Overdetermined Steady-State Initialization Problems in Object-Oriented Fluid System Models

Francesco Casella, Filippo Donida
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci, 32 - 20133 Milano ITALY
e-mail: casella@elet.polimi.it

Bernhard Bachmann
Fachbereich Mathematik und Technik
Fachhochschuele Bielefeld
Am Stadtholz 24 – 33609 Bielefeld GERMANY

Peter Aronsson
MathCore Engineering AB,
Teknikringen 1B, SE-583 30 Linköping, SWEDEN

Abstract

The formulation of steady-state initialization problems for fluid systems is a non-trivial task. If steady-state equations are specified at the component level, the corresponding system of initial equations at the system level might be overdetermined, if index reduction eliminates some states. On the other hand, steady-state equations are not sufficient to uniquely identify one equilibrium state in the case of closed systems, so additional equations are required. The paper shows how these problems might be solved in an elegant way by formulating overdetermined initialization problems, which have more equations than unknowns and a unique solution, then solving them using a least-squares minimization algorithm. The concept is tested on a representative test case using the OpenModelica compiler.

1 Introduction

The Modelica language is finding more and more applications in the field of thermo-fluid system modeling, due to the many advantages of the declarative, object-oriented approach. In this context, it is very often the case that steady-state initialization is required.

Specifying a well-posed steady state initialization problem in an object-oriented language is a non-trivial task for some fundamental reasons. From an end-user point of view, the ideal situation is to select a "steady-state initialization" option on the system components, without worrying too much about the actual internal implementation. This means that each component model should contain an initial equation section, with conditionally activated initial equations that express the steady state condition for that model. In this way, initial equations are specified locally within each model. Unfortunately, a well-posed initialization problem can only be formalized at the aggregate system level, i. e., on the system of DAEs describing the complete system. On one hand, index reduction can lead to a reduced number of states, if ideal pipes with zero pressure loss are used or ideal controllers are employed, so that some of the locally specified initial conditions are redundant. On the other hand, some model structures (e. g., closed systems) may be such that the locally specified steady-state conditions are not sufficient to completely determine the initial state.

The actual type and number of independent initial equations required to uniquely determine a consistent steady-state initialization thus depends in a non-trivial way on the connection topology of the system. It is therefore impossible for the library designer to write local steady-state initial equations

which are always good, because that depends on how the specific model will be connected to other ones. Furthermore, it is exceedingly hard for the end user to determine the exact structure of the required initial equations, because this would require a deep knowledge of the inner mathematical details of the single models, and of the mathematical properties arising from the interconnection of the models. The former requirement is against the principle of encapsulation: one should not necessarily be aware of the implementation details of an object in order to use it; the latter can be even more difficult for large systems.

The aim of this paper is to demonstrate how an elegant and user-friendly solution to this problem can be obtained by formulating overdetermined initialization problems, with particular reference to fluid systems. No extension to the Modelica language is needed. Three representative examples will be presented, then solved using the Open Modelica compiler and the methods presented in [1].

2 A Simple Circuit Model

The approach proposed in this paper will be demonstrated on a small case study: the simplified model of a heating circuit. The system includes an accumulator to pressurize the circuit, a pump, a heater (pipe with prescribed heat flow), a valve and a radiator (pipe with convective heat transfer to a fixed temperature sink), connected in a closed loop configuration (Fig. 1).

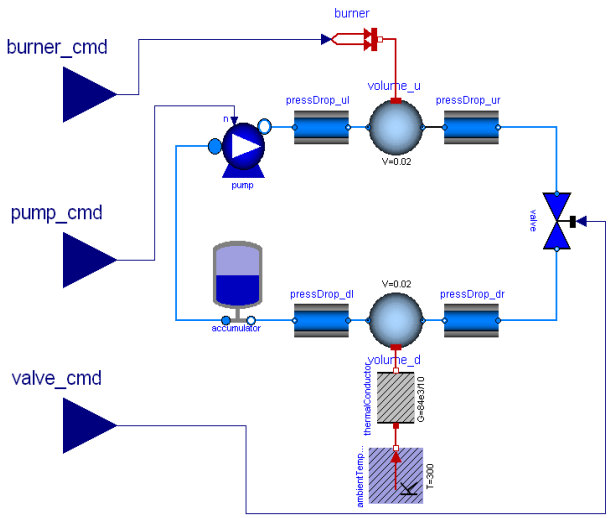

Figure 1. Flow diagram of the test case

The original model was built using components from the Modelica_Fluid library [2]. In order to overcome the current limitations of the OpenModelica compiler, the SimpleFluid library has been developed. The aim of this small library is to capture the essential mathematical structure of fluid system models, while avoiding advanced language features, such as the semiLinear operator and the replaceable packages of the Modelica.Media library, currently not supported by the compiler. These simpler models are more than adequate to demonstrate the proposed approach; the library will be updated with more complex models and test cases as the OpenModelica compiler is improved.

2.1 Connectors

The fluid connectors of the library are similar to the connectors of the ThermoPower library [3]-[4]:

```
connector FlangeA "Type-A connector"
 Types.Pressure p "Pressure";
 flow SI.MassFlowRate w "Mass flowrate";
 output Types.SpecificEnthalpy hAB
    "Specific enthalpy of fluid flowing A->B";
 input Types.SpecificEnthalpy hBA
    "Specific enthalpy of fluid flowing B->A";
end FlangeA;

connector FlangeB "Type-B connector"
  Types.Pressure p "Pressure";
  flow SI.MassFlowRate w "Mass flowrate";
  input Types.SpecificEnthalpy hAB
    "Specific enthalpy of fluid flowing A->B";
  output Types.SpecificEnthalpy hBA
    "Specific enthalpy of fluid flowing B->A";
end FlangeB;
```

Locally re-defined types are used in order to set reasonable non-zero default start values for the thermodynamic properties. The reader is referred to [3] for details about the connector design.

Thermal transfer is described by standard Modelica.Thermal.HeatTransfer connectors and components.

2.2 Medium models

Medium properties are computed by a replaceable medium model, similar to the BaseProperties model of the Modelica.Media standard library. The base model contains the pressure p, temperature T, density ρ, specific enthalpy h, and specific energy u of the fluid, as well as the partial derivatives with respect to pressure and enthalpy which are needed for the mass and energy balance equations.

The test cases described in this paper use a model of a compressible liquid with constant specific heat at

constant pressure, constant compressibility and constant thermal expansion coefficient.

2.3 Pump

Currently, a trivial pump model is employed, with prescribed flow rate: this could represent a pump equipped with an ideal mass flow rate controller. The prescribed flow rate is given by an input signal connector. The enthalpy increase due to the specific work added to the fluid is not taken into account, as it is negligible compared to the heat tranfer in the heater and radiator models.

2.4 Accumulator

Accumulators are usually employed to pressurize liquid-filled circuit and accommodate the expansion and contraction of the fluid due to the thermal expansion effect. Typical accumulators are built using a tank partially filled with air, so that the amount of water contained depends on the air pressure. The model includes the three-way T junction to the circuit, so that it has two fluid connectors.

A simple linear model has been used to compute the amount of liquid contained in the accumulator:

$$M = Cp\ ; \tag{1}$$

where C is the hydraulic capacitance; since the pressure has been selected as a state, the mass balance equation of the model

$$\frac{dM}{dt} = w_1 - w_2\ , \tag{2}$$

is written as

$$C\frac{dp}{dt} = w_1 - w_2\ , \tag{3}$$

w_1 and w_2 being the inlet and outlet mass flow rates. Since the flow rate of fluid going into and out of the accumulator is usually much smaller than the flow rate in the circuit, trivial energy balance equations are assumed, where the specific enthalpy of the fluid going out of the T junction is always equal to the enthalpy of the incoming fluid.

The steady-state equation for this component, which contains a dynamic mass balance, should be

$$\frac{dM}{dt} = 0\ ; \tag{4}$$

given the choice of states, the initial equation in the model is written as:

$$\frac{dp}{dt} = 0\ . \tag{5}$$

2.5 Lumped volume

Mass and energy storage are represented by the classical lumped-parameter mass and energy balance equations. Pressure and temperature are used as states.

$$M = \rho\ V \tag{6}$$

$$U = M\,u \tag{7}$$

$$\frac{dM}{dt} = V\left[\left(\frac{d\,\rho}{dp}\right)_T \frac{dp}{dt} + \left(\frac{d\,\rho}{dT}\right)_p \frac{dT}{dt}\right] \tag{8}$$

$$\frac{dh}{dt} = \left(\frac{dh}{dp}\right)_T \frac{dp}{dt} + c_p \frac{dT}{dt} \tag{9}$$

$$\frac{dU}{dt} = M\frac{dh}{dt} + \frac{dM}{dt}h - V\frac{dp}{dt} \tag{10}$$

The mass and energy balance equations

$$\frac{dM}{dt} = w_1 - w_2 \tag{11}$$

$$\frac{dU}{dt} = w_1 h_1 - w_2 h_2 + Q \tag{12}$$

are thus written using the results of equations (8)-(10). Here, M and U are the mass and energy of the fluid contained in the model, V is the volume, c_p is the specific heat at constant pressure, h_1 and h_2 are the specific enthalpies of the fluid entering and exiting the volume, and Q is the heat flow entering the volume.

The steady-state equations for this component are:

$$\frac{dM}{dt} = 0 \tag{13}$$

$$\frac{dU}{dt} = 0\ . \tag{14}$$

Given the choice of states, these equations can be more conveniently reformulated as

$$\frac{dp}{dt} = 0 \tag{15}$$

$$\frac{dT}{dt} = 0\ . \tag{16}$$

2.6 Pressure loss model

In order to avoid trouble with hard nonlinearities at this stage, a simple linear pressure loss has been assumed:

$$w = K\,\Delta\,p\ , \tag{17}$$

where Δp is the pressure drop across the component, w is the mass flow rate through it, and K is a constant flow coefficient. Future version of the model wil consider a density-dependent, quadratic pressure loss.

The energy balance is an isentalpic transformation between the inlet and the outlet.

2.7 Valve

The valve model is similar to the pressure loss, except that the flow coefficient can be modulated by varying the valve opening input signal u from 0 to 1.

$$w = K\,u\,\Delta p\ . \tag{18}$$

2.8 Pipe

Each pipe is described by a simple symmetric lumped-parameter model, with one volume describing mass and energy storage, and two adjacent pressure loss models describing the momentum balance.

2.9 Choice of physical parameters

The nominal operating point of the circuit assumes a flow rate of 1 kg/s, a thermal power of 84 kW, and a convective heat transfer to the environment such that the temperature of the radiator is 10 K above the ambient value of 300 K, while the heater temperature is 330 K. The pressure loss in the valve is 1 bar, as well as the pressure loss in the pipes, which is equally divided between the two half-pressure-loss models. The hydraulic capacitance of the accumulator is 3 kg/bar.

3 Initialization problems

3.1 Steady-state initialization of a closed circuit

The components of the circuit model have 5 potential state variables: the pressure and temperature of the two volumes, and the pressure of the accumulator.

Since the circuit is closed, the total mass of the fluid in the circuit must be constant, because there is no mass flow rate entering or leaving the system. Therefore, the system equations, by their very nature, imply that

$$\sum_j \frac{dM_j}{dt} = 0 \tag{19}$$

where M_j are the masses of the fluid in the components with storage, i.e., the accumulator and the two pipe volumes. If one now sums the initial equations (4) and (13) for the accumulator and volume components, the same equation is obtained. This means that the simulation equations and the steady-state equations for a closed system will always be linearly dependent. The corresponding initialization problem is therefore singular, and has an infinite number of solutions, corresponding to different amounts of liquid in the circuit or, equivalently, to different levels of pressure in the circuit.

It is important to note that no single component has singular initialization equations: the singularity only arises at the system level. It is therefore convenient to leave all the steady-state equations in the single components, and add one more initial equation at the system level, e. g. by specifying the pressure at one point of the circuit or, alternatively, by specifying the total mass of liquid in the circuit. This leads to an overdetermined system of initial conditions, which has one more equation than unknowns, but now has one unique solution.

3.2 Steady-state initialization with zero-pressure-loss flow components

Suppose that the pressure loss due to friction in the radiator is small, compared to other pressure losses in the circuit. In order to avoid highly nonlinear stiff equations, and to reduce the number of states in the system, a possible modelling option is to neglect the pressure loss entirely, i. e. use the equation:

$$\Delta p = 0 \tag{20}$$

in place of equation (17) for the two pressure loss models of the radiator. This might be an interesting option for control-oriented models, where a reduced number of states is often sought.

As a consequence, the pressure within the radiator volume and the pressure within the accumulator are bound to be equal, so that the resulting system has index 2. The index reduction algorithm gets rid of one of the two pressure states, so that there now is one more redundant initial equation, compared to the previous case, even though the overdetermined system of equation still has one unique solution.

As in the previous case, this situation does not depend on equations which are local to a single sub-model, but rather depends on the system-level structure of the overall model, due to the way the sub-model are connected. It is therefore very convenient if the user doesn't have to change the local initialization option for any sub-model, and still get

the unique steady-state solution for the initialization problem.

3.3 Steady-state initialization with idealized controllers (inverse simulation)

So far, open-loop simulation problems have been considered, in which the three inputs corresponding to the three actuators (pump speed, valve opening, burner power) are prescribed functions of time. One could then study a closed-loop control problem, in which, e. g., the burner power is used to control the radiator temperature to a given set point, using a PI controller. In this case, one more steady-state equation would be needed for the controller state, but this would not cause any further imbalance between the initial states and the initial equations. However, it would be necessary to tune the parameters of the PI controller in order to obtain a stable and satisfactory performance.

In some cases, one could be interested in evaluating the transient of the control variable (the burner power) corresponding to some external disturbance, assuming a very tight control, without worrying about the actual tuning of the controller itself. This kind of study is carried out easily in an a-causal context, by just removing the equation which assigns the prescribed value to the control variable, and adding an equation which prescribes the value of the controlled variable to be equal to the set point. This kind of approach is also know as inverse simulation problem (see, e.g., [5]). The prescribed set-point must be smooth enough in order for the inverse simulation problem to have a well-defined solution, but this is outside the scope of the initialization problem.

In the specific case considered in this paper, one could prescribe the value of the radiator temperature, in order to obtain the corresponding value of the heater power input. This can be done both with the system described in Sect. 3.1, as well as with the system described in Sect. 3.2. In both cases, since the radiator temperature is one of the system states, the connection of the plant model to the idealized controller will enforce an algebraic constraint on a differentiated variable; index reduction will have to be applied in order to get an index-1 DAE, and thus one more state will be eliminated. Once again, since this is a system-level issue, it would be nice not to be obliged to change the initialization options inside any specific sub-model of the plant, but rather keep the resulting overdetermined initial equation system, which still has one unique solution.

4 Numerical results

The three test cases described in Sect. 3 have been set up in the SimpleFluid library, described in Sect. 2. The problems have then been solved using the OpenModelica Compiler (OMC) version 1.4.3 [6]. The current solution algorithm is summarized here:

- The Modelica code is flattened, obtaining the declarations of all variables, parameters and constants, as well as the full set of equations and initial equations.
- Index reduction is applied, in order to obtained a reduced-order, index-1 system.
- The initialization problem $f(z) = 0$ is built, by adding the initial equations to the set of index-1 DAEs of the system; z is the vector including the algebraic variables, the state variables, and the state derivatives, while f is the vector of the residual functions. Note that, in general, $\dim(f) \geq \dim(z)$.
- The initialization problem is then solved by minimizing the norm of the residual vector $F(z)=\sum_j f_j^2(z)$, by using the Sequential Quadratic Programming optimization code described in [7]; the start values of all variables are used as an initial guess for the iterative algorithm. If the initialization problem has one solution, the minimum is unique and characterized by a zero residual.

OMC successfully solves all the three initialization problems described in Sect. 3, finding the corresponding initial steady state, provided that:

- all the thermodynamic variables (pressures, temperature, densities) are given a meaningful, non-zero start value – this is accomplished by extending the standard SI unit types with suitable default start attributes within SimpleFluid;
- the pressure and temperature states of the volumes and of the accumulator are given a start value close enough to the steady-state value.

Unfortunately, convergence of the initialization problem seems to be rather sensitive to the start values of the temperatures in the volumes: a start value of 300 K instead of 330 K for the heater volume is enough to make the algorithm fail.

5 Improvements and future work

Several improvement actions are proposed in this section, which will be tested in future versions of OMC.

First of all, the size of the optimization problem corresponding to the initialization problem can be roughly halved by just removing alias variables from the flattened model. Although this sounds like a trivial operation, care must be exercised in order to avoid getting rid of user-defined start values, which might have been applied to only one of the variables in the alias set. For this purpose, it might be useful to define a suitable priority indicator for start value modifiers, and select the alias variable with the highest priority start value in the set.

In order to further reduce the size of the optimization task, BLT partitioning of the initialization equation set could be performed, in order to split the original problem into smaller problems, to be solved sequentially. As the incidence matrix has more rows than columns, one has more degrees of freedom in selecting the row/column permutation than it is possible in the standard square problem. This is an open topic for further research. Tearing methods could also be very beneficial in this context.

Better scaling must be ensured to improve the robustness of the minimization algorithm. Currently, the state and algebraic variables z of the initialization problem, and the equation residuals $f(z)$ are directly used in the optimization problem. Some equations and some variables thus have a predominating influence on the optimization problem, due to bad scaling. For example, in the test case discussed in this paper, the mass flow rates have an order of magnitude of 1, while the pressures are around 10^6; the mass balance equations have residuals (i.e., flow rates) of the order of 1, while the energy balance equations might easily give residuals (i.e. powers) of the order of 10^5. This might explain the failure of the initialization algorithm even for small changes in the start values of the temperatures, since they mainly affect the energy balances, which have a larger influence on the residual norm than the mass balances.

To improve this situation, the algebraic and state variables might be normalized with their nominal values; the state derivatives might be normalized with the nominal values of the corresponding states, assuming a typical time scale of 1 second. On the equation side, residuals could be normalized with scale factors obtained by a Monte Carlo approach: these could be estimated by computing the residuals with random small variations of the corresponding values around their start values.

Convergence of the minimization algorithm might be improved by introducing penalty functions which are added to the objective function when the unknown variables gets out of their min-max interval. In fact, confidence intervals for the initial value are usually known, which are much narrower than min-max values during simulation – new minStart and maxStart attributes for Real types could be defined in Modelica, in order to specify the range during initialization.

Finally, homotopy methods might be considered in order to improve the robustness of the convergence for not too accurate choices of the start values.

In order to be able to evaluate the impact of all these actions, it is important to be able to monitor the progress of the iterative minimization algorithm, step by step. Improved diagnostic features (e.g., logging of iteration variable values) should then be implemented in OMC, which could also be useful for the diagnostics of the nonlinear solvers during simulation.

As the robustness of the initialization algorithm and the diagnostic capabilities are improved, it will be possible to increase the complexity of the test cases, first by introducing density-dependent, quadratic pressure losses in flow models, and then by trying more complex systems with larger numbers of equations and states.

6 Conclusions

Steady-state initialization problems for fluid systems are often naturally specified in terms of overdetermined systems of initial equations, having more equations than unknowns, but possessing just one unique solution. These problems can be solved using minimization algorithms. The paper motivates the need of such problems with reference to a simple test case, and presents results obtained with the OpenModelica compiler. Suggestions to improve the robustness of the OpenModelica solver are also given. The Modelica source code of all the test cases is available from the authors; contributions to improve the algorithms within the OpenModelica Compiler are welcome.

7 References

[1] Bachmann B., Aronsson P., Fritzson P.. "Robust Initialization of Differential-Algebraic Equations", *Proceedings of the 5rd Modelica Conference*, Vienna, Austria, 4-5 Sep 2006, pp. 607-614. http://www.modelica.org/events/modelica2006/Proceedings/sessions/Session6a2.pdf

[2] Casella F, Otter M., Proelss K., Richter C., Tummescheit H., "The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks", *Proceedings of the 5rd Modelica Conference*, Vienna, Austria, 4-5 Sep 2006, pp. 631-640. http://www.modelica.org/events/modelica2006/Proceedings/sessions/Session6b1.pdf

[3] F. Casella, A. Leva, "Modelling of thermo-hydraulic power generation processes using ModelicaModular Modelling in an Object Oriented Database", *Mathematical and Computer Modelling of Dynamical Systems, Modelling of Systems*, v. 12, n. 1, pp 19-33, 2006.

[4] F. Casella, A. Leva, "Modelica open library for power plant simulation: design and experimental validation", *Proceedings 3rd International Modelica Conference,* Linköping, Sweden, Nov 2003, pp. 41-50. http://www.modelica.org/Conference2003/papers/h08_Leva.pdf

[5] M. Thümmel, G. Looye, M. Kurze, M. Otter, J. Bals, "Nonlinear Inverse Models for Control", *Proc. 5th International Modelica Conference,* Hamburg, Germany, Mar 2005, pp. 267-279. http://www.modelica.org/events/Conference2005/online_proceedings/Session3/Session3c3.pdf

[6] Peter Fritzson, et al. "The Open Source Modelica Project", *Proc. 2nd International Modelica Conference*, 18-19 March, 2002. Munich, Germany See also: http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html

[7] M. J. D. Powell, "The NEWUOA software for unconstrained optimization without derivatives", *Proc. 40th Workshop on Large Scale Nonlinear Optimization*, Erice, Italy, 2004, paper DAMTP 2004/NA05.