Proceedings
of the 3rd International Modelica Conference,
Linköping, November 3-4, 2003,
Peter Fritzson (editor)

Rüdiger Franke, Manfred Rode, Klaus Krüger
ABB Corporate Research, ABB Utilities GmbH, Germany:
On-line Optimization of Drum Boiler Startup
pp. 287-296

Paper presented at the 3rd International Modelica Conference, November 3-4, 2003, Linköpings Universitet, Linköping, Sweden, organized by The Modelica Association and Institutionen för datavetenskap, Linköpings universitet

All papers of this conference can be downloaded from
http://www.Modelica.org/Conference2003/papers.shtml

On-line Optimization of Drum Boiler Startup

Rüdiger Franke, Manfred Rode
ABB Corporate Research
Wallstadter Str. 59
68526 Ladenburg, Germany

Klaus Krüger
ABB Utilities GmbH
Kallstadter Str. 1
68309 Mannheim, Germany

E-Mail: {Ruediger.Franke, Manfred.Rode, Klaus Krueger}@de.abb.com

Abstract

On-line optimization of industrial processes is increasingly important to minimize cost and environmental impact of a plant during its operation. The new Modelica.Media and Modelica.Fluid base libraries allow the dynamic modeling of process systems [4]. Their application to dynamic optimization is discussed on a tutorial example.

The optimization is based on a simple non-linear drum boiler model from the literature [1]. The model is implemented in Modelica using the new Modelica.Media and Modelica.Fluid base libraries.

The model exhibits three control inputs: feed water flow rate, heat input, and position of a valve at the steam outlet. A PI control is embedded into the model for the feed water flow. The remaining two control inputs are optimized. Optimization results are compared with a straightforward control strategy.

On-line optimization based on a sophisticated boiler model is currently being applied to a 700 MW coal fired power plant.

1 Introduction

The primary aim of on-line optimization of industrial processes is to minimize cost and environmental impact of a plant during its operation. This cost includes energy and raw material consumption, losses due to off-spec production, waste and exhaust treatment as well as maintenance. Especially for complex processes with many mutually interacting decision variables and constraints, a secondary aim often is to automatically find a feasible and reproducible operation.

A dynamic model can serve as basis for on-line optimization. An optimal control problem is formulated for the model. The problem is solved in real-time and the optimization results are applied to the process.

On-line optimization is especially interesting as simulation models are being reused. However, the optimization is computationally expensive. This is because control trajectories are being sought and as constraints have to be fulfilled for state and output trajectories. Time discretization leads to large numbers of optimization variables and constraints, as opposed to a comparable small number for design optimization problems. That is why advanced numerical optimization methods are required to solve the large-scale optimization problems in real-time.

The on-line optimization can be performed repeatedly in a control loop. The resulting Nonlinear Model based Predictive Control (NMPC) is advantageous if a process model is available or effordable and if

- multiple controlled and manipulated variables need to be considered
- state constraints have to be fulfilled
- the control problem is non-linear

The example discussed here exhibits all these features.

2 Drum boiler model

2.1 Generic drum model

The drum boiler model from [1] is used. Inside the drum there is water in two phases: liquid an vapour. The thermodynamic state of both phases is assumed to be at the phase boundary. Feed water enters the drum and saturated steam leaves the drum. A furnace supplies energy for heating up and evaporating the feed

water. The model assumes a global mass balance

$$\frac{dm}{dt} = q_{m,W} - q_{m,S} \tag{1}$$

for feed water entering and steam leaving the drum and with the mass

$$m = \rho_v V_v + \rho_l V_l + m_D. \tag{2}$$

See the code in figure 2 for a list of variables. The global energy balance is

$$\frac{dU}{dt} = q_F + q_{m,W} h_W - q_{m,S} h_S. \tag{3}$$

with the internal energy

$$U = \rho_v h_v V_v + \rho_l h_l V_l - p(V_v + V_l) + m_D c_{p,D} T_D \tag{4}$$

considering vapour phase, liquid phase, volume work, and the thermal energy in the surrounding metal, respectively. It is assumed that the specific enthalphy of steam leaving the boiler is equal to the vapour enthalpy: $h_S = h_v$. Furthermore, ideal heat transfer between the water inside the drum and the surrounding metal is assumed. Consequently the metal temperature is equal to the saturation temparature of water for the pressure inside the drum: $T_D = T_{sat}(p)$. The constant total volume inside the drum boiler is

$$V_t = V_v + V_l. \tag{5}$$

Thermal stress occures in the thick walled drum if there are spatial temperature differences, which are caused by fast temperature variations, e.g. during startup. As this stress leads to fatigue or even destruction, it needs to be hold within given limits. Here the thermal stress is modeled proportional to the time derivative of the metal temparature

$$\sigma_D = k \frac{dT_D}{dt}. \tag{6}$$

Note that the modeling of temperature gradients is also of practical importance when no measurements of wall temperatures are available.

2.2 Mathematical model analysis and transformation

The physical oriented generic model formulation given in subsection 2.1 can directly be formulated in Modelica. Mathematical details are treated automatically by the Modelica tool Dymola. Nevertheless the model is analyzed in this section, in order to outline important mathematical details.

The model forms a system of differential and algebraic equations (DAE). It has a number of disadvantages when applied to on-line optimization. The balance equations are formulated for mass and internal energy that are not measured. Drum pressure, temperature and liquid water level are important quantities for drum boiler control. The physical oriented model defines them via an implicit non-linear relationship, being numerically disadvantageously.

Moreover, equation (6) for thermal stress causes a high-index DAE, as the time derivative of drum temperature is used.

That is why the model shall be transformed into a more appropriate form prior to its application. In [1] it is proposed to use pressure p and volume of liquid V_l as state variables. It is explained, how the model equations for mass and energy balance can be transformed accordingly. This results into

$$e_{1,1} \frac{dV_l}{dt} + e_{1,2} \frac{dp}{dt} = q_{m,W} - q_{m,S}, \tag{7}$$

$$e_{2,1} \frac{dV_l}{dt} + e_{2,2} \frac{dp}{dt} = q_F + q_{m,W} h_W - q_{m,S} h_S \tag{8}$$

with

$$e_{1,1} = \rho_l - \rho_v \tag{9}$$

$$e_{1,2} = V_v \frac{\partial \rho_v}{\partial p} + V_l \frac{\partial \rho_l}{\partial p} \tag{10}$$

$$e_{2,1} = \rho_l h_l - \rho_v h_v \tag{11}$$

$$e_{2,2} = V_v \left(h_v \frac{\partial \rho_v}{\partial p} + \rho_v \frac{\partial h_v}{\partial p} \right) \tag{12}$$

$$+ V_l \left(h_l \frac{\partial \rho_l}{\partial p} + \rho_l \frac{\partial h_l}{\partial p} \right) \tag{13}$$

$$- V_{vl} + m_D c_{p,D} \frac{\partial T_{sat}(p)}{\partial p} \tag{14}$$

Furthermore, equation (6) can be re-written as

$$\sigma_D = k \frac{\partial T_D}{\partial p} \frac{dp}{dt}. \tag{15}$$

```
package WaterPhaseBoundaryIF97
  "Physical properties for water at phase boundary at boiling and dew curves"
  extends Modelica_Media.Interfaces.PartialMedium(
    mediumName = "WaterIF97",
    substanceNames = fill("", 0),
    incompressible = false,
    reducedX = true,
    MassFlowRate(quantity="MassFlowRate.WaterIF97"));
  // basic property definitions required for each medium model
  redeclare model BaseProperties
    extends;
    parameter Integer region = 0 "specify region 1 (liquid) or 2 (vapour)";
  equation
    assert(region == 1 or region == 2,
           "WaterPhaseBoundaryIF97 medium model only valid for regions 1 and 2");
    T = Modelica_Media.Water.IF97.BaseIF97.Basic.tsat(p);
    if region == 1 then
      d = Modelica_Media.Water.IF97.BaseIF97.Regions.rhol_p(p);
      h = Modelica_Media.Water.IF97.BaseIF97.Regions.hl_p(p);
    else
      d = Modelica_Media.Water.IF97.BaseIF97.Regions.rhov_p(p);
      h = Modelica_Media.Water.IF97.BaseIF97.Regions.hv_p(p);
    end if;
    u = h - p/d;
  end BaseProperties;
end WaterPhaseBoundaryIF97;
```

Figure 1: Property model for water at phase boundary between liquid and vapour

This model can now easily be applied to study the system dynamics and to determine relevant terms for a control application. After the non-linear state-space transformation, the implicit dependency of pressure and liquid volume has become linear.

2.3 Model implementation

The model implementation consists of three steps: The selection of appropriate medium models, the implementation of the evaporator component model, and the assembling of a complete system model allowing the simulation of the drum boiler.

The Modelica.Media library contains accurate properties for water and steam according to the IAPWS/IF97 standard [5]. Here a new medium model is formulated for the phase boundaries at the boiling and dew curves, using available low level function calls for property evaluation.

Figure 1 shows the Modelica formulation. The medium model is defined as a package assembling general information, like medium name, and actual property definitions. The package is derived from the predefined Modelica.Media.Interfaces.PartialMedium. As the medium model is for a single substance, the flag for reduced mass fraction vector X is set to true, resulting in $\dim(X) = n - 1 = 0$ for $n = 1$ substance. Substance names are not defined.

The model BaseProperties contains the actual property definitions. It defines saturation temperature, density, enthalpy and specific total energy as functions of pressure. The region parameter is used to determine at which boundary the properties are evaluated.

This medium model can now be used to formulate the evaporator component model. Note that in Modelica the physically oriented model is directly formulted. Model transformations required for efficient and robust simulation, as e.g. discussed in subsection 2.2, are left to the Modelica translator.

Figure 2 shows the Modelica formulation. Considering that just the model given in subsection 2.1 is implemented, the listing appears relatively long. This is because many parameters and variables are involved that are declared one per line.

The evaporator model first imports the re-used libraries Modelica.Fluid.Interfaces and Model-

```
model Evaporator
  import Modelica_Fluid.Interfaces.*;
  import Modelica.SIunits.Conversions.*;
  import SI = Modelica.SIunits;
  // property and interface declarations
  package Medium = WaterPhaseBoundaryIF97;
  Medium.BaseProperties medium_a(region=1, p=port_a.p) "Medium in port_a";
  Medium.BaseProperties medium_b(region=2, p=port_b.p) "Medium in port_b";
  FluidPort_a port_a(redeclare package Medium = Medium);
  FluidPort_b port_b(redeclare package Medium = Medium);
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heatPort;
  Modelica.Blocks.Interfaces.OutPort V(redeclare type SignalType = SI.Volume)
    "liquid volume (level)";
  Modelica.Blocks.Interfaces.OutPort sigma_D "Thermal stress in metal";
  // public parameters
  parameter SI.Mass m_D=300e3 "mass of surrounding drum metal";
  parameter SI.SpecificHeatCapacity cp_D=500
    "specific heat capacity of drum metal";
  parameter SI.Volume V_t=100 "total volume inside drum";
  parameter SI.Pressure p_start=from_bar(1) "initial pressure";
  parameter SI.Volume V_start=67 "initial liquid volume";
protected
  SI.Pressure p(start=p_start, stateSelect=StateSelect.prefer)
    "pressure inside drum boiler";
  SI.Volume V_v "volume of vapour phase";
  SI.Volume V_l(start=V_start, stateSelect=StateSelect.prefer)
    "volumes of liquid phase";
  SI.SpecificEnthalpy h_v=medium_b.h "specific enthalpy of vapour";
  SI.SpecificEnthalpy h_l=medium_a.h "specific enthalpy of liquid";
  SI.Density rho_v=medium_b.d "density in vapour phase";
  SI.Density rho_l=medium_a.d "density in liquid phase";
  SI.Mass m "total mass of drum boiler";
  SI.Energy U "internal energy";
  SI.Temperature T_D=heatPort.T "temperature of drum";
  SI.HeatFlowRate q_F=heatPort.Q_dot "heat flow rate from furnace";
  SI.SpecificEnthalpy h_W=port_a.e "feed water enthalpy";
  SI.SpecificEnthalpy h_S=medium_b.h "steam enthalpy";
  SI.MassFlowRate qm_W=port_a.m_dot "feed water mass flow rate";
  SI.MassFlowRate qm_S=port_b.m_dot "steam mass flow rate";
equation
  // balance equations
  m = rho_v*V_v + rho_l*V_l + m_D;
  U = rho_v*V_v*h_v + rho_l*V_l*h_l - p*V_t + m_D*cp_D*T_D;
  der(m) = qm_W + qm_S;
  der(U) = q_F + qm_W*h_W + qm_S*h_S;
  T_D = medium_a.T;
  // ideal heat transfer between metal and water
  V_t = V_l + V_v;
  // pressure and specific total enthalpies at ports
  port_a.p = p;
  port_b.p = p;
  port_b.E_dot = semiLinear(port_b.m_dot, port_b.e, h_v);
  port_a.E_dot = semiLinear(port_a.m_dot, port_a.e, h_l);
  // thermal stress
  sigma_D.signal[1] = 60*der(T_D);
  // liquid level
  V.signal[1] = V_l;
end Evaporator;
```

Figure 2: Evaporator component model

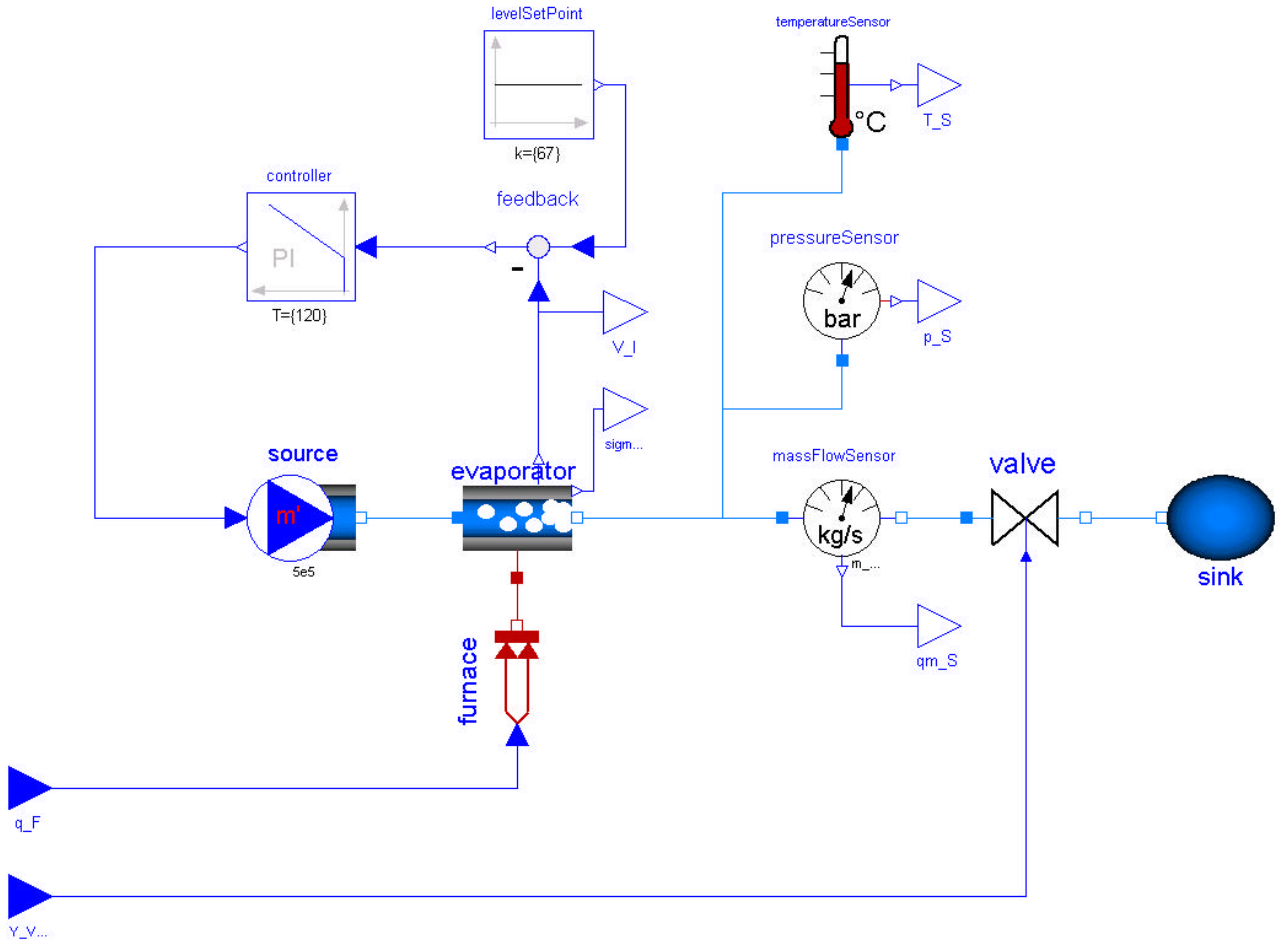

Figure 3: DrumBoiler system model in Dymola

ica.SIunits. Then the medium models and ports are declared together with parameters in a public section. The medium model defined above is instantiated twice: one for the liquid phase and one for the vapour phase. Internal model variables are declared in a protected section. Finally the model equations are stated and internal model variables are assigned to the public ports.

Note the use of semiLinear to define the energy flow through each port. This mechanism enables the treatment of reversible flow, see [4]. For example at port_a, either water with given total enthalphy port_a.e may enter, or liquid with enthalpy h_l may leave the evaporator.

A further important detail is the stateSelect attribute defined for pressure p and liquid volume V_l. This tells the translator to do the model transformation outlined in subsection 2.2.

Having the medium model and the evaporator component model ready, a complete system model is assembled in the third step, adding a feed water pump, a valve at the steam outlet, sensors, and an ambient reference point. The composition of a system model is easiest done graphically. Figure 3 shows the complete drum boiler model assembled with Dymola.

The feed water flow needs to be controlled so that the water level inside the drum is kept at its set point. A PI control loop is added to the system model for this purpose. These additional component models are taken from the standard Modelica.Blocks library.

2.4 Model translation

Prior to numerical calculations, the Modelica model is translated to a mathematical system of differential-algebraic equations (DAE) and further transformed to a system of ordinary differential equations (ODE)

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t], \tag{16}$$
$$\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \mapsto \mathbb{R}^{n_x},$$
$$\mathbf{y}(t) = \mathbf{g}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t], \tag{17}$$
$$\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \mapsto \mathbb{R}^{n_y}.$$

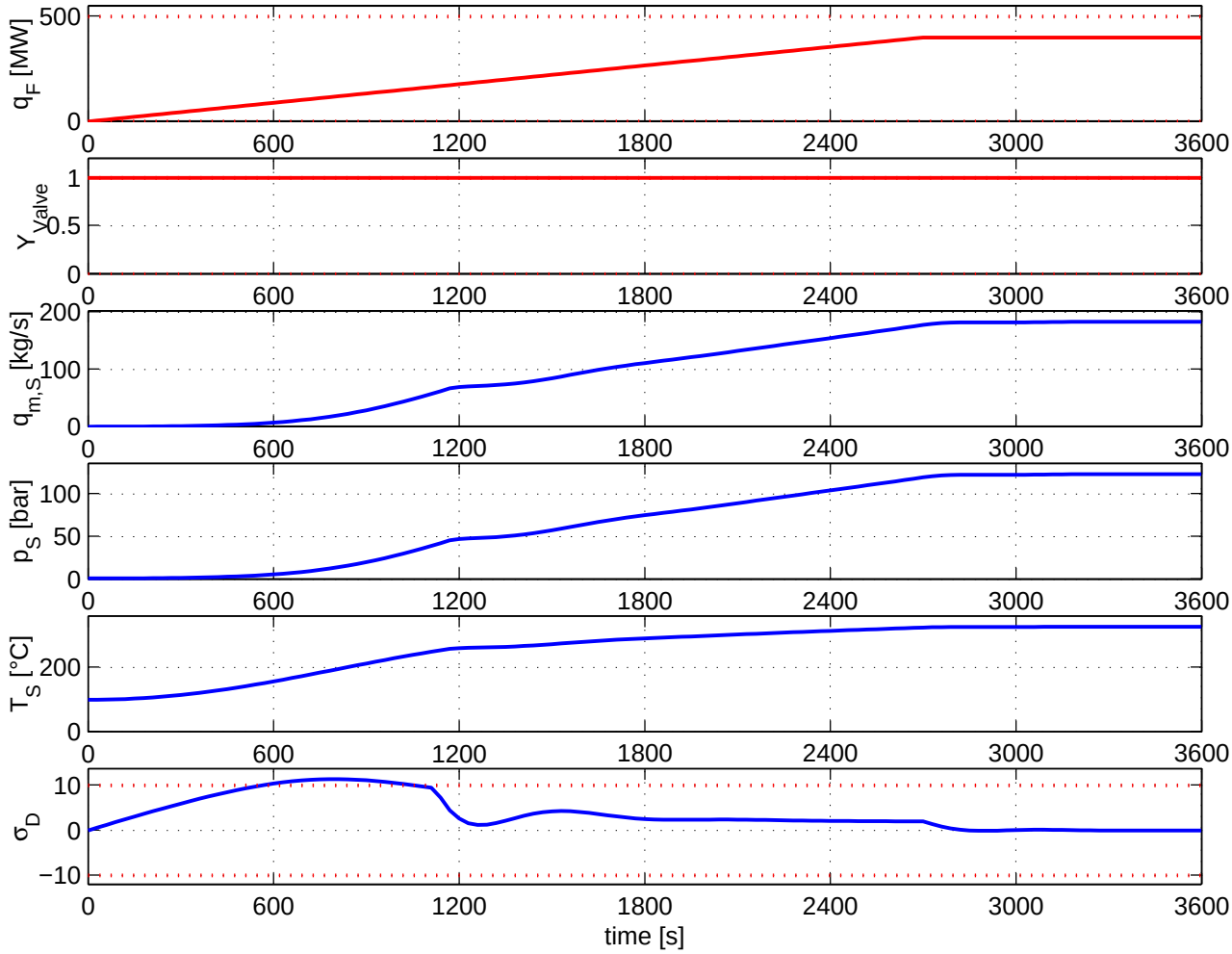

Figure 4: Simulation results applying a ramp to the heat flow and holding the valve open.

Model variables are internal continuous-time states $\mathbf{x} \in \mathbb{R}^{n_x}$, controlled inputs $\mathbf{u} \in \mathbb{R}^{n_u}$, constant parameters $\mathbf{p} \in \mathbb{R}^{n_p}$, and model outputs $\mathbf{y} \in \mathbb{R}^{n_y}$.

In the drum boiler example there are

$$\mathbf{u} = (q_F, Y_{Valve}) \tag{18}$$

$$\mathbf{x} = (V_l, p_S, x_{PI}) \tag{19}$$

$$\mathbf{y} = (T_S, p_S, q_{m,S}, V_l, \sigma_D) \tag{20}$$

with x_{PI} coming from the PI controller.

The model exhibits a nonlinear dynamics caused by material properties of water and steam, the large range of operation passed during startup, and the embedded control of the water level.

3 Boiler startup problem

During startup, a specified set point for steam temperature, pressure and mass flow rate shall be reached as fast and efficient as possible, considering constraints on process variables. The most important constraints arise from thermal stress on thick-walled parts that are heated up, σ_D in the example treated here, cf. (6).

The boiler startup can be simulated for a given control strategy using e.g. Dymola [2] or Simulink [12]. Here the HQP optimization solver is applied, see next section, which provides initial-value simulation as a subfunctionality. Figure 4 shows simulation results when increasing the heat flow in the form of a ramp during 45 minutes and holding the valve at the steam outlet open. With this strategy, the startup takes about 45 minutes. The constraint on thermal stress is violated.

4 Optimal boiler startup

The boiler startup problem can be treated as optimal control problem minimizing an objective function subject to constraints.

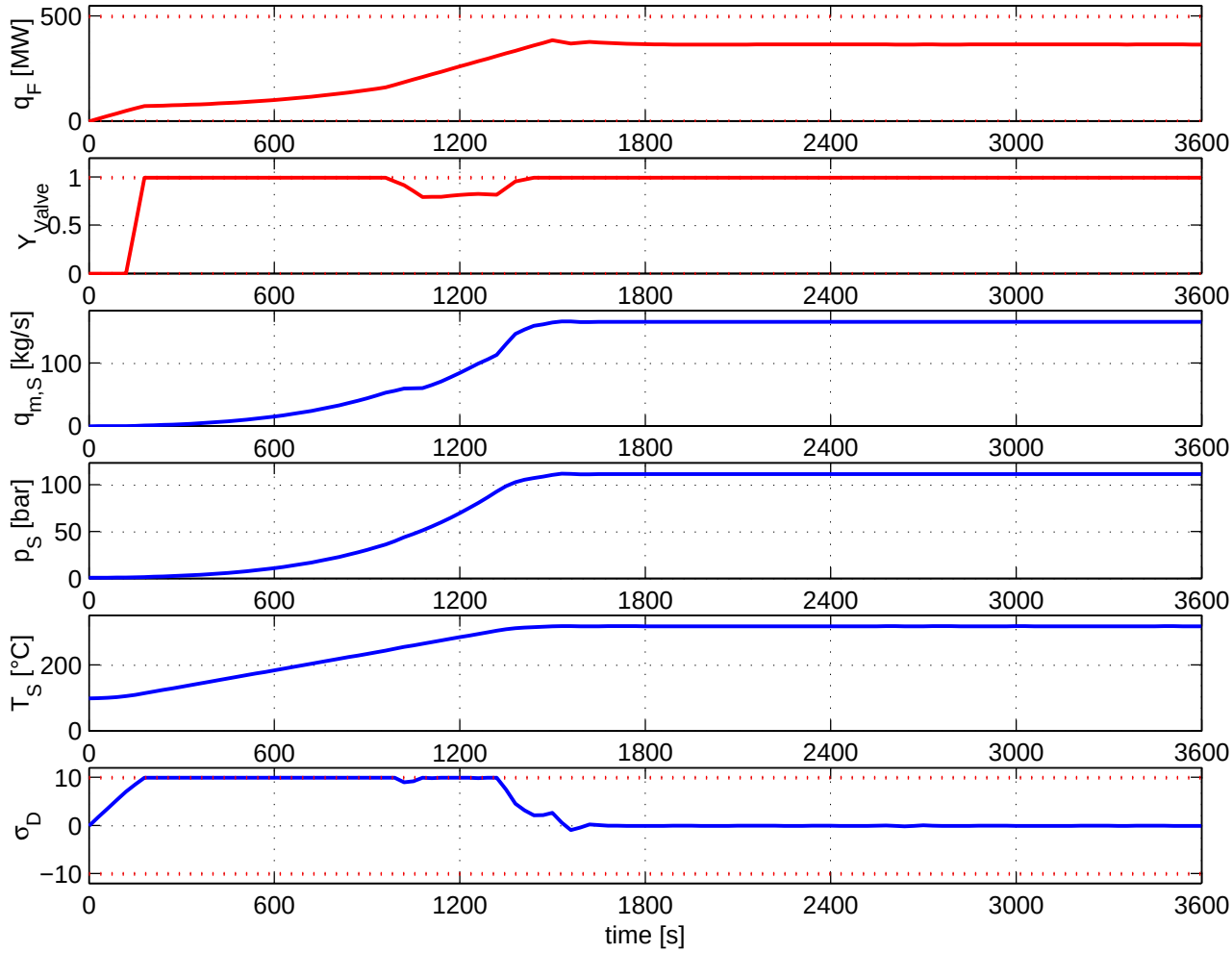

Figure 5: Optimal startup control minimizing the optimization objective subject to constraints.

In the example treated here, the objective is to minimize the deviation of generated steam pressure and mass flow rate from given reference points over the time horizon $[t_0, t_f]$

$$J = \int_{t=t_0}^{t_f} \mathbf{w}^T \left\{ \begin{bmatrix} p_S(t) - p_{ref} \\ [q_{m,S}(t) - q_{m,ref}] \end{bmatrix}^2 \right\} dt \rightarrow \min_{\mathbf{u}(t)} \qquad (21)$$

with the reference point being $p_{ref} = 110\,\text{bar}, q_{m,ref} = 180\,\text{kg/s}$ and the vector of weighting terms $\mathbf{w} = (10^{-3}, 10^{-4})^T$. The objective shall be minimized subject to the system model (16).

Constraints that have to be fulfilled over the entire optimization horizon $t \in [t_0, t_f]$ can be divided into control input constraints and state or output constraints. Input constraints are the control bounds

$$0 \leq q_F \leq 500\,MW \qquad (22)$$

$$0 \leq Y_{Valve} \leq 1 \qquad (23)$$

and the rate-of-change bound on the heat flow

$$-25\,MW/min \leq dq_F/dt \leq 25\,MW/min \qquad (24)$$

The thermal stress is formulated as output constraint

$$-10 \leq \sigma_D \leq 10. \qquad (25)$$

Figure 5 shows the solution of the optimal startup control problem. As a result of the dynamic optimization, the startup time can be reduced to less than 30 minutes, fulfilling all constraints. This is mainly due to better exploitation of allowed limits and the mutual dependence of multiple process variables. Especially the thermal stress limit is exploited during almost the complete startup. The valve position is used to fine control the startup, especially at times when the heat flow is limited by the rate-of-change bound (between 1000 and 1500 seconds).

5 Solver issues

The dynamic optimization problem was solved using the HQP code [7]. The optimizer accesses the same compiled model (16) as a simulator. In fact the model is loaded dynamically as Simulink S-function. During each optimization iteration, HQP solves the model differential equations and internally derived sensitivity equations, in order to evaluate optimized control input trajectories and to find directions for further improvement, respectively. The dynamic optimization problem is transformed to a non-linear optimization problem with the method of control vector parameterization. In total 121 parameters (free optimization variables) are introduced to describe the two control trajectories. 183 additional optimization variables are introduced for discrete-time states. The state constraint is evaluated at 120 sampling time points. The solution of the optimization problem takes 34 seconds on a PC Pentium IV, 1.8 GHz. It can be reduced down to 5 seconds applying a fixed step-size implicit integration rule to differential and sensitivity equations (cf. inline integration for real-time simulation [3]).

While accessing the same model as a simulation solver, an optimization solver generally has stronger requirements on a model. HQP implements a sparse Sequential Quadratic Programming algorithm that is of quasi Newton type and considered state-of-te art for large-scale non-linear optimization. The model needs to be smooth with respect to the optimization variables, allowing the determination of model sensitivities. This strongly limits the use of features like discrete events, reversible flows, and the like, that can be treated by simulation solvers without any problem.

Model sensitivities are obtained by integrating sensitivity equations together with model equations. The sensitivity equations are based on model Jacobians, see [6] for more details. That is why a model being translated for optimization should contain Jacobians, besides the compiled model equations. Note that the used Simulink S-function format supports Jacobians and that Dymola can generate them.

The Modelica.Fluid and Modelica.Media libraries were designed such that model Jacobians can be derived automatically by a Modelica tool. This is especially important for medium models accessing external functions that can not be differentiated automatically by a Modelica tool. An annotation syntax exists to refer to Jacobian information as available.

The automatic generation of model Jacobians does not work for high-index DAE's and medium models providing first order derivatives only, as the example treated here. This is because the first order derivatives are already used for the transformed model, cf. subsection 2.2. Second order derivatives of the state dependent medium properties would be needed for the model Jacobian. Even higher order derivatives would be needed for a DAE index > 2. If a medium model is formulated using internal Modelica functions, like Water.IF97 used here, a Modelia tool might apply algorithmic differentiation to automatically obtain required derivatives [9].

6 On-line application

The method discussed in this paper is being applied to a 700 MW coal fired power plant. The model used there is significantly more rigorous, esp. with respect to the thermal stress models, see also [10]. It considers additional important components like the furnace, economizer, superheaters, headers, spray water injection and long pipes.

A number of new challanges arise in an on-line application, ranging from repeaded update of the solution in a control loop, on-line identification of transient initial states and numerical robustness, through issues of the integration with the control system, supervision of the optimization and fall-back strategy, up to user interface design and acceptance by operating staff. It is not in the scope of this paper to discuss these issues. More details about implications on the optimization finds in [8]. More application specific information is given in [11].

7 Conclusions

This paper discusses the application of a drum boiler simulation model taken from the literature to dynamic optimization. The model is formulated in Modelica exploiting the new Modelica.Media and Modelica.Fluid base libraries. The mathematical model transformation performed automatically by the Dymola tool is outlined.

Based on the model, the optimal control problem is specified with an objective function and constraints. The optimal control problem is solved as large-scale non-linear optimization problem. For the example, the startup time can be reduced from 45 minutes to less

than 30 minutes, while constraint on thermal stress is fulfilled better.

The new Modelica.Fluid and Modelica.Media libraries allow the formulation of thermo-fluid models from a physical point of view. The object-oriented design supports a flexible design. Many mathematical details that traditionally had to be treated by a human modeler have been automated, making the modeling more efficient and allowing better re-use. Such details include the treatment of high-index DAEs, non-linear state-space transformations and the automatic detection of flow directions for multiple inter-connected fluid ports, including support for flow reversal.

The new libraries are applicable to on-line optimization. While making the job for human modelers easier, the libraries pose high requirements on a Modelica translator for the generation of efficient simulation code. Most important are the analytical treatment of connection equations and the elimination of common sub-expressions for multiple property evaluations at the same point, e.g. in inter-connected components. Dymola offers these features.

Reversible flows must be used carefully in optimization models as they are treated discontinuously, causing problems for the sensitivity analysis. A model allowing for reversible flow does not cause problems if flow does not revert, however, switching may also occur due to small numerical errors if flows are zero. For models with known unidirectional flows, one would like to be able to fix the direction in the model and to enforce correct results with optimization constraints instead. An other point that might be improved by tool vendors is the automatic differentiation of medium functions, as required for DAE index reduction and for the calcualtion of model sensitivities. Currently derivatives have to be provided explicitly together with medium functions in a model library.

References

[1] K.J. Åström and R.D. Bell. Drum-boiler dynamics. *Automatica*, 36:363–378, 2000.

[2] Dynasim AB. Dymola: Dynamic Modeling Laboratory. http://www.dynasim.se.

[3] H. Elmquist, S.E. Mattsson, and H.Olsson. New methods for hardware-in-the-loop simulation of stiff models. In *Proceedings of the 2nd International Modelica Conference*. Oberpfaffenhofen, Germany, March 2002.

[4] H. Elmqvist, H. Tummescheit, and M. Otter. Modeling of thermo-fluid systems – Modelica.Media and Modelica.Fluid. In *Proceedings of the 3rd International Modelica Conference*. Linköping, Sweden, November 2003.

[5] W. Wagner et. al. The IAPWS Industrial Formulation 1997 for the thermodynamic properties of water and steam. *Transactions of the ASME*, 122:150–182, 2000.

[6] R. Franke. Formulation of dynamic optimization problems using Modelica and their efficient solution. In *Proceedings of the 2nd International Modelica Conference*. Oberpfaffenhofen, Germany, March 2002.

[7] R. Franke, E. Arnold, and H. Linke. HQP: a solver for nonlinearly constrained large-scale optimization. http://hqp.sourceforge.net.

[8] R. Franke, K. Krüger, and M. Rode. Nonlinear model predictive control for optimized startup of steam boilers. In *Proceedings of the GMA-Kongress 2003*. Baden-Baden, Germany, June 2003.

[9] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, volume 19 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 2000.

[10] K. Krüger, R. Franke, and M. Rode. Optimization of boiler startup using a nonlinear boiler model and hard constraints. In *Proceedings of the 15th International Conference on Energy, Costs, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS 2002)*, volume II, pages 1310–1318. Berlin, Germany, July 2002.

[11] M. Rode, R. Franke, and K. Krüger. Optimize IT model predicteve control for boiler start-up (BoilerMax). *ABB Review*, 3:30–36, 2003.

[12] The MathWorks, Inc. Simulink: for model-based and system level design. http://www.mathworks.com.