Investigating Steady State Initialization for Modelica models

Hans Olsson[1] Erik Henningsson[2]

[1] Dassault Systemes AB, Sweden, hans.olsson@3ds.com

[2]Dassault Systemes AB, Sweden, erik.henningsson@3ds.com

Abstract

This paper investigates steady-state initialization both symbolically and numerically, and in particular demonstrates new ways of adapting the symbolic methods for finding steady-state solutions for Modelica models extending ideas that were previously manually implemented in libraries. The methods are compared on realistic Modelica models in Dymola.

Keywords: initialization, fluid models, differential algebraic equation, static simulation, steady state

1 Introduction

Steady state behavior for models is an important study – for several reasons. The steady state solution can be the goal of the particular study, when studying how parameters influence its characteristics, – and also a starting point for normal simulation studies, since starting from a steady-solution avoids uninteresting transients.

The goal of this paper is to investigate strategies for finding steady states of Modelica models. We survey different approaches and discuss their benefits and challenges. Additionally, we present strategies for automatic treatment of structural and numerical singularities arising due to the steady-state formulation. Such problems are for example encountered in fluid models. To the best of our knowledge, such automatic handling specifically for steady-state initialization has not previously been discussed in the literature even if several of the methods have been discussed.

In addition to true steady-state solutions where all derivatives are zero and the solution is unchanging, we as an outlook consider have quasi steady-state where some states are changing, but the solution is fundamentally time-invariant – e.g., a vehicle running at a constant velocity. This creates unique challenges that will be discussed later.

The stability of the steady-state solution could be analyzed by a linearization at that point, but we will not consider it in detail.

2 Variants of initialization

Normally Modelica models are initialized using the model equations, initial equations, fixed start-values as described in (Mattsson 2002), and supplemented by selecting non-fixed start-values as described in section 8.6.2 “Recommended selection of start-values” of Modelica 3.5 (Olsson (editor), 2021).

The problems are often numerically challenging and homotopy methods can be useful for handling that; see (Sielemann 2011), (Sielemann 2012), and (Casella 2011).

Note that the initialization is applied after the index-reduction, and we will thus primarily consider the ODE or index-1 DAE-formulation of the model that due to index-reduction may require additional conditions. Note that historically the index reduction algorithm, (Pantelides 1988) was proposed specifically to find those initial conditions.

Note that the difference between the ODE and index-1 formulation is important here as there might be initial conditions and/or start-values involving the algebraic variables. Additionally, index reduction sometimes lead to dynamic state selection, (Mattsson 2000) and the combination with steady state initialization poses specific problems that will be considered later.

Various libraries have different mechanisms for conditionally disabling initial equations and start-values.

2.1 Symbolic steady state initialization

Steady state initialization changes this to instead of selecting start-values we set derivatives to zero. This might be seen as simply changing from computing a solution $x(t)$ from (this is most easily seen in the ODE case):

$$\begin{aligned} \dot{x} &= f(x(t), t) \\ x(0) &= x_0 \end{aligned} \tag{1}$$

to computing it from:

$$\begin{aligned} \dot{x}(t) &= f(x(t), t) \\ 0 &= \dot{x}(0) \end{aligned} \tag{2}$$

However, even if this works in some cases there are number of issues in most cases:

- The Jacobian $\partial f / \partial x$ might be singular, either structurally or just at the initial point; indicating that there are multiple acceptable steady-state solutions; this is discussed in (Casella 2012). It can be difficult to give good numeric diagnostics for this, and we will return to this in Section 3.

- Some initial conditions might be specified, either to handle the singularity above or in order to get a specific quasi-steady-state solution.

These problems can be overcome using specific remedies as will be explained later.

2.2 Dynamic steady state finding

Another alternative is to numerically integrate forward from the normal initial conditions until a dynamic steady-state is reached (up to a certain precision). This is often simpler, and handles the issues above. Furthermore, by introducing pseudo dynamics, purely static models can be transformed to dynamic models. The dynamics are crafted so that the solution tends to a steady state equal to the static solution of the original model. This technique is often employed to break up algebraic loops for more robust solving of static problems. For both of these application it is important to automatically detect that the steady state has been reached so that the simulation can be terminated in a timely fashion.

However, there are other potential downsides when using the dynamic steady state initialization:

- The dynamic simulation usually takes longer than the static initialization.
- Double integrators – especially in combination with "quasi-steady-state", will tend to infinity.
- The model may not have a steady-state solution; but instead tend to a periodic solution, quasi-periodic, or even have a strange attractor.
- The solution may have state events during the solution. There might also be time events if integrating forward in time.

Methods to detect periodic steady states have previously been considered in the Modelica context. (Kuhn 2017) investigates techniques to automatically identify periodic steady states in Modelica models of electrical AC systems. An additional example is the Electrified Powertrains Library (EPTL), which feature components that terminate a simulation when a periodic steady state is reached.

It might be possible to avoid time events and periodic solutions by integrating from minus infinity towards zero, and use implicit Euler with large step-sizes to artificially dampen oscillations.

3 Symbolic steady state initialization

We will now consider the specific issues related to symbolic steady state initialization, and our general approach for solving it. Specifically we will describe how the changes to the basic approach handles various cases.

3.1 Outline of symbolic approach

The goal of the symbolic approach is to set up the modified steady-state problem to ensure that it is structurally non-singular, and numerically non-singular at the desired steady state solution.

We start from $\dot{x} = f(x(t), t)$ but instead of setting the entire $\dot{x}(0) = 0$ we use the well-known maximum-bipartite-matching; e.g., (Cormen 1990) for finding which elements of $\dot{x}$ that should be set to zero.

The matching is similar to the matching for transient simulation where we require a perfect (one-to-one) matching of derivatives to these equations. The matched variables and equations are also sorted into strongly connected components and each of them solved separately; but we will not discuss that in detail.

However, for initialization we instead attempt to match variables to $\dot{x} = f(x(t), t)$, prioritized to first match states and then derivatives of states so that all equations are matched to some variable – but not all variables is matched to an equation.

The states and derivatives of states that were unmatched in this initialization problem are then set to default values (normally zero for derivatives) and start-value for states. This matching is based on the variable incidences for each equation, but modified as will be explained later.

This procedure is then amended for an index-1 DAE by adding algebraic equations (and initial equations) and first matching all auxiliary variables.

An important aspect is that when matching states to the equations we prioritize them to get the result of section 8.6.2 "Recommended selection of start-values" of Modelica 3.5 (Olsson (editor) 2021).

Note that this is only used for solving the initialization problem and we then use the equations in the usual way for dynamic simulations.

This basic approach was implemented in Dymola in 2004 based on the similarity with the normal state-initialization. There are multiple variants of this; note that we symbolically manipulate the equations but do not aim to symbolically solve them in contrast to (Ochel 2014).

3.2 Avoiding structural singularities

For fluid models we have seen two specific causes of singularities. The first issue for fluid models, already handled in models in (Casella 2012), can be simplified to two tanks connected in a cycle where the outflow, f, from each tank depend on its mass, x, and some parameter A:

$$\begin{aligned} f_j &= g(x_j, A_j) \\ \dot{x}_1 &= f_2 - f_1 \\ \dot{x}_2 &= f_1 - f_2 \end{aligned} \tag{3}$$

This can be integrated forward in time, but if we attempt to compute the steady-state solution we get a redundant equation:

$$\begin{aligned} 0 &= f_2 - f_1 \\ 0 &= f_1 - f_2 \end{aligned} \tag{4}$$

The proposed solution for such cases is to consider all trivial equations when derivatives are set to zero, and see if they form cycles (as in this case). Trivial equations are equations that can be written on the form a=+/-b. This automates the procedure from (Casella 2012).

DOI 10.3384/ecp21181119

As previously indicated the symbolic selection of initial conditions is based on matching variables to equations based on their incidence. We thus modify the equation graph causing the cycle (in this case the two derivative-equations) by adding an incidence to an extra algebraic variable in them and introducing a new equation with incidence to the derivative-variables of these equations (the variable is called h below; the equation is left empty) which gives the graph:

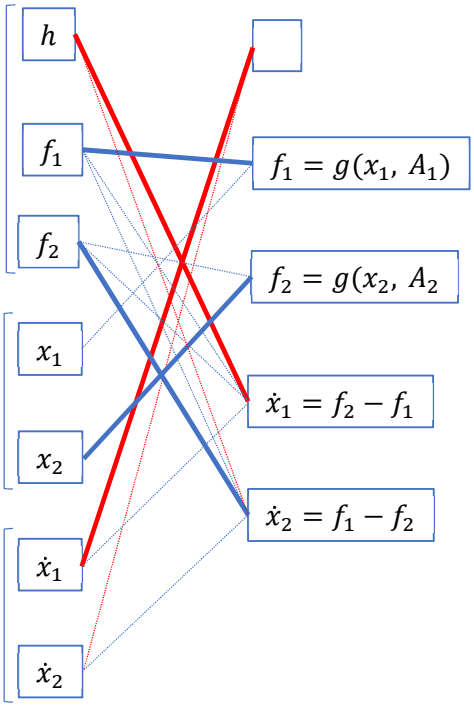

Figure 1 Matching for two tanks

The variables are in three groups, since we first match h, f_1, f_2; then attempt to match x_1, x_2 and finally attempt to match $\dot{x}_1, \dot{x}_2$. The matchings are solid lines and other incidences are dotted thinner lines.

The unmatched derivative $\dot{x}_2$ is set to zero, and the unmatched state x_1 becomes an arbitrary start-value (there is a priority depending on whether start-values are set and at what level, such that the start-value that is "highest up" is prioritized). The other state is initialized based on $f_1 = f_2$ giving $g(x_1, A_1) = g(x_2, A_2)$.

We then modify the new matching to remove the extra equation and extra variable h, but keep the previously matched variables. (There might be other ways of getting to this desired matching without temporarily adding an extra variable and equation and then removing them). This matches $\dot{x}_1$ with $\dot{x}_1 = f_2 - f_1$; and the equations are then as usual sorted into strongly connected components where each component in this case is scalar; and only the one involving x_2 require the solution of a non-linear equation.

From a modeling perspective another possibility would be to have an initial equation for the total mass in the system, replacing the arbitrary start-value for one of the masses. The chosen approach naturally handles that, and the initial equation can either be conditional on steady-state initialization or always applied.

3.3 Avoiding singularity at the solution

The second issue for fluid models can occur for any differentiated media such as the function for density as a function of the states:

$$\rho = d(T,p) \tag{5}$$

Which is differentiated to give:

$$\dot{\rho} = d_T(T,p)\dot{T} + d_p(T,p)\dot{p} \tag{6}$$

If we just solve the equations without symbolic processing as suggested in (Casella 2012) this is unproblematic.

However, if we want to perform a structural analysis to see which derivatives we can set to zero we get a problem. Specifically we might attempt to match T or p to this equation during steady initialization and attempt to compute them from it – because they influence $d_T(T,p)$ and $d_p(T,p)$.

Numerically this will not be well-behaved when we approach the steady-state solution since the influence of T and p on $d_T(T,p)\dot{T} + d_p(T,p)\dot{p}$ gradually disappear as the derivatives approach zero.

We handle that by modifying the incidence for all initialization equations by removing checking what happens if the state-derivatives are set to zero. If the equation no longer contains a non-derivative variable we remove the incidence from the equation to that variable when finding the steady-state solution.

3.4 Higher order derivatives

The original description separated variables into states and derivatives of states. For higher order derivatives that naturally occur in mechanical systems that is not always straightforward -- consider a simple rotational model:

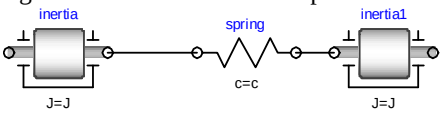

Figure 2 Two inertias

The rotational speed, w, is both a state and a derivative; and it may even have a start-value.

However, the state-variable w can be matched to the equation der(phi)=w; i.e. it will be treated as if it only were a derivative-variable. Thus such higher order derivatives cause no direct problem, and since the state is matched to equations its start-value is ignored (assuming there is no fixed=true). We might in the future prioritize start-values for such combinations of state/derivative instead of setting derivatives to zero, but other related issues are more important.

3.5 Dynamic State Selection

Consider the pendulum in Cartesian coordinates.

```
model Pendulum
  Real vx,vy;
```

```
  Real x(start=0.5),y(start=0.7);
  Real f;
equation
  x^2+y^2=1;
  der(x)=vx;
  der(y)=vy;
  der(vy)=-9.81+f*y;
  der(vx)=f*x;
end Pendulum;
```

Note that the start-values are inconsistent guess-values.

In Modelica tools the pendulum is usually solved using the dynamic dummy derivative method, since there is no static selection of states that generate a good solution.

Note that there are two dynamic dummy derivative systems, each selecting one state among two possibility – one for positions and one for velocities.

However, that is an implementation detail and from a user perspective we prefer to hide those dummy variables and instead give start-values for the normal variables. Using high order derivatives all differentiated equations can be written as:

$$x^2 + y^2 = 1 \quad (7a)$$
$$2\dot{x}x + 2\dot{y}y = 0 \quad (7b)$$
$$2\ddot{x}x + 2\dot{x}^2 + 2\ddot{y}y + 2\dot{y}^2 = 0 \quad (7c)$$
$$\ddot{y} = -9.81 + fy \quad (7d)$$
$$\ddot{x} = fx \quad (7e)$$

Simply ignore the dummy derivatives?

Hiding the dummies causes two problems: the first is that it looks as if we need two steady-state conditions and have four free derivatives (two velocities and two accelerations) that we could set to zero for steady-state initialization, but in reality there are only two free derivatives in total (one velocity and one acceleration).

If we ignore that and set the two velocities to zero that directly collapses one existing equation:

$$2\dot{x}x + 2\dot{y}y = 0 \xrightarrow{yields} 0 = 0 \quad (8)$$

Similarly setting the two accelerations to zero simplifies another existing equation:

$$2\ddot{x}x + 2\dot{x}^2 + 2\ddot{y}y + 2\dot{y}^2 = 0 \xrightarrow{yields} 2\dot{x}^2 + 2\dot{y}^2 = 0 \quad (9)$$

Which have the following Jacobian with respect to the velocities:

$$\begin{bmatrix} 2x & 2y \\ 4\dot{x} & 4\dot{y} \end{bmatrix} \quad (10)$$

When the velocities goes to zero the second row tends to zero.

Solving that problem leads to the second problem, that some choices of velocity and acceleration-variables for steady state lead to singular systems, which is exactly the reason we used dynamic dummy-derivatives in the first place. It is not certain that a dummy-derivative system is singular at exactly the steady state solution, but it seems likely and will occur in this case if we set

$$\ddot{y} = \dot{y} = 0 \quad (11)$$

which gives the following Jacobian with respect to $x, y, \dot{x}, \ddot{x}, f$ (where the second and third rows are zero at the solution):

$$\begin{bmatrix} 2x & 2y & 0 & 0 & 0 \\ 2\dot{x} & 0 & 2x & 0 & 0 \\ 2\ddot{x} & 0 & 4\dot{x} & 2x & 0 \\ 0 & -f & 0 & 0 & -y \\ -f & 0 & 0 & 1 & -x \end{bmatrix} \quad (12)$$

Directly using the dummy derivatives

An alternative would then to be to attempt to set the actual derivatives of the dynamic dummy derivative method to zero. But that seems underspecified, since it involves projecting the original derivatives using an unspecified projection matrix.

This approach might work using an iterative scheme where the projection matrix is recomputed until convergence, but it seems needlessly complicated and it is not obvious that it will generate a unique solution.

Proposed solution

The proposed remedy here involves modifying the initial equations in a consistent way.

First we set *all* derivatives in each dummy derivative system to zero as if we ignore the dummy derivatives, and then we balance that by removing the corresponding differentiated constraint that becomes identically zero for that choice (after verifying that it is in fact the case).

Additionally we propagate these zeros to other dummy derivative systems, since normally the acceleration constraints also involve velocities.

For the pendulum this means that we want to solve:

$$x^2 + y^2 = 1 \quad (13a)$$
$$\dot{x} = 0 \quad (13b)$$
$$\dot{y} = 0 \quad (13c)$$
$$\ddot{y} = -9.81 + fy \quad (13d)$$
$$\ddot{x} = fx \quad (13e)$$
$$\ddot{x} = 0 \quad (13f)$$
$$\ddot{y} = 0 \quad (13g)$$

and verify that the differentiated constraints are zero:

$$2\dot{x}x + 2\dot{y}y = 0 \quad (14a)$$
$$2\ddot{x}x + 2\dot{x}^2 + 2\ddot{y}y + 2\dot{y}^2 = 0 \quad (14b)$$

4 Future work: Quasi Steady State

Quasi steady state, where some derivatives have non-zero values is important in practice. The goal would be to directly study the model in operation without unnecessary transients. A realistic example would be a model of a car running at 60 km/h.

The goal of this section is two-fold; both to indicate how a tool in the future could automatically find these solutions, and also to demonstrate the problem to avoid the unintended solutions shown here.

4.1 Why quasi steady state is complicated

Let us start by a simple example of two connected inertias.

DOI 10.3384/ecp21181119

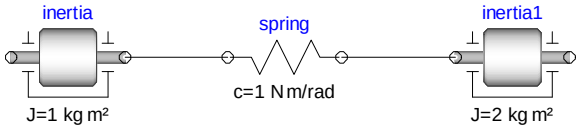

Figure 3 Two Inertias, Quasi Steady State

Assuming we want quasi steady-state and set inertia.w(start=5, fixed=true); something odd happens in this example. Just setting all other derivatives to zero makes the other inertia starts stationary and then we get a periodic solution for the angular velocities.

If we use a damper (or spring-damper) the states will as default be in the damper and thus as default the damper will have derivative zero, but if we set options to avoid states in the damper there is a risk of a similar non quasi-steady-state solution.

Note that it is not necessarily that we give a rotational velocity – a more realistic case is that we attach an engine (can even be a simple constant torque) to one inertia and some losses such as bearing-friction to the other. The result is the same, we can set one acceleration to zero, but not the other; and the solution will then start with a similar transient until reaching a quasi steady state.

Obviously this is not the desired quasi-steady solution, and the idea with investigating a small example is to find an approach that can be applied to a large system, like a car.

One approach is that instead of selecting initial conditions among the existing derivatives we add the initial equations $\dot{w} = 0$ and $\ddot{w} = 0$, and that is similar to treating w=5 as a normal non-initial equation during index-reduction. However, if we had another spring (or spring-damper) followed by an inertia we would need to set $\dot{w} = 0$ and $\ddot{w} = 0$ for this new inertia, etc.

Partially this is just prioritizing setting high order derivatives to zero, i.e. $\dot{w} = 0$ instead of $w = 0$, but $\ddot{w}$ does not even exist in the model, and even constructing $\ddot{w}$ in the first inertia does not create the next one. Thus further investigations are needed. We can also consider a clutch instead of a spring-damper, and a solution can have the two inertias rotating with different (steady-state) speeds.

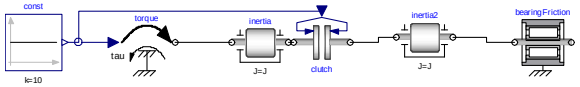

Figure 4 Advanced Quasi Steady State

An additional complication occurs if there are multiple disconnected mechanical systems – the initialization procedure outlined here will allow them to move independently of each other.

This also indicates that the original name we used for the approach "DefaultSteadyState" is misleading, since using steady state as "default" for a few variables does not guarantee a meaningful result.

4.2 Simple quasi steady state theory

Assume we have a system:

$$\dot{x} = f(x) \tag{15a}$$

and we want to generalize the notion steady state to cases where the derivative is non-zero but the system does not change. The simplest possibility is that the derivative is constant which gives that the following should be valid for all points in time:

$$\dot{x} = f(x + \dot{x}t) \tag{15b}$$

This implies that the derivative is in the non-trivial null-space

$$\ker\left(\frac{\partial f}{\partial x}\right) \tag{16}$$

and such null-spaces exist if the model is translationally invariant, such that y=T(x,p) behaves the same as x, with the restriction that the transformation has determinant 1 (and is differentiable in p). In general quasi steady state meaning that all points on the trajectory are "equivalent" seems like a good definition for quasi steady state, and smooth transformations allow that, and for a general fixed transformation we have

$$\dot{x} = \left(\frac{\partial T}{\partial x}\right)^{-1} f\big(T(x,p)\big) \tag{17}$$

For translational invariance $\left(\frac{\partial T}{\partial x}\right) = I$ and we can select a trajectory as a varying transformation of one point (since it always gives the same derivative). The restriction is then that the derivatives are given as $\frac{\partial T}{\partial p}$.

The benefit of this formulation can be seen for second order 1D-mechanical systems where we directly see that velocities should be in this null-space. Compare this with setting $\ddot{w} = 0$ (i.e., setting the third order derivative of the angle to zero) that require differentiating the model equations an additional time. Both formulations ensure quasi steady state behavior.

The null-space can be fairly simple, e.g. all absolute positions (with equal weight), and without relative positions. If we have a rotational motion with gears it is less trivial, but still straightforward.

4.3 Multi-dimensional fixed speed rotations

However, if we want to consider a car turning with a fixed steering angle (i.e. with a constant yaw) it becomes complicated. In general we propose the equation:

$$\left(\frac{\partial T(x_0,p)}{\partial p}\right)\dot{p} = f\big(T(x_0,p)\big) \tag{18}$$

The desired solution for rotation can be seen a pure rotation – but around an unknown point in space, and for the positions the second derivatives are thus non-zero (and proportional to the distance from the rotation center for absolute coordinates). If we use $p = \{r_x, r_y, \theta\}$ as invariant position and angle of the main object and then relative coordinates, x, we have the invariants:

$$\ddot{r} = f\big(\dot{r},\theta,\dot{\theta},x,\dot{x}\big) = e^{R\theta} f\big(e^{-R\theta}\dot{r},0,\dot{\theta},x,\dot{x}\big) \tag{19a}$$

$$\ddot{\theta} = g\big(\dot{\theta},x,\dot{x}\big) \tag{19b}$$

$$\ddot{x} = h\big(\dot{\theta},x,\dot{x}\big) \tag{19c}$$

with the solution $\ddot{x} = 0, \dot{x} = 0, \ddot{\theta} = 0, \ddot{r} = \dot{r}R\dot{\theta}$. For a rocket in space all positions and angles can be invariant, but for more earthly applications, such as a typical car (on a tarmac that is big, flat, and even), the positions in the plane are the invariant positions, and the yaw angle the invariant angle (the yaw velocity is $\dot{\theta}$).

This seems straightforward, but it is not clear how to perform this when only using the equations - without any additional knowledge. Additionally if the road surface depends on the position or is slanted, there are no solutions that rotate with a fixed speed.

However, this is the desired solution if we want to set the speed for the car (or have the engine running), and have the steering wheel off-center (or in general anything that makes the car non-symmetric) – i.e. it is the natural extension of the simple translational case for quasi steady state to 3-dimensions and also the special case of 2-dimensions (Höbinger and Otter, 2008).

Note that this is mostly restricted to mechanical systems, due to lack of invariants in other cases. Electrical systems are invariant with respect to the potentials – even more generally than mechanical systems (the potential can be any time-varying function), but grounding the circuit eliminates that – and grounding is used for normal simulation as well.

5 Evaluating the methods

There are three factors we want to consider for these methods:

- Does is find the desired steady-state solution?
- How easy is it to set up the problem?
- How quickly do we compute the solution?

5.1 Furuta pendulum

In order to dynamically find a steady state solution we currently have to add dampers to all joints to ensure that the steady state solution is asymptotically stable. And then set flag to find dynamic steady-state.

To generate the symbolic steady state solution the dampers in Figure 5 are not necessary (although possible), just setting the symbolic steady-state flags suffice. However, unless we set the guess-value for R2.phi close to zero this generates an unstable steady-state solution.

The symbolic handling automatically detects that R1.phi is a free variable, since the rotation axis is aligned with gravity. It is thus selected as a fixed start value in accordance with Section 3.2.

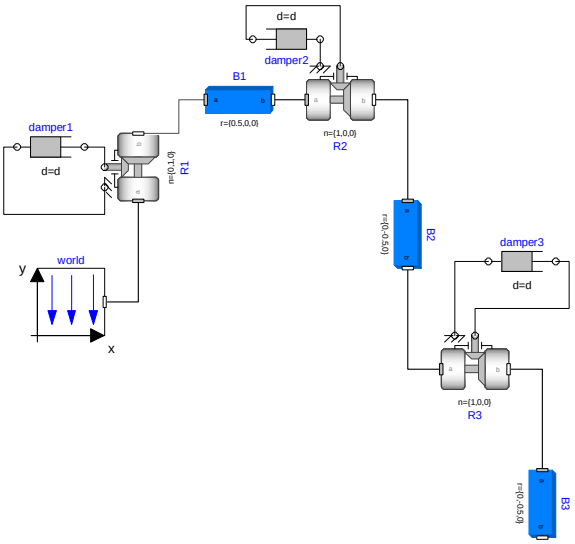

Figure 5 Damped Furuta Pendulum

5.2 Three tanks

This is a simple model in the Fluid package in the standard library.

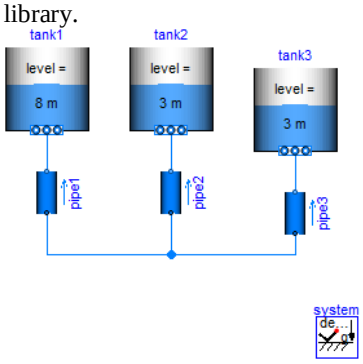

Figure 6 Three Tanks Steady State

Merely simulating the model gives the following plot (converging to a steady state):

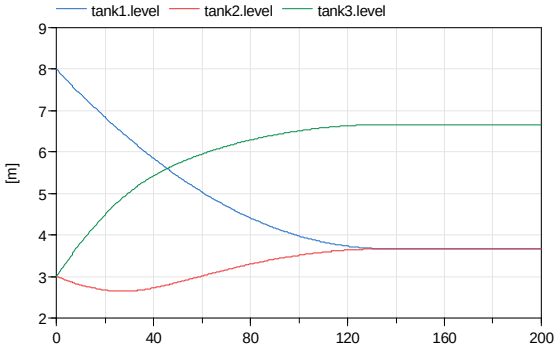

Figure 7 Solution for Three Tanks Steady State

There is a global setting for initialization in the model which has three relevant possibilities:

- FixedInitial – the default giving the plot
- Steady-state initial
- Initial guess-value

DOI 10.3384/ecp21181119

The initial guess-value in combination with steady-state flags finds a desired solution, and reports that one tank level and all tank temperatures are free initial values.

The built-in steady-state initial setting also finds a steady-state solution and reports the same free initial values, but additionally generate a diagnostic that there are four redundant consistent initial conditions are automatically removed:

```
  Removed the following equations which
are redundant and consistent:
der(tank3.U) = der(tank3.m)*tank3.medium.u
  + tank3.m*der(tank3.medium.u);
der(tank1.U) = der(tank1.m)*tank1.medium.u
  + tank1.m*der(tank1.medium.u);
der(tank2.U) = der(tank2.m)*tank2.medium.u
  + tank2.m*der(tank2.medium.u);
pipe1.port_a.m_flow+pipe2.port_a.m_flow
   +pipe3.port_a.m_flow = 0.0;
```

5.3 HeatLosses in MultiBody

This example model is interesting because it mixes different domains and has dynamic state selection.

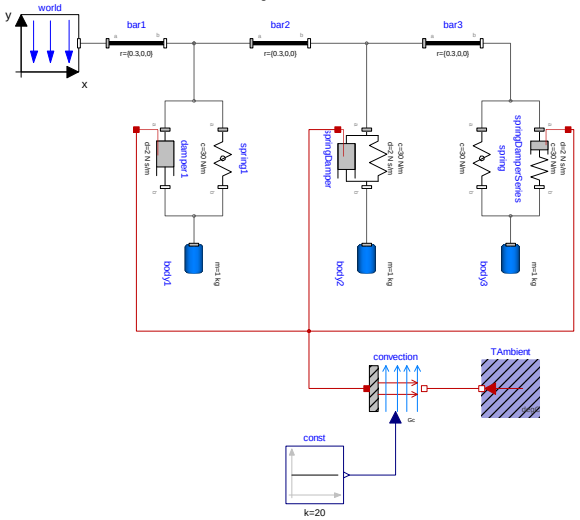

Figure 8 Multiple Springs with Heatlosses

The heat-losses are due to friction in the dampers, and should be zero in steady-state (confirmed by examining the solution at steady-state).

The symbolic steady-state settings directly finds the steady-state solution (the straight lines in the diagram) matching the dynamic solution.

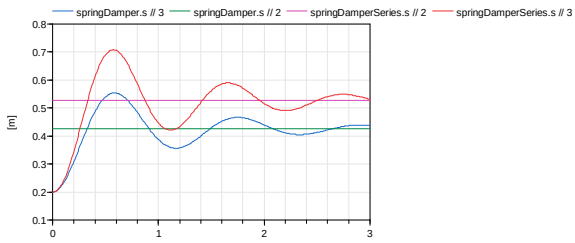

Figure 9 Solution for Multiple Springs with Heatlosses

The dynamic steady-state does not quickly find that the solution has converged, since there is a hidden slowly damped oscillation.

5.4 Quasi-steady state vehicle

Using constant torque for a simple translational vehicle with a non-rigidly attached trailer:

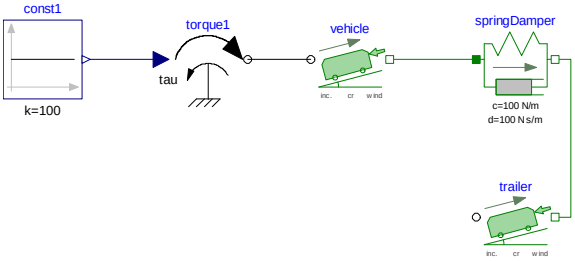

Figure 10 Quasi Steady-State for Simple Vehicle

Using dynamic steady state automatically finds the steady-state velocity (after 1 minute).

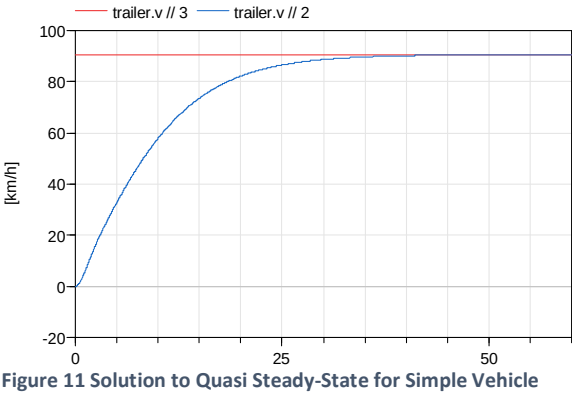

Figure 11 Solution to Quasi Steady-State for Simple Vehicle

Since the quasi-steady symbolic solution is not yet implemented we would have to manually add the corresponding initial equations based on Section 4.2 (the first two give that the derivatives are in the null-space and the final one is setting a second order derivative to zero):

```
vehicle.v=trailer.v;
vehicle.a=trailer.a;
vehicle.a = 0;
```

directly giving the solid line above.

5.5 Implementation notes

The symbolic steady state initialization with the improvements listed in section 3 were implemented already in Dymola 2020, but Dymola 2022 and 3DExperience 2022x adds the possibility of ignoring some state initializations in the model which makes it easier to perform the tests. The flags used are:

```
Advanced.Translation.
 DefaultSteadyStateInitialization=false;
Advanced.Translation.
 DefaultSteadyStateInitializationLevel=1;
```

Finding the dynamic steady-state initialization was implemented in Dymola 2022, and is enabled by the flag:

```
Advanced.Simulation.
```

```
SteadyStateTermination=true;
```

By default the simulation is automatically terminated when all state derivatives have an absolute value less than 2 % of the scale of the corresponding state, taking into account the time scale of the simulation. This default tolerance is chosen in accordance with the common definition of *settling time* within control theory (Tay, Mareels and Moore 1998). The tolerance can be modified by

```
Advanced.Simulation.
   SteadyStateTerminationTolerance
```

for details see (Dassault Systèmes 2021).

5.6 Evaluation

Both methods quickly find a steady-state solution, and are fairly easy to set up.

The downside of finding dynamic steady-state is that one must ensure that the system is sufficiently damped to reach a steady state; whereas the downside of symbolic steady state is that it may find an unwanted steady-state solution.

6 Outlook for standardization

The new features are specific to Dymola, and not standard Modelica.

The Modelica Language has no features for switching between setting values for states and steady-state initialization. However, many Modelica models have such settings, locally and/or using an inner component to have a "global" setting.

The examples show that such settings, when they exist, work similarly as the tool-setting for symbolically solving the steady-state problem. This means that any standardization effort needs to ensure that the existing models can seamlessly switch to the new formulation which will be an additional effort.

One downside of having the steady-state setting in the model is that although index-reduction automatically handles algebraic couplings for the dynamic equations that does not automatically remove initialization equations. However, that is not a major issue as tools can detect such redundant initial equations and automatically remove them.

A specific issue with having steady-state settings locally in components is that this can easily set up quasi steady-state problems generating undesirable solutions as indicated above.

7 Conclusions

This paper demonstrates that Modelica allows powerful initialization techniques, both symbolic and numeric. The methods have been implemented in Dymola 2022 and 3DEXPERIENCE 2022x.

However, to be standardized in Modelica this must be integrated in models complementing the existing model settings and it must be possible to detect and preferably solve quasi steady state problems.

Acknowledgements

Some of this work has been part of the ModeliScale research project, and we also acknowledge the customers providing us with challenging models.

References

Dassault Systèmes. (2021) Dymola 2022: *Dymola, Dynamic Modeling Laboratory, User Manual 2A: Model Development Tools.* Dassault Systèmes AB, Lund, Sweden.

Casella, Francesco, and Michael Sielemann, and Luca Savoldelli (2011) "Steady-state initialization of object-oriented thermo-fluid models by homotopy methods" In *Proceedings of the 8h International Modelica Conference.* 86-96

Casella, Francesco (2012): On the Formulation of Steady-State Initialization Problems in Object-Oriented Models of Closed Thermo-Hydraulic System.

Cormen, Thomas H. and Charles E. Leiserson, and Ronald L. Rivest (1990) *Introduction to Algorithms.* MIT Press. ISBN 0-07-013143-0

Höbinger, Mathias and Martin Otter (2008): "PlanarMultiBody A Modelica Library for Planar Multi-Body Systems". In: *Proceedings of 6th International Modelica Conference* 549-546

Ochel, Lennart, and Bernhard Bachmann and Francesco Casella (2014): "Symbolic Initialization of Over-determined Higher-index Models" In: *Proceedings of the 10h International Modelica Conference.* 1179 – 1187.

Olsson, Hans (editor) (2021): Modelica A Unified Object-Oriented Language for Systems Modeling Language Specification Version 3.5.
URL: https://specification.modelica.org/maint/3.5/MLS.pdf

Kuhn, Martin Raphael (2017): "Periodic Steady State Identification for use in Modelica based AC electrical system simulation". In: *Proceedings of the 12th International Modelica Conference.* 493 – 505.

Mattsson, Sven Erik, and Hans Olsson and Hilding Elmqvist (2000) "Dynamic Selection of States in Dymola". In: *Modelica Workshop 2000.* 61 –67.

Mattsson, Sven Erik, and Hilding Elmqvist, and Martin Otter, and Hans Olsson (2002) "Initialization of hybrid differential-algebraic equations in Modelica 2.0". In: *Proceedings of 2nd International Modelica Conference.* 9-15.

Pantelides, Constantinos (1988) "The Consistent Initialization of Differential-Algebraic Systems" In *SIAM Journal on Scientific and Statistical Computing 9:2, pg 213-231.*

Sielemann, Michael, and Francesco Casella, and Martin Otter, and Christoph Clauß, and Jonas Eborn, and Sven Erik Mattson, and Hans Olsson (2011) "Robust Initialization of Differential-Algebraic Equations Using Homotopy" In *Proceedings of the 8h International Modelica Conference.*

Sielemann, Michael (2012) "Probability-One Homotopy for Robust Initialization of Differential-Algebraic Equations" In *Proceedings of the 9h International Modelica Conference.* 223-236.

DOI 10.3384/ecp21181119

Tay, Teng-Tiow, and Ivan Mareels, and John B. Moore (1998)
High Performance Control. Birkhäuser Basel.