Steady-state initialization of object-oriented thermo-fluid models by homotopy methods

Francesco Casella* Michael Sielemann† Luca Savoldelli*
* Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci, 32, 20133 Milano, Italy
† Deutsches Zentrum für Luft- und Raumfahrt, Institute of Robotics and Mechatronics
Münchner Strasse 20, 82234 Wessling, Germany

Abstract

The steady-state initialization of large object-oriented thermo-hydraulic networks is a difficult problem, because of the sensitivity of the convergence to the initial guesses of the iteration variables. This paper proposes an approach to this problem based on homotopy transformation, detailing specific criteria for model simplifications in this physical domain. The approach is successfully demonstrated on large power plant test cases, having several hundreds of iteration variables.

Keywords: Thermo-hydraulic systems, power plants, steady-state initialization problems.

1 Introduction

Steady state initialization of large thermo-fluid network is hard and often fails, even when using state-of-the-art nonlinear solvers. This hampers the use of object-oriented models in applications such as power plant simulation, because of the difficulties encountered in getting a newly built model to actually simulate.

Currently, the only way to solve this problem is to manually set good initial guesses for all the iteration variables of the problem. This is rather inconvenient, since the number of such variables can easily grow beyond a hundred or even a thousand, and also because any tiny change to the model, or to the version of the Modelica tool used to simulate it, can lead to a different set of iteration variables and thus require a further setting of intial guesses. This makes the initialization activity tedious and very far from the concepts of modularity and object-orientation.

This paper presents an alternative approach to the problem, based on homotopy transformation. The proposed strategy is demonstrated by means of a prototype solver code on large-scale power plant test cases.

The paper is structured as follows: Section 2 gives the basic of homotopy-based initialization of object-oriented models and presents the test implementation of the solver. Section 3 introduces criteria for the formulation of simplified models in the domain of thermo-hydraulic networks. Section 4 illustrates experimental results obtained large-scale models of combined-cycle power plants, while Section 5 gives concluding remarks.

2 Homotopy-based initialization of object-oriented models

2.1 Problem definition

To encode initialization problems in Modelica, language constructs such as initial equation sections are defined. They introduce additional constraints, which, together with all equations and algorithms that are utilized during simulation, constitute the initialization problem. The solution can then be used to assign all variables, derivatives and pre-variables consistent values.

Formally, the resulting problem is an initial value problem for a system of differential algebraic equations (DAE), $0 = F(\dot{x}, x, w, t)$. Variables x are the state variables, w are the algebraic unknowns, and t is time. The initialisation problem prescribed by the model introduces conditions such as the steady-state condition $\dot{x} = 0$ at some time $t = t_0$. The differential algebraic equation system is usually index reduced, i.e. it has index 1, which means that the following expression be regular

$$\begin{bmatrix} \frac{\partial F}{\partial \dot{x}} & \frac{\partial F}{\partial w} \end{bmatrix}.$$

Formally, this problem usually results in a nonlinear system of algebraic equations that has to be

solved numerically. Unfortunately, this does not always work robustly for industrial problems as the frequently utilised gradient-based local algorithms such as damped Newton Method offer local convergence properties only (even when using so-called globalizations such as trust regions).

Several alternative methods are discussed in literature to solve the present problem more robustly. Homotopy continuation is one of them and it is considered in this article to address the need for more robust initialisation.

2.2 Established homotopy methods

Informally, using homotopy to solve nonlinear algebraic equation systems can be defined as follows. First, one starts with a simple problem whose solution is known or easy to obtain and then continuously deforms this simple problem to the difficult problem of interest. Conceptually, this appears to be simple. However, several details of these methods have to be taken into account. In particular, the existence of the homotopy path between the start and a solution, finite length, and nonexistence of singularities along the track are not guaranteed.

In order to construct a homotopy, one needs the system of residual equations of interest, $F(x)$, and another one that is easy to solve $\tilde{F}(x)$. Here and in the remainder of this section, a generic vector of unknowns x is indicating, including the state derivatives, states, and algebraic unknowns. The two sets of residual equations are then deformed from one to the other via a homotopy or continuation parameter λ. A simple example of such a deformation is a linear convex combination. In any case, the homotopy is then a system of equations with one higher dimension and denoted by

$$\rho(x,\lambda) = 0.$$

The homotopy parameter is typically restricted to some range, e.g. $[0,1]$, such that $\rho(x,0) = \tilde{F}(x) = 0$ is solved easily and $\rho(x,1) = F(x) = 0$ is the system of interest.

Many general-purpose homotopies are defined in literature. For example, the Newton homotopy [3] is defined as:

$$\rho(x,\lambda) = \lambda F(x) + (1-\lambda)(x-x_0), \qquad (1)$$

where x_0 is a tentative estimate for the solution of $F(x) = 0$. Other similar methods exist, such as the fixed point homotopy and the affine homotopy. All such methods exhibit convergence failure modes, as shown in Fig. 1, which render them not sufficiently robust to alleviate the convergence issues described in the introduction. Examples of these convergence failure modes are infinite loops without reaching $\lambda = 1$ (isolae), which occur for the Newton homotopy, and components of the solution vector wandering off toward $\pm\infty$, as observed for the fixed point and affine homotopies. Furthermore, bifurcations may arise along the continuation paths, which are non-trivial to handle numerically for industrial problems.

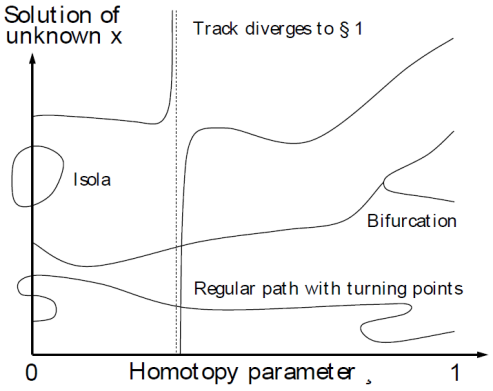

Figure 1: Problematic homotopy paths

Additional reasons why established homotopy methods are not considered a feasible solution to alleviate the need for a more robust initialisation method are given in [1].

2.3 Problem-specific homotopy

In the established homotopies mentioned in the previous section, two rather unrelated systems of equations are continuously deformed into each other; the radical difference between the two systems of equation is arguably the cause of the singular homotopy pahts.

Therefore, we propose to introduce problem-specific homotopies, where the simplified system is derived from the actual system of interest and close enough to it so as to avoid that the homotopy to the actual problem of interest be free of singularities. The formulation of the simplified systems is introduced by domain experts, allowing them to infuse their knowledge about the physics of the problem into the way the equation system is solved. The approach is fully compatible with object-orientation and declarative modeling.

2.4 Test implementation

In order to validate the methodology a test implementation was developed. It was based on the equation-

based object-oriented modelling language Modelica® and the compiler Dymola® in versions 7.3 and 6.1.

Using this test implementation, homotopies such as

$$\rho(x,\lambda) = \lambda F(x) + (1-\lambda)\tilde{F}(x)$$

can be formulated in a declarative way, where $F(x)$ is the actual problem and $\tilde{F}(x)$ is the simplified one. For this purpose,a function `homotopy()` was introduced, that takes two input arguments, namely `actual`, the expression describing the actual problem, and `simplified`, the expression corresponding to the simple problem. The Modelica compiler then expanded this function according to the above-described homotopy. For example, the expression

```
homotopy(actual=a*b, simplified=c/d)
```

was expanded to

$$\lambda(a \cdot b) + (1-\lambda)(c/d).$$

This idea has later on been included in version 3.2 of the Modelica language specification, where a built-in `homotopy()` operator with the same semantics has been introduced.

From the numerical side, the test implementation utilised the LOCA continuation algorithms of Trilinos [2]. A list of the main features of the test implementation is given here.

First of all, the algorithm provided three options for the implementation of the `homotopy()` function. In order to numerically solve the simplified problem as easily as possible, a version of the function that returned the `simplified` argument was inlined, in order to obtain the maximum structural simplification of the corresponding system of equations. For the homotopy transformation, it was expanded to the given homotopy expression. For the dynamic simulation of the system, after initialization, an inlined version returning the `actual` argument was used.

The user was able to manually prescribe whether to use homotopy initialisation or not. This is an important feature for library development and debugging, and may be useful for end users, too (e.g., if a local gradient based solver converges to a mathematically valid, but physically unreasonable solution or when a local gradient based solver does not converge and a user does not want to wait at the start of each simulation until the software realised this).

The user was able to specify that the simplified problem only should be solved. This feature is essential for library development, when one must analyze the properties of the simplified model, to understand whether it is good enough to provide a robust numerical initialization to the homotopy transformation, or if it is necessary to proceed further in the simplification process.

Verbose information on the homotopy was optionally provided, which was useful for library development and debugging, and the homotopy traces of all the iteration variables of the nonlinear system of equations were recorded for later visualisation and analysis.

Last, but not least, the underlying solver was able to follow homotopy traces with turning points, should they arise during the transformation.

3 Homotopy-based initialization of thermo-fluid network models

3.1 Basic principles

The basic idea is to formulate a simplified model which is easier to solve without the need of accurate start values, but which is on the other hand close enough to the actual problem to avoid singularities during the homotopy transformation. Three goals must be pursued:

1. The simplified model should approximate the actual model around the nominal operating point of the plant, in order to have a solution which is close to that operating point, and thus physically meaningful.

2. The simplified model should be close enough to the actual model that the homotopy transformation from the simplified to the actual problem gives rise to smooth transformations of all the iteration variables, with no singularites, no bifurcations, and possibly no turning points, even though the latter ones can be handled by continuation algorithms such as LOCA.

3. The numerical solution of the simplified model should converge with rough (default, or nominal-parameter based) initial guess values, either set by default or based on parameters specifying the nominal operating point. This avoids the need of manually setting start values for the iteration variables of the specific system at hand, whose set is difficult or impossible to determine a-priori by the end user, as it is usually the result of sophisticated (and often proprietary) tearing algorithms.

3.2 Formulating the simplified model

The simplified model should be initialized at steady state as the actual problem, in order to avoid unphysical situations, so the initial equations (der(x) = 0) are not changed. The general guideline is to approximate a few model equations so that the implicit system of equations corresponding to the steady-state initialization problem has the minimum number of unknowns and is as linear as possible. Noting that a linear problem can be solved by the standard Newton algorithm exactly in one iteration, one can expect that in general the less nonlinear the problem is, the less sensitive to start values the convergence will be.

Ideally, the simplified model could be obtained by linearizing the actual model close to the initial steady state. In practice, this is not a good idea for two reasons. First of all, this strategy would require to modify a large fraction of the model equations (all the nonlinear ones). Secondly, obtaining such linearisation requires to know the steady state values of all the variables, which are yet to be determined. The idea is then instead to simplify only those few equations that mostly contribute to the nonlinearity of the large implicit system of equations of typical steady-state initialization problems.

Power plant models are essentially thermo-hydraulic networks with non-trivial fluid models (ideal gases with temperature-dependent c_p and vaporizing fluids, usually water), exchanging heat by convection through heat exchanger walls. The main sources of nonlinearities in the steady state initialization problem are now listed.

1. *Momentum balance equations*: pressure-flow rate relationships are usually quadratic and depend on upstream properties, such as density and viscosity, which in turn depend on thermal variables and on the flow direction.

2. *Energy balance equations* have the form $\sum_j w_j h_j + \sum_j Q_j = 0$, thus are nonlinear in the mass flow rate - specific enthalpy products $w_j h_j$.

3. The *upstream enthalpy* appearing in energy balance equations of components allowing flow reversal depends on the direction of the flow.

4. *Flow-dependent heat transfer coefficients* γ introduce nonlinearities in heat transfer equations $Q = \gamma S(T_{fluid} - T_{wall})$.

5. *Temperature-enthalpy relationships* are nonlinear in both ideal gas and water/steam models.

6. *Controllers influencing flow rates* through, e.g., valve openings, pump speeds, etc., and whose controlled variables are instead related to energy flows or storage, e.g., turbine power, boiler pressure, introduce nonlinear couplings between hydraulic and thermal equations.

7. *Controllers with control signal saturations* introduce nonlinearities in the system model.

Note that many other nonlinear equations which are present in the model are irrelevant for the steady-state initialization, because they only involve the dynamic behaviour, which is by definition not considered if all derivatives are zero. For example, the dependency on pressure and temperature (or specific enthalpy) of the fluid compressibility $\frac{d\rho}{dp}$, which enters the left-hand-side of dynamic mass balances, is irrelevant in the determination of the steady state. Therefore, if structural analysis is applied to the initialization problem, the computation of those quantities will be moved after the core implicit system of equations in the BLT transformation, and they will be computed explicitly as a function of the already computed thermodynamic states, e.g., (p,T) or (p,h). Consequently, it is only necessary to worry about those equations and those variables which are strictly necessary to solve the steady-state equations, where one assumes that all derivatives are equal to zero.

3.2.1 Momentum balance equations

The most important source of nonlinearity in the steady-state initialization problem is given by the momentum equations, which are usually quadratic in the flow rate, due to the friction term. When low pressure losses are modelled, the flow rate is highly sensitive to pressure errors: a small error in the pressures during the first Newton iterations can cause large errors in the flow rates, which in turn cause large errors in the energy balance equations, possibly bringing the specific enthalpies out of their validity range of the fluid model. Furthermore, the dependence of the momentum balance on the fluid properties introduces a nonlinear coupling between the hydraulic equations, describing pressure-flow relationships, and the thermal equations, describing energy storage and heat transfer.

All these problems are removed if the momentum balances are substituted with linear constant-coefficient pressure-flow rate relationships. These can be based on nominal operating data (nominal pressure drop, nominal flow rate), which are often already in-

cluded among the component parameters, and are usually known in advance from overall plant design data. Friction losses can be approximated by a linear function passing through the origin and the nominal operating point:

$$w = \frac{w_{nom}}{\Delta p_{nom}} \Delta p \tag{2}$$

Static head terms can be computed using a constant known nominal density:

$$\Delta p_{static} = \rho_{nom} g H \tag{3}$$

The flow characteristic of turbines can be approximated by a linear pressure-flowrate realationship:

$$w = \frac{w_{nom}}{\Delta p_{nom}} \Delta p \tag{4}$$

Control valves can be represented by a simplified equation, where the flow rate is both proportional to the pressure difference Δp and to the valve opening α:

$$w = \alpha \frac{w_{nom}}{\Delta p_{nom}} \Delta p. \tag{5}$$

This equation is still significantly nonlinear and might cause problems, in particular if α is the output of a controller (e.g., in the case of level controller for drum boilers). In this case, it is possible to further simplify the equation, making it linear, by removing the dependency on Δp:

$$w = \alpha w_{nom}. \tag{6}$$

Pump characteristics cannot be reasonably represented by a curve passing through the origin. In this case, it is convenient to use a linearised version of the characteristic curve, computed around the nominal flow rate, head, and pump rotational speed.

The simplified models are then written together with their actual counterparts, using the `homotopy()` operator. A few examples from the ThermoPower library are shown here for the sake of the example:

```
// Pressure loss component
pin - pout =
  homotopy(smooth(1, Kf*squareReg(w,wnom*wnf))/rho,
           dpnom/wnom*w) "Flow characteristics";

// Valve for incompressible fluid
w = homotopy(FlowChar(theta)*Av*sqrt(rho)*sqrtR(dp),
             theta/thetanom*wnom/dpnom*dp);

// Pump
function df_dq = der(flowCharacteristic, q_flow);
head = homotopy((n/n0)^2*flowChar(q*n0/(n + n_eps)),
       df_dq(q0)*(q - q0)+
       (2/n0*flowChar(q0) - q0/n0*df_dq(q0))*(n - n0)
       + head0);

// Turbine
w = homotopy(Kt*partialArc*sqrt(p_in*rho_in))*
               sqrtReg(1 - (1/PR)^2),
             wnom/pnom*p_in);
```

In some cases, the structure of the system of equations corresponding to the simplified steady-state initialization is such that, with these simplifications, the steady-state hydraulic equations are completely decoupled from the steady-state thermal equations. In those cases, the BLT algorithm will split the system of equations into two smaller subsystems. First, the hydraulic equations alone will be solved, determining the pressures and flow rates. Since all the involved equations are now linear, the problem is solved easily and without any concern about convergence and initial guess values. Subsequently, the thermal equations will be solved, but since the flow rates are now known, a major source of nonlinearity, i.e., the $w_j h_j$ products in energy balances, will be gone, thus making it easier to solve the thermal equations as well.

Other cases will not be this easy. Consider for example a Rankine cycle with a circulation boiler. Even though the simplified flow equation for the turbine is linear, it is apparent how the steam flow rate essentially depends on the heat input, which determines how much steam is produced. Consequently, the hydraulic and thermal equations will be coupled in this case, even when considering the simplified model. Anyway, a larger part of the equations in this system will be linear, thus easing the convergence of the nonlinear solver. More opportunities for efficient tearing will also be available, since a larger fraction of equations can be symbolically turned into an explicit assignment.

3.2.2 Energy balance equations

The nonlinearity in this case stems from the $w_j h_j$ products in the steady-state energy balances. It is not as hard as in the case of momentum balances for small pressure losses, but it can still give rise to significant problems: if during iterations, the mass flow rate is wrong by a factor of, say, two, then also enthalpy changes will be off by the same factor, which could cause out-of-bounds problems with the fluid property computation routines.

The best way to get rid of this problem is to use the nominal flow rates instead of the actual flow rates for the simplified initialization problem; by doing so, the energy balance equations become linear, and are thus solved without major problems. Unfortunately, specifying all the nominal flow rates is rather impractical for multiple-port mixing components such as storage and storage-less mixing volumes, steam drums, steam headers, etc. Furthermore, if all of those nominal values were not set to correct values, considerable errors could arise in the computation of the enthalpies, that

could hamper the convergence of the simplified problem.

A reasonable compromise, which allows to get rid of most $w_j h_j$-type nonlinearities in typical power plant models, is to using the nominal flow rate in the energy balance equations for both sides of heat exchangers, instead of the actual flow rate. The values of the nominal primary and secondary flow rates are usually known for all heat exchangers in a plant, only two parameters are needed per heat exchanger, and a lot of nonlinear equations are turned into linear equations, since there are $2N$ such energy balance equations in a heat exchanger having N discrete volumes on each side, and there are usually many heat exchangers in a given plant model. It is assumed that the few remaining nonlinear energy balance equations (contained in mixers, drums, steam headers, etc.) will be handled by the nonlinear solver without major problems.

Note that this approximation effectively removes the dependency between the flow rate and the outlet temperature of the fluid. This might then prove problematic in all those cases where a temperature controller is used to keep the outlet temperature at a given set point, because the corresponding simplified equations might become singular or ill-conditioned. In those cases, it is necessary to open the temperature feedback loop in the simplified problem - see below Sect. 3.2.6.

3.2.3 Dependence of the upstream enthalpy on the direction of the flow in energy balances

If flow reversal is allowed, the specific enthalpy of fluids entering and leaving control volumes where mass and energy balances are formulated are calculated using the upstream discretisation scheme, e.g.:

$$h = if\ w > 0\ then\ h_{entering}\ else\ h_{internal} \tag{7}$$

The discontinuity might be smoothed out in the neighbourhood of $w = 0$, but in any case these equations introduce a strong nonlinearity, if not a discontinuity, in the steady-state equations.

If the hydraulic equations of the simplified problem are completely decoupled from the thermal equations, then this is not a problem: the values of all flow rates will be computed by solving the linear hydraulic equations; then, the flow rate w will no longer be an unknown when (7) will be solved. In general, this decoupling cannot be performed, as discussed in the previous sub-section. Upon initialization, however, one can assume that the flow rate will have the design direction, so a simplified equation can be written under that assumption, e.g.:

$$h = h_{entering} \tag{8}$$

For example, this is how the specific enthalpy at the inlet port of a mixing volume is computed in the ThermoPower library:

```
hi = homotopy(if not allowFlowReversal
          then inStream(inlet.h_outflow)
          else actualStream(inlet.h_outflow),
       inStream(inlet.h_outflow));
```

3.2.4 Flow-dependent heat transfer coefficients

Convective heat transfer is represented by equations such as

$$Q = \gamma S(T_{fluid} - T_{wall}) \tag{9}$$

Simpler models assume a constant heat trasfer coefficient γ, so the equation is linear. More accurate models instead compute γ as a function of Reynolds and Prandtl numbers, which depend on the flow rate, as well as on the density, viscosity and thermal conductivity of the fluid, and possibly also on the wall temperature. All these dependencies introduce considerable nonlinearities, as well as coupling between the hydraulic and thermal equations.

The obvious strategy for simplified problem formulation is to use the nominal value of γ instead of the actual one, thus making equation (9) linear, e.g.:

```
wall.gamma[j] = homotopy(
   gamma_nom*noEvent(abs(infl.m_flow/wnom)^kw),
   gamma_nom);
```

3.2.5 Temperature-enthalpy relationships

Temperature profiles and transferred thermal power in heat exchangers are determined by the interplay between heat transfer, which is driven by temperature differences, and convective transport of heat by the fluid, which is described by enthalpy differences. The temperature-enthalpy relationships are therefore involved in the steady-state equations describing heat exchangers, namely $h = h(T)$ for ideal gases and $T = T(p,h)$ for vaporizing fluids.

In the case of ideal gases, the function is approximately linear over significant ranges of T, since its derivative, the specific heat c_p, does not change too much with the temperature. This is also the case for the vaporizing fluid, as long as the function is evaluated on the correct side of the saturation curve: the c_p of liquid water does not change dramatically with temperature, nor does the c_p of steam, with the exception

of the transcritical region and of a thin region just outside the saturation curve. On the other hand, substituting those functions with linear approximations which are consistent with each other is not trivial and would require substantial changes to the code of the original fluid model.

The strategy for the homotopy is then to rely on the fact that these functions are only mildly nonlinear, so they should not cause major convergence issues, of course as long as they are called in their range of validity. Therefore, the corresponding function calls are left untouched in the simplified model.

It is however essential to select reasonable start values for the gas temperatures, so that the guess values used for the first Newton iterations are already in the correct temperature range, as far as c_p is concerned; the precise numerical value of start attribute is not critical. As concerns the vaporizing fluid properties, start values should be selected so that the first Newton iterations compute the properties on the correct side of the saturation curve, i.e., subcooled liquid or superheated steam.

The user input is therefore a very rough temperature value for the gas side (say, 400 rather than 600 or 800 K, for standard flue gas heat exchanger), and the indication of a nominal pressure and of the phase (liquid or vapour) for the vapour size, that can be used internally in the model to compute start values of the specific enthalpies corresponding to well-subcooled liquid and well-superheated steam.

Evaporating pipes are less critical from this point of view, because in a two-phase mixture the temperature-enthalpy relationship becomes flat, i.e. the temperature no longer depends on the enthalpy, but only on the pressure, which usually does not change much across the pipe length.

3.2.6 Controllers acting on flows and controlling energy-related quantities

It is often the case that the plant model is complete with controllers, and that the goal is to initialize the whole controlled system in steady state. If the controller is active and contains some integral action on the error, the steady-state equations are equivalent to the equation

$$y = y_{sp}, \tag{10}$$

where y_{sp} is the value of the set point. This equation, coupled with the rest of the plant model, implicitly determines the value of the control value, e.g. a valve opening or a pump rotational speed.

If the control variable directly influences a flow rate, and the controlled variable is manly determined by the energy flows, (10) introduces a strong nonlinear coupling between the hydraulic equations and the thermal equations, thus hampering the solver convergence.

Consider the following example. The last economizer stages of a heat recovery steam generator usually allow to modulate a recirculation flow in order to control the outlet temperature of the preheated water to the desired value. In order to change this value, valves or pump speeds must be changed, that can also affect the water/steam flow through the evaporator and superheater, thus greatly influencing all the thermal power transfer phenomena across the steam generator. During the first iterations of the nonlinear solver, the gas temperature near the exhaust can be quite different from its design value: this causes the recirculation flows to be also different from the design values, thus influencing the evaporator and superheater flows, which in turn affect the temperature of the gas heating them. In the end, the solver might get stuck far away from the sought after solution even when considering the simplified equations for the physical model.

Should this happen, it usually is possible to roughly estimate what the value of the control variable will be in the nominal operating point of the plant. It is then possible to remove the above-described nonlinear coupling by using a simplified model of the controller that just outputs the start value of the control variable. Of course this means that the steady-state operating point of the simplified model will be slightly off with respect to the correct value, but this is not a problem, as long as the operating point is physically meaningful and not too far from the exact solution. The homotopy transformation will then slowly introduce the closed-loop controller action, thus smoothly bringing the controlled variables to their set points at the end of the homotopy transformation.

In some cases, an explicit controller model is not included in the plant model, and the steady-state operating point is just obtained by adding equations such as (10) for the desired outputs (inverse initialization). In this case, those equations should use the homotopy operator to blend the prescribed control value (simplified model) with the prescribed output value, e.g.:

```
0 = homotopy(valve_opening - valve_opening_nominal,
             T_out - T_out_nominal);
```

3.2.7 Controllers with control signal saturations

It is often the case that controllers in controlled plant models include saturations, i.e., limitations in the control variable range. If the saturation limits are wide

enough, then they are actually irrelevant: the controller is modulating, and the effect of the controller is equivalent to the steady-state equation on the integral action. This is in turn equivalent to (10), which implicitly determines the value of the control variable, within the saturation limits. On the other hand, if an out-of-bound control action would be required to attain the set point, then the saturation is engaged, equation (10) no longer holds and is replaced by either

$$u = u_{max} \tag{11}$$

or

$$u = u_{min}. \tag{12}$$

Irrespective of the way the saturating controller is actually implemented (e.g. with or without anti-windup action), the above scenarios always hold, indicating a strongly nonlinear behaviour of the corresponding system of equations. In other words, letting the solver figure out which controllers are in a modulating state, which are saturated high and which are saturated low corresponds to solving a highly nonlinear problem, with potentially combinatorial complexity, which can cause serious convergence problems to the solver.

Doing so is however not necessary in general, since the status of the controllers in the nominal operating point is usually well known. In case it is known in advance that the controller will be modulating, then the saturation limits can be removed in the simplified model, thus making the model linear. In case it is known in advance that the control output will be saturated at the maximum or minimum limits, then the saturation equation is replaced with an equation stating that the control output is fixed at the maximum or minimum value.

Note that this functionality can be merged with the functionality described in the previous subsection. Summing up, the simplified model should either remove the saturation limits from the output, or hold the output at a fixed value, which might be a specific nominal value, the maximum, or the minimum, depending on the situation.

3.3 Solving the simplified model

It is apparent that all the above-described simplification strategies reduce the couplings between equations and the nonlinear effects, compared to the actual initialization problem. In order to take full advantage of these simplifications and ensure the highest chance of convergence, it is recommended that the tool applies structural analysis and optimization (BLT transformation, tearing, etc.) to the simplified initialization problem, obtained by replacing all instances of the homotopy() operator with their simplified argument.

When this is done, then the iteration variables of the initialization problem, i.e, the tearing variables, typically belong to these categories:

- Gas-side temperature distributions in heat exchangers
- Wall temperatures distributions in heat exchangers
- Water/steam side enthalpy distributions in heat exchangers
- Steam drum pressures
- A few other flow rates and pressures

The first two sets will need very rough start values (say 400, 600 or 800 K, depending on the heat exchanger); there will be no need at all to provide estimates of the actual temperature distributions within heat exchangers. The third set also requires very rough start values (subcooled liquid or superheated steam, depending on the case). Therefore, appropriate start values can be set for all these variables in the model, based on a a couple of numerical parameters in the heat exchanger component, whose precise value is not at all critical for convergence. Steam drum start values can be easily supplied based on nominal operating point data. If the flow rates belong to heat exchanger components, a nominal value is already available, since it is required to perform the energy balance equation simplification, so it can also be used to set the start value.

In some cases, there might still be a very few remaining iteration variables that don't have any meaningful start value, causing the solver to fail. These can be fixed on a case by case basis, or by adding suitable start and/or nominal parameters to the corresponding library model. Ideally, required start values should be inferred from parameters of the component which give information about the nominal operating point, without the need of extra ad-hoc input by the end-user.

3.4 Steady-state initialization far from the nominal operating point

The simplified problem has been designed to approximate the actual problem at the nominal operating point. What if one wants to initialize the plant at a

different operating point, e.g. 40% load, instead of the nominal 100% load?

One idea in this case is to use the homotopy() operator to parameterize the signal sources that define the operating point. For example, if the load set-point is generated by a step or ramp source, one might write `homotopy(40, 100)` as the offset value. This means that the simplified initialization problem is actually solved with an offset of 100%, i.e., at full load; during the homotopy transformation, while the problem is brought to its actual form, the load is also progressively reduced to 40%, thus eventually converging to the required steady state.

It has been verified in a number of test cases (see next Section) that this additional mismatch between simplified and actual model does not lead to any singularity of the solution during the homotopy transformation, and guarantees successful convergence of the actual initialization problem.

Should this not be the case, two homotopy transformations should be performed in sequence: first the simplified problem at nominal load should be transformed to the actual model, also at nominal load; then, the load set point should be reduced, therefore realising a quasi-static change of the operating point from full load to partial load, which should pose no problems. Unfortunately this is not possible with the current definition of the homotopy() operator, which only allows for a single, system-wide transformation.

4 Experimental results

The general ideas illustrated in the previous sections has been implemented in the version 3 of the ThermoPower library [4]. The library has then been used to build a series of test cases of increasing complexity, culminating in the complete model of a combined-cycle power plant, whose heat recovery steam generator (HRSG) includes 15 different heat exchangers. A few extra parameters for nominal values required by the simplified model had to be added to the formerly developed heat exchanger models; however, they are a very small fraction of the number of parameters already needed to set up those models and, as noted in the previous section, their numerical values need not be precise by any means. Furthermore, and more important, these parameters are set once and for all in a given plant model and need not be changed on a case-by-case basis depending on the choice of start values of the Modelica tool.

For the simpler cases, homotopy was actually not necessary to solve the initialization problem, but as the complexity increased, more and more cases fail to initialize when the built-in solver of Dymola is used, because of initial guesses which are not accurate enough. All the simplified models converged without problems (as long as the nominal parameter gives a correct order of magnitude for all the iteration variables) and the homotopy paths of all the iteration variables proved to be smooth and devoid of turning points or worse singularities.

The three largest and hardest-to-solve cases are briefly documented here. The model describes a complete combined-cycle power plant. The three levels of pressure HRSG includes 15 heat exchangers, each one discretized by finite volumes and with flow-dependent heat transfer coefficients. The steam turbine system includes a condenser model and a pumping system model, so the water/steam cycle is closed. The turbines operate in sliding pressure; control loops are included to control the steam drum levels, the superheater outlet temperatures, the economizer outlet temperature, and the combined electrical power output of the gas and steam turbines. Three variants have been considered:

1. reference plant model, initialized at 100% load;
2. reference plant model, initialized at 60% load;
3. detailed plant model, with two parallel HRSGs, common steam collector and steam turbine system, also initialized at 100% load;

The initialization problem of case 1. has 345 iteration variables. During the homotopy transformation, no variable shows bifurcations or turning points. Most variables change by less than 5% between $\lambda = 0$ and $\lambda = 1$. The outputs of the superheaters and reheaters temperature controllers (which are fixed to the start value at $\lambda = 0$ and work in closed loop at $\lambda = 1$) show the biggest variations, but change smoothly and without singularities during the homotopy transformation.

The computation of the transformation took 8 steps and 40 seconds using the test implementation, running on a 2.26 GHz P9300 Intel processor. For the sake of the example, Fig. 2 shows some representative plots of iteration variables during the transformation.

Case 2 has the same number of iteration variables, but now the homotopy transformation also involves bringing down the load from the nominal 100% value to 60%, so it is a bit more involved, because the values of the initial steady state significantly differ from the nominal values. This time, the transformation required 37 steps and took 100 seconds to compute. Fig.

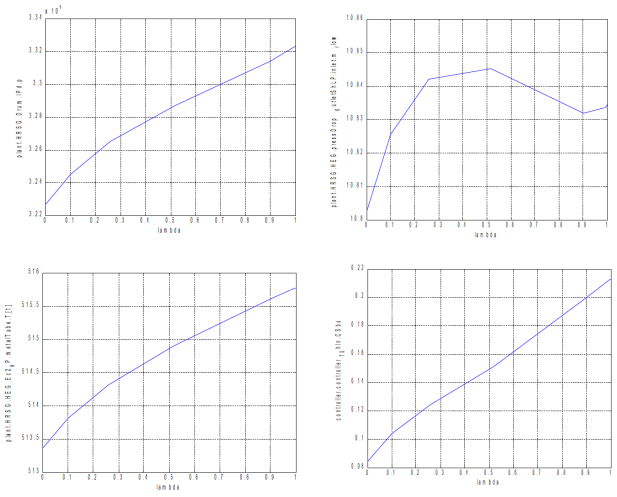

Figure 2: Homotopy paths for 100% load initialization

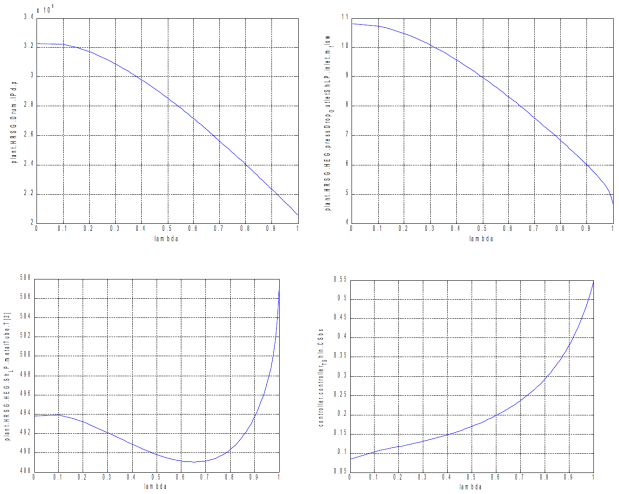

Figure 3: Homotopy paths for 60% load initialization

3 shows the plots of the same iteration variables considered in the previous case: it is apparent how the change in the values is now substantial, since it involves a large change in the operating point, but the transformation is nevertheless smooth and without singularities.

Case 3, which includes two HRSG models in parallel, has 673 iteration variables. The transformation took 7 steps and 170 seconds of CPU time to be performed. The plots of the iteration variables are similar to those shown in Figure 2 as expected, since also in this case the system is initialized at 100% load.

As a final consideration, note that the experimental code uses a brute-force numerical approach to compute the Jacobian, which can be computed in a much more efficient way by exploiting its sparsity pattern. Furthermore, not much time has been devoted to the optimal setting of the continuation solver. A production-quality implementation is therefore expected to be substantially faster to perform the transformation shown above.

5 Conclusions and outlook

A strategy for robust and reliable steady-state initialization of large thermo-hydraulic system has been presented in this paper. The basic idea is to simplify a few selected equations in order to form a simplified initialization problem that is easily solved, without need of setting accurate start values for the iteration variables; subsequently, smoothly transform this problem into the actual problem of interest, getting its initialization by continuity.

The proposed strategy has been demonstrated by means of a test implementation of the `homotopy()` operator, applied to large models of combined-cycle power plants with up to 671 iteration variables. All the examined test cases were solved successfully and the homotopy paths of all the iteration variables did not show singular behaviour of any sort, thus confirming the validity of the selection criteria for the simplified model.

By adding a few more parameters to the model, indicating nominal values (without need of particular accuracy), the proposed method completely eliminated the need by the end user of setting start values on the particular problem at hand, in order to ensure convergence. The authors thus argue that they have demonstrated a truly modular and object-oriented approach to reliable steady-state initialization for large thermo-hydraulic networks.

The availability of built-in, fast and numerically well-behaved homotopy methods in Modelica tools would make this approach a lot more user-friendly than using the prototype implementation employed for this study, which was only meant to demonstrate the soundness of the proposed approach from the point of view of the mathematical modelling involved.

6 Acknowledgement

The financial support of EDF under contract 5900058671 (Development of an efficient method for power plant modelling) is gratefully acknowledged by the first and last authors.

References

[1] Sielemann M., Casella F., Otter M., Clauss C., Eborn J., Mattsson S.E., Olsson H., Robust Initialization of Differential-Algebraic Equations Using Homotopy. Submitted to Modelica Conference 2011, Dresden, Germany, 20–22 March 2011.

[2] Heroux M. A., Bartlett R. A., Howle V. E., Hoekstra R. J., Hu J. J., Kolda T. G., Lehoucq R. B., Long K. R., Pawlowski R. P., Phipps E. T., Salinger A. G., Thornquist H. K., Tuminaro R. S., Willenbring J. M., Williams A., Stanley K. S. An overview of the Trilinos project, ACM Transactions on Mathematical Software, 31, 397–423, 2005.

[3] Chow S. N., Mallet-Paret J., Yorke J. A. (1978): Finding Zeroes of Maps: Homotopy Methods That are Constructive With Probability One. Mathematics of Computation 32, pp. 887-899, 1978.

[4] Casella F., Leva A., Modelica Open Library For Power Plant Simulation: Design And Experimental Validation. Proceedings 3rd International Modelica Conference, Linköping, Sweden, Nov. 3-4, 2003, pp. 41-50.