# Simulating a Pneumatics Network using the DLR ThermoFluidStream Library

Peter Junglas[1*], Raphael Gebhart [2]

[1]PHWT-Institut, PHWT Vechta/Diepholz, Am Campus 2, 49356 Diepholz, Germany; *peter@peter-junglas.de*

[2]Institute of System Architectures in Aeronautics, German Aerospace Center (DLR), Münchener Str.20, 82234 Weßling, Germany

SNE 35(2), 2025, 117-124, DOI: 10.11128/sne.35.tn.10736
Selected ASIM WS 2025 Postconf. Publication: 2024-05-05
ReReceived Revised: 2025-05-15; Accepted: 2025-05-31
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

**Abstract.** Modeling and simulation of pneumatics networks is still a challenging task, plagued by initialization problems even in sophisticated environments such as the Modelica Fluid Library. The recently proposed DLR ThermoFluid Stream Library uses a promising new approach to cope with such problems. Therefore, it should be a convenient basis for a more specialized pneumatics library.
The essential concepts and components of such a library are presented, with a special focus on the notorious tee branch components. Their dynamic behaviour is very complex, since it couples the effects of dynamic pressure changes and friction losses, and often leads to stability problems. Results of systematic tests as well as more realistic models are discussed. They show that even though some problems with stability remain in special examples, the new library generally allows for the simulation of pneumatics networks using realistic tee branch models, which are more accurate than previous implementations.

## Introduction

Modeling and simulation of pneumatic systems is a non-trivial endeavour, since it combines the turbulent flow of a compressible medium in a usually large pipe network with the highly non-linear behaviour of components such as actuators and valves [1]. A starting point for the mathematical description could be a non-linear partial differential equation describing the fluid flow, coupled with a set of ordinary differential equations modeling the mechanical behaviour of the corresponding components. Of course, this direct approach is usually unfeasible not only due to high computational demands, but because it requires a lot of fine-grained parameters to describe the model and provides much more data than is necessary for typical applications.

Applying a divide-and-conquer strategy, different modeling approaches are used according to the complexity of the system or component under study: A simple tee branch can be analyzed using the full power of a CFD simulation [2], while for more complex situations a coarse grained finite volume approach is employed, using discretizations in one or two dimensions [3]. To cope with very complex systems, one even reduces the description of many components to a zero-dimensional model, using ordinary differential or even purely algebraic equations to describe their behaviour, disregarding any spatial resolution. This approach is adopted in the Modelica Fluid library ("MFL") [4] for most of its components.

For the modeling of large pneumatic networks the MFL has been used in [5]. Unfortunately, most models studied there didn't run in standard Modelica environments, unless the behaviour of some components – especially the tee branches – had been simplified drastically. This is due to the structure of the model: For a pipe network the MFL approach leads to a large system of nonlinear equations, which needs very precise starting values to make the initialization converge.

The recently presented DLR ThermoFluidStream Library ("TFS") [6] has been invented to address these problems. For this purpose, it adds the inertial pressure of the fluid, promoting the mass flows to state variables. Additionally, it uses a clever approximation scheme that decouples the equations of the components, without destroying the correct behaviour in static or quasi-static models [7]. Furthermore, the flows generally have fixed directions, which simplifies the modeling. As a consequence, the initialization usually works, even starting with vanishing mass flow, which should make it a suitable approach for the modeling of pneumatic pipe networks. It is the basis of the specialized PneuBibTFS library presented here, which is freely available from [8].

To show that the TFS library is up to this task, we will closely follow the lines of [5]: After a short introduction to the library and its basic components, a special focus will be on the modeling of tee branches, where several alternatives will be presented and extensively tested. Finally several variants of complete networks containing time-varying consumers will be analyzed and compared to the simplified versions presented in [5].

## 1 Using the DLR ThermoFluidStream Library

The problems with the initialization of models in different application areas are well-known, and a solution based on interpolating between the complete model and a simplified version has been proposed [9] and applied to thermofluid models [10]. Unfortunately, it only works in very special cases, especially not for pneumatic network models [5].

The TFS library addresses the initialization problem by introducing two major changes to the usual description of thermofluid models. They will be described briefly in the following, more details and motivations can be found in [7].

Integrating the Euler equation along a stream line leads to the "Newton's law like" pressure balance

$$\Delta r = \Delta q + \Delta p + \Delta p_{ext},$$

where $\Delta q$ is the dynamic pressure difference due to change of velocity, $\Delta p$ the pressure difference at the end points of the stream line, $\Delta p_{ext}$ the pressure due to additional forces such as gravity or friction and $\Delta r$ the pressure difference due to the inertia of the fluid, given by

$$\Delta r = -L\frac{d\dot{m}}{dt}.$$

The *inertance* $L$ is independent of the thermodynamical state of a fluid and very small for gases. Since one is usually interested only in quasi-static processes, the inertial pressure difference $\Delta r$ is neglected in the MFL library.

In the TFS library, models include this term, where $L$ is generally defined as a globally set small value – since one is not really interested in the transient behaviour –, but can be set for each component individually.

The second ingredient of the TFS library is the introduction of the *steady mass flow pressure* $\hat{p}$, which is defined by splitting the total pressure as

$$p = \hat{p} + r.$$

Its change $\Delta\hat{p}$ along a stream line generally depends on the total pressure and the mass flow. In the steady state $r$ vanishes, therefore the approximation

$$\Delta\hat{p} = f(p,\dot{m}) \approx f(\hat{p},\dot{m})$$

is generally sufficient for quasi-static simulations.

It leads to a decoupling of the component equations along the stream direction. This reduces the large set of nonlinear equations for the complete system to small-sized equations inside the components, thereby making the initialization problem feasible.

An important element in the design of a Modelica library is the connector. Instead of the stream connector used in the MFL library [11] the TFS library defines different connectors for ingoing and outgoing flows. They both use the mass flow $\dot{m}$ as flow variable and the initial pressure $r$ as normal (*potential*) variable.

Additionally they contain the thermodynamic state as input or output variable, respectively. It is usually given by the pressure, the specific enthalpy and a set of mass fractions. Here, the steady mass flow pressure $\hat{p}$ is used instead of the total pressure, thereby implementing the approximation scheme described above.

Based on these ideas, the freely available TFS library contains many of the components that are needed for pneumatics simulations. The specialized pneumatics library PneuBibTFS mainly just contains wrappers around the TFS counterparts, which reduce the number of parameters to the few needed here, and fix the medium to SimpleAir. This further reduces possible non-linearities in the medium model, leading to enhanced stability.

Basic elements of PneuBibTFS generated in this way are:

- `Pipe`: a straight pipe with pressure loss according to Cheng [12].
- `Bend`: a curved pipe with pressure loss from the MFL dissipation library.
- `Tank`: an isothermal pressure tank with explicit inflow and outflow ports.

- `PressureSource`, `PressureSink`: simple source and sink with given pressure.
- `MassFlowSource`, `MassFlowSink`: source and sink that define an input or output mass flow. This is non-trivial in TFS, since the mass flow is a state variable. The components work by combining a pressure source or sink with a control valve from TFS that uses a PT1 dynamic to obtain the given mass flow.
- `MassFlowSourceLin`: source that uses a linear valve component to obtain a given input mass flow.
- `CVActuatorLin`: actuator using a linear valve to obtain a given output mass flow.

The linear mass flow source/sink components are simpler than their controlled counterparts and can lead to more stable models. Furthermore, they are used here to make results comparable to those of [5]. The critical tee branch components have to be created from scratch, they will be studied extensively in the following.

## 2 Modeling Tee Branches

As has been shown in [5], the modeling of the tee branch components is crucial for the stability of pneumatic network models.

This is mainly a consequence of their complex behaviour, combining pressure drops due to internal friction with dynamic pressure changes caused by the changed cross sections of the fluid flow.

In the case of splitting flows the division of the mass flows depends on incoming and outgoing pressures, which leads to a nonlinear coupling across the complete model. Using MFL-based components, even simple models did only run – i. e. survive the initialization phase –, when the tee branches were simplified drastically by completely disregarding all dynamical pressure changes.

Using the TFS approach instead, the mass flows become state variables, which breaks most of such loops. Since the flow directions are generally fixed in TFS, one now needs two different components: a splitter `TeeBranchS` and a junction `TeeBranchJ`, which join or split along the straight direction (cf. Fig. 1). For simplicity, we will only consider tee branches with a 90° angle and identical cross sections $A$ at all three ports. This is a common situation in many pneumatics networks.

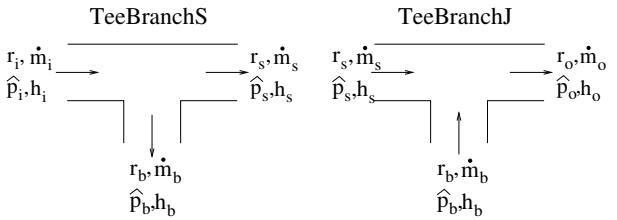


**Figure 1:** Tee Branch components.

The basic equations to describe the behaviour of a tee branch have been formulated in [13] and are widely used in applications. They rely on two functions $\zeta_{cs}$ and $\zeta_{cb}$ that describe the pressure losses across the straight and branch directions.

Since they contain a part of the dynamical pressure effects, they can be negative in certain cases, giving an actual pressure rise. Unfortunately, their concrete form varies largely in the literature [5]; we will use simple polynomials that fit published data.

The basic component `TeeBranchS` is simplified further by assuming constant temperature and density, using the density of the incoming flow everywhere. This avoids additional nonlinear loops inside the component and leads to the following equations:

$$
\begin{aligned}
0 &= \dot{m}_i + \dot{m}_s + \dot{m}_b \\
\rho &= \rho(\hat{p}_i, h_i) \\
\Delta p_s &= -\frac{1}{2\rho A^2}\zeta_{cs}\left(\frac{\dot{m}_b}{\dot{m}_i}\right)\dot{m}_i^2 \\
\Delta p_b &= -\frac{1}{2\rho A^2}\zeta_{cb}\left(\frac{\dot{m}_b}{\dot{m}_i}\right)\dot{m}_i^2 \\
\Delta p_{dyn,s} &= \frac{1}{2\rho A^2}(\dot{m}_i^2 - \dot{m}_s^2) \\
\Delta p_{dyn,b} &= \frac{1}{2\rho A^2}(\dot{m}_i^2 - \dot{m}_b^2) \\
\hat{p}_s &= \hat{p}_i + \Delta p_{dyn,s} + \Delta p_s \\
\hat{p}_b &= \hat{p}_i + \Delta p_{dyn,b} + \Delta p_b \\
h_s &= h_i \\
h_b &= h_i
\end{aligned}
$$

It is important to note that the equations to calculate the dynamic pressure differences $\Delta p_{dyn}$ along the straight or branch direction are using the total input mass flow, while only a part of this mass flow reaches the corresponding output. This formulation is used, because the mass flow split is unknown beforehand.

The error introduced here is made up for by including the difference to the correct dynamical pressure in the $\zeta$-functions – which makes clear, why they can have negative values.

These are the same equations that have been used in [5] for the split case, if one identifies $\hat{p}$ and $p$. In the TFS context, one needs additional equations describing the behaviour of the $r$ variables. They can be derived from results in [7] or directly read off the component `SplitterN` provided in the TFS library:

$$L\dot{m}_i = r_i - r_{mix}$$
$$L\dot{m}_s = r_s - r_{mix}$$
$$L\dot{m}_b = r_b - r_{mix}$$

where $r_{mix}$ is an internal variable that is defined implicitely by the component equations.

A different approach to the modeling of a tee branch splitter uses the `DynamicSplitter` that is provided by the TFS library (cf. Fig. 2). It contains `DynamicPressureInflow/Outflow` components that compute dynamic pressure differences from the cross section area and the inlet/outlet velocity, which are given as parameter values. This leads exactly to the dynamic pressure differences from above.

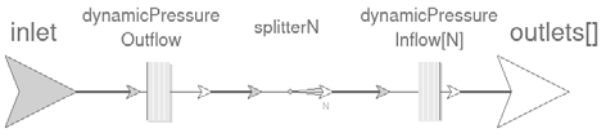


**Figure 2:** DynamicSplitter component.

The complete `TeeBranchS1` component adds a `SplitterPressureLoss` that computes the pressure loss caused by friction and the correction of the dynamical pressure, again using the $\zeta$-functions (cf. Fig. 3). Basically, it reproduces the equations from above, with two small differences: The `DynamicPressureInflow/Outflow` include the temperature changes that are due to the – usually adiabatic, not isothermal – pressure change, and the frictional pressure computation uses the density at the outputs of the `DynamicSplitter`, not at the inlet. This corrects a part of the approximations that are made in the simpler `TeeBranchS`.

Furthermore, its approach is more modular and easier to understand.

On the other hand, its Modelica implementation consists of 147 equations altogether, compared to only 28 equations for the simpler component. Luckily, the Modelica preprocessing usually gets rid of this overhead.

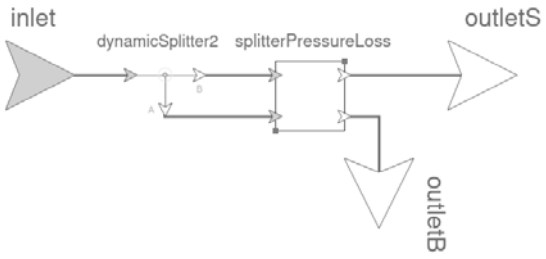


**Figure 3:** Alternative component TeeBranchS1.

To get even better results, one can use the component `TeeBranchS2`, which calculates the dynamic pressure differences by correctly using the densities of the input and output streams instead of using the input density everywhere.

The price is the addition of two nonlinear equations inside the component. A similar approach can be used with the `SplitterPressureLoss`, which would lead to two more nonlinear equations.

The construction of corresponding joining elements `TeeBranchJ`, `TeeBranchJ1` and `TeeBranchJ2` completely follows the lines above. The basic difference lies in the handling of the $r$ variables when mixing input streams. The proper equations again can be found in [7] or in the component `JunctionN` from the TFS library.

## 3 Testing Tee Branches

The various tee branch components have been tested thoroughly using similar models as in [5], a typical example for the joining case is shown in Fig. 4. Here, the mass flows at the inflows and the pressure at the outflow are given explicitely.

The results for this example using the three different `TeeBranchJ` components and the `TeeBranch1` component from [5] are shown in Fig. 5. The plots for the basic MFL and TFS based components are almost identical, which is expected, since they use basically the same equations. The deviation at the beginning is due to the different initialization methods: MFL starts with a given value of $\dot{m}$, while TFS starts here with $\dot{m} = 0$ and winds it up using the inertance equation.

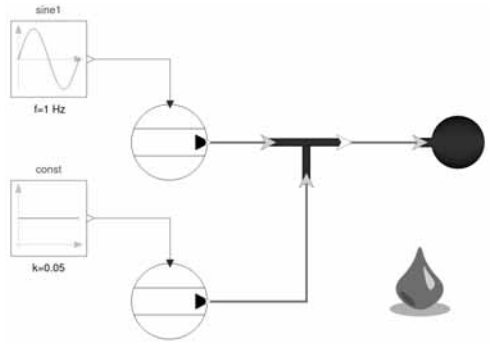


**Figure 4:** Model for testing a TeeBranchJ component.

The slight phase difference is not caused by the inertance, but by the PT1 dynamic of the mass flow controller used in TFS. Much larger are the pressure differences between the three TFS components, especially for the straight branch. At this point, this seems to indicate that a better modeling of the density changes could be useful.

More important than the exact results – which depend on the choice of the $\zeta$-functions anyhow – is the question of stability: Do the models run immediately, only with special initial values or doesn't the initialization converge? To check this, test models similar to Fig. 4 have been analyzed, using different kinds of boundary conditions:

- a: pressure given at inflow, mass flow at outflow
- b: mass flow given at inflow, pressure at outflow
- c: pressure given at inflow and outflow

The results are unexpected: Only in the simple case, where two mass flows are given (case a for the splitter, case b for the joiner), all four components work. For the other cases, the MFL models work always, the basic TFS components in most cases, the more advanced TFS components never. The problem here is not the initialization, all models start and run for a (very) short time. Then the pressure values diverge rapidly. Obviously, the differential equations used here are highly unstable. In some cases, the problem can be fixed by using non-zero initial conditions for the mass flows, but often even very good starting points – coming from the working MFL model! – don't lead to a stable solution.

In additional tests a small pipe has been added either at the incoming or the outgoing straight branch. For the stable cases this leads to a problem with an MFL model: The splitter doesn't run with a pipe at output [5].

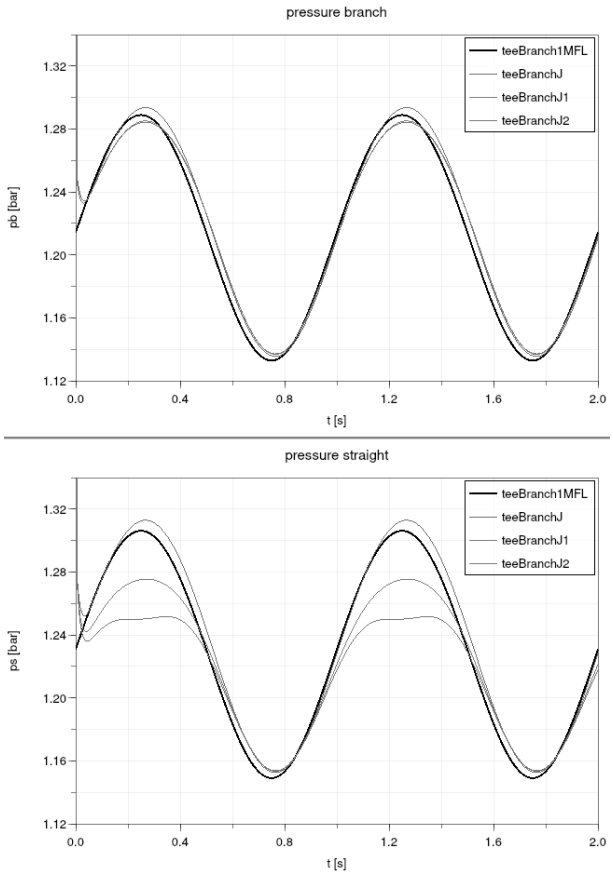


**Figure 5:** Comparison of the pressure drops in join mode.

The corresponding TFS models are not affected, they all work with the additional pipe on either side. In the unstable cases, the situation is more complicated, but generally, the situation gets worse in the MFL case, while in the TFS case several models that didn't run before, get stabilized by the additional pipe.

In conclusion, the tests show that the TFS approach does not solve all problems, due to the inherent instability of the basic equations. This apparently gets worse when the change in density is included. But at least it works in many cases, and the addition of pipes sometimes stabilizes a model.

## 4 Modeling Pneumatic Networks

To check the performance of the PneuBibTFS library in more realistic situations, the basic example model from [5] has been studied, which contains one `TeeBranchJ` and four `TeeBranchS` components, together with sev-

eral pipes and curves, a pressure source, a few consumers and an auxiliary tank. Since the tank uses dedicated inflow and outflow ports, it is connected to the network via a loop consisting of a splitter and a joiner (cf. Fig. 6).

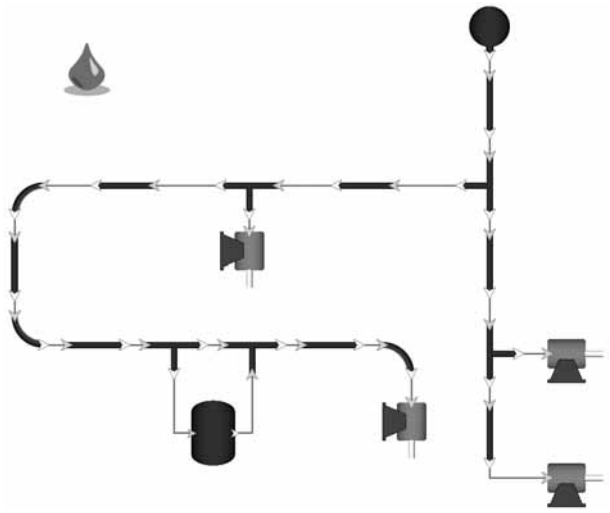

**Figure 6:** Model of the simple pneumatics network 1.

Starting with an empty tank (i. e. $p = p_0$), running the model works without problems and leads to results that are similar to those from [5] (cf. Fig. 7). If one replaces all tee branches by their more sophisticated versions, the models still run and reproduce the results of Fig. 7 within the plot accuracies. But in the MFS case, the model didn't run at all, unless one replaced the basic tee branch model by a very simplistic model based on substitutional pipe lengths. This shows that the somewhat unconvincing conclusions from the teebranch test results are much clearer in larger models: While the initialization problems in the MFL case get much more serious for larger models, in the TFS approach, the instabilities are largely mitigated.

A slightly extended example has been studied in [5] that contains an auxiliary tank between the two consumers on the right side. Building this model with PneuBibTFS, the simulation stops immediately with the error message

```
Positive mass flow rate at Volume outlet.
```

Apparently, in the initial phase of the simulation the medium flows into the tank through the outflow port, which is caught by an assertion. The TFS library includes a variable – hidden inside the `DropOfCommons` component – to reduce the assertion level from *error* to *warning*. Doing this, the model runs fine and produces the expected results. The backflow issue is a minor initialization problem and can be safely ignored here.

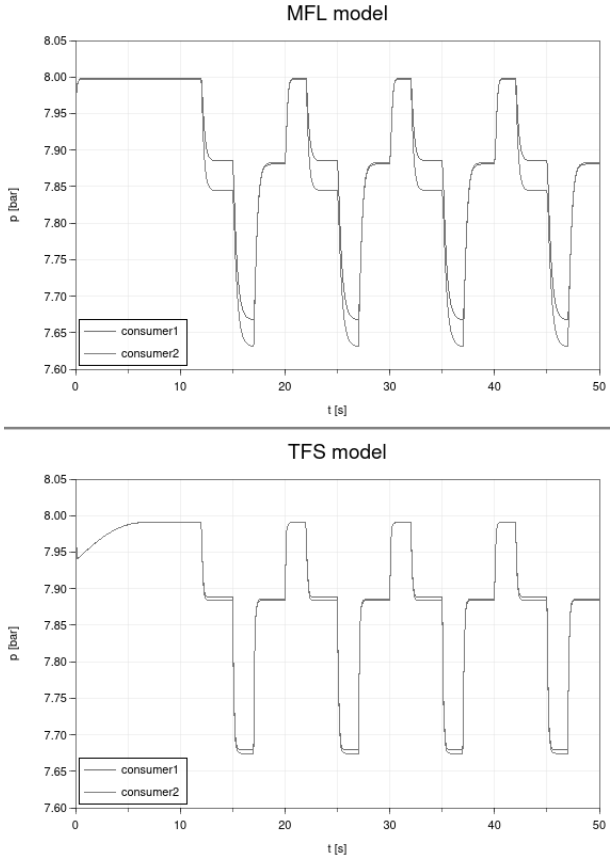


**Figure 7:** Simulation results of example network 1.

Increasing the simulation time one runs into another problem: The simulation stops at $t = 80$ s, one has hit the instability region. Taking a closer look at the model, one finds, that at this moment the consumer near the first tank is switched on for the first time. To increase the stability, a small pipe has been added between the splitter and the joiner that form the loop containing the tank (cf. Fig. 8). This works fine and the model now runs for long simulation times. Fig. 9 displays the pressure curves at the two consumers at the right side for the MFL and TFS variants. It shows clearly that the simplifications, which had been necessary to make the MFL model run, lead to significant deviations in the results.

Finally, one of the real-world models from [5], coming from an industrial partner, has been ported to PneuBibTFS. It contains almost 60 components, among them three pumps, one tank, 12 consumers and 17 tee branches.

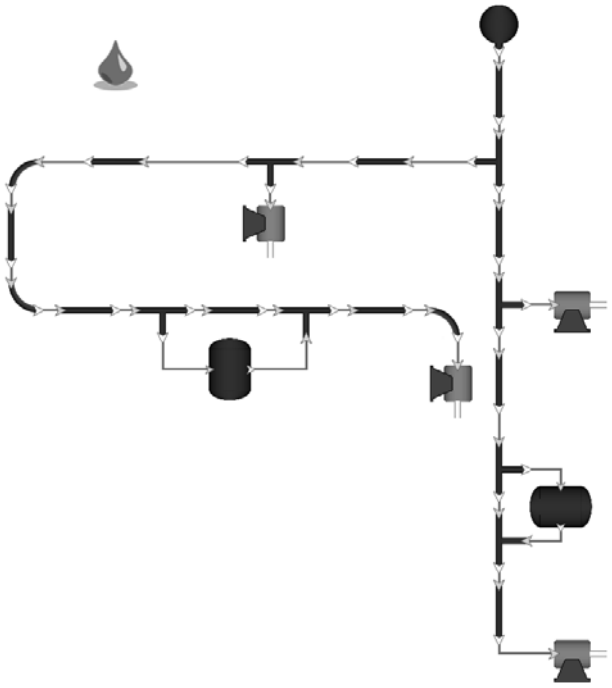

**Figure 8:** Model of the enlarged pneumatics network 2.

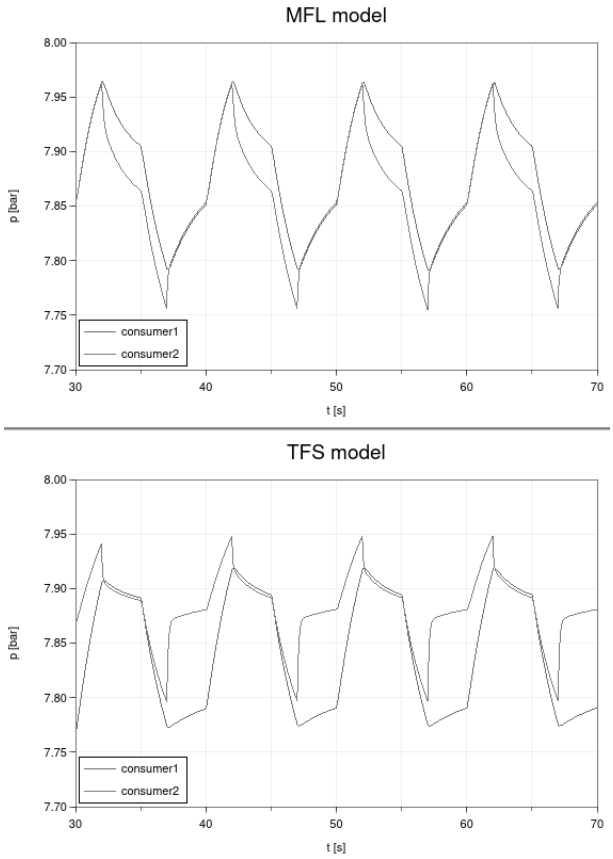


**Figure 9:** Simulation results of example network 2.

The MFL version only contains the simplistic tee branch component and has about 4500 equations. For the port to TFS the flow directions have to be specified everywhere. Furthermore, the tank again has to be included via a small loop, and its initial pressure has been set to the (identical) pump pressures. The final model has only 1750 equations, since the TFL library has a much simpler structure than the MFL library.

The simulation of the TFS-based model stopped after 1 s with the usual blowup of all pressures. Additionally, several flows had the wrong direction, which is due to the identical pressures of all pumps. To ensure the correct flow directions, the pressure of one pump has been increased marginally. With this change, the model runs immediately and qualitatively reproduces the results of the MFL version.

## 5 Conclusions

Though the PneuBibTFS library still has problems with stability in special examples, it allows for the simulation of pneumatics networks using a realistic tee branch model. In many cases, the TFS-based methods work much better than an MFL-based approach, especially for larger models. If problems appear, they can often be cured by insertion of auxiliary pipes.

Comparison with the MFL-based results show significant differences, which are due to the very crude tee branch models used there. Apparently, the omission of the proper dynamic pressure changes introduced considerable errors.

Using the more detailed tee branch models that take into account local variations in density and temperature changed the results only marginally. Since these components reduce the general stability of the model, one should stick to the basic `TeeBranchS` and `TeeBranchJ` components.

In [5] the use of OpenModelica [14] as a modeling and simulation tool introduced additional problems. This has changed completely, all PneuBibTFS models that run in Dymola [15], work in OpenModelica as well, and vice versa. This is due to two effects: On the one hand, the OpenModelica simulator has been enhanced considerably in the last years [16], on the other hand, the new models are much simpler conceptionally, since they don't lead to huge monolithic nonlinear equations . An interesting point for improvement is the model-

An interesting point for improvement is the modeling of the tanks: In reality, a tank is often connected to the pipe network using a simple port. It works as a buffer, the flow direction changes according to the pressure differences between the tank and the network. To model such a tank, one also needs a tee branch model that works with different flow directions. For such purposes, the TFS library has been enhanced to allow for bidirectional flows [17]. This leads to more complex components that are more tightly coupled. Whether such models deliver better results and – more importantly – are more stable, is an interesting question.

Clearly, the most important open point is the question of stability. Probably, the difficulties in solving the MFL-based nonlinear equations and the instability of the TFS-based differential equations are related. It would be interesting to study the instability of the basic tee branch equations in more detail and to find out, whether there exist more stable formulations, as well as how the stabilization in larger models actually works.

A basic conclusion from [5] with respect to the MFL library was: *The fundamental problem of initialization seems to be still far from being solved.*

In the light of the results presented here, it seems to be justified to claim that the TFS library has solved the initialization problem, at least for the class of models that have been studied here.

**Publication Remark**. This contribution is the improved version of the conference version published in Tagungsband Kurzbeiträge ASIM WS GMMS/STS 2025, ARGESIM Report AR 48,
ISBN ebook: 978-3-903347-66-3,
Volume DOI 10.11128/arep.48, p. 65-72.

## References

[1] Beater P. *Pneumatic drives*. Berlin Heidelberg New York: Springer. 2007.

[2] Aigner D. Gleichung zur Berechnung der hydraulischen Verluste der Rohrvereinigung - kalibriert mit Ergebnissen numerischer und physikalischer Modelle. *3R-international*. 2008;47(1).

[3] Fischer F, Schmitz K. Distributed Parameter Pneumatics. In: *Proc. 15th Int. Modelica Conference*. Aachen, Germany. 2023; pp. 85–44.

[4] Franke R, Casella F, Sielemann M, Proelss K, Otter M. Standardization of thermo-fluid modeling in Modelica.Fluid. In: *Proc. 7th Int. Modelica Conference*. Como, Italy. 2009; pp. 122–131.

[5] Drente P, Junglas P. Simulating a simple pneumatics network using the Modelica Fluid library. *SNE Simulation Notes Europe*. 2015;25(2):85–92.

[6] Zimmer D, Weber N, Meißner M. The DLR ThermoFluid Stream Library. *Electronics*. 2022;11(22).

[7] Zimmer D. Robust object-oriented formulation of directed thermofluid stream networks. *Mathematical and Computer Modelling of Dynamical Systems*. 2020; 26(3):204–233.

[8] Junglas P. *Pneumatics Network library in Modelica*. URL `https://www.peter-junglas.de/fh/simulation/pneubib.html`

[9] Sielemann M, Casella F, Otter M, Clauß C, Eborn J, Mattsson SE, Olsson H. Robust Initialization of Differential-Algebraic Equations Using Homotopy. In: *Proc. 8th Int. Modelica Conference*. Dresden, Germany. 2011; pp. 75–85.

[10] Casella F, Michael Sielemann LS. Steady-state initialization of object-oriented thermo-fluid models by homotopy methods. In: *Proc. 8th Int. Modelica Conference*. Dresden, Germany. 2011; pp. 1–11.

[11] Franke R, Casella F, Otter M, Sielemann M, Elmqvist H, Mattson SE, Olsson H. Stream connectors – an extension of Modelica for device-oriented modeling of convective transport phenomena. In: *Proc. 7th Int. Modelica Conference*. 2009; pp. 108–121.

[12] Cheng NS. Formulas for friction factor in transitional regimes. *Journal of Hydraulic Engineering*. 2008; 134(9):1357–1362.

[13] Miller DS. *Internal Flow Systems*. Cranfield, UK: The British Hydromechanics Research Association, 2nd ed. 1990.

[14] Fritzson P, Pop A, Abdelhak K, Ashgar A, Bachmann B, Braun W, Bouskela D, Braun R, Buffoni L, Casella F, Castro R, Franke R, Fritzson D, Gebremedhin M, Heuermann A, Lie B, Mengist A, Mikelsons L, Moudgalya K, Ochel L, Palanisamy A, Ruge V, Schamai W, Sjölund M, Thiele B, Tinnerholm J, Östlund P. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Modeling, Identification and Control*. 2020;41(4):241–285.

[15] Brück D, Elmqvist H, Mattsson SE, Olsson H. Dymola for multi-engineering modeling and simulation. In: *Proc. 2nd Int. Modelica Conference*. Oberpfaffenhofen, Germany. 2002; pp. 55-1–55-8.

[16] Pop A, Östlund P, Casella F, Sjölund M, Franke R. A new OpenModelica compiler high performance frontend. In: *Proc. 13th Int. Modelica Conference*. Regensburg, Germany. 2019; pp. 689–698.

[17] Zimmer D, Weber N, Meißner M. Robust Simulation of Stream-Dominated Thermo-Fluid Systems: From Directed to Non-Directed Flows. *SNE Simulation Notes Europe*. 2021;31(4):177–184.