x_j) h_l$$

where h_g and h_f are the gas and liquid enthalpies; ρ, x_{nc} are the mixture density and non-condensable mass fraction in the tank, all of them calculated by the state functions.

6.2.2 Abs_Tank_Vol1D

This component represents a *1D* tank volume. It can either be used to model the gas or the liquid part of a Tank. Operative Tanks will be built using two of these *1D* tank volumes in a topological model. The first control volume of the liquid part takes the lower position. The last liquid control volume will interact with the first gas control volume; last gas control volume takes the upper position of the tank. Liquid, vapor and a non-condensable gas can be present in any of the *n control volumes* in which the tank is discretized.

6.2.2.1 Conservation equations

The mass, energy and momentum equations are basically the same as in the Pipe component with variable cross area (see §5.3.8), but adding the moving grid terms corresponding to the discretization of a variable volume. Heat conduction and diffusive mass exchange are included. It is supposed to have "n" variable control volumes of the same size according to a staggered grid.

Global mass, non-condensable mass and energy equations:

$$V_i \rho'_i + V'_i \rho_i = (m_{jun,i-1} - m_{jun,i}) + (m_{dif,i-1} - m_{dif,i}) + (m_{fch,i-1} - m_{fch,i})$$

$$V_i (x_{nc,i}' \rho + x_{nc,i} \rho'_i) + V'_i x_{nc,i} \rho = (m_{nc\_jun,i-1} - m_{nc\_jun,i}) + (m_{dif,i-1} - m_{dif,i}) + (m_{nc\_boi,i-1} - m_{nc\_boi,i})$$

$$(u'_i \rho_i + u_i \rho'_i) V_i + u_i \rho_i V_i' = (mh)_{jun,i-1} - (mh)_{jun,i} + q_{wall,i} - P_i V'_i - g \rho_i V_i vel_i + (q_{dif,i-1} - q_{dif,i})$$

where for each control volume i (V_i),

- $m_{jun,i}$: Mass flow at the exit of control volume no. i *relative* to the staggered *moving* grid
- $m_{dif,i}$: Diffusive non-condensable mass flow at the exit of control volume no. i
- $m_{nc\_boi,i}$: non-condensable "boiling" flow at the exit of control volume no. i
- $m_{fch,i}$: Phase change (boiling or condensation) mass flow at the exit of control volume no. i
- ρ_i : Mixture density at control volume no. i
- $x_{nc,i}$: Non-condensable mass fraction at control volume no. i
- u_i , h_i : Mixture total specific energy and enthalpy at control volume no. i ($u=u_{static} + vel^2/2$)
- $q_{wall,i}$: Heat exchanged with the wall at control volume no. i
- $q_{dif,i}$: Diffusive (by conduction) heat flux between volumes i, i+1

The enthalpy flows $(mh)_{jun}$ and the non-condensable mass flows $(m)_{nc\_jun}$ are calculated using the upstream cell conditions:

$$(mh)_{jun,i} = m_{jun,i} h_{i,up} + m_{dif,i} h_{nc,i,up} + (mh)_{fch,i}$$

$$m_{nc\_jun,i} = m_{jun,i} \; x_{nc,i,up}$$

$h_{i,up}$ is the up-stream enthalpy of the fluid mixture and $h_{nc,i,up}$ is the up-stream enthalpy of non-condensable gases. $(mh)_{fch,I}$ are the phase change enthalpy flows, see below.

Diffusive mass and enthalpy flows.

Assuming a Lewis number of one, the diffusive flows are calculated as follows (assumed to be null at the gas/liquid interface and at the inlet/outlet of the tank):

$$m_{dif,i} = (\mu / \rho)_{i-1/2} (\partial \rho / \partial L)_{i-1/2} (x_{nc,i-1} - x_{nc,i}) A_{jun,i}$$

$$q_{dif,i} = (\lambda / \rho)_{i-1/2} (\partial \rho / \partial L)_{i-1/2} (T_{i-1} - T_i) A_{jun,i}$$

Phase change mass flows.

A) Condensation mass flows. When the temperature of a *gas* control volume is less than or equal to Tsat (quality less than one), the following formula estimates the global condensation at the gas volume i:

$$m_{fch,i} = -f_{\text{dropletDrop\_speed}} \rho_{sat\_liq,i} A_i (1 - \alpha_i); \qquad mh_{fch,i} = m_{fch,i} h_{sat\_liq,i}$$

where α is the void fraction (calculated by the state equations) of the corresponding control volume and $f_{dropletDrop\_speed}$ is the droplet drop speed defined by the user as a boundary condition in the experiment file. Droplets can remain as a fog in the ullage part ($f_{dropletDrop\_speed}$ = 0) or can drop to the liquid at a given speed.

B) Generalized boiling mass flows. When the temperature of a *liquid* control volume is greater than or equal to Tsat (quality greater than zero), the following formula calculates the global boiling at a liquid volume i:

$$m_{fch,i} = \rho_{sat\_vap,i} vel_{buble,i} A_i \alpha_i; \quad mh_{fch,i} = m_{fch,i} h_{sat\_vap,i}$$

where $vel_{bubble,i}$ is the bubble rising speed calculated according to A4.4.

When a non-condensable mass flow is injected in the liquid side, the corresponding "boiling" flow in the liquid volumes is similarly calculated:

$$m_{nc\_boi,i} = \rho_i x_{nc,i,up} vel_{buble,i} A_i \alpha_i$$

C) Film boiling mass flows. When the wall temperature of a *liquid* control volume is greater than Tsat and the Tfluid <= Tsat, the following formula calculates the film boiling rate at the tank inner walls:

$$m_{fch,i} = \rho_{sat\_vap,i} vel_{buble,i} A_i N_{bub,i} / V_i; \quad mh_{fch,i} = m_{fch,i} h_{sat\_vap,i}$$

A_i is the mean cross area of volume no. i (depending on time). $N_{bub,i}$ is the total volume of bubbles at volume no. i and $vel_{bubble,i}$ is the bubble rising speed calculated according to A4.4. The bubble volume evolution is calculated from the equation of a convective flow of bubbles at the local rising speed:

$$N'_{bub,i} = P_{bub,i} + (N_{bub} vel_{bub} / L)_{i-1} - (N_{bub} vel_{bub} / L)_i)$$

P_{bub} is the bubbles production calculated from the film boiling heat flux:

$$P_{bub,i} = q_{film\_boil} / \Delta h \ / \rho_{sat\_vap,i}$$

where Δh is the latent heat and q_{film_boil} is the film boiling heat exchange calculated in 0, see §6.2.2.4.

Momentum equations (junction mass flow equation):

$$0.5(L_{i-1} + L_i) \cdot m'_{jun,i} = A_{i-1}\left[P + qn - 0.25\xi \cdot \rho \cdot vel|vel|\right]_{i-1} - A_i\left[P + qn + 0.25\xi \cdot \rho \cdot vel|vel|\right]_i - 0.5(\rho_{i-1}V_{i-1} + \rho_i V_i) \cdot (g + vel'_{grid,i}) - 0.5(P_{i-1} + P_i)(A_{i-1} - A_i)$$

where,

L_i	: Height of control volume no. i (depending on time)
A_i	: Mean cross area of control volume no. i (depending on time)
qn_i	: Numeric dissipation
$vel_{grid,i}$	: Grid velocity in the middle of control volume no. i (see below)
vel_i	: Mean velocity *at* control volume no. i. (= vel_{grid} + m_{jun_mean}/A)

The grid acceleration will be calculated in the inherited components from the gas/liquid interface equations. It is assumed that all the control volumes have the same rate of change which is coherent with uniform volume discretization, as in:

$$vel_{grid,i} = vel_{grid,i} + V'_i / A_i; \qquad V'_i = V'_g / n$$

where V'_g is the rate of change of the gas volume calculated in §6.2.3.3. The artificial dissipation, qn(i), is calculated as follows:

$$qn(i) = -Damp(m_{jun}(i+1) - m_{jun}(i)) / Av_{sound}(i)$$

Momentum equations are applied at the boundaries of any discretized volume with the exception of the last liquid volume and the first gas volume, where the gas/liquid interface is placed. Similarly, the mass flows at the inlet of the first liquid control volume and the last gas control volume will be calculated in the junction components connected to the tank.

6.2.2.2 Absorption and desorption

The liquid volume of 1D tank needs an additional equation for the diluted gas:

Diluted non-condensable mass equation:

$$V_i(x^d_{nc,i}{}'\rho + x^d_{nc,i}\rho'_i) + V'_i\, x^d_{nc,i}\rho = (m^d_{jun,i-1} - m^d_{jun,i}) + (m^d_{dif,i-1} - m^d_{dif,i})$$

where for each control volume i (V_i),

$x^d_{nc,i}$	: Diluted gas mass fraction at control volume no. i
$m^d_{jun,i}$	: Diluted gas mass flow at the exit of control volume no. i
$m^d_{dif,i}$	: Diffusive diluted gas mass flow at the exit of control volume no. i

As for the NCG conservation equation, the convective diluted gas mass flows is calculated using the upstream cell conditions: $m^d_{jun,i} = m_{jun,i}\, x^d_{nc,i,up}$ The absorption/desorption terms inside the tank are modeled as a diffusion term, because the gas in the ullage volume diffuses through the gas/liquid interface surface. Consequently they are represented in terms of diffusive fluxes. To keep the notation consistent with the previous implementation, the absorbed/desorbed gas mass flow into the liquid is indicated with m_{abs}. According to a Godunov's discretization the conservative variables at cell *i* could be computed by the fluxes at the face interfaces *i+1/2* and *i-1/2.* Considering the diffusive flux as governed by Fick's law a single flux is described by:

$$f_{abs,i-1/2} = D_{i-1/2}\, A_{i-1/2}\, (\partial \rho x_d^{nc} / \partial L)\big|_{i-1/2}$$

The description of the derivative of the density and the mass fraction could be described with a simple difference of the east and west cells. By calling ΔL_i the size of the *i*-th cell, the first the term on the right hand side of the previous equation is:

$$(\partial \Xi / \partial \eta)\big|_{i-1/2} = \frac{\frac{1}{2}(\Xi_i - \Xi_{i-1})}{\Delta \eta} = \frac{\frac{1}{2}(\Xi_i - \Xi_{i-1})}{\frac{1}{2}\left(\frac{\Delta L_{i-1}}{2} + \frac{\Delta L_i}{2}\right)} = \frac{\Xi_i - \Xi_{i-1}}{\frac{\Delta L_{i-1}}{2} + \frac{\Delta L_i}{2}},$$

In case of evenly spaced cell the flux is then approximated by a second order central difference.

The diffusion coefficient $D_{i-1/2}$ and the cross section area of the tank $A_{i-1/2}$ in a specific cell are computed with a linear interpolation between the adjacent cell nodes:

$$D_{i-1/2} = D_{i-1} + \frac{D_i - D_{i-1}}{\frac{\Delta L_{i-1}}{2} + \frac{\Delta L_i}{2}}\left(\frac{\Delta L_{i-1}}{2}\right)$$

The boundary condition at the interface between the liquid and the gas phase is given by the maximum value of absorbed gas that can be in the liquid.

Diffusion coefficients are computed after the Stoke-Einstein relation

$$D_{AB} = \frac{kT}{6\pi R_A \mu_B}$$

where *k* is the Boltzmann constant, μ is the viscosity of the liquid and R is the radius of the gas molecule. This formula is an approximation so, if better data are available they should be used. An easy way to adjust the diffusion coefficient is to adjust the diffusion turbulence factor that could account for both coefficient inaccuracy and the absorption enhancement caused by turbulence.

The user may experience small variations of mass when using this option (around 0.1% of the total mass). The resulting drop in the tank pressure due to the absorption effect is strongly influenced by the number of nodes available in the tank. This is related to the way the Tank component is discretized. IF it is not possible to increase the number of nodes on the final simulation, it is advisable to tune the diffusion turbulence coefficient in order to have the appropriate change in pressure.

6.2.2.3 Pressure, temperature and quality calculation

At each discretized volume, the non-derivative state variables (pressures, qualities and temperatures) will be calculated using the state functions. See §5.3.3.2 and §4.5.3.1.

6.2.2.4 Heat Exchange with Walls

The term *q_wall(i)* appearing in the energy equation will permit the exchange of heat with the walls:

$$q_{wall,i} = hc_i A_{wet,i}(tp\_in.T(i) - T_i),$$

where $A_{wet,i}$ is the wet area of volume number i depending on time; *tp_in* is the name of the thermal port connected to the volume. Port temperatures tp_inT(i) behave as the wall internal temperatures, to be determined in the connected components. Two options are available calculating the heat exchange coefficients hc:

- "WallheatExch = Boiling" uses the correlations indicated in appendix A4.3 [RD-21].
- "WallheatExch = NoBoiling" assumes natural convection correlations.

The term q_{wall} includes the natural convection for the liquid or gas volumes and the film boiling heat exchange if any (depending on the options and wall/fluid conditions). The film boiling contribution will not normally heat the liquid bulk because it is employed in vaporizing the liquid, forming vapor bubbles that are considered as a phase change flows (see the conservation equations terms). The user can factorize the film boiling heat exchange coefficient, see the input data description in §6.3.4.4.

6.2.3 Abs_Tank_Cryo_1D

This component is topological and represents the set of the gas and the liquid parts of a *1D* tank volume. The particular geometrical shapes and heat exchanges (using the appropriate geometrical functions) will be defined later in the inherited components.

6.2.3.1 Topology

The "Abs_Tank_Cryo_1D" component is built topologically with two 1D fluid volumes (Abs_Tank_Vol1D type component), one simulating the liquid, and the other for the gas.

Each of these 1D volumes has two fluid ports, one of them connecting the liquid with the gas (interface) and the other connecting the tank to the external fluid ports (junctions). The following sections describe the different mass and energy exchanges at the interfaces.

6.2.3.2 Gas/liquid interface. Mass & energy exchange equations

Vaporization exchanges:

Two possibilities are foreseen:

A) "heatExch = EvapPool". Energy-balance approach:

Assuming a very thin layer at Tsat, the equilibrium conditions would allow calculating the evaporation (*positive or negative*) flow rate through this layer:

$$m_{vap\_st} = A(h_g(T_g - T_{sat}) + h_l(T_l - T_{sat}))/(h_{sat\_vap} - h_{sat\_liq})$$

$$q_{gas\_to\_liq} = A \cdot h_g(T_g - T_{sat}) - m_{vapor} \cdot h_{sat\_vap}$$

Tg, Tl are the gas and liquid temperatures of the control volumes nearest to the gas/liquid interface. The saturation enthalpies, h_{sat_vap}, h_{sat_liq} and the saturation temperature, Tsat, are those calculated at the partial vapor pressure of the gas volume nearest to the liquid side.

Similarly, hg, hl are the gas and the liquid side heat exchange coefficients. All these quantities were calculated by the "Abs_Tank_Vol1D" component. Heat exchange coefficients are factorized with f_{turb} which is a correcting factor defined by the user in the BOUND block of the experiments. For example:

```
BOUNDS
Tank_LH2.turb_fac = 0.1 -- timeTableInterp( TIME, turb_table1D)
...
```

B) "heatExch = Diffusion". Diffusion approach:

The calculation is made according to Appendix A4.2

The vaporization model will be switched off if no liquid conditions are present on the liquid side.

Boiling / Condensation exchanges.

These contributions were calculated by the "Abs_Tank_Vol1D" component and they correspond to the terms m_{fch} at the last liquid control volume and at the first gas control volume, see §6.2.2.1.

Absorption / Desorption.

These phenomena are accounted in the "Abs_Tank_Vol1D" component and they correspond to the term $m^d{}_{dif}$ as in §6.2.2.2.

6.2.3.3 Gas/liquid interface. Pressure equation coupled with wall compressibility

A closing equation is still needed to calculate the rate of change of the geometrical volumes acting in the 1D momentum equations (see §6.2.2.1). The following physical conditions are imposed:

$$P_g = P_l$$

$$(V_g + V_l)' = (V_g + V_l) k_{wall} (\partial P / \partial t)$$

where,

P_g, P_l: Gas/liquid pressures calculated at the control volumes nearest to the interface

K_{wall}: Compressibility due to wall expansibility (input data, 1/Pa)

V_g V_l: total gas and liquid volumes dynamically calculated in §6.2.2.1

Note that previous equations together with the momentum equations of the gas/liquid volumes form a linear box automatically detected and solved by EcosimPro.

6.2.3.4 Over-flowing an outlet

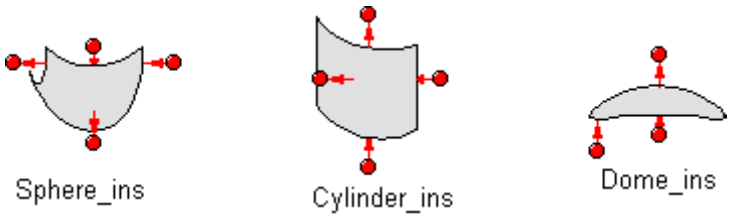
The 1D Tank models take into account the fact that the liquid surface elevation can move through the outlets. The same formulation as in the single tanks is used. See §6.2.1.4

6.3 OPERATIONAL COMPONENTS

6.3.1 Sphere_ins, Cylinder_ins, Dome_ins

6.3.1.1 Description

These components represent 2D thermal insulation plates with four resistive thermal ports. The numbers of nodes (in vertical and radial direction) are input parameters.



6.3.1.2 Construction Parameters

Name	Type	Description
nr	INTEGER	Number of thermal nodes in thickness direction
nz	INTEGER	Number of thermal nodes in vertical direction (slabs, see Figure 6-3)

6.3.1.3 Ports

Name	Type	Parameters	Direction	Description
tpr_in	thermal	(n = nz)	IN	Thermal inlet radial port
tpr_out		(n = nz)	OUT	Thermal outlet radial port
tpz_in	thermal	(n = nr)	IN	Thermal inlet axial port
tpz_out		(n = nr)	OUT	Thermal outlet axial port

6.3.1.4 Data

Name	Type	Description	Units
Di	REAL	Inside diameter	m
Do	REAL	Outside diameter	m
L	REAL	Cylinder or dome length	m
To	REAL	Initial temperature	K
cp	REAL	Wall Specific heat if mat=None	J/kg/ºK
k	REAL	Wall conductivity if mat=None	W/m/ºK
mat	ENUM	Material	-
rho	REAL	Wall density if mat=None	

Note: Wall materials are selected via the ENUMERATIVE variable "mat" which provides a list of materials (see §3.9).

6.3.1.5 Formulation

The mesh of the plate is defined by the values of *nr* (number of nodes in the radial direction) and *nz* (number of nodes in the axial direction). The temperature at each node is stored in an array called T[nr,nz]. The qr[nr+1, nz] array contains the heat flows across nodes in the radial direction, and qz[nr, nz+1] contains the heat flows across nodes in the axial direction.

Cylindrical geometry.

The thermal capacitance Ci of each node is given by:

$$\Delta S_i = \pi\left((D_{in} + 2i\Delta r)^2 - (D_{in} + 2(i-1)\Delta r)^2\right)/4;$$
$$C_i = \rho_{mat} \cdot cp_{mat} \cdot \Delta S_i \cdot \Delta z; \qquad where$$
$$\Delta z = L/nz; \qquad \Delta r = (D_{out} - D_{in})/nr/2$$

The internal heat flows in the axial and radial directions are given by:

$$qz_{i,j} = \lambda_{mat} \Delta S \cdot (T_{i,j-1} - T_{i,j})/\Delta z$$
$$qr_{i,j} = \lambda_{mat} 2\pi \cdot \Delta z \cdot (T_{i-1,j} - T_{i,j})/\log\left((D_{in} + (2i-1)\Delta r)/(D_{in} + (2i-3)\Delta r)\right)$$

Spherical geometry.

In this case the following expressions differ from the cylindrical case:

$$\Delta S_i = 2/3\pi\left((R_{sph} + i\Delta r)^3 - (R_{sph} + (i-1)\Delta r)^3\right)/R_{sph}; \qquad R_{sph} = D_{in}/2$$

$$qz_{i,j} = \lambda_{mat} \Delta S \cdot (T_{i,j-1} - T_{i,j})/(R_{sph}\Delta\theta_j)$$

$$qr_{i,j} = \frac{4\pi \cdot \lambda_{mat} \cdot \Delta z \cdot (T_{i-1,j} - T_{i,j}) \cdot (D_{in} + 2(i-1)\Delta r)}{(D_{in} + (2i-3)\Delta r)^2 \left(2/(D_{in} + (2i-3)\Delta r) - 2/(D_{in} + (2i-1)\Delta r)\right)}$$

where "θ" is the elevation angle in spherical coordinates (centred in the cells). Note that j=nz corresponds to the pole, θ=90 degrees:

$$\theta_j = a\tan((R_{sph} - z_j)/a_j); \qquad a_j = \sqrt{R_{sph}^2 - (R_{sph} - z_j)^2} \qquad z_j = L - (j - 0.5)\Delta z$$

The energy equation for the corresponding diffusive nodes is:

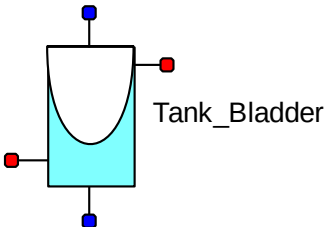
$$C_i \frac{dT_{i,j}}{dt} = \left(qr_{i,j} - qr_{i+1,j}\right) + \left(qz_{i,j} - qz_{i,j+1}\right)$$

At the ends of the plate, a conductivity equation will be considered for the last half nodes using the port temperature as the right condition.

6.3.2 Tank_Bladder

6.3.2.1 Description

This component represents a bladder tank containing two separated fluids at two different temperatures. The bladder motion is calculated as a function of the volume pressures and the bladder resistance. This component can be used as an accumulator or a pressurized tank.



6.3.2.2 Ports

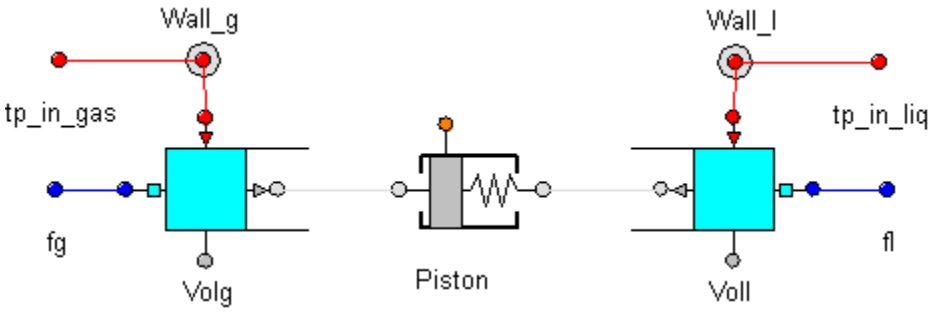
Name	Type	Parameters	Direction	Description
fg	Fluid		IN	Inlet/Outlet fluid ports
fl	Fluid		IN	Tank pressure (Pa)
tp_in_gas	thermal	(n = 1)	IN	Thermal port to lower dome
tp_in_liq	thermal	(n = 1)	IN	Thermal port to upper dome

6.3.2.3 Data

Name	Type	Description	Units
D	REAL	Gas/Liquid surface diameter	m
P_o	REAL	Initial Pressure in Tank	Pa
T_g	REAL	Initial gas temperature	K
T_l	REAL	Initial liquid temperature	K
Vog	REAL	Initial gas volume	m^3
Vol	REAL	Initial liquid volume	m^3
cp_w	REAL	Wall Specific heat if mat=None	J/kg/ºK
dP_vs_Vol	TABLE_1D	Bladder resistance: Gas-Liquid deltaP vs Gas Volume	
mat	Material	Wall Material	-
rho_w	REAL	Wall density if mat=None	kg/m3
th	REAL	Wall thickness	m
x_nco	REAL	Initial non-condensable mass fraction	-
z_bottom	REAL	Elevation of the tank bottom relative to a body axis system	m

6.3.2.4 Topology

The Tank_Bladder component is built topologically with two fluid chambers, one "piston", and two THERMAL "Dnode" capacities simulating the liquid and the gas walls of the tank.



6.3.2.5 Formulation

Each chamber is inherited from a variable volume component, see §5.3.4.The fluid volumes allow exchanging heat through the walls, as indicated in the topology scheme. The walls, simulated by two diffusive nodes, can exchange heat through the thermal ports (exterior).The two fluid volumes are mechanically connected by the piston, but there is no fluid connection, so two different working fluids should be defined in the connected fluid loops.

Bladder movement. The rate of change of the geometrical volumes (V') intervening in the volume equations is calculated by the piston equations simulating the bladder. A small piston mass (= gas volume) has been supposed to prevent oscillating responses in the bladder motion. The viscous force cv is assumed to be zero.

$$F = -F_{ext} + A_{1,pis} \cdot P_1 - A_{2,pis} \cdot P_2$$

$$V^{'} = A_{I/F} x'$$

The external force acting on the bladder is interpolated from the input data table "dP_vs_Vo", the pressure increase needed to have a certain gas volume:

```
Piston.s_Fext.signal[1] = A * linearInterp1D(dP_vs_Vol, Vg])
```

End stops (use of EL ZONEs)

- When ($\frac{x - x_{min}}{x_{max} - x_{min}} \leq 0$ and F<0) OR ($\frac{x - x_{\min}}{x_{\max} - x_{\min}} > 1$ and F>0) $\Rightarrow$ x'= 0
- In the rest of the cases: $x'' = \frac{F}{M_{pis}}$

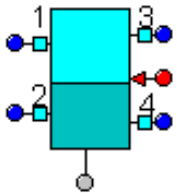
X_{min}, x_{max} are calculated from the initial liquid and gas cavities. The mechanical stops prevent over expanding the bladder, just limiting the gas/liquid capacities to the geometrical values and capturing the corresponding pressure oscillations due to a sudden stop of the liquid mass flow.

6.3.3 Tank_Single

6.3.3.1 Description

Inherited from the Abstract Component "Abs_Tank", this component represents a 0D tank volume. It is the simplest model of a pressurized tank calculating the liquid surface motion under two phase conditions.

This fluid capacity accounts for geometrical volume changes under pressure variation due to wall compressibility. Note that this symbol also represents spherical or cylindrical (vertical and horizontal cylinders) shapes; see the geometrical input data table.



Liquid, vapor and a possible non-condensable gas are assumed to be at the same equilibrium temperature. One phase state (sub cooled or superheated) is also included in the formulation.

6.3.3.2 Ports

Name	Type	Parameters	Direction	Description
f[4]	fluid		IN	Inlet/Outlet fluid ports
tp_in	thermal	(n= 1)	IN	Thermal port
meas_out	vol_measure	See §5.2.2.2	OUT	Tank outlet signals

6.3.3.3 Data

Name	Type	Description	Units
L	REAL	Volume length. L=0, a sphere. L<0, horizontal cylinder	m
Vo	REAL	Initial fluid volume	m^3
init_option	ENUM	Option to specify the initial thermodynamic state	
P_o	REAL	Initial Pressure	Pa
T_o	See	Initial temperature	K
x_o	§5.1.3.3	Initial quality	-

Name	Type	Description	Units
x_nco		Initial non-condensable mass fraction	-
rho_o		Initial density	kg/m^3
ht_option	ENUM	Wall-fluid heat transfer option (see §A3)	
hc_dat	REAL	Heat transfer coef. if ht_option = constant	W/m^2*K
mat	Material	Wall Material	-
rho_w	REAL	Wall density if mat=None	Kg/m3
cp_w	REAL	Wall Specific heat if mat=None	J/kg/ºK
th	REAL	Wall thickness	m
kappa_wall	REAL	Compressibility due to wall expansibility, dV_VdP	1/Pa
overfl_f	REAL	Ratio of the characteristic outlet flanges diameter to tank diameter - for overflowing calculation (see §6.2.1.4)	-
z_bottom	REAL	Bottom elevation relative to a z fixed axis	m
iside[10]	INTEGER	Port sides (1 inlet, 2 outlet)	

Note 1: The "*iside(i)*" data determines if the port "*i*" is on the inlet or on the outlet side. See §5.3.3.3

Note 2: Not all the initialization inputs are necessary. See §5.1.3.3 about possible initializations

Note 3: L<0 means that the volume is a *horizontal* cylinder of length abs(L). The liquid level will be calculated accordingly

6.3.3.4 Formulation

The calculation is described on the abstract component "Abs_Tank". See §6.2.1. The operational component only fixes the number of fluid ports (4). Non used ports can be closed by a DeadEnd component.

The single Tank model simulates the liquid level passing through a fluid port in such a way that the exiting flow can be liquid or gas depending on the liquid level (see §6.2.1.4). In case of a sphere, the level will be calculated accordingly with the wall shape.

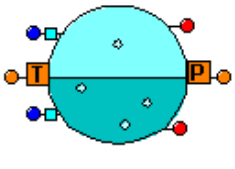
6.3.4 Tank_Sphere, Tank_CylDomes, Tank_ CylDomesSph

6.3.4.1 Description

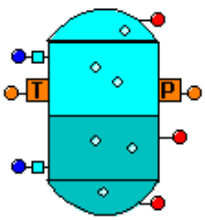
Inherited from the Abstract Component "Abs_Tank_Cryo_1D", these components simulate a two-phase Tank containing two fluids, a liquid which can evaporate (even boiling) and a pressurization gas receiving the vapor. Vapor condensation is also calculated along with the liquid level evolution due to filling or emptying processes.

The walls have *fixed* 2D spatial discretization. The fluid temperatures are simulated by means of two 1D moving grid spatial discretizations for the liquid and gas sides, both exchanging heat with the walls. The heat and mass exchange at the liquid/gas interface is also modeled. Both fluid capacities account for geometrical volume changes under pressure variation due the wall compressibility

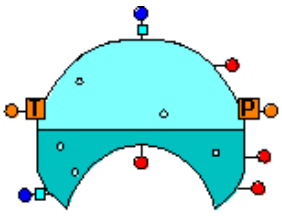
Three kinds of Tanks are considered depending on the geometrical shapes: Tank_Sphere, Tank_CylDomes and Tank_CylDomesSph.



Tank_Sphere



Tank_CylDomes



Tank_CylDomesSph

6.3.4.2 Construction Parameters

Name		**Description**
nd	See Figure 6-3	Number of thermal nodes in the wall vertical direction (slabs)
nr		Number of thermal nodes in the wall thickness direction
ng		Number of gas control volumes
nl		Number of liquid control volumes
AbsorOption	ENUM	Activate/deactivate absorption -desorption option. *Use this option only for particular studies*

The numbers of fluid and thermal nodes (see figure Figure 6-3) should be based on the following:

- If it is not necessary to calculate the thermal stratification in the liquid or in the gas sides, use only one fluid volume for each side, mainly for the gas side where a more effective mixture (3D effects) would take place if an incoming pressurization jet is present.
- The number of control volumes in the liquid or gas cavities should be 1 if the corresponding inlet/outlet ports are at an intermediate lateral position (inlet/outlet flows are supposed to be connected to the bottom of the liquid side or to the top of the gas side).

6.3.4.3 Ports

Name	**Type**	**Parameters**	**Direction**	**Description**
f[2]	fluid		IN	Inlet/Outlet fluid ports
s_pres	signal	(n = 1)	OUT	Tank pressure (Pa)
s_temp		(n = 1)	OUT	Tank temperature (K)
tp_in_Dome2	thermal	(n = nd)	IN	Thermal port to lower dome
tp_in_Dome3		(n = nd)	IN	Thermal port to upper dome

6.3.4.4 Data

Name	**Type**	**Description**	**Units**
R1	REAL	Sphere radius	m
R2	REAL	Lower dome radius	m
R3	REAL	Upper dome radius	m
RZ	REAL	Cylindrical part radius	m
th_s	REAL	Semi sphere wall thickness	m
th_2	REAL	Lower dome wall thickness	m
th_3	REAL	Upper dome wall thickness	m
th_Z	REAL	Cylinder wall thickness	m
z_bottom	REAL	Elevation of the tank bottom relative to a body axis system	m
heatExch	ENUM	Gas/Liquid exchange option (see note 1)	
P_o	*See note 3*	Initial Pressure in Tank	Pa
T_g		Initial gas temperature	K
T_l		Initial liquid temperature	K
x_nco		Initial non-condensable mass fraction in gas volume	-
Vl_per		Initial fill level (*Must be greater than 0*)	%
mat	Material	Wall Material	-
cp_w	REAL	Wall Specific heat if mat=None	J/kg/ºK
k_w	REAL	Wall conductivity if mat=None	W/m/ºK
rho_w	REAL	Wall density if mat=None	kg/m3
kappa_wall	REAL	Compressibility due to wall expansibility (see §6.2.3.3)	1/Pa
overfl_f	REAL	Ratio of the characteristic flanges diameters to tank diameter - for overflowing calculation (see §6.2.1.4, note 3)	-
wallHeatExch	ENUM	= Boiling➔film boiling and bubble rising calculation (note 2)	
f_boi	REAL	Multiplier of film boiling correlation, see §6.2.2.4	
UserDefSolubData	Absorption/ desorption	Binary solubility coefficients if UserDefSolubData = TRUE	
A_coef_sol			K

Name	Type	Description	Units
B_coef_sol	data		
Diff_Turb_Factor		Factor weighting diffusion of diluted gases	-
xd_nco		Pre-existing gas mass fraction diluted in the liquid phase	-

Note 1: The "heatExch = Diffusion" uses the correlations indicated in Appendix A4.2. "heatExch = EvapPool" uses the *Energy-balance approach* (see §6.2.3.2) concerning the gas/liquid mass exchange. "heatExch = noExchange" will deactivate mass and heat exchange (in case of non-known correlations, supercritical case for example)

Note 2: If "wallHeatExch = Boiling" and the wall temperature surpasses the saturation one, then the number, speed and volume of the bubbles rising to the gas side from the inner wall of the liquid side will be calculated together with the corresponding extra pressurization of the Tank (see §6.2.2 and A4.3)

Note 3: The liquid side can be initialized in gas conditions (T_l=T_g) to allow the simulation of filling processes. In this case, use low values (1e-4) of overfl_f datum and low values of the initial fill level (Vl_per) but greater enough to cover the flange of the liquid entry. You can select low initial pressures and to choose initial conditions in vapor (x_nco = 0) or in non- condensable gases (x_nco = 1).

Note 4: Each instance of a 1D tank will produce three boundaries (automatically loaded in the default experiment) to control the rotational axial speed and the vaporization and condensation flow rates at the liquid/gas interface:

OMEGA: Angular velocity of the tank (rad/s). *This variable is not used; no angular acceleration is taken into account for the moment*

turb_fac: Turbulent factor law modulating vaporization mass flows. This factor will multiply the heat exchange coefficients at the gas liquid interface for the evaluation of the vaporization rate, §6.2.3.2

dropletDrop_speed: Droplet drop speed in the tank (m/s), in case of two phase flow in the ullage part of the tank, see §6.2.2.1, subparagraph "Condensation mass flows"

6.3.4.5 Formulation

These components inherit the formulation of the Abstract Component "Abs_Tank_Cryo_1D" where two 1D fluid volumes (gas and liquid parts) are linked by the liquid/gas interface, see §6.2.3.

6.3.4.6 Topology

This component (*which already inherits two 1D fluid volumes*) incorporates several walls with 2D (radial and axial) temperature spatial discretization. The walls are modeled as the insulation plates, see §6.3.1.5. Geometrical data are described below:

- Tank_Sphere: two spherical walls
- Tank_CylDomes: two domes and one cylinder
- Tank_CylDomesSph: two domes, one cylinder and one semi sphere

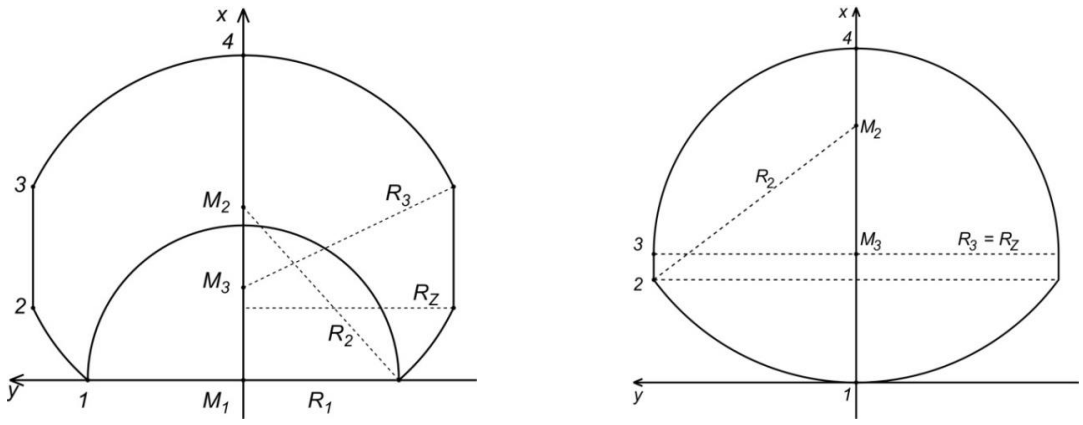


Figure 6-2: Cross section through LH2/LOX tank compartments

Wet areas and moving grid definition

The calculation of the wet areas and other geometrical parameters needed to the moving grid definition is made here. For example, in case of a sphere the following functions are used:

```
-- Call to the surface shape routines

Tank_Shapes (2, GRAV,0, OMEGA*57.29578, 100*Vl/V_tot, 0.1, \
       Voll.f1.rho,  Voll.meas.signal[9], 0, R2, R2, R2, R2, V_tot, ST_option, \
       A, A_wet, COG, ERR_ID,  Lh, nl, Al, Awl, ng, Ag, Awg)
```

The *Tank_Shapes* function calculates the surface shapes taking into account either the gravity and spin acceleration (see Annex A5), or only the geometrical shapes [RD-22] and assuming a flat gas/liquid interface. *Currently, the spin and lateral acceleration effects are deactivated because this function does not work with the GCC compiler*.

The application of Tank geometrical formulae to the calculation of the liquid height, wet areas, etc. from the calculated liquid volume *is implicit (cubic equations)*

Cover matrix concept

The heat exchange between the moving fluid grid and the fixed grid wall temperatures is calculated by using the cover matrix concept:

```
-- Cover matrix concept: Inner walls to gas and liquid

Cover_2Sh2Fl_inv(nl, ng, nd, Awl, Awg, A_tot, \
      Dome2_Wall.tpr_in.Tk, Dome3_Wall.tpr_in.Tk, Voll.tp_in.Tk, Volg.tp_in.Tk)

Cover_2Sh2Fl(nl, ng, nd, Awl, Awg, A_tot, Tvl, Tvg, Tf_Dome2, Tf_Dome3)
Cover_2Sh2Fl(nl, ng, nd, Awl, Awg, A_tot, hcl, hcg, hc_Dome2, hc_Dome3)
```

The *Cover_2Sh2Fl* (*Cover_4Sh2Fl*) functions (see Annex A6) consider the coverage of two moving fluid grids by means of two (four) arbitrarily dimensioned surface vectors:

COVER MATRIX CONCEPT: Mapping between the fluid node areas and the thermal wall areas

↓ **positive gravity**

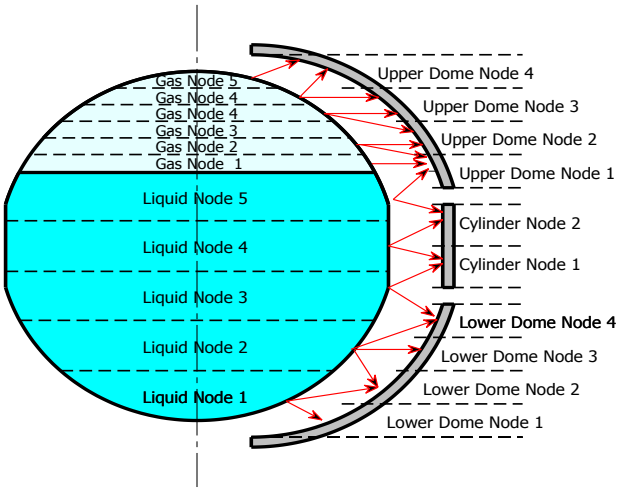


Figure 6-3: 1D TANKS. Fluid volume and wall discretization

Overflowing an outlet port

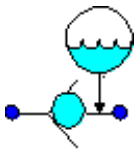
Present Tank models can simulate the liquid level passing through a fluid port in such a way that the exiting flow can be liquid or gas depending on the liquid level. See §6.2.1.4.

6.3.5 CheckValve_FlSep

6.3.5.1 Description

This component represents a fluid separator allowing the simulation of a common circuit pressurizing different tanks with different liquids. It can be used for the definition of two "separated" **real** fluids working in the same loop (the liquid in a tank and its pressuring gas, for example).

The component includes a non-return valve that prevents the contact of different kinds of vapors connected by a common pressurization system, see the example below.



6.3.5.2 Ports

Name	Type	Parameters	Direction	Description
f1	Fluid	-	IN	Inlet/outlet fluid ports
f2	Fluid	-	OUT	

6.3.5.3 Data

Name	Type	Description	Units
Ao	REAL	Junction area when fully open	m^2
dpmin	REAL	DP for complete opening. =0, `always opened`	Pa
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m
tao	REAL	Time constant of area change filter	s
zetaf	REAL	Forward loss coefficient	-
fluid	ENUM	Real fluid - downstream side only	
fluid_nc	ENUM	Pressuring gas - both sides	
RealPresGas	BOOLEAN	TRUE, pressuring gas real properties will be considered if an equivalent real fluid is available	

Note 1: The available fluids are described in §4.1.2.

Note 2: Only the following pressuring gases *He, O2, H2, N2 and CO2* will consider real fluid properties in the upstream circuit of this component if RealPresGas = TRUE.

Note 3: Use *dpmin* = 0 when no check valve is present in the circuit. If this is the case, the simulation will be valid if no significant reverse flow of vapors is produced at the point where this component is placed. Mixtures of two real fluids (fractional distillation) are not allowed.

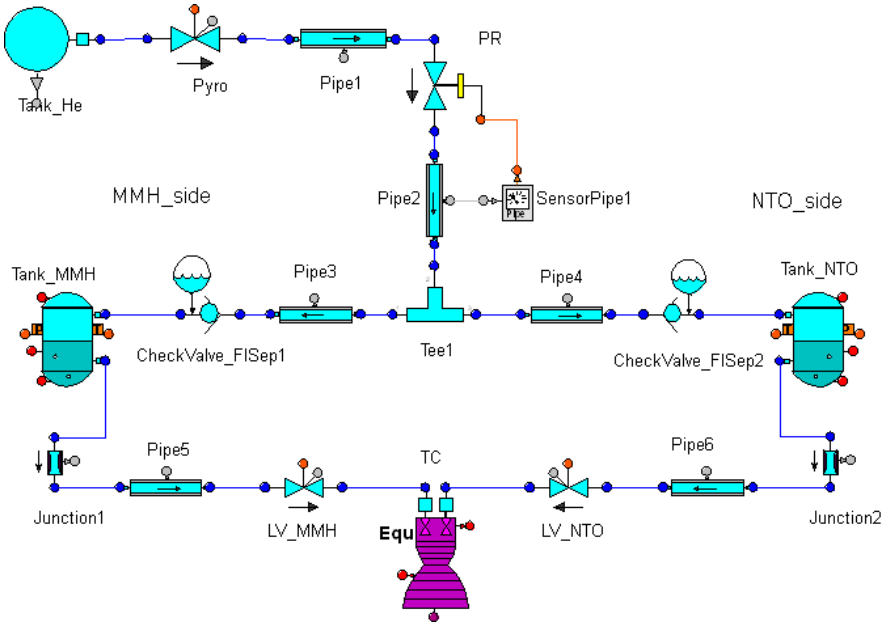
6.3.5.4 Formulation

This component performs the following functions:

- It prevents reverse flow (case of dpmin != 0) as the *ValveCheck* component does, see §5.4.17.5
- It is loaded with the definitions of the main fluid (vaporizing liquid downstream of the component) and the pressuring gas (both sides). See §5.4.24
- It checks that all possible interconnected CheckValve_FlSep components define only one pressuring gas. If not, the simulation will be stopped with a message

- It checks that all possible interconnected tanks at the liquid side have only one CheckValve_FlSep component connected to them. If not, the simulation will be stopped with a message

Upstream of this component, only one fluid, the pressuring gas, will work as the main fluid; the initial non-condensable mass fraction of the upstream components is not relevant (assumed to be zero). Downstream of this component both fluids will be considered, as in the *WorkingFluid* component, so it is important to correctly define the initial non-condensable mass fraction of downstream components. The example below shows how to use the CheckValve_FlSep component in a typical MMH/NTO engine.

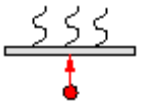


Note in this example that the common pressurization circuit can use real properties of Helium (see input flag "RealPresGas") which can be decisive in calculating the behavior of the Helium tank at high pressures and low temperatures where the Helium properties are not close to those of a perfect gas. The Joule-Thomson effect through the pressure regulator valve will be simulated.

6.3.6 Bound_QT_ext

6.3.6.1 Description

This component represents Ground & flight heat fluxes model.



6.3.6.2 Ports

Name	Type	Parameters	Direction	Description
tpr_out	THERMAL	(n = nz)	IN	Thermal outlet port to the tank wall (or insulation)

Nz is the number of axial (vertical) thermal nodes in the wall (or insulation) to which the Bound_QT_ext component is connected.

6.3.6.3 Data

Name	Type	Description	Units
Q_FS	TABLE_2D	Ground convective heat flux (kW/m^2)	kW/m^2
Q_FV	TABLE_2D	Flight convective heat flux (kW/m^2)	kW/m^2
Q_RS	TABLE_1D	Sun radiation heat flux (kW/m^2)	kW/m^2

Name	Type	Description	Units
Q_RT	TABLE_1D	Earth radiation heat flux (kW/m^2)	kW/m^2
S_ext	REAL	External surface (m^2)	m^2
T_RA	TABLE_1D	Sink temperature (K)	K
eps_a	REAL	Absorption coefficient	
eps_e	REAL	Emission coefficient	

Note 1: Heat fluxes are defined positive going to the Tank.

Note 2: It is assumed that the ground convective heat fluxes are only applied for negative times, and the flight convective heat fluxes for positive times, so the tables should be defined covering the corresponding times.

6.3.6.4 Formulation

This component performs the following functions:

- For each thermal node (nz), the ground and flight convective heat fluxes are interpolated in the respective tables (Q_FS, Q_FV) as a function of time (first independent variable) and wall temperature
- For each thermal node (nz), the sun and earth radiation heat fluxes are interpolated in the respective tables (Q_RS, Q_RT) as a function of the wall temperature
- Explicit radiative heat flux from the sink temperature (T_RA) to the wall tank will be calculated
- Each thermal node is assumed to have the same effective surface area (=S_ext/nz)

All heat fluxes are summed according to:

$$q_{wall,i} = S_{ext} / nz \left[\begin{array}{l} Q_{FS}(t, T_{wall,i}) + Q_{FV}(t, T_{wall,i}) + eps_a Q_{RS}(T_{wall,i}) + eps_a Q_{RT}(T_{wall,i}) + \\ eps_e \, 5.6696 \cdot 10^{-8} \, (T_{RA}^4(t) - T_{wall,i}^4) \end{array} \right]$$

where "t" is the current time and $Q_{FS}(t, T_{wall,i})$, ..., $T_{RA}(t)$ are the above-mentioned table interpolation functions. $T_{wall,i}$ is the external wall temperature at node i.

The wall temperature (or insulation temperature, depending on which solid this component is connected to) is calculated accommodating the calculated heat flux to the one calculated in the wall or insulation component. For example, if the solid is one of the 1D Tank or its insulations, an implicit calculation will be done iterating in Twall to adjust the heat flux by solid conduction to the expression above.

7. TURBO_MACHINERY LIBRARY

7.1 OVERVIEW

TURBO_MACHINERY is an ECOSIMPRO library for the *transient* simulation of pumps, turbines and compressors. Its most important features are the following:

- Pump model provided with user-defined dimensionless turbo-pump characteristic curves adapted to positive and negative speeds and flow zones.
- Turbine and Compressor components provided with user-defined dimensionless performance maps as a function of the reduced axial speed and pressure ratio.
- Special turbo machinery components allowing simple calculation of generalized performance maps as a function of the nominal performances and other significant design data.
- Programming of the turbo-machinery components is non-dependent on the working fluid type: The properties of the selected working fluid (calculated inside the corresponding thermodynamic function) will depend on the fluid, but the non-dimensional parameters of the performances map are equally defined for all kinds of fluids.

The generalized performance maps of the TURBO_MACHINERY library allow robustly analyzing the transients during the startup and shutting down processes, where the reduced axial speed and flow are far away from the nominal values.

The TURBO_MACHINERY components can be connected to FLUID_FLOW_1D components with the aim of simulate a complete rocket engine cycle. Rocket models where turbo machinery plays an important role will be easily evaluated using this Library.

7.1.1 Component classification

The components of the TURBO_MACHINERY Library are listed below. Two different types of pumps/turbines are available: one "generic" model (with a G on the symbol) if the off-design characteristics are unknown, and one specific model which can only be used with tables, for well-defined turbo machinery.

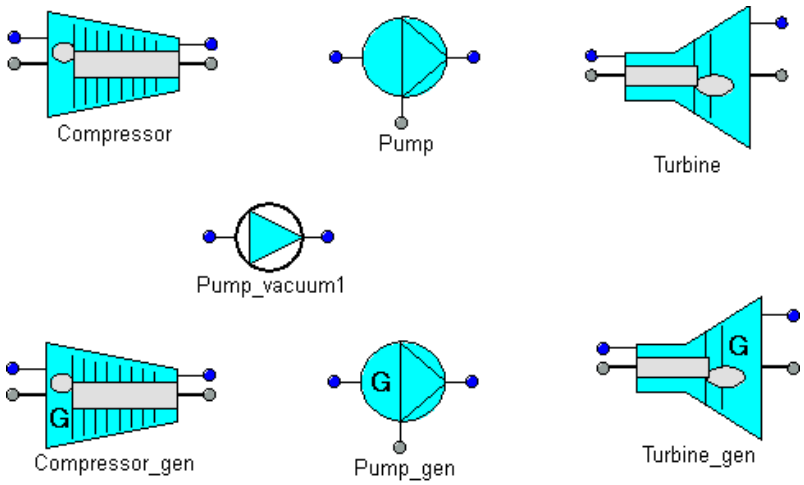


Figure 7-1 TURBO_MACHINERY palette of symbols

All these components behave externally as resistive elements (see §5.1.1). Component ports are resistive because they calculate the mass flow. Nevertheless, every model of a turbo-machinery component includes an internal capacitive element receiving/giving the mechanical work.

7.1.2 Building a model

TURBO_MACHINERY library is as an important complement of the FLUID_FLOW_1D library for the simulation of turbines, compressors and pumps. The rules for the fluid library will apply here. The following more specific recommendations and tips must also be considered:

- As for any other FLUID_FLOW_1D component turbines and compressors must be initialized with the **burnerGasesOption** parameter, which has two options:
  - *burnerGasesOption* = FLUID_PROPERTIES.*Chemicals* if the component is placed downstream of a combustor. Convection, mixing and properties of the chemical constituents are calculated dynamically. Working fluid is treated as a mixture of variable chemicals composition calculated with the Perfect gas or the Van der Walls state of equation.
  - *burnerGasesOption* = FLUID_PROPERTIES.*noBurnGases* if the component is **not** downstream of a combustor. Mixture of a (real) fluid with a non-condensable gas is considered.
- Turbo-machinery components marked with a "**G**" use "adjustable" generic maps. They include simple but physical general performance maps (see A8) that can be used for design purposes in a large range of conditions (positive and negative flow and speed regimes):
  - For Turbines and Compressors, choose the characteristic speed parameter, the nominal pressure ratio, efficiency, etc. according to the state of the art and the system to be modeled. Then, adjust the beta parameter (characteristic blade angle) to the actual speed triangle (dependent on the calculated mass flow). *Several iterations may be needed adjusting design parameters*.
  - For Pumps, use the specific speed parameter Ns to better adjust the performance maps dependent on the pump type (axial or centrifugal).
  - If more detailed performances than the generic maps contained in A8 are known, then it is always possible to modify the functions given in Appendix A8 with more accurate correlations.
- The rest of turbo-machinery components use specific performance maps introduced by the user. These maps should *cover the ranges explored by the simulation*. One typical problem in these cases is that the maps supplied by the builders may have different kinds of non-dimensional parameters, not always compatible with the ESPSS interface. In these cases, we recommend the following:
  - From the knowledge of the maps, perform a sensibility study on the original input parameters obtaining dimensioned performances in mass flow, torque, axial speed, pressure ratio, etc.
  - From these results, obtain the non-dimensional performances used by ESPSS (mass flow coefficient and specific torque, see §7.2.6.6 or head rise and reduced torque, see §7.2.4.6).
  - Migrating the previous results (extended as much as possible to large ranges of variation of the non-dimensional parameters) to the EcosimPro input data tables format.
- As a general rule, it is recommended to initialize a circuit with the same conditions as in the nearest inlet boundary. Simulating engine startup, the external turbine temperature should prevail with respect to the feeding turbine conditions. Global data variables (pressure and temperature) are useful to initialize the same conditions in different components.
- The internal volume of the turbo-machinery components can be used to calculate the temperature variation of the working fluid circulating by the turbomachinery, even during the phases with no flow but with a permanent torque applied to the pump or compressor component.
- Turbines, Compressors and Pumps with "adjustable" maps can use "Design" option (see the input data description). Activate this option only for pure fluids, not for burned gases, because the model will not know the fluid properties at the initial conditions, when the rated conditions will be applied.
- *The turbines cannot work with liquids. There is no model for hydraulic turbines, but might be modelled with a pump working with reverse flow.*

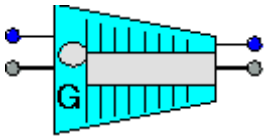
The examples in reference RD-5 are a template for more complicated models.

7.2 OPERATIONAL COMPONENTS

7.2.1 Compressor_gen

7.2.1.1 Description

This component simulates a generic compressor. It is provided with *calculated* (no input) but adjustable dimensionless characteristic curves adapted to positive and negative speeds and flow zones. Adjustable dimensionless pressure ratio curves are a function of: *beta design parameter* (see the input data variables), reduced speed and inlet Mach number. Efficiency curves are a function of: reduced speed and pressure ratio parameters.



7.2.1.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids
rpm_opt	ENUM	Two option: fixed or dynamic (see note 2)

Note 1: Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

Note 2: If "*rpm_opt*" = fixed, then the dynamic equation for the axial speed will be deactivated assuming constant axial speed (= rpm_cte) and the axial torque will be an output; if "rpm_opt" = dynamic, the axial speed will be calculated dynamically.

7.2.1.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet / Outlet fluid ports
f2	fluid	burnerGasesOption	OUT	
sh_in	MECHANICAL.shaft		IN	Inlet shaft port
sh_out	MECHANICAL.shaft		OUT	Outlet shaft port

7.2.1.4 Data

Name	Type	Description	Units
Ain	REAL	Characteristic Inter-blade flow area	m^2
Aout	REAL	Outlet flow area	m^2
beta	REAL	Characteristic blade-axe angle	degrees
R	REAL	Characteristic blade-tip radius	m
I	REAL	Mobile parts inertia	kg. m^2
N_nom	REAL	Nominal characteristic speed	-
PI_nom	REAL	Nominal pressure ratio	-
eta_nom	REAL	Nominal efficiency	-
P_o	REAL See §5.1.3.3	Initial Pressure	Pa
T_o		Initial temperature	K
x_nco		Initial non-condensable mass fraction	-
vsound_outlet	BOOLEAN	TRUE, sound speed calculated at outlet, FALSE, at inlet	-
x	REAL	X coordinate relative to a body axis system	m
y	REAL	Y coordinate relative to a body axis system	m
z	REAL	Elevation relative to a body axis system	m
design	ENUM	=active, rated conditions are used (see note2)	-
m_op	REAL	Rated mass flow	(kg/s)

Name	Type	Description	Units
rpm_op	REAL	Rated axial speed	(rpm)
Pin_op	REAL	Rated inlet pressure	(Pa)
Tin_op	REAL	Rated inlet temperature	(K)
Tout_op	REAL	Rated outlet temperature	(K)
rpm_cte	REAL	Constant axial speed (see note on construction parameter)	(rpm)

Note 1: *vsound_outlet* =FALSE is not valid with two fluid mixtures

Note 2: If "*design*" = non active, the rated conditions will not be used and the original R, eta and Ain will be used. If "design" = active, then the rated conditions will be used to calculate the geometrical design (operational tip radius, inter-blade flow area and efficiency. *Activate this option **only for pure fluids**, not for burned gases, because the model will not know the fluid properties at the initial conditions, when the rated conditions will be applied.*):

```
h_in  = FL_prop_vs_pT(f1.fluid, Pin_op, Tin_op, fprop_enthalpy,ier)
rho_in= FL_prop_vs_pT(f1.fluid, Pin_op, Tin_op, fprop_density, ier)
s_in  = FL_prop_vs_pT(f1.fluid, Pin_op, Tin_op, fprop_entropy, ier)
h_out = FL_prop_vs_pT(f1.fluid, Pin_op*PI_nom, Tout_op, fprop_enthalpy, ier)
H_ise = FL_prop_vs_ps(f1.fluid, Pin_op*PI_nom, s_in, fprop_enthalpy, ier)

eta_op = (h_in -H_ise) / (h_in-h_out)
R_op   = N_nom * vsound / (2*PI/60*rpm_op)
Ain_op = m_op *tan(PI*beta/180) /vsound /N_nom /rho_in
```

7.2.1.5 Topology

This component is built topologically with a FLUID_FLOW_1D Volume component receiving the head and the mechanical work produced in the mobile parts, and a Junction simulating the outlet.

The following sections describe the equations for calculating the pressure drop and the mechanical work assumed to be applied to the inlet of the compressor volume.

7.2.1.6 Compressor Power

The value of the power *W* is obtained using the "*GasTurbo_pow*" function (see Appendix A8.3).

This function basically estimates efficiency as a function of the reduced speed and pressure ratio parameters. It also needs the inlet mass flow (see Inlet Mass flow Equation below) to calculate the power. The rest of the arguments are input data (eta_nom, N_nom,…) or calculated values in the connected pipes components (bound pressures and enthalpies, etc.).

7.2.1.7 Mechanical Balance

The mechanical balance is used to calculate the axial speed dynamically:

$$I_{mech} \cdot \omega^{'} = T_{shaft,port} - T$$

where:

ω = mechanical speed (rad/s)
I_{mech} = Compressor mechanical inertia (kg m2)
T = Torque, calculated as W/ω

7.2.1.8 Energy Conservation Equation

The mechanical work (T* ω) extracted from the compressor is simulated as an enthalpy flow.

Compressor.Volume.mh = Inlet_Compressor.mh - T*ω

where the variable *mh* is the enthalpy flow.

7.2.1.9 Inlet Mass flow Equation:

The inlet mass flow equation is expressed dynamically in accordance with the compressor pressure rise as follows:

$$I \cdot m_{in}' = (P + 0.5\rho v^2)_{Compr,\,inlet} - (P + 0.5\rho v^2)_{Compr,\,vol} + (\Pi_{nom} - 1)P_{in} \cdot dp\_rel$$

where:

m_{in} = inlet mass flow
P= static pressures
I = fluid inertia (1/m)

dp_rel is obtained using "*GasTurbo_dpComp*" function (see Appendix A8.2). This function basically estimates the relative pressure drop of the compressor as a function of the *beta design parameter*, the reduced speed and the inlet Mach number.

The inlet Mach number Min is related to the inlet mass flow:

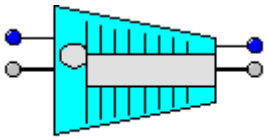
$$M_{in} = \frac{m_{in}}{\rho_{in} A_{in} c}$$

c is the sound speed calculated at the outlet volume or at inlet according to the "*vsound_outlet*" option

7.2.2 Compressor

7.2.2.1 Description

This component simulates a compressor for gases. Compressor map curves (dimensionless torque and mass flow coefficients) are input data tables depending on the dimensionless speed and pressure ratio coefficients.



7.2.2.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

7.2.2.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet / Outlet fluid ports
f2	fluid	burnerGasesOption	OUT	
sh_in	MECHANICAL.shaft		IN	Inlet shaft port
sh_out	MECHANICAL.shaft		OUT	Outlet shaft port

7.2.2.4 Data

Name	Type	Description	Units
Aout	REAL	Outlet flow area	m^2
R	REAL	Characteristic blade-tip radius	m
I	REAL	Mobile parts inertia	kg. m^2

Name	Type	Description	Units
P_o	REAL	Initial Pressure	Pa
T_o		Initial temperature	K
x_nco	See §5.1.3.3	Initial non-condensable mass fraction	-
vsound_outlet	BOOLEAN	TRUE, sound speed calculated at outlet, FALSE, at inlet	-
x	REAL	X coordinate relative to a body axis system	m
y	REAL	Y coordinate relative to a body axis system	m
z	REAL	Elevation relative to a body axis system	m
ST_vs_N_PI	TABLE 2D	Table of non-dimensional torque vs N and PI	-
Qplus_vs_N_PI	TABLE 2D	Table of non-dimensional mass flow vs N and PI	-

Note: *vsound_outlet*=FALSE is not valid with two fluid mixtures

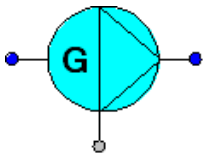
7.2.2.5 Topology & formulation

This component is built topologically with a FLUID_FLOW_1D Volume component receiving the head and the mechanical work, and a Junction simulating the outlet. The formulation is very similar to the Turbine component (see §7.2.6.6 and following), but changing the sign of the Torque. The default maps supplied are adapted for compressors.

7.2.3 Pump_gen

7.2.3.1 Description

This component simulates a pump for liquids. It is provided with fixed or user-defined dimensionless turbo-pump characteristic curves adapted to positive and negative speeds and flow zones, and valid for several types of pumps.



7.2.3.2 Construction Parameters

Name	Type	Description
rpm_opt	ENUM	Two option: fixed or dynamic (see note 3)
comp_cav	ENUM	Flag to consider fluid compressibility under cavitation conditions at inlet.

Note: If "*rpm_opt*" = fixed, then the dynamic equation for the axial speed will be deactivated assuming constant axial speed (= rpm_cte) and the axial torque will be an output; if "rpm_opt" = dynamic, the axial speed will be calculated dynamically.

7.2.3.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	noBurnGases	OUT	Inlet / Outlet fluid ports
f2	fluid	noBurnGases	OUT	
sh_in	MECHANICAL.shaft		IN	Shaft port

7.2.3.4 Data

Name	Type	Description	Units
I	REAL	Pump inertia	kg.m^2
Ns	REAL	Pump specific speed	
P_o	REAL	Initial Pressure	Pa
T_o	See	Initial temperature	K
x_nco	§5.1.3.3	Initial non-condensable mass fraction	-
m_o	REAL	Initial mass flow	Kg/s

Name	Type	Description	Units
rpm_cte	REAL	Constant axial speed	rpm
rpm_o	REAL	Initial pump speed	rpm
rpm_r	REAL	Pump speed at rated conditions	rpm
tdh_r	REAL	Total dynamic head (at rated conditions)	m
Q_r	REAL	Volume flow (at rated conditions)	m^3/s
eff_r	REAL	Efficiency at rated conditions	-
design	ENUM	=active, rated conditions are used (see note 4)	-
Pin_op	REAL	Rated inlet pressure	Pa
Pw_op	REAL	Rated pump power	W
Tin_op	REAL	Rated inlet temperature	K
tdh_correction	BOOLEAN	If TRUE, the pump head is modified to account for high compressible liquids	
x	REAL	X coordinate relative to a body axis system	m
y	REAL	Y coordinate relative to a body axis system	m
z	REAL	Elevation relative to a body axis system	m
User_Curves	BOOLEAN	TRUE: user defined map curves (see note 1)	
wh_vs_theta_r	TABLE_2D	Dimensionless head curves defined by the user.	
wbeta_vs_theta_r	TABLE_2D	Dimensionless torque curves defined by the user.	
NPSH_dyn	BOOLEAN	FALSE, NPSH accounts only for static pressure (see note 2)	
dtdh_vs_NPSH_Q	TABLE_2D	Table of loss of head vs NPSH_m and flow_m3_s	%
kappa_cav_vs_NPSH	TABLE_1D	Compressibility due to pump cavitation (see note 3)	1/Pa

Note 1: User_Curves = TRUE, then pump curves are defined by the user. FALSE, default performances will be used according to RD-46, Chapter 7.2, see §7.2.3.6

Note 2: NPSH_dyn = FALSE means that the NPSH has **not** been calculated with the usual formulation (including dynamic pressure at the inducer inlet eye), but with an alternative one considering only the static pressure upstream of the inducer

Note 3: "comp_cav" = active (see the construction parameters) means that, under cavitation conditions at pump inlet, there will be an extra compressibility of the liquid fluid circulating through the pump:

$$V' = V\, k_{cav} P',$$

where K_{cav} is the interpolated value in the input data table "kappa_cav_vs_NPSH"

Note 4: Activate "*design* = *active*" to calculate rated efficiency from the operational conditions Pin_op , Tin_op and Pw_op. If "*design* = *not_active*", these inuts will not be used but the *eff_r* input data.

7.2.3.5 Topology

The Pump component is built topologically with a FLUID_FLOW_1D Volume component receiving the head and the mechanical work, and a Junction simulating the outlet. The sections below describe the equations used to calculate the head and the mechanical work of the pump assumed to be applied to the inlet of the pump volume.

7.2.3.6 Pump Performances

Generalized Pump Performances

It is usually rather difficult to find pump curves that include the non-normal zones, which is why this component has a set of curves covering all zones of pump operation for 3 different specific speeds: Ns = 25 corresponding to a pure centrifugal pump, Ns=147 corresponding to mixed pump, and Ns=261 corresponding to an axial pump. The specific speed is defined as:

$$Ns = \frac{rpm\sqrt{Q / n_s}}{(TDH / n_{st})^{0.75}}$$

where:

n_s = number of suctions
n_{st} = number of stages
Q = volumetric flow (m3/s)
TDH = Actual total dynamic head of the pump (m)

The pump curves (head and torque) are introduced by means of fixed 2D data tables named "wh_vs_theta_Ns" and "wt_vs_theta_Ns" defined as functions of the specific speed and a dimensionless variable θ that *preserves homologous relationships in all zones of operation*. The θ parameter is:

$$\theta = \pi + \tan^{-1}(v/n)$$

"v" and "n" are the reduced flow and speed variables:

$$v = Q/Q_R = m_{in}/\rho_{in}/Q_R; \qquad Q_R : \text{Nominal } volumetric\ flow$$
$$n = 30\omega/\pi/rpm_R; \qquad rpm_R : \text{Nominal } speed\ (rpm)$$

where:

m_{in} = inlet mass flow (see §7.2.3.9)
ρ= liquid density
ω = rotational speed (see §7.2.3.7)

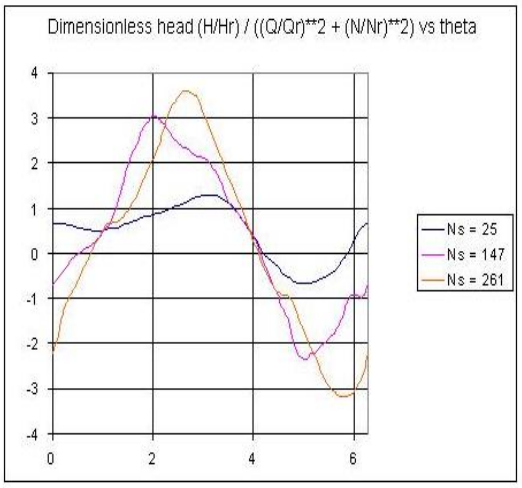
The dimensionless characteristics (head and torque) are defined as follows:

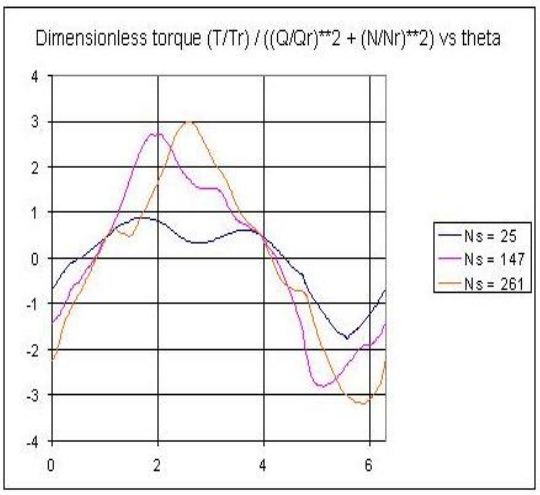
$$\text{wh} = \frac{TDH}{TDH_R(n^2+v^2)}; \qquad \text{wt} = \frac{T}{T_R(n^2+v^2)}$$

T and TDH are the torque and the total dynamic head respectively. Sub index R means "rated" (nominal) conditions (input data). The nominal torque is calculated from the other nominal parameters:

$$T_R = \frac{9{,}806\rho_{up}TDH_RQ_R}{eff_R rmp_R 2\pi/60}$$

The figures below show the referred pump dimensionless characteristic as a function of the specific speed Ns andθ. They are taken from RD-46 (Chapter 7.2, *Dimensionless-Homologous Turbopunp Characteristics*):





The dimensionless performances will be calculated interpolating in these curves for the actual values of the specific speed Ns and θ :

wh = wh_vs_theta_Ns (Ns, θ) (dimensionless pump TDH)

wt = wt_vs_theta_Ns (Ns, θ) (dimensionless pump torque)

Using the definition of the wh, wt, the actual torque T and the pressure rise TDH (expressed as the total dynamic head in meters) will be calculated.

User-defined Pump performances

In this case, the dimensionless head and torque values (wh, wt) are interpolated in user defined tables, wh_vs_theta_r and wbeta_vs_theta_r.

The same procedures as before will be used, with the difference that the pump performances will depend not only on the "θ" variable (angular distance coordinate), but on a new independent parameter, r, the polar distance coordinate, defined as follows:

$$r = \sqrt{v^2 + n^2})$$

7.2.3.7 Mechanical Balance

The mechanical balance is used to calculate the axial speed dynamically:

$$I_{mech} \cdot \omega' = T_{shaft,port} - T$$

where:

ω = mechanical speed (rad/s)
I_{mech} = Pump mechanical inertia (kg m2)
T = Torque calculated using the non-dimensional pump performances

7.2.3.8 Energy Conservation Equation

The mechanical work (Torque*w) received by the pump volume is simulated as an enthalpy flow.

Pump.Volume.mh = Inlet_Pump.mh + T*omega

where the variable *mh* is the enthalpy flow. It must be emphasized that this formulation with an internal volume receiving the work is used to simulate the fluid heating in every situation, even when the pump works without flow (null efficiency).

The current efficiency in off-design conditions can be calculated accordingly with the definition of the nominal torque (isentropic enthalpy rise assumed to be the pump head, $9.806 \cdot TDH = (h_{is} - h_{in})$). Then,

$$\eta = m\, 9.806\, TDH \,/\, T\omega$$

The "*tdh_correction*" option accounts for high compressible liquids modifying the effective outlet pressure:

$$\begin{aligned} P_{out} &= P_{in} + \rho_{in} \cdot 9.806 \cdot TDH && (\text{tdh\_correction} = \text{FALSE}) \\ &= P_{vs\_hs}(h_{in} + 9.806 \cdot TDH, s_{is}) && (\text{tdh\_correction} = \text{TRUE}) \end{aligned}$$

$$s_{is} = s_{vs\_ph}(P_{in}, h_{in})$$

where "P_{vs_hs}" and "s_{vs_ph}" are thermodynamic functions.

7.2.3.9 Inlet Mass flow Equation:

The inlet mass flow equation is expressed dynamically in accordance with the pump pressure rise as follows:

$$I \cdot m_{in}' = P_{out} - (P + 0.5\rho v^2)_{Pump,vol} + 9.806 \cdot \rho_{in} \cdot dtdh\_cav(NPSH, Q)$$

where:

m_{in} = inlet mass flow
P= static pressure.Pout is the off-design total outlet pressure in steady conditions, see above.
ρ = density

I = Pump inlet fluid inertia (1/m)
TDH = total dynamic head, in meters
dtdh_cav = interpolated value in the data table "dtdh_vs_NPSH_Q"

The NPSH is calculated in meters:

NPSH = (Pin+0.5ρ_{in}vel^2 – Psat(Tin)) / (ρ_{in}9.806); NPSH_dyn = TRUE

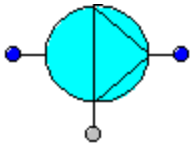
NPSH = (Pin – Psat(Tin)) / (ρ_{in}9.806); NPSH_dyn = FALSE

Note that the inlet pressure is the static value calculated in the capacitive component placed upstream the pump, typically a pipe. The lower value of the inlet flow area (pipe diameter), the lower Pin.

7.2.4 Pump

7.2.4.1 Description

This component simulates a pump for liquids. Turbo-pump map curves (torque and head rise) are input data tables depending only on the reduced mass flow.



7.2.4.2 Construction Parameters

Name	Type	Description
comp_cav	ENUM	Flag to consider fluid compressibility under cavitation conditions at inlet.

7.2.4.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	noBurnGases	OUT	Inlet / Outlet fluid ports
f2	fluid	noBurnGases	OUT	
sh_in	MECHANICAL.shaft		IN	Shaft port

7.2.4.4 Data

Name	Type	Description	Units
I	REAL	Pump inertia	kg.m^2
P_o	REAL	Initial Pressure	Pa
T_o		Initial temperature	K
x_nco	See §5.1.3.3	Initial non-condensable mass fraction	-
m_o	REAL	Initial mass flow	Kg/s
rpm_o	REAL	Initial pump speed	rpm
psi_vs_phi	TABLE	Dp coeff.: Psi+ = f(Phi+ (m^3))	m^2
Cplus_vs_phi	TABLE	Torque coeff.: C+ = f(Phi+ (m^3))	m^5
x	REAL	X coordinate relative to a body axis system	m
y	REAL	Y coordinate relative to a body axis system	m
z	REAL	Elevation relative to a body axis system	m
NPSH_dyn	BOOLEAN	FALSE, NPSH accounts only for static pressure (see note 1)	
dtdh_vs_NPSH_Q	TABLE_2D	Table of loss of head vs NPSH_m and flow_m3_s	%
kappa_cav_vs_NPSH	TABLE_1D	Compressibility due to pump cavitation (see note 1)	1/Pa

Note 1: NPSH_dyn = FALSE means that the NPSH has **not** been calculated with the usual formulation (including dynamic pressure at the inducer inlet eye), but with an alternative one considering only the static pressure upstream of the inducer.

Note 2: Parameter "comp_cav" = active (see the construction parameters) means that, under cavitation conditions at pump inlet, there will be an extra compressibility of the liquid fluid circulating by the pump. This will produce a mass flow discontinuity between the inlet and the outlet because of the change in the fluid volume:

$$V' = V\, k_{cav} P',$$

where K_{cav} is the interpolated value in the input data table "kappa_cav_vs_NPSH"

7.2.4.5 Topology

The Pump component is built topologically with a FLUID_FLOW_1D Volume component receiving the head and the mechanical work, and a Junction simulating the outlet. The sections below describe the equations used to calculate the head and the mechanical work of the pump assumed to be applied to the inlet of the pump volume.

7.2.4.6 Pump Performances

The pump curves (head and torque) are entered by means of input data tables:

Independent variable:

- Mass flow coefficient: $\varphi^+ = m/(\rho_{in}\omega)$ (m3)

Dependent variables:

- Head rise coefficient: $\psi^+ = \Delta P/(\rho_{in}\omega^2)$ (m2)
- Reduced torque: $C^+ = T/(\rho_{in}\omega^2)$ (m5)

where:

m_{in} = inlet mass flow
ρ= outlet density
T= Consumed torque
ω = rotational speed

The head rise and needed torque coefficients are computed with 1-D tables, as a function of mass flow coefficient only. Rotational speed is not taken into account. The pump model described here has the disadvantage of being less general than the *Pump_gen* component: the coefficients used (φ^+, ψ^+ and C^+) are not dimensionless. Hence, the characteristics differ for each pump, even for geometrically similar pumps (with impellers having the same angles and proportions).

7.2.4.7 Mechanical Balance, NPSH Calculation & Conservation Equations

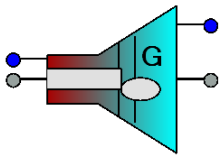
Same formulation as in the *Pump_gen* component. See §7.2.3.7 and following.

7.2.5 Turbine_gen

7.2.5.1 Description

This component simulates a turbine for gases. It is provided with *calculated* (no input) but adjustable dimensionless characteristic curves adapted to positive and negative speeds and flow zones, and valid for several types of turbines.

Adjustable dimensionless pressure ratio curves are a function of: *beta design parameter* (see the input data variables), reduced speed and inlet Mach number. Efficiency curves are a function of: reduced speed and pressure ratio parameters.



7.2.5.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids
rpm_opt	ENUM	Two option: fixed or dynamic (see note 3)

Note 1: Select *burnerGasesOption = Chemicals* only for components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

Note 2: If "*rpm_opt*" = fixed, then the dynamic equation for the axial speed will be deactivated assuming constant axial speed (= rpm_cte) and the axial torque will be an output; if "rpm_opt" = dynamic, the axial speed will be calculated dynamically.

7.2.5.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet / Outlet fluid ports
f2	fluid	burnerGasesOption	OUT	
sh_in	MECHANICAL.shaft		IN	Inlet shaft port
sh_out	MECHANICAL.shaft		OUT	Outlet shaft port

7.2.5.4 Data

Name	Type	Description	Units
Ain	REAL	Characteristic Inter-blade flow area	m^2
Aout	REAL	Outlet flow area	m^2
beta	REAL	Characteristic blade-axe angle	degrees
R	REAL	Characteristic blade-tip radius	m
I	REAL	Mobile parts inertia	kg. m^2
N_nom	REAL	Nominal characteristic speed	-
PI_nom	REAL	Nominal pressure ratio	-
eta_nom	REAL	Nominal efficiency	-
P_o	REAL See §5.1.3.3	Initial Pressure	Pa
T_o		Initial temperature	K
x_nco		Initial non-condensable mass fraction	-
vsound_outlet	BOOLEAN	TRUE, sound speed calculated at outlet, FALSE, at inlet	-
x	REAL	X coordinate relative to a body axis system	m
y	REAL	Y coordinate relative to a body axis system	m
z	REAL	Elevation relative to a body axis system	m
design	ENUM	=active, rated conditions are used (see note2)	-
m_op	REAL	Rated mass flow	(kg/s)
rpm_op	REAL	Rated axial speed	(rpm)
Pin_op	REAL	Rated inlet pressure	(Pa)
Tin_op	REAL	Rated inlet temperature	(K)
Tout_op	REAL	Rated outlet temperature	(K)
rpm_cte	REAL	Constant axial speed (see note on construction parameter)	(rpm)

Note 1: *vsound_outlet* =FALSE is not valid with two fluid mixtures

Note 2: If "*design*" = non active, the rated conditions will not be used and the original R, eta and Ain will be used. If "design" = active, then the rated conditions will be used to calculate the geometrical design (operational tip radius, inter-blade flow area and efficiency). *Activate this option* ***only for pure fluids,***

not for burned gases, because the model will not know the fluid properties at the initial conditions, when the rated conditions will be applied.:

```
h_in  = FL_prop_vs_pT(f1.fluid, Pin_op, Tin_op, fprop_enthalpy,ier)
rho_in= FL_prop_vs_pT(f1.fluid, Pin_op, Tin_op, fprop_density, ier)
s_in  = FL_prop_vs_pT(f1.fluid, Pin_op, Tin_op, fprop_entropy, ier)
h_out = FL_prop_vs_pT(f1.fluid, Pin_op/PI_nom, Tout_op, fprop_enthalpy,ier)
H_ise = FL_prop_vs_ps(f1.fluid, Pin_op/PI_nom, s_in, fprop_enthalpy, ier)
eta_op = (h_in-h_out) / (h_in -H_ise)
R_op = N_nom * vsound / (2*PI/60*rpm_op)
Ain_op = m_op *tan(PI*beta/180) /vsound /N_nom /rho_in
```

This option ("design" = active) is only available with pure fluids

7.2.5.5 Topology

This component is built topologically with a FLUID_FLOW_1D Volume component receiving the pressure and mechanical drops produced in the mobile parts, and a Junction simulating the outlet. The following sections describe the equations used to calculate the pressure drop and the mechanical work assumed to be applied to the inlet of the turbine volume.

7.2.5.6 Turbine Power

The value of the power *W* is obtained using "*GasTurbo_pow*" function (see Appendix A8.3).

This function basically estimates the efficiency as a function of the reduced speed and pressure ratio parameters. It also needs the inlet mass flow (see Inlet Mass Flow Equation below) to calculate the power. The rest of the arguments are input data (eta_nom, N_nom,…) or calculated values in the connected pipe components (bound pressures and enthalpies, etc.).

7.2.5.7 Mechanical Balance

The mechanical balance is used to calculate the axial speed dynamically:

$$I_{mech} \cdot \omega' = T_{shaft,port} - T$$

where:

ω = mechanical speed (rad/s)
I_{mech} = Turbine mechanical inertia (kg m2)
T = Torque, calculated as W/ω

7.2.5.8 Energy Conservation Equation

The mechanical work (T* ω) extracted from the turbine is simulated as an enthalpy flow.

Turbine.Volume.mh = Inlet_Turbine.mh - T* ω

where the variable mh is the enthalpy flow.

7.2.5.9 Inlet Mass flow Equation:

The inlet mass flow equation is expressed dynamically in accordance with the turbine pressure drop as follows:

$$I \cdot m_{in}' = (P + 0.5\rho v^2)_{Turb,inlet} - (P + 0.5\rho v^2)_{Turb,vol} - (\Pi_{nom} - 1)P_{in} \cdot dp\_rel$$

where:

m_{in} = inlet mass flow
P= static pressures
I = fluid inertia (1/m)

dp_rel is obtained using the "*GasTurbo_dpTurb*" function (see Appendix A8.1). This function basically estimates the relative pressure drop of the turbine as a function of the *beta design parameter*, the reduced speed and the inlet Mach number.

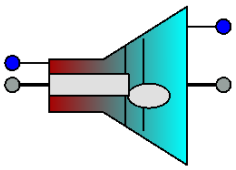
The inlet Mach number Min is related to the inlet mass flow: $M_{in} = \frac{m_{in}}{\rho_{in} A_{in} c}$

c is the sound speed calculated at the outlet volume or at inlet according to the "*vsound_outlet*" option.

7.2.6 Turbine

7.2.6.1 Description

This component simulates a turbine for gases. Turbine map curves (dimensionless torque and mass flow coefficients) are input data tables depending on the dimensionless speed and pressure ratio coefficients.



7.2.6.2 Construction Parameters

Name	Type	Description
burnerGasesOption	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* onlyfor components placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

7.2.6.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	OUT	Inlet / Outlet fluid ports
f2	fluid	burnerGasesOption	OUT	
sh_in	MECHANICAL.shaft		IN	Inlet shaft port
sh_out	MECHANICAL.shaft		OUT	Outlet shaft port

7.2.6.4 Data

Name	Type	Description	Units
Aout	REAL	Outlet flow area	m^2
R	REAL	Characteristic blade-tip radius	m
I	REAL	Mobile parts inertia	kg. m^2
P_o	REAL	Initial Pressure	Pa
T_o		Initial temperature	K
x_nco	See §5.1.3.3	Initial non-condensable mass fraction	-
vsound_outlet	BOOLEAN	TRUE, sound speed calculated at outlet, FALSE, at inlet	-
x	REAL	X coordinate relative to a body axis system	m
y	REAL	Y coordinate relative to a body axis system	m
z	REAL	Elevation relative to a body axis system	m
ST_vs_N_PI	TABLE 2D	Table of non-dimensional torque vs N and PI	-
Qplus_vs_N_PI	TABLE 2D	Table of non-dimensional mass flow vs N and PI	-

Note: vsound_outlet =FALSE is not valid with two-fluid mixtures

7.2.6.5 Topology

This component is built topologically with a FLUID_FLOW_1D Volume component receiving the pressure and mechanical drops produced in the moving parts, and a Junction simulating the outlet.

The sections below describe the equations used to calculate the pressure drop and the mechanical work assumed to be applied to the inlet of the turbine volume.

7.2.6.6 Turbine Performances

Performance maps (mass flow coefficient and specific torque) are entered by means of 2D input data tables:

Independent variables:

- Speed coefficient: $N = \frac{r \cdot \omega}{v_{son}}$
- Total pressure ratio: $\Pi = P_{o1} / P_{o2}$

Dependent variables:

- Mass flow coefficient: $Q^{+} = m_{map} \cdot v_{son} /(r^2 P_{o1})$
- Specific torque: $ST = T /(r \ m_{map} \cdot v_{son})$

where:

m_{map} = mass flow
v_{son}= sound speed
T= Consumed torque
ω = rotational speed

Independent variables are easily calculated because they are obtained from the dynamic rotational speed and the boundary pressures. Then, the mass flow and torque will be computed by interpolating into the 2D input data tables representing the turbine maps.

Therefore, for a good function in transients, data must be provided for a wide range of speed coefficients (from 0 to 2 is a good choice) and pressure ratios (for instance, from 1 to 3 for a nominal pressure ratio of 2). The turbine model is very general because the parameters are scaled with the mean radius and the speed of sound. Therefore, the given turbine characteristics remain valid for geometrically similar turbines, and for different fluids (in first order approximation).

7.2.6.7 Mechanical Balance

The mechanical balance allows calculation of the axial speed dynamically:

$$I_{mech} \cdot \omega^{'} = T_{shaft,port} - T$$

where:

ω = mechanical speed (rad/s)
I_{mech} = Turbine mechanical inertia (kg m2)
T = Torque

7.2.6.8 Energy Conservation Equation

The mechanical work (*Torque*w*) extracted from the turbine is simulated as an enthalpy flow.

Turbine.Volume.mh = Inlet_ Turbine.mh - T*omega

where the variable *mh* is the enthalpy flow. It must be emphasized that this formulation with an internal volume allows the simulation of the fluid temperature drop through the turbine.

7.2.6.9 Inlet Mass flow Equation:

The inlet mass flow equation is expressed dynamically in accordance with a delay time (inertia terms):

$$\tau \cdot m_{in}{}' = (m_{map} - m_{in}); \qquad \tau = I \cdot r^2 / v_{son}$$

where:

m_{in} = inlet mass flow
P= static pressures
I = fluid inertia (1/m)

8. COMB_CHAMBERS LIBRARY

8.1 OVERVIEW

COMB_CHAMBERS is an ECOSIMPRO library for the *transient* simulation of liquid, solid and hybrid rocket engines. The most important features are the following:

- The properties of the combustion gases (transport and heat capacity) are obtained from the CEA coefficients (see §4.4.1) for an arbitrary mixture of chemical reactants. The equilibrium molar fractions of a mixture of reactants are derived from the Minimum Gibbs Energy Method.
- Non adiabatic 1D combustor components: the equilibrium combustion gases are calculated using previous capabilities. The chamber conditions will be derived from the general transient conservation equations along a 1D axial discretization.
- Inclusion of optional models for the calculation of non-equilibrium combustors and for the liquid droplets evaporation. The non-equilibrium model does not include finite rate chemistry but only time-delay parameters.
- Ideal or non-adiabatic exhaust nozzles provided with a 1D axial discretization. A special nozzle component allows simple simulation of the film cooling injection.
- Solid/hybrid combustor components including an evaporation model for the solid (and liquids) propellants together with a 1D reaction delay model for the combustor core.
- Ramjet/scramjet combustor components including a supersonic intake, shock capture, a 1D model for the liquid fuel evaporation together with a 1D reaction delay model.
- Pre-burners and main thrusters are topologically built by means of a combustor, two injectors with its cavities and a nozzle for the main thrusters. They can work either under liquid, two-phase or gas injection conditions. The bubble collapse calculation is included in the model of cavities.
- The combustion gases generated in a chamber are *transported* (using standard FLUID_FLOW_1D components) to the turbines or to other chambers where any of the previously combusted gases will be considered a new reactant. *Convection, mixing and properties of the combusted gases chemical constituents are calculated dynamically in the transport equations of the ESPSS components)*.
- Cooling jacket component: Several models are available: one with a complete 3D wall temperature distribution and another including the injection torus.
- Modeling of solid propellant starters, igniters and thermal coating protection are available using combustor components.

The COMB_CHAMBERS components can be connected to FLUID_FLOW_1D, TANKS or TURBO_MACHINERY components for the simulation of a rocket engine cycle. Models with one or more chambers (staged engines) can be evaluated. This library is designed to give a detailed analysis of the transients during the startup and shut down processes, where the valve sequences are decisive.

The numeric method used for the resolution of the subsonic (*or supersonic in case of a scramjet combustor*) sections of a combustor is based on the *transient* conservation equations**.** Combustor models (we are referring now to the transient models, the STEADY library will perform similar calculations under steady conditions for design purposes, see Chapter 11) calculate the chamber pressure and the mixture ratio as a function of the combustor geometry, chemistry and the physical boundaries whereas in CEA code Pc and MR were imposed. The pros of the transient approach are:

- Transient phenomena (including pressure/temperature peaks at the startup and shutdown processes) are taken into account.
- The number of implicit equations is reduced, the state variables being dynamic.
- The wall heat exchange (non-adiabatic terms) and the pressure drops are taken into account.
- Transient formulation allows including vaporization / non-equilibrium phenomena.

The cons of this method are:

- The characteristic time (integration time step) can be very low. Nevertheless, numerical instabilities are normally smoothed.
- The total pressure is not strictly conserved along the 1D combustor volumes. Typical errors are between 0.5 and 1%, mainly produced near the throat where the Mach number is close to 1.

Concerning the *supersonic sections* of nozzles, a resolution method based on the transient conservation equations would have numeric problems (passage from subsonic to supersonic regime, shock waves if non-adapted conditions, etc.), so a 1D quasi-steady implicit method has been implemented for these components, including non-isentropic effects (heat exchange) under frozen or equilibrium conditions.

8.1.1 Component Classification

The components of the COMB_CHAMBERS Library are listed in the figure below showing the inheritance hierarchy:

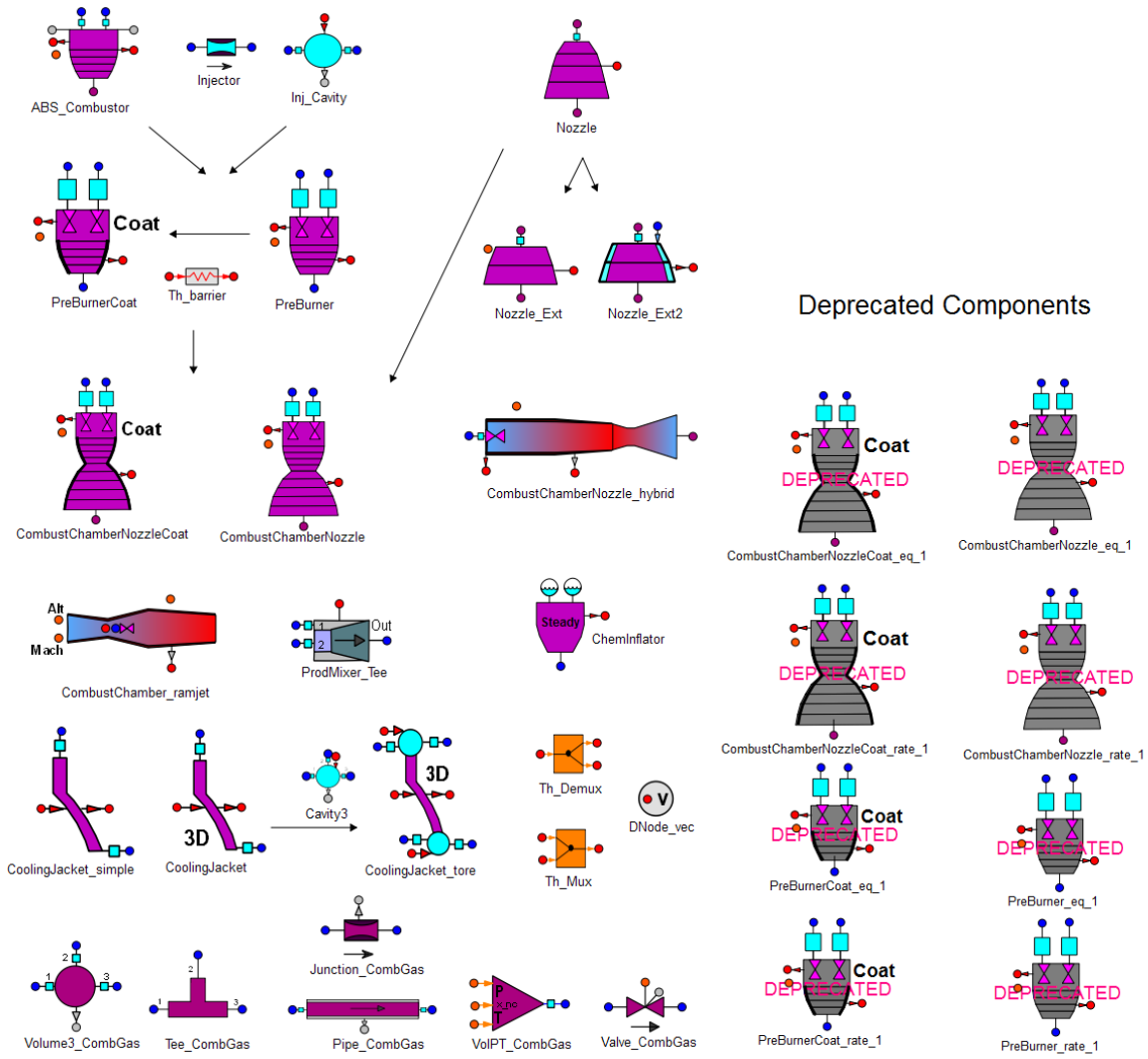


Figure 8-1 COMB_CHAMBER palette of symbols

The existing components have the following types (see §5.1.1):

- Cavities, Regenerative Circuit and Nozzle components are C (capacitive) elements
- Pre-burner outlets, Injectors and chemical inflator components are M (resistive) elements

- Propellant fluid ports of *Preburners* and *CombustChamberNozzle* components are capacitive, and can be connected to any FLUID_FLOW_1D library resistive component. Thermal ports should be connected to a Cooling Jacket component or other thermal component

First row *(ABS_Combustor, Nozzle, Injector and Inj_Cavity*) components are **only** used for building Combustion Chambers and Preburners. In order to avoid possible errors, the symbols of these components are not provided since they must be properly interconnected to each other (see §8.3.5.5).

Preburner components include a combustor, two injectors and two cavity components, and an outlet resistive fluid port where the mass flow is calculated. This port must be connected to a FLUID_FLOW_1D capacitive component so the combustion gases can be conducted to another combustor. *CombustChamberNozzle* components include a combustor, two injectors and two cavity components, and a non-adiabatic 1D Nozzle. They have a special exit port so a new nozzle extension component (with or without film cooling injection) can be connected.

CombustChamberNozzle_hybrid component include a Combustor_hybrid, an injector with cavity for oxy injection and a non-adiabatic 1D Nozzle component. *CombustChamber_ramjet* component include a Combustor_ramjet, an intake, and an injector and a cavity components for fuel injection.

Nozzle_Ext and Nozzle_Ext2 components are used for representing an extension of the nozzle without or with a film cooling around it respectively. Several *CoolingJacket* components are provided including simple or 3D thermal models.

Th_mux, Th_demux are multiplexer/demultiplexer components allowing splitting or joining a vectorized thermal port into several ones, so that different sized Cooling Jacket components can be connected to a chamber.

Components (*Pipe_CombGas, Tee_CombGas, etc*) ending with "_CombGas" are taken from the FLUID_FLOW_1D library and ready to be used for the combusted gases transport. The use of these components is equivalent to take them from the FLUID_FLOW_1D library with the parameter *burnerGasesOption* frozen to *Chemicals*. *Note that the mixing and transport processes of the combusted gases are performed by any FLUID_FLOW_1D component*.

The *ProdMixer_Tee* simulates a mixture of two pure fluids or a pure fluid with combusted gases. The fluid at the outlet of this component is always a mixture of Chemicals. To simulate a mixture of two combusted gases, a standard Tee component has to be used instead of this one.

As can be seen in Figure 8-1 above, two main models of combustor and main thruster are available: with no extension names (but with optional non-equilibrium and vaporization models) and the "…**Coat**" combustors (also having a thermal coating protection). "xxx_Rate" and "xxx_eq" combustor components have become deprecated components because their characteristics are now included in the components with no extension names.

8.1.2 Building a Model

It is recommended to read the User Manual and start off with the ROCKETS_EXAMPLES library and the execution of the different examples contained in it.

8.1.2.1 Mono/bi propellant combustors

The propellant fluids must be defined (as for any FLUID_FLOW_1D model, see §5.1.3.2) by inserting two "WorkingFLuid" components in the circuits connected to the respective injector ports.

Mono propellant combustors can be modeled by plugging (*Deadend* FLUID_FLOW_1D library component) one of the injector inlets. For compatibly reasons, the plugged inlet must be connected to a "WorkingFLuid" component that uses the same fluids as the other injector. If the active injector is the reducer, the minimum mixture ratio (MR data) must have a negative value.

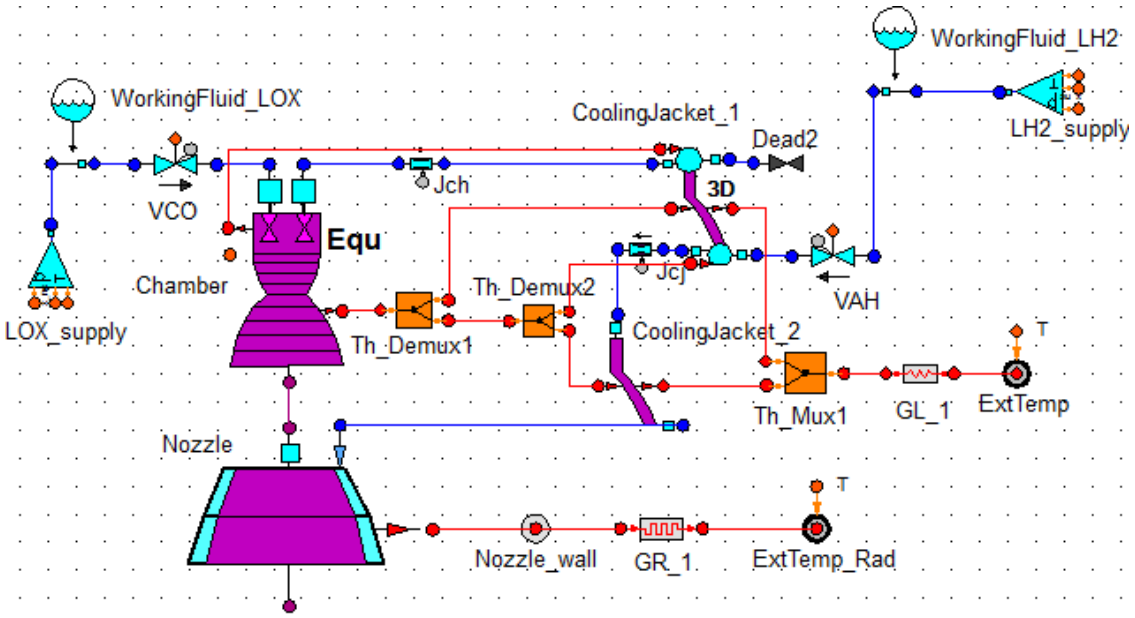
8.1.2.2 Geometry arrangement

Preburner, CombustChamberNozzle and *Nozzle* components have been modeled using a 1D discretization. They have been provided with a thermal port which can be connected to a Cooling Jacket

or other thermal components. The number of thermal nodes should be the same as the number of fluid nodes. Indeed, the cooling channel geometry and node distribution should match the combustion chamber for coherence (matching grid).

A *CombustChamberNozzle* component can be defined in which both the subsonic (combustor) and the supersonic sections will be covered by a *Cooling_Jacket* component. Then, if necessary, connect a *Nozzle_ext* to the chamber; the *Nozzle_ext* may or may not be covered by another *Cooling_Jacket*.

In other cases, where there is no direct correspondence between the combustor/nozzle nodes and the cooling jackets, it is always possible to use Multiplexer/Demultiplexer components to split a chamber thermal port into two or more different CoolingJackets or other thermal components. See the example below (Test_ChamberNozzle_mux) taken from the ROCKET_EXAMPLES library. In this case the chamber is connected to two different cooling jackets with some intermediate nodes (use of the second thermal demux) directly connected to the cooling jacket torus:



The definition of the nozzle and chamber geometry (diameters and dx length as a function of the axial position) needs some common data affecting several components (Chamber, Nozzle, Cooling Jacket). To facilitate data entry, the following common variables (see the input data of the respective components) are used in these components. Two options are available:

A) "dx_input" flag is defined as FALSE (default case):

- "Dc_vs_L" ("Dd_vs_L") tables: Normalized subsonic (supersonic) chamber diameters vs. normalized axial position. "Normalized" means that the "y" values of the diameters table are non-dimensional diameters i.e., divided by the throat diameter. "x" values are the axial lengths divided by the total subsonic (or supersonic) length.
- "dxc_vs_L" ("dxd_vs_L") tables: Subsonic (supersonic) mesh size distribution vs. normalized axial position. "x" values have the same meaning as for the "Dc_vs_L" ("Dd_vs_L") tables. "y" values are the **relative** mesh size depending on the axial position (x): The lower y value the lower mesh size. For example, a "y" value of 0.5 at x=0 and a "y" value of 2 at x=1 means that the mesh at the inlet (x=0) is four times smaller than at the exit (x=1). The same effect is obtained if the "y" value is 1 at x=0 and 4 at x=1.

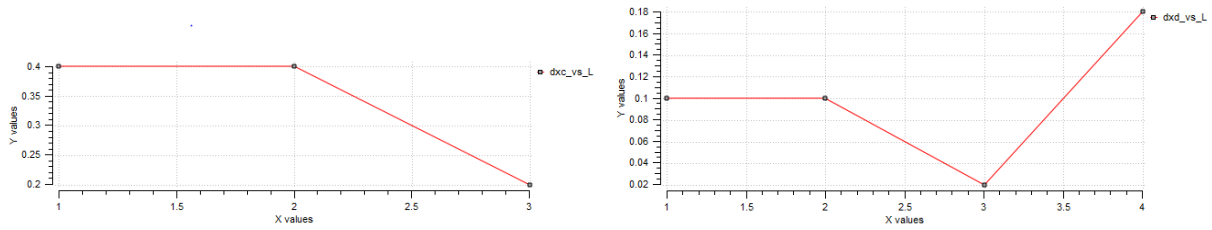
  The mesh size for each node is calculated by iteration. Tables "D_vs_L" and "dx_vs_L" will not be affected by a change in the number of nodes, in the length of the nozzle or in the throat diameter (scalars).

B) "dx_input" flag is defined as TRUE:

- "Dc_vs_L" ("Dd_vs_L") tables: Same meaning as before

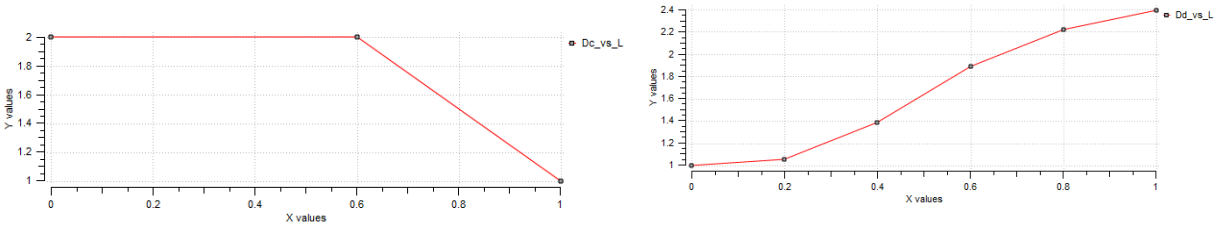
- "dxc_vs_L" ("dxd_vs_L") tables: In this case, these tables will directly give the normalized node lengths vs. the node number. Then, the x values of the table must be the node number. The sum of the y values (non-dimensional dx values) must be the simulated nozzle length divided by the total length, this being checked by the code. The execution will be stopped if it is not ok. Note that it is possible to simulate only a part of the total nozzle length defined in diameter profile.

For example, in the model of the previous figure we want to discretize the subsonic part of the chamber into 3 fixed nodes (dx_input=TRUE) of lengths 40%, 40% and 20% respectively. The supersonic part of the chamber is divided into 4 fixed nodes of lengths 10%, 10%, 2% and 18% respectively, so covering only the 40% of the total length of the nozzle. The 3rd node is connected to the cooling jacket torus. Then, dxc_vs_L, dxd_vs_L tables should be as follows. Note that the x values are the node number:



Chamber node discretization (subsonic and supersonic parts)

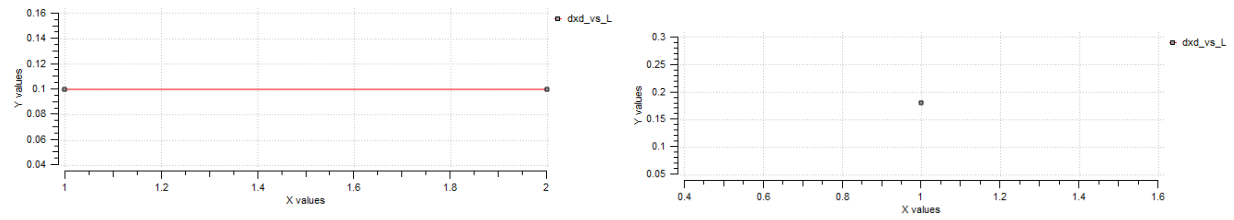
Subsonic and supersonic nozzle profiles are defined independently of previous tables. For example Dc_vs_L, Dd_vs_L can be as follows: (throat diameter half of the injection plate diameter). Note that now the x values are the non-dimensional lengths:



Chamber profile (subsonic and supersonic parts)

The cooling jackets node discretization must be coherent with the chamber. Then, the Dc_vs_L, Dd_vs_L tables should be the same as for the chamber. In our case, the first cooling jacket is connected to the subsonic part and to the first two supersonic nodes. The second cooling jacket is connected to the fourth node of the supersonic part. Then, dxc_vs_L of the first cooling jacket is the same table as for the chamber. The second cooling jacket will not use any dxc_vs_L table value.

dxd_vs_L tables of both cooling jackets (dxd_input=TRUE) should cover respectively the first two supersonic nodes (first cooling jacket) and the fourth supersonic node (second cooling jacket):

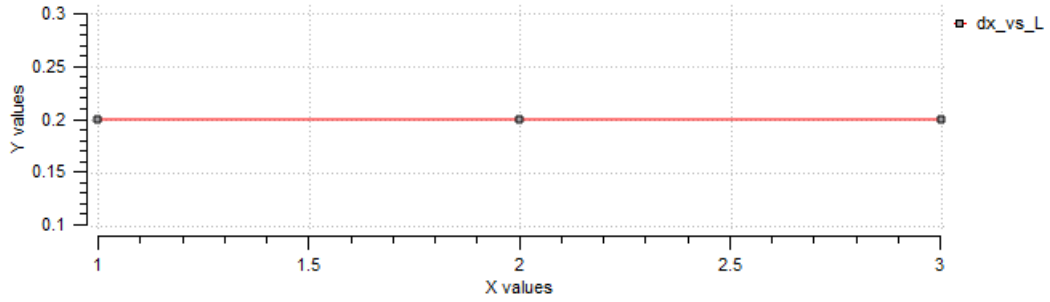


Cooling jackets node discretization

The throat diameter, subsonic and supersonic lengths (Dt, Lc and Ld) are defined independently of previous tables, with common values for the chamber and for the cooling jackets. The throat diameter of the cooling jackets can be different than the chamber one. The partial nozzle lengths are as follows:

Element	Lc (Length of the convergent part. To normalize subsonic axial position)	Ld (Total length of divergent part. To normalize supersonic axial position)	Ld1 (Nozzle axial length from throat to inlet - 0 if Lc > 0)	Ld2 (Nozzle axial length from throat to outlet)
Main Thruster	0.5	2	0	0.4 * 2
Nozzle extension	0	2	0.4 * 2	1.0 * 2
cooling jacket 1	0.5	2	0	0.2 * 2
cooling jacket 2	0	2	0.22 * 2	0.4 * 2

The "Nozzle extension" component will be used to complete the nozzle part without cooling jacket. Dd_vs_L for this part of the nozzle must be the same as for the other components (chamber and cooling jackets). dxd_vs_L table (the reminding part of the nozzle, i.e. from 40% to 100%) can be defined as follows (three nodes of constant length, 20%, 20% and 20% for example):



Finally, we point out that the nozzle profile is normally a common input for all these components. To examine the influence of the nozzle exit to throat area ratio, this ratio can be forced without changing the nozzle profile: simply change the input **AR** to a value other than 0. AR is an input modifying the exit/throat area ratio of a given profile without changing the relative shape. For a conic nozzle, this is equivalent to changing the nozzle angle.

8.1.2.3 Initialization and ignition control

- As a general rule, the combustor should be initialized with the external conditions. Then, under vacuum conditions, the combustor and the regenerative circuit can be initialized even with pressures lower than the saturation pressure at the external temperature (a minimum of 100 Pa is advised). If a proper calculation of the start-up is not necessary, use higher values of the initial pressure
- The ignition flag (a time dependent input data that will always be a BOUND in the experiment file, see the example below) allows you coarsely simulate an igniter. By default, its value is one (ignition activated). Nevertheless, it will be more physic to set this variable to zero while no self-ignition is foreseen and no ignition order has been commanded
- The valve opening-closing sequences must be treated with care. In principle, if only steady conditions are wanted, the simulation can begin with all the valves in their final position. Other times it will be necessary to test more physical opening laws using input data tables with the valve position as a function of time (see the example below). As a general rule, it is advisable to:

  A) Enter suitable values for the minimum/maximum Mixture Ratio and for the liquid vaporization delay ("MR_min, MR_max, tau_v" input data). *Lower vaporization delays can be more stable*.

  B) Initialize the chamber with the same vapor as that contained in the injector that will open first. Combustor chamber can be initialized with only reducer vapor, oxidizer vapor or non-condensable gas (x_nco = 0, -1, 1, respectively, see the combustor input dada). *Combustors using preburner gases are always initialized with non-condensable gases, see §5.1.3.3.*

  C) Activate the *IgnitionFlag=1* when the MR at the injection level (laws of valve opening) is enough for the chemical heat to vaporize all the injected liquids. In some situations, it might be a good idea

to set the ignition laws to always active. To do that, no limitations on the minimum/maximum MR should be selected (MR_min=-1, MR_max=1e20). In any case, the numerical model will try to automatically limit the liquid vaporization rate inside the combustor to physically prevent very low temperatures under unsuitable MRs during startup or shutdown processes, see §8.1.2.5.

Combustors can be initialized under burned conditions, choosing as input data the initial MR, chamber pressure and temperature. Nevertheless, a transient will be produced due to the non-consistence between the imposed chamber conditions and the non-stabilized feeding conditions. The model "*Test_chambeNozzle_eq*" of the ROCKETS_EXAMPLES library uses ignited initial conditions.

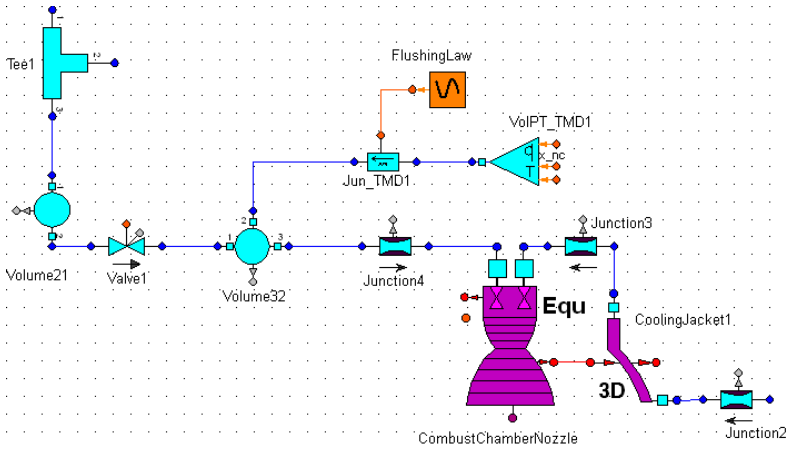
- *The combustion chamber also includes a solid propellant starter model to "help" the ignition process*; the starter gas mass flows, composition and their temperature are time-dependent input data, *not calculated*. By default, these values are zero (no starter). These inputs are automatically added to the default experiment file. Here below is an extract of the "*GGEngine*" model experiment using time-dependent tables (ROCKETS_EXAMPLES library):

```
EXPERIMENT exp1 ON GGEngine.default
DECLS
   -- Table defining the ignition law and Starter gases mass flow
   TABLE_1D Ignit_GG  = {{0, 2.0, 2.001, 100},  {0, 0, 1, 1}}
   TABLE_1D DM_law = {{0, 2.0, 2.5, 3.0, 3.5, 100}, {0, 0, 0.9, 0.9, 0.0, 0.0}}
…
   -- Table defining the valves law
   TABLE_1D VGH_law  = {{0, 3., 3.5, 100},  {0, 0, 1, 1}}
   TABLE_1D VGO_law  = {{0, 3., 3.5, 100},  {0, 0, 1, 1}}…
…
BOUNDS
   -- function interpolating in the table of ignition law
   Gas_generator.Combustor.IgnitFlag = timeTableInterp( TIME, Ignit_GG)
   Gas_generator.Combustor.starter_T = 840
   Gas_generator.Combustor.starter_m = timeTableInterp( TIME, DM_law)   …
…
   -- function interpolating valve positions
   VGO.s_pos.signal[1] = timeTableInterp( TIME, VGO_law)
…
END EXPERIMENT
```

The mass fractions of the solid propellant gases are input data. The solid propellant gases must be composed of gases from a predefined set of chemicals {H2O, CO, CO2, N2, H2, He}, where most of the starter gases produced in a particular starter powder are included.

8.1.2.4 Flushing modeling

During startups and shutdowns, it may be necessary to use flushing of non-condensable gas to prevent reverse flow at injectors and to at temperate the chamber. This can be done by connecting a *lateral* line to the inlets of the chamber. See the example below:



- Choose a law of flushing (mass flow) during a particular time schedule using the AnalogSignal CONTROL component

- Select the appropriate flushing boundary conditions using the VolPT_TMD component. Note that the non-condensable mass fraction imposed at this boundary should be 1.

8.1.2.5 Equilibrium vs. Non-Equilibrium combustor models

Using default combustor options GasLiqOption=FALSE, rateOption=FALSE, the following behavior will be considered:

- In non-ignited (frozen) conditions, only the injected vapors and the non-condensable gases (the mixture of which is calculated in the successive chamber volumes) will contribute to the chamber pressurization. Injected liquids are assumed to be lost.
- In ignited (equilibrium) conditions it is assumed that all the injected liquid will be vaporized within a delay time, but with a relaxation of the injected liquid mass flow at very low temperatures, see §8.2.3.3. The heat for the vaporization will be taken from the combustion heat. It is assumed that the actual concentrations of vapors at the exit of the first chamber volume will react immediately.

Using combustor option GasLiqOption=TRUE, the following behavior will be considered:

- Convection of liquids is calculated assuming same speed as in the gas side. The droplet size is assumed to be an input data that can be "modulated" in time and along the chamber by the vaporization factors "f_v[i]", where "i" is the chamber volume number (x position). "f_v" factors are boundary (time dependent) variables, automatically loaded in the default experiment).

```
BOUNDS
  Chamber.Combustor.f_v[1] = 1    -- use a table if f_v depends on time
   Chamber.Combustor.f_v[2] = 1    --     ""                    ""
…
```

- Then, the *heat and the vaporization flows at the droplet interface* are calculated as a function of the liquid properties (latent heat, vapor pressure, ...) and the current chamber conditions (P/T, Reynolds number, ...). The vapor mixture equations will account for these source terms in the successive chamber volumes taking into account the chamber length and the residence time, see §8.2.4.2.

Using combustor option rateOption=TRUE, the following behavior (first approach of a non-equilibrium chamber based on time-delay parameters) will be considered:

- In non-ignited (frozen) conditions, no reaction will take place; only vapors will be produced from liquid jets. In ignited conditions, besides the vaporization process, it is assumed that any chemical species in vapor phase (reactants and products can react with each other) will follow a *global* reaction rate defined as a characteristic burning time, see §8.2.4.3.
- The "rate" model can be used to estimate the amount of liquid not being evaporated and exiting the chamber (so the efficiency of the chamber) both under frozen or ignited conditions. If the option **liquidExitAllowed** is TRUE, the amount of liquid exiting a chamber will be added to the gas mixture passing by turbines and other FLUID_FLOW components downstream of the chamber.

All the combustor models include an automatic limitation of the vaporization rate when the chamber temperature is very low (vaporization in the vacuum), the "rate" option being the more realistic of the two. *Note that the mixture exiting a combustor is calculated accordingly with the perfect gas or Van der Waals state of equation, so temperatures below the* triple point *are supported with difficulty*.

8.1.2.6 Staged combustion models

One or more instances of interconnected preburners and main chambers can be used. Combusted chemical species will be considered as new reactants for the downstream chambers. Convection and mixing of the chemical constituents are now calculated by the transport equations for any component (from the FLUID_FLOW_1D and TURBO_MACHINERY libraries) downstream of a combustor.

As for any other FLUID_FLOW_1D component, the injector ports must be initialized with the appropriate Inj_oxyGases/Inj_redGases parameter, which has two options (see input data description):

- = FLUID_PROPERTIES.*Chemicals* if the respective injector is placed downstream of a combustor.
- =FLUID_PROPERTIES.*noBurnGases* if the respective injector is **not** downstream of a combustor.

8.1.2.7 Numerical limitations

From a numerical point of view, besides the tips given in §5.1.3.7, the following recommendations and limitations should be considered:

- In general, we recommend using the STEADY library (see §11.1) for design purposes. Use the COMB_CHAMBERS and the rest of ESPSS libraries for transient aspects (startups, shutdowns, etc.) and for particular sensitivity or failure studies starting from a state saved when a transient has been stabilized (see §3.7).
- The more species in a reaction, the more calculation time is required. The reaction products (in combustors and nozzles) are calculated implicitly (minimum Gibbs function), not only the chemical species concentration but also the pressure and temperature, and the same for any calculation time and discretized chamber section. In some occasions the implicit calculations can fail during abrupt chamber extinctions or ignitions: try then to verify/modify the input data and the boundaries to more "easy" conditions. Two situations are particularly sensible to this drawback:

  a) Nozzle equilibrium options (frozen_th, frozen_nz = FALSE): activate these options at later times (not during startup); see the *"PressureFedEngine_eq"* example of the ROCKET_EXAMPLES library.

  b) The chamber wall temperatures become implicit variables if an insulation (or radiative) thermal component is *directly* connected to the chamber thermal port. To avoid this, use a diffusive thermal node in between the combustor and the insulation.
- The combustor temperature can be very low during ignition or shutdown processes due to the vaporization of injected liquids. To account for this situation to the extent possible, the vaporization is automatically reduced as long as the chamber temperature approaches the triple point.

  Consequently, in some abrupt startup/shutdown simulation cases the combustor model can be unstable. It is advisable in these circumstances to use a solid propellant starter model that can be physically necessary, or to limit the startup/shutdown simulation to some delayed/modified ignition laws, or to initialize the chamber with higher pressures and different values of the vaporization delays, tau_v. See §8.1.2.3, §8.2.3.3, §8.2.4.2 and the different application examples of the ROCKET_EXAMPLES library.
- Combustors or Nozzles components should use an associated thermal component (a cooling Jacket or a diffusive thermal node) calculating the thermal wall temperatures explicitly. If no regenerative circuit is used, use the diffusive thermal component available in the COMB_CHAMBERS library calculating a different temperature for each chamber node (axial heat flush neglected). The diffusive THERMAL component imposes constant wall temperatures.
- Try to reduce ABS_ERROR and REL_ERROR integration parameters in case of non-convergence (or when the simulation becomes very stiff), but not to values lower than 1e-6.
- In some rare cases, mainly using hydrocarbons, the calculation of the chemical equilibrium can find some difficulties because of the formation of solid graphite. *Currently solid combustion products are not correctly treated in the product gas mixture. They are considered as gases.*
- *Combustor injectors do not allow negative flow. In case of chamber pressure greater than the injector pressure, zero flow is assumed.*
- Do not initialize combustor cavities with non-condensable gases unless this will be really necessary, for example when simulating flushing phenomena with helium. The feeding process of liquid propellant into a cavity initially filled with gas can be high CPU consuming.
- Control loops regulating valves during the startup/shutdowns can introduce instabilities because these controls are normally valid under quasi-steady conditions. It is preferable to regulate the opening/closing valves by means look-up tables with well-known sequences in time according to the physics of the combustor. See for example the experiment file of the "*GGEngine*" model (Vxxx_law tables) in the ROCKETS_EXAMPLES library and §8.1.2.3
- *Building engine cycles, the turbines cannot work with liquids. There is no model for hydraulic turbines, but might be modelled with a pump working with reverse flow.*

8.2 ABSTRACT COMPONENTS. MAIN FORMULATION

8.2.1 Inj_Cavity

Inherited from a "Capacity" (FLUID_FLOW_1D library, see §5.3.3), this component represents the combustion cavities upstream of the injectors with thermal ports allowing heat exchange.

8.2.1.1 Conservation and State Equations

The same formulation as in §5.3.6: The conservation equations are valid even if the fluid conditions are liquid, vapor or *homogeneous* two phase flow. Bubble collapse will be calculated.

8.2.1.2 Molar fraction calculation

This component is also used for calculating the propellant molar fractions from the injected fluids:

$$N_{chem} = 1; \qquad (Case\ of\ \ pure\ \ fluid)$$

$$N_k = y_k; \qquad MW_{mix} = \sum_{k=1,Nchem} y_k MW_k; \qquad (Case\ of\ \ previously\ burned\ \ gases)$$

In case of a pure fluid, *chem* is the chemical corresponding to the *main* fluid (see §4.1.2). In case of previously combusted gases, y_k are the molar fractions at the inlet of the cavity calculated by an upstream combustor). MW_k is the molecular weight of the chemical constituent k.

8.2.2 Injector

This component represents the injectors of a combustor chamber simulated as a concentrated load loss with constant throat area and with sonic flow limitation.

The same formulation as in the orifice component of the FLUID_FLOW_1D library is used. See §5.3.2. *The only exception is that the injector does not allow negative flow*. This is because, in the present version, FLUID_FLOW_1D components cannot handle a mixture of chemicals.

8.2.3 ABS_Combustor (Equilibrium Option)

This component represents a non-adiabatic combustion process inside a chamber for liquid or gas propellants. The transient conditions (gases composition, pressures, temperatures, mass flows and heat exchanged with the walls) will be derived from general 1D area varying transient conservation equations. The equilibrium combustion gases are calculated using the minimum Gibbs energy method as a function of the propellant's mixture molar fractions and enthalpies, and the chamber pressure.

8.2.3.1 Mixture ratio equation

A mixture equation between the injected propellants and the combustion gases is applied. From the definition of the mixture ratio (MR) and derivation, the following dynamic equation gives the MR evolution:

$$\left.\begin{aligned} MR &= Mass_{oxy} / Mass_{red} \\ Mass_{red} &= \frac{(\rho \cdot V)_{chamber}}{1+MR} \end{aligned}\right\} \Rightarrow \quad m_{oxy} = MR \cdot m_{red} + MR' \cdot \frac{(\rho \cdot V)_{chamber}}{1+MR}$$

where,

MR	*:* Mixture ratio
ρ	*:* Fluid mixture density
V	*:* Chamber volume
m_{oxy}, m_{red}	*:* Propellant mass flows

8.2.3.2 Ignition/Extinction Mode (Discrete Block)

Combustion takes place when the mixture ratio is within the allowed limits and the ignition flag is active.

```
-- Detection of burning mode according to the MR
```

```
WHEN (NOT(Burning) AND MR_inj > MR_min AND MR_inj < MR_max
                   AND IgnitFlag > 0.5) THEN
        Burning = TRUE
        WRITE("\n ******     Comb_Chamber: Ignition  ...  \n \n")
END WHEN
```

Extinction comes if both the mixture ratio at injection level and in the chamber (see above) is ten times out of the allowed limits. The following WHEN is used:

```
      -- Detection of low/hight MR preventing burning

WHEN (Burning AND (max(MR_inj,MR) < MR_min/10 OR IgnitFlag<0.5 OR \
                   min(MR_inj,MR) > MR_max*10)  ) THEN
        Burning = FALSE
        WRITE("\n ******     Comb_Chamber: Extinction  ... \n \n")
END WHEN
```

At extinction time, a re-initialization of all dynamic variables at the current MR is procured assuming that all burned gases have flown out. Other more sophisticated models can be programmed in the DISCRETE BLOCK to simulate more accurate ignition processes.

8.2.3.3 1D Conservation Equations

The 1D set of equations is implemented according to a staggered grid (see §5.3.8.3) in which the P/rho/x variables are defined in the center of the volumes and the mass flows at the junction between the volumes.

Gas mixture mass and energy equations:

$$V_i \rho'_i = m_{jun,i-1} - m_{jun,i} + \text{starter\_m}$$

$$(u'_i \rho_i + u_i \rho'_i) V_i = (mh)_{jun,i-1} - (mh)_{jun,i} + \text{starter\_mh} + q_{wall,i}; \qquad i = 1, n$$

where:

$m_{jun,i}$: Mass flow at the exit of volume no. i
$mh_{jun,i}$: Gas mixture enthalpy flow at the exit of volume no. i
ρ_i , u_i : Gas mixture density and total energy at volume no. i. ($u=u_{st}+0.5\ v^2$)

- At injector level (i=0), the junction mass flows will be calculated by the Injector components, see §8.2.2. The effective **liquid** mass flow entering into the chamber will be computed as follows:

  1. Burning=FALSE, only the injected vapors and non-condensable gases contribute to the chamber pressurization. Injected liquids are assumed to be lost.
  2. Burning=TRUE, it is assumed that all the injected liquid will be vaporized within a delay time, tau_v (injected gas is not modified):

$$m'_{red,liq} = \left((1 - x_{red}) m_{red} \tanh\left(3 \frac{T_{nsub} - T_{tr,red}}{T_{tr,red}}\right) - m_{red,liq} \right) / tau_v$$

$$m'_{oxy,liq} = \left((1 - x_{oxy}) m_{oxy} \tanh\left(3 \frac{T_{nsub} - T_{tr,oxy}}{T_{tr,oxy}}\right) - m_{oxy,liq} \right) / tau_v$$

  where $m_{oxy,liq}$, $m_{red,liq}$ are the effective liquid injected mass flow; m_{oxy}, m_{red} are the total injector mass flow; x_{red} and x_{oxy} are the gas (vapors and non-condensable gas) mass fractions in the cavities. τ_{vap} is the characteristic vaporization time, **tau_v**. The hyperbolic tangent term is added to produce a relaxation of the injected liquid mass flow at very low temperatures (ignition process).

- Total enthalpies "h_{jun}" are calculated using the upstream cell conditions: $h_{jun,i} = h_{i-1} = (u + P/\rho)_{i-1}$. P, T conditions will be derived from the combustion equations, see §8.2.3.4.

As before, at injector level the enthalpy flows will be computed in correspondence with the effective liquid and gas (vapors and non-condensable gas) mass flows that will be multiplied by the corresponding liquid and gas cavity enthalpies. This allows injection conditions (liquid, gas or two-phase flow) to be taken into account.

- The starter terms ("starter_m" and "starter_mh") are source terms in the mass and energy conservation equations only for the first chamber volume. The composition of the solid propellant gases is an input data within a predefined set of chemicals (starter_mh = f(starter_T, powder_composition); starter_m, starter_T and powder composition being input data.
- Node velocities "v" are calculated using the adjacent junction mass flows and the junction densities:

$$v_i = \frac{1}{A_i}\cdot\left(\frac{m_{jun,i-1}}{\rho_i + \rho_{i-1}} + \frac{m_{jun,i}}{\rho_i + \rho_{i+1}}\right) \quad i=2,n; \qquad v_1 = \frac{m_{jun,1}}{A_1\rho_1}$$

For the first volume, it is assumed that only the outlet junction will count. For the last volume, the throat density should be used (transmitted by the nozzle port).

Mass equations for vapors / non-condensable / solid propellant gases:

The vapor and the non-condensable conservation equations take into account the mixture process:

$$(x'_{red,i}\,\rho_i + x_{red,i}\,\rho'_i)\,V_i = x_{red,i-1}m_{jun,i-1} - x_{red,i}m_{jun,i}$$
$$(x'_{oxy,i}\,\rho_i + x_{oxy,i}\,\rho'_i)\,V_i = x_{oxy,i-1}m_{jun,i-1} - x_{oxy,i}m_{jun,i}$$
$$(x'_{nc,i}\,\rho_i + x_{nc,i}\,\rho'_i)\,V_i = x_{nc,i-1}m_{jun,i-1} - x_{nc,i}m_{jun,i}$$
$$(x'_{pw,i}\,\rho_i + x_{pw,i}\,\rho'_i)\,V_i = x_{pw,i-1}m_{jun,i-1} - x_{pw,i}m_{jun,i}$$

where:

$x_{red,i}$	: Reducer vapor mass fraction at volume no. i
$x_{oxy,i}$	: Oxidizer vapor mass fraction at volume no. i
$x_{nc,i}$	: Non condensable mass fraction at volume no. i
$x_{pw,i}$	: Solid propellant gases mass fraction at volume no. i

The mass flow of the injected vapors and liquids (depending on the burning mode, see before) and injected non-condensable gases will be added as source terms to the respective conservation equation of the *first* chamber volume. In the same way, the mass flow of the solid propellant gases, "starter_m", will be added as source a term to the solid propellant gases conservation equation of the *first* chamber volume.

Under burning conditions, only the mass fractions of the first chamber volume will be used to compute the molar fraction of the reactants. Subsequent volumes will use as reactant the product of the upwind volume (see §8.2.3.4).

Momentum equations:

Momentum equations are basically the same as in the Pipe component:

$$0.5(L_{i-1}+L_i)\cdot m'_{jun,i} = A_{i-1}\left[P + qn + \rho\cdot v^2 - 0.25\xi\cdot\rho\cdot v|v|\right]_{i-1} - A_i\left[P + qn + \rho\cdot v^2 + 0.25\xi\cdot\rho\cdot v|v|\right]_i - 0.5(P_{i-1}+P_i)(A_{i-1}-A_i)$$

Artificial dissipation, qn(i), is calculated as follows:

$$qn(i) = -Damp\,\frac{m_{jun}(i+1) - m_{jun}(i)}{A}V_{sound}(i)$$

Damp is a global boundary of the FLUID_FLOW_1D library to be defined in the experiment file, see *§5.1.3.6*. Momentum equations are applied to the exit of any volume in which the combustor is discretized with the exception of the last volume that should end with the throat to avoid numerical

problems (transition from subsonic to supersonic flow). The outlet mass flow will be calculated by the inherited components (preburners) or by the nozzle connected to the combustor.

8.2.3.4 Combustion gases properties calculation

For each chamber volume the combustion gas properties (product's molar fraction, heat and transport properties) are calculated using the Minimum Gibbs energy method as a function of the propellant molar fractions, pressure and specific enthalpy.

First chamber volume:

Reducer contribution: $N_{k,1} = x_{red,i}\frac{y_{k,red}}{MW_{mix,red}}$; $MW_{mix,red} = \sum_{k=1,Nchem} y_{k,red} MW_{k,red}$

Oxidizer contribution: $N_{k,1} = N_{k,1} + x_{oxy,i}\frac{y_{k,oxy}}{MW_{mix,oxy}}$; $MW_{mix,xoy} = \sum_{k=1,Nchem} y_{k,oxy} MW_{k,oxy}$

Solid propellant gases contribution: $N_{k,1} = N_{k,1} + x_{pw,i}\frac{y_{k,pw}}{MW_{mix,pw}}$; $MW_{mix,pw} = \sum_{k=1,Nchem} y_{k,pw} MW_{k,pw}$

Non condensable (NC) contribution: $N_{NC,1} = N_{NC,1} + \frac{x_{NC,1}}{MW_{NC}}$

Nchem is extended to any chemical treated by the FLUID_PROPERTIES library, see §**4.1.2**
MW_k : is the molecular weight of the chemical constituent k
$x_{NC,1}$: Mass fraction of non-condensable gas at volume 1, see conservation equations
$x_{red,i}$; $x_{oxy,i}$; $x_{pw,i}$: Mass fractions of vapors and solid propellant gases, see conservation equations
y_{k_red}; y_{k_oxy}; y_{k_pw}: Molar fraction of chemical k of the reducer (oxidizer) mixture
$N_{k,1}$: Number of moles of the chemical constituent k of the reactant mixture at volume 1

The propellants (reducer and oxidizer mixtures) molar fractions have been calculated by the cavity components. Propellants can be formed by any allowed fluid mixture using the FLUID_FLOW library, including that of a previous combustion.

Then, the number of moles, N_k is normalized. With this entry calculated and using the "dynamic" enthalpy value obtained from the conservation equations of the first combustor volume, it is possible to call the Minimum Gibbs energy method (see Appendix A7) to obtain the equilibrium temperature and the molar fraction of the products:

$$(y_{k\_eq}, T_{eq})_1 = f_{minGibbs}(N_{k,1}, h_1 - v_1^2/2, P_1)$$

Two possibilities are foreseen after calling the previous function: *A/ Equilibrium and B/ frozen* flow. In the latter case (no ignition), molar fractions remain constant: $y_{k\_eq,1} = N_{k,1}$

Subsequent chamber volumes:

A/ Under frozen conditions, the molar fractions of the subsequent volumes will be calculated as for the first volume. The temperature calculation from the enthalpy value will require an iteration procedure, in this case less complicated than in equilibrium conditions.

B/ Under equilibrium conditions, subsequent volumes will consider that the molar fraction of the products of the previous volume will act as the inlet propellant mixture, so the Minimum Gibbs energy method can be applied to any combustor volume.

$$(y_{k\_eq}, T)_i = f_{minGibbs}((N_k)_{i-1}, h_i - v_i^2/2, P_i); \qquad \text{i:volume number}$$

Calculation of chamber pressures (combustor efficiency):

The effective combustion gas constants (R_i, Cp_i, $cond_i$, $visc_i$) will be derived using the 'mixture properties' equations as a function of $y_{k\_eq,i}$. See §4.5.1. The pressure is obtained from the perfect gas state equation:

$$P_i = \rho_i \cdot R_i \cdot T_i \, \eta$$

where η is the combustor efficiency. Note that the pressure equation and the Minimum Gibbs function become an algebraic loop.

Finally, the molar fraction of the products of the last volume will be transmitted to the outlet port to be used by the Nozzle component or in another possible chamber.

8.2.3.5 Heat exchanged with the walls

The term q_{wall} appearing in the energy conservation equation permits the exchange of heat through a THERMAL port. Both convection and radiations are modeled:

$$q_{wall,i} = hc_i A_{wet,i}(T_{i,aw} - tp.T(i)) + \sigma \cdot A_{wet,i}(T_i^4 - tp.T(i)^4)$$

hc_i is the wall heat exchange coefficient and $A_{wet,i}$ is the chamber wet area of volume i. *tp* is the name of the thermal port (with n nodes in axial direction) connected to the Combustor. σ is the Stefan-Boltzmann constant = $5.67.10^{-8}$ $W \cdot m^{-2} \cdot K^{-4}$. *tp.T(i)* behaves as the internal wall temperature, to be determined in the connected wall component. The walls are not included in this component but can be represented by THERMAL components or by the Cooling Jacket component. T_i is the fluid temperature and $T_{i,aw}$ is the adiabatic wall temperature defined as:

$$T_{i,aw} = T_i(1 + \mathrm{Pr}_{i,ref}^{0.33} \frac{\gamma_i - 1}{2} M_i^2); \qquad \mathrm{Pr}_{i,ref} = (Cp_i \lambda_i / \eta_i)_{ref}$$

The reference conditions are calculated at a temperature halfway between the wall and the free stream static temperature. It is assumed that the mixture composition does not change between these two temperatures. The wall heat exchange coefficient hc is calculated using empirical correlations according to Bartz [RD-23]:

$$hc_i = 0.026 \cdot \eta_{i,ref}^{0.2} \cdot (\lambda / \eta)_{i,ref}^{0.6} \cdot Cp_{i,ref}^{0.4} \cdot m_{th}^{0.8} / A_i^{0.9} \cdot (\pi D_{th} / 4 / R_{curv})^{0.1}$$

where:

$\eta_{i,ref}$: viscosity of combusted gases at volume no. i and T_{ref} temperature
$\lambda_{i,ref}$: conductivity of combusted gases at volume no. i and T_{ref} temperature
R_{curv}: Curvature radius of the throat
D_{th}: Throat diameter
m_{th}: Throat mass flow
A_i: Cross section at volume no. i

The heat exchanged with the fluid is transmitted through the thermal port making: $tp.q(i) = q_{wall,i}$.

8.2.4 ABS_Combustor (Non-equilibrium Option)

This component represents non-equilibrium, non-adiabatic combustion process for liquid or gas propellants. The transient conditions (gases composition, pressures, temperatures, mass flows and heat exchanged with the walls) will be derived from general 1D area varying transient conservation equations. *The non-equilibrium model does not include finite rate chemistry but time-delay parameters on the chemical equilibrium and a model calculating liquid transport and evaporation.*

The ABS_*Combustor* formulation is upgraded with new options: **GasLiqOption** for the liquid propellant vaporization models and **rateOption** to choose equilibrium or reaction delay method:

- *GasLiqOption = UserDefined*: Same model as in the *ABS_Combustor,* equilibrium option.
- *rateOption = FALSE*: Same model as in the *ABS_Combustor,* equilibrium option.
- *GasLiqOption = Advanced*: The convection of liquids and its vaporization are calculated. See the formulation in the following paragraphs.
- *rateOption = TRUE*: This is a non-equilibrium chamber model based on time-delays. See the formulation in the following paragraphs.

8.2.4.1 Ignition/Extinction mode (Discrete Block)

The same formulation is used as in the *ABS_Combustor,* equilibrium option.

8.2.4.2 1D Conservation equations

The mass, energy and momentum equations include those of the equilibrium Combustor component plus the ones concerning the vaporization model. Liquid phase conservation equations are included.

Gas mixture mass and energy equations:

$$V_i \rho'_i = m_{jun,i-1} - m_{jun,i} + starter\_m + m_{vap\_red,i} + m_{vap\_oxy,i}$$

$$(u'_i \rho_i + u_i \rho'_i) V_i = (mh)_{jun,i-1} - (mh)_{jun,i} + starter\_mh + q_{wall,i} + q_{vap\_red,i} + q_{vap\_oxy,i}; \qquad i = 1, n$$

where,

$m_{jun,i}$	: Gas mixture mass flow at the exit of volume no. i
$(mh)_{jun,i}$	: Gas mixture enthalpy flow at the exit of volume no. i
ρ_i , u_i	: Gas mixture density and total energy at volume no. i. ($u=u_{st}+0.5\ v^2$)
v_i	: Mean velocity *at* volume no. i
$m_{vap\_oxy,i}$	: Vaporized liquid mass flow of oxidizer at volume no. i
$m_{vap\_red,i}$	: Vaporized liquid mass flow of reducer at volume no. i

- At injector level, the gas mass and enthalpy flows (m_jun_0) will be calculated by the Injector components taking into account the quality calculated in the Cavities components. The liquid contributions will be considered in the droplet conservation equations.

  $$m_{red} = m_{red,inj} x_{red}; \qquad m_{oxy} = m_{oxy,inj} x_{oxy}$$

  where $m_{oxy,inj}$, $m_{red,inj}$ are the injector mass flows (liquid or gas) and x_{red} and x_{oxy} are the vapor mass fractions in the cavities. Then $m_{jun,0} = m_{red} + m_{oxy}$. The hyperbolic tangent term is added to produce a relaxation of the injected vapor mass flow at very low temperatures
- Total enthalpies "h_{jun}" are calculated using the upstream cell conditions: $h_{jun,i} = h_{i-1} = (u + P/\rho)_{i-1}$. P, T conditions will be derived from the combustion equations (see §8.2.4.3). Node velocities "v" are calculated as in the *ABS_Combustor,* equilibrium option.
- Starter terms are included in the same way as in the *ABS_Combustor,* equilibrium option.

Vaporization flows, "m_{vap}", and enthalpies flows, "q_{vap}" (source term for the gas mixture conservation equations), are calculated by the droplet vaporization models. Two models are available (see function *GasLiq_exch)*:

A/ User defined model (GasLiqOption == UserDef).

The vapor mass flows are calculated assuming a characteristic vaporization time modulated with user defined vaporization factors:

$$m_{vap\_red,i} = f_{vap,i} M_{liq\_red,i} / \tau_{vap}; \qquad m_{vap\_oxy,i} = f_{vap,i} M_{liq\_oxy,i} / \tau_{vap}$$

$$q_{vap\_red,i} = m_{vap\_red,i} h(T_{liq\_red,i}); \qquad q_{vap\_oxy,i} = m_{vap\_ozy,i} h(T_{liq\_oxy,i})$$

B/ Droplet vaporization model (GasLiqOption == Advanced).

Assuming a very thin saturated layer between the droplets and the surrounding gases, the conservation equations establish that the sum of convective heat plus enthalpy mass flow are the same at both sides of the layer.

Then, the following set of equations is applied at each combustor volume i, allowing the calculation of the mass and energy exchanges through this layer:

$$m_{vap\_red,i} = A_{liq\_red,i} \left(hc_i (T_i - T_{sat\_red,i}) + hcond_i (T_{liq\_red,i} - T_{sat\_red,i}) \right) / (h_{vap\_red,i} - h_{liq\_red,i})$$

$$q_{vap\_red,i} = m_{vap\_red,i} h_{vap\_red,i} - A_{liq\_red\text{-}gas,i} hc_i (T_i - T_{sat\_red,i})$$

$$hcond_i = 2\lambda_{liq\_red,i} / D_{droplet\_red}$$

$$A_{liq\_red,i} = f_{vap,i} 6 M_{liq\_red,i} / \rho_{liq\_red} / D_{droplet\_red}$$

$$m_{vap\_oxy,i} = A_{liq\_oxy,i}\left(hc_i(T_i - T_{sat\_oxy,i}) + hcond_i(T_{liq\_oxy,i} - T_{sat\_oxy,i})\right)/(h_{vap\_oxy,i} - h_{liq\_oxy,i})$$

$$q_{vap\_oxy,i} = m_{vap\_oxy,i} h_{vap\_oxy,i} - A_{liq\_oxy\text{-}gas,i} hc_i(T_i - T_{sat\_oxy,i})$$

$$hcond_i = 2\lambda_{liq\_oxy,i} / D_{droplet\_oxy}$$

$$A_{liq\_oxy,i} = f_{vap,i} 6M_{liq\_oxy,i} / \rho_{liq\_oxy} / D_{droplet\_oxy}$$

Where for each propellant (reducer & oxidizer):

$M_{liq,i}$: Liquid mass at volume no. i, see the equations for liquids later on
$T_{liq,i}$: Liquid droplet temperatures, volume no. i
τ_{vap} : Characteristic vaporization time, **tau_v**, see the chamber input data
$f_{vap,i}$: Vaporization factor, time dependant input data "**f_v[i]**", see §8.1.2.5
T_i : Gas temperatures, volume no. i
hc_i : Heat exchange coefficient at gas side. Same value is used as is §8.2.3.5
$\lambda_{xxx,i}$: Droplet conductivity
$T_{sat,i}$: Saturation temperature calculated at the partial vapor pressure of volume no. i
$h_{vap,i}$, $h_{liq,i}$: Saturation enthalpies calculated at the partial vapor pressure of volume no. i
$D_{droplet}$: Mean droplet diameter
$A_{liq,i}$: Equivalent exchange area between the droplets and the gas

The droplet diameter has been modulated by the vaporization factor. This factor is an input datum depending on the time and on the volume number.

In theory, assuming a known droplet size at the injection plate, the droplet diameter evolution could be determined by "simple" equations relating the evaporated mass flow with the liquid mass conservation equations. Nevertheless, due to the high penetration and breakup of the liquid jets, it seems more realistic to assume a known (*adjusted*) droplet size at each chamber volume, the number of droplets being determined by the current liquid mass.

Limiting the vapor mass flow at low temperatures.

In both methods (used defined and droplet model), the vapor mass flow is weighted with a factor to prevent vaporization at very low temperatures (frozen liquid):

$$m_{vap,i} = m_{vap,i} \tanh\left(3\frac{T_i - T_{sat,i}}{T_{sat,i}}\right)$$

In normal situations, T>Tsat, thus making the value of the hyperbolic tangent be equal to one.

Burning rate:

The burned gas mass flow (second source term for the vapor conservation eqs., see below) is calculated assuming a *global* characteristic burning time [RD-25]. It is assumed that any species (vapor or burned gas) present in the gas mixture contributes to the global reaction rate, so the burning rate will be proportional to the total gas mixture density:

$$m_{bu,i} = f_{bu,i}\rho_i V_i / \tau_{bu} * \tanh((TIME - t\_burn)/\tau_c)$$

where (see the chamber input data):

τ_{bu} : Characteristic burning time, tau_b
τ_c : Ignition time delay, tau_c, with respect to the ignition order (t_burn)
$f_{bu,i}$: Burning factors at volume no. i.

The burning factors are automatically set to one if the burning conditions are true: mixture ratio within the allowed limits and ignition flag activated. Otherwise the burning factors are set to zero.

Mass equations for vapors / non-condensable / solid propellant *gases:*

The vapor and non-condensable mass conservation equations take into account the burned gas production and the vaporization terms previously calculated:

$$(x'_{red,i}\rho_i + x_{red,i}\rho'_i)V_i = x_{red,i-1}m_{jun,i-1} - x_{red,i}m_{jun,i} + m_{vap\_red,i} - x_{red,i}m_{bu,i}$$
$$(x'_{oxy,i}\rho_i + x_{oxy,i}\rho'_i)V_i = x_{oxy,i-1}m_{jun,i-1} - x_{oxy,i}m_{jun,i} + m_{vap\_oxy,i} - x_{oxy,i}m_{bu,i}$$
$$(x'_{nc,i}\rho_i + x_{nc,i}\rho'_i)V_i = x_{nc,i-1}m_{jun,i-1} - x_{nc,i}m_{jun,i}$$
$$(x'_{pw,i}\rho_i + x_{pw,i}\rho'_i)V_i = x_{pw,i-1}m_{jun,i-1} - x_{pw,i}m_{jun,i}$$

where

$x_{red,i}$: Reducer vapor mass fraction at volume no. i
$x_{oxy,i}$: Oxidizer vapor mass fraction at volume no. i
$x_{nc,i}$: Non condensable mass fraction at volume no. i
$x_{pw,i}$: Solid propellant gases mass fraction at volume no. i
$x_{bu,i}$: Burned gases mass fraction at volume no. i = 1 - $x_{red,i}$ – $x_{oxy,i}$
$m_{bu,i}$: Burned gases mass flow at volume no. i

The mass flow of the injected vapors (m_{red}, m_{oxy}) and injected non-condensable gases will be added as source terms to the respective conservation equation of the *first* chamber volume. In the same way, the mass flow of the solid propellant gases, "starter_m", will be added as source a term to the solid propellant gases conservation equation of the *first* chamber volume.

Conservation equations for liquids:

The mass and enthalpy flows for liquids are calculated assuming that Vel_gas=Vel_liq and neglecting Cp_liq derivatives:

$$M'_{liq\_red,i} = M_{liq\_red,i-1}vel_{i-1}/L_{i-1} - M_{liq\_red,i}vel_i/L_i - m_{vap\_red,i}$$
$$(M\cdot T)'_{liq\_red,i} = (M\cdot T)_{liq\_red,i-1}vel_{i-1}/L_{i-1} - (M\cdot T)_{liq\_red,i}vel_i/L_i - q_{vap\_red,i}/Cp_{liq\_red,i}$$
$$M'_{liq\_oxy,i} = M_{liq\_oxy,i-1}vel_{i-1}/L_{i-1} - M_{liq\_oxy,i}vel_i/L_i - m_{vap\_oxy,i}$$
$$(M\cdot T)'_{liq\_oxy,i} = (M\cdot T)_{liq\_oxy,i-1}vel_{i-1}/L_{i-1} - (M\cdot T)_{liq\_oxy,i}vel_i/L_i - q_{vap\_oxy,i}/Cp_{liq\_oxy,i}$$

where Li is the length of volume no. i.

Momentum equations:

Same formulation as in the "ABS_Combustor" (Non-equilibrium Option).

8.2.4.3 Combustion gases properties calculation

It is supposed that any molar fraction follows a *global* reaction rate in accordance with the previously mentioned burning time:

$$y'_{k\_bu,i} = (y_{k\_eq,i} - y_{k\_bu,i})/\tau_{bu}$$

where

$y_{k\_eq,i}$: Equilibrium molar fraction of the chemical constituent k at volume no. i.
$y_{k\_bu,i}$: Actual burned molar fraction of the chemical constituent k at volume no. i.
τ_{bu} : Characteristic burning time

For each chamber volume i, the combustion gases *equilibrium* composition (needed for the calculation of the *actual* burned gases composition) is calculated using the Minimum Gibbs energy method as a function of the gas mixture molar fractions, pressure and the enthalpy.

The gas mixture molar fractions are calculated as follows:

Reducer vapor contributions: $N_{k,i} = x_{red,i}\dfrac{y_{k\_red}}{MW_{mix,red}}$; $\quad MW_{mix,red} = \sum_{k=1,Nchem} y_{k\_red}MW_{k,red}$

Oxidizer vapor contributions: $N_{k,i} = N_{k,i} + x_{oxy,i}\dfrac{y_{k\_oxy}}{MW_{mix,oxy}}$; $\quad MW_{mix,xoy} = \sum_{k=1,Nchem} y_{k\_oxy}MW_{k,oxy}$

Solid propellant gas contribution: $N_{k,1} = N_{k,1} + x_{pw,i}\frac{y_{k,pw}}{MW_{mix,pw}}$; $MW_{mix,pw} = \sum_{k=1,Nchem} y_{k,pw} MW_{k,pw}$

Non condensable (NC) contribution: $N_{NC,i} = N_{NC,i} + \frac{x_{NC,i}}{MW_{NC}}$

Burned gas contribution: $N_{k,i} = N_{k,i} + x_{bu,i}\frac{y_{k\_bu,i}}{MW_{mix,bu}}$; $MW_{mix,bu} = \sum_{k=1,Nchem} y_{k\_bu,i} MW_{k\_bu}$

where

Nchem : is extended to any chemical treated by the FLUID_PROPERTIES library, see §**4.1.2**
MW_k : Molecular weight of the chemical constituent k
$x_{NC,i}$: Mass fraction of non-condensable gas at volume i, see the conservation equations
$x_{red,i}$; $x_{oxy,i}$; $x_{pw,i}$: Mass fractions of vapors and solid propellant gases, see conservation equations
y_{k_red}; y_{k_oxy}; y_{k_pw}: Molar fraction of chemical k of the reducer (oxidizer) mixture
$y_{k\_bu,i}$: Molar fraction of chemical k of the burned gas mixture at volume i
$N_{k,i}$: Number of moles of the chemical constituent k of the reactant mixture at volume i

The vapor (reducer and oxidizer mixtures) molar fractions have been calculated by the cavity components and can include any allowed fluid mixture using the FLUID_FLOW library, including that of a previous combustion. The burned molar fractions, y_{k_bu}, are calculated dynamically.

Once the number of moles of the reducer/oxidizer/burned gases mixture has been evaluated, and using the "dynamic" enthalpy value obtained from the conservation equations, it is possible to call to the Minimum Gibbs energy method (see Appendix A7) to obtain the equilibrium combustion gases composition:

$$(y_{k\_eq}, T_{eq})_i = f_{minGibbs}(N_{k,i}, h_i - v_i^2/2, P_i)$$

Two possibilities are foreseen after calling the previous function: *A/ Equilibrium* and *B/ frozen flow*. In the latter case (no ignition) the molar fractions remain constant:

$$y_{k\_eq,i} = N_{k,i}$$

Chamber pressure calculation:

The effective combustion gas constants (R_i, Cp_i, $cond_i$, $visc_i$) will be derived using the 'mixture properties' equations as a function of $y_{k\_bu,i}$. see §4.5.1. The pressure is obtained from the perfect gas equation: $P_i = \rho_i \cdot R_i \cdot T_i$. With respect to the non-equilibrium option, the pressure equation and the Minimum Gibbs function do *not* become an algebraic loop because the actual molar fractions ($y_{k\_bu,i}$) of the mixture are dynamic variables.

The molar fraction of the products of the last volume will be transmitted to the outlet port to be used by the Nozzle component or in another possible chamber.

8.2.4.4 Heat exchanged with the walls: See §8.2.3.5

8.2.5 Combustor

Inherited from an "ABS_Combustor", this component represents equilibrium, non-adiabatic 1D Combustor with an outlet fluid port. This component is typically used as a Pre-burner. Calculated molar fractions of combusted gases will be transmitted to other Chambers or ESPSS components.

This component also has the option *liquidExitAllowed* for including, or not, the amount of liquid exiting a chamber in the gas mixture passing by turbines or other FLUID_FLOW components. As the amount of liquid (oxidizer or reducer) has been already calculated with the evaporation model, the different mass fractions of the mixture of gases (vapor plus products) and liquids can be calculated:

- `Gas / liquid mass fractions`

```
x_g[i] = rho[i]*V[i] / (rho[i]*V[i] + M_lf[i]+M_lo[i]) --
```

```
x_lf[i] = M_lf[i] / (rho[i]*V[i] + M_lf[i]+M_lo[i])
x_lo[i] = M_lo[i] / (rho[i]*V[i] + M_lf[i]+M_lo[i])
```

where Nsub is the last chamber volume. `rho[]`is the gas mixture density and `M_lf[], M_lo[]` are the liquid mass of fuel and oxidizer calculated in the equations shown in previous paragraphs. `x_g[], x_lf[], x_lo[]` are the mass fractions of gas and liquids (fuel and oxidizer).

Finally, in case of *liquidExitAllowed*= TRUE, the mass and enthalpy flows to be transferred to the downstream component can be calculated as a perfect mixture:

```
f_out.m = m_jun[Nsub] + m_lf[Nsub] + m_lo[Nsub]
f_out.mh = f_out.m * (x_g[Nsub]*h[Nsub] + x_lf[Nsub]*h_lf + x_lo[Nsub]*h_lo)
```

`m_jun[], m_lf[], m_lo[]` refer to the gas and liquids mass flows calculated in the equations shown in previous paragraphs. It is assumed that `h_lf, h_lo` are the oxidizer and reducer saturated liquid enthalpies. In case of *liquidExitAllowed* = FALSE, the mass and enthalpy flows to be transferred to the downstream component are only the gas part:

```
f_out.m = m_jun[Nsub]
f_out.mh = m_jun[Nsub]* h[Nsub]
```

8.2.5.1 Combustion gas mass flow

The combustion gas mass flow is calculated using a similar equation as for the junction momentum equation (see §5.3.2.1), where the left magnitudes correspond to the last combustor volume and the right ones are those of the connected component (port variables).

8.2.5.2 Setting the combusted gases properties

The mass fractions of every chemical species produced in the combustion are transferred to the outlet port variables. Convection and mixing of the chemical constituents can be then calculated dynamically (using the transport equations) in any ESPSS component downstream of a combustor. These components will calculate properties according to the current chemical compositions, not with the combustor properties.

8.2.6 ABS_Combustor_hybrid

This component represents a non-adiabatic combustion process inside a chamber for hybrid or solid combustors. The transient conditions (gases composition, pressures, temperatures, mass flows and heat exchanged with the walls) will be derived from general 1D area varying transient conservation equations. The combustion gases are calculated using the minimum Gibbs energy method as a function of the propellant's mixture (rubber gases, solid constituents and injected oxidizer if any) enthalpies, and of the current chamber pressure also calculated.

The hybrid combustor includes an *optional* non-equilibrium model for the products calculation (time-delay parameters), a model calculating the oxidizer evaporation and another one for the release of vapors from the solid propellant (grain) together with the calculation of the grain temperature and heat exchanges.

8.2.6.1 1D Conservation equations

In the gas core of the combustor, similar equations that in the "Combustor" component are used, so the convection of liquids and droplets vaporization (oxidizer) and the combusted gases calculation follow the formulation given in §8.2.4. *The differences are:*

- Control volumes are not constant and the corresponding source terms (see §5.3.3.1) are added in the conservation equations. The derivative of volume i is: $V'i = M'_{sp,i}/\rho_{sp}$, see 8.2.6.4.
- The evaluation of source terms due to the solid propellant "evaporation" is explained below.

8.2.6.2 Solid propellant "evaporation" model

Two models are available for the source terms in the gas mixture conservation equations (m_{vap_red}, q_{vap_red}) corresponding to the solid propellant contribution (see function *GasSol_exch*):

A/ User defined model (GasSolOption = stdHybrid/stdSolid).

The solid propellant consumption *for each node "i"* is calculated assuming an empirical regression law, function of the mass flow or the pressure depending on the option:

$$r_{sp,i} = a_{sp} G_i^{a_{sp}} \quad (GasSolOption = stdHybrid); \qquad r_{sp,i} = a_{sp} P_i^{a_{sp}} \quad (GasSolOption = stdSolid)$$

$$m_{vap\_red,i} = r_{sp,i} \rho_{sp} A_{sp,i}$$

$$A_{sp,i} = f_{s,i} \pi D_{sp,i} L_i$$

where

a_sp, b_sp : Regression rate constants (input data)
ρ_{sp} : Grain density (input data)
G/P : Gas mass flow per unit of area or pressure, volume no. i
$D_{sp,i}$: Internal diameter of the solid propellant grain, volume no. i
L_i : Length of each chamber node no. i
$A_{sp,i}$: Equivalent exchange area between the grain and the gas, volume no. i
$f_{s,i}$: *grain/fluid interface* factor, time dependent input data
$A_{sp,i}$: Equivalent exchange area between the grain and the gas

f_s are boundary variables, automatically loaded in the default experiment*, to account for non-cylindrical surface areas of the grain (modulation of the grain /fluid interface area)*.

```
BOUNDS
   Chamber.Combustor.f_s[1] = 1    -- use a table if f_v depends on time
   Chamber.Combustor.f_s[2] = 1    --     ""                    ""
...
```

B/ Advance model (GasSolOption = vapModel).

Assuming a thin vapor layer between the grain and the surrounding gases, the conservation equations establish that the sum of convective heat plus enthalpy mass flow are the same at both sides of the layer. Then, the following set of equations is applied at each combustor volume i, allowing the calculation of the mass and energy exchanges through this layer:

$$m_{vap\_red,i} = A_{sp,i}\left(hc_i (T_i - T_{sat}) + hcond_i (T_{sp,i} - T_{sat})\right) / LH_{sp}$$

$$q_{vap\_red,i} = m_{vap\_red,i} LH_{sp} - A_{sp,i} hc_i (T_i - T_{sat})$$

$$hcond_i = 2\lambda_{sp,i} / th_{sp,i}$$

where:

T_{sat} : "Saturation" temperature of grain (input data)
LH_{sp} : "Latent" heat for grain evaporation (input data)
$\lambda_{sp,i}$: Grain conductivity (input data)
$T_{sp,i}$: "Liquid" grain temperatures, volume no. i
T_i : Gas temperatures, volume no. i
hc_i : Heat exchange coefficient at gas side, volume no. i
$hcond_i$: Heat exchange coefficient by conduction, volume no. i
$th_{sp,i}$: Grain thickness , volume no. i

As in the User defined model, the exchange area Asp,i between the grain and the gas is modulated by the *grain/fluid interface* factors.

The heat and enthalpy exchange with the gas phase is evaluated with the same formula in both cases:

$$q_{vap\_red,i} = m_{vap\_red,i} LH_{sp} - A_{sp,i} hc_i (T_i - T_{sat})$$

In this respect we point out the importance of the values of LH_{sp} ("Latent" heat for the grain evaporation) and $T_{sp,i}$ ("liquid" grain temperature) that are input data in the current model.

8.2.6.3 Combustion gases properties calculation

The mass fractions of the of the solid propellant constituents are input data (rubComp[] variable) within a predefined set of constituents {*HTPB, IPDI, RubUsr, KNO3_a, Al_cr, S_a, NH4NO3_IV, NH4CLO4_I*}. Note that the solid propellant "constituents" are reactants.

The combusted gases are calculated as in the *ABS_Combustor, but accounting for* rubber gases, solid constituents and injected oxidizer if any see §8.2.3.4 and §8.2.4.3.

8.2.6.4 Solid propellant consumption and temperature

The solid propellant consumption (grain mass and thickness evolution) is simply determined by the amount of released vapors:

$$M'_{sp,i} = -m_{vap\_red,i}$$

$$th_{sp,i} = 4M_{sp,i} / (\rho_{sp}\pi(D^2_{sp,i} - D^2_i)L_i); \qquad D_{sp,i} = D_i - 2th_{sp,i}$$

The grain temperature $T_{sp,i}$ is calculated accounting for heat exchanges at the wall and at the gas core:

$$(M \cdot T)'_{sp,i} = -m_{vap\_red,i}T_{sp,i} + (\mathrm{hcond}_i(\mathrm{T_{sat}} - \mathrm{T_{sp,i}}) - q_{wall,i})/Cp_{sp}$$

where:

$M_{sp,i}$: Grain mass, volume no. i
CP_{sp} : Grain heat capacity
q_{wall} : Heat exchanged between the wall and the grain

8.2.6.5 Heat exchanged with the walls

The term q_{wall} appearing in the energy conservation equation permits the exchange of heat through a THERMAL port. The walls (which can be represented by THERMAL components or by the Cooling Jacket component) are not included in this component:

$$th_{sp} > 10^{-6} \qquad q_{wall,i} = 2\frac{\lambda_{sp}}{th_{sp}} A_{wet,i}(T_{sp,i} - tp.T(i))$$

$$th_{sp} < 10^{-6} \qquad q_{wall,i} = hc_i A_{wet,i}(T_i - tp.T(i))$$

tp is the name of the thermal port (with n nodes in axial direction) connected to the Combustor grain (if present). The heat exchanged with the grain is transmitted through this port: $tp.q(i) = q_{wall,i}$. tp.T(i) is the wall temperature, to be determined in the connected wall component. The heat exchange coefficient is calculated using empirical correlations according to Dittus-Boelter correlation, A3.2.

8.2.7 Combustor_ramjet

This component represents equilibrium, non-adiabatic combustion process for air-breathing engines with subsonic or supersonic combustion including shock wave capturing depending on the ambient conditions.

The transient performances (combusted gases, pressures, temperatures, mass flows and heat exchanged with the walls) are derived from general 1D area varying transient conservation equations as for a Pipe component (Roe scheme is not included yet).

In the gas core of the combustor, similar equations as in the *ABS_Combustor,* non-equilibrium option, are used, so the convection of liquids and droplets vaporization (in case of liquid fuel injection) and the combusted gases calculation follow the formulation given in §8.2.4.

The difference is in the fuel injection which is multipoint, so any chamber node can have a particular fuel injection rate. The model calculating the particular fuel injection rate is very simple, just multiplying the total injected mass flow by an array input data ***A_rel_red[]*** (see §8.3.7.4).

The component also includes an *optional* non-equilibrium model for the calculation of products (time-delay parameters) and a model calculating the fuel evaporation (see §8.2.4.3).

8.2.8 ABS_Nozzle

This component represents a 1D supersonic nozzle in quasi-steady conditions. Two possibilities are foreseen in the main body of this component:

- an "ideal" nozzle using a variable gamma approximation
- a non-adiabatic, non-isentropic nozzle

The mathematical model explained below can be applied either to a complete nozzle or to a nozzle extension. The important point is to know the inlet total conditions calculated from the upstream enthalpy/entropy conditions and from the mass flow calculated in the complete nozzle component:

- If the nozzle is connected to a chamber (case of a complete nozzle), then the total conditions will be those of the exit of the chamber transmitted by the nozzle port
- If the nozzle is connected to another nozzle (case of a nozzle extension), then the total conditions will be those of the exit of the upwind nozzle. Inlet/outlet mass flows are the same

The throat calculation (choked mass flow, see below) will only be implemented in the complete nozzle.

8.2.8.1 Throat calculation

The choked throat conditions (P_{th}, T_{th} and vel_{th}) can be calculated with the following three equations assuming that the total enthalpy and the static entropy are known (those of the last combustor volume):

$$h_{ch,tot} = h(N_{k,th}, T_{th}) + vel_{th}^{\,2}/2$$
$$s_{ch} = s(N_{k,th}, P_{th}, T_{th})$$
$$vel_{th} = \sqrt{\gamma_{th} R_{th} T_{th}}; \qquad m_{th} = A_{th} vel_{th} P_{th}/(R_{th} T_{th})$$

where:

γ_{th} : Isentropic coefficient. This is a function of P_{th}, T_{th}, $N_{k,th}$ (see §8.2.8.4)
s : Entropy. This is a function of P_{th}, T_{th}, $N_{k,th}$ (see §4.4.1.1)
h : Static enthalpy. This is a function of T_{th}, $N_{k,th}$ (see §4.4.1.1)

$N_{k,th}$ is the number of moles of the chemical constituent k of the burned gases at throat. It is calculated from the Minimum Gibbs energy method:

$$(N_k, T)_{th} = f_{minGibbs}(N_{k,ch}, s_{ch}, P_{th})$$

where sub index "ch" denotes the conditions at the exit of the combustor. Two possibilities are foreseen calling the previous function: *A/ Equilibrium* and *B/ frozen flow*. In the latter case, molar fractions remain constant. The equations above are solved iterating in pressure and in temperature.

To cover subsonic conditions, the effective throat mass flow is calculated using a dynamic momentum equation (see §5.3.2.1) where the left pressure corresponds to the combustor exit, and the right one to the external pressure. Of course, under normal steady conditions, the dynamic mass flow will be limited to the critical mass flow, m_{th}, previously calculated.

8.2.8.2 "Ideal" supersonic nozzle

The nozzle is divided into Nsup +1 sections. Assuming an isentropic *frozen* expansion between the nodes i and i+1, the temperature and pressure for each section can be calculated as follows:

$$\theta_i = \frac{T_{tot}}{T_i} = 1 + \left(\frac{\gamma_i - 1}{2}\right) \cdot M_i^{\,2}; \qquad \delta_i = \frac{P_{tot}}{P_i} = \theta_i^{\frac{\gamma_i}{\gamma_i - 1}}$$

where,

γ_i : Burned gases isentropic coefficient at section no. i. This is a function of T
M_i : Burned gases Mach number at section no. i
T_{tot} : Burned gases total temperature (input)
P_{tot} : Burned gases total pressure (input)

The closing condition to calculate the Mach number knowing the area ratio is:

$$M_i \frac{A_i}{A_{throat}} = \left(\frac{2\theta}{\gamma_i + 1} \right)^{\frac{\gamma_i + 1}{2(\gamma_i - 1)}}; \quad \gamma_i = f(T_i)$$

The equations above are solved iterating in Mach and in temperature.

8.2.8.3 Non-adiabatic, non-isentropic supersonic nozzle

It is assumed that the non-adiabatic process takes place in two separate steps:

A/ *Heat losses*. From section i to i+1, the following losses in enthalpy and entropy will take place:

$$h_{i+1,tot} = h_{i,tot} - q_{wall,i} / m_{th}$$
$$s_{i+1} = s_i - q_{wall,i} / m_{th} / T_i$$

where $q_{wall,i}$ is calculated in §8.2.3.5 (Bartz correlations). For the first nozzle station (i=1) the total enthalpy and the static entropy are known (those of the exit of upstream component).

B/ *Expansion*. Assuming now that the isentropic relations are valid we have:

$$h_{i+1,tot} = h(N_{k,i+1}, T_{i+1}) + vel_{i+1}^{\ 2} / 2$$
$$s_{i+1} = s(N_{k,i+1}, P_{i+1}, T_{i+1})$$
$$m_{th} = \rho_{i+1} vel_{i+1} A_{i+1}$$

$N_{k,i}$ is the number of moles of the chemical constituent k at the nozzle station i. It is calculated from the Minimum Gibbs energy method: $(N_k\ T)_{i+1} = f_{minGibbs}(N_{k,i}, s_i, P_{i+1})$

The three equations above with three unknowns (pressure, temperature and speed) are solved iterating in pressure and temperature. Two possibilities are foreseen after calling the previous function: *A/ Equilibrium* and *B/ frozen flow*. In the latter case, molar fractions remain constant.

8.2.8.4 Isentropic coefficient calculation

The following numeric derivatives are done to calculate the isentropic coefficient at equilibrium conditions:

$$\partial \ln v / \partial \ln T = 1 + \ln N_{tot,dT} / \ln(1+\varepsilon)$$
$$\partial \ln v / \partial \ln P = 1 + \ln N_{tot,dP} / \ln(1+\varepsilon)$$

where $N_{tot,dT}$, $N_{tot,dP}$ are the total number of moles (increments with respect to one) due to separate perturbations in P/T ($\Delta P = \varepsilon P$; $\Delta T = \varepsilon T$, ε=1e-4):

MinGibbsEnergy_PT(mix, N_k, T(1+ε), P, $N_{k,dT}$, $N_{tot,dT}$)
MinGibbsEnergy_PT(mix, N_k, T, P(1+ε), $N_{k,dP}$, $N_{tot,dP}$)

N_k is the number of moles of the chemical constituent k at equilibrium conditions. $N_{k,dT}$, $N_{k,dT}$ are the number of moles after perturbation in P, T. The calculation of the specific heat at constant pressure, cp, has two terms: cp_{fr} and cp_{re}. The frozen one is calculated in §4.4.1.1 and the reactive term is calculated as follows:

$$cp = cp_{fr} + cp_{re}$$
$$cp_{re} = \sum_k (\partial \ln N_k / \partial \ln T) N_k h_k / T / MW_{mix}$$

where:

$$\partial \ln N_k / \partial \ln T = (\ln(N_{k,dT} / MW_{mix,dT}) - \ln(N_k / MW_{mix})) / \ln(1+\varepsilon)$$

Then calculation of the specific heat at constant volume and the isentropic coefficient is:

$$cv = cp + 8.3144 \cdot (\partial \ln v / \partial \ln T)^2 / MW_{mix} / (\partial \ln v / \partial \ln P)$$
$$\gamma = -\frac{cp}{cv(\partial \ln v / \partial \ln P)}$$

In frozen conditions, $\partial \ln v / \partial \ln P$ is equal to one.

8.2.8.5 Heat exchanged with the walls

The same formulation is used as in §8.2.3.5. Ideal nozzles assume that the quantity of heat exchanged with the walls is small with respect to the enthalpy flow, so the isentropic equations are assumed to be still valid.

8.2.8.6 Thrust calculation

The expressions below are needed to calculate the thrust F and the ISP at section no. i:

$$F_i = m_{th} M_i \sqrt{\gamma_i R_{gas} T_i} + (P_i - P_{out}) A_i$$
$$ISP_i = F_i / m_{th}$$

where

P_{out}	:	External boundary condition in pressure
R_{gas}	:	Burned gases constant at nozzle exit
m	:	Mass flow at throat

8.2.9 Nozzle_Ext2

This component represents 1D supersonic nozzle with a film cooling fluid inlet. It is an aggregated component of an "ABS_Nozzle" and a Junction allowing disposing of a fluid port in which the film is injected.

Two possibilities are foreseen in the main body of this component:

- No interaction between the film cooling and the main flow of the nozzle
- A perfect mixing between two flows

In both cases, the static pressure inside the nozzle at the film inlet position is calculated as if there would not be any perturbation in the upstream nozzle flow:

$$P_{in} = P\_vs\_sT(N_{k,in}, s_{in}, T_{in})$$

"P_vs_sT" is a thermodynamic function calculating pressure from entropy and temperature. Index "in" refers to the nozzle station where the film is injected. At this point, the entropy is known by the connected upstream nozzle. Inlet temperature will depend on the film mixture option (see paragraphs below). The temperature of the injected film is calculated with the corresponding thermodynamic function using the CEA enthalpy reference of the injected flow:

$$T_{inj} = Tmix\_vs\_h(N_{k,inj}, h_{inj})$$

Once the nozzle inlet pressure P_{in} is known the film mass flow (including the sonic limitation) can be calculated by the aggregated Junction component (see §5.3.2).

8.2.9.1 No interaction option

It is assumed that there will not be any influence on the main nozzle flow. The inlet nozzle temperature will be the exit temperature of the upstream nozzle. The same applies for the enthalpy and the entropy of the first nozzle station.

8.2.9.2 Perfect mixture option

It is assumed that there will be a total mixture of two flows (that of the upstream nozzle and that of the injected film):

$$m_{mix} = m_{in\_noz} + m_{inj}; \quad h_{tot} = (mh_{in\_noz} + mh_{inj}) / m_{mix}$$

where $m_{in,noz}$ is the main mass flow calculated by the upstream connected nozzle. m_{inj} is the injected mass flow calculated by the Junction component. The implicit calculation of the mixture temperature is made throughout the following equation:

$$h_{mix} = h(N_{k,mix}, T_{in}) + \frac{1}{2}\left(\frac{m_{mix}}{P_{in}/(R_{in}T_{in})}\right)^2$$

where $N_{k,mix}$ are the molar fractions of the mixture calculated in a similar way as in paragraph 8.2.3.4. Then, the entropy of the mixture can be calculated:

$$s_{mix} = h(N_{k,mix}, P_{in}, T_{in})$$

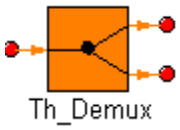
Finally, these enthalpy and entropy values are associated with the first station of nozzle, so that the same equations as in §8.2.8 will apply to calculate each one of the nozzle stations.

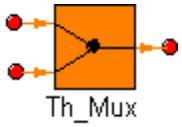
8.3 OPERATIONAL COMPONENTS

8.3.1 Thermal Mux/Demux

8.3.1.1 Description

Thermal demultiplexers allow splitting a thermal port into two different thermal ports and vice versa. See §8.1.2.2 with an example of application.





8.3.1.2 Construction Parameters

Name	Type	Description
n1	CONST INTEGER	First outlet (inlet for the Th_mux) thermal port dimension
n1	CONST INTEGER	Second outlet (inlet for the Th_mux) thermal port dimension

Note 1: n1, n2 determine the number of wall thermal nodes of each one of the split/joined thermal ports. The third thermal port must be dimensioned with n=n1+n2.

Note 2: If a *Th_Demux* component is connected to a Chamber, the sum of n1, n2 must be Nsub+Nsup, i.e., the number of subsonic plus supersonic stations, see §8.3.1.5 and §8.3.5. Similarly, when collecting two thermal ports (*Th_Mux* component), the dimension of the outlet port (and the associated connected component) must be n1+n2.

8.3.1.3 Ports (Th_Demux)

Name	Type	Parameters	Direction	Description
tp_in	Thermal	(n =n1+ n2)	IN	Inlet thermal port
tp_out1		(n = n1)	OUT	First outlet thermal port
tp_out2		(n = n2)	OUT	Second outlet thermal port

8.3.1.4 Ports (Th_Mux)

Name	Type	Parameters	Direction	Description
tp_in1	Thermal	(n = n1)	IN	First inlet thermal port
tp_in2		(n = n2)	IN	Second inlet thermal port
tp_out		(n =n1+ n2)	OUT	Outlet thermal port

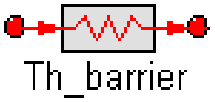
8.3.1.5 Formulation

No equations are added to this component. Heat flux and temperatures contained in a vectorized thermal port are split/joined into two thermal ports.

8.3.2 Th_barrier

8.3.2.1 Description

This component simulates a thermal barrier (coating) between the combusted gases and the chamber wall.



8.3.2.2 Construction Parameters

Name	Type	Description
Nsub	INTEGER	Number of chamber fluid sections
Nsup	INTEGER	Number of nozzle fluid sections

8.3.2.3 Ports

Name	Type	Parameters	Direction	Description
tp_in	thermal	(n = Nsub+Nsup)	IN	Thermal inlet port
tp_out		(n = Nsub+Nsup)	OUT	Thermal outlet port

8.3.2.4 Data

Name	Type	Description	Units
th	REAL	Thermal barrier thickness (m)	m
Dt	REAL	Chamber throat diameter	m
Lc	REAL	Chamber length of subsonic part. To normalize subsonic axial position	m
Dc_vs_L	TABLE 1D	Normalized subsonic chamber diameters vs normalized axial position	-
dxc_input	BOOLEAN	FALSE, dxc_vs_L = weighting function for subsonic mesh size distribution. TRUE, normalized node lengths vs node number	-
dxc_vs_L	TABLE 1D	Weighting function for subsonic mesh size distribution or normalized node lengths vs node number	-
Ld	REAL	Total length of supersonic part. To normalize axial position	m
Ld1	REAL	Nozzle axial length from throat to coating inlet position (0 if Lc > 0)	m
Ld2	REAL	Nozzle axial length from throat to coating outlet position	m
Dd_vs_L	TABLE 1D	Normalized nozzle diameters vs normalized axial position	-
dxd_input	BOOLEAN	FALSE, dxc_vs_L = weighting function for supersonic mesh size distribution. TRUE, normalized node lengths vs node number	-
dxd_vs_L	TABLE 1D	Weighting function for supersonic mesh size distribution or normalized node lengths vs node number	-
To	REAL	Initial temperature (K)	K
mat	ENUM	Material of the thermal barrier	
rho	REAL	Density of the thermal barrier if mat=None (kg/m^3)	kg/m^3
cp	REAL	Specific heat of the thermal barrier if mat=None (J/kg*K)	J/kg*K
k	REAL	Conductivity of the thermal barrier if mat=None (W/m*K)	W/m*K

Coating materials are selected via the ENUMERATIVE variable "mat" in which a list of materials is provided (see §3.9).

Chamber diameters, wet areas and mesh size of each node are function of the axial position through the non-dimensional geometry tables. *See §8.1.2.2 for more details.*

8.3.2.5 Formulation

Heat transfer is calculated in a similar way than in §6.3.1.5 for a cylindrical domain but with only one thermal diffusive node along the thickness for each chamber station (axial discretization with variable diameters).

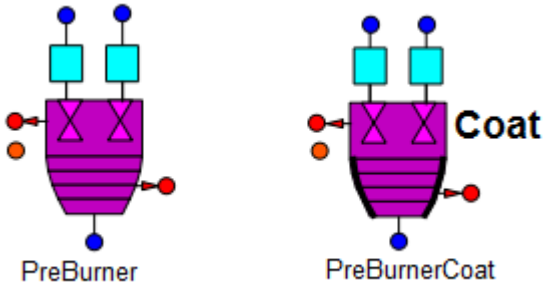
For each axial node, the conductivity equation (using the specific properties of the thermal barrier) calculates the heat flux between the coating temperature (assumed to be the one calculated by the diffusive equation) and the external one calculated by the connected thermal component simulating the chamber wall.

The internal heat flux used in the heat diffusive equation will be calculated in the chamber (Bartz correlations).

8.3.3 PreBurner

8.3.3.1 Description

These components represent non adiabatic 1D Preburners for liquid or gas propellants built by means of a combustor (equilibrium or "rate" model), two injectors and two cavities. They can be connected to any FLUID_FLOW_1D component. The properties and composition of combusted gases can be transmitted by the outlet fluid port to another possible combustor (staged engines) or turbines.



Combustor components are doubled depending on whether or not they have a thermal coating protection (_Coat).

8.3.3.2 Construction parameters

Name	Type	Description	
Nsub	CONST INTEGER	Number of nodes along the chamber axial direction	
Inj_redGases	SET_OF(Chemicals)	Select Chemicals only for red_injector downstream of a burner	
Inj_oxyGases	SET_OF(Chemicals)	Select Chemicals only for oxy_injector downstream of a burner	
GasLiqOption	ENUM DropExchange	"_rate" models	UserDef, user-defined characteristic vaporization time
liquidExitAllowed	BOOLEAN		TRUE, it allows exit of liquids
rateOption	BOOLEAN		TRUE, burning rate model active

Notes:

1. Nsub determines the number of fluid and wall thermal nodes
2. Select burnerGasesOption = Chemicals only for injectors placed downstream of another combustor. Keep burnerGasesOption = noBurnGases (default value) in other cases.
3. Combustor components have two new options: GasLiqOption concerning the propellant vaporization models and rateOption to choose equilibrium or reaction delay method, see §8.2.4:
   - GasLiqOption = Advanced: Convection of liquids and its vaporization is calculated. Liquid droplets size is tuned in time and position through the "f_v[]" factors.
   - GasLiqOption = UserDefined: No convection of liquid is calculated. Vaporization is assumed to be within a delay time after the injection plate if the ignition order is given (boundary IgnitFlag=1), or null if no ignition order is given.
   - rateOption = FALSE: All vapors will react instantaneously.
   - *rateOption = TRUE*: this is a first approach of a non-equilibrium chamber based on a time-delay between the equilibrium and the actual burned gases composition.
4. Combustor components also have the option *liquidExitAllowed* to include or not the amount of liquid exiting a chamber in the gas mixture passing by turbines and other FLUID_FLOW components.
   - Use this option if the amount of liquid is not very big. Note that the mixture exiting a combustor is calculated accordingly with the perfect gas or Van der Waals state of equation, so temperatures below the triple point are difficultly supported. See §8.1.2.5

8.3.3.3 Ports

Name	Type	Parameters	Direction	Description
f_out	fluid	Chemicals (fixed)	OUT	Outlet combustion gases port
f_oxy		burnerGasesOption	IN	Inlet oxidizer port
f_red		burnerGasesOption	IN	Inlet fuel port
tp	thermal	(n = Nsub)	OUT	Thermal port
tp_inj	thermal	(n=1)	OUT	Injector thermal port
s_pres	control	(n=1)	OUT	Chamber pressure signal

The outlet port can only be connected to a *FLUID_FLOW type* component (fluid type port).

8.3.3.4 Data

Name	Type	Description	Units
Dt	REAL	Chamber throat diameter	m
Rcurv	REAL	Curvature radius at throat	m
Lc	REAL	Chamber length of subsonic part	m
dx_input	BOOLEAN	Mesh size distribution option. See note 1	-
Dc_vs_L	TABLE 1D	Normalized subsonic chamber diameters vs normalized axial position	-
dx_vs_L		Weighting function for mesh size distribution or normalized node lengths vs node number, depending on "dx_input" data	-
P_ch	REAL	Initial Chamber pressure	Pa
T_ch	REAL	Initial Chamber temperature	K
x_nco	REAL	Initial non-condensable mass fraction*, see §8.1.2.3*	-
Tcav_oxy	REAL	Initial oxidizer cavity temperature	K
Tcav_red	REAL	Initial reducer cavity temperature	K
A_inj_oxy	REAL	Effective injection area for oxidizer	m^2
A_inj_red	REAL	Effective injection area for gas reducer	m^2
V_cav_oxy	REAL	Oxidizer cavity volume	m^3
V_cav_red	REAL	Reducer cavity volume	m^3
capa	REAL	Chamber to Cavities heat capacity	J/K
cond	REAL	Chamber to Cavities conductance	W/K
emiss	REAL	Emissivity of the combustion gases	-
MR_ini	REAL	Initial mixture ratio; > 0, ignited chamber initialization at P_ch, T_ch	-
MR_min	REAL	Minimum Mixture Ratio allowing combustion	-
MR_max	REAL	Maximum Mixture Ratio allowing combustion	-
eta	REAL	Combustor efficiency (only for equilibrium components)	-
starter_y[]	REAL	Mass fractions of the starter gases (H2O,CO,CO2,N2,H2,He)	-
tau_b	for RateOption and GasLiqOption	Burning characteristic time, see note 4	s
tau_c		Ignition time delay	s
D_dr_fu		Fuel droplets nominal size	m
D_dr_ox		Oxidizer droplets nominal size	m
tau_v		Vaporization characteristic time, see note 5	s
th_coat	for "Coat" models	Thermal barrier thickness	
mat_coat		Material of the thermal barrier	-
k_coat		Conductivity of the thermal barrier if mat=None	W/m*K
cp_coat		Specific heat of the thermal barrier if mat=None	J/kg*K
rho_coat		Density of the thermal barrier if mat=None	kg/m^3

Notes:

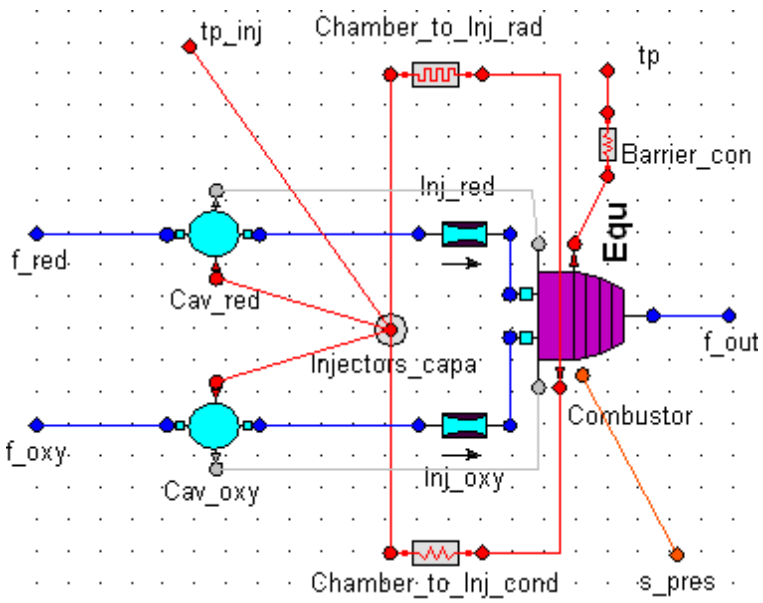
1. *Dc_vs_L, dx_vs_L* tables allow variable geometry, variable mesh size in the chamber. If dx_input = FALSE, "dx_vs_L" is a table with the relative mesh size vs. axial position. If dx_input = TRUE, "dx_vs_L" is a table with the normalized node lengths vs node number. The discretized (Nsub) wet

areas and diameters will be interpolated using these tables from L=0 to L=Lc. Dt is used for the normalization of diameters and Lc to that of the axial lengths**.** *See §8.1.2.2 for more details*

2. The mass fractions of the starter gases are input data (starter_y[] variable) within a predefined set of chemicals {H2O, CO, CO2, N2, H2, He} where most of the starter gases produced in a particular starter powder are included. By default this composition is: {0.26, 0.19, 0.07, 0.21, 0.27, 0} [RD-44]
3. Ignition flag and starter gases mass flow are not input data but boundary conditions automatically loaded in the default experiment. By default, ignition is activated with *no* starter gases (see §8.1.2.3 & §8.2.3.4).
4. Very low values (10^{-6} seconds) of the burning and vaporization times (only for the "Rate" model) result in the equilibrium Combustor component behavior
5. "tau_v" is used to prevent low temperatures at ignition injecting liquids, see §8.2.3.3. "_rate" models calculate this time if GasLiqOption=Advanced, see §8.2.4.2
6. The vaporization factors f_v[], see §8.1.2.5 and §8.2.4.2, are boundary conditions automatically loaded in the default experiment scaling the nominal droplets size, and thus the vaporization rate. By default, they are set to 1.

8.3.3.5 Topology

Below, the schematic of a *PreBurneCoat* component with a thermal barrier is shown. Combustor components without a coating barrier do not have the "Barrier_con" component.



8.3.3.6 Formulation

No special formulation has been added to these topological components that gather all the capabilities of the combustors, injectors and cavities. See Sections 8.2.1 to 8.2.5

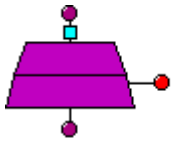
The thermal components included allow the calculation of thermal fluxes between the chamber and the cavities. The main thermal fluxes at chamber walls are simulated by the "tp" thermal port which can be connected to a Cooling Jacket for example.

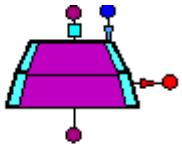
8.3.4 Nozzle Extension

8.3.4.1 Description

These components represent non-adiabatic, non-isentropic 1D supersonic nozzle in quasi-steady conditions. The nozzle port allows the concatenation of several nozzle extensions, each one having

possible different thermal connections. Nozzle_Ext2 behaves as a nozzle extension but includes a fluid inlet port allowing simulating a film cooling injection.





8.3.4.2 Construction parameters

Name	Type	Description	
Nsup	CONST INTEGER	Number of supersonic nodes	
type	ENUM NozzleType	Calculation option. "*IdealNozzle*" or "*NonIsentropic*"	
burnerGasesOption	SET_OF (Chemicals)	For Nozzle_Ext2	Type of mixture of fluid
FilmType	ENUM		Film cooling calculation option

Select *burnerGasesOption = Chemicals* only for nozzle injections placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

Note: In normal situations we recommend using the NonIsentropic option which is more robust and precise than the ideal option.

8.3.4.3 Ports

Name	Type	Parameters	Direction	Description
np_in	NozzlePort		IN	Inlet combusted gases port
np_out	NozzlePort		OUT	Outlet combusted gases port
f_film	fluid	burnerGasesOption	OUT	Film cooling inlet fluid port
tp	thermal	(n = Nsup)	OUT	Thermal port
desing_noz	signal	(n=2)		Input output variables

Note 1: Nozzle type ports can only be connected to a nozzle type component. The film injection port must be connected to a *FLUID_FLOW type* component (fluid type port).

Note 2: "desing_noz" port contains the following input output signal values:

- desing_noz.signal[1] = Ivac_nozzle[Nsup];
- desing_noz.signal[2] = Thrust
- desing_noz.signal[3] = AR (exit/throat area ratio);
- desing_noz.signal[4] = Ld (total nozzle length)

8.3.4.4 Data

Name	Type	Description	Units
Eta	REAL	Nozzle efficiency	-
Frozen_th	BOOLEAN	Flag forcing frozen conditions in the throat	-
Frozen_nz	BOOLEAN	Flag forcing frozen conditions in the supersonic nozzle	-
emiss	REAL	Emissivity of the combustion gases	-
D_throat	REAL	Throat diameter	m
Rcurv	REAL	Curvature radius at throat	m
Ld	REAL	Total length of supersonic part. To normalize axial position	m
Ld1	REAL	Nozzle axial length from throat to inlet	m
Ld2	REAL	Nozzle axial length from throat to outlet	m
dx_input	BOOLEAN	Mesh size distribution option. See note 1	-
Dd_vs_L	TABLE 1D	Normalized nozzle diameters vs normalized axial position	-
dx_vs_L		Weighting function for mesh size distribution or normalized node lengths vs node number, depending on "dx_inpu"t data	-
AR	REAL	Exit/throat area ratio: 0, no Dd_vs_L profile modification	-
P_ini	REAL	Initial pressure for iterative calculations	Pa
T_ini	REAL	Initial temperature for iterative calculations	K

Name	Type	Description	Units
w_film	REAL (only for Nozzle_Ext2 comp.)	Width of the film cooling channel at inlet (m)	m
x_jun_in		Inlet junction X coordinate relative to a body axis system	m
y_jun_in		Inlet junction y coordinate relative to a body axis system	m
z_jun_in		Inlet junction z coordinate relative to a body axis system	m

Notes:

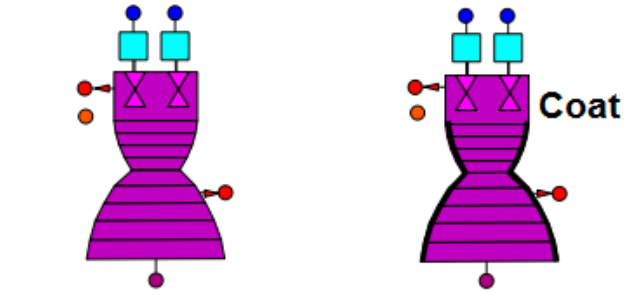
1. Dd_vs_L, dx_vs_L tables allow variable geometry, variable mesh size in the nozzle. If dx_input = FALSE, "dx_vs_L" is a table with the relative mesh size vs. axial position. If dx_input = TRUE, "dx_vs_L" is a table with the normalized node lengths vs node number. The discretized (Nsup) wet areas and diameters will be interpolated using these tables from L=Ld1 to Ld2. D_throat is used for the normalization of diameters and Ld to that of the axial lengths. See §8.1.2.2.
2. AR is an input modifying the exit/throat area ratio of a given profile without changing the relative shape. For a conic nozzle this is equivalent to changing the nozzle angle.
3. frozen_th, frozen_nz = FALSE means equilibrium composition. frozen_nz = FALSE is only available for NonIsentropic nozzles. Ideal nozzles always use frozen conditions.

This component incorporates the mass, energy and momentum equations in a steady regime according to an area-varying non-uniform mesh1D spatial discretization. *See §8.2.8* and *§8.2.9.*

8.3.5 CombustChamberNozzle

8.3.5.1 Description

These components represent non adiabatic 1D combustor-nozzle chambers for liquid or gas propellants built by means of a combustor (equilibrium or "rate" model), a nozzle, two injectors and two cavities.



CombustorChamberNozzle components are doubled depending on whether or not they have a thermal coating protection (_Coat).

8.3.5.2 Construction parameters

Name	Type	Description	
Nsub	CONST INTEGER	Number subsonic nodes	
Nsup	CONST INTEGER	Number of supersonic nodes	
nozzle_type	ENUM NozzleType	Nozzle calculation option	
Inj_redGases	SET_OF(Chemicals)	Select Chemicals only for red_injector downstream of a burner	
Inj_oxyGases	SET_OF(Chemicals)	Select Chemicals only for oxy_injector downstream of a burner	
rateOption	BOOLEAN	For "_rate" models	TRUE: burning rate model active
GasLiqOption	ENUM DropExchange		UserDef: user-defined characteristic vaporization time

Notes:

1. Nsub+Nsup determine the number of fluid and thermal nodes. The "tp" port should be connected to a Cooling Jacket or to a diffusive thermal node, where the same amount of nodes must be set.

2. "nozzle_type" parameter has two possibilities: "IdealNozzle", an isentropic approximation, or "NonIsentropic", so a non-adiabatic approximation will be used.
3. Select *burnerGasesOption = Chemicals* only for injectors placed downstream of another combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.
4. These components have two options: GasLiqOption concerning the propellant vaporization models and rateOption to choose equilibrium or reaction delay method, see §8.2.4:
   - GasLiqOption = Advanced: Convection of liquids and its vaporization is calculated. Liquid droplets size is tuned in time and position through the "f_v[]" factors.
   - GasLiqOption = UserDefined: No convection of liquid is calculated. Vaporization is assumed to be within a delay time just after the injection plate if the ignition order is given (boundary IgnitFlag=1), or null if no ignition order is given.
   - rateOption = FALSE: All vapors will react instantaneously.
   - rateOption = TRUE: this is a first approach of a non-equilibrium chamber based on a time-delay between the equilibrium and the actual burned gases composition.

8.3.5.3 Ports

Name	Type	Parameters	Direction	Description
f_oxy	fluid	burnerGasesOption	IN	Inlet oxidizer port
f_red		burnerGasesOption	IN	Inlet fuel port
np_out	NozzlePort		IN	Outlet combusted gases port
tp	thermal	(n = Nsub+Nsup)	OUT	Thermal port to the Cooling Jacket
tp_inj	thermal	(n=1)	OUT	Injector thermal port
s_pres	control	(n=1)	OUT	Chamber pressure signal

The outlet port can only be connected either to a *Nozzle_Ext* component or be kept unconnected.

8.3.5.4 Data

Name	Type	Description	Units
Dt	REAL	Chamber throat diameter	m
Rcurv	REAL	Curvature radius at throat	m
Lc	REAL	Chamber length of subsonic part. To normalize axial position	m
dxc_input	BOOLEAN	Subsonic mesh size distribution option. See note 1	-
Dc_vs_L	TABLE 1D	Normalized subsonic chamber diameters vs normalized axial position	-
dxc_vs_L		Weighting function for subsonic mesh distribution or normalized node lengths vs node number, depending on "dxc_input" data	-
Ld	REAL	Total length of supersonic part. To normalize axial position	m
Ld2	REAL	Nozzle axial length from throat to outlet	m
dxd_input	BOOLEAN	Supersonic mesh size distribution option. See note 1	-
Dd_vs_L	TABLE 1D	Normalized nozzle diameters vs normalized axial position	-
dxd_vs_L		Weighting function for supersonic mesh distribution or normalized node lengths vs node number, depending on "dxd_input" data	-
AR_sup	REAL	Nozzle Exit/throat area ratio: 0, no Dd_vs_L profile modification	-
Eta	REAL	Nozzle efficiency	-
frozen_th	BOOLEAN	Flag forcing frozen conditions in the throat	-
Frozen_nz	BOOLEAN	Flag forcing frozen conditions in the supersonic nozzle	-
P_ch	REAL	Initial Chamber pressure	Pa
T_ch	REAL	Initial Chamber temperature	K
x_nco	REAL	Initial non-condensable mass fraction, *see §8.1.2.3*	-
Tcav_oxy	REAL	Initial oxidizer cavity temperature	K
Tcav_red	REAL	Initial reducer cavity temperature	K
A_inj_oxy	REAL	Effective injection area for oxidizer	m^2
A_inj_red	REAL	Effective injection area for gas reducer	m^2

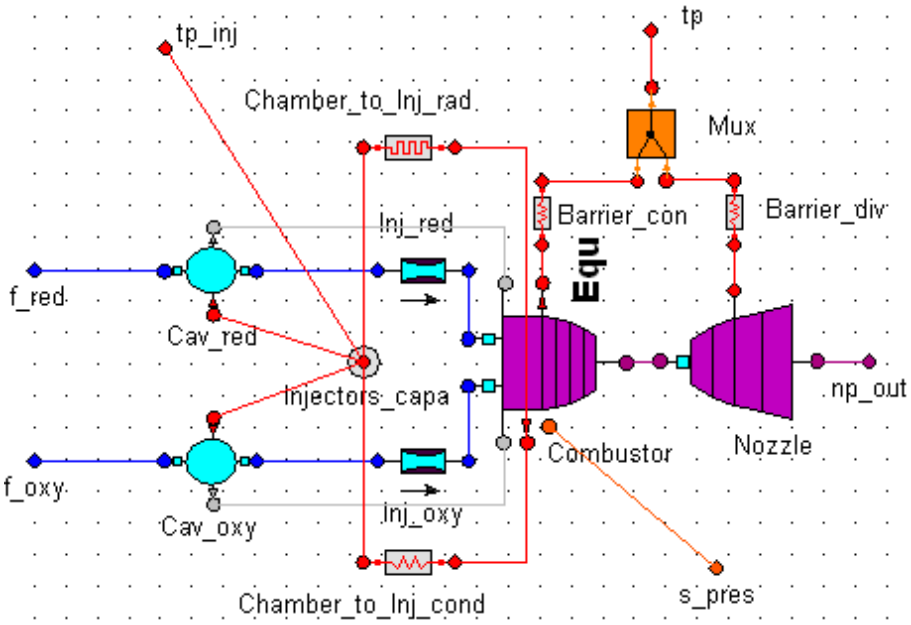
Name	Type	Description	Units
V_cav_oxy	REAL	Oxidizer cavity volume	m^3
V_cav_red	REAL	Reducer cavity volume	m^3
capa	REAL	Chamber to Cavities heat capacity	J/K
cond	REAL	Chamber to Cavities conductance	W/K
emiss	REAL	Emissivity of the combustion gases	-
MR_ini	REAL	Initial mixture ratio; > 0, ignited chamber at P_ch, T_ch conditions	-
MR_min	REAL	Minimum Mixture Ratio allowing combustion	-
MR_max	REAL	Maximum Mixture Ratio allowing combustion	-
eta	REAL	Combustor efficiency (only for equilibrium components)	-
starter_y[]	REAL	Mass fractions of the starter gases (H2O,CO,CO2,N2,H2,He)	-
tau_b	for RateOption and GasLiqOption	Burning characteristic time, see note 4	s
tau_c		Ignition time delay	s
D_dr_fu		Fuel droplets nominal size	m
D_dr_ox		Oxidizer droplets nominal size	m
tau_v		Vaporization characteristic time, see note 5	s
th_coat	for "Coat" models	Thermal barrier thickness	m
mat_coat		Material of the thermal barrier	-
k_coat		Conductivity of the thermal barrier if mat=None	W/m*K
cp_coat		Specific heat of the thermal barrier if mat=None	J/kg*K
rho_coat		Density of the thermal barrier if mat=None	kg/m^3

Notes:

1. *Dc_vs_L, dxc_vs_L* and *Dd_vs_L, dxd_vs_L* tables allow variable geometry, variable mesh size in the chamber and in the nozzle respectively. If dx.._input = FALSE, "dx..._vs_L" is a table with the relative mesh size vs. axial position. If dx.._input = TRUE, "dx..._vs_L" is a table with the normalized node lengths vs node number. The discretized (Nsub+Nsup) wet areas and diameters will be interpolated using these tables from L=0 to L=Lc in the subsonic zone and from L=0 to L=Ld2 in the supersonic zone. Dt is used for the normalization of diameters and Lc, Ld to the normalization of the subsonic, supersonic axial lengths respectively. *See §8.1.2 for more details.*
2. The mass fractions of the starter gases are input data (starter_y[] variable) within a predefined set of chemicals {H2O, CO, CO2, N2, H2, He} where most of the starter gases produced in a particular starter powder are included. By default, this composition is: {0.26, 0.19, 0.07, 0.21, 0.27, 0} [RD-44]
3. Ignition flag and starter gas mass flow are not input data but boundary conditions automatically loaded in the default experiment. By default, ignition is activated with *no* starter gases (see §8.1.2.3 & §8.2.3.4).
4. Very low values (10^{-6} seconds) of the burning and vaporization times (only for the "Rate" model) result in the equilibrium Combustor component behavior.
5. "tau_v" is used in to prevent low temperatures at ignition injecting liquids, see §8.2.3.3. "_rate" models calculate this time if GasLiqOption=Advanced, see §8.2.4.2.
6. The vaporization factors f_v[], see §8.1.2.5 and §8.2.4.2, are boundary conditions automatically loaded in the default experiment scaling the nominal droplets size, so the vaporization rate. By default, they are set to 1.
7. frozen_th = FALSE means equilibrium composition at throat. frozen_nz option = FALSE (varying equilibrium composition in the supersonic sections) is only available for NonIsentropic nozzles. Ideal nozzles always use frozen conditions.
8. AR is a new input to modify the aspect ratio of a given profile without changing the relative shape. For a conic nozzle this is equivalent to changing the nozzle angle.

8.3.5.5 Topology

Shown below is the schematic of a *CombustChamberNozzleCoat* component with a thermal barrier. Combustor components without any coating barrier do not have the "Barrier_con" component.



8.3.5.6 Formulation

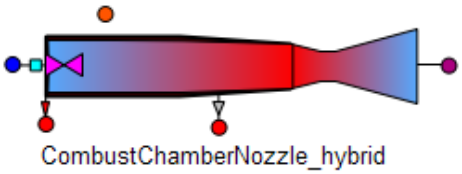
No special formulation has been added to these topological components that gather all the capabilities of the combustors, injectors, cavities and nozzles. See Sections 8.2.1 to 8.2.5.

The thermal components included allow the calculation of thermal fluxes between the chamber and the cavities. The main thermal fluxes at chamber walls are simulated by the "tp" thermal port which can be connected to a Cooling Jacket for example.

8.3.6 CombustChamberNozzle_hybrid

8.3.6.1 Description

This component represents non-adiabatic 1D combustor-nozzle chamber for *solid and hybrid propellants* built by means of a hybrid combustor, an oxidizer injector with its cavity and a Nozzle component..



8.3.6.2 Construction parameters

Name	Type	Description
Nsub	CONST INTEGER	Number subsonic nodes
Nsup	CONST INTEGER	Number of supersonic nodes
nozzle_type	ENUM NozzleType	Nozzle calculation option
Inj_oxyGases	SET_OF(Chemicals)	Type of mixture of fluids
rateOption	BOOLEAN	TRUE: burning rate model active
grainCons	BOOLEAN	If "FALSE, the grain consumption is deactivated, so that the grain thickness will remain constant for particular studies

Notes:

1. Nsub+Nsup determine the number of fluid and thermal nodes. The "tp" port should be connected to a diffusive thermal node, where the same amount of nodes must be set.

2. "nozzle_type" parameter has two possibilities: "IdealNozzle", an isentropic approximation, or "NonIsentropic", so a non-adiabatic approximation will be used.
3. Select *Inj_oxyGases = Chemicals* only for injectors placed downstream of another combustor. Keep *Inj_oxyGases = noBurnGases* (default value) in other cases.
4. Two options are available to choose equilibrium or reaction delay method, see §8.2.4:
   - rateOption = FALSE: All vapors will react instantaneously just when they are released from the grain or the droplets (oxidizer).
   - rateOption = TRUE: this is a first approach of a non-equilibrium chamber based on a time-delay between the equilibrium and the actual burned gases composition.

8.3.6.3 Ports

Name	Type	Parameters	Direction	Description
f_oxy	fluid	burnerGasesOption	IN	Inlet oxidizer port
np_out	NozzlePort		IN	Outlet combusted gases port
tp	thermal	(n = Nsub+Nsup)	OUT	Wall thermal port
tp_inj	thermal	(n=1)	OUT	Injector thermal port
s_pres	control	(n=1)	OUT	Chamber pressure signal

The outlet port can only either be connected to a *Nozzle_Ext* component or be kept unconnected.

8.3.6.4 Data

Name	Type	Description	Units
Dt	REAL	Chamber throat diameter	m
Rcurv	REAL	Curvature radius at throat	m
Lc	REAL	Chamber length of subsonic part. To normalize axial position	m
dxc_input	BOOLEAN	Subsonic mesh size distribution option. See note 1	-
Dc_vs_L	TABLE 1D	Normalized subsonic chamber diameters vs normalized axial position	-
dxc_vs_L		Weighting function for subsonic mesh distribution or normalized node lengths vs node number, depending on "dxc_input" data	-
Ld	REAL	Total length of supersonic part. To normalize axial position	m
Ld2	REAL	Nozzle axial length from throat to outlet	m
dxd_input	BOOLEAN	Supersonic mesh size distribution option. See note 1	-
Dd_vs_L	TABLE 1D	Normalized nozzle diameters vs normalized axial position	-
dxd_vs_L		Weighting function for supersonic mesh distribution or normalized node lengths vs node number, depending on "dxd_input" data	-
AR_sup	REAL	Nozzle Exit/throat area ratio: 0, no Dd_vs_L profile modification	-
Eta	REAL	Nozzle efficiency	-
frozen_th	BOOLEAN	Flag forcing frozen conditions in the throat	-
Frozen_nz	BOOLEAN	Flag forcing frozen conditions in the supersonic nozzle	-
P_ch	REAL	Initial Chamber pressure	Pa
T_ch	REAL	Initial Chamber temperature	K
x_nco	REAL	Initial non-condensable mass fraction (-)	-
A_inj_oxy	REAL	Effective injection area for oxidizer	m^2
V_cav_oxy	REAL	Oxidizer cavity volume	m^3
capa	REAL	Chamber to Cavities heat capacity	J/K
cond	REAL	Chamber to Cavities conductance	W/K
emiss	REAL	Emissivity of the combustion gases	-
eta	REAL	Combustor efficiency (only for rateOption = FALSE)	-
tau_b	for "Rate" models	Burning characteristic time, see note 4	s
tau_c		Ignition time delay	s
D_dr_ox		Oxidizer droplets nominal size	m
ht_option	ENUM	Solid propellant-internal fluid heat transfer option	-
hc_dat	REAL	Heat transfer coefficient – only if ht_option = Ht_constant	W/m^2*K

Name	Type	Description	Units
rug	REAL	Equivalent grain rugosity	m
k_f	REAL	Multiplier of the friction factor	-
GasSolOption	See §8.2.6 and note 2	If stdHybrid/stdSolid: user defined regresion rate constants: $r_{sp,i} = a_{sp} G_i^{a_{sp}}\ (GasSolOption = stdHybrid)$; $r_{sp,i} = a_{sp} P_i^{a_{sp}}\ (GasSolOption = stdSolid)$ If vapModel, no use of a_sp, b_sp constants, see §8.2.6	-
a_sp		Solid propellant regression rate constant	m^3/kg
b_sp		Solid propellant regression rate exponent	-
LH_sp		Enthalpy of formation of UsrDef rubber, RubUsr	J/kg
Tsat_sp		Solid propellant evaporation temperature	K
k_sp		Solid propellant thermal conductivity	W/m*K
rho_sp		Solid propellant density	Kg/m3
rubComp[]		Mass fractions of solid propellant *constituents* – HTPB, IPDI, RubUsr, KNO3_a, Al_cr, S_a, NH4NO3_IV, NH4CLO4_I	-
rubUsrForm[]		RubUsr's formula according to. the following atoms list -H, O, S, N, C, Ar, He, Al, K, Cl-	
th_sp_ini	REAL	Solid propellant initial thickness	m
th_vs_L	TABLE_1D	Dimensionless solid propellant thickness vs dimensionless axial position. *Using non-homogeneous grain thickness along the chamber allows to design particular thrust profiles*	-

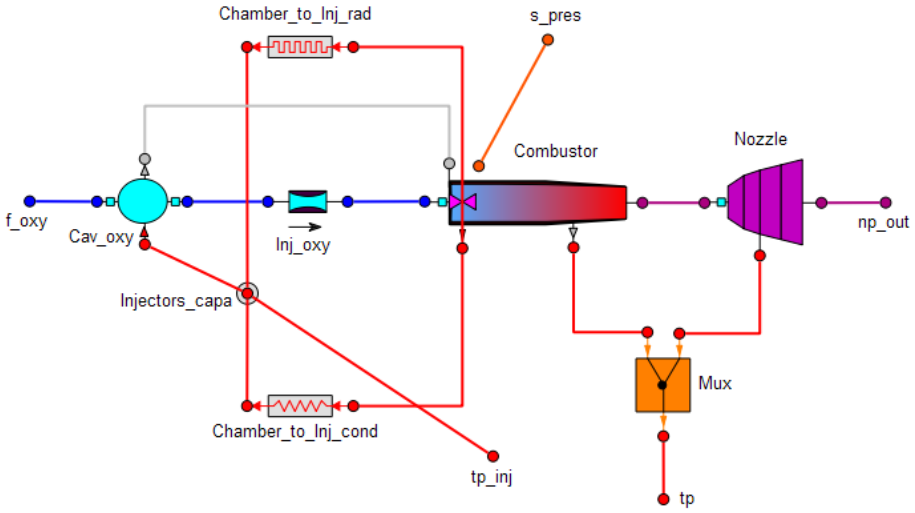
Notes:

1. *Dc_vs_L, dxc_vs_L* and *Dd_vs_L, dxd_vs_L* tables allow variable geometry, variable mesh size in the chamber and in the nozzle respectively. If dx..._input = FALSE, "dx..._vs_L" is a table with the relative mesh size vs. axial position. If dx..._input = TRUE, "dx..._vs_L" is a table with the normalized node lengths vs node number. The discretized (Nsub+Nsup) wet areas and diameters will be interpolated using these tables from L=0 to L=Lc in the subsonic zone and from L=0 to L=Ld2 in the supersonic zone. Dt is used for the normalization of diameters and Lc, Ld to the normalization of the subsonic, supersonic axial lengths respectively. *See §8.1.2 for more details.*
2. The mass fractions of the of the solid propellant *constituents* are input data (**rubComp**[] variable) within a predefined set of constituents {HTPB, IPDI, RubUsr, KNO3_a, Al_cr, S_a, NH4NO3_IV, NH4CLO4_I}.

   Note that the solid propellant "constituents" are reactants. Two predefined rubbers are considered: HTPB (C10H15.4O0.07) and IPDI (C12H18O20.07N2). For a user defined rubber, select a mass fraction for "**RubUsr**" type, introduce its formula within the "**rubUsrForm**" input data array and set "LH_sp" as the enthalpy of formation of the UsrDef rubber, normally low with respect to the reaction heat calculated by the code.
3. Ignition flag is not input data but boundary condition automatically loaded in the default experiment. By default ignition flag is activated with *no* starter gases. The starter gases (ready to react if the ignition is activated) have the same composition as the solid propellant constituents, so the boundary variable *starter_m* (see §8.1.2.3) *represents an additional mass flow of solid propellant constituents apart from those derived from the regression law*.
4. The grain factors f_s[], see §8.2.6.2, are boundary variables automatically loaded in the default experiment to modulate the grain/fluid interface area, *thus the grain consumption rate*. By default, they are set to 1 (circular shape).
5. The vaporization factors f_v[], see §8.1.2.5 and §8.2.4.2, are boundary conditions automatically loaded in the default experiment scaling the nominal droplets size (so the vaporization rate) in case of oxidizer liquid injection. By default, they are set to 1.
6. frozen_th = FALSE means equilibrium composition at throat. frozen_nz option = FALSE (varying equilibrium composition in the supersonic sections) is only available for NonIsentropic nozzles. Ideal nozzles always use frozen conditions.
7. AR is an input to modify the aspect ratio of a given profile without changing the relative shape. For a conic nozzle this is equivalent to changing the nozzle angle.

8.3.6.5 Topology

Shown below is the schematic of a *CombustChamberNozzle_hybrid* component.



8.3.6.6 Formulation

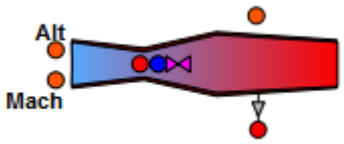
No special formulation has been added to this topological component that gathers all the capabilities of the hybrid combustors and nozzles. See §8.2.6.

The thermal components included allow the calculation of thermal fluxes between the chamber and the cavities. The main thermal fluxes at chamber walls are simulated by the "tp" thermal port which can be connected to a thermal network.

8.3.7 CombustChamber_ramjet

8.3.7.1 Description

This component represents non equilibrium, non-adiabatic 1D ramjet/scramjet combustors for air-breathing engines built by means of an intake, a supersonic/subsonic combustor and a fuel injector with its cavity components.



8.3.7.2 Construction parameters

Name	Type	Description
Inj_redGases	SET_OF (Chemicals)	Select Chemicals only for fuel injector downstream of a burner
nodes	INTEGER	Number of nodes
rateOption	BOOLEAN	TRUE: burning rate model active
givenOutletPress	BOOLEAN	"TRUE, Static outlet pressure different than the far inlet pressure

Notes:

1. *nodes* determine the number of fluid and thermal nodes along the axial direction. The "tp" port should be connected to a diffusive thermal node, where the same amount of nodes must be set.
2. Select Inj_redGases = *Chemicals* only for fuel injectors placed downstream of another combustor. Keep *Inj_oxyGases = noBurnGases* (default value) in other cases.
3. Two options are available to choose equilibrium or reaction delay method, see §8.2.4:

- rateOption = FALSE: The fuel vapors will react instantaneously just when they are injected (or vaporized in case of liquid injection).
- rateOption = TRUE: this is a first approach of a non-equilibrium chamber based on a time-delay between the equilibrium and the actual burned gases composition.

8.3.7.3 Ports

Name	Type	Parameters	Direction	Description
f_red	fluid	burnerGasesOption	IN	Inlet fuel port
s_alt	control	(n=1)	IN	Altitude
s_mach	control	(n=1)	IN	Mach number
tp	thermal	(n = nodes)	OUT	Thermal port of the chamber
tp_inj	thermal	(n=1)	OUT	Injector thermal port

Signals "s_mach" and "s_alt" give the flight conditions to be imposed at the inlet of the intake. The static pressure, temperature and velocity will be calculated and applied as initial conditions in the combustor.

8.3.7.4 Data

Name	Type	Description	Units
TPR_table	TABLE	Intake Total Pressure Recovery vs. flight Mach number	-
P_out	REAL	Outlet Pressure if givenOutletPress = TRUE	Ps
Lc	REAL	Chamber length. To normalize axial position	m
dx_vs_L	TABLE_1D	Mesh size distribution vs non-dimensional axial position	-
a	REAL	Reference cross section width	m
b	REAL	Reference cross section height	m
a_vs_L	TABLE_1D	Dimensionless section width vs dimensionless axial position	-
b_vs_L	TABLE_1D	Dimensionless section height vs dimensionless axial position	-
A_inj_red	REAL	Effective injection area for fuel	m^2
V_cav_red	REAL	Fuel injector cavity volume	m^3
capa	REAL	Chamber to Cavities heat capacity	J/K
cond	REAL	Chamber to Cavities conductance	W/K
emiss	REAL	Emissivity of the combustion gases	-
eta	REAL	Combustor efficiency (only for equilibrium components)	-
rug	REAL	Rugosity	m
k_f	REAL	Multiplier of the friction factor	-
starter_y[]	REAL	Mass fractions of the starter solid propellant gases -H2O,CO,CO2,N2,H2,He-	-
tau_b	for "Rate" models	Burning characteristic time	s
tau_c		Ignition time delay	s
D_dr_fu	REAL	Characteristic fuel droplets size in case of liquid injection	m
MR_max	REAL	Maximum mixture ratio allowing ignition (minimum fuel concentration)	-
x_nco	REAL	Initial non-condensable mass fraction (-)	-
zetaf	REAL	Fuel injector loss coefficient; 0, calculated	-

Notes:

1. The default TPR table (input data) gives very important losses at high Mach numbers. The user is responsible of providing appropriate TPR tables according to the design of the intake.
2. To facilitate the data insertion of the cross sections and grid sizes as a function of the axial position, the following tables were defined at the input data interface:

➢ "a_vs_L", "a_vs_L": Non-dimensional cross sections vs. normalized axial position. "Non-dimensional" means that the "y" values of the table are divided by the nominal width and height. "x" values are axial lengths divided by the total length, *and must be between 0 and 1*

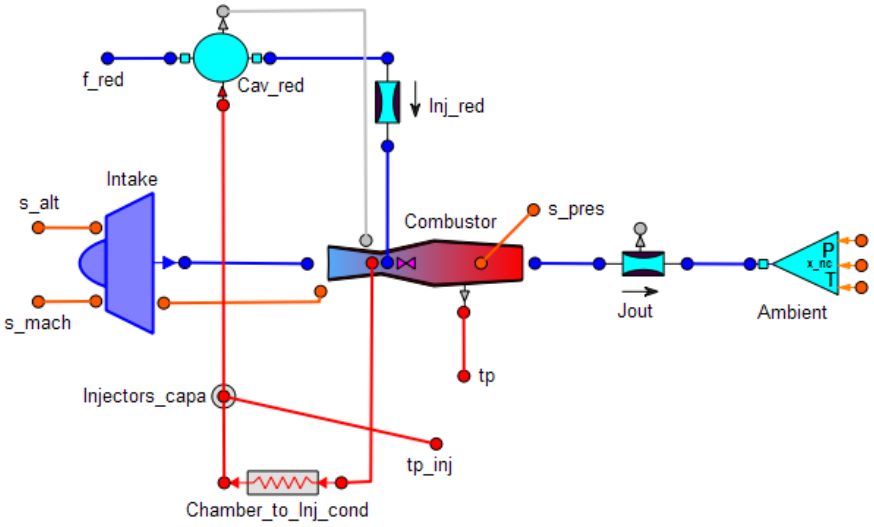
- "dxc_vs_L": Mesh size distribution vs. normalized axial position. "x" values are axial lengths divided by the total length, *and must be between 0 and 1.* "y" values are the **relative** mesh size depending on the axial position (x): The lower y value the lower mesh size. For example, a "y" value of 0.5 at x=0 and a "y" value of 2 at x=1 means that meshes at inlet (x=0) are four times smaller than at the exit. The same effect is obtained if the "y" value is 1 at x=0 and 4 at x=1.

  Tables "a_vs_L", "b_vs_L" and "dx_vs_L" will not be affected by a change in the number of nodes, in the length of the pipe or in the nominal diameter (scalars). Default values are: {{0,1},{1,1}}, so constant diameter and uniform grid size distribution will be used.

3. The mass fractions of the starter gases are input data (starter_y[] variable) within a predefined set of chemicals {H2O, CO, CO2, N2, H2, He} where most of the starter gases produced in a particular starter powder are included. By default, this composition is: {0.26, 0.19, 0.07, 0.21, 0.27, 0} [RD-44]
4. Ignition flag and starter gas mass flow are not input data but boundary conditions automatically loaded in the default experiment. By default, ignition is activated with *no* starter gases (see §8.1.2.3 & §8.2.3.4).
5. Very low values (10^{-6} seconds) of the burning and vaporization times (only for the "Rate" model) result in the equilibrium Combustor component behavior.
6. The vaporization factors f_v[], see §8.1.2.5 and §8.2.4.2, are boundary conditions automatically loaded in the default experiment scaling the nominal droplets size in case of fuel liquid injection. By default, they are set to 1.
7. *The fuel injection is multipoint, so any chamber node can have a particular fuel injection rate. The array of relative fuel injection areas vs. node number,* ***A_rel_red[]****, is a boundary variable automatically loaded in the default experiment, see* §10.8.1

8.3.7.5 Topology

Shown below is the schematic of a *CombustChamber_ramjet* component.



8.3.7.6 Formulation

This topological component gathers all the capabilities of the ramjet combustors, injectors, cavities and intakes. See sections §8.2.7 and §5.4.22.

Signals "s_mach" and "s_alt" give variable flight conditions to be imposed at the inlet of the intake. The static pressure, temperature and velocity will be calculated and applied as initial conditions in the combustor *and as external conditions at outlet (Ambient component)*:

```
INIT
    atm_cond(s_alt.signal[1], s_mach.signal[1], P_o, T_o, v_o)
    m_o = P_o/(RGAS/28.9644*T_o)*v_o*(a*linearInterp1D(a_vs_L,1) *
    b*linearInterp1D(b_vs_L,1))
```

```
CONTINUOUS
    Ambient.s_pres.signal[1] = intake.meas_out.signal[2]
    Ambient.s_temp.signal[1] = intake.meas_out.signal[3]
    Ambient.s_xNonCond.signal[1] = 0
```

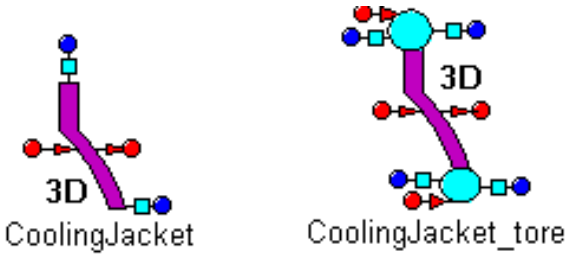
P_o, T_o and v_o are the calculated ambient conditions according to the standard atmosphere and inlet Mach number. *The intake outlet conditions (combustor inlet) are calculated by iterating in the inlet velocity to fulfill a pressure recovery function of the inlet Mach number and a fixed discharge pressure of the combustor, P_o, the ambient pressure, see §5.4.22.*

The thermal components included allow the calculation of thermal fluxes between the chamber and the cavities. The main thermal fluxes at chamber walls are simulated by the "tp" thermal port which can be connected to a thermal network.

8.3.8 CoolingJacket and CoolingJacket_tore

8.3.8.1 Description

These components represent a Regenerative Circuit of a Chamber that does or does not include inlet/outlet tores. A 3D geometry (built by means of several 3D walls around the channels) will be taken into account. The *CoolingJacket_tore* component is an aggregate of a *CoolingJacket* and two fluid cavities (see §5.4.2) representing the torus with two fluid ports each, so that multiple cooling jacket connections or bleedings can be simulated. Use a "*DeadEnd*" component of the FluidFlow_1D library to close any unused fluid ports.



8.3.8.2 Construction Parameters

Name	Type	Description
Nsub	CONST INTEGER	Number subsonic nodes
Nsup	CONST INTEGER	Number of supersonic nodes
nr	CONST INTEGER	Number of radial thermal nodes at internal and external wall thickness
ns	CONST INTEGER	Number of azimuthal thermal nodes between the channels and in the channels. See Figure 8-2

Nsub+Nsup are the number of fluid nodes of the channels. The meaning of "nr" and "ns" is in Figure 8-2.

1. Put Nsub = 0 if the Cooling Jacket is connected to a *Nozzle*
2. Put Nsup = 0 if the Cooling Jacket is connected to a *PreBurner*
3. Put Nsub != 0 and Nsup !=0 if the Cooling Jacket is connected to a *CombustChamberNozzle*

8.3.8.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	IN	Inlet/Outlet fluid port
f2	fluid	burnerGasesOption	IN	Inlet/Outlet fluid port
f1_lat	fluid	burnerGasesOption	IN	Lateral inlet fluid port
f2_lat		burnerGasesOption	IN	Lateral outlet fluid port
tp_ch	thermal	(n = Nsub+Nsup)	IN	Thermal port to chamber
tp_ex		(n = Nsub+Nsup)	OUT	Thermal port to the exterior
tp_tore_in		(n =1)	IN	Thermal port connected to inlet torus
tp_tore_out		(n =1)	IN	Thermal port connected to outlet torus

8.3.8.4 Data

Name	Type	Units	Units
Dt	REAL	Cooling jacket throat diameter	m
Lc	REAL	Length of the convergent part. To normalize subsonic lengths	m
dxc_input	BOOLEAN	Subsonic mesh size distribution option. See note 1	
Dc_vs_L	TABLE 1D	Normalized convergent diameters vs normalized axial position	-
dxc_vs_L		Weighting function for conv. mesh size distribution or normalized node lengths vs node number, depending on "dxc_input" data	-
Ld	REAL	Total length of divergent part. To normalize supersonic lengths	m
Ld1	REAL	Nozzle axial length from the throat to the inlet (0 if Lc != 0)	m
Ld2	REAL	Nozzle axial length from the throat to the outlet	m
dxd_input	BOOLEAN	Supersonic mesh size distribution option. See note 1	
Dd_vs_L	TABLE 1D	Normalized divergent diameters vs normalized axial position	-
dxd_vs_L		Weighting function for div. mesh size distribution or normalized node lengths vs node number, depending on "dxd_input" data	-
AR_sup	REAL	Nozzle Exit/throat area ratio: 0, no Dd_vs_L profile modification	-
n_ch	INTEGER	Number of channels	-
init_option	ENUM	Option to specify the initial thermodynamic state	
P_o	REAL See §5.1.3.3	Initial Pressure	Pa
T_o		Initial temperature	K
rho_o		Initial density	kg/m3
x_nco		Initial non-condensable mass fraction	-
x_o		Initial quality	-
ht_option	ENUM	Wall-fluid heat transfer option (see §A3 with the different options)	
hc_dat	REAL	Heat transfer coefficient if ht_option = Ht_constant	W/m^2*K
k_f	REAL	Multiplier of the friction factor	-
mat_e	ENUM	Chamber external Material	-
cp_e	REAL	External wall Specific heat if mat=None	J/kg/K
K_e	REAL	External wall conductivity if mat=None	W/m/K
rho_e	REAL	External wall density if mat=None	
mat_i	ENUM	Chamber internal Material	-
cp_i	REAL	Internal wall Specific heat if mat=None	J/kg/K
K_i	REAL	Internal wall conductivity if mat=None	W/m/K
rho_i	REAL	Internal wall density if mat=None	
rug	REAL	roughness	m
w_ch	REAL	Channel width at throat section	m
t_ch	REAL	Channel height at throat section	m
th_e	REAL	Jacket outer wall thickness at throat section	m
th_i	REAL	Jacket inner wall thickness at throat section	m
wc_vs_L	TABLE 1D See note 2	Non-dimensional channel widths vs. non-dimensional axial position	
tc_vs_L		Non-dimensional channel heights vs. non-dimensional axial position	
ti_vs_L		Non-dim. inner wall thickness vs non-dimensional axial position	
te_vs_L		Non-dim. outer wall thickness vs non-dimensional axial position	
R1_tore	REAL *(Only for the torus option)*	Radius of the upper torus	m
R1_tore		Radius of the lower torus	m
r1_tore		Radius of the circular section of the upper torus	m
r2_tore		Radius of the circular section of the lower torus	m

Notes:

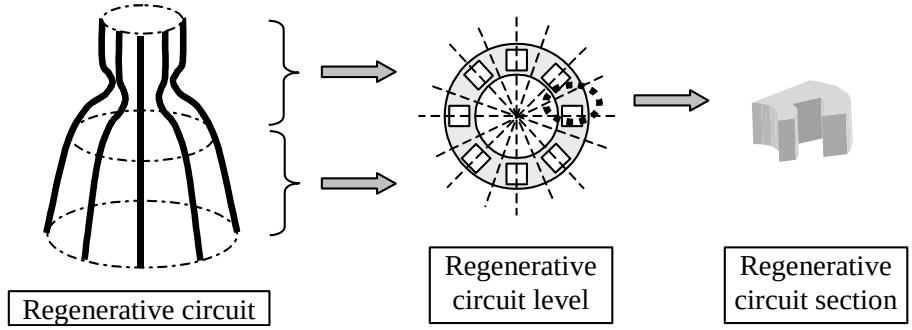
1. *Dc_vs_L, dxc_vs_L* and *Dd_vs_L, dxd_vs_L* tables allow variable geometry, variable mesh size in the convergent/divergent parts. If dx.._input = FALSE, "dx.._vs_L" is a table with the relative mesh size vs. axial position. If dx.._input = TRUE, "dx.._vs_L" is a table with the normalized node lengths vs node number. These tables will be used to discretize (Nsub+Nsup) and to interpolate the channel profile from L=0 to L=Lc in the convergent zone and from L=Ld1 to L=Ld2 in the divergent zone. Dt is used for the normalization of diameters and Lc, Ld to the normalization of the

convergent/divergent axial lengths respectively. *See §8.1.2 for more details.* Ld1 must be zero if a convergent part (Nsub != 0, Lc != 0) is included.

2. Non dimensional channel widths, heights and thickness tables must be entered as a function of the non-dimensional length from *the cooling jacket inlet to the exit*, i.e. the x vector of these tables is normalized with "Lc+Ld", and must *start with x=0 and finish with* x=1. "Y" values are normalized with the corresponding "w_ch, t_ch, th_e, th_e" values. Default tables {{0,1},{1,1}} means constant geometry along x. Note that nozzle profile and mesh tables (*Dc_vs_L,dxc_vs_L, …*) have a different x origin (the injection plane) to be compatible with the chamber geometry input data.
3. Wall materials are selected via the ENUMERATIVE variable "mat" which `provides a list of materials (see §3.9).

8.3.8.5 Topology

It is constructed by aggregation of one "Tube" (FLUID_FLOW_1D library) representing the channels and five 3D walls around them.



The cooling jacket is divided into a variable (*Nsub+Nsup*) number of sections in axial direction. Every section is made of:

- one fluid node of the "Tube" component (FLUID_FLOW_1D library), which simulates the cooling channels and,
- five slides of the "wall_3D" components, which simulates the metallic walls. They are arranged according to the following sketch:

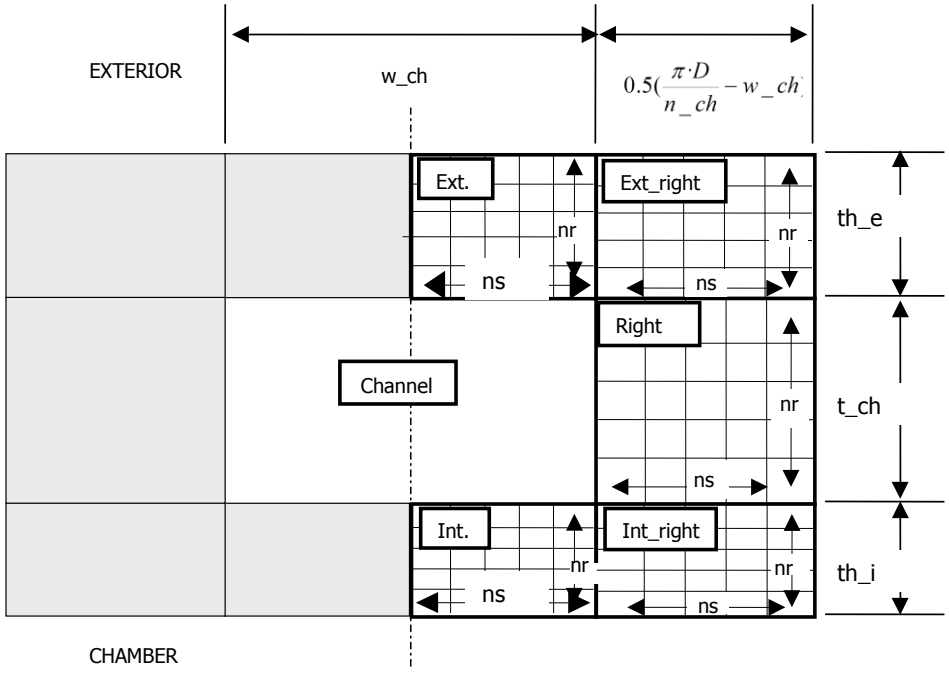


Figure 8-2 Cooling Jacket wall mesh

The *CoolingJacket_tore* component has two cavities simulating the torus. They are linked to the channels with two junctions.

8.3.8.6 Formulation

Channels

Channels are simulated by only one *Tube* component making its "*num*" data (see §5.4.15.4) equal to the number of channels. The mesh size of each node is a function of the axial position through the non-dimensional geometry tables. See §8.1.2.2.

The rectangular channel geometry (widths, heights, wet areas) is similarly calculated but using the interpolated widths and heights values as follows: $A_{wet,i} = 2(a_i + b_i) \cdot l_{ch,i}$, where

$$a_i = w\_ch \cdot Interp(x_i/(Lc + Ld),\ wc\_vs\_L)$$
$$b_i = t\_ch \cdot Interp(x_i/(Lc + Ld),\ tc\_vs\_L)$$

The wall thickness will be equally interpolated using the "te_vs_L", "ti_vs_L" tables

Heat conduction

The "wall_3D" components used to simulate the walls will calculate the heat conduction in every direction including the axial one. This thermal component has thermal ports in radial and in azimuth directions allowing the exact calculation of heat conduction through the channel corners. The formulation is as in §6.3.1.5 but for a rectangular domain.

Left/Right boundary conditions

For reasons of symmetry, the left and the right sides are adiabatic:

$$\text{wall}_{\text{right}}(i).\text{tpx\_out.q}[j] = 0, \qquad j = 1, nr$$
$$\text{wall}_{\text{int}}(i).\text{tpx\_in.q}[j] = 0, \qquad \text{wall}_{\text{int\_right}}(i).\text{tpx\_out.q}[j] = 0,\ j = 1, nr$$
$$\text{wall}_{\text{ext}}(i).\text{tpx\_in.q}[j] = 0, \qquad \text{wall}_{\text{ext\_right}}(i).\text{tpx\_out.q}[j] = 0,\ j = 1, nr$$
$$i = 1, nl \qquad (\text{number of fluid nodes})$$

where:

wall$_{xxx}$: Name of one of the walls around the channel (see figure above)
tpx_in : Name of thermal port of any "wall" component along x direction (inlet side)
tpx_out: Name of thermal port of any "wall" component along x direction (outlet side)

External side (Ambient)

The heat flux in this section (port "*tp_ex*") is calculated as the sum of all wall nodes.

$$\text{tp\_ex.q}[i] = 2\ \text{n\_ch}\left(\sum_{ns} \text{wall}_{\text{ext}}(i).\text{tpy\_out.q}[j] + \sum_{ns} \text{wall}_{\text{ext\_right}}(i).\text{tpy\_out.q}[j]\right)$$

The temperatures of the external walls in contact with the exterior are assumed to be the same:

$$\text{tp\_ex.T}[i] = \text{wall}_{\text{ext}}(i).\text{tpy\_out.T}[j] \qquad j = 1,\ldots, ns$$
$$\text{tp\_ex.T}[i] = \text{wall}_{\text{ext\_righ}}(i).\text{tpy\_out.T}[j] \qquad j = 1,\ldots, ns$$

where,

tpy_in : Name of thermal port of any "wall" component along y direction (inlet side)
tpy_out: Name of thermal port of any "wall" component along y direction (outlet side)

Internal side (Combustion chamber)

Same equations as before applied to the "tp_ch" chamber thermal port

$$tp\_ch.q[i] = 2\ n\_ch(\sum_{ns} wall_{int}(i).tpy\_in.q[j] + \sum_{ns} wall_{int\_right}(i).tpy\_in.q[j])$$
$$tp\_ch.T[i] = wall_{int}(i).tpy\_in.T[j] \qquad j = 1,...,ns$$
$$tp\_ch.T[i] = wall_{int\_right}(i).tpy\_in.T[j] \qquad j = 1,...,ns$$

Channel / Wall interfaces

The wet surfaces of the walls ("*wall_int*", "*wall_right*" and "*wall_ext*") in contact with the channel refrigerant, are assumed to be at three different temperatures (one for each side):

$$Channel.tp\_in.T[i] = wall_{int}(i).tpy\_out.T[j] \qquad j = 1,...,ns$$
$$Channel.tp\_lat.T[i] = wall_{right}(i).tpx\_in.T[j] \qquad j = 1,...,nr$$
$$Channel.tp\_out.T[i] = wall_{ext}(i).tpy\_in.T[j] \qquad j = 1,...,ns$$

Channel.tp_in and Channel.tp_out are the names of the three thermal ports of the channel component. The corresponding heat flux between the channel and the walls around it is calculated as follows:

$$Channel.tp\_in.q[i] = 2\sum_{ns} wall_{int}(i).tpy\_out.q[j]$$
$$Channel.tp\_lat.q[i] = -2\sum_{nr} wall_{right}(i).tpx\_in.q[j]$$
$$Channel.tp\_out.q[i] = 2\sum_{ns} wall_{ext}(i).tpy\_in.q[j]$$

The channel heat fluxes (*Channel.tp_in.q(i*), *Channel.tp_lat.q(i) and Channel.tp_out.q(i)*) are also calculated by the "Tube" component using as input the wall/fluid temperatures and two-phase hydraulic correlations for the film coefficient evaluation. Then, a set of implicit nonlinear equations is formed which is solved by EcosimPro. This also applies for the combustion chamber side, where the heat fluxes (*tp_ch.q[]*) are calculated by the "*Combustor*" component connected to the "*RegCircuit*" component. On the external side (ambient), the heat flow is defined according to the thermal component connected to this port".

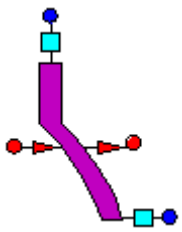
Torus

The inlet/outlet torus collecting the channel flows is modeled by two fluid cavities (see §5.4.2) with three fluid ports (one used to connect the channel to the cavities).

8.3.9 CoolingJacket_simple

8.3.9.1 Description

This component represents a Regenerative Circuit of a Chamber. A *simplified* 3D geometry (built by means of several 3D bars around the channels) will be taken into account.



8.3.9.2 Construction parameters

Name	Type	Description
Nsub	CONST INTEGER	Number subsonic nodes
Nsup	CONST INTEGER	Number of supersonic nodes

Note: Nsub+Nsup determine the number of fluid nodes of the channels.

1. Put Nsub = 0 if the Cooling Jacket is connected to a *Nozzle*
2. Put Nsup = 0 if the Cooling Jacket is connected to a *PreBurner*
3. Put Nsub != 0 and Nsup !=0 if the Cooling Jacket is connected to a *CombustChamberNozzle*

8.3.9.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid	burnerGasesOption	IN	Inlet/Outlet fluid port
f2		burnerGasesOption	IN	Inlet/Outlet fluid port
tp_ch	thermal	(n = Nsub+Nsup)	IN	Thermal port to chamber
tp_ex		(n = Nsub+Nsup)	OUT	Thermal port to the exterior

8.3.9.4 Data

Name	Type	Units	Units
Dt	REAL	Cooling jacket throat diameter	m
Lc	REAL	Length of the convergent part. To normalize subsonic lengths	m
dxc_input	BOOLEAN	Subsonic mesh size distribution option. See note 1	
Dc_vs_L	TABLE 1D	Normalized convergent diameters vs normalized axial position	-
dxc_vs_L		Weighting function for conv. mesh size distribution or normalized node lengths vs node number, depending on "dxc_input" data	-
Ld	REAL	Total length of divergent part. To normalize supersonic lengths	m
Ld1	REAL	Nozzle axial length from the throat to the inlet (0 if Lc != 0)	m
Ld2	REAL	Nozzle axial length from the throat to the outlet	m
dxd_input	BOOLEAN	Supersonic mesh size distribution option. See note 1	
Dd_vs_L	TABLE 1D	Normalized divergent diameters vs normalized axial position	-
dxd_vs_L		Weighting function for div. mesh size distribution or normalized node lengths vs node number, depending on "dxd_input" data	-
AR_sup	REAL	Nozzle Exit/throat area ratio: 0, no Dd_vs_L profile modification	-
n_ch	INTEGER	Number of channels	-
init_option	ENUM	Option to specify the initial thermodynamic state	
P_o	REAL See §5.1.3.3	Initial Pressure	Pa
T_o		Initial temperature	K
rho_o		Initial density	kg/m3
x_nco		Initial non-condensable mass fraction	-
x_o		Initial quality	-
ht_option	ENUM	Wall-fluid heat transfer option (see §A3 with the different options)	
hc_dat	REAL	Heat transfer coefficient if ht_option = Ht_constant	W/m^2*K
k_f	REAL	Multiplier of the friction factor	-
mat_e	ENUM	Chamber external Material	-
cp_e	REAL	External wall Specific heat if mat=None	J/kg/K
K_e	REAL	External wall conductivity if mat=None	W/m/K
rho_e	REAL	External wall density if mat=None	
mat_i	ENUM	Chamber internal Material	-
cp_i	REAL	Internal wall Specific heat if mat=None	J/kg/K
K_i	REAL	Internal wall conductivity if mat=None	W/m/K
rho_i	REAL	Internal wall density if mat=None	
rug	REAL	roughness	m
w_ch	REAL	Channel width at throat section	m
t_ch	REAL	Channel height at throat section	m
th_e	REAL	Jacket outer wall thickness at throat section	m
th_i	REAL	Jacket inner wall thickness at throat section	m
wc_vs_L	TABLE 1D See note 2	Non-dimensional channel widths vs. non-dimensional axial position	
tc_vs_L		Non-dimensional channel heights vs. non-dimensional axial position	

Name	Type	Units	Units
te_vs_L		Non-dimensional outer and inner wall thickness vs. non-dimensional axial position	
ti_vs_L			

Notes:

1. *Dc_vs_L, dxc_vs_L* and *Dd_vs_L, dxd_vs_L* tables allow variable geometry, variable mesh size in the convergent/divergent parts. If dx.._input = FALSE, "dx.._vs_L" is a table with the relative mesh size vs. axial position. If dx.._input = TRUE, "dx.._vs_L" is a table with the normalized node lengths vs node number. These tables will be used to discretize (Nsub+Nsup) and to interpolate the channel profile from L=0 to L=Lc in the convergent zone and from L=Ld1 to L=Ld2 in the divergent zone. Dt is used for the normalization of diameters and Lc, Ld to the normalization of the convergent/divergent axial lengths respectively. *See §8.1.2 for more details.* Ld1 must be zero if a convergent part (Nsub != 0, Lc != 0) is included.
2. Non dimensional channel widths, heights and thickness tables must be entered as a function of the non-dimensional length from *the cooling jacket inlet to the exit*, i.e. the x vector of these tables is normalized with "Lc+Ld", and must *start with x=0 and finish with* x=1. "Y" values are normalized with the corresponding "w_ch,t_ch, th_e, th_e" values. Default tables {{0,1},{1,1}} means constant geometry along x. Note that nozzle profile and mesh tables (*Dc_vs_L,dxc_vs_L, …*) have a different x origin (the injection plan) to be compatible with the chamber geometry input data.
3. Wall materials are selected via the ENUMERATIVE variable "mat" in which a list of materials is provided (see §3.9).

8.3.9.5 Topology

It is constructed by aggregation of one "Tube" (FLUID_FLOW_1D library) representing the channels and three 1D bars around them:

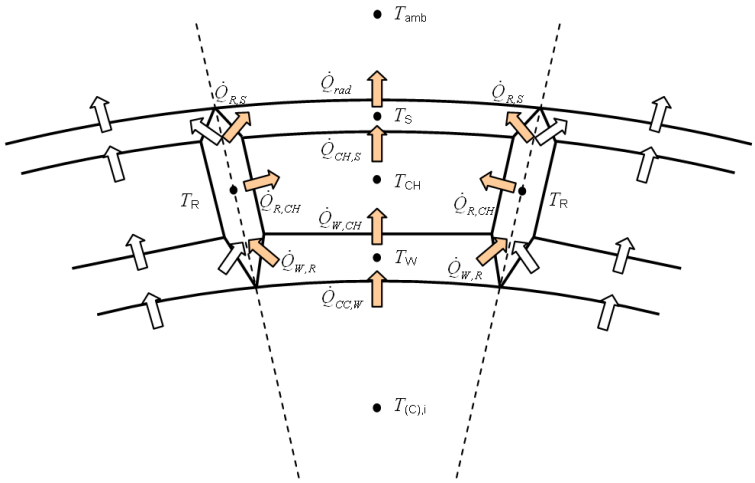


Figure 8-3 Simplified Cooling Jacket wall disposition

The philosophy is the same as in the "CoolingJacket" component with the difference that here the five 3D-walls are replaced by three 1D-bars with only one thermal node per section. So, the cooling jacket is divided into a variable (*Nsub+Nsup*) number of sections in axial direction. Each section is made up of:

- one of the fluid nodes of the "Tube" component (FLUID_FLOW_1D library), which is simulating the cooling channels and,
- three slides of the "bar_1D" components, which are simulating the metallic walls. They are arranged according to Figure 8-3.

8.3.9.6 Formulation

The channels are simulated by only one *Tube* component making its "*num*" data (see §5.4.15.4) equal to the number of channels.

The channel geometry (wet areas, thickness and the mesh axial length for each fluid node) is built in the same way as in the RegCircuit component.

The equations used to interconnect the three bars are taken from RD-25. Basically, heat fluxes at wall interfaces are calculated assuming linear temperature distribution between the center and the boundaries in the different walls and applying the conduction equation for each wall interface in every direction including the axial direction: $Q = 2A_{bound}\lambda(T_{wall}-T_{bound})/th$.

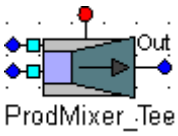
At channel/wall interfaces, the conduction heat flux is forced to be equal to that calculated in the "Tube" component (see §5.3.8.2, term ΔQ), so that the interface temperature (T_{bound}) can be calculated.

Finally, the heat storage dynamic equations will calculate the wall temperatures at each level (axial direction, in correspondence with the channel fluid nodes) and for each one of the bars (internal, lateral and external, see Figure 8-3).

8.3.10 ProdMixer_Tee

8.3.10.1 Description

This component simulates a mixture of two pure fluids or a pure fluid with combusted gases. The fluid at the outlet of this component is always a mixture of Chemicals. To simulate a mixture of two combusted gases, a standard Tee component has to be used instead of this one. The *ProdMixerPipe* does not longer exists because a simple Pipe replace it.



A combustion chamber with more than two injectors can then be modeled by placing this component upstream of one of the chamber injectors and collecting two flows, one feeding combusted gases coming from a pre-burner and the other feeding a pure fluid coming from a pump.

This component does not allow reverse flow.

8.3.10.2 Construction Parameters

Name	Type	Description
burnerGasesOptionP1	SET_OF(Chemicals)	Type of mixture of fluids
burnerGasesOptionP2	SET_OF(Chemicals)	Type of mixture of fluids

Select *burnerGasesOption = Chemicals* only for ports placed downstream of a combustor. Keep *burnerGasesOption = noBurnGases* (default value) in other cases.

8.3.10.3 Ports

Name	Type	Parameters	Direction	Description
f1	Fluid	burnerGasesOption	IN	Inlet/Outlet fluid ports
f2	Fluid	burnerGasesOption	IN	Inlet/Outlet fluid ports
fout	Fluid	Chemicals	OUT	Outlet fluid port for the ProdMixer_Tee
tp_in	THERMAL	n=1	IN	Thermal port connected to the wall

The *ProdMixer_Tee* is topological composed by a PodMixer_Vol and a Junction. Ports f[1] and f[2] must be inlets. Port f[3] (not visible) is the volume outlet being connected to the inlet of outlet Junction whose outlet port is *fout*.

8.3.10.4 Data

Name	Type	Description	Units

Name	Type	Description	Units
Vo	REAL	Initial fluid volume	m^3
L	REAL	Volume length - if zero, the volume is assumed to be a sphere	m
Pw	REAL	Wet perimeter. (0, circular cross area)	m
Re_lam	REAL	Laminar Reynolds number	-
T_o	REAL	Initial temperature	K
P_o	REAL	Initial Pressure	Pa
mat	ENUM	Wall material	
cp_w	REAL	Wall Specific heat - only if mat=None (J/kg*K)	J/kg*K
rho_w	REAL	Wall density - only if mat=None (kg/m^3)	kg/m^3
th	REAL	Wall thickness (m)	m
fld_add	REAL	Pressure drop coefficient for losses different than friction (-)	-
rug	REAL	Rugosity (m)	m
Aout	REAL	Outlet junction area	m^2
zeta_out	REAL	Loss coefficient of outlet junction	-
z_bottom	REAL	Elevation at the bottom relative to a z fixed axis	m

The fluid volume (Vo) is the only geometric data which is always needed. If the other two geometric data are left as zero, the volume geometry is considered to be a sphere.

L	Pw	Geometry
0	0	Sphere, with radius calculated from the total volume
>0	0	Circular cylinder, with radius calculated from the cross area
>0	>0	Non Circular cylinder
0	>0	Circular cylinder, with radius calculated from the wet perimeter

8.3.10.5 Formulation

This component uses a similar formulation as in a fluid volume, but accounts for the enthalpy shifts between CEA chemicals and pure fluids. Note that the chemical composition at the inlet ports is known, and depends on the fluid type (pure fluid or combusted products). In case of pure fluids, the corresponding enthalpy shift is added to the inlet enthalpy in the energy equation so that the same CEA reference is used. Then, once the mixture equations in mass and enthalpy are applied (see §5.3.3.1), the calculated mass fractions are assigned to the outlet port, so the gas composition can be known in any component placed downstream of it.

The complete thermodynamic state (pressures, temperature and transport properties) is calculated using the `ProdMixer_prop` and `Trans_mix` functions according to the current chemical compositions. Two possible cases are considered:

- Van der Waals mixture

  ```
  VDWMix_state_vs_ru(mix,x,rho,u,P,T,drho_dp,drho_dh,vsound,Rg,gamma,Cp)
  ```

- Perfect gas mixture thermodynamic functions (see §4.5.1):

  ```
  Tmix_vs_u(mix, y, u, T)
  CpMW_mix (mix, y, T, MW, Cp)
  Rg = 8314.4/MW
  gamma = Cp / (Cp - Rg)
  P = rho * T * Rg
  vsound = sqrt(gamma*Rg*T)
  ```

where `Tmix_vs_u`, `CpMW_mix` are thermodynamic functions, and "mix" refers to the available chemicals. The inputs of the equations are x[i] (index i is extended to chemicals), rho, and u, all of them dynamic variables as in the Volume component.

Friction pressure drop is computed using the same formulation as for Pipes by means of the mean speed in the volume mixture, the equivalent hydraulic diameter and the evaluation of the friction coefficient. Then, the pressure losses are assigned to the fluid ports.

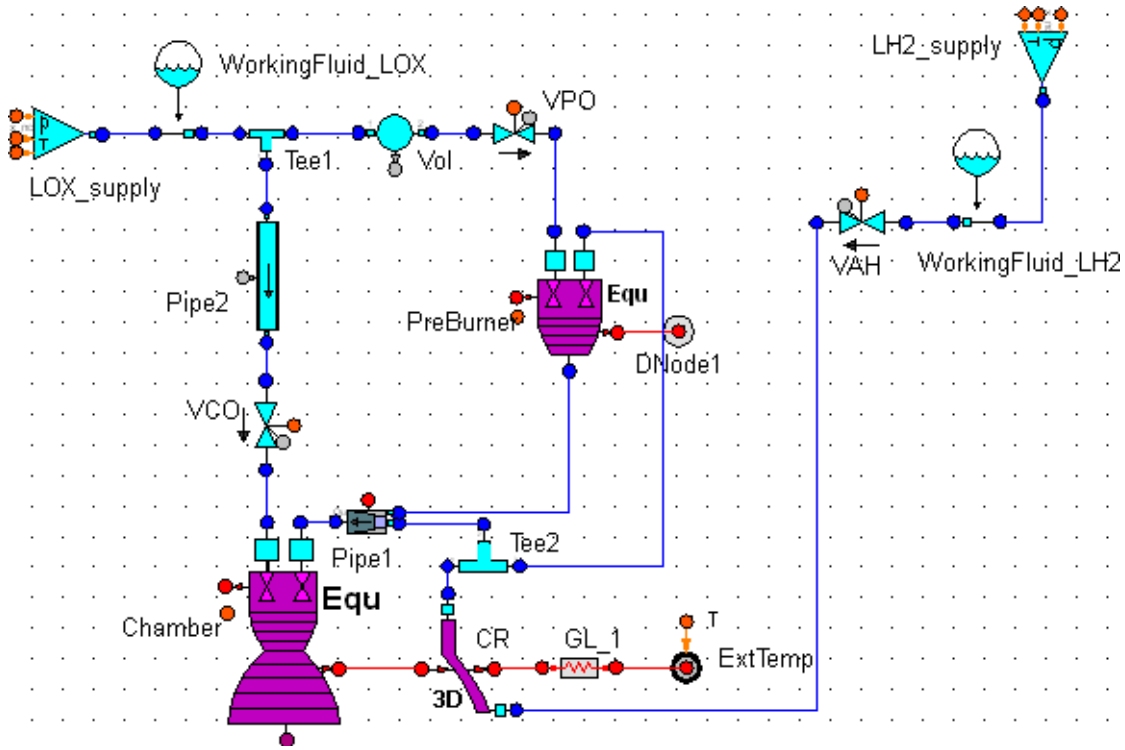
The following equation calculates the thermal wall temperature:

$$q_{in} = h_{film} A_{wall} (T_{wall} - T)$$
$$T_{wall}^{'} = (q_{port} - q_{in}) / (cp_{wall} \cdot m_{wall})$$

where qin is the heat flux given by the connected thermal port. The film coefficient is calculated according to A3.2.

8.3.10.6 Application example

The following model ("Test_PreBurner" of the ROCKETS_ESAMPLES library) represents a case using a ProdMixer_tee component to simulate a chamber with tree injectors:

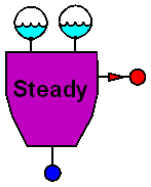


Simulation results can be visualized running this model ("exp1" experiment) showing the delay of the Preburner products entering into the main chamber and the effects of the pure fluid mixed with the Preburner products.

8.3.11 ChemInflator

8.3.11.1 Description

This component represents a steady model of a combustor as in CEA code with a fluid outlet port giving outlet gas properties vs MR, solid propellant gases and chamber pressure.



8.3.11.2 Construction parameters

Name	Type	Description
Nsub	CONST INTEGER	Number of nodes along the chamber axial direction

Note: Nsub determines the number of fluid and wall thermal nodes

8.3.11.3 Ports

Name	Type	Parameters	Direction	Description
f	fluid	Chemicals	OUT	Outlet combustion gases port
tp	thermal	(n = Nsub)	OUT	Thermal port

The outlet port can only be connected to a *FLUID_FLOW type* component (fluid type port), such as for example a Pipe of a Nozzle component.

8.3.11.4 Data

Name	Type	Description	Units
Dt	REAL	Chamber throat diameter	m
Rcurv	REAL	Curvature radius at throat	m
Lc	REAL	Chamber length of subsonic part	m
dx_input	BOOLEAN	FALSE, dx_vs_L = weighting function for mesh size distribution. TRUE, normalized node lengths vs node number	-
Dc_vs_L	TABLE 1D	Normalized subsonic chamber diameters vs normalized axial position	-
dx_vs_L		Weighting function for mesh size distribution or normalized node lengths vs node number	-
P	REAL	Imposed chamber pressure	Pa
fluid_oxy	ENUM	Oxidizer fluid name	-
fluid_red	ENUM	Reducer fluid name	-
P_oxy		Pressure of the oxidizer fluid	Pa
P_red	REAL	Pressure of the reducer fluid	Pa
T_oxy		Temperature of the oxidizer fluid	K
T_red	REAL	Temperature of the reducer fluid	K
emiss	REAL	Emissivity of the combustion gases	-
x	REAL	X coordinate relative to a body axis system	m
y	REAL	Y coordinate relative to a body axis system	m
z	REAL	Elevation relative to a body axis system	m
eta	REAL	Combustor efficiency (only for equilibrium components)	-
starter_y[]	REAL	Molar fractions (H2O,CO,CO2,N2,H2,He) of the starter gases	-
frozen_th	BOOLEAN	Flag forcing frozen conditions in the throat	-

Notes:

1. *Dc_vs_L,dx_vs_L* tables allow variable geometry, variable mesh size in the chamber. The discretized (Nsub) wet areas and diameters will be interpolated using these tables from L=0 to L=Lc. Dt is used for the normalization of diameters and Lc to that of the axial lengths**.** *See §8.1.2.2 for more details*
2. The mixture ratio is not included in the list of data because it will be a boundary that must be defined in the experiment file as a function of time.

8.3.11.5 Formulation

The properties of combusted gases and temperature distribution along the Nsub subsonic sections are calculated using the Minimum Gibbs energy method at imposed pressure and MR. Injection conditions are calculated directly from the input data.

The different static conditions and heat transfers along the chamber wall are calculated by iteration as in the Nozzle component (see *§8.2.8.3) but taking the subsonic solution.*

9. COUPLING TO EXTERNAL SOFTWARE

9.1 OVERVIEW

This last chapter demonstrates EcosimPro's capabilities when coupled to external software. Two applications are selected:

- Zooming: EcosimPro can be coupled to 3D CFD software acting as a simple EcosimPro component. The goal of such a coupling is to give the user the opportunity to simulate components that cannot be summarized by a 1D or 0D component. Moreover, it makes it possible to simulate any geometry provided it is connected to the rest of the EcosimPro circuit by ports.
- System optimization: EcosimPro can also be coupled to an external optimizer whose purpose is to find the best combination of parameters leading to the objectives imposed by the user. Using such an interface, every EcosimPro system can be optimized. As an example of application, given an objective of maximizing the thrust of a GG cycle while respecting a pressure constraint in the combustion chamber, the optimizer will provide the optimal valve or throat areas, or any parameter specified by the user.

The rest of this chapter provides examples of the coupling interface with external software programs: Fluent 6.3.26 for the zooming application, and Max for the optimization application. It should be stressed that these examples are offered as proofs of concept only and thus do not constrain the user to follow the given approach.

9.1 COUPLING TO AN EXTERNAL CFD CODE

9.1.1 Introduction

In some cases, a 1D view of a component can be too simplistic to model complex 3D reality. Therefore, it was thought to use external 3D/2D CFD software to simulate this problematic component. This software should be treated as a simple EcosimPro component, having the same ports as the 1D element.

The 3D CFD software chosen is Fluent (version 6.3.26 is used here). Fluent is a commercial CFD software program of proven efficiency and accuracy in many problems. Moreover, it offers the possibility to resolve a wide range of flows including multiphase/species, reacting flows, moving geometries and so on, using a variety of physical/numerical models. Another advantage of Fluent is that you can access and modify solver variables during a computation using "User Defined Functions". Fluent can therefore be driven by another software program such as EcosimPro.

This section is intended to summarize the features added to both software programs and to show how they communicate with each other.

The following section presents the general process of the coupling method. It begins with the sequence of operations to obtain a "coupled solution". Then, a description is given of the general interface including the EcosimPro files needed to encapsulate the external component, the user-defined functions used in Fluent and the scheme file created to setup the Fluent model. After that, a more in-depth description of Fluent's modeling capabilities and limitations is presented in order to justify the assumptions made. The section concludes by pointing out known limitations of the interface and areas for possible improvement.

9.1.2 Coupling Process

9.1.2.1 General interface

The sequence of operations linking EcosimPro to Fluent is fairly straightforward and is shown in Figure 9-1. The additional features needed to ensure this coupling and their roles are diagrammed in Figure 9-2.

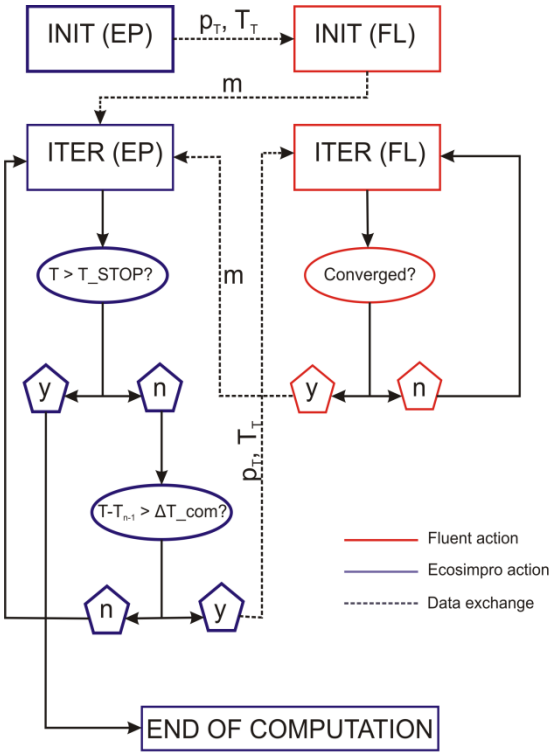


Figure 9-1: Schematic view of the coupling process between EcosimPro and an external CFD software program.

The sequence of operations of the coupled process is articulated as follows. After EcosimPro initializes, a journal file is created containing every relevant variable that EcosimPro has to transmit to Fluent. This includes initial conditions, the path of the computational grid, solver parameters, etc.

Then, Fluent is launched by EcosimPro using a system call implemented in the external C++ library, and directly starts to read the created journal file. This journal file is case-specific and is overwritten at each EcosimPro experiment. Another journal file ("setup.scm") contains all the operations to be performed by Fluent during the simulation. Some of these operations require data written in the first file (such as the time step, the duration of the experiment, etc.). This second journal file is thus parameterized by the first one and contains four main parts:

1. Model setup:

   This part includes the physical modeling (choice of turbulence model, for instance) and the numerical modeling (discretization accuracy, etc.).

2. Boundary conditions:

   This part hooks the "C library" (UDF) to Fluent. In practical terms, it notifies Fluent that the boundary info has to be fetched by the external library.

3. Fluid properties:

   Like the boundary conditions definition, the scheme file notifies Fluent that the data regarding fluid material properties is given by a User-Defined Function which uses the EcosimPro tables to return the property at the given conditions.

4. Initializing the computation and integration:

   Here, the flow field is initialized using the EcosimPro initial values. Afterwards, time integration begins using the communication interval and time steps set by EcosimPro.

This scheme file dictates how Fluent behaves by translating EcosimPro user inputs into Fluent commands.

During the integration, the only exchange between Fluent and EcosimPro is made by simple text files containing boundary data. At each communication interval, Fluent waits for the text files containing the boundary condition parameters to be created. The EcosimPro integration lasts for an interval defined by the user: Δt_{COM}. After this interval is passed, any data needed by Fluent are written, and Fluent integration can begin. Just as Fluent was before, EcosimPro is idle and waits for the boundary data to be written by Fluent. This is done after Fluent reaches convergence.

Such a process is termed "a loosely coupled" interaction, in contrast to the "strongly coupled" case where there are several information exchanges at each communication interval.

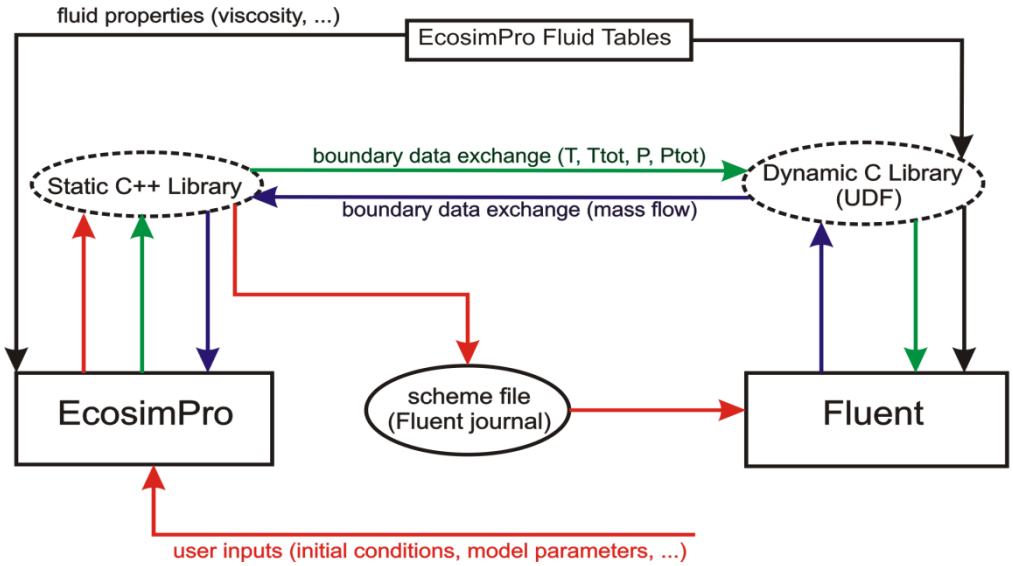


Figure 9-2: Schematic view of the data exchange between EcosimPro, external CFD software and the additional libraries.

It is important to note that this process is the same for steady or unsteady cases. In either case, Fluent uses its unsteady solver, and integrates the solution in time according to the time-step transmitted by EcosimPro. Indeed, in the steady case, this interval should be long enough to allow EcosimPro to converge after receiving a Fluent input and to minimize the number of communication steps. In the unsteady case, this interval should ideally be as short as the EcosimPro integration interval or at least short enough to guarantee a rather smooth evolution of the quantities. It is all a matter of trade-off between time accuracy and computational time.

9.1.2.2 EcosimPro

EcosimPro usually provides two ways of defining a component. It can be resistive (for a given pressure it returns the mass flow) or capacitive (the opposite). The information exchange is made by "ports". In this particular case, the external component (the Fluent 2D/3D case) is a two-port resistive component. The reasons why a resistive component was chosen are explained in the next section.

The Fluent component behaves as an EcosimPro junction, except that total temperature and pressure have to be computed and transmitted to Fluent. For that, ideal gas relations are used.

For this purpose, an "EL" file had to be created: ExternalComponent.el. The basic algorithm is shown below.

```
DISCRETE
  WHEN (Communication==TRUE) THEN
    DT = TIME - TIME_old
    N_TIME = N_TIME+1
    IF (DT < 1e-10) THEN
      N_TIME = 0
      launch_external_model(dim,axi,order,casefile,...)
```

```
END IF
    set_external_model_vars
    min =   (1-alpha)*min  + alpha * get_external_model_var("min")
mout =  (1-alpha)*mout  + alpha * get_external_model_var("mout")
Communication = FALSE
    Communication = TRUE AFTER dt_com
    TIME_old = TIME
  END WHEN
```

In the DISCRETE section, all operations regarding the communication between EcosimPro and the external model are implemented.

First, the external model is launched and initialized by an external "C++" function launch_external_model whose parameters correspond to global parameters transmitted to the external model. These parameters are summarized in Table 9-1. This function is implemented in a static library (see Figure 9-2), invoked by the keyword "C++".

Variable	Type	Meaning
TABLE_PATH	STRING	folder containing EcosimPro tables
RUN_PATH	STRING	folder where the data will be read and written
casefile	STRING	Case file containing the grid of the external component (.cas)
init	STRING	initial solution (.dat), optional, leave blank if none
dim	INTEGER	dimension of the external model case (2=2D ; 3=3D)
axi	INTEGER	1 if axisymmetric 2D problem -- 0 if else
order	INTEGER	discretization accuracy: 1 = 1st order ; 2 = 2nd order
n_steps	INTEGER	number of Fluent integration steps per Δt_COM
Δt_COM	REAL	communication time interval between the EXTERNAL model and
alpha	REAL	under-Relaxation coefficient (1: no under-relaxation ; 0: strong
A_in	REAL	inlet section (m^2)
A_out	REAL	outlet section (m^2)

Table 9-1: Summary of the parameters used for the coupling with Fluent

After the initialization, the information exchange is guaranteed by other functions from this library. ”set_external_model_var” and “get_external_model_var” are respectively used to transmit total pressure and temperature to Fluent and to retrieve the mass flow. Some under-relaxation can be used in order to speed up the convergence of the coupled system for steady case. It is even advised to set alpha = 0.5 to avoid stability concerns that can occur at sharp transients such as the initialization of the flow.

```
CONTINUOUS
  f1.MR = f2.MR
  f1.q_comb = f2.q_comb
  f1.fluid = fluid
  f1.fluid = f2.fluid
f1.n_fluid = f2.n_fluid
f1.m = min
  f2.m = -mout
f1.mh = f1.m * f1.h
  f2.mh = f2.m * f2.h
f1.Q = f1.m/ f1.rho
f2.Q = f2.m/ f2.rho
f2.m_nc = 0
f1.m_nc = 0
  Ti = f1.P/(f1.rho*R)
To = f2.P/(f2.rho*R)
  M_2i = f1.v*f1.v/(gamma*R*Ti)
M_2o = f2.v*f2.v/(gamma*R*To)
Tti = Ti*(1+.5*(gamma-1)*(M_2i))
  pti = f1.P*(Tti/Ti)**3.5
```

```
Tto = To*(1+.5*(gamma-1)*(M_2o))
  pto = f2.P*(Tto/To)**3.5
```

In the CONTINUOUS part, the port information is updated at each EcosimPro integration step. According to its resistive behavior, the mass flow computed by the external model is transmitted to the neighboring components. Other variables, such as enthalpy, pressure and temperature, are extrapolated from the neighboring component ports.

During integration, several outputs are generated:

- Two folders, "EtoF" and "FtoE", are created and contain the data exchanged between Fluent and EcosimPro. These folders are cleaned before every new experiment.
- After each communication step, the complete field of the external component is saved under the name *fluent_comp* followed by the integration time. These files can be further analyzed in Fluent.
- A log file, containing all the messages from Fluent is also generated under the name *outputfile.trn*. It is used to check the convergence and that the simulation ended properly.

9.1.2.3 Fluent

9.1.2.3.1 Introduction

In general, a CFD simulation passes through several steps: CAD and grid generation, pre-processing, simulation and post-processing. The Fluent package including Gambit is able to perform all these steps.

Gambit is responsible for (simple) CAD generation and meshing. Some considerations on that point are given below. The pre-processing (i.e. selecting boundary conditions, solver and physical options), simulation itself and post-processing are done using Fluent. Since the parameters of EcosimPro must be transmitted to Fluent, the pre-processing process must be automated. This involves several assumptions in order to use it in a general framework. Choice of boundary conditions, numerical and physical options are discussed below. A detailed explanation about all these options can be found in the Fluent UG [RD-26].

9.1.2.3.2 Grid generation

As seen earlier, any two-port Fluent component can be easily integrated into EcosimPro. Beforehand, however, the geometry and grids have to be created. This can be done using Gambit, the ANSYS (Fluent) grid generator for creating simple geometries and meshing them. Keep in mind that grids are an essential element of a CFD simulation, and that no single grid is suitable for all flow conditions.

Also notice that the geometry is fixed once and for all. Therefore, the length of a pipe for example cannot be changed by setting a simple parameter in EcosimPro, but rather, it involves recreating the whole geometry and the mesh. Nevertheless, "scale" functionality is available in Fluent for stretching the geometry and the mesh in one or more directions. This feature is not included in the present interface, since stretching the mesh can create a strong anisotropy spoiling the accuracy and causing some stability concerns.

The mesh should therefore be designed very carefully according to the flow conditions. An initial idea of the range of Reynolds and Mach numbers is needed to construct the grid.

9.1.2.3.3 Boundary conditions

Fluent offers several kinds of boundary conditions. Among them, only few are suitable for defining an EcosimPro external component: "Pressure inlet", "Pressure outlet" and "Mass flow outlet". The "Pressure outlet" boundary condition can be used to set a "target mass flow rate" and could be seen in this case as a "Mass flow outlet" boundary.

However, in certain circumstances, this condition cannot be used. This topic is addressed later. The boundary conditions are listed here:

- Pressure inlet. This condition requires several inputs:

- Total pressure
- Total temperature
- Flow direction
- Static pressure
- Turbulent parameters
- Multiphase parameters

In Fluent, the variables used in the computations are the density, static pressure, velocity components, static temperature and scalar variables, if any (eddy viscosity, mass fraction, etc.). The static pressure, interpolated from the first adjacent cells, is used along with the total pressure input to calculate the Mach number by means of isentropic relations. The static temperature can be computed knowing the Mach and the total temperature at inlet. The density and the velocity magnitude can be computed using respectively constitutive relations or the Mach number and the speed of sound. Individual velocity components are set using the flow direction input. Turbulence parameters are imposed as such. The static pressure input is only used at the initialization or if the flow at inlet is supersonic.

- Pressure outlet
  - Static pressure
  - Backflow parameters
  - Target mass flow rate

The only input at a "Pressure outlet" is the static pressure, meaning that all other values are extrapolated from the interior of the domain. Backflow parameters are needed in case there are zones of reversed flow at the outlet. In this case, the inputs are the same as a pressure inlet boundary.

Finally, the target mass flow rate option can be used (except in multiphase flows). The computation iterates then on the imposed static pressure to obtain the correct mass flow rate using Bernoulli's relation.

- Mass flow inlet
  - Mass flow rate or mass flux
  - Total temperature
  - Flow direction
  - Static pressure
  - Turbulent parameters
  - Multiphase parameters

The computation process for a "mass flow inlet" boundary is comparable to a "pressure inlet" except that the total pressure is adapted to obtain the correct mass flow. The imposed mass flow gives a relation between velocity and density. Another relation between the known static pressure (extrapolated from the interior), the static temperature and the density is given by the constitutive law. Knowing the total enthalpy with the given total temperature, and using both previous relations, we can then find the static temperature and the other unknowns.

Thus, we either have the possibility to impose a resistive condition (given pressure, it returns the flowrate) or a capacitive one (given the flowrate, it returns the pressure). However, a purely capacitive component cannot be achieved. Indeed, it requires information on the mass flow on both boundaries, and assuming non accumulation of mass, this mass flow must be equal on both sides. Imposing mass flow on both sides leads to an ill-posed problem, since it lacks information about the pressure.

Other combinations can be achieved, namely resistive/capacitive, capacitive/resistive or resistive/resistive. The first combination was tested but since a hybrid component is not very suited to be encapsulated in EcosimPro, this solution is dropped. The last combination is then kept, leading to a purely capacitive, "junction-like" component.

The imposition of the boundary conditions is one of the main aspects of the coupling. Except at the initialization, this is the only moment which allows the two programs to communicate with each other. Two approaches are possible to impose the boundary conditions in Fluent without using the Graphical User Interface (GUI). At first, a scheme file can be used to access and modify the boundary inputs. However, this approach can only impose constant values on boundaries.

The second approach is selected for being much more flexible. It involves the concept of "User Defined Function". UDFs are written in "C" language and use several predefined macros to access and modify solver variables during the computation. A detailed guide on UDF is given in Fluent UDF [RD-27].

In particular, the DEFINE_PROFILE macro can be hooked to a well-defined boundary input (i.e. total pressure, total temperature, etc) and local values of this variable can be imposed on the whole boundary surface.

Currently, only constant profiles are imposed, but the extension to a variable profile is straightforward. It could provide more accuracy and an increased convergence rate in certain cases. For the test cases considered (i.e. pipe flow and nozzle flow) the influence is negligible.

Here, the total and static temperature and pressure are transmitted by EcosimPro to Fluent through text files. Total values are used at the inlet and static values at the outlet. These files are created in the "EtoF" folder and contain as prefix the label of the boundary ("inlet" or "outlet") followed by the label of the variable ("pT", "TT", "p" or "T") and end with the index of the communication interval. Be aware that this communication interval does not correspond to a physical time. The value starts off at zero at initialization and increases at each communication step.

Also, during the transient of an EcosimPro simulation, it may occur that the inlet pressure is lower than the outlet pressure. In that case, the boundary conditions must be inverted (i.e. pressure inlet at physical outlet and conversely). Therefore, both total and static information is transmitted at each boundary.

In summary, at each communication interval, each variable (total and static pressure and temperature) is read from a different text file. If the pressure drop p_{in} - p_{out} is negative, the boundary conditions are inverted. Once the values are read, every boundary profile is automatically updated using the DEFINE_PROFILE macro.

9.1.2.3.4 Flow/fluid modeling

Fluent is a general solver for simulating compressible, turbulent flows. In the current study, multiphase/species flows were not taken into account, but the extension is possible. Here, a general model setup is chosen, namely a turbulent, compressible, perfect gas flow with variable properties. This general setup can be easily modified by the user provided a little experience in Fluent.

- Turbulence modeling

  In Fluent, there are several ways of handling the viscous term. The simplest is to neglect it. Euler equations are then solved. Secondly, laminar viscous flow can be selected and Navier-Stokes equations are solved. Thirdly, turbulent flow can be assumed and RANS equations are solved (advanced turbulence modeling approach can be used like LES or Reynolds Stress Modeling but are much more expensive than RANS). In this general framework, RANS modeling is selected using the Spalart-Allmaras one-equation model. This model combines the advantages of simplicity, robustness and efficiency and conversely to k-ε class of models, it has the correct near-wall behavior. This model can be combined with wall functions if the mesh close to the wall is too coarse. Indeed, Fluent automatically detects that the wall layer is not resolved and uses wall functions to bridge the wall zone to the bulk of the flow. Moreover, even if the turbulence is mainly created close to walls, its intensity at the inlet boundary is relevant. Correlations relating the inlet eddy viscosity with the Reynolds number and the hydraulic diameter exist. A constant eddy viscosity computed according to these relations is imposed at inlet.

- Fluid properties

  There are many ways to handle fluid properties in Fluent. Unfortunately, most of them cannot be used in a general framework. Listed below are several procedures that were considered:

- Fluent material database:

  The first possibility is to resort to the Fluent material database. It is fairly easy to use and only requires matching up the name of the specified material. The drawback to this approach is that its database is incomplete. For instance, when selecting "air", it assumes that its material properties (dynamic viscosity, specific heat and thermal conductivity) are constant. For other materials, however, variations of these coefficients are implemented. This method cannot be global, and is not consistent with the way materials are handled in EcosimPro.

- Piecewise linear/polynomial evolution:

  Fluent can also be used to implement piecewise evolution of the material properties with respect to the temperature only. Unfortunately, thirty points at most can be given, meaning that there can be no systematic and accurate approach to translate the EcosimPro table into a piecewise linear evolution.

- DEFINE_PROPERTY macro:

  The DEFINE_PROPERTY macro method is very similar to the DEFINE_PROFILE method used to impose the boundary conditions. Using a "C" function, one can locally impose the material property knowing the local temperature and pressure for example by reading the EcosimPro tables. This method is used here even though the DEFINE_PROPERTY macro is defined for all properties except the specific heat Cp. Indeed, the computation of the enthalpy requires an integration of Cp and requires a smooth evolution. Nevertheless, in Fluent 6.4, a specific DEFINE_CP macro will be added to use material properties that vary as a function of the pressure and temperature.

- Real Gas models:

  Finally, Fluent provides two ways of using real gas models. The first solution is to implement user-defined constitutive laws and evolution of material properties. The second one resorts to the NIST real gas model based on the REFPROP v7.0 database. Unfortunately, the limitations of these approaches are numerous: only available with pressure-based solver, single phase and species flow.

None of these methods provides a complete, easy-to-use and accurate way to deal with material properties. The third approach is considered since it combines flexibility and accuracy. It should be complete in version 6.4. Furthermore, the extension of the DEFINE_CP should be straightforward.

- Numerical parameters/options

The Fluent solver uses a finite volume technique. Each of the governing equations is integrated on every control volume to yield algebraic relations for the discrete dependant variables (velocity, pressure, temperature and conserved scalars). Basically, there are two methods available in Fluent to solve the discretized equations: pressure-based and density-based solvers. Initially, the pressure-based one was better suited for low-speed incompressible flows and the density-based one was used for high-speed compressible flows. However, both models have been extended to operate for a wide range of flows. Unfortunately, the density-based solver is a recent feature in Fluent and is not yet available for multiphase flows. It is discarded and only the pressure-based solver is considered here.

The most important options of the pressure-based solver are detailed below:

- Gradient Option:

  One can distinguish between the node-based and the cell-based option for the evaluation of gradients. Both are based on the Green-Gauss theorem. Even so, it is recommended to select the node-based option for unstructured and especially for tetrahedral meshes to preserve second order accuracy. This option can be easily modified and depends on the type of grid used.

- Pressure velocity coupling:

  Fluent provides several segregated algorithms to resolve the pressure/velocity system (SMIPLE(C), PISO). A fully coupled procedure is also available and yields robustness and efficiency at the cost of higher memory requirements. This approach is selected.

- Spatial discretization scheme:

  A spatial discretization scheme is needed since values at cell faces are needed to compute convective fluxes. Many different schemes are available in Fluent. Among them, the first order upwind scheme is known to be fairly accurate on uniform grids (i.e. when the flow is aligned with the grid) and can provide a higher convergence rate than a second order scheme. Otherwise, when dealing with highly non-uniform flows or grids (grids made of triangles or tetrahedrons are non-uniform in the sense that the flow is never aligned with the grid) second order accuracy is needed. Among higher order schemes, the second order upwind is a good tradeoff between accuracy and computational cost.

- Time integration:

  The pressure based solver offers two pressure-velocity coupling procedures that are suited for transient simulations: PISO and Fractional Step Method. The first option is chosen, since it allows using quite a large time step without any loss of stability. Furthermore, the user can choose between first and second order implicit time integration. The second option is used even if it is slightly more memory consuming.

- Multigrid:

  Multigrid techniques are used relatively often to increase the convergence rate on dense grids. For the test cases considered here, the grids are rather light and multigrid techniques are not needed. However, they can be a worthwhile solution for larger applications.

9.1.3 Known Limitations & Suggestions

- External Component:
  - One external component can be used at a time in an EcosimPro circuit. Using more components would require as many Fluent licenses as components.
  - In this example, a two-port (inlet/outlet) component is assumed. The extension to a three (or more) ports component is not difficult but requires modifying the Fluent UDF library and the scheme file.
- Fluid/Flow modeling:
  - The fluid is assumed to be a perfect gas, single phase/single fluid. Fluent offers the possibility of modeling multiphase/fluid flows. However, the extension of the whole library is not straightforward.
  - As discussed earlier, the specific heat cannot be user-specified. This limitation should be removed in the next version of Fluent. If not, a workaround can be found.
- Numerical stability/efficiency
  - It is known that the loosely coupled approach turns out to be rather unstable. A strongly coupled approach would be more advisable for unsteady problems.
  - For two dimensional cases, the computational time is affordable. If three dimensional geometries are faced, especially in unsteady computations, the computational time will increase dramatically. Therefore, running simulations in parallel could be investigated.
- Geometry/Grid generation
  - As a possible extension of the interface, a solution should be found to avoid creating new

geometry every time the length or cross section has to be changed. As stated earlier, it is possible to scale the overall model to give it the desired dimensions, but the grid is scaled accordingly, which is not advised. However, if this scaling is combined with automatic grid adaptation, it could be a feasible solution.

- Grid generation does not only depend on the geometry of the component but also on the flow conditions. In some cases, the same grid can be submitted to a Reynolds number varying of orders of magnitude. One possible solution would be to resort to automatic grid adaptation. This feature is already available in Fluent.

9.2 COUPLING TO AN EXTERNAL OPTIMIZER

9.2.1 Introduction

Max is an optimizer whose goal is to provide a powerful tool for various engineering applications. In particular, mathematical and physical models developed in EcosimPro environment can be optimized thanks to the genetic algorithms implemented in Max without requiring a deep knowledge of their internal architecture.

Both single- and multi-objective optimization -with or without constraints- are available and can be accelerated by the use of meta-models based on radial basis function networks.

This section is divided into two parts:

- a short review of the genetic algorithms implemented in Max;
- a description of the coupling interface.

9.2.2 Description of Max

9.2.2.1 Genetic algorithms

Genetic algorithms (GA) were designed by Holland in the 1970s, and improved and made well known by Goldberg [RD-28] in the 1980s. Genetic algorithms are becoming increasingly widely used in mechanical and aerodynamic problems, e.g. preliminary design of turbines [RD-29], aerodynamic optimization using CFD [RD-30 to RD-34] optimization of target pressure distributions for inverse design methods [RD-35 and RD-36], multi objective aerodynamic shape optimization [RD-37], and multidisciplinary optimization of wing plan-form design [RD-38].

Artificial genetic algorithms mimic natural behavior in terms of biological evolution in order to reach the best possible solution to a given problem. Weak individuals tend to die before reproducing, while the stronger ones live longer and bear many offspring, who often inherit the qualities that enabled their parents to survive.

A genetic algorithm is summarized as follows. An initial population is generated by randomly selecting individuals in the whole design space. Then, pairs of individuals are selected from this population based on their objective function values. The performance of an individual is measured by its fitness. Then, each pair of individuals undergoes a reproduction mechanism to generate a new population in such a way that fitter individuals will spread their genes with higher probability. The children replace their parents. As this proceeds, inferior traits in the pool die out due to lack of reproduction. At the same time, strong traits tend to combine with other strong traits to produce children who perform better.

The reproduction cycle is governed by genetic operators. Numerous genetic operators have been proposed in the literature. Three of them are used here, namely: selection, recombination, and mutation.

- The selection operator chooses two individuals from the population that will mate to generate an offspring. Max implements several different selection operators, which are probabilistically chosen along with a recombination operator as described below.

- The recombination operators take two individuals selected by the selection operator that will be mated to generate an offspring, which tends to preserve the favorable characteristics of the parent individuals. Special attention is paid to the construction of the recombination algorithm so that it does not excessively degrade the diversity of the population. In Max, selection and recombination operators are probabilistically chosen to form a pair of operators that are combined to generate an offspring.
- Mutation introduces new genetic material into a heterogeneous population. The mutation operator implemented in Max introduces a perturbation to each design parameter with a small probability.

9.2.2.2 Genetic algorithms with meta-models

In order to speed up the optimization process (and decrease the number of calls to the external function), the genetic algorithms are combined with meta-models. Indeed, since the computation of the objectives and constraints can take hours (with CFD or FEM models for instance), and as the number of function evaluations is rather high with GAs (to fix ideas: often more than 500 times the number of design parameters), it is of paramount importance to reduce the number of calls to the expensive accurate task.

Therefore, the method implemented in Max for single-objective problems is organized as follows: first, a database is built by running the exact model on a set of initial random individuals. This database is used to create an approximate model based on radial basis functions (RBF) network. Then, a design iteration is performed, which consists of:

- executing the GA on the approximated model (and not on the accurate one)
- after the end of the GA process, computing the best solution found by the optimizer thanks to the accurate model
- adding this new point to the database in order to enrich the model and be able to construct a better approximation

As already mentioned, the approximation used in the computations is performed by radial basis functions. RBF are linear models. Examples can be found in Bishop [RD-39].

The goal of the approximator is to find the weights minimizing the error between the real outputs and those predicted by the model.

Of course, the quality of the model increases with the number of design loops, since new points are added to the database. Consequently, the move limit procedure adapts the search range of the variables as the approximate optimum improves.

Moreover, in order to guarantee diversity in the population, a *merit function* is combined to the fitness of each individual. This function takes into account the average distance of an individual with the other individuals, and favors the solutions far away from their neighbors.

To deal with constraints in a mono-objective framework, specific procedures have been implemented, as described in the next sections.

9.2.2.3 How to deal with the constraints

The handling of constraints in genetic algorithms is a delicate task, which explains the large number of techniques devised to address it.

In mono-objective optimization, Michalewicz [RD-40] and Coello [RD-41] classified them in six categories:

1. lethalization or death penalty techniques;
2. penalization techniques;
3. methods separating the objectives and the constraints;
4. methods with decoders or constraint-preserving operators;

5. repair algorithms;
6. hybrid methods.

The penalization of individuals violating the constraints is the most popular approach in the GA community. A new objective function, called *global objective function*, is defined by adding a penalty to each unfeasible solution. For instance, Joines and Houck [RD-42] proposed a method with dynamic coefficients, whose values are computed with respect to the number of the current generation:

Penalization techniques provide good results without significant modification of the standard evolutionary algorithm. However, the difficulty in the choice of the parameters constitutes their main drawback, because no general rule can be applied to determine their values.

To alleviate this problem, Deb proposed a constraint tournament selection method that works as follows [RD-43]:

- when two individuals are compared, the feasible one is always preferred to the unfeasible one
- if both individuals are feasible, the one with the best objective function value is chosen
- if both individuals are unfeasible, the one with the least violation of the constraints is preferred

This technique has demonstrated itself to be successful, especially in providing feasible solutions in highly constrained problems. But as a drawback, the optimizer might focus in local extrema too rapidly if the design space is scarcely populated; thus it is strongly suggested to use as many samples as possible so that multiple extrema (if they exist) are well represented.

Both methods (Joines and Houck's and Deb's) have been implemented in Max GAs. However, the handling of the constraints must not only be performed the GA, but also during the calculation of the global fitness function used for validation (which is required to compare the best solutions found by the GA). In this case, it is thus necessary to construct a global fitness function taking into account the constraints, as in penalization. Therefore, three different penalty methods have been implemented:

- static penalty
- Joines and Houck's dynamic penalty
- Bean and Hadj-Alouane's adaptive penalty [RD-43]

9.2.3 Coupling Process

Max is built on object-oriented architecture and has been converted to an API class for fast and efficient integration with external tasks like EcosimPro.

Indeed, Max can be compiled as a static library. This kind of libraries can be easily called from EcosimPro. In this example, a single EL file has been created to call the Max main function. Thanks to that EL file, the Max main function can be used from every EcosimPro experiment. This main function has several arguments that will be described in the rest of this section.

9.2.3.1 Building the function to optimize

The first step consists in building an experiment inside a partition. The OPTIMIZATION library must be explicitly called by mentioning USE OPTIMIZATION at the top of the EL experiment file. An experiment function must be written right after: it will contain the definition of the parameters, objectives and constraints to optimize.

In this particular case, the function to optimize is a second order polynomial with two variables (x_1 and x_2).

$$f(x_1,x_2) = 100(x_2-x_1^2)^2+(1-x_1)^2.$$

Their domain of definition is [-2,2]. The minimum is located at (1,1) and equals zero.

```
FUNCTION NO_TYPE fcn_rosenbrock (OUT INTEGER nDesignVariables,
       OUT REAL designVariables[],
       OUT REAL responseVariables[],
       OUT INTEGER successSwitch)

       DECLS
              REAL x1,x2,f
       BODY
              x1 = designVariables[1]
              x2 = designVariables[2]
              f = 100*(x2-x1**2)**2+(1-x1)**2
              responseVariables[1]= f
              successSwitch= 1
END FUNCTION
```

The *designVariables* are real (continuous) variables, whereas the *responseVariables* are the functions (objectives and constraints) needed by the optimization problem.

Note that the objectives must be stored in the response vector before the constraints (in other words: the objectives must always have lower indices than the constraints in the response vector).

The *successSwitch* variable is used to indicate that the EcosimPro computation finished properly. It can be seen as an additional test judging whether the computation can be added to the optimization samples. On the other hand, whenever a fatal error occurs and eventually causes the premature end of the integration, Max will automatically set this parameter to zero and continue the optimization process.

9.2.3.2 Defining the optimization problem

After the definition of a function computing the equations governing the behavior of the component and defining the objectives and constraints, the variables and parameters of the optimization have to be defined, as well as an object of *ExternalOptimizer* class (which will be called to optimize the function).

```
EXPERIMENT optimization ON test.default

       DECLS -- setting the parameters for MAX optimization
              INTEGER nDesignVariables= 2
              INTEGER nResponseVariables= 1
              INTEGER nObjectives= 1
              INTEGER objectiveType[0]=  0   -- minimize
              INTEGER nConstraints= 0 -- no constraints
              INTEGER constraintType[2]= (1, 1}   -- upper bound
              INTEGER optimizationAlgorithm= 1
              INTEGER reproductionCycleCount= 20
              INTEGER populationSize= 20
              INTEGER designLoopCount= 10
              INTEGER initialDatabaseStrategy= 0
              REAL designVariables[2]= {0.5, 0.5}
              REAL designVariablesLower[2]= {0.0, 0.0}
              REAL designVariablesUpper[2]= {2.0, 2.0}
              REAL constraintBound[2]= {0, 0}
              REAL solutionSet[1000]

       OBJECTS
              ExternalOptimizer optimizer -- defining an object

       INIT -- set initial values for variables
              (...)

       BOUNDS -- set expressions for boundary variables: v = f(t,...)
              (...)

       BODY -- launching the optimization
              optimizer.Optimize(fcn_rosenbrock,
                     nDesignVariables,
                     designVariables,
                     designVariablesLower,
```

```
                        designVariablesUpper,
                        nResponseVariables,
                        nObjectives,
                        objectiveType,
                        nConstraints,
                        constraintTyp e
                        constraintBound,
                        solutionSet,
                        optimizationAlgorithm,
                        reproductionCycleCount,
                        populationSize,
                        sampleCountRatio,
                        designLoopCount,
                        initialDatabaseStrategy)
    END EXPERIMENT
```

In the declaration field, all the variables required by the optimization must be set. They are listed in Table 9-2:

Variable	Type	Description
fcn_rosenbrock	FUNC_PTR	name of the function to optimize
nDesignVariables	INTEGER	number of design variables
designVariables	REAL[]	vector of design variables
designVariablesLower	REAL[]	vector of lower bounds for the design variables
designVariablesUpper	REAL[]	vector of upper bounds for the design variables
nResponseVariables	INTEGER	number of response variables
nObjectives	INTEGER	number of objectives
objectiveType	INTEGER[]	vector defining the goals of the objectives (0: minimize ; 1: maximize)
nConstraints	INTEGER	number of constraints
constraintType	INTEGER[]	vector defining the natures of the constraints
constraintBound	REAL[]	vector defining the bounds of the constraints
solutionSet	REAL[]	vector containing the solution obtained by the algorithm (see remark R1 for more detail)
optimizationAlgorithm	INTEGER	algorithm used for the optimization (see remark R2 for more detail)
reproductionCycleCount	INTEGER	number of generations of the genetic algorithm
populationSize	INTEGER	size of the population of the genetic algorithm
designLoopCount	INTEGER	number of design loops for the genetic algorithm combined with approximation models (see remark R3 for more detail)
initialDatabaseStrategy	INTEGER	should be set to 1 if an existing database has to be used

Table 9-2: Summary of the parameters used in the optimization library

Here are some remarks about three variables:

- (R1) *solutionSet*: this vector contains the solution found by the algorithm. In single-objective optimization, it consists of a vector whose first elements are the design variables, followed by the objectives (and possibly the constraints)
- (R2) *optimizationAlgorithm*: the user can choose between 2 optimization algorithms:
  1. single-objective genetic algorithm (with advanced operators)

2. single-objective genetic algorithm combined with an approximate model based on radial basis function networks (to speed up the process when time consuming models are used)

   Multidisciplinary optimization is not yet incorporated into the EcosimPro environment, although this functionality is available within Max.

- (R3) *designLoopCount*: when the GA is combined with an approximate model, the process is divided into two levels: a general level consisting in building the approximate model and running the exact simulation, and a local level (the GA applied to the approximate model). The number of design loops corresponds to the number of runs of the GA on the approximate models. (cf. Figure 9-3)
- (R4) *initialDatabaseStrategy*: this variable is only used when algorithm 2 or 4 is selected. It separates the processes of database creation and optimization. When the database is built, one file containing the database information (*db_out.dat)* is created in the *EcosimPro/bin* directory. If the *initialDatabaseStrategy* is set to 1, Max will look for *db_in.dat* in the same folder. The user should then rename *db_out.dat* so that it can be used as input.

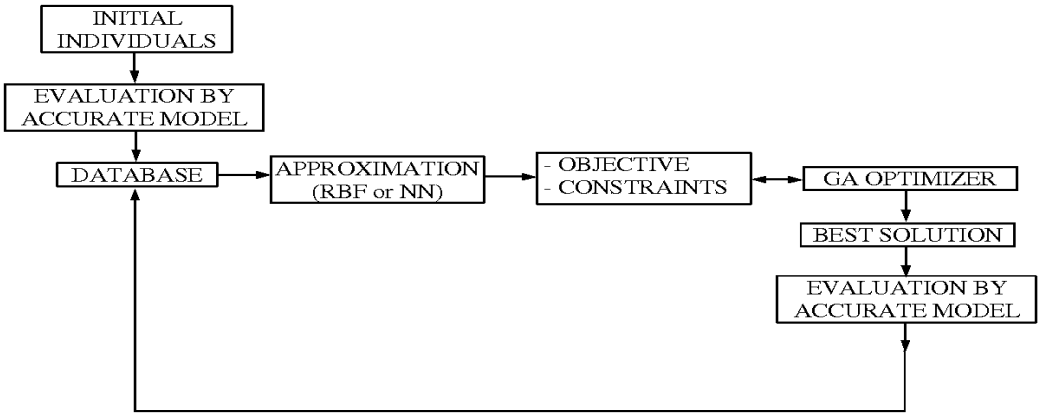


Figure 9-3: Flow-chart of the GA combined with meta-model (in 1-objective optimization)

In general, when the duration of the EcosimPro experiment is very short (i.e. simple models or steady cases), using GA is recommended. The latter requires more function evaluation than when it is combined with an approximate model, but does not require building an approximate model (which is the most time consuming step when the response time of the EcosimPro experiment is very short).

The number of function evaluations and the accuracy of the results for the simple problem described above are summarized in Table 9-3:

Optimization algorithm	**Number of function**	**Best response value**
Classical GA	2253	2.3×10^{-13}
Approximated GA (50)	60	5.6×10^{-8}
Approximated GA (100)	110	2.7×10^{-12}

Table 9-3: Comparison in terms of efficiency and accuracy of the classical GA and the approximated model on the Rosenbrock function.

Figure 9-4 shows a comparison between the convergence history of the GA with and without metamodel. It is seen that convergence occurs much faster with the meta-model.

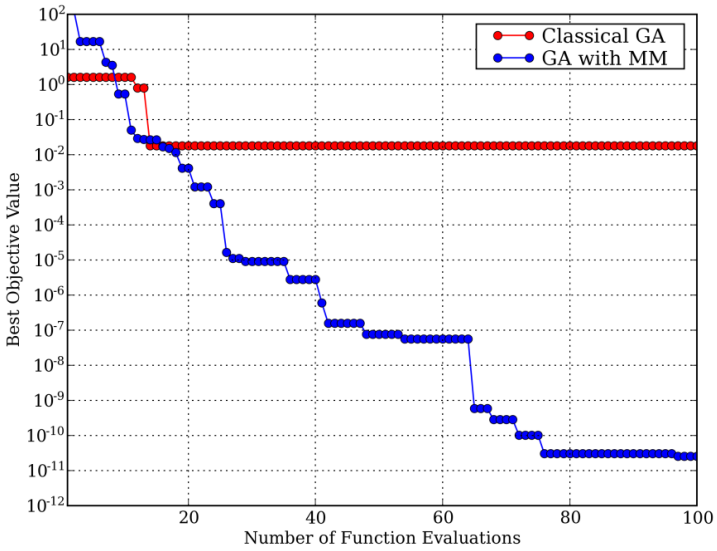


Figure 9-4: Convergence history of the GA with and without meta-model on the Rosenbrock function

9.2.3.3 Running the optimization

Once the parameters of the problem are defined, and the object *opti* created, the optimization process can be launched by clicking on the *Run* button in EcosimPro.

Max creates a report in a file called *Max_solution.txt*. Should the solution found by the algorithm be unfeasible, this would be indicated in the report. In all cases, when there are constraints, their values for the best design are written.

Furthermore, the *db_out.dat* file is related to the database, and is created in the *EcosimPro/bin* directory only if algorithm 2 is used.

10. APPLICATION EXAMPLES

Besides the example given in Chapter 3 (a two-phase water hammer case), additional interesting *transient* application examples provided with the FLUID_FLOW_1D_EXAMPLES and ROCKETS_EXAMPLES libraries are presented below. The corresponding simulated systems can be used as templates for more complicated models.

10.1 COLD THRUSTER

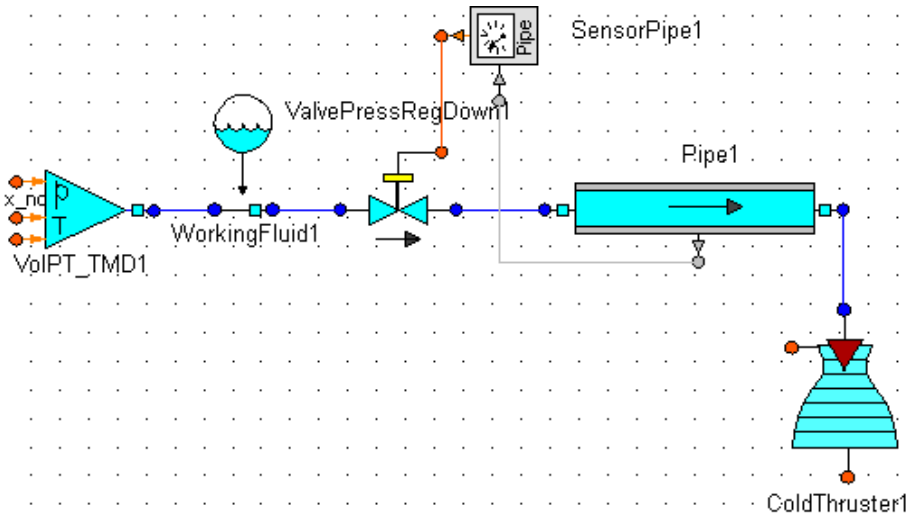
Library:	FLUID_FLOW_1D_EXAMPLES
Model Name:	test_coldThruster
Partition Name:	default
Experiment Name:	exp1

10.1.1 Model Description

The following example shows the FLUID_FLOW_1D capabilities concerning subsonic/supersonic transitions in a nozzle using fluids with real properties. The nozzle is simulated by a particular Pipe component, which takes into account variable area, inertia forces, pressure losses and density changes.

Shock capture inside the supersonic part is modeled depending on the outlet pressure.

The model with ESPSS is representative of a typical cold thruster system including a pressure regulation valve from High Pressure (HP) to Intermediate Pressure (IP), a pipe for taking gases at IP to the thruster and finally the thruster itself provided with an on/off valve:

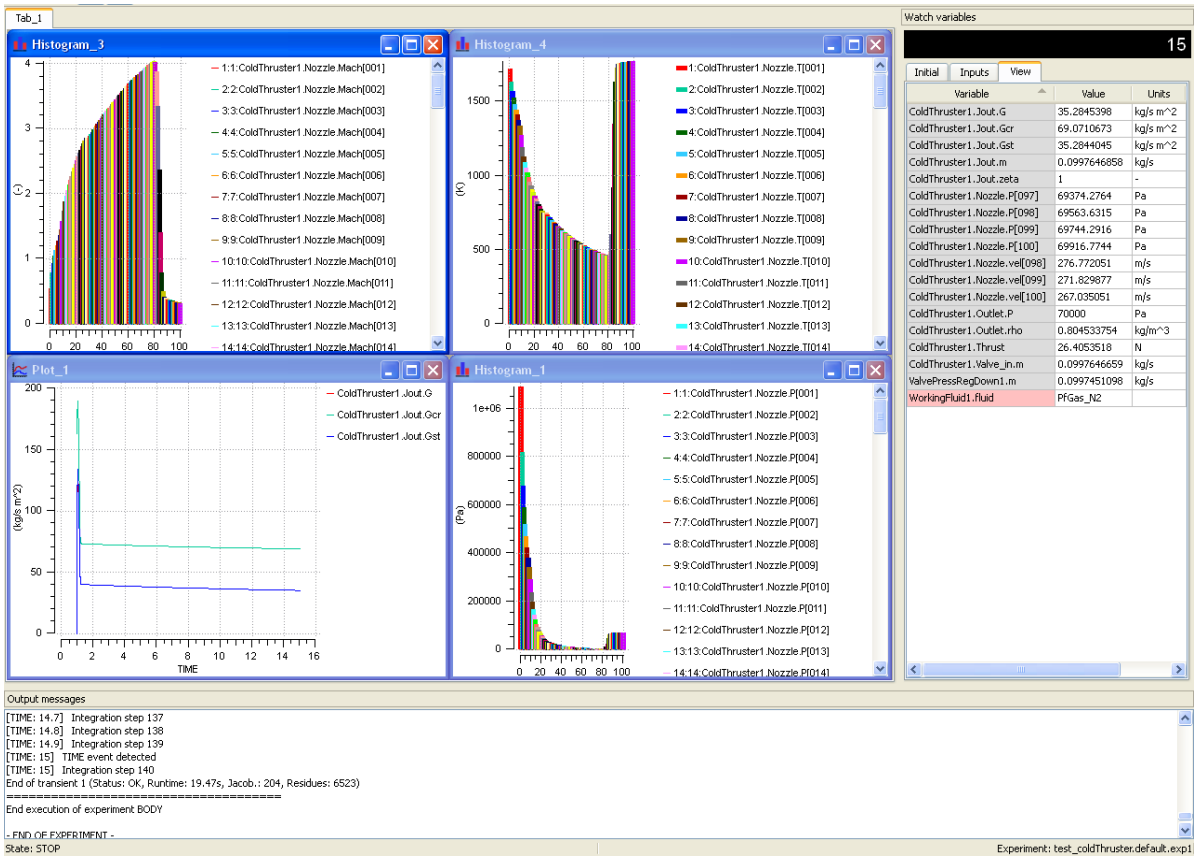


10.1.2 Results

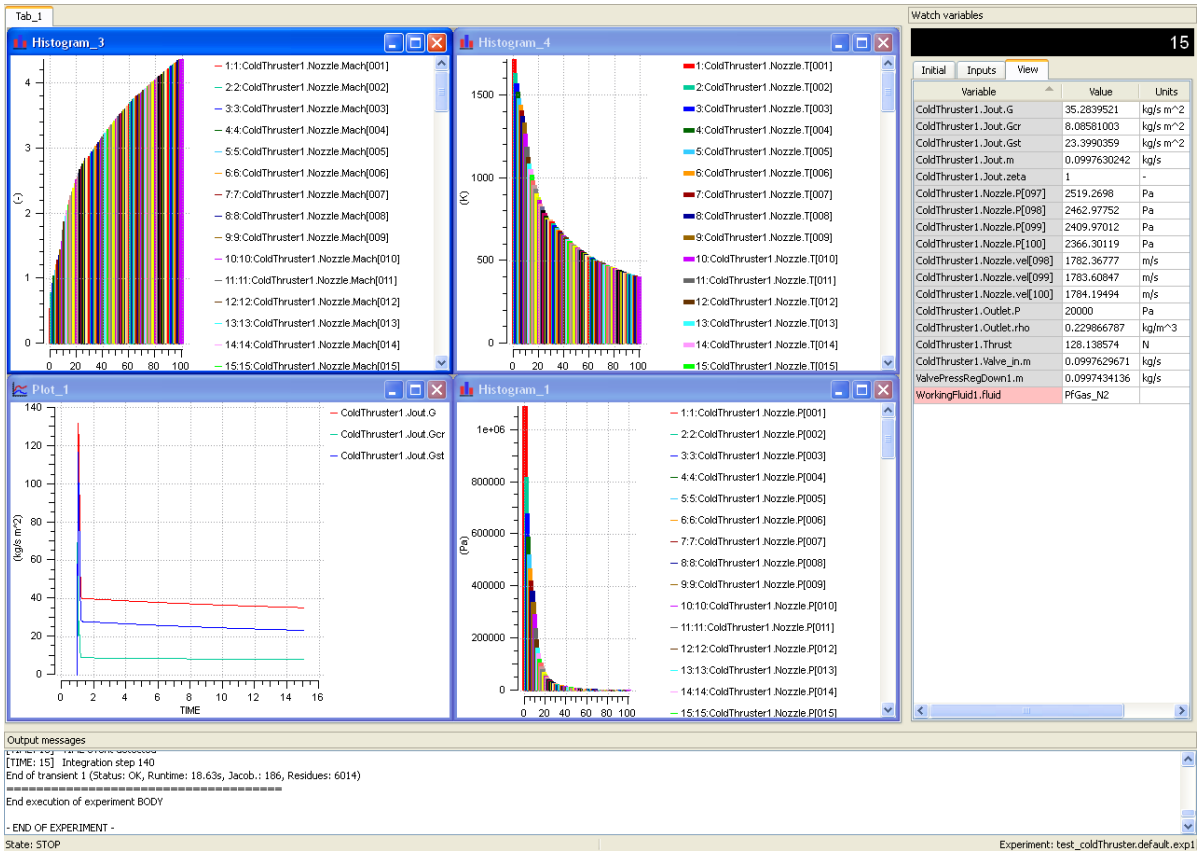
The plots shown below are direct output from the EcosimPro monitor:

- The ESPSS model can correctly simulate the three stationary cases: shock inside the nozzle, outside the nozzle and attached to the exit
- The ESPSS model can correctly simulate transient cases passing from adapted conditions to non-adapted conditions and vice versa
- Real fluid conditions can be calculated, even if the fluid is condensing inside the nozzle due to the cold temperatures reached in supersonic flow

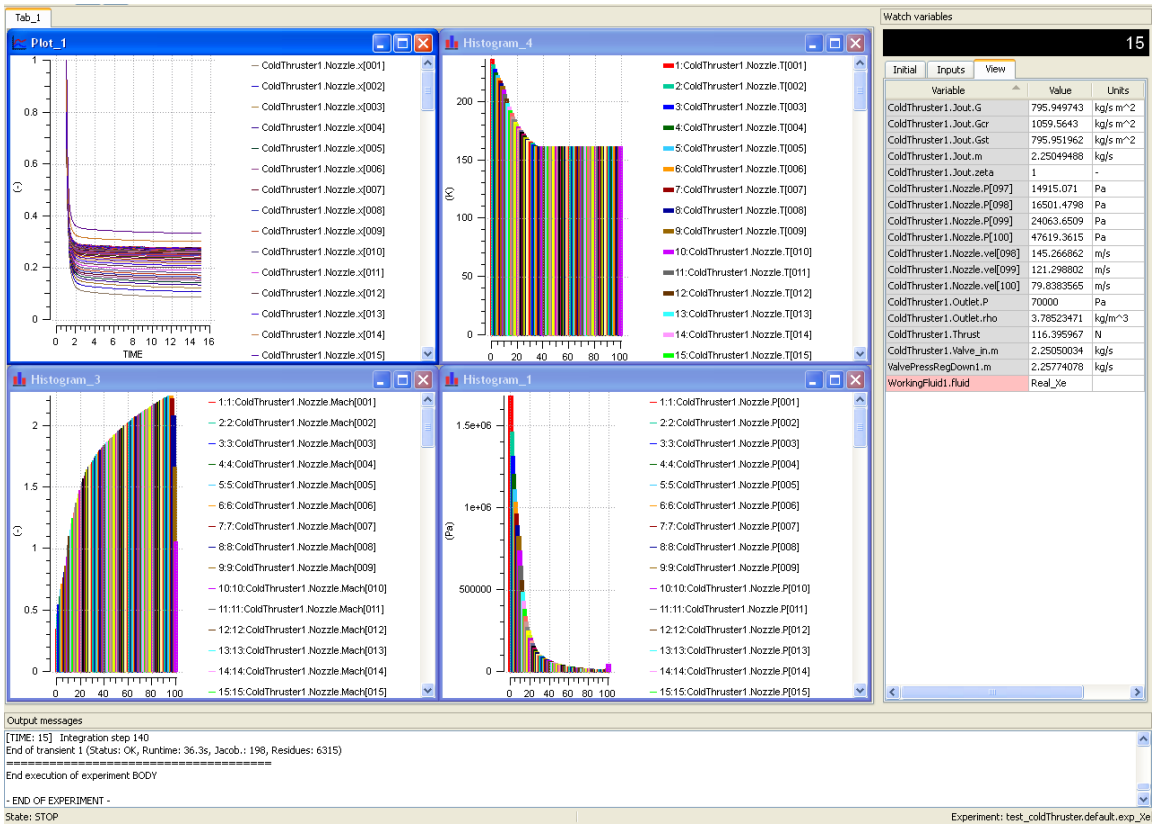
Perfect gas N2 (Tin= 2000 K; Pin = 20 bar; Pamb = 0.7 bar): Shock inside the nozzle:



Transition to Pamb = 0.2 bar: Shock attached to the nozzle exit



Real Xe properties (Tin= 250 K; Pin = 20 bar; Pamb = 0.7 bar): Liquefaction in supersonic region

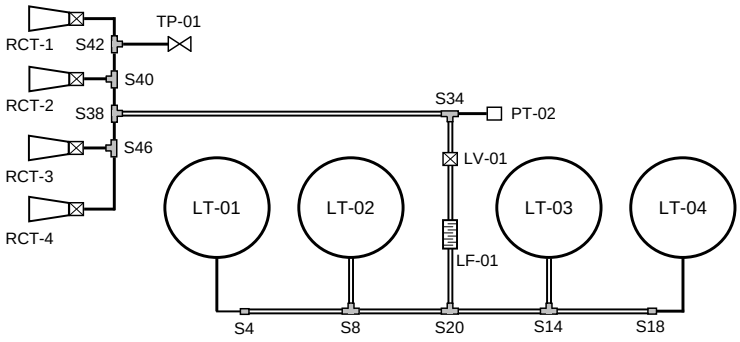


10.2 RCS CASE. THRUSTER FEEDING SYSTEM

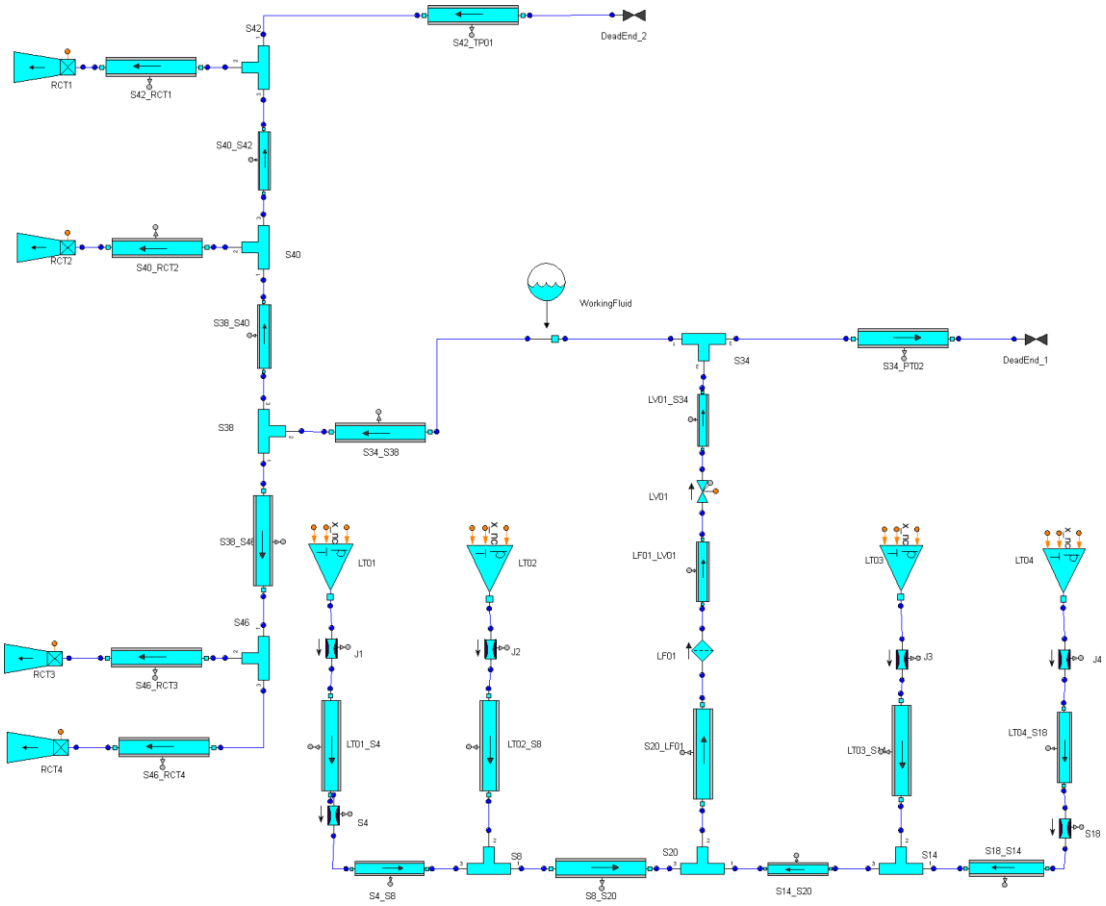
Library:	FLUID_FLOW_1D_EXAMPLES
Model Name:	Test_RCS
Partition Name:	default
Experiment Name:	exp2

10.2.1 Description of Model

This model is representative of a satellite propulsion system with hydrazine**.** *The thrusters are simulated using an empirical simulation model, so no combustor components (no chemical reactions) are included in this case***.**



The EcosimPro schematic diagram of the model appears in the figure below, which includes the main feeding lines:



The main data are described below:

PIPE	LENGTH [mm]	DIAMETER [in]	PIPE	LENGTH [mm]	. DIAMETER [in]
LT01-S4	290	0.25	S34-S38	113	0.375
LT02-S8	275	0.25	S38-S40	189	0.25
LT03-S14	273	0.25	S40-S42	1980	0.25
LT03-S18	287	0.25	S42-RCT1	1750	0.25
S4-S8	2355	0.375	S40-RCT2	1684	0.25
S8-S20	212	0.375	S38-S46	1498	0.25
S20-S14	1647	0.375	S46-RCT3	1579	0.25
S14-S18	2117	0.375	S46-RCT4	3771	0.25
S20-LF01	1127	0.375	TP01-S42	520	0.25
LF01-LV01	263	0.375			
LV01-S34	113	0.375			

The RCT (Reaction Control Thruster) Performance Characteristics are coded in a particular component (RCT.el) in which the following equations are directly coded:

$$thrust: \quad T[N] = A \cdot \sqrt{P_{in}[bar]} + B$$
$$specific\ impulse: \quad I_{sp}[Ns/Kg] = 2.04 \cdot P_{in}[bar] + 2203.5$$
$$mass\ flow\ rate: \quad \dot{m}[Kg/s] = T/I_{sp}$$

RCT	A	B
1	6.628	-7.669
2	6.588	-7.619
3	6.641	-7.713
4	6.602	-7.388

10.2.2 Results

The following phenomena can be shown:

- Simultaneous opening of the 4 RCT valves (tanks initial pressure of 25 bar, tank liquid temperature kept constant at 20°C) produces large shifts in the line pressures
- Steady conditions can be easily calculated in a few CPU seconds by running the model

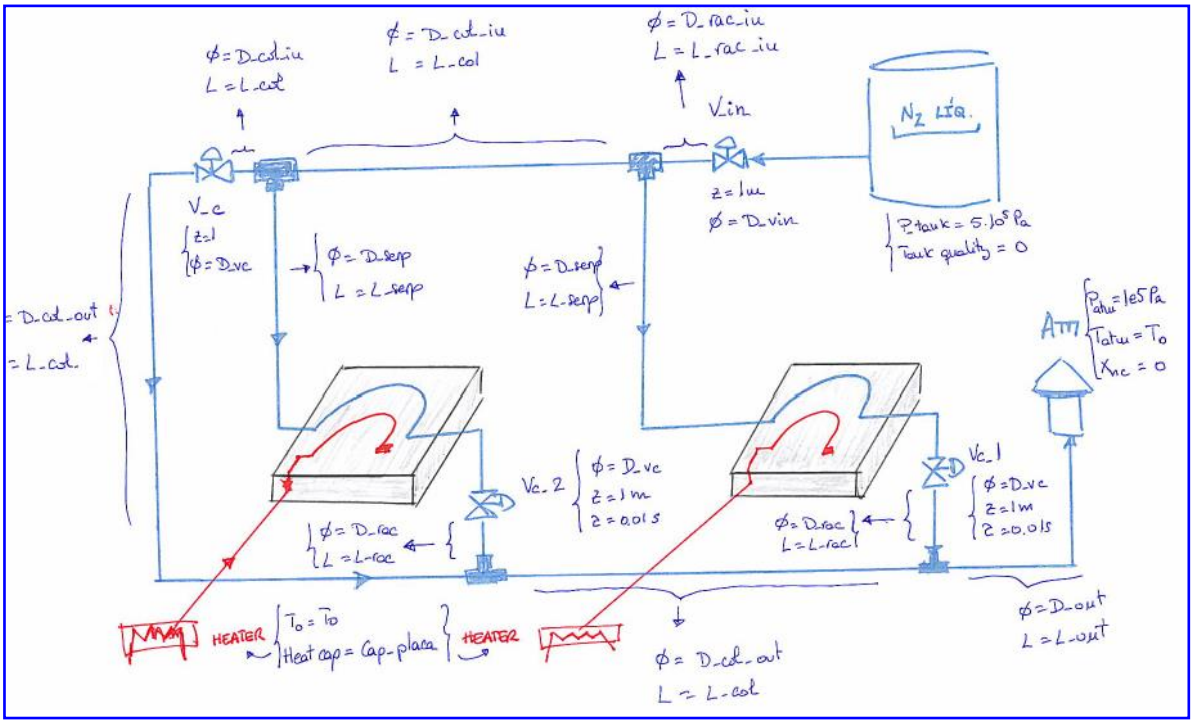
10.3 COLD PLATE TEST BENCH

Library:	FLUID_FLOW_1D_EXAMPLES
Model Name:	Cold_Plates_Model
Partition Name:	default
Experiment Name:	exp1

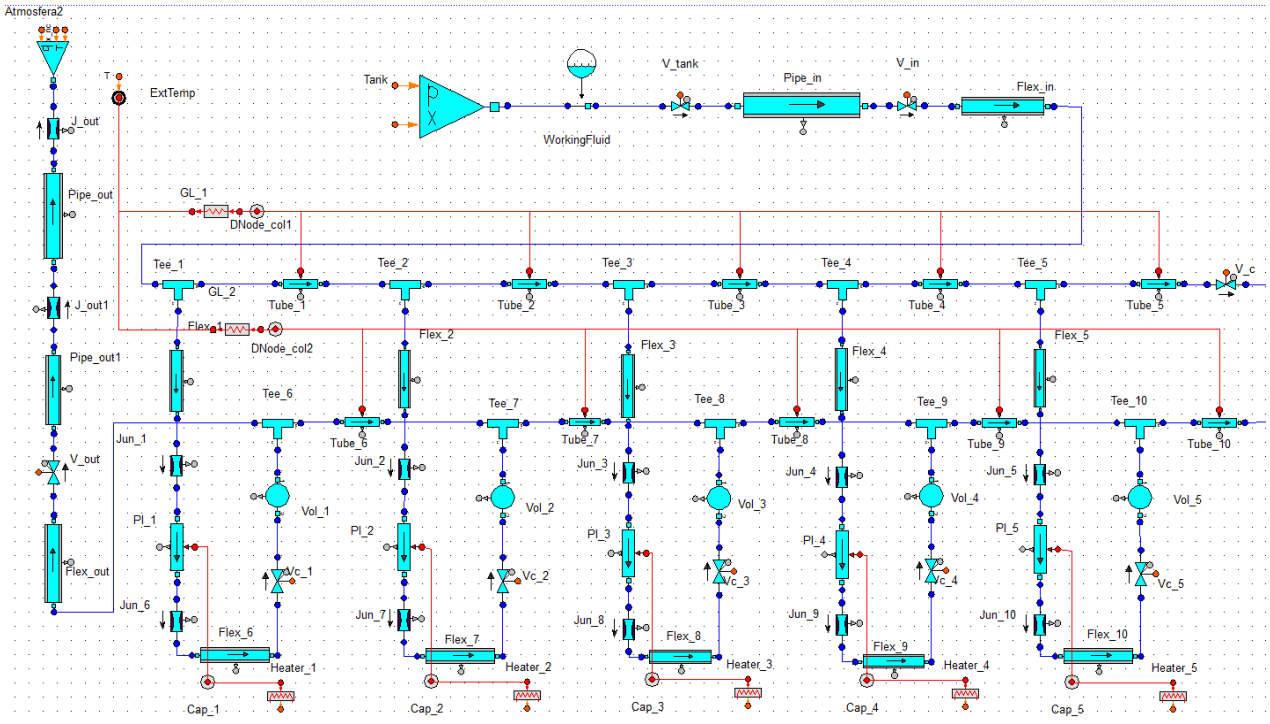
10.3.1 Model Description

This example shows the FLUID_FLOW_1D capabilities related to the design of a test bench with several plates to be cooled by N_2. The plates must dissipate a thermal flux of 400 W so that an electronic device can be tested.

The temperature control consists of a two-phase flow pipe system with nitrogen going through the plates in order to keep the plate temperature at a certain value. The N2 is taken from a conventional tank with N2 in saturated liquid conditions:



The corresponding ESPSS model is:



The plates (aliased "cap") are modelled by Dnode THERMAL components whereas the internal tubes circulating two-phase N2 are modelled by the Tube component.

The valves (Vc1, Vc2, etc.) regulating the two phase flow have the following temperature control included in the CONTINUOUS block of the model (the "edit source code" button in the schematic view of EcosimPro):

```
…
DISCRETE

    WHEN (Cap_1.tp_in.Tk[1] > (T_set1 + 5)) THEN      --Valve1 switch on
            Vc_active[1] = TRUE
    END WHEN
    WHEN (Cap_1.tp_in.Tk[1] < T_set1) THEN            --Valve1 switch off (no flow
            Vc_active[1] = FALSE
    END WHEN
    WHEN (Cap_2.tp_in.Tk[1] > (T_set2 + 5)) THEN
            Vc_active[2] = TRUE
    END WHEN

    --Idem Valve2 control
    WHEN (Cap_2.tp_in.Tk[1] < T_set2) THEN
            Vc_active[2] = FALSE
    END WHEN
    WHEN (Cap_3.tp_in.Tk[1] > (T_set3 + 5)) THEN
            Vc_active[3] = TRUE
    END WHEN
    …

CONTINUOUS
    --Idem Valve2 control
    Vc_1.s_pos.signal[1] = ZONE (Vc_active[1]) 1 OTHERS 0
    Vc_2.s_pos.signal[1] = ZONE (Vc_active[2]) 1 OTHERS 0
    Vc_3.s_pos.signal[1] = ZONE (Vc_active[3]) 1 OTHERS 0
    Vc_4.s_pos.signal[1] = ZONE (Vc_active[4]) 1 OTHERS 0
    Vc_5.s_pos.signal[1] = ZONE (Vc_active[5]) 1 OTHERS 0
    …
```

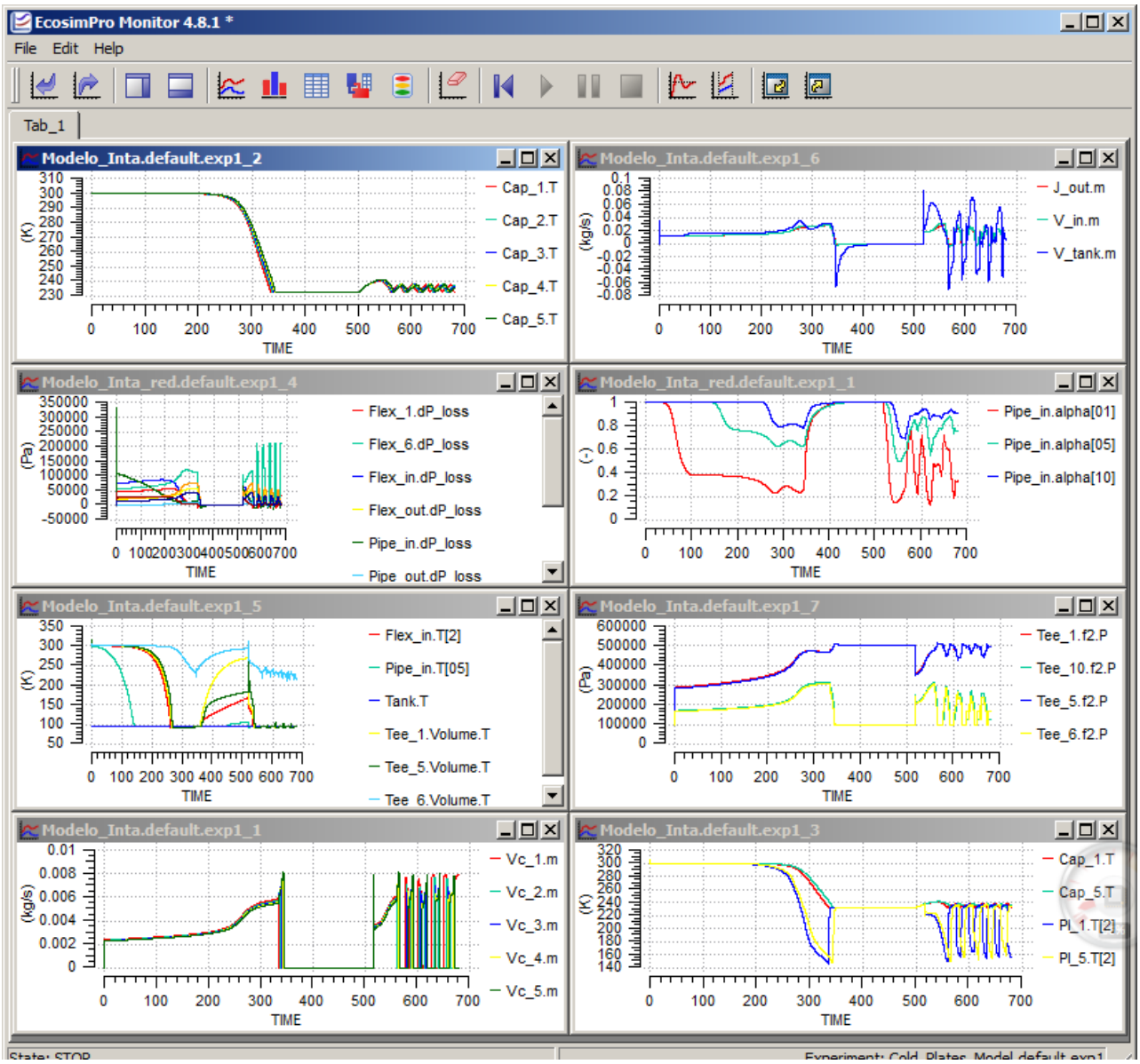
Similar results may be obtained by having the model diagram include equivalent CONTROL components to carry out the temperature comparisons (P controlers for example) and feed their outlet signals back to the inlet control port of the valves.

10.3.2 Results

The following phenomena can be seen:

- A first transient phase where the system goes from the initial ambient conditions to the imposed plate temperature, in which the model calculates the temperature distribution, quality (two-phase flow distribution) and pressure losses of the piping
- At TIME = 400 seconds, the heat flux is applied to the plates producing the opening and closing of the control valves
- In these circumstances, the model is able to calculate the efficiency (temperature overshoot) on the plate control system as per the valve dimensions, tubing, temperature threshold, etc.

Below the main plots of the simulation:



10.4 LOX PRESSURIZATION SYSTEM

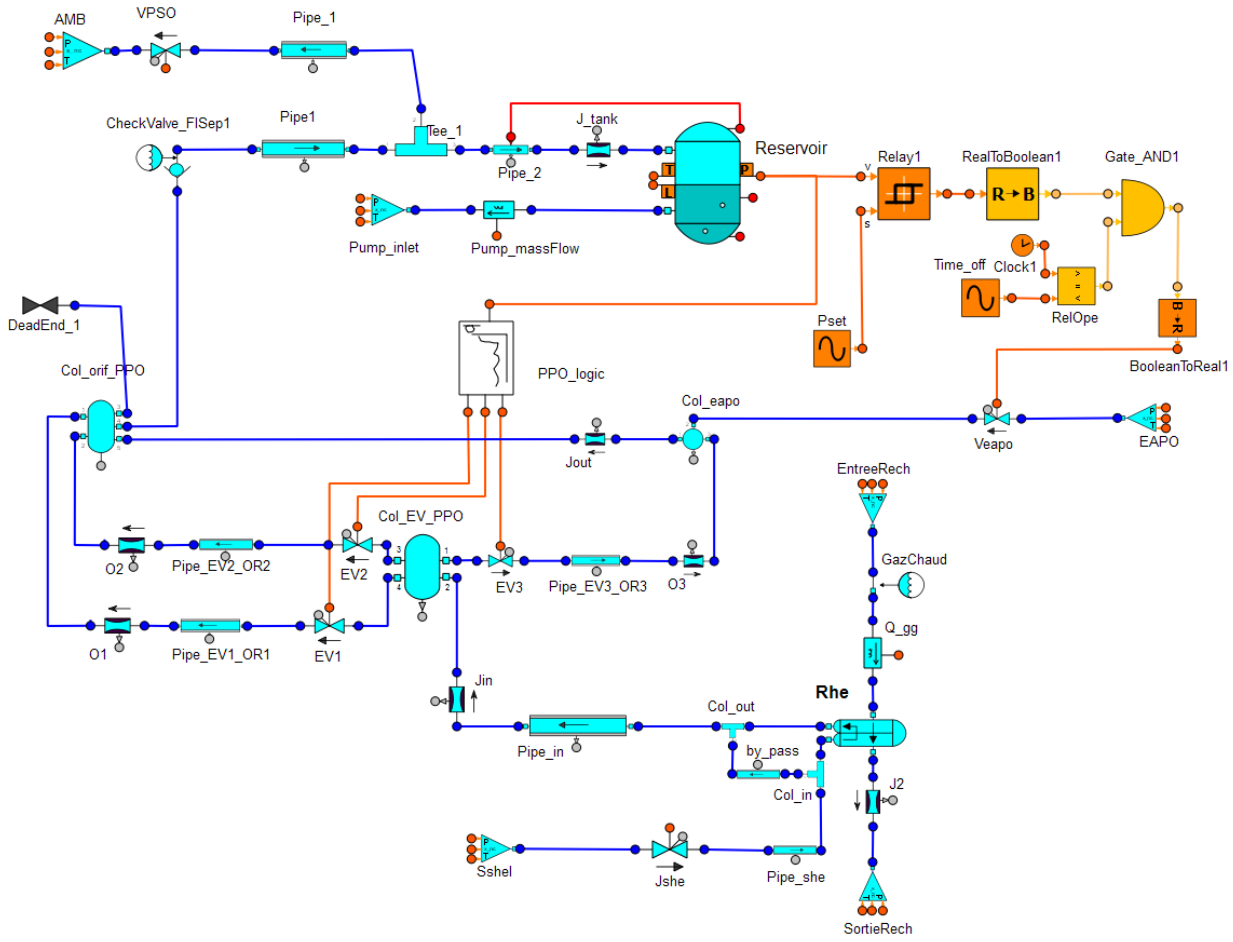
Library:	ROCKET_EXAMPLES
Model Name:	PressureCircuit_LOX
Partition Name:	default
Experiment Name:	exp1

10.4.1 Description of Model

A model showing a pressurization system by means of electro-valves is presented below. The Tank (containing LOX, He and GOX vapors as pressuring gas) is simulated with the more complex component capturing vaporization, fluid temperature distribution, etc. Liquid exiting the tank through the pump is represented by the single Jun_TMD component (Pump_massflow).

The pressurization system consists of a set of three latch valves (PPO subsystem). One of them is normally open. The second will open when the tank pressure goes down to predefined thresholds, but

variable in time --see the PPO_logic component source code (*Prlogic* type component)-- and will close when the tank pressure rises to another predefined threshold. The third one is normally closed.



The PPO collector is pressurized with He coming from the SShel Sphere at very cold temperature which is heated when passing through the He reheater RHE inside the turbine exhaust lines ("GazChaud" flow). The reheater subsystem includes a passive control temperature thanks to the bypass lines.

Finally, the EAPO subsystem pressurizing the tank since hearth is also modeled. This system will pressurize the Tank from TIME -245 till -10 seconds; see the control components "Time_off". "RelOpe". The Relay control component is responsible to switch the valve between +-0.15 bar around the set point imposed by "Pset". For times lower than -245 seconds, the Tank is pressurized by boiling exiting vapors to the atmosphere through the valve VSPO.

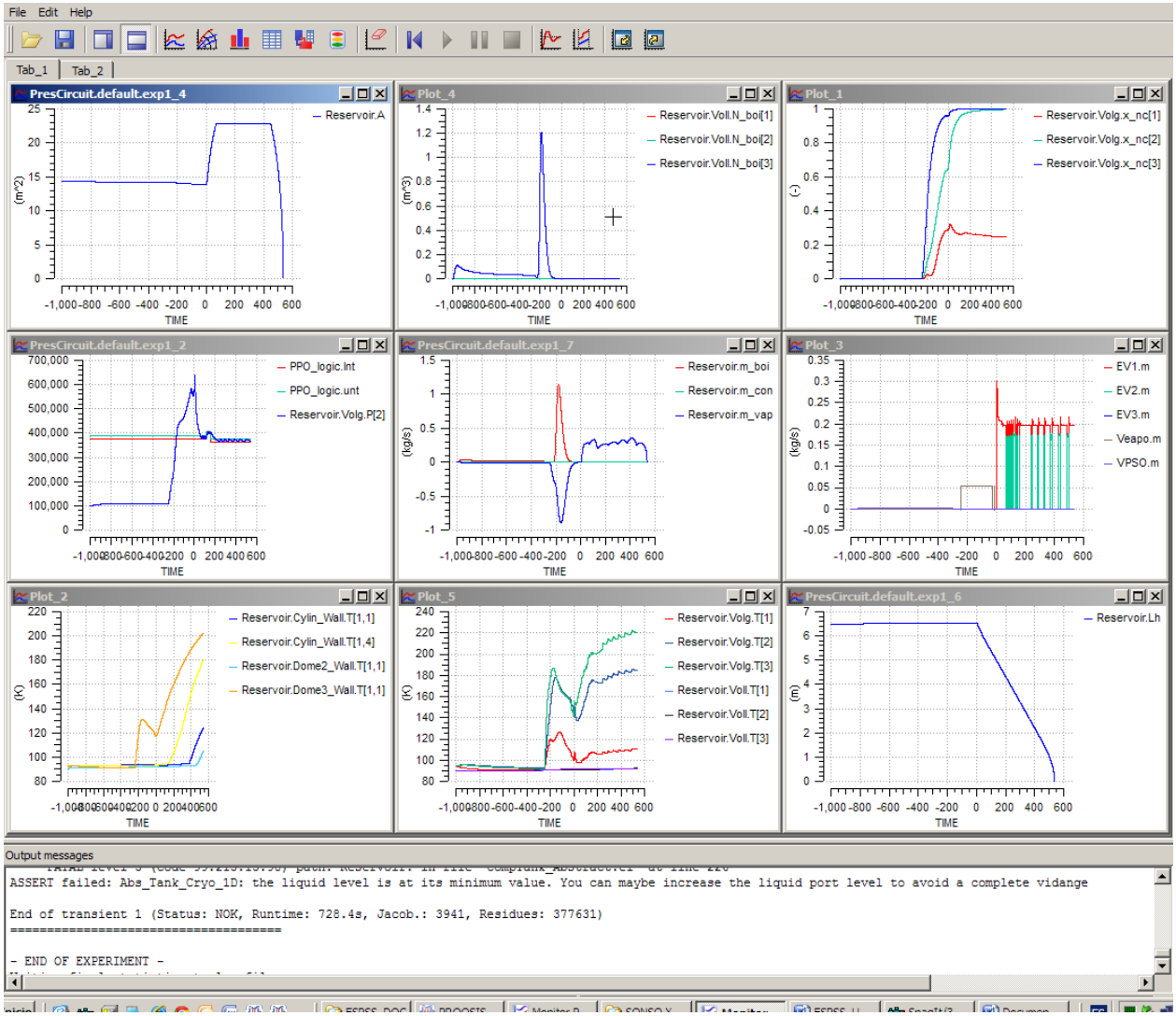
The boundaries, valve activation orders and some input data redefinition (initial conditions) are given in the experiment file.

10.4.2 Results

Below are the main plots obtained. The following is of note:

- For time before -300s, the Tank remains in boiling pressurization because no gas is injected inside; on the contrary, a small outlet vapor flow exiting the Tank by the VPSO valve compensates any possible heat leaks.
- Before the startup of the engine at TIME-300, the EAPO subsystem pressurizes the Tank at P=5 bar. Then, at TIME = -10 this system is switched off and the PPO subsystem is activated.
- The tank pressure evolves (in accordance with the latch valve activations) between the predefined thresholds. The vaporization level is very important in determining the number of valve activations.

- The RHE model reproduces the real behaviour of the system well.
- The plot with variables N_boi[] gives the amount of bubbles rising in the liquid due to film boiling near the walls.

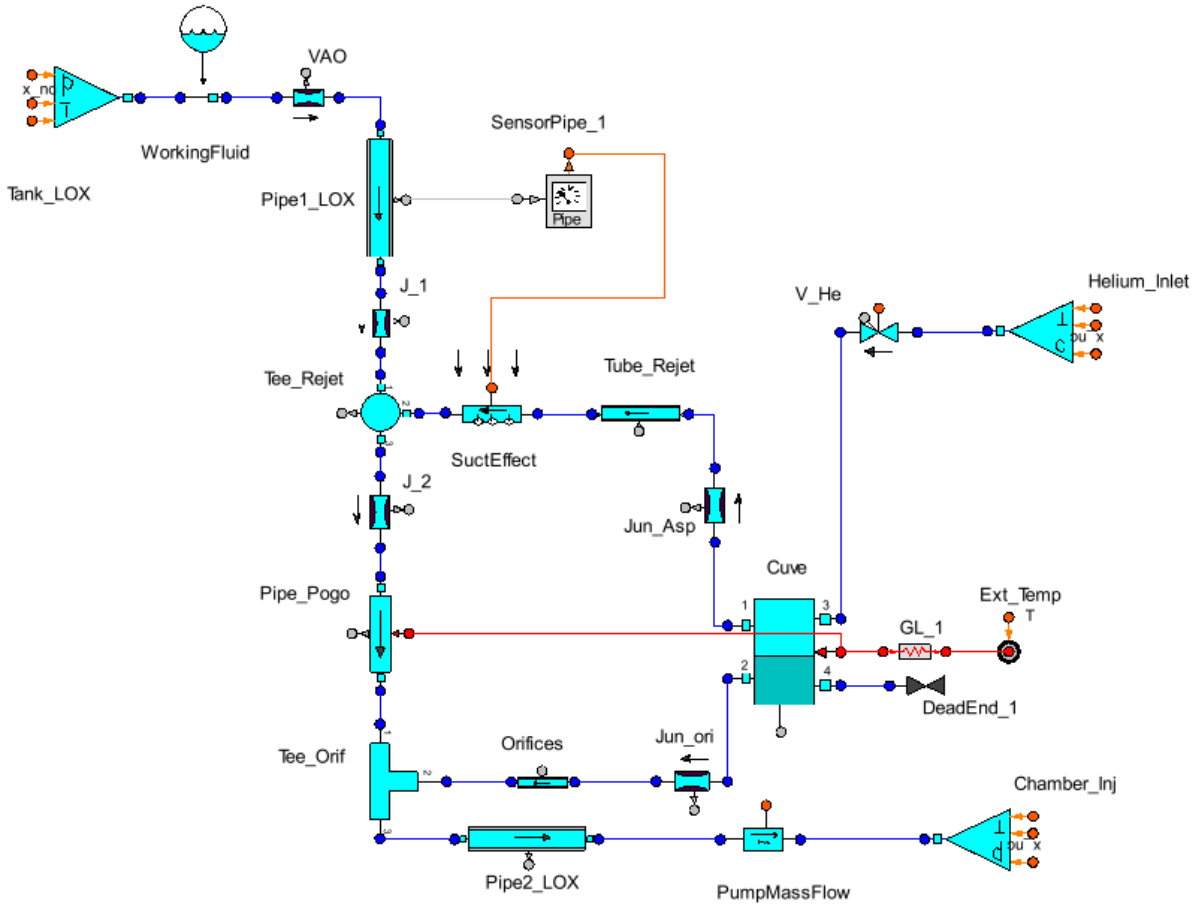


10.5 POGO MODEL

Library:	ROCKET_EXAMPLES
Model Name:	PogoSystem
Partition Name:	default
Experiment Name:	exp1

10.5.1 Model description

A model showing a device to smooth the POGO oscillations in a cryogenic feeding line is presented in this example. The following physical phenomena will be retained: overflowing a tank outlet (reject tube), damping of the pressure oscillations with non-condensable bubbles in a liquid line, pressure rise due the hydraulic height, suction and plugging effects in a tube, etc.



The system consists of an accumulator tank (Cuve) working under two-phase conditions (liquid and vapor Oxygen) with Helium. The homogeneous equilibrium model (HEM) is applied

The tank is pressurized with Helium to maintain a gas cavity in the accumulator. The excess of helium is rejected to the main liquid line through the "Tube_Rejet" pipe. Another communication between the liquid in the lower part of the tank and the main line is made through holes (Orifices component with inertia) giving a new characteristic response time to the system

All the orifices connecting the main line with the accumulator have been modeled by only one pipe with the same orifice length and an equivalent hydraulic diameter

The main line mass flow is imposed by a Jun_TMD component ("PumpMassFlow") simulating the Pump.

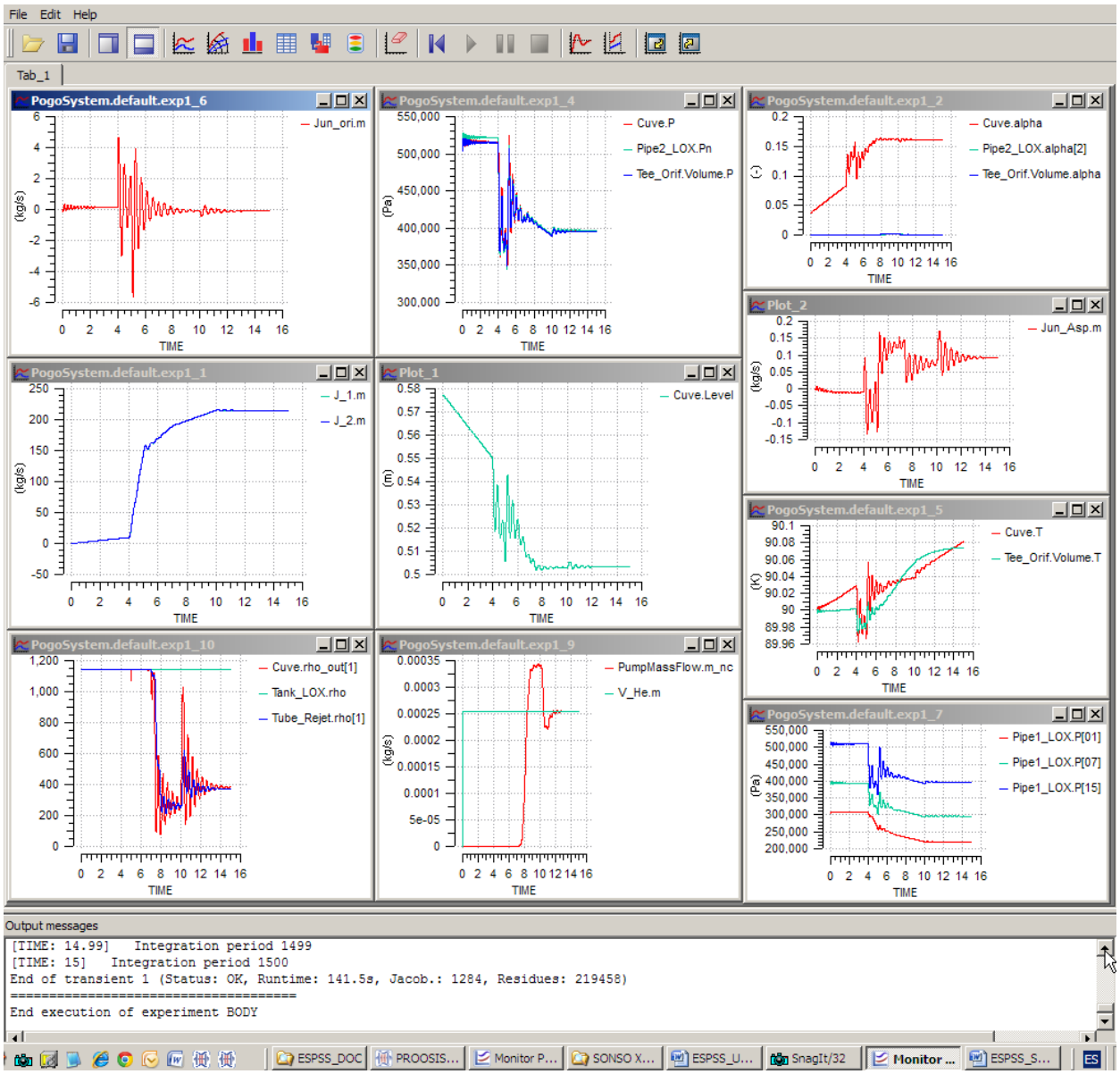
The boundaries, valve activation orders and some input geometrical data are given in the experiment file.

10.5.2 Results

Without an accumulator system, the pressure oscillations in the main line caused by the start up of the pump would be very sharp. The following plots show how much these oscillations are reduced. Similarly, the frequency of the oscillation is also reduced. Herebelow are the main plots and conclusions obtained:

- Depending on the ΔP calculated by the "SuctEffect" and "Orifices" components and on the Helium mass flow (V_He valve), different stationary liquid levels can be foreseen in the accumulator.
- For low rates of Helium, the equilibrium is in a semi-plugged situation, so that the tube flow regime is two-phase with Helium.
- Trade-offs performed over some geometrical dimensions (orifices, tube and accumulator) show a very non-linear behavior of the system.

- Line pressures plot shows the Δz of the line (20 meters).



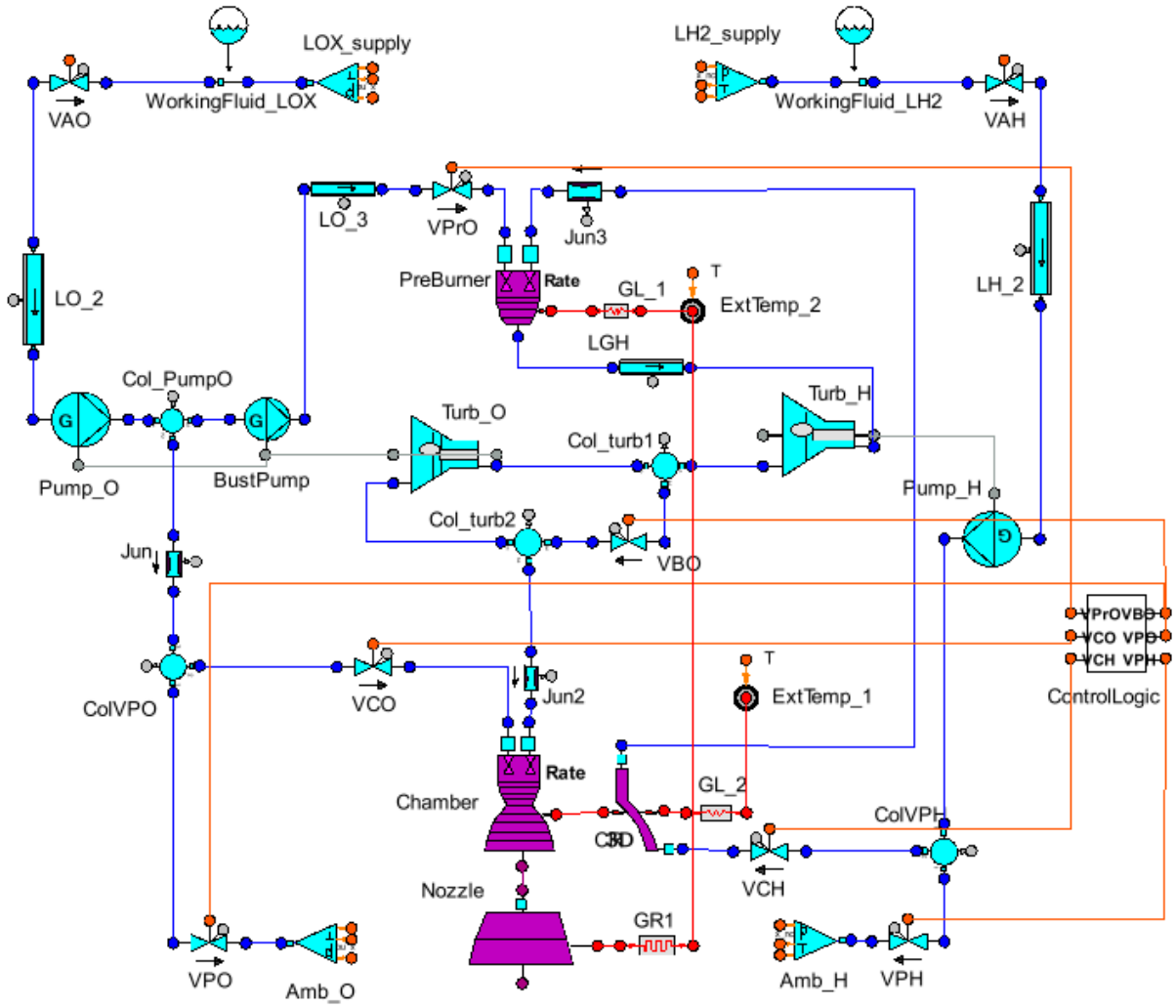
10.6 STAGED ENGINE CYCLE

Library:	ROCKET_EXAMPLES
Model Name:	StagedEngine_rate
Partition Name:	default
Experiment Name:	exp1

10.6.1 Model description

This model represents a Staged engine cycle type. Input data are fictitious values, the aim of this example being only to demonstrate the capabilities of ESPSS Libraries regarding this type of engine.

The default performance maps of the turbines (subsonic) have been used in this model.



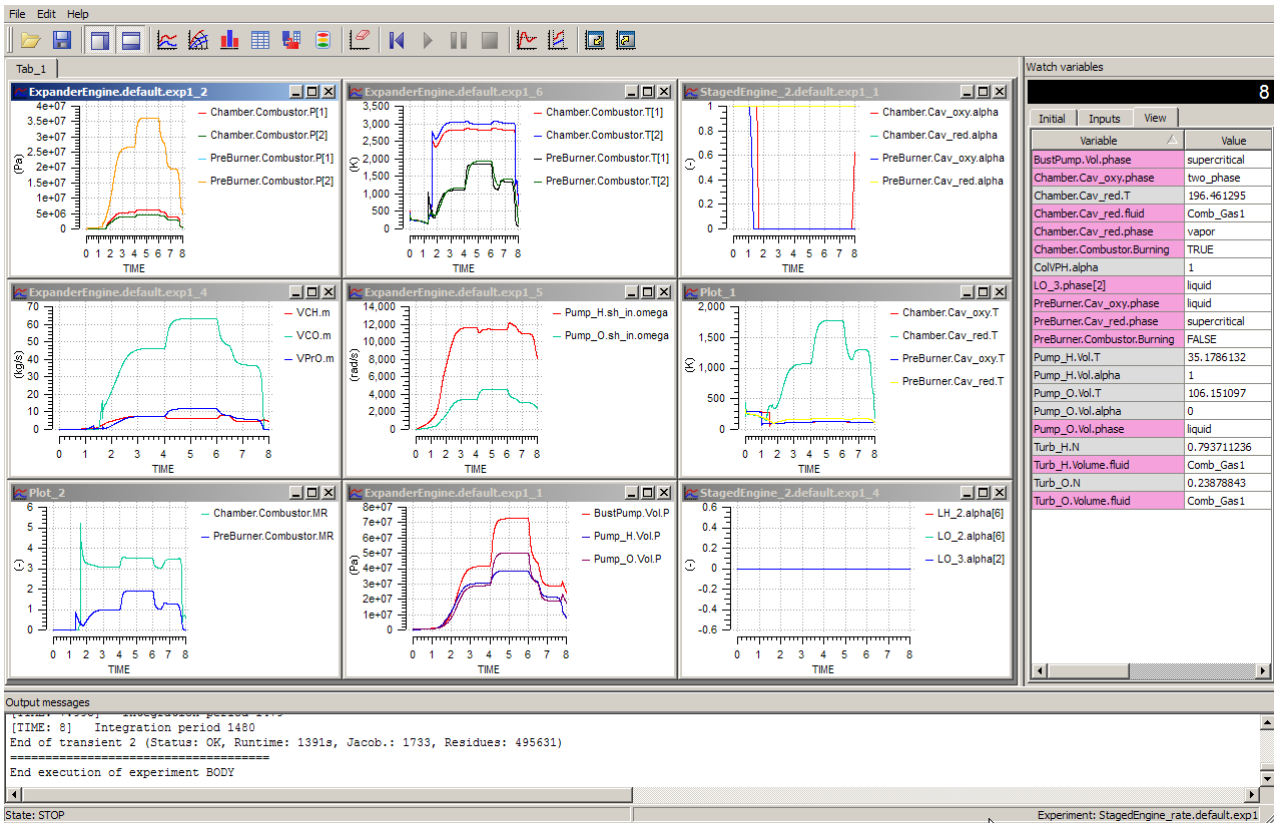
The boundaries, valve activation orders and some input geometrical data are given in the experiment file.

10.6.2 Results

Here below are the main plots obtained. The following is of note:

- In this model, the "combustor_rate" model is used. With the default burning and evaporation times, the chamber temperatures are less than the equilibrium ones.
- The model is able to simulate the start-up and the shut-down of the engine, including the relevant phenomena: cavities priming, liquid vaporization, pressure/temperature rise in the chamber during ignition, cavitations in the feeding lines, etc.
- Two steady working points have been simulated by controlling the VPrO valve. See the input data tables on the "ControlLogic" component.

As for the other engine models, the molar fraction of the products entering and exiting the combustion chambers can be found by exploring the results on the monitor tool.



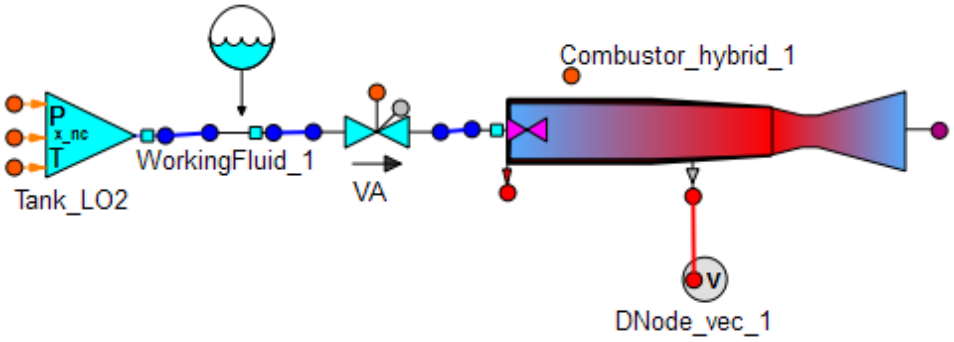
10.7 SOLID/HYBRID ENGINES

Library:	ROCKET_EXAMPLES
Model Name:	Test_hybrid
Partition Name:	default
Experiment Name:	exp1, exp2

10.7.1 Model description

This model represents an example of a Hybrid/Solid rocket engine with a main chamber containing a solid propellant, optionally fed with a liquid or gaseous oxidizer.

Input data are fictitious values, the aim of this example being only to demonstrate the capabilities of the ESPSS Libraries in this type of engines.



The system consists of a main thruster (aliased Combustor_hybrid_1 in this case) internally provided with a cylindrical combustion chamber and a nozzle, see §8.3.6.5, plus a liquid oxidizer injector system (closed while simulating a solid rocket engine).

The main dimensions are:

Cylinder: L = 1 m; D = 0.2 m *(Uniformly in 90% of the length; linear decreasing till throat)*
Grain propellant thickness = 0.03 m *(Uniformly in 90% of the length; linear to 0 at throat))*
Nozzle D_out/D_th ratio = 2.4

Grain factors are set to 5 (effective grain surface area with respect to a cylindrical one). See §8.3.6.4 for the input data description.

10.7.2 Results for a solid propellant case (exp2)

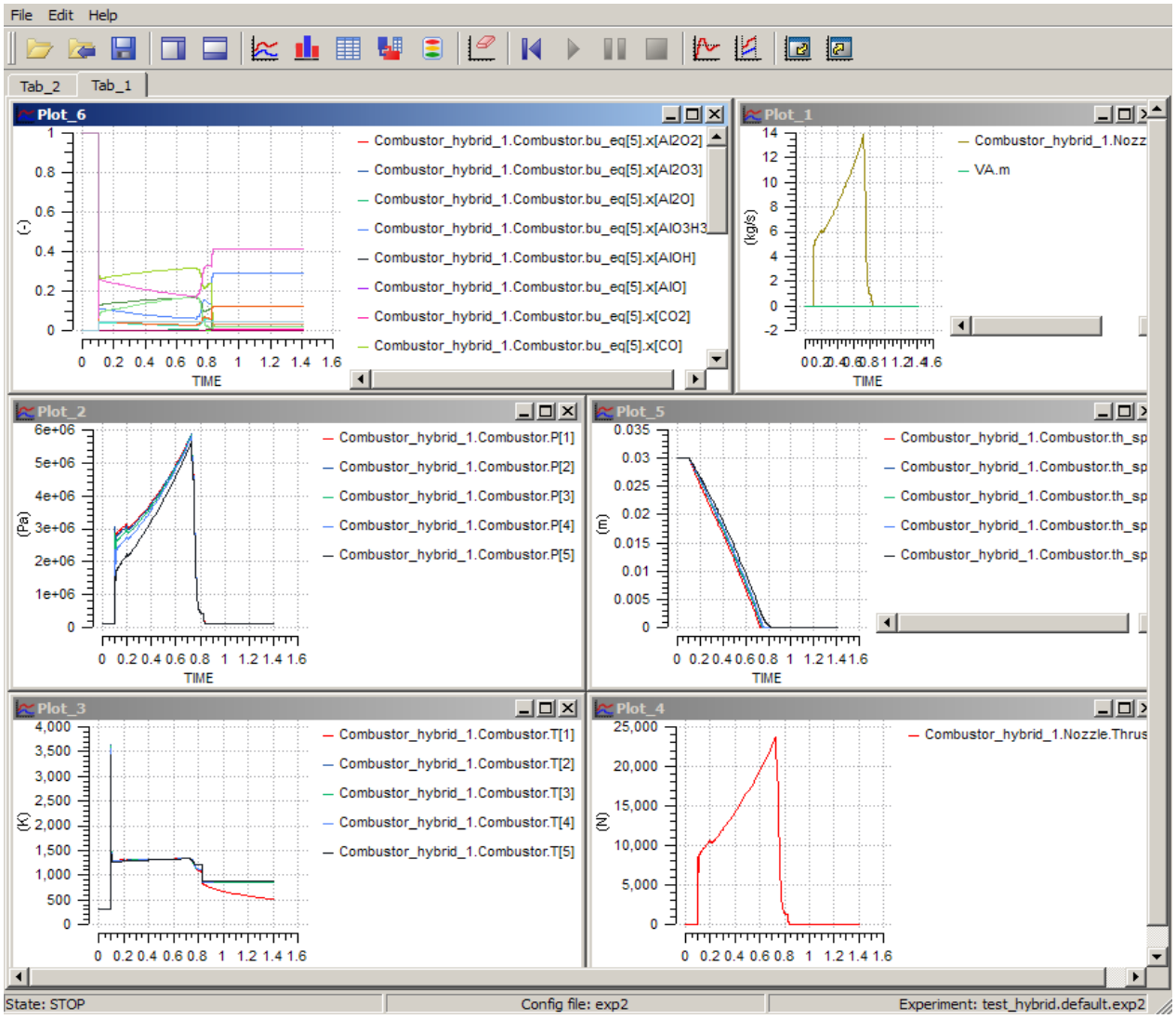
The boundaries (*closed LOX valve*), the time-dependant law to control the ignition, and some input data related to the combustion chamber and the solid propellant are given in the experiment file. *Note that in this case, the starter needs to be active (during 0.2 s) to help the release of rubber*:

```
EXPERIMENT exp2 ON test_hybrid.default
DECLS
...
BOUNDS
        Combustor_hybrid_1.Combustor.IgnitFlag = step(TIME,0.1)
        Combustor_hybrid_1.Combustor.starter_m = 0.2*(step(TIME,0.1) - step(TIME,0.2)) – starter
        ...
        -- Grain/gas exchange area factors :
        Combustor_hybrid_1.Combustor.f_s[01] = 5 ...
        Combustor_hybrid_1.Combustor.f_s[10] = 5

        VA.s_pos.signal[1] = 0. -- closed LOX valve
        Combustor_hybrid_1.np_out.P = 100000
        Combustor_hybrid_1.tp_inj.q[1] = 0
BODY
        Combustor_hybrid_1.Dt = 0.06      -- Small throat because NO LOX injection

        WorkingFluid_1.fluid_nc = PfGas_Air
        Combustor_hybrid_1.x_nco = 0

        --Solid propellant characteristics
        Combustor_hybrid_1.GasSolOption = stdSolid
        Combustor_hybrid_1.a_sp = 1e-005 -- 3e-3 -- constant regression rate (3 mm/s !!)
        Combustor_hybrid_1.b_sp = 0.45 -- 0 --
        Combustor_hybrid_1.tau_c = 1e-3
        Combustor_hybrid_1.tau_b = 1e-4
        Combustor_hybrid_1.Tsat_sp = 400

        -- Mass fractions of solid propellant constituents
        Combustor_hybrid_1.rubComp[HTPB] = 0.5
        Combustor_hybrid_1.rubComp[IPDI] = 0.
        Combustor_hybrid_1.rubComp[RubUsr] = 0.
        Combustor_hybrid_1.rubComp[KNO3_a] = 0
        Combustor_hybrid_1.rubComp[Al_cr] = 0.1
        Combustor_hybrid_1.rubComp[S_a] = 0
        Combustor_hybrid_1.rubComp[NH4NO3_IV] = 0
        Combustor_hybrid_1.rubComp[NH4CLO4_I] = 0.4
        ...
```

Some plots obtained are shown below. The model calculates the chemical composition of the products at the equilibrium temperature and pressure in correspondence with the grain composition, the consumption of solid propellant and the corresponding outlet mass flow for the given geometry.

Please note the following:

- The model calculates the P/T 1D axial distributions during the startup, quasi-steady conditions and shutdown
- A small pressure surge is detected at start up; then, the thrust increases because the grain/gas effective area is increasing with the grain consumption.
- The model is able to evaluate the influence of different grain compositions coupled with the shape of the grain and the combustor geometry.



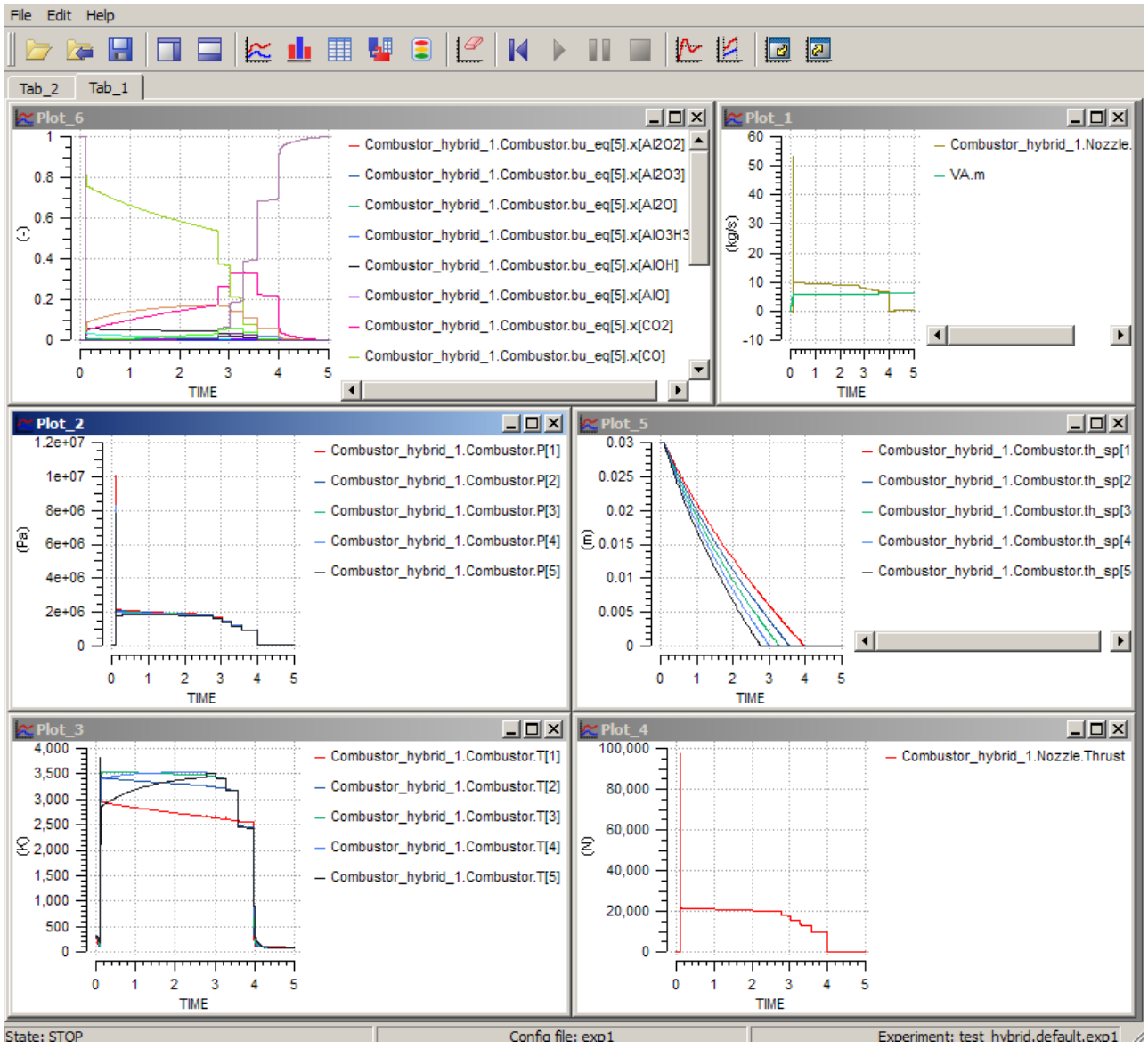
10.7.3 Results for a hybrid propellant case (exp1)

The boundaries, *the time-dependant law to control the LOX valve opening* and ignition, and some input data related to the combustion chamber and the solid propellant are given in the experiment file:

```
EXPERIMENT exp1 ON test_hybrid.default
DECLS
        TABLE_1D law_VAH = {{0., 0.1,0.2, 9.5, 10., 100} , {0, 1, 1, 1 , 0. , 0. } }
...
BOUNDS
        Combustor_hybrid_1.Combustor.IgnitFlag = step(TIME,0.1)
        ...
        -- Liquid oxygen injection conditions
        Tank_LO2.s_pres.signal[1] = 100e5
```

```
Tank_LO2.s_temp.signal[1] = 90
Tank_LO2.s_xNonCond.signal[1] = 0
VA.s_pos.signal[1] = 0.5*timeTableInterp(TIME,law_VAH)   -- open LOX valve

-- Grain/gas exchange area factors :
Combustor_hybrid_1.Combustor.f_s[01] = 5 ...
BODY
Combustor_hybrid_1.Dt = 0.10        -- Greater throat because LOX injection
VA.Ao = 1e-4                        -- LOX injection area

--Solid propellant characteristics
Combustor_hybrid_1.GasSolOption = stdHybrid
Combustor_hybrid_1.a_sp = 1e-5 ---- constant regression rate (3 mm/s !!)
Combustor_hybrid_1.b_sp = 0.9
...
-- Molar fractions of solid propellant constituents
Combustor_hybrid_1.rubComp[HTPB] = 0.9
Combustor_hybrid_1.rubComp[IPDI] = 0.
Combustor_hybrid_1.rubComp[RubUsr] = 0.
Combustor_hybrid_1.rubComp[Al_cr] = 0.1
...
```

Below some plots obtained.



It is appreciated the extinction of the solid propellant at different stages, the latest nodes being finishing before because they stayed at higher mass flow.

The high pressure peak (nearly an explosion) during the startup is of note; it is due to the fact that we suppose valid the rubber release regression law (proportional to the mass flow) during this first stage. Then, the temperature evolution is different for each node because of the local mixture ratio depends on the LOX consumption and rubber "vapor" released calculated by the code throughout the chamber.

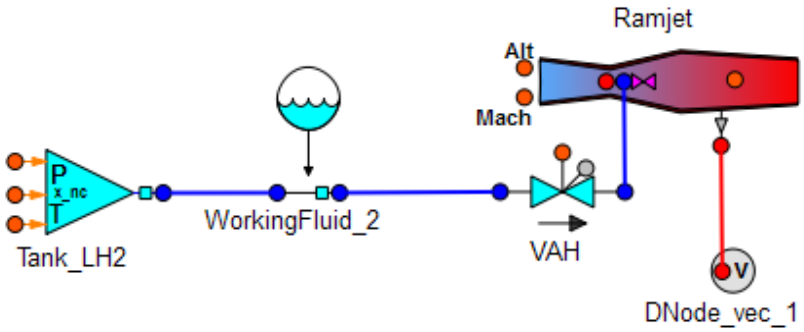
The model calculates the chemical composition of the products and the equilibrium temperature and pressure in correspondence with the grain composition and oxidizer injection conditions, the consumption of solid propellant and the corresponding outlet mass flow for the given geometry. Note that the simulation results in this case are strongly dependent on the oxidizer injection conditions (VA area and injection pressure) because the combusted gas composition depends on the mixture of the evaporated LOX, also simulated in the ESPSS Combustor_hybrid component.

10.8 RAMJET ENGINE (T- CCN -008)

Library:	ROCKET_EXAMPLES
Model Name:	Test_ramjet
Partition Name:	default
Experiment Name:	exp1

10.8.1 Model description

This model represents an example of a Ramjet engine with a main chamber taking the air from the atmosphere and fed with the LH2 fuel. Input data are fictitious values, the aim of this example being just to show the ESPSS Libraries capabilities with regard to this type of engine.

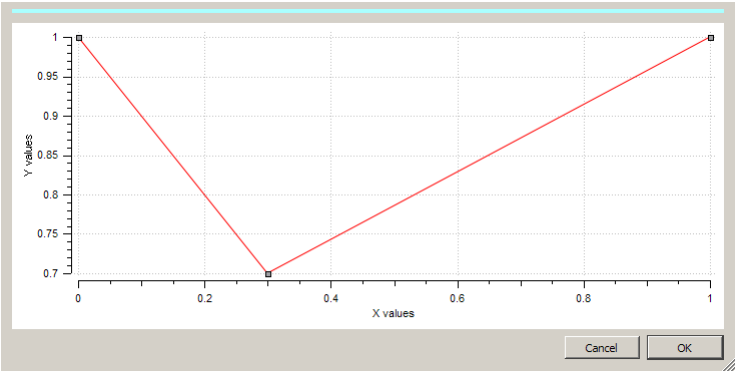


The system consists of a main thruster (aliased "Ramjet" in this case) internally provided with a non-ideal intake, an area varying supersonic combustion chamber, plus a liquid fuel injector system. The main dimensions are (see §8.3.7.4, §8.3.6.4 for the input data description):

- Combustor length: L = 1 m;
- Inlet cross section width and height= 0.5 m
- Number of nodes: **30**

Combustor height Profile →

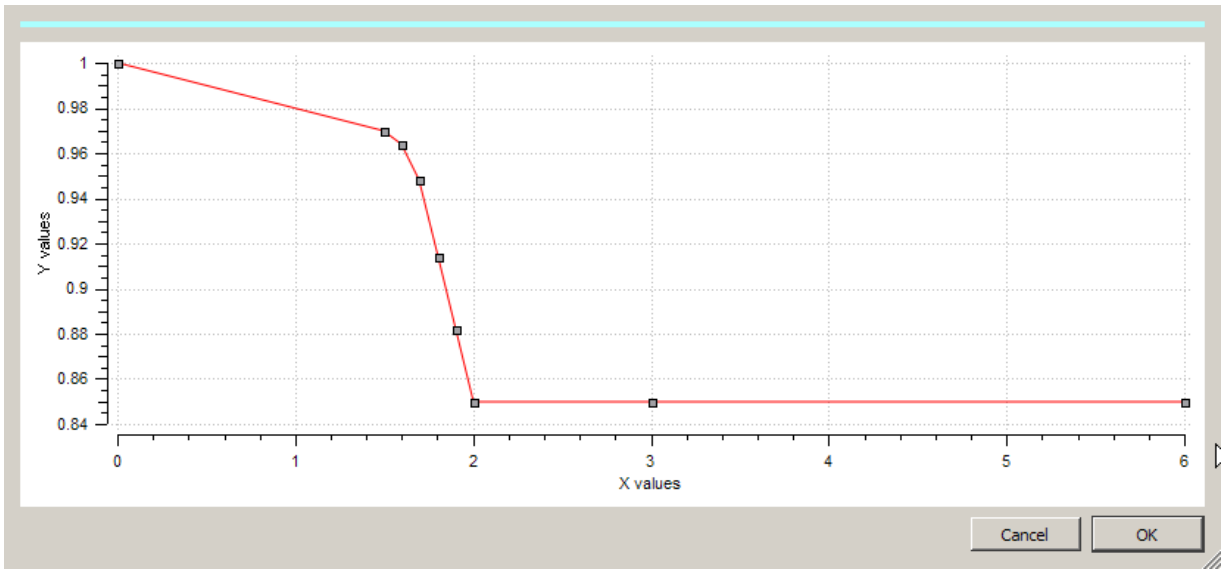
Constant width



Liquid LH2 injection conditions are:

Pressure = 20 bars; Temperature: 21 K
A_inj_red = 1E-4 m2
VAH.Ao = 1E-3 m2

The Total Pressure Recovery vs. flight Mach number at the intake is an "arbitrary" input data table as shown below. It should be noted that the inlet pressure of the combustor is coupled to the intake equations and to the fixed discharge pressure, P_o, the ambient pressure:



The boundaries, the time-dependant law to control the valve opening and the Flight conditions, are given in the experiment file:

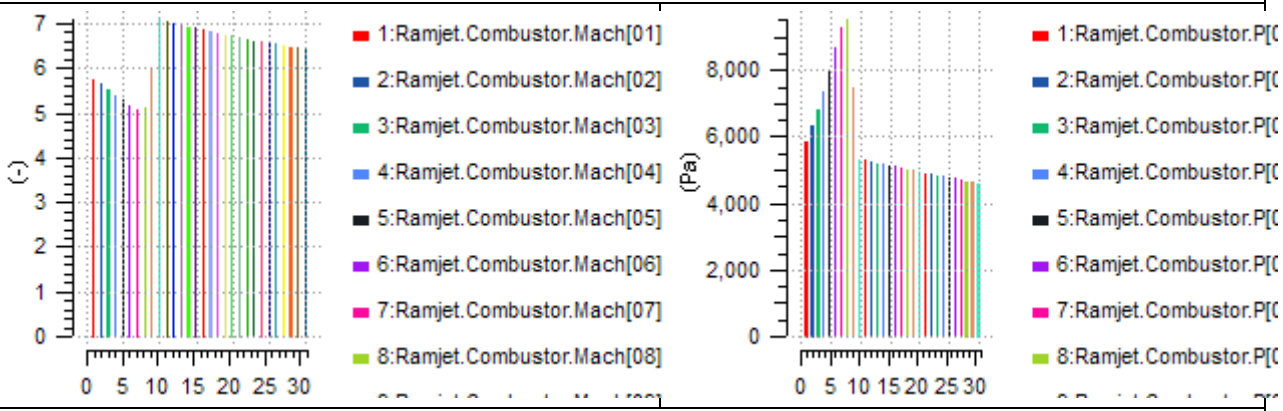
```
...
BOUNDS
        FLUID_FLOW_1D.Damp = 1
        FLUID_FLOW_1D.GRAV = 9.806
        FLUID_PROPERTIES.VDW_option = 0

        Ramjet.Combustor.IgnitFlag = step(TIME,15.01)      -- ignition time
        Ramjet.Combustor.starter_T = 1000
        Ramjet.Combustor.starter_m = 0
        Ramjet.tp_inj.q[1] = 0

        Tank_LH2.s_pres.signal[1] = 20e5
        Tank_LH2.s_temp.signal[1] = 21
        Tank_LH2.s_xNonCond.signal[1] = 0
        VAH.s_pos.signal[1] = 0.1*step(TIME,10.01)         -- LH2 injection time

        Ramjet.s_alt.signal[1] = 20000                     -- Altitude
        Ramjet.s_mach.signal[1] = 6--*min(1,TIME*10)       -- Flight Mach number
        ...
        Ramjet.Combustor.A_rel_red[15] = 0.2
        Ramjet.Combustor.A_rel_red[16] = 0.2
        Ramjet.Combustor.A_rel_red[17] = 0.2
...
```

LH2 injection nodes are from node 13 to 17 (30 nodes in total uniformly distributed):

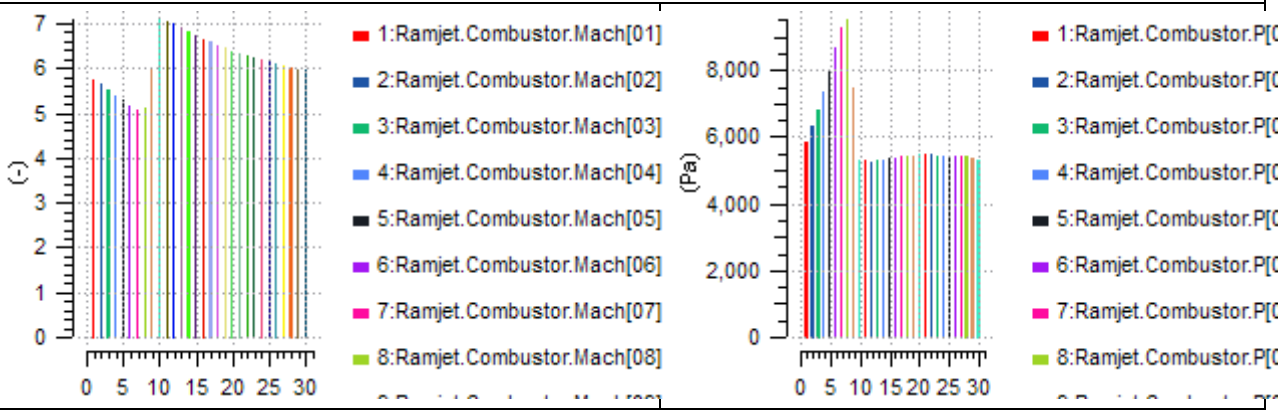
10.8.2 Results

Below are the main plots obtained before and after LH2 injection, and after stabilized combustion at a flight Mach number of 6.

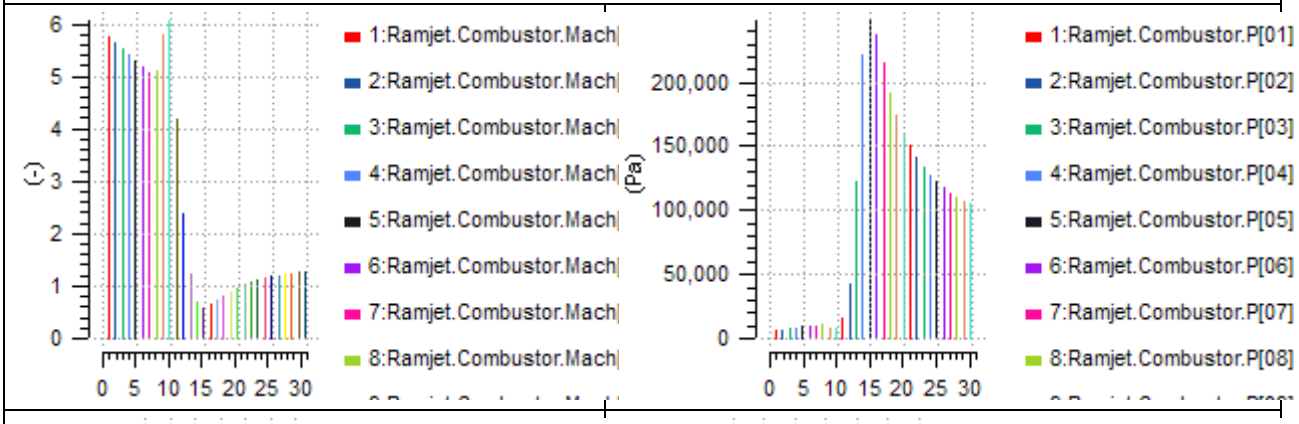
Flight Conditions: M =6, Alt = 20000 m (before LH2 injection)

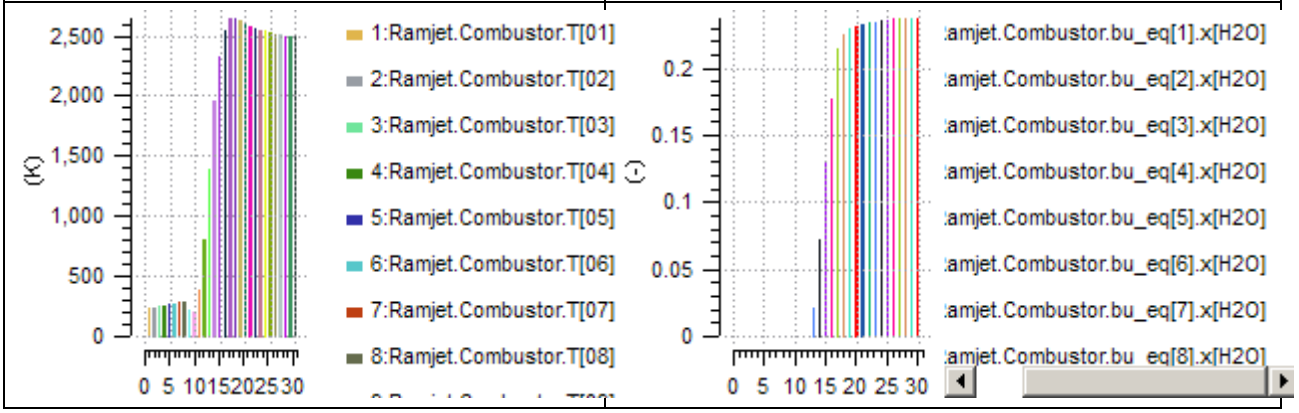


(after LH2 injection, time = 10 s)



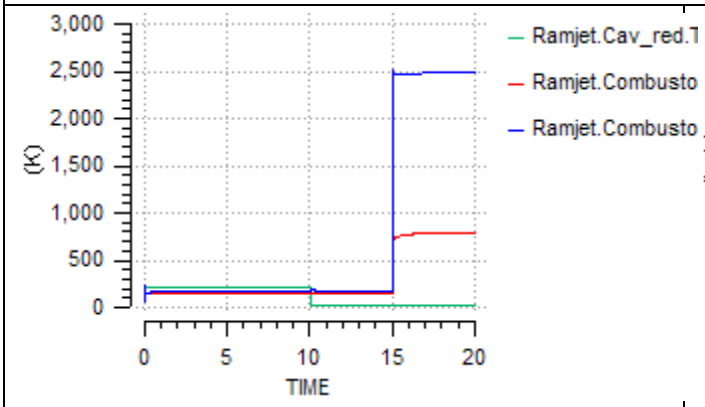
(after combustion, time = 15 s)



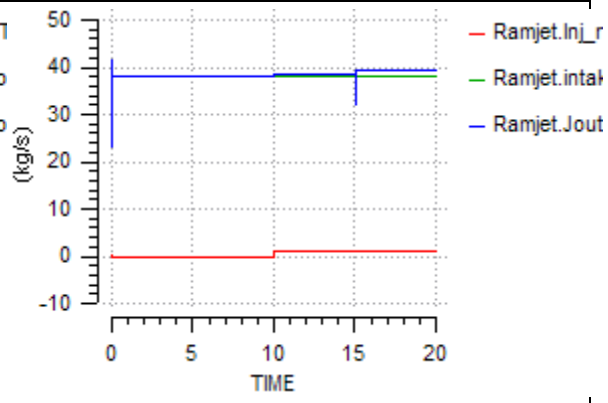


The time histories of the mass flow, thrust and other magnitudes in response the different engine conditions are presented below:

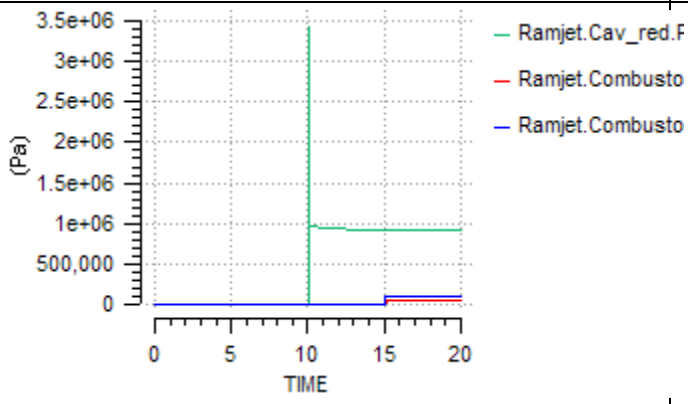
Flight Conditions: M =6, Alt = 20000 m
(Time histories)



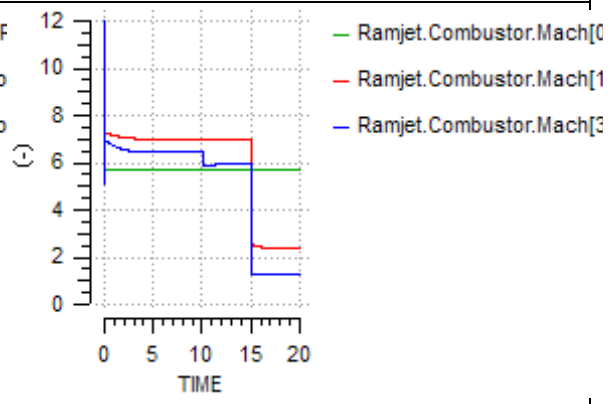
Temperatures



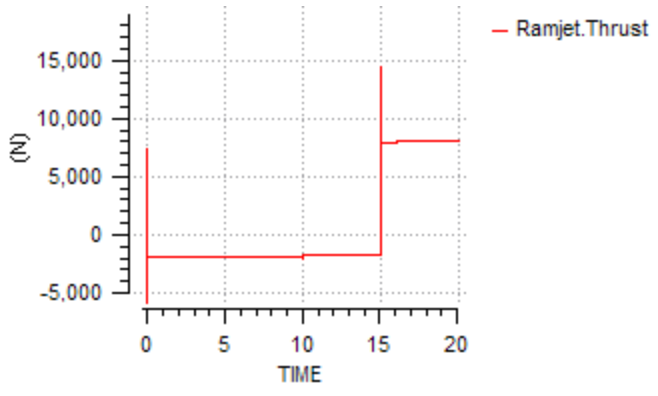
Mass flows



Pressures



Mach numbers



Thrust

It is interesting to observe how the ESPSS model can simulate the influences of friction, geometry, flight & LH2 injection conditions, etc. on the thrust, combustion efficiency and Mach evolutions.

Of course, with no combustion, the thrust is negative. It becomes positive only with a good coupling of a supersonic combustion within a "small" range of LH2 injection. It can be tested for example that by increasing the LH2 flow rate over a limit, the combustor becomes subsonic with negative thrust.

11. STEADY LIBRARY

11.1 OVERVIEW

The *STEADY* library contains a complete set of ready to use components (combustors, nozzles, turbines, pumps, cooling-jackets, pipes and valves) for calculating the steady performances of liquid rocket engine cycles under design and analysis (off-design) conditions. These two kinds of analyses performed by the STEADY library are characterized by the following:

A) Design Models:

Design models contain the design conditions (normally non-dimensional engine performance) as part of the components' input data. Experiments built with the default *partition* (automatic ordering of the whole cycle equations made by EcosimPro) will calculate the engine "size" and the nominal operating conditions. This sizing comprises the turbo-machinery and valves. Complete engine size (as the nozzle throat diameter) can be done imposing particular mission requirements as shown in paragraph 11.4

Typically, the design conditions are as follows (see Table 11-1):

- Pump and turbine efficiencies, pressure ratios, characteristic speed, …
- Combustor pressure and Mixture ratio
- Relative or absolute pressure drops of some particular valves. Pressure drop in bypass valves will be calculated instead to adjust the cycle performances

Calculated operating conditions are the cycle mass flows, valve areas, pressures and temperatures as well as the turbo machinery powers and axial speeds.

B) Analysis Models

Analysis (off-design) models do not contain the *design* conditions as part of the components' input data, but the geometry (engine size) and the nominal operating conditions, which necessarily have some common variables with the design conditions, such as the nominal efficiencies, turbines pressure ratios, nominal pump speeds, etc. Off design conditions will be calculated accordingly with the current boundary conditions and *using the performance maps of turbines and pumps*.

Design and analysis models are different because of the different input data they need. Nevertheless, the *STEADY* library has been prepared to run design and off-design calculations using a single model by means of the *design partition,* which is an EcosimPro capability to transform some data (nominal mass flows and axial speeds, valve areas,…) to unknowns adding the design conditions (efficiencies, chamber pressure, mixture ratios, etc.) as boundaries (see §11.1.2.3 for more details). This method (use of a design partition) has the following advantages:

- A single model serves for design and analysis calculations under two different partitions (design and default respectively)
- The nominal operating conditions and engine size (previously calculated in a design partition) can be loaded in an analysis (off design) experiment by means of the RESTORE command which enters the design partition results as input data.

Other important features of the STEADY library are the following:

- STEADY components directly calculate performance with basically the same accuracy as in the transient libraries, but under steady conditions. The fluid properties (two-phase flow) functions and the chemical equilibrium functions for combustion are taken from the FLUID_PROPERTIES & COMB_CHAMBER libraries, as they serve the same purpose needed for the steady state.

  The only restriction is that mixtures of non-condensable gases and a liquid in pipes are not allowed; no compatibility issues arise, however, because the use of non-condensable gases is normally required for the startup and shut-down process.

- *STEADY cycle models are solved implicitly*. The system of equations incorporates most of the components simultaneously because the time derivatives have been excluded in the formulation.

- Ports have been rewritten for the Steady State library permitting multiple connections of components (splitting and collection of flows).
- The STEADY library can be connected to other ESPSS libraries. The interface component assumes some limitations, such as no reverse flow at the interface and no flow of non-condensable gases.

Several test cases have been studied to validate the Steady State library. These tests range from simple flow lines with few components to full rocket engine cycles. Their results have been carefully assessed in the STEADY_EXAMPLES library and partially documented in *RD-5.*

11.1.1 Components classification

The components of the STEADY Library are listed in Figure 10-1 below.

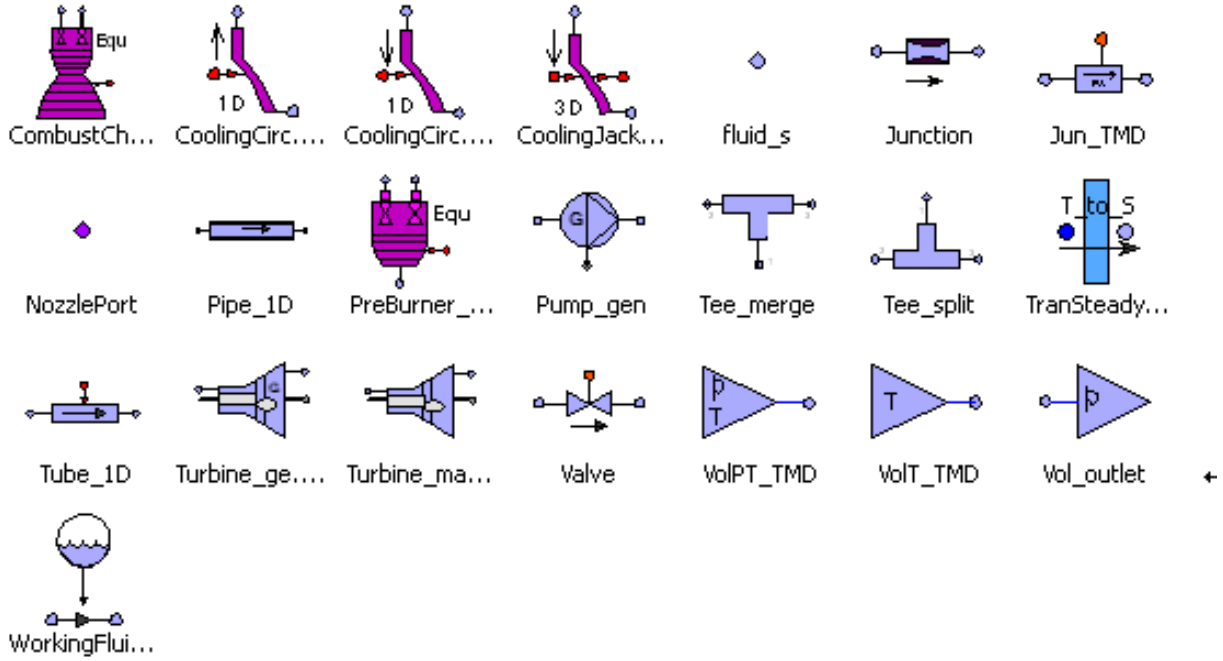


Figure 11-1 STEADY palette of symbols

The existing components have **no** capacitive / resistive type: in contrast to the ESPSS transient libraries, STEADY components can be linked to each other without respecting any particular rules. The tranSteady_IF component is designed to permit the connection of STEADY components with transient components of the ESPSS libraries for mission analysis.

Another important property of the STEADY components is that connections are not single type (*Tee components can be avoided*). Multiple connections will simulate ideal splitter/collection of flows:

- Outlet ports can be directly split into several inlet ports
- Inlet ports can collect several outlet ports

11.1.2 Building a Model

It is recommended to start by using the STEADY_EXAMPLES library and executing the different examples contained in the library while reading the User Manual.

11.1.2.1 Geometry arrangement

As in the COMB_CHAMBER library, *Preburner, CombustChamberNozzle* and *Nozzle* components have been modeled using a 1D discretization. The same procedures and rules concerning the definition of chamber, nozzle & cooling jacket profiles (see §8.1.2.2) apply here.

11.1.2.2 Design models

Let's consider the "expander_cycle_design" case of the STEADY_EXAMPLES library:

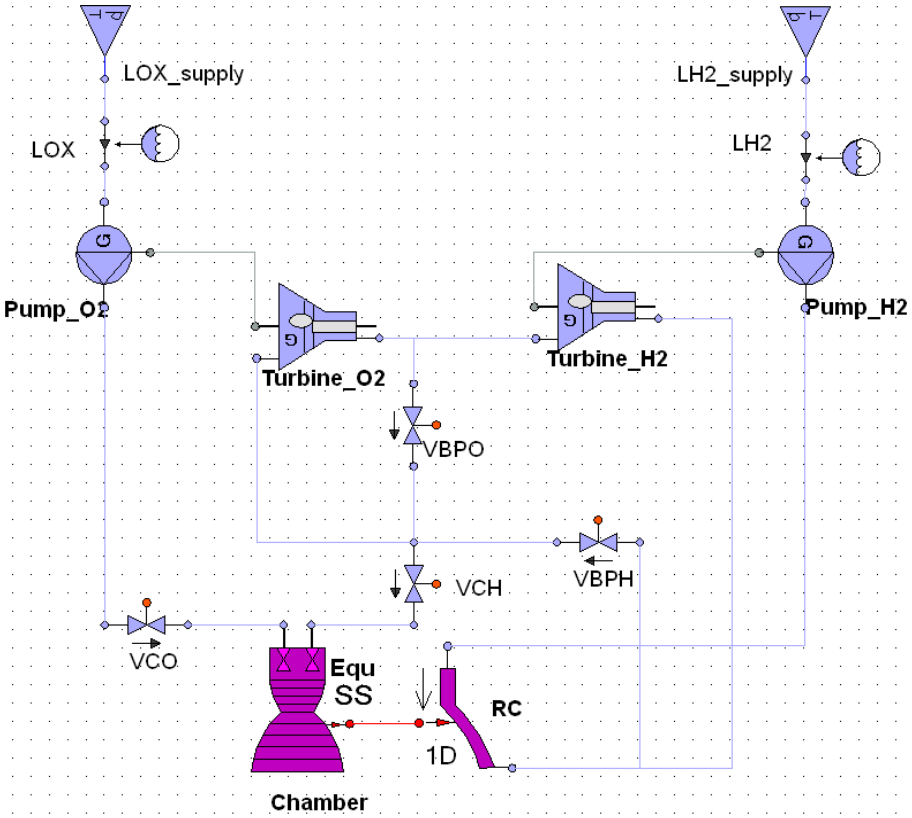


Figure 11-2 Expander cycle example

All the components have activated design mode parameters: "type" = Design, known_PI_tt, known_Ns, etc., depending on the respective component, where "type" is a construction parameter.

Design models contain the design conditions as part of the input data of the components: In the present case, imposed/calculated design parameters are as follows (see the input data description in §11.3):

Component	Imposed parameters (design conditions)			Calculated Operating Values	
Turbines	PI_tt_o	Eta_o (effic.)	See note 5	Rpm, torque, …	flow, pressures temperatures, …
Pumps	Ns (spec. speed)	Eta_o (effic.)			
Chamber	Pc & MR (mixture ratio)	dP_per_oxy, dP_per_red	See note 2	Inj_oxy.A Inj_red.A	flow, pressures temperatures, … (see note 1)
Cooling Jacket	dP_design	Tw_ design	See note 3	t_ch_cal, th_i_cal	
Valves	VCO.dP_design	VCH.dP_design	See note 4	Valve areas (*.A)	

Table 11-1. Imposed/calculated parameters in Design Mode

Notes

1. Calculated variables are internal to the respective component and include flows (m), power, torque, axial speeds, pressures and temperatures
2. Inj_oxy.A, Inj_oxy.A are the injector areas

3. "t_ch_cal" and "th_i_cal" are the channel width and the internal wall thickness at throat position. For this component, choose Off design mode for fixed geometry and calculated dP, dT
4. Not all the junction/valve pressure drops are imposed, only the ones with "dp_type= UserGiven". The user should determine which valves are calculated "FromPorts", see §11.2.2 (normally, bypass valves where the pressure bounds are determined by the imposed turbine pressure ratio)
5. Besides imposing efficiency for turbines and pumps, the user should determine which turbines (pumps) impose PI_tt (NS) values depending on the cycle topology (see §11.2.2).

Default partition experiments in Design Models

Default partition[2] experiments automatically calculate the operating values (see Table 11-1) of a cycle for some design conditions. To do this, follow these two steps:

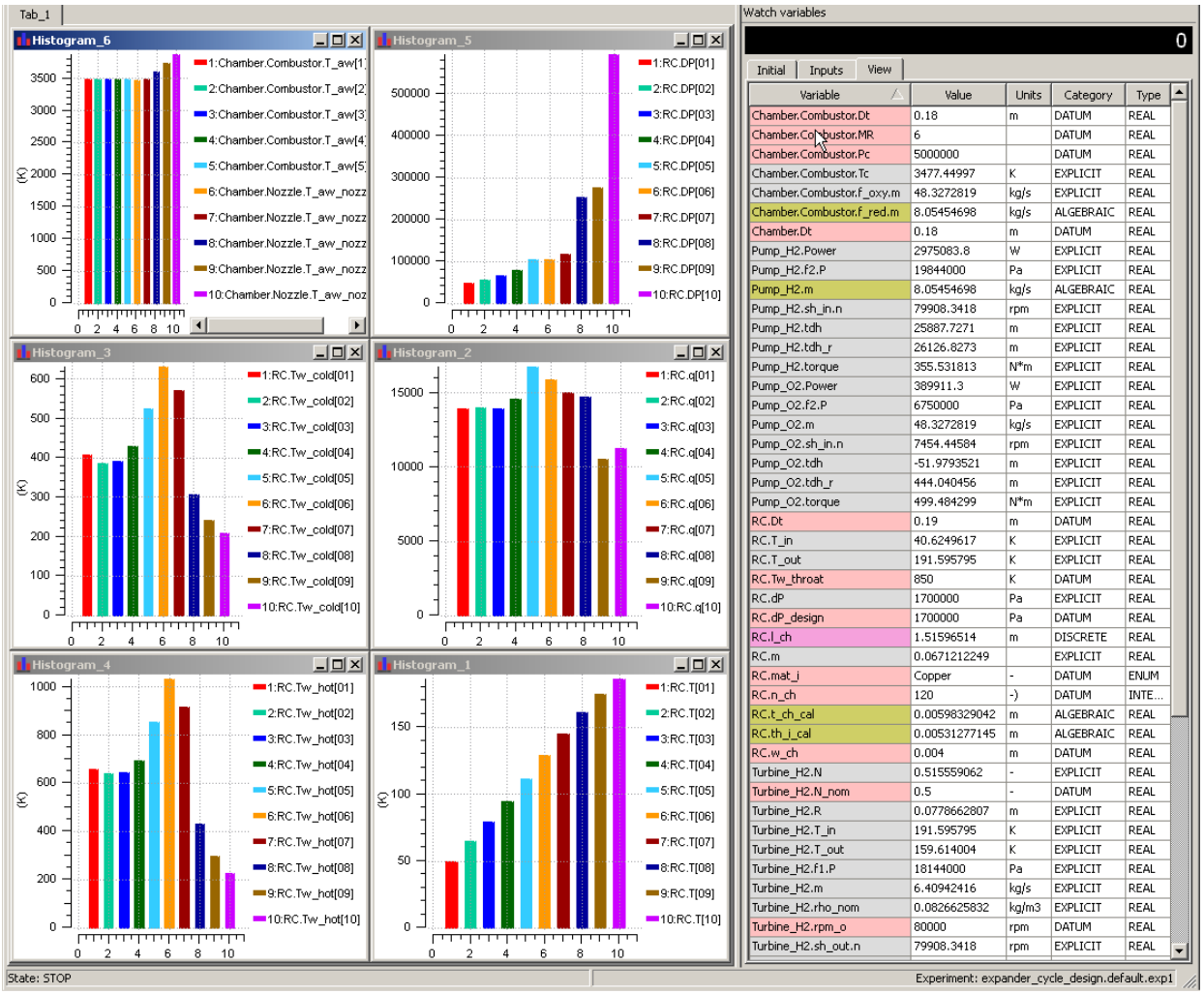
- Press the "Generate model" button on the EcosimPro GUI. This compiles the graphical model and generates the default partition. (Note that EcosimPro will internally find an automatic ordering of all the model equations, searching for the appropriate algebraic variables to be solved).
- Generate the default experiment. *The INIT block of the experiment can be normally deleted* because the initial input data entered in the components is enough (a few sets of initial values of pressures and mass flows). *The component INT block already provides the necessary initialization.*

Below is the experiment used for the "Expander" cycle model. Note that global variables such as "m_fu" are used to initialize unknowns in several components:

```
EXPERIMENT exp1 ON expander_cycle_design.default
INIT

BOUNDS
        STEADY.dp_lam = 3000
        VBPH.s_pos.signal[1] = 1
        VBPO.s_pos.signal[1] = 1
        VCH.s_pos.signal[1] = 1.
        VCO.s_pos.signal[1] = 1.

BODY
        Pc = 50e5
        MR = 6
        m_fu = 8      -- to init iterative calculations
        RC.t_ch = 0.0006 – initial value for algebr. variable
        RC.th_i = 0.0004 –     ".
        …
        -- Below, possible redefinitions of the input data values (Input data control)
        Turbine_H2.PI_tt_o = 2.4
        Turbine_O2.PI_tt_o = 1.2
        Pump_O2.Ns = 10
        VCO.dP_design = 10e5
        VCH.dP_design = 5.5e5

        TIME = 0
        DEBUG_LEVEL = 3
        TOLTYPE= ABSTOL
        STEADY()
    END EXPERIMENT
```

The calculated values are plotted out below:

[2] In some cases, the algebraic variables proposed by the default partition may need to be modified (partition edition), adding or suppressing particular variables (mass flows for example) as algebraic.



Channel dP is nearly exponential with the axial length because the channels have constant flow areas. A more homogeneous pressure drop can be obtained by modifying the dimensionless channel widths and heights vs. axial position (see the CoolingCircuit input data wc_vs_L and tc_vs_L).

11.1.2.3 Analysis Models

The analysis model for the previous example is the "expander_cycle_offDesign" of the STEADY_EXAMPLES library. It represents the same cycle as before but with all the design mode parameters of the respective components set as Off_Design mode (type = Off_Design, Off_D_pump, Off_D_turb, etc.).

Analysis models need a different model setup than the design models as the design mode parameters must be fixed before a partition can be generated, and *because the analysis model is using performance maps of turbines and pumps*.

Design Partition experiments in Analysis Models

A *design* partition in an *analysis* model is an alternative way of calculating the *nominal* operating point and "size" (required turbo-machinery power, valve areas …) of a cycle. This method (using a design partition) has the following advantages: a) the same model serves for Design and Analysis calculations because in both cases design mode parameters are set to "Off Design" value, b) the nominal operating conditions (previously calculated in a design partition) can be automatically loaded in an experiment by means of the RESTORE command.

To calculate the nominal operational point and size of a cycle, follow these steps:

- Define a *design* partition in the Analysis model. This is not very complicated and in the end this exercise should always be done to get a good idea of the variables being calculated when designing the cycle. The rules that should be *generally* followed generating the design partition (use of the Design Partition wizard) are:

  a) Set the following Data (*nominal operational point and engine sizing*) as unknowns:

  1 Apply the filter "*.A*" to choose any Junction, Valve and injector area as an unknown variable
  2 Apply the filters "*.m_o", "*.rpm_o" and "dP_o" to choose operational mass flow, nominal speed and pressure rise of Pumps as unknown variables
  3 Apply the filter "*.rpm_o" to choose the operational speed of Turbines as an unknown variable
  4 Apply the filter "*.Ain" to choose the characteristic inter-blade flow area of Turbine as an unknown variable (only for generic Turbines)
  5 Apply the filter "*.N_nom" to choose the characteristic speed of Turbine as an unknown variable (only for map Turbines)

  Note: Data to be converted to unknowns must not have been re-defined with model variables

  b) Set the following variables (*design conditions*) as boundaries ("able to be selected" category):

  1 Apply the filter *.dP to choose Junctions, Valves and injectors pressure drops as known variables (choose only those that in a design model would be "UserGiven" type, see §11.2.2)
  2 Apply the filter *.MR to choose imposed combustor mixture ratios (MR)
  3 Apply the filter *.Pc to choose imposed combustor chamber pressures
  4 Apply the filter *.eta_calc to choose imposed efficiency in turbines & pumps
  5 Apply the filter *.PI_tt_calc to choose imposed PI_tt in turbines (choose only the turbines that in Design mode would be "known_PI_tt" type, see §11.2.2)
  6 Apply the filter *.Ns_calc to choose imposed Ns in pumps (choose only the Pumps that in Design mode would be "Know_Ns" type, see §11.2.2)
  7 Apply the filter *.n to choose nominal axial speed in pumps
  8 Apply the filter *.signal[* to choose imposed valve positions

  Note: "*_calc" are local model variables with the actual value of the efficiency and pressure ratio depending on the performance map. Setting these variables to a value is equivalent to setting the operational point in a particular point of the performance map.

  Table 10-2 below summarizes these rules for turbines and pumps:

Turbines

Input to be calculated	Boundary
rpm_o (Nominal speed)	eta_calc = eta
Ain (characteristic inter-blade flow area, (1))	PI_tt_calc = PI_tt_o

Pumps

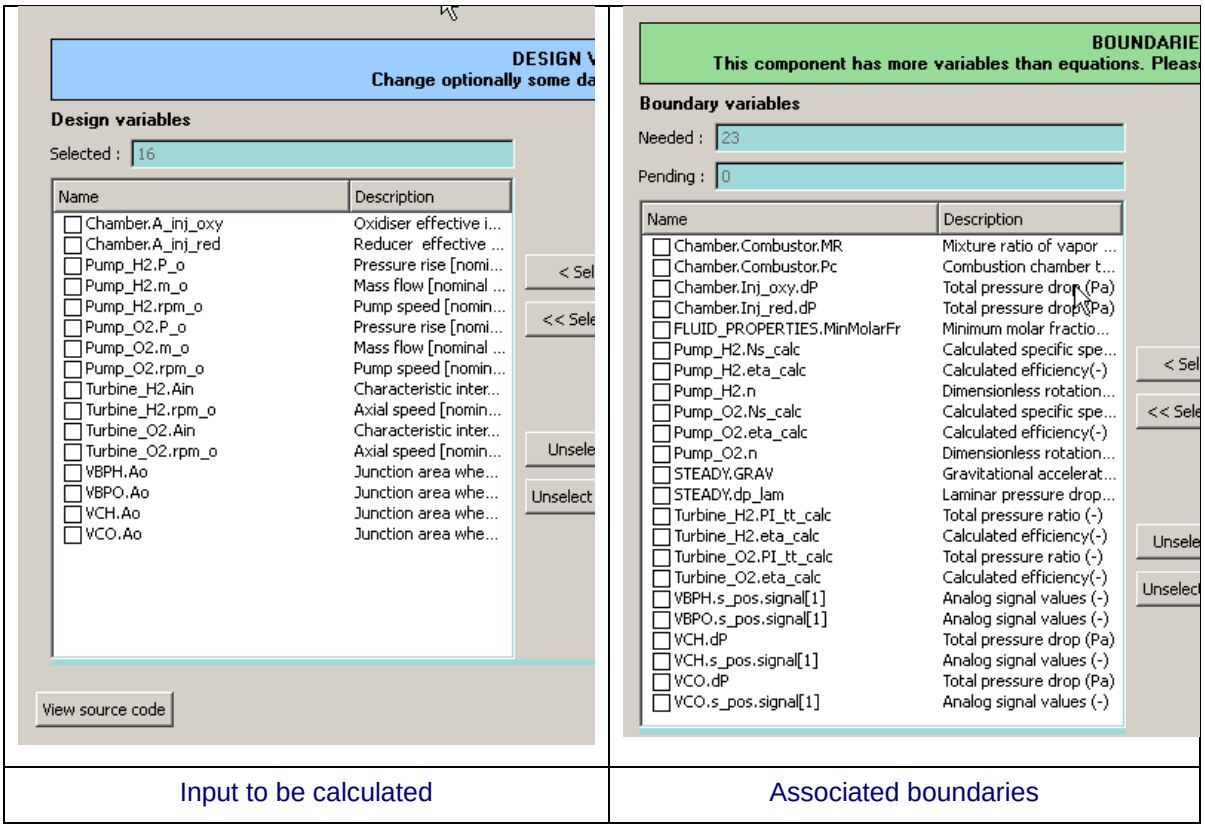
Input to be calculated	Boundary
dP_o (pressure rise)	eta_calc = eta
m_o (Nominal mass flow)	Ns_calc = Ns
rpm_o (Nominal speed)	n =1 (2)

Table 11-2. Imposed/calculated parameters in Design Partitions

Notes:

1. "Ain" is used for generic turbines. In case of user defined maps, this parameter is replaced by the N_nom parameter
2. "n" is a local pump variable defined as the ratio of the current axial speed and the nominal one
3. Ns is the pump characteristic speed; see the input data description

Below is a printout of the design partition wizards for the present case:



Input to be calculated	Associated boundaries

c) *As general rule*, select the proposed algebraic variables by the wizard breaking the nonlinear boxes.

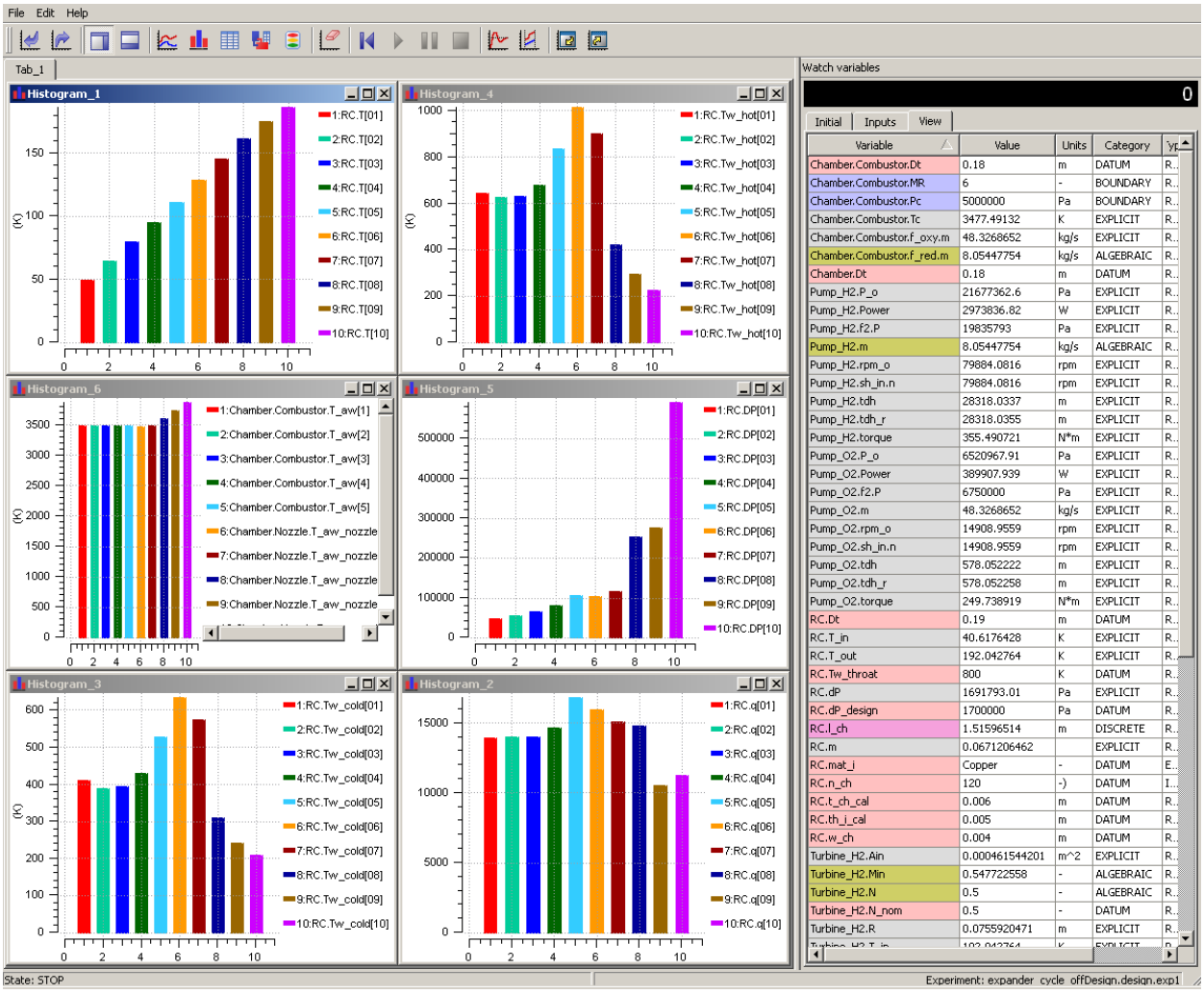
- Generate the default experiment. The INIT block of the experiment can normally be deleted because the initialization entered in the components is enough (a few sets of pressures and mass flows). *The components' INT block normally already provides the necessary initialization.*

  Below is the experiment used for the Expander" cycle model:

```
EXPERIMENT exp1 ON expander_cycle_offDesign.design
INIT
    -- initial values for algebraics

BOUNDS
    -- Set equations for boundaries: boundVar = f(TIME;...)
    Chamber.Combustor.MR = MR
    Chamber.Combustor.Pc = Pc
    Chamber.Inj_oxy.dP = 7.5e5
    Chamber.Inj_red.dP = 7.5e5
    VCH.dP = 5.5e5
    VCO.dP = 10e5

    Pump_H2.Ns_calc = Pump_H2.Ns
    Pump_H2.eta_calc = 0.7
    Pump_H2.n = 1
    Pump_O2.Ns_calc = Pump_O2.Ns
    Pump_O2.eta_calc = 0.7
    Pump_O2.n = 1
    Turbine_H2.PI_tt_calc = Turbine_H2.PI_tt_o
    Turbine_H2.eta_calc = 0.7
    Turbine_O2.PI_tt_calc = Turbine_O2.PI_tt_o
    Turbine_O2.eta_calc = 0.7
    VBPH.s_pos.signal[1] = 1
```

```
        VBPO.s_pos.signal[1] = 1
        VCH.s_pos.signal[1] = 1
        VCO.s_pos.signal[1] = 1
        STEADY.GRAV = 9.806
        STEADY.dp_lam = 3000

    BODY
        -- Below, possible redefinitions of the input data values (Input data control)
        Pc = 50e5
        MR = 6
        RC.t_ch = 0.006 -.
        RC.th_i = 0.005 -.
        ...
        TIME = 0
        DEBUG_LEVEL = 3
        TOLTYPE= ABSTOL -- FRACTOL --
        STEADY()
        SAVE_STATE("design")
    END EXPERIMENT
```

Note the SAVE_STATE command after the STEADY calculation. This command will produce an ASCII file in the experiment folder with the value of the model variables. The plot of calculated values is presented below:



Calculated operational values are saved with the same name as the one used for the input data: for example, valve areas are saved in the *.Ao variables (input data), calculated nominal axial speeds are

saved in *.rpm_o variables, etc. The local variables *.A, sh_in.n, etc. will have the same values as the nominal ones for the design partition.

The following paragraphs describe the results from the same expander cycle but working with methane instead of H2. The exercise, which would normally involve a drastic change for a typical simulation tool, is really simple with the ESPSS libraries. As long as the same model and partition are used as for the LH2/LOX case, the following changes are made in the experiment file:

- First of all, change the reducer working fluid to Real_CH4, the inlet/initial temperatures and the chamber mixture ratio to 3 with the same chamber pressure:

```
LH2.fluid = Real_CH4
LH2_supply.P_o = 3e5
LH2_supply.T_o = 110   -- (methane liquid conditions)
Pump_H2.T_o = 110
RC.T_o = 150

Pc = 50e5
MR = 3
```

  When we try to simulate this case, the model does not converge at all. What happens, we suspect, is that now the PI_tt of the turbines may be inappropriate. Results seem to be more suitable imposing higher PI_tt values for the O2 turbine (from 1.2 to 1.5), but this is not enough to converge: mass flow circulating through the bypasses is negative.

- However, if we then decrease the pressure in the chamber the model begins to converge. It appears that for the disposition of turbines and pumps of Figure 11-2, this cycle is limited to maximum of about 45 bars in the combustor using CH4. Below are the results using CH4 at 40 bar:

Default partition experiments in analysis models

Default partition experiments in an analysis model will calculate the off-design operation of a cycle. As mentioned before, a design partition should have been previously generated over the same model. Then, the calculated operational point in the design partition can be loaded in an analysis experiment by means of the RESTORE command which enters the design partition results as input data. More precisely, the following rules apply to generating experiments for off-design analysis:

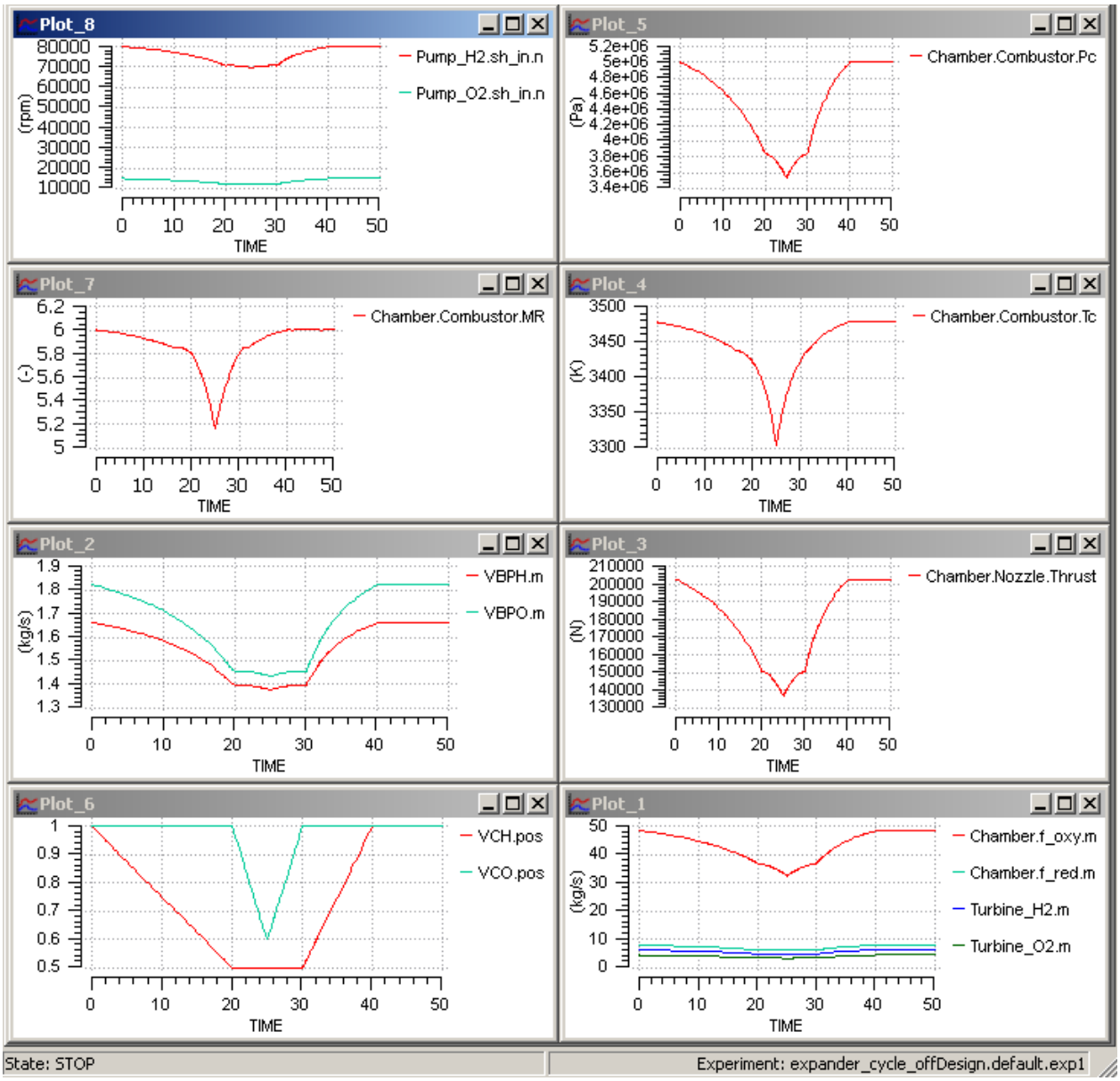
- Generate the *default* partition in the Analysis model. EcosimPro will automatically generate the default partition and will internally find the ordering of the whole model equations, searching for the appropriate algebraic variables to be solved
- Generate default experiment. The INIT block of the experiment can be deleted. The initial input data entered in the components (a few sets of pressures and mass flows) is normally enough.

  Below is the experiment used to calculate the "Expander" cycle off-design behavior opening and closing the VCH/VCO valves:

```
EXPERIMENT exp1 ON expander_cycle_offDesign.default

DECLS
    -- opening / closing law for VCH valve
    TABLE_1D law1 = { {0, 20, 30, 40, 50} ,    {1,0.5,0.5, 1, 1} }

    -- opening / closing law for VCO valve
    TABLE_1D law2 = { {0,20, 25, 30, 50} ,    {1, 1,0.6, 1, 1} }

INIT
    -- initial values for algebraics (not needed)

BOUNDS
    -- Set equations for boundaries: boundVar = f(TIME;...)
    STEADY.GRAV = 9.806
    STEADY.dp_lam = 3000
    VBPH.s_pos.signal[1] = 1
    VBPO.s_pos.signal[1] = 1
    VCH.s_pos.signal[1] = timeTableInterp(TIME,law1)
    VCO.s_pos.signal[1] = timeTableInterp(TIME,law2)

BODY
    -- restore a previous state
    SET_INIT_ACTIVE(FALSE)--deactivateINIT blocks because the complete sate will be read from a file
    RESTORE_STATE("@STEADY_EXAMPLES@/experiments/expander_cycle_off+design.design/exp1/design")

    TSTOP = 20
    TIME = 0
    CINT = 1

    REPORT_MODE = IS_STEP
    REL_ERROR = 1e-4
    ABS_ERROR = 1e-4
    INTEG()
END EXPERIMENT
```

Note how the design parameters have been loaded using the RESTORE_STATE command from a design partition experiment. Remember that analysis studies require that the design conditions be entered as input data. The command RESTORE_STATE will produce a complete loading of the input data from a design partition experiment.

The calculated transient is:



11.1.2.4 Numerical limitations and advises

The STEADY library directly calculates the Design and the Off-design operating conditions of a complete cycle. This involves solving complex algebraic loops. While in transient model the drawbacks are the numerical difficulties simulating *startup* and shutdown processes (very stiff equations under transient conditions), now in steady conditions, the difficulties are those to find the solution of a high non-linear implicit equation system *starting* from some "arbitrary" initial values of the unknowns.

Default partitions (the ECOSIMPRO capability to internally find an automatic ordering of all the model equations, searching for the appropriate algebraic variables to be solved) normally success solving these non-linear implicit equation systems for very different cycle topologies; the degree of convergence is normally good even if the initial values of the algebraic variables are far from the solution. *Nevertheless, in some cases the user should conduct the partition wizards in a more physical way choosing alternative algebraic variables.* The most frequent problems a user may encounter are:

- The user must be especially careful when building a design model which leads to a well-described set of algebraic equations. *Wrong design options (see §11.2.2)* can lead to over-conditioned cycles (no partition can be generated) or under-conditioned cycles (strange boundaries conditions).

- The model equations system can sometimes have local minimums that make it nearly impossible to find a solution if the initial guest of the algebraic variables is on the wrong side of the solution. In this case, change the initial value of the unknowns and try to initialize the unknowns closer to the solution.
- Negative mass flows: Unlike for the other ESPSS transient libraries, the STEADY library is NOT adapted to negative flows in junctions or pipes (and of course in turbomachinery). A cycle disposition that would lead to reverse flow, in a bypass valve for example, will not converge. See §11.2.3.1.
- Imposed design conditions (chamber pressure, efficiencies, turbine pressure ratio, etc.) are not compatible with the cycle disposition or with the performance maps. In these cases, calculations can end with a simulation crash because there is no physical solution.
- Initial values for iteration in algebraic variables are not coherent: this means that rather than having initial guess values close to the solution, it is better to provide compatible values between the different components. For example, the initial pressure of serial turbines being compatible with the respective PI_tt values, the initial value of the pump pressure rise should be compatible with the chamber pressure, etc.
- Design conditions for the temperature of the cooling jacket wall can be difficult to obtain. Sometimes, an unphysical imposed temperature can give negative thickness.
- In Analysis mode, the valve position can be changed within limits; beyond some point, however, model assumptions can fail: no sonic condition in chamber, out of the performance maps, etc.
- In some cases, it is recommended to activate the "Disable equation reduction" option in the EcosimPro Options wizard or in the Partition wizard. In this way, the original components equation setup is lesser manipulated, reducing sometimes the risk of choosing unphysical closing equations solving the non-linear equation system of the cycle model.

11.2 LIBRARY ITEMS

11.2.1 Library Variables

The following variables are visible in any STEADY model and can be redefined in the experiment file (BOUND specification).

NAME	INITIAL	DESCRIPTION	UNITS
dp_lam	1000	Laminar pressure drop	Pa
GRAV	REAL	9.806	m/s2
PI_30	REAL	Conversion factor from rpm to rad/s (-)	-

11.2.2 Type Switches

The ENUM type global variables define series of labels representing options or global constants to be used in the code:

NAME	DESCRIPTION	Values
DesignType	Design option	{Design, Off_Design}
DesignType_Comb	Design option for Pre-burners	{Design, Off_Design, Design_MR}
PumpType	PumpType	{known_Ns, known_flow, Off_D_pump}
TurbineType	TurbineType	{known_mflow,known_PI_tt, known_pressures, Off_D_turb}
DeltaP	Valve DeltaP option	{UserGiven, FromPorts, Percentage}
PumpStage	Pump number of stages & suctions	{One, Two_S, Two_P}
RatioType	Mass flow ratio option in Tees	{NoRatio, f2_vs_f1, f3_vs_f1

Most components use some of these switches that enable switching the model between Design and Off-Design mode. The different options you can select are explained here:

DesignType = Design:

- Pipes, Junctions and Valves calculate their cross-sections from the calculated mass flow (from ports) and ΔP value either taken from the ports ("FromPorts" sub-option), or fixed from the user as a DATA ("UserGiven" sub-option). You need to determine which valves/Junctions are "UserGiven" or "FromPorts" (typically valves for turbine bypasses or adjusting a given pressure rise of a pump).
- Cooling circuits calculate the channel width and the internal wall thickness ("t_ch_cal" and "th_i_cal") at throat position of the Cooling Jacket from known wall temperature and ΔP.
- Combustion chambers calculate the inlet mass flows from given chamber pressure, mixture ratio (MR) and nozzle throat diameter (input data).

DesignType = Design_MR:

- Only for Pre-burners. Chamber pressure and mass flows are calculated from a given MR (input data) and from the calculated mass flows (from ports). This option is useful when the pre-burner pressure is determined by the downstream component, typically a turbine at given pressure ratio and known valves pressure drops.

DesignType = Off_Design:

- Pipes, Junctions and Valves have a given geometry. Mass flow and ΔP are calculated by the model, depending on the problem.
- Cooling circuits have a given geometry and calculate ΔP and wall temperatures from the rest of the model variables, depending on the problem.
- Combustion chambers have a given nozzle throat diameter. Chamber pressure and MR are calculated from the rest of the model variables, depending on the problem.

PumpType = known_Ns (as full design conditions, no performance map)

- Pumps calculate needed torque and power from mass flow and pressures (taken from the ports), and efficiency (user given). Rotation speed is calculated from the specific speed Ns (input data).

PumpType = known_flow (as partial design conditions, no performance map)

- Pumps calculate needed torque and power from mass flow & pressures (taken from the ports) and efficiency (user given). Rotation speed is conditioned by another mechanically linked pump.

PumpType =Off_D_pum (as off-design conditions, use of performance map)

- Pumps calculate *current* rotation speed, torque and power from the pump nominal conditions (input data), map performances and from the rest of the model, depending on the problem.

TurbineType = known_PI_tt (as full design conditions, no performance map)

- Turbines calculate mass flow (from ports variables), torque, rotation speed (from the mechanical port) and efficiency (user given). Pressure ratio is forced from the specified PI_tt value (input data).

TurbineType = known_mflow (as partial design conditions, no performance map)

- Turbines calculate mass flow (from ports variables), torque, rotation speed (from the mechanical port) and efficiency (user given). Pressure ratio is conditioned by the connected components (from port variables).

TurbineType = known_ pressures (as partial design conditions, no performance map)

- The same as before but forcing the known inlet pressure when calculating the outlet turbine properties.

TurbineType =Off_D_turb (as off-design conditions, use of performance map)

- Turbines calculate *current* rotation speed, torque and power from the turbine nominal conditions (input data), performance map and from the rest of the model variables, depending on the problem.

Concerning the Tee components, and depending on the problem, mass flow ratios through splits can be user-fixed or an output of the design calculation. Alternatively, multiple port connections can be used, avoiding the Tee component. The flow ratio of the different branches will then be calculated depending on the connected components and their respective pressure drop option: for example, if a turbine splits its outlet flow into a bypass valve and another turbine, the flow ratio will be calculated depending on the bypass valve option (fixed dP or calculated with the conditions of a predefined chamber pressure for example).

Likewise, turbine pressure ratio can be fixed or calculated *… Using Off-Design conditions for any component prevents the user from worrying about which options to use, and the default partition always works assuming the nominal operating conditions and geometry are known. As explained before (see §11.1.2.3) a Design partition can be built under Off-Design models to obtain the nominal operation conditions from the design conditions.*

11.2.3 Port Types

Two new ports have been created: the steady state *fluid_s* port that represents the basis and the rationale on which the whole library is coded, and the *nozzle port*, to better manage nozzle connections, similar to the transient nozzle port.

11.2.3.1 *"fluid_s"* port

This port has the following variables:

NAME	DESCRIPTION	UNITS
P	Total pressure	Pa
fluid	Working fluid name	
h	Total enthalpy (specific enthalpy + specific kinetic energy ½ V²)	J/Kg
m	Total mass flow	Kg/s
mh	Total enthalpy * flow	W
N_eq[Chemicals]	Molar fraction of the chemicals constituents. Only defined for the components placed downstream of a combustion chamber	-
n_fluid	Number of times the working fluid is defined (*always 1 if model is correct*)	-

The enthalpy flow is calculated in the port itself:

$$mh = m * h$$

The ports can connect several components. SUM variables will be summed; EQUAL variables will be propagated to all components.

Components should all have two ports, one IN and one OUT, thus defining the mass flow direction. Enthalpy flow mh is summed in IN ports of components (SUM IN specification). Enthalpy h is given in the OUT ports of all components (EQUAL OUT specification). *These port specifications are valid assuming only positive mass flows*. The conservation equations of the component are not valid with negative mass flow, both in design and analysis modes.

Please note that, unlike for the FLUID_FLOW_1D library, the pressure at the fluid port is the total pressure, not the static one.

11.2.3.2 *"NozzlePort"* port

This port has the following variables:

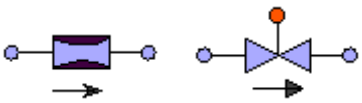
NAME	DESCRIPTION	UNITS
P	Static pressure (Pa)	Pa
T	Static temperature (K)	K
h	Total enthalpy (J/kg)	J/kg
m	Total mass flow (kg/s)	kg/s
s	Entropy (J/kg*K)	J/kg*K
x_eq[Chemicals]	Mass fraction of species (-)	-

11.3 OPERATIONAL COMPONENTS

11.3.1 Junction and Valve Components

11.3.1.1 Description

These components represent a concentrated load loss with constant or variable throat area and sonic flow limitation.



11.3.1.2 Construction Parameters

Name	Type	Description
dp_type	DeltaP	Valve DeltaP option (UserGiven, FromPorts or Percentage, see §11.2.2)
type	DesignType	Design, calculated flow area; OffDesign, fixed flow area

11.3.1.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid_s		IN	Inlet fluid port number 1
f2	fluid_s		OUT	Outlet fluid port number 2
s_pos	analog_signal	n=1	IN	Valve position signal

Valve position port is only available in Valves.

11.3.1.4 Data

Name	Type	Description	Units
Ao	REAL	Junction area when fully open [nominal if OffD; initial for iterative calculations if Design]	m^2
m_o	REAL	Initial mass flow and temperature. *To initialize fluid conditions in case of collection of flows upstream the valve*.	kg/s
T_o	REAL		K
dP_design	REAL	Absolute pressure drop in Design mode	Pa
dP_per	REAL	Relative pressure drop in Design mode	%
zeta	REAL	Loss coefficient	-

11.3.1.5 Formulation

In Off-Design mode, this component calculates the concentrated pressure drop in a junction, as in the corresponding transient component (see §5.3.1 and §5.3.2), but cancelling transient terms, so the mass flow needs to be calculated implicitly.

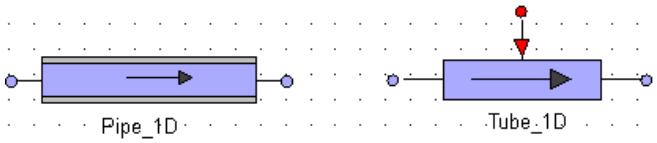
The flow areas of orifices in design calculations accountsfor critical flow conditions using the same formula as in a transient case (see §5.3.2.3).

11.3.2 Pipe_1D and Tube_1D Components

11.3.2.1 Description

Both components simulate a cylindrical1D fluid vein in steady conditions. The Pipe_1D component includes the wall thermal nodes but with no thermal connection with the exterior. The Tube_1D component does not include the thermal wall, but it is provided with a thermal port to be connected to a more accurate simulation of the wall using, for example, a Cylinder of the THERMAL library.

The Tube component can also exchange heat with a thermal port but does not include the wall nodes, which should be simulated by a THERMAL component like a "Cylinder" or a "Dnode" component.



11.3.2.2 Construction Parameters

Name	Type	Description
n_bends	CONST INTEGER	Number of bends
nodes	CONST INTEGER	Number of control volumes
type	DesignType	Design, Fixed dP; OffDesign, fixed flow area

Note: Design mode would calculate the Pipe diameter for a given delta P.

11.3.2.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid_s	-	IN	Inlet / outlet fluid ports
f2	fluid_s	-	OUT	
tp_in	THERMAL.thermal	(n = nodes)	IN	Thermal port

11.3.2.4 Data

Name	Type	Description	Units
num	REAL	Number of parallel identical tubes	-
L	REAL	Pipe length	m
D	REAL	Nominal pipe inner diameter	m
dP_design	REAL	Pressure drop in design mode	Pa
R_bend	REAL[n_bends]	Ratio of curvature of bends	m
alpha_bend		Bend angles (0, no bend)	degrees
rug	REAL	Roughness	m
fld_add	REAL	Pressure drop coefficient for additional losses	-
k_f	REAL	Multiplier of the friction factor	-
P_o	REAL	Initial Pressure	Pa
T_o		Initial temperature	K
m_o		Initial mass flow	kg/s
ht_option		Heat transfer option (see §A3)	
hc_dat	REAL	Heat transfer coef. if ht_option = constant	W/m^2*K

Name	Type	Description	Units
mat (See note 1)	Only in case of a Tube_1D comp. (See note 2)	Material	-
E_wall		Wall Elasticity Modulus , if mat=None	Pa
v_wall		Wall Poisson Modulus , if mat=None	-
cp_wall		Wall specific heat , if mat=None	J/kg/K
rho_wall		Wall density, if mat=None	Kg/m3
e_wall		Wall thickness	m
h_outside		Heat transfer coefficient with the environment	K
T_outside		Ambient temperature for the pipe	K

Note 1: Wall materials are selected via the ENUMERATIVE variable "mat" in which a list of materials is provided. Wall data only concerns the Pipe component.

Note 2: *h_outside, T_outside* are simple constant parameters to simulate the heat exchange with the exterior. For other more complex exchanges, use the "*Tube_1D*" components.

11.3.2.5 Formulation

Pipe & Tube components incorporate the mass, energy and momentum equations in steady regime according to a 1D centered spatial discretization. Equations are similar as in §5.3.8 just canceling the transient terms. The upwind scheme available in the transient library is not yet included in the formulation.

No fluid mixture equations are considered. Because the density is no longer a dynamic variable in steady conditions, the transient scheme rho-u is replaced by the p-h scheme. The tube takes into account the enthalpy variation due to the heat flux and the pressure drop due to the friction along the pipe. As for the transient version of the component the steady state tube is divided into volumes and junctions. The pressure drop and the enthalpy variation are calculated at the end of each volume, on the junction. The governing equations are the following:

Mass conservation

$$m = \rho_i vel_i A_i$$

Momentum conservation. For node i:

$$P_{i+1} = P_i - 0.5\xi_i \cdot m^2 / \rho_i / A_i$$
$$\xi_i = fr_i \Delta L / D_i$$

where the friction factor fr is a function of Re and roughness rug=Dh, as defined in the transient model.

Energy conservation. For node i:

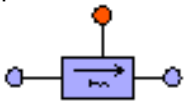
$$q_i = m(h_{i+1} - h_i)$$

where the q is the heat exchanged with the walls. It is evaluated as for the transient case, see §5.3.8.2

11.3.3 Jun_TMD

11.3.3.1 Description

This component represents an orifice with imposed mass flow.



11.3.3.2 Ports

Name	Type	Parameters	Direction	Description
f1	fluid_s		IN	Inlet / Outlet fluid ports
f2	fluid_s		OUT	
s_massflow	CONTROL.analog_signal	(n = 1)	IN	Imposed mass flow (kg/s)

11.3.3.3 Formulation

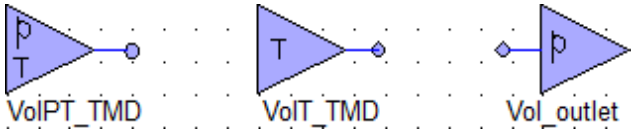
The mass flow is directly imposed by the signal port

$$m_{controlled} = s\_massflow.signal[1]$$

11.3.4 VolPT_TMD, VolT_TMD, Vol_outlet

11.3.4.1 Description

These components represent time dependent (TMD) boundary conditions in P-T.



11.3.4.2 Construction Parameters

Name	Type	Description
type	DesignType	Design option

Note: Design mode is only active for the VolT_TMD component.

11.3.4.3 Ports

Name	Type	Parameters	Direction	Description
f	fluid_s		IN	Inlet/Outlet fluid port

11.3.4.4 Data

Name	Type	Description	Units
P_o	REAL	Imposed pressure	Pa
T_o	REAL	Imposed temperature	K

11.3.4.5 Formulation

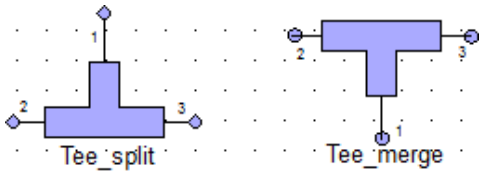
Depending on the type, the following variables (boundaries) will be fixed:

- VolT_TMD: Inlet boundary condition in T and in P if Design option is active
- VolPT_TMD: Inlet boundary condition in P-T
- Vol_outlet: Outlet boundary condition in P

11.3.5 Tee_split, Tee_merge

11.3.5.1 Description

These components represent Tees for spliting or merging several flows.



11.3.5.2 Construction Parameters

Name	Type	Description
type	RatioType	Mass flow option

11.3.5.3 Ports

Name	Type	Tee_split	Tee_merge
f1	fluid_s	Inlet	Outlet
f2	fluid_s	Outlet port 2	Inlet port 2
f3	fluid_s	Outlet port3	Inlet port 3

11.3.5.4 Data

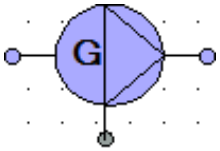
Name	Type	Description	Units
m_o	REAL	Initial mass flow [only to init]	Kg/s
mflow_ratio (for splitter)	REAL	Mass flow ratio - depending on type	-
m2_m_calc_o (for merge)	REAL	Initial mass flow ratio [guess]	-

For closed cycles, in most cases, the mass flow is determined by the preburner (staged combustion) or by bypass valves (expander), and therefore it is given from the ports, and the pressure ratio should be calculated in the turbine component. This is why the use of TEE components, where the mass flow ratio between the branches is fixed, is suggested besides the use of the known_mflow turbine switch.

11.3.6 Pump_gen

11.3.6.1 Description

This component simulates a pump for liquids in steady conditions. It is provided with fixed dimensionless pump characteristic curves valid for several types of pumps (Pump specific speed).



11.3.6.2 Construction Parameters

Name	Type	Description
ptype	PumpType	known_Ns, fixed specific speed; known_flow, calculated; Off_D_pump, Off-Design

11.3.6.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid_s		IN	Inlet / Outlet fluid ports
f2	fluid_s		OUT	
sh_in	MECHANICAL.shaft		IN	Shaft port

11.3.6.4 Data

Name	Type	Description	Units
Stagetype	PumpStage	One: 1 stage; Two_S: 2 serial stage; Two_P; 2 parallel stages	
Ns	REAL	Pump specific speed [nominal if OffD; fixed if known_Ns]	
eta_o	REAL	Efficiency [nominal if OffD; fixed if Design]	-
dP_o	REAL	Pressure rise [nominal if OffD; for iterative calculations if Design]	Pa
T_o	REAL	Temperature [nominal if OffD; for iterative calculations if Design]	K
m_o	REAL	Mass flow [nominal if OffD; for iterative calculations if Design]	kg/s
rpm_o	REAL	Pump speed [nominal if OffD; for iterative calculations if Design]	rpm

11.3.6.5 Formulation

The Pump component incorporates the mass, energy and momentum equations as in the equivalent transient Pump (see §7.2.3), but in steady regime, i.e. cancelling all the transient terms, so the equation system become implicit. Nevertheless, a simplification is made for the calculation of dh_ise in pumps by an approximate expression depending on the inlet density and deltaP. Assuming a perfect liquid behavior:

```
dh_is = (f2.P-f1.P)/rho
```

rho is the inlet density

In design conditions there is no need to use performance maps because the torque and pressure head TDH (interpolated in the map tables in Off-Design conditions) will be fixed to the rated values:

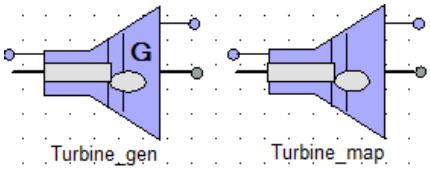
```
tdh_r = P_o/rho_r/9.806
torque_r = m_o*P_o/rho_r/eta/(rpm_o*PI/30)
```

11.3.7 Turbines

11.3.7.1 Description

These components simulate a turbine for gases in steady conditions.

The Turbine_gen is provided with *calculated* (no input) but adjustable dimensionless characteristic curves. The Turbine_map has user defined curves (dimensionless torque and mass flow coefficients) as input data tables depending on the dimensionless speed and pressure ratio coefficients



11.3.7.2 Construction Parameters

Name	Type	Description
turbine_type	TurbineType	known_PI_tt: fixed (user given) PI_tt; known_mflow: mass flow imposed by ports known_pressures: inlet and outlet pressure imposed by ports; Off_D_turb: Off-Design conditions, use of performances maps

See §11.2.2.

11.3.7.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid_s		IN	Inlet / Outlet fluid ports
f2	fluid_s		OUT	
sh_in	MECHANICAL.shaft		OUT	Shaft port

11.3.7.4 Data

Name	Type	Description	Units
N_nom	REAL	Nominal characteristic speed [if OffD]	-
rpm_o	REAL	Axial speed [nominal if OffD; initial guest if Design]	rpm
eta_o	REAL	Turbine efficiency [nominal if OffD; fixed if Design]	-
PI_tt_o	REAL	Pressure ratio [nominal if OffD; fixed if known_PI_tt]	-
P_o	REAL	Initial inlet pressure for iterative calculations	Pa
T_o	REAL	Initial inlet temperature for iterative calculations	K
m_o	REAL	Initial mass flow rate for iterative calculations	kg/s
Ain	REAL	Character. inter-blade flow area [if OffD, for generic turbines]	m^2

Name	Type	Description	Units
beta	REAL	Characteristic blade-axe angle [if OffD, for generic turbines]	deg
Qplus_vs_N_PI	TABLE 2D	Dimensionless flow Q+ [if OffD, for map turbines]	-
ST_vs_N_PI		Dimensionless torque ST [if OffD, for map turbines]	-

The performance maps are a function of N/N_nom and PI_tt/PI_tt_nom. (Transient turbine maps are a function of N and PI_tt). The mass flow coefficient and specific torque are defined as:

- Mass flow coefficient: $Q^+ = m_{map} \cdot v_{son} /(r^2 P_{o1})$
- Specific torque: $ST = T /(r \ m_{map} \cdot v_{son})$

Independent variables are:

- Speed coefficient: $N = r \cdot \omega / v_{son}$
- Total pressure ratio: $\Pi = P_{o1} / P_{o2}$

where:

m_{map} = mass flow
v_{son}= sound speed
T= Consumed torque
ω = rotational speed

11.3.7.5 Formulation

The Turbine components incorporate the mass, energy and momentum equations as in the equivalent transient Turbines (see §7.2.5 and 7.2.6), but in steady regime, i.e. cancelling all the transient terms, so the equation system becomes implicit.

In design conditions there is no need to use performance maps because the efficiency and pressure ratio PI_tt (interpolated in the map tables in Off Design conditions) will be fixed to the rated values.

The map reconstruction (turbine sizing) from the operational conditions is made as follows:

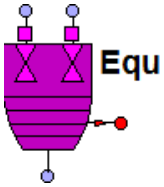
```
eta_op = (h_in-h_out) / (h_in -H_ise)
R_op = N_nom * vsound / (2*PI/60*rpm_op)
Ain_op = m_op *tan(PI*beta/180) /vsound /N_nom /rho_in
```

When the combusted gases feed a turbine, their properties depend on their composition and on the local temperature. As in the transient turbines the enthalpy is calculated with the current chamber gas composition and at the local turbine temperature.

11.3.8 PreBurner_eq component

11.3.8.1 Description

This component represents a non-adiabatic 1D Preburner for liquid or gas propellants built by means of a combustor and two injectors. The properties and composition of combusted gases will be transmitted by the outlet fluid port to other possible combustors (staged engines) or turbines.



11.3.8.2 Construction Parameters

Name	Type	Description
Nsub	CONST INTEGER	Number of nodes along the chamber axial direction

Name	Type	Description
type	DesignType	Design, fixed MR & chamber pressure, calculated injectors areas; OffDesign, fixed injectors areas

Note: Nsub determines the number of fluid and wall thermal nodes.

11.3.8.3 Ports

Name	Type	Parameters	Direction	Description
f_out	fluid_s		OUT	Outlet combustion gases port
f_oxy			IN	Inlet oxidizer port
f_red			IN	Inlet fuel port
tp	thermal	(n = Nsub)	OUT	Thermal port

The outlet port can only be connected to a *STEADY type* component (fluid_s type port).

11.3.8.4 Data

Name	Type	Description	Units
Dt	REAL	Chamber throat diameter	m
Rcurv	REAL	Curvature radius at throat	m
Lc	REAL	Chamber length of subsonic part	m
Dc_vs_L	TABLE 1D	Normalized subsonic chamber diameters vs. normalized axial position	-
dx_vs_L		Weighting function for mesh size distribution or normalized node lengths vs. node number	-
dx_input	BOOLEAN	FALSE, dx_vs_L = weighting function for mesh size distribution. TRUE, normalized node lengths vs. node number	-
A_inj_oxy	REAL	Effective injection area for oxidizer [if Off-D]	m^2
dP_per_oxy	REAL	Oxidizer injector design pressure drop [if Design]	%
zeta_inj_oxy	REAL	Oxidizer injector loss coefficient (-)	-
A_inj_red	REAL	Effective injection area for gas reducer [if Off-D]	m^2
dP_per_red	REAL	Oxidizer injector design pressure drop [if Design]	%
zeta_inj_red	REAL	Reducer injector loss coefficient (-)	-
emiss	REAL	Emissivity of the combustion gases	-
MR_o	REAL	Mixture Ratio [initial if Off-D; assigned if D] (-)	-
Pc_o	REAL	Chamber pressure [initial if Off-D; assigned if D](Pa)	Pa
Tc_o	REAL	Combustion temperature [to initialize] (K)	K
m_o	REAL	Initial combusted mass flow [only for init]	Kg/s
eta_c	REAL	Combustor efficiency	-
eta_cf	REAL	Nozzle efficiency	-

Notes: *Dc_vs_L,dx_vs_L* tables allow variable geometry, variable mesh size in the chamber. The discretized (Nsub) wet areas and diameters will be interpolated using these tables from L=0 to L=Lc. Dt is used for the normalization of diameters and Lc for that of the axial lengths**.** *See §8.1.2.2 for more details.*

11.3.8.5 Formulation

This component incorporates the mass and energy equations as in the equivalent transient component (see Sections 8.2.1 to 8.2.5), but in steady regime, i.e. cancelling all the transient terms and those corresponding to non-condensable gases, so the equation system becomes implicit:

The first step is to calculate the molar fractions of the reactants:

Reducer contribution: $N_{k,1} = x_{red}\frac{y_{k,red}}{MW_{mix,red}}$; $MW_{mix,red} = \sum_{k=1,Nchem} y_{k,red} MW_{k,red}$

Oxidizer contribution: $N_{k,1} = N_{k,1} + x_{oxy}\frac{y_{k,oxy}}{MW_{mix,oxy}}$; $MW_{mix,xoy} = \sum_{k=1,Nchem} y_{k,oxy} MW_{k,oxy}$

Nchem is extended to any chemical treated by the FLUID_PROPERTIES library, see §4.1.2
MW_k : is the molecular weight of the chemical constituent k
x_{red}; x_{oxy}: Mass fractions of reducer and oxidizer
y_{k_red}; y_{k_oxy}: Molar fraction of chemical k of the reducer (oxidizer) mixture
$N_{k,1}$: Number of moles of the chemical constituent k of the reactant mixture

x_{red}, x_{oxy} are calculated from the injector mass flows:

$$x_{red} = 1/(MR+1); \qquad x_{oxy} = MR/(MR+1)$$

$$m = f\_red.m + f\_oxy.m; \; f\_oxy.m = MR * f\_red.m$$

The next step is to calculate the enthalpy mixture of reactants:

$$h_{mix} = x_{red}(h_{red} - dH_{red}) + x_{oxy}(h_{oxy} - dH_{oxy}) - q_{wall}/m$$

h_{red}, h_{oxy} are the injector enthalpies calculated by the upstream (injectors) chamber components and dH_{red}, dH_{oxy} are the enthalpy shifts between the real properties reference and the CEA reference.

The term q_{wall} is the heat exchange with the walls (a cooling jacket for example, or some other thermal component), making the equations system still strongly coupled (implicit) with the other cycle components. The formulation used to calculate the heat exchange (Bartz correlation) is the same as in the corresponding transient component (see §8.2.3.5). Here, q_{wall} is the sum for all the chamber wall sections evaluated at the corresponding local Mach number depending on the local section ratio. The local Mach number is calculated as in Section §8.2.8.2, choosing the subsonic solution.

The number of moles, Nk, is then normalized. This number and the enthalpy mixture h_{mix} can be used to call the Minimum Gibbs energy method (see Appendix A7) to obtain the equilibrium temperature and the molar fraction of the products:

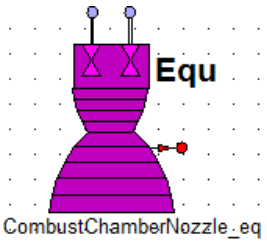
$$(y_{k\_eq}, T_{eq})_1 = f_{minGibbs}(N_{k,1}, h_1 - v_1^2/2, Pch)$$

In design conditions the mixture ratio MR and the chamber pressure Pch are input data, while the injector areas will be calculated. In Off-Design conditions MR and PC will became algebraic variables to be calculated as per the calculated mass flows at injector and chamber exits.

11.3.9 CombustChamberNozzle_eq component

11.3.9.1 Description

This component represents a non-adiabatic 1D Main Thruster for liquid or gas propellants built by means of a combustor and two injectors and a supersonic nozzle.



11.3.9.2 Construction parameters

Name	Type	Description
Nsub	CONST INTEGER	Number of subsonic nodes along the chamber axial direction
Nsup	CONST INTEGER	Number of supersonic nodes along the chamber axial direction
type	DesignType	Design, fixed MR & chamber pressure, calculated injectors areas; OffDesign, fixed injectors areas

11.3.9.3 Ports

Name	Type	Parameters	Direction	Description
f_oxy	fluid_s		IN	Inlet oxidizer port
f_red			IN	Inlet fuel port
tp	thermal	(n = Nsub)	OUT	Thermal port

11.3.9.4 Data

Name	Type	Description	Units
Dt	REAL	Chamber throat diameter	m
Rcurv	REAL	Curvature radius at throat	m
Lc	REAL	Chamber length of subsonic part	m
Ld	REAL	Total length of supersonic part. To normalize supersonic axial position	m
Ld2	REAL	Nozzle axial length from throat to outlet	m
Dc_vs_L	TABLE 1D	Normalized subsonic chamber diameters vs. normalized axial position	-
dx_vs_L		Weighting function for mesh size distribution or normalized node lengths vs. node number	-
dx_input	BOOLEAN	FALSE, dx_vs_L = weighting function for mesh size distribution. TRUE, normalized node lengths vs. node number	-
Dd_vs_L	TABLE 1D	Normalized nozzle diameters vs. normalized axial position	-
dxd_vs_L		Weighting function for supersonic mesh size distribution or normalized node lengths vs. node number	-
dxd_input	BOOLEAN	FALSE, dxc_vs_L = weighting function for supersonic mesh size distribution. TRUE, normalized node lengths vs. node number	-
AR_sup	REAL	Nozzle Exit/throat area ratio: 0, no Dd_vs_L profile modification	-
A_inj_oxy	REAL	Effective injection area for oxidizer [if Off-D]	m^2
dP_per_oxy	REAL	Oxidizer injector design pressure drop [if Design]	%
zeta_inj_oxy	REAL	Oxidizer injector loss coefficient	-
A_inj_red	REAL	Effective injection area for gas reducer [if Off-D]	m^2
dP_per_red	REAL	Oxidizer injector design pressure drop [if Design]	%
zeta_inj_red	REAL	Reducer injector loss coefficient	-
emiss	REAL	Emissivity of the combustion gases	-
MR_o	REAL	Mixture Ratio [initial if Off-D; assigned if D]	-
Pc_o	REAL	Chamber pressure [initial if Off-D; assigned if D]	Pa
Tc_o	REAL	Combustion temperature [to initialize]	K
m_o	REAL	Initial combusted mass flow [only for init]	Kg/s
eta_c	REAL	Combustor efficiency	-
eta_cf	REAL	Nozzle efficiency	-
frozen_th	BOOLEAN	Flag forcing frozen conditions in the throat	
frozen_nz	BOOLEAN	Flag forcing frozen conditions in the nozzle	
P_ext	REAL	External pressure (only for thrust evaluation)	Pa

Notes:

1. *Dc_vs_L,dxc_vs_L* and *Dd_vs_L,dxd_vs_L* tables allow variable geometry, variable mesh size in the chamber and in the nozzle respectively. The discretized (Nsub+Nsup) wet areas and diameters will be interpolated using these tables from L=0 to L=Lc in the subsonic zone and from L=0 to L=Ld2 in the supersonic zone. Dt is used for the normalization of diameters and Lc, Ld for the normalization of the subsonic, supersonic axial lengths respectively. *See §8.1.2 for more details.*
2. frozen_th = FALSE means freezing the gas composition in nozzle based on the equilibrium composition at the throat.
3. AR is a new input that modifies the aspect ratio of a given profile without changing the relative shape. For a conic nozzle this is equivalent to changing the nozzle angle

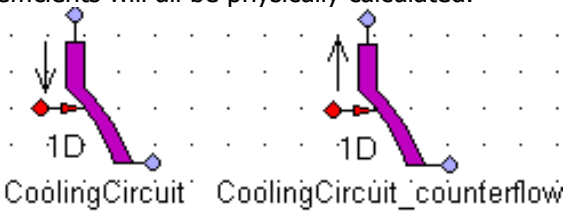
11.3.9.5 Formulation

The CombustChamberNozzle_eq component is built topologically and includes the formulation of a combustor (see *§11.3.8.5)* and that of a non-isentropic nozzle (see*§8).* Additionally, the outlet mass flow is assumed to be sonic, as described in *§8.2.8.1.* When the chamber pressure (analysis mode) is below the external pressure, a subsonic transition will be considered.

11.3.10 CoolingCircuit and CoolingCircuit_counterflow components

11.3.10.1 Description

These components represent a Regenerative Circuit of a Chamber. A 1D geometry (built by means of an equivalent 1D wall around the channels) will be taken into account. Fluid and wall temperatures, pressure losses and heat exchange coefficients will all be physically calculated.



Since the direction of the flow in the Steady State library must be given at the schematic design stage, two components are foreseen: a co-flow and a counter flow cooling jacket.

11.3.10.2 Construction parameters

Name	Type	Description
Nsub	CONST INTEGER	Number subsonic nodes
Nsup	CONST INTEGER	Number of supersonic nodes
type	DesignType	Design, Fixed dP and wall temperature; OffDesign, calculated

Nsub+Nsup are the number of fluid nodes of the channels.

1. Put Nsub = 0 if the Cooling Jacket is connected to a *Nozzle*
2. Put Nsup = 0 if the Cooling Jacket is connected to a *PreBurner*
3. Put Nsub != 0 and Nsup !=0 if the Cooling Jacket is connected to a *CombustChamberNozzle*

11.3.10.3 Ports

Name	Type	Parameters	Direction	Description
f1	fluid		IN	Inlet/Outlet fluid port
f2			IN	Inlet/Outlet fluid port
tp_ch	thermal	(n = Nsub+Nsup)	IN	Thermal port to chamber

11.3.10.4 Data

Name	Type	Units	Units
Dt	REAL	Cooling jacket throat diameter	m
Lc	REAL	Length of the convergent part. To normalize subsonic axial position	m
Dc_vs_L	TABLE 1D	Normalized convergent chamber diameters vs. normalized axial position	-
dxc_input	BOOLEAN	FALSE, dxc_vs_L = weighting function for conv. mesh size distribution. TRUE, normalized node lengths vs. node number	
dxc_vs_L	TABLE 1D	Weighting function for conv. mesh size distribution or normalized node lengths vs. node number	-
Ld	REAL	Total length of divergent part. To normalize supersonic axial position	m
Ld1	REAL	Nozzle axial length from the throat to the inlet (0 if Lc != 0)	m
Ld2	REAL	Nozzle axial length from the throat to the outlet	m

Name	Type	Units	Units
Dd_vs_L	TABLE 1D	Normalized divergent diameters vs. normalized axial position from the throat	-
dxd_input	BOOLEAN	FALSE, dxc_vs_L = weighting function for div. mesh size distribution. TRUE, normalized node lengths vs. node number	
dxd_vs_L	TABLE 1D	Weighting function for div. mesh size distribution or normalized node lengths vs. node number	-
AR_sup	REAL	Nozzle Exit/throat area ratio: 0, no Dd_vs_L profile modification	-
n_ch	INTEGER	Number of channels	-
init_option	ENUM	Option to specify the initial thermodynamic state	
P_o	REAL	Initial Pressure	Pa
T_o		Initial temperature	K
k_f	REAL	Multiplier of the friction factor	-
mat_i	ENUM	Chamber internal Material	-
K_i	REAL	Internal wall conductivity if mat=None	W/m/K
rug	REAL	roughness	m
w_ch	REAL	Channel width at throat section	m
t_ch	REAL	Channel height at throat section	m
th_i	REAL	Jacket inner wall thickness at throat section	m
wc_vs_L	TABLE 1D See note 2	Non-dimensional channel widths vs. non-dimensional axial position	
tc_vs_L		Non-dimensional channel heights vs. non-dimensional axial position	
ti_vs_L		Non-dim. inner wall thickness vs. non-dimensional axial position	
dP_design	REAL	Pressure drop [in design mode; initial if OffDesign]	Pa
Tw_ design	REAL	Wall temperature at throat [in design mode]	K
eta_q_throat	REAL	Heat flux efficiency at throat	-
etaq_vs_L	REAL	Dimensionless heat flux efficiency vs. dimensionless axial position	-

Notes:

1. *Dc_vs_L,dxc_vs_L* and *Dd_vs_L,dxd_vs_L* tables allow variable geometry, variable mesh size in the convergent/divergent parts. These tables will be used to discretize (Nsub+Nsup) and to interpolate the channel profile from L=0 to L=Lc in the convergent zone and from L=Ld1 to L=Ld2 in the divergent zone. Dt is used for the normalization of diameters and Lc, Ld for the normalization of the convergent/divergent axial lengths respectively. *See §8.1.2 for more details.* Ld1 must be zero if a convergent part (Nsub != 0, Lc != 0) is included.
2. Non dimensional channel width, height and thickness tables must be entered as a function of the non-dimensional length from *the cooling jacket inlet to the exit*, i.e. the x vector of these tables is normalized with "Lc+Ld", and must *start with x=0 and finish with* x=1. "Y" values are normalized with the corresponding "w_ch,t_ch, th_e, th_e" values. Default tables {{0,1},{1,1}} mean constant geometry along x. Note that nozzle profile and mesh tables (*Dc_vs_L,dxc_vs_L, …*) have a different x origin (the injection plan) to be compatible with the chamber geometry input data.
3. Wall materials are selected via the ENUMERATIVE variable "mat" in which a list of materials is provided (see §3.9).

11.3.10.5 Formulation

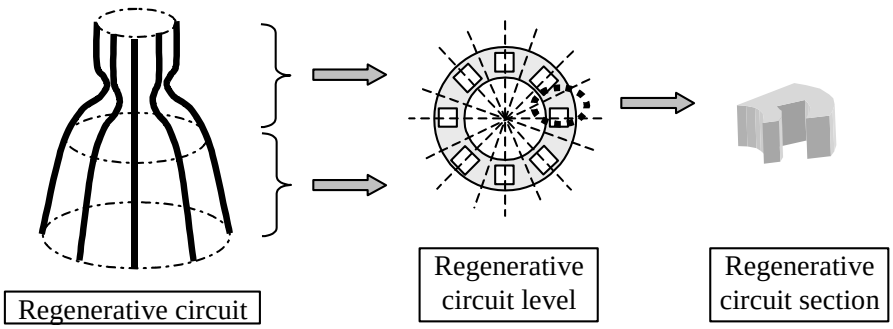


Figure 11-3 Cooling jacket disposition

The cooling jacket is divided into a variable (*Nsub+Nsup*) number of sections in axial direction. Every section is made of:

- a fluid node which is simulating the cooling channels as in the "Tube" component (see §11.3.2.5). The mesh size of each node is function of the axial position through the non-dimensional geometry tables. See §8.1.2.2.

  The rectangular channel geometry (widths, heights, wet areas) is also calculated using the interpolated widths and heights as follows:

$$A_{wet,i} = 2(a_i + b_i) \cdot l_{ch,i}$$

  where

$$a_i = w\_ch \cdot Interp(x_i/(Lc + Ld),\ wc\_vs\_L)$$
$$b_i = t\_ch \cdot Interp(x_i/(Lc + Ld),\ tc\_vs\_L)$$

- a wall section representing the inner wall between the chamber and the channels. The wall thickness will be interpolated using the "ti_vs_L" tables. The heat conduction is simplified with respect to the 3D model. For each section node, the heat exchanged at the combustor/nozzle side is equal to the heat taken by the fluid channels:

  -- heat flux from the wall cold temperature to the cooling fluid:

$$q[i] = hc[i] * A\_noz[i]/n\_ch\ (Tw\_cold[i] - T[i])$$

  -- heat flux through the internal walls:

$$q[i] = (Tw\_hot[i]-Tw\_cold[i]) * ki\_var[i]/ti[i] * A\_noz[i])$$

  where *A_noz* is the chamber/nozzle combusted gases wet area, depending on the section (subsonic or supersonic). *Tw_cold, Tw_hot* are the wall temperatures at the channel/chamber sides respectively. The heat exchange coefficient *hc* is calculated as for the Tube component, see §5.3.8.2.

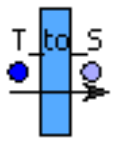
  The heat exchanged q must be equal to the one calculated by the combustor or nozzle component according to the Bartz correlation, see §8.2.3.5

In design conditions two additional equations are set: the pressure drop and wall temperature at the throat are imposed, whereas the wall thickness and channel height at the throat section are calculated. In analysis mode, these two geometrical variables will be considered as input data, whereas the pressure drop and the wall temperature are calculated.

11.3.11 Component TranSteady_IF

11.3.11.1 Description

This component connects STEADY components with transient (FLUID_FLOW_1D, TANKS, etc.) components.



11.3.11.2 Ports

Name	Type	Parameters	Direction	Description
f1	fluid		OUT	FLUID_FLOW_1D fluid port
f2	fluid_s		OUT	STEADY fluid port

Note: The transient port (FLUID_FLOW_1D fluid port) must be connected to a capacitive transient component such as a Pipe or a Tank.

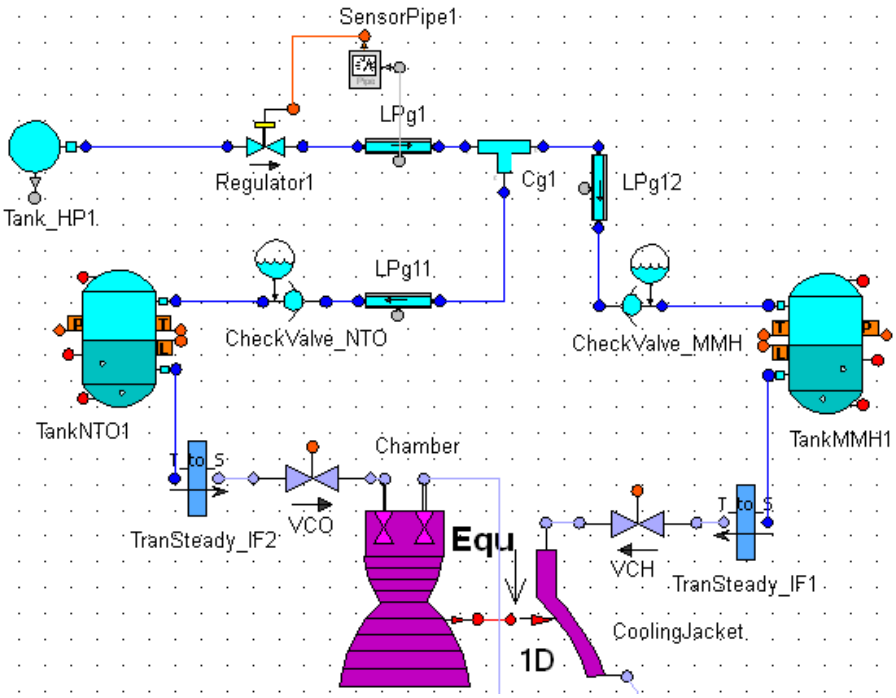
11.3.11.3 Data

Name	Type	Units	Units
x_jun	REAL	X coordinate relative to a body axis system	m
y_jun	REAL	Y coordinate relative to a body axis system	m
z_jun	REAL	Elevation relative to a body axis system	m

11.3.11.4 Formulation

This component transfers the transient values of pressure, enthalpy and working fluid to the steady model. This makes it possible to study a transient boundary condition (for example a Tank) connected to a quasi-steady model.

The following example (the pressure-fed engine of the STEADY_EXAMPLES library) shows how a transient pressurization system can be coupled to a steady combustor:



Transient / steady models coupling have some limitations:

- The default partition (automatic model generation by EcosimPro) can lead to very complex non-linear equations system
- The transient boundary conditions must lead to *positive* flows in the steady components

The experiment of this pressure fed engine is shown below. Note that in this case it has been necessary to initialize some algebraic variables to reach convergence:

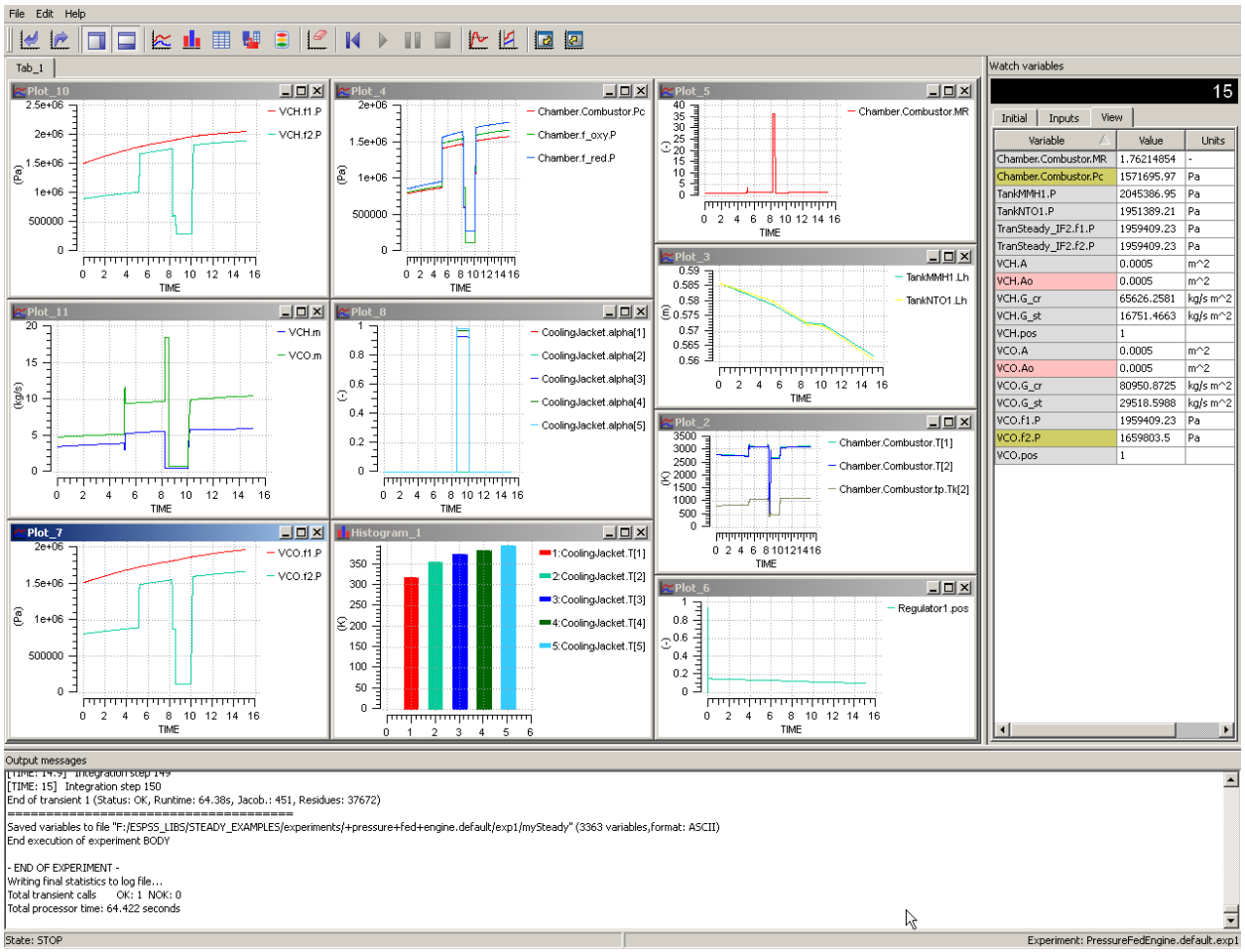
```
EXPERIMENT exp1 ON PressureFedEngine.default

DECLS
        TABLE_1D VCO_law = {{0, 5.10, 5.150, 8.5, 8.55, 10, 10.2, 100}, {0.3, 0.3, 1, 1, 0.03, 0.03, 1, 1}}
        TABLE_1D VCH_law = {{0, 5.15, 5.200, 8.2, 8.25, 10, 10.2, 100}, {0.3, 0.3, 1, 1, 0.03, 0.03, 1, 1}}

INIT
        -- Algebraic variables
        Chamber.Inj_red.f1.P = 15e5
        Chamber.Inj_oxy.f1.P = 15e5
```

```
BOUNDS
        -- Set expressions for boundary variables: v = f(t;...)
        FLUID_FLOW_1D.Damp = 0.8
        FLUID_FLOW_1D.GRAV = 9.806
        FLUID_FLOW_1D.GRAVx = 0.
        FLUID_FLOW_1D.GRAVy = 0.
        STEADY.GRAV = 9.806
        STEADY.dp_lam = 3000
        FLUID_FLOW_1D.Re_lam = 2000
        TankMMH1.OMEGA = 0.
        TankMMH1.tp_in_Cylin.q[1] = 0
        TankMMH1.tp_in_Dome2.q[1] = 0
        ...
        TankNTO1.turb_fac = 1.
        VCH.s_pos.signal[1] = timeTableInterp( TIME, VCH_law)
        VCO.s_pos.signal[1] = timeTableInterp( TIME, VCO_law)

BODY
        HP = 200e5
        RP = 20e5
        z_reg = 0.9
        pg0 = 15e5 -- 15e5 -- initial gas line pressure
        tg0 = 275 -- initial gas line temper
        th0 = 295 -- initial liquid MMH line temper.
        to0 = 295 -- initial liquid NOO line temper.
        VCH.Ao=0.0005
        VCO.Ao=0.0005

        Chamber.P_ext = 0.01e5
        Chamber.A_inj_oxy = 0.0005 -- 0.0005 --
        Chamber.A_inj_red = 0.0005 -- 0.0005 --
        Chamber.Dt = 0.15 --0.15
        CoolingJacket.dP_design = 1e5
        CoolingJacket.n_ch = 50 -- 120
        CoolingJacket.t_ch = 0.005
        CoolingJacket.mat_i = Steel321
        Chamber.frozen_th = TRUE

        CINT = 0.1
        REL_ERROR = 1e-4
        ABS_ERROR = 1e-4
        REPORT_MODE = IS_STEP
        IMETHOD = DASSL_SPARSE
        TIME = 0.0
        TSTOP = 15
        INTEG()
END EXPERIMENT
```

The simulation is a transient case (analysis mode, default partition) where the tank initial conditions are still below the nominal pressure. The sequence is the following:

- Engine valves opened at 30% during the first 5 seconds
- Valves to 100% in a few milliseconds
- Quasi-steady conditions for 8 seconds
- Then, an engine valve is commanded to close (minimum closing is 3%, below this value the steady model seems not to be consistent)
- An engine restart at TIME = 10 is commanded

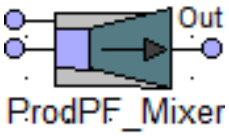
During the actuations, the Tanks are pressurized from 15 (initial conditions) to 20 bar according to the FLUID_FLOW_1D (pressure regulator) and TANKS transient model components.



11.3.12 ProdPF_Mixer

11.3.12.1 Description

The "ProdPF_Mixer" permits the simulation of a mixture of combusted gases and pure fluids or the mixture of two different combusted gases. A combustion chamber with more than 2 injectors can then be modelled placing this component upstream of one of the chamber injectors and collecting two flows, one, for example, feeding combusted gases coming from a pre-burner and the other feeding a pure fluid coming from a pump.



11.3.12.2 Construction parameters

Name	Type	Description
type	DesignType	Design, fixed m_in1 to m_out ratio; OffDesign, calculated

11.3.12.3 Ports

Name	Type	Parameters	Direction	Description
f1	Fluid_s		IN	Inlet 1 fluid port
f2	fluid_s		IN	Inlet 2 fluid port
f3	fluid_s		OUT	Outlet fluid port

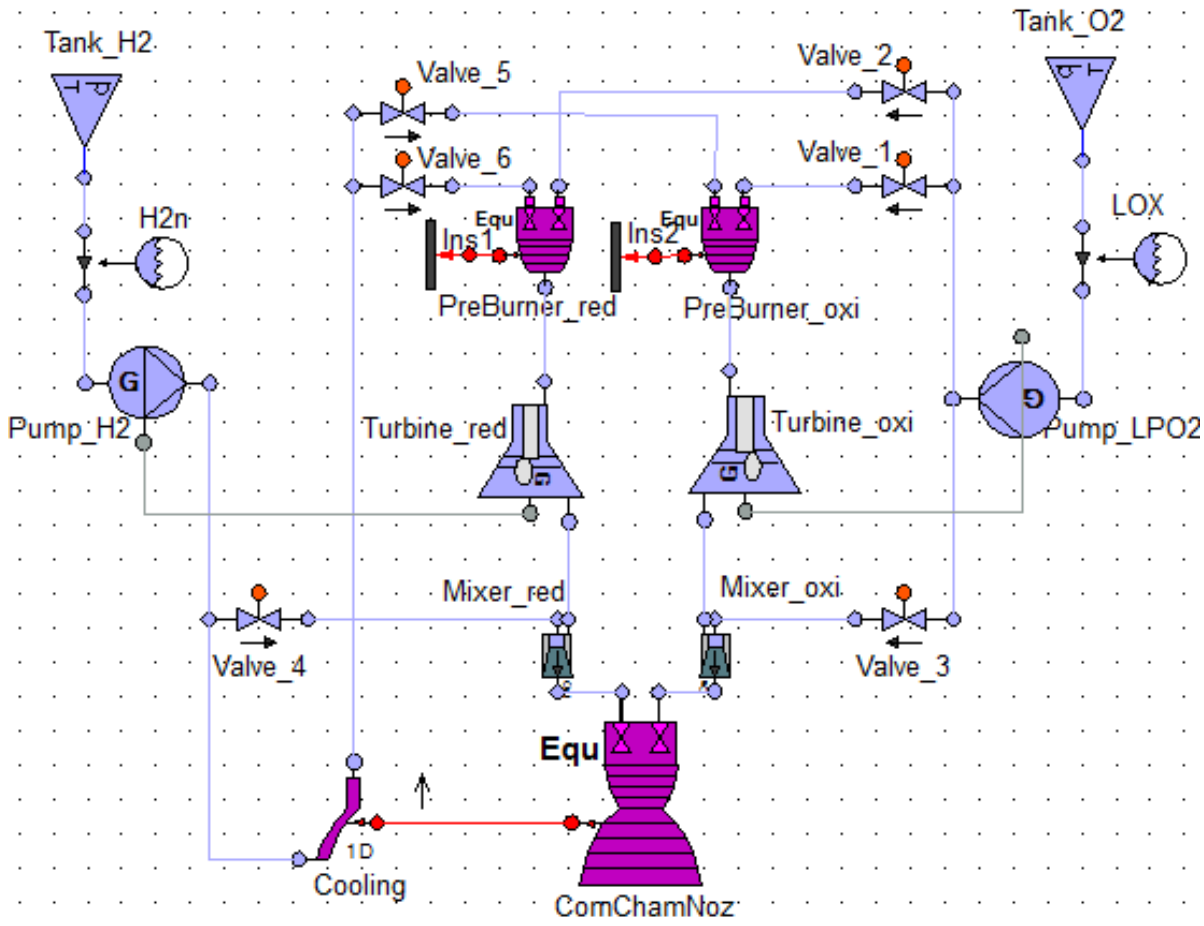
11.3.12.4 Data

Name	Type	Units	Units
mflow_ratio	REAL	Mass flow ratio m_in1/m_out - depending on type	-
m_o		Initial outlet mass flow rate for iterative calculations	kg/s

11.3.12.5 Formulation

This component uses a similar formulation as for its equivalent transient "PordMixer_tee" of the COMB_CHAMBERS library (see §) but in steady conditions (no transient terms included).

The following example ("ssme" of the STEADY_EXAMPLES library) shows a model of a simplified SSME engine with a main chamber of 4 injectors:

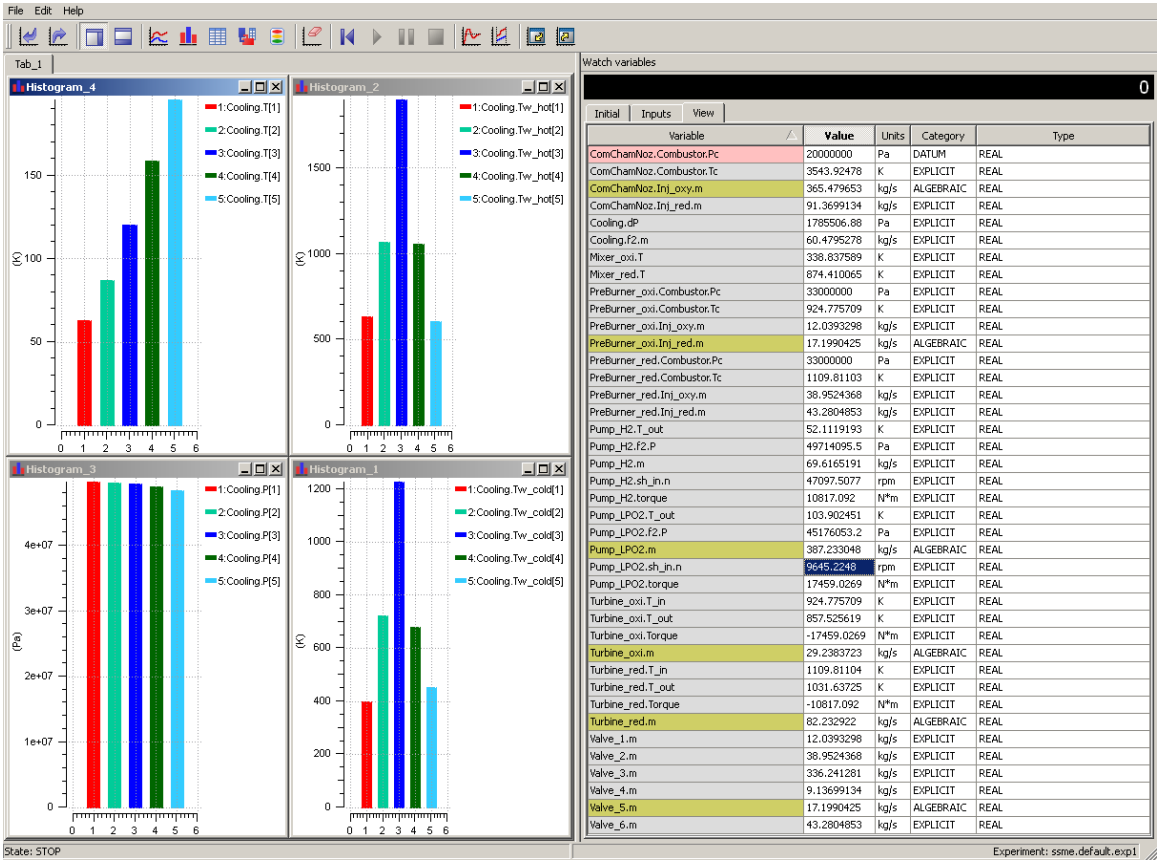


Shown below is the experiment of this model:

```
EXPERIMENT exp1 ON ssme.default
DECLS
INIT
BOUNDS
    -- Set equations for boundaries: boundVar = f(TIME;...)
        STEADY.GRAV = 9.806
        STEADY.dp_lam = 3000
        Valve_1.s_pos.signal[1] = 1
        Valve_2.s_pos.signal[1] = 1
        Valve_3.s_pos.signal[1] = 1
```

```
        Valve_4.s_pos.signal[1] = 1
        Valve_5.s_pos.signal[1] = 1
        Valve_6.s_pos.signal[1] = 1
        -- Valve_7.s_pos.signal[1] = 1
        -- Valve_8.s_pos.signal[1] = 1
BODY
        Cooling.Dt = 0.3
        Cooling.n_ch = 200
        Cooling.t_ch = 0.015

        PreBurner_oxi.MR_o = 0.7
        PreBurner_red.MR_o = 0.9
        ComChamNoz.MR_o = 4 -- 4.5
        Mixer_red.mflow_ratio = 0.1 -- 0.2
        Mixer_oxi.mflow_ratio = 0.92

        DEBUG_LEVEL = 3
        STEADY()
END EXPERIMENT
```

The results can be seen below:



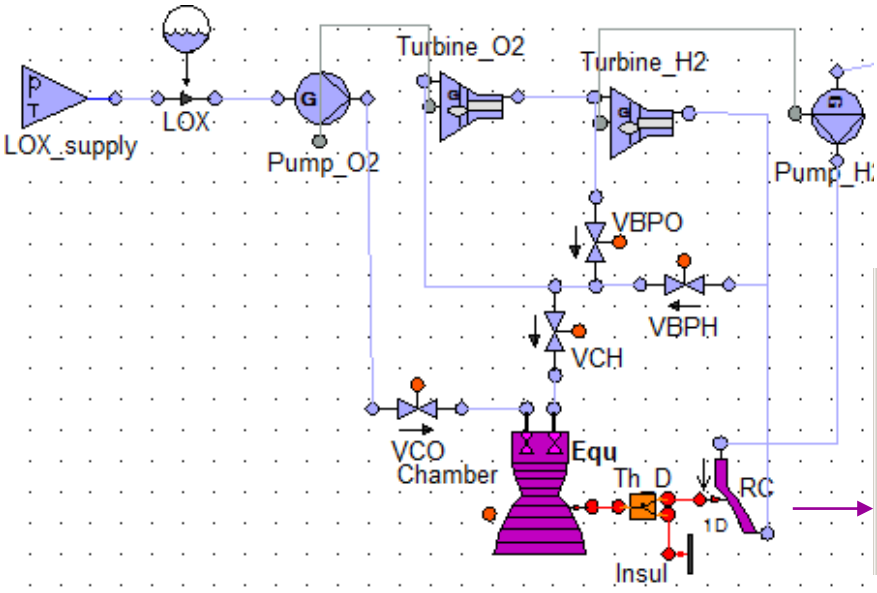
11.4 CYCLE DESIGN & MISSION REQUIREMENTS

The STEADY library can be coupled with the design of a cycle fulfilling mission requirements. The steps to be followed are:

- Build an "elastic" cycle containing the design conditions (efficiencies, performance map, etc) as part of the components' input data. Geometrical data can be proportional to some global parameters (Dt)

- Mass estimation and mission requirements can be coded in the continuous block of the model as a function of the actual engine performances and the engine dimensions
- The dimensions (throat diameter...) fulfilling mission requirements will be *implicitly calculated by EcosimPro*. Parametric studies in **Pc**, **MR** can be performed to optimize the engine mass

The "*expanderEngine*_mission" model of the STEADY_EXAMPLES library is an example of an "elastic" expander cycle fulfilling mission requirements. It is advisable to edit this model to have a template of the steps previously mentioned. Below is a picture of this cycle:



Self-sizing geometrical definition

Nsub	INTEGER	5	
Nsup	INTEGER	2	
		DATA	
Dt	REAL	1.1 * Dt	m
Lc	REAL	5 * Dt	m
Ld	REAL	20 * Dt	m
Ld1	REAL	0	m
Ld2	REAL	10 * Dt	m
Dc_vs_L	TABLE 1D	Edit	-
dxc_vs_L	TABLE 1D	Edit	-
dxc_input	BOOLEAN	FALSE	-
Dd_vs_L	TABLE 1D	Edit	-
dxd_vs_L	TABLE 1D	Edit	-
dxd_input	BOOLEAN	FALSE	-

The initial vehicle mass M_o and the propellant mass are decisive parameters on rocket performance, as the main goal is to elevate the maximum payload to the desired orbit.

Below is a printout of the simple function used for the evaluation the masses of the cycle components from the engine performances (mass flows, thrust, etc.) and the actual engine size (Dt, etc.):

```
FUNCTION NO_TYPE massExpander(
    -- Material properties
        REAL sigma_cc ,
        REAL sigma_nz ,
        REAL sigma_tank ,
        ...
BODY
        Dc = 2 * Dt   -- convergent diameter of the chamber
        t_rc = PI*Dt/n_ch_rc - w_ch_rc  -- inter-channel distance of the cool.jacket

    -- TurboPumps mass estimation
        M_tp = A_cof_tp * (Torque_oxi**B_cof_tp + Torque_red**B_cof_tp)

    -- Chamber/Nozzle mass estimation
        Acc = PI * Dc * Lc
        tcc = Pc/2 * Dc / (2 * sigma_cc)
        M_cc = rho_cc*Acc*tcc + 3*(Pc/40e5) + 3*(Dc/0.1)**2

    -- torus/pipes/valves ...
        Anz = PI * 0.7 * Dt * (1 + sqrt(AR)) * Ld
        tnz = Pc * Dt / (2 * sigma_nz)
        M_nz = rho_nz * Anz * tnz
```

```
-- Injectors mass estimation
    t_inj = sqrt(3 / 16 * 2 * Pc * Dc ** 2 / sigma_cc)
    M_inj = 3 * rho_cc * PI * (Dc / 2) ** 2 * t_inj

-- Cooling Jacket mass estimation
    M_rc = rho_rc * L_rc * (t_ch_rc*t_rc*n_ch_rc + PI*Dc*ti_rc)

-- Propulsant mass  ("tb" is the burning time, to be calculated implicitly by
    M_prop = (m_oxi + m_red) * tb  --   m_oxy, m_red are the propellant mass flows
    ...
-- Total mass (including pay load, M_PL)
    M_o = M_PL + M_prop + M_cc + M_nz + M_rc + M_inj + M_ta + M_tp + M_mis
```

Of course, in a real case, this function must be adjusted to more realistic data, but the procedure when coupling this function with the cycle performances should not change. The above mass relations are encapsulated in the "massExpander" function.

New variables and equations can be added to the schematic diagram using the "Edit source code" button from the "Schematic view". In this case, we add the call of the "masses" function to the CONTINUOUS block of the model:

```
...
CONTINUOUS
        Thrust = Chamber.meas_out.signal[1]
        Isp = Chamber.meas_out.signal[2]

        massExpander(sigma_u_cc, sigma_u_nz, sigma_tank, rho_cc, rho_nz, rho_rc, rho_tank, Pc, Thrust,
        Isp,    Chamber.meas_out.signal[3],    Pump_O2.f1.P,    Pump_O2.sh_in.T,    Pump_O2.f1.m,
        Pump_O2.sh_in.omega, Pump_H2.f1.P, Pump_H2.sh_in.T, Pump_H2.f1.m, Pump_H2.sh_in.omega,
        M_PL, tb, Dt, Chamber.Lc, Chamber.Ld, AR, RC.n_ch, RC.w_ch, RC.t_ch, RC.th_i, RC.Ld2 - RC.Ld1,
        Mprop, M_inj, Mcc, Mnz, Mrc, M_tp, M_ta, M_o)
        ...

END COMPONENT
```

Once the masses have been estimated, the Tsiolkovsky equation gives a relation between the mission requirements, the specific impulse and the initial and final mass of the rocket.

The maximum acceleration requirement gives a new equation for the determination of the engine thrust that becomes coupled with the determination of the chamber diameter and the other requirements:

These relations are added into the CONTINUOUS block of the model after the mass function call:

```
-- Tsiolkovsky eq. relating mission requirement
    M_o = (M_o - Mprop) * exp(Delta_v / Isp)

-- Maximum acceleration eq.
    Accel_max = Thrust / (M_o - Mprop) / 9.806
```

In our example, the mission requirements, which are used in the mass relations, are declared as data in the model source editor:

- DeltaV = 2500 m/s
- M_PL = 7500 kg
- Max accel. = 4 m/s2

By imposing these requirements directly in a *design* partition of the model, EcosimPro is able to calculate the geometry and the corresponding engine performances. As described in previous sections, in design partitions some data can be converted into new unknowns; for example, in this case, Dt is selected as a

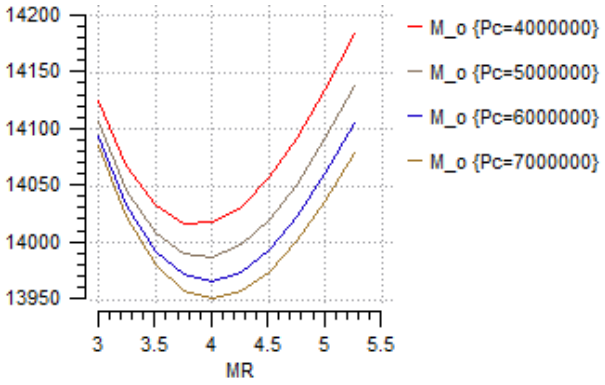
new unknown to be designed. Then a new equation, the maximum acceleration condition, must be imposed. In this example, the other mission requirements are already data in the model.

In the experiment herein, a parametric study has been performed to determine the evolution of the total mass depending on the imposed chamber pressure Pc and mixture ratio MR:

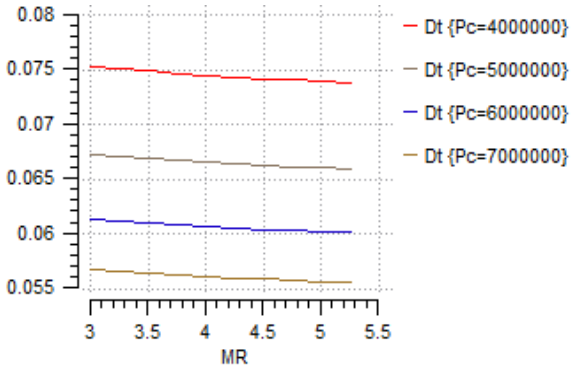
```
EXPERIMENT exp1 ON expanderEngine_mission.design
DECLS
INIT
    -- initial values for algebraics
BOUNDS
        STEADY.GRAV = 9.806
        STEADY.dp_lam = 3000
        VBPH.s_pos.signal[1] = 1
        VBPO.s_pos.signal[1] = 1
        VCH.s_pos.signal[1] = 1.
        VCO.s_pos.signal[1] = 1.
        Accel_max = 0.4
BODY
        M_PL = 7500
        Delta_v = 2500
        Chamber.P_ext = 100

    -- algebraic variable initialisation
        Dt = 0.07
        tb = 1000
        VBPH.f1.m = 0.2
        m_fu = 1
        m_ox = 5
        RC.t_ch = 0.008
        RC.th_i = 0.008

    -- Some data can be redefined here …
        RC.mat_i = Copper -- SS_321 --
        RC.Tw_throat = 1066
        RC.w_ch = 0.002 -- 0.004

    -- Perform the calculation for several chamber pressure and MR
        FOR(j IN 1,4)
            Pc = 40e5 + (j-1)*10e5
            NEW_BRANCH("Pc=$Pc")
            FOR(i IN 1,12)
                MR = 3 + (i-1)*0.25
                STEADY()
            END FOR
        END FOR

END EXPERIMENT
```

The results are plotted in the following figures:

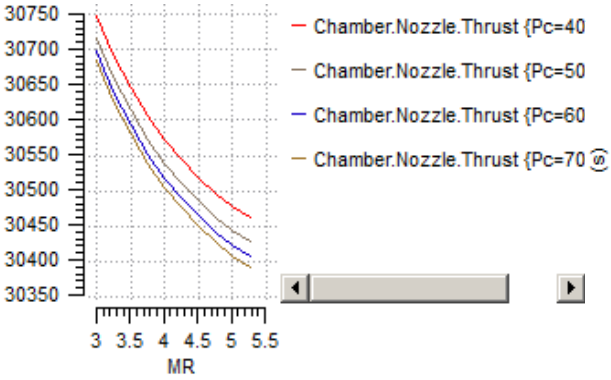
- A minimum engine mass can be found at an MR of 4, less than the typical value of a GG engine
- The greater the chamber pressure, the lower the engine mass. This has the limitation of positive mass flow by the bypass valves



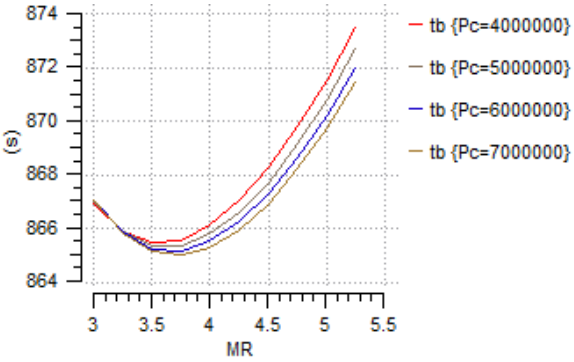
Total engine mass (kg)



Throat diameter (m)



Thrust (N)



Burning time (s)

APPENDICES

A1. CRITICAL FLOW CALCULATION

References

[1] IEC 60534 2 1. Industrial process control valves. Part 2-1: Flow capacity. Sizing equations for fluid flow under installed conditions. September -1998

The critical mass flow can be calculated in two ways depending on the Gcr_ideal option:

- Gcr_ideal = TRUE → Ideal (theoretical) critical flow expansion,
- Gcr_ideal = FALSE → Empirical correlation

A1.1 EMPIRICAL CORRELATION: *FL_GCRIT_FUN* FUNCTION

This function determines the critical flow per unit of flow area using approximated correlations. Liquid, vapor or two-phase flow regimes are considered:

A/ Vapor or two-phase flow. The compressible choking is calculated as follows:

$$(\rho \mathrm{v})_{crit,compr} = \rho c \left(\frac{2 + (\gamma - 1) M^2}{\gamma + 1} \right)^{(\gamma+1)/2/(\gamma-1)}$$

$$M = v / c; \qquad \gamma = \frac{\rho c^2}{P}$$

where ρ, v,c are the density, velocity and sound speed of the upstream conditions, which have been calculated by the corresponding upstream volume.

B/ Liquid flow. The sub-cooled choking is calculated as follows (only for real fluid and perfect liquids), see Reference [1], Sections 6.1.2 and 8.4:

$$(\rho \mathrm{v})_{crit,subcooled} = \sqrt{2 \cdot \rho \cdot (\min(P_{crit}, P) - \alpha \max(P_{nc}, P_{sat}))}$$

$$\alpha = 0.96 - 0.28 \sqrt{\max(0, \min(1, P_{sat} / P_{crit}))}$$

P_{sat} is the saturation pressure upstream as calculated by the state functions. The sub-cooled choking is assumed to be zero for perfect gases.

It is noted that the sub cooled critical mass flow is also used under supercritical pressures to have a continuous expression when passing through the critical pressure, which is fundamental to integrating the fluid conservation equations.

To preventunrealistic values, the actual pressure is limited to the critical one. Otherwise, the effects of non-condensable gases are taken into account by using the maximum of the non-condensable partial pressure and the vapour pressure.

The effective critical mass flow will be evaluated as:

$$(\rho \mathrm{v})_{crit} = \left((\rho \mathrm{v})^8_{crit,compr} + (\rho \mathrm{v})^8_{crit,subcooled} \right)^{0.125}$$

A1.2 IDEAL (THEORETICAL) CRITICAL FLOW EXPANSION: *GCR_CAL* FUNCTION

This function determines the critical flow per unit of flow area by calculation of isentropic expansions until the flow reaches the sonic speed for any upstream conditions. This method needs to iterate and to calculate what the conditions (liquid, gas or two-phase) will be at the throat.

Knowing the upstream conditions of the throat, the inlet entropy can be calculated:

$$s_{up} = FL\_prop\_vs\_ph(P_{up}, h_{up} - 0.5vel_{up}^2, fprop\_entropy)$$

where "*FL_prop_vs_ph*" is one of the functions available in the FLUID_PROPERTIES library. h_{up}, vel_{up} and P_{up} are the upstream conditions determined by the state of the nearest fluid volume to the throat.

Because supersonic conditions are allowed in pipes,vel_{up} is limited to the sound speed when evaluating the upstream entropy in the previous equation.

The critical flow will be calculated by iterating critical pressure (P_{crit}) in the throat at constant entropy (s_{up}). Then, the critical throat conditions (T_{crit}, h_{crit}, ρ_{crit}, …) including the sound speed (c_{crit}) can be calculated from the FLUID_PROPERTIES functions using variables P_{crit} and s_{up} as input.

If the critical throat conditions are two-phase flow (and this is determined by the property functions, see §4.2.1, "*thermo_prop*" function), the following expression for the sound speed (continuous at phase changes) will be used instead (see §4.4.5.4):

$$c_{crit}^2 = \frac{T_{crit}\left(dP/dT / \rho_{crit}\right)^2}{x\left(\varepsilon \cdot c_{p,v} - T_{crit}\, dP/dT\, v_v((1+\varepsilon)\beta_v - \kappa_v\, dP/dT)\right) + (1-x)\left(\varepsilon \cdot c_{p,l} - T_{crit}\, dP/dT\, v_l((1+\varepsilon)\beta_l - \kappa_l\, dP/dT)\right)}$$

$$dP/dT = \frac{x c_{p,v} + (1-x)c_{p,l}}{T_{crit}(x\beta_v v_v + (1-x)\beta_l v_l)}$$

$$\varepsilon = 0 \quad \Rightarrow \text{"frozen" sound speed}$$

Indices "v" and "l" indicate vapor and liquid saturated conditions. "v" is the specific volume and "x" is the quality (vapor mass fraction). cp, β, κ, are the specific heat, the volumetric expansivity and the isothermal compressibility. All these variables are returned by FLUID_PROPERTIES library function "*thermo_prop*", see §4.2.1.

The following closing condition must be satisfied (total enthalpy conservation):

$$c_{crit} = \sqrt{2(h_{up} - h_{crit})}$$

Finally, the critical mass flow per unit of area will be evaluated as:

$$(\rho v)_{crit} = \rho_{crit} c_{crit}$$

A2. PRESSURE LOSS FUNCTIONS

References

[1] Idelchik, I.E.; Handbook of Hydraulic Resistance, 3rd edition, Research Institute for Gas Purification, Moscow, Russia, 1994.

[2] Engineering Data Book III. Wolverine Tube. Inc

A2.1 FRICTION FACTOR CALCULATION. FUNCTION HDC_FRIC

The function "hdc_fric" incorporates the evolution of the friction factor as a function of the local Reynolds number ($\rho vD/\mu$) and the relative roughness.

The friction factor (f) is calculated by means of a simple correlation valid for laminar, turbulent and transient flow.

$$f = 8\cdot\left[\left[\frac{8}{\mathrm{Re}}\right]^{12} + \frac{1}{(A+B)^{3/2}}\right]^{\frac{1}{12}}$$

where

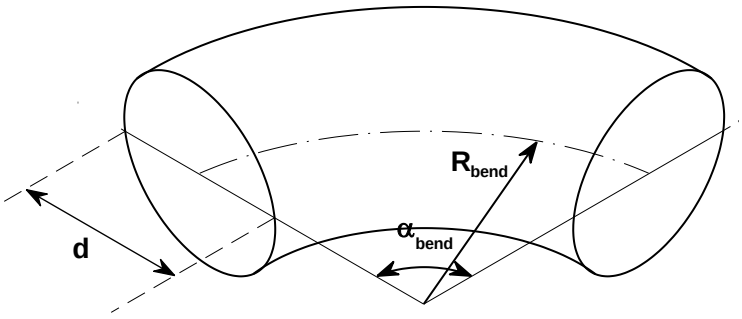
$$A = \left[2.457 \ln \frac{1}{(7/\mathrm{Re})^{0.9} + 0.27\varepsilon/D}\right]^{16}$$

$$B = \left[\frac{37530}{\mathrm{Re}}\right]^{16}; \qquad \varepsilon = roughness$$

A2.2 ELBOW PRESSURE LOSS FUNCTION

This function calculates the bend pressure drop coefficient. It depends on the relative radius of curvature, R_{bend}/D, the relative roughness, R/D, and the bend angle, α:

According to Idelchik, I.E [1], ("Handbook of Hydraulic Resistance), the total resistance coefficient of pipe bends is the product of the following coefficients:



a/ angle effect:

$$\xi_{angle} = 0.957\frac{\alpha}{90} + 0.226\sqrt{\frac{\alpha}{90}} + 0.407\sin(\alpha) - 0.833\sin(\alpha/2)$$

b/ radius effect:

$$\xi_{radius} = 0.21/\sqrt{\frac{R}{D}} \qquad (R/D > 1)$$

$$0.21/(\frac{R}{D})^{2.5} \qquad (R/D < 1)$$

c/ roughness effect:

$$\xi_{rug} = \quad \min(\ 2{,}1 + 10^6 (\frac{Rug}{D})^2) \qquad (R/D > 1.5)$$

$$\min(\ 2{,}1 + 10^3 \frac{Rug}{D}) \qquad (R/D < 1.5)$$

Thus, the pressure drop coefficient is:

$$\xi_{bend} = \xi_{angle} \cdot \xi_{radius} \cdot \xi_{rug}$$

A2.3 TWO-PHASE FRICTION FACTOR CALCULATION. FRIEDEL CORRELATION

The following formulation is taken from reference 2:

The correlation method by Friedel (1979) uses a two-phase multiplier:

$$\Delta p_{frict} = \Delta p_L \ \Phi_{fr}^2$$

where Δp_L is calculated for the liquid-phase flow as:

$$\Delta p_L = 4 f_L \left(\frac{L}{d_i}\right) \dot{m}_{total}^{\ 2} \left(\frac{1}{2\rho_L}\right)$$

The liquid friction factor f_L and liquid Reynolds number are obtained from

$$f = \frac{0.079}{\mathrm{Re}^{0.25}} \qquad \mathrm{Re} = \frac{\dot{m}_{total} d_i}{\mu}$$

Using the liquid dynamic viscosity μ_L. His two-phase multiplier is

$$\Phi_{fr}^2 = E + \frac{3.24FH}{Fr_H^{0.045} We_L^{0.035}}$$

The dimensionless factors Fr_H, E, F and H are as follows:

$$Fr_H = \frac{\dot{m}_{total}^{\ 2}}{g d_i \rho_H^2}$$

$$E = (1-x)^2 + x^2 \frac{\rho_L f_G}{\rho_G f_L}$$

$$F = x^{0.78}(1-x)^{0.224}$$

$$H = \left(\frac{\rho_L}{\rho_G}\right)^{0.91}\left(\frac{\mu_G}{\mu_L}\right)^{0.19}\left(1-\frac{\mu_G}{\mu_L}\right)^{0.7}$$

The liquid Weber We_L is defined as

$$We_L = \frac{\dot{m}_{total}^2 d_i}{\sigma \rho_H}$$

Using the following alternative definition of the homogeneous density ρ_H based on vapor quality:

$$\rho_H = \left(\frac{x}{\rho_G} + \frac{1-x}{\rho_L}\right)^{-1}$$

A3. FILM COEFFICIENT CALCULATION

References

[1] Chen, J.C., 1966, "A correlation for Boiling Heat Transfer to Saturated Fluid in Convective Flow" I.Eng. Chem. Process Des. Dev., 5, pp. 322-329.

[2] Dougall, R.S., Rohsenow, W.H.; Film Boiling on the inside of the inside of the Vertical Tubes with Upward Flow of the Fluid at Low Qualities, M.I.T. Report No. 9079.26, (1963)

A3.1 PIPES

The calculation of the heat exchange coefficient in pipes takes into account two-phase regimes. Four main possibilities (HT_option) are foreseen calculating the heat exchange coefficient for Pipes: Use *ht_option=* HT_tube to take into account one- and two-phase regimes (boiling and condensation regimens at the wall/fluid interfase are included). Use the HT_tube_1ph option to consider one-phase correlations, even under two-phase flow. If necessary, use HT_critCond when passing near the critical point. Use HT_boiling for film boiling correlation in particular *vertical* pipes if necessary (cases of low flow):

*A/ Single phase correlation. (**HT_tube_1ph** option)*

Laminar and turbulent Nusselt numbers:

$$Nu_{lam} = 4$$

$$Nu_{tur} = 0.023\,Re^{0.8}\,Pr^{0.4}$$

Where, the Prandtl and Reynolds numbers are:

$$Re = \rho\ vel\ dia\ /\ vis$$

$$Pr = Cp\ vis\ /\lambda$$

The equivalent Nusselt number for pipes covering transitions zones is:

$$Nu = (N_{lam}{}^{16} + Nu_{tur}{}^{16})^{1/16}.$$

Then, the single phase film coefficient is calculated as follows:

$$h_{sp} = Nu(\lambda / D)$$

λ is the fluid conductivity.

Note that the properties used to evaluate Re, Pr, etc. are those of the two-phase mixture in case of two-phase flow. There may be some problems when passing from supercritical to two-phase flow, especially near the critical point. Indeed, at the critical point some property derivatives are not defined and do not have finite values. Thus, the correlations using Cp run the risk of having discontinuities. If that happens, the simulation becomes slow, and it is advised to use ht_option = ***HT_critCond***, so the heat exchange coefficient is calculated at Pr=1, without dependence on the cp value.

*B/ Two-phase correlations. (**HT_tube** option)*

B1/ Condensation (two-phase or vapor and $T_{wall} < T_{sat}$). This method is based on Boyko & Kruzhilin's correlation, appropriate for film-wise condensation in uniform channels under forced convection conditions:

$$h_{cond} = h_{sp}\sqrt{1 + x(\rho_l / \rho_g - 1)}; \quad x : mean\ quality$$

$$\rho_l, \rho_g : saturated\ liquid\ and\ vapor\ density$$

B2/ Superheated Condensation (Quality =1 and $T_{wall} < T_{sat}$). The method used for the calculation of the heat transfer coefficient is shown below. The method was chosen in order to keep continuity between single and two phase regimes:

$$h = \frac{h_g (T_{sat} - T_{wall}) + h_{cond} (T - T_{sat})}{(T - T_{wall})}$$

where h_g is the vapor single phase film coefficient. h_{cond} is the condensation correlation film coefficient considering the actual pressure saturation properties.

B3/ Boiling (Quality < 0.7 and $T_{wall} > T_{sat}$). According to Chen [1], for vapor quality < 0.7 and if the stratification is not severe, several steps must be followed in order to calculate the film coefficient in vaporization conditions:

First of all, the convective film coefficient for liquid must be calculated:

$$h_l = 0.023\ Re_l^{0.8} Pr_l^{0.4} \lambda_l / D$$

where the sub index "l" makes reference to saturated liquid conditions. The inverse of the Lockhart-Martinelli parameter is calculated:

$$1/X_{tt} = (x/(1-x))^{0.9} (\rho_l / \rho_g)^{05} (\mu_g / \mu_l)^{0.1}$$
$$\mu_g, \mu_l : saturated\ vapour\ and\ liquid\ viscosity$$

Then, the convective boiling contribution is calculated as follows: $h_{con} = F\ h_l$, where:

$$F = 2.35\,(1/X_{tt} + 0.213)^{0.736} \quad (1/X_{tt} > 0.1)$$
$$= 1 \quad (1/X_{tt} < 0.1)$$

The nucleate boiling contribution is calculated as follows:

$$h_{nuc} = B\,(T_{wall} - T_{sat})^{0.24} (P_{sat,Twall} - P_{sat})^{0.75},$$

where:

$$B = 0.00122 \frac{\lambda_l^{0.79} cp_l^{0.45} \rho_l^{0.49} S}{\sigma^{0.5} \mu_l^{0.29} (\rho_g (h_g - h_l))^{0.24}}$$

$$S = \frac{1}{1 + 2.53e^{-6}\, Re_{2ph}^{1.17}}, \quad Re_{2ph} = Re_l\, F^{1.25}$$

where σ is the surface tension; h_g - h_l is the vaporization latent heat. Finally, the combined boiling film coefficient is $h_{Chen} = h_{con} + h_{nuc}$

B4/ Boiling (Quality > 0.9 and $T_{wall} > T_{sat}$). For vapor quality x >0.9 a post-dry-out correlation due to Dougall & Rohsenow [2] is used:

$$h_g = 0.023\ Re_g^{0.8} Pr_g^{0.4} \lambda_g / D$$
$$\Phi = x + (1 - x)\, \rho_g / \rho_l$$
$$h_{DR} = h_g \Phi$$

B5/ Boiling (0.7<Quality < 0.9 and $T_{wall} > T_{sat}$). For vapor quality 0.7 < x < 0.9, cubic spline interpolation is performed between the Chen & Dougall - Rohsenow correlations.

B6/ Subcooled Boiling (Quality =0 and $T_{wall} > T_{sat}$). The method used is to calculate the heat transfer coefficient as to ensure continuity between the single and two phase regimes:

$$h = \frac{h_l (T_{sat} - T_l) + h_{Chen} (T_{wall} - T_{sat})}{(T_{wall} - T)}$$

where h_l is the liquid single phase film coefficient. h_{Chen} is the Chen correlation film coefficient considering the actual pressure saturation properties.

*C/ Film boiling correlation (**HT_boiling** option)*

Use HT_boiling for film boiling correlation in particular vertical pipes if necessary (cases of low flow). This option only applies if Twall > Tsat and Tfluid<=Tsat. The same formulation as in appendix 0 for Tanks will be used. If Twall <= Tsat or Tfluid > Tsat, the one phase correlation (*HT_tube_1ph*) will be used.

A3.2 TANKS

1/. *Forced convection*. Laminar and turbulent Nusselt numbers:

$Nu_{lam} = 4;$

$$Nu_{tur} = 0.023\ Re^{0.8} Pr^{0.4}$$

The Nusselt number for a vessel with an entering jet is:

$$Nu_{jet} = 2.2\ \left(\frac{w_{in}}{\pi D \mu} \cdot Pr\right)^{0.632}$$

where w_{in} is the entering mass flow.

The forced convection Nusselt number is:

$$Nu_f = (Nu_{lam}^{16} + Nu_{tur}^{16} + Nu_{jet}^{16})^{1/16}$$

2/. *Natural convection*. Calculation of the Grashoff, Rayleigh and Nusselt numbers:

$$Grashoff = 2 \cdot g \cdot D^3 (T_{wall} - T) \cdot \frac{(\rho/\nu)^2}{T_{wall} + T}$$

$$Rayleigh = Grashoff \cdot Prandtl$$

$$Nu_{n1} = 2. + 0.392 Grashoff^{0.25} \quad (Rayleigh < 1.e5)$$

$$Nu_{n2} = 0.52\ Grashoff^{0.25} \quad (1.e5 < Rayleigh < 1.e9)$$

$$Nu_{n3} = 0.13\ Grashoff^{0.3333} \quad (Rayleigh > 1.e9)$$

The natural convection Nusselt number covering transitions zones is:

$$Nu_n = (Nu_{n1}^{16} + Nu_{n2}^{16} + Nu_{n3}^{16})^{1/16}$$

3/ The equivalent film coefficient for tanks covering transitions zones is

$$Nu = (Nu_n^{16} + Nu_f^{16})^{1/16}$$

$$h_{tank} = Nu\ (\lambda / D)$$

A4. HEAT AND MASS TRANSFER INSIDE TANKS

The following sections provide an overview of the heat and mass transfer process and describe the various phenomena involved. The phenomena expected inside a propellant or pressure tank are closely related to the upper stages currently in production in the Ariane 5 program.

References

[1] Ostrach, S., "Natural Convection in Enclosures. Advances in Heat Transfer, 8:161–227, 1972

[2] VDI-Gesellschaft Verfahrenstechnik und Chemieingenieurwesen (Herausgeber), "VDI-Wärmeatlas, Band 8. Springer-Verlag, BerlinHeidelberg,1997

[3] Becker, M., "Heat Transfer – A Modern Approach, New York: Plenum Press, 1986"

[4] Baehr, H. D. und K. Stephan, "Wärme- und Stoffübertragung. Springer Verlag, Berlin, 1996

[5] Bird, B. et. al., "Transport Phenomena. New York - London, John Wiley & Sons, 1960

[6] Carey, van P., "Liquid-Vapor Phase-Change Phenomena. Hebron, Taylor and Francis, 1992

[7] Lienhard, J. H., "Corresponding stats correlations for the spinoidal and homogeneous nucleation temperatures, J. Heat Transfer, 104,379-382,1982

A4.1 CONVECTIVE HEAT TRANSFER

Energy transfer occurs within a moving fluid not only by conduction but also due to its motion. As a result, the energy is not only transferred by conduction, but also by the transported enthalpy and kinetic energy of the fluid. Therefore, the convective heat transfer is a combination of conduction and transport of energy.

In practice, basically, the heat transfer between fluid and walls is important. Considering a plain wall with the temperature ϑ_w and a fluid with the bulk temperature ϑ (not the temperature inside the boundary layer) moving along the wall, the resulting heat flux density is proportional to the difference between the two temperatures.

$$\dot{q}_w = -\alpha\left(\vartheta - \vartheta_w\right)$$

The proportionality factor α is defined as the heat transfer coefficient.

Close to the wall where the fluid has no velocity, the energy can only be transported by conduction. Therefore, the heat flux density at the wall can be derived:

$$\alpha = -\lambda \frac{\left(\frac{\partial \vartheta}{\partial y}\right)}{\vartheta - \vartheta_w}$$

To calculate the heat transfer coefficient, the temperature field needs to be known. This requires accurately knowing the velocity field. Because both the flow field as well as the temperature field are described by differential equations that are only solvable in very simple cases, it is obvious that exact models of the heat transfer coefficient only exist for those cases. In the most realistic cases the heat transfer coefficients need to be evaluated with experimental results.

The convective heat transfer is more effective than pure conduction due to the motion of fluid. This motion is induced either by a forced flow or by a density gradient driven acceleration field.

Equations for the heat transfer calculation for free and forced convection are described next with special emphasis on the application. For that purpose the Nusselt number Nu is defined, which is basically a dimensionless temperature gradient averaged over the heat transfer surface. It represents the relation between convection and conduction at the same temperature difference.

$$Nu = \frac{\alpha \cdot l}{\lambda}$$

Consequentially, the heat transfer coefficient is determined by the equation:

$$\alpha = \frac{Nu \cdot \lambda}{l}$$

In this context the characteristic length l is the length of the considered wall segment in case of Nu as the Nusselt number or the overflow length of the flow in case of Nu as the local Nusselt number.

In the following, the Nusselt number is given as a function of dimensionless parameters. Those parameters are the Prandtl number Pr:

$$\mathrm{Pr} = \frac{\eta\, c_p}{\lambda}$$

the Grashoff number Gr:

$$Gr = \frac{g\, l^3\, \beta \left| T - T_w \right|}{\nu^2}$$

and for forced flow the Reynolds number Re:

$$\mathrm{Re} = \frac{w\, l\, \rho}{\eta}$$

with w as the flow velocity of the bulk flow. Sometimes the product of Prandtl and Grashoff number is used as the so-called Rayleigh number Ra:

$$Ra = Gr\, \mathrm{Pr}$$

The surface temperature T_w of the inner tank surface varies with time due to the temperature increase/decrease during compression/expansion, respectively, and additionally, by external heat loads. The temperature difference between the surface boundary layer and the bulk gives a density difference which induces Helium convection flow inside the vessel. The resulting heat transfer can be generally described with the help of the Nusselt theory basing on experimental results.

The heat flux Q_w is due to heat exchange between ullage and the tank surface by natural or forced convection:

$$\dot{Q}_w = -\alpha\, A\, (T_g - T_w)$$

- *Free convection between internal wall surface and fluid:*

Detailed research in the field of free convection inside rectangular vessels is described in [1]. Experimental researches have shown that the mechanisms of heat transfer for that configuration can be divided into three parts. Depending on the Grashoff number, the heat transfer inside a vessel is caused either by heat conduction or by boundary layer flow. Those two main regions are linked by a transition region. In addition, it turned out that the ratio between the height l and the width d of the vessel influences the validity of these regions.

For pressurant tanks and propellant tanks, it is proposed to apply the correlations for free convection inside spherical vessels to calculate the heat transfer within the liquid and gas part for tanks [2]. The proposed equations are:

$Nu_{free} = 5.9$ for $Ra < 10^4$,

$Nu_{free} = 0.59 \cdot Ra^{0.25}$ for $10^4 < Ra < 10^9$

$Nu_{free} = 0.13 \cdot Ra^{\frac{1}{3}}$ for $10^9 < Ra < 10^{12}$

In addition, the following correlation is given in this context:

$Nu_{free} = 0.098 \cdot Ra^{0.345}$ with a validity range given for $3 \cdot 10^8 < Ra < 5 \cdot 10^{11}$.

In case of a spherical pressurant vessel, the diameter d of the tank is used as characteristic length l. For the ullage or liquid part of a propellant tank, the respective volume V is used to derive the diameter d of an equivalent large sphere:

$$d = 2\left(\frac{3V}{4\pi}\right)^{\frac{1}{3}}$$

- *Free convection between ullage and liquid at phase interface:*

The correlations take account of the influence of the Prandtl number on the flow along a horizontal plate.

For a horizontal plate with heat dissipation on the upside or cooling on the bottom side the following correlation is given for the *laminar* range

$Nu = 0.766 \cdot \left(Ra \cdot f\left(\mathrm{Pr}\right)\right)^{\frac{1}{5}}$ $\left(Ra \cdot f\left(\mathrm{Pr}\right)\right) < 7 \cdot 10^4$

and for the *turbulent* range, respectively.

$Nu = 0.15 \cdot \left(Ra \cdot f\left(\mathrm{Pr}\right)\right)^{\frac{1}{3}}$ $\left(Ra \cdot f\left(\mathrm{Pr}\right)\right) > 7 \cdot 10^4$

where $f\left(\mathrm{Pr}\right) = \left(1 + \left(\frac{0.322}{\mathrm{Pr}}\right)^{\frac{11}{20}}\right)^{-\frac{20}{11}}$

and $0 < \mathrm{Pr} \le \infty$. The inverse case, a horizontal plate with heat dissipation at the bottom side and cooling on the upside, is at laminar conditions for the range $10^3 < \left(Ra \cdot f\left(\mathrm{Pr}\right)\right) < 10^{10}$ characterized by

$Nu = 0.6 \cdot \left(Ra \cdot f\left(\mathrm{Pr}\right)\right)^{\frac{1}{5}}$ using the same function $f\left(\mathrm{Pr}\right)$.

- *Forced convection:*

The phenomenon of forced convection can be divided into two parts. On the one side, the region of laminar convection flow exists where the conservation equations are valid and solvable for simple cases. In the second region with turbulent flow, the heat transfer in the boundary layer is increased by the turbulent fluctuation motion of the fluid and additional assumptions have to be considered for solving the conservation equations.

- *Laminar forced convection:*

The exact solution [3] of the laminar forced convection is given by

$$Nu_{lam} = 0.664 \cdot \sqrt{\mathrm{Re}} \cdot \sqrt[3]{\mathrm{Pr}} \cdot f(\mathrm{Pr})$$

This equation is valid for $0.6 < \mathrm{Pr} < 2000$ in case of flow towards a plane plate and perpendicular across tubes, wires, etc. and $5 \cdot 10^5 < \mathrm{Re} < 10^7$. The function $f(\mathrm{Pr})$ is given by following tabled values:

Pr	$f(\mathrm{Pr})$
0,01	0,7194
0,1	0,9092
0,7	0,9922
1	1
10	1,0179
100	1,0199

A 3rd order polynomial fitting function may be used as given next:

$$f(\mathrm{Pr}) = 10^{0.99831 + 0.04715 \log(\mathrm{Pr}) - 0.03232 \log^2(\mathrm{Pr}) + 0.007 \log^3(\mathrm{Pr})}$$

- *Turbulent forced convection:*

If the Reynolds number exceeds 10^5, the laminar convection changes to the turbulent one. For a Reynolds number $5 \cdot 10^5 < \mathrm{Re} < 10^7$, the Nusselt number has to be calculated by:

$$Nu_{turb} = \frac{0.037 \cdot \mathrm{Re}^{0.8} \cdot \mathrm{Pr}}{1 + 2.443 \cdot \mathrm{Re}^{-0.1} \left(\mathrm{Pr}^{2/3} - 1\right)}$$

This equation is valid for a Prandtl number $0.6 < \mathrm{Pr} < 2000$ in case of flow towards a plane plate and perpendicular across tubes, wires, etc. The fluid properties need to be calculated using a mean value of the wall and liquid temperature as already required for the laminar forced convection.

- *Transition region:*

Within the transition region between laminar and turbulent convection a good agreement with experimental results can be obtained by quadratic superposition of the equations of laminar and turbulent convection.

$$Nu_{forced} = \sqrt{Nu_{lam}^2 + Nu_{turb}^2}$$

Because values of Nu_{lam} are negligible within the turbulent region and vice versa, this superposition equation can be used in the entire region $10 < \mathrm{Re} < 10^7$.

- *Superposition of free and forced convection:*

In case of superposition of free and forced convection at inclined walls the following approach is used. If the free and the forced convection are of same direction, the Nusselt number is given by

$$Nu = \sqrt[3]{Nu_{free}^3 + Nu_{forced}^3}$$

This equation is valid for $0.1 < \mathrm{Pr} < 100$. If free and forced flow is of opposite direction the Nusselt number is calculated by

$$Nu = \sqrt[3]{Nu_{forced}^3 - Nu_{free}^3} \qquad \text{if } \frac{Nu_{free}}{Nu_{forced}} < 0.8$$

A4.2 MASS TRANSFER IN THE ULLAGE OF THE TANK

- *Correlation between heat and mass transfer coefficients*

The solutions of many mass-transfer problems at low mass transfer rates can be obtained by analogy with corresponding problems in heat transfer. Therefore, a presentation of new correlations is not necessary, but only a conversion into the analogous mass-transfer correlations. That is, to obtain concentration profiles instead of temperature profiles; the Prandtl number Pr is replaced with the Schmidt number Sc and T by c:

$$Nu = f\,(Re, Pr, \frac{d}{l}, ...)$$

$$Sh = f\,(Re, Sc, \frac{d}{l}, ...)$$

Note that this analogy assumes (i) constant physical properties, (ii) a small rate of mass transfer, (iii) no chemical reactions in the fluid, (iv) no viscous dissipation, (v) no emission or absorption of radiant energy, and (vi) no pressure diffusion, thermal diffusion or forced diffusion.

For free convection around submerged objects of any given shape it can be similarly shown that:

$$Nu = f\,(Gr, Pr)$$
$$Sh = f\,(Ar, Sc)$$

in which Ar is the Archimedes number for binary diffusion. In the application for the propellant tank model, the following equations for the Sherwood number Sh are used.

The mass flow of vapour transferred to the phase boundary depends on both the pressure difference between the partial pressure p_v of the vapour and the total pressure p in the ullage and the pressure difference between the partial pressure p_s of the vapour at the phase boundary and the total pressure p in the ullage. The mass transfer is defined by:

$$\dot{m} = \rho_v \cdot \beta \cdot A \cdot \ln\left(\frac{p - p_s}{p - p_v}\right)$$

where ρ_v is the density of the condensable component at total pressure p and temperature T in the ullage. The mass transfer coefficient β can be approximately traced back to the heat transfer coefficient using Lewis' correlation [4].

$$\beta = \frac{\alpha}{c_{pv} \cdot \rho_v} \cdot Le^{-2/3}$$

The following definition gives the relation to between heat and mass transfer coefficients:

$$Le^{\frac{1}{3}} = \frac{Sh}{Nu} = \frac{\beta\lambda}{\delta\alpha}$$

Therefore, the Lewis number can be determined by

$$Le = \frac{a}{\delta} = \frac{Sc}{\mathrm{Pr}}$$

where a is the thermal conductivity and δ the binary diffusivity. The Lewis number is the ratio between the Schmidt number and the Prandtl number and is used in case of laminar free convection.

The heat of evaporation is expressed by:

$$\dot{Q}_{ev} = \dot{m}_{ev}\ \Delta h_v$$

The heat of condensation is expressed by:

$$\dot{Q}_{co} = \dot{m}_{co}\ \left(h_v - h_l \right)$$

- *Diffusion coefficient for binary gas mixtures*

The mass diffusivity δ for a binary system is a function of temperature, pressure, and composition, whereas the dynamic viscosity and thermal conductivity for a pure gas are functions of temperature and pressure. The data available on the diffusivity for most binary mixtures are, moreover, quite limited in range and accuracy. The available correlations of the diffusivity are limited of scope and are based more on theory than on experiment.

For binary gas mixtures at low pressures, the following equation for estimation of δ at low pressures has been developed [5]:

$$\delta_{12} = 2.6622 \cdot 10^{-2} \cdot \frac{\sqrt{T^3 \frac{\tilde{M}_1 + \tilde{M}_2}{2\,\tilde{M}_1 \tilde{M}_2}}}{p\,\sigma_{12}^2\,\Omega_{12}}$$

In the foregoing equation, the *Chapman-Enskog* kinetic theory is used. Ω_{12} is a dimensionless function of the temperature and of the intermolecular potential field for one molecule of He and one of N_2O_4. It is convenient to approximate this potential field by the Lennard-Jones function:

$$\varphi_{12}(r) = 4\,\varepsilon_{12}\left[\left(\frac{\sigma_{12}}{r}\right)^{12} - \left(\frac{\sigma_{12}}{r}\right)^{6}\right]$$

The *Lennard-Jones* parameters σ_{12} and ε_{12} could be determined directly from accurate measurements of σ_{12} over a wide range of temperature. Suitable data are, however, extremely rare. Fairly satisfactory the estimations can be made for non-polar, non-reacting molecule pairs by combining the *Lennard-Jones* parameters of species 1 and 2 empirically:

$$\sigma_{12} = \frac{1}{2}\left(\sigma_{11} + \sigma_{22}\right)$$

$$\varepsilon_{12} = \sqrt{\varepsilon_{11} \cdot \varepsilon_{22}}$$

This way it is possible to predict measured values of the binary diffusivity within an average deviation of about 6% by use of viscosity data on pure species 1 and 2 or within about 10% if the *Lennard-Jones* parameters for 1 and 2 are estimated from boiling-point data. The collision potential Ω_{12}^* are the result of the potential energy of equation and are given as a function of T_{12}^*:

$$T_{12}^* = \frac{kT}{\varepsilon_{12}}$$

$$\Omega_{12}^* = 1.43984 \cdot T_{12}^{*\,-0.51626}, \qquad T_{12}^* < 1.7234$$

$$\Omega_{12}^* = 1.24063 \cdot T_{12}^{*\,-0.24267}, \qquad T_{12}^* \leq 5.9475$$

$$\Omega_{12}^* = 1.06603 \cdot T_{12}^{*\,-0.1576}$$

Another approximation to the collision potential is given by *Neufeld et al.:*

$$\Omega_{12}^{*} = \frac{1.06036}{T_{12}^{*\,0.1561}} + \frac{0.193}{\exp\left(0.47635 \cdot T_{12}^{*}\right)} + \frac{1.03587}{\exp\left(1.52996 \cdot T_{12}^{*}\right)} + \frac{1.76474}{\exp\left(3.89411 \cdot T_{12}^{*}\right)}$$

The relevant values for the *Lennard-Jones* parameters are listed in the following table for He, N_2O_4, NO_2, MMH, O_2, H_2, N_2:

fluid	He	N_2O_4	NO_2	MMH	O_2	H_2	N_2
ε_{11} / k	10.22	347	210	347	113	3.8	91.5
σ_{11}	2.576	4.621	3.765	4.621	3.433	2.915	3.681

Lennard-Jones Parameters for the relevant Molecules

A4.3 BOILING HEAT AND MASS TRANSFER IN THE LIQUID POOL OF THE TANK

The convective heat transfer can be characterized using thermodynamic properties like density, viscosity, volume expansion and geometrical properties. In contrast, additional properties that describe the phase transition are relevant for characterization of boiling processes. These properties are the enthalpy of vaporization, the boiling temperature, the vapor density and the surface tension. Furthermore, the condition of the heating surface material with its microscopic structure is important. Due to this large number of driving factors it is more difficult to develop correlations for boiling heat transfer coefficients than for any other type of heat transfer. As no complete theory has been developed for boiling heat transfer yet, only empirical models are developed using experimental investigations [6].

Among the additional properties, the different types of boiling heat transfer complicate the development of such a closed model. The type of boiling heat transfer can be subdivided in the act of flow conducting and the order of magnitude for superheating. Therefore, a classification into pool boiling and flow boiling has been performed in the theory.

Depending on the liquid temperature, the pool boiling can be additionally subdivided into sub-cooled boiling and boiling of saturated liquids. Depending on the type of boiling, the pool boiling and the flow boiling are subdivided into nucleate boiling, film boiling and the transition between nucleate and film boiling.

- *Pool boiling*

In pool boiling the flow is driven by free convection inside the vessel. Therefore, pool boiling is also called boiling in free flow regime. The regimes of boiling are schematically shown in Figure A-1. In this schematic the heat flux density is shown as function of wall superheating. One of the boiling conditions is the fact that the wall temperature exceeds the saturation temperature of the liquid. Within this regime pool boiling can be subdivided into *four* different types:

If the wall temperature is slightly above the saturation temperature, evaporation on the liquid surface takes place and the heat transfer at the wall is driven by free convection, because wall superheating is not strong enough to enable bubble formation.

With increasing wall temperature, the onset of nucleate boiling (ONB) condition will occur (section d-e in Figure A-1; isolated bubble regime). The bubbles leave the heating surface as soon as their diameter exceeds the critical bubble (departure) diameter.

With further increasing of the heat flux fully developed pool boiling occurs (section e-f in Figure A-1; regime of slugs and columns). The heat flux at the wall can be only increased up to a critical value (point f).

With further increasing of the wall temperature the regime of partial film boiling is encountered (section f-g in Figure A-1; transition boiling) which describes the transient regime between nucleate boiling and film boiling. In this regime the heat flux decreases with increasing wall temperature down to the minimum film boiling heat flux which is reached in point g. At film boiling (section g-h in Figure A-1; film boiling)

conditions the liquid is not in direct contact with the heating surface anymore. There is a closed vapor film between heating surface and liquid surface and heat transfer takes place by conduction and radiation.

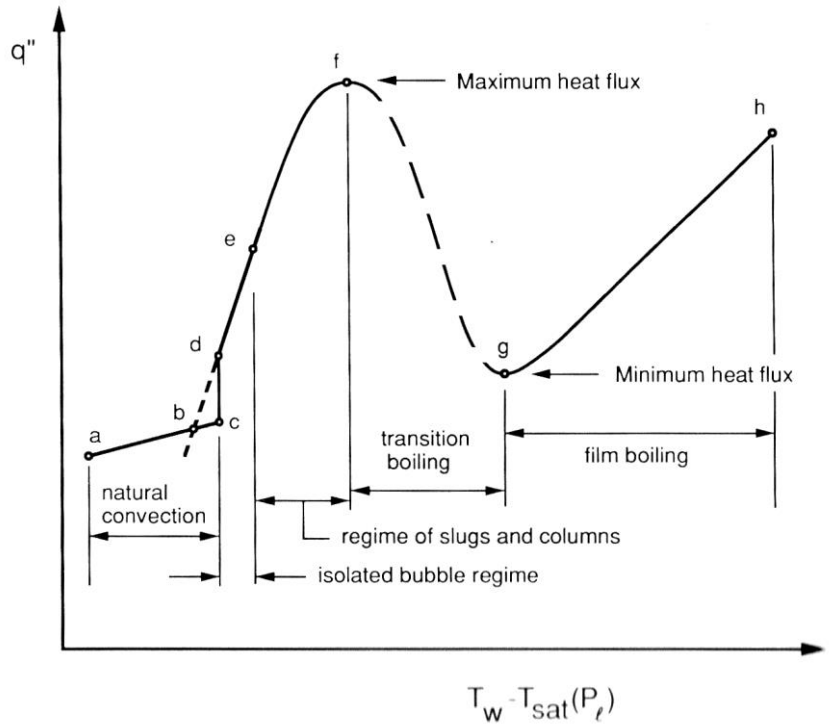


Figure A-1: Pool boiling regimes for an independently controlled surface temperature

- *Quiescent boiling*

The superheating of the wall is defined by $\Delta T_s = (T_w - T_s) \leq (T_{ONB} - T_s)$ in quiescent boiling conditions, but there is no bubble formation at the wall to observe. T_{ONB} is the wall temperature of beginning nucleate boiling. Therefore, the correlations for free convection can be used in case of quiescent boiling. The following correlations are given for determination of the Nusselt number of laminar and turbulent free convection:

$$Nu_{lam} = 0.6 \cdot (Gr \cdot \mathrm{Pr})^{1/4}$$

$$Nu_{turb} = 0.15 \cdot (Gr \cdot \mathrm{Pr})^{1/3}$$

The Grashoff number is built using the characteristic length in this context. It corresponds to the diameter in case of a vertical tube. The transition from laminar to turbulent boundary layer flow is in the region of $10^7 < Gr \cdot \mathrm{Pr} < 10^8$. Previous equations are valid for $2 < \mathrm{Pr} < 100$.

- *Nucleate boiling:* $T_w \leq T_s + \Delta T_{crit}$

Different models and correlations have been proposed to calculate the nucleate boiling heat transfer. These correlations can be subdivided into two groups.On one hand are purely empirical correlations as given in the literature. More widespread, however,are semi-empirical correlations that are based on empirical models but adjusted by experimentally investigated parameters.

Many of the early nucleate boiling models such as the one of *Rohsenow* model assume that the formation and rising of bubbles causes secondary flows that help convective heat transfer at the heating surface. At this type of heat transfer the heat first is transferred from the heating surface to the liquid, just like at any type of convection, and then it is transferred into the liquid driven by the rising of the bubble.

This empirical model advises the postulation of a correlation similar to the forced convection heat transfer. From experimental investigations it is known that the influence of the sub-cooling of the liquid decreases with increasing heat flux. Therefore, the heat transfer coefficient inside the bubble can be written as

$$\alpha = \frac{\dot{q}}{T_w - T_s}$$

As a result, Rohsenow's correlation gives:

$$\dot{q} = \frac{\eta_l \cdot \Delta h_v}{L_b} \cdot \left(\frac{c_{pl} \cdot (T_w - T_s)}{\Delta h_v \, \mathrm{Pr}_l^s \, C_{sf}} \right)^{\frac{1}{r}}$$

where L_b is called the capillary length scale of a bubble:

$$L_b = \left(\frac{\sigma}{g \cdot (\rho_l - \rho_v)} \right)^{\frac{1}{2}}$$

In most cases the constant C_{sf} is determined using experimental results especially determined for the corresponding combination of heating surface and liquid. The same proceeding is applied for the exponents r and s. Within the simulation tool EKSIM the following values are in use:

	LH2	**LOX**	**H_2O polished steel**
C_{sf}	0.01625	0.0149	0.0132
1/r	2.468	2.431	3.03
s	1.7	1.7	1.0

Originally, values of $r = 0.33$ and $s = 1.7$ were recommended for this correlation.

The evaporated mass flow can be calculated by:

$$\dot{m}_{ev} = \frac{\dot{q}}{\Delta h_v}$$

The bubble frequency f is simply obtained by:

$$f = \frac{\dot{m}_{ev}}{\rho_v V_b}$$

- *Onset of nucleate boiling:*

Nucleate boiling of a liquid is only possible up to a maximum heat flux. Further increase of the heat flux results in the onset of nucleate boiling, i.e. non wet areas at the heating surface will be formed which in turn results in a reduction of the heat flux.

The maximum heat flux at nucleate boiling is called "critical heat flux" (CHF). It can be determined using the correlation of Kutateladze and Zuber, respectively, which is based on the similarity between column flooding phenomena in distillation and the CHF condition:

$$\dot{q}_{\max} = K \cdot \Delta h_v \cdot \rho_v \cdot \left(\frac{\sigma \cdot (\rho_l - \rho_v) \cdot g}{\rho_v^2} \right)^{\frac{1}{4}}$$

where the constant K is given for an infinite large horizontal plate: $K = 0.149$. This is only valid if the dimensions of the heated plate are significantly larger than the capillary dimensions of the vapor bubbles $L / L_b >> 30$.

For consideration of the liquid sub-cooling K is given by

$$K = 0.16 \cdot \left(1 + k_2 \cdot Ph\right)$$

Ph is the phase number which is a dimensionless measure of the sub-cooled liquid:

$$Ph = \frac{c_{pl} \cdot \left(T_s - T_l\right)}{\Delta h_v}$$

The value k_2 is given:

$$k_2 = C_o \left(\frac{\rho_l}{\rho_v}\right)^m$$

The constant C_0 and the exponent m are given by Kutateladze as $C_o = 0.065$ and $m = 0.8$.

The temperature difference ΔT_{crit} at critical heat flux is:

$$\Delta T_{crit} = \left(\frac{\dot{q}_{crit}}{\eta_l \, \Delta h_v \, L_b}\right)^r \cdot \frac{C_{sf}}{c_{pl}} \, \mathrm{Pr}^s \, \Delta h_v$$

- *Leidenfrost point:*

The Leidenfrost point is related to the minimum heat flux condition which is the boundary between the transition boiling and the film boiling regime and corresponds approximately to the lowest heat flux that will sustain stable film boiling.

Using the model formulation of Zuber, the following equation will be used for the calculation of the minimum heat flux $\dot{q}_{f,\min}$:

$$\dot{q}_{f,\min} = C_2 \, \rho_v \, \Delta h_v \cdot \left(\frac{\sigma \cdot g \cdot \left(\rho_l - \rho_v\right)}{\left(\rho_l + \rho_v\right)^2}\right)^{\frac{1}{4}}$$

For the value C_2 the results of Berenson is used who found $C_2 = 0.09$ as a good fit to the pool boiling experiments.

The temperature difference $\Delta T_{f,\min}$ at minimum film boiling heat flux is:

$$\Delta T_{f,\min} = \left(\frac{0.09}{0.425}\right)^{\frac{4}{3}} \frac{\rho_v \, \Delta h_{v,f}}{\lambda_v} L_b \cdot \left(\frac{g \cdot \left(\rho_l - \rho_v\right) \cdot \eta_v}{\left(\rho_l + \rho_v\right)^2}\right)^{\frac{1}{3}}$$

- *Film boiling:* $T_w > T_s + \Delta T_{f,\min}$

An empirical correlation for calculation of the film boiling Nusselt number is given by Berenson correlation. For the upward-facing horizontal infinite horizontal surface for water and R-113 at atmospheric pressure it is:

$$Nu = 0.425 \cdot \left[\left(\frac{Ra_b}{Ja_v} \right) \cdot Bo^{\frac{1}{2}} \right]^{\frac{1}{4}}$$

where Ra_b is the film boiling Rayleigh Number:

$$Ra_b = \frac{g \cdot \rho_v \cdot (\rho_l - \rho_v) \cdot l^3 \cdot \mathrm{Pr}_v}{\eta_v^2}$$

and Bo the corresponding Bond number:

$$Bo = \frac{g \cdot (\rho_l - \rho_v) \cdot l^2}{\sigma}$$

The Jacob number Ja_v is given by:

$$Ja_v = \frac{c_{pv} \cdot (T_w - T_s)}{\Delta h_v'}$$

where $\Delta h_v'$ is the effective enthalpy of evaporation:

$$\Delta h_v' = \Delta h_v + 0.5 \cdot c_{pv} \cdot (T_w - T_s)$$

For the corresponding film boiling heat flux the following equation is obtained:

$$\dot{q}_f = 0.425 \cdot \left(\frac{\lambda_v^3 \, g \, \rho_v (\rho_l - \rho_v) \Delta h_v'}{\eta_v (T_w - T_s) L_b} \right)^{\frac{1}{4}} \cdot (T_w - T_s)$$

The evaporated mass flow at film boiling can be calculated with:

$$\dot{m}_{ev} = \frac{\dot{q}_f}{\Delta h_v'}$$

The bubble frequency f is simply obtained by:

$$f = \frac{\dot{m}_{ev}}{\rho_v V_b}$$

- *Partial film boiling:* $T_w > T_s + \Delta T_{crit} \wedge T_w \leq T_s + \Delta T_{f,\min}$

The transition region between nucleate boiling and film boiling is called partial film boiling or transition boiling. Transition boiling has been traditionally interpreted as a combination of nucleate and film boiling alternately occurring over the heated surface. Compared to the other boiling regions there are only few investigations published regarding this phenomenon. Most of the results have been determined by performing quenching experiments.

Quenching studies typically have been initiated by heating a solid body to a high temperature and suddenly immersing the body into a pool of liquid. The body temperature is typically high enough that film boiling occurs at its surface immediately after immersion. As the body cools, its surface temperature drops to the point that the boiling process shifts to the transition boiling regime, followed by transition to nucleate boiling and finally to natural convection.

If the superheating of the wall is increased from the nucleate boiling value up to the one characterizing the maximum heat flux density at nucleate boiling, often there is no partial film boiling detectable, but the wall temperature directly increases up to the corresponding film boiling one. Consequently, the maximum temperature of the heater is exceeded abrupt in many cases. This behavior is called boiling crisis.

The heat flux density at partial film boiling often depends on the contact angle of the forward or backward moving border of contact. If the contact angle of the forward moving liquid is significantly smaller than the one of the backward moving liquid the heat flux density is significantly smaller at decreasing superheating of the wall. However, if the angle of contact decreases with decreasing wall superheating, unexpected increasing of the heat flux density can be observed. Because the structure of the surface has an important influence on the dynamic angle of contact, it must be taken into account. Although those influences are clearly identified in the literature, there are no corresponding models available.

The most widespread imagination is that partial film boiling is a superposition of nucleate boiling on the one hand and film boiling on the other hand and that the heat flux density can be determined using an interpolation of the wall temperature at maximum and minimum heat flux density, respectively. As described in, Ramilison and Lienhard propose the following correlation:

$$Bi^* = 3.74 \cdot 10^{-6} \cdot \left(Ja^*\right)^2 \cdot K$$

where

$$Bi^* = \frac{\left(\dot{q} - \dot{q}_{fb}\right) \cdot \sqrt{a_h \cdot \tau}}{\lambda_h \cdot \left[T_w - T_{SAT}\right] \cdot K}$$

$$Ja^* = \frac{\rho_h \cdot c_{ph} \cdot \left(T_{dfb} - T_W\right)}{\rho_v \cdot \Delta h_v}$$

$$\tau = \left[\frac{\sigma}{g^3 \cdot \left(\rho_l - \rho_v\right)}\right]^{\frac{1}{4}}$$

$$K = \frac{\lambda_l / a_l^{1/2}}{\lambda_l / a_l^{1/2} + \lambda_h / a_h^{1/2}}$$

In this correlation $\dot{q}_{fb}$ is the heat flux density at film boiling at equal wall temperature and τ is the characteristic Taylor wavelength of the liquid surface. All parameters with subscript h correspond to the properties of the heating surface. T_{dfb} is the minimum temperature of film boiling and the point where the liquid starts to touch the heating surface. The following approximation for calculation of T_{dfb} is proposed by Ramilison and Lienhard:

$$\frac{T_{dfb} - T_s}{T_{hn} - T_s} = 0.97 \cdot \exp\left(-0.00060 \cdot \theta_a^{1.8}\right)$$

In this context $\theta_a[°]$ is the angle of contact for the moving forward meniscus and T_{hn} is the temperature of nucleate boiling. T_{hn} can be determined using the correlation proposed by Lienhard [7]:

$$T_{hn} = \left[0.932 + 0.077 \cdot \left(\frac{T_{SAT}}{T_c}\right)^9\right] \cdot T_c$$

where T_c is the critical temperature of the liquid.

The correlation presented above provides good predictions of the experimental investigations of the authors for combinations of different liquids and heater materials.

Within the simulation tool EKSIM actually the following linear interpolation model is in use for the heat flux:

$$\dot{q}_t = \dot{q}_{crit} + \frac{\dot{q}_{f,\min} - \dot{q}_{crit}}{\Delta T_{f,\min} - \Delta T_{crit}} \left(\Delta T_{ws} - \Delta T_{crit} \right)$$

The evaporated mass flow at transition boiling can be calculated with:

$$\dot{m}_{ev} = \frac{\dot{q}_t}{\Delta h_v'}$$

The bubble frequency f is simply obtained by:

$$f = \frac{\dot{m}_{ev}}{\rho_v V_b}$$

A4.4 BUBBLE RISE IN LIQUID PROPELLANT UNDER REDUCED ACCELERATION

Bubble rise behaviour can be estimated by the theoretical model from 3. This model is based on measurements of air bubbles rising in water at 1 g, see Fig. 3-2. By exchanging the fluid properties and the acceleration level the model is applied to the current situation. Bubble distribution functions are not known. They must be evaluated either by tests for large bubble amount and movement or under real flight conditions.

THEORETICAL MODEL

The theoretical model as described hereafter is used and is valid within infinite spread liquid.

A simple relationship exists between the resistance ç and the rise velocity w_s of a bubble which is valid for rigid and moving phase interfaces:

$$w_s^2 = \frac{4}{3}\left(1 - \frac{\rho_B}{\rho}\right)\frac{a\, d_B}{\varsigma} \qquad (.1)$$

In the following the derived laws will be presented for the different forms of bubble occurring in liquids (properties to insert from liquid):

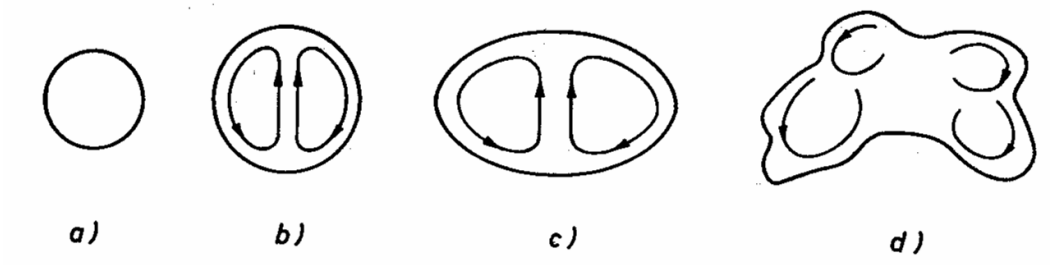


Figure A-2: Different bubble shapes

- region (a) spherical bubble without inner circulation
- region (b) spherical bubble with inner circulation
- region (c) elliptical bubble with inner circulation

3 Brauer, H., "Grundlagen der Einphasen- und Mehrphasenströmung. Verlag Sauerländer, 1971“

- region (d) irregular formed bubble (umbrella bubbles) with inner circulation

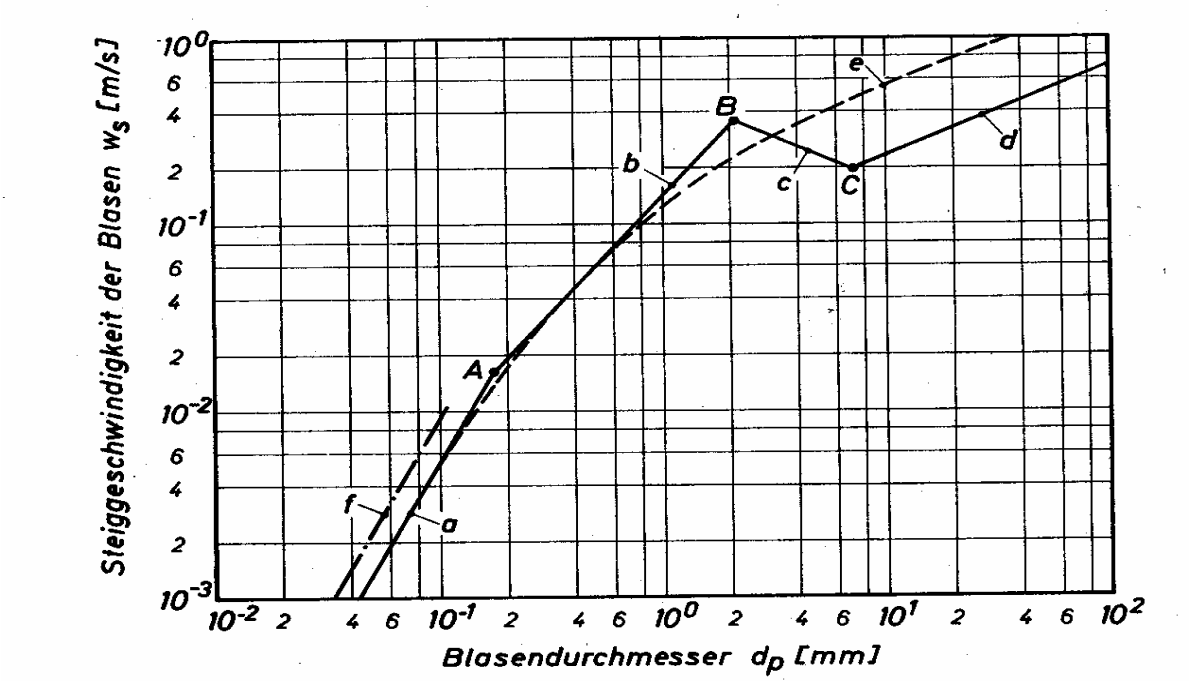


Figure A-3: Rise Velocity of Air Bubbles (a-b-c-d) and rigid Spheres (e) in Water; Curve (f) is valid under Consideration of Hadamard - Rybczynski Correction

Within region ***a*** spherical bubbles occur with rigid interfaces. From Stokes derived law for rigid spheres are valid. The rise velocity is under neglect of ρ_B against ρ:

$$w_s = \frac{1}{18}\frac{a\, d_B^2}{\nu} \tag{.2}$$

The Reynolds number is defined as:

$$\mathrm{Re}_s = \frac{w_s\, d_B}{\nu} \tag{.3}$$

The validity is between $0 \le \mathrm{Re}_s \le 2.2$. The maximum bubble diameter is therefore:

$$d_{a-b} = \left(2.448\ \ a^{-024}\ \nu^{0.48}\right)^{-072} \tag{.4}$$

Within region ***b*** spherical bubbles occur with inner circulation. The rise velocity is:

$$w_s = 0.136\,\frac{a^{0.76}\, d_B^{1.28}}{\nu^{0.52}} \tag{.5}$$

The validity is between $2.2 < \mathrm{Re}_s \le 4.02\ K^{0.214}$. K is a liquid characteristic number containing only properties of the liquid:

$$K = \frac{\mathrm{Re}_s^4\, Fr_s}{We_s^3} \tag{.6}$$

Froude and Weber number are named with Fr_s and We_s:

$$Fr_s = \frac{w_s^2}{a\, d_B} \tag{.7}$$

$$We_s = \frac{w_s^2\, d_B\, \rho}{\sigma} \tag{.8}$$

The upper limit of region ***b*** is dependent from the surface tension. The crossing of this limit represents the starting of bubble deformation from spherical to elliptical.

The maximum bubble diameter is therefore:

$$d_{b-c} = \left(197.237\, \frac{\sigma\, \nu^{1.04}}{\rho\, a^{1.52}} \right)^{-3.56} \tag{.9}$$

Within region ***c*** elliptical bubbles occur with inner circulation. The rise velocity is:

$$w_s = 1.91 \sqrt{\frac{\sigma}{\rho\, d_B}} \tag{.10}$$

Within region ***c*** the Weber number We_s is constant 3.64. The upper limit is:

$$\mathrm{Re}_s = 3.1\, K^{0.25} \tag{.11}$$

The maximum bubble diameter is therefore:

$$d_{c-d} = \left(7.156\, \frac{\sigma}{\rho\, a} \right)^{0.5} \tag{.12}$$

Within region ***d*** irregular shaped bubbles occur with inner circulation. The rise velocity is:

$$w_s = 0.714 \sqrt{a\, d_B} \tag{.13}$$

The rise height h_s can be simply obtained via integration of rise velocity w_s with time t:

$$\int_o^h dh_s = \int_0^t w_s(t)\, dt \tag{.14}$$

THEORETICAL MODEL ACCORDING TO COMOLET

Another model is proposed in 4. The authors distinguish 3 different regions of bubble behaviour and present an up-to-date overview of existing experimental correlations on the loss coefficient ç for each region. The 3 correlations use a non-dimensional velocity w_+ as a function of the non-dimensional diameter d_+ in the form:

$$w_+ = k \cdot Mo^m \cdot \left(k_1 d_+ + \frac{k_2}{d_+} \right)^n \tag{.15}$$

with

4 Di Marco, P. et al, "Experimental Study on rising Velocity of Nitrogen Bubbles in FC-72. International of Thermal Sciences, Vol. 42, pp. 435-446, 2005"

$$w_+ = We^{0.5} Bo^{-0.25}; \quad \text{and} \quad d_+ = Bo^{0.5}; \tag{.16}$$

that is:

$$w_+ = \frac{w}{w_o}; \quad w_o = \left(\frac{g\sigma(\rho_l - \rho_v)}{\rho_l^2} \right)^{0.25} \tag{.17}$$

and

$$d_+ = \frac{d}{d_o}; \quad d_o = \left(\frac{\sigma}{g(\rho_l - \rho_v)} \right)^{0.5} \tag{.18}$$

The Mo number can be expressed as non-dimensional gravity acceleration, in the sense that:

$$Mo = \frac{g}{g_o}; \quad g_o = \left(\frac{\sigma^3}{\rho_l^2 \nu_l^4 (\rho_l - \rho_v)} \right) \tag{.19}$$

Finally, the values for the different constants are defined as follows:

For the region **a**, the *Stokes* relation is valid, if $d_+ < 2Mo^{1/6}$ and with the parameters:

$$k = \frac{1}{12}; \quad m = -\frac{1}{4}; \quad k_1 = 1; \quad k_2 = 0; \quad n = 2 \tag{.20}$$

For the region **b**, the *Wallis* relation is valid, if $2Mo^{1/6} \le d_+ \le 3.8Mo^{1/14}$ and with the parameters:

$$k = \frac{7}{50}; \quad m = -\frac{1}{8}; \quad k_1 = 1; \quad k_2 = 0; \quad n = 1.25 \tag{.21}$$

For the region **c** and **d**, the *Comolet* relation is valid, if $d_+ > 3.8Mo^{1/14}$ and with the parameters:

$$k = 1; \quad m = 0; \quad k_1 = 0.51; \quad k_2 = 2.14; \quad n = 0.5 \tag{.22}$$

A5. INTERFACE TO THE FORTRAN SURFACE SHAPE ROUTINES

For the design of satellite tanks the distribution of the liquid propellant inside the tank of zero-g or small-g levels is of interest. In this context a program written in FORTRAN 90 language has been developed [1].

As part of the ESPSS project an EcosimPro[2] interface to this program for calculation of equilibrium shapes of liquid surfaces in propellant tanks is described.

The interface is written in EcosimPro language EL [2] and can be included in existing simulations just by embedding the EcosimPro library `SURFACE_SHAPE_DLL`.

References

[1] Weiß, J.: *Equilibrium Shapes of Liquid Surfaces in Tanks of various Types with Rotational Symmetry*, University of Applied Sciences, Bremen, 2004.

[2] EcosimPro 3.4.2, EA International, Madrid, 2006.

[3] EUCES-RIBRE-TN-0005, "Generation of Message Windows within EcosimPro"

[4] Weiß, J.: *Subroutine LH2_LOX*, University of Applied Sciences, Bremen, 2004.

[5] Compaq Visual Fortran 6.6A, Compaq Computer Corporation, 2000.

A5.1 STRUCTURE

The described code combines the EcosimPro environment with the FORTRAN 90 code of the surface shape calculation routines, see Figure A-4.

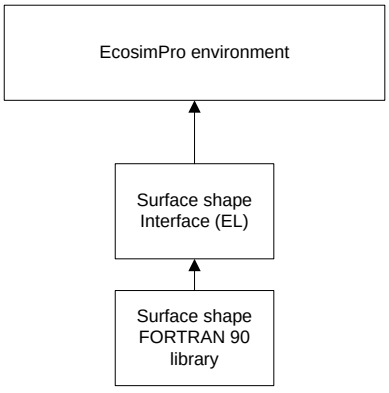


Figure A-4: Surface shape interface structure

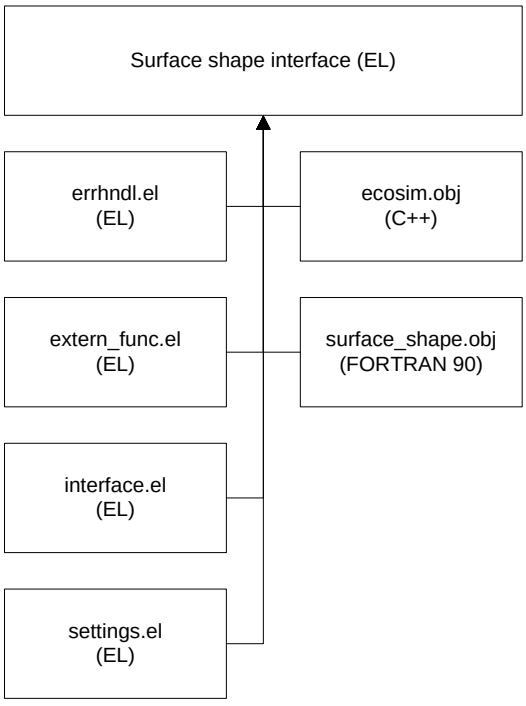


Figure A-5: Interface file structure

The Interface to the Fortran 90 surface shape routines is divided into six files: four .el-files written in the EcosimPro language EL and two external object-files (.obj).

A5.2 FUNCTIONALITY

The functionality of the files of the interface and the interface itself is described below.

errhndl.el

This file contains the error handling system of the interface. It mainly uses the error codes generated by the FORTRAN 90 subroutines [4]. In case of an error a message window will be generated using a C++ function which is stored in the ecosim.obj file.

Errors can be divided into two categories: input error codes 1-15 and internal error codes 101-113 (see tables below).

The internal error codes are returned by the `DERKF` routine which is a part of the public available `DEPAC` package for solving differential equations. `DERKF` is a fifth order Runge-Kutta code, for details see [1].

Error code	Description	Internal Error code	Description
1	`TANK_COMPARTMENT` unknown (1=LH2; 2=LOX)	101	An apparent infinite loop has been detected.
2	Acceleration along x-axis [g_0] out of boundaries $-10 \le ACC\_X \le 0$	102	Length of `rwork` array must be at least 30+7·`neq`.
3	Acceleration along y-axis [g_0] is not supported in this version and must be zero	103	Length of `iwork` array must be at least 30.
4	Rotation rate [degrees/second] out of boundaries $0 \le ACC\_ROT \le 100$	104	`info(1)` must be set to 0 for the start of a new problem and must be set to 1 following an interrupted task.
5	Filling level out of boundaries $FILL\_LEVEL\_TOL \le FILL\_LEVEL \le 100$	105	`info(2)` must be 0 or 1 indicating scalar and vector error tolerances, respectively.
6	Filling level tolerance out of boundaries $0.1 \le FILL\_LEVEL\_TOL \le 10$	106	`info(3)` must be 0 or 1 indicating the interval or intermediate-output mode of integration, respectively.
7	(Density/Surface tension) out of boundaries $0 < \frac{\rho}{\sigma} \le 1$	107	The number of equations `neq` must be a positive integer.
8	Radius `R1` [m] out of boundaries, $0 < R_1 \le 1$	108	The relative error tolerances `rtol` must be non-negative. In the case of vector error tolerances no further checking of `rtol` components is done.
9	Radius `R2` (LH2) [m] out of boundaries, $0 < R_2 \le 1$	109	The relative error tolerances `atol` must be non-negative. In the case of vector error tolerances no further checking of `atol` components is done.
10	Radius `R3` (LH2) [m] out of boundaries, $0 < R_3 \le 1$	110	You have called the code with `t=tout`. This is not allowed on continuation calls.
11	Radius `RZ` [m] out of boundaries, $0 < R_Z \le 1$	111	You have changed the value of `t` from `(R1)` to `(R2)`. This is not allowed on continuation calls.
12	`XP3` (LH2) [m] out of boundaries, $x_{P2} < x_{P3} \le 1$	112	By calling the code with `tout=(R1)` you are attempting to change the direction of integration. This is not allowed without restarting.
13	Radius `R2` (LOX) [m] out of boundaries, $0 < R_2 \le 1$	113	Invalid input was detected on successive entries. It is impossible to proceed because you have not corrected the problem, so execution is being terminated.
14	Radius `R3` (LOX) [m] out of boundaries, $0 < R_3 \le 1$		
15	`XP3` (LOX) [m] out of boundaries, $x_{P2} < x_{P3} \le 1$		

extern_func.el

The extern_func.el file links the two extern C++ functions EMESSAGE and STOP_ALL to the EcosimPro environment. Both functions are stored in the ecosim.obj file.

interface.el

This file contains the main components of the interface code. All external FORTRAN 90 functions are linked within this file. Furthermore, a conversion routine is included to get SI units within EcosimPro environment.

The subroutine call within EcosimPro for calculation of equilibrium shapes of liquid surfaces in the propellant tank compartments is

```
SURF_SHAPE(TANK_COMPARTMENT,ACC_X,ACC_Y,ACC_ROT,FILL_LEVEL,FILL_LEVEL_TOL,
           RHO_LIQUID,SIGMA_LIQUID,R1,R2,R3,RZ,XP3,FREE_SURF,A_WET,
           A_WET_ABS,COG,SEP_LINE,ERR_ID)
```

All input and output parameters are described below.

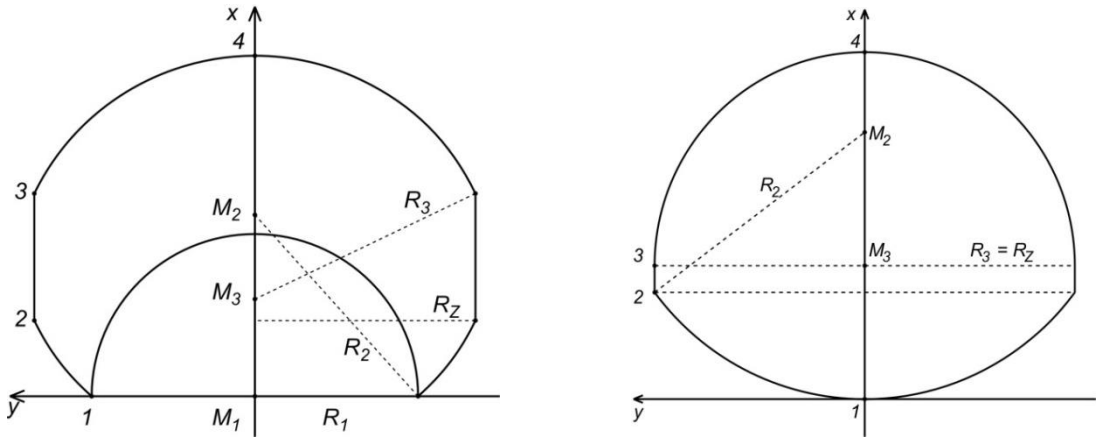


Figure A-6: Cross section through LH2/LOX tank compartments

The input parameters of the SURF_SHAPE subroutine are shown in Table A-1. For more details see [1].

Input parameter	Description
TANK_COMPARTMENT	Tank compartment (1: LH2; 2: LOX)
ACC_X	Linear acceleration along x-axis [g_0]
ACC_Y	Linear acceleration along y-axis [g_0], not supported in this version
ACC_ROT	Rotation rate [°/s]
FILL_LEVEL	Fill level [%]
FILL_LEVEL_TOL	Tolerance for fill level [%]
RHO_LIQUID	Density of the liquid [kg/m³]
SIGMA_LIQUID	Surface tension of the liquid [N/m]

R1	Radius R_1 [m]
R2	Radius R_2 [m]
R3	Radius R_3 [m]
RZ	Radius R_Z [m]
XP3	X-coordinate of point 3 [m]

Table A-1: Input parameters of the SURF_SHAPE subroutine

Output parameter	Description
FREE_SURF	Free surface [m²]
A_WET	Array of four wet tank surfaces [%] [1]: Upper part of sphere (R_3) [2]: Cylinder (R_Z) [3]: Lower part of sphere (R_2) [4]: Semi sphere (R_1)
A_WET_ABS	Same as A_WET but with absolute values [m²]
COG	Centre of gravity x-coordinate [m]
SEP_LINE	Array of (x,r)-coordinates of points of the separation line [m] (SEP_LINE(2,101))
ERR_ID	Error code, see previous section

Table A-2: Output parameters of the SURF_SHAPE subroutine

settings.el

This file contains setting parameters for modification of the error handling system.

Setting parameter	Description
err_surf	Switch for error handling system TRUE/FALSE TRUE: Error messages FALSE: No error messages
err_ign_surf	Switch for error message options TRUE/FALSE TRUE: The "ignore future messages" option is available FALSE: The "ignore future messages" option is not available

Table A-3: Setting parameters for the interface

ecosim.obj

Ecosim.obj is a compiled C++ file that contains the two functions EMESSAGE and STOP_ALL. EMESSAGE generates a message window and STOP_ALL stops the simulation immediately. The functionality is described more in detail in [3]. The ecosim.obj file has to be linked to the EcosimPro partition that contains the experiments.

surface_shape.obj

This file contains the surface shape calculation functions. It is the compiled file of the FORTRAN 90 code. The surface_shape.obj file has to be linked to the EcosimPro partition that contains the experiments.

Additional requirements

Because the original surface shape is written in the FORTRAN 90 language and compiled with the Compaq Visual Fortran 6.6A [5] an additional file is required.

The original surface shape routines use some Win32 API functions which are stored in a special library. This library file, `DFWIN.lib`, is distributed with the Compaq Visual Fortran compiler. A copy of this library file needs to be stored in the `\lib` subfolder of the EcosimPro environment.

A5.3 PERFORMANCE& CONCLUSION

First simple performance tests have been executed. For that purpose the following iteration loop has been applied to the EcosimPro interface to the FORTRAN 90 program.

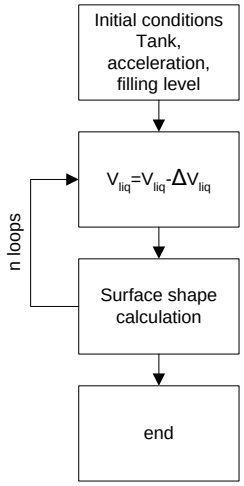


Figure A-7: Performance test iteration loop

To simulate different calculation conditions the tank compartments have been completely drained under both zero gravity and 1 g conditions even though these combinations are not realistically.

An EcosimPro interface to the FORTRAN 90 routines for calculation of equilibrium shapes of liquid surfaces in propellant tanks [1]1 g conditions even though these combinations are not realistically.

The results of the performance tests are shown in Figure A-8.

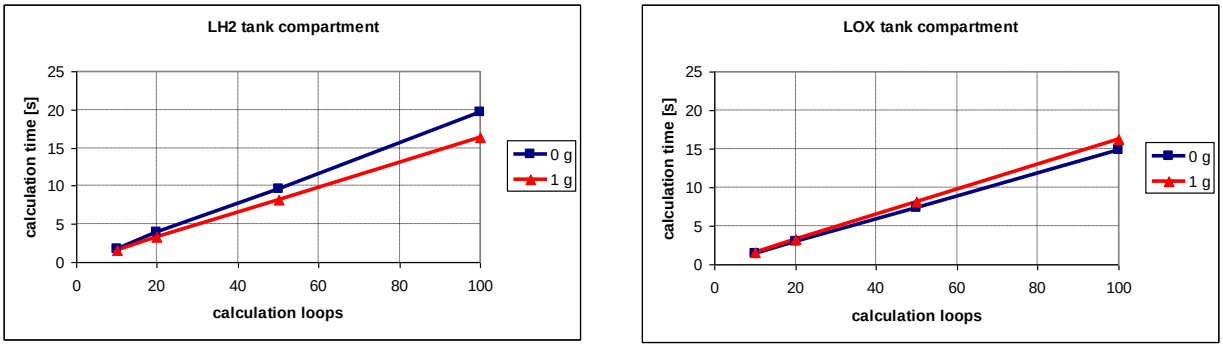


Figure A-8: Performance test results

These first performance tests show that the average time per calculation is about t_{calc}=0.14 s…0.2 s.

EcosimPro interfaces with the FORTRAN 90 routines to calculate equilibrium shapes of liquid surfaces in propellant tanks [1].

Based on first performance test calculation loops it can be concluded that the use of this tool within any simulation should considered before because the iteration takes a long time. The FORTRAN 90 object file used for this test has been compiled using a Compaq Visual Fortran 6.6A compiler at maximum optimization level.

A6. COVERAGE OF 2 ARBITRARILY DIMENSIONED SURFACE VECTORS

In case of a 1D ullage or liquid propellant discretization for the propellant tank, the level of moving liquid due to propellant in- or outflow makes it advantageous to apply a moving grid with an equally but time-dependent volume distribution for each compartment.

For a connection between the ullage or propellant to a fixed discretized tank shell, a coverage matrix **A** [n *x* m] and its inverse $\mathbf{A^{-1}}$ [m *x* n] have to be evaluated for each time step to calculate the coverage degree of the wall related fixed surfaces $\vec{S}_u$ [n] to the varying e.g. ullage surfaces $\vec{S}_o$ [m].

The relation for the coverage matrix is:

$$\vec{S}_u = \overline{\overline{A}} \cdot \vec{S}_o$$

or vice versa for its inverse:

$$\vec{S}_o = \overline{\overline{A^{-1}}} \cdot \vec{S}_u$$

Starting with re-initialization of the matrix coefficients $A_{i,j}$ and the start value of the surface summand s_u :

$$A_{i,j} = 0\,; \qquad i = 1,n\,; \qquad j = 1,m$$

$$s_u = 0$$

The outer loop counts from 1 to n surface elements with index i e.g. over all left hand side surfaces. The inner loop counts from 1 to m with an additional condition to identify the next cross over of the right hand side surface elements. As there exits several possibilities for overlapping, the logic presented below can be applied:

$FOR\ \ i = 1,n$

$s_u = s_u + S_{u\,i}$

$j = 0$

$s_o = S$

$WHILE\ \left(j < m \wedge s_o \le s_u\right)$

$j\,+\!= 1$

$s_o = s_o + S_{o\,j}$

$IF\ \left(s_u \le s_o \wedge s_u > s_o - S_{o\,j} \wedge s_o - S_{o\,j} \le s_u - S_{u\,i}\right)\ THEN$

$$A_{i,j} = \frac{S_{u\,i}}{S_{o\,j}}$$

$ELSE\ IF\ \left(s_u \ge s_o \wedge s_o > s_u - S_{u\,i} \wedge s_o - S_{o\,j} \ge s_u - S_{u\,i}\right)\ THEN$

$$A_{i,j} = 1$$

$ELSE\ IF\ \left(s_u \ge s_o \wedge s_o > s_u - S_{u\,i} \wedge s_o - S_{o\,j} < s_u - S_{u\,i}\right)\ THEN$

$$A_{i,j} = \frac{\left(s_o - \left(s_u - S_{u\,i}\right)\right)}{S_{o\,j}}$$

$ELSE\ IF\ \left(s_u < s_o \wedge s_u > s_o - S_{o\,j} \wedge s_o - S_{o\,j} \ge s_u - S_{u\,i}\right)\ THEN$

$$A_{i,j} = 1 - \frac{(s_o - s_u)}{S_{oj}}$$

$$END\ IF$$

$$END\ WHILE$$

$$END\ FOR$$

The reverse counter logic for the inner loop with index j starting at m+1 is:

$$j = m+1$$
$$WHILE\ (j > 1 \wedge s_o \le s_u)$$
$$j -- = 1$$

The reverse counter loop for the outer loop with index i starting at n is:

$$FOR\ (i = n; i > 0; i-- = 1)$$

A6.1 HEAT FLUX CONNECTOR FOR N SURFACES TO M SURFACES FOR DEFINED Q

In EcosimPro, a PORT connection for external heat fluxes contains the heat flux and the surface of contact.

The definition for heat exchange averaging between surfaces $\vec{S}_u$ (left) and $\vec{S}_o$ (right) of a *solid adaptor* to connect ***n*** surfaces to ***m*** surfaces of arbitrary distribution per element $S_{u\,i}$ and $S_{o\,j}$, respectively, results in the following conversion equation for the heat fluxes with index o:

$$\dot{Q}_{o\,j} = \sum_{i=1}^{n} A_{j,i}^{-1} \cdot \dot{Q}_{u\,i}\ ; \qquad j = 1, m$$

A6.2 HEAT FLUX AND TEMPERATURE CONNECTOR FOR N SURFACES TO M SURFACES FOR DEFINED Q, T

In EcosimPro, a general PORT connection for heat fluxes contains the heat flux, the temperature and the surface of contact.

The definition for heat exchange and temperature averaging between surfaces $\vec{S}_u$ (left) and $\vec{S}_o$ (right) of a *solid adaptor* to connect ***n*** surfaces to ***m*** surfaces of arbitrary distribution per element $S_{u\,i}$ and $S_{o\,j}$, respectively, results in the following conversion equation for the heat fluxes with index o:

$$\dot{Q}_{o\,j} = \sum_{i=1}^{n} A_{j,i}^{-1} \cdot \dot{Q}_{u\,i}\ ; \qquad j = 1, m$$

and for the temperatures with index u:

$$T_{u\,i} = \frac{1}{S_{u\,i}} \sum_{j=1}^{m} A_{i,j} \cdot \dot{Q}_{o\,j}\ ; \qquad i = 1, n$$

A6.3 HEAT FLUX CONNECTOR FOR K SURFACES TO N + M SURFACES FOR DEFINED Q

In EcosimPro, a PORT connection for external heat fluxes contains the heat flux and the surface of contact.

The definition for heat exchange averaging between surfaces $\vec{S}_u$ (left) and $\vec{S}_{o1}$ (right 1), $\vec{S}_{o2}$ (right 2) of a *solid adaptor* to connect ***k*** surfaces to ***n***surfaces via $\mathbf{A^{-1}_1}$ and ***m***surfaces via $\mathbf{A^{-1}_2}$ of arbitrary distribution per element $S_{u\,i}$ and $S_{o1\,j}$, $S_{o2\,j}$, respectively, results in the following conversion equation for the heat fluxes with index o1:

$$\dot{Q}_{o1\,j} = \sum_{i=1}^{k} A^{-1}_{1\,j,i} \cdot \dot{Q}_{u\,i}\,; \qquad j = 1,n$$

and for the heat fluxes with index o2:

$$\dot{Q}_{o2\,j} = \sum_{i=1}^{k} A^{-1}_{2\,j,i} \cdot \dot{Q}_{u\,i}\,; \qquad j = 1,m$$

A6.4 HEAT FLUX CONNECTOR FOR K SURFACES TO N+M SURFACES FOR DEFINED Q,T

In EcosimPro, a general PORT connection for heat fluxes contains the heat flux, the temperature and the surface of contact.

The definition for heat exchange and temperature averaging between surfaces $\vec{S}_u$ (left) and $\vec{S}_{o1}$ (right 1), $\vec{S}_{o2}$ (right 2) of a *solid adaptor* to connect ***k*** surfaces to ***n*** surfaces via $\mathbf{A^{-1}_1}$ and ***m*** surfaces via $\mathbf{A^{-1}_2}$ of arbitrary distribution per element $S_{u\,i}$ and $S_{o1\,j}$, $S_{o2\,j}$, respectively, results in the following conversion equation for the temperatures with index o1:

$$T_{o1\,j} = \frac{1}{S_{o1\,j}} \sum_{i=1}^{k} A^{-1}_{1\,j,i} \cdot S_{u\,i} \cdot T_{u\,i}\,; \quad j = 1,n$$

and for the temperatures with index o2:

$$T_{o2\,j} = \frac{1}{S_{o2\,j}} \sum_{i=1}^{k} A^{-1}_{2\,j,i} \cdot S_{u\,i} \cdot T_{u\,i}\,; \quad j = 1,m$$

The transformation for the heat fluxes with index u is:

$$\dot{Q}_{u\,i} = \sum_{j=1}^{n} A_{1\,i,j} \cdot \dot{Q}_{o1\,j} + \sum_{j=1}^{m} A_{2\,i,j} \cdot \dot{Q}_{o2\,j}\,; \qquad i = 1,k$$

A6.5 HEAT FLUX CONNECTOR FOR K,L,O SURFACES TO N SURFACES FOR DEFINED Q,T

In EcosimPro, a general PORT connection for heat fluxes contains the heat flux, the temperature and the surface of contact.

The definition for heat exchange and temperature averaging between surfaces $\vec{S}_{u1}$ (left 1), $\vec{S}_{u2}$ (left 2), $\vec{S}_{u3}$ (left 3) and $\vec{S}_o$ (right) of a *solid adaptor* to connect ***k, l , o*** surfaces to ***n*** surfaces via the matrixes $\mathbf{A_{21}}$, $\mathbf{A_{31}}$ and $\mathbf{A^{-1}_{11}}$, $\mathbf{A^{-1}_{12}}$, $\mathbf{A^{-1}_{13}}$ of arbitrary distribution per element $S_{u1\,i}$, $S_{u2\,i}$, $S_{u3\,i}$ and $S_{o\,j}$, respectively, results in the following conversion equation for the temperatures with index u2:

$$T_{u2\,i} = \frac{1}{S_{u2\,i}} \sum_{j=1}^{n} A_{21\,i,j} \cdot S_{o\,j} \cdot T_{o\,j}\,; \quad i = 1,l$$

and for the temperatures with index u3:

$$T_{u3\,i} = \frac{1}{S_{u3\,i}} \sum_{j=1}^{m} A_{31\,i,j} \cdot S_{o\,j} \cdot T_{o\,j}\,; \quad i = 1,o$$

The transformation for the heat fluxes with index o is:

$$\dot{Q}_{o\,j} = \sum_{i=1}^{k} A^{-1}_{11\,j,i} \cdot \dot{Q}_{u1\,i} + \sum_{i=1}^{l} A^{-1}_{12\,j,i} \cdot \dot{Q}_{u2\,i} + \sum_{i=1}^{o} A^{-1}_{13\,j,i} \cdot \dot{Q}_{u3\,i}; \qquad j = 1,n$$

A6.6 HEAT FLUX CONNECTOR FOR K, L, O SURFACES TO N, M SURFACES FOR DEFINED Q, T

In EcosimPro, a general PORT connection for heat fluxes contains the heat flux, the temperature and the surface of contact.

The definition for heat exchange and temperature averaging between surfaces $\vec{S}_{u1}$ (left 1), $\vec{S}_{u2}$ (left 2), $\vec{S}_{u3}$ (left 3) and $\vec{S}_{o1}$ (right 1), $\vec{S}_{o2}$ (right 2) of a *solid adaptor* to connect ***k, l , o*** surfaces to ***n, m*** surfaces via the matrixes $\mathbf{A_{21}}$, $\mathbf{A_{22}}$, $\mathbf{A_{31}}$, $\mathbf{A_{32}}$ and $\mathbf{A^{-1}_{11}}$, $\mathbf{A^{-1}_{12}}$, $\mathbf{A^{-1}_{13}}$, $\mathbf{A^{-1}_{21}}$, $\mathbf{A^{-1}_{22}}$, $\mathbf{A^{-1}_{33}}$ of arbitrary distribution per element $S_{u1\,i}$, $S_{u2\,i}$, $S_{u3\,i}$ and $S_{o1\,j}$, $S_{o2\,j}$ respectively, results in the following conversion equation for the temperatures with index u2:

$$T_{u2\,i} = \frac{1}{S_{u2\,i}} \left(\sum_{j=1}^{n} A_{21\,i,j} \cdot S_{o1\,j} \cdot T_{o1\,j} + \sum_{j=1}^{m} A_{22\,i,j} \cdot S_{o2\,j} \cdot T_{o2\,j} \right); \quad i = 1, l$$

and for the temperatures with index u3:

$$T_{u3\,i} = \frac{1}{S_{u3\,i}} \left(\sum_{j=1}^{n} A_{31\,i,j} \cdot S_{o1\,j} \cdot T_{o1\,j} + \sum_{j=1}^{m} A_{32\,i,j} \cdot S_{o2\,j} \cdot T_{o2\,j} \right); \quad i = 1, o$$

The transformation for the heat fluxes with index o1 is:

$$\dot{Q}_{o1\,j} = \sum_{i=1}^{k} A^{-1}_{11\,j,i} \cdot \dot{Q}_{u1\,i} + \sum_{i=1}^{l} A^{-1}_{12\,j,i} \cdot \dot{Q}_{u2\,i} + \sum_{i=1}^{o} A^{-1}_{13\,j,i} \cdot \dot{Q}_{u3\,i}; \qquad j = 1, n$$

and for the heat fluxes with index o2 is:

$$\dot{Q}_{o2\,j} = \sum_{i=1}^{k} A^{-1}_{21\,j,i} \cdot \dot{Q}_{u1\,i} + \sum_{i=1}^{l} A^{-1}_{22\,j,i} \cdot \dot{Q}_{u2\,i} + \sum_{i=1}^{o} A^{-1}_{33\,j,i} \cdot \dot{Q}_{u3\,i}; \qquad j = 1, m$$

A7. GIBBS MINIMIZATION AS AN APPROACH TO EQUILIBRIUM

References

[1] Çengal, Y. A., and M. A. Boles: Thermodynamics, An Engineering Approach, 3rd Edition, McGraw-Hill pp. 810-815 (1998).

[2] Gordon, S., and B. J. McBride: Computer Program for the Calculation of Complex Chemical Equilibrium Compositions with Applications; I. Analysis. NASA Reference Publication 1311 (1994). available for download as a PDF file via ftp link at: http://letrs.lerc.nasa.gov/cgi-bin/GLTRS/browse.pl?1994/E-8017.html

[3] Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling: Numerical Recipes: The Art of Scientific Computing, CambridgeUniversity Press, pp. 269-272 (1986).

(Following write-up comes from http: //courses.washington.edu/mengr524/Handouts/gibbs.pdf)

A7.1 GENERAL THEORY

The general idea behind thermochemical equilibrium is that all spontaneous reactions occur in the direction that increases the overall entropy of the universe (meaning both the system under consideration and the surroundings that represent the rest of the universe). When the composition of the system arrives at a point where the total entropy for the system plus surroundings reaches a maximum, it becomes "stuck" since movement in any direction in composition space involves an entropy decrease, and thus cannot spontaneously occur without violating the second law. Thus, the equilibrium state we seek is this end state, which is reached by all spontaneously reacting systems if given enough time.

There are several kinds of equilibrium problems. A few of the more important ones are the following:

• The system can be a box that is fixed in volume and is adiabatic. In this case, as the reactions progress, both the temperature and pressure within the box will change. Thus, both the final temperature and the pressure are unknowns and these are provided by the solution. This system is an idealization of what occurs in a calorimetry bomb.

• The system can be fixed in volume, but be in contact with an environment at a known temperature. As the reaction progresses, any energy released by the reaction will be carried away as heat to keep the box at the same temperature. The pressure will change in response to the change in the total number of moles as the reaction progresses. Thus, the final temperature is known, but the final pressure is provided as part of the solution.

• The system can be adiabatic, but is allowed to expand or contract as the reaction progresses to keep the pressure constant (e.g., an adiabatic piston/cylinder system). In this case, the final pressure is known but the final temperature is part of the solution. Many common combustion systems most closely approach this model, e.g., Bunsen burners, combustion in boilers, gas turbine combustors. In these systems the combustion is approximately adiabatic and the pressure is constant (here the specific volume of the gas expands to accommodate the increase in temperature and any changes in mole numbers).

• The system is a box that is in contact with an environment at a known temperature and the volume expands or contracts to keep the pressure constant. Here, both the final temperature and pressure are known, and the box undergoes both a heat and work interaction with the surroundings while the reaction progresses. This case is important when you know both the final temperature and pressure of a process and you are looking for the corresponding equilibrium composition. An example might be the composition at the tail pipe of a car, where you have measured the exhaust temperature and you know the pressure is atmospheric.

We will focus on the latter case. The approach for the others is similar. The starting point is the increase in entropy principle, which says:

$$S_{gen} = \Delta s_{system} + \Delta s_{surroundings} \qquad (1)$$

For any spontaneous process, $S_{gen} \geq 0$. Since the environment is assumed to be at a constant temperature, $\Delta s_{surroundings} = q/T$, where T is the temperature of the surroundings and q is the heat flow into the

surroundings (if you have forgotten why this is true, remember, this is the formal definition of s). This means that for any spontaneous process:

$$0 \leq \Delta s_{system} + q_{surroundings}/T_{surroundings} \quad (2)$$

or in differential form: $0 \leq ds_{system} + \delta q_{surroundings}/T_{surroundings}$. The differential first law for our box is:

$$\delta q - \delta w = du \quad (3)$$

For a reversible process, the work term is Pdv. From equation 2, q can be expressed as $\delta q_{system} \leq T ds_{system}$ (note that we have changed our reference point for q from the surroundings to the system, which involves a change of sign for q). Substituting these into the first law gives: $Tds - Pdv \geq du$, or

$$0 \geq du + Pdv - Tds \quad (4)$$

Note that this equation holds for any of the four types of equilibrium systems noted above. The definition of Gibbs free energy is:

$$g = u + Pv - Ts \quad (5)$$

Taking the derivative, we get:

$$dg = du + Pdv + vdP - Tds - sdT \quad (6)$$

We are following an equilibrium process that occurs at constant pressure and temperature (case 4 from above). In this case, dP=dT=0, so the equation becomes:

$$dg = du + Pdv - Tds \quad (7)$$

Comparing this to equation 4, we see that for a spontaneous chemical reaction at constant T and P, the change in Gibbs energy must be negative (i.e., a positive change violates the second law):

$$0 \geq dg \quad (8)$$

This means that spontaneous reaction will occur at a fixed temperature and pressure until the Gibbs free energy reaches a minimum point in composition space, and then it will become "stuck". This will be the equilibrium point.

The Gibbs free energy is a function of pressure, temperature, and composition (i.e., the moles of the various components that are present, e.g., H2O, CO2, etc.). This functionality can be formally written as:

$$g = g(T, P, N1, N2, \ldots N_{NS}) \quad (9)$$

Here, Nj is the number of moles of species j in the box, and the index NS is the total number of species in the system. Taking the total derivative of g gives:

$$dg = \frac{\partial g}{\partial T}_{P,N} dT + \frac{\partial g}{\partial P}_{T,N} dP + \sum \frac{\partial g}{\partial N_j}_{P,T,N} dN_j \quad (10)$$

Here the summation is over all the species present. Since T and P are fixed, these terms drop out. This leaves our equilibrium condition as:

$$dg = 0 = \sum \mu_j dN_j \quad (11)$$

Here μj is the chemical potential, which is defined as: $\mu_j = \frac{\partial g}{\partial N_j}_{P,T,N_j}$ (12)

The chemical potential can be thought of as the change of Gibbs free energy of a mixture caused by the addition of a differential amount of species j when the T, P, and other mole numbers are held constant. For ideal gases, this is the Gibbs free energy of the individual species since they do not interact in a mixture:

$$g_j = u_j + Pv_j - Ts_j = h_j - Ts_j \quad (13)$$

Now we can expand h in terms of enthalpy of formation and also expand s to express the pressure correction for ideal gases:

$$g_j = h^0_{f,j} + (h_j - h_{0,j}) - T[s^0_j - R\ln(P_j / P_0)] \quad (14)$$

Here, $h^0_{f,j}$ is the enthalpy of formation at 298 K, hj is the enthalpy at the target temperature, hj,0 is the enthalpy at 298 K, s^0_j is the 1 atm entropy at the target temperature, R is the gas constant, Pj is the partial pressure of the component, and P0 is 1 atm (all of these are simple look-up values from standard thermodynamics tables). It is usually customary to separate out the properties that depend just on temperature, so:

$$g_j = h^0_{f,j} + (h_j - h_{0,j}) - Ts^0_j + RT\ln(P_j / P_0) \quad (15)$$

Next, we define: $g^*_j = h^0_{f,j} + (h_j - h_{0,j}) - Ts^0_j$ (16)

We also split up the pressure term as follows:

$$\ln(\frac{P_j}{P_0}) = \ln(\frac{P_j}{P}\frac{P}{P_0}) = \ln(\frac{N_j}{N}\frac{P}{P_0}) = \ln(\frac{N_j}{N}) + \ln(\frac{P}{P_0}) \quad (17)$$

Here, P is the system pressure and N is the total number of moles in the system. This leads us to an operational equation for calculating gj:

$$g_j = g^*_j + RT\ln N_j - RT\ln N + RT\ln(\frac{P}{P_0}) \quad (18)$$

If you know T and P, you can get g*, and the only unknowns are the mole numbers of species j and the total number of moles in the system. Substituting this into equation 11 gives us the operational equation for the minimization:

$$\mathrm{dg} = 0 = \sum g_j dN_j = \sum[g^*_j + RT\ln N_j - RT\ln N + RT\ln(\frac{P}{P_0})]dN_j \quad (19)$$

This is the point of departure for the equilibrium constant approach. In that approach, an equilibrium reaction is hypothesized, and this is used to reduce all the dNj to one variable, which is then divided into the zero. The resulting equation contains (within the gj terms) the variables Nj and N. Using algebraic manipulation and atom balances, the Nj and N terms are reduced to a single variable, which is solved (this approach is detailed in most standard thermodynamics texts, e.g., Çengal and Boles (1998), and is not discussed further here). This approach is essentially impossible to execute for complex systems, so we move to the general Gibbs minimization approach which is the basis for all the equilibrium codes.

A7.2 SOLUTION VIA LAGRANGE MULTIPLIERS.

It is key to recognize that the Nj in equation 19 are not independent variables. They are constrained such that the number of moles of each element in the system must remain constant (i.e., if you start with 4 moles of oxygen atoms, this must stay constant as the reaction progresses). These constraint equations are best developed by example. Assume a system starts with 1 mole CO2 and 2 moles H2O (for a total of 4 moles of O-atom going in). We assume the equilibrium mixture contains CO2, H2O, OH and O2. The constraint equation for oxygen atoms is an expression of the fact that there must be 4 moles of O-atom in the products:

$$4 = 2NCO2 + NH2O + NOH + 2NO2 \quad (20)$$

One approach to such constrained optimization problems is the method of Lagrange (or undetermined) multipliers. (A simple application of the approach is to constrain a cylinder to contain a certain volume, and find the length and diameter dimensions that minimize the total external surface area.)

We start by generalizing the constraint condition as:

$$0 = \sum_{i=1}^{NS} a_{i,j}N_j - b_i \quad (21)$$

Here, bi is the number of moles of element i in the system, and ai,j is number of atoms of element I in one molecule of species j (e.g., there are 2 atoms of O in one molecule of CO2, so ai,j=2). There will be one of these equations for each element in the system (i=1,2,…NE), where NE is the total number of elements in the system). If we generalize these equations as functions:

$$0 = \varphi_i(N_1, N_2, \ldots N_{NS}) \tag{22}$$

Here we have one constraint equation for each element. We can take the total derivative of this:

$$d\varphi_i = 0 = \left(\frac{\partial \varphi_i}{\partial N_1}\right) dN_1 + \left(\frac{\partial \varphi_i}{\partial N_2}\right) dN_2 + \ldots + \left(\frac{\partial \varphi_i}{\partial N_{NS}}\right) dN_{NS} \tag{23}$$

Now we make a linear combination of equation 23 (one for each element) with the differential dg from equation 10 (with the P and T derivatives already set to 0):

$$dg + \lambda_1 d\varphi_1 + \lambda_2 d\varphi_2 + \ldots = \left[\frac{\partial g}{\partial N_1} + \lambda_1 \frac{\partial \varphi_1}{\partial N_1} + \lambda_2 \frac{\partial \varphi_2}{\partial N_1} + \ldots\right] dN_1 + \left[\frac{\partial g}{\partial N_2} + \lambda_1 \frac{\partial \varphi_1}{\partial N_2} + \lambda_2 \frac{\partial \varphi_2}{\partial N_2} + \ldots\right] dN_2 + \ldots \tag{24}$$

The λi's can be anything, and we define them such that all the bracketed terms simultaneously go to zero (but we do not yet have any way of calculating their values yet):

$$0 = \left[\frac{\partial g}{\partial N_j} + \lambda_1 \frac{\partial \varphi_1}{\partial N_j} + \lambda_2 \frac{\partial \varphi_2}{\partial N_j} + \ldots\right] \tag{25}$$

According to the discussion following equation 12, the "g" differential is just gj. The terms involving the constraint equations are obtained by differentiating equation 21:

$$\lambda_i \frac{\partial \varphi_i}{\partial N_j} = \lambda_i a_{i,j} \tag{26}$$

Substituting into equation 25 yields:

$$0 = g_j + \sum_{i=1}^{NE} \lambda_i a_{i,j} \tag{27}$$

This gives a system of "j" equations (equation 27), one for each species. We also have "i" constraint equations, one for each element:

$$0 = \sum_{j=1}^{NS} a_{i,j} N_j - b_i$$

Finally, we have the condition that the total number of moles in the system must equal the sum of the individual mole numbers:

$$0 = \sum_{j=1}^{NS} N_j - N \tag{28}$$

Equations 21, 27, and 28 form a system of NS+NE+1 equations. Note that gj is defined by equation 18 and it contains the unknowns Nj and N. Thus the unknowns are the Nj (there are NS of these), the λi (there are NE of these), and N. Thus, the number of unknowns matches the number of equations and we have a closed algebraic system. Note that it is not necessary to hypothesize any equilibrium reactions; all you need to do is specify the species you expect to appear in your system and you can find the equilibrium solution.

A7.3 NUMERICAL SOLUTION.

The approach used by the code is to solve equations 21, 27, and 28 via a Newton Raphson method for non-linear equations. The method is outlined by Press et al. (1986). This involves some art in addition to the mathematics.

We search for the zeros of a function f:

$$0=f(x1,x2,\ldots xN) = f(X) \qquad (29)$$

To do this we first expand f as a Taylor series:

$$f(\mathbf{X}+\delta\mathbf{X}) = f(\mathbf{X}) + \sum_{i=1}^{N} \frac{\partial f}{\partial x_i}\delta x_i + O(\mathbf{X}^2) + \ldots \qquad (30)$$

Next, you neglect the higher-order terms. We are looking for the point where f(X)=0, so you want to choose your corrections (δxi) such f(X+δX)->0. Setting the left side of equation 30 to zero, you obtain:

$$\sum_{i=1}^{N} \frac{\partial f}{\partial x_i}\delta x_i = -f(\mathbf{X}) \qquad (31)$$

The idea is that you have a system of f equations (N in number) and equation 31 then expands into a matrix whose solution yields the correction values (δxi). These are then applied to the original estimates:

$$xnew = xold + \delta x \qquad (32)$$

The process is repeated until you converge. Now we enter into the art part of the problem. First, we take equation 27 and expand to open up the gj term:

$$0 = g_j^* + RT\ln\left(\frac{P}{P_0}\right) + RT\ln N_j - RT\ln N + \sum_{i=1}^{NE}\lambda_i a_{i,j} \qquad (33)$$

Next, we divide by RT to non-dimensionalize the equation:

$$0 = \frac{g_j^*}{RT} + \ln\left(\frac{P}{P_0}\right) + \ln N_j - \ln N + \sum_{i=1}^{NE}\frac{\lambda_i a_{i,j}}{RT} \qquad (34)$$

To apply equation 31 in as linear way as possible, we choose non-linear correction variables[5]. These are $\Delta\ln N_j$, $\Delta\ln N$, and $\pi_i=-(\lambda_i/RT)$. This makes the derivatives in equation 31 become:

$$\frac{\partial f_j}{\partial(\ln N_j)} = 1 \qquad (35a)$$

$$\frac{\partial f_j}{\partial(\ln N)} = -1 \qquad (35b)$$

$$\frac{\partial f_j}{\partial(\pi_i)} = -a_{i,j} \qquad (35c)$$

Substituting these into equation 31 yields[6]:

5The question is why do we not use $\Delta\pi_i$? This is a very subtle point that is more art than science. It is argued (Gordon and McBride, 1994) that the iterations are uninfluenced by starting each new iteration with the Lagrange multipliers set equal to zero. Thus, $\pi_i = \Delta\pi_i$ for each iteration. This will influence the form of equation 36.

6Note that the right side of equation 36 would normally contain the summation of $a_{i,j}\pi_i$, but since $\pi_i=0$ at the start of each iteration, these terms drop out.

$$\Delta \ln N_j - \Delta \ln N - \sum_{i=1}^{NE} a_{i,j}\pi_i = -\frac{g_j}{RT} \quad (36)$$

There will be one of these equations for each species (j=1,2,…NS). Next we go after equation 21. Here we rewrite the equation in terms of lnNj:

$$0 = \sum_{j=1}^{NS} a_{i,j} \exp(\ln N_j) - b_i \quad (37)$$

The derivative from equation 31 becomes:

$$\frac{\partial f_i}{\partial(\ln N_j)} = \sum_{j=1}^{NS} a_{i,j} \exp(\ln N_j) = \sum_{j=1}^{NS} a_{i,j} N_j \quad (38)$$

Substituting into equation 31 yields:

$$\sum_{j=1}^{NS} a_{i,j} N_j \Delta \ln N_j = b_i - \sum_{j=1}^{NS} a_{i,j} N_j \quad (39)$$

We will have one of these equations for each element (i=1,2,…NE). Finally, we turn to equation 28. Again, we convert this to a log variable:

$$0 = \sum_{j=1}^{NS} \exp(\ln N_j) - \exp(\ln N) \quad (40=$$

Taking the derivatives in equation 31 yields:

$$\frac{\partial f}{\partial(\ln N_j)} = \sum_{j=1}^{NS} \exp(\ln N_j) = \sum_{j=1}^{NS} N_j \quad (40a)$$

$$\frac{\partial f}{\partial(\ln N)} = -\exp(\ln N) = -N \quad (40b)$$

So substituting these equations into equation 31 yields:

$$\sum_{j=1}^{NS} N_j \Delta \ln N_j - N\Delta \ln N = N - \sum_{j=1}^{NS} N_j \quad (41)$$

There will be one of these equations. Equations 36, 39, and 41 can be collected together into a matrix format. This is illustrated by an example in which the system contains 5 species and 3 elements:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -a_{11} & -a_{21} & -a_{31} & -1 \\ 0 & 1 & 0 & 0 & 0 & -a_{12} & -a_{22} & -a_{32} & -1 \\ 0 & 0 & 1 & 0 & 0 & -a_{13} & -a_{23} & -a_{33} & -1 \\ 0 & 0 & 0 & 1 & 0 & -a_{14} & -a_{24} & -a_{34} & -1 \\ 0 & 0 & 0 & 0 & 1 & -a_{15} & -a_{25} & -a_{35} & -1 \\ a_{11}N_1 & a_{12}N_2 & a_{13}N_3 & a_{14}N_4 & a_{15}N_5 & 0 & 0 & 0 & 0 \\ a_{21}N_1 & a_{22}N_2 & a_{23}N_3 & a_{24}N_4 & a_{25}N_5 & 0 & 0 & 0 & 0 \\ a_{31}N_1 & a_{32}N_2 & a_{33}N_3 & a_{34}N_4 & a_{35}N_5 & 0 & 0 & 0 & 0 \\ N_1 & N_2 & N_3 & N_4 & N_5 & 0 & 0 & 0 & -N \end{bmatrix} \begin{bmatrix} \Delta\ln N_1 \\ \Delta\ln N_2 \\ \Delta\ln N_3 \\ \Delta\ln N_4 \\ \Delta\ln N_5 \\ \pi_1 \\ \pi_2 \\ \pi_3 \\ \Delta\ln N \end{bmatrix} = \begin{bmatrix} -g_1/RT \\ -g_2/RT \\ -g_3/RT \\ -g_4/RT \\ -g_5/RT \\ b_1^0 \\ b_2^0 \\ b_3^0 \\ N^0 \end{bmatrix}$$

Here:

$$b_i^0 = b_i - \sum_{j=1}^{NS} a_{i,j} N_j \qquad N^0 = N - \sum_{j=1}^{NS} N_j$$

So the approach is to make reasonable initial guesses for Nj, calculate the gj's from equation 18 (knowing P and T), solve the matrix for the correction factors, and use the correction factors to get the revised values of Nj. The process is repeated with, as mentioned in the footnotes, πi reset to zero for each iteration.

A7.4 A SLICK NUMERICAL TRICK

The difficulty here is that practical calculations for hydrocarbon air systems can involve the order of 70 species and 4 elements, resulting in a 75x75 matrix. The sparseness of the upper left corner of the matrix suggests that substitution may result in a smaller number of more complex equations to solve (also, as the systems grow, this sparse region becomes most of the matrix). With the following substitution, we can reduce this monster matrix to one that is NE+1 in size.

We start by solving equation 36 for ΔlnNj:

$$\Delta \ln N_j = \Delta \ln N + \sum_{i=1}^{NE} a_{i,j}\pi_i - \frac{g_j}{RT} \quad (42)$$

This is substituted into equation 39 in place of the ΔlnNj term:

$$\sum_{j=1}^{NS} a_{i,j}N_j\left[\Delta \ln N + \sum_{i=1}^{NE} a_{i,j}\pi_i - \frac{g_j}{RT}\right] = b_i - \sum_{j=1}^{NS} a_{i,j}N_j \quad (43)$$

We recognize that the i that appears in the first and last summation terms, and the bi relates to the equation itself, while the i in the middle summation term is actually summed. Calling the unsummed i (the equation index) as k, and rearranging gives:

$$\sum_{i=1}^{NE}\sum_{j=1}^{NS} a_{k,j}a_{i,j}N_j\pi_i + \left[\sum_{j=1}^{NS} a_{k,j}N_j\right]\Delta \ln N = b_k - \sum_{j=1}^{NS} a_{k,j}N_j + \sum_{j=1}^{NS} a_{k,j}N_j\frac{g_j}{RT} \quad (44)$$

There will be k=1,2,…,NE of these equations. We also make the same substitution of equation 42 into equation 41:

$$\sum_{j=1}^{NS} N_j\left[\Delta \ln N + \sum_{i=1}^{NE} a_{i,j}\pi_i - \frac{g_j}{RT}\right] - N\Delta \ln N = N - \sum_{j=1}^{NS} N_j \quad (45)$$

Rearranging this to put the correction variables on the left side yields:

$$\sum_{i=1}^{NE}\sum_{j=1}^{NS} a_{i,j}N_j\pi_i + \left(\sum_{j=1}^{NS} N_j - N\right)\Delta \ln N = N - \sum_{j=1}^{NS} N_j + \sum_{j=1}^{NS} N_j\frac{g_j}{RT} \quad (46)$$

There will be one of these equations. Equations 44 and 46 can be cast into a matrix form as shown above, although now the matrix will be (NE+1)x(NE+1) in size and the solution vector will be: (π1, π2,…, πNE, ΔlnN).

The solution procedure is as follows. First, you make an initial guess for Nj and N. (The solution procedure is very robust, and the standard crude initial guesses are N=0.1, and Nj=0.1/NS. (NNS is usually set a little off from the others in value to prevent a zero divide problem in the first iteration. From this convergence is almost always quickly achieved.) You use these, along with the known T and P to calculate gj from equations 16 and 18. You solve the matrix represented by equations 44 and 46 to get the correction vector, (π1, π2,…, πNE, ΔlnN). Next, you use the values from the correction vector to calculate ΔlnNj using equation 42. Then you correct each of the variables:

$$\ln N_{j,new} = \ln N_{j,old} + e\Delta \ln N_j \quad (47a)$$

$$\ln N_{new} = \ln N_{old} + e\Delta \ln N \quad (47b)$$

You iterate until you locate the equilibrium point. The factor "e" is a self-adjusting underrelaxation parameter (it varies between 0 and 1). If e=1, the solution may numerically diverge for poor initial guesses, so e is calculated via an empirical procedure outlined in Gordon and McBride (1994) that is based on the existing N and Nj. In practice, e is much less than 1 at the start of a calculation and it reaches 1 as the problem approaches convergence.

The value e is calculated as follows: Assign a parameter SIZE=-ln10-8. Then define a parameter:

$$e_1 = \frac{2}{\max(5|\Delta \ln N|, |\Delta \ln N_j|)} \quad (48)$$

For those species where ln(Nj/N)≤-SIZE, and ΔlnNj≥0, find a second parameter:

$$e_2 = \min \left| \frac{-\ln \frac{N_j}{N} - 9.2103404}{\Delta \ln N_j - \Delta \ln N} \right| \qquad (49)$$

Finally:

$$e=\min(1, e1, e2) \qquad (50)$$

A8. GENERALIZED PERFORMANCE MAPS

Generalized Gas-Turbo performances maps consist of three main functions:

A8.1 FUNCTION GASTURBO_DPTURB

This function calculates the turbine pressure ratio using non dimensional parameters:

NAME	TYPE	DESCRIPTION	UNITS
N	IN REAL	Reduced speed	-
Min	IN REAL	Inlet Mach number	-
beta	IN REAL	Main blade axe angle	degrees
N_nom	IN REAL	Nominal reduced speed	-

Formulation

Two signed parabolic zones (negative flow and normal operation) are programmed near a Mach number Mo = 0:

$$dp\_rel = sign(M_{in} - M_0)\left[\frac{M_{in} - M_0}{N_{nom}}\tan(\beta)\right]^2; \qquad N_{Nom} = \frac{r\cdot\omega_{Nom}}{cson}$$

where,

ω is the angular speed
β is the blade angle
r is a characteristic radius
cson is the sound speed in the turbine volume

The speed contribution itself is calculated as follows:

$$dp\_rel = dp\_rel - 0.2 * sign(N)\left(\frac{N}{N_{nom}}\right)^2$$

A8.2 FUNCTION GASTURBO_DPCOMP

This function calculates the compressor pressure ratio using non dimensional parameters:

NAME	TYPE	DESCRIPTION	UNITS
N	IN REAL	Reduced speed	-
Min	IN REAL	Inlet Mach number	-
beta	IN REAL	Main blade axe angle	degrees
N_nom	IN REAL	Nominal reduced speed	-

Formulation

Two signed parabolic zones (negative flow and normal operation) are programmed near a Mach number Mo = N/tan(β)

$$dp\_rel = -k\cdot sign(M_{in} - M_0)\left[\frac{M_{in} - M_0}{N_{nom}}\tan(\beta)\right]^2; \qquad N_{Nom} = \frac{r\cdot\omega_{Nom}}{cson}$$

where,

If Min > N/ta(β), k = 1. Others k = 0.13 (surge zone)
ω is the angular speed
β is the blade angle
r is a characteristic radius

cson is the sound speed in the turbine volume

The speed contribution itself is calculated as follows:

$$dp\_rel = dp\_rel + sign(N)\left(\frac{N}{N_{nom}}\right)^2$$

A8.3 FUNCTION GASTURBO_POW

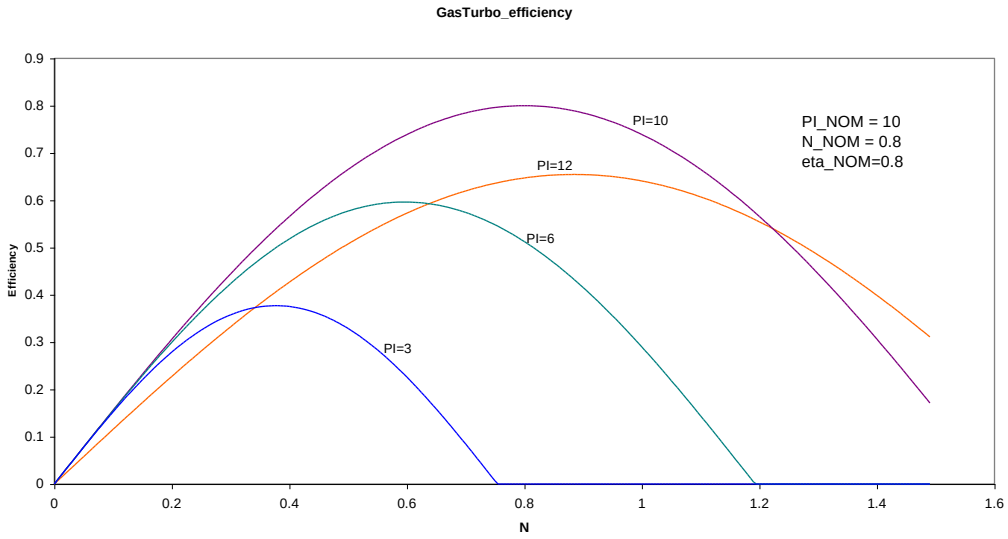
This function calculates the efficiency and the power balance of compressor and turbine components.

NAME	TYPE	DESCRIPTION	UNITS
N	IN REAL	Reduced speed	-
N_nom	IN REAL	Nominal reduced speed	-
PI_nom	IN REAL	Nominal inlet/outlet total pressure ratio	-
eta_nom	IN REAL	Nominal efficiency	-
h_in	IN REAL	Inlet enthalpy	J/Kg
h_out	IN REAL	Outlet enthalpy	J/Kg
m	IN REAL	Mass flow	Kg/s
p_in	IN REAL	Inlet pressure	Pa
p_out	IN REAL	Outlet pressure	Pa
eta	OUT REAL	Efficiency	-

Efficiency estimation:

$$eta = eta_{nom}\cdot\left(\frac{\Pi_{tt}-1}{\Pi_{nom}-1}\right)^{0.5}\cdot sin\left(\frac{0.5\cdot\pi\cdot\left(N/N_{nom}\right)}{\left((\Pi_{tt}-1)/(\Pi_{nom}-1)\right)^{0.5}}\right).$$

For Π_{tt} greeter than = Π_{nom} the factor multiplying etanom is inverted with power 1 to take into account efficiency degradation. Next figure shows the behavior of this expression:



Power balance

Depending on the pressure and flow values, it is possible to distinguish four different cases:

a) $p_{out} > p_{in}$ (Compressor)

- If the mass flow is positive (coming into the compressor volume), the power absorbed by the fluid is calculated as $W = m \cdot (H_{ise,out} - h_{in}) / eta$
- On the contrary, if the mass flow is negative, the power taken from the fluid (as a turbine) is calculated as $W = -eta \cdot m \cdot (h_{out} - H_{ise,in})$

b) $p_{out} < p_{in}$ (Turbine)

- If the mass flow is positive (incoming into the turbine volume), the power taken from the fluid is calculated as $W = eta \cdot m \cdot (H_{ise,out} - h_{in})$
- On the contrary, if the mass flow is negative, the power absorbed by the fluid (as a compressor) is calculated as $W = -m \cdot (h_{out} - H_{ise,in}) / eta$

A9. THE CAPACITIVE PIPE COMPONENT

At inlet and outlet, this component receives as input the following flow variables from the ports f1 and f2: volumetric flows Q, mixture mass flows m, non-condensable mass flows m_{nc} and mixture total enthalpic flows mH. These variables are stocked junction-wise, on the first and last junctions of the component.

The equations governing the flow within this component are solved numerically to retrieve the profiles of all the relevant flow variables.

Amongst them, the first cell and last cell values of mixture density, non-condensable mass fraction, velocity and mixture total enthalpy are used to define the output ports of the component, in a first-order approximation:

$$\begin{aligned} f1.\rho &= \rho_1 & f2.\rho &= \rho_2 \\ f1.x^{nc} &= x_1^{nc} & f2.x^{nc} &= x_2^{nc} \\ f1.u &= u_1 & f2.u &= u_2 \\ f1.H &= H_1 & f2.H &= H_2 \end{aligned}$$

The port pressures are reconstructed from the closest cell pressure and half a pressure drop due to the source terms:

$$f1.p = p_1 + \frac{1}{4}\rho_1 u_1 |u_1| A_1 \zeta_1 - \frac{g}{2}\rho_1 A_1 \Delta L_1 + qn_1$$

$$f1.p = p_{nodes} + \frac{1}{4}\rho_{nodes} u_{nodes} |u_{nodes}| A_{nodes} \zeta_{nodes} - \frac{g}{2}\rho_{nodes} A_{nodes} \Delta L_{nodes} + qn_{nodes}$$

where qn_1 and qn_{nodes} are artificial viscosity terms only used with the centred scheme. Their expression is given below in equation (9.13).

A9.1 DISCRETIZED SET OF EQUATIONS

The discretized conservation equations for mixture mass, non-condensable mass fraction and mixture energy are written:

$$\begin{aligned} &\forall i = 1,\ldots,nodes: \\ &\rho_i' V_i = f_{mass,j} - f_{mass,j+1} + S_{mass,i} \end{aligned} \tag{9.1}$$

$$\rho_i x_i^{nc\,\prime} V_i = \begin{cases} 0 & \text{if only condensable} \\ f_{nc,j} - f_{nc,j+1} + S_{nc,i} - \rho_i' x_i^{nc} V_i & \text{else} \end{cases} \tag{9.2}$$

$$\left(\rho_i' E_i + \rho_i E_i'\right) V_i = f_{ene,j} - f_{ene,j+1} + S_{ene,i} \tag{9.3}$$

This general form is identical whether the domain is collocated or staggered.

Considering the cells are numbered by *i=1,...,nodes*, with $x=x_i$ being the cell center location, and the junctions are numbered by *j=1,...,nodes+1* and located at $x=x_j=x_{i-1/2}=x_i-1/2\Delta L_i$. In a collocated configuration, all the variables are discretized in the cell centers x_i. In a staggered configuration, the state variables (pressure, density, velocity, etc) are discretized in the cell centers x_i, and the flow variables (mass flow, enthalpy flow, etc) are discretized in the cell interfaces $x_j=x_{i-1/2}$.

If the scheme chosen is centred, the mixture momentum equation is discretized following the staggered approach:

$$\dot{m}_j' \Delta L_j = f_{mom,i-1} - f_{mom,i} + S_{mom,j} \quad \forall j = 2,\ldots,\text{nodes} \tag{9.4}$$

where the dual cell size is given by $\Delta L_j = \frac{1}{2}(\Delta L_{i-1} + \Delta L_i)$. The first and last junction-wise mass flows are directly computed from the port mass flows:

$$\dot{m}_1 = f1.\dot{m}$$
$$\dot{m}_{nodes+1} = -f2.\dot{m}$$

where the minus sign comes from the sign convention of the fluid network.

If the upwind scheme is selected, the momentum equation is discretized following the collocated approach:

$$\dot{m}_i' \Delta L_i = f_{mom,j} - f_{mom,j+1} + S_{mom,i} \qquad \forall i = 1,\ldots,nodes \tag{9.5}$$

A9.2 CENTRED SCHEME

1/. Flux terms. The inner fluxes are computed through calls to a dedicated function. This function gives as output:

$$f_{mass,j} = \dot{m}_j \qquad \forall j = 2,\ldots,nodes \tag{9.6}$$
$$f_{nc,j} = \dot{m}_j^{nc} \qquad \forall j = 2,\ldots,nodes \tag{9.7}$$
$$f_{mom,j} = \left(p_i + qn_i + \rho_i u_i^2\right)A_i \qquad \forall j = 1,\ldots,nodes \tag{9.8}$$
$$f_{ene,j} = \dot{m}H_j \qquad \forall j = 2,\ldots,nodes \tag{9.9}$$

where the following equations are applicable for the expressions of $\dot{m}_j^{nc}$, $\dot{m}H_j$ and qn_i :

$$\dot{m}_j^{nc} = \dot{m}_j \text{donor\_cell}(\dot{m}_j, x_{i-1}^{nc}, x_i^{nc}) \qquad \forall j = 2,\ldots,nodes \tag{9.10}$$
$$\dot{m}H_j = \dot{m}_j \text{donor\_cell}(\dot{m}_j, H_{i-1}, H_i) \qquad \forall j = 2,\ldots,nodes \tag{9.11}$$
$$\dot{m}_j^{nc} = \frac{1}{2A_i}\left(\frac{\dot{m}_j}{\text{donor\_cell}(\dot{m}_j,\rho_{i-1},\rho_i)} + \frac{\dot{m}_{j+1}}{\text{donor\_cell}(\dot{m}_{j+1},\rho_{i1},\rho_{i+1})}\right) \qquad \forall i = 2,\ldots,nodes-1 \tag{9.12}$$
$$qn_i = -\max(100|u_i|,1)k_d Damp \frac{\dot{m}_{j+1} + \dot{m}_j}{A_i} \max(|u_i|, c_i) \qquad \forall i = 1,\ldots,nodes \tag{9.13}$$

where *Damp* is a free parameter of order 1, and k_d is a multiplier. Still to define are the expressions of velocity for the first and last cell. They are given by:

$$u_1 = \frac{1}{2A_1}\left(f1.Q + \frac{\dot{m}_2}{\text{donor\_cell}(\dot{m}_2,\rho_1,\rho_2)}\right) \tag{9.14}$$

$$u_{nodes} = \frac{1}{2A_{nodes}}\left(-f2.Q + \frac{\dot{m}_{nodes}}{\text{donor\_cell}(\dot{m}_{nodes},\rho_{nodes-1},\rho_{nodes})}\right) \tag{9.15}$$

Moreover, the mixture mass, non-condensable mass and energy numerical fluxes still need values for the first and last junctions. These are naturally given by the port flow variables:

$$f_{mass,1} = f1.\dot{m} \qquad f_{mass,nodes+1} = -f2.\dot{m}$$
$$f_{nc,1} = f1.\dot{m}^{nc} \qquad f_{nc,nodes+1} = -f2.\dot{m}^{nc}$$
$$f_{ene,1} = f1.\dot{m}H \qquad f_{ene,nodes+1} = -f2.\dot{m}H$$

2/. Source terms. Using the centred scheme, all the source terms are evaluated point wise:

$$S_{mass,i} = -\rho_i \kappa_{wi} dp_i V_i \qquad \forall i = 1,\ldots,nodes \tag{9.16}$$

$$S_{nc,i} = -\rho_i x_i^{nc} \kappa_{wi} dp_i V_i \qquad \forall i = 1,...,nodes \tag{9.17}$$

$$S_{mom,j} = -\frac{1}{4}\left(\rho_{i-1}u_{i-1}|u_{i-1}|A_{i-1}\zeta_{i-1} + \rho_i u_i|u_i|A_i\zeta_i\right) + \frac{g}{2}\left(\rho_{i-1}V_{i-1} + \rho_i V_i\right) + \frac{1}{2}\left(p_{i-1} + p_i\right)\left(A_i - A_{i-1}\right) \quad \forall \mathrm{i} = 2,..., \mathrm{nodes} \tag{9.18}$$

$$S_{ene,i} = q_i \qquad \forall i = 1,...,nodes \tag{9.19}$$

where:

- κ_w is the wall compressibility factor, function of the pipe material, diameter and wall thickness;
- $V_i = A_i \Delta L_i$ is the volume of the cell *i*;
- *dp* represents the time variation of the pressure due to wall compressibility. It is approximated by $\rho' / \left.\frac{\partial \rho}{\partial p}\right|_h$;
- As the mixture momentum equation is discretized on a staggered mesh, the corresponding friction term $-\frac{1}{2}\rho_j u_j |u_j| A_j \zeta_j$ in junction *j* is approximated by an arithmetic average of the counterparts in cell centers *i-1* and *i*:

$$-\frac{1}{2}\rho_j u_j |u_j| A_j \zeta_j = -\frac{1}{4}\left(\rho_{i-1}u_{i-1}|u_{i-1}|A_{i-1}\zeta_{i-1} + \rho_i u_i |u_i| A_i \zeta_i\right) \tag{9.20}$$

- same for the gravity term $\rho_j g A_j \Delta L_j$:

$$\rho_j g A_j \Delta L_j = \frac{g}{2}\left(\rho_{i-1}A_{i-1}\Delta L_{i-1} + \rho_i A_i \Delta L_i\right) \tag{9.21}$$

where g represents the gravitational acceleration, if any. It is computed as the scalar product of $\bar{g} = \left(g_x, g_y, g_z\right)$ with the direction of the pipe in the global axis system (*x,y,z)*:

$$g = \frac{1}{L}\left(g_x \Delta x + g_y \Delta y + g_z \Delta z\right) \tag{9.22}$$

For a pipe of length $L = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$.

- $\left(p\Delta A\right)_j$ is the extra source term accounting for the variable cross-section effect. There can be several ways to implement this term, and here the following formulation has been chosen:

$$\left(p\Delta A\right)_j = \frac{1}{2}\left(p_{i-1} + p_i\right)\left(A_i - A_{i-1}\right) \tag{9.23}$$

- *q* is the heat transfer to the wall. It is computed through the use of a Fourier law.

A9.3 UPWIND SCHEME

1/. Flux terms. The inner junction fluxes are computed through calls to a dedicated function following the upwind Roe scheme. Here follows a brief description of this function.

The Roe numerical flux on the junction *j=i-1/2* is computed from this first order expression:

$$\vec{f}_{i-1/2} = \frac{1}{2}\left(\vec{f}\left(\vec{u}_{i-1}\right) + \vec{f}\left(\vec{u}_i\right)\right) - \frac{1}{2}\left|\tilde{J}_{i-1/2}\right|\Delta\vec{u}_{i-1/2} \tag{9.24}$$

where **u** is the vector of conservative variables in cell *i*. The centred part of these expressions is an average of fluxes written as:

$$f_{mass}^{central}(\vec{u}_i) = \rho_i u_i A_i \tag{9.25}$$

$$f_{nc}^{central}(\vec{u}_i) = \rho_i u_i x_i^{nc} A_i \tag{9.26}$$

$$f_{mom}^{central}(\vec{u}_i) = (p_i + \rho_i u_i^2) A_i \tag{9.27}$$

$$f_{ene}^{central}(\vec{u}_i) = \rho_i u_i H_i A_i \tag{9.28}$$

The upwind part $-\frac{1}{2}\left|\tilde{J}_{i-\frac{1}{2}}\right|\Delta\bar{u}_{i-\frac{1}{2}}$ is implemented through the eigenstructure of the linearized Jacobian matrix, also called Roe matrix $\tilde{J} = \tilde{R}\left|\tilde{\Lambda}\right|\tilde{L}$:

$$\left|\tilde{J}_{i-1/2}\right|\Delta\vec{u}_{i-1/2} = \sum_{k=1}^{4}\tilde{\alpha}^k\left|\tilde{\lambda}^k\right|\tilde{R}^k \tag{9.29}$$

where $\tilde{x}$ represents the Roe average of the quality *x* between states *i-1* and *i*, $\tilde{R}$ are the right eigenvectors of the matrix, $\tilde{\lambda}^k$ its eigenvalues, and $\tilde{\alpha}^k$ are the wave strengths computed as:

$$\tilde{\vec{a}} = \left(\tilde{\alpha}^1, \tilde{\alpha}^2, \tilde{\alpha}^3, \tilde{\alpha}^4\right)^T = \tilde{L}\Delta\vec{u} \tag{9.30}$$

where $\tilde{L} = \left(\tilde{L}^1, \tilde{L}^2, \tilde{L}^3, \tilde{L}^4\right)$ is the left eigenmatrix regrouping the left eigenvectors of the linearized Jacobian.

As a first step to compute $\tilde{R}$, $\tilde{L}$, $\tilde{\lambda}^k$ and $\tilde{\alpha}^k$, the pressure derivatives with respect to conservative variables need to be computed. Let $\pi = \pi(\rho, \rho x^{nc}, \rho u, \rho E)$ be the functional dependence of the pressure to the conservative variables, its derivatives π_ρ, $\pi_{\rho x^{nc}}$, $\pi_{\rho u}$ and $\pi_{\rho E}$ are computed in a thermodynamic function, taking into account the flow case (possibly two-component and two-phase). Then the Roe averages needed to construct these eigenmatrices are computed:

- Those of some primitive variables, i.e. the velocity $\tilde{u}$, mixture total enthalpy $\tilde{H}$, non condensable mass fraction $\tilde{x}^{nc}$ and mixture density $\tilde{\rho}$.
- Those of the pressure derivatives with respect to conservative variables, i.e. $\tilde{\pi}_\rho$, $\tilde{\pi}_{\rho x^{nc}}$, $\tilde{\pi}_{\rho u}$ and $\tilde{\pi}_{\rho E}$. This is done by numerical integration. Several integration methods are provided, and a correction on the integrated averages is also possible (available through the *expert_mode* flow case in the GUI).
- If there is a phase transition between cells *i-1* and *i*, these averages are computed appropriately, taking into account the phase transition in their derivation.
- From these averages the averaged mixture sound speed $\tilde{c}$ is derived

An entropy fix is performed on $\left|\tilde{\lambda}^k\right|$ in order to avoid unphysical expansion shocks. Several entropy fixes are available through the *expert_mode* flow case in the GUI.

A switch in the GUI enables using either the first order numerical flux (9.24), allowing piecewise constant solution profiles, or a high resolution version, allowing a piecewise quadratic solution in smooth regions, and switching back to piecewise constant near strong gradients, thanks to the use of slope limiters.

The high resolved numerical flux of Roe follows this expression:

$$\vec{f}_{i-1/2}^{\,hr} = \frac{1}{2}\left(\vec{f}\left(\bar{\vec{u}}_{i-1/2}^{+}\right) + \vec{f}\left(\bar{\vec{u}}_{i-1/2}^{-}\right)\right) - \frac{\left|\tilde{J}_{i-1/2}\right|}{2}\Delta\vec{u}_{i-1/2}^{\,hr} \tag{9.31}$$

where $\bar{\vec{u}}_{i-1/2}^{+}$, $\bar{\vec{u}}_{i-1/2}^{-}$ and $\Delta u_{i-1/2}^{hr} = \bar{\vec{u}}_{i-1/2}^{+} - \bar{\vec{u}}_{i-1/2}^{-}$ are variables reconstructed at the left and right side of the junction *j-1/2* by the MUSCL approach (Monotone Upstream-centered Scheme for Conservation Laws), and then limited by an appropriate slope limiter.

When selecting the high order upwind scheme, the upwind part is always reconstructed. Through the *expert_mode*, the experimented user can decide whether or not to reconstruct as well the central part of the flux, which limiter to use, and which set of variables to reconstruct (primitive or conservative).

Finally, a last *expert_mode* option concerns the preconditioning: for low Mach number flows, selecting the *precond* parameter to *altern* could improve the accuracy of the results. Select a Turkel preconditioning helps only for steady computations. By default, no preconditioning is applied.

2/. Source terms. Using the upwind scheme, all the source terms that do not initially contain spatial derivatives can be implemented in a pointwise evaluation:

$$\forall i = 1,\ldots,nodes: \tag{9.32}$$
$$S_{mass,i} = -\rho_i \kappa_{wi} dp_i V_i$$

$$S_{nc,i} = -\rho_i x_i^{nc} \kappa_{wi} dp_i V_i \tag{9.33}$$

$$S_{mom,i} = \rho_i g V_i - \frac{1}{2}\rho_i u_i |u_i| A_i \zeta_i \tag{9.34}$$

$$S_{ene,i} = q_i \tag{9.35}$$

In order to satisfy the balance of equations at each time step, the last term $(p\Delta A)_i$, namely the area-variation term of the mixture momentum conservation equation, should be implemented using the same stencil [*j,j+1*] as the flux terms: as the flux term derivative $\left.\frac{\partial \vec{f}}{\partial x}\right|_i$ is discretized on two junctions *i-1/2* and *i+1/2*:

$$\left.\frac{\partial \vec{f}}{\partial x}\right|_i = \frac{\vec{f}_{i+1/2} - \vec{f}_{i-1/2}}{dx} \tag{9.36}$$

Likewise, in an equation discretized in cell *I*, the source term ss requiring an upwinding must first be split into the closest junction-wise corresponding terms:

$$S_i = S^{+}_{j=i-1/2} + S^{-}_{j+1=i+1/2} \tag{9.37}$$

$S^{+}_{i-1/2}$ is the upwinded part of $S_{i-1/2}$ and represents the part of the source term $S_{j=i-1/2}$ upwinded from the left cell *i-1*. Similarly, $S^{-}_{i+1/2}$ is the part of $S_{i+1/2}$ downwinded from the right cell *i+1*. For the case of the area-variation term $(p\Delta A)_i$, it is split this way:

$$(p\Delta A)_i = (p\Delta A)^{+}_{j=i-1/2} + (p\Delta A)^{-}_{j+1=i+1/2} \tag{9.38}$$

Here the term $(p\Delta A)_{j=i-1/2}$ is discretized as:

$$(p\Delta A)_j = \left(\frac{p}{A}\Delta A\right)_j = \frac{p_i A_i + p_{i-1} A_{i-1}}{A_i + A_{i-1}}\left(A_i - A_{i-1}\right) \tag{9.39}$$

Moreover, the source term should be also discretized in the same way the flux terms were discretized, i.e. using an upwind approach based on eigenvalue absolute values, see (9.29). This is done through a correct derivation of $S^{+}_{i-1/2}$ and $S^{-}_{i+1/2}$. The way the upwinding from $S_{j=i-1/2}$ to $S^{+}_{i-1/2}$ is performed follows the approach used for the numerical flux.

All junction-wise source terms are gathered into a vector, and this vector S_j (or simply *S*) is first decomposed, using the left eigenvectors $\tilde{L}^k$ of Roe's matrix to define the characteristic source term S_{char}, in a similar way the wave strength alpha were defined, recall (9.30):

$$S^k_{ichar} = \sum_{i=1}^{4} L^k_i S^i \tag{9.40}$$

The upwinding is then performed on this characteristic source terms, following the rule:

$$S^{k,+}_{char} = \begin{cases} S^{k,}_{char} & \text{if} \quad \tilde{\lambda}^k \geq 0 \\ 0 & \text{if} \quad \tilde{\lambda}^k < 0 \end{cases} \tag{9.41}$$

$$S^{k,-}_{char} = \begin{cases} 0 & \text{if} \quad \tilde{\lambda}^k \geq 0 \\ S^{k,}_{char} & \text{if} \quad \tilde{\lambda}^k < 0 \end{cases}$$

Note that the upwinding is indeed done on the sign of the propagative speeds $\tilde{\lambda}^k$ and not on the sign of the actual characteristic source term S_{char}.

Practically in the code, this splitting is implemented as follows:

$$\begin{aligned} S^{k,+}_{char} &= \frac{1}{2} S^k_{char} \left(1 + sign2\left(\tilde{\lambda}^k, \varepsilon_{lin}\right)\right) \\ S^{k,-}_{char} &= \frac{1}{2} S^k_{char} \left(1 - sign2\left(\tilde{\lambda}^k, \varepsilon_{lin}\right)\right) \end{aligned} \tag{9.42}$$

Where $sign2\left(\tilde{\lambda}^k, \varepsilon_{lin}\right)$ is similar to the function *sign(x)*, giving -1 for *x<0* and 1 for *x>=0*, but with a linear smoothing of extent ε_{lin} around *0*, in order to provide a smooth transition. ε_{lin} is governed by the numerical parameter *eps_lin* and is $O(\tilde{c})$.

This numerical artifact is essential for nozzle transonic flows. One can see it as a transonic fix for the source upwinding, in the same way the flux upwinding needs an entropy fix for transonic flows. This parameter can be modified by the user through the *expert_mode* flow case selection in the GUI (*source_upwinding_smoothing*). Usually a value of *0* is default for gas flows, whereas a value of 20 is necessary for liquid flows to smooth the upwinding.

Finally the characteristic upwinded source terms are transformed back into conservative variables:

$$S^{k,\pm} = \sum_{i=1}^{4} \tilde{R}^{k}_{\ i} S^{i,\pm}_{char} \tag{9.43}$$

These two upwinded source vectors are combined and added to the cell-wise source terms that did not require upwinding:

$$S^{total}_i = S^{+}_{i-1/2} + S^{-}_{i+1/2} + S_i \tag{9.44}$$

A10. THE RESISTIVE PIPE COMPONENT

At inlet and outlet, this component receives as input the following state variables from the ports f1 and f2: mixture density ρ , mixture pressure *p*, non-condensable mass fraction x^{nc} , velocity *u* and mixture total enthalpy *H*. These variables are stocked junction-wise, on the first and last junctions of the component. The output of this resistive component is the following flow variables defined in the ports f1 and f2: volumetric flows *Q*, mixture mass flows $\dot{m}$, non-condensable mass flows $\dot{m}^{nc}$ and mixture total enthalpic flows $\dot{m}H$. These are computed this way:

$$f1.\dot{m}H = f1.\dot{m} * f1.H \qquad f2.\dot{m}H = f2.\dot{m} * f2.H$$
$$f1.Q = f1.\dot{m} / f1.\rho \qquad f2.Q = f2.\dot{m} / f2.\rho$$
$$f1.\dot{m}^{nc} = f1.\dot{m} * f1.x^{nc} \qquad f2.\dot{m}^{nc} = f2.\dot{m} * f2.x^{nc}$$

The mass flows $f1.\dot{m}$ and $f2.\dot{m}$ are required. If the centred scheme is selected, they are defined using junction-wise mass flows:

$$f1.\dot{m} = -\dot{m}_1$$
$$f2.\dot{m} = \dot{m}_{nodes+1}$$

where the minus sign comes from the sign convention in the fluid network. Ideally $f1.\dot{m}$ and $f2.\dot{m}$ should be equal as the mass flow is conserved through all the components (if the computation converged). If the upwind scheme is selected, this equality is directly imposed:

$$f1.\dot{m} = -f2.\dot{m}$$
$$f2.\dot{m} = \frac{\sum_{i=1}^{nodes} \dot{m}_i}{nodes}$$

where $\dot{m}_i$ are here cell-wise mass flows.

A10.1 DISCRETIZED SET OF EQUATIONS

The discretized conservation equations for mixture mass, non-condensable mass fraction and mixture energy are written:

$$\forall i = 1,...,nodes:$$
$$\rho_i^{'} V_i = f_{mass,j} - f_{mass,j+1} + S_{mass,i} \tag{10.1}$$

$$\rho_i x_i^{nc'} V_i = \begin{cases} 0 & \text{if only condensable} \\ f_{nc,j} - f_{nc,j+1} + S_{nc,i} - \rho_i^{'} x_i^{nc} V_i & \text{else} \end{cases} \tag{10.2}$$

$$\left(\rho_i^{'} E_i + \rho_i E_i^{'}\right) V_i = f_{ene,j} - f_{ene,j+1} + S_{ene,i} \tag{10.3}$$

This general form is identical whether the domain is collocated or staggered.

Considering the cells are numbered by *i=1, ...,nodes*, with $x = x_i$ being the cell center location, and the junctions are numbered by *j=1,...,nodes+1* and located at $x = x_j = x_{i-\frac{1}{2}} = x_i - \frac{1}{2}\Delta L_i$. In a collocated configuration, all the variables are discretized in the cell centers x_i . In a staggered configuration, the state

variables (pressure, density, velocity, etc) are discretized in the cell centers x_i, and the flow variables (mass flow, enthalpy flow, etc) are discretized in the cell interfaces $x_j = x_{i-\frac{1}{2}}$.

If the selected scheme is the centred one, the mixture momentum equation is discretized following the staggered approach:

$$\dot{m}_j' \Delta L_j = f_{mom,i-1} - f_{mom,i+1} + S_{mom,j} \qquad \forall j = 1,\ldots,\text{nodes}+1 \tag{10.4}$$

where the dual cell size is given by $\Delta L = \frac{1}{2}\left(\Delta L_{i-1} + \Delta L_i\right)$. If the upwind scheme is selected, that equation is discretized following the collocated approach:

$$\dot{m}_i' \Delta L_i = f_{mom,j} - f_{mom,j+1} + S_{mom,i} \qquad \forall \text{i} = 1,\ldots,\text{nodes} \tag{10.5}$$

The numerical fluxes and source terms are detailed in the following sections.

A10.2 CENTRED SCHEME

1/. Flux terms. The flux terms are computed through calls to a dedicated function, giving as output:

$$f_{mass,j} = \dot{m}_j \qquad \forall \text{j} = 1,\ldots,\text{nodes}+1 \tag{10.6}$$

$$f_{nc,j} = \dot{m}_j^{nc} \qquad \forall \text{j} = 1,\ldots,\text{nodes}+1 \tag{10.7}$$

$$f_{mom,j} = \left(p_i + \text{qn}_i + \rho_i \text{u}_\text{i}^2\right)\text{A}_i \qquad \forall \text{j} = 0,\ldots,\text{nodes}+1 \tag{10.8}$$

$$f_{ene,j} = \dot{m}H_j \qquad \forall \text{j} = 1,\ldots,\text{nodes}+1 \tag{10.9}$$

where the following equations are applicable for the expressions of $\dot{m}_j^{nc}$, $\dot{m}H_j$, u_j and qn_i:

$$\dot{m}_j^{nc} = \dot{m}_j \text{donor\_cell}\left(\dot{m}_j, x_{i\text{-}1}^{nc}, x_i^{nc}\right) \qquad \forall \text{j} = 1,\ldots,\text{nodes} \tag{10.10}$$

$$\dot{m}H_j = \dot{m}_j \text{donor\_cell}\left(\dot{m}_j, H_{i\text{-}1}, H_i\right) \qquad \forall \text{j} = 1,\ldots,\text{nodes} \tag{10.11}$$

$$\dot{m}_j^{nc} = \frac{1}{2A_i}\left(\frac{\dot{m}_j}{\text{donor\_cell}\left(\dot{m}_j, \rho_{i\text{-}1}, \rho_i\right)} + \frac{\dot{m}_{j+1}}{\text{donor\_cell}\left(\dot{m}_{j+1}, \rho_{i1}, \rho_{i+1}\right)}\right) \qquad \forall \text{i} = 1,\ldots,\text{nodes} \tag{10.12}$$

$$qn_i = -\max\left(100\left|u_i\right|, 1\right) k_d \, Damp \, \frac{\dot{m}_{j+1} + \dot{m}_j}{A_i} \max\left(\left|u_i\right|, c_i\right) \qquad \forall \text{i} = 1,\ldots,\text{nodes} \tag{10.13}$$

where *Damp* is a free parameter of order 1, and k_d is a multiplier. It means the values of non-condensable mass fraction x^{nc}, total enthalpy *H*, velocity *u*, density ρ, pressure *p*, cross-section *A* and artificial viscosity *qn* need to be known for the indexes *i=0* and *i=nodes+1*.

This is done through the application of ghost cells, which centers are located on *i=0* and *i=nodes+1*, and for which all needed properties are computed from the boundary conditions and the inner cell values.

2/. Source terms.

Using the centred scheme, all the source terms are evaluated point wise:

$$S_{mass,i} = -\rho_i \kappa_{wi} dp_i V_i \qquad \forall i = 1,\ldots,nodes \tag{10.16}$$

$$S_{nc,i} = -\rho_i x_i^{nc} \kappa_{wi} dp_i V_i \qquad \forall i = 1,\ldots,nodes \tag{10.17}$$

$$S_{mom,j} = -\frac{1}{4}\left(\rho_{i-1}u_{i-1}|u_{i-1}|A_{i-1}\zeta_{i-1} + \rho_i u_i|u_i|A_i\zeta_i\right) + \frac{g}{2}\left(\rho_{i-1}V_{i-1} + \rho_i V_i\right) + \frac{1}{2}\left(p_{i-1} + p_i\right)\left(A_i - A_{i-1}\right) \quad \forall \mathrm{i} = 2,\ldots,\mathrm{nodes} \tag{10.18}$$

$$S_{ene,i} = q_i \qquad \forall i = 1,\ldots,nodes \tag{10.19}$$

where:

- κ_w is the wall compressibility factor, function of the pipe material, diameter and wall thickness;
- $V_i = A_i\Delta L_i$ is the volume of the cell *i*;
- *dp* represents the time variation of the pressure due to wall compressibility. It is approximated by $\rho' / \left.\frac{\partial\rho}{\partial p}\right|_h$;
- As the mixture momentum equation is discretized on a staggered mesh, the corresponding friction term $-\frac{1}{2}\rho_j u_j|u_j|A_j\zeta_j$ in junction *j* is approximated by an arithmetic average of the counterparts in cell centers *i-1* and *i*:

$$-\frac{1}{2}\rho_j u_j|u_j|A_j\zeta_j = -\frac{1}{4}\left(\rho_{i-1}u_{i-1}|u_{i-1}|A_{i-1}\zeta_{i-1} + \rho_i u_i|u_i|A_i\zeta_i\right) \tag{9.20}$$

- same for the gravity term $\rho_j g A_j \Delta L_j$:

$$\rho_j g A_j \Delta L_j = \frac{g}{2}\left(\rho_{i-1}A_{i-1}\Delta L_{i-1} + \rho_i A_i \Delta L_i\right) \tag{9.21}$$

where g represents the gravitational acceleration, if any. It is computed as the scalar product of $\bar{g} = \left(g_x, g_y, g_z\right)$ with the direction of the pipe in the global axis system (*x,y,z)*:

$$g = \frac{1}{L}\left(g_x\Delta x + g_y\Delta y + g_z\Delta z\right) \tag{9.22}$$

For a pipe of length $L = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$.

- $\left(p\Delta A\right)_j$ is the extra source term accounting for the variable cross-section effect. There can be several ways to implement this term, and here the following formulation has been chosen:

$$\left(p\Delta A\right)_j = \frac{1}{2}\left(p_{i-1} + p_i\right)\left(A_i - A_{i-1}\right) \tag{9.23}$$

- *q* is the heat transfer to the wall. It is computed through the use of a Fourier law.

A10.3 UPWIND SCHEME

1/. Flux terms. All the numerical fluxes are computed through calls to a dedicated function following the upwind Roe scheme. This method is fully described in Appendix9.

2/. Source terms. When the upwind scheme is selected, all the source terms follow exactly the same discretization as for the capacitive pipe described in Appendix9. Please refer to that appendix for details.

A11. ASSESSMENT OF THE COMPONENT JUNCTION

Ref:http://www.daerospace.com/HydraulicSystems/OrificeFlowDesc.php

An orifice is a sudden flow restriction of short length (zero length for sharp edge orifice). Orifices are treated as either a sharp edge orifice or a short tube orifice. Orifices are primarily used to control flow or create a pressure differential (drop). Orifices may be fixed or variable (valve). Many types of valves and flow devices can essentially be viewed as orifices. In valves, there can be numerous flow passages, but usually somewhere in the flow passage is a restriction that controls flow, which is why a valve often behaves like an orifice.

Flow characterization for orifices is done mathematically. However, by understanding the nature of the mathematical equations the behavior of fluid flow in orifices (as well as pipes and servos) can be understood and intuition developed. Also, the equations can be used to model component and system behavior for enhanced analysis and understanding. The orifice flow equation is a key equation for hydraulic systems.

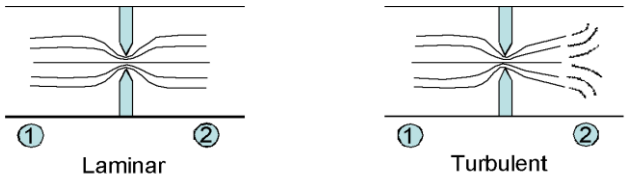


Figure 1 Flow Streamlines for Laminar and Turbulent Orifice Flow

Like pipe flow, fluid flow in orifices can be either laminar or turbulent (see Figure 1).

In laminar flow, each fluid particle follows a well defined trajectory, with velocity only in the direction of flow.

In turbulent flow (most common in hydraulic systems due to small line diameters and small orifices) each particle flows in the general direction (velocity) of the flow, but is subjected to fluctuating cross current velocities. Equations for computing orifice flow are different for laminar and turbulent flow. Determination of laminar or turbulent is determined using the Reynolds number,

$$\mathrm{Re} = \frac{\rho v d_h}{\mu} = \frac{v d_h}{\upsilon} \qquad d_h = \frac{4A}{S} \qquad (1)$$

where Dh is the hydraulic diameter; A flow section area; S flow section perimeter; v flow velocity; μ dynamic viscosity v kinematic viscosity

For low values of Re, flow is laminar. For high values of Re, flow is turbulent.

From Bernoulli's equation, the total energy loss is the energy converted to heat by friction of particles against the wall and each other

$$\Delta p = (p_1 + 1/2\rho v_1^2 + \rho g z_1) - (p_2 + 1/2\rho v_2^2 + \rho g z_2) \qquad (2)$$

Assuming, away from the orifice, that v1 = v2 and A1 = A2, the flow becomes a product of the area and speed

$$Q = Av = A\sqrt{\frac{2}{\rho\xi}\left(p_1 - p_2\right)} \qquad (3)$$

where ξ is a dimensionless loss coefficient, representing the energy loss associated with the pressure drop. For hydraulic systems, this equation is normally written as

$$Q = \alpha_d A\sqrt{\frac{2}{\rho}\left(p_1 - p_2\right)} \qquad (4)$$

where α_d is the discharge coefficient and represents the energy loss in the fluid. Equation (4) is the orifice flow equation. The discharge coefficient is the key element to estimate for laminar and turbulent flow regimes. Inspection of the equation (4) indicates that the flow rate varies proportionally with area if the Δp

is held constant, and that the flow rate varies with the square root of Δp if the flow area is held constant. Figure 2 shows notional charts of the flow behavior.

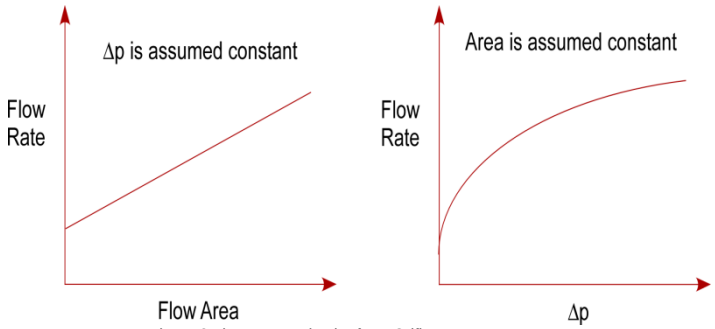


Figure 2 Flow Rate Behavior for a Orifice

Turbulent Orifice Flow: For a sharp edge orifice, with turbulent flow and with orifice flow area, Ao << A (pipe flow area), the theoretical αd is:

$$\alpha_d = \pi/(\pi+2) = .611 \qquad (5)$$

For short tube orifices of length L, pipe diameter d, and orifice diameter do, Figure 3 graphically shows the variation in α_d for two equations.

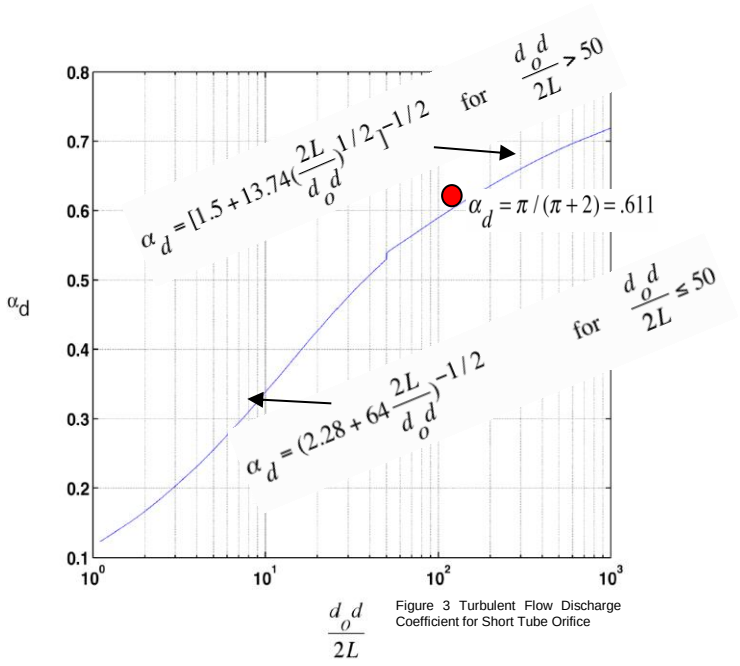


Figure 3 Turbulent Flow Discharge Coefficient for Short Tube Orifice

Laminar Orifice Flow: Equation

$$Q = \alpha_d A \sqrt{\frac{2}{\rho}\left(p_1 - p_2\right)}$$

can be used in the turbulent-laminar (transition) region and the laminar flow region using

$$\alpha_d = \delta\sqrt{\mathrm{Re}} \qquad (6)$$

where

$$\delta = \frac{\alpha_{d,turb}}{\sqrt{\mathrm{Re}_{crit}}} = \frac{0.611}{\sqrt{20}} \approx 0.137 \quad \text{(Sharp Edged Orifice)}$$

$$\delta = \frac{\alpha_{d,turb}}{\sqrt{\mathrm{Re}_{crit}}} = \frac{0.611}{\sqrt{80}} \approx 0.068 \quad \text{(Rounded Off Orifice)}$$

δ is called the laminar flow coefficient and depends on orifice geometry. αd,turb comes from equation (5) $\alpha_d = \pi/(\pi+2) = .611$. Recrit is the critical (breakpoint) Reynolds number found during empirical testing for the type of orifice (see Figure 4). These equations are theoretical, but have been validated by experiment.

Equation (6) is valid up to the critical Reynolds number (see Figure 4). Experimental data for α_d as a function of Recrit for various orifices are shown in the figure below.

The sloped lines can be used for α_d in the laminar flow region. At the breakpoint, turbulent flow occurs and α_d = 0.611 should be used.

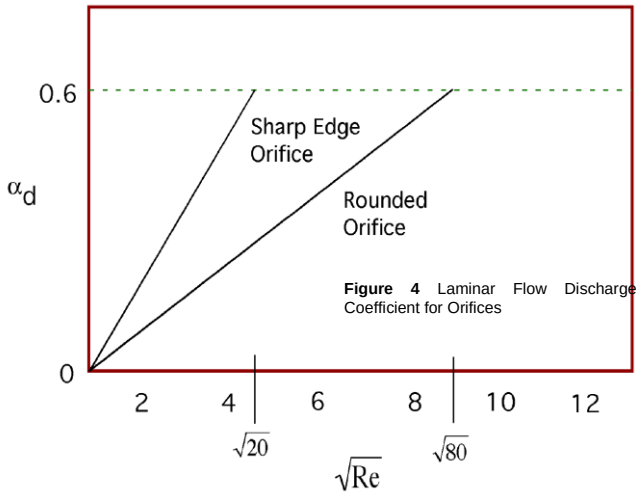


Figure 4 Laminar Flow Discharge Coefficient for Orifices

CONCLUSION for ESPSS: The real laminar behavior in sharp orifices may occur for ESPSS.Re_lam ≈ 53 (=1/ δ^2 with δ=0.137) or for rounded orifices ESPSS.Re_lam= 216 (with δ=0.068).