Object-Oriented Modeling and Optimal Control: A Case Study in Power Plant Start-Up

Casella, Francesco; Donida, Filippo; Åkesson, Johan

2011

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Object-Oriented Modeling and Optimal Control: A Case Study in Power Plant Start-Up

Francesco Casella* Filippo Donida** Johan Åkesson***

* Dipartimento di Elettronica ed Informazione, Politecnico di Milano, Milano, Italy (e-mail: casella@elet.polimi.it).
** Dipartimento di Elettronica ed Informazione, Politecnico di Milano, and Motus s.r.l., Milano, Italy (e-mail: donida@motus-it.com)
*** Department of Automatic Control, Lund University, Lund, Sweden and Modelon AB, Lund, Sweden (e-mail: johan.akesson@control.lth.se)

Abstract: Modeling and optimization of complex systems traditionally have required significant programming efforts in order to encode the model dynamics, the cost functions and the constraints in a format suitable for state of the art numerical algorithms. The availability of dedicated languages for physical modeling has made the design process simpler, but often tools have been limited to a particular optimization algorithm. In this paper, we present a case study where a combined cycle power plant model has been developed using first principles in the modeling language Modelica. Based on the model, an optimal control problem for the start-up of the power plant has been formulated in the Optimica extension and solved using the open source software JModelica.org. The results demonstrate how high-level modeling languages and tools can be used to bridge the gap between the engineering need for intuitive description formats and the interfaces of efficient numerical algorithms.

Keywords: Nonlinear dynamic optimization, Power plant control, Object-oriented modeling, Modelica, Optimica

1. INTRODUCTION

Optimization of industrial processes has been a strong trend during the last decades, resulting in a large body of research in the field as well as significant industrial adoption. Applications of optimization techniques range from plant-wide set-point optimization to improvement of economic performance, up to on-line Model Predictive Control (MPC) to track set-points and to reject disturbances. In the quest for improved performance to meet increasing competition, increasingly rigorous, and thereby complex, models are employed when formulating and solving optimization problems. This trend has increased the effort required to encode models suitable for optimization. Also, it is typical that existing tools use proprietary description formats which limits portability and reuse of models for different purposes.

The approach demonstrated in this paper is instead based on the non-proprietary object-oriented modelling language Modelica and on the open source platform JModelica.org, Åkesson et al. (2010). This approach overcomes some of the difficulties often associated with development of models to be used for optimization purposes by i) relying on an open language supported by several tools for physical system modeling, ii) using a high-level language, the Optimica extension, for formulation of the dynamic optimization problem, and iii) using a computational platform allowing the same model to be simulated, optimized and analyzed in a fully integrated way.

Modelica models have been used in the past to solve optimal control problems in the field of power plant control, see, e.g., Krueger et al. (2004), requiring extensive manual coding to interface the simulation code generated by the Modelica compiler with nonlinear optimization codes. The main contribution of this paper is to demonstrate the applicability of fully integrated object-oriented modelling and optimization techniques and tools to solve such problems.

The plant model, albeit simple enough to be handled by numerical optimization methods, is entirely derived from first principle laws such as mass, energy and momentum balance equations, or as the equations of state of the working fluid. The model is built according to the object-oriented principle: single components are described in terms of a-causal differential-algebraic equations, then the components are joined by means of additional a-causal connection equations to form the complete system model. The optimization problem addressed in this case study is the minimum time start-up of a combined-cycle power plant, subject to a constraint on the level of stress in the steam turbine shaft, which determines the consumption of residual life-time of the component. The goal is to increase the availability and flexibility of such plants without incurring in the additional costs caused by the life-time...
reduction of one of the most expensive components in the
system, i.e., the steam turbine.

The paper is structured as follows: Section 2 gives some
background information about object-oriented modelling
and dynamic optimization; Section 3 describes the plant
model, while in Section 4 the optimal start-up problem is
formulated and the numerical results are briefly discussed.
Finally, Section 5 summarizes the results and gives per-
spectives for future work.

2. BACKGROUND

2.1 Physical modeling and Modelica

The Modelica language, The Modelica Association (2010),
is dedicated to modeling of complex heterogeneous physi-
cal systems. The language is object-oriented and imple-
ments concepts such as classes, components (class in-
stances), and inheritance. In addition, Modelica supports
declarative equation-based modeling. This formalism en-
ables the user to state declarative differential and alge-
braic relations, without the need to manually convert
the model to an ordinarily differential equation (ODE) by
solving for the derivatives, a procedure that is commonly
required when using block-based modeling systems such as
Simulink. Accordingly, the underlying mathematical for-
malism of Modelica models is that of differential-algebraic
equations (DAEs). In addition to declarative equations,
Modelica offers means to model, explicitly, a-causal physi-

cal interfaces which can be used to connect components
together. Modelica also implements support for hybrid
constructs such as conditional and instantaneous equa-
tions, boolean and integer equations and sampled systems,
which are however outside the scope of the work presented
in this paper.

Modelica is an open language, and is continuously updated
and maintained. Several tools are supporting the language,
including Dymola, Dassault Systemes (2010), Simulation
X, ITI GmbH (2010), and MapleSim, Maplesoft (2010).
Also, a freely available standard library provides basic
models in different fields, including electronics, multi-body
systems, and thermodynamics.

2.2 Dynamic optimization

In the 50s and 60s, two different methods for solving
optimization problems with constraints in the form of
dynamic systems were presented. In 1957, Bellman formu-
lated the celebrated principle of optimality, and showed
that dynamic programming was applicable to a broad
range of applications, Bellman (1957). Further, Pontryagin
and co-workers presented the maximum principle in 1962,
Pontryagin et al. (1962).

However, both dynamic programming and the maximum
principle have practical drawbacks that make them hard
to apply to large-scale systems. For example, in the presence
of inequality constraints, the activation sequence must be
known a priori. Also, it may be difficult to find initial
guesses for adjoint variables.

In the last two decades, a new family of methods has
emerged to overcome these difficulties. These methods are
referred to as direct methods. Direct methods attempt to
solve dynamic optimization problems by transcribing the
original infinite-dimensional dynamic problem into a finite-
dimensional static optimization problem.

There are two main branches within the family of direct
methods, referred to as sequential and simultaneous meth-
ods, see Binder et al. (2001) for an overview. The sequen-
tial methods rely on state of the art numerical integrators,
typically also capable of computing state sensitivities, and
standard nonlinear programming (NLP) codes. The con-
trols are then usually approximated by piece-wise poly-
nomials, which render the controls to be parameterized
by a finite number of parameters. These parameters are
then optimized. See e.g., Vassiliadis (1993) for details on
sequential methods.

Simultaneous methods, on the other hand, are based on
collocation, and approximate both the state and the con-

trol variables by means of piece-wise polynomials, see
Biegler et al. (2002) for an overview. This strategy requires
a fine-grained discretization of the states, in order to
approximate the dynamic constraint with sufficient accu-

racy. Accordingly, the NLPs resulting from application of
simultaneous methods are very large, but also sparse.
In order to solve large-scale problems, the structure of the
problem needs to be explored. A detailed treatment of
direct collocation methods is given in Betts (2001).

A popular extension of the single shooting algorithms is
multiple shooting Bock and Plitt (1984). In a multiple
shooting algorithm, the optimization interval $[t_0,t_f]$ is
divided into a number of segments. New optimization
variables corresponding to the initial conditions for the
states in each segment, are then introduced. This enables
the dynamics, as well as the sensitivity equations, to be
integrated independently in each segment. In order to
enforce continuity of the state profiles, equality constraints
are introduced in the optimization problem to ensure con-

tinuous state profiles. Extensions of the multiple shooting
algorithm include the DAE case, see Bock et al. (1988),
and application to real-time model predictive control, see
Diehl et al. (2002).

2.3 Optimica

While the Modelica language is well-suited for modeling
of physical systems, it does not natively support formu-
lation of optimization problems. In order to complement
Modelica with language constructs dedicated to optimiza-
tion, the Optimica extension, Åkesson (2008), has been
proposed. Optimica adds to Modelica syntactic and se-

mantic means to encode cost functions, constraints and
what quantities to optimize, which in turn can be used to
formulate optimal control problems, parameter estimation
problems and design optimization problems. In order to
highlight the main features of Optimica, a small example
is considered:

$$\min_{u(t)} t_f$$

subject to the dynamic constraint

$$\dot{x}(t) = v(t), \quad x(0) = 0$$

$$\dot{v}(t) = u(t), \quad v(0) = 0$$

and
JModelica.org is an extensible Modelica-based open source platform for optimization, simulation and analysis of complex dynamic systems. The main objective of the project is to create an industrially viable open source platform for optimization of Modelica models, while offering a flexible platform serving as a virtual lab for algorithm development and research.

The JModelica.org platform is composed of several different software components, including a compiler front-end, code generation modules for C and XML, a run-time system in C providing evaluation-oriented model interfaces, and a Python interface for user interaction. A schematic picture of the platform architecture is shown in Figure 1.

The JModelica.org platform implements a simultaneous optimization method based on Lagrange polynomials on finite element with Radau points, see Biegler et al. (2002) for an overview. The algorithm is implemented in C and relies on the API functions for the JModelica.org run-time system. In particular, Jacobians and sparsity information are exploited in order to increase the efficiency of the algorithm. The non-linear program resulting from collocation is then solved by the solver IPOPT, Wächter and Biegler (2006).

The scripting language Python, Python Software Foundation (2009), is used to provide a user interaction environment in JModelica.org. This choice is motivated by the availability of Python packages for numerical and scientific computations, as well as for visualization. The JModelica.org compilers are interfaced with Python, as are the API run-time functions for model equation evaluation.

For a full description of the JModelica.org platform, see Åkesson et al. (2010).

### 3. COMBINED-CYCLE PLANT MODELING

The case study considered in this paper is the minimum time start-up of a combined-cycle power plant. The main limiting factor in this transient is the thermo-mechanical stress on the outer surface of the steam turbine rotor. This stress is proportional to the difference between the surface temperature, which closely follows the superheated steam temperature due to the relatively high heat transfer coefficient, and the average temperature across the whole body of the rotor, which lags behind due to the rotor’s thermal inertia. The most stressed part is the slice of the shaft corresponding to the first row of blades at the turbine inlet; this is modeled by only considering the axial heat...
flow while neglecting the longitudinal heat flow, which is much less relevant.

The Modelica object diagram of a representative, yet simple, model of the process undet consideration is shown in Fig. 2. For reasons of simplicity, a one-pressure-level generator is considered. The plant data have been derived from those of a three-levels of pressure steam generator used in a previous study by Casella and Pretolani (2006), where the start-up sequence of a combined-cycle plant was studied with a detailed process model, optimized by trial-and-error.

Starting from the lower left, the gas turbine model generates a prescribed flow of exhaust gas at a prescribed temperature. Both quantities are a function of the electrical load input signal and follow the typical behavior of large units with IGV-controlled exhaust temperature: from 100% down to about 50% load, the exhaust temperature is kept constant by reducing the exhaust gas flow via the IGVs; for lower load levels, the IGVs cannot be closed further, so the exhaust gas flow is approximately constant, and the exhaust temperature decreases. The typical curves are well-approximated by piecewise linear functions, which are smoothed out at the corner points in order to avoid numerical problems with the optimization algorithm, that requires smooth equations. Since the fundamental dynamics of the gas turbine is much faster than that of the steam cycle, an algebraic model is employed for simplicity.

The turbine exhaust gases enter the hot side of a counter-current heat exchanger, and are then discharged to the atmosphere. Each segment of the hot side is modeled by a dynamic energy balance equation, accounting for the energy stored in the gas and for the heat transfer to the cold side. Changes in mass storage and pressure losses are neglected, since their effects are irrelevant for this application. The convective heat transfer to the cold side is then modeled by components that are connected to the heat ports of both sides. The economizer and the superheater are also modeled by dynamic energy balance equations, while the changes in mass storage and the pressure losses are also neglected. The evaporator model instead requires dynamic mass and energy balance equations, and is written assuming thermodynamic equilibrium between the liquid and the vapour phase. For all the three water/steam components, the temperature of the surrounding steel walls is assumed to be the same as the fluid’s, and the corresponding heat storage is modeled accordingly in the energy balance equation. This assumption is reasonable, since the heat transfer coefficients are large enough to allow neglecting the temperature differences between the fluid and the wall, at least in a first approximation model.

The feedwater system is approximated by a prescribed flow rate source with fixed temperature, driven by the drum level controller. A PI level controller is included in the model, to stabilize the level dynamics and keep the void fraction around 0.5. There is no need of optimal level control for this kind of application, so the model to be optimized can include the closed-loop level dynamics, comprehensive of the PI feedback loop.

Finally, the superheated steam enters the steam turbine, which is modeled as an expansion to the condenser pressure, assuming a constant isentropic efficiency. The turbine model also exposes a thermal port, corresponding to the surface where the inlet steam comes into contact with the turbine rotor, exchanging heat by convective heat transfer. This port is in turn connected to a thermal model of the shaft, given by Fourier’s heat equation, discretized by the finite difference method. This thermal model allows to compute the surface and average temperatures of the rotor shaft, and thereby the surface stress level, which is proportional to the difference between these two temperatures.

In order to keep the model as simple as possible, to ease the convergence of the optimization algorithms, the following additional assumptions have been made. First of all, the gas, the subcooled liquid and the superheated steam are modeled as incompressible fluids with constant $c_p$, while the properties of saturated liquid and vapour in the evaporator are approximated by low-order polynomials. Secondly, lumped-parameter models are assumed for the heat exchanger segments, with just one temperature state for each side. The finite-difference turbine rotor model has 8 nodes and six states, in order to accurately describe the temperature profile that determines the surface stress level. The resulting nonlinear model has 15 state variables and 141 algebraic variables. The most severe non-linearity is given by the change of derivative of the gas turbine curves at 50% load. Other milder nonlinearities are caused by the fluid equations of state and by the flow-specific enthalpy products in the energy balance equations.

4. OPTIMAL START-UP PROBLEM SET-UP AND NUMERICAL RESULTS

As already stated in the Introduction, the goal of the optimization problem is to reach the full load level as fast as possible, while limiting the peak stress value on the rotor surface; this peak value determines the life-time consumption during the entire start-up phase, under the hypothesis that the stress curve follows a simple cycle, i.e., it is monotonically non-decreasing up to the peak value, and then monotonically non-increasing after the peak has been reached. Since the steam cycle is assumed to operate in a pure sliding pressure mode, the full load state is reached when the load level of the turbine $u(t)$ (which is the control variable) has reached 100% and the normalized value of the evaporator pressure $p_{ev}$ has reached the target reference value $p_{ref}^T$. A Lagrange-type cost function,
penalizing the sum of the squared deviations from the target values, drives the system towards the desired set-point \((p_{\text{evap}}, u) = (p_{\text{ref} \text{evap}}, 1)\) as quickly as possible. It turns out that the relative weight of the two deviations is not that important, as long as normalized values having the same order of magnitude are employed.

The main limiting factor in the start-up problem is the constraint acting on the thermal stress in the steam turbine, \(\sigma(t)\). In addition, inequality constraints are imposed on the rate of change of the gas turbine load: on one hand, the load is forbidden to decrease, in order to avoid as much as possible the multiple cycling of the stress level during the transient; on the other hand, it cannot exceed the maximum rate prescribed by the manufacturer.

The start-up optimization problem is then formally stated as:

\[
\begin{align*}
\min_{u(t)} & \int_{t_0}^{t_f} (p_{\text{evap}}(t) - p_{\text{evap}}^{\text{ref}})^2 + (u(t) - 1)^2 \, dt \\
\text{subject to} & \quad \sigma(t) \leq \sigma_{\text{max}} \\
& \quad \dot{u}(t) \leq du_{\min} \\
& \quad \dot{u}(t) \geq 0
\end{align*}
\]  

and to the DAEs representing the plant dynamics. The initial state for the DAE represents the state of the plant immediately after the connection of the electric generator to the grid. It is assumed that the initial rotor temperature is uniform and equal to the steam temperature, which roughly corresponds to a warm start-up case. A more realistic initial distribution would require modelling the boiler and turbine start-up phases, which is beyond the scope of this paper.

The optimization problem was solved using the direct collocation method in JModelica.org. The algorithm was set up to use 40 collocation points and cubic interpolation polynomials to approximate the state profiles. Also, the control variable was approximated by a piecewise linear function with as many segments as finite elements.

The resulting non-linear program had 22663 variables and was solved by IPOPT in 107 seconds on a PC with a 2.60Ghz Intel Core 2 Duo processor.

The nonlinear optimization algorithm requires an initial trajectory of all the problem variables as a first guess for the nonlinear solvers. In this case, a reference simulation was used, where the gas turbine load was increased from the initial value of 15% to the final value of 100% in 10000 seconds.

The result of the optimization is shown in Figure 3. During the first 200 seconds, the gas turbine load is increased at the maximum allowed rate and the stress builds up rapidly, until it reaches the target limit. Subsequently, the load is slowly increased, in order to maintain the stress level approximately constant at the prescribed limit. When the 50% load level is reached, further increases of the load do not cause additional increase of the gas exhaust temperature, and therefore cause only small increases of the steam temperature, due to the steam generator dynamics. It is then possible to resume increasing the load at the maximum allowed rate, while the stress level starts to decrease. The full load is reached at about 4200 s.

Figure 4 shows the corresponding temperature transients in the turbine shaft: the upper curve shows the live steam temperature, while the middle and low curves show the turbine rotor surface and mean temperatures.

This result is qualitatively comparable to the results obtained by Casella and Pretolani (2006) with a detailed...
plant model, even though the mismatch in the initial conditions of the rotor temperature distribution and the absence of the intermediate and low pressure circuits do not allow to exactly superimpose the results.

5. SUMMARY AND CONCLUSIONS

In this paper, we have shown how the high level modeling languages Modelica and Optimica have been used to solve a start-up optimization problem for a combined-cycle power plant. The open source platform JModelica.org has been used to obtain computational results by means of a direct collocation algorithm.

The use of high-level design tools has enabled short design iteration cycles, where tuning of the model and the optimization problem, rather than the details of their implementation, has been put into focus. Since the power plant model relies on library components, it is well suited for reuse and modular extension with more rigorous models regarding, e.g., the thermodynamic media properties.

Future enhancements to this work include using more accurate 1D models of the heat exchangers and of the fluid properties, including the boiler start-up and turbine start-up phases in the optimization, and using the optimization algorithm for NMPC control, possibly by closing the loop on a more accurate plant model to evaluate the behaviour of the control system in a realistic set-up.

REFERENCES


