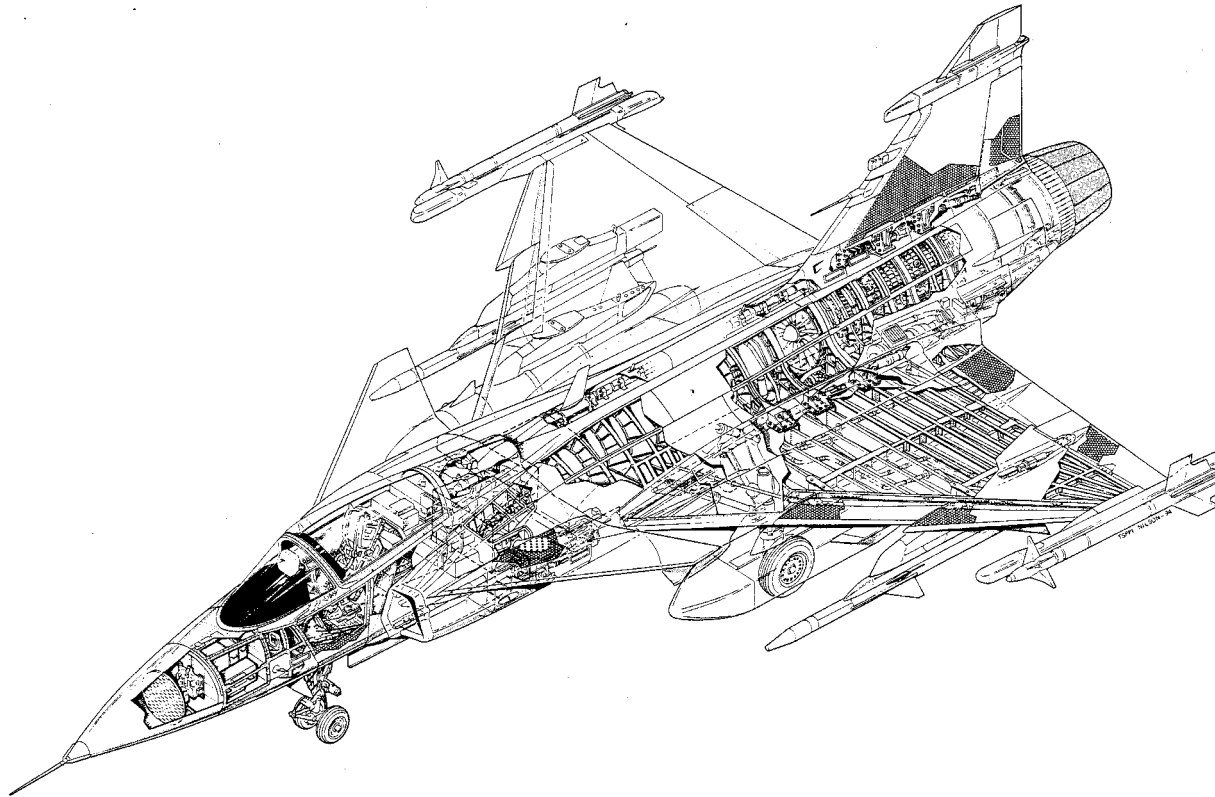


# Fuel-Air Tank System Library

Hans Ellström, SAAB, Linköping. Sweden

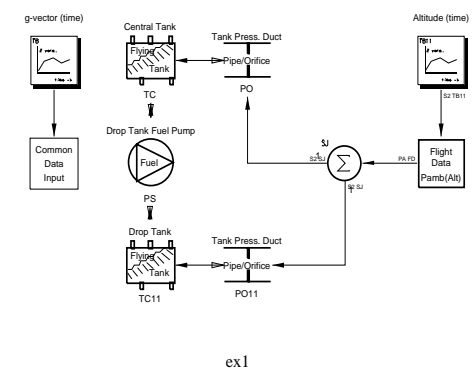
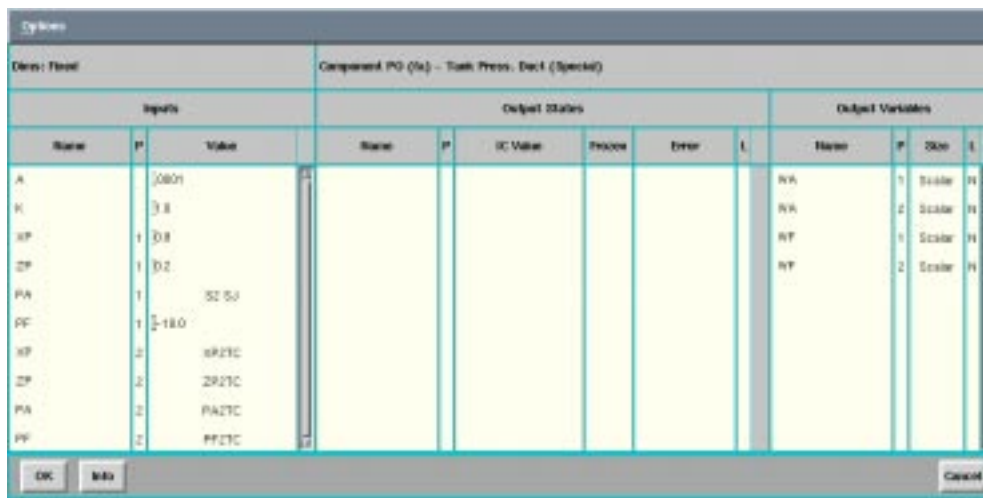


*Saab 39 Gripen*

# Transfer of old Fortran code (extract from input data)

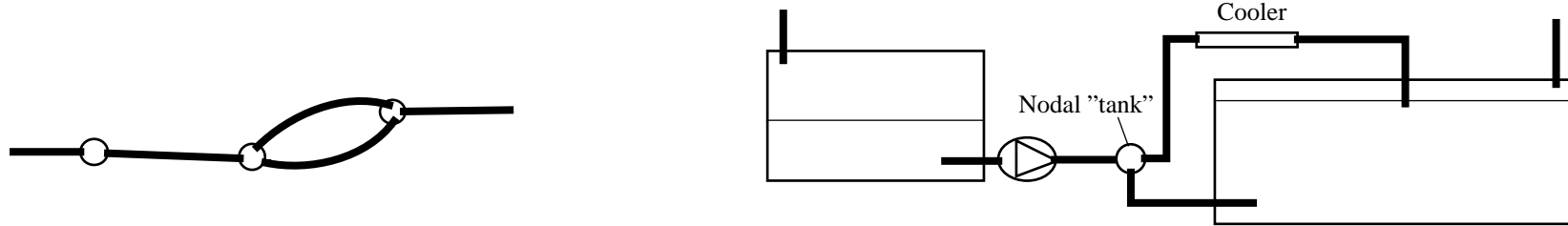
11 'FRTU Pr. to 3F'	33 35 5 73 0 3 0 7.05 10	! Acc. to prel IN sub test
12 'TUBE FRTU to 3F '	35 3 73 8 0 1 0 7.05 2.5	
13 'RADARCOOL W CV'	47 1 78 11 1 1 0 .933 10.8	! 860224
14 'PRIM JP 2B-1B'	47 36 78 33 8 3 -1 .21 1.1	! 851108
15 'SEK JP 2B-1B'	7 36 34 33 1 2 -1 1.47 .8	! 851108
16 'OUT JP 2B-1B'	36 2 33 35 7 1 1 1.68 2.0	! z=1.8+0.2, 1.25 m tube 5/4" + SOV 5 cm2
17 'PRIM JP VT-DRAIN'	47 37 78 27 0 3 -1 .093 1	! 851108
18 'SEK JP VT-DRAIN'	6 37 28 27 1 2 -1 .647 1	! 851108
19 'MIX JP VT-DRAIN'	37 2 27 80 6 1 2 .74 2.1	! z=2.0 + 0.1 for exit tube
20 'ARTU to L WING'	38 49 6 85 0 3 0 10.35 15.0	! 92-02-10
21 'L WINGPUMP '	11 49 47 85 2 2 4 0 0	

into Easy5



## A pure nodal-network concept is used.

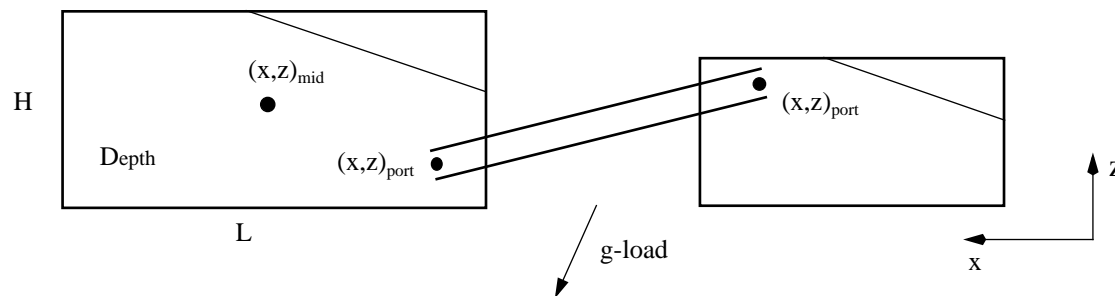
The tanks hold the states of the model and all connections between tanks are resistive.



## 2D-Layout and rectangular tanks

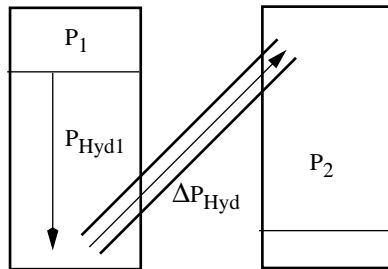
With 2D-layout and rectangular tanks you get a good approximation of the reality without overloading the code and data input. The fuel level in a tank, given the fuel content and the g-load vector, is fairly easy to calculate.

All geometrical data are concentrated to the tanks.



# Both Fuel and Air

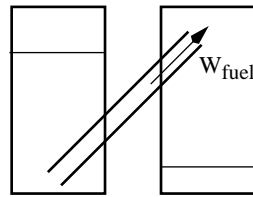
If you want to have a single model of a complete tank system with ventilation/pressurization you are almost forced to admit both fuel or/and air in the same pipe. The pure fuel flow case or pure air flow case, with both ends fuel-covered or not, are trivial, but there are (2 x) 4 different fuel - air flow cases.



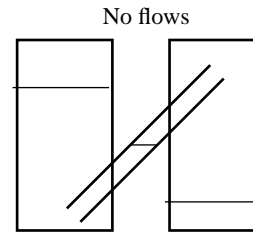
$$\Delta P_{air} = P_1 + P_{Hyd1} - P_2 = \text{pressure between ends}$$

$$\Delta P_{fuel} = \Delta P_{air} + \Delta P_{hyd} = \text{driving pressure in pipe}$$

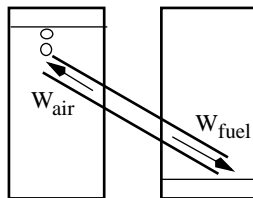
g-load  $\Delta P_{Hyd} =$  hydraulic pressure in the pipe helping the flow from tank 1 to tank 2



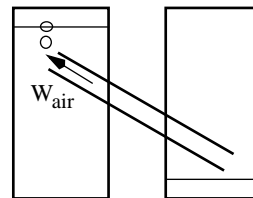
$$\Delta P_{air} > 0, \Delta P_{fuel} > 0$$



$$\Delta P_{air} > 0, \Delta P_{fuel} < 0$$



$$\Delta P_{air} < 0, \Delta P_{fuel} > 0$$



$$\Delta P_{air} < 0, \Delta P_{fuel} < 0$$

## SI-units throughout

SI-units are used solely and completely. Even inputs have to be given in pure SI-units. It has been considered to change to more relevant multiples when inputting data (e.g. cm<sup>2</sup> for pipe cross sections), but this is abandoned. If you stick to pure SI-units, you always know what to input, the risk of making wrong is minimized, *and most important, the underlying code becomes simple, clear and safe.*

## Simple and fast rather than extremely accurate

There is always a balance between the engineers need for an accurate model and the man-hours needed to create and maintain that accurate model and the CPU-time to execute it. The approach of the Fuel-Air Library is to keep it as simple as possible. E.g. the flow function for air that is used, is not the exact expression, but a simplified expression, with only squares and square-roots.

No Easy5 switch states are so far introduced in the library. There are dead-bands for covered/non-covered ports, but these are internal in the code. The inaccuracy, that you get by not using switch states, has to be tolerated.

## Fixed time-step

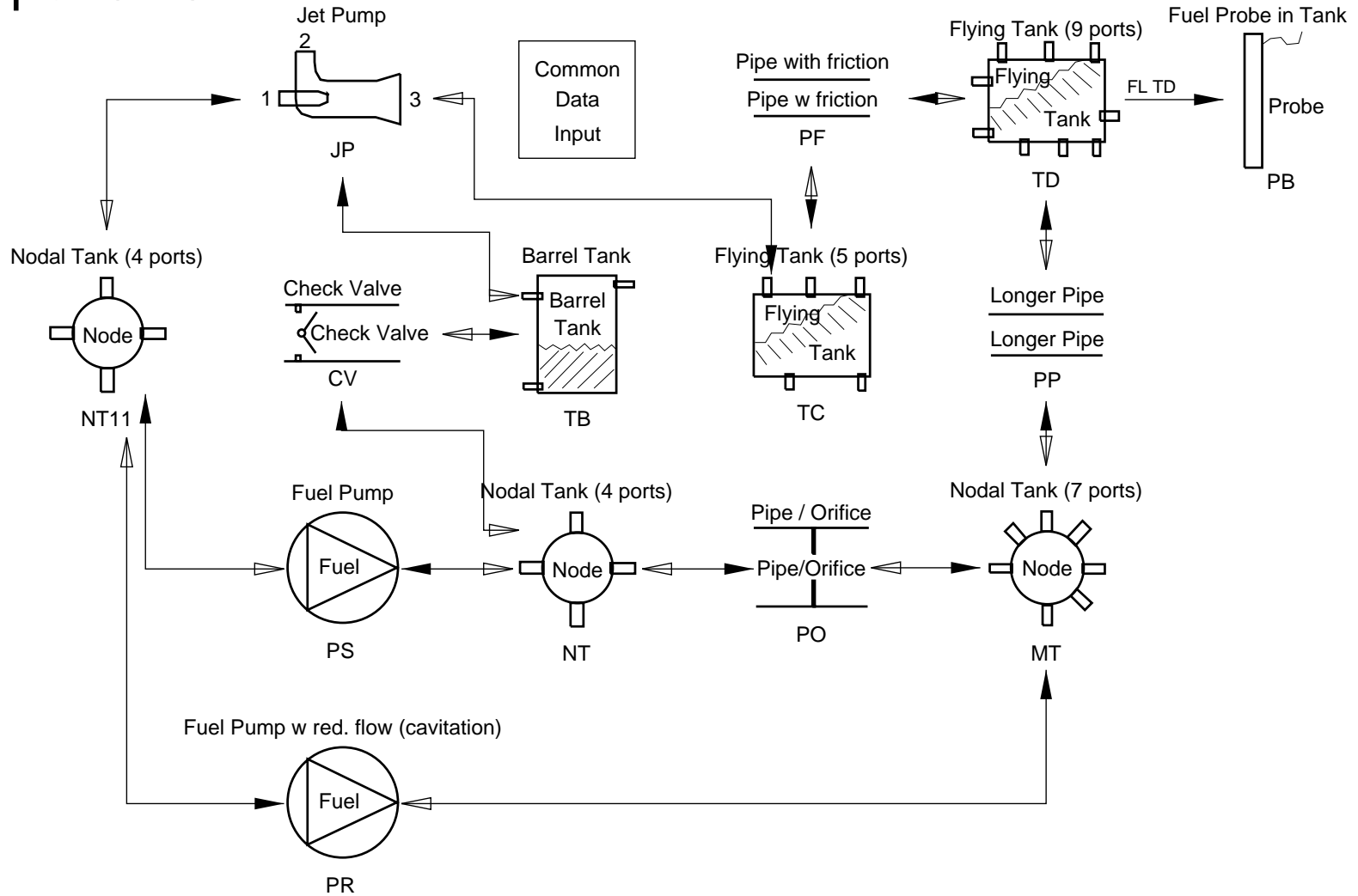
The reluctance to use switch states is one reason why fixed time-step algorithms has to be used in Fuel-Air Library. There is also a pipe module with friction. This module is iterative (the friction is dependent of the flow) and thus implicit. Variable time-step algorithms need truly explicit models without internal memories to work flawlessly (the same states and input variables shall always yield the same state derivatives output).

## The Fuel-Air Tank System Library is self-contained

The Fuel-Air Tank System Library is a self-contained library. It is not an add-on to the ECS Library or the Hydraulic Library. The only modules you might have to add are the Function Tables, regulators, etc in the General Purpose Library to create input to your simulation profiles or make some regulating functions.



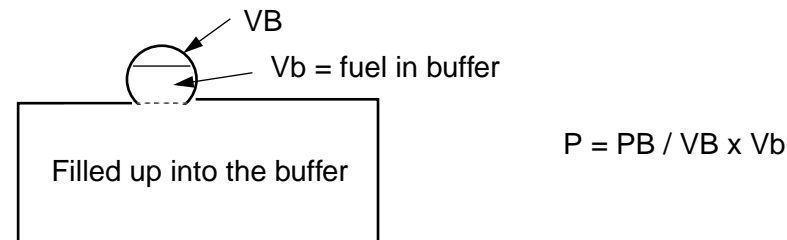
# Components



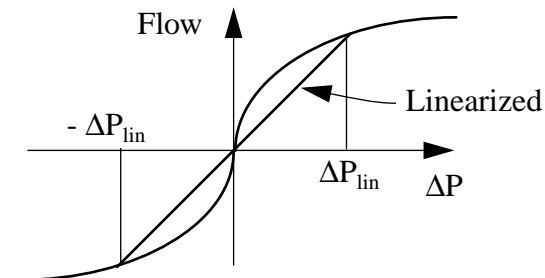
# The Common Data Input, CD

In this component all data that are global to the model are set.

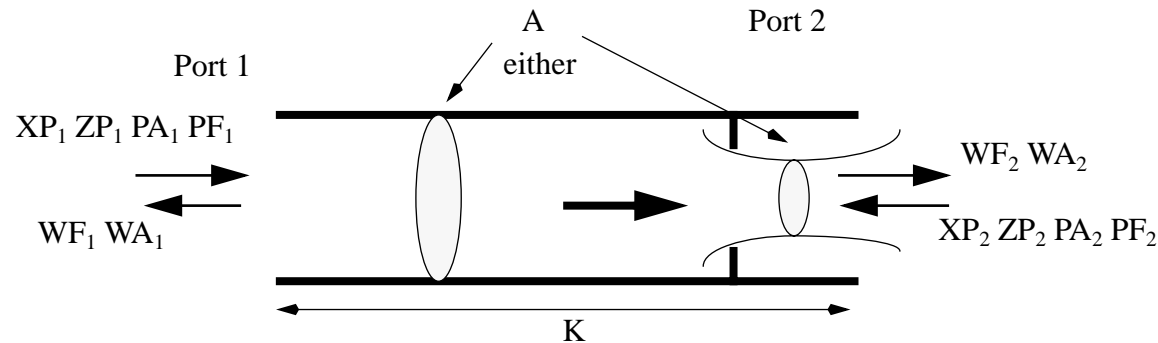
1. The fluid (default fuel JP8) and gas (default air at 20 C) properties.
2. The g-load vector (gx and gz).
3. The volume in the “nodal tanks”.
4. The volume of the filled tank buffer,  $V_B$ , and the associated filled buffer pressure,  $P_B$ .  $P_B / V_B$  makes up a bulk modulus of a filled tank or node.



5. Delta-pressure limits for linearization around zero flow in pipes, etc.



## PO Pipe / Orifice / Venturi / Valve



$$\text{Dynamic head } q = (P1 - P2) / K$$

Input: Geometry and pressures  
Output: Flows, positive outwards

The PO component is the most common connection between two tanks or nodal points. The pressure drop in the connection is represented by a cross section area and the corresponding loss coefficient (number of lost dynamic heads). The flow calculation is simple for fuel but for air the more complicated flow function approach is used. It is a component, that may be used for all types of connections that are more or less discrete in space, e.g. holes, short pipes, orifices, valves and venturis. If there is a chance of high speed air flow, with possible choking, then this component should be used.

## PP Longer Pipe

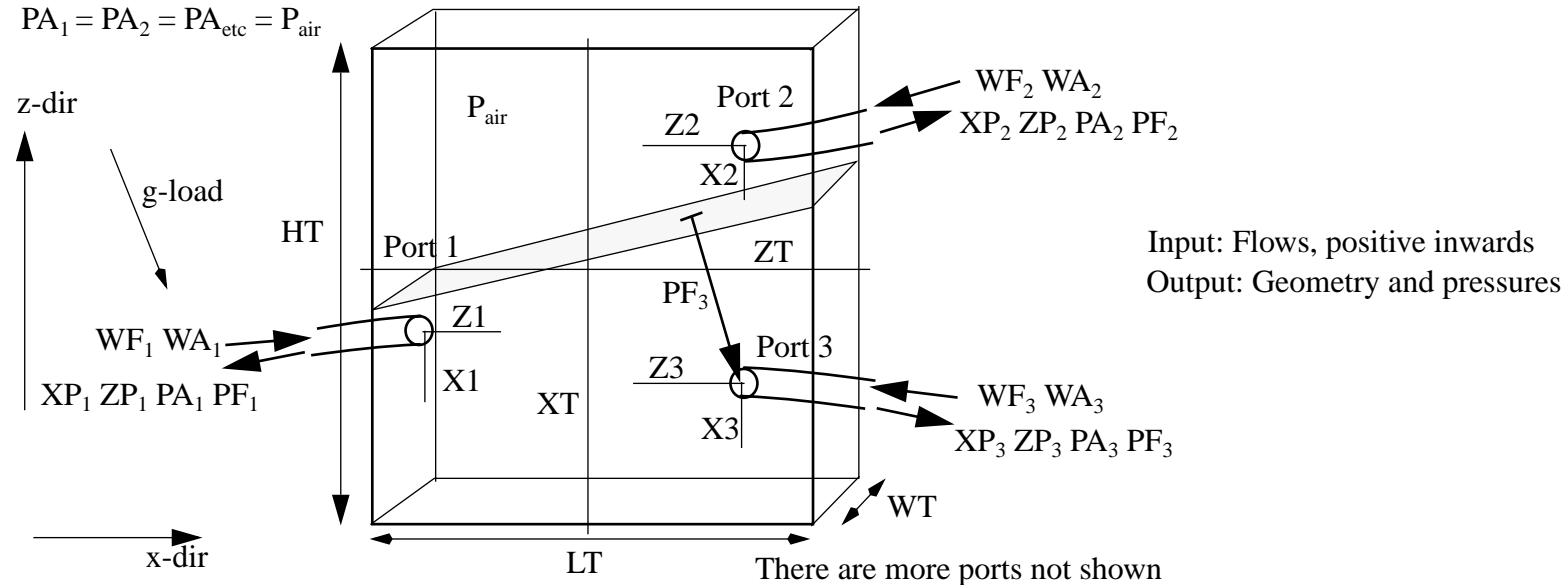
The PP component is mainly intended for longer pipes, in case of air flow, where the flow speed is moderate. The flow calculation code is simple and fast. If high speed, choked conditions may occur, you should combine the PP component with a PO component at the outlet, for best results.

## PF Pipe with discrete loss coefficient and friction

The PF component is a PO component with friction added. This means a more realistic description of the flow-pressure drop behavior in longer pipes than in the PO or PP component with their constant loss coefficients, but it costs a little more CPU-consumption. The Colbrooke-White expression for pipe friction is used.

# TC (5 ports), TD (9 ports) 2-dim Flying Tank

$$PA_1 = PA_2 = PA_{etc} = P_{air}$$



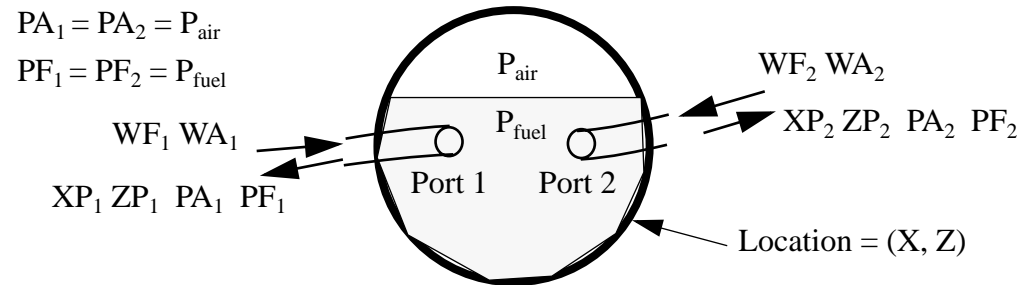
The TC and TD components are 2-dimensional, rectangular tanks, that can be exposed to a g-load in any direction in the 2-D plane. They form the basis for modelling flying tank systems. Most of the geometrical data of a tank system model is placed in these tanks.

Ports that have non-set values ( $ZP = 0.99999$ ) are not calculated.

## The chosen states

The physically most correct states in a tank are the fuel and the air mass content, as these are both conserved states. Due to the inconvenience to specify the initial state of the air as a mass, the state of the air is changed to the more familiar and measurable pressure.

# NT (4 ports), MT (7 ports) Nodal point "Tank"



There are more ports not shown

The NT or MT is the normal way to connect, merge and branch different "pipes". It may be looked upon as a node in a network of "pipe"-resistors. Its main function is to dynamically calculate the pressure in such a nodal point and also keep in memory if the node is filled with fuel or not. With a dead-band, less than 40% fuel in the node means air-filled, more than 60% means fuel-filled.

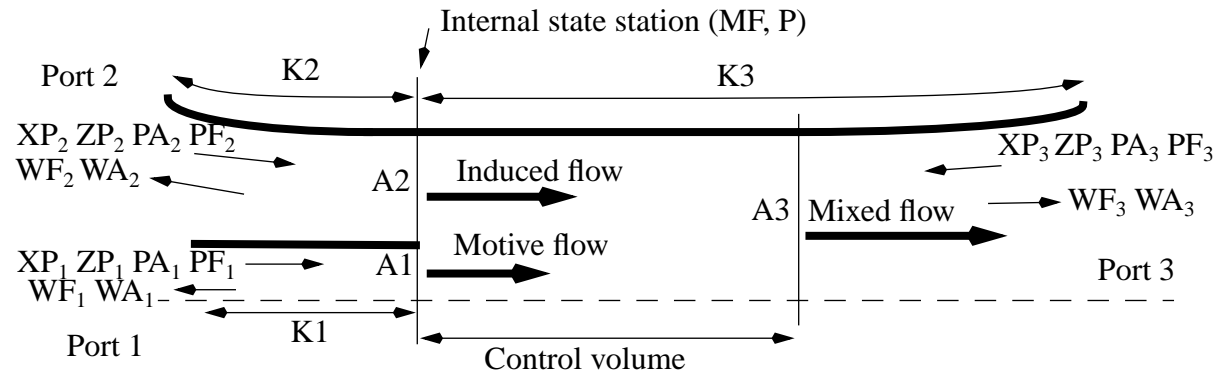
Pressure equation for both "real tanks" and "nodal point tanks"

$$PA = \frac{m_{air}}{V_{air}} \cdot R_{air}T + \frac{PB}{VB} \cdot V_{fuelinbuffer}$$

and its derivative, which has to be used in Easy5, as the pressure is a state.

$$\frac{dP}{dt} = \frac{dm_{air}}{dt} \cdot \frac{R_{air}T}{V_{air}} + \frac{PB}{VB} \cdot \frac{dV_{fuelinbuffer}}{dt} \quad V_{air} \geq V_{buffer} = VB$$

# JP Jet Pump for both fuel and air.



The JP component is a Jet Pump defined by its geometrical data and loss coefficients.

The jet pump is treated as a nodal point representing the cross section at the motive nozzle outlet/mixing channel inlet, which is connected to the three "pipes",

- motive flow nozzle
- induced flow inlet
- mixing channel

The motive nozzle and induced flow inlet are treated as PO components with the addition that the momentum out of them is also calculated. The constant area portion of the mixing channel serves as a control volume for the calculation of the flow through the channel.

The cross section nodal point acts as a normal nodal point "tank" (NT). It acts as the fuel/air buffer described in the NT component. Thereby for instance both fuel-fuel and fuel-air situations can be handled automatically in a simulation. With fuel in the motive flow, it is easy to suck too much air out of the nodal point. Thus the pressure in the nodal point is limited to be above vacuum, via its derivative.

## TB Barrel Tank ( 3 ports)

A non-flying ground tank, described by its bottom area and height. Only the vertical z-positions are used.

## CV Check Valve

The check valve is treated as a orifice (PO). Below an opening differential pressure the valve is closed and above a fully open differential pressure it is fully open. In between these pressures the valve area is linearly interpolated.

## PS Fuel Pump. Single curve.

The PS component is a simple way to describe a fuel Pump by a Single curve (Delta-P, Flow). The Delta-P-values must be monotonously increasing.

If there is air at the inlet, there is neither fuel or air going through the pump, so far.

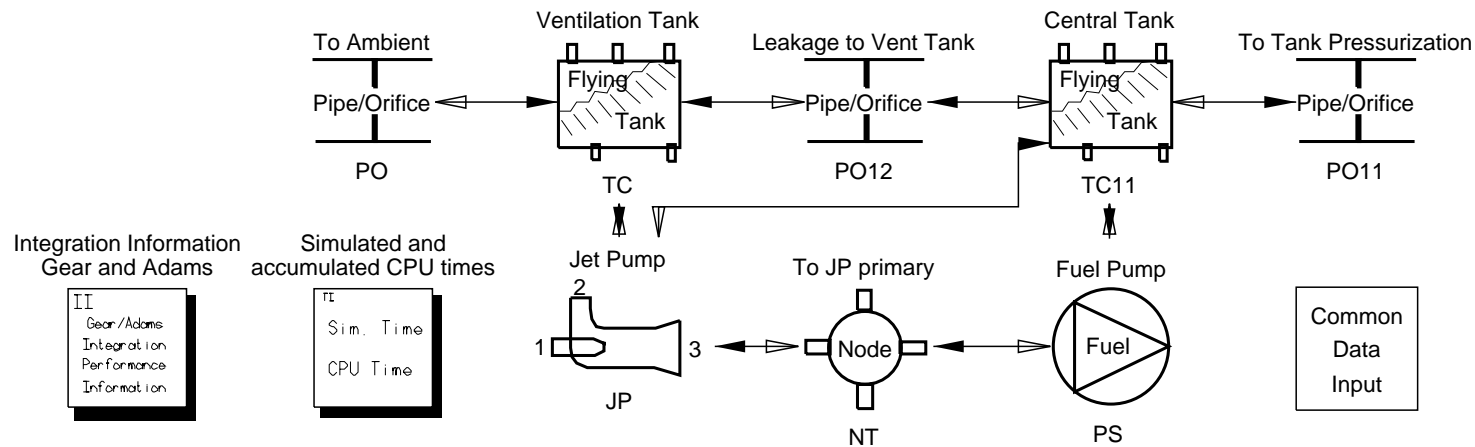
## PB Probe in Tank

The PB component is a simple linear Probe, that can be used to extract the fuel level information from the tank component TC or TD.

As you may have noted there are no heat transfer components. The system is supposed to be isothermal.

## Example 1. Jet Pump in Vent Tank

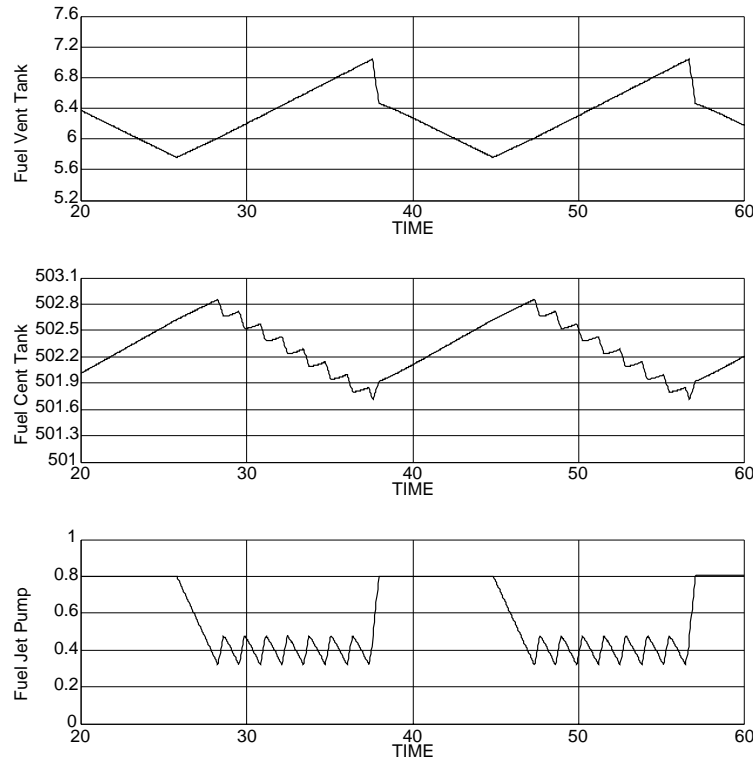
To demonstrate the fuel - air capability the following simplified part of a larger system is used. The jet pump is also probably the most tricky component.



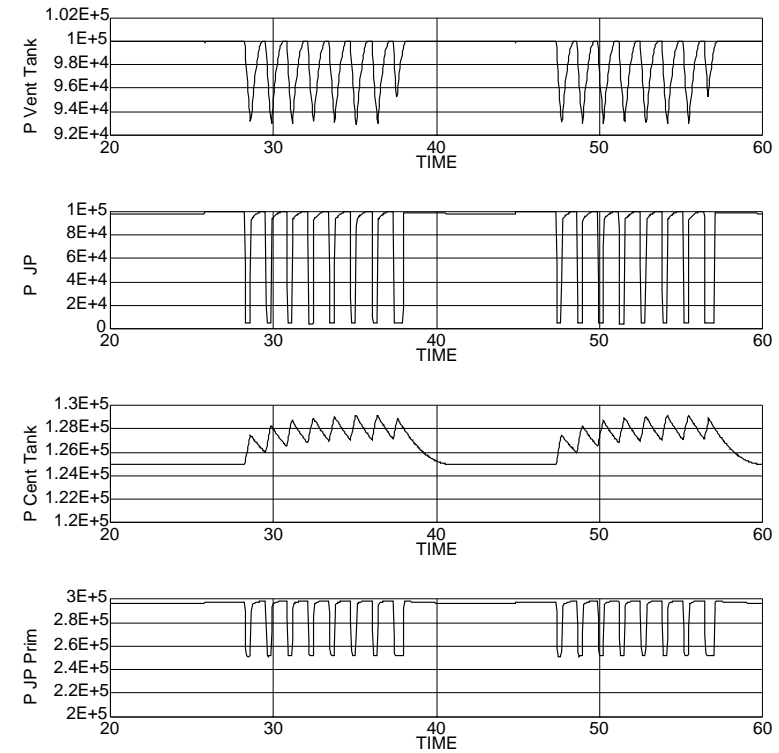
A fuel driven jet pump (JP 0.3 cm<sup>2</sup> prim, 1.3 cm<sup>2</sup> mix) is installed to suck the fuel that has been drained or has leaked into the ventilation tank (TC) back into the central tank (TC11). The jet pump is driven by a fuel pump (PS 4 bar no-flow, 0.6 kg/s free-stream). The ventilation tank is connected to ambient atmosphere (PO with 100 kPa in the other end) and the central tank is connected to some tank pressurization (PO11 with 125 kPa in the other end). The fuel drain/leakage to the vent tank is modeled by a small orifice (PO12). In Common Data (CD (don't forget the CD, nothing happen without it)) the buffer volume and pressure is changed to 0.1 liter and 2 MPa to get a quick pressure reponse to fuel changes. Dead-band (ZDB) is 1 cm. This simple model can be run with all of the integration algorithms, but Heun or Fixed R-K seems to give the best accuracy/CPU-yield. The variable time-step algorithms don't like the internal switches and time-lagging memories in the Fuel Library and get a little upset.

## Result of the simulation (Heun, dt = 0.4 ms)

Jet Pump in Vent Tank

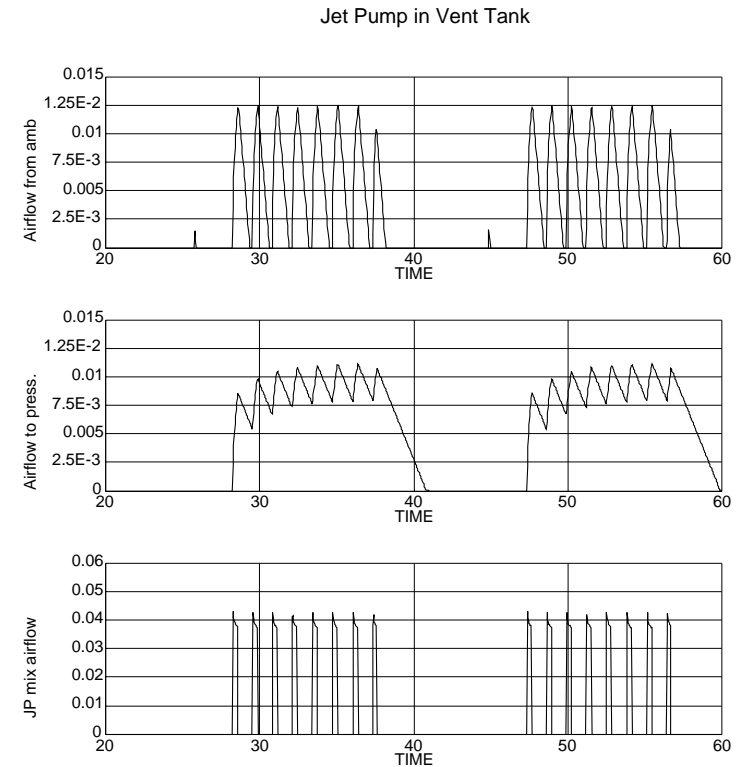
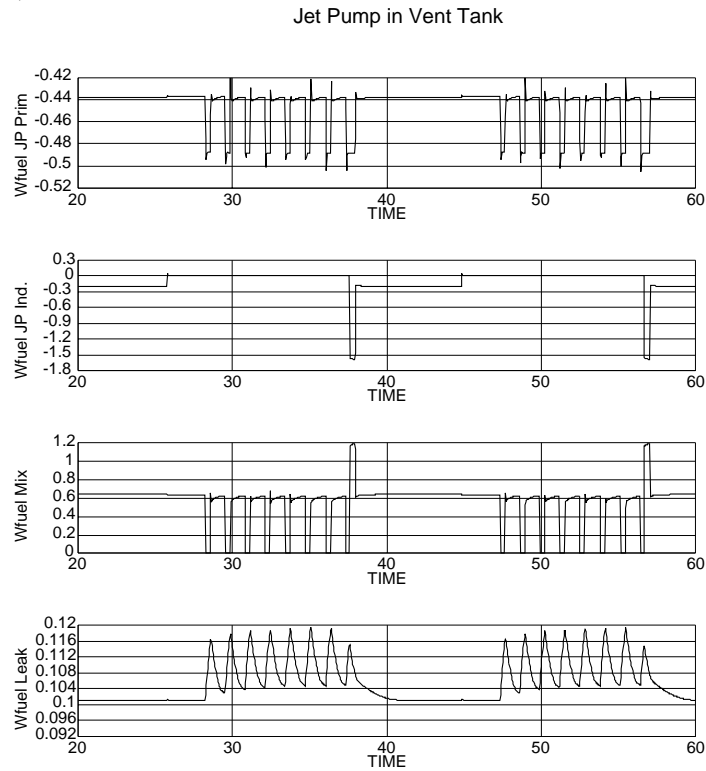


Jet Pump in Vent Tank



The large scale up and downs are due to the covered - not covered dead-band at the induced flow inlet in the vent tank. When the inlet is not-covered, air is sucked through the jet pump. But the primary, motive fuel flow has to be blown out of the jet pump too. This is causing the small scale up and downs. You can see the 40 - 60% dead-band in the "jet pump node" in the left lower plot above.

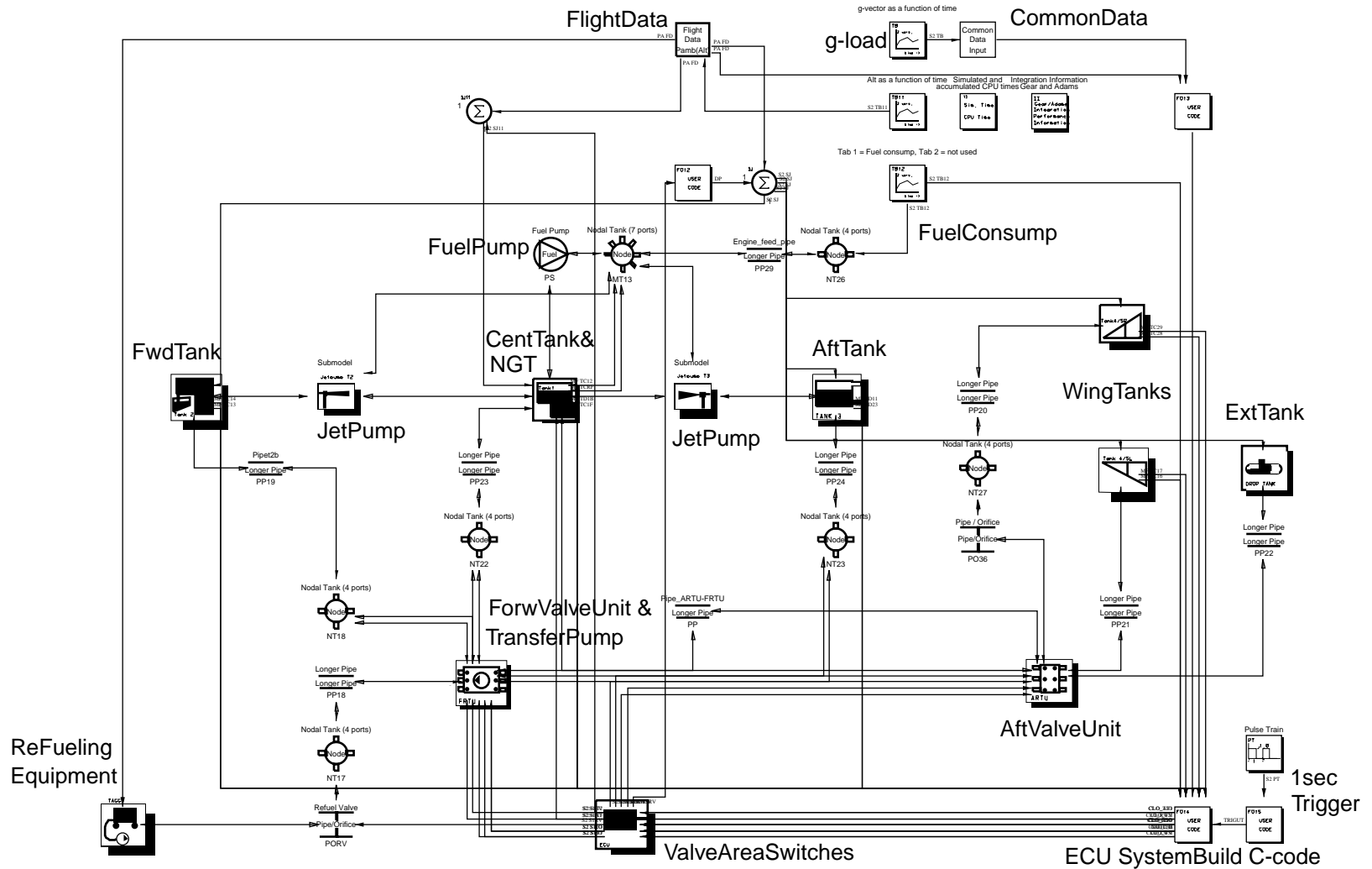
## Results, cont.



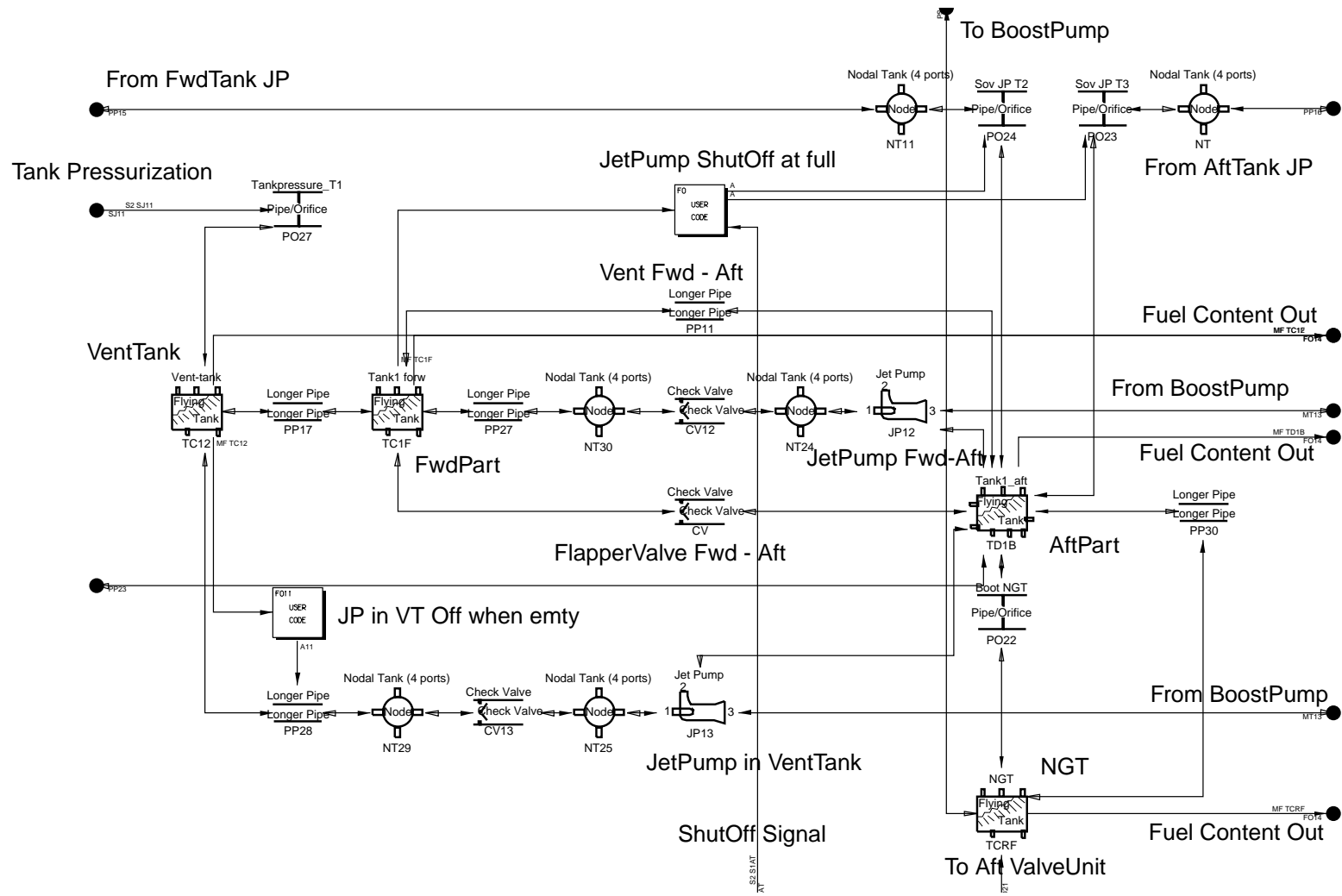
## Conclusion of this example

The Fuel-Air Library is able to handle the “both fuel and air” situation in a stable and reliable manner.

# Example 2. JAS39 Gripen fuel system

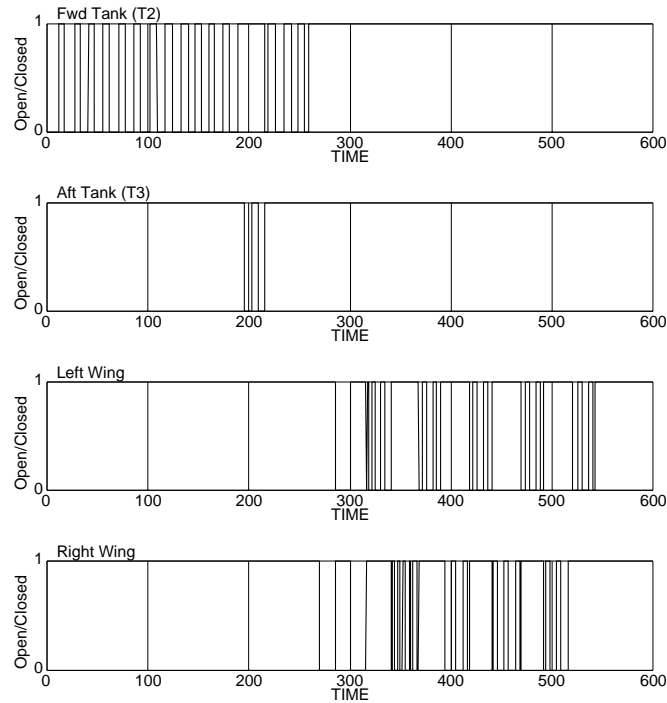


# Submodel of the Central Tank ("Tank 1")



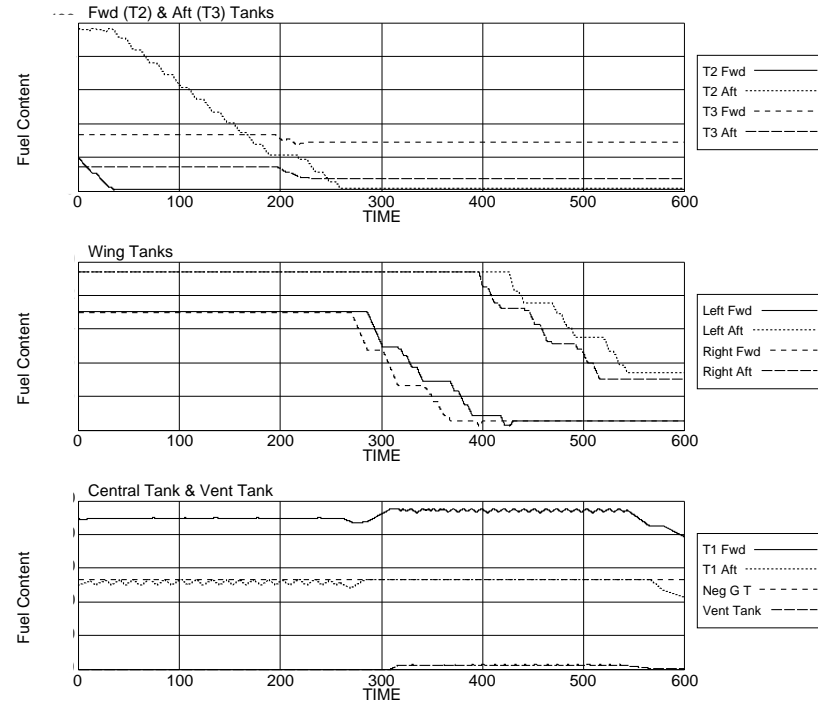
# JAS39 Fuel System. Simulating a straight-forward engine fuel-consumption. (Heun, dt = 1 ms)

Closed Valve Flags (1=Close, 0=Open), First test with ECU



Model: With ECU, Runid: simulation, Case: 1, Display: 5. 10-MAY-2000, 14:21:25

Fuel content in all tanks, first test with ECU



Model: With ECU, Runid: simulation, Case: 1, Display: 2. 10-MAY-2000, 14:21:25

We are in the process of debugging the integration of the SystemBuild model of the ECU. You see that the valve commands from the ECU are nervous in a way they shouldn't. The de-fueling sequence is not correct and the left and right wing tanks are not in phase either. There is something wrong in the C-code or the interfacing Fortran-C-calling routines.

Another astonishing discovery was, that running on SUN instead of HP we got a different behavior. The valve commands weren't nervous, but the de-fueling sequence was even more incorrect.



## Concluding remarks

I really appreciate the user-friendliness of EASY5. As far as I know there is no other tool that encourage you to build your own, fully authorized components and libraries. (Just klick the Macro-button....). It is a good frame-work for your own ideas.

The demonstrated Fuel-Air Library is by now well-proven. At SAAB we use it, not only in the JAS39 Fuel System but in other minor liquid and gas flow applications, all the scale down to simple flow - pressure drop calculations of single pipes.

If there is a true interest to test and use the Fuel-Air Library, this could in some way be arranged, there has already been some discussion about "commercialization".

