LUND UNIVERSITY

Operator Interaction and Optimization in Control Systems

Åkesson, Johan

2003

Document Version:
Publisher's PDF, also known as Version of record

Operator Interaction and Optimization in Control Systems

Johan Åkesson

Department of Automatic Control
Lund Institute of Technology
Lund, December 2003

To Katarina - my loving wife and partner

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 LUND
Sweden

ISSN 0280–5316
ISRN LUTFD2/TFRT--3234--SE

Printed in Sweden,
Lund University, Lund 2003

Contents

Acknowledgments

In the course of the work presented in this thesis, many people have helped me and inspired me. I would like to take this opportunity to express my gratitude to you all.

First I would like to thank my supervisor Karl-Erik Årzén for skillfully guiding me through my first three years as a PhD student. Karl-Erik is a never-ending source of ideas and inspiration, and I look forward to continue this collaboration. As my second supervisor, Per Hagander has been the ultimate discussion partner, and I have very much enjoyed our, not always control related, discussion sessions. I also would like to thank Karl Johan Åström for bringing my attention to the field of automatic control. When Karl Johan hired me as a trainee during the summer of 1999, he also introduced me to the exciting field of control, and for this I am very grateful. Further, would like to thank Tore Hägglund for valuable comments regarding my work on Grade Change Control.

During the academic year of 1999/2000 I was an exchange student at the University of California, San Diego. During this time I had the privilege to attend courses taught by Robert E. Skelton and Robert Bitmead. My experiences from theses courses strongly affected my decision to apply for the PhD program in Lund. I am also indebted to Robert E. for inviting me to work with his research group in the spring of 2000.

It is a very pleasant experience to work at the Department of Automatic Control in Lund. I would like to thank my friends and colleagues at the Department for making my working days brighter and for eagerly discussing whatever issue I have come up with - control-oriented or not. Henrik Sandberg is the perfect room-mate. Apart from being a terrific source of information about linear systems, he is also the DJ of our room. He even plays Gessle from time to time and, needless to say, he is thus a very good DJ. Pop on! The work on Flexible Real-Time Scheduling of MPC Controllers presented in Chapter 6 is a joint work with Dan Henriksson, a collaboration I have very much enjoyed.

Leif Andersson keeps our computer systems healthy - his work has undoubtly saved me many hours of computer hassle. Also, Leif is a keen teacher of the mysteries of Linux, and I very much appreciate his teaching efforts. Anders Blomdell knows the answer to any questions

about programming (at least as far as I can tell), and I would like to thank him for generously sharing his knowledge with me whenever I run into a syntax error. I also would like to thank Rolf Braun for letting me mess around in his lab, it has been very rewarding. The dynamic trio on the fifth floor consisting of Eva Schildt, Britt-Marie Mårtensson and Agneta Tuszynski has helped me with numerous administrative (and other) matters. I don't know what the Department would be like without you girls, and I am quite sure that I don't want to know.

In the spring of 2003 I had the opportunity to get a first-hand experience of the paper making industry when I spent a week at the Stora Enso Hylte Mill. I would like to thank Per Malmros at Stora Enso for making this possible. I also would like to thank Olli Saarela for inviting me to visit KCL, Helsingki, in the spring of 2003.

I am grateful for the financial support provided by the EC/GROWTH CHEM project, aimed at developing advanced decision support systems for different operator-assisted tasks, e.g., process monitoring, data and event analysis, fault detection and isolation, and operations support, within the chemical process industries. The CHEM project is funded by the European Community under the Competitive and Sustainable Growth Programme (1998-2002) under contract G1RD-CT-2001-00466. The work presented in Chapter 2 was supported by the Swedish Research Council for Engineering Sciences, grant TFR 281-1998-618.

Sometimes, when my mind has been a bit too set on work, my dear friends have helped me focus on non-control related essentials in life such as whisky, computer gaming and cooking. I am very happy to have you all. My parents, Lena and Bo-Evert Åkesson have supported me and always encouraged me to do the things I have set my mind to do, from day one - you are the best! As my more practically oriented brother, Erik keeps reminding me that a good monkey wrench may be an excellent idea when I might have suggested something a bit more theoretical. It is very rewarding to be your brother.

Finally, I would like to thank my beloved wife Katarina for her love and encouragement. You have shown me what *really* matters and I am a very lucky man to have you by my side. I love you.

Johan

1

Introduction

1.1 Background and motivation

In many control systems a human operator, or user, is an essential ingredient. The role of the operator may include interaction at several levels, ranging from actually closing the control loop to mere supervision of an automatic system. As an example, consider an airline pilot. During take off and landing and the associated climb and decent, the pilot actively controls the aircraft and thereby closes the control loop. In contrast, during cruise flight, an auto-pilot controls the aircraft, and the role of the pilot is then to supervise its operation.

For a comparison, consider an embedded control system, e.g. tracking and focus control in a DVD-player. The problem is certainly technically challenging, requiring advanced hardware as well as sophisticated control techniques. Never the less, the user of a DVD-player controls the equipment at a high level, and is actually placed far outside of the control loop.

As an example from the process industry, consider a paper production plant. The plant consists of the wet end, where the pulp is put on the wire, the drying section consisting of several steam heated cylinders and finally the reeling part. Several operators attend a plant of this size and complexity. The task of the operators is highly dependent on the state of operation of the plant. In stationary operation the operators mainly supervise the production and maintain the process equipment. However, if a sheet break occurs, the operators are very

much involved in recovering a normal state of operation. These activities require team work, as several control loops must be manipulated manually and, at the same time, the paper web should be led through the machine. Other cases where there is a high degree of operator interaction is during start-ups, shut-downs and grade changes.

The presence of an operator in the control loop offers challenges for the control designer that require special attention in the control system design. There are at least four important aspects that must be considered for such control systems. The list is not exhaustive, but should capture the essence of the problem.

- Safety
  Many control systems are mission critical, in the sense that failure may damage equipment, environment or humans. Failure may also have less dramatical consequences, but never the less serious; the financial impact of production loss may be severe. The challenge for the control designer is to ensure safe operation of equipment, even in the face of erroneous behavior of the operator.

- Performance
  In order to utilize process equipment efficiently, it is important to strive to maximize the performance of the control system. In commercial applications, the monetary incitements to optimize performance may be huge, and in competitive businesses even a requirement for survival.

- Interaction
  The mechanism through which the operator communicates with the control system is also of importance. The design of the user interface should support the operator in the decision making process and enable efficient operation of the plant.

- Acceptance
  Usually, the operator is responsible for the operation of industrial equipment. In this situation, it is important that the operator trusts the control system and know its functionality and its strengths as well as its limitations. Therefore, operator acceptance is a key parameter to consider in the design of such systems.

This thesis treats examples of control problems where these issues are important. Mainly, the applications will be taken from the process industry, but examples are taken also from other fields, with the intention to illustrate the trade-off between the often conflicting objectives listed in this section.

1.2 Contributions

In this section, the contributions of the thesis are summarized.

Chapter 2: Manual control

Control of the inverted pendulum is a classic control problem, and stabilization of pendulums are part of literally all control theory educations. In Chapter 2, a variation of the classic problem is treated. The aim of the control system is to enable the operator, or pilot, to control the velocity of the pivot point of the pendulum, while keeping the pendulum in upright position. Although representing another area of application, this example illustrates the trade-offs between safety and performance present also in many process industries. The motivation of the problem is taken from avionics; the task of a control system for an unstable fighter aircraft is to stabilize the plane, while enabling the pilot to perform agile maneuvers. In the pendulum application, the control system should guarantee safe operation (stability), regardless of the reference commands given by the operator. Also, the remaining control authority should be used efficiently to enable swift maneuvers. The example in this chapter serves as an illustrative example of a control system where safety, performance and operator interaction must be regarded. The chapter is based on the publication

Åkesson, J. and K. J. Åström (2001): "Safe manual control of the Furuta pendulum." In *Proceedings 2001 IEEE International Conference on Control Applications (CCA'01)*, pp. 890–895. Mexico City, Mexico.

Chapter 3: A grade change framework

Dynamic optimization is commonly used in the process industry to improve control performance. Specifically, the performance of produc-

tion transitions, or grade changes, may be significantly improved using optimization techniques. Chapter 3 focuses on methods for increasing operator acceptance of optimization techniques. A decision support system based on a method for dynamic optimization in combination with a tool for sequential control, `JGrafchart`, is proposed. The approach is motivated by the observation that operators, when performing manual grade changes, commonly apply a sequence of actions to the plant. Example of such actions are steps and ramps. By parameterizing the optimal control trajectory as a sequence of actions, it may be implemented in `JGrafchart`, which offers a structured and hierarchical way of representing complex sequences, as well as an execution environment. The presentation is based on the publication

Åkesson, J. and K.-E. Årzén (2003): "A framework for grade changes: An optimization and sequential control approach." Preliminary accepted to ESCAPE-14.

Chapter 4: Tools for MPC

Model predictive control (MPC) is a control strategy that has won wide-spread use in the process industry. Its main advantages is that it handles, explicitly, control and state variable constraints and offers a structured method for dealing with MIMO plants. However, an MPC controller is quite complex to implement and analyze. In particular, the behavior of the associated optimization algorithm that is executed on-line complicates the analysis. Also, the computational complexity is large. When implemented, the controller offers a rather intuitive interface for tuning, but the internal functionality of the controller is often hidden for the user. Intuitive and easy to use tools that enable detailed analysis of the controller behavior is therefore desirable. In Chapter 4, tools for analysis of a standard formulation of a linear MPC controller are presented.

Chapter 5: Integral action - a disturbance observer approach

One of the most basic requirements for a process control system is to achieve error-free tracking. This is usually achieved by introducing integral action in the control loop, commonly implemented by PID controllers. Centralized control of potentially large MIMO plants, for example MPC, calls for other solutions, however. A well established

technique for ensuring offset-free tracking is the use of a disturbance observer, assuming constant load disturbances. It is straight forward to show that this method actually translates into integral action in the SISO case, and also for the case of square plants. However, if there are additional measured signals, the standard method fails, and results in stationary errors. This chapter offers a rigorous treatment of this case, and provides a method for ensuring integral action resulting in error-free tracking. The material is based on the publication

Åkesson, J. and P. Hagander (2003): "Integral action - a disturbance observer approach." In *Proceedings of European Control Conference*.

Chapter 6: Compensation of computational delay in MPC

The computational complexity of MPC controllers often prevents its application to plants where fast sampling is required. The delay introduced by on-line solution of the associated optimization program can be significant, and may even jeopardize the stability of the system. In Chapter 6 the behavior of a specific MPC implementation is studied in detail, down to the behavior of the optimization algorithm during individual samples. Two mechanisms affecting control performance are identified. Firstly, the iterative optimization algorithm minimizes a pre-specified cost function, and produces a sequence of solutions, where the last one is optimal. Secondly, there is a computational delay associated with each optimization iteration, which potentially reduces performance. Using recent results for stabilizing sub-optimal MPC, a strategy for reducing the effects of computational delay is proposed.

This chapter represents joint work with Dan Henriksson. The tools used for analysis of the MPC controller were supplied by the author of this thesis, whereas the real time simulations, and the associated tools, were supplied by Dan. The results regarding the delay compensation scheme are a result of joint work. The material is based on the following publications

Henriksson, D., A. Cervin, J. Åkesson, and K.-E. Årzén (2002a): "Feedback scheduling of model predictive controllers." In *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium*. San Jose, CA.

Henriksson, D., A. Cervin, J. Åkesson, and K.-E. Årzén (2002b): "On dynamic real-time scheduling of model predictive controllers." In *Proceedings of the 41st IEEE Conference on Decision and Control*. Las Vegas, NV.

Henriksson, D., J. Åkesson, and K.-E. Årzén (2004): "Flexible real-time implementation of model predictive control using sub-optimal solutions." Submitted to the 2004 American Control Conference, Boston, MA.

1.3 Future directions

The topics treated in this thesis offer several interesting subjects for future research. In particular, in Chapter 3 a method capturing many of the aspects of operator interaction and optimization techniques is outlined. The proposed method offers a novel technique for operator support for grade changes, in combining state of the art optimization techniques and a tool for sequential control, `JGrafchart`. However, in order to create a framework for grade changes applicable to large and complex processes, further research is necessary. For example, the issues of robustness, scalability and hierarchical structuring of grade change sequences have to be investigated in detail. For an extended discussion of future directions in this field, see Section 3.9.

2

Manual control

2.1 Introduction

Many systems are controlled by a combination of automatic and manual control, aircrafts beeing the typical example where stability augmentation systems have been used for a long time to simplify the task of the pilot. The combination of manual and automatic control is particularly crucial for unstable systems with actuator constraints. A severe problem with such systems is that the feedback loop is broken when the actuator saturates and the system may reach a state where stability cannot be recovered. The system can be driven to such a state unintentionally by manual control or by a combination of manual control and disturbances. The Swedish Fighter JAS 39 Gripen is an aircraft which is unstable in some flight conditions, in order to achieve high performance. The flight control system is mission critical because in some flight conditions the unstable mode is so fast that a pilot cannot stabilize the system. The flight control system should thus fulfill the dual task of stabilizing the aircraft without restricting the maneuverability unnecessarily. The saturation in this case is actually a rate saturation caused by the hydraulic actuators driving the control surfaces, see [Rundqwist *et al.*, 1997].

The essence of the problem can be captured in the following formulation. Consider an unstable system with actuator saturation, see Figure 2.1. Find a control strategy that stabilizes the system and pro-

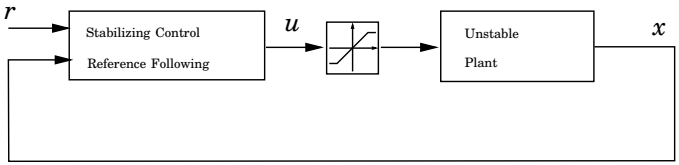

Figure 2.1 An illustration of the problem formulation dealt with in this work.

vides facilities for manual control. The strategy should be such that the pilot can manipulate the system easily but without driving the system unstable. In this paper we discuss a problem of this type, namely stabilization of a Furuta pendulum, which is a pendulum hinged on a rotating arm, see [Furuta *et al.*, 1994]. The tasks are to keep the pendulum stabilized in the upright position while permitting manual control of the arm. These tasks are conflicting because both have to be done using the same control variable, the torque on the pendulum. Since the torque is limited, manual control could drive the torque to saturation. When this happens it might not be possible to stabilize the inverted pendulum. This problem is similar to the aircraft problem. In both cases we have to stabilize an unstable system. The constraints are, however, different because there is a rate saturation in the aircraft case.

The problems associated with control system constraints are well documented in the literature. Ideas from [Åström and Brufani, 1997], [Patcher and Miller, 1998] and [Teel, 1996] will be used in the analysis and controller design. This chapter gives contributions on the translation of a theoretically motivated controller into a working implementation.

2.2 Preliminaries

In this section we will define the nonlinearity of interest, the saturation function. Further, a mathematical model for the Furuta pendulum will be presented. Finally a linear controller design will be performed. The linear controller will be needed in the nonlinear control design.

Table 2.1 Parameters of the Furuta Pendulum

Parameter	Description
r	Length of the arm
l	Distance from the pivot point to the center of mass for the pendulum
m_p	Mass of the pendulum, including weight
J_p'	Moment of inertia of the pendulum, with respect to the center of mass
J_p	Moment of inertia of the pendulum, with respect to the pivot point
J_a'	Moment of inertia of the motor and arm
J_a	Moment of inertia for the motor, arm and pendulum with θ fixed to 0

The saturation nonlinearity

The saturation function that will be used in the following is assumed to be defined by

$$\sigma_\mu(x) = \begin{cases} -\mu & x < -\mu \\ x & -\mu \le x \le \mu \\ \mu & x > \mu \end{cases}$$

Pendulum model

Consider the Furuta pendulum in Figure 2.2. The physical parameters are given in Table 2.1 and numerical values are found in Appendix A. The angle of the pendulum, θ, is defined to be zero when in upright position and positive when the pendulum is moving clockwise. The angle of the arm, φ is positive when the arm is moving in counter clockwise direction. Further, the central vertical axis is connected to a DC motor which applies a torque proportional to the control signal u. By introducing the coefficients

$$\begin{aligned} a &= J_a = J_a' + m_p r^2 & b &= J_p = J_p' + m_p l^2 \\ c &= m_p r l & d &= m_p g l \end{aligned}$$

Figure 2.2 The Furuta pendulum

the equations of motion for the Furuta pendulum may be written

$$\begin{aligned} c\ddot{\varphi}\cos\theta - b\dot{\varphi}^2\sin\theta\cos\theta + b\ddot{\theta} - d\sin\theta &= 0 \\ (a + b\sin^2\theta)\ddot{\varphi} + 2b\dot{\varphi}\dot{\theta}\sin\theta\cos\theta + c\ddot{\theta}\cos\theta - c\dot{\theta}^2\sin\theta &= \sigma_\mu(u). \end{aligned} \tag{2.1}$$

As mentioned above, a linear model of the system will be needed for controller design in the following. Introduction of the state vector

$$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} \theta & \dot{\theta} & \dot{\varphi} \end{bmatrix}^T$$

and linearization of the system (2.1) around

$$x_0 = \begin{bmatrix} 0 & 0 & \dot{\varphi}_0. \end{bmatrix}^T$$

gives

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ \alpha & 0 & 0 \\ \gamma & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \beta \\ \delta \end{bmatrix} \sigma_\mu(u) \tag{2.2}$$

where

$$\alpha = \frac{ab\dot{\varphi}_0^2 + ad}{ab - c^2} \quad \beta = \frac{-c}{ab - c^2}$$
$$\gamma = \frac{-bc\dot{\varphi}_0^2 - cd}{ab - c^2} \quad \delta = \frac{b}{ab - c^2}$$

Notice that the linearized model is parameterized with respect to $\dot{\varphi}$. The reason for this is that our objective is to perform velocity control of the Furuta pendulum arm. This means that we will force this state to have different values during a typical control sequence, and in particular, $\dot{\varphi}_0$ cannot assumed to be zero. Since this term is also very influential on the resulting linearized model, it will be taken into account in the design procedure.

A gain scheduled controller

In the following sections a nonlinear controller will be designed. In this procedure a linear controller that stabilizes the pendulum system without the saturation nonlinearity is assumed. Since all state variables are available as measurements from the real pendulum, state feedback will be used for this purpose. However, a critical feature of the linearized model is that it is highly dependent on the arm velocity $\dot{\varphi}_0$. As we can see, $\dot{\varphi}_0$ enters both α and γ as a quadratic term. In order for a controller based on the linearized state space model to successfully control the system for a wide range of $\dot{\varphi}$, this fact must be accounted for. A simple, but effective, way to do this is to use gain scheduling. A gain scheduling controller based on linear state feedback can be written

$$u = L(\dot{\varphi})(r - x) \tag{2.3}$$

where r is the reference signal to be tracked and $L(\dot{\varphi}_0) = (l_1\ l_2\ l_3)$. r is a vector; $r^T = (0\ 0\ x_3^r)$, where the two zeros indicate the desired values of the states θ and $\dot{\theta}$. The feedback gains $L(\dot{\varphi})$ were calculated using LQ design assuming constant values of $\dot{\varphi}$ ranging from 0 to 20 rad/s with a resolution of 1 rad/s. The values were stored in a table.

At runtime, linear interpolation was used to obtain an approximation of the correct feedback vector. This approximation proved to work well, and moreover, it was essential to perform successful control over the full range of arm velocities.

Friction

There were also other practical complications. There was substantial friction in the motor driving the arm and in the slip rings. The friction gave a significant limit cycle oscillation. This is illustrated in the experiment shown in Figure 2.3 where the limit cycle in the arm angle has an amplitude of about 0.45 rad. Several methods for friction compensation are discussed in [Olsson *et al.*, 1998]. A friction compensator based on the Coulomb friction model

$$F_f(\dot{\varphi}, v) = \begin{cases} F_c^+ & \dot{\varphi} > 0 \\ F_c^+ & \dot{\varphi} = 0, \quad u > F_c^+ \\ u & \dot{\varphi} = 0, \quad F_c^- < u < F_c^+ \\ F_c^- & \dot{\varphi} = 0, \quad u < F_c^- \\ F_c^- & \dot{\varphi} < 0 \end{cases} \tag{2.4}$$

was designed. An estimate of the friction force was calculated using this model and added to the control signal computed from the control law (2.3). The control signal is thus

$$v = L(\dot{\varphi})(r - x) + \hat{F}_f$$

where $\hat{F}_f$ is an estimate of the friction based on the model (2.4). In the following, it is assumed that friction compensation is used but we do not show it explicitly in the controller expressions to simplify the presentation. Figure 2.3 shows the drastic improvements obtained by friction compensation. Friction usually changes with the operating condition and friction compensation should therefore be adaptive, [Olsson *et al.*, 1998]. This was not done in the experiment, but if necessary the parameters of the friction model were re-tuned at the start of the experiments.

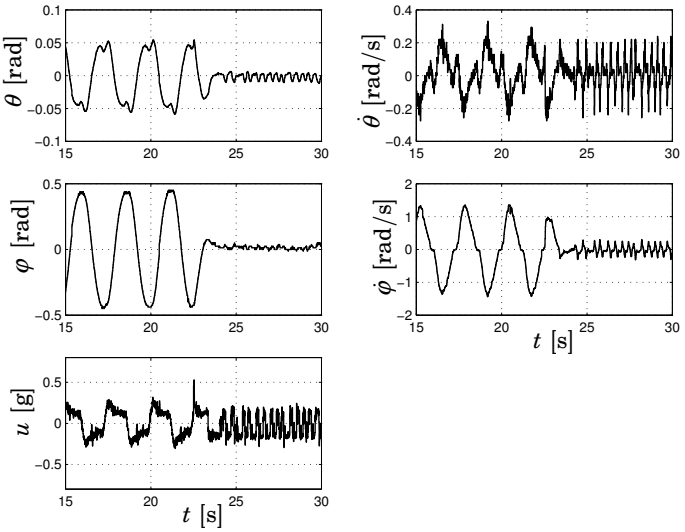

Figure 2.3 The effect of friction compensation. Notice the significant reduction of the amplitude of the limit cycle for the arm angle, ϕ. Initially, there is no friction compensation. Friction compensation is activated at $t = 23$ s.

2.3 Phase plane analysis

We will now analyze the behavior of the linearized saturation constrained system. This means that the control signal saturation is the only nonlinearity in the model, which simplifies the analysis. A phase plane analysis will be performed, which will lead us to a design of a nonlinear controller.

The prime objective of the controller is to stabilize the states $x_1 = \theta$ and $x_2 = \dot{\theta}$. Motivated by the fact that the state x_3 does not affect x_1 and x_2 we will start analyzing the subsystem

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \alpha x_1 + \beta \sigma_\mu(u). \end{aligned} \tag{2.5}$$

Now, assume a control law on the form

$$u = -l_1 x_1 - l_2 x_2.$$

The controller gains l_1 and l_2 are chosen to stabilize the system. When $|u| \leq \mu$ the system operates linearly, and is then stable. The interesting question is how the system behaves when the control saturates, i.e. $|u| > \mu$. To answer this question, let us examine the phase plane of the system.

When the control signal saturates, the system is described by

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \alpha x_1 \pm \beta\mu. \end{aligned}$$

This system has two equilibria:

$$(x_1^{eq}, x_2^{eq}) = (p, 0) = \left(\frac{\beta\mu}{\alpha}, 0\right) \tag{2.6}$$

$$(x_1^{eq}, x_2^{eq}) = (-p, 0) = \left(-\frac{\beta\mu}{\alpha}, 0\right). \tag{2.7}$$

Further we can conclude that the equilibrium points are unstable, since the system matrix has the eigenvalues $\lambda_1 = \sqrt{\alpha}$ and $\lambda_2 = -\sqrt{\alpha}$. To reveal the qualitative behavior of the system, consider the following expressions;

$$\begin{aligned} \ddot{\theta} &= \alpha\theta \pm \beta\mu \\ \dot{\theta}\ddot{\theta} &= \alpha\dot{\theta}\theta \pm \beta\mu\dot{\theta} \\ \frac{1}{2}\frac{d}{dt}\dot{\theta}^2 &= \frac{1}{2}\alpha\frac{d}{dt}\theta^2 \pm \beta\mu\frac{d}{dt}\theta \\ \frac{1}{2}\dot{\theta}^2 &= \frac{1}{2}\alpha\theta^2 \pm \beta\mu\theta + C \\ 0 &= \alpha\left(\theta \pm p\right)^2 - \dot{\theta}^2 - p\alpha + C \end{aligned}$$

where the last expression describes a hyperbolic function. The hyperbolic axes are given by $\dot{\theta} = \pm\sqrt{\alpha}(\theta \pm p)$ and corresponds to the stable and unstable eigenvectors of the system matrix given by the system (2.5).

Figure 2.4 shows the behavior of the system (2.5). In the figure, the axes of the hyperbolic trajectories can be seen, as well as the lines, $u = \mu$ (dashed) and $u = -\mu$ (dashed). We can also see that the state trajectories are stable for some initial conditions, but diverge for others. That is, the phase plane is divided into one stable, and two unstable regions.

In the plot, five different regions are marked, corresponding to different modes of operation.

- Ω_0: In this region the system operates linearly, i.e. the control law is
$$u = -l_1 x_1 - l_2 x_2.$$
and the trajectories converges to $(x_1, x_2)^T = (0, 0)$. The region is defined by the lines $u = \mu$ and $u = -\mu$.
- Ω_1^+: The control signal saturates, $u = \mu$. The region is bounded from below by the line $u = \mu$ and from above by the line corresponding to the stable eigenvector of the right equilibrium. The most significant characterization of this region is that the solutions are stable; trajectories starting in Ω_1^+, converge to origin.
- Ω_1^-: Equivalent to Ω_1^+, but bounded by the lines $u = -\mu$ and the line corresponding to the stable eigenvector of the left equilibrium.
- Ω_2^+: Also in this region the control signal saturates. The difference from Ω_1^+ and Ω_1^- is that trajectories in this region are unstable.
- Ω_2^-: Equivalent to Ω_2^+; unstable.

To conclude, we have shown that there exists a region of attraction for the constrained system. An important observation is that saturating control inputs do not necessarily cause instability, and that the stable region is significantly larger than the one implied by the saturation limits.

Let us now discuss the consequences of introducing also the state $x_3 = \dot{\varphi}$ in the model,

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \alpha x_1 + \beta \sigma_\mu(u) \\ \dot{x}_3 &= \gamma x_1 + \delta \sigma_\mu(u) \end{aligned} \tag{2.8}$$

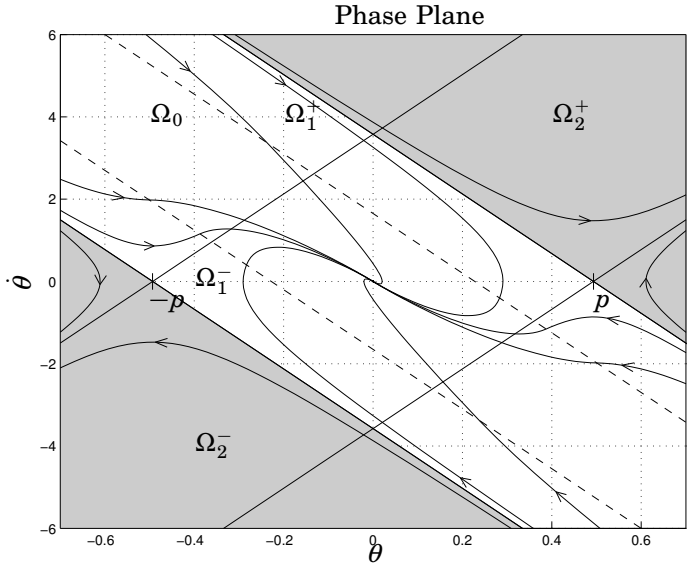

Figure 2.4 Phase plane for the constrained system (2.5). The solid curves are actual state trajectories. The dashed lines marks the saturation limits. The important conclusion to be drawn from this phase plot, is that the state space is divided into a stable region (unshaded) and two unstable regions (shaded).

The control law is now assumed to be

$$u = \sigma_\mu(-l_1 x_1 - l_2 x_2 - l_3(x_3 - x_3^r)). \tag{2.9}$$

That is, we strive to achieve $(x_1\ x_2\ x_3^r - x_3)$=(0 0 0), and in particular, we want this relation to be true in stationarity for constant x_3^r.

As stated above, the states x_1 and x_2 are not directly affected by x_3. That is, for any value of x_3, the behavior of x_1 and x_2 when the control signal saturates is determined by the hyperbolic expression (2.8). Thus, the orthogonal projection of the hyperbolic axes on the $x_1 - x_2$ plane will be as in figure 2.4. Further the planes $(\sqrt{\alpha}s \pm p,\ s,\ t)$, $s, t \in R$, constitute boundaries between important regions in the state space. Trajectories starting in the region between these planes may, but does not necessarily, converge. Trajectories starting outside

this subset however, will never converge. This observation is essential, and will be used in the following.

Now, the location of the linear region Ω_0 is critical. The expression for the saturation planes that determine Ω_0 are

$$x_2 = -\frac{1}{l_2}(\pm\mu + l_1 x_1 + u_c)$$

where

$$u^c = -l_3(x_3 - x_3^r)$$

As we can see, the orthogonal projection of the linear region onto the $x_1 - x_2$ plane will be translated in the x_2 direction depending on the value of u_c. The location of the linear region determines which of the two saturated trajectory sets that governs the behavior of the system. Moreover, if the projection of the linear region Ω_0 overlaps with any of the regions Ω_2^+ or Ω_2^- the system may enter these unstable regions from where recovery is not possible. This is equivalent to the regions Ω_1^+ and Ω_1^- disappearing. In order to prevent this, the existence of the regions Ω_1^+ and Ω_1^- should always be enforced, so that the projection of the linear region never overlaps with the unstable regions.

To conclude, if we succeed in restricting the movements of the projected linear region as describe above, the states of the system will not diverge for any combination of set point changes in x_3^r. This result assumes that the projection of the initial conditions are contained in the region of attraction in Figure 2.4, which is reasonable.

2.4 A cascaded saturations controller

With this insight, we are ready to suggest a controller for the system (2.8), that prevents destabilization the system for all possible changes in the reference value x_3^r. As above, we assume that the initial conditions are not unrecoverable. The troublesome term in the control law, and also the one that has to be restricted, is u_c. Motivated by this fact, we introduce the revised control law

$$u = \sigma_\mu(-l_1 x_1 - l_2 x_2 + \sigma_{\mu^c}(u^c)).$$

This control law offers the possibility to restrict u^c, in such way that the regions Ω_1^+ and Ω_1^- always exist in the phase plane.

This control strategy using cascaded saturations is well known and documented in the literature. Work on similar problems reported in [Teel, 1996]. An example of its applications is given in [Burg *et al.*, 1996]. The controller structure presented here also has common characteristics with the one suggested in [Åström and Brufani, 1997].

It now remains to calculate admissible values for μ^c. The condition for guaranteed stability is, as stated above, existence of the regions Ω_1^+ and Ω_1^-. The situation may be illustrated as in Figure 2.5. We will assume that the boundary line between Ω_1^+ and Ω_2^+ and the line $u = \mu$ are parallel. This can be achieved by introducing a diagonal weighting matrix $Q \geq 0$ and a control signal weight $R > 0$ in the LQ-design, and choosing the elements

$$q_{11} = \alpha q_{22} \quad q_{22} = \frac{2\rho R\sqrt{\alpha}}{\beta} - \rho^2 R, \quad -\frac{2\sqrt{\alpha}}{\beta} \leq \rho \leq 0$$

This choice of Q gives $(l_1\ l_2)=(\rho\sqrt{\alpha} \quad \rho)$ and saturation lines parallel to the stable eigenvectors. This result is for the reduced system (2.5), but can be generalized to the full system (2.8).

Further assume that we desire a safety margin for stability. In Figure 2.5 the margin is defined by εq where $q = \beta\mu/\sqrt{\alpha}$ and $\varepsilon \in [0, 1]$. This gives us an upper bound for μ^c:

$$\mu^c < -l_2(1 - \varepsilon)q - \mu$$

In Figure 2.6 and Figure 2.7 simulations of the linear constrained system controlled by the cascaded saturations controller are shown. In the simulations, the saturation limits were set to $\mu = 0.07$. As we can see, the controller successfully stabilizes the system, and provides reference following for the pendulum arm velocity. From the phase plot in Figure 2.6 it is clear how the controller prevents the system from leaving the region of attraction. Also, the system does not enter the safety region defined by the dash-dotted lines. The step responses can be seen in Figure 2.7 for a few different step sizes.

The benefit from the cascaded saturations controller is obvious, if compared to the performance of a pure linear controller. In the simulations, stability is lost for some combinations of set point changes for

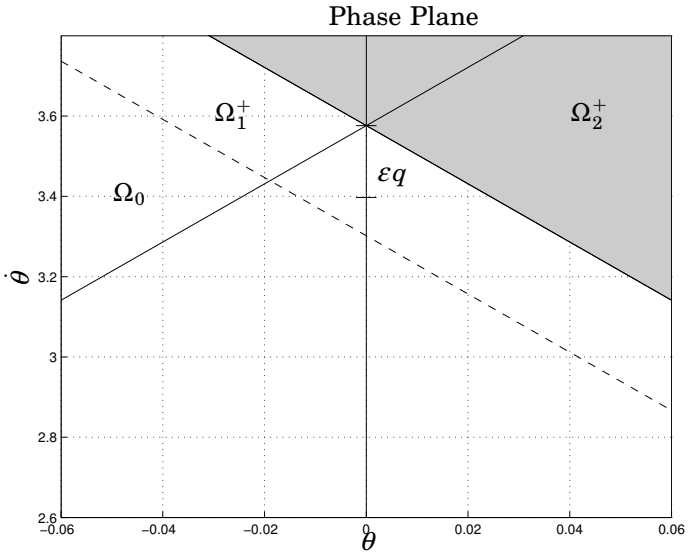

Figure 2.5 Calculation of admissible μ^c. ε may be used to obtain desired minimal size of the region Ω_1^+.

the state x_3 when the linear controller is used. When the cascaded saturations controller is engaged, the system will remain in the specified region in the state space.

This result can readily be interpreted in terms of the control task, safe manual control. Given that the state of the system initially is contained in the stable regions, as described above, there exists no reference trajectory that will drive the system out of these regions. That is, the controller guarantees that stability is preserved, independently of the applied reference.

The analysis presented in this section solves the problem for a linear plant, with constrained input. However, our analysis does not support the same claims for the constrained nonlinear plant with gain scheduled control. The region of attraction, which is the basis for the analysis will be different for the nonlinear pendulum model and it will

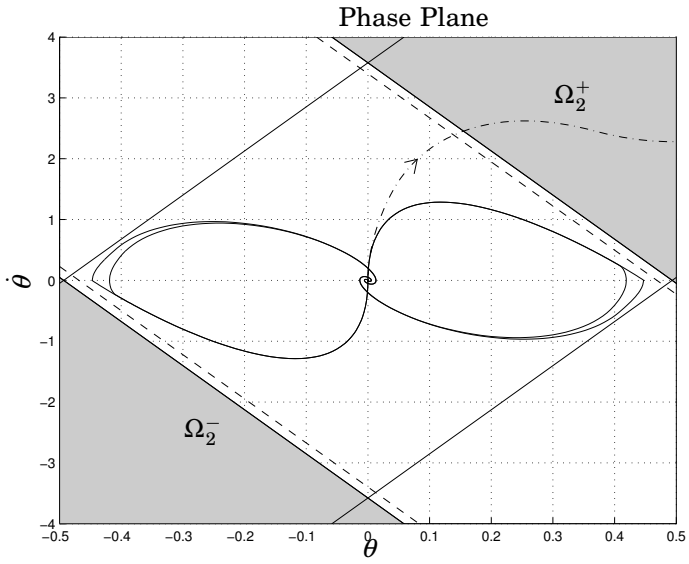

Figure 2.6 A $\theta - \dot{\theta}$ plane plot of two simulations. The solid curve shows the system behavior if the cascaded saturations controller is engaged. The dashed curve represents a simulation of the system controlled by a regular linear controller. As we can see, the linear controller fails to stabilize the system.

be necessary to investigate the full nonlinear problem.

Having said this, we shall now explore the applicability of the cascaded saturations controller on the real Furuta pendulum. Our ambition is to show that the controller designed for the linear model can, with some modifications, be made to work in practice on the real plant.

2.5 Experiments

One of the objectives in this work has been to implement a well working control strategy for a real Furuta pendulum. When applying a control strategy designed for a linear model to a real plant as in our case, the potential difference between the model and the plant may be a source

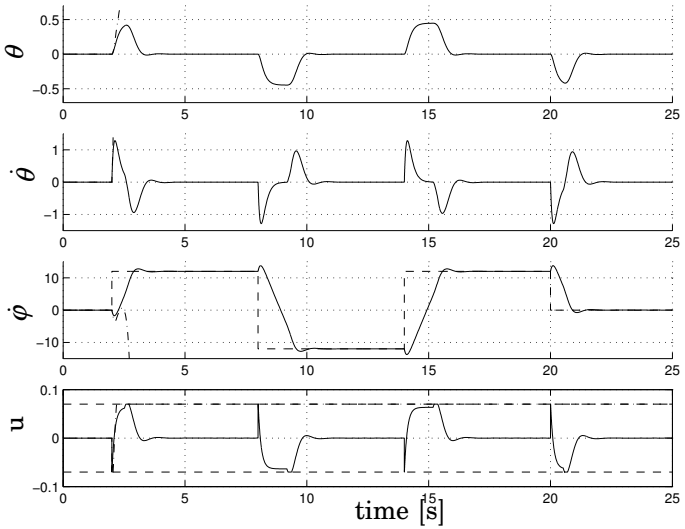

Figure 2.7 The state trajectories for the simulations. The solid curves represents the case when the cascaded saturations controller is engaged, where as the dashed-dotted curves shows the response when a pure linear controller is used. The control signal u is normalized with respect to g.

of problems. This application proved to be no exception. Various problems, for example friction, noise and the inherent nonlinearities of the pendulum made the implementation far from straight forward. The friction problem was solved by introducing friction compensation in the control loop. The compensation algorithm was based on Coulomb friction with stiction and improved the performance considerably. Our main tool in dealing with the difference between the linear and nonlinear model has been gain scheduling. This technique was used for all controller parameters that depended on the system model. The procedure was described for the state feedback vector L, see equation (2.9), above, but also the saturation limit μ^c depends on the linearized model and should thus be gain scheduled. However, these modifications were not sufficient. The value μ^c also depends on the actuator saturation limit μ, which is different from the effective saturation limits in the

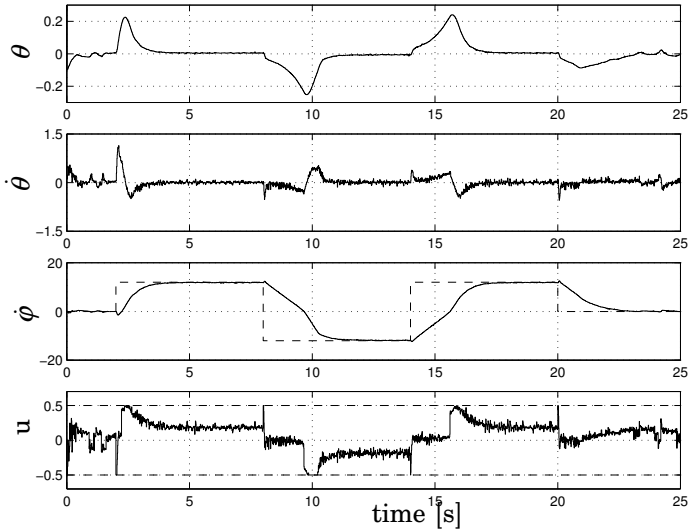

Figure 2.8 Results from experiments performed on the real Furuta pendulum. The control signal u is also here normalized with respect to g.

presence of friction and friction compensation. The difference in the control signal magnitude compared to the simulations accounts for the friction. The design parameter ε was set to 0.5 in the experiments.

With these modifications, the controller achieved the results shown in Figure 2.8. As we can see, the controller produces slower step responses compared to the simulations, but it keeps the pendulum in upright position and in stationarity achieves $x_3^r - x_3 = 0$. Under the same circumstances, a pure linear state feed back controller failed to stabilize the system. The fact that the responses are slower may be explained by the fact that the modifications made to the controller resulted in a less aggressive controller.

2.6 Results and conclusions

In this paper, we have investigated the behavior of an unstable system subject to manual control, in the presence of control signal saturation, theoretically and practically. The inverted pendulum has served as an illustrative example for the analysis and design as well as the experimental part of this work.

The analysis of the constrained linearized pendulum model resulted in a controller design, that solved the stabilization problem. The nonlinear case, however, is not clear and would be a natural extension of this work. The cascaded saturations controller was also implemented on a real Furuta pendulum. The controller succeeded, after some modifications striving to make it less aggressive, in stabilizing also the real pendulum.

2.7 Relations to other work

In the course of this work, we have also investigated other approaches to solve the problem of constrained control. In particular, we have investigated the applicability of the methods presented in [Gilbert and Tan, 1991] and [Gilbert *et al.*, 1994] which are based on very elegant analysis using admissible sets and the reference governor. There were some difficulties in applying these ideas to the Furuta pendulum, although they performed well in simulations. The main reason was due to the difficulty in computing the admissible sets accurately for the real pendulum which has friction and measurement noise.

2.8 References

Burg, T., D. Dawson, C. Rahn, and W. Rhodes (1996): "Nonlinear control of an overhead crane via the saturating control approach." In *Proc. of the 1996 IEEE International Conference on Robotics and Automation*, pp. 3155–3160. Minneapolis, Minnesota.

Furuta, K., M. Yamakita, S. Kobayashi, and M. Nishimura (1994): "A

new inverted pendulum apparatus for eduction." In *IFAC Symposium on Advances in Control Education, Boston, MA*, pp. 191–196.

Gilbert, E., I. Kolmanovsky, and K. Tan (1994): "Nonlinear control of discrete-time linear systems with state and control constraints: A reference govenor with global convergence properties." In *Proc. 33rd Conference on Decision and Control*, pp. 144–149. Lake Buena Vista, Florida.

Gilbert, E. and K. Tan (1991): "Linear systems with state and control constraints: The theory and application of maximal output admissible sets." *IEEE Trans. on Automatic Control*, **36:9**, pp. 1008–1020.

Olsson, H., K. J. Åström, C. C. de Wit, M. Gäfvert, and P. Lischinsky (1998): "Friction models and friction compensation." *European Journal of Control*.

Patcher, M. and R. Miller (1998): "Manual flight control with saturating actuators." *IEEE Control Systems*, February, pp. pp. 10–19.

Rundqwist, L., K. Stål-Gunnarsson, and J. Enhagen (1997): "Rate limiters with phase compensation in JAS 39 Gripen." In *Proc. European Control Conference*. Saab Military Aircraft, Linköping, Sweden.

Åström, K. and S. Brufani (1997): "Manual control of an unstable system with a saturating actuator." In *Proc. 36th Conference on Decision and Control*, pp. 964–965. San Diego, California.

Teel, A. (1996): "A nonlinear small gain theorem for the analysis of control systems with saturation." *IEEE Trans. on Automatic Control*, **41:9**, pp. 1256–1270.

3

A framework for grade changes

3.1 Introduction

Typically, chemical processes are designed and optimized for steady state operation. Also, processes are often controlled by local and independent control loops implemented in a Digital Control System (DCS). A DCS may achieve satisfactory control in stationary operation but often it offers little help to the operator for controlling transient phenomena like grade changes, start-ups and shut-downs, [Ihalainen and Ritala, 1996]. In these cases, the control performance relies on the expertise of the operator. In many cases this may be satisfactory, whereas for critical or frequent grade changes efficient operator support would be desirable.

As a typical example of a process where grade changes are important, consider a paper machine. As discussed in Section 1.1, during normal operation, the control system of a paper machine operates in stationary mode, striving to keep critical process variables such as basis weight and moisture level at the specified target values, which define a certain paper grade. However, a control system optimized for stationary operation is not adequate for grade changes, as it would produce to slow transitions and unnecessarily large production losses. Efficient support for grade changes is of major importance to achieve

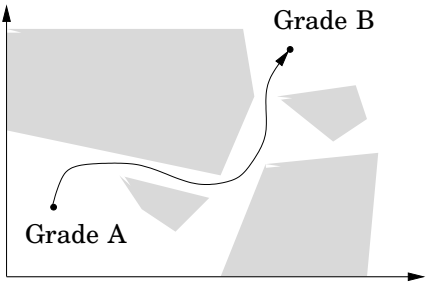

Figure 3.1 A grade change transfers a process from one operating condition to another. The shaded region represents constraints that must be respected.

superior production performance in this type of plants, see [Forsman, 2002] for a recent case study.

There has also been a shift towards diversification and tailored products in chemical process industry. This trend has made grade changes a more frequent mode of operation in many industries, which also motivates development of improved operator support for this type of control mode, [McQuillin and Huizinga, 1994]. In addition, there are usually safety regulations that must be respected, e.g. process variables that may not exceed specified values.

The aim of a grade change may be complex. A grade change can be seen as a state transition, where the task is to transfer, safe and as fast as possible, the state of a process from a starting point to a terminal point in the state space of the process. In addition, there are usually constraints on the control variables and process variables that must be respected. Typically, actuators like pumps and valves have a limited region of operation, and critical process parameters should be kept within certain safety intervals. The problem is illustrated in Figure 3.1.

The framework outlined in this chapter has been developed without a specific target application in mind. Rather, the aim has been to formulate a widely applicable method suitable for process industries in various fields, such as pulp and paper, petroleum and food industries. An important assumption, however, is that the target process is continuous, as opposed to a batch process. Application of the proposed

method to batch processes would require somewhat modified assumptions, and would be an interesting extension.

Industrial application of optimal control profiles requires operator acceptance. In many cases, the operating personnel are responsible for the operation and safety of the equipment. Operator confidence in any control method to be used is therefore crucial. This observation has important consequences for design of operator support systems. Specifically, it is important to recognize that the mechanism for presenting and executing grade changes is a key issue. To the authors knowledge, little work seems to be done in this area compared to the efforts to develop dynamic optimization schemes.

In this chapter we consider dynamic optimization in combination with tools for sequential control as building blocks for an operator support system for grade changes. The development in the field of dynamic optimization has been significant during the last decade. *On-line* applications like MPC have won widespread use in industry, see for example [Qin and Badgwell, 2003]. Another branch of the theory is concerned with *off-line* optimization applied to large scale systems, see for example [Biegler *et al.*, 2002] for an overview.

We propose the use of Grafcet, see [David and Alla, 1992], to provide a more operator friendly representation of grade change sequences. In particular, a dynamic optimization problem is formulated and solved off-line so that the solution is given by a sequence of actions often used by operators, e.g. steps and ramps.

3.2 Relations to previous research

There are several possible approaches to the grade change problem. A very active field of current research is Nonlinear Model Predictive Control (NMPC). This control strategy explores a nonlinear dynamic process model, and solves, at each sampling instant, a dynamic optimization problem. Several successful applications in the process industry have been reported. A few illustrative examples can be found in [Bemporad and Morari, 1999; Magni *et al.*, 1999], where NMPC applications based on nonlinear and hybrid dynamic models are described.

The benefits of using NMPC in a grade change context are obvious. Firstly, the control strategy performs well for grade changes where

the dynamics of the process changes with the operating conditions (given that a good nonlinear model is available). Secondly, the NMPC controller may be employed also during stationary operation, and possibly improve control performance further. The application of NMPC to complex plants is not straight forward, however. On-line solution of nonlinear optimization programs is computationally demanding, which requires long sampling intervals. Also, the optimization algorithm may fail due to infeasibilities or local minima, although techniques for reducing such problems have been developed. From an operators point of view, an NMPC controller represents a complex control structure. Without profound knowledge of the internal principle of the controller it may be difficult to understand its behavior, which may be, although optimal with respect to a given criteria, foreign to normal operator procedures. As argued above, operator acceptance is critical in process industries where the operator is responsible for the safety and functionality of the plant. In this respect, there remains open research problems in the field of NMPC.

Some industrial control systems also offers support for improved grade change performance. In [Forsman, 2002], implementation of a control system from ABB Process Industries AB for a paper machine is described. The control system software also offers a grade change function; "Fast grade change", which may be used to improve performance. In [Murphy and Chen, 1999] and [Murphy and Chen, 2000] written by ABB researchers, a technique for performing grade changes aimed at paper machines is described. The method described there assumes linear dynamic first order process models with time delay, which relate the important controlled and manipulated process variables. The strategy explores operator expertise in offering highly machine dependent design parameters, notably transition speed, for the operator to choose. In [Murphy and Chen, 1999] a real application example is reported, showing significantly improved grade change performance compared to a transition executed manually by operators.

In [Ihalainen and Ritala, 1996] an approach to grade changes based on dynamic optimization for a paper making process is reported. The idea presented in that paper is based on the use of "valley functions", and provides an interesting discussion about choice of optimization criterion and consequences thereof for the dynamic optimization algorithm.

The method for decision support for grade changes that will be described in the following uses model based optimization as a key ingredient. The quality and availability of a dynamic model describing the plant of interest will inevitably have a major impact on the performance of the resulting grade change sequence. Modeling of paper machines, however, is currently an active field of research, and significant efforts is done in the field, see for example [Askaner, 2003; Balderud *et al.*, 2001].

3.3 Assumptions

In many process industries, the controlled process is accessed through a DCS, where the DCS implements local controllers, such as PID controllers, tuned for stationary operation of the plant. Normally, the DCS takes care of the stationary control of the plant, but it is usually possible for operators to put loops in manual mode and set control variables directly. For each grade, or stationary point, we assume that there is a corresponding set of reference values for the local controllers. Reference values for *all* loops may not specified, they can also be generated internally in the DCS, for example in the case of cascaded control loops. Typically, during production transitions, critical control loops are put in manual mode by the operators, and the control variables, or possibly reference values for low level control loops, are then manipulated directly during the transition. A more elaborate description of operating procedures for grade changes is given in [Sundarraman and Srinivasan, 2000].

Using the standard terminology of control theory, this transition control strategy corresponds to feedforward control, whereas the control strategy in stationary operation corresponds to feedback control. The combination of feedback and feedforward is a well established control strategy [Åström, 2002]. For a schematic view, see Figure 3.3.

It is natural to regard the transition problem as an open loop optimal control problem for the plant. That is, we would like to find the optimal solution, according to some criteria, that transfers the system from one grade to another. The solution will include a control variable trajectory, u, as well as the expected response of the plant, y. A natural way to use the optimal trajectories for process transitions is to

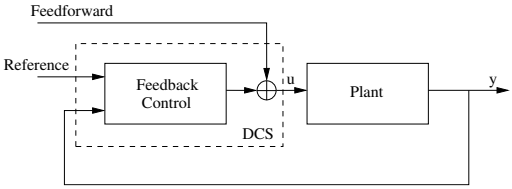

Figure 3.2 A structural figure of the proposed frame work

use u as a feedforward signal for the DCS, and let y be the reference trajectory for the local control loops. This way, feedforward is used to achieve performance, whereas feedback is used to obtain robustness.

We will assume that the optimization of the transition sequence is performed *off-line*. Potentially, the optimization may be time consuming, and there may also be a variation in calculation time between different grade changes. With this approach it is also possible to maintain a data base that stores transition sequences for frequently performed grade changes.

In Figure 3.3 the structure of the proposed grade change framework can be seen. During stationary operation, the reference values for the DCS are fixed. When a grade change is initiated, the transition commands are instead supplied by a grade change controller, supervised by the operator. When the process has reached the new grade, the controller returns to the stationary control mode. The proposed process transition scheme utilizes the DCS for feedback and feedforward control, as opposed to using primarily manual control during transitions.

3.4 Sequential control and JGrafchart

Grade changes are sequential by nature. In order to ensure operator acceptance of a grade sequence generation tool it is important to use formats and notations that are well accepted. A common way of expressing sequential actions in automation systems is to use Grafcet (Sequential Function Charts; IEC61131-3, [Lewis, 1996]). In Grafcet a sequence is represented by a sequence of steps interconnected by tran-

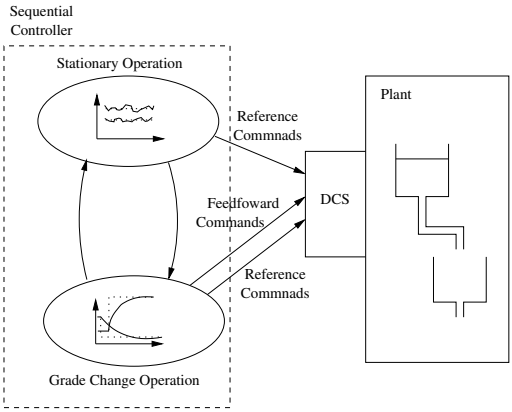

Figure 3.3 A schematic figure of the proposed grade change framework. A sequential controller is used to apply reference commands to the DCS.

sitions. The steps contain actions that are executed when the step is activated, when it is deactivated, or periodically while the step is active. The transitions contain boolean conditions. When a step is active, the conditions of the transitions following the step are evaluated, and when one of them becomes fulfilled, the preceding steps are deactivated and the succeeding steps are activated. A grade change sequence could be represented by steps representing step changes or ramp changes in reference signals or feed-forward signals. Using Grafcet it is also straight forward to express parallelism and synchronization.

Grafcet is commonly used as a language to express sequential control structures in DCS and PLC systems. However, the execution of the Grafcet is often implemented at a low level in the control system, without the possibility of operator interaction. In particular, the graphical Grafcet representation is normally not shown to the operator. The graphical feature of Grafcet is usually used only for programming and reconfiguration purposes, possibly performed by an expert rather than an operator. A key benefit of using Grafcet for sequential control is then left unexplored. By enabling the operator to follow the Grafcet execution graphically, operator acceptance and understanding for the control system as well as the process could be increased.

Using the graphical representation of Grafcet, a sequence could be executed automatically, semi-automatically, or manually. In automatic mode the sequence is executed without operator intervention, whereas in semi-automatic and manual mode the operator is required to either acknowledge the actions before they are executed or execute them fully manually, i.e., in the latter case the grade sequence could be viewed as a suggestion.

`JGrafchart` (see Figure 3.4) is the name of a Java-based version of Grafchart, see [Johnsson, 1999], an object-oriented extension to Grafcet/Sequential Function Charts that also includes part of the functionality of Statecharts, currently under development [Årzén *et al.*, 2002]. The language elements are placed on a workspace and connected together graphically. After compilation the function charts are executed by the runtime system within the editor. `JGrafchart` includes a number of language features that support abstraction and re-use. It is also possible to implement graphical user interfaces and user dialog management. Hence, using `JGrafchart` it is possible to present the generated grade change sequences to the operator in a familiar format and to also execute the grade changes.

In the following, a few elementary operations representing common operator actions expressed as Grafcet elements will be introduced. Such operations include application of steps and ramps to the plant. By exploring the hierarchical features of Grafchart, very complex grade change procedures may be implemented in a structured manner. This gives us a suitable platform for a decision support system for grade changes.

3.5 Dynamic optimization

During the last five decades, the theory of dynamic optimization has received much attention. In 1957, Bellman formulated the celebrated Principle of Optimality, and showed that Dynamic Programming was applicable to a broad range of applications, [Bellman, 1957]. Following this work, the dynamic programming technique has been applied to various fields, notably inventory control, economics, statistics, engineering etc. For a modern description, see [Bertsekas, 2000a],[Bertsekas, 2000b]. However, although a very elegant theory,

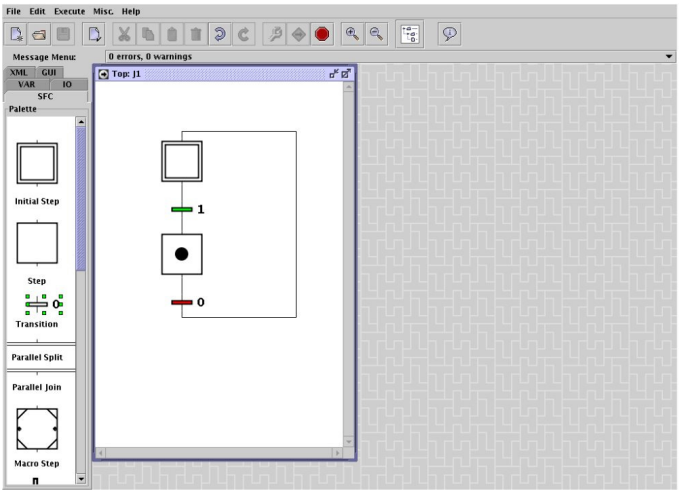

Figure 3.4 The screen layout of `JGrafchart`.

dynamic programming has proven difficult to apply to large optimal control problems. In particular, the presence of nonlinear dynamics and state or control variable constraints may lead to computationally intractable problems. However, recent initiatives have shown that it is possible to calculate approximate (with pre-specified error bounds) solutions of the dynamic programming problem, increasing the applicability of the technique to complex systems, see [Lincoln, 2003].

Another branch of the theory is concerned with open loop optimal control, as opposed to dynamic programming which is a closed loop strategy. The former branch is closely related with the calculus of variations. Using this formulation, the solution of the optimal control problem includes solution of a two-point boundary-value problem. For a detailed treatment, see [Bryson and Ho, 1975]. There are several variations of this theme. Also, numerical algorithms for solving trajectory optimization problems are available, for example RIOTS [Schwartz, 1999].

The applicability of a particular optimization method is highly model dependent. For example, in the case of linear dynamics,

quadratic cost function and no constraints, the solution is a linear state feedback control law which may be computed from the solution of the associated Ricatti equation. For nonlinear system models or in the presence of constraints, the situation is more involved. The framework presented in this chapter is intended for large and complex plants, and accordingly, the dynamic optimization method of choice should be suited for this type of problems.

Commonly, chemical processes are modeled using Differential Algebraic Equations (DAE) obtained from first-principle equations or identification experiments. It is therefore reasonable to formulate the optimization problem so that this, quite general, model structure can be used. A common formulation of an optimal control problem of this type is the following

$$\begin{aligned} &\min P(x(t_f), u(t_f), t_f) \\ &\text{subject to} \\ \bar{f}(\dot{x}(t), x(t), u(t)) &= 0 \quad \text{(DAE dynamics)} \\ x(t_0) &= x_0 \quad \text{(initial conditions)} \\ g(x(t_f), u(t_f)) &= 0 \quad \text{(terminal constraint)} \\ r_e(\dot{x}(t), x(t), u(t) &= 0 \quad \text{(equality path constraints)} \\ r_i(\dot{x}(t), x(t), u(t) &\leq 0 \quad \text{(inequality path constraints)} \end{aligned} \tag{3.1}$$

where P is a scalar objective function, $x(t) \in R^n$ is the state of the system and $u(t) \in R^m$ is the control signal. This formulation covers many common optimal control formulations, e.g. minimum time problems. The apparent limitation that state and control variables are only penalized at the final time is not severe. For example, an integral performance measure is easily included by adding an additional state governed by the dynamics

$$\dot{x}_i(t) = L(x(t), u(t))$$

where L is a suitable function.

In the field of process control, there are two major groups of algorithms for solving the dynamic optimization problem (3.1) in an off-line

context: simultaneous and sequential. Using simultaneous methods, both the state space and control variable space is discretized, yielding a finite dimensional non linear programming (NLP) problem. The number of optimization variables is generally very large, and specialized NLP algorithms are required, see [Biegler *et al.*, 2002]. Also, using a simultaneous method, the dynamics of the DAE system may be satisfied only at the optimal point. An advantage of this method is that state and control variable constraints are straight forward to include.

Sequential methods, or control parameterization methods, on the other hand, use standard DAE solvers to integrate the system equations and evaluate the cost function. Only the control variable space is discretized, yielding a significantly lower number of optimization variables in the NLP problem. A main advantage of sequential methods is that the solution at each NLP iteration is feasible with respect to the dynamics of the system, which enable premature termination of the algorithm. On the downside it should be mentioned that integration of the DAE equations, which is required at each iteration, may be computationally demanding. Also, path constraints requires special attention. For a detailed treatment of sequential methods, see [Vassiliadis, 1993].

Both methods use polynomials to parameterize the control profiles. The control interval $[t_0\ t_f]$ is usually divided into a fixed number of segments (or elements), where a polynomial of predefined degree is used to represent the control signal in each segment. Steps and ramps correspond to zero and first order polynomials, and either method could in principle be used to generate grade change sequences suitable for Grafcet representation.

A control parameterization method

To demonstrate the idea in this chapter, a simplified version of the sequential method described in [Vassiliadis, 1993] has been implemented, mainly because of its conceptual simplicity and the fact that it does not require a specialized NLP solver. The original method is quite general, and is applicable to multistage DAE systems with general junction conditions and inequality as well as equality path constraints on state and control variables. Junction conditions are useful for example for describing system behavior at the boundary between two stages in a chemical reaction. In this formulation, implementation of

the method requires a DAE integrator and an algorithm for solving nonlinear programming (NLP) problems. In this section a simplified version of this scheme, still capturing the important features, is described. We assume that the system dynamics is the same over the optimization horizon, that the dynamics is given by an ODE and that path constraints are only enforced for the control variables. These restrictions simplifies presentation as well as implementation, but they could easily be removed to obtain a more generally applicable grade change frame work. The notation introduced in the following closely follows the presentation in [Vassiliadis, 1993].

We consider the following simplified problem

$$\begin{aligned} &\min P(x(t_f), u(t_f), t_f) \\ &\text{subject to} \\ &\dot{x} = f(x(t), u(t)), \quad \text{(ODE dynamics)} \\ &x(t_0) = x_0 \quad \text{(initial conditions)} \\ &g(x(t_f), u(t_f)) = 0 \quad \text{(terminal constraint)} \\ &u^L \leq u(t) \leq u^H \quad \text{(inequality path constraints)} \end{aligned} \tag{3.2}$$

where P, $x(t)$ and $u(t)$ have the same interpretation as previously. The main differences compared to problem (3.1) is that ODE instead of DAE dynamics is assumed and that only path constraints on the control variables are included.

Control parameterization Control parameterization is a key issue, both for the optimization method, and for the Grafcet representation of the grade change sequence. In particular, parameterization of the control variables significantly affects the convergence of the NLP algorithm. For example, a very large number of parameters makes the search problem harder, and the risk of obtaining sub-optimal solutions due to local minima increases.

Commonly, Lagrange polynomials are used to parameterize the control signal. Using polynomials, the degree of the polynomials is a key design parameter. For our purposes, zero and first order polynomials correspond to steps and ramps, which makes this approach very well suited for the application at hand. In Figure 3.5, a typical example of

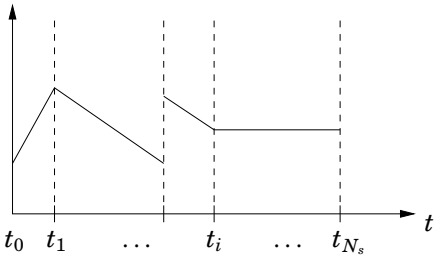

Figure 3.5 A control variable trajectory parameterized by first order polynomials.

a control trajectory parameterized by first order polynomials is shown. The elements of the control horizon are defined by the switching instants t_i, $i = 1 \ldots N_s$, where N_s is the number of segments. For each segment i in the horizon, the control variable u_j is defined by

$$\begin{aligned} u_j^{[i]} &= \sum_{k=1}^{M_j} u_{ijk} \phi_k^{(M_j)}(\tau^{[i]}) \\ t &\in [t_{i-1}\ t_i] \end{aligned} \tag{3.3}$$

where u_{ijk} is the *k:th* polynomial weighting coefficient over the *i:th* segment of the *j:th* control variable,

$$\tau^{[i]} = \frac{t - t_{i-1}}{t_i - t_{i-1}} \in [0\ 1]$$

is the normalized time over the *i:th* segment and ϕ are Lagrange polynomials. The order of the Lagrange polynomials (which equals $M_j - 1$) may be chosen arbitrarily. However, we will only use zero and first order Lagrange polynomials, which are given by

$$\begin{cases} \phi_1^{(1)}(\tau) = 1 \\ \phi_1^{(2)}(\tau) = \dfrac{\tau - \tau_2}{\tau_1 - \tau_2} \\ \phi_2^{(2)}(\tau) = \dfrac{\tau - \tau_1}{\tau_2 - \tau_1} \end{cases} \tag{3.4}$$

The parameters τ_1 and τ_2 may be chosen arbitrarily, but if $\tau_1 = 0$ and $\tau_{M_j} = 1$, then control variable inequality constraints for constant and piecewise linear polynomials are easily enforced by

$$u^L \leq u_{i.1} \leq u^H \quad M_j = 1,\ i = 1 \ldots N_s$$
$$u^L \leq u_{i.k} \leq u^H \quad M_j = 2,\ i = 1 \ldots N_s,\ k = 1, 2$$

It is also possible to enforce other types of constraints, for example, continuity of the control trajectory is obtained by including

$$u_{i-i,j,M_j} = u_{ij0}$$

as a constraint in the optimization procedure.

This control variable parameterization gives the following set of optimization parameters

$$p = \{u_{ijk},\ h_i j\}$$

where u_{ijk} is defined as above and h_{ij} is the length of the *i:th* segment of the *j:th* control variable. The lengths of the segments are usually used as optimization variables instead of the switching times. The choice of Lagrange polynomial order and the number of segments are key design parameters that must be chosen carefully for each application. Also, this choice has significant impact on the performance of the optimization algorithm. For example, a large number of segments may lead to an over-parameterized problem yielding slow convergence in the NLP algorithm. In order to reduce the number of optimization parameters, it is possible to let the switching times be fixed, in which case only the u_{ijk}*:s* are optimization variables. Also, in the case of multiple control variables, the switching times may be defined for each control variable, common for all variables or a combination thereof. In particular, the number of segments need not be the same for all control variable trajectories.

The infinite dimensional dynamic optimization problem has now been translated to a finite dimensional approximation of the original problem. This approximation procedure enables the use of a standard NLP algorithm for optimizing the criteria

$$P(x(t_f), u(t_f), t_f) = P(p).$$

Gradient information Most NLP algorithms require information about the gradient of the optimization criteria with respect to the optimization variables, and sometimes also the second derivative i.e. the Hessian. For dynamic optimization, this problem is considered in [Rosen and Luus, 1991], where three methods for gradient evaluation are considered. The first, and most intuitive method is to estimate the gradient by calculating finite differences. This is also the easiest method to implement, but it may require careful scaling of perturbations, which yield a less robust method. A second method explores the adjoint system associated with the Hamiltonian function, that appears in the solution of the optimal control problem based on Pontryagin's maximum principle, [Bryson and Ho, 1975]. Thirdly, the state trajectory sensitivity equations may be used. [Rosen and Luus, 1991] offers a thorough discussion about which method should be used, concluding that the use of the sensitivity equations is computationally less demanding than using the adjoint system, and more accurate than using finite differences.

The sensitivity equations for a given parameter p can be written

$$\frac{dx_p(t)}{dt} = \frac{\partial f}{\partial x} x_p(t) + \frac{\partial f}{\partial u} \frac{\partial u(t)}{\partial p}. \tag{3.5}$$

This ODE describing the evolution of $x_p(t) = \frac{\partial x(t)}{\partial p}$ over the optimization horizon can be integrated, using the additional relations

$$\begin{aligned} x_p(t_0) &= 0 \\ x_p(t_i^+) &= x_p(t_i^-) + \left(f(x(t_i^-), u(t_i^-)) - f(x(t_i^+), u(t_i^+)) \right) \frac{\partial t_i}{\partial p} \end{aligned} \tag{3.6}$$

which represent initial conditions and boundary conditions at the switching times. The second equation (3.6) accounts for the fact that the *full* derivative of of the state trajectories with respect to the parameters must be evaluated. Derivatives of the control variable trajectories $u(t)$ with respect to the optimization parameters p, $\frac{\partial u(t)}{\partial p}$, can be found in [Vassiliadis, 1993].

The gradient of the cost function with respect to the optimization

variables may now be evaluated from the expression

$$\frac{\partial P}{\partial p}(t_f) = \frac{\partial P}{\partial x}\left(x_p(t_f) + f(x(t_f), u(t_f))\frac{\partial t_f}{\partial p}\right) + \frac{\partial P}{\partial u}\frac{\partial u}{\partial p}(t_f) + \frac{\partial P}{\partial t_f}\frac{\partial t_f}{\partial p}. \tag{3.7}$$

The gradient information can now readily be used by the NLP algorithm to determine a search direction. Obviously this method for calculating gradients involves numerical solution of several differential equations, which is potentially computationally demanding. However, it is possible to solve simultaneously, the ODE:s representing the system dynamics (which is necessary in order to evaluate the cost function) and the sensitivity equations for increased efficiency.

It is also possible to evaluate second derivatives using similar techniques. In [Vassiliadis *et al.*, 1999] it is shown the the performance of the NLP algorithm in terms of faster convergence can be improved using this method. Evaluation of second derivatives using sensitivities was not included in the implementation described here, however, but would be a reasonable extension.

Implementation

The algorithm has been implemented in Matlab using `ode45` to solve differential equations and Optimization Toolbox to solve the NLP problem. The Optimization Toolbox function `fmincon`, which implements a Sequential Quadratic Programming (SQP) algorithm (see for example [Fletcher, 1987]) was used to solve the NLP problem. The algorithm requests, at each iteration, gradient information, which is calculated by integration of the sensitivity equations. Cost function evaluations are also initialized by the optimization algorithm.

3.6 Grafchart representation

The mapping between the solution of the optimal control problem and a corresponding Grafchart sequence is a key part in the proposed method. Using the proposed method, generation of grade change sequences should be done off-line, where the grade change Grafcets are

generated automatically from the solution of the optimization problem. For a particular plant, a library of frequently occurring grade changes may be created and used by the operating personnel. Used in this context, the Grafcet representation offers two important benefits.

Firstly, it serves as a complement to trend plots and offers a structured and hierarchical view of complex transition procedures. For a complex transition, the number of control variables may be very large, and structuring the transition sequence into different levels of detail can be most helpful for the operator. For example, there may be subsystems of the process that requires detailed but not critical operation, and accordingly, the corresponding Grafchart sub-sequence could be encapsulated and accessed only if needed by the operator. The hierarchical structuring of the grade change information assists the operator in maintaining an overview of the procedure.

Secondly, when implemented in `JGrafchart`, or an equivalent environment, it enables the operators to execute and supervise a grade change at a desired level of detail. As pointed out previously, a sequence can be executed with different levels of operator involvement. Simple tasks can be executed automatically, whereas critical sequences may be executed when acknowledged by the operator. For a detailed description of `JGrafchart`, see [Olsson, 2002].

Three types of Grafchart elements have been defined in order to express Grafchart sequences, namely steps, ramps and function tables, see Figure 3.6.

- The step element
  One of the most basic operations performed by operators is to apply a step to a control variable or a reference value. When the step element is activated, an instantaneous change in the associated control variable value is performed, which is implemented as an action that is executed when the step becomes activated. The transition following the step element specifies the length of the time interval that should elapse before proceeding to the next element in the sequence. This is expressed as a condition evaluating to `true` when the step element has been active for the specified amount of time.

- The ramp element
  The ramp element is defined similarly. When the ramp element is

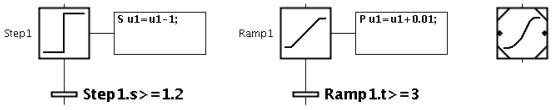

Figure 3.6 Grafchart elements for representation of grade change sequences. The variable $u1$ is assumed to be the name of a control variable.

activated, an action is executed at each cycle updating the associated control variable. When the transition condition specifying the time for which the ramp element is active evaluates to `true`, the execution proceeds.

- The function table element
  Potentially, more complex functions have to be used. For example, it was suggested above that the expected response of the plant should be used as reference values for local control loops. The predicted response of a complex plant is generally not possible to parameterize using only steps and ramps, which calls for a more sophisticated element. The function table element offers the possibility of defining arbitrary functions.

Using these three basic elements as building blocks, very complex grade change procedures may be described. In addition, other features of Grafchart, such as synchronization elements may be used to express parallel and synchronized tasks. Another useful element of the Grafchart language is the macro step, which can be used to encapsulated sub-sequences and thereby achieve hierarchical structuring of grade change procedures.

Figure 3.7 shows an example of a typical sequence encapsulated in a macro step. The time function representing the control variable trajectory is shown in the left plot and the corresponding Grafcet in the right figure. In Figure 3.8 a more complex sequence involving two control variables, $u1$ and $u2$, as well as synchronization elements is shown.

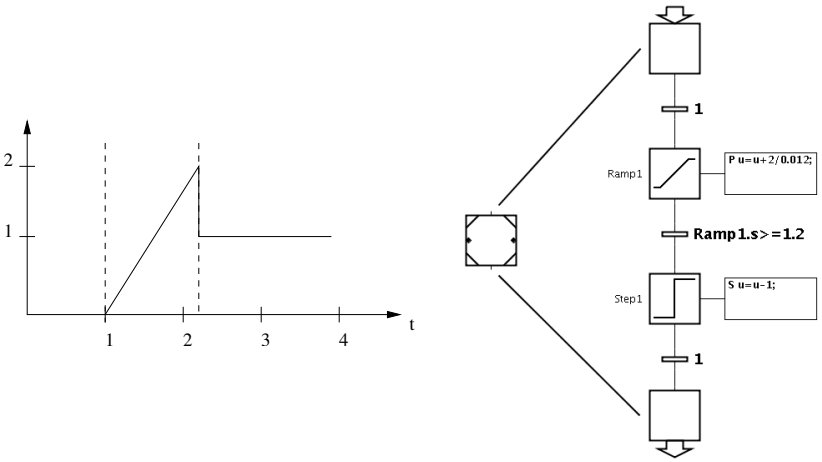

Figure 3.7 A sequence of a ramp and a step represented as a time function and a Grafcet sequence respectively. The update interval of the runtime system is assumed to be 10 ms.

3.7 An example

The quadruple-tank laboratory process, see Figure 3.9, has been used to demonstrate the proposed grade change framework. The model presented here is derived in [Johansson, 1997]. The process consists of four tanks, organized in pairs (left and right), where water from the two upper tanks flows into the two lower tanks. A pump is used to pour water into the upper left tank and the lower right tank. A valve width fixed position is used to allocate pump capacity to the upper and lower tank respectively. A second pump is used to pour water into the upper right tank and lower left tank. The control variables are the pump voltages. Let the states of the system be defined by the water levels of the tanks (expressed in cm) x_1, x_2, x_3 and x_4 respectively. The maximum level of each tank is 20 cm. The dynamics of the system is

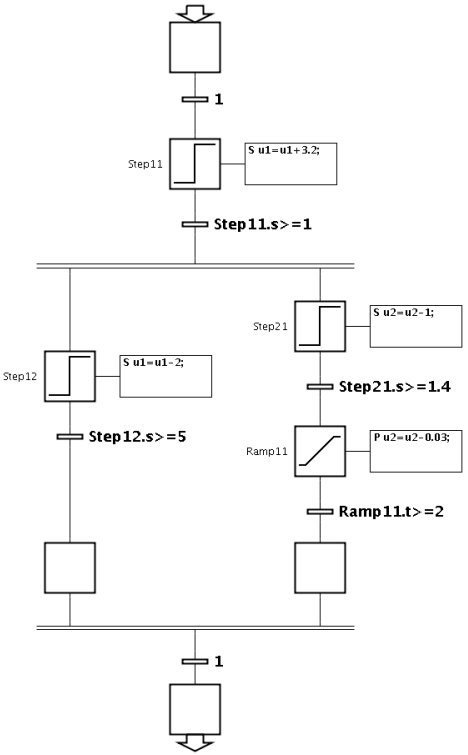

Figure 3.8 A more complex Grafchart representing a grade change sequence.

given by

$$
\begin{aligned}
\dot{x}_1 &= -\frac{a_1}{A_2}\sqrt{2gx_1} + \frac{a_3}{A_1}\sqrt{2gx_3} + \frac{\gamma_1 k_1}{A_1}u_1 \\
\dot{x}_2 &= -\frac{a_2}{A_2}\sqrt{2gx_2} + \frac{a_4}{A_2}\sqrt{2gx_4} + \frac{\gamma_2 k_2}{A_2}u_2 \\
\dot{x}_3 &= -\frac{a_3}{A_3}\sqrt{2gx_3} + \frac{(1-\gamma_2)k_2}{A_3}u_2 \\
\dot{x}_4 &= -\frac{a_4}{A_4}\sqrt{2gx_4} + \frac{(1-\gamma_1)k_1}{A_4}u_1
\end{aligned}
\tag{3.8}
$$

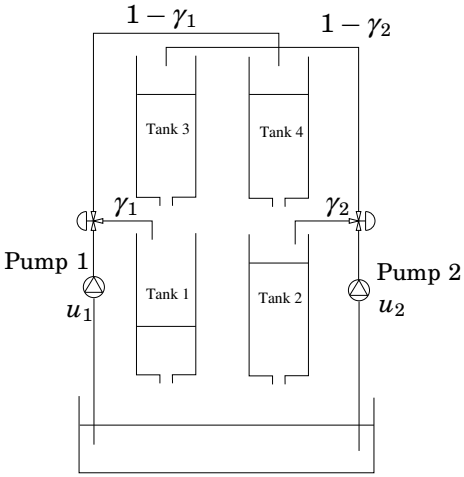

Figure 3.9 A schematic picture of the quadruple tank process

where the A_i:s and the a_i:s represent the cross section area of the tanks and the holes respectively. The parameters γ_i:s determine the position of the valves which control the flow rate to the upper and lower tanks respectively. The control signals are given by the the u_i:s. The objective is to control the levels of the two lower tanks, i.e. x_1 and x_2. Numerical values of the parameters are given in Table 3.1.

An interesting feature of this multivariate process is that the linearized system has an adjustable zero. By adjusting the parameters γ_1 and γ_2 it is possible to obtain a zero with negative real part, or a non minimum phase zero with positive real part. For the simulations, the valve positions were set to $\gamma_1 = 0.30$ and $\gamma_2 = 0.30$, yielding a non-minimum phase system.

In order to evaluate the proposed scheme, three grades were defined, see Table 3.2. Grade changes transferring the plant from grade A to B, grade B to C and grade C to A were calculated using the optimization procedure outlined above. The optimization criterion was

$$P(x(t_f), u(t_f), t_f) = \int_0^{t_f} (u^0 - u(t))^T R(u^0 - u(t))dt \qquad (3.9)$$

Table 3.1 Parameter values of the Quadraple Tank

Parameters	Values	Unit
A_1, A_2	28	[cm^2]
A_3, A_4	32	[cm^2]
a_1, a_2	0.071	[cm^2]
a_3, a_4	0.057	[cm^2]
k_1, k_2	3.33, 3.35	[cm^3/Vs]
k_c	0.50	[V/cm]
g	981	[cm/s^2]

Table 3.2 Stationary values for grade A, B, C.

Grade	x_1^0	x_2^0	x_3^0	x_4^0	u_1^0	u_2^0
A	7.0000	12.0000	3.1720	6.3169	2.8870	2.4321
B	9.0000	12.0000	4.9060	5.2061	2.6209	3.0247
C	7.0000	14.0000	2.7689	8.1416	3.2776	2.2723

where $R = I$, reflecting the fact that the control energy should be small. The target values for the control variables u_0 are assumed to be the values of the new grade. A terminal constraint ensuring that $x(t_f)$ corresponds to the new grade was also imposed.

The parameterization of the control variable trajectories is an important part of the optimization procedure. The solution obtained by the optimization algorithm, and also the convergence of the algorithm, are highly dependent of this choice.

As a first attempt, the control variables of the quadruple tank were parameterized by first order Lagrange polynomials, using 2 segments for each variable. The final time of the optimization horizon was assumed to be fixed, yielding one switching time for each control variable. Continuity of the control trajectories was also imposed. This choice of control variable parameterization results in 10 optimization variables; 4 polynomial weighting coefficients and one interval length for each

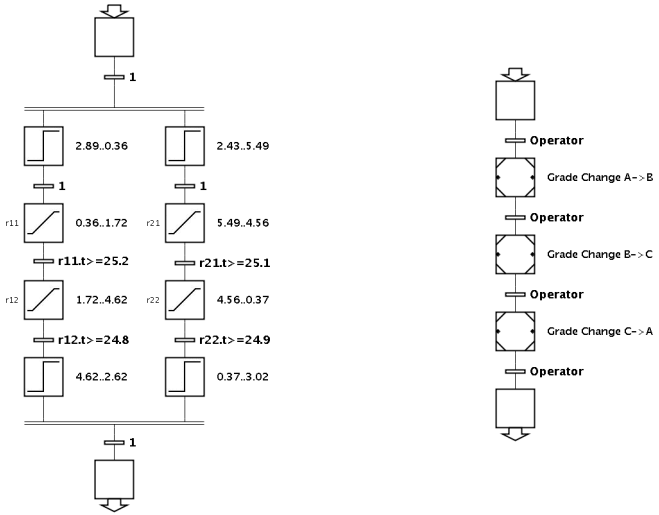

Figure 3.10 Grade change Grafcets.

control variable.

Figure 3.10 shows the Grafcet representing the optimal sequence for transferring the plant from grade A to B. Apart from steps and transitions, parallel split and join elements are used to obtain synchronization between the two sequences. The right Grafcet in Figure 3.10 shows the hierarchical possibilities of `JGrafchart`. Each macro step encapsulates a grade change sequence; the transitions are usually triggered by the operator. For complex grade changes, it should be possible for operators to supervise the procedure more closely and accept manually sub-sequences at an increased level of detail. Figure 3.11 shows the state and control profiles produced by the optimization algorithm. This simple example corresponds to a production schedule in which the grades A, B, and C should be produced.

In a second attempt, piecewise constant polynomials and two segments for each control variable was assumed. The optimization horizon was assumed to be fixed, and was determined individually for each grade change. This control variable parameterization yields 6 op-

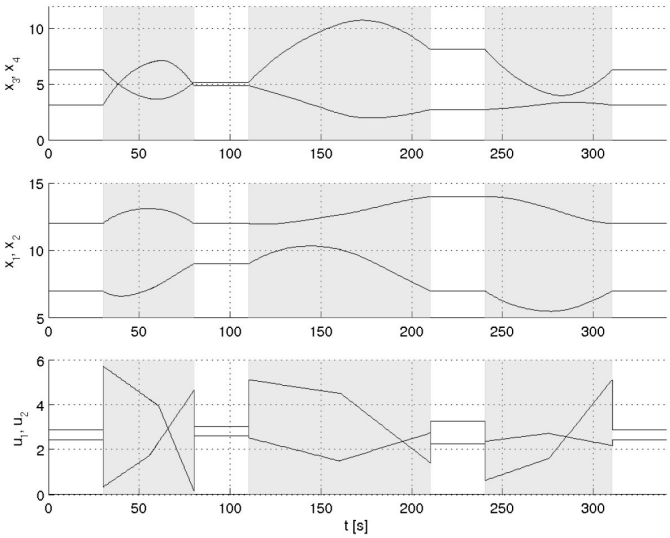

Figure 3.11 The plot shows a simulation run, where the optimal grade change sequences are applied to the plant. The shaded areas mark the grade changes.

timization parameters; two polynomial weighting coefficients and one switching time for each control variable. The resulting grade changes are shown in Figure 3.12. As we can see, the objective of each grade change is fulfilled; at the end of the optimization horizon the state of the process corresponds to the new grade.

It is tempting to increase the complexity of the control variable parameterization, in order to improve performance. It is reasonable to assume that the additional degree of freedom would render a better approximation of the corresponding optimal control problem without any restriction on the control variable parameterization. In a third attempt, piecewise linear polynomials and four segments for each control variable were assumed. As before, continuity of the control variables as well as fixed optimization horizon were assumed. This parameterization yields a more complex search problem for the optimization algorithm, and potentially degraded performance in terms of slow con-

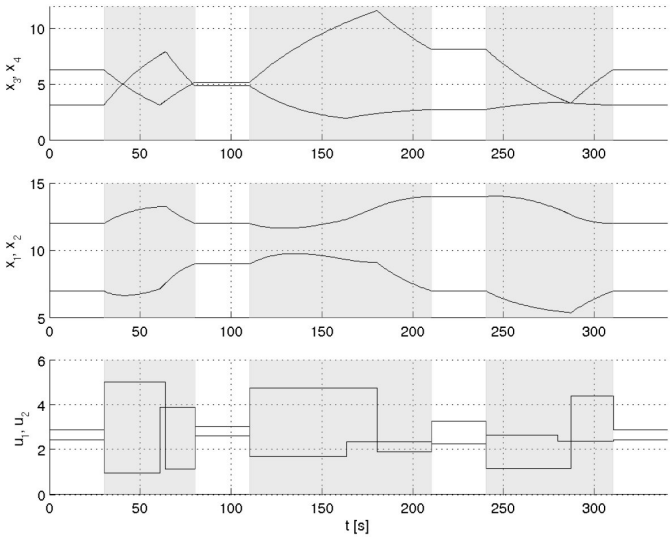

Figure 3.12 The optimal solution assuming 2 segments and piecewise constant control variable trajectories.

vergence. In this case however, no such problems were encountered. The result of the optimization is shown in Figure 3.13.

An interesting question is how well do the optimal solutions presented above approximate the solution of the optimal control problem without restrictions on the control variable parameterization. In order to investigate this, the Matlab toolbox RIOTS (Recursive Integration Optimal Trajectory Solver) was employed. See [Schwartz, 1999] for details on the usage of, and the theory behind this tool. The optimization problem with cost function (3.9) and terminal equality constraint was solved using RIOTS for the grade changes defined above. A large number of segments were used to discretize the optimization horizon. The resulting trajectories are shown in Figure 3.14 for the grade A to grade B transition. For comparison, the solutions described previously are included. As we can see, the simple control variable parameterization based on piecewise constant polynomials approximate the RIOTS

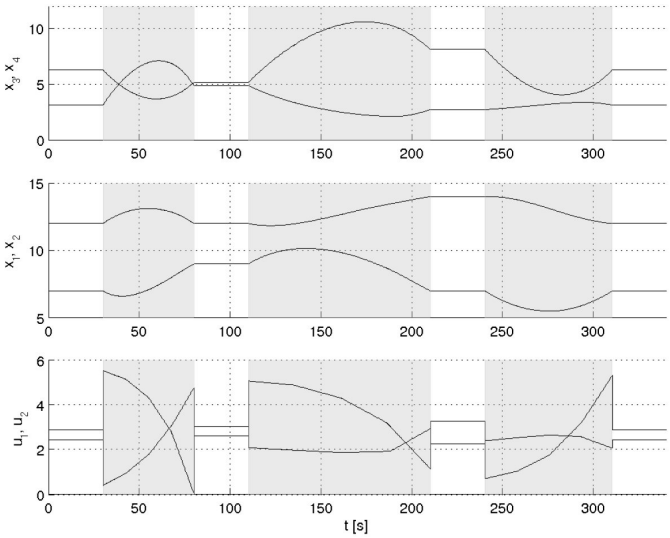

Figure 3.13 The optimal solution assuming 4 segments and piecewise linear control variable trajectories.

solution quite well, if the state trajectories are considered. In the case of four segments and piecewise linear control trajectories, the difference is neglectable. To further quantify the performance improvement obtained by the increased level of detail in the control variable parameterization, the optimal values of the cost function are summarized in Table 3.3. Apparently, in this case, little improvement is achieved by introducing additional degrees of freedom in the control variable parameterization. Consequently, the grade change sequences in this example are well suited for Grafchart representation, since close to optimal performance can be achieved using only a small number of elements. This conclusion, however, cannot be drawn in general. Accordingly, a similar analysis should be done in each particular application.

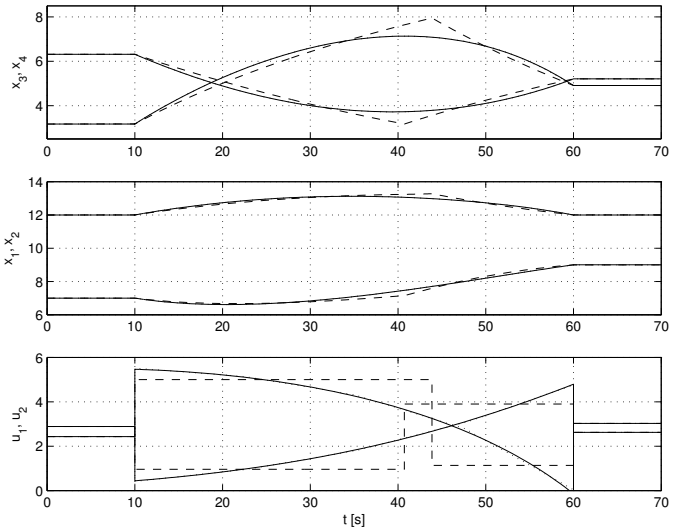

Figure 3.14 The solid curves represent the optimal solution obtained using RIOTS. For comparison, the dashed curves representing the case with piecewise constant control variables and the dotted curves representing the case with piecewise linear control trajectories with four segments are included. Notice that the difference between the latter case and the RIOTS solution is barely visible in the plot.

3.8 Conclusions

In this chapter we have shown how dynamic optimization methods can be used to generate grade change sequences expressed in the sequential language Grafcet. The motivation for the proposed method is the importance of operator acceptance of decision support systems. In `JGrafchart`, grade change sequences are expressed in a structured and hierarchical manner, offering an operator friendly complement to regular time plots. Further, `JGrafchart` enables automatic, semi-automatic or manual execution of grade change sequences, offering an environment suitable for operator support systems.

Table 3.3 Optimal cost function values obtained with different control variable parameterizations.

Grade change	PWC, 2 seg.	PWL, 2 seg.	PWL, 4 seg.	RIOTS
A→B	1.5309	1.2190	1.2148	1.2143
B→C	1.1255	0.8950	0.8664	0.8655
C→A	0.9753	0.7900	0.7864	0.7861

3.9 Future work

The work presented in this chapter can be extended in several directions, with the objective of creating a framework for grade changes applicable to large and complex plants. An important issue that must be treated is robustness. For example, it is necessary for the method to be robust to modeling errors and disturbances, but also to uncertain initial conditions. In the method outlined in this chapter, local control loops implemented the DCS ensures some robustness by introducing feedback. However, feedback could be introduced directly in the Grafcet sequence as well. In the method presented above, the grade change sequence evolves with time, i.e. the transitions in a Grafcet sequence are triggered by time conditions. An attractive complement would be to let the transition conditions depend on the evolution of the plant, e.g. state triggered transitions. In this way additional robustness could be achieved.

Another convenient extension would be to generate Grafcet sequences automatically, given the optimal solution from the optimization algorithm. This problem includes development of methods for automatic hierarchical structuring of sequences, as well as graphical layout of Grafcet elements. An related issue is scalability. The presented framework is intended for large and complex processes, and accordingly, finding methods for structuring that grants for good scalability is essential.

By parameterizing the control variable sequences as steps and ramps, the obtained solution is obviously sub-optimal. Also, the number of segments, i.e. the number of steps and ramps, also affects the

performance of the method. In this chapter, RIOTS has been used to investigate sub-optimality of solutions. An extended comparative study, quantifying the sub-optimality of the proposed method would be interesting.

As noted previously, the availability of a dynamic model is important in order for the proposed method to perform well. Specifically, the structure of the model influences the choice of optimization method. In this chapter, the use of dynamic models based on non-linear ODE:s has been explored. However, other types of models may require a different approach to the dynamic optimization problem.

3.10 References

Askaner, M. (2003): "Simulerad kartongmaskin sparar pengar." *Process Nordic*, **No 1**.

Balderud, J., C. Haag, and D. I. Wilson (2001): "Large-scale dynamic paper machine models." In *6th World Congress of Chemcal Engineering*. Melbourne, Australia.

Bellman, R. (1957): *Dynamic Programming*. Princeton University Press, Princeton, N.J.

Bemporad, A. and M. Morari (1999): "Control of systems integrating logic, dynamics and constraints." *Automatica*, **35**, pp. 407–427.

Bertsekas, D. P. (2000a): *Dynamic Programming and Optimal Control, vol 1*. Athena Scientific.

Bertsekas, D. P. (2000b): *Dynamic Programming and Optimal Control, vol 2*. Athena Scientific.

Biegler, L., A. Cervantes, and A. Wächter (2002): "Advances in simultaneous strategies for dynamic optimization." *Chemical Engineering Science*, **57**, pp. 575–593.

Bryson, A. E. and Y.-C. Ho (1975): *Applied optimal control*. Hemisphere Publishing Corporation.

David, R. and H. Alla (1992): *Petri nets and Grafcet: Tools for modelling discrete events systems*. Prentice-Hall International (UK) Ltd.

Forsman, K. (2002): "Lägesrapport från världens mest automatiserade pappersbruk." In *Reglermöte 2002*. Linköping, Sweden.

Ihalainen, H. and R. Ritala (1996): "Optimal grade changes." In *Proceedings from Control Systems '96*, pp. 213–216. Halifax, Canada.

Johansson, K. H. (1997): *Relay Feedback and Multivariable Control*. PhD thesis ISRN LUTFD2/TFRT--1048--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Johnsson, C. (1999): *A Graphical Language for Batch Control*. PhD thesis ISRN LUTFD2/TFRT--1051--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Lewis, R. (1996): *Programming Industrial Control Systems Using IEC 1131-3*. IEE, London.

Lincoln, B. (2003): *Dynamic Programming and Time-Varying Delay Systems*. PhD thesis ISRN LUTFD2/TFRT--1067--SE.

Magni, L., G. Bastin, and V. Wertz (1999): "Multivariable nonlinear predictive control of cement mills." *IEEE Transactions on Control Systems Technology*, **7:4**, pp. 502–508.

McQuillin, D. and P. W. Huizinga (1994): "Reducing grade change time through the use of predictive multi-variable control." In *Proceedings from Control Systems '94*, pp. 275–281. Stockholm, Sweden.

Murphy, T. F. and S.-C. Chen (1999): "Transition control of papermaking processes: Paper grade change." In *Proceedings of International Conference on Control Applications*, pp. 1278–1283. Kohala Coast-Island of Hawai'i, Hawai'i, USA.

Murphy, T. F. and S.-C. Chen (2000): "Fast grade change for paper making processes." In *Control Systems 2000*. Victoria, Canada.

Olsson, R. (2002): "Exception handling in recipe-based batch control." Technical Report Licentiate thesis ISRN LUTFD2/TFRT--3230--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Qin, S. J. and T. A. Badgwell (2003): "A survey of industrial model predictive control technology." *Control Engineering Practice*, **11**, pp. 733–764.

Rosen, O. and R. Luus (1991): "Evaluation of gradients for piecewise constant optimal control." *Comput. chem. Engng.*, **15:4**, pp. 273–281.

Årzén, K.-E., R. Olsson, and J. Åkesson (2002): "Grafchart for procedural operator support tasks." In *Proceedings of the IFAC World Congress, Barcelona, Spain*.

Schwartz, A. (1999): "The riots home page." `http://www.accesscom.com/~adam/RIOTS/`.

Åström, K. J. (2002): *Introduction to Control*. Department of Automatic Control, Lund Institute of Technology.

Sundarraman, A. and R. Srinivasan (2000): "Decision support for monitoring transitions in chemical plants." In *Third International Conference on Loss Prevention (Saftey, Health and Environment) in the Oil, Chemical and Process Industries*. Singapore.

Vassiliadis, V. (1993): *Computational solution of dynamic optimization problem with general differnetial-algebraic constraints*. PhD thesis, Imerial Collage, London, UK.

Vassiliadis, V. S., E. B. Canto, and J. R. Banga (1999): "Second-order sensitivities of general dynamic systems with application to optimal control problems." *Chemical Engineering Science*, **54**, pp. 3851–3860.

4

Tools for model predictive control

4.1 Introduction

The key feature that distinguishes MPC from most other control strategies is the receding horizon principle. An MPC controller solves, at each sampling instant, a finite horizon optimal control problem. Only the first value of the resulting optimal control variable solution is then applied to the plant, and the rest of the solution is discarded. The same procedure is then repeated at each sampling instant, and the prediction horizon is shifted forward one step. Thereby the name receding horizon control. This strategy includes solving *on-line* an optimal control problem, which enables the controller to deal *explicitly* with MIMO plants and constraints. On the downside are the computational requirements. Solving the optimization problem may introduce computational delay, which, if not considered, may degrade control performance. Also, MPC is a model based control strategy, and a model of the process to be controlled is a necessary requirement for MPC.

Historically, there has been two major selling points for MPC; it works well for MIMO plants, and it takes constraints into account *explicitly*. Both these issues arise frequently in many practical applications, and must be dealt with in order for a control design to be successful. MPC has been particularly successful in the area of pro-

cess control, which is also the field from where MPC originates. Traditionally, MPC has been mainly applied to plants with slow dynamics, where the computational delay is small compared to typical sampling intervals. However, recent reports of MPC applications include plants with fast dynamics, ranging from air plane control to engine control. For a review of industrial use of MPC, including a historical review of the evolution of MPC, see [Qin and Badgwell, 2003]

During the last decade, there has been significant research efforts to sort out the theoretical issues of MPC. Notably, the problem of formulating a stabilizing MPC scheme has received much attention. As a result, several techniques to ensure stability have been presented, see [Mayne *et al.*, 2000] for a review. The theory for MPC based on linear systems is well developed, and strong results ensuring robust stability exists, see [Maciejowski, 2002] for an overview. Also, the optimization problem resulting from linear MPC is a Linear Inequality Constrained Quadratic Programming (LICQP) problem, which is a convex optimization problem, and efficient solution algorithms exist. In particular, existence of a unique global minimum is guaranteed. For non-linear system, there exist MPC formulations that guarantee stability under mild conditions. However, the resulting optimization problem is, in general, non-convex and usually no guarantee of finding a global minimum exits. Non-linear MPC remains a very active field of research and recent results have shown that optimality is not a necessary condition for stability, see [Scokaert *et al.*, 1999]. These promising results show that many non-linear MPC schemes may be stabilizing although finding the global minimum is difficult.

In this chapter, Matlab tools for a standard formulation of linear MPC will be presented. State estimation, error-free tracking and stability are important issues that will be covered in the following. The aim of developing the MPC tools have been to simulate, in detail, the behavior of an MPC controller. In particular, the tools has been used to study the effects of computational delay as described in Chapter 6. A detailed analysis of controller behavior also requires understanding of the algorithm used to solve the LICQP problem. Accordingly, the tools include implementations of common LICQP algorithms for this purpose.

4.2 Linear model predictive control

In this section, an MPC formulation based on linear discrete-time state space models will be described. The presentation is based on [Maciejowski, 2002].

Receding horizon control

The MPC scheme makes use of the receding horizon principle, illustrated in Figure 4.1. At each sample, a finite horizon optimal control problem is solved over a fixed interval of time, the prediction horizon. We assume that we would like the controlled variables, $z(k)$, to follow some set point trajectory, $r(k)$. The optimal control problem is formulated using a cost function penalizing deviations of the controlled variables as well as variations in the control signal. A common choice is to use a quadratic cost function, which in combination with a linear system model yields a finite horizon LQ problem. Figure 4.1 shows the predicted optimal trajectories $\hat{z}(k+i|k)$ and $\hat{u}(k+i|k)$ starting at time k.

The predictions necessary to solve the optimization problem is obtained using a model of the controlled system. The prediction of the controlled variable z is performed over an interval with length H_p samples. The control signal is assumed to be fixed after H_u samples. H_p and H_u are referred to as the prediction horizon and the control horizon respectively. The distinction between the prediction and control horizons is useful since the number of decision variables in the optimization problem increases with H_u, but is independent of H_p. Normally, $H_u < H_p$ in order to reduce the complexity of the optimization problem.

When the solution of the optimal control problem has been obtained, the value of the first first control variable in the optimal trajectory, $u(k|k)$, is applied to the process. The rest of the predicted control variable trajectory is discarded, and at the next sampling interval the entire procedure is repeated.

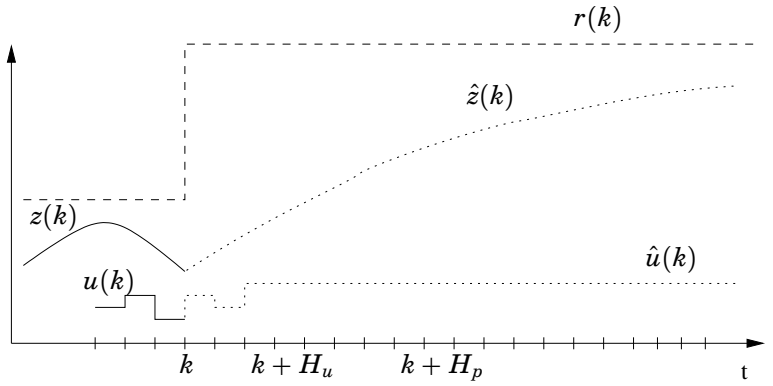

Figure 4.1 The idea of MPC. Here $r(k)$ is the set point trajectory, $z(k)$ represents the controlled output and $u(k)$ the control signal.

Model assumptions

We assume that a model on the form

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= C_y x(k) \\ z(k) &= C_z x(k) + D_z u(k) \\ z_c(k) &= C_c x(k) + D_c u(k) \end{aligned} \tag{4.1}$$

is available. Here $y(k) \in R^{p_y}$ is the measured output, $z(k) \in R^{p_z}$ the controlled output and $u(k) \in R^m$ the input vector. The state vector is $x(k) \in R^n$. The MPC controller should also respect constraints on control variables as well as the constrained outputs, $z_c(k) \in R^{p_c}$

$$\begin{aligned} \Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max} \\ u_{min} \leq u(k) \leq u_{max} \\ z_{min} \leq z_c(k) \leq z_{max} \end{aligned} \tag{4.2}$$

where $\Delta u(k) = u(k) - u(k-1)$ are the control increments.

The distinction between controlled and constrained variables is natural, since only the controlled variables have specified reference values.

This distinction is not made in [Maciejowski, 2002], but is quite useful. For example, there may be plant variables that must respect constraints, without having corresponding reference values. In some cases the constrained variables may not be included in the set of measured variables. In this case, an observer can be used to obtain estimates of such variables. The constraints are then enforced for the *estimated* outputs, which may not be equal to the true constrained outputs. The same argument applies to the controlled outputs, which will be discussed further in the section dealing with error free tracking.

An optimal control problem

We will now formulate the optimal control problem that is the core element of the MPC algorithm. Consider the following quadratic cost function:

$$J(k) = \sum_{i=H_w}^{H_p+H_w-1} \|\hat{z}(k+i|k) - r(k+i|k)\|_Q^2 + \sum_{i=0}^{H_u-1} \|\Delta\hat{u}(k+i|k)\|_R^2 \quad (4.3)$$

where $\hat{z}(k+i|k)$ are the predicted controlled outputs at time k and $\Delta\hat{u}(k+i|k)$ are the predicted control increments. The matrices $Q \geq 0$ and $R > 0$ are weighting matrices, which are assumed to be constant over the prediction horizon. The length of the prediction horizon is H_p, and the first sample to be included in the horizon is H_w. H_w may be used to shift the control horizon, but in the following presentation we will assume that $H_w = 0$. The control horizon is given by H_u. In the cost function (4.3) $\Delta u(k)$ is penalized rather than $u(k)$, which is common in LQ control. The reason for this is that for a non-zero set point, $r(k)$, the corresponding steady state control signal $u(k)$ is usually also non-zero. By avoid penalizing $u(k)$, this conflict is avoided. A different method that has been used is to introduce a set point also for the control variable, $r_u(k)$, and to penalize deviations of $u(k)$ from u_r. This approach may be implemented in the above formulation by choosing $C_z = 0$ and $D_z = I$, and thereby let $u(k)$ be part of the controlled variables.

The cost function (4.3) may be rewritten as

$$J(k) = \left\| \mathcal{Z}(k) - \mathcal{T}(k) \right\|_{\mathcal{Q}}^2 + \|\Delta\mathcal{U}\|_{\mathcal{R}}^2$$

where

$$\mathcal{Z}(k) = \begin{bmatrix} \hat{z}(k|k) \\ \vdots \\ \hat{z}(k+H_p-1|k) \end{bmatrix} \quad \mathcal{T}(k) = \begin{bmatrix} r(k|k) \\ \vdots \\ r(k+H_p-1|k) \end{bmatrix}$$

$$\Delta\mathcal{U}(k) = \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+H_u-1|k) \end{bmatrix} \quad \mathcal{Q} = \begin{bmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q \end{bmatrix}$$

$$\mathcal{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R \end{bmatrix}$$

By deriving the prediction expressions, we can write

$$\mathcal{Z}(k) = \Psi x(k) + \Gamma u(k-1) + \Theta\Delta\mathcal{U}(k) \tag{4.4}$$

where

$$\Psi = \begin{bmatrix} C_z \\ C_z A \\ C_z A^2 \\ \vdots \\ C_z A^{H_p-1} \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} D_z \\ C_z B + D_z \\ C_z AB + C_z B + D_z \\ \vdots \\ C_z \sum_{i=0}^{H_p-2} A^i B + D_z \end{bmatrix}$$

$$\Theta = \begin{bmatrix} D_z & 0 & \cdots & 0 \\ C_zB + D_z & D_z & & \\ C_zAB + C_zB + D_z & & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ C_z\sum_{i=0}^{H_u-2} A^iB + D_z & & \cdots & D_z \\ \vdots & & \ddots & \vdots \\ C_z\sum_{i=0}^{H_p-2} A^iB + D_z & & \cdots & C_z\sum_{i=0}^{H_p-Hu-1} A^iB + D_z \end{bmatrix}$$

Also, let

$$\mathcal{E}(k) = \mathcal{T}(k) - \Psi x(k) - \Gamma u(k-1).$$

This quantity could be interpreted as the free response of the system, if all the decision variables at $t = k$, $\Delta\mathcal{U}(k)$, were set to zero. Inserting the prediction expressions into the cost function (4.3) we obtain

$$J(k) = \Delta\mathcal{U}^T\mathcal{H}\,\Delta\mathcal{U} - \Delta\mathcal{U}^T\mathcal{G} + \mathcal{E}^T\mathcal{Q}\,\mathcal{E} \tag{4.5}$$

where

$$\begin{aligned} \mathcal{G} &= 2\Theta^T\mathcal{Q}\,\mathcal{E}(k) \\ \mathcal{H} &= \Theta^T\mathcal{Q}\,\Theta + \mathcal{R} \end{aligned}$$

The problem of minimizing the the cost function (4.5) is a quadratic programming (QP) problem. If $\mathcal{H}$ is positive definite, the problem is convex, and the solution may be written on closed form. Positive definiteness of $\mathcal{H}$ follows from the assumption that $\mathcal{Q} \geq 0$ and $\mathcal{R} > 0$. The solution is given by

$$\Delta\mathcal{U} = \frac{1}{2}\mathcal{H}^{-1}\mathcal{G}\,.$$

Notice that the matrix $\mathcal{H}^{-1}$ does not depend on k, and may be precalculated. In fact, the controller is linear, and may be calculated offline. This will be discussed in detail in Section 4.2.

Constraints

Let us now introduce constraints on the constrained and control vari-

ables, z_c and u. In general, linear constraints may be expressed as:

$$W\Delta\mathcal{U}(k) \leq w \tag{4.6}$$
$$F\mathcal{U}(k) \leq f \tag{4.7}$$
$$G\mathcal{Z}_c(k) \leq g \tag{4.8}$$
$$\tag{4.9}$$

This formulation allows for very general constraints, but in the following, only the constraints specified by (4.2) will be considered. Using (4.2), we obtain

$$W = F = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & & & \\ 0 & 1 & & \vdots \\ & -1 & & \\ \vdots & & \ddots & 0 \\ & & & 1 \\ 0 & \cdots & 0 & -1 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & & & \\ 0 & 1 & & \vdots \\ & -1 & & \\ \vdots & & \ddots & 0 \\ & & & 1 \\ 0 & \cdots & 0 & -1 \end{bmatrix}$$

$$w = \begin{bmatrix} \Delta u_{max} \\ -\Delta u_{min} \\ \vdots \\ \Delta u_{max} \\ -\Delta u_{min} \end{bmatrix} \quad f = \begin{bmatrix} u_{max} \\ -u_{min} \\ \vdots \\ u_{max} \\ -u_{min} \end{bmatrix} \quad g = \begin{bmatrix} z_{max} \\ -z_{min} \\ \vdots \\ z_{max} \\ -z_{min} \end{bmatrix}$$

Notice that W and F may not be of the same size as G, though the structure is the same. Since $\mathcal{U}(k)$ and $\mathcal{Z}_c(k)$ are not explicitly included in the optimization problem, we rewrite the above constraints in terms of $\Delta\mathcal{U}(k)$. This gives

$$\begin{bmatrix} \mathcal{F} \\ G\Theta_c \\ W \end{bmatrix} \Delta\mathcal{U} \leq \begin{bmatrix} -\mathcal{F}_1 u(k-1) + f \\ -G(\Psi_c x(k) + \Gamma_c u(k-1)) + g \\ w \end{bmatrix} \tag{4.10}$$

where

$$\mathcal{F} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & & & \\ 1 & 1 & & \vdots \\ -1 & -1 & & \\ \vdots & & \ddots & 0 \\ 1 & & & 1 \\ -1 & \cdots & & -1 \end{bmatrix} \quad \mathcal{F}_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ \vdots \\ 1 \\ -1 \end{bmatrix}$$

The definitions of Ψ_c, Θ_c and Γ_c are equivalent to those of Ψ, Θ and Γ. As we can see, the left side of the inequality is not dependent on k, and could be calculated off-line. The right side depends on the last control signal and the present estimation of the state vector, and should thus be evaluated at each sample.

The optimization problem can now be rewritten using (4.3) and (4.10)

$$\begin{aligned} &\min J(k) = \Delta\mathcal{U}^T \mathcal{H} \Delta\mathcal{U} - \Delta\mathcal{U}^T \mathcal{G} + \mathcal{E}^T Q \mathcal{E} \\ &\text{subject to } \Omega\Delta\mathcal{U} \leq \omega \end{aligned}$$

The problem is still recognized as a quadratic programming problem, but now with linear inequality constraints. The problem is convex, but due to the constraints, it is not possible to write the solution on closed form. Rather iterative algorithms have to be employed. This issue will be discussed further in Section 4.3

Stabilizing MPC algorithms

The problem of formulating stabilizing MPC schemes has received much attention in the last decade. The conditions for stability are now quite well understood, and several techniques for ensuring stability exist, including methods based on terminal penalty, terminal equality constraint and terminal sets, see [Mayne *et al.*, 2000] for an overview. These results are valid both for linear and non-linear MPC, although for linear systems the situation is less complicated due to the convexity of the optimization problem.

The following theorem summarizes the important features of a stabilizing MPC scheme based on a terminal equality constraint [Bem-

porad *et al.*, 1994]. Without lack of generality we assume that $r(k)$ is zero.

THEOREM 4.1
Consider the system (4.1) controlled by the receding horizon controller based on the cost function (4.3), subject to the constraints (4.10). Let $r(k) = 0$. Further assume terminal constraints $\hat{x}(k + H_p + 1) = 0$ and $\hat{u}(k + H_u) = 0$, $Q \geq 0$ and $R > 0$ and that $(Q^{\frac{1}{2}} C_z, A)$ is a detectable pair. If the optimization problem is feasible at time k, then the origin is stable, and $z(k)^T Q z(k) \to 0$ as $k \to \infty$.
Proof:
Let

$$\Delta \mathcal{U}_k^* = (\Delta \hat{u}_k^*(k), \Delta \hat{u}_k^*(k+1), \ldots, \Delta \hat{u}_k^*(k + H_u - 1))$$

denote the optimal control sequence at time k. Obviously,

$$\Delta \mathcal{U}_{k+1} = (\Delta \hat{u}_k^*(k+1), \ldots, \Delta u_k^*(k + H_u - 1),\ 0)$$

is then feasible at time $k + 1$. Consider the function

$$V(k) = J(k, \Delta \mathcal{U}_k^*, x(k))$$

with $r(k) = 0$. Then we have the following relations:

$$\begin{aligned} V(k+1) &= J(k+1, \Delta \mathcal{U}_{k+1}^*, x(k+1)) \\ &\leq J(k+1, \Delta \mathcal{U}_{k+1}, x(k+1)) \\ &= V(k) - z(k+1)^T Q z(k+1) \\ &\quad - \Delta u(k)^T R \Delta u(k). \end{aligned} \tag{4.11}$$

Since $V(k)$ is lower-bounded and decreasing,

$$z(k)^T Q z(k) \to 0$$

and

$$\Delta u(k)^T R \Delta u(k) \to 0$$

as $k \to \infty$. Further, using the fact that $(Q^{\frac{1}{2}} C_z, A)$ is a detectable pair, it follows that

$$\|x(k)\| \to C < \infty$$

as $k \to \infty$. □

REMARK 4.1
To prove the stronger result that the origin is asymptotically stable, the additional assumption that the system (4.1) has no transmission zeros at $q = 1$ from u to z could be imposed. Notice also that the sensible assumption that $Q > 0$ implies that $z(k) \to 0$ as $k \to \infty$, which is, however, automatically achieved if the transmission zero condition is fulfilled. □

There are several important points that should be stressed. The assumption that there exist an initial feasible solution (subject to the constraints (4.10) and $\hat{x}(k + H_p + 1) = 0$) is non trivial; for some regions in the state space a feasible solution may not exist. However, in an actual application, it is often possible to use the control signal $\Delta\hat{u}(k + 1)$ calculated at the previous sample, in hope that the optimization problem at the next sample is feasible. Also, using a more sophisticated stabilizing technique, such as a terminal penalty and a terminal set the feasible region may be extended.

Recent results explore another important feature of Lyapunov-based MPC stability proofs; optimality is not necessary for stability. In [Scokaert *et al.*, 1999] it is shown how stability can be achieved by ensuring feasibility of the solution, with respect to some criterion, rather than optimality. Using this technique, it is recognized that it is sufficient to ensure that the cost function is decreasing in each sample in order for the stability proof to work. An application of these results is found in Section 6, where premature termination of the optimization algorithm is considered in order to reduce the effects of computational delay. Another area of importance is non-linear MPC, where finding the optimal solution may be difficult due to multiple local minima.

State estimation

The algorithm for obtaining the optimal control signal at each sample assumes that the present state vector is available. Since this is often not the case, state estimation is required. The celebrated separation principle, stating that the optimal control and optimal estimation problems solved independently, yields a globally optimal controller for linear systems, suggests an attractive approach. We let the solution of the optimization problem be based on an estimate of the state vector, $\hat{x}(k)$ instead of the true state vector $x(k)$. For this purpose, a Kalman

filter,

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K(y(k) - C_y\hat{x}(k)).$$

can be used. If the covariance matrices of the states, W, and the measurement noise, V, are assumed to be known, the gain matrix K may be obtained by solving an algebraic Riccati equation, see for example [Åström and Wittenmark, 1990].

Apart from estimating the state of the system, an estimator could be used to estimate disturbances, assuming that a disturbance model is available. For example, error-free tracking may be achieved by including a particular disturbance model in the observer.

Error-free tracking

In practical applications, there are always modeling errors and disturbances present. The MPC formulation described above contains no explicit mechanism to deal with these complications. In order for the controller to be useful in practice, these problems have to be considered. Commonly, the controller is designed so that it contains integral action, which ensures zero steady-state error. There are several methods for achieving integral action in a controller. For SISO systems, introduction of integral action is quite straight forward. A common approach is to introduce an integrator state in the state space model:

$$\begin{bmatrix} x(k+1) \\ x_i(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C_z & I \end{bmatrix} x(k) + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ I \end{bmatrix} r(k)$$
$$y(k) = \begin{bmatrix} C_y & 0 \end{bmatrix}$$
$$z(k) = \begin{bmatrix} C_z & 0 \end{bmatrix}.$$

A stabilizing feedback control law may then be calculated based on the extended model. The integrator state is implemented in the controller, and used for feedback together with the true or estimated states. From the definition of the extended system model, it is clear that in steady state, $z = r$. This approach does not work so well for MPC controllers. In particular, it is not clear how the integral state should be introduced in the cost function in order for the integral action to work properly.

A different approach to achieve integral action is to use a disturbance observer. In summary, the extended system model

$$\begin{bmatrix} x(k+1) \\ v_a(k+1) \\ d(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 & B \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ v_a(k) \\ d(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u(t)$$

$$z(k) = y_z(k) = \begin{bmatrix} C_z & 0 & 0 \end{bmatrix} \begin{bmatrix} x(k)^T & v_a(k)^T & d(k)^T \end{bmatrix}^T$$

$$y_a(k) = \begin{bmatrix} C_a & I & 0 \end{bmatrix} \begin{bmatrix} x(k)^T & v_a(k)^T & d(k)^T \end{bmatrix}^T$$

is used. It is assumed that the number of controlled outputs, z, equals the number of inputs, u. Additional outputs, if any, are denoted y_a. Also, the controlled variables are assumed to be included in the set of measured variables. A detailed treatment of this method is given in Chapter 5.

Blocking factors

In some situations it may be advantageous to let the control signal be fixed over several consecutive predicted samples. In this way, the control horizon may be increased without increasing the complexity of the optimization problem. Also, ringing behavior of the control signal may be avoided. For example, suppose that the control horizon has to be increased in order to increase closed loop performance. If the control horizon is increased, the time to solve the optimization problem will also increase. If this is not acceptable, one approach might be to include only every other decision variable in the optimization problem, assuming that the the control signal is fixed over two consecutive sampling intervals. We denote the set of predicted sample indexes for which the control signal is allowed to vary by I_u.

In the MPC formulation given above this means that some $\Delta\hat{u}(k+i|k)$:s are set to zero for certain i:s. This means that the corresponding columns in the matrices Θ, W and F may be neglected.

In a similar way it is possible to generalize the prediction horizon. Instead of including all predicted values in the interval $[k...k+H_p-1]$, we introduce the set I_p consisting of all sample indexes for which the corresponding predicted output values are included in the cost function

and for which the constraints are enforced. The last point is critical. It may be tempting to introduce a sparse set of predicted sampling instants in order to obtain a longer prediction horizon. However, since the inter-sample behavior is neglected, this may lead to the constraint in effect being violated at some points.

This generalization is easily introduced by neglecting the rows of the matrices Ψ, Γ, Θ and G corresponding to sample indexes not present in I_p.

Using the notation introduced above, the cost function may be rewritten as

$$J(k) = \sum_{i \in I_p} \|\hat{z}(k+i|k) - r(k+i|k)\|_Q^2 + \sum_{i \in I_u} \|\Delta\hat{u}(k+i|k)\|_R^2 . \qquad (4.12)$$

Linear properties of the MPC controller

The behavior of the MPC controller is intrinsically nonlinear, since constraints on state and control variables are taken into account. However, if no constraints are present in the problem formulation, the controller is linear. Also, the controller behaves linearly during operation when no constraints are active. In the first case, the control law, could (and should) be calculated off-line, whereas in the second case, the optimization procedure must be done each sample. There are however, methods for avoiding on-line solution of the optimization problem. Using the observation that the MPC control law is piecewise linear in the states, it is possible to calculate, off-line, all possible control laws. The on-line optimization problem is then transformed into a search problem, where the objective is to find the appropriate partition in the state space, identifying the corresponding control law. This approach is described in [Bemporad *et al.*, 2002].

We will now analyze the linear properties of the MPC controller. The analysis is valid for the case when no constraints are present or the controller operates so that no constraints are active. In this case,

the minimizing solution of the quadratic programming problem is

$$\begin{aligned}\Delta\mathcal{U}(k) &= (\Theta^T\mathcal{Q}\,\Theta+\mathcal{R})^{-1}\Theta^T\mathcal{Q}\,\mathcal{E}(k)\\ &= (\Theta^T\mathcal{Q}\,\Theta+\mathcal{R})^{-1}\Theta^T\mathcal{Q}\begin{bmatrix}\begin{bmatrix}I\\ \vdots\\ I\end{bmatrix} & -\Gamma & -\Psi\end{bmatrix}\begin{bmatrix}r(k)\\ u(k-1)\\ \hat{x}(k)\end{bmatrix}\\ &= \bar{K}_s\begin{bmatrix}r(k)\\ u(k-1)\\ \hat{x}(k)\end{bmatrix}\end{aligned}$$

Now, since only the first of the predicted control signals are applied we can write the control law as

$$\begin{aligned}\Delta u(k|k)=\Delta u(k) &= K_s\begin{bmatrix}r^T(k) & u^T(k-1) & \hat{x}^T(k)\end{bmatrix}^T\\ &= \begin{bmatrix}K_{sr} & K_{su} & K_{sx}\end{bmatrix}\begin{bmatrix}r^T(k) & u^T(k-1) & \hat{x}^T(k)\end{bmatrix}^T\end{aligned}$$

where K_s is given by the first m rows of $\bar{K}_s$. This control law is linear, and the constant gain matrix K_s may be calculated off-line. The block diagram of the controller may now be drawn as in Figure 4.2. In this figure, the transfer function (matrix) of the plant is given by $P(z)$, and $H_u(z)$ and $H_y(z)$ represents the observer. This block diagram is readily converted into a feedback system on standard form shown in Figure 4.3, with

$$\begin{aligned}P(z) &= C_y(zI-A)^{-1}B\\ F(z) &= K_{sr}\\ H(z) &= -K_{sx}H_y(z)\\ K(z) &= \frac{z}{z-1}\left[I-\frac{1}{z-1}K_{su}-\frac{z}{z-1}K_{sx}H_u(z)\right]^{-1}\\ H_y(z) &= (zI-A+KC_y)^{-1}K\\ H_u(z) &= (zI-A+KC_y)^{-1}B\end{aligned}$$

It is now straight forward to apply standard linear analysis methods. For example, the poles and zeros of the closed loop system may be calculated, as well as the sensitivity of the system.

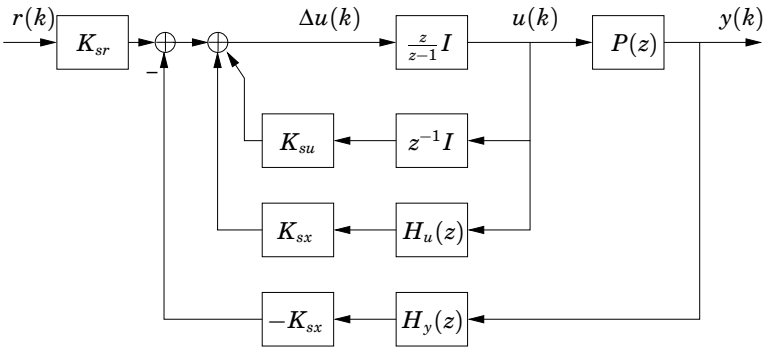

Figure 4.2 The block diagram for the MPC controller

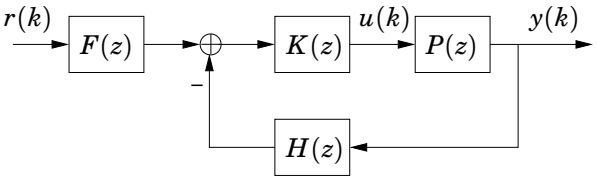

Figure 4.3 A standard feedback structure.

4.3 Quadratic programming algorithms

An important element of the MPC algorithm described above is the algorithm for solving the LICQP problem. The problem at hand is

$$\begin{aligned} &\min J(k) = \Delta u^T H \Delta u - \Delta u^T G + E^T Q E \\ &\text{subject to } \Omega \Delta u \leq \omega. \end{aligned}$$

This problem has several nice features. For example, the objective function is convex, since it is quadratic with positive definite Hessian. Also, the constraints are also convex. Given these conditions, it is a well known result that a local minimum, if it exists, is also a global minimum. (See for example Theorem 4.3.8 in [Bazaraa *et al.*, 1993].) When designing numerical algorithms, this property is of course very valu-

able, since we know in advance that if we find a minimum, it is indeed a global minimum.

There exist several algorithms for constrained optimization, see for example [Fletcher, 1987]. For quadratic programming problem the two most common approaches are primal-dual internal point methods and active set methods [Maciejowski, 2002].

The active set algorithm assumes an initial point in the decision variable space that fulfills the constraints. The active set is defined as the set of all active constraints at this point. A constraint is said to be active if a particular point in the search space is at the boundary of the feasible region defined by the constraint. In the case of linear constraints, these boundary surfaces are given by hyper-planes. In each iteration step, a quadratic programming problem with linear equality constraints (namely those in the active set) is solved. The solution of this problem may be written on closed form. Possibly this solution leads to the introduction of a new constraint into the active set. By calculating the Lagrange multipliers for the problem at each iteration, it is possible to conclude if a constraint may be relaxed, that is, removed from the active set. The algorithm terminates when the gradient of the associated Lagrange function is identically zero, and all Lagrange multipliers are positive.

One problem remains to deal with; the feasible initial point. In order to start the active set algorithm, we need a feasible solution, that is, we would like to find a solution that fulfills the constraints $\Omega \Delta u \leq \omega$. Of course, such a solution is not likely to be unique. Several methods exist for obtaining the desired solution. For example, the problem may be cast as an LP problem, and solved by the simplex algorithm. Another and possibly more attractive alternative is given in [Fletcher, 1987]. This strategy employs an active set technique similar to the one for solving the main quadratic programming problem. However, in this case, the objective function is defined at each iteration as the sum of the violated constraint functions.

Primal-dual interior point methods on the other hand, explores the Karush-Kuhn-Tucker conditions explicitly. The name of this family of QP algorithms stems from the fact that the primal and the dual problems are solved simultaneously. It is important to note however, that the term *interior point* refers to the fact that the algorithm maintains a solution in the interior of the *dual* space. In fact, the primal solution

may not be feasible during the optimization run, except at the optimal point. This constitutes an important difference between active set methods and primal-dual interior point methods. Specifically, in the former case it is possible to terminate the optimization algorithm prematurely and still obtain a feasible, but sub-optimal, solution whereas in the latter case, the algorithm may terminate only when the optimal solution is found.

Concerning performance, both methods have advantages and disadvantages when applied to MPC. Rather, the key to achieve good performance lies in exploring the special structure of the MPC QP problem. See [Bartlett *et al.*, 2000] for a comparison between interior point and active set methods.

4.4 MPC tools for Matlab

In this section, Matlab tools implementing the algorithms described in the previous sections are presented. The main objective for implementing the MPC tools has been to enable detailed study of the behavior of an MPC controller. In this section, the functionality of the tools is described briefly, see Appendix B for a detailed description.

A quadratic programming solver

Obviously, a quadratic programming solver is essential for the implementation of the MPC controller. A solver of this type is available for example from the Matlab Optimization Toolbox; `quadprog`. However, the aim of the development of the Matlab tools has been to create a complete implementation of an MPC controller. Since this also includes an algorithm to solve the QP problem, a solver based on active sets as well as a primal-dual interior point QP solver have been implemented. In addition, an algorithm for finding an initial feasible solution for the active set solver has been implemented. The following Matlab functions implement the necessary algorithms:

- `getfeasible`
  This function obtains a feasible point given the constraints $Ax \leq b$. The algorithm is described in [Fletcher, 1987, p. 166].

- `qp_as`
  This is an active set solver, based on [Fletcher, 1987, p. 240]. The algorithm finds the solution of the optimization problem

$$\begin{aligned} &\min \frac{1}{2} x^T H x + f^T x \\ &\text{s.t.} \\ &Ax \leq b \end{aligned}$$

  using an initial solution x_0 as starting point. The possibility of starting the algorithm from an initial solution is quite useful when implementing the MPC controller. Usually, the solution obtained at the previous sample is a good initial guess, and the number of iterations may be reduced by supplying this solution to the QP algorithm

- `qp_ip`
  This function solves the same quadratic problem problem as `qp_as`. The algorithm implements the Mehrotra predictor-corrector primal-dual interior point method based on [Wright, 1997].

The MPC controller described in the next section enables the user to explore either quadratic programming method, as well as the Optimization Toolbox function `quadprog`.

MPC tools

The Matlab functions presented in this section implement an MPC controller as described in Section 4.2. The tools enable the user to simulate a linear or nonlinear plant model, with an MPC controller engaged. Most of the features described in Section 4.2 are implemented, including constraint handling, observer support, blocking factors and error-free tracking. Also, a basic tool for analyzing the linear properties of the controller is supplied.

In summary, the same assumptions as in Section 4.2 are made for the Matlab implementation of the MPC controller. We assume that a linear discrete time model on the form (4.1), with linear inequality constraints (4.2) is available. Also, we assume that the design variables H_w, H_p, H_u and the blocking factors are specified, as well as the

weighting matrices Q, R and, if applicable, W and V for the design of a Kalman filter.

An important goal in designing the MPC tools has been to enable the user to experiment with different configurations of the MPC controller. Consequently, the MPC tools support several modes of operation:

- `Mode 0`: State feedback.
- `Mode 1`: State feedback with explicit integrators.
- `Mode 2`: Observer-based output feedback.
- `Mode 3`: Observer-based output feedback with explicit integrators.
- `Mode 4`: Observer-based output feedback with a disturbance model that gives error free tracking.

Both state and output feedback is supported. Also, two different strategies for achieving error-free tracking are provided; explicit integrators and an observer based solution. Both methods are described in Section 4.2.

The MPC tools are implemented in five functions:

- `MPCInit`
  Typically, `MPCInit` is used to initialize the MPC controller. Most of the matrices needed to solve the optimization problem may be calculated off-line in advance, for example the Hessian of the QP problem. `MPCInit` implements pre-processing of matrices, which drastically improves the performance of the controller.
- `MPCSim`
  `MPCSim` simulates a linear plant model controlled by the MPC controller. Reference trajectories as well as input disturbance trajectories can be supplied. The function produces the state and control variable trajectories, as well as the predicted trajectories for the controlled and control variables. The latter feature enables the user to examine the predicted solutions obtained at each sample, and explore the effect of different choices of prediction horizons.

- `MPCfrsp`
  As noted in Section 4.2, the MPC controller behaves linearly if no constraints are active. It is then interesting to use standard tools for analysis of linear systems. The function `MPCfrsp` provides frequency response plots for relevant transfer functions corresponding to, e.g., the closed loop system and input and output sensitivity functions.
- `MPCOptimizeSol`
  `MPCOptimizeSol` is a low level function used by other functions rather than by the user. This function provides the control signal of the MPC controller given the current state measurement (or estimation) and reference value.
- `MPCController`
  Most real world plants are nonlinear. It is therefore of interest to investigate the performance of the linear MPC controller applied to a nonlinear plant model, if it is available. This functionality is provided by the simulation environment Simulink for Matlab. The MPC controller has been adapted to Simulink, and is implemented by a Matlab S-function. The S-function is a standard block in Simulink, which may in turn be connected to arbitrary blocks. For an example, see Figure 4.4. The name of the S-function is `MPCController`, and is intended for use with the Simulink S-function block. It takes the argument `md`, which is a data structure created by the `MPCInit` function. The `MPCController` block takes as its inputs a vector signal consisting of the measured outputs of the plant and the reference value. Its output is a vector signal consisting of the control signal u, and the internal state estimation of the controller, $\hat{x}$. The latter may, apart from the estimated states of the plants, also include estimated disturbances or integrator states.

Using the MPC tools described above, it is possible to simulate, at a high level of detail, the behavior of an MPC controller. For example, the consequences of various assumptions regarding measurements, integral action schemes, choice of QP algorithm and prediction horizons are easily explored. Also, the use of Simulink significantly increases the applicability of the tools, enabling the user to simulate the behavior of the MPC controller when applied to nonlinear plants.

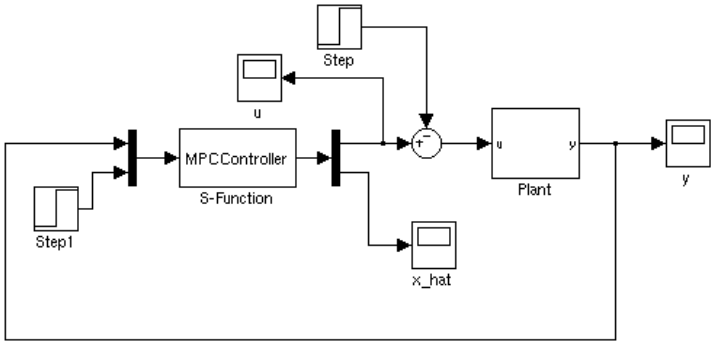

Figure 4.4 A Simulink model where the MPC controller is used to control a nonlinear plant.

4.5 Case studies

In this section, two cases studies are reported, illustrating the main functionality of the MPC tools.

The Quadruple Tank

In Chapter 3, the Quadruple Tank process was described. The process is nonlinear, and in order to apply the MPC tools, a linearized model must be derived. Introducing $\Delta x = x - x^0$, $\Delta u = u - u^0$ and $\Delta y = y - y^0$, we obtain

$$\begin{aligned}
\Delta \dot{x} &= \begin{bmatrix} -\frac{1}{T_1} & 0 & \frac{A_4}{A_1 T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{bmatrix} \Delta x + \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2) k_2}{A_3} \\ \frac{(1-\gamma_1) k_1}{A_4} & 0 \end{bmatrix} \Delta u \\
\Delta y &= \begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix} \Delta x
\end{aligned} \tag{4.13}$$

Table 4.1 Stationary values corresponding to $\gamma_1 = 0.25$ and $\gamma_2 = 0.35$.

Variables	Values	Unit
x_1^0, x_2^0	8.2444, 19.0163	[cm]
x_3^0, x_4^0	4.3146, 8.8065	[cm]
u_1^0, u_2^0	3, 3	[V]

where

$$T_i = \frac{A_i}{a_i}\sqrt{\frac{2x_i^0}{g}}.$$

The stationary operating conditions, x^0 and u^0, are given in Table 4.1. The particular choice of valve positions $\gamma_1 = 0.25$ and $\gamma_2 = 0.35$ yields a non-minimum phase system. The measured outputs of the system are the levels of the lower tanks, represented by a voltage ranging from 0 to 10 V. The objective of the control system is to control, independently, the level of the lower tanks, while preventing overflow in any of the four tanks. The maximum level of the tanks is 20 cm, corresponding to 10 V. In the simulations, the tank level constraints were set to 19.8 cm to ensure some safety margin. Also, the operation of the pumps is limited to 0 to 10 V. We notice that the stationary level of the second tank is close to the maximum level. The combination of a non-minimum phase system and an operating point close to a constraint yields a quite challenging control problem, where two of the main benefits of the MPC controller, namely constraint handling and MIMO support, will be useful. The plant was discretized using the sampling interval $h = 3$ s, resulting in a discrete time model used for the MPC control design.

Control of the linearized model In Table 4.2, the controller parameters used in the simulations are summarized. The choice of prediction and control horizons have been made considering the time constants of the system. Too short horizons may cause instability. However, the prediction and control horizons must not be chosen to large, since it would result in an unnecessarily complex QP problem to solve at each sample. In order to increase the horizons without increasing

Table 4.2 MPC controller parameters for the Quadruple Tank simulations

Parameter	Value
H_p	30
H_w	1
H_u	10
I_p	Only every other sample was included in the optimization problem.
I_u	Every other control increment was assumed to be constant.
Q	diag(4, 1)
R	diag(0.01, 0.01)
W	diag(1, 1, 1, 1)/diag(1, 1, 1, 1, 1, 1)
V	diag(0.01, 0.01)

the number of optimization variables, the blocking factor 2 has been specified for both the control and prediction horizons.

Obviously, since all states are not measurable, an observer must be used. In the first simulation, controller mode 2 was used: a Kalman filter was designed, but no mechanism to ensure integral action was assumed. The result of the simulation is shown in Figure 4.5, represented by the dashed curves. The dotted curves represent the set points for the lower tanks. A step change in the set point of level 1 of size 6 cm is applied at $t = 60$ s, while the set point of level 2 is held constant. As we can see, the controller achieves the correct set points, and in particular, the level of tank 2 does not exceed 20 cm. At $t = 600$ s, a unit step disturbance is applied to the input channel 2. As expected, the MPC controller does not manage to achieve error-free tracking in the presence of load disturbances.

In a second design, control mode 4 was assumed. In this case, the disturbance observer described in Chapter 5 was used to estimate the state vector and the disturbance states. The response of the system is identical to the previous case during the step response. However, the input disturbance is now rejected.

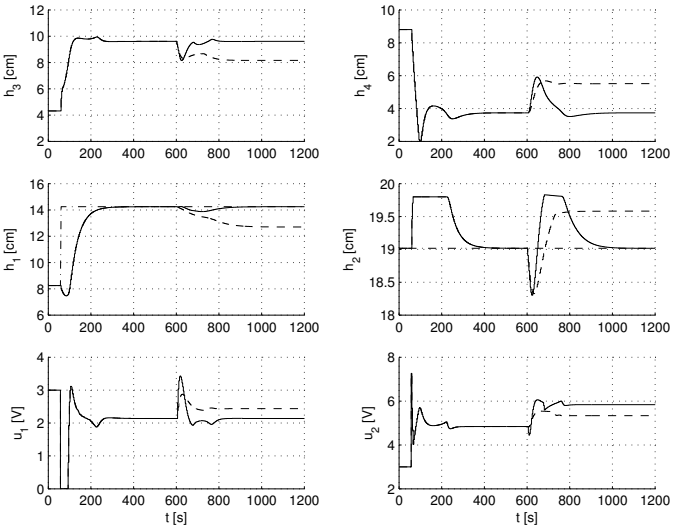

Figure 4.5 Simulations of the MPC controller applied to the linearized plant.

Control of the nonlinear model As noted above, the true Quadruple Tank system is nonlinear. A more realistic scenario would then be to apply the MPC controller designed above to the nonlinear plant model. The plant model was implemented in Simulink, and an S-function block representing the MPC controller was connected to the plant model block. The design parameters of the MPC controller were identical to the previous case, see Table 4.2. Also the same simulation scenario was assumed. The result of the simulation can be seen in Figure 4.6. The dashed curves represent the case when an observer without disturbance states is employed. As we can see, the controller fails to achieve error-free tracking even when no disturbances are present, which is due to the fact that there is a model-mismatch between the linear and the nonlinear plant. Obviously, the controller also fails to compensate fully for the load disturbance. If the disturbance observer is employed however, the controller gives zero steady-state error. This is shown by the solid curves. In both cases, however, the controller respects the constraints: no tank level exceed 20 cm.

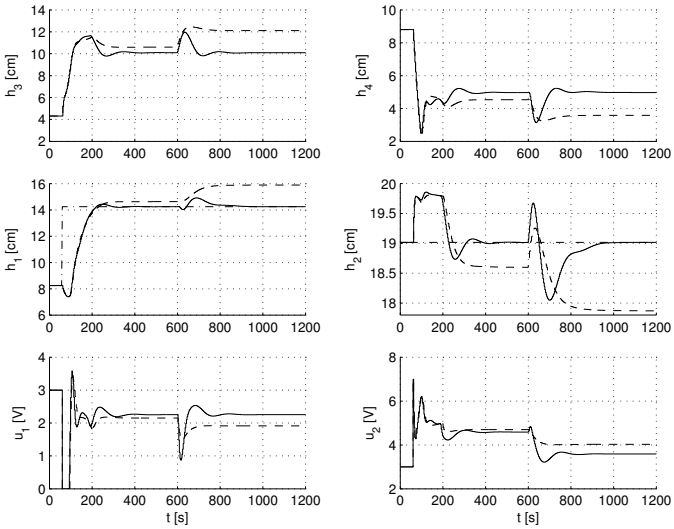

Figure 4.6 Simulations of the MPC controller applied to the nonlinear plant.

The Helicopter Process

As an example of a process with fast dynamics, we consider the helicopter process shown in Figure 4.7. The helicopter consists of an arm mounted to a base, enabling the arm to rotate freely. The arm carries the helicopter body on one end and a counterweight on the other end. The helicopter body consists of a bar connected to the arm in such a way that it may rotate around the arm axis. To the bar is connected two propellers, which may be used to maneuver the plant. A schematic image of the process is shown in Figure 4.7, where the elevation angle, θ_e, measures the angle of the arm with respect to the horizontal plane, the travel angle, θ_r, measures the rotational position of the arm and the pitch angle, θ_p, measures the angle of the bar. A simple dynamical

Table 4.3 Parameter values of the helicopter process

Parameter	Value	Unit	Description
J_e	0.91	[kgm^2]	Moment of inertia about elevation axis
l_a	0.66	[m]	Arm length from elevation axis to helicopter body
K_f	0.5		Motor Force Constant
F_g	0.5	[N]	Differential force due to gravity and counter weight
T_g	$l_a F_g$	[Nm]	Differential torque
J_p	0.0364	[kgm^2]	Moment of inertia about pitch axis
l_h	0.177	[m]	Distance from pitch axis to either motor
J_t	0.91	[kgm^2]	Moment of inertia about travel axis

model of the system is given by

$$\begin{aligned} \ddot{\theta}_e &= (K_f l_a / J_e)(V_f + V_b) - T_g / J_e \\ \ddot{\theta}_r &= -(F_g l_a / J_t) \sin \theta_p \\ \ddot{\theta}_p &= (K_f l_h / J_p)(V_f - V_b) \end{aligned} \tag{4.14}$$

where the coefficients are given in Table 4.3. The input signals to the system are V_f and V_b which represent the voltages fed to the propeller motors. In addition, the elevation and pitch angles are constrained, so that

$$-0.5 \leq \theta_e \leq 0.6 \quad -1 \leq \theta_p \leq 1.$$

The process was linearized around the stationary point

$$(\theta_e^0,\ \theta_r^0,\ \theta_p^0,\ V_f^0,\ V_b^0,) = (0,\ 0,\ 0,\ T_g/(2K_f l_a),\ T_g/(2K_f l_a))$$

and then discretized using the sampling interval $h = 0.2$ s. This gives a discrete-time state space model of the process to be used to design the MPC controller.

The task of the control system is to control, independently, the elevation angle and the rotation angle. The main constraint is that the

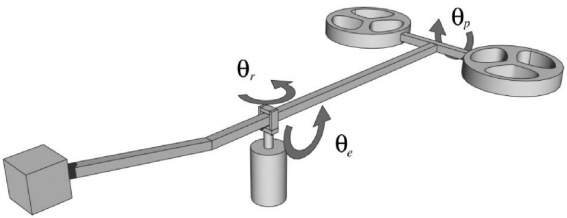

Figure 4.7 The helicopter process.

pitch angle must not be too large, reflecting the fact that for large θ_p, the control authority in the θ_e direction is small. This is not accounted for in the linearized model, and the controller is likely to have degraded performance when θ_p is large.

The design parameters of the MPC controller are given in Table 4.4. Again, the choice of prediction and control horizons have been made considering the dominating time constants of the system. We also assume that the entire state vector is measurable, enabling the use of the state feedback configuration of the MPC controller corresponding to mode 0. In order to achieve fast sampling, (which requires a QP problem of moderate complexity) while maintaining sufficiently long prediction and control horizons, the blocking factor 2 was specified for both horizons.

The result of the simulation when the MPC controller is applied to the nonlinear plant (4.14) is shown in Figure 4.8. Reference trajectories specifying step sequences for the elevation and rotation angles respectively are applied to the system. As we can see, the controller manages to track the set points, while keeping the pitch angle within the specified limits.

Table 4.4 MPC controller parameters for the Helicopter simulations

Parameter	Value
H_p	30
H_w	1
H_u	10
I_p	Only every other sample was included in the optimization problem.
I_u	Every other control increment was assumed to be constant.
Q	diag(1,1)
R	diag(0.1,0.1)

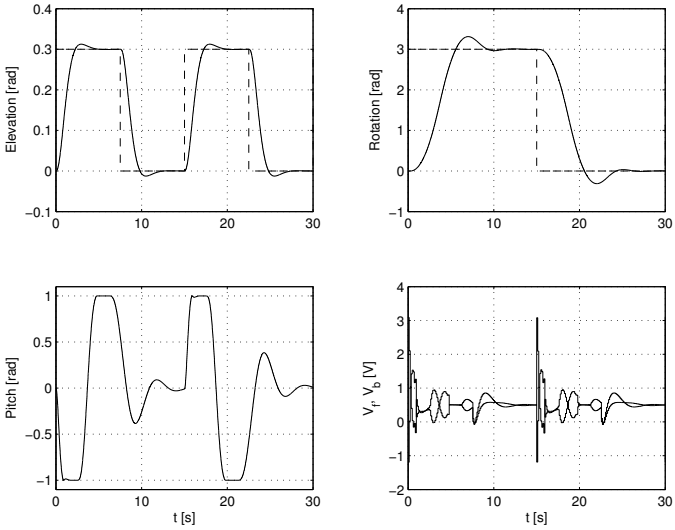

Figure 4.8 A simulation of the MPC controller applied to the nonlinear helicopter plant.

4.6 References

Åström, K. J. and B. Wittenmark (1990): *Computer Controlled Systems—Theory and Design*. Prentice-Hall, Englewood Cliffs, New Jersey.

Bazaraa, M. S., H. D. Sherall, and C. M. Shetty (1993): *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, Inc.

Bemporad, A., L. Chisci, and E. Mosca (1994): "On the stabilizing property of SIORHC." *Automatica*, **30:12**, pp. 2013–2015.

Fletcher, R. (1987): *Practical Methods of Optimization*. John Wily & Sons Ltd.

Maciejowski, J. M. (2002): *Predictive Control with Constraints*. Pearson Education.

Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert (2000): "Constrained model predictive control: Stability and optimality." *Automatica*, **36:6**, pp. 789–814.

Qin, S. J. and T. A. Badgwell (2003): "A survey of industrial model predictive control technology." *Control Engineering Practice*, **11**, pp. 733–764.

Scokaert, P. O. M., D. Q. Mayne, and J. B. Rawlings (1999): "Suboptimal model predictive control (feasibility implies stability)." *IEEE Transactions of Automatic Control*, **44:3**, pp. 648–654.

Wright, S. J. (1997): *Primal-Dual Interior-Point Methods*. SIAM.

5

Integral action - a disturbance observer approach

5.1 Introduction

Integral action is often needed to achieve robustness to modeling errors and disturbance attenuation. In addition, error free tracking of constant reference signals may be achieved by integral action. Integral action may be introduced in several ways. A common approach is to extend the state vector to include integrator states, $\dot{x}_i = r - y$. However, alternatives exist. In this chapter, we investigate the use of disturbance observers. In particular, this method is commonly suggested in MPC applications, [Maciejowski, 2002].

In this chapter we will show how assumptions on the disturbance model may be used to guarantee integral action in output feedback MIMO controllers. The case of $m \times m$ plants is a straight forward generalization of the SISO case. The main contribution of this chapter is the generalization to non-square plants, where the number of outputs exceeds the number of inputs.

Without lack of generality, we could assume that the measured output vector is partitioned as $y = \left[y_z^T y_a^T\right]^T$, where y_z represents the

controlled outputs and y_a are the additional measured outputs (if any). We will also assume that the number of inputs of the system equals the number of controlled outputs. The case when the number of inputs equals the number of outputs will be treated first. Thereafter, the controller will be generalized to handle the case when there are additional measured outputs.

5.2 Square plants

In this section, a controller with integral action for square plants will be developed. The term *square plant* refers to the fact that the transfer function matrix of a system with an equal number of inputs and outputs is square. We will assume that the plant model is given by

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{5.1}$$

where $x \in R^n$, $u \in R^m$ and $y \in R^p$. In this case, $y = y_z$ and y_a is not present. (A, B) is assumed to be controllable a pair and (A, C) is assumed to be an observable pair. A standard way of introducing integral action for such systems is described in [Åström, 2002]. In this approach, a constant load disturbance, d, acting on the plant input is assumed. Using an augmented system description, a composite model may be written as

$$\begin{aligned} \dot{x}_e &= \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} x_e + \begin{bmatrix} B \\ 0 \end{bmatrix} u = A_e x_e + B_e u \\ y &= \begin{bmatrix} C & 0 \end{bmatrix} x_e = C_e x_e \end{aligned} \tag{5.2}$$

where $d \in R^m$ represents the input disturbance and $x_e = \begin{bmatrix} x^T & d^T \end{bmatrix}^T$.

The main idea is to use an observer based on this extended system to estimate the input disturbance d, and to use the disturbance estimation in the control law.

Observability

Before proceeding, it should be verified that the extended system model is observable. The result is straight forward to derive using the PBH test, and is summarized in the following lemma.

LEMMA 5.1—OBSERVABILITY
The system (5.2) is observable if and only if the system (5.1) is observable and has no zeros at $s = 0$.
Proof:
Using the PBH test we obtain the rank condition

$$\text{rank} \begin{bmatrix} A - sI & B \\ 0 & -sI \\ C & 0 \end{bmatrix} = n + m$$

If $s \neq 0$ it is readily verified that the matrix has full rank if and only if

$$\text{rank} \begin{bmatrix} A - sI \\ C \end{bmatrix} = n.$$

Since (A,C) is assumed to be an observable pair this is always the case. For $s = 0$ we have that

$$\text{rank} \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} = n + m$$

That is, the system may not have any transmission zeros at s=0, in which case this matrix looses rank. □

REMARK 5.1
The condition that the plant may not have zeros at $s = 0$ is equivalent to that $G(0)$ must be invertible, where $G(s)$ is the transfer function of the plant. This condition is identical to that given for SISO systems in [Åström, 2002]. An interesting observation is that this condition appears as a condition for integral stabilizability (see [Campo and Morari, 1994]), where the objective is to stabilize a system using a controller containing integral action. The same condition, but for a more general case, is given in [Davison and Goldenberg, 1975]. □

Controller structure

In order to analyze the properties of the controller it is necessary to make some assumptions about the controller structure. We will assume that the control law is given by linear feedback, from the state and disturbance estimations. The observer is assumed to be given by

$$\dot{\hat{x}}_e = A_e\hat{x}_e + B_e u + K(y - C_e\hat{x}_e). \tag{5.3}$$

where A_e, B_e and C_e are defined according to the augmented system (5.2). Introducing the feedback law

$$u = -L_x\hat{x} - \hat{d} + L_r r \tag{5.4}$$

we obtain the following equations for the controller:

$$\begin{aligned}\begin{bmatrix} \dot{\hat{x}} \\ \dot{\hat{d}} \end{bmatrix} &= \begin{bmatrix} A - BL_x - K_xC & 0 \\ -K_dC & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix} + \\ &+ \begin{bmatrix} K_x \\ K_d \end{bmatrix} y + \begin{bmatrix} BL_r \\ 0 \end{bmatrix} r \\ u &= -L_x\hat{x} - \hat{d} + L_r r.\end{aligned} \tag{5.5}$$

The feedback and observer gains are assumed to be chosen so that $A - BL_x$ and $A_e - KC_e$ are stable matrices.

This choice of control law, u, has a strong intuitive appeal. It is clear that if $\hat{d}$ is an accurate estimation of d, the effect of the disturbance is canceled. Also, the choice of controller structure is not restrictive in the sense that any design method that yields stable $A - BL_x$ and $A_e - KC_e$ matrices may be used.

Before we proceed, we notice that in stationarity, the following identities hold

$$\begin{aligned}&(A - BL_x)\hat{x} + K_x(y - C\hat{x}) + BL_r r = 0, \quad K_d(y - C\hat{x}) = 0 \\ &\Rightarrow y = -C(A - BL_x)^{-1}BL_r r.\end{aligned}$$

If we choose the gain L_r such that

$$L_r = (C(-A + BL_x)^{-1}B)^{-1} \tag{5.6}$$

the controller will have a unique equilibrium for $y = r$. This follows from the fact that K_d is invertible (since $A_e - KC_e$ is invertible), which implies that $y - \hat{C}\hat{x} = 0$ in stationarity. Notice also that

$$\text{rank} \begin{bmatrix} A - BL_x & B \\ C & 0 \end{bmatrix} = n + m$$

for any stabilizing L_x. This follows from Lemma 1. But this is equivalent to the matrix $C(A - BL_x)^{-1}B$ having full rank. The inverse in expression (5.6) thus exists.

It is also clear that the controller has m eigenvalues that are zero, which implies that the controller will have integral action. We shall now investigate in detail this property of the controller.

Integral action

In this section an input - output representation of the controller is derived. The aim is to show that the controller contains integral action, which was one of the initial objectives of the design.

Using (5.5), the control law may be written as

$$\begin{aligned} U(s) = &(I - L_x\Phi(s)B)L_rR(s) - L_x\Phi(s)K_xY(s) + \\ &+ \frac{1}{s}K_d(I - C\Phi(s)K_x)(C\Psi(s)BL_rR(s) - Y(s)) \end{aligned} \tag{5.7}$$

where

$$\begin{aligned} \Psi(s) &= (sI - A + BL_x)^{-1} \\ \Phi(s) &= (sI - A + BL_x - K_xC)^{-1} \end{aligned} \tag{5.8}$$

We assume that the matrix $(-A + BL_x + K_xC) = \Phi(0)^{-1}$ has full rank. The first term of the controller expression represents feedback and feedforward terms with finite gain. From the assumptions above, it follows that the transfer function $\Psi(s)$ is stable.

The second term in the controller expression represents integral action acting on $r_f - y$, where r_f is the filtered reference signal. The stationary gain of the filter used to obtain r_f is $C\Psi(0)BL_r$, which by design is equal to identity. The integral gain is

$$K_i = K_d(I + C\Psi(0)K_x)^{-1} = K_d(I - C\Phi(0)K_x) \tag{5.9}$$

so K_i is bounded and has full rank.

Robustness and sensitivity

The properties of the controller may also be illustrated by using the concepts of robustness and sensitivity. In practice, there is always a mismatch between the true plant and the plant model. It is of course desirable that modeling error does not severely degrade the control performance. In particular, stability and steady state tracking properties should not be affected.

Let us assume that the true plant has the transfer function $P^0(s)$ and that the available model is given by $P(s)$. If we assume an output uncertainty model, we have that

$$P^0(s) = (I + \Delta_P(s))P(s).$$

In summary, the following relation holds:

$$\Delta_Y(s) = S^0(s)\Delta_P(s) \tag{5.10}$$

where $S^0(s)$ is the sensitivity function of the true system and $Y^0(s) = (I + \Delta_Y(s))Y(s)$, [Glad and Ljung, 2000]. For small $\Delta_P(s)$, $S^0(s)$ could usually be approximated by $S(s)$. The sensitivity function is readily identified as the transfer function from an output disturbance v to the measured output $y = Cx + v$. The main result is summarized in the following theorem.

THEOREM 5.1
Let the system (5.1) be controlled by (5.5) and let the sensitivity function of the closed loop system be $S(s)$. Then

$$S(0) = 0.$$

Proof:
Introduce the auxiliary variable v, representing the output disturbance:

$$y = Cx + v.$$

The result is proven by showing that y does not depend on v in stationarity. Using the controller expression (5.2), we obtain the equations

$$\dot{\hat{x}} = A\hat{x} + Bu + K_x(Cx + v - C\hat{x})$$
$$\dot{\hat{d}} = K_d(Cx + v - C\hat{x})$$

In stationarity we have that $\dot{x} = 0$, $\dot{\hat{x}} = 0$ and $\dot{\hat{d}} = 0$, yielding

$$Cx + v - C\hat{x} = 0 \tag{5.11}$$

since K_d is invertible. Using the expression (5.4) we obtain that

$$0 = A\hat{x}_s - BL_x\hat{x}_s + BL_r r_s = (A - BL_x)\hat{x}_s + BL_r r_s$$

where subscript s indicates that the relation holds for constant values of $\hat{x}$ and r. Since $A - BL_x$ is assumed to be a stable matrix, we have that

$$\hat{x}_s = (-A + BL_x)^{-1}BL_r r_s. \tag{5.12}$$

In particular, due to the choice of L_r, and the expressions (5.11) and (5.12) it is clear that

$$y_s = Cx_s + v_s = C\hat{x}_s = r_s.$$

Obviously, y is not dependent on v in stationary, and thus $S(0) = 0$. □

This result implies that the controller is robust in the sense that the steady state tracking properties are not affected by modeling errors.

Example

In Figure 5.1 the controller described in this section is applied to a simple second order system, given by

$$\begin{aligned}
\dot{x} &= \begin{bmatrix} a & -1 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ b \end{bmatrix} u \\
y_z &= \begin{bmatrix} 1 & 0 \end{bmatrix} x \\
y_a &= \begin{bmatrix} 0 & 1 \end{bmatrix} x.
\end{aligned}$$

For the true plant, the parameters are $a = -0.2$ and $b = -1.2$, whereas for the model used for control design, the parameters are $a = -0.5$ and $b = -1$. Notice that there is a mismatch between the the real plant and the model used for control design. The plot shows the response

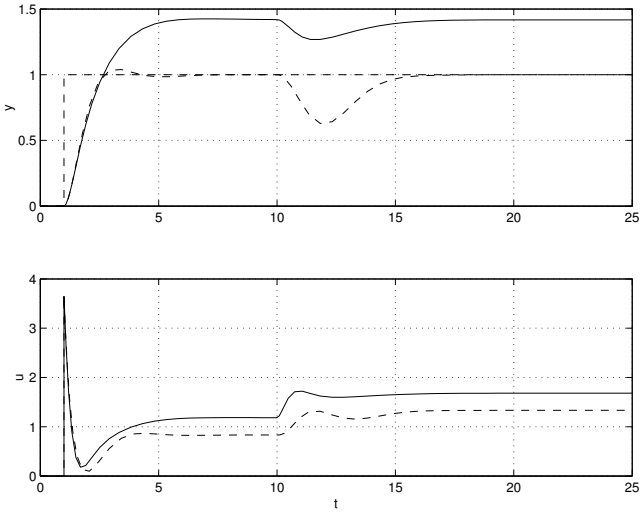

Figure 5.1 The dashed line represents the case when only the controlled output is measured, whereas the solid line represents the case when both states are measured.

for a reference step input at $t = 1$ s and an input step disturbance applied at $t = 10$ s. In the first simulation, represented by the dashed curves, only one measurement is used by the controller, and the scheme works as intended. However, if a controller using both available state measurements is employed, there will be a stationary error. In the first case $S(0) = 0$ as expected, but in the second case we have that

$$S(0) = \begin{bmatrix} 0.28 & 1.39 \\ 0.14 & 0.72 \end{bmatrix}$$

This apparent paradox will be discussed in detail in the next section.

5.3 Non-square plants

Now, suppose that apart from the controlled variables, which are assumed to be measured, there are additional measured signals. We may without lack of generality assume that y is partitioned as $y = \left[y_z^T \; y_a^T \right]^T$, where y_z are the controlled variables and y_a the additional measured outputs. It is reasonable to assume that the number of controlled variables equals the number of inputs. This gives that $y_z \in R^m$ and $y_a \in R^{p_a}$, $p_a \geq 1$. Notice that r specifies reference values for y_z, whereas there are no reference values for y_a. Certainly the additional measured variables can be used to estimate the state of the plant. However, it is easily demonstrated that the method presented above for introducing integral action into the controller may fail in such cases. If we assume that both states are available for measurements in the previous example, Figure 5.1 shows the result. Control performance is significantly degraded - even if more information is available. The controller has integral action, but this is obviously not enough to give the desired steady state tracking properties because of the bias in the additional measurement signals. In the presence of model - plant mismatch, as in the example above, the measured signals are not compatible with the dynamics of the model.

One way to deal with this situation is to determine the confidence associated with the two sets of measured signals y_z and y_a. Since the outputs y_z are controlled, a reasonable assumption is that we are confident in those, i.e. there is no bias in y_z. The fact that we in effect integrate the deviations of y_z from r supports this assumption.

To recover the properties of the controller described in the previous section, the following augmented model is introduced

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{v}_a \\ \dot{d} \end{bmatrix} &= \begin{bmatrix} A & 0 & B \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v_a \\ d \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u \\ y_z &= \begin{bmatrix} C_z & 0 & 0 \end{bmatrix} \begin{bmatrix} x^T & v_a^T & d^T \end{bmatrix}^T \\ y_a &= \begin{bmatrix} C_a & I & 0 \end{bmatrix} \begin{bmatrix} x^T & v_a^T & d^T \end{bmatrix}^T \end{aligned} \tag{5.13}$$

where $x \in R^n$, $u \in R^m$, $d \in R^m$, $y_z \in R^{p_z}$, $y_a \in R^{p_a}$ and $v_a \in R^{p_a}$. As

previously, an input disturbance d is assumed. Also, the disturbance model is augmented to include an output disturbance v_a acting on the additional measured outputs y_a. This assumption reflects the fact that confidence is put in the controlled outputs.

The results derived in the previous section will now be generalized to the modified assumptions stated above.

Observability

In order to use the model (5.13) for state and disturbance estimation, it must be verified that the model is indeed observable. The result is summarized in the following lemma.

LEMMA 5.2—OBSERVABILITY
The system (5.13) is observable if the system (5.1) is observable and has no zeros at $s = 0$.
Proof:
Using the PBH test we obtain the rank condition

$$\operatorname{rank}\begin{bmatrix} A - sI & 0 & 0 \\ 0 & -sI & 0 \\ 0 & 0 & -sI \\ C_z & 0 & 0 \\ C_a & I & 0 \end{bmatrix} = n + m + p_a$$

If $s \neq 0$ it is readily verified that the matrix has full rank if and only if

$$\operatorname{rank}\begin{bmatrix} A - sI \\ C_z \\ C_a \end{bmatrix} = n.$$

Since $(A, [C_z^T \; C_a^T]^T)$ is assumed to be an observable pair this is always the case. For $s = 0$ we have that

$$\operatorname{rank}\begin{bmatrix} A & 0 & B \\ C_z & 0 & 0 \\ C_a & I & 0 \end{bmatrix} = n + m + p_a$$

But this is equivalent to the matrix

$$\text{rank} \begin{bmatrix} A & B \\ C_z & 0 \end{bmatrix} = n + m.$$

having full rank. □

REMARK 5.2
This condition is very similar to the one given in the previous section, the plant may not have zeros at $s = 0$. Notice that the condition applies to the sub plant with input u and output y_z, and does not include the additional outputs y_a. No additional constraints are thus introduced in the presence of extra measurement signals. □

Controller structure

We will use the same controller structure as previously. By using the estimator given by equation (5.3) and the control law (5.4), the modified controller may now be written as

$$\begin{aligned} \begin{bmatrix} \dot{\hat{x}} \\ \dot{\hat{v}}_a \\ \dot{\hat{d}} \end{bmatrix} &= \begin{bmatrix} A - BL_x - K_x C & -K_{xa} & 0 \\ -K_v C & -K_{va} & 0 \\ -K_d C & -K_{da} & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{v}_a \\ \hat{d} \end{bmatrix} + \\ &+ \begin{bmatrix} K_{xz} & K_{xa} \\ K_{vz} & K_{va} \\ K_{dz} & K_{da} \end{bmatrix} \begin{bmatrix} y_z \\ y_a \end{bmatrix} + \begin{bmatrix} BL_r \\ 0 \\ 0 \end{bmatrix} r \\ u &= -L_x \hat{x} - \hat{d} + L_r r. \end{aligned} \tag{5.14}$$

As previously, we assume that $A - BL_x$ and $A_e - KC_e$ are stable matrices. L_r is chosen as

$$L_r = (C_z(-A + BL_x)^{-1}B)^{-1}$$

The controller then has the unique equilibrium $r = y_z$. The same arguments given previously apply also in this case.

We notice that the order of the controller is now increased compared to the design in the previous section. One extra state for each additional measured signal is introduced. We also notice that the controller has m eigenvalues equal to zero, which implies the presence of integrators.

Integral action

As we have seen, the presence of integrators in the controller alone is not sufficient to ensure zero steady state tracking error. In this section we will establish that the controller gives integral action acting on $e_f = r_f - y_{zf}$, where the filters used to obtain r_f and y_{zf} have special properties.

Introducing

$$\Phi_e^{-1}(s) = \begin{bmatrix} sI - A + BL_x + K_xC & K_{xa} \\ K_vC & sI + K_{va} \end{bmatrix}$$
$$C_t = \begin{bmatrix} C_z & 0 \\ C_a & I \end{bmatrix}$$

The following expressions are obtained:

$$\begin{bmatrix} \hat{X}(s) \\ \hat{V}_a(s) \end{bmatrix} = \Phi_e(s) \left(\begin{bmatrix} K_x \\ K_v \end{bmatrix} Y(s) + \begin{bmatrix} BL_r \\ 0 \end{bmatrix} R(s) \right)$$
$$\hat{D}(s) = \frac{1}{s} K_d \left(Y(s) - C_t \begin{bmatrix} \hat{X}(s) \\ \hat{V}_a(s) \end{bmatrix} \right)$$

The controller may then be written as

$$\begin{aligned}
U(s) &= -L_x\hat{X}(s) - \hat{D}(s) + L_r R(s) \\
&= \left(I - \left[\begin{array}{cc} L_x & 0 \end{array} \right] \Phi_e(s) \left[\begin{array}{c} B \\ 0 \end{array} \right] \right) L_r R(s) \\
&- \left[\begin{array}{cc} L_x & 0 \end{array} \right] \Phi_e(s) \left[\begin{array}{c} K_x \\ K_v \end{array} \right] Y(s) \\
&+ \frac{1}{s} K_d \left(I - C_t \Phi_e(0) \left[\begin{array}{c} K_x \\ K_v \end{array} \right] \right) (C\Psi(s)BL_r R(s) - Y(s)) \\
&= M_0(s)R(s) - \left[\begin{array}{cc} M_1(s) & M_2(s) \end{array} \right] \left[\begin{array}{c} Y_z(s) \\ Y_a(s) \end{array} \right] + \\
&+ \frac{1}{s} K_d \left[\begin{array}{cc} M_3(s) & M_4(s) \end{array} \right] \left[\begin{array}{c} C_z\Psi(s)BL_r R(s) - Y_z(s) \\ C_a\Psi(s)BL_r R(s) - Y_a(s) \end{array} \right]
\end{aligned}$$

The definitions of $\Psi(s)$ and $\Phi(s)$ are given by (5.8). We also make the assumption that the matrix $\Phi_e(0)$ has full rank and that $\Psi(0)$ is stable. It then follows that the matrices $M_0(0)$, $M_1(0)$ and $M_2(0)$ represent finite feedback and feedforward gains.

Now, the aim of the controller is somewhat more elaborate than in the previous case; integral action is desired to act on $r_f - y_{zf}$. In order for the proposed controller to achieve this, it must be verified that $M_4(0) = 0$ and that $M_3(0)$, the integral gain, has full rank. We start by verifying that $M_4(0) = 0$. By direct calculation we obtain:

$$\begin{aligned}
M_4(0) &= \left(I - C_t \Phi_e(0) \left[\begin{array}{c} K_x \\ K_v \end{array} \right] \right) \left[\begin{array}{c} 0 \\ I \end{array} \right] \\
&= \left(\left[\begin{array}{c} 0 \\ I \end{array} \right] - C_t \left[\begin{array}{cc} -A + BL_x + K_x C & K_{xa} \\ -K_v C & K_{va} \end{array} \right]^{-1} \left[\begin{array}{c} K_{xa} \\ K_{va} \end{array} \right] \right) \\
&= \left(\left[\begin{array}{c} 0 \\ I \end{array} \right] - C_t \left[\begin{array}{c} 0 \\ I \end{array} \right] \right) = 0
\end{aligned}$$

where we have used the identity

$$\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}^{-1} \begin{bmatrix} X_{12} \\ X_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}.$$

As for $K_i = K_d M_3(0)$ we have

$$\begin{aligned} K_i &= K_d \left(I - C_t \Phi_e(0) \begin{bmatrix} K_x \\ K_v \end{bmatrix} \right) \begin{bmatrix} I \\ 0 \end{bmatrix} \\ &= \left(K_{dz} - \begin{bmatrix} K_d C & K_{da} \end{bmatrix} \Phi_e(0) \begin{bmatrix} K_{xz} \\ K_{vz} \end{bmatrix} \right). \end{aligned} \tag{5.15}$$

But this expression is recognized as the Schur-complement of $\Phi_e^{-1}(0)$ with respect to the matrix

$$\begin{bmatrix} -A + BL_x + K_x C & K_{xa} & K_{xz} \\ K_v C & K_{va} & K_{vz} \\ K_d C & K_{da} & K_{dz} \end{bmatrix}. \tag{5.16}$$

It is clear that the matrix (5.15) having full rank is equivalent to (5.16) having full rank. Now, let us rearrange the elements of this matrix by permuting the rows and columns, an operation that preserves the rank of the matrix

$$\begin{bmatrix} -A + BL_x + K_x C & K_{xz} & K_{xa} \\ K_d C & K_{dz} & K_{da} \\ K_v C & K_{vz} & K_{va} \end{bmatrix}. \tag{5.17}$$

If we take the Schur-complement of

$$\begin{bmatrix} K_{dz} & K_{da} \\ K_{vz} & K_{va} \end{bmatrix}$$

with respect to (5.17) we obtain

$$\Phi(0) - \begin{bmatrix} K_{xz} & K_{xa} \end{bmatrix} \begin{bmatrix} K_d \\ K_v \end{bmatrix}^{-1} \begin{bmatrix} K_d \\ K_v \end{bmatrix} C = \Psi(0).$$

The invertability of $[K_d^T \; K_v^T]^T$ follows from the matrix $A_e - KC_e$ having full rank. Further, $\Psi(0)$ is assumed to have full rank by design, and we can conclude that the matrix (5.15) has indeed full rank.

We have now shown that the integral action property of the controller is recovered. As in the previous case, we have integral action acting on

$$C_z\Psi(s)BL_rR(s) - Y_z(s).$$

Due to the choice of L_r, this gives integral action acting on $r - y_z$ in stationarity. The integral gain, K_i, is given by (5.15).

Robustness and sensitivity

Let us now investigate the robustness properties of the controller. The critical feature of the controller is that $r = y_z$ also in the presence of modeling errors. To establish this, we use the relation

$$\Delta_Y(s) = S(s)\Delta_P(s).$$

However, now we are concerned with the controlled outputs y_z, and not all of the measured outputs as previously. The additional measured outputs are not controlled, and are likely to be influenced by disturbances as well as modeling errors. For this reason we use

$$\Delta_{Y_z}(s) = S_z(s)\Delta_P(s).$$

That is, only deviations of the controlled outputs from the nominal case $r = y_z$ are considered. In this case $S_z(s)$ consists of the first m rows of the sensitivity function matrix $S(s)$. It follows that $S_z(s)$ is the transfer function from v to y_z. In particular we would like to show that $S_z(0)$ is zero, which would imply that the controller is robust to modeling errors in steady state. We have the following result:

THEOREM 5.2
Let the system (5.1) be controlled by (5.14), and let the sensitivity function of the closed loop system be $S(s) = \begin{bmatrix} S_z(s)^T & S_a(s)^T \end{bmatrix}^T$. Then

$$S_z(0) = 0.$$

Proof:
Introduce the auxiliary variables v_a and v_z, representing the output disturbances:

$$y_z = C_z x + v_z, \quad y_a = C_a x + v_a$$

The same technique as in theorem (5.1) is used; it will be shown that y_z does not depend on v_z or v_a in stationarity. The equations for the controller may be written as

$$\begin{aligned}
\dot{\hat{x}} &= A\hat{x} + Bu + K_{xz}(C_z x + v_z - C_z\hat{x}) + \\
&\quad + K_{xa}(C_a x + v_a - C_a\hat{x} - \hat{v}_a) \\
\dot{\hat{v}}_a &= K_{vz}(C_z x + v_z - C_z\hat{x}) + K_{va}(C_a x + v_a - C_a\hat{x} - \hat{v}_a) \\
\dot{\hat{d}} &= K_{dz}(C_z x + v_z - C_z\hat{x}) + K_{da}(C_a x + v_a - C_a\hat{x} - \hat{v}_a)
\end{aligned}$$

In stationarity we have that $\dot{x} = 0$, $\dot{\hat{x}} = 0$ and $\dot{\hat{d}} = 0$ which yields

$$C_z x + v_z - C_z\hat{x} = 0, \quad C_a x + v_a - C_a\hat{x} - \hat{v}_a = 0 \tag{5.18}$$

since $\left[K_d^T K_v^T\right]^T$ is invertible. Using the expression (5.4) we obtain that

$$0 = A\hat{x}_s - BL_x\hat{x}_s + BL_r r_s = (A - BL_x)\hat{x}_s + BL_r r_s$$

where subscript s indicates that this relation holds in stationarity, as before. Since $A - BL_x$ is assumed to be a stable matrix, we have that

$$\hat{x}_s = (-A + BL_x)^{-1} BL_r r_s. \tag{5.19}$$

In particular, due to the choice of L_r, and the expressions (5.18) and (5.19) it is clear that

$$y_{zs} = C_z x_s + v_{zs} = C_z\hat{x}_s = r_s.$$

Obviously, y_z is not dependent on v in stationary, which implies that $S_z(0) = 0$. □

This result proves that the robustness property described in the previous section is recovered.

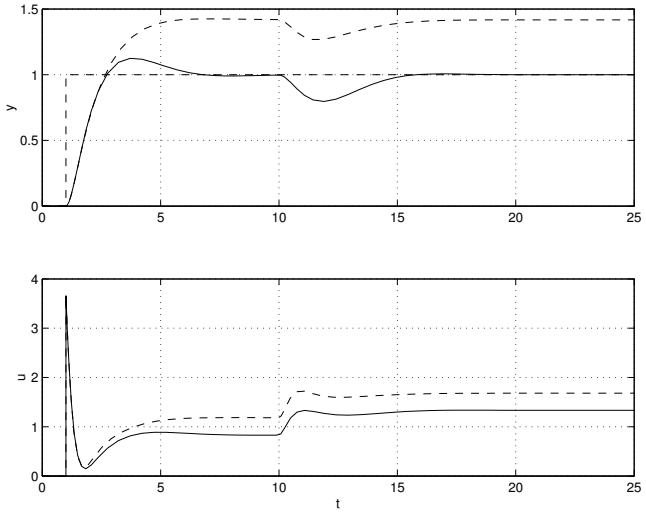

Figure 5.2 Responses of the controllers derived in Section 2 (dashed) and section 3 (solid).

Example

In Figure 5.2, the improved controller structure is simulated, using the same system as in the previous section. The system has one input, and two measured outputs, of which one is controlled. There is also a modeling error present. As we can see, the improved controller performs well, and achieves zero steady state tracking error. In addition,

$$S(0) = \begin{bmatrix} 0.00 & 0.00 \\ 0.20 & 1.00 \end{bmatrix}$$

and in particular $S_z(0) = 0$.

5.4 Conclusions

In this chapter the design of centralized controllers including integral action has been discussed. The proposed solution explores disturbance observers of a certain structure. The case of square plants, including SISO plants, has been generalized to the case of plants with additional measured signals.

During the course of this work, articles treating similar problems have been published, see [Pannocchia and Rawlings, 2003], which we became aware of after the submission of [Åkesson and Hagander, 2003]. Notably, [Pannocchia and Rawlings, 2003] reports strong results for how disturbance observers achieve offset-free control for quite general disturbance models in the context of Model Predictive Control. Some of the results presented in this chapter follow as a specialization of those given in [Pannocchia and Rawlings, 2003]. In the results presented here, the emphasis on integral action is stronger.

5.5 References

Åkesson, J. and P. Hagander (2003): "Integral action - a disturbance observer approach." In *Proceedings of European Control Conference*.

Åström, K. J. (2002): "Model uncertainty and feedback." In Albertos and Sala, Eds., *Iterative Identification and Control*. Springer Verlag.

Campo, P. J. and M. Morari (1994): "Achievable closed-loop properties of systems under decentralized control: Conditions involving the steady-state gain." *IEEE Transactions on Automatic Control*, **39:5**, pp. 932–942.

Davison, E. J. and A. Goldenberg (1975): "Robust control of a general servomechanism problem: The servo compensator." *Automatica*, **11**, pp. 461–471.

Glad, T. and L. Ljung (2000): *Control Theory: Multivariable and Nonlinear Methods*. Taylor & Francis.

Maciejowski, J. M. (2002): *Predictive Control with Constraints*. Pearson Education.

Pannocchia, G. and J. B. Rawlings (2003): "Disturbance models for offset-free model predictive control." *AIChE Journal*, **49:2**, pp. 426–437.

6

Compensation of computational delay in MPC

6.1 Introduction

Model predictive control (MPC), see, e.g., [Garcia *et al.*, 1989; Richalet, 1993; Qin and Badgwell, 2003], has been widely accepted industrially during recent years, mainly because of its ability to handle constraints explicitly and the natural way in which it can be applied to multivariable processes. The computational requirements of MPC, where typically a quadratic optimization problem is solved on-line in every sample, have previously prohibited its application in areas where fast sampling is required. Therefore MPC has traditionally only been applied to slow processes, mainly in the chemical industry. However, the advent of faster computers and the development of more efficient optimization algorithms, see, e.g., [Cannon *et al.*, 2001], has led to applications of MPC also to processes governed by faster dynamics. Some recent examples include [Dunbar *et al.*, 2002; Dunbar and Murray, 2002].

From a real-time implementation perspective, however, the execution time characteristics associated with MPC tasks still poses many

interesting problems. Execution time measurements show that the computation time of an MPC controller varies significantly from sample to sample. The variations are due to, e.g., reference changes or external disturbances. To cope with this, an increased level of flexibility is required in the real-time implementation.

The highly varying execution times introduce delays which are hard to compensate for. The longer time spent on optimization the larger the latency, i.e., the delay between the sampling and the control signal generation. The latency has the same effect as an input time delay, and if it is not properly compensated for it will affect the control performance negatively. However, since the optimization algorithms used in MPC are iterative in nature, and, typically, reduce the quadratic cost for each iteration step, it is possible to abort the optimization before it has reached the optimum, and still fulfill the stability conditions.

Stability of model predictive control algorithms has been the topic of much research in the field. For linear systems, the stability issue is well understood, and also for nonlinear systems there are results ensuring stability under mild conditions. For an excellent review of the topic see [Mayne *et al.*, 2000]. In summary, there are two main ingredients in most stabilizing MPC schemes; terminal penalty and terminal constraint. These two tools has been used separately or in combination to prove stability for many existing MPC algorithms. It is also well known that feasibility, rather than optimality, is sufficient to guarantee stability, see for example [Scokaert *et al.*, 1999]

In this chapter, the trade-off between computational delay and optimization is quantified by the introduction of a delay-dependent cost index, which constitutes the main contribution of this chapter. (A preliminary simulation study was presented in [Henriksson *et al.*, 2002a].) The index is based on a parameterization of the cost function in the MPC formulation. The key observation is that the computational delay may significantly degrade control performance, and premature termination of the optimization algorithm may be advantageous over actually finding the optimum. In this chapter it is shown how a novel termination criterion can be employed to improve control performance.

Another contribution of the chapter, is the application of the termination criterion and cost index in a real-time scheduling context. Traditional real-time scheduling of control tasks is based on task models assuming constant, known, worst-case execution times for all tasks.

However, the large variations in execution time for MPC tasks make a real-time design based on worst-case bounds very conservative and gives an unnecessary long sampling period. Hence, more flexible implementation schemes than traditional fixed-priority or deadline-based scheduling are needed.

In feedback scheduling, [Årzén *et al.*, 2000; Cervin *et al.*, 2002], the CPU time is viewed as a resource that is distributed dynamically between the different tasks based on, e.g., feedback from CPU usage and quality-of-service (QoS). For controller tasks the quality-of-service corresponds to the control performance. Another approach that can be tailored towards MPC is scheduling of imprecise computations [Liu *et al.*, 1991; Liu *et al.*, 1994]. Here, each task is divided in a mandatory part (finding a feasible solution) and an optional part (QP optimization), which are scheduled separately. The dynamic scheduling strategy proposed in this chapter schedules the optional parts of the MPC tasks using the cost indices as dynamic task priorities.

The rest of the chapter is organized as follows. The MPC formulation is given in Section 2. Section 3 describes the delay-dependent cost index which is used to dynamically trade-off computational delay and optimization. Section 4 describes a dynamic scheduling scheme for scheduling of multiple MPC controllers. Section 5 contains a case study, where the proposed strategy is compared to conventional scheduling techniques. Finally the conclusions are given in Section 6.

6.2 MPC formulation

The MPC formulation used in this chapter is equivalent to the one given in Chapter 4, but with minor changes in notation. We assume a discrete linear process model on the form

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ y(k) &= C_y x(k) \\ z(k) &= C_z x(k) + D_z u(k) \end{aligned} \tag{6.1}$$

where $y(k)$ is the measured output, $z(k)$ the controlled output, $x(k)$ the state vector, and $u(k)$ the input vector. The function to minimize at time k is

$$J(k, \Delta\mathcal{U}, x(k)) = \sum_{i=1}^{H_p} \|\hat{z}(k+i|k) - r(k+i)\|_Q^2 + \sum_{i=0}^{H_u-1} \|\Delta\hat{u}(k+i|k)\|_R^2 \tag{6.2}$$

where $\hat{z}$ is the predicted controlled output, r is the current set-point, $\hat{u}$ is the predicted control signal, H_p is the prediction horizon, H_u is the control horizon, $Q \geq 0$ and $R > 0$ are weighting matrices, and $\Delta u(k) = u(k) - u(k-1)$. It is assumed that $H_u < H_p$ and that $\hat{u}(k+i) = \hat{u}(k+H_u-1)$ for $i \geq H_u$.

Introducing $\Delta\mathcal{U} = \left(\Delta\hat{u}(k)^T \ldots \Delta\hat{u}(k+H_u-1)^T\right)^T$ and $\mathcal{U}$ and $\mathcal{Z}$ equivalently, the state and control signal constraints may be expressed as

$$W\Delta\mathcal{U} \leq w \quad F\mathcal{U} \leq f \quad G\mathcal{Z} \leq g \tag{6.3}$$

This formulation leads to a convex linear-inequality constrained quadratic programming problem (LICQP) to be solved at each sample. The problem can be written on matrix form as

$$\min_{\theta} \; V(k) = \theta^T \mathcal{H}\, \theta - \theta^T \mathcal{G} + \mathcal{C} \quad \text{s.t.} \quad \Omega\theta \leq \omega. \tag{6.4}$$

where $\theta = \Delta\mathcal{U}$ and the matrices $\mathcal{H}$, $\mathcal{G}$, $\mathcal{C}$, Ω, and ω depend on the process model and the constraints, see Chapter 4. Only the first element of $\Delta\mathcal{U}$ is applied to the process. The optimization is then repeated in the next sample in accordance with the receding horizon principle.

Feasibility and optimality

Recall the MPC stability Theorem 4.1. The important feature in the proof of this theorem that will be explored in this chapter is embedded in equation (4.11). In order for the stability proof to work, we must ensure that $V(k)$ is decreasing, which, however, does not require optimality of the control sequence $\Delta\mathcal{U}$. See e.g. [Scokaert *et al.*, 1999] for a thorough discussion on this topic. Rather, having fulfilled the stability condition $V(k+1) < V(k)$, the optimization may be aborted prematurely without losing stability. This fact will be explored in Section 6.3 where a novel termination criterion is presented. In the simulations, the terminal constraint $\hat{u}(k+H_u)$=0 has been relaxed, in order to

increase the feasibility region of the controller. To remove this complication, the control signal, u, rather than the control increments, Δu, could be included in the cost function. Notice, however, that the important feature of the stability proof that will be explored is the inequality (4.11) and that other, more sophisticated, stabilizing techniques may well be used instead.

QP solver

There are two major families of algorithms for solving LICQPs; *active set methods* [Fletcher, 1991] and *primal-dual interior point methods*, e.g., Mehrotra's predictor-corrector algorithm, [Wright, 1997]. Both types of methods have advantages and disadvantages when applied to MPC, as noted in [Bartlett *et al.*, 2000] and [Maciejowski, 2002]. Rather, the key to efficient algorithms lies in exploration of the structure of the optimization problem generated by the MPC algorithm.

Recent research has also suggested interesting, and fundamentally different MPC algorithms, see e.g. [Kouvaritakis *et al.*, 2002] and [Bemporad *et al.*, 2002], known as explicit MPC. Here, the optimization problem is solved *off-line* for all $x(k)$, resulting in an explicit piecewise affine control law. At run-time, the problem is then transformed into finding the appropriate (linear) control law, based on the current state estimation. However, when the complexity of the problem increases, so does the complexity of the problem of finding the appropriate control law at each sample.

In this chapter we will use an MPC algorithm based on the on-line solution of a QP problem. The value of the cost function at each iteration in the optimization algorithm is of importance. Specifically, if the decay of the cost function is slow, it may be a good choice to terminate the optimization algorithm, and use the sub-optimal solution, rather than allowing the algorithm to continue and thereby introduce additional delay in the control loop. In the scheduling case, long execution times will also affect the performance of other control loops.

From this point of view, there is a fundamental difference between an active set algorithm and a typical primal-dual interior point method. The active set algorithm explicitly strives to decrease the cost function in each iteration, whereas a primal-dual interior point algorithm rather tries to find, simultaneously, a point in the primal-dual space that fulfills the Karush-Kuhn-Tucker conditions. In the latter case, the

duality gap is explicitly minimized in each iteration, rather than the cost function. With these arguments, and from our experience using both types of algorithms, we conclude that an active set algorithm is preferable for our application.

6.3 Termination criterion

We will now introduce a delay-dependent cost index, which will be used on-line to determine when to abort the MPC optimization and output the control signal. This cost index is based on a parameterization of the cost function (6.2).

Assuming a constant time delay, $\tau < h$, the process model (6.1) can be extended (see, e.g., [Åström and Wittenmark, 1997]) to

$$\begin{aligned} \tilde{x}(k+1) &= \tilde{\Phi}\tilde{x}(k) + \tilde{\Gamma}u(k) \\ y(k) &= \tilde{C}_y\tilde{x}(k) \\ z(k) &= \tilde{C}_z\tilde{x}(k) + D_z u(k) \end{aligned} \tag{6.5}$$

where

$$\begin{aligned} \tilde{x}(k) &= \begin{pmatrix} x(k) & u(k-1) \end{pmatrix}^T \\ \tilde{\Phi} &= \begin{pmatrix} \Phi & \Gamma_1(\tau) \\ 0 & 0 \end{pmatrix}, \quad \tilde{\Gamma} = \begin{pmatrix} \Gamma_0(\tau) \\ 1 \end{pmatrix} \\ \tilde{C}_y &= \begin{pmatrix} C_y & 0 \end{pmatrix}, \qquad \tilde{C}_z = \begin{pmatrix} C_z & 0 \end{pmatrix} \end{aligned}$$

The matrices $\mathcal{H}$, $\mathcal{G}$, $\mathcal{C}$, Ω, and ω in (6.4) all depend on the system matrices and thus on the delay. Ideally, these matrices should be updated based on the current computational delay. However, on-line recomputation of these matrices is too time-consuming.

However, using the representation (6.5) it is possible to evaluate the cost function (6.2) assuming a constant computational delay, τ, over the prediction horizon. The assumption that the delay is constant over the prediction horizon is in line with the assumptions commonly made in the standard MPC formulation, e.g., that reference values will be constant over the prediction horizon. Thus, for each iterate, $\Delta\mathcal{U}_i$, produced by the optimization algorithm, we compute

$$J_d(\Delta\mathcal{U}_i, \tau) = \Delta\mathcal{U}_i^T \mathcal{H}(\tau)\Delta\mathcal{U}_i - \Delta\mathcal{U}_i^T \mathcal{G}(\tau) + \mathcal{C}(\tau) \tag{6.6}$$

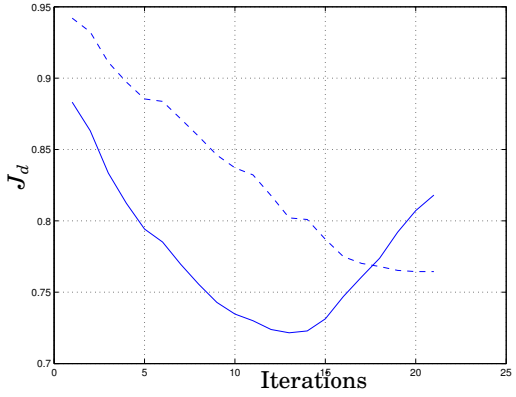

Figure 6.1 The solid curve shows the delay-dependent cost index J_d, and the dashed curve shows the original cost function used in the QP-algorithm.

This cost index penalizes not only deviations from the desired reference trajectory, but also performance degradation due to computational delay. There are two major factors that affect the evolution of J_d. On one hand, an increasing τ, corresponding to an increased computational delay, may degrade control performance and cause J_d to increase. On the other hand, J_d will decrease for successive Δu_i:s since the quality of the control signal has improved. Figure 6.1 shows the evolution of J_d during an optimization run. In the beginning of the optimization, J_d is decreasing rapidly, but then increases due to computational delay. In this particular example, the delayed control trajectory seems to achieve a lower cost than the original. This situation may occur since the cost functions are evaluated for non-optimal control sequences, except for the last iteration. Notice, however, that for the optimal solution, J_d is higher than the original cost. The proposed termination strategy is then to compare the value of $J_d(\Delta u_i, \tau_i)$ with the cost index computed after the previous iteration, i.e., $J_d(\Delta u_{i-1}, \tau_{i-1})$. If the cost index has decreased since the last iteration, we conclude that we gained more by optimization than we lost by the additional delay. On the other hand, if the cost index has increased, the optimization is aborted. Notice that the matrices needed to evaluate J_d should be calculated off-line.

In the MPC formulation used in this chapter we will assume a process model without delay. Another possible approach would be to include a fixed-sample delay in the process description. However, since the computational delay will be highly varying, compensating for the maximum delay may become very pessimistic and lead to decreased obtainable performance. We will also assume that the control signal is actuated as soon as the optimization algorithm terminates, not to induce any unnecessary delay.

6.4 Dynamic real-time scheduling of MPCs

The cost index and termination criterion described above, will now be applied in a dynamic real-time scheduling context. Controller tasks are often implemented as tasks on a microprocessor using a real-time kernel or a real-time operating system (RTOS). The real-time kernel or OS uses multiprogramming to multiplex the execution of the tasks on the CPU. To guarantee that the time requirements and time constraints of the individual tasks are all met, it is necessary to schedule the usage of the CPU time.

During the last two decades, scheduling of CPU time has been a very active research area and a number of different scheduling models and methods have been developed [Buttazzo, 1997; Liu, 2000]. The most common, and simplest, model assumes that the tasks are periodic, or can be transformed to periodic tasks, with a fixed period, T_i, a known worst-case execution time, C_i, and a *hard deadline*, D_i. The latter implies that it is imperative that the tasks always meet their deadlines, i.e., that the actual execution time in each sample is always less or equal to the deadline.

MPC tasks, however, do not fit this traditional task model very well, mainly because their highly varying execution times. On the other hand, MPC offers two features that distinguish it from ordinary control algorithms from a real-time scheduling perspective. Firstly, as we have seen in the previous sections, it is possible to abort the computation and thereby reduce the execution time. Secondly, the cost index contains relevant information about the state of the controlled process. Thus, the cost index can be viewed as a real-world quality-of-service measure for the controller, and be used as a dynamic task priority by

the scheduler. This also enables a tight and natural connection between the control and the real-time scheduling.

The MPC algorithm can be divided into two parts. The first part consists of finding a starting point fulfilling the constraints in the MPC formulation (constraints on the controlled and control variables and the terminal equality constraint) and to iterate the QP optimization algorithm until the stability condition of Theorem 1 is fulfilled. The second part consists of the additional QP iterations that further reduce the value of the cost function. Based on this insight, the MPC algorithm can be cast into the framework of scheduling of imprecise computations [Liu *et al.*, 1991; Liu *et al.*, 1994]. Using their terminology, the first part of the control algorithm will be called the mandatory sub-task, and the second part will be called the optional sub-task. The mandatory sub-tasks will be given the highest priority, whereas the optional sub-tasks will be scheduled based on the values of the MPC cost indices. Combining this strategy with the trade-off between optimization and computational delay, we get the following dynamic scheduling scheme of the optional sub-tasks

```
now = currentTime;
for (each optional MPC sub-task) {
   if (J_d(now) > prev J_d) {
      abort optimization;
      actuate plant;
   }
}
determine MPC task i with highest J_d;
schedule MPC task i for one iteration;
if (optimum__reached_i) {
   actuate plant;
}
```

It should be noted that comparing cost indices directly may not be appropriate when the controllers have different sampling intervals, prediction horizons, weighting matrices, etc. In those cases, it would be necessary to scale the cost indices to obtain a fair comparison. The scheduling could also use feedback from the derivatives of the cost functions, as well as the relative deadlines of the different controllers.

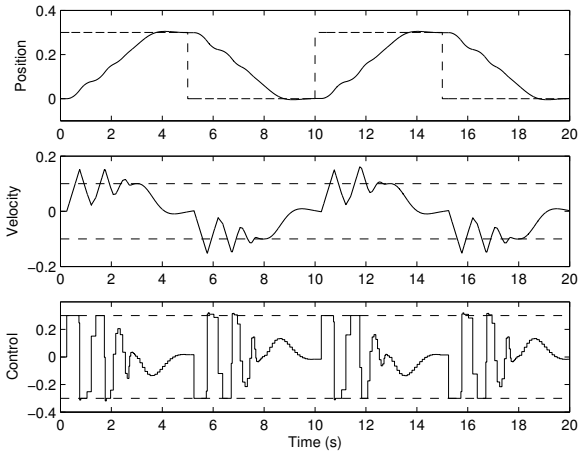

Figure 6.2 Control performance when the optimization algorithm is allowed to finish in every sample. The bad performance is a result of considerable delay and jitter induced by the large variations in execution time. During the transients the long execution times causes the control task to miss its next invocation, inducing sampling jitter. The dashed lines in the velocity and control signal plots show the constraints used in the MPC formulation.

6.5 Case study

The proposed termination criterion and dynamic real-time scheduling strategy have been evaluated in simulation using a second order system, a double-integrator:

$$\begin{aligned} \dot{x} &= \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} x \end{aligned} \tag{6.7}$$

The plant was discretized using the sampling interval $h = 0.1$ s. In the simulations, $z = x_1$ was set to be the controlled state and the constraints $|u| \leq 0.3$ and $|x_2| \leq 0.1$ were enforced.

The MPC controller used in the simulations was implemented as described in Section 6.2, with prediction horizons $H_p = 50$ and $H_u = 20$

and weighting matrices $Q = 1$ and $R = 0.1$.

Simulation environment and implementation

Real-time MPC control of the double-integrator process was simulated using the TRUETIME toolbox [Henriksson *et al.*, 2002b]. Using TRUETIME it is possible to perform detailed co-simulation of the MPC control task executing in a real-time kernel and the continuous dynamics of the controlled process. Using the toolbox it is easy to simulate different implementation and scheduling strategies and evaluate them from a control performance perspective.

In the standard implementation, the MPC task is released periodically and new instances may not start to execute until the previous instance has completed. This implementation will allow for task overruns without aborting the ongoing computations. The control signal is actuated as soon as the task has completed.

In the dynamic scheduling scheme, the MPC task is divided into a mandatory and an optional part as described in Section 6.4. The mandatory part is scheduled with a distinct high priority, whereas the priority of the optional part is changed depending on the current value of the cost index compared to other running MPC tasks.

Simulation of one MPC controller

The first simulations consider the case of a single MPC task implemented according to the standard task model described in the previous section. Figure 6.2 shows the result of a simulation where the optimization is allowed to finish in each sample. Delay and jitter induced by the large variations in execution time compromise the optimal control performance. The constraints are shown by the dashed lines in the velocity and control signal plots. As seen in the plots the constraints are violated at some points. This is due to the computational delay, which is not accounted for in the MPC formulation.

Figure 6.3 shows a simulation, utilizing the termination criterion proposed in Section 6.3. The cost index (6.6) is evaluated after each iteration, and if it has increased since the last iteration, the optimization is aborted and the current control signal is actuated. As can be seen from the simulations, the control performance has increased significantly.

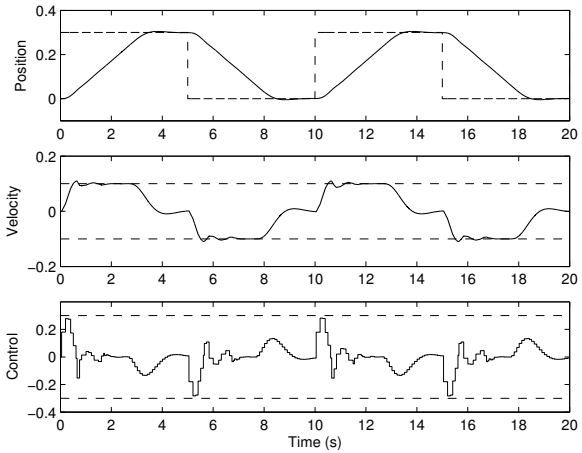

Figure 6.3 Control performance obtained using the proposed sub-optimal approach where the QP optimization may be aborted according to the termination criterion described in Section 6.3. The performance is increased substantially compared to Figure 6.2.

Optimization	Full	Sub-optimal
Total time [s]	0.1055	0.0692
Mandatory time [s]	0.0302	0.0297
Number of iterations	8.87	5.66
Number of necessary iterations	1.70	1.89

Table 6.1 Average values per sample for a simulation.

Figure 6.4 shows a comparison of the number of iterations needed for full optimization (top) and the number of iterations after which the optimization was aborted due to an increasing value of J_d (bottom). The execution time of each iteration was 10 ms. Average values for computation times and the number of iterations in the QP optimization algorithm in each sample is summarized in Table 6.1. The number of necessary iterations denotes the number of QP iterations

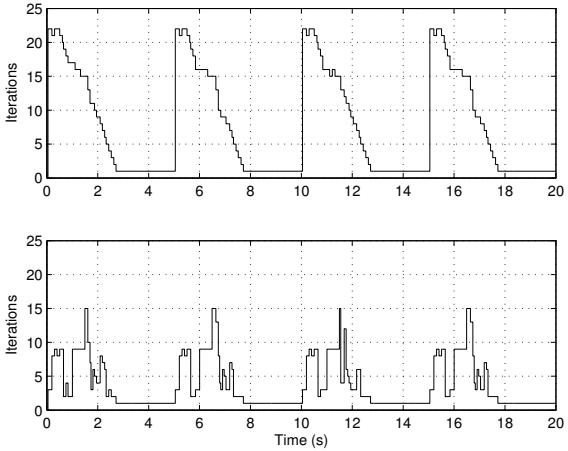

Figure 6.4 Number of iterations for the QP solver. The top plot shows the number of iterations to find the optimum. The bottom plot shows the number of iterations after which the optimization is terminated and the sub-optimal control is actuated.

needed to fulfill the stability condition. It can be seen that the total execution time of the MPC task is reduced by 35 percent by using the proposed termination criterion. The execution time for the mandatory part of the algorithm is roughly constant for both approaches. In the full optimization case, the execution time will exceed the 100 ms sampling period during the transients, causing the control task to miss deadlines and experience sampling jitter.

To quantify the simulation results, the *performance loss*

$$J = \int_0^{T_{sim}} \left(\|z(t) - r(t)\|_Q^2 + \|\Delta u(t)\|_R^2 \right) dt \tag{6.8}$$

was recorded in both cases. The weighting matrices, Q and R, were the same as those used in the MPC formulation. The performance loss was scaled with the loss for an ideal simulation. The ideal case was obtained by simulating full optimization and zero execution time in each sample. The results are given in Table 6.2.

Strategy	Loss
Ideal case	1.0
Full optimization	1.35
Sub-optimal	1.09

Table 6.2 Performance loss for the different implementations in the single MPC case.

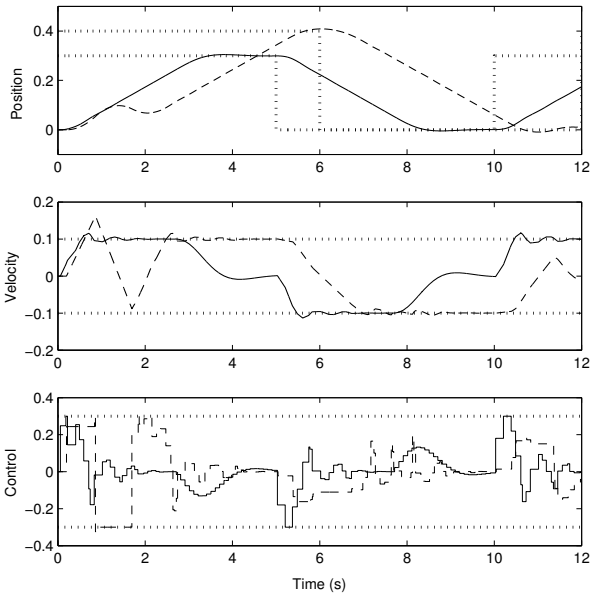

Figure 6.5 Control performance using fixed-priority scheduling where MPC1 (solid) is given the highest priority. MPC2 (dashed) is constantly preempted by the higher priority task, consequently degrading its performance.

Dynamic scheduling of two MPC tasks

In the following simulations the dynamic scheduling strategy proposed in Section 6.4 will be compared to ordinary fixed-priority scheduling. Two MPC controllers are implemented and executed by two different

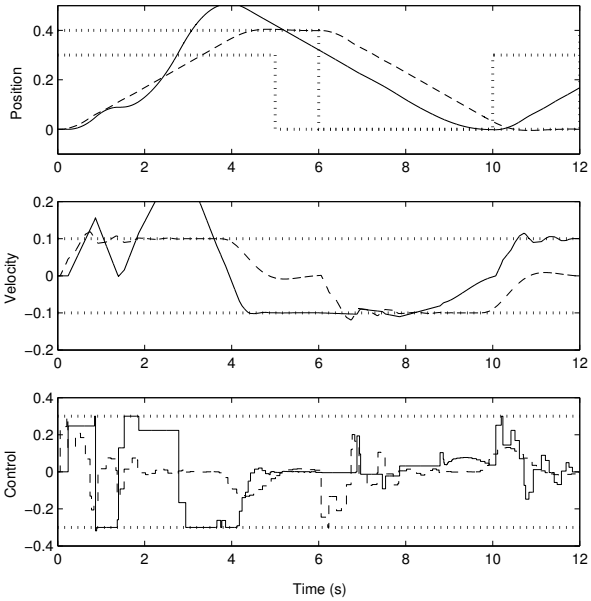

Figure 6.6 Control performance using fixed-priority scheduling where MPC2 (dashed) is given the highest priority. Comparing with Figure 6.5 it can be seen that the performance is worse using this priority assignment.

tasks running concurrently on the same CPU controlling two different double-integrator processes. Both MPC controllers are designed with the same prediction and control horizons, sampling periods, and weighting matrices in the MPC formulation.

Both controllers were given square-wave reference trajectories, but with different amplitudes and periods. The reference trajectory for MPC1 had an amplitude of 0.3 and a period of 10 s. The corresponding values for MPC2 were 0.4 and 12 s. The different reference trajectories will cause the relative computational demands of the MPC tasks to vary over time. Therefore, it is not obvious which controller task to give the highest priority. Rather, this should be decided on-line based on the current state of the controlled process.

The simulation results are shown in Figures 6.5-6.7. The first two

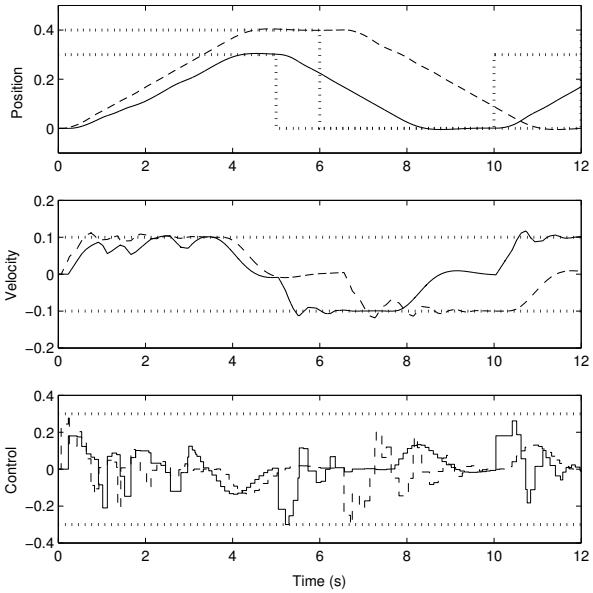

Figure 6.7 Control performance using the dynamic scheduling approach. Scheduling based on cost functions makes sure that the most urgent task gets access to the processor, thus increasing the overall performance.

simulations show the fixed-priority cases. MPC1 is given the highest priority in the first simulation, and MPC2 is given the highest priority in the second simulation. It is seen that we get different control performance, depending on how we choose the priorities. By giving MPC2 the highest priority, the performance in this particular simulation scenario is considerably better than if the priorities are reversed.

The performance using dynamic scheduling based on the cost index (6.6) is shown in Figure 6.7, and the performance is improved significantly. Figure 6.8 shows a close-up of the computer schedule during one sample. After both tasks have completed the mandatory parts of their algorithms, the execution trace (the dynamic priority assignments) is determined based on the values of the cost functions of the individual tasks. These values after each iteration are shown in the figure. The

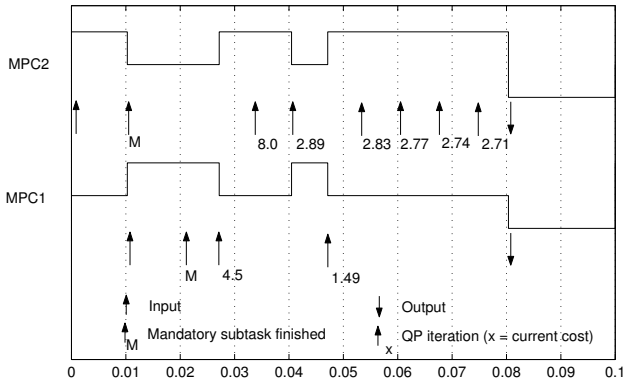

Figure 6.8 Computer schedule for the first sample using the dynamic scheduling approach (high = running, medium = preempted, low = idle). The figure shows the completion of the mandatory part, as well as the value of the cost index after each iteration of the QP solver.

Strategy	Loss
Ideal case	2.0
Fixed priority / MPC1 highest priority	2.47
Fixed priority / MPC2 highest priority	2.79
Dynamic cost-based scheduling	2.43

Table 6.3 Performance loss for the different scheduling strategies.

termination criterion aborts both tasks at time 0.08.

The scaled performance loss (6.8) was recorded for the individual control loops and added up to obtain a total loss for each of the different scheduling strategies. The results are summarized in Table 6.3. It can be seen that the improvement using the dynamic scheduling is less significant in the case where MPC1 is given the highest priority. This is, however, due to the particular reference trajectories applied in this simulation.

Using the proposed dynamic scheduling strategy we arbitrate the computing resources according to the current situation for the con-

trolled processes, and the varying computational demands caused by reference changes and other external signals are taken into account at run-time. It should be noted that the control performance obtained using the dynamic cost-based scheduling would have been the same if the reference trajectories for the two controllers had been switched. As we have shown this would not have been the case using ordinary fixed-priority scheduling.

6.6 Conclusions

In this chapter we have shown how a novel termination criterion can be employed to improve the performance of sub-optimal, stabilizing MPC. A new delay-dependent cost index has been presented that quantifies the trade-off between improving control signal quality resulting from successive iterations in the optimization algorithm and potential control performance degradation due to computational delay. The criterion provides guidance for when to terminate the optimization algorithm, while preserving the stability properties of the MPC algorithm.

It has also been shown how a delay-dependent cost index can be used in the context of dynamic real-time scheduling. The cost index has been used to provide the scheduling algorithm with information to be used for deciding which of two MPC controllers should be allocated execution time. Using the index for scheduling, it has been shown how the overall control performance may be significantly improved compared to traditional fixed-priority scheduling.

6.7 References

Årzén, K.-E., A. Cervin, J. Eker, and L. Sha (2000): "An introduction to control and scheduling co-design." In *Proceedings of the 39th IEEE Conference on Decision and Control*. Sydney, Australia.

Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*. Prentice Hall.

Bartlett, R. A., A. Wächter, and L. T. Biegler (2000): "Active set vs. interior point strategies for model predictive control." In *Proceedings of the American Control Conference*. Chicago, Illinois.

Bemporad, A., L. Chisci, and E. Mosca (1994): "On the stabilizing property of SIORHC." *Automatica*, **30:12**, pp. 2013–2015.

Bemporad, A., M. Morari, V. Dua, and E. N. Pistikopoulos (2002): "The explicit linear quadratic regulator for constrained systems." *Automatica*, **38:1**, pp. 3–20.

Buttazzo, G. C. (1997): *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers.

Cannon, M., B. Kouvaritakis, and J. A. Rossiter (2001): "Efficient active set optimization in triple mode MPC." *IEEE Transactions on Automatic Control*, **46:8**, pp. 1307–1312.

Cervin, A., J. Eker, B. Bernhardsson, and K.-E. Årzén (2002): "Feedback-feedforward scheduling of control tasks." *Real-Time Systems*, **23**, pp. 25–53.

Dunbar, W. B. and R. M. Murray (2002): "Model predictive control of coordinated multi-vehicle formations." In *Proceedings of the 41st IEEE Conference on Decision and Control*. Las Vegas, NV.

Dunbar, W. B., M. B. William, R. Franz, and R. M. Murray (2002): "Model predictive control of a thrust-vectored flight control experiment." In *Proceedings of the 15th IFAC World Congress on Automatic Control*. Barcelona, Spain.

Fletcher, R. (1991): *Practical methods of optimization 2nd ed.* John Wiley & Sons Ltd.

Garcia, C. E., D. M. Prett, and M. Morari (1989): "Model predictive control: Theory and practice – a survey." *Automatica*, **25:3**, pp. 335–348.

Henriksson, D., A. Cervin, J. Åkesson, and K.-E. Årzén (2002a): "On dynamic real-time scheduling of model predictive controllers." In *Proceedings of the 41st IEEE Conference on Decision and Control*. Las Vegas, NV.

Henriksson, D., A. Cervin, and K.-E. Årzén (2002b): "TrueTime: Simulation of control loops under shared computer resources." In *Proceedings of the 15th IFAC World Congress on Automatic Control*. Barcelona, Spain.

Kouvaritakis, B., M. Cannon, and J. Rossiter (2002): "Who needs QP for linear MPC anyway?" *Automatica*, **38:5**, pp. 879–884.

Liu, J., K.-J. Lin, W.-K. Shih, A. Yu, J.-Y. Chung, and W. Zhao (1991): "Algorithms for scheduling imprecise computations." *IEEE Trans on Computers*.

Liu, J., W.-K. Shih, K.-J. Lin, R. Bettati, and J.-Y. Chung (1994): "Imprecise computations." *Proceedings of the IEEE*, **82:1**, pp. 83–94.

Liu, J. W. S. (2000): *Real-Time Systems*. Prentice-Hall.

Maciejowski, J. M. (2002): *Predictive Control with Constraints*. Prentice-Hall.

Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert (2000): "Constrained model predictive control: Stability and optimality." *Automatica*, **36:6**, pp. 789–814.

Qin, S. J. and T. A. Badgwell (2003): "A survey of industrial model predictive control technology." *Control Engineering Practice*, **11**, pp. 733–764.

Richalet, J. (1993): "Industrial application of model based predictive control." *Automatica*, **29**, pp. 1251–1274.

Scokaert, P. O. M., D. Q. Mayne, and J. B. Rawlings (1999): "Suboptimal model predictive control (feasibility implies stability)." *IEEE Transactions of Automatic Control*, **44:3**, pp. 648–654.

Wright, S. J. (1997): *Primal-Dual Interior-Point Methods*. SIAM.

A

Parameter values for the Furuta pendulum

In order to obtain the parameter values of the Furuta pendulum, the process was disassembled, weighted and measured. The following measurements were made:

m_{pa}	$0.020\,kg$	Mass of pendulum
M	$0.015\,kg$	Mass of pendulum weight
l_p	$0.421\,m$	Pendulum Length
r	$0.245\,m$	Arm length
r_{cm}	$0.044\,m$	Distance from center of rotation to center of mass of arm
m_a	$0.165\,kg$	Mass of the arm
J_m	$0.0000381\,kgm^2$	Moment of inertia of motor and tachometer, from data sheet
ω_p	$5.23\,rad/s$	Natural frequency of pendulum with respect to its pivot point
ω_a	$7.12\,rad/s$	Natural frequency of arm with respect to its center of rotation

A.1 Moment of Inertia of the Pendulum

The moment of inertia of the pendulum is straight forward to calculate from

$$J_p = \left(\frac{1}{3}m_{pa} + M\right) l_p^2 = 0.00384\, kgm^2.$$

It is possible however, to obtain a verification of the value of the moment of inertia of the pendulum from the expression

$$J_p = \frac{mgl}{\omega_p^2} = 0.00377\, kgm^2.$$

where l (distance between center of rotation and center of mass of the pendulum with weight) is calculated from

$$l = \frac{m_{pa}/2 + M}{m_{pa} + M} l_p = 0.301\, m.$$

As we can see, there is a close correspondence between the theoretically calculated and experimentally determined value of the moment of inertia for the pendulum.

A.2 Moment of Inertia of the Arm Assembly

The moment of inertia of the pendulum arm was harder to determine, since it was not possible to detach the potentiometers at the back of the arm or the attachment device with the slip rings. Therefore, this calculation had to rely on experiments. We again use the relation

$$J_{arm} = \frac{m_a g r_{cm}}{\omega_a^2} = 0.00141\, kgm^2.$$

The moment of inertia of the entire arm assembly including motor is then

$$J_a' = J_{arm} + J_m = 0.00144\, kgm^2$$

B

Matlab tools for MPC

In this appendix the syntax of the MPC tools described in Chapter 4 is given. The tools are intended for use with Matlab R13, but should work also with Matlab R12. The tools require Control System Toolbox and, if the Simulink extension is to be used, also Simulink. The quadratic programming solver `quadprog` may be used to solve the MPC optimization problem, but this feature requires Optimization Toolbox. It is not necessary in order to use the tools however.

MPCinit

Purpose

Initializes the MPC data structure.

Syntax

```
md = MPCInit(Ad, Bd, Cyd, Czd, Dzd, Ccd, Dcd, Hp,
             Hw, zblk, Hu, ublk, du_max, du_min,
             u_max, u_min, z_max, z_min, Q, R, W, V,
             h, cmode, solver)
```

Input arguments

`Ad, Bd, Cyd`	System matrices for the plant; A_d, B_d and C_d.
`Czd, Dzd`	Matrices defining the controlled outputs; C_z and D_z.
`Ccd, Dcd`	Matrices defining the constrained outputs; C_c and D_c.
`Hw, Hp, Hu`	Integers defining the prediction and control horizons; H_w, H_p and H_u.
`zblk, ublk`	Blocking factors defining the sets I_p and I_u.
`u_min, u_max`	Control variable limits; u_{min} and u_{max}.
`du_min, du_max`	Control increment limits; Δu_{min} and Δu_{max}.
`z_min, z_max`	Controlled variable limits; z_{min} and z_{max}.
`Q, R`	Weighting matrices for the cost function.
`W, V`	Weighting matrices for the Kalman filter design, if applicable.
`h`	Sampling interval.
`cmode`	Controller mode.
`solver`	Solver to be used for the quadratic programming problem.

Output arguments

`md` Contains the pre-computed matrices needed by the MPC controller.

Description

`MPCInit` creates the data structure used by the MPC controller. A discrete time model, with sampling interval h, of the controlled system is assumed,

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= C_y x(k) \\ z(k) &= C_z x(k) + D_z u(k) \\ z_c(k) &= C_c x(k) + D_c u(k) \end{aligned} \tag{B.1}$$

where $z(k)$ are the controlled outputs, $y(k)$ the measured outputs and $z_c(k)$ the constrained outputs. The constraints of the system are given by

$$\begin{aligned} \Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max}, \quad k \in I_u \\ u_{min} \leq u(k) \leq u_{max}, \quad k \in I_u \\ z_{min} \leq z_c(k) \leq z_{max}, \quad k \in I_p \end{aligned} \tag{B.2}$$

where $\Delta u(k) = u(k) - u(k-1)$ are the control increments. I_p and I_u are the sets of samples for which the controlled and control variables are included in the cost function and for which the constraints are enforced. The first sample to be included in the optimization procedure is H_w, and the total number of samples in the prediction horizon is H_p. The entries in the array `z_blk` indicates the time distance between two consecutive predicted samples. Also, the last entry in `z_blk` is used as distance between the remaining samples (if `length(z_blk)` $< H_p$).

EXAMPLE B.1
Assume that $H_w = 1$, $H_p = 6$ and `z_blk` = [1 2 2 5]. Then the samples [1 2 4 6 11 16] will be included in the cost function evaluation. □

Equivalently, H_u indicates the number of predicted control moves to be calculated at each sample. The array `u_blk` contains the blocking factors indicating over which sampling intervals the control signal should be constant. For example, a blocking factor of 2 indicates that 2 consecutive control signals are equal.

EXAMPLE B.2
Assume that $H_u = 3$, `u_blk` = [1 2]. Then it is assumed that at each sample, for the predicted control signal, $\hat{u}$, we have that $\hat{u}(k+1) = \hat{u}(k+2)$ and $\hat{u}(k+3) = \hat{u}(k+4)$. Only $\hat{u}(k)$, $\hat{u}(k+1)$ and $\hat{u}(k+3)$ will be calculated. □

Q and R are weighting matrixes for the cost function, where Q penalizes the controlled outputs and R penalizes control increments. W and V are weighting matrices for the design of the Kalman filter used to estimate the state and load disturbances. W is the covariance matrix of the state and V is the covariance matrix of the measurement noise.

The controller supports several control modes. `cmode` specifies which mode should be used. The following modes are supported:

- `Mode 0`: State feedback
  The arguments W and V are not used, and may be set to `[]`. C_y is assumed to be identity.
- `Mode 1`: State feedback and explicit integrators
  The integrators act the controlled outputs, specified by C_z. The Q-matrix should be extended to include weights for the integrator states as well as the controlled outputs. The arguments W and V are not used, and may be set to `[]`.
- `Mode 2`: Observer based output feedback
  The design of the Kalman filter is based on the covariance matrices W and V.
- `Mode 3`: Observer based output feedback with explicit integrators
  The integrators act on the controlled outputs, specified by C_z. The Q-matrix should be extended to include weights for the integrator states as well as the controlled outputs. The controlled variables should be the same as the first p_z (p_z = `dim(z)`) measured variables.
- `Mode 4`: Disturbance observer based output feedback
  The disturbance model ensures that the controller achieves error free tracking. Constant load disturbances are assumed on the control input, and the system is augmented to include also the disturbance states. If the number of measured outputs exceed the number of inputs, constant load disturbances on the additional

measured outputs are assumed. The controlled variables should be the same as the first p_z measured variables. Also, the number of controlled outputs should be equal to the number of inputs.

The argument solver should be a string containing one of the alternatives `qp_as`, `qp_ip` or `quadprog`, indicating which solver should be used to solve the quadratic programming problem. Notice that `quadprog` requires Optimization Toolbox.

See Also

`MPCOptimizeSol`, `MPCSim` and `MPCfrsp`.

MPCOptimizeSol

Purpose

This function solves the MPC optimization problem.

Syntax

```
[duPred, zPred, J] = MPCOptimizeSol(x_est,u_last,du_old,
                                    r,md)
```

Input arguments

`x_est`	The current estimate of the state vector
`u_last`	The control signal applied to the plant at the last sample.
`du_old`	The optimal decision vector from the last sample that is used to hot start the quadratic programming algorithm.
`r`	The reference vector for the controlled variables.
`md`	The data structure containing pre-computed matrices.

Output arguments

`duPred`	Optimal control increment trajectory.
`zPred`	Optimal trajectory of the controlled variables.
`J`	Optimal value of the cost function.

Description

`MPCOptimizeSol` solves the MPC optimization problem. `duPred` and `zPred` are the predicted control increments and controlled outputs for the specified prediction horizons.

An estimate of the current state vector is given by `x_est`, and `u_last` is the last applied control input. As an initial starting point for the optimization algorithm, the last predicted control input vector, `du_old`, is supplied. `r` is the desired set point for the controlled outputs and md is the data structure containing matrices needed to solve the optimization problem.

See Also

MPCInit, MPCSim and MPCfrsp

MPCSim

Purpose

Simulates a linear system model controlled by the MPC controller.

Syntax

```
[x, u, y, z, zPredTraj, uPredTraj] = MPCSim(md,r,d)
```

Input arguments

`md`	Data object containing the pre-computed matrices.
`r`	Reference trajectory for the controlled variables.
`d`	Input disturbance trajectory.

Output arguments

`x`	State trajectory of the plant.
`u`	Control variable trajectory.
`y`	Measured variable trajectory.
`z`	Controlled variable trajectory.
`zPredTraj`	Optimal predicted trajectories for the controlled variables at each sample.
`uPredTraj`	Optimal predicted trajectories for the control variables at each sample.

Description

`MPCSim` simulates the MPC controller specified by the data object `md`. The reference trajectory for the controlled outputs is given by `r` and a load disturbance acting on the input is given by `d`.

See Also

`MPCInit`, `MPCOptimizeSol` and `MPCfrsp`

MPCController

Purpose

S-function for simulation of the MPC controller in the Simulink environment.

Syntax

```
[sys, x0, str, ts] = MPCController(t, x, u, flag, md)
```

Input arguments

`t`	Supplied by the Simulink environment.
`x`	Supplied by the Simulink environment.
`u`	Supplied by the Simulink environment.
`flag`	Supplied by the Simulink environment.
`md`	Data object containing the pre-computed matrices needed by the MPC controller. Supplied as a parameter in the S-function block.

Output arguments

`sys`	Needed by the Simulink environment.
`x0`	Needed by the Simulink environment.
`str`	Needed by the Simulink environment.
`ts`	Needed by the Simulink environment.

Description

`MPCController` is an S-function implementing the MPC controller intended for use with Simulink. The argument md, which is the only user supplied argument, contains the data structures needed by the controller. The input to the S-function block is a vector signal consisting of the measured outputs and the reference values for the controlled outputs. The output of the S-function block is a vector signal consisting of the control variables and the estimated state vector, potentially including estimated disturbance states.

See Also

`MPCOptimizeSol` and `MPCInit`

MPCfrsp

Purpose

Calculates the linear controller corresponding to the MPC controller if no constraints are active.

```
[Sys_CL, Sys_S, Sys_CS, Sys_SU, F, H, K, h] = MPCfrsp(md)
```

Input arguments

`md`	MPC data object.

Output arguments

`Sys_CL`	Closed loop system from r to z.
`Sys_S`	Sensitivity function from v to y.
`Sys_CS`	Complimentary sensitivity function: from n to y.
`Sys_SU`	Control signal sensitivity function: from n to u.
`F`	See block diagram.
`H`	See block diagram.
`K`	See block diagram.
`h`	A handle to the figure where the transfer functions are plotted.

See Figure B.1 for explanation of the notation.

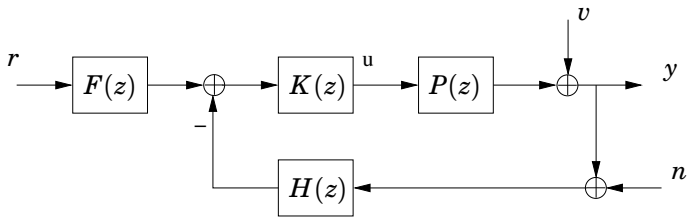

Figure B.1 Block diagram used for calculation of the linear controller.

Description

`MPCfrsp` calculates the frequency responses of the MPC controller. When no constraints are active, the MPC controller is a linear controller and may be analyzed using linear methods. Some important transfer functions are also plotted.

See Also

`MPCInit`

qp_as

Purpose

Solves a quadratic program using an active set method.

Syntax

```
[xopt, lambda, J, x_hist] = qp_as(H,f,A,b,x0)
```

Input arguments

`H`	H-matrix (Hessian) of the QP problem.
`f`	f-vector of the QP problem.
`A`	A-matrix defining linear constraints.
`b`	b-vector defining linear constraints.
`x0`	Initial solution guess.

Output arguments

`xopt`	The optimal solution.
`lambda`	Lagrange multipliers.
`J`	The optimal value of the cost function.
`x_hist`	History of x during the optimization run. Each column in `x_hist` represents the solution at the corresponding iteration.

Description

`qp_as` solves the quadratic programming problem

$$\min \frac{1}{2}x^T Hx + f^T x$$

s.t.

$$Ax \leq b$$

The algorithm is based on [Fletcher, 1987, p. 240].

See Also

`get feasible`

qp_ip

Purpose

Solves a quadratic program using a primal-dual interior point method.

Syntax

```
[xopt, lambda, J, x_hist] = qp_ip(H,f,A,b,x0)
```

Input arguments

`H`	H-matrix (Hessian) of the QP problem.
`f`	f-vector of the QP problem.
`A`	A-matrix defining linear constraints.
`b`	b-vector defining linear constraints.
`x0`	Initial solution guess.

Output arguments

`xopt`	The optimal solution.
`lambda`	Lagrange multipliers.
`J`	The optimal value of the cost function.
`x_hist`	History of x during the optimization run. Each column in `x_hist` represents the solution at the corresponding iteration.

Description

`qp_ip` solves the quadratic programming problem

$$\min \frac{1}{2}x^T Hx + f^T x$$

s.t.

$$Ax \leq b$$

The algorithm is based on [Wright, 1997]. An initial solution is supplied in `x0`, but is not used efficiently by the algorithm in the current implementation.

getfeasible

Purpose

Finds a feasible solution subject to linear inequality constraints.

Syntax

```
[x, as, iter] = getfeasible(A,b)
```

Input arguments

`A`	A-matrix defining linear constraints.
`b`	b-vector defining linear constraints.

Output arguments

`x`	A feasible solution.
`as`	The indices of active constraints, if any.
`iter`	The number of iterations.

Description

`getfeasible` finds a feasible vector that fulfills the linear inequality constraints

$$Ax \leq b.$$

The algorithm is based on [Fletcher, 1987, p. 166]

See Also

`qp_as`

B.1 References

Fletcher, R. (1987): *Practical Methods of Optimization*. John Wily & Sons Ltd.

Wright, S. J. (1997): *Primal-Dual Interior-Point Methods*. SIAM.