Symbolic Initialization of Over-determined Higher-index Models

Lennart Ochel Bernhard Bachmann
lennart.ochel@fh-bielefeld.de bernhard.bachmann@fh-bielefeld.de
Bielefeld University of Applied Sciences, Department of Mathematics and Engineering
Am Stadtholz 24 - 33609 Bielefeld, Germany

Francesco Casella
francesco.casella@polimi.it
Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria
Piazza Leonardo da Vinci, 32 - 20133 Milano, Italy

Abstract

The quantity of initial equations required in an object-oriented model can only be determined at system level. Since Modelica models are generally designed by components, it is difficult to calculate the amount of initial equations needed at system level, especially when changes are applied to the model, e.g. by adding or removing components. Therefore, it is more convenient to define initial equations at component level. Due to component connections, algebraic dependencies between states may be introduced, which eventually lead to the removal of states when symbolic index reduction algorithms are applied. In this process, the corresponding initial equations are not automatically removed, which results in an over-determined initial system.

This paper describes an algorithm that detects such redundant equations and determines if they are consistent or not. Consistent redundant initial equations can thus be removed automatically, and inconsistent ones can be reported to the modeler. A prototype of the algorithm is implemented in OpenModelica, tested on several representative cases, and compared to previously presented concepts.

Keywords: initialization; higher-index; simulation; over-constrained

1 Introduction

1.1 Statement of the Problem

Initial equations in Modelica are usually defined at the component level, and they are as many as the dynamic variables of the component, i.e., the potential states.

When connections are made, connection equations can induce algebraic constraints on dynamic variables. The dummy derivatives algorithm is used by many Modelica tools to dispose of some potential states and obtain an index-1 problem. As a result, there will be more initial equations than states, leading to an over-constrained initialization problem.

It is agreed that index reduction is necessary in object-oriented modeling to achieve full modularity without compromises, and suitable means to handle it have been developed over time, so it is obviously necessary to extend the handling to initialization as well. In the majority of cases the over-constrained initialization problem turns out to be consistent, and should therefore be handled automatically, without any intervention by the end user; inconsistent initialization problems should be reported in a user-friendly way.

1.2 Overview of Existing Solutions

OpenModelica has been using a numerical approach to solve this problem for a long time, as discussed in [1]. The initialization problem is turned into an optimization problem, where the sum of square of all residuals is minimized; if the problem is consistent, then the minimum is zero, otherwise the inconsistency is spread among equations, which is generally not a good idea. However, solving an optimization problem is much harder and time-consuming than solving a system of equations, and it might easily be possible to get trapped in local minima. Also, convergence problems quickly get worse when increasing the size of the system, up to the point where a solution cannot be reliably found unless guess values very close to the solution are given to all the problem unknowns. This gets even worse in the case of hybrid models, where some parts of the system contain discrete equations. Even for determined initialization systems, the numerical approach is only applicable in very special cases. In practice, this limits the application of the method to very simple models. So far, large-scale and hybrid models are reliably initialized using the symbolic initialization method described in [2], which has been further developed to also handle over-determined systems.

To our knowledge, there is no other Modelica tool available that has a general approach to handle over-determined initialization problems. The package *ModelicaTest*, which is offered by the Modelica Association together with the Modelica standard library (MSL), has been extended to provide a free-accessible set of appropriate test models. This can be used to compare the excellence of the initialization capabilities of different tools.

1.3 Structure of the Paper

In Section 2, an simple introductory example is discussed in detail to demonstrate the problem being tackled in this paper. In Section 3, an algorithm is presented to locate redundant equations which arise when symbolic index reduction is applied, and detect whether they are consistent or not. In Section 4, it is shown how the proposed algorithm works. A few simple test cases and the results of a more involved test case are discussed. Section 5 concludes the work with final remarks and suggestions for future work.

2 Introductory Example

This section describes how over-constrained initialization problems arise, by means of a simple electric circuit model, where two series-connected capacitors are connected to a constant voltage source. A graphical representation is shown in Figure 1 and an equation-based model description where all alias variables have been removed is shown in Listing 1.

Both of the capacitors introduce a potential state u_i. Due to component-based modeling, both capacitors may also introduce initial equations, for example $u_i = 5$.

An algebraic constraint among potential states is introduced by connecting the capacitors in parallel to a voltage source, so this model has index-2. Hence, symbolic index reduction is used (see [3], [4]) to transform this system into an equivalent index-1 system of lower order. During this process, one of the potential

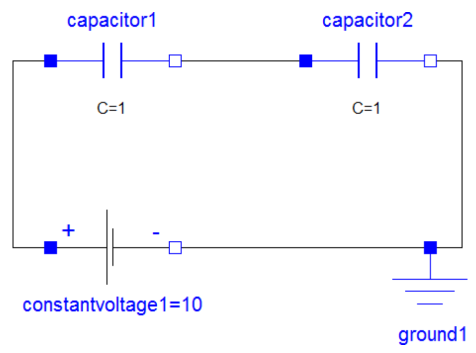


Figure 1: Introductory example - object diagram

states becomes an ordinary algebraic variable and the additional equation $0 = der(u_1) + der(u_2)$ gets introduced to keep the dynamic system determined.

```
model example
  Real u = 10;
  Real u1;
  Real u2;
  Real i;
  parameter Real C = 1;
initial equation
  u1 = 5;
  u2 = 5;
equation
  i = C*der(u1);
  i = C*der(u2);
  u = u1 + u2;
end example;
```

Listing 1: Introductory example - flat Modelica model

After the dynamic system is transformed to index-1, the initial equations will be added. As a result, an over-determined system arises that needs to get matched. There will be at least one unmatched equation for each redundant initial equation. Also note that there is no unique matching (besides the matching within each strong component) due to the over-determined system structure; different matchings are possible, which leave out different unmatched equations and possibly lead to different sets of strong components. Figure 2 shows one possible matching that will be used for the next steps; the gray equation is the unmatched one.

If all unmatched equations are consistent, then they can be removed and the initial solution can be calculated using robust and efficient algorithms designed for square problems. In order to check the consistency, the matching digraph gets first transformed into a directed graph by replacing each non-matching edge with an arc going from the E-node to the V-node, then by collapsing each V-node with its matching E-node.

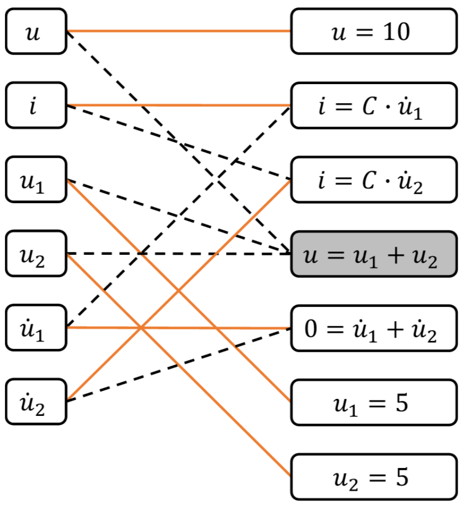
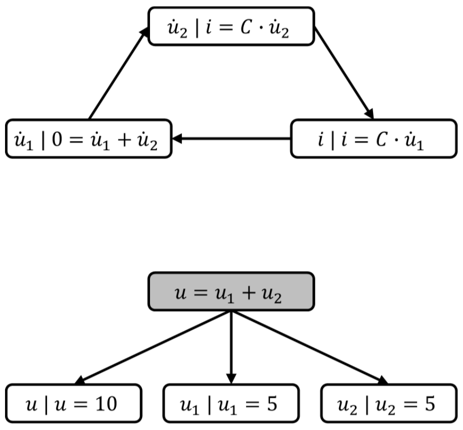


Figure 2: Introductory example - matching

Next, Tarjan's algorithm [5] gets applied on the directed graph, ignoring the unmatched equations, to find out strong components. The result of this procedure is presented in Figure 3, with one strong component at the top of the graph.

The symbolic consistency check will be performed on this graph. The basic idea is to only consider the sub-graph which involves the initial equations and the unmatched equations, in this case the lower part of Figure 3, and to recursively and symbolically solve the equations starting from the sinks and going up to the unmatched equations. In this case, once the three sink node equations have been solved, the gray equation becomes $10 = 5+5$, which is equivalent to $0 = 0$, and thus redundant.



$\dot{u}_2 \mid i = C \cdot \dot{u}_2$
$\dot{u}_1 \mid 0 = \dot{u}_1 + \dot{u}_2$
$i \mid i = C \cdot \dot{u}_1$
$u = u_1 + u_2$
$u \mid u = 10$
$u_1 \mid u_1 = 5$
$u_2 \mid u_2 = 5$

Figure 3: Introductory example - directed graph of the unmatched equation

In general, however, it will not be possible to symbolically solve all the chain of equations leading to the unmatched equation. It is therefore essential to select a particular matching that makes this possible. A concrete algorithm that does so is presented in the next section.

3 Algorithm for Redundant Equation Detection

The goal of this algorithm is to find redundant equations and remove them if they are consistent, or issue an error if they are not. A symbolic approach is preferred, because it avoids the need of setting more or less arbitrary numerical thresholds and using iterative solvers. In general, solving the full initialization problem symbolically is not feasible, because it often contains large and nonlinear coupled systems of equations that cannot be solved in closed form. Therefore, a symbolic approach should aim at finding a suitable subset of the initialization problem that is easy to solve symbolically, to check if there are redundant and consistent equations, so as to remove them.

In a system with n equations and m variables, with $k = n - m$ too many equations, there are $\binom{n}{k}$ possible sub-sets of equations that may be removed to make the problem square. However, the resulting problem need not only be square, but also have a solution, and many of these sets cannot be removed, because they contain essential (and not redundant) constraints. Hence, an algorithm is needed that efficiently finds those sets that can be removed without losing essential information.

In addition to that, each set of removed equations corresponds to a unique matching (ignoring the different matchings within strong components) of the remaining system. Depending on this matching, a consistence check can be performed by recursively evaluating the sorted subsets. The effort for this evaluation strongly depends on the selected equation dependencies, i.e., on the selected matching.

In practice, such a subset can normally be found, because initial equations are usually linear, involving one unknown, e.g. $x = x_0$, or $\mathrm{der}(x) = 0$. State derivatives usually show up linearly in balance equations, because they stem from the derivative of some basic quantity (mass, energy, momentum, charge) via differentiation. Connection equations, which usually provide the constraints that make the problem high index, are also linear. In most cases, it should then be possible to symbolically prove that the problem is either consistent or inconsistent, by means of symbolic computations, because the equations to be solved symbolically will belong to the above-mentioned categories.

3.1 Proposed Algorithm

1. Apply index reduction and state variable change to the dynamic problem, using the dummy derivatives algorithm.

2. Add the initial equations to the set to form the initialization problem.

3. Build the corresponding E-V graph.

4. Run the matching algorithm; some unmatched equations will remain at the end of the process (one for each redundant initial equation).

5. Transform the E-V digraph into a directed graph by first replacing each non-matching edge with an arc going from the E-node to the V-node, then by collapsing each V-node with its matching E-node.

6. Run Tarjan's algorithm on the directed graph, ignoring the unmatched equations, to find the strongly connected components, and collapse each strong component in a single node.

7. Starting from the sink nodes and proceeding recursively towards the source nodes (unmatched equations), symbolically solve each equation (or system of equations) for its unknown(s) and substitute the result in all the nodes that have arcs pointing to the solved equation node and are needed to validate the unmatched equations.

   If the symbolic solution of one equation is not possible (e.g., a sub-expression becomes $0/0$), then try to change the matching from step 4 for the corresponding equation and go back to step 5. If there is no other matching, then the algorithm aborts, and it is neither possible to draw any conclusion on the consistency of the problem, nor to reduce the problem to a square one.

8. If all the unmatched source nodes contain equations equivalent to $0 = 0$, then the over-determined system is consistent, and it is possible to turn it into a square equivalent system by just removing all the unmatched equations. If there are one or more source nodes containing equations equivalent to $0 = 1$, then the system is inconsistent. For diagnostic purposes, it is possible to report for each node the set of connected equations which are inconsistent. This will help the end user to identify the source of the inconsistency and possibly remove it.

The calculation of the matching in step 4 is essential for this approach. Due to the over-determined system structure, in general various matchings are possible. If a matching for all variables exists that contains no algebraic loops, then it is preferred and should be tried first. Therefore, Tarjan's tearing algorithm described in Cellier's book [6] is applied to the over-determined equation system.

A recursive evaluation of the equation system is possible if and only if during sequential evaluation each next equation depends on exactly one more unknown variable. This means that within the equation system at least one equation exists, which depends just on one variable. Tarjan's algorithm detects this fact, matches the variable to the equation, and reduces the corresponding bipartite graph by removing both nodes and all corresponding edges. The same must hold for the reduced set of equations and can be repeated until all variables are matched.

If during the algorithm all remaining equations depend on at least two unknown variables no matching without algebraic loops exists.

The proposed algorithm extends the existing symbolic initialization method of OpenModelica [2], which is capable to initialize complex hybrid models.

3.2 Numeric Fall-back Case

Instead of step 7 and 8 another possible approach would be to leave the unmatched equations out of the problem, solve it, then numerically evaluate the residuals and check if they are small enough. This might be non-trivial in some cases when the involved quantities are very large due to the choice of measurement units and to the size of the system under consideration.

This kind of approach has also the disadvantage that it is not possible to find a different matching if the system ends up in a local singularity.

4 Discussion Based on Selected Examples

The proposed algorithm has been tested using the package *OverdeterminedInitialization* from *ModelicaTest*, which has been first created for this purpose at the 80th Modelica Design Meeting. This test package contains a list of models from different domains (Electrical, Mechanics, and Fluid) for test purpose, which become over-constrained after index-reduction. The different test cases can be used to cover all kinds of the different issues, that may occur during this process.

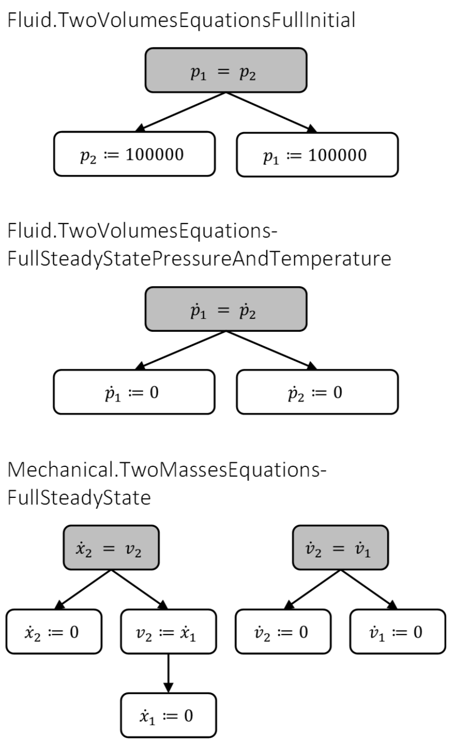


Figure 4: Directed graph of the unmatched equations of some simple examples from the package *OverdeterminedInitialization*

Because of the low complexity of the most test cases, they end up with a similar subsystem for the consistence check as the introductory example. Therefore, the consistence check can be performed by evaluating the remaining subgraphs recursively towards the unmatched equations (see Figure 4).

4.1 Fluid Model of Two Volumes

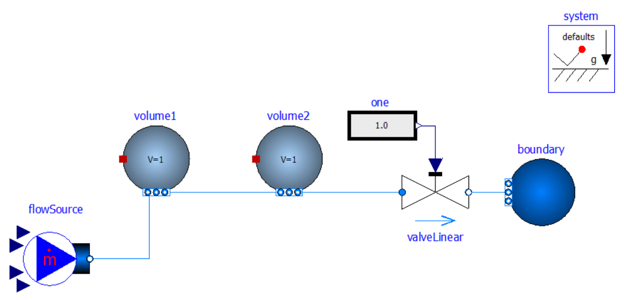


Figure 5: *TwoVolumesEquationsFullSteadyState-MassAndEnergy*

One of the most complex examples of the package is the fluid model *TwoVolumesEquationsFullSteady-StateMassAndEnergy*, in which equations and initial equations refer to stored mass and energy within each volume as differentiated variables, but pressure and temperature are forced to be states using the Modelica stateSelect attribute. Figure 5 shows a graphical representation using MSL components. Note that, due to the way the Modelica.Fluid components are designed, it is not possible to reproduce this situation, so a textual equation-based model is used for testing the algorithm. This model and the respective dependence graph are listed in the appendix. Using transformations like alias elimination, a reduced version can be generated that contains just 10 variables and 11 equations. Figure 6 shows the reduced dependence graph with one possible matching.

The example contains 11 equations, and is overconstrained due to one equation. Therefore, there are 11 sets of equations (each of cardinality one) that may be removed. Depending on the selected set of equations several cases can occur:

1. recursively evaluable systems
2. systems containing algebraic loops
3. systems with local singularities
4. system with structural singularities

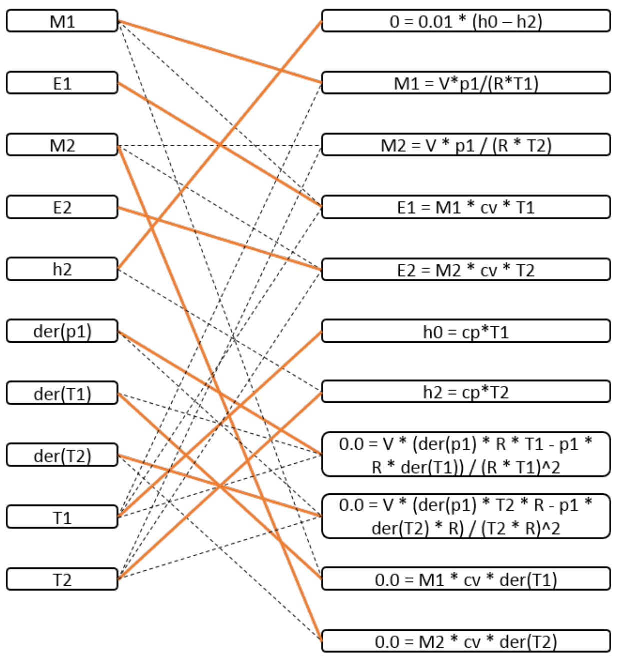


Figure 6: A matching of the remaining equation system of *TwoVolumesEquationsFullSteadyState-MassAndEnergy*

Four sets end up in case 1, which is the most desirable case. Here all equations can be solved recursively towards the unmatched ones to check the consistency of the system. This happens often in simple cases as described above. Unfortunately, there is no guarantee that such a case will always show up in more complex examples. But, if such a case exists, it will be always captured.

Three sets end up in case 2, which is the most common case for real-world problems. In general, there is no way to avoid algebraic loops. However, this case will only need to be taken into consideration if there is no recursively evaluable system available. In that case, advanced symbolic solvers are needed to compute the solution of these loops symbolically. If this is not possible a modified version of the algorithm could switch to a numeric fall-back mode.

Two sets end up in case 3, which occurs, for instance, if a subexpression is evaluated to $0/0$. In this case the selected set of unmatched equations can not be used to determine whether the system is consistent or not. The proposed algorithm detects this in step 7, and, if possible, this case is avoided by rejecting the corresponding set and trying another one, in the hope of finding a recursively evaluable one.

Two sets end up in case 4, which gives a non-valid matching for the system. It is not possible to use the selected set of unmatched equations for any conclusion. Because, the proposed algorithm only selects matchable sets of equations, this case is never reached. Furthermore, due to this fact the possible sub-sets of equations that may be reduced is much less than $\binom{n}{k}$.

4.2 Electrical 3-Phase System

This test model was already presented in [1]:

Consider the following electrical 3-phase power system of Figure 7, where two generating units VS1 and VS2 are connected via a transmission line modeled by components LR1 and LR2.

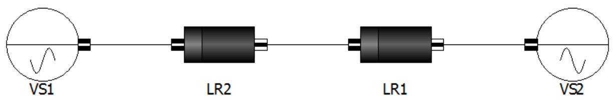


Figure 7: An electrical power system with two generating units VS1 and VS2 connected via a transmission line

The connectors are written in dq0-coordinates implementing the potential variable u_dq0 and the flow variable i_dq0. These quantities are constant in case of a non-distributed steady state, which is generally assumed during the initialization process. Introducing the Park-Transformation P the 3-phase rotating system (voltages u_abc and currents i_abc) can be calculated from the dq0-representation and vice versa.

The transmission line (LR1 and LR2) is modeled by a purely inductive and resistive component, based on the Modelica Electrical Library. Since LR1 and LR2 are connected in series, giving a higher index system, index reduction has to be applied for simulation purposes.

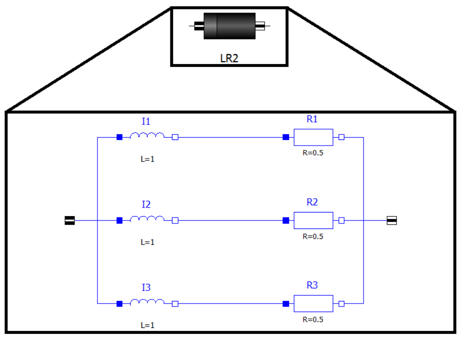


Figure 8: LR2 component with dq0-connectors

The voltage source is described similarly using the Modelica Standard Library combined with the dq0-connectors.

4.2.1 Results Using the Proposed Algorithm

This system covers some important pitfalls. After index reduction, the initial system contains three equations too many. The described approach is not able to find a matching without algebraic loops. Therefore, the resulting loop (with a size of 27) has to be solved symbolically. This is currently not supported from the OpenModelica back-end and might be in general impossible.

As fall-back case the system can be solved numerically during runtime as described in 3.2. For the concrete case of the 3-phase system, the resulting algebraic loop becomes singular if a wrong set of equations is removed. A more advanced solution is described in the following subsection.

4.2.2 Future Work

As stated above, a symbolic solution for the 3-phase system is still needed. One possible idea is to change the consistence check as follows:

Depending on the matching, the following three equations might get selected as the set of removed equations:

```
0.0 = 0.5773502691896258 * (der(LR2.I1.i) +
      der(LR2.I2.i) + der(LR2.I3.i));
```

```
0.0 = LR2.Park[2,1] * der(LR2.I1.i) +
      LR2.Park[2,2] * der(LR2.I2.i) +
      LR2.Park[2,3] * der(LR2.I3.i) +
      der(LR2.Park[2,1]) * LR1.i_abc[1] +
      der(LR2.Park[2,2]) * LR1.i_abc[2] +
      der(LR2.Park[2,3]) * LR1.i_abc[3];

0.0 = LR2.Park[1,1] * der(LR2.I1.i) +
      LR2.Park[1,2] * der(LR2.I2.i) +
      LR2.Park[1,3] * der(LR2.I3.i) +
      der(LR2.Park[1,1]) * LR1.i_abc[1] +
      der(LR2.Park[1,2]) * LR1.i_abc[2] +
      der(LR2.Park[1,3]) * LR1.i_abc[3];
```

They are quite similar to the next three equations, that are part of the matched system and involved in the algebraic loop:

```
0.0 = 0.5773502691896258 * (der(LR2.I1.i) +
      der(LR2.I2.i) + der(LR2.I3.i));

0.0 = LR1.Park[2,1] * der(LR2.I1.i) +
      LR1.Park[2,2] * der(LR2.I2.i) +
      LR1.Park[2,3] * der(LR2.I3.i) +
      der(LR1.Park[2,1]) * LR1.i_abc[1] +
      der(LR1.Park[2,2]) * LR1.i_abc[2] +
      der(LR1.Park[2,3]) * LR1.i_abc[3];

0.0 = LR1.Park[1,1] * der(LR2.I1.i) +
      LR1.Park[1,2] * der(LR2.I2.i) +
      LR1.Park[1,3] * der(LR2.I3.i) +
      der(LR1.Park[1,1]) * LR1.i_abc[1] +
      der(LR1.Park[1,2]) * LR1.i_abc[2] +
      der(LR1.Park[1,3]) * LR1.i_abc[3];
```

Instead of solving the entire algebraic loop symbolically, it might be possible to transform each of the removed equations into an equation of the matched system, and thus prove its redundancy.

The first equation of both sets are already equal, so there is no further consistence check needed. By applying common-sub-expression elimination techniques and advanced alias elimination also the other two equations can be transformed into the other two. For that, it is just needed to figure out that `LR1.Park` is alias of `LR2.Park` and `der(LR1.Park)` is alias of `der(LR2.Park)`.

5 Conclusions

This paper has discussed a symbolic algorithm that handles the initialization problem of over-determined systems. The presented approach has not to deal with numerical thresholds and there is no risk of trapping into local minima. Also hybrid models can be handled efficiently. This is a major improvement with respect to the existing numerical approach within OpenModelica.

This paper focused on symbolic techniques to determine potential redundant equations and ways to analyze whether the system is consistent or not. These work well for problems, which end up in an recursively evaluable initial system. Furthermore, the developed algorithm takes care of singularities, if they occur during the consistency check. More complex problems end up with systems including algebraic loops. If they are not solvable symbolically, a numerical fall-back solution as well as advanced symbolic techniques are proposed.

References

[1] B. Bachmann, P. Aronsson, P. Fritzson, "Robust Initialization of Differential-Algebraic Equations", In Proceedings 5th International Modelica Conference, Vienna, Austria, Sep. 4-5, 2006, pp. 607-614

[2] L. Ochel, B. Bachmann, "Initialization of Equation-Based Hybrid Models within OpenModelica", In EOOLT 2013 Proceedings, Nottingham, UK, April 19, 2013, pp. 97-103

[3] C.C. Pantelides, The Consistent Initialization of Differential-algebraic Systems. SIAM Journal on Scientific and Statistical Computing, Vol. 9, No. 2, 1988

[4] S.E. Mattsson, G. Söderlind, Index Reduction in Differential-algebraic Equations Using Dummy Derivatives. SIAM Journal on Scientific Computing, Vol. 14, No. 3, 1993

[5] R. Tarjan, Depth-first search and linear graph algorithms. SIAM Journal on Computation, Vol. 1, No. 2, 1972

[6] F.E. Cellier, E. Kofman, Continuous System Simulation, Springer, 2006

A Test Model TwoVolumesEquationsFullSteadyStateMassAndEnergy

```
model TwoVolumesEquationsFullSteadyStateMassAndEnergy
  "Two volumes containing an ideal gas with a zero dp connection"
  Real M1(stateSelect=StateSelect.avoid, start=1.0),
       M2(stateSelect=StateSelect.avoid, start=1.0),
       E1(stateSelect=StateSelect.avoid, start=1.0),
       E2(stateSelect=StateSelect.avoid, start=1.0),
       p1(stateSelect=StateSelect.prefer, start=1.0),
       p2(stateSelect=StateSelect.prefer, start=1.0),
       T1(stateSelect=StateSelect.prefer, start=1.0),
       T2(stateSelect=StateSelect.prefer, start=1.0),
       w0, w1, w2, h1, h2;
  parameter Real V = 1;
  parameter Real R = 400;
  parameter Real cp = 1000;
  parameter Real cv = cp-R;
  parameter Real h0 = cp*300;
  parameter Real Kv = 1e-7;
initial equation
  der(M1) = 0;
  der(E1) = 0;
  der(M2) = 0;
  der(E2) = 0;
equation
  der(M1) = w0 - w1;
  der(E1) = w0*h0 - w1*h1;
  der(M2) = w1 - w2;
  der(E2) = w1*h1 - w2*h2;
  M1 = V*p1/(R*T1);
  M2 = V*p2/(R*T2);
  E1 = M1*cv*T1;
  E2 = M2*cv*T2;
  h1 = cp*T1;
  h2 = cp*T2;
  w0 = 0.01;
  w2 = Kv*p2;
  p1 = p2;
end TwoVolumesEquationsFullSteadyStateMassAndEnergy;
```

Listing 2: Test model *TwoVolumesEquationsFullSteadyStateMassAndEnergy*

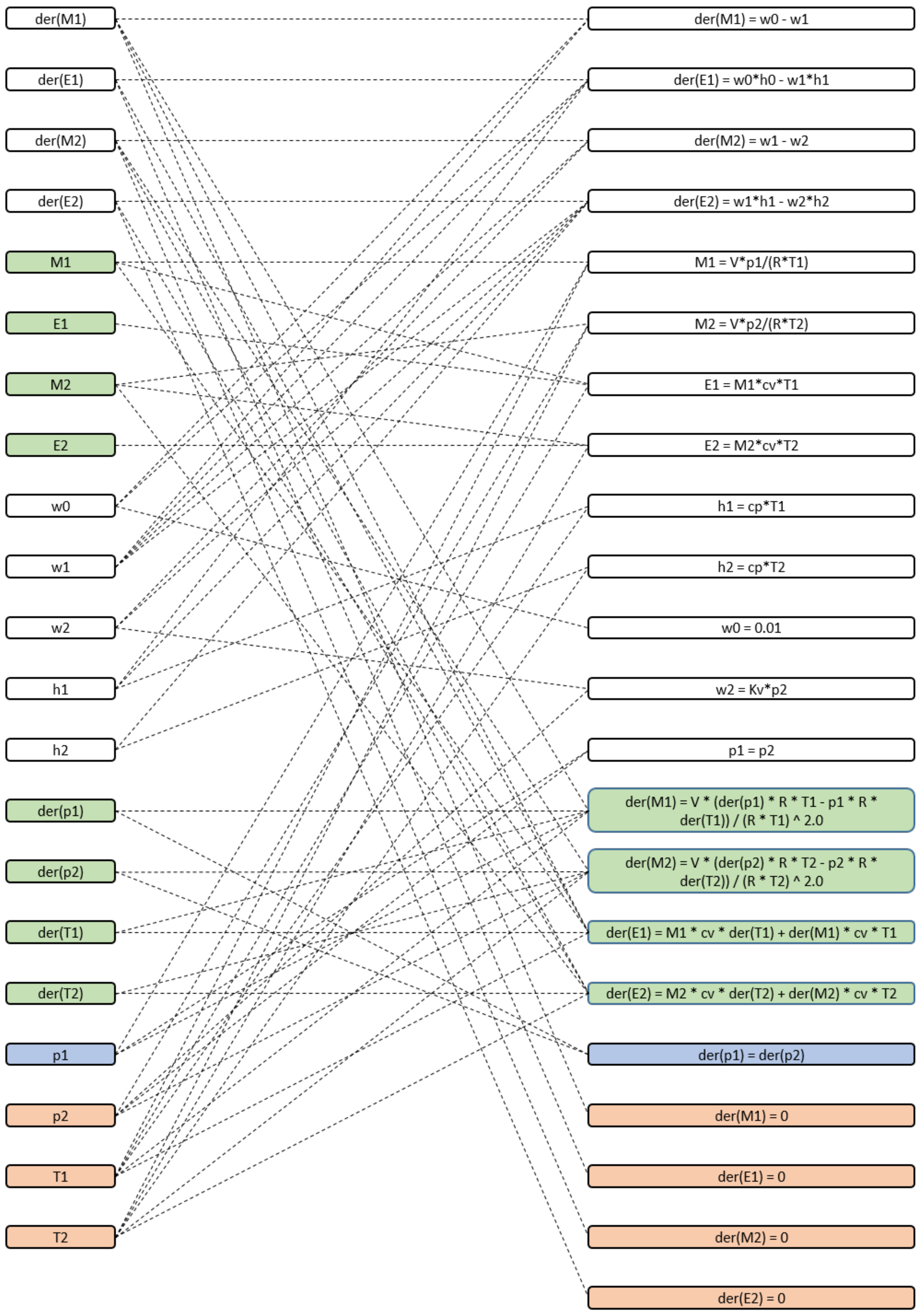


Figure 9: *TwoVolumesEquationsFullSteadyStateMassAndEnergy* - dependence graph