On the Formulation of Steady-State Initialization Problems in Object-Oriented Models of Closed Thermo-Hydraulic Systems

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

Abstract

The object-oriented formulation of steady-state initialization for models of closed thermo-hydraulic systems yields singular problems, due to system-wide structural issues. The paper proposes how to solve this problem in an object-oriented fashion, by means of an additional component that helps to uniquely determine the initial conditions of the system. A method based on the analysis of the null space of the Jacobian of the initialization problem and on suitable annotations is also proposed to provide the end user with meaningful, high-level, context-relevant diagnostic messages, in case the singular problem arises. This diagnostic method can also be applied to other cases, such as closed systems with constant density fluid and electrical circuits lacking a ground connection.

Keywords: Thermo-hydraulic system modelling, Initialization, User-friendly error diagnostics

1 Introduction

Dynamic modelling of thermo-hydraulic systems in Modelica is becoming increasingly popular in many application fields: energy conversion systems, air conditioning and ventilation plants for civil and airborne applications, etc.

The steady-state initialization of such models is well-known to be a critical issue. The two typical alternatives available to practitioners are:

- set the initial values of all the state variables to estimates of the steady-state one (the exact value is not known a priori), then simulate a relaxation transient until a steady-state is reached;
- set the initial derivatives of all the state variable to zero and let the tool numerically solve for the exact steady-state values.

The first choice has the advantage that finding a consistent initial state is numerically easy, so the simulation always starts, but the ensuing transient can be problematic, because of potentially large swings of flow variables that can require a very long simulation time and possibly lead to solver failures.

The second choice is more appealing: if the solver converges, the time spent to solve the steady-state initialization problem is typically much lower than the time spent simulating the (meaningless) relaxation transient. However, there are three categories of issues that can prevent getting the desired initial state:

1. the initialization problem is well-posed, but a solution is not found because of convergence failure of the iterative solver (typically, the initial guess values are not close enough to the solution);
2. the initialization problem is structurally well-posed, but the values of some parameters are such that a physically valid solution does not exist;
3. the initialization problem is not structurally well-posed, e.g., it is singular.

Recently proposed improvements [7, 4] based on simplified models and homotopy transformations, have been proved effective to enhance the likelyhood of convergence (issue 1.) and also to point out problems stemming from the wrong parametrization of the model (issue 2.). However, they do not address issue 3. at all.

If the thermo-hydraulic system under consideration is closed, i.e., it does not exchange mass with the outside world, initialization problems where all components are set up for steady-state initialization turn out to be singular. The singularity arises at the system level, so it doesn't take place if parts of the closed system are first tested separately, with suitable boundary conditions that make the system an open one, but only

when the final closed system is assembled. This behaviour can be puzzling for inexperienced users.

The goal of this paper is to propose an elegant, object-oriented way to completely determine the initial conditions for closed systems, as well as a method based on numerical analysis and suitable annotations to issue meaningful diagnostics when such system-level singularities do arise, guiding the end-user towards the solution of the problem.

The paper is structured as follows: in Section 2, the structure of steady-state initialization problems in closed circuits is analysed; in Section 3, an object-oriented solution to the problem is proposed and then applied in Section 4 to two case studies. Section 5 discusses how a tool can report the singularity to the end user in a meaningful way, while Section 6 ends the paper with concluding remarks.

2 Singular initialization problems of closed systems

Consider a generic closed thermo-hydraulic system. Dynamic models of such a system contain mass balance equations of three kinds. The first corresponds to mass balances inside control volumes where the stored mass can change over time:

$$\frac{dM_i}{dt} = \sum_j w_{i,j}, \quad i = 1,\ldots,N_d, \tag{1}$$

where M_i is the mass contained within the i-th volume, $w_{i,j}$ are the flow variables of its fluid ports, i.e, all the mass flow rates crossing the component boundary through the ports assumed positive when entering, and N_d is the number of control volumes having such dynamic balances. The second kind corresponds to mass balances inside components where the change over time of stored mass is neglected:

$$0 = \sum_j w_{i,j}, \quad i = N_d+1,\ldots,N_d+N_s, \tag{2}$$

where $w_{i,j}$ has the same meaning as above, and N_s is the number of control volumes with such static balances. The third kind corresponds to the mass balance equations formulated over infinitesimally small control volumes spanning each connection set of fluid ports, which are automatically generated by the compiler:

$$0 = \sum_m w_{k,m}, \quad k = 1,\ldots N_c \tag{3}$$

where N_c is the number of connection sets and, for the k-th connection set, $w_{k,m}$ are the flow variables of all the ports belonging to it.

It is essential to note that both in (1) and (2), *all* mass flow rates appearing in the mass balance equations correspond to flow through the ports. Thus, source and sink components, which represent flows exchanged between the system and the outer world, are explicitly excluded. The system is closed, as it cannot exchange mass with the outer environment, but only with other components belonging to it.

Assume now one wants to initialize the system in steady state. In an object-oriented formulation of the system model, which is formed by connecting component models through their ports, one typically selects a steady state initialization option for each component, possibly via some system-level default option which is passed to all the components via inner/outer constructs. This approach is followed by the `Modelica.Fluid` library [6]. This option adds an steady-state initial equation for each control volume with dynamic mass balance:

$$\frac{dM_i}{dt} = 0, \quad i = 1,\ldots,N_d. \tag{4}$$

The system of equations (1)-(4) is always singular, because its equations are not linearly independent.

Proof: the sum of all dynamic balance equations (1) minus the sum of all initial equations (4) plus the sum of all static balance equations (2) yields

$$0 = \sum_{i=1\ldots N_s+N_d,j} w_{i,j}. \tag{5}$$

On the other hand, the sum of all connection equations (3) yields

$$0 = \sum_{k=1\ldots N_c,m} w_{k,m}. \tag{6}$$

Now, each flow variable in the right-hand-side of (5) belongs to one and only one connection set (for unconnected ports, default connection sets are automatically generated). Therefore each variable appearing in (5) appears once and only once in (6). It follows that the difference between equation (5) and (6), which is a non-trival linear combination of (1)-(4), gives

$$0 = 0. \tag{7}$$

Hence, equations (1)-(4) are not linearly independent and thus the system is singular, q.e.d.

The physical interpretation of this singularity is that the initial conditions (4) do not give any information on the total amount of mass contained in the system at initialization.

DOI 10.3384/ecp12076215

3 Object-oriented formulation of well-posed initialization problems

3.1 Mathematical formulation

One way to make the initialization problem non-singular is to avoid the initial equation (4) for one of the control volumes in the circuit, substituting it with some other equation that makes the problem well posed. This is convenient for those systems where one component has the specific purpose of enforcing the pressure at some point of the circuit.

For example, closed pressurized circuits containing liquid water usually feature dedicated components which determine the pressure level of the system and accommodate the thermal expansion of the fluid: accumulators in domestic heating systems, pressurizers in cooling circuits for PWR nuclear reactors, etc. In this case, the initial conditions for those components are not given as (4), but rather by specifying the initial value of the pressure (and possibly of other variables, depending on the actual level of detail of the model).

Unfortunately, this approach is not always convenient for two reasons. One is that some closed systems that use a compressible fluid as working medium may not contain such a specialized component. The other one, which is more fundamental, is that the extra initial condition which is needed to make the initialization well-posed might refer to the entire system and not just to a specific component. For example, it is often the case that refrigeration circuits must be initialized so that the *total mass* of the fluid contained in the circuit, also known as the *charge*, has a certain value.

In this paper, a modular approach is proposed to solve this issue. The idea is to add an extra component to the system model which contains the equation

$$0 = w_{N_d+N_s+1,1} + w_b, \tag{8}$$

where $w_{N_d+N_s+1,1}$ is the flow variable of the only port of this additional component and w_b is an *unknown* parameter, as well as an extra *initial equation* to completely determine the initial state, such as, e.g.

$$p = p_{start} \tag{9}$$

(where p is the port pressure and p_{start} a known parameter), or, e.g.,

$$\sum_i M_i = M_{start} \tag{10}$$

where M_{start} is again a known parameter.

Including this extra component to the system adds two more equations, e.g., (8),(9) or (8),(10), and two more unknowns, $w_{N_d+N_s+1,1}$ and w_b, to the initialization problem, which thus remains balanced.

If (8) is added to the set of static mass balance equations (2), equation (5) will change to

$$0 = w_b + \sum_{i=1\ldots N_s+N_d+1,j} w_{i,j}, \tag{11}$$

and the difference between (11) and (6) will now yield

$$0 = w_b \tag{12}$$

as the term w_b is not a flow through a port, so it is not cancelled out. Therefore, the mass balance equations and initial equations are no longer linearly dependent, and the value of the flow rate w_b potentially entering the system will be computed to be zero during initialization, so the additional component will have no effect on the model during simulation.

It is not trivial to prove that the initialization problem will be non-singular in general. The user building the model should select the extra initial condition (e.g., (9) or (10)) so that the corresponding system of equations uniquely determine the initial state of the system, based on his understanding and expertise.

If a compressible fluid model is employed, typically (9) is a good choice, leading to a numerical problem which is easier to solve than the one obtained by adding (10).

3.2 Modelica formulation

The Modelica code of the initialization component is now presented, based on the mathematical formulation laid out in the previous sub-section, in order to be compatible with the `Modelica.Fluid` library.

```
model ClosedSystemInitializer
  replaceable package Medium =
      Modelica.Media.Interfaces.PartialMedium;
  parameter Medium.AbsolutePressure p_start;
  final parameter Medium.MassFlowRate
      w_b(fixed = false) = 0;
  Modelica.Fluid.Interfaces.FluidPort_a
      port(redeclare package Medium = Medium,
           m_flow(min = 0),
           p(start = p_start));
  Modelica.Blocks.Interfaces.RealInput
      initialConditionResidual;
equation
  0 = port.m_flow + w_b;  // Mass balance
  port.h_outflow  = 0;
  port.Xi_outflow = zeros(Medium.nXi);
initial equation
  0 = initialConditionResidual;
end ClosedSystemInitializer;
```

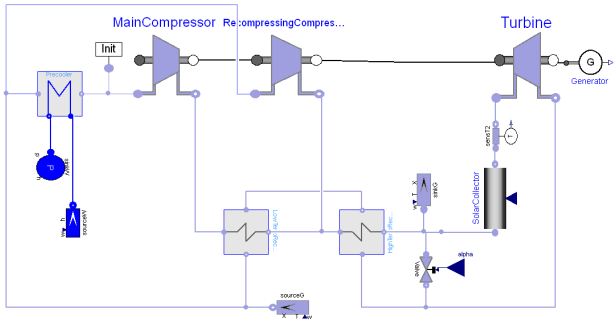

Figure 1: Brayton cycle plant model.

When instantiating the model, the extra initial condition can be specified by connecting a RealExpression block containing its residual to the component's input port.

Note that the `min` annotation on the fluid port specifies that the fluid will never flow out of it, so the values of the stream variables `h_outflow` and `Xi_outflow` (which must be given in order to get a balanced model) are never actually used outside the component itself, when computing the incoming stream quantity via the `inStream()` operator [6]. Therefore, once the initial solution of the problem has been found, in particular `w_b = 0`, this additional component has no influence at all on the remaining equations of the closed system.

In order to improve the convergence of the solver when (10) is used as initial condition, it might be possible to use a homotopy transformation, where the simplified initial equation is (9). The initial equation section is correspondingly changed to:

```
initial equation
  0 = homotopy(
        actual = initialConditionResidual,
        simplified = port.p - p_start);
```

4 Example applications

4.1 Brayton cycle for power generation

Advanced energy conversion cycles are being considered for high-temperature heat sources, such as central receiver solar plants and IV generation nuclear plant, using supercritical CO_2 as a working medium and a closed Brayton cycle configuration [1, 5], featuring two separate compressors to achieve optimal efficiency of the overall cycle. The object diagram of the plant model is shown in Fig. 1. In order to control the pressure levels in the system, in particular at the main compressor inlet, it is possible to add or remove mass from two points of the circuit, through appropriate sub-systems which are modelled here as ideally controlled flow rate sources or sinks. When the plant model is considered together with the pressure controller model in a closed-loop configuration, then a full steady-state formulation of the initialization problem, as in (4), is well-posed, because the pressure level (and thus the mass) of the fluid contained in the circuit is determined implicitly by the controller to be equal to the set point.

During the control system design phase, though, it is usually necessary to analyse the dynamic behaviour in open loop around equilibrium points, which in this case means there is no pressure controller and both flow rates are set to zero, so that the system is effectively closed.

In this case, a convenient way to make the initialization problem well-posed is to connect the `ClosedSystemInitializer` component at the compressor inlet, so that it sets the initial pressure at that point. This condition, together with the flow characteristics of the turbo-machines and with the thermal interaction with the heat source and sink, uniquely determines the amount of gas contained in the system and the pressure and temperature distribution.

The model was set up using the ThermoPowerLight library, which is a simplified version of the ThermoPower library [2, 3] currently under development for optimization studies, and successfully solved using Dymola 2013.

4.2 Refrigeration cycle

Refrigeration systems are usually carefully sealed, in order to avoid leaks of refrigerant fluid to the environment, so they always qualify as closed system. The static and dynamic behaviour of the system is heavily influenced by the amount of refrigerant which is initially loaded in the system (the *charge*).

A simple model of a refrigeration circuit, built with the ThermoPower library, is shown in Fig. 2. Homogeneous flow and constant heat transfer coefficients are assumed for simplicity; pressure losses along the condenser and evaporator are lumped at the pipe ends.

The initialization problem has been set up by adding an extra condition specifying the total mass contained in the condenser and evaporator pipes. In order to avoid convergence problem, homotopy-based initialization is performed, starting from a simplified model where the condenser inlet pressure is fixed. The model was successfully initialized in steady-state using Dymola 2013.

DOI 10.3384/ecp12076215

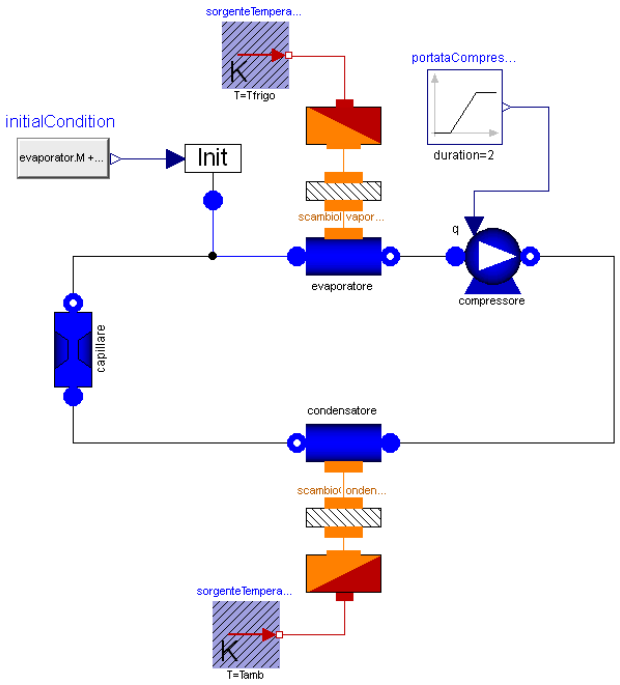

Figure 2: Refrigeration cycle plant model.

5 Diagnostics of ill-posed initialization problems

This section addresses the problem of giving end-users meaningful feedback in case they do not specify the initialization problem correctly and fall into the singular case presented in Sect. 2.

The situation with currently available Modelica tools is not satisfactory. The complete steady-state initialization problem is usually large (hundreds or thousands of unknowns, depending on the degree of detail of the model) and strongly non-linear. The presence of linearly dependent equations makes its Jacobian singular. As a consequence, during iterations the unknowns, which typically include pressures, temperatures and specific enthalpies of control volumes in the model, will fluctuate wildly, causing repeated out-of-bounds errors from the fluid property calculation functions, until the solver eventually gives up. The error messages typically shown to the end user will point to these out-of-bounds errors and, possibly, to the very high condition number of the Jacobian. By no means this diagnostic information points explicitly to the actual root cause, which has been shown in Sect. 2.

One possible solution to this problem is sketched in the remainder of this Section, based on numerical analysis and suitable annotations in the model library.

5.1 Numerical identification of singular subsystems

Let the initialization problem be formulated in residual form (as it is usually the case if nonlinear solvers are to be employed):

$$F(x,\dot{x},v,p) = 0 \tag{13}$$

where $x \in \Re^n$ is the vector of state variables, $v \in \Re^m$ is the vector of algebraic variables, and $p \in \Re^q$ is the vector of unknown parameters (having the attribute `fixed =false`). The analysis of under- and over-constrained initialization problems, though extremely interesting, is outside the scope of this paper, so it is assumed here that the initialization problem is square, i.e., the function returning the residuals of the dynamic and initial equations of the system is $F : \Re^n \times \Re^n \times \Re^m \times \Re^q \to \Re^{2n+m+q}$. Define the vectors of unknowns

$$z = \begin{bmatrix} x \\ \dot{x} \\ v \\ p \end{bmatrix}, \tag{14}$$

$z \in \Re^{2n+m+q}$, and let F_z be the Jacobian matrix of the function F with respect to z. The ill-posed initialization problem described in Section 2 leads to a singular Jacobian, since it is shown there that there exist a non-zero vector v such that

$$F_z v = 0 \tag{15}$$

having non-zero entries (plus/minus one) only in correspondence to the mass balance equations and initial equations. If many disconnected closed systems exist in the model, then there will be a corresponding number of linearly independent vectors v_j that satisfy eq. (15). Note that there might also be other such vectors because of other parts of the problem being singular on there own. The set of linearly independent v_j's satisfying eq. (15) spans the so-called nullspace or kernel of F_z.

It is now possible to identify the set(s) of linearly dependent equations in the initialization problem by numerically computing the nullspace of the Jacobian F_z, i.e. a set of orthonormal vectors v_j that forms a basis for the nullspace. For each of these vectors, all entries whose absolute value is greater than a suitable small threshold identify the equations in the initial problem that are part of a singular subsystem, which can be reported to the end-user.

The selection of the threshold might be critical, because due to numerical approximations, the zero entries will actually have a small non-zero value, and it might not be trivial to avoid false positives or false negatives. It is then essential to use a state-of-the-art numerically robust algorithm to compute the orthonormal basis of the nullspace of F_z, with the lowest possible effect effect of numerical rounding errors on the result.

Since the sum of square of the elements of each basis vector is one, it is expected that there will be a sharp difference between the elements that correspond to the involved equations (which will have order of magnitude of one) and the other ones (which will have order of magnitude of machine ε, around 10^{-16} for double precision arithmetic). Some experimentation in real-life-sized test cases is however necessary to fine tune such a method, but this has not been possible yet for the lack of available implementation of the method in Modelica compilers.

Another issue to be taken care of is the proper handling of cases when $N > 1$ disconnected closed subsystems are present in the model. In this case, the basis of the nullspace will contain N orthonormal vectors, but there is no guarantee that the non-zero entries of one of them will only refer to one sub-system. The reason of this fact is that the orthonormal basis of the nullspace is not unique, as other perfectly valid bases can be found by taking one and multiplying its vectors by an orthogonal matrix, i.e., obtaining a basis which is rotated with respect to the previous one. It is then possible that the basis vectors which are obtained from the SVD decomposition are linear combinations of the ones that each refer to one singular sub-system.

However, once that these basis vectors $v_1, \ldots, v_N$ have been found for the null space, obtaining vectors whose non-zero entries point to one sub-system is fairly straightforward, by means of a pivoting algorithm. The idea is to look for linear combinations of the originally found vectors that have the least possible number of non-zero entries. A possible sketch of the algorithm is:

1. Build the matrix $M = [m_{i,j}] = [v_1\ v_2\ \ldots\ v_N]$.
2. Look for the rows with more than one element having absolute value greater than the threshold; if none are found, stop.
3. If at least row one is found, select among them the elements $m_{i,j}$ and $m_{i,k}$ which have the largest absolute value, such that $|m_{i,j}| \geq |m_{i,k}|$.
4. Subtract column j multiplied by $m_{i,k}/m_{i,j}$ from column k.
5. Go to step 2.

When the algorithm terminates, the elements above threshold of each column of M correspond to the singular equations of just one sub-system.

Last, but not least, it is essential to point out that the proposed analysis must be performed on the *full initialization problem* (13), prior to any optimization such as alias elimination, symbolic simplification, and BLT partitioning of the problem. Such optimizations make it easier and more efficient to solve the problem from a numerical point of view, but effectively destroy the structural information that is required to issue meaningful diagnostic errors to the end user.

For example, equations (4) could be used to statically determine that the initial values of the mass derivatives are zero, so these equations could be eliminated from the set of initial equations and all instances of $\frac{dM_i}{dt}$ could be replaced by zeros in equation (1). Obviously, this makes it impossible to point out to the root cause of the problem, which lies in the equations (4). It is of course possible to first try solving the problem with all optimizations active, and only if that fails, generate the Jacobian F_z of the full problem and evaluate it using the values of the unknowns thus found.

As a final remark, although these methods can be fairly expensive in terms of CPU load, the computation is only performed once after the initialization has failed, and a waiting time of a few seconds (or even of a minute or two) is largely preferrable to quickly getting diagnostic output that gives no meaningful information in order to trace the root cause of the problem.

5.2 High-level error diagnostics for the user

It is possible to infuse in the model additional expert knowledge from the modeller, which can further help the end user to identify the root cause of the singularity. In the case under consideration, thanks to the analysis carried out in Section 2, the expert library developer knows that a bad choice of steady-state initial equations will lead to a singular problem, in which those equations will form a singular sub-system. It would then be possible to annotate those equations with meaningful error messages, that will be reported to the end-user in case they are found to be part of such a singular sub-system, e.g.:

```
initial equation
  der(M) = 0
```


DOI 10.3384/ecp12076215

```
annotation(PartOfSingularSystemError =
  "Ill-posed initial conditions for closed
   system.\n  Please connect a
   ClosedSystemInitializer component to
   the system to completely specify the
   initial state");
```

Instead of just reporting the raw set of equations which form a singular subsystem, the tool could report their associated error annotation strings, that would help even an inexperienced user to fix the problem quickly. A hot link to the documentation of the `ClosedSystemInitializer`, e.g. marked-up using a `modelica://` URL, could lead with one click to a documentation page that explains the problem in more detail, and possibly even link to this paper on the Web for further information. Note that annotations on equations are allowed by the Modelica language grammar, though they have never been used so far for any definite purpose.

5.3 Application to other cases

The mechanism proposed in this Section could also be used for diagnostic purposes in other situations that suffer of the same syndrome, namely, a sub-set of the system equations being singular due to the system-wide structural issues.

A first example is models of thermo-hydraulic systems with constant density fluid, which is a modelling assumptions sometimes used if the working medium is liquid and one is not interested in the very fast pressure dynamics, nor in the thermal expansion and buoyancy effects. In this case, the sets of equations (1) and (4) are empty, and all the mass balance equations of the control volumes that make up the system are contained in set (2). The linear combination of equations mentioned in Section 2 still yields $0 = 0$, so the system is singular. In this case, however, the singularity does not involve the initial equations, so both the initialization problem and the simulation problem are singular. An annotation could then be applied to the static mass balances of the components using incompressible fluid, e.g. with error message "Closed system with constant density fluid yields singular system of equations. Please add a component such as pressurizer or accumulator to determine the pressure in the circuit uniquely".

Another example is the well-known case of electrical circuits with missing ground component. In this case, the system of dynamic equations is singular because the pin voltages are not uniquely determined. Various solutions have been proposed, based on structural analysis, to issue meaningful diagnostic messages to the end user. Using the numerical method proposed in this paper, it would suffice to add the `PartOfSingularSystemError` annotation to the equation `v = p.v - n.v` contained in the `OnePort` partial model, e.g. with error message "Electrical circuit without ground connection yields singular system of equations. Please connect a ground component to the circuit where appropriate".

6 Conclusions

Models of closed thermo-hydraulic circuits can lead to singular steady-state initialization problems due their system-wide structure. This paper proposes a way to solve this problem in an object-oriented fashion, by means of an additional component that helps to uniquely determine the initial conditions of the system.

A method based on the analysis of the null space of the Jacobian of the initialization problem is also proposed to provide the end user with meaningful, high-level, context-relevant diagnostics, by suitably annotating those equations that might potentially lead to such singular problems. This allows the library designer to infuse expert knowledge about potential system-level issues, helping inexperienced end-users to pin-point the root cause of the problem easily, contrary to the current state-of-the-art with existing tools, that will output hard-to-understand low-level numerical diagnostic messages.

The proposed method for high-level singular system diagnostics can also be applied to other cases, such as closed circuits with constant-density fluid, or electrical circuits lacking a ground connection.

References

[1] F. Casella and P. Colonna. Development of a Modelica dynamic model of solar supercritical CO2 Brayton cycle power plants for control studies. In *Proceedings of the Supercritical CO2 Power Cycle Symposium*, pages 1–7, Boulder, Colorado, USA, May 24–25 2011.

[2] F. Casella and A. Leva. Modelica open library for power plant simulation: Design and experimental validation. In P. Fritzson, editor, *Proceedings 3rd International Modelica Conference*, pages 41–50, Linköping, Sweden, Nov. 3–4 2003. Modelica Association. http://www.elet.polimi.it/upload/casella/thermopower/.

[3] F. Casella and A. Leva. Modelling of thermo-hydraulic power generation processes using Modelica. *Mathematical and Computer Modeling of Dynamical Systems*, 12(1):19–33, Feb. 2006.

[4] F. Casella, M. Sielemann, and L. Savoldelli. Steady-state initialization of object-oriented thermo-fluid models by homotopy methods. In C. Clauss, editor, *Proceedings 8th International Modelica Conference*, pages 86–96, Dresden, Germany, Mar. 20–22 2011. Modelica Association.

[5] V. Dostal. *A Supercritical Carbon Dioxide Cycle for Next Generation Nuclear Reactors*. PhD thesis, MIT, 2004.

[6] R. Franke, F. Casella, M. Sielemann, K. Proelss, M. Otter, and M. Wetter. Standardization of thermo-fluid modeling in Modelica.Fluid. In F. Casella, editor, *Proceedings 7th International Modelica Conference*, pages 122–131, Como, Italy, Sep. 20–22 2009. The Modelica Association.

[7] M. Sielemann, F. Casella, M. Otter, C. Clauß, J. Eborn, S. E. Mattsson, and H. Olsson. Robust initialization of differential-algebraic equations using homotopy. In C. Clauss, editor, *Proceedings 8th International Modelica Conference*, pages 75–85, Dresden, Germany, Mar. 20–22 2011. Modelica Association.

 DOI 10.3384/ecp12076215