ISSN 0280-5316
ISRN LUTFD2/TFRT--5900--SE

Optimization of the Start-up Procedure of a Combined Cycle Power Plant

Alexandra Lind
Elin Sällberg

Lund University
Department of Automatic Control
June 2012

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
June 2012

Document Number
ISRN LUTFD2/TFRT--5900--SE

Author(s)
Alexandra Lind
Elin Sällberg

Supervisor
Johan Åkesson, Dept. of Automatic Control, Lund University, Sweden
Stéphane Velut, Modelon, Sweden
Tore Hägglund, Dept. of Automatic Control, Lund University, Sweden (Examiner)

Sponsoring organization

Title and subtitle
Optimization of the Start-up Procedure of a Combined Cycle Power Plant (Optimering av uppstart av gaskombikraftverk)

Abstract
In the electricity market of today, with increasing demand on electricity production on short notice, the combined cycle power plant stands high regarding fast start-ups and efficiency. In this thesis it has been shown how the start-up procedure of a combined cycle power plant can be optimized using JModelica.org, proposing a way to minimize the start-up time while maximizing the power production during start-up. The physical models have been developed in Modelica, adapted to suit optimization purposes and extended to optimization problems with the Optimica extension. Constraints keeping the lifetime consumption of critically stressed components under control have been limiting factors and the plant models have been successfully optimized to full load.

Keywords
Combined Cycle Power Plants, Start-up, Dynamic optimization, Optimica, Control, Modelica, Modeling

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-87

Recipient's notes

Security classification

Acknowledgements

We would like to thank our main supervisor Stéphane Velut at Modelon AB for sharing his experiences with dynamic optimization and JModelica.org. Thank you for your enthusiastic support and guidance, we have really enjoyed working with you!

Our supervisors Johan Åkesson, Modelon AB / Department of Automatic Control at Lund University, and Per-Ola Larsson, Modelon AB, shall also be thanked for valuable advice on optimization issues and JModelica.org tool support.

Many thanks to our supervisors Stephanie Gallardo Yances and Kilian Link at Siemens AG, Energy Sector, Erlangen, Germany. You have been a great support in our thesis with your process knowledge of the combined cycle power plant. We have felt very welcome in your team!

Together with our supervisors an article about this project has been written for the Modelica Conference 2012, see [1]. Thank you for your support during this process.

Finally we would like to thank our examiner Tore Hägglund at the Department of Automatic Control at Lund University.

The German Ministry BMBF has partially funded this work (BMBF Förderkennzeichen: 01IS09029C) within the ITEA2 project OPENPROD (http://www.openprod.org). Modelon's contribution to this work was partially funded by Vinnova within the ITEA2 project OPENPROD (dnr: 2010-00068).

Abstract

In the electricity market of today, with increasing demand on electricity production on short notice, the combined cycle power plant stands high regarding fast start-ups and efficiency. In this thesis it has been shown how the start-up procedure of a combined cycle power plant can be optimized using JModelica.org, proposing a way to minimize the start-up time while maximizing the power production during start-up. The physical models have been developed in Modelica, adapted to suit optimization purposes and extended to optimization problems with the Optimica extension. Constraints keeping the lifetime consumption of critically stressed components under control have been limiting factors and the plant models have been successfully optimized to full load.

Keywords: Combined Cycle Power Plants, Start-up, Dynamic optimization, Optimica, Control, Modelica, Modeling

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In a time when the production from renewable energy sources is steadily growing the demand for complementary electricity production on short notice is high. Large fluctuations during the day require power generators to react quickly to maintain the balance between demand and production. Deregulation of the electric power market also allows private investors to install power plants and supply power to the grid, which has increased the competition on the electricity market. The requirements between demand and supply have to be maintained while offering electricity at the lowest cost.

When considering fast start-ups and efficiency, the combined cycle power plant stands high in comparison with other electricity production methods. In this thesis, the start-up procedure of a combined cycle power plant is studied. The aim is to minimize the start-up time while keeping the lifetime consumption of crucial power plant components under control and maximizing the amount of power output produced.

Several previous studies that deal with optimization of the start-up of combined cycle power plants have been made. In Casella and Pretolani, [2], optimization with a trial-and-error method is presented where the results are obtained by simulating simplified Modelica power plant models. The study has been carried out to develop simplified models that can be used to automatically compute the optimal transients with an optimization software and the models were based on the Modelica ThermoPower library, see Casella and Leva, [3]. A model-based approach for optimizing the gas turbine load trajectory has been studied in Casella et al., [4]. A simplified model is developed based on interpolated locally identified linear models and the procedure aims at deriving the gas turbine load profile described by a parameterized function. A minimum-time problem is solved to determine the parameters of the parametrized function. Shirakawa et al. proposed an optimal design method combining dynamic simulation and non-linear programming in [5].

1.2 Aim of Thesis

The aim of the master thesis is to make the start-up procedure of a combined cycle power plant more efficient, with respect to the start-up time and power production, while limiting the thermal stress in the heat recovery steam generator.

The plant models are described in the object-oriented modeling language Modelica. All models are developed by Siemens AG, Energy Sector, in cooperation with Modelon AB, and are based on elementary models from first principle equations of mass and energy. The physical models have been developed using the commercial Modelica simulation environment, Dymola [6], and they have been adapted to suit optimization purposes. The tool used for optimization is like proposed in [7] the Modelica based open source platform JModelica.org.

1.3 Siemens AG Energy Sector

Siemens AG was founded in 1847 and is today, with its 360 000 employees around the world, occupying leading markets and technology positions within the areas of Energy, Healthcare, Industry, and Infrastructure & Cities. The Energy Sector is one of the world's leading suppliers of a wide range of products, solutions and services within Energy Service, Fossil Power Generation, Oil and Gas, Power Transmission, Solar and Hydro and Wind Power. Siemens is the leading service partner of approximately one-fifth of all large-scale and industrial power plants worldwide. Much of the energy service is focused on increasing the efficiency and capacity of existing power plants [8].

1.4 Modelon AB

Modelon AB is specialized in providing solutions, services and technology for the research and development of dynamic systems and offers services in physical modeling, simulation and optimization, and model-based control design. Modelon is active in providing technology and solutions in the Modelica language and is provider of commercial Modelica libraries as well as Modelica language development. Trademarks of Modelon AB include Optimica, FMI Toolbox and JModelica.org, where JModelica.org is the tool used in this project for dynamic optimization. The head office of Modelon is in Lund, Sweden.

Chapter 2

Background on Combined Cycle Power Plants

2.1 Thermodynamics

In thermodynamics, a system refers to a fluid in a confined space that is characterized by its state, which for pure substances is defined by the two properties temperature and pressure. A thermodynamic cycle is a system that goes through a sequence of different states, exchanging heat with its environment, and finally ends up in its initial state. In a thermodynamic cycle a system can also perform work on its surroundings. Such a system is called a heat engine. In theory, the most efficient cycle for converting a given amount of heat energy into work is the Carnot cycle.

An ideal evaporation and condensation cycle can be seen in Figure 2.1 where the heat, Q_{in}, is applied to the system evaporating the water from point 1 to point 2. While the water is evaporating the temperature T does not rise but the entropy S increases from S_1 to S_2. Between point 2 and 3 the evaporated water expands without any heat infusion, resulting in a temperature decrease. Here work is performed on the surrounding. At point 3 a condensation of the steam begins. Q_{loss} is taken from the system when the water condensates without a temperature change. From point 4 to point 1 the pressure builds up again, using for example a pump, and the temperature increases. Here work is supplied to the system. At point 1 the evaporation starts again and the cycle is closed.

The efficiency of a thermal process is defined by $\eta_c = W/Q$, where W is the work attained and Q is the heat transferred to the system. The efficiency of an ideal Carnot cycle is defined as the Carnot efficiency and it can be shown to be:

$$\eta_c = 1 - \frac{T_{min}}{T_{max}}, \tag{2.1}$$

where
η_c = Carnot efficiency,
T_{max} = Temperature of the energy supplied, [K],
T_{min} = Ambient temperature, [K].

The efficiency of an actual process is always lower than the Carnot efficiency due to exergetic and energetic losses. The efficiency of a system can be improved in three

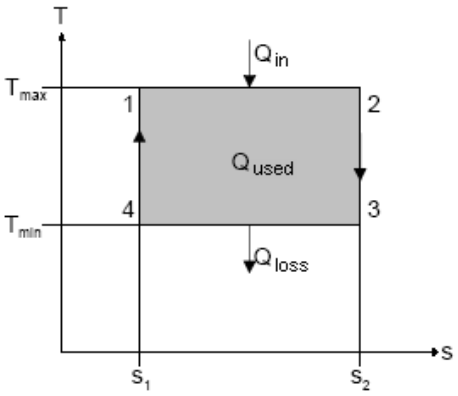

Figure 2.1: The Carnot cycle.

ways, either by raising the maximal temperature in the cycle, releasing the waste heat at a lower level or by improving the process in order to minimize the internal energetic losses [9].

In a process with only one thermodynamic cycle it seems impossible to make all these improvements at once. The goal could thus be achievable if one combines two cycles, one with high process temperatures and one with a low end temperature. An example of such a process is a combined cycle power plant [9].

2.2 The Combined Cycle Power Plant

A combined cycle is a combination of two thermal cycles in one power plant, where the topping cycle is the cycle operating at the higher temperature and the bottoming cycle is the cycle operating at a lower temperature level. The waste heat that the topping cycle produces is used in the process of the bottoming cycle and the efficiency is therefore higher for the combined cycle than that of one cycle alone. In the commercial power generation of today the combined cycle power plants consist of a gas topping cycle and a steam/water bottoming cycle [9]. The net efficiency world record of a CCPP is 60.75% and was reached by Siemens in 2011 [10].

2.2.1 Basic Principles

The basic principles of a combined cycle power plant can be seen in Figure 2.2. The plant consists mainly of three parts, the gas turbine (GT), the heat recovery steam generator (HRSG), also called boiler, and the steam turbine (ST).

The principles of the GT are quite simple. First ambient air is drawn into the turbine where it is compressed and used to burn a combustion medium. Hot gas with a temperature of about 1000°C is produced and is expanded in the turbine where it is used to drive both the compressor and the generator. The exhaust gas leaves the turbine at ambient pressure and a temperature of about $450 - 650$°C depending on the design of the turbine. The GT is the key component of the power plant, generating about 60-70 % of the total electricity output.

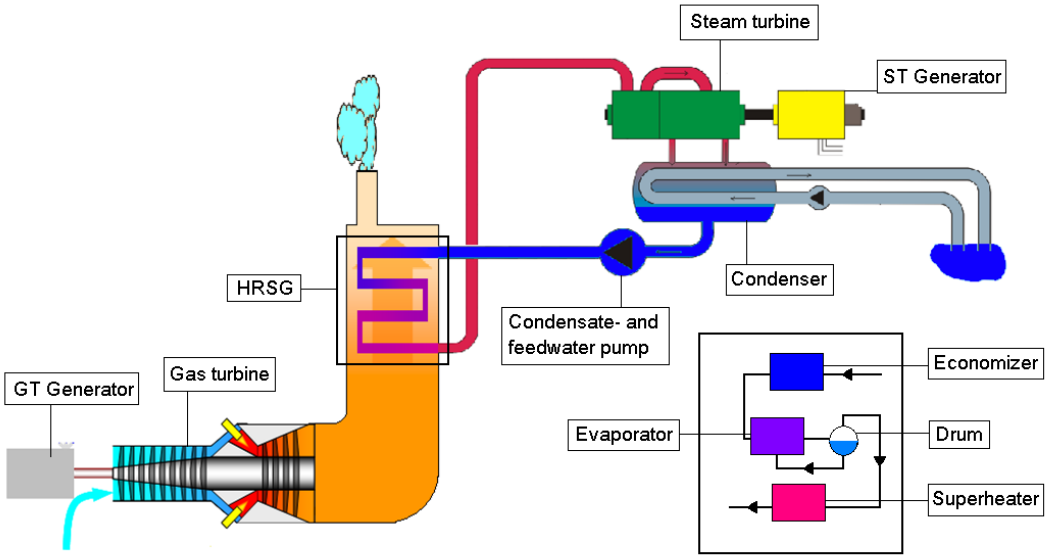

Figure 2.2: The CCPP.

The most important part of a CCPP is the HRSG. In this part of the plant the heat of the exhaust gas from the GT is used to produce hot steam which will drive the ST. The HRSG consists mainly of three different components, the economizer, the evaporator and the superheater. In the economizer the exhaust gas at the lowest temperature is used to heat water. Subcooled water leaves the economizer and enters the evaporator. Here the water is circulated and hotter exhaust gas from the GT is used to evaporate the water. A mix between water and steam enters a drum where it is separated and saturated water vapour leaves the drum and enters the superheater, where finally the hottest exhaust gas is used to heat the steam to its maximum temperature. After the steam has been dried in the superheater it is ready to enter the ST where it expands and generates power. The ST stands for the remaining 30-40 % of the total power output. Finally the exhaust steam from the ST is condensed and led back to the economizer where the heat recovery process starts over. Some of the energy generated in the ST is used to pump the feedwater back into the HRSG [9].

2.2.2 The Different Cycles

The concept of combining a gas turbine cycle with an HRSG and steam turbine cycle can easier be understood when having the different cycles in an S-T diagram. In the following figures, the bowed curve shows the border to the two-phase region, i.e under the curve is the region where the medium is a mix between steam and water. In Figure 2.3 the area above the curve where the entropy and temperatures are such that the medium is solely water, has been marked. The area above the curve where the medium is solely steam has as well been marked in Figure 2.3. Above the maximum value of the curve the pressure is so high that there is no longer a clear difference between the two phases. Figure 2.3 shows the Carnot cycle described above working in the two phase region.

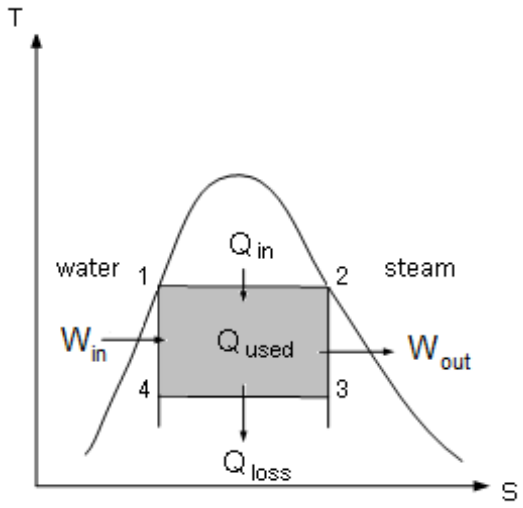

Figure 2.3: The Carnot cycle in the two phase region.

An ideal process including the ST and the HRSG can be described with a slightly modified Carnot cycle called the Rankine cycle, see Figure 2.4. The evaporator works in the two phase region, evaporating the water from point 1 to 2. The superheater heats up the steam from point 2 to 3 and work output is achieved when the steam expands in the ST from point 3 to 4. The steam is completely condensed from point 4 to 5 and work is added when the water is compressed by the pump from 5 to 6. Finally the economizer heats up the water from point 6 to 1. The process differs from the Carnot cycle described above in Figure 2.3 in two ways. One of them is that the water condensates completely between point 4 to 5. There are two reasons why this is necessary; first of all it is very difficult to only partially condensate wet steam, secondly it is much harder to compress a mix between water and steam. Another difference from the Carnot cycle is the superheating process between point 2 and 3. The superheating is beneficial because a higher temperature gives higher efficiency and dry steam is necessary to not damage the blades of the ST.

The ideal process for a GT is called a Bryton cycle, see Figure 2.5. From point 1 to 2 the air is compressed. From point 2 to 3 the air is heated with some kind of fuel combustion. Between 3 and 4 the air is expanded without heat infusion and work is achieved. From point 4 to 1 the air is cooled, i.e we loose the exhaust heat from the turbine.

The heat that is lost between point 4 and 1 in the Bryton cycle is the heat that is used as a heat source in the Rankine cycle, i.e the heat that is used in the HRSG to warm up, evaporate and superheat the water and steam, see Figure 2.6. This way of combining two different thermodynamic cycles is what makes a combined cycle so efficient.

2.2.3 Additional Features

A number of improvements can be made to the combined cycle power plant in order to reach higher efficiency. The power plant described above is a simple single-pressure

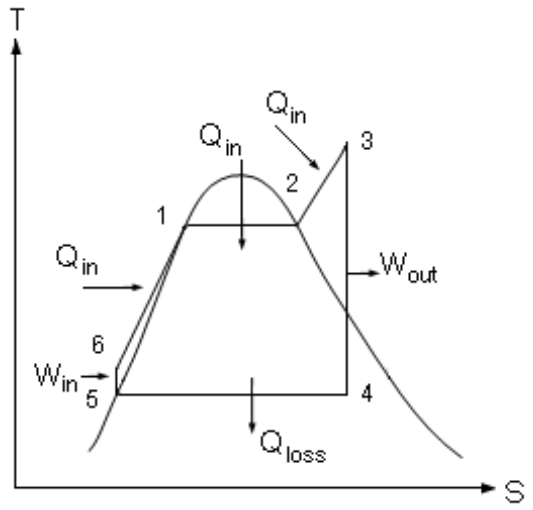

Figure 2.4: The Rankine cycle.

cycle. The HRSG can, however, work at different pressure stages to reach higher efficiency. The more heat recovered, the higher efficiency of the plant. Normally CCPPs are constructed with three pressure stages: high pressure (HP), intermediate pressure (IP) and low pressure (LP).

Another type of improvement is the so called reheating. After the expansion of the HP steam in the ST, some steam at IP is heated again before total expansion in the ST. The reheater takes more exergy out of the HRSG and increases the ST enthalpy drop resulting in a higher ST output. The additionally gained output can be seen in Figure 2.7. The usable energy is the coloured area which is clearly increased thanks to the reheating. Reheating also results in a reduction in the moisture content, since the last part of the ST expansion line is moved further to the right in the *S*-*T* diagram.

2.2.4 The Start-up Procedure

The start-up of a CCPP is normally scheduled as follows.

1 The GT is first accelerated to full speed, no load, and it is synchronized to the grid.

2 The load of the GT is increased and the boiler starts producing steam. The generated steam is not led to the ST but is bypassed to a condenser.

3 When the steam quality is high enough, the bypass valve is slowly closed and the steam can drive the ST.

Depending on how long the plant has been shut down, different kinds of start-ups are considered. A start-up from a stand-still of more than 17.5 hours is defined as a cold start, a start-up from a stand-still of 7 to 17.5 hours is defined as a warm start and if the plant has been shut down for less than 7 hours the start-up is defined as a hot start, see Table 2.1. The warmer the plant is when beginning a new start-up, the faster it is possible to reach full capacity.

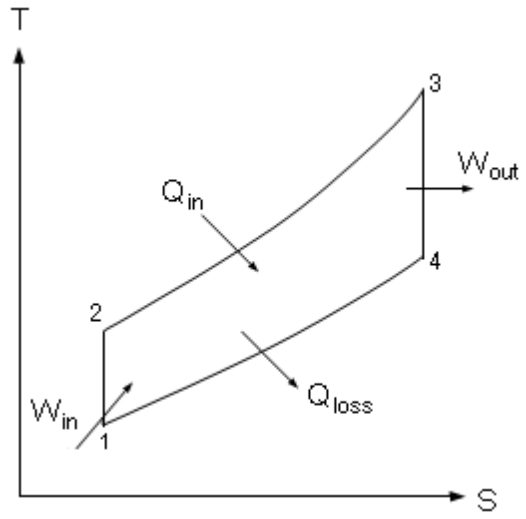

Figure 2.5: The Bryton cycle.

Table 2.1: Hot, warm and cold starts of a CCPP are defined from the time of the stand-still.

cold	warm	hot
$t > 17.5$ h	$7 \leq t \leq 17.5$ h	$t < 7$ h

The different kinds of start-ups are defined with respect to the temperature and pressure in the boiler where one of the most critical components during start-up, the drum in the evaporator, is located. During the second phase of the start-up, fast temperature changes resulting in high thermal stress as well as mechanical stress from high pressures can easily damage the drum if these properties are not controlled. The lifetime of the boiler is directly coupled to the stress it is subject to during the start-up. Since the drum is one of the most expensive components in a power plant, it is of high interest to keep the lifetime of this component as long as possible.

Another critical component is the header in the HP superheater. In the header, steam is divided into different tubes of the superheater, and at the outlet of the header large stresses can occur during the second phase of the start-up.

A reduction of the start-up time of the CCPP is typically achieved by maximizing the loading rates of both turbines while maintaining the lifetime consumption of critically stressed components, such as the drum and the header, under control. Even the ST is subject to large stress constraints, but this occurs in the last phase of the start-up. This thesis focuses on the optimization of the second phase, that is the loading of the GT.

The profile of some important parameters during the start-up procedure can be seen in Figure 2.8. The numbers on the time axis represent the different phases described above.

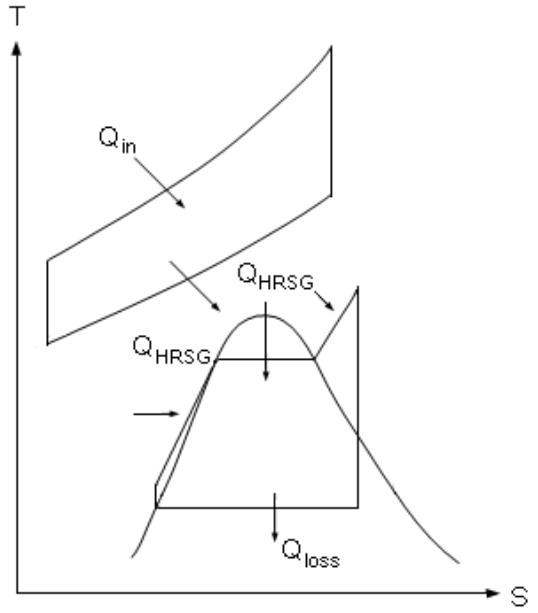

Figure 2.6: The heat recovery between the different cycles.

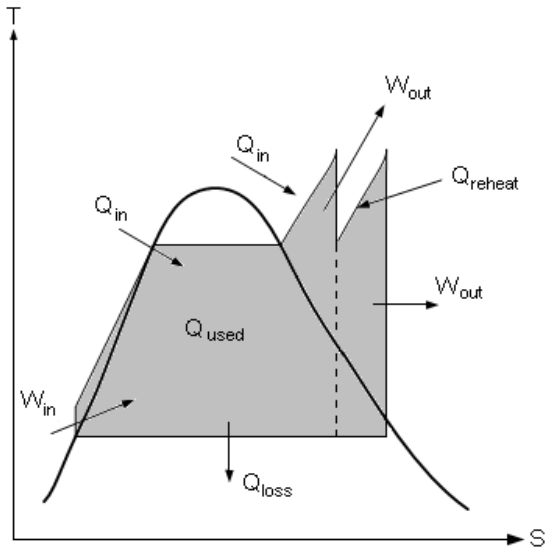

Figure 2.7: Reheating in the CCPP.

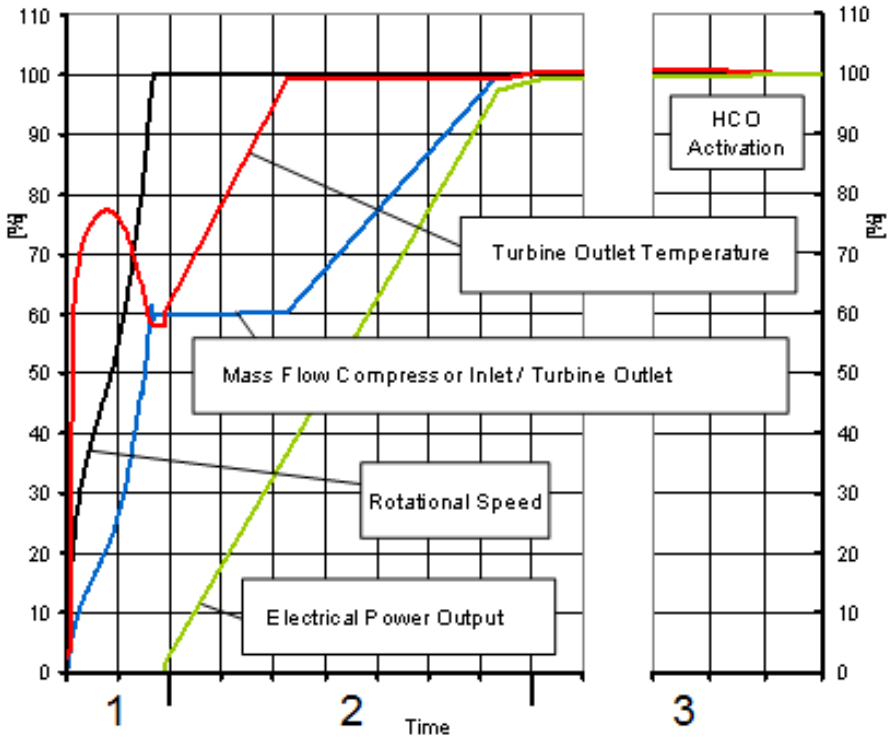

Figure 2.8: The start-up procedure of a CCPP. The numbers on the time axis represent the first, second and third phase of the start-up described above. Provided by Siemens AG, Energy Sector, Germany.

Chapter 3

Dynamic Optimization

3.1 Differential Algebraic Equations

Dynamical processes can often be modeled using *differential algebraic equations* (DAEs) consisting of differential equations describing the mass and energy balances and algebraic equations ensuring thermodynamic and physical relations.

A DAE system is a generalization of an *ordinary differential equation* (ODE) system. ODE systems are given by the general form:

$$x^n = F(x(t), x'(t), ..., x^{n-1}(t), t); \forall t \in [t_0, t_f] \tag{3.1}$$

where n is the order of the system, $x'(t)$ is the first derivative of x and $x^{n-1}(t)$ is the $n-1$ derivative of x. All ODE systems of order n can be transformed to a system of order 1 by introducing $n-1$ additional variables

$$\begin{aligned} z_1 &= x \\ z_2 &= x' \\ &\vdots \end{aligned}$$

and the first order DAE system has the general form:

$$F(x(t), x'(t), t) = 0; \forall t \in [t_0, t_f]. \tag{3.2}$$

The variable x can be divided into

$$x = (z, y)^T \tag{3.3}$$

where the variable z is called the *differential variable* and y is the *algebraic variable*. The algebraic variables are the variables whose derivatives are not a part of the DAE system. Equation (3.2) can thus be written:

$$F(z(t), z'(t), y(t), t) = 0; \forall t \in [t_0, t_f]. \tag{3.4}$$

The differentiation index of the DAE system is n if an ODE is obtained after n differentiations of equation (3.4). A DAE system of index zero is thus equivalent to an ODE system.

3.2 The Dynamic Optimization Problem

In the optimal control problem the aim is to find the control variable(s) that minimizes the *objective* (or *cost*) *function* and fulfil the model equations given by equation (3.1). The optimization variables are the time-dependent control variables and the optimal solution to the system can for example be how to control the process from one stationary state to another stationary state while minimizing the amount of resources spent.

The general DAE optimization problem used in optimal control problems can be stated as in [11]:

$$\min_{u(t)} \varphi(z(t), y(t), u(t), t_f) \tag{3.5}$$

subject to

$$0 = F(z(t), z'(t), y(t), u(t), t), \tag{3.6}$$
$$0 = G(z(t), y(t), u(t), t), \tag{3.7}$$
$$z(0) = z_0, \tag{3.8}$$
$$H_s(z(t_s), y(t_s), u(t_s), t_s) = 0 \text{ for } s \in \{1, ..., n_S\}. \tag{3.9}$$

The upper and lower bounds are given as:

$$z^L \leq z(t) \leq z^U \tag{3.10}$$
$$y^L \leq y(t) \leq y^U \tag{3.11}$$
$$u^L \leq u(t) \leq u^U \tag{3.12}$$
$$t_f^L \leq t_f \leq t_f^U \tag{3.13}$$

where

φ is a scalar objective function,
F are the right hand sides of differential equation constraints,
G are algebraic equation constraints, assumed to be index one,
H_s are additional point conditions at fixed times t_s,
z are differential state profile vectors,
z_0 are the initial values of z,
y are algebraic state profile vectors,
u are control profile vectors,
t_f is the final time.

The objective function φ, that is to be minimized, can have multiple forms of formulations; one given by the *Lagrange* form:

$$\varphi = \int_{t_0}^{t_f} L(z(t), y(t), u(t), t) dt. \tag{3.14}$$

The *minimal time* formulation is formulated if the time interval $t \in [t_0, t_f]$ is unknown. The endpoints t_0 and/or t_f become free optimization variables and the dynamic optimization problem is formulated with point constraints at the time t_f. The objective function is in this case given by:

$$\varphi = \int_{t_0}^{t_f} 1 dt. \tag{3.15}$$

3.3 Collocation Method

The DAE system in Section 3.2 is discretized and converted into a *nonlinear programming* (NLP) problem with a simultaneous approach using a direct collocation method on finite elements [11].

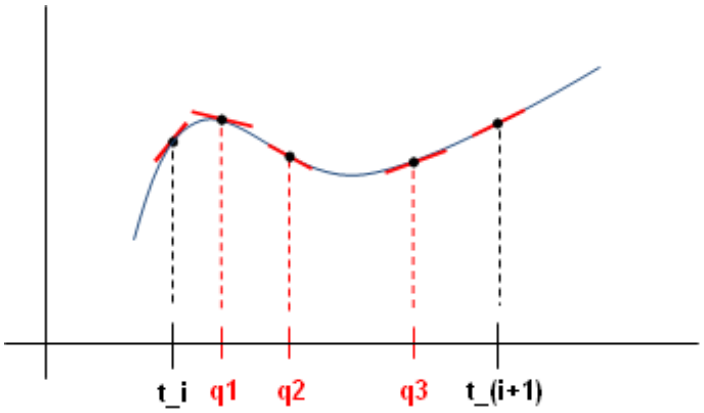

Figure 3.1: An element formed with the collocation method. Three collocation points (q_1, q_2 and q_3) are placed in the element given as the interval $t \in [t_i, t_{i+1}]$.

The NLP is formed by approximating control and state profiles by piecewise polynomials on finite elements $(t_0 < t_1 < ... < t_{n_e} = t_f)$, where n_e is the number of finite elements. A number of Radau collocation points, n_{cp}, are placed in the elements. The number of collocation points in each element corresponds to the stage order of the Runge-Kutta scheme and determines as well the degree of the approximated polynomial, see Figure 3.1. This transforms the DAE to an NLP given by:

$$\min_{x \in \mathbf{R^n}} f(x) \tag{3.16}$$

subject to

$$g_i(x) \leq 0, i = 1, ..., m, \tag{3.17}$$
$$h_j(x) = 0, j = 1, ...l, \tag{3.18}$$
$$x^L \leq x \leq x^U \tag{3.19}$$

where

$$x = (\frac{dz}{dt}_{k,q}, z_k, y_{k,q}, u_{k,q}, t)^T,\ q = 1, ..., n_{cp},\ k = 1, ..., n_e, \tag{3.20}$$
$$f : \mathbf{R^n} \longrightarrow \mathbf{R} \tag{3.21}$$

and m and l are the number of inequality and equality constraints respectively. The conditions $g_i(x) \leq 0$ and $h_j(x) = 0$ define the constraints where g_i gives the inequality constraints and h_j the equality constraints. $g_1, ..., g_m$ and $h_1, ..., h_l$ are defined on $\mathbf{R^n}$ as real valued functions.

The bounds on the differential variables can be enforced at all collocation points by appropriate point constraints and the method makes sure that the initial conditions and

the differential equation are fulfilled at all collocation points. The last Radau collocation point in each element coincides with the last point of the element and this enforces continuity from one element to the next. [11]

See Biegler 2010, [12], for further reading on collocation methods.

3.4 NLP Solver Strategies

3.4.1 Lagrange Multipliers and the Karush-Kuhn-Tucker (KKT) Theorem

An important method used for solving NLP problems of the form (3.16) - (3.19) is the method of *Lagrange Multipliers*. The *Lagrange function* is defined as

$$L(x;\mathbf{u},\mathbf{v}) = f(x) + \sum_{i=1}^{m} u_i g_i(x) + \sum_{j=1}^{l} v_j h_j(x), \qquad \mathbf{u} \leq \mathbf{0}, \mathbf{v} \text{ free.} \tag{3.22}$$

A point $x \in \mathbf{R^n}$ is called feasible if it satisfies (3.16) - (3.19) and a feasible solution $\bar{x}$ to the optimization problem is found if

$$f(x) \geq f(\bar{x}) \text{ for all feasible } x.$$

If $\bar{x}$ is a local minimum the *Karush - Kuhn - Tucker (KKT) Theorem* states that there exists constants u_i and v_j such that

$$\begin{aligned} \nabla_x L(\bar{x};\mathbf{u},\mathbf{v}) &= \mathbf{0}, \\ \mathbf{u} &\leq \mathbf{0}, \\ u_i g_i(\bar{x}) &= 0 \text{ for all } i, \end{aligned} \tag{3.23}$$

assuming that the functions g_i are differentiable and that the functions h_j are continuously differentiable at $\bar{x}$, and that their gradients at $\bar{x}$ satisfy the *Constraint Qualifications (CQ)*

$$\begin{cases} \sum_{i=1}^{m} \lambda_i \nabla g_i(\bar{x}) + \sum_{j=1}^{l} \mu_j \nabla h_j(\bar{x}) = \mathbf{0} \\ \lambda_i \geq 0 \text{ for all } i \end{cases} \Longrightarrow \begin{cases} \lambda_i = 0 \text{ for all } i \\ \mu_j = 0 \text{ for all } j. \end{cases}$$

The Karush - Kuhn - Tucker (KKT) conditions describe the necessary *first-order optimality conditions* and it is necessary that $\bar{x}$ satisfies (3.23) in order for $\bar{x}$ to be a local minimum.

If the objective function f is assumed to be twice continuously differentiable the *Hessian* of the function, $\nabla^2 f(x)$, is defined and the *sufficient conditions* for $\bar{x}$ being a local minimum is that

$$\nabla^2 f(\bar{x}) \text{ is a KKT-point}$$

and that

$$\mathbf{d}^T \nabla^2_{xx} L(\bar{x};\bar{\mathbf{u}},\bar{\mathbf{v}})\mathbf{d} > 0 \tag{3.24}$$

for all $\mathbf{d} \neq 0$ satisfying:

$$\begin{aligned} \nabla g_i(\bar{x})^T \mathbf{d} &\leq 0, i \in I, \\ \nabla h_j(\bar{x})^T \mathbf{d} &= 0, j = 1,...,l. \end{aligned} \tag{3.25}$$

3.4.2 Interior Point Method

With the *interior point method* NLP problems can be solved given an objective function, f, and constraints, g, of the form:

$$\min_{x \in \mathbf{R^n}} f(x) \tag{3.26}$$

subject to

$$g^L \leq g(x) \leq g^U \tag{3.27}$$

$$x^L \leq x \leq x^U. \tag{3.28}$$

g in equation (3.27) includes the constraints in equations (3.16) - (3.19). The functions f and g can be non-linear and non-convex, but should be twice continuously differentiable.

Bounded slack variables are used to replace an inequality constraint by an equality constraint [13]. As an example a new bounded slack variable s_i is defined as $g_i - s_i = 0$ with $g_i^L \leq s_i \leq g_i^U$ replacing equation (3.27) by $c(x) = 0$. The problem can thus be formulated as

$$\min_{x \in \mathbf{R^n}} f(x) \tag{3.29}$$

subject to

$$c(x) = 0 \tag{3.30}$$

$$x \geq 0, \tag{3.31}$$

assuming that all variables have only lower bounds of zero.

The problem is transferred to an interior point method (barrier method) with the auxiliary barrier problem formulation given as

$$\min_{x \in \mathbf{R^n}} \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \ln(x_i) \tag{3.32}$$

subject to

$$c(x) = 0. \tag{3.33}$$

A logarithmic barrier term replaces the constraints where the size of the barrier parameter $\mu > 0$ influences the effect of the barrier term. If any of the variables x_i approach their bound the barrier objective function goes to infinity. As $\mu \to 0$ the optimal solutions converge to an optimal solution of the original problem.

The barrier method uses the technique where the solution strategy is to solve a sequence of barrier problems. Starting with a moderate value of μ and a starting point given by the user the corresponding barrier problem is solved. The value of μ is then decreased and the corresponding barrier problem is solved. Decreasing the value of μ with each run solves the problem with better accuracy.

Chapter 4

Programs and Tools

4.1 Modelica

Modelica [14] is a declarative, object-oriented modeling language for modeling dynamical systems. The language is mainly equation based and classes, also called models, are described by high-level differential, algebraic and discrete equations, which are solved by first transforming them into a set of explicit differential equations. The declarative concept means that only the logic of the computation needs to be set up, i.e the dynamical equations of the system should be defined. The exact control flow, i.e how the system of equations should be solved, does not need to be specified. [15] [16]

A Modelica model consists mainly of variables, parameters and equations. A simple example of a model of an integrator can be seen in Figure 4.1. The model consists

```
model IntegratorEx
  extends Modelica.Blocks.Interfaces.BlockIcon;

  parameter Real k(unit="1")=1 "Integrator gain";
  parameter Real y_start "Initial or guess value of output";

  Modelica.Blocks.Interfaces.RealOutput y(start=y_start, fixed=true
      ) "Output signal"
  Modelica.Blocks.Interfaces.RealInput u "Input signal"

equation
  der(y) = k*u;

end IntegratorEx;
```

Figure 4.1: A Modelica model of an integrator.

of parameters for the integrator gain and the output `start` value. Two variables, the input which is the variable to be integrated, and the output which is the result of the integration, are declared. The dynamic of the integrator is specified with a differential equation under the `equations` section. One of the benefits when using object-oriented programming languages is the possibility of inheritance and in Modelica, this is done through the `extends` statement. A model can extend several other models at the same time.

When a variable is declared, some properties of the variable can be given; for example its `max`- and `min` values, its `nominal` value and its `start` value, see Figure 5.4 for an example. The `max`- and `min` values specify in what range the variable is defined, while the `nominal` value is used for scaling the variables, see Section 5.2 *Scaling*. A number of built-in libraries are available in Modelica, for example the `Modelica.SIunits`, which makes it possible to specify the unit of a parameter when declaring it. The `Modelica.SIunits` have different properties which are taken into account during simulation and optimization.

Before simulating a model all states need to be given an initial value. The default initial value for most variables is zero, which is used unless another value is given through the `start` attribute. The `start` value is however only taken as a guess value and may be changed by the solver unless another attribute, `fixed`, is set to `true`. Another way of initializing a model is by use of the `initial equation` construct. The variables are then initialized by solving the system of equations given, taking the fixed start values into account.

Modelica is a free language and it can be used in both commercial and free simulation environments like Dymola [6] and JModelica.org [17].

4.2 Dymola

Dymola (Dynamic Modeling Laboratory), is a commercial environment for modeling and simulating dynamical behaviours of different kinds of physical systems. It is based on the modeling language Modelica. Since Modelica is an open language, users have the possibility to both create their own model libraries and to modify already existing libraries in Dymola. [6]

One of the benefits with Dymola is the graphical interface where it is possible to get a clear overview of the model that is developed. The model can also be viewed as pure Modelica code and it is possible to change and develop the model both in the graphical and the textual view. [6]

Models in Dymola usually consist of single components which are modeled separately and joined by a drag and drop technique. To describe how different components interact, a `connector` class is used. The `connector` class describes the additional connection equations and in the graphical view the `connector` is shown as a line between the joined components. It is possible to model physical connections between components. Other variables and states, which are not only defined as inputs and outputs, can be connected with the connection equations.

Figure 4.2 shows the model of the integrator described above in the graphical view of Dymola. To see how different components can be connected to a larger model, see Figure 5.1 which shows one of the CCPP models optimized in the thesis.

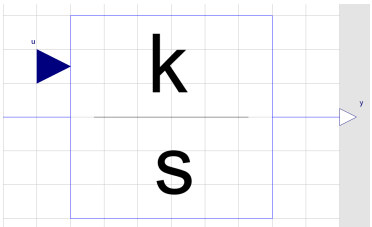

Figure 4.2: The IntegratorEx model in the graphical view of Dymola.

4.3 Optimica

Optimica is an extension of the Modelica language that enables high-level formulation of optimization problems based on Modelica models. The extension mainly consists of an additional class `optimization` which includes the attributes `objective`, `finalTime`, `startTime` and `static`. The `objective` attribute specifies the type of objective function that should be used in the problem. `startTime` specifies at what time the optimization starts and `finalTime` when it ends.

An optimization class can inherit from a Modelica model with the `extends` statement seen in Figure 4.3. The full model can also be described explicitly in the optimization class, in the case of simple model implementations. Another supplement is the constraint section which can handle both equality- and inequality path constraints and point constraints.[18]

Similarly to a Modelica model, an optimization class consists of an equation section where equations for describing the model can be defined if the model is implemented directly in the optimization class. In the case when the optimization class extends a model, this section only consists of the equations related to the optimization problem. These are usually differential equations including the cost function.

The Modelica built in type `Real` also has two other attributes in Optimica; `free` and `initialGuess`. The `free` attribute lets the user decide if a variable or parameter should be free in the optimization while the `initialGuess` attribute makes it possible to provide an initial guess for variables and parameters. By extending or creating an instance of a Modelica model in an optimization class in Optimica these features make it possible to solve different kinds of optimization problems using the model, for example a minimal time problem or a minimization of some kind of cost function. Optimica was first developed by Åkesson in 2007, see [18] for further reading.

```
optimization costFinalTime (objective=cost(finalTime), startTime=0,
                            finalTime=1)
  extends CCPP.JM.Modul1DOF;     //extends the Modelica model
                                 //Modul1DOF.mo

  Real gradT (max=max_value);    //gradient constraint defined
                                 //as maximum attribute
  parameter Real u_ref=1;        //reference value: GT power load
  Real u(nominal=1);             //GT power load with nominal
                                 //attribute

  parameter Real alpha = 1e-4;   //weight on GT power cost
  parameter Real beta = 1;       //weight on derivative of GT
                                 //power cost

  Real cost(min=-1);             //total cost with min attribute

  Real cost_u(start=cost_u_start, fixed=true, min=-1);
                                 //GT power cost
  Real cost_der_u(start=cost_der_u_start, fixed=true, min=-1);
                                 //derivative of GT power cost
  parameter Real cost_u_start=0;
                                 //start value of GT power cost
  parameter Real cost_der_u_start=0;
                                 //start value of derivative of
                                 //GT power cost

equation
  cost=cost_der_u+cost_u;                //total cost
  gradT=evaporator.T - wall.layer[2].T; //gradient
  u = integrator.y;                      //GT power
  der(cost_u) = alpha*(u-u_ref)^2;              //GT power cost
  der(cost_der_u) = beta*(der_u)^2;             //derivative of GT
      power cost

constraint
  //gradT  <=  max_nbr;             //gradient constraint defined
                                    //in Optimica constraint
                                        section

end costFinalTime;
```

Figure 4.3: An Optimica optimization class extending the Modelica model Modul1DOF.mo.

4.4 JModelica.org

4.4.1 Description

In this project, the tool used for optimization is the open source platform *JModelica.org* [17].
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. [17]

The JModelica.org platform uses a simultaneous optimization method based on Lagrange polynomials on finite elements with Radau points. The differential algebraic equations (DAE) describing the dynamical process are transformed to a non-linear programming (NLP) problem by approximating control and state profiles and the NLP problem is solved by the solver IPOPT, see Section 4.4.4 [11].

JModelica.org supports the optimization extension Optimica, see Section 4.3, and for user interaction JModelica.org relies on the Python language [19]. Python is a clear and commonly used programming language which can be used as an object-oriented language. It is recommended to use the PyLab environment [20] to run the Python files which interact with JModelica.org. Some additional Python packages may also be useful, for example SciPy [21] and NumPy [22], which together create an open-source software for mathematics, science, and engineering. Another additional package matplotlib [23] is used for plotting through the PyLab environment. See Åkesson et al. [24] for a thorough description of the JModelica.org platform.

4.4.2 Model Objects

When working with Modelica and Optimica models in JModelica.org the models are first compiled and then loaded as a model object in the Python interface. JModelica.org can create three different types of model objects; `FMUModel`, `JMUModel` and `FMUXModel`. In this thesis only the first two are used and thus only these will be described.

FMUModel

For simulation purposes, an `FMUModel` is used. A Functional Mock-up Unit (FMU) follows the FMI (Functional Mock-up Interface) standard and is created by compiling a Modelica model in JModelica.org or any other tool which supports the FMU export. In the FMU the DAE is converted into an ODE which makes simulations faster and more efficient. When compiling in JModelica.org the compiler function `compile_fmu` is imported and the compilation is performed:

```
# Import the compiler function
from jmodelica.fmi import compile_fmu
# Specify Modelica model and model file
model_name = 'myPackage.myModel'
mo_file = 'myPackage.mo'
# Compile the model, return argument is the file name of the FMU
fmu_name = compile_fmu(model_name, mo_file) .
```

The FMU file, `fmu_name`, is thereafter loaded as an `FMUModel` in JModelica.org:

```
# Import FMUModel from jmodelica.fmi
from jmodelica.fmi import FMUModel
myModel = FMUModel(fmu_name)
```

and can be simulated using the Assimulo package [25]. For a more detailed description of import and export of FMU:s in JModelica.org, see [26].

JMUModel

A `JMUModel` object can be used for both optimization and simulation purposes. To create a `JMUModel`, a Modelica or Optimica model is first compiled to a JMU using the method `compile_jmu`:

```
# Import compile_jmu
from jmodelica.jmi import compile_jmu
# Specify Modelica model and model file
model_name = 'myPackage.myModel'
mo_file = 'myPackage.mo'
# Compile model
jmu_name = compile_jmu(model_name, mo_file) .
```

A JMU is a compressed file following a JModelica.org specific standard that is close to the FMI standard. The DAE is in the JMU not transferred to an ODE which makes simulations of JMU:s less efficient and more time consuming than FMU:s. After compilation the JMU file is loaded into JModelica.org and the `JMUModel` is created by:

```
# Import JMUModel
from jmodelica.jmi import JMUModel
# Load model
myModel = JMUModel(jmu_name)
```

and can be optimized using state of the art numerical methods or simulated with the Assimulo package.

The compiler options used in the project are compiled using:

```
compile_jmu(model_name, mo_file, compiler_options)
```

and they are briefly described below.

`enable_variable_scaling`**:** The `nominal` attribute is used to scale variables in the model when set to `True` (default `False`). `True` is used in the project and read as following:

```
compiler_options = { "enable_variable_scaling":True } .
```

`extra_lib_dirs`**:** The value of this option is appended to the value of the MODELICAPATH environment variable for determining in what directories to search for libraries [27].

`state_initial_equations`: A new option has been added during the thesis that executes the following steps in the compiler:

1. Parameters corresponding to the start values of the states are added:
   `parameter Real _start_<state_name> = <start_attribute_value>;.`
2. The initial equations in the model, including variables with `fixed=true` are discarded.
3. Initial equations, one for each state, on the form:
   `_start_<state_name> = <state_name>` are instead added.

The user can then use the parameters introduced by the compiler to set the start values of the states [28]. `True` is used in the project (default `False`). This compiler option makes it possible to define the starting point of a simulation or optimization by a previous simulation or optimization result file. Existing initial equations in the Modelica/Optimica models are overwritten and the model is initialized by the previous simulation or optimization.

4.4.3 Methods

Simulation

For simulation in JModelica the method `simulate` is used. The return argument is a result object containing the simulation result. The method can be used on both an `FMU`- and a `JMUModel` and is invoked in the following way:

```
res = model_sim.simulate(start_time=0, final_time=1.0, algorithm='
    AssimuloFMIAlg' input=input_object, options={}) .
```

The attributes `start_time` and `end_time` are simply the times when the simulation should start and end. Via the attribute `algorithm` the simulation algorithm can be changed. The default algorithm for an `FMUModel` is `'AssimuloFMIAlg'` that uses the Sundials [29] *CVode* solver as default.

When simulating a model with a free input, for example a model which is also used for optimization, the attribute `input` can be used. The `input_object` defines the input trajectories of the simulation and is a 2-tuple consisting of the names of the input variables and their trajectories. The default simulation algorithm `'AssimuloFMIAlg'` has a number of options which can be specified by the attribute `options`. Some of the options used in this thesis are:

`solver`: Specifies the simulation method that is to be used. (Default CVode)

`initialize`: If set to `True`, the initializing algorithm defined in the `FMUModel` is invoked, otherwise it is assumed that the user has manually invoked `model.initialize()`. (Default `True`)

`write_scaled_result`: Set this parameter to `True` to write the scaled result to file. If the parameter value is `False`, then the variable scaling factors of the model are used to reproduce the unscaled variable values. (Default `False`)

See [27] for further information.

Optimization

After loading and compiling a `JMUModel` the model can be optimized by calling the method `JMUModel.optimize`, returning a result object containing the optimization result. The `optimize` method has the parameters `algorithm` and `options`. When calling the optimize method with the default optimization algorithm and with default options the method is invoked by:

```
res = JMUModel.optimize() .
```

The default algorithm is `'CollocationLagrangePolynomialsAlg'` which is based on direct collocation on finite elements using the IPOPT solver algorithm for obtaining the numerical solution. The algorithm used in the project is set by default. See Section 3.3 for collocation methods and Section 4.4.4 for a description of IPOPT.

Adding optimization options is done by first obtaining an options object that gives the user the possibility to access the documentation and the default option values. The method `optimize_options` is invoked by:

```
opts = JMUModel.optimize_options()
```

and the options object is a Python dictionary that is set using:

```
opts['n_e'] = 5 .
```

The optimization algorithm can then be invoked with the new option:

```
res = JMUModel.optimize(options = opts) .
```

The collocation-based optimization algorithm provides a number of options, a brief explanation of some of the the options used in the thesis will be given here.

`n_e`**:** Number of elements of the finite element mesh (default 50).

`n_cp`**:** Number of collocation points in each element (default 3).

`blocking_factors`**:** Blocking factors are specified by a vector of integers, where each entry in the vector corresponds to the number of elements for which the control profile should be kept constant (default not used). For example, the blocking factor specification `[2,1,5]` means that u_0 = u_1 and u_3 = u_4 = u_5 = u_6 = u_7 assuming that the number of elements is 8. [27]

`init_traj`**:** Initial guess trajectory used for initialization of the optimization problem (default `None`).

`write_scaled_result`**:** Write the scaled optimization result if set to `True`. This option is only applicable when automatic variable scaling is enabled. Only for debugging use. (Default `False`)

For further specification of optimization options consult [27].

4.4.4 IPOPT

The open-source software *IPOPT* (Interior Point OPTimizer) is a package for large-scale non-linear optimization using an *interior point method*, see Section 3.4.2. IPOPT is suitable for large problems with a large number of constraints and variables. The Jacobian matrix of the constraint function is assumed to be sparse. [13]

The algorithm is used to solve the NLP resulting from collocation (see Section 3.3) and it aims for finding the global minimizer. If the problem is non-convex many stationary points, local minimizers, exist and depending on the starting point and algorithmic choices the method converges to the closest local minimizer.

If the algorithm has found a point satisfying the first-order optimality conditions for the barrier problem (see definition in Section 3.4.1) the algorithm is finished and has found a stationary point for the NLP (given in equations (3.16)-(3.19)). This stationary point can though be a minimizer as well as a maximizer or a saddle point. [13]

When using a line search method in the algorithm the direction from the Newton-method should be guaranteed to be a descent direction:

$$f(x+\lambda d) < f(x) \text{ when } 0 < \lambda < \delta$$
$$\text{for } d \neq 0 \text{ and } \delta > 0.$$

If no acceptable step size is found the algorithm temporarily ignores the objective function (feasibility restoration phase) and solves the problem to minimize the constraint violation. This will result in that a feasible point is found and the IPOPT algorithm can be continued, or a local minimizer of the constraint violation is obtained indicating that the problem is locally infeasible. [13]

IPOPT Output

At each iteration of the optimization IPOPT provides a summary of how it solves the problem and how close the algorithm is to finding a feasible solution. See Figure 4.4 for an example of the output that IPOPT returns.

```
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  1.6109693e+01 1.12e+01 5.28e-01   0.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  1.8029749e+01 9.90e-01 6.62e+01   0.1 2.05e+00    -  2.14e-01 1.00e+00f  1
   2  1.8719906e+01 1.25e-02 9.04e+00  -2.2 5.94e-02   2.0 8.04e-01 1.00e+00h  1
```

Figure 4.4: The output provided by IPOPT while running the non-linear optimization algorithm.

In Figure 4.5 an explanation of the output is provided for all columns of the IPOPT output. The solver aims at finding a solution while the constraint violation `inf_pr` and the dual infeasibility `inf_du` go to zero. The size of the primal search direction should as well go to zero in the end as the objective function reaches the optimal value. In a typical optimization run the value of the barrier parameter goes to zero and there will be a decrease in the number listed in column 5. The IPOPT algorithm closes with some statistics of the computational effort and with an `EXIT` message, informing if the optimization has terminated successfully. [13]

IPOPT Options

The user can set a large number of options that change the algorithmic behaviour and other aspects of IPOPT. They are described in the "IPOPT Options" section of the

col #	header	meaning
1	iter	iteration counter k (r: restoration phase)
2	objective	current value of objective function, $f(x_k)$
3	inf_pr	current primal infeasibility (max-norm)
4	inf_du	current dual infeasibility (max-norm)
5	lg(mu)	$\log_{10}$ of current barrier parameter μ
6	||d||	max-norm of the primal search direction, $\|\Delta x_k\|_\infty$
7	lg(rg)	$\log_{10}$ of Hessian perturbation δ_x (--: none, $\delta_x = 0$)
8	alpha_du	dual step size $\alpha_k^{z,\max}$
9	alpha_pr	primal step size α_k^x
10	ls	number of backtracking steps $l + 1$

Figure 4.5: List of the columns of the output provided by IPOPT.

IPOPT documentation available at [30]. IPOPT options are set using the syntax

```
opts['IPOPT_options']['max_iter'] = 200
```

where in this example the maximum number of iterations for the solver is set to 200. The linear solver MUMPS is used by default in IPOPT. In this thesis, large models have been optimized and the MA27 linear solver [31] has been a better option due to its more robust properties.

A brief explanation of some of the options used in the thesis will be given here.

`max_iter`: The maximum number of iterations (default 3000).

`tol`: IPOPT terminates successfully if the (scaled) NLP error is below the value of `tol`. This value is set to 1e-4 in the project (default 1e-8).

`nlp_scaling_method`: Selects the technique used for scaling the problem internally before it is solved (default "gradient-based").

`nlp_scaling_max_gradient`: Maximum gradient after NLP scaling. This option is only used if `nlp_scaling_method` is chosen as "gradient-based". This value is set to 1e4 in the project (default 100).

`mu_strategy`: Determines which barrier parameter update strategy is to be used (default "monotone").

Chapter 5

Models

5.1 Plant Models

In this project three models of a CCPP with different complexities have been considered. All models consist of one GT and a high pressure (HP) stage of an HRSG , see Dymola model *CCPP1* in Figure 5.1, model *CCPP2* in Figure 5.2 and model *CCPP3* in Figure 5.3. The models are developed in Modelica [14] using the commercial modeling and simulation environment Dymola [6] and are based on elementary models from first principle equations of mass and energy. Discontinuities have been smoothed and all equations are twice continuously differentiable.

Single components are modeled separately according to the object oriented principle and joined by additional connection equations to form the complete system model. The possibility of using inheritance in the Modelica language has been made use of and the CCPP1 and CCPP2 models have been constructed by inheriting from the same base model. The base model provides the properties for the GT, HRSG and boundaries used in both models. The third model, CCPP3 is more complex than CCPP1 and CCPP2 and is thus constructed from another base model.

Some of the components in the Dymola models are not connected by visible connector lines but only by Modelica equations. This is the case for the output of the integrator at the valve opening, which is connected to the real expression at the valve just above it. The two outputs of the GT are also connected by Modelica equations to the two real expressions to the right of the GT, see Figures 5.2 and 5.3.

In all three models the GT model computes temperature and mass flow of the gas entering the HRSG at every load. The water side is modeled by dynamic balance equations whereas the gas side is static. An ideal level control is assumed in the evaporator model and it computes the water/steam flow through the HRSG.

Pressure and specific enthalpy have been chosen as states in the balance equations on the water side. Correlations to compute temperature as well as density and its derivatives with respect to pressure and enthalpy need therefore to be derived. Polynomial approximations expressed as Taylor expansions from the phase boundaries have been chosen. This leads to optimization friendly and accurate medium properties and also a continuous transition of temperature and density across the phase boundaries.

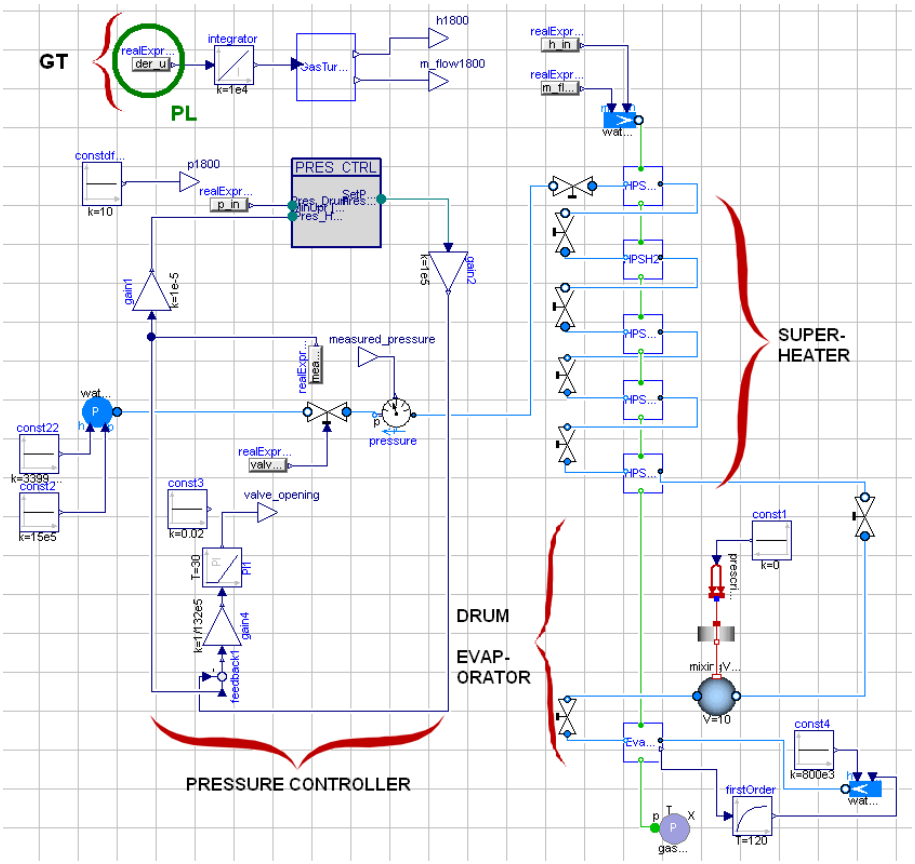

Figure 5.1: Dymola model CCPP1. The main components are marked and the input PL (Power Load) is circled.

5.1.1 CCPP1 Model

The HP stage is in the CCPP1 model represented by an HP superheater and an HP evaporator. To attain better accuracy, the superheater is described by five partial components with different tube geometries, that are operating at different pressures. CCPP1 has a pressure controller where the bypass valve controls the pressure in the water circuit to limit large pressure transients, see component *Pressure Controller* in Figure 5.1.

The component that is subject to high stress during start-up transients is in this model considered to be the wall in the evaporator drum, see component *Drum* in Figure 5.1. The wall is a spatially discretized model with three spacial layers. Input to the CCPP1 Model is the *GT Power Load* (marked as PL in Figure 5.1).

5.1.2 CCPP2 Model

In the CCPP2 model the HP stage is like in CCPP1 represented by an HP superheater with five partial components and an HP evaporator. CCPP2 is constructed by inheritance from the same base model as CCPP1, with the exception that CCPP2 does not have a built in pressure controller, see Figure 5.2. The bypass valve is in this model

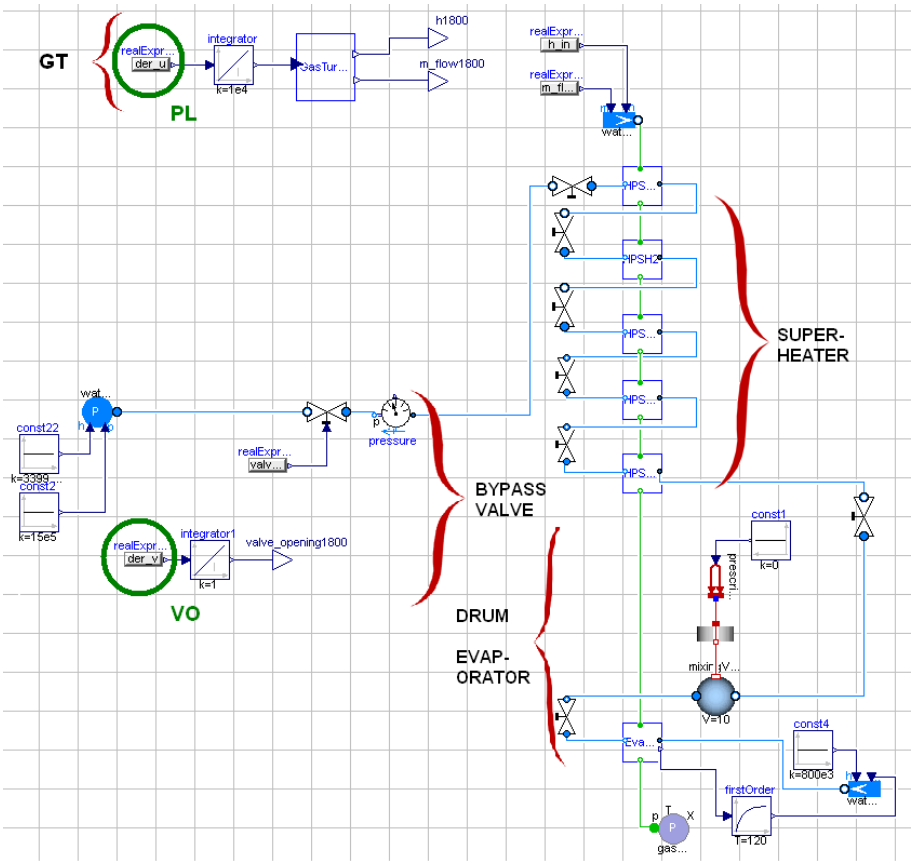

Figure 5.2: Dymola model CCPP2. The main components are marked and the inputs PL (Power Load) and VO (Valve Opening) are circled.

controlled by the input *Valve Opening* (marked as VO in Figure 5.2) where opening and closing the valve controls the pressure in the water circuit. The second input is in CCPP2 the *GT Power Load* (marked as PL in Figure 5.2).

The wall in the evaporator drum is considered to be the component that is subject to high stress during start-up transients, see component *Drum* in Figure 5.2. The wall is spatially discretized, in the same way as in CCPP1, with three layers.

5.1.3 CCPP3 Model

The CCPP3 model is the most complex model considered in the thesis and it is modeled with an additional IP reheater apart from an HP superheater and an HP evaporator. The reheater is described by three partial components and the superheater has four partial components with different tube geometries that are operating at different pressures.

The model has two inputs; *GT Power Load* (marked as PL in Figure 5.3) and *Valve Opening* (marked as VO in Figure 5.3) and the bypass valve is in this model controlled by the input VO which controls the pressure in the water circuit.

An additional component that has been modeled in CCPP3 is the header of the part of the superheater operating at highest temperature, see component *Header* in

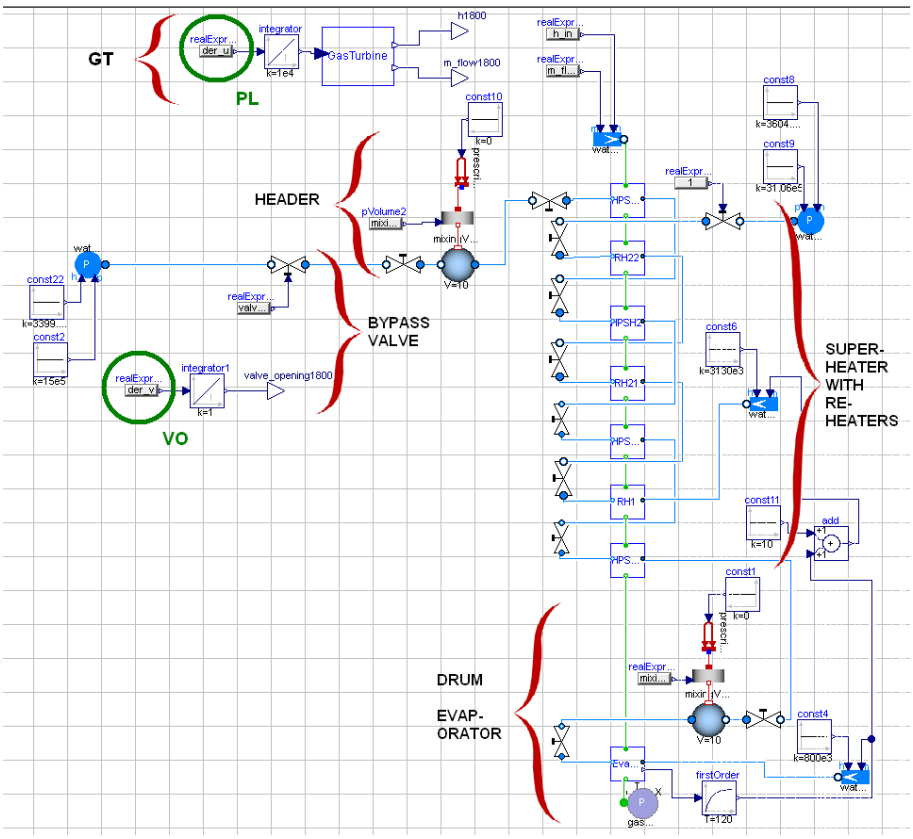

Figure 5.3: Dymola model CCPP3. The main components are marked and the inputs PL (Power Load) and VO (Valve Opening) are circled.

Figure 5.3. The header is in this model considered as a component subject to high stress during start-up transients together with the evaporator drum.

5.2 Scaling

When modeling thermodynamic systems the models contain variables that differ in several orders of magnitude. Examples from the project include pressures, temperatures and mass flows, and it is necessary to scale the variables to get good performance of the numerical algorithms.

The built in Modelica `nominal` attribute has been used to automatically scale the models. The nominal attributes have been set when declaring the variables:

```
Real pressure(start=102.3e3, nominal=1e4); .
```

The compiler option `enable_variable_scaling` is set to `True` and this scales all variables by their nominal value before applying the numerical algorithm. It is desirable that all variables take values close to one, but this is not possible due to variations in variable values during simulation and optimization. The nominal values do not apply to equa-

tions and by multiplying the variable with its nominal value the model equations are fulfilled:

```
T = f(p)
```

is replaced by the equation

```
T_scaled*T_nom = f(p_scaled*p_nom) .
```

5.3 Units

Units have been defined as an extension of the `Modelica.SIunits` with additional attributes corresponding to properties of the unit which are taken into account during compilation. An example for the specific enthalpy is given in Figure 5.4.

```
type SpecificEnthalpy =  Modelica.SIunits.SpecificEnthalpy(start =
    0.8e6, min = 1e4, max = 3.6e7, nominal = 1.0e5);
```

Figure 5.4: Extension of the `Modelica.SIunits` with attributes `start`, `min`, `max` and `nominal`.

Here the built in Modelica nominal attribute has been used as well as the minimum and maximum attributes that provide implicit constraints to the optimization problem. Almost all states and some parameters in the models are declared with their units, assuring that the model is scaled well and that the physical limits are held.

Chapter 6

Problem Formulation

6.1 Method

As described in Section 2.2.4, the start-up of a CCPP can be divided into three phases. This thesis focuses on the optimization of the second phase, that is the loading of the GT, where the aim is to make the start-up procedure of a CCPP more efficient, with respect to the start-up time and the power production. Even so, both phase 1 and phase 2 have been modeled, see definitions in Figure 6.1. Phase 1 is considered as a non-controlled phase where the GT starts rotating and is accelerated to net frequency, see green markings in Figure 6.1. This phase is modeled only for simulation purposes, to give an initial point for the controlled phase 2 that is optimized. See simulation of phase 2 with zero-load marked as blue and optimization result marked as red in Figure 6.1. It is possible to choose different initial values for phase 1 which also gives the possibility to use different start-up conditions for the optimization phase. Different conditions can be given depending on if a warm or a hot start is to be investigated. In this thesis, only a hot start has been considered. The start-up is considered to be complete when the GT has reached its full load, i.e. its maximum power output.

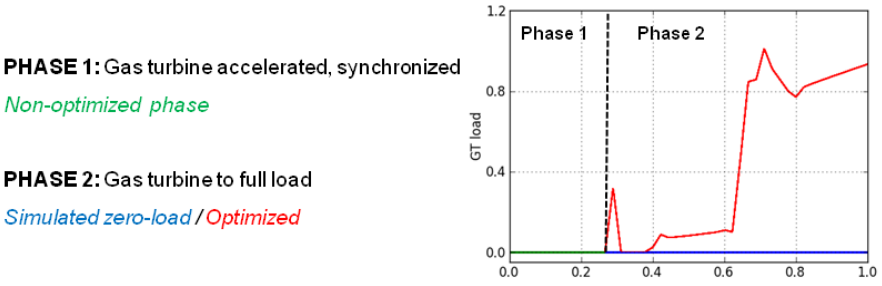

Figure 6.1: The two phases considered in the thesis are defined. Simulation of phase 1 is marked as green. Phase 2 is optimized with a zero-load input trajectory from simulation, see blue curve. The optimization result is marked as red.

The first steps towards optimizing a dynamic system is to construct a model suitable for optimization. This model should be simple enough to allow for optimization techniques to find a solution, but the model must be fairly detailed to be able to represent the actual process. The models used in this thesis are implemented for opti-

mization purposes based on elementary models from first principle equations of mass and energy, see Chapter 5. Three Modelica models have been optimized in the thesis; *CCPP1*, *CCPP2* and *CCPP3*. See Dymola models in Figures 5.1 - 5.3. To include the optimization problem, the model equations written in the Modelica language are combined with the optimization specification written in the Modelica extension Optimica and together they form the complete optimization model.

It is important to understand the processes and dynamics in the system before optimizing and therefore all models have first been simulated with possible inputs and loads. Simulation has also been necessary to acquire feasible initialization trajectories for the optimization problems, see Section 6.6 *Initialization*. For simulation purposes, only `FMUModel` objects have been used, see Section 4.4.2. The FMU has either been created by compiling a Modelica model in JModelica.org or in Dymola, and Dymola has served as a reference when evaluating the results from JModelica.org. When optimizing it is only possible to use `JMUModels`, see Section 4.4.2 for further reading.

Two control variables have been considered in the thesis, see Section 6.2 *Degrees of Freedom*. The cost functions are given on the *Lagrange* form or formulated with the *minimal time* formulation. The Lagrange form maximizes the produced power output during start-up and aims at minimizing the start-up time, while the minimal time formulation only aims for minimizing the start-up time, see Section 6.3 *Cost Functions*. The constraints that are considered as most limiting in the thesis are the limits in the walls of the heat recovery steam generator (HRSG) due to stress caused by pressure- and temperature gradient transients during the start-up. See Section 6.4 *Constraints*, where it is described how the constraints in the walls of the drum and header have been taken into account in the thesis.

From the Modelica models CCPP1, CCPP2 and CCPP3 five optimization models have been defined with the Optimica extension. *Optimization Problem 1* is extended from CCPP1, *Optimization Problem 2* is extended from CCPP2 and *Optimization Problems 3, 4 and 5* are extended from CCPP3. The degrees of freedom have been specified together with the cost function and depending on the complexity of the Modelica model constraints have been defined. See definitions of the optimization models in Sections 6.2 - 6.4. A summary of the optimization models is given in section 6.5 *Optimization Models*.

The optimization settings that are used are presented in Section 6.7 and the optimization results have been analyzed by means of the discretization in the collocation algorithm by varying the number of elements and the blocking factors. The accuracy of the discretization of the dynamics of the optimized solutions are verified by simulating the system using the optimal control profiles as input. This is to make sure that the method of collocation and the number of elements gives a fine enough discretization to represent the dynamics in the models.

6.2 Degrees of Freedom

Two control variables have been considered in the thesis: the power load u of the GT and the opening v of the bypass valve. The degrees of freedom in the optimization are defined as the time-derivative of the control variables, i.e. du/dt (marked as PL - Power Load in Figure 5.1) and dv/dt (marked as VO - Valve Opening in Figure 5.1), and are parametrized by piecewise constant signals. This is achieved by introducing an integrator between the du/dt control variable and the GT, see Figure 6.2 for a schematic figure of the inputs. The use of the derivatives of the control variables as degrees of

freedom is preferable when the inputs are chosen to be piecewise constant since a continuous profile for the actual PL and the VO then will be obtained through the integrator.

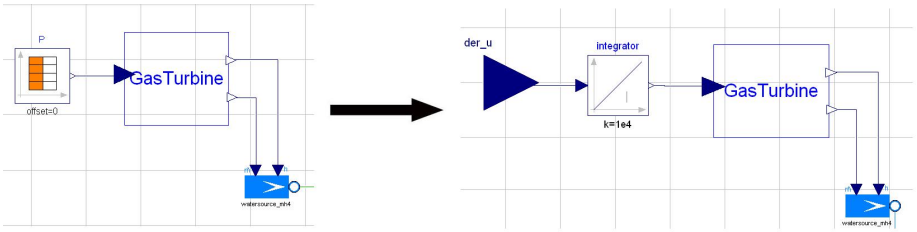

Figure 6.2: The input table used when simulating is replaced by the der_u control variable and an integrator in the optimization model.

Optimization Problem 1: In the first optimization problem the GT load u is chosen to be the only control variable. The input u has been allowed to be both monotonic and non-monotonic. The bypass valve is in this case manipulated to control the pressure at the superheater outlet by the control loop seen in Figure 5.1.

Optimization Problems 2-5: In the second, third, fourth and fifth optimization problem, the pressure controller is removed and both degrees of freedom, du/dt and dv/dt, are used, see Figures 5.3 and 5.2. Non-monotonic inputs have been used.

6.3 Cost Functions

The cost functions used in the thesis are written on the *Lagrange* form as in equation (3.14) and with the *minimal time* formulation as in equation (3.15). When using the Lagrange form the optimization problems have been formulated using a quadratic cost function where the integrand L penalizes the deviation of the GT load u from its reference value u_{ref} as well as the use of the inputs:

Optimization Problem 1:

$$L = \alpha(u - u_{ref})^2 + \beta(\frac{du}{dt})^2. \tag{6.1}$$

Optimization Problems 2-4:

$$L = \alpha(u - u_{ref})^2 + \beta(\frac{du}{dt})^2 + \gamma(\frac{dv}{dt})^2. \tag{6.2}$$

The reference value for $u(t)$ is normalized to 1, which corresponds to its full load. The cost function punishes a deviation for the control variables du/dt and dv/dt from the desired steady state value of zero. This formulation maximizes the produced power output during start-up and should also result in a fairly short start-up time.

The values of the cost function weights, α, β and γ, have been varied and explored to find the formulation that weights the deviation of u from full GT load as most important without compromising with the optimization problem's robustness. In order

to find a converging optimization solution the optimization problem should be well defined and well scaled, see section 5.2 *Scaling*. Like all variables the cost function should not attain too large or too small values, and has in the project been allowed to be in the interval from 1e-4 to 1e4.

Optimization Problem 5: The *minimal time* formulation has been used in Optimization Problem 5, and the cost function is in this case given by:

$$\varphi = \int_{t_0}^{t_f} 1 dt. \tag{6.3}$$

The time interval $t \in [t_0, t_f]$ is unknown and the endpoint is t_f, the free optimization variable. The dynamic optimization problem is formulated with point constraints at the time t_f where the GT power load u is to be at 100% of its full load.

6.4 Constraints

The most limiting factor during the start-up procedure of a CCPP is the stress due to pressure- and temperature gradient transients in the walls of HRSG. The wall of the HP drum and the wall of the header of the highest temperature superheater have been studied in the thesis. The models contain more constraints in the form of minimum and maximum attributes and correctly defined these should not be limiting during optimization. Other constraints in the models are the design of some components, for example diameters of valve openings etc.

As a starting point a simpler optimization constraint is considered by using a constant valued constraint on the temperature gradient in the walls of the HRSG, instead of a stress constraint. First a constraint is considered in the wall of the evaporator drum in:

Optimization Problems 1 and 2:

$$|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}|. \tag{6.4}$$

A constraint on the temperature gradient in the wall of the superheater header has also been added and considered in:

Optimization Problems 3 and 5:

$$|T_{wallHeader} - T_{SH}| < |\nabla T_{maxHeader}| \tag{6.5}$$

and used in the optimization together with the evaporator drum constraint. The values of $\nabla T_{maxDrum}$ and $\nabla T_{maxHeader}$ are assumed to have constant values which have been derived from simplified power plant control methods.

As an improvement the maximum temperature gradients have been modeled as a function dependent of the pressure of the HRSG component. This is denoted as the *basic stress model* in the thesis and the allowed temperature gradient is given a limit defined from table values. The basic stress model is based on a simplified model where both the thermal and mechanical stresses are considered. A higher pressure induces a higher mechanical stress and this allows the thermal stress to attain larger values. This is due to that the mechanical and thermal stresses operate in reverse spacial directions.

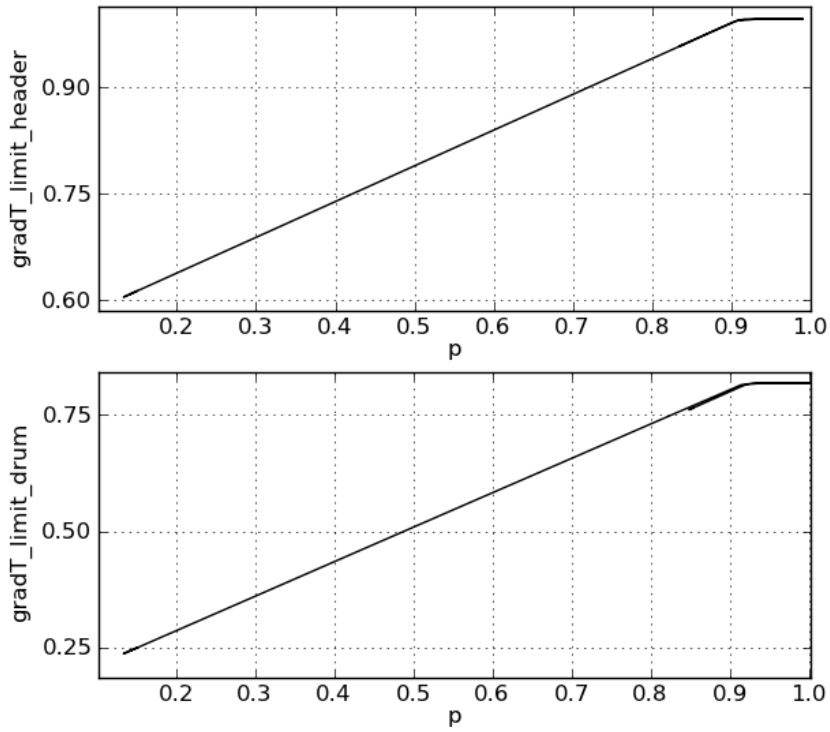

Figure 6.3: Maximum constraints considered in the basic stress model for the evaporator drum and the superheater header. All temperature gradient values and pressures have been normalized.

The basic stress model has been used in:

Optimization Problem 4:

$$|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}(p)| \tag{6.6}$$

$$\nabla T_{maxDrum}(p) = \min(\nabla T_{Drum}(p), \nabla T_{maxDrum}) \tag{6.7}$$

where $\nabla T_{maxDrum}(p)$ is the gradient constraint value at a certain pressure, $\nabla T_{Drum}(p)$ is a function of the pressure p for $p \leq p_{workDrum}$ and $\nabla T_{maxDrum}$ is the gradient constraint given as a constant for $p > p_{workDrum}$. The constraint on the header of the superheater is implemented in the same way from table values in the basic stress model:

$$|T_{wallHeader} - T_{SH}| < |\nabla T_{maxHeader}(p)| \tag{6.8}$$

$$\nabla T_{maxHeader}(p) = \min(\nabla T_{Header}(p), \nabla T_{maxHeader}). \tag{6.9}$$

See Figure 6.3 for the values considered in the basic stress model for the drum and the header.

The constraints on the temperature gradients can either be defined under the constraints section in the Optimica script, see example in section 4.3 *Optimica*, or in the

Modelica model as `min` or `max` attributes, see section 4.1 *Modelica*. IPOPT handles the two implementations different; in the case of an Optimica constraint IPOPT rewrites this into a Dynamic Optimization Problem (DOP) equation and this thus increases the number of equations in the DOP and so the problem complexity. In every iteration IPOPT must check if this constraint has been violated and this makes the optimization process more time consuming.

The favourable way of implementing constraints such as minimum and maximum values for variables is to define the limits in the Modelica model using `min` and `max` attributes. The values are instead set as boundaries for the variable and it is not possible for IPOPT to search for the optimal solution outside the boundaries. The attributes are set when declaring the variables in the Modelica model:

```
Real gradT(start = 0, max = max_value).
```

When the bypass valve opening is a degree of freedom, an additional constraint on the opening derivative is introduced:

$$|\frac{dv}{dt}| < |\frac{dv}{dt}|_{max}. \tag{6.10}$$

6.5 Optimization Models

Five optimization models have been optimized in the thesis. The models contain the Modelica model that is extended in the Optimica optimization specification. Here the properties of the optimization models are summarized.

Optimization Problem 1:

- Extended from Modelica model CCPP1, see Figure 5.1.
- Degree of freedom: $\frac{du}{dt}$. The GT load u is the only control variable, the bypass valve is controlled by the control loop.
- The input u is allowed to be both monotonic and non-monotonic.
- The cost function is given as: $L = \alpha(u-u_{ref})^2 + \beta(\frac{du}{dt})^2$.
- Drum constraint:
  $|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}| = 0.5$.

Optimization Problem 2:

- Extended from Modelica model CCPP2, see Figure 5.2.
- Degrees of freedom: $\frac{du}{dt}$ and $\frac{dv}{dt}$. The GT load u and the bypass valve are controlled in the optimization.
- The input u is non-monotonic.
- The cost function is given as: $L = \alpha(u-u_{ref})^2 + \beta(\frac{du}{dt})^2 + \gamma(\frac{dv}{dt})^2$.
- Drum constraint:
  $|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}| = 0.5$.

Optimization Problem 3:

- Extended from Modelica model CCPP3, see Figure 5.3.
- Degrees of freedom: $\frac{du}{dt}$ and $\frac{dv}{dt}$. The GT load u and the bypass valve are controlled in the optimization.
- The input u is non-monotonic.
- The cost function is given as: $L = \alpha(u - u_{ref})^2 + \beta(\frac{du}{dt})^2 + \gamma(\frac{dv}{dt})^2$.
- Constraints:
  Drum: $|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}| = 0.5$ and
  Header: $|T_{wallHeader} - T_{SH}| < |\nabla T_{maxHeader}| = 0.75$.

Optimization Problem 4:

- Extended from Modelica model CCPP3, see Figure 5.3.
- Degrees of freedom: $\frac{du}{dt}$ and $\frac{dv}{dt}$. The GT load u and the bypass valve are controlled in the optimization.
- The input u is non-monotonic.
- The cost function is given as: $L = \alpha(u - u_{ref})^2 + \beta(\frac{du}{dt})^2 + \gamma(\frac{dv}{dt})^2$.
- Constraints given by basic stress model:
  Drum:

$$|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}(p)|$$
$$\nabla T_{maxDrum}(p) = \min(\nabla T_{Drum}(p), \nabla T_{maxDrum}),$$

  Header:

$$|T_{wallHeader} - T_{SH}| < |\nabla T_{maxHeader}(p)|$$
$$\nabla T_{maxHeader}(p) = \min(\nabla T_{Header}(p), \nabla T_{maxHeader}),$$

  see Figure 6.3.

Optimization Problem 5:

- Extended from Modelica model CCPP3, see Figure 5.3.
- Degrees of freedom: $\frac{du}{dt}$ and $\frac{dv}{dt}$. The GT load u and the bypass valve are controlled in the optimization.
- The input u is non-monotonic.
- The cost function is given by the minimal time formulation: $\varphi = \int_{t_0}^{t_f} 1 dt$ with a point constraint at the time t_f where the GT power load u is to be at its full load.
- Constraints:
  Drum: $|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}| = 0.5$ and
  Header: $|T_{wallHeader} - T_{SH}| < |\nabla T_{maxHeader}| = 0.75$.

6.6 Initialization

Initialization is an important issue both when it comes to simulation and optimization of dynamical systems. For simulation, only the initial point needs to be specified whereas in dynamic optimization a whole initial guess trajectory needs to be given. In both cases, all states need to be initialized. It is preferable when initializing that all collocation points have values that fulfil the model equations to the specified accuracy. In the case of optimization, it is also preferable that the input guess trajectories do not violate any constraints in the optimization problem so that all trajectories are feasible.

For simulations there are a number of options that can be set, for example if the simulation should be initialized or not, see section 4.4.3. If `initialize` is set to `True` the model is initialized using the initial equations stated in the model taking the fixed start values into account. If not, no initialization problem is solved but the simulation continues from the last point of the previous simulation which is stored in the simulation model. To initialize the non-controlled phase, `initialize` has been set to `True` whereas in the controlled phase, `initialize` has been set to `False`. In this way, the second simulation just continues simulating from the last point of the first simulation.

The closer an initial guess trajectory is to the optimal solution, the easier it often is for the optimizer to to find a solution. This can turn out by means of less iterations or that it is possible to refine the discretization and still achieve an optimal solution. With respect to this, the optimization has been done iteratively, starting with a very simple initial guess trajectory which most often is quite far from the optimal solution. In every step the previous optimization result is used as initial guess trajectory, improving the initial guess in every iteration. See section 4.4.3 for more information on the optimization method in JModelica.org.

Since "optimization by hand" is undesirable, the first optimization is initialized by first simulating the model, using a simple (zero-load) input trajectory, see blue boxes in Figure 6.4. This results in feasible trajectories for the optimization that do not violate the defined constraints. Since phase 2 is initialized with the end point of phase 1 there are no risks of discontinuities between the non-controlled and the controlled phase. The simulation result of phase 2 is then used as initial guess trajectory for the first optimization, which is performed with a quite coarse discretization. The result of this optimization is then used as an initial guess trajectory for the next optimization, where the discretization has been refined, see red boxes in Figure 6.4. The iteration continues until the discretization is fine enough, i.e when no essential differences can be seen between the optimization results. See final result as the yellow box in Figure 6.4.

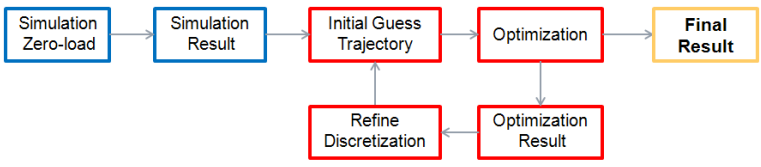

Figure 6.4: To get a more accurate solution with a fine discretization an iterative optimization technique has been used in the thesis.

6.7 Optimization Settings

The direct collocation method discretizes the initial guess trajectories and the problem is formulated as a non-linear programming (NLP) problem that IPOPT can be applied to. This implies that the differential equation is approximated by a discrete time counterpart and depending on the number of elements and the number of collocation points that have been set for the optimization the accuracy of the solution varies. The initial guess trajectories are differently discretized and a higher number of elements and collocation points gives the solution higher accuracy. The computation time used during the iterations is though increased [27].

The number of elements, n_e, in the optimization interval has been varied between 10 and 45 and the number of collocation points in every element was fixed to $n_{cp} = 3$. The optimization option `blocking_factors` is used to limit the input to piecewise constant values in intervals containing one or more elements. The example in Figure 6.5 shows an input with 10 elements and blocking factors `[1,3,3,3]`. This keeps the input du/dt first constant for one element and then constant for three elements in the last three blocks. The GT power load thus has a constant derivative in these blocks.

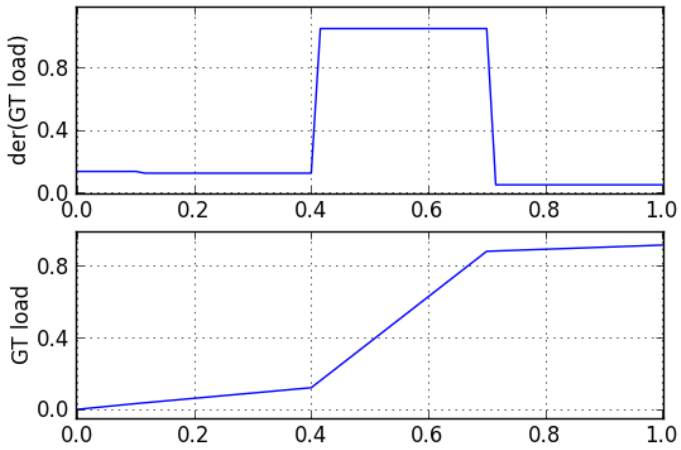

Figure 6.5: The input du/dt is kept piecewise constant by blocking factors.

In the iterative process where a finer discretization was achieved the blocking factors were as well made more fine. The optimization attribute `finalTime` was also varied to adjust the optimization time so that it suits the process dynamics. The overall relative tolerance for the interior point solver was chosen to be 10^{-4}.

Chapter 7

Results

7.1 Simulation

To understand the processes and dynamics in the system the models have been simulated with possible inputs and loads. In Figure 7.1 the CCPP1 model has been simulated using a rapidly increasing GT load input with values from a table. The dashed line shows phase 1 of the start-up, where the GT is accelerated to the electricity net

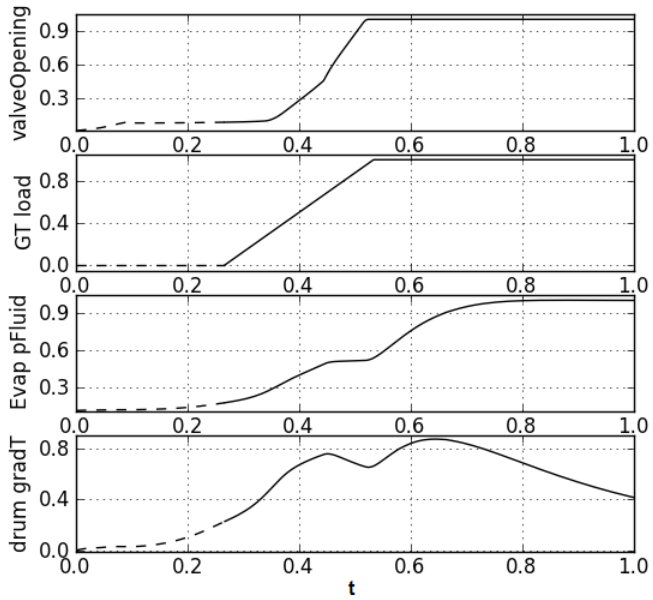

Figure 7.1: Simulation of CCPP1: dashed curve is the non-controlled phase (phase 1) and solid curve is the controlled phase (phase 2). From top: the bypass valve opening, the GT power output, the pressure in the superheater on the steam/water side and the temperature gradient in the wall of the drum. All results and times have been normalized.

frequency. Phase 1 is defined as the non-controlled phase and this phase is pre-defined and not optimized. The solid line shows phase 2 which is the phase that is optimized in this thesis. This rapid ramp up of the GT violates the temperature gradient constraint in the drum defined as $|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}| = 0.5$ and this clearly shows that any arbitrary GT load is not enough if one wishes to keep the constraints in the HRSG during start-up.

7.2 Optimization Problem 1

In Optimization Problem 1 the continuous-time optimization model contains 28 continuous time states and 456 scalar equations. After collocation in JModelica.org the discrete NLP is represented by 34598 equations when using 45 elements. The optimization is done iteratively, starting with a quite coarse discretization with $n_e = 10$ and ending with $n_e = 45$ where the first optimization is initialized with guess trajectories resulting from simulations of the CCPP model with zero GT power load. The blocking factors have been varied and analyzed as well as the `finalTime` variable in the optimization.

When analyzing the values of the weights α and β in the cost function $\beta = 0$ appeared to give the best convergence. The value of α was chosen so that the cost function was kept in the order of 1e4 or below.

7.2.1 Monotonic / Non-monotonic Input

When Optimization Problem 1 has been optimized the power output has been allowed to either both increase and decrease during start-up (non monotonic power output) or to only increase (monotonically increasing power output). Both cases have been optimally controlled to full load and the optimization results are shown in Figures 7.2 and 7.3. The solid line trajectory represents the solution for the non monotonic power output and the dashed trajectory represents the monotonically increasing power output. The results have been obtained using the iterative process and they have `n_e=45` and `blocking_factors=ones(n_e)`.

The optimal time for the GT to reach 95% of full load is approximately the same in both cases: 0.724 and 0.723 for the monotonic and non-monotonic load profile, respectively, see Table 7.1. From Figure 7.2, it can be seen that the temperature gradient constraint becomes rapidly active in spite of the low GT load. This is due to that the pressure controller keeps the bypass valve closed which results in a low mass flow through the valve and a high pressure in the HRSG, giving large temperature gradients in the wall of the drum. There is also hot steam in the HRSG due to transients from phase 1 and from the fact that the start-up is considered as a hot start. From the simulated zero-load trajectories it can be observed how the gradient increases even though there is no GT load.

At about $t = 0.37$, the GT load is rapidly increased from about 10% to 80%, at an optimal rate that steadily maintains the gradient constraint active. At about $t = 0.44$, the non-monotonic load profile reaches a maximum of 90% before decreasing to 80% at $t = 0.51$. This behaviour is related to the optimization formulation that penalizes deviations from the reference load of 100% and may therefore lead to overshoots before the gradient constraint becomes too constraining. The overshoot that is allowed when the input penalty coefficient β is zero is not observed in the case of a monotonically increasing load. The dip in the temperature gradient observed at about $t = 0.48$ is due to

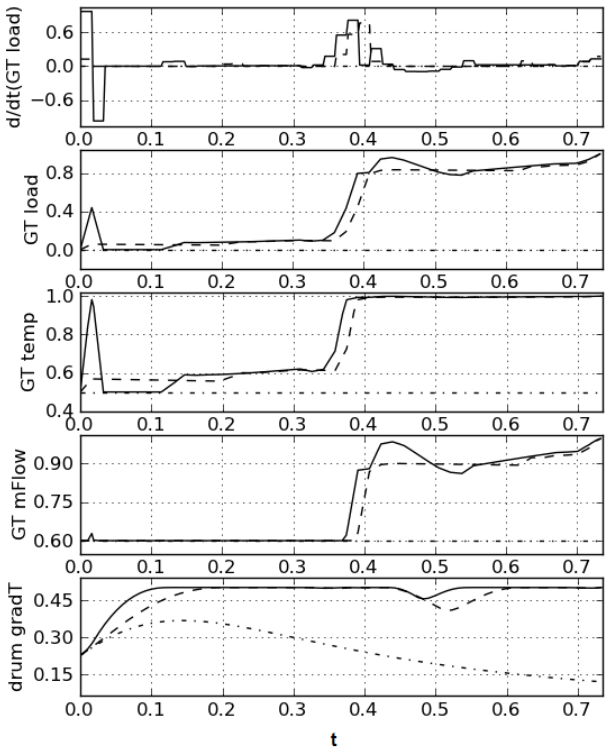

Figure 7.2: Optimal start-up trajectories for Optimization Problem 1: dashed (monotonically increasing power output) and solid (non monotonic power output) curves. Simulated initial guess trajectories: dash-dot curve. From top: the derivative of the power output, the GT power output, the GT outlet temperature, the GT mass flow and the temperature gradient in the wall of the drum. All results and times have been normalized.

the limited degree of freedom and its amplitude decreases with an increasing discretization level. In the case of a monotonically increasing load, the gradient dip cannot be avoided and is rather independent on the discretization level. After scaling the value of the objective function is 0.77 for the monotonically increasing power output case and 0.73 in the non monotonic case.

7.2.2 The Impact of Discretization

As mentioned in Section 6.6 *Initialization*, all results have been produced using an iterative technique where the result from the previous optimization has been used as new initial trajectory in every iteration, together with a finer discretization. In Figure 7.4 some solutions of Optimization Problem 1 with different discretizations can be seen. See also Table 7.1 for the objective function values, the time it takes to reach 95% of full GT load and the maximum pressure on the water side in the HRSG for

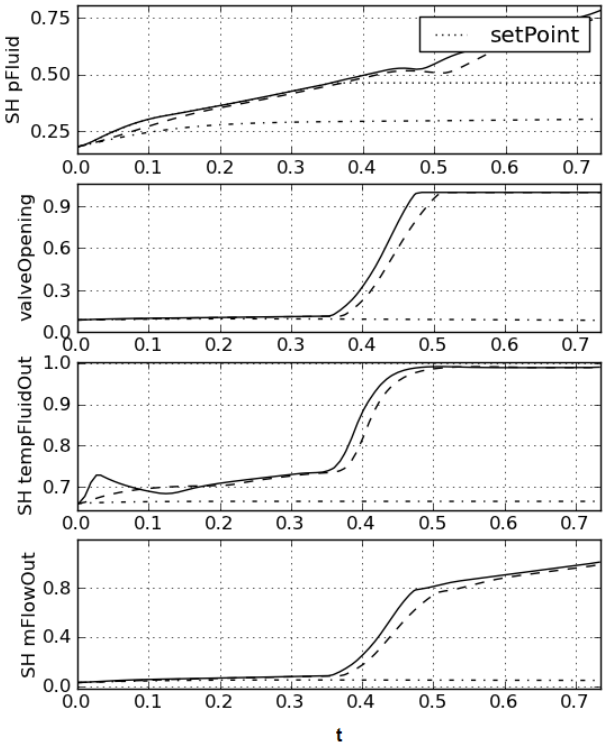

Figure 7.3: Optimal start-up trajectories for Optimization Problem 1: dashed (monotonically increasing power output) and solid (non monotonic power output) curves. Simulated initial guess trajectories: dash-dot curve. From top: the pressure in the superheater on the steam/water side and the pressure control loop set-point, the bypass valve opening in the pressure controller, the outlet temperature and the outlet mass flow from the superheater on the steam/water side. All results and times have been normalized.

the optimal solutions. The dotted line is the result of the first optimization, with a simulation result using a zero-load profile as initial guess trajectory. The number of elements, n_e, is 10 and the blocking factors are `[1,3,3,3]`. The dashed line shows the result of an optimization problem a couple of iterations later. Here `n_e=25` and `blocking_factors=ones(n_e)`. Finally the solid line shows the result with highest accuracy, with `n_e=45` and `blocking_factors=ones(n_e)`. The dotted/dashed line is the initial trajectory used for the first iteration.

It can clearly be seen how the optimizer finds a quite good solution already in the first attempt, from an initial guess trajectory very far from the optimal solution. The solution does however not reach full load. There is a quite large interval in the middle where the temperature gradient constraint is not active. A finer discretization makes the solution reach full load and a quick load peak can be observed in the beginning of the optimization. The dip in the gradient is substantially smaller and the constraint is

Table 7.1: Summary of results from Optimization Problem 1 showing the optimization `finalTime`, specification if the GT load was monotonic or non-monotonic, the number of elements (n_e), the objective function value for the optimal solution, the time it takes to reach 95% of full GT load and the maximum pressure on the water side in the HRSG.

`finalTime`	Non-/Monotonic	n_e	Objective Function	Time: 95 % GT Load	Max HRSG Pressure
1	Non Mono	10	1	Does not reach 95%	0.85
1	Non Mono	25	0.75	0.764	1
1	Non Mono	45	0.74	0.757	1
0.73	Non Mono	45	0.73	0.72	0.78
0.47	Non Mono	45	0.71	0.43	0.54
0.73	Mono	45	0.77	0.72	0.76

active at an earlier stage. When refining the discretization further the load peak is even higher, but except from that the two finer discretizations are essentially the same. The gradient increases even faster in the beginning but there is no difference in the dip.

7.2.3 The Impact of `finalTime`

Another parameter that effects the solution is the final time stated in the optimization problem. Figure 7.5 shows the solution of Optimization Problem 1 for three different final times. All solutions have been produced by iterations and have `n_e=45` and `blocking_factors=ones(n_e)`. The final times have been normalized. See Table 7.1 for for the objective function values, the time it takes to reach 95% of full GT load and the maximum pressure on the water side in the HRSG for the optimal solutions.

All solutions reach full load in time, the two solutions with shortest final time reach full load exactly at the final time stated, while the solution with the longest final time also has time to keep a stable input of full load for approximately 0.2 time units. It was possible for the optimization with final time = 0.47 to converge to full load since that final time coincides with a peak at the GT load profile.

The load profiles are very similar except from in the beginning where short load peaks can be seen. The peak is higher the shorter the final time is. There is a slight difference between the times where the temperature gradient becomes active in the beginning of the solution, this is probably due to the discretization. The gradient of the problem with the shortest final time becomes active first and the gradient of the problem with the longest final time becomes active last. There are no essential differences on the dip of the gradient constraint.

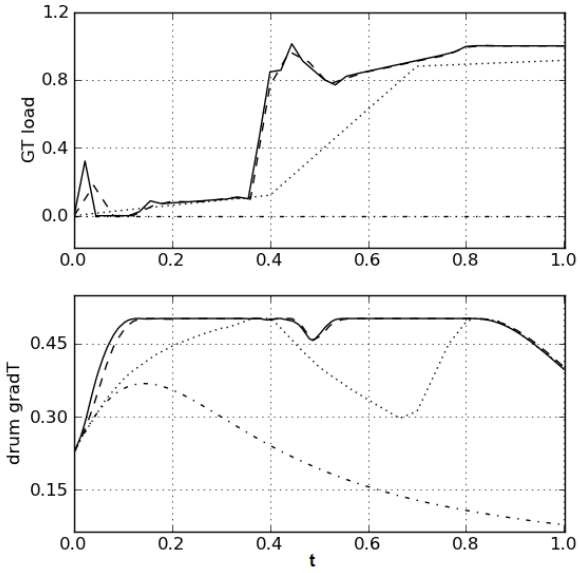

Figure 7.4: The impact on the result from the discretization. The dotted line: `n_e=10` and `blocking_factors=[1,3,3,3]`. The dashed line: `n_e=25` and `blocking_factors=ones(n_e)`. The solid line: `n_e=45` and `blocking_factors=ones(n_e)`. The dotted/dashed line is the initial trajectory used for the first iteration.

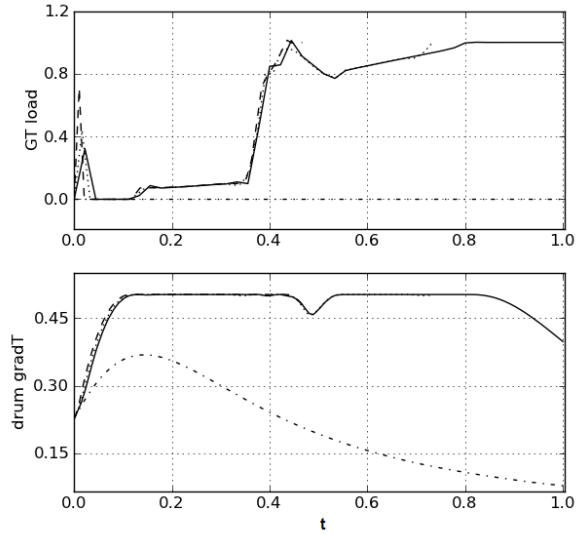

Figure 7.5: The impact on the result from the final time. The dashed line: final time = 0.47. The dotted line: final time = 0.73. The solid line: final time = 1. The dotted/-dashed line is the initial trajectory used for the first iteration. All times are normalized and all trajectories have `n_e = 45` and `blocking_factors=ones(n_e)`

7.3 Optimization Problem 2

When solving Optimization Problem 2 the input signal representing the power output load was defined as non monotonic. The second input, the opening of the bypass valve, could vary from closed to fully open with a derivative in the interval $[-0.5, 0.5]$.

The optimization is done iteratively, starting with a quite coarse discretization with $n_e = 10$ and ending with $n_e = 25$. The blocking factors are varied with the number of elements and `blocking_factors = ones(22)` is used when $n_e = 25$. The first optimization is initialized with guess trajectories resulting from simulations of the CCPP model with zero GT load. The optimization problem contains 28 continuous time states and 389 scalar equations. After collocation in JModelica.org the discrete NLP consists of 18772 equations when using 25 elements.

When analyzing the values of the weights in the cost function the same α and β values were used as in Optimization Problem 1. These values gave good convergence together with a non zero value of γ. The γ value was chosen so that the cost function was kept in the order of 1e4 or below and the contribution from the α term was weighted as most important. These cost function weights were then used in Optimization Problems 3 and 4.

The model has been successfully optimized to full load, see results in Figure 7.6 where the solid trajectory represents the solution of the optimization problem. See Table 7.2 for the objective function value, the time it takes to reach 95% of full GT load and the maximum pressure on the water side in the HRSG for the optimal solution.

The optimal time for the GT to reach 95% of full load is approximately $t = 0.75$, see Figure 7.6 and Table 7.2. The temperature gradient constraint is active from $t = 0.06$ and the GT load can not increase as rapidly as initiated after $t = 0.04$. At about $t = 0.12$ the GT load increases steadily at almost constant rate to not violate the temperature gradient constraint until it reaches its maximum value at $t = 0.84$. The dip in the temperature gradient that was observed in the 1 DOF case in Optimization Problem 1 is not observed. The gradient constraint is not completely active around $t = 0.1$ which most likely is due to the discretization.

The bypass valve is opened at $t = 0$ and is fully opened at $t = 0.12$, inducing that the power load can be increased more rapidly for $t < 0.5$, comparing to the results for Optimization Problem 1. Opening the bypass valve gives a decrease in pressure and since the temperature is directly coupled to the pressure this decreases the temperature in the evaporator. The temperature gradient in the drum decreases which gives the GT operational space to increase the load. After scaling the value of the objective function is 0.15, see Table 7.2, and this value also includes a contribution from the dv/dt term in the cost function.

The total power produced during the start-up procedure corresponds to the area under the graph of the power output. Even though the GT reaches full load later than in Optimization Problem 1, this model produces more GT power during the start-up, which shows the profit of using an extra degree of freedom.

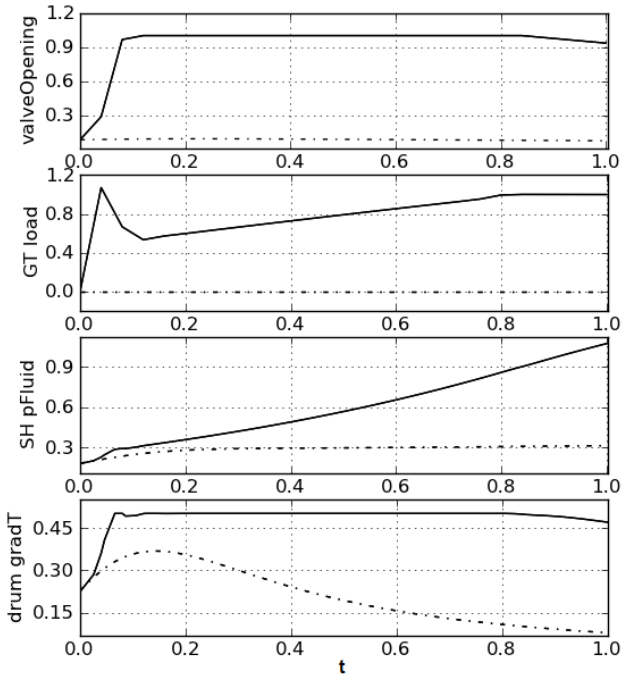

Figure 7.6: Optimal start-up trajectories for Optimization Problem 2: solid (non monotonic power output) curve. Simulated initial guess trajectories: dash-dot curve. From top: the bypass valve opening, the GT power output, the pressure in the superheater on the steam/water side and the temperature gradient in the wall of the drum. All results and times have been normalized.

Table 7.2: Summary of results from Optimization Problems 2-3 showing the objective function value for the optimal solution, the time it takes to reach 95% of full GT load and the maximum pressure on the water side in the HRSG.

Optimization Problem	Objective Function	Time: 95% GT Load	Max HRSG Pressure
2	0.15	0.75	1.07
3	0.34	0.45	0.82
4	0.47	0.56	0.81

7.4 Optimization Problem 3

Optimization Problem 3 has been successfully optimized to full load, see results in Figures 7.7 and 7.8, where the dashed trajectory represents the solution of Optimization Problem 3. See Table 7.2 for the objective function value, the time it takes to reach 95% of full GT load and the maximum pressure on the water side in the HRSG for the optimal solution. The optimization was done iteratively, starting with $n_e = 10$ and ending with $n_e = 25$. The blocking factors were varied with the number of elements and `blocking_factors = [1,1,1,1,1,2,2,3,5,8]` was used when $n_e = 25$. The Optimization Problem consists of 569 scalar equations and 39 continuous time states. After collocation in JModelica.org the discrete NLP consists of 26824 equations when using 25 elements.

Constraints on the header and drum walls were used to keep the temperature gradients under the critical levels $\nabla T_{maxDrum}$ and $\nabla T_{maxHeader}$:

$$|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}|$$
$$|T_{wallHeader} - T_{SH}| < |\nabla T_{maxHeader}|.$$

The degrees of freedom were du/dt and dv/dt. The GT load input u was non-monotonic and the bypass valve was controlled in the optimization so that the opening of the bypass valve could vary from closed to fully open with a derivative in the interval $[-0.5, 0.5]$. The optimal time for the GT to reach 95% of full load was approximately $t = 0.45$, see Figure 7.7 and Table 7.2.

The GT load can not increase as rapidly as initiated after $t = 0.04$ since at the end time of the second block ($0.04 < t \leq 0.08$, since the degree of freedom du/dt is piecewise constant) the header constraint is active. The header temperature gradient constraint is active from $t = 0.08$ until $t = 0.3$. The drum temperature gradient constraint is active at different times from $t = 0.16$ and it is the only active temperature gradient constraint when $t > 0.3$. Around $t = 0.6$ the drum temperature gradient constraint is active for the longest time sequence.

From $t = 0.12$ the GT load increases with a rate that varies to not violate the header drum constraint. After $t = 0.2$ the GT load increases with a steady almost constant rate until it reaches about 80% of full load at $t = 0.34$. The drum temperature gradient constraint is not active during this time period. From $t = 0.34$ the GT load increases at a low rate to not violate the drum temperature gradient constraint until it reaches its maximum value of 1 at $t = 0.88$. The bypass valve is opened at $t = 0$ and is fully opened at $t = 0.48$. The valve though closes at $t = 0.08$ giving a rise in the HRSG pressure and the drum temperature gradient.

After scaling the value of the objective function is 0.34, see Table 7.2, and this value also includes a contribution from the dv/dt term in the cost function.

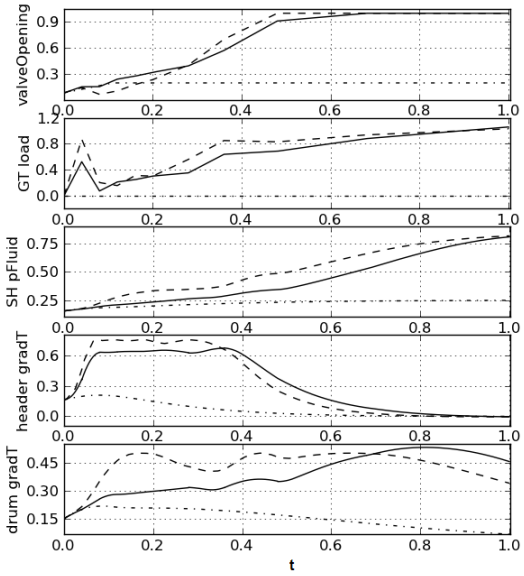

Figure 7.7: Optimal start-up trajectories for Optimization Problems 3 and 4: dashed (Optimization Problem 3) and solid (Optimization Problem 4) curves. Simulated initial guess trajectories: dash-dot curve. From top: the bypass valve opening, the GT power output, pressure in the superheater on the steam/water side, the temperature gradient in the wall of the header and the temperature gradient in the wall of the drum. All results and times have been normalized.

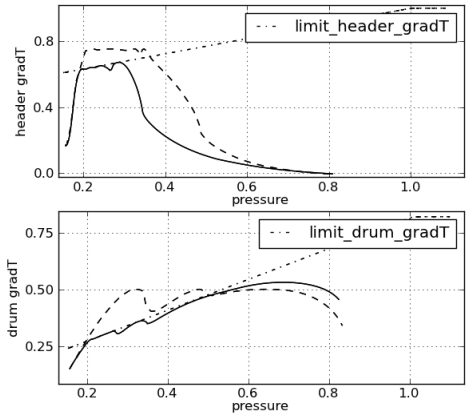

Figure 7.8: Optimal start-up trajectories for Optimization Problems 3 and 4: dashed (Optimization Problem 3) and solid (Optimization Problem 4) curves. Maximum constraints considered in the basic stress model: dash-dot curve. From top: the temperature gradient in the wall of the header as a function of pressure and the temperature gradient in the wall of the drum as a function of pressure. All results and times have been normalized.

7.5 Optimization Problem 4

In Optimization Problem 4 the model has successfully been optimized to full load, see results in Figures 7.7 and 7.8, where the solid trajectory represents the solution of Optimization Problem 4. See Table 7.2 for the objective function value, the time it takes to reach 95% of full GT load and the maximum pressure on the water side in the HRSG for the optimal solution. The optimization result was obtained by iterating the optimization process starting with a simulated initial guess trajectory with zero GT power load. The number of elements were increased from $n_e = 10$ to $n_e = 25$ and the blocking factors `blocking_factors = [1,1,1,1,2,2,3,5,8]` was used when $n_e = 25$. The optimization problem consists of 576 scalar equations and 39 continuous time states. After collocation in JModelica.org the discrete NLP consists of 27132 equations when using 25 elements.

Constraints on the header and drum walls were used to keep the temperature gradients under the critical levels defined by the basic stress model:
Drum:

$$|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}(p)|$$
$$\nabla T_{maxDrum}(p) = \min(\nabla T_{Drum}(p), \nabla T_{maxDrum}),$$

Header:

$$|T_{wallHeader} - T_{SH}| < |\nabla T_{maxHeader}(p)|$$
$$\nabla T_{maxHeader}(p) = \min(\nabla T_{Header}(p), \nabla T_{maxHeader}),$$

see Figure 6.3.

The GT load input u was non-monotonic and the bypass valve was controlled in the optimization so that the opening of the bypass valve could vary from closed to fully open with a derivative in the interval $[-0.5, 0.5]$. The degrees of freedom were du/dt and dv/dt.

The optimal time for the GT to reach 95% of full load was approximately $t = 0.56$, see Figure 7.7 and Table 7.2. Like in Optimization Problem 3 the GT load can not increase as rapidly as initiated after $t = 0.04$, since at the end time of the second block ($0.04 < t \leq 0.08$, since the degree of freedom du/dt is piecewise constant) the header constraint is active, see Figures 7.7 and 7.8.

The header temperature gradient constraint is active from pressures $p = 0.19$ to $p = 0.25$ corresponding to the time period $t = 0.07$ until $t = 0.24$. The drum temperature gradient constraint is active from pressures $p = 0.2$ to $p = 0.55$ corresponding to the times $t = 0.09$ to $t = 0.7$. The GT load is thus more constrained at lower pressures than in Optimization Problem 3, see Figure 7.8.

The basic stress model allows the drum temperature gradient to attain larger values when $t > 0.7$ (the pressure in the HRSG is larger than 0.55) compared to the constant constraint used in Optimization Problem 3. The header is though harder constrained in Optimization Problem 4 compared to Optimization Problem 3 giving a lower rate of increase of the GT load compared to Optimization Problem 3. From $t = 0.08$ the GT load increases with a rate that varies to not violate the header drum constraints. The bypass valve is opened at $t = 0$ and is fully opened at $t = 0.65$. The pressure is kept at low values when the temperature gradient constraints are active and the pressure can increase at a higher rate when $t > 0.5$.

After scaling the value of the objective function is 0.47, see Table 7.2, and this value also includes a contribution from the dv/dt term in the cost function.

7.6 Optimization Problem 5

Optimization Problem 5 uses the cost function given by the minimal time formulation and this problem has been successfully optimized to full load, see results in Figure 7.9. The optimization was done iteratively using $n_e = 10$ and a variation of the blocking factors. `blocking_factors = [1,1,2,3,3]` was used for the result presented here. The Optimization Problem consists of 292 scalar equations and 39 continuous time states. After collocation in JModelica.org the discrete NLP consists of 11135 equations when using 10 elements.

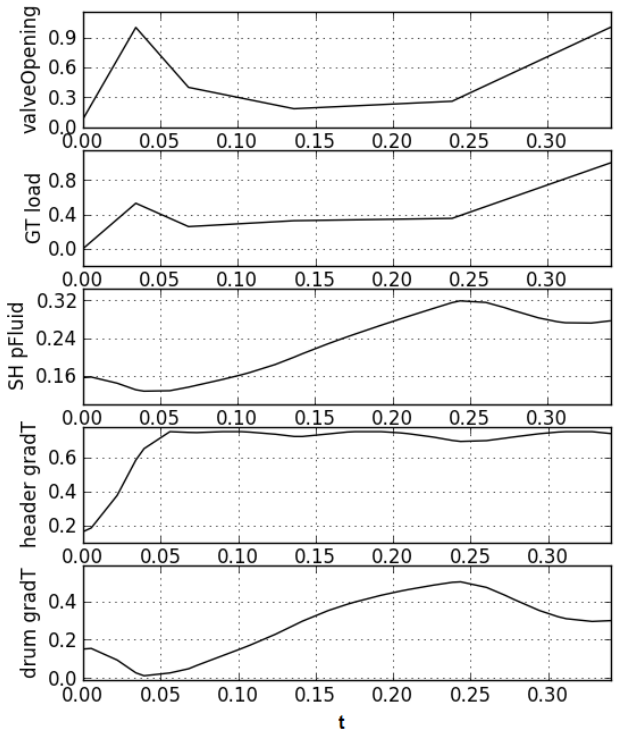

Figure 7.9: Optimal start-up trajectories for Optimization Problem 5. From top: the bypass valve opening, the GT power output, the pressure in the superheater on the steam/water side, the temperature gradient in the wall of the header and the temperature gradient in the wall of the drum. All results and times have been normalized.

The degrees of freedom were du/dt and dv/dt. The GT load input u was non-monotonic and the bypass valve was controlled in the optimization so that the opening of the bypass valve could vary from closed to fully open with a derivative in the interval $[-0.5, 0.5]$. Constraints on the header and drum walls were used to keep the temperature gradients under the critical levels $\nabla T_{maxDrum}$ and $\nabla T_{maxHeader}$:

$$|T_{wallDrum} - T_{evap}| < |\nabla T_{maxDrum}|$$
$$|T_{wallHeader} - T_{SH}| < |\nabla T_{maxHeader}|.$$

The final time variable in the optimization, t_f was defined with the following `min`, `max` and `nominal` attributes:

```
finalTime(free=true,min=0.13,initialGuess=0.87,max=2.67,nominal=0
    .67).
```

The optimal time for the GT to reach 100% of full load was approximately $t = 0.34$, see Figure 7.9. The objective function value is with this formulation the value of the minimized t_f in the optimization.

The GT load can not increase as rapidly as initiated after $t = 0.03$ since at the end time of the second block $(0.03 < t \leq 0.07)$ the header constraint is active. The header temperature gradient constraint is active from $t = 0.06$ and until $t = 0.33$. The drum temperature gradient constraint is active at $t = 0.24$. From $t = 0.07$ the GT load increases with a rate that varies to not violate the header constraint. After $t = 0.07$ the GT load increases with a steady almost constant rate until it reaches about 40% of full load at $t = 0.24$. The drum temperature gradient constraint is not active during this time period. From $t = 0.24$ the GT load increases at a high rate that does not violate the drum and header temperature gradient constraint until it reaches its maximum value of 1 at $t = 0.34$.

The bypass valve is opened at $t = 0$ and is fully opened at $t = 0.34$. The valve though closes at $t = 0.04$. The pressure is only increased to 0.28 and does therefore not reach the value defined as the operational pressure (=1 when normalizing).

7.7 Simulation of Optimization Results

To verify the accuracy of the optimized solutions the optimization results have been used to simulate the CCPP models. The optimal control profiles have been used as input when simulating the system. A longer time horizon than what has been optimized has been simulated to see how the system reacts when operating the GT at full load. All simulations using optimization results as input and the actual optimization results are very similar, which verifies that the accuracy is fine enough with the discretization used.

From Optimization Problem 1 the GT load profile was extracted and used as input when simulating the CCPP1 model, see simulation results in Figures 7.10 and 7.11. In Figure 7.10 the optimization results were obtained using the optimization final time $t = 1$ and in Figure 7.11 the optimization results had $t = 0.47$ as final time, see optimization results in Section 7.2.3. The simulation time was in both cases $t = 2$ and the final value of GT power load, u, from the optimization was held constant for the rest of the simulation. The non-controlled phase (phase 1) has been plotted as dashed and the simulation results of the controlled phase (phase 2) are plotted as solid.

The control profiles from the optimal solution for Optimization Problem 5 were used as inputs when simulating the CCPP3 model, see simulation results in Figure 7.12. From the optimal solution for Optimization Problem 5, when the optimization final time was $t = 0.34$, the profiles of the GT power load, u, and the valveOpening, v, were extracted and used as inputs for the controlled phase (phase 2). The simulation time was $t = 1.67$ and the final values of the GT power load, u, and the valveOpening, v, were held constant at their values at the final time of the optimization until the end of the simulation. The dash-dot curve is the non-controlled phase (phase 1), the solid curve is the controlled phase (phase 2) without further opening of the bypass valve and

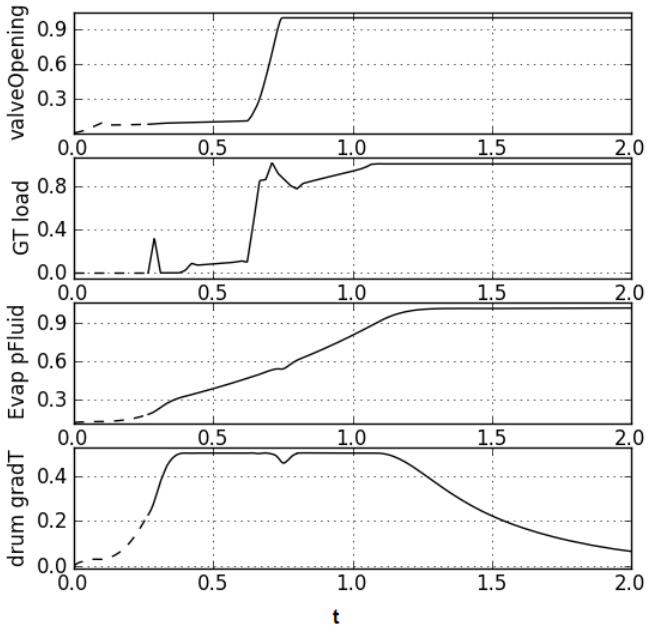

Figure 7.10: Simulation of optimal solution for Optimization Problem 1 when the optimization final time was 1 s: dashed curve non-controlled phase (phase 1) and solid curve controlled phase (phase 2). From top: the bypass valve opening, the GT power output, the pressure in the superheater on the steam/water side, the temperature gradient in the wall of the drum. All results and times have been normalized.

the dashed curve is the controlled phase (phase 2) with further opening of the bypass valve.

When ramping up the GT like simulated in Figures 7.11 and 7.1 the start-up of phase 2 violates the drum temperature gradient constraint (=0.5) in a later stage of the start-up when the optimization phase has finished. The drum temperature gradient can be held below its defined maximum value by opening the bypass valve further after the optimization phase has ended, see dashed curve in Figure 7.12. This modification of the bypass valve keeps the constraints fulfilled when operating the GT at full load.

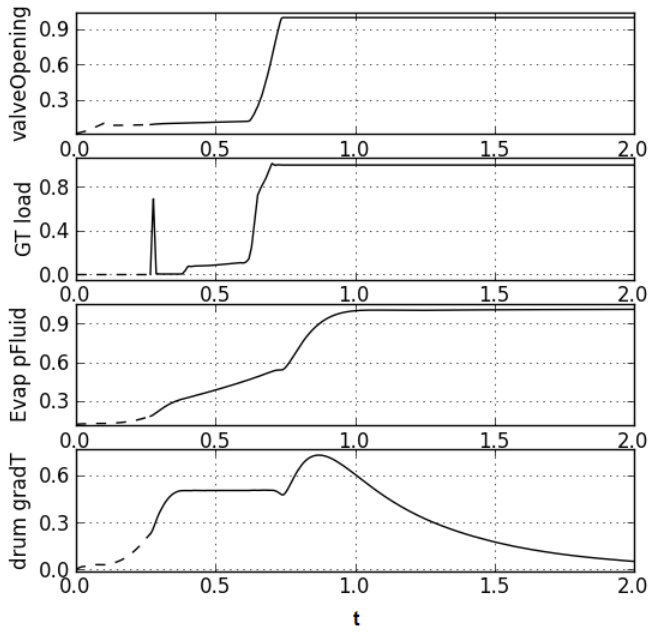

Figure 7.11: Simulation of optimal solution for Optimization Problem 1 when the optimization final time was 0.47 s: dashed curve non-controlled phase (phase 1) and solid curve controlled phase (phase 2). From top: the bypass valve opening, the GT power output, the pressure in the superheater on the steam/water side, the temperature gradient in the wall of the drum. All results and times have been normalized.

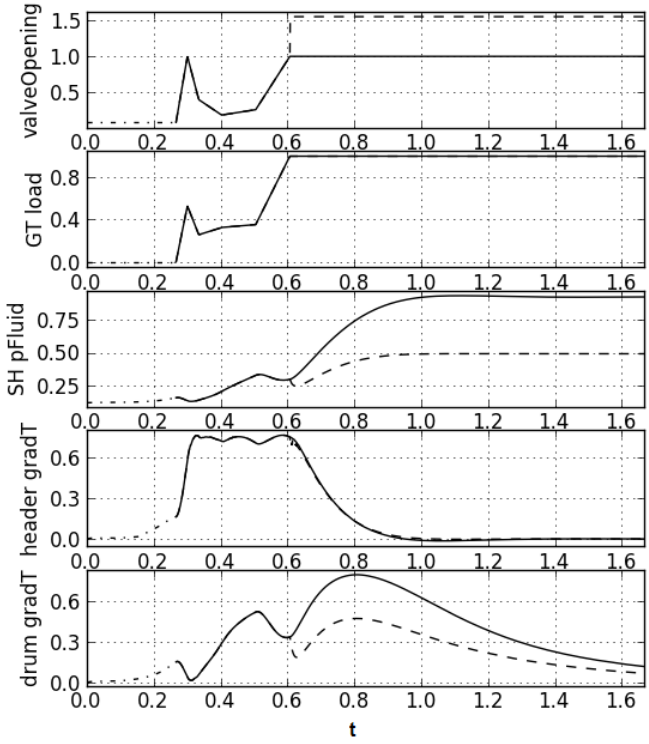

Figure 7.12: Simulation of optimal solution for Optimization Problem 5 when the optimization final time was 0.34 s: dash-dot curve non-controlled phase (phase 1), solid curve controlled phase (phase 2) without further opening of the bypass valve and dashed controlled phase (phase 2) with further opening of the bypass valve. From top: the bypass valve opening, the GT power output, the pressure in the superheater on the steam/water side, the temperature gradient in the wall of the header and the temperature gradient in the wall of the drum. All results and times have been normalized.

Chapter 8

Discussion

When starting up a power plant, the most desirable goal does not necessarily have to be to reach full load as fast as possible. To achieve as much power output as possible during the start-up procedure could be just as important. The results show how a larger amount of produced power during start-up can be achieved when adding the opening of the bypass valve as a degree of freedom.

8.1 Optimization Problems 1 and 2

When using the pressure controller in Optimization Problem 1 (1 DOF), it has been shown to function in a far from optimal way since the pressure is controlled so that the load cannot increase during the first 0.35 t.u. There is a delay in the heat transfer to the drum after the GT load has been started to load up and the dip in the temperature gradient can not be eliminated when using the pressure control loop in Optimization Problem 1 since an increase of the GT load at this time could give an overshoot of the temperature gradient in the drum. The set point of the controller could be modified so that the bypass valve can be opened earlier in the start-up, lowering the pressure in the HRSG and giving the load more operational space where it does not violate the gradient constraint. When the bypass valve opening is used as a degree of freedom in Optimization Problem 2 (2 DOF) the valve is opened earlier, lowering the pressure in the HRSG and allowing the load to be increased earlier in the start-up. The results from Optimization Problem 2 produces the most power during the start-up and the benefits from using two degrees of freedom instead of one is clear.

To get a realistic model, control loops that are used in the real plant need to be included in the optimization. An important step towards improving the models would be to work on the tuning of control loops and to adapt them to optimization purposes of the optimization problem considered. This could be done before optimizing the model but could as well be explored when working with the optimization since values of set points etc. can be set depending on how the optimizer handles the control loop. In the project an attempt was made to control the evaporator level with a control loop, but due to robustness properties the ideal level control evaporator was proved to simplify the optimization process.

Analysis of the impact from the the final time variable in Optimization Problem 1 shows that the load profiles all follow the same trajectory for $t < 0.47$. Decreasing the final time only refines the discretization and takes the process dynamics during

this horizon into account. It seems important to try many different final time values to understand the process and understand how the input used during the optimization phase affects the system dynamics later on in the start-up. What discretization that is needed in the optimization depends on the process dynamics and using 25 number of elements has been shown to be enough when optimizing this process.

The Optimization Problem 2 model produces more steam in an earlier phase of the start-up due to the faster ramp up of the GT load. It is though not taken into account in this thesis if the produced steam is of sufficient quality to start the third phase of the start-up procedure; the loading of the ST. A more complete picture of the efficiency of the start-up could be attained by modifying the objective function and adding more complex and thorough descriptions of possible objectives, so that the efficiency is maximized and the economical costs during the whole start-up transient is minimized. Thus the economical aspects of not only the load produced could be taken into account.

8.2 Optimization Problems 3 and 4

When comparing Optimization Problems 3 and 4, it is clear that the time to reach full load is more or less the same even though Optimization Problem 4 is harder constrained at lower pressures comparing to Optimization Problem 3. The optimizer compensates by using the full potential of the drum gradient constraint for $t > 0.7$. The rate of increase of the GT load is slightly larger for Optimization Problem 4 when $t > 0.7$ and it can be observed that the pressure increases at a higher rate than in Optimization Problem 3. Even though the time to reach full load is approximately the same the profile of Optimization Problem 4 keeps the lifetime consumptions of the stressed components at a level used in actual power plant controls.

The temperature gradient in the header is dependent on the mass flow through the bypass valve and the higher mass flow the larger temperature gradient. In Figure 7.8 it can be seen that when the header gradient constraint is active the optimizer keeps the valve opening fairly small but as soon as the constraint is not active any more it opens it and the pressure is increased to the working pressure.

The header constraint is active earlier in the start-up phase comparing to the drum constraint. This is due to that the hot exhaust gas from the GT enters the header first when the gas is of the highest temperature. The exhaust gas reaches the drum with a time delay and the exhaust gas is of lower temperature than when reaching the header. The drum consists of water in the two phase region and the gradient is therefore coupled to the pressure in the component. Steam is though decoupled from pressure and the header temperature gradient is more dependent on the GT exhaust gas temperature than the pressure.

The basic stress model used in Optimization Problem 4 uses constraints that are typically used in power plant control. Since Optimization Problem 3 violates these constraints, the result from Optimization Problem 4 is the most preferable choice.

8.3 Optimization Problem 5

The result of Optimization Problem 5 is quite different from the other solutions with 2 DOF. This is most obvious when observing the valve opening which opens rapidly to almost fully opened at the beginning of the start-up and then closes to 1/3 for a while,

to open fully again in the end. This is probably due to the header constraint which the optimizer tries to keep down by limiting the mass flow through the valve.

A problem with the minimal time formulation used in this problem is that there are no constraints that are coupled to the HRSG pressure. The solution found only ramps up the pressure to about 40 % of the working pressure and the drum gradient constraint is therefore barely active. A better formulation would be to put a final time constraint on the pressure or the steam quality so that the ST has the best conditions to start up as fast as possible.

8.4 Simulation of Optimization Results

Simulation of the start-up transients after the optimization has proved to give important information about the results obtained, regarding where in the complete start-up procedure this solution finishes. Is it possible to start up the GT optimally where phase 2 ends or is it necessary to ramp up the pressure in a certain way before phase 3 can start? When simulating the results from Optimization Problem 1, with a short final time for the optimization, the optimization does not take care of the ramp up of the pressure and therefore the temperature gradient constraint is violated when keeping the GT at full load.

The same phenomena has been observed when using the minimal time formulation and the only way to keep the temperature gradient constraint when simulating was proven to be to open the bypass valve to more than 100 %. This can be interpreted as adding another valve to the process that can be opened after phase 2 has finished and another process takes by. Since phase 3 is not modeled in this thesis it is difficult to say how this process would handle the effects of the optimal input profiles used in phase 2.

The minimal time solution has shown that the GT can be ramped up to full load in a very short time with only one active constraint in the header. At such low pressures the drum constraints is not active and an alternative formulation of phase 2 could be to divide the GT load ramp up into two separate processes. One where only the header constraint is restricting and that this process finishes when the GT is at full load. The next process would then include how the ST could be started and ramped up optimally taking into account how the most optimal ramp up of the pressure is obtained so that the steam quality is sufficient to start the ST.

8.5 Additional Remarks

None of the optimization problems formulated in this thesis put any penalization or even uses du/dt in the cost function. The problem is still well defined thanks to the constraints on the temperature gradients that limit the input du/dt. The weight on dv/dt could possibly have been increased in Optimization Problem 2 since the valve is not kept fully opened in the end of the optimization. Not penalizing the input can give ill-defined problems that can have several different solutions.

The rate of increase of the GT power load has been unlimited in all optimizations done in this thesis. This and the fact that the GT power load is allowed to decrease gives a peak in the GT power load in the beginning of the start-up phase for all non-monotonic GT load cases. From an optimization point of view this is a satisfying result, since it is clear that the optimizer is trying to make the load reach its reference value as fast as possible. The GT load must though decrease to not violate the temperature

gradient constraints. In actual power plants, such a fast increase in load could damage the GT or even not be physically applicable. By penalizing the use of the du/dt input such peaks could be avoided in the optimization.

The increase- and decrease rate of the GT power load has deliberately been assumed to be larger than the standard of today's GT:s. Because of this and the fact that simplified CCPP models have been used, the results obtained in this thesis are not applicable in power plants of today but is a suggestion of how to handle the constraints that limit the start-up. Increasing the allowed rate of the GT has been shown to decrease the start-up time in the models and a plausible method of operating the GT could be to allow decreases in the power load rate during start-up.

Chapter 9

Experiences with JModelica.org

An important part of this thesis has been to test the optimization models and optimization problem formulations. The work has been in cooperation with the developers of the optimization tool JModelica.org. We will in this section comment on which issues during the project that were challenging. The solutions and strategies that were worked out during the thesis will also be described.

9.1 Iterative Optimization Method

The iterative method described in Section 6.6 *Initialization* was developed in a quite late stage of the project. This method, however, made the procedure of finding a solution with a fine discretization considerably easier than before. One method for varying the number of elements and blocking factors that sometimes was successful was to include the number of elements that were added in one block at the end of the optimization interval:

$$\begin{cases} \texttt{n\_e = 10} \\ \texttt{blocking\_factors = [1,3,3,3]} \end{cases} \Longrightarrow \begin{cases} \texttt{n\_e = 15} \\ \texttt{blocking\_factors = [1,3,3,3,5] .} \end{cases} \tag{9.1}$$

The iterative technique could preferably be automatized in next coming projects, for example by constructing a loop which automatically chooses the number of elements and blocking factors for the next optimization problem, depending on whether the previous optimization has converged or not and for what discretization it converged.

9.2 Scaling

Scaling turned out to be a very important issue when optimizing. In order to find a converging optimization solution the optimization problem should be well defined and well scaled, see sections 5.2 *Scaling* and 6.3 *Cost Functions*. With no scaling, different parameters have values of different magnitudes during simulation and optimization. This can give problems with the convergence and therefore the aim has been to scale all variables so that they are in the interval from 1e-4 to 1e4. In this thesis, the scaling has been done using only the nominal values in Modelica. A better way to scale the variables would be to shift the variables with their mean values before scaling with a nominal value, which then would be of the same order of magnitude as the standard

deviation. To simplify the scaling procedure a python script for checking the variable scaling was used.

9.3 New Compiler Option

As mentioned in Section 4.4.2 *Model Objects*, a new compiler option called `state_initial_equations` has been added to the compiler during the project. This option makes it possible to initialize an optimization from a previous simulation or optimization result which in the beginning of the project was not possible in JModelica.org.

9.4 Interaction with the JModelica.org Developers

The additional features in JModelica.org that have been added during the project are resulting from the continuous interaction between the tool developers and us. To be a part of the development of the JModelica.org platform has been a very interesting part of the thesis.

9.5 Complementary Tools

Two tools have been used in the project; Dymola and JModelica.org. Dymola has mainly been used for the CCPP model development in this project. The collaborating group at Siemens AG, Energy Sector, prefers the Modelica modeling language for simulation purposes and therefore it was natural to use Dymola as modeling tool. When considering optimization the Modelica extension Optimica together with the optimization and simulation tool JModelica.org has proved to be a well suited environment.

The possibility to start with a Modelica simulation model developed in any Modelica based tool is an advantage with this optimization technique. An external optimization problem is then added to the Modelica model and the model can be optimized in JModelica.org. This method of optimizing is a very straightforward way of working with optimization and models from different contributors can be combined with this technique.

9.6 Project Management

When optimizing Modelica models from different contributors a system that separates the modeling group's work from the optimization group's work would be preferable when using an optimization tool that is under development. To be able to analyze the JModelica.org tool stability separately from the optimization models stability and robustness changes in the different environments should be logged and handled separately. In the project a system using *trac* [32] and *Subversion (svn)* [33], where it is possible to log and track all changes to models and scripts, has been used.

In this project the models that were optimized were still under development and some model changes affected the properties and the convergence of the optimization problems. We suggest that the optimization group *checks out* the modeling groups models when they are considered by the modeling group to be of high enough quality. If modeling errors are found in a later stage the optimization group should be informed and these changes should be made while determining if the changes affect the stability

of the optimization problem. In the end of this project, the models were considered fixed to be able to analyze optimization properties without influence of model changes.

9.7 Desired Features in JModelica.org

The tool used for model development in this project would not necessarily have had to be Dymola. Any other tool supporting the Modelica language could as well have been used. The models could also have been implemented directly in the Optimica files. A desired additional feature in JModelica.org would be some kind of graphical interface that is comparable to Dymola. This would make JModelica.org an attractive tool for model development as well as for simulation and optimization.

Chapter 10

Conclusions

In this thesis it has been shown how a start-up procedure of a combined cycle power plant can be optimized with respect to the start-up time and the power production during start-up, using JModelica.org. The thermal stress in the heat recovery steam generator has been considered as the most limiting constraint when starting up the GT to full load, i.e. its maximum power output.

Five types of optimization models have been considered; one where the load u of the GT is the sole degree of freedom and four where both the load u and the opening v of the bypass valve are degrees of freedom. All optimization problems have been successfully controlled to their maximum power output reference value of 100% and it has been observed that by adding the opening of the bypass valve as degree of freedom a larger amount of power during start-up is produced.

The models have been adapted to suit optimization purposes concerning the start-up of the GT and thus the ST has not been modeled. The next step towards achieving more realistic results could be to close the steam cycle and to include more components in the model. More constraints could as well be used and additional degrees of freedom could be added. It has not been taken into consideration when it is most optimal to start the ST and if the optimization of the GT loading should take this factor into account. One improvement could thus be to find when, during the start-up procedure, the ST should be started and to determine when and how much of the steam should pass the bypass valve. Another improvement could be to include economical aspects and minimize the fuel spent during start-up while maximizing the produced power. The work presented in this thesis is one step towards an optimal power plant control and could be used with an on-line strategy such as model predictive control.

Bibliography

[1] Lind, A., Sällberg, E., Velut, S., Gallardo Yances, S., Åkesson, J., Link, K. Start-up Optimization of a Combined Cycle Power Plant. In: Modelica Conference, 2012.

[2] Casella, F. and Pretolani, F. Fast Start-up of a Combined-Cycle Power Plant: A Simulation Study with Modelica. In: Modelica Conference, pp. 3-10, Vienna, Austria, 2006.

[3] Casella, F., and Leva, A. Modelica open library for power plant simulation: design and experimental validation. In: Proceedings of 3rd International Modelica Conference, pp. 41-50. Linkoping, Sweden, 2003.

[4] Casella, F., Farina, M., Righetti, F., Scattolini, R., Faille, D., Davelaar, F., Tica, A., Gueguen, H. and Dumur, D. An optimization procedure of the start-up of combined cycle power plants. In: 18th IFAC World Congress, pp. 7043-7048. Milano, Italy, 2011.

[5] Shirakawa, M., Nakamoto, M. and Hosaka, S. Dynamic simulation and optimization of start-up processes in combined cycle power plants. In: JSME International Journal, vol. 48 (1), pp. 122-128, 2005.

[6] Dassault Systemes. Dymola, http://www.3ds.com/products/catia/portfolio/dymola, 2012, viewed 2012-06-12.

[7] Casella, F., Donida, F. and Åkesson, J. Object-oriented modeling and optimal control: a case study in power plant start-up. In: 18th IFAC World Congress, pp. 9549-9554. Milano, Italy, 2011.

[8] Siemens AG. http://www.siemens.com, 2012, viewed 2012-06-12.

[9] Kehlhofer, R., Warner, J., Nielsen, H., Bachmann, R. Combined-Cycle Gas and Steam Turbine Power Plants, second edition, PennWell Publishing Company, 1999.

[10] Siemens AG. Press release, http://www.siemens.com/press/en/presspicture/?press=/en/presspicture/pictures-photonews/2010/pn201002/pn201002-02.htm, 2012, viewed 2012-06-12.

[11] Biegler, L., Cervantes, A., Wachter, A. Advances in simultaneous strategies for dynamic optimization, Chemical Engineering Science 57, pp. 575-593, 2002.

[12] Biegler, L. Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes, SIAM, 2010.

[13] Biegler, L., Wächter, A. On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming, Mathematical Programming 106(1), pp. 25-57, 2006.

[14] Modelica Association. The Modelica Association, https://www.modelica.org, 2012, viewed 2012-06-12.

[15] Fritzon, P. Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, Wiley-Interscience, 2004.

[16] Otter, M. Modelica Overview, https://modelica.org/education/educational-material/lecture-material/english/ModelicaOverview.pdf, 2009, viewed 2012-06-12.

[17] Modelon AB. JModelica Home Page. http://www.jmodelica.org, 2009, viewed 2012-06-12.

[18] Åkesson J. Languages and Tools for Optimization of Large-Scale Systems, PhD Thesis ISRN LUTFD2/TFRT–1081–SE, Regler, 2007.

[19] Python Software Foundation. Python Programming Language - Official Website, http://www.python.org/, 2012, viewed 2012-06-12.

[20] PyLab Home Page. http://www.scipy.org/PyLab, 2010, viewed 2012-06-12.

[21] Enthought, I. SciPy, http://www.scipy.org, 2012, viewed 2012-06-12.

[22] Oliphant, T. Numpy Home Page, http://numpy.scipy.org/, 2012, viewed 2012-06-12.

[23] Hunter, J., Dale, D., Droettboom, M. matplotlib: python plotting, http://matplotlib.sourceforge.net/, 2008, viewed 2012-06-12.

[24] Åkesson, J., Årzen, K.E., Gafvert, M., Bergdahl, T., and Tummescheit, H. Modeling and optimization with Optimica and JModelica.org - languages and tools for solving large-scale dynamic optimization problems. In: Computers and Chemical Engineering, vol. 34 (11), pp. 1737-1749, 2010.

[25] Modelon AB. Assimulo. http://www.jmodelica.org/assimulo, 2009, viewed 2012-06-12.

[26] Andersson, C., Åkesson, J., Führer, C., Gäfvert, M. Import and Export of Functional Mock-up Units in JModelica.org. In: 8th International Modelica Conference 2011. Modelica Association, 2011.

[27] Modelon AB. JModelicaUsersGuide-1.6.0, http://www.jmodelica.org/page/236, 2009, viewed 2012-06-12.

[28] Modelon AB. JModelica Trac, http://trac.jmodelica.org/ticket/1810, 2012, viewed 2012-06-12.

[29] Sundials. https://computation.llnl.gov/casc/sundials/main.html, 2012, viewed 2012-06-12.

[30] Wächter, A. IPOPT Documentation, http://www.coin-or.org/Ipopt/documentation/, 2010, viewed 2012-06-12.

[31] STFC Rutherford Appleton Laboratory, Harwell Subroutine Library. MA27 Documentation, http://www.hsl.rl.ac.uk/, 2012, viewed 2012-06-12.

[32] Edgewall software. trac, Integrated SCM and project Management http://trac.edgewall.org/, 2012, viewed 2012-06-21.

[33] TortoiseSVN. http://tortoisesvn.net/, 2012, viewed 2012-06-21.