MODEL PREDICTIVE CONTROL
and optimization

Lecture notes
Regulering med PID og MPC
Torsdag 18. oktober 2001

Dr. ing.
David Di Ruscio

Division of Process Automation
Department of Technology
Telemark University College

Mars 2001

April 22, 2004

Forelesningsnotater
Institutt for prosessautomatisering
Avdeling for teknologiske fag
Høgskolen i Telemark
Kjølnes ring 56
N-3914 Porsgrunn

To Christina Maria, Daniel Magnus and Marit.

Preface

This report contains material to be used in a course in Model Predictive Control (MPC) at Telemark University College. The first four chapters are written for the course and the rest of the report is collected from earlier lecture notes and published work of the author.

Contents

Part I

Optimization and model predictive control

Chapter 1

Introduction

The material which is presented in this report is a collection of earlier lecture notes and published work on Model Predictive Control (MPC) presented by the author. The focus is on discrete time linear systems and predictive control based on state space models. However, the theory which is presented is general in the sense that it is able to handle MPC based on any linear dynamic model. The main MPC algorithm and the notation which is presented is derived from the subspace identification method DSR, Di Ruscio (1994), (1995). This algorithm is denoted Extended Model based Predictive Control (EMPC). The name comes from the fact that the method can be derived from the Extended State Space Model (ESSM) which is one of the basic matrix equations for deriving the DSR method. An interesting link is that the subspace matrices from the DSR method also can be used directly in the MPC method. The EMPC method can off-course also be based on a general linear state space model.

One of the advantages of the EMPC method is that important MPC methods such as the Generalized Predictive Control (GPC) algorithm and Dynamic Matrix Control (DMC) pull out as special cases. The GPC algorithm is based on an input-output CARIMA model, which is an ARMAX model in terms of control deviation variables. The DMC method is based on Finite Impulse Response (FIR) and step response models. The theory presented is meant to be general enough in order to make the reader able to understand MPC methods which is based on linear models, e.g., state space models. Another variant is the MPC method presented by Rawlings.

The main advantage constraining the description to linear models is that it results in a linear prediction model. The common control objective used in MPC is the Linear Quadratic (LQ) objective (cost function). An LQ objective, a linear prediction model and linear constraints gives rise to a so called Quadratic Programming (QP) optimization problem. A QP problem can be solved within a finite number of numerical operations. A QP problem is a convex optimization problem with a unique minimum, if the problem is feasible. The QP problem makes the resulting MPC algorithm robust for process control.

On the other side, if the process model is allowed to be non-linear then in general also the prediction model will be non-linear. This leads to a non-linear optimization method which usually is solved by a Sequential Quadratic Programming (SQP) method. A non-linear MPC is not guaranteed to converge within reasonable computing time. Furthermore, a non-linear optimization problem often has problems with local minima and convergence problems. Hence, a non-linear MPC method may not be robust for on-line process control.

Chapter 2

Model predictive control

2.1 Introduction

Model Predictive Control (MPC) is a control strategy which is a special case of the optimal control theory developed in the 1960 and lather. MPC consists of an optimization problem at each time instants, k. The main point of this optimization problem is to compute a new control input vector, u_k, to be feed to the system, and at the same time take process constraints into consideration (e.g. constraints on process variables). An MPC algorithm consists of

- **Cost function**
  A control objective, J_k, (or cost function) which is a scalar criterion measuring e.g., the difference between future outputs, $y_{k+1|L}$, and some specified (future) reference, $r_{k+1|L}$, and at the same time recognizing that the control, u_k, is costly. The price on control is therefore also usually measured in, J_k. Hence, the objective is a measure of the process behaviour over the prediction horizon, L. This objective is minimized with respect to the future control vectors, $u_{k+1|L}$, and only the first control vector, u_k, is actually used for control. This optimization process is solved again at the next time instant, i.e, at $k := k+1$. This is sometimes called an receding horizon control problem.

- **Constraints**
  One of the main motivation behind MPC is that constraints on process variables simply can be treated. Common constraints as input amplitude constraints and input rate of change constraints can be treated far more efficient than in conventional control systems (PID-control). This usually leads to a simple inequality constraint, $Au_{k+1|L} \leq b$, which is added to the optimization problem.

- **Prediction model**
  The main drawback with MPC is that a model for the process, i.e., a model which describes the input to output behaviour of the process, is needed. Mechanistic models derived from conservation laws can be used. Usually, however in practice simply data-driven linear models are used. A promising choice which has got great attention is to use the models identified by the subspace

identification methods, e.g., the state space model (A, B, D, E) or even the subspace matrices, $(\tilde{A}_L, \tilde{B}_L)$, from the DSR method can with advantage be used for MPC. This may be referred to as model free MPC. The use of DSR leads to a fast implementation of MPC. The model is primarily used to predict the outputs, $y_{k+1|L}$, (and the states) over the prediction horizon. The process model is usually used to construct a PM. The purpose of the PM is to describe the relationship between the future outputs and the future control inputs to be computed. The PM is a part of the optimization problem and is needed for this reason.

Another advantage of MPC is that cross coupling in multiple input and multiple output (MIMO) systems are taken into consideration in an optimal way. MPC is a simple method for controlling MIMO systems.

It is also important to note that the MPC method with advantage can be used for operator support. In some cases we are only interested in obtaining suggestions for the control action, and not to feed back the computed control, u_k, to the process. The MPC method can be used to (at each time instant) compute the future optimal controls, $u_{k|L}$. Hence, we have a methodology to compute control suggestions which may be a valuable tool for the process operators. Note that a conventional control system can not be used for this purpose.

2.2 The control objective

The common control objective used in connection with MPC is given by the scalar function

$$J_k = \sum_{i=1}^{L}((y_{k+i} - r_{k+i})^T Q_i (y_{k+i} - r_{k+i}) + u_{k+i-1}^T P_i u_{k+i-1} + \Delta u_{k+i-1}^T R_i \Delta u_{k+i-1}), \tag{2.1}$$

where L is defined as the prediction horizon, $Q_i \in \mathbb{R}^{m\times m}$, $P_i \in \mathbb{R}^{r\times r}$ and $R_i \in \mathbb{R}^{r\times r}$ are symmetric and positive semi-definite weighting matrices specified by the user. In some cases and for some MPC methods we simply chose $Q_i = qI_m$, $P_i = pI_r$ and $R_i = r_0 I_r$ for some positive parameters q, p and r_0. The more general choice is to specify Q_i, P_i and R_i as diagonal weighting matrices. Often, P is chosen as zero in order to obtain MPC with offset-free control, i.e., $y = r$ in steady state. The weighting matrices are almost always chosen as time invariant matrices, i.e., the weighting matrices are constant over the prediction horizon L so that $Q_1 = Q_2 = \ldots = Q_L$, $P_1 = P_2 = \ldots = P_L$ and $R_1 = R_2 = \ldots = R_L$.

The problem of choosing the weighting matrices are usually process dependent and must usually be chosen by trial and error. However, if Q, P and R are chosen as diagonal positive definite matrices and if the prediction horizon L is large (infinite) then it can be proved some remarkable properties of the closed loop (controlled) system. The closed loop system with the optimal control is guaranteed to be stable even if the open loop system is unstable. Furthermore, for SISO systems we are guaranteed a phase margin of 60° or more and an infinite gain margin (i.e. the gain in the loop can be increased by a factor $0.5 \leq k \leq \infty$. For details we refer to a course in advanced optimal control.

The control objective, J_k, is also often denoted a cost function. Note also that another common symbol for it simply is $\mathcal{J}_k := J_k$. The control objective can be written on matrix form. The main point of doing this is to remove the summation sign from the problem and the solution. This will simplify the discussion considerably and lead to a Quadratic Programming (QP) problem. A matrix formulation of the objective J_k will be used throughout this report. The matrix equivalent to (2.3) is given by

$$J_k = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + u_{k|L}^T P u_{k|L} + \Delta u_{k|L}^T R \Delta u_{k|L}, \quad (2.2)$$

where $Q \in \mathbb{R}^{Lm \times Lm}$, $P \in \mathbb{R}^{Lr \times Lr}$ and $R \in \mathbb{R}^{Lr \times Lr}$ are symmetric and positive semi-definite block diagonal weighting matrices. The control problem is

$$u_{k|L}^* = \arg\min_{u_{k|L}} J_k(u_{k|L}) \quad (2.3)$$

subject to a prediction model and process variable constraints if specified.

The control objective is motivated from the requirement to hold the process outputs $y_{k+1|L}$ as close as possible to some specified references $r_{k+1|L}$ but at the same time minimize the control (energy) $u_{k|L}$, recognizing that the control is costly.

A promising choice is to put $P = 0$. The reason for this is that the MPC control will give offset-free control (if the constraints are not active), and that the problem is independent of target values and non-zero offset on the control inputs. Note that $P = 0$ is the default choice in the GPC and DMC methods. This is also usually used in the EMPC method. By simply choosing $P = 0$ a lot of practical problems regarding non-zero mean process variables are avoided.

2.3 Prediction models for use in MPC

A strictly proper linear dynamic process model can always be written as a prediction model (PM) which takes the standard form

$$y_{k+1|L} = F_L u_{k|L} + p_L, \quad (2.4)$$

where L is the prediction horizon, $F_L \in \mathbb{R}^{Lm \times Lr}$ is a (constant) matrix derived from the process model, $p_L \in \mathbb{R}^{Lm}$ is a vector which in general is dependent of a number of inputs and outputs older than time k as well as the model parameters. Note that in some cases that F_L and p_L may be identified directly. The PM (2.4) can be used directly in MPC algorithms which are computing the actual control input vectors, $u_{k|L}$.

Some algorithms for MPC are computing process deviation variables, i.e., computing the vector of $\Delta u_{k|L}$ of future control deviation variables. Then $u_k = \Delta u_k + u_{k-1}$ is used as the actual control vector. For this case it is convenient with a PM on the form

$$y_{k+1|L} = F_L^{\Delta} \Delta u_{k|L} + p_L^{\Delta}. \quad (2.5)$$

The main point of writing the PM in a form as (2.4) or (2.5) is that the future predictions is directly expressed as a function of the unknown future control vectors which are to be computed by the MPC algorithm. We will in the following sections illustrate how a PM can be build from different linear models.

Most MPC algorithms and applications are based on a linear dynamic model of the process. A state space model yields a general description of a linear dynamic system. However, many of the MPC applications are based on special case input and output models such as Finite Impulse Response (FIR) and step response models, e.g. the Matrix Algorithm Control (MAC) algorithm and Dynamic Matrix Control (DMC) algorithm. These algorithms are not general because they realize upon models which only can approximately describe some special case linear dynamic systems, e.g., stable systems and systems without integrators. Another method which has got great attention is the Generalized Predictive Control (GPC) method, which is based upon a Auto Regression Moving Average with eXogenous/extra inputs (ARMAX) model on deviation form, i.e., a so called CARIMA model. One pont of this report is that all these methods can simply be described within the same framework. This leads to the Extended Model Predictive Control (EMPC) algorithm, Di Ruscio (1997). See also the paper Di Ruscio and Foss (1998) and Chapter 6. The EMPC method can be based on any linear model. The theory is derived from the theory of subspace identification and in particular the DSR algorithm, Di Ruscio (1995), (1996). Another point of using the state space approach or the extended state space model approach is that the state space model or the subspace matrices from DSR can be used directly in the MPC algorithm. The MAC, DMC and GPC methods pulls out as special cases of the EMPC algorithm.

2.3.1 Prediction models from state space models (EMPC$_1$)

Any strictly proper deterministic linear dynamic system can be written as a state space model

$$x_{k+1} = Ax_k + Bu_k, \tag{2.6}$$

$$y_k = Dx_k. \tag{2.7}$$

The only proper case when the output equation is of the form $y_k = Dx_k + Eu_k$ is treated in Chapter 11. Hence, it is important to now how a prediction model (PM) for the use in MPC simply can be build from the state space model (2.6) and (2.7). The PM is simply

$$y_{k+1|L} = F_L u_{k|L} + p_L, \tag{2.8}$$

where

$$F_L = \begin{bmatrix} O_L B & H_L^d \end{bmatrix}, \tag{2.9}$$

$$p_L = O_L A x_k. \tag{2.10}$$

Note that if the states is not measured, then, x_k may be computed from the knowledge of a number of past inputs and outputs over the past horizon, J. This is one of the options in the EMPC algorithm. See below for details. The states can also be estimated in a state observer, e.g. using the Kalman filter (gain) estimated from the DSR algorithm.

Equations (2.9) and (2.10) can be proved as follows. One of the basic matrix equations for strictly proper systems in the subspace identification theory is

$$y_{k|L} = O_L x_k + H_L^d u_{k|L-1}. \tag{2.11}$$

Putting $k := k+1$ in (2.11) gives

$$y_{k+1|L} = O_L(Ax_k + Bu_k) + H_L^d u_{k+1|L-1}. \tag{2.12}$$

This can be written in matrix form identical to (2.8) with F_L and p_L given in (2.9) and (2.10)

The term p_L in (2.10) is dependent upon the present state, x_k. An especially simple way of computing an estimate for the present state, x_k, is as presented in Di Ruscio (1997), i.e.,

$$x_k = A^{J-1} O_J^\dagger y_{k-J+1|J} + (C_{J-1} - A^{J-1} O_J^\dagger H_J^d) u_{k-J+1|J-1}, \tag{2.13}$$

where $u_{k-J+1|J-1}$ and $y_{k-J+1|J}$ is defined from the known past inputs and outputs, respectively.

$$y_{k-J+1|J} = \begin{bmatrix} y_{k-J+1} \\ y_{k-J+2} \\ \vdots \\ y_{k-1} \\ y_k \end{bmatrix} \in \mathbb{R}^{Jm}, \quad u_{k-J+1|J-1} = \begin{bmatrix} u_{k-J+1} \\ u_{k-J+2} \\ \vdots \\ u_{k-1} \end{bmatrix} \in \mathbb{R}^{(J-1)r}. \tag{2.14}$$

Here, J is a user specified horizon into the past. We may simply chose the minimum J to ensure existence of the solution (2.13), i.e. chose J so that $\text{rank}(O_J) = n$. We can simply chose $J = n - \text{rank}(D) + 1$ when $m < n$ and $J = 1$ when $m \geq n$.

Any linear dynamic model has a state space equivalent. A simple method of building a state space model from a known input and output model is to generate data (Y, U) from the known input-output model and then identify the state space model by using the DSR method. Real process data is however to be preferred. The PM is then constructed as above.

Note that a FIR model with M terms can be expressed as a state space model of order M. The system matrix A in the corresponding state space model can be build in MATLAB by the command

```
>> A=diag(ones(M-1,1),1)
>> B=[h1;h2;...;hM]
>> D=eye(1,M)
```

The system matrix B consists of the impulse responses. See Example 2.1 and 2.2 for illustrations of building a state space model from FIR and step response models. Furthermore one should note that more general ARMAX and CARIMA models also have a state space equivalent. See Chapter 12 for some examples.

2.3.2 Prediction models from state space models (EMPC$_2$)

A prediction model in terms of process deviation variables can be derived from (2.8) by using the relationship $u_{k|L} = S\Delta u_{k|L} + cu_{k-1}$. The matrices S and c consists of ones and zeroes, see Section 6 for the definitions. Hence, we have

$$y_{k+1|L} = F_L^{\Delta}\Delta u_{k|L} + p_L^{\Delta}, \tag{2.15}$$

where

$$F_L^{\Delta} = F_L S, \tag{2.16}$$

$$p_L^{\Delta} = p_L + F_L c u_{k-1}, \tag{2.17}$$

where p_L is given by (2.10) and with advantage (2.13) if the states is not available. For further details concerning the problem of building a PM from a states space model (A, B, D) and/or the subspace matrices $\tilde{A}_L, \tilde{B}_L$ we refer to Section 6. See also Di Ruscio and Foss (1998).

The term p_L^{Δ} is not unique. We will in the following present an alternative formulation which have some important advantages. The presentation is in the same spirit as the alternative presented in Section 6.3.2. Taking the difference of (2.8) gives

$$y_{k+1|L} = y_{k|L} + O_L A\Delta x_k + F_L\Delta u_{k|L}. \tag{2.18}$$

Using (2.18) recursively gives

$$y_{k+1|L} = y_{k-J+1|L} + O_L A\sum_{i=1}^{J}\Delta x_{k-i+1} + F_L\sum_{i=1}^{J}\Delta u_{k-i+1|L}. \tag{2.19}$$

2.3.3 Prediction models from FIR and step response models

Consider the state space model in (2.6) and (2.6). An expression for y_k can be expressed as

$$y_k = DA^i x_{k-i} + DC_i u_{k-i|i}, \tag{2.20}$$

where C_i is the extended controllability matrix, i.e., $C_0 = 0$, $C_1 = B$, $C_2 = \begin{bmatrix} B & AB \end{bmatrix}$ and so on. Assume now that the process is stable, i.e., A has all eigenvalues inside the unit circle in the complex plane. In this case we have that $A^M \approx 0$ when $M = i \geq 1$ is large. Hence,

$$\begin{aligned} y_k &= DC_M u_{k-M|M}, \\ A^M &\approx 0. \quad \text{for some model horizon, } M, \end{aligned} \tag{2.21}$$

where

$$DC_M = \begin{bmatrix} H_1 & H_2 & \dots & H_M \end{bmatrix} = \begin{bmatrix} DB & DAB & \dots & DA^{M-1}B \end{bmatrix}, \tag{2.22}$$

is a matrix of impulse response matrices. The input output model (2.21) is called a FIR model and M is defined as the model horizon. Using (2.21) in order to express y_{k+1} and subtracting y_k gives

$$y_{k+1} = y_k + C_M\Delta u_{k+1-M|M}. \tag{2.23}$$

$$\Delta u_{k+1-M|M} = u_{k+1-M|M} - u_{k-M|M} \tag{2.24}$$

The input output model (2.23) is called a step response model.

The model horizon is typically reported to be in the range $20 \leq M \leq 70$, Seborg *et al* (1989). As illustrated above, the parameters in the FIR and the step response model are related to the impulse response matrices of the state space model. The parameters in C_M is often obtained directly by system identification. However, there may be a huge number of parameters to be estimated and this problem may be ill-conditioned compared to only identifying the model matrices (A, B, D).

A PM can be build from (2.21) and (2.23) in different ways:

1. Via the state space model matrices (A, B, D). See Section 2.3.1.
2. Via the subspace matrices (i.e., the extended state space model matrices) $(\tilde{A}_L, \tilde{B}_L)$. See Chapter 6 for details.
3. Direct derivation as illustrated in Examples 2.3 and 2.4.

See Examples 2.3, 2.4 and 2.5 for illustrations of building a PM from FIR and step response models. The FIR and step response models can also be converted to a state space model and then constructing the PM as in Section 2.3.1. See also Examples 2.1 and 2.2.

2.3.4 Prediction models from models with non zero mean values

We have so far and in Section 2.3.1 based our discussion of how to make a prediction model from state space models of the form

$$x_{k+1} = Ax_k + Bu_k, \tag{2.25}$$

$$y_k = Dx_k. \tag{2.26}$$

However, in many practical cases the model is obtained by linearizing a physical model as described in Appendix A, or even more important, the model is identified from centered data or data where some constant values are removed. Hence, we may have a model of the form

$$x_{k+1} = Ax_k + Bdu_k, \tag{2.27}$$

$$dy_k = Dx_k, \tag{2.28}$$

where

$$du_k = u_k - u^0, \tag{2.29}$$

$$dy_k = y_k - y^0, \tag{2.30}$$

and $u^0 \in \mathbb{R}^r$ and $y^0 \in \mathbb{R}^m$ are constant vectors.

A simple solution to the problem of making a prediction model is to first transform the model in (2.27) and (2.28) to a model of the form (2.25) and (2.26). This is presented in detail in Section 11. One (insignificant) drawback with this is that the transformed model will have one additional state. This additional state is an integrator (i.e., there is an eigenvalue equal to one in the A matrix in the transformed model), which take care of the non-zero constant trends. Hence, all the theory which is developed from state space models of the form (2.25) and (2.26) can be used without modifications.

However, the presented theory on MPC algorithms may be modified to take properly consideration to possible nonzero vectors u^0 and y^0. Using (2.8) we have that

$$dy_{k+1|L} = F_L du_{k|L} + p_L, \tag{2.31}$$

where

$$\begin{aligned} F_L &= \begin{bmatrix} O_L B & H_L^d \end{bmatrix}, & (2.32)\\ p_L &= O_L A x_k. & (2.33) \end{aligned}$$

Noticing that $dy_{k+1|L}$ and $du_{k|L}$ are deviation variables, we have that the PM of the actual future outputs can be expressed by

$$y_{k+1|L} = F_L u_{k|L} + p_L^0, \tag{2.34}$$

where

$$p_L^0 = p_L + \begin{bmatrix} I_m \\ \vdots \\ I_m \end{bmatrix} y^0 - F_L \begin{bmatrix} I_r \\ \vdots \\ I_r \end{bmatrix} u^0, \tag{2.35}$$

where p_L is as before and given in (2.33). Note that the indicated matrices with identity matrices I_m and I_r are of dimensions $\mathbb{R}^{Lm}$ and $\mathbb{R}^{Lr}$, respectively.

If not the state, x_k, in (2.33) can be measured, then a state observer (e.g., Kalman filter) kan be constructed from the state space model (2.27)-(2.30). The Kalman filter identified by using the DSR method, Di Ruscio (1996), can with advantage be used. However, the state estimate in (2.13) can be modified similarly as we modified the PM above. We have

$$x_k = A^{J-1} O_J^\dagger dy_{k-J+1|J} + (C_{J-1} - A^{J-1} O_J^\dagger H_J^d) du_{k-J+1|J-1}, \tag{2.36}$$

where $du_{k-J+1|J-1}$ and $dy_{k-J+1|J}$ is defined from the known past inputs and outputs, and the known constant vectors u^0 and y^0, as follows

$$dy_{k-J+1|J} = y_{k-J+1|J} - \begin{bmatrix} y^0 \\ \vdots \\ y^0 \end{bmatrix} \in \mathbb{R}^{Jm}, \tag{2.37}$$

$$du_{k-J+1|J-1} = u_{k-J+1|J-1} - \begin{bmatrix} u^0 \\ \vdots \\ u^0 \end{bmatrix} \in \mathbb{R}^{(J-1)r}. \tag{2.38}$$

The extended output and input vectors $y_{k-J+1|J}$ and $u_{k-J+1|J-1}$, respectively, are as defined in (2.14). Furthermore, J is a user specified horizon into the past. See further comments in Section 2.3.1.

One should note that the methods which are computing control deviation variables, $\Delta u_k = u_k - u_{k-1}$, and based upon the PM formulation in Section 6, are insensitive to non-zero mean values on u_k (when $P = 0$ in the objective 2.3).

2.3.5 Prediction model by solving a Diophantine equation

The original GPC algorithm is based on an input-output model of the form

$$A(z^{-1})y_k = z^{-d}B(z^{-1})u_{k-1} + \frac{C(z^{-1})}{\Delta}e_k, \tag{2.39}$$

where $0 \leq d$ is a specified delay, e_k are white noise and $\Delta = 1 - z^{-1}$ is the differentiating operator. This is a so called Controller Auto-Regressive Integrated Moving-Average (CARIMA) model. Another frequently used name for it is an Auto-Regressive Integrated Moving-Average with eXtra inputs (ARIMAX). CARIMA is motivated from the fact that u_k is a control variable. The main point of the differentiator $\Delta = 1 - z^{-1}$ is to obtain an PM in terms of control deviation variables, i.e. to obtain a PM of the form (2.15). Another advantage is that integral action in the controller is obtained, i.e., resulting in zero steady state offset between y_k and the reference r_k. The resulting controller is insensitive to non-zero mean control variables and constant disturbance values. Most important, it leads to an MPC which are computing control deviation variables $\Delta u_{k|L}$.

In the following we will discuss the SISO case. The theory can be extended to MIMO systems, as described lather in this section. However this is not so numerically practical compared to the state space approach. For SISO systems we have that the polynomials in (2.39) are given by

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{na} z^{-na}, \tag{2.40}$$

$$B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{nb} z^{-nb}, \tag{2.41}$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + c_2 z^{-2} + \ldots + c_{nc} z^{-nc}, \tag{2.42}$$

where na, nb and nc are the order of the $A(z^{-1})$, $B(z^{-1})$ and $C(z^{-1})$ polynomials, respectively.

The prediction of the jth output y_{k+j} $\forall\ 1 \leq j \leq L$ is given by

$$y_{k+j} = G_j(z^{-1})\Delta u_{k+j-d-1} + F_j(z^{-1})y_k. \tag{2.43}$$

In the following we will discuss the SISO case where the noise polynomial is equal to $C(z^{-1}) = 1$. The theory can simply be extended to the colored noise case. The polynomials $G_j(z^{-1})$ and $F_j(z^{-1})$ are obtained as described in the following. First solve the Diophantine equation

$$1 = E_j(z^{-1})\tilde{A}(z^{-1}) + z^{-j}F_j(z^{-1}), \tag{2.44}$$

$$\tilde{A}(z^{-1}) = \Delta A(z^{-1}), \tag{2.45}$$

$$\Delta = 1 - z^{-1}, \tag{2.46}$$

where ($\tilde{A}(z^{-1})$ is obtained by multiplying the two polynomials Δ and $\tilde{A}(z^{-1})$, i.e.,)

$$\tilde{A}(z^{-1}) = \tilde{a}_0 + \tilde{a}_1 z^{-1} + \tilde{a}_2 z^{-2} + \ldots + \tilde{a}_{na+1} z^{-(na+1)}, \tag{2.47}$$

for the unknown coefficients in the polynomials $E_j(z^{-1})$ and $F_j(z^{-1})$. These polynomials is of the form

$$E_j(z^{-1}) = e_{j0} + e_{j1} z^{-1} + e_{j2} z^{-2} + \ldots + e_{j,j-1} z^{-(j-1)}, \tag{2.48}$$

Note that we when $C(z^{-1}) = 1$ have that $e_{j0} = 1$. Since $\tilde{A}(z^{-1}) = \Delta A(z^{-1})$ is of order $na + 1$, then the product of the polynomials $E_j(z^{-1})\Delta A(z^{-1})$ must be of order $j + na$. Requiring that the two terms on the left hand side of the Diophantine Equation (2.44) are of the same order, then we have that $F_j(z^{-1})$ must be of order na, i.e.,

$$F_j(z^{-1}) = f_{j0} + f_{j1}z^{-1} + f_{j2}z^{-2} + \ldots + f_{j,na}z^{-na}. \tag{2.49}$$

The role of the $F_j(z^{-1})$ polynomial is very important since it decides how many old outputs which are to be used in order to predict the future outputs. Hence, remark that for a single output system a number of $na + 1$ old outputs are used. Hence, the future predictions will be a function of the known outputs in the vector $y_{k-na|na+1}$.

Once, $E_j(z^{-1})$ is known, then we compute the coefficients in the $G_j(z^{-1})$ polynomials from the equations

$$G_j(z^{-1}) = E_j(z^{-1})B(z^{-1}) \ \forall \ j = 1, \ldots, L. \tag{2.50}$$

Hence, the G_j polynomials are found by multiplying two known polynomials. Note that the coefficients in the polynomials $E_j(z^{-1})$, $G_j(z^{-1})$ and $F_j(z^{-1})$ are different for different numbers j. Hence, we have to solve $j = L$ Diophantine equations, i.e. for $1 \leq j \leq L$, in order to obtain the PM. The resulting PM can be written in the standard prediction model form

$$y_{k+1|L} = F_L^{GPC}\Delta u_{k|L} + p_L^{GPC}. \tag{2.51}$$

It is important to note that the matrices in the PM (2.51) is related to the PM obtained in Section 2.3.1 as

$$F_L^{GPC} = F_L^{\Delta} = F_L S. \tag{2.52}$$

where F_L is related to the equivalent state space model matrices (A, B, D) as given by (2.9). The term p_L^{GPC} can be obtained directly from the state space model as described in Di Ruscio (1997), see also Chapter 6 and Proposition 3.4.

The coefficients in the polynomials $E_j(z^{-1})$ and $F_j(z^{-1})$ can be obtained recursively. The simplicity of this process is the same for the MIMO and SISO cases. Hence, the following procedure can also be used to define the PM for MIMO systems. First define initial polynomials for the recursion directly as

$$E_1(z^{-1}) = I_{m\times m}, \tag{2.53}$$

$$F_1(z^{-1}) = z(I_{m\times m} - \tilde{A}(z^{-1}). \tag{2.54}$$

Then for $j = 1, \ldots, L-1$, the coefficients/matrices, $f_{j+1,i}$, in the remaining F_j polynomials are computed as follws. For each j do

$$R_j = f_{j,0}, \tag{2.55}$$

$$f_{j+1,i} = f_{j,i+1} - R_j\tilde{a}_{i+1} \quad \forall \ i = 0, 1, \ldots, na-1 \tag{2.56}$$

$$f_{j+1,i} = -R_j\tilde{a}_{i+1} \quad i = na \tag{2.57}$$

See Example 2.6 for a demonstration of this recursion scheme for obtaining the polynomials. Furthermore, this recursion scheme is implemented in the MATLAB function **poly2gpcpm.m**.

2.3.6 Examples

Example 2.1 (From FIR to state space model)
A FIR model

$$y_k = h_1 u_{k-1} + h_2 u_{k-2} + h_3 u_{k-3}, \tag{2.58}$$

can be simply written as a state space model

$$\begin{bmatrix} x^1_{k+1} \\ x^2_{k+1} \\ x^3_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^1_k \\ x^2_k \\ x^3_k \end{bmatrix} + \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} u_k, \tag{2.59}$$

$$y_k = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^1_k \\ x^2_k \\ x^3_k \end{bmatrix}. \tag{2.60}$$

Note that the order of this state space model, which is $M = 3$, in general is different from the order of the underlying system. The theory in Section 2.3.1 can then be used to construct a prediction model.

Example 2.2 (From step response to state space model)
A step response model can be derived from the FIR model in (2.58), i.e.,

$$y_{k+1} = a_0 y_k + h_1 \Delta u_{k-1} + h_2 \Delta u_{k-2} + h_3 \Delta u_{k-3}, \tag{2.61}$$

with $a_0 = 1$. This can simply be written as a state space model

$$\begin{bmatrix} x^1_{k+1} \\ x^2_{k+1} \\ x^3_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & a_0 \end{bmatrix} \begin{bmatrix} x^1_k \\ x^2_k \\ x^3_k \end{bmatrix} + \begin{bmatrix} h_1 \\ h_1 + h_2 \\ h_1 + h_2 + h_3 \end{bmatrix} \Delta u_k, \tag{2.62}$$

$$y_k = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^1_k \\ x^2_k \\ x^3_k \end{bmatrix}. \tag{2.63}$$

Note that the order of this state space model, which is $M = 3$, in general is different from the order of the underlying system. The theory in Section 2.3.1 can then be used to construct a prediction model.

Example 2.3 (Prediction model from FIR model)
Assume that the process can be described by the following finite impulse response (FIR) model

$$y_k = h_1 u_{k-1} + h_2 u_{k-2} + h_3 u_{k-3}. \tag{2.64}$$

Consider a prediction horizon, $L = 4$. The future predictions is then

$$y_{k+1} = h_1 u_k + h_2 u_{k-1} + h_3 u_{k-2}, \tag{2.65}$$

$$y_{k+2} = h_1 u_{k+1} + h_2 u_k + h_3 u_{k-1}, \tag{2.66}$$

$$y_{k+3} = h_1 u_{k+2} + h_2 u_{k+1} + h_3 u_k, \tag{2.67}$$

$$y_{k+4} = h_1 u_{k+3} + h_2 u_{k+2} + h_3 u_{k+1}. \tag{2.68}$$

This can be written in matrix form as follows

$$\overbrace{\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ y_{k+4} \end{bmatrix}}^{y_{k+1|4}} = \overbrace{\begin{bmatrix} h_1 & 0 & 0 & 0 \\ h_2 & h_1 & 0 & 0 \\ h_3 & h_2 & h_1 & 0 \\ 0 & h_3 & h_2 & h_1 \end{bmatrix}}^{F_4} \overbrace{\begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ u_{k+3} \end{bmatrix}}^{u_{k|4}} + \overbrace{\begin{bmatrix} h_3 & h_2 \\ 0 & h_3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{k-2} \\ u_{k-1} \end{bmatrix}}^{p_4}. \tag{2.69}$$

Which can be expressed in standard (prediction model) form as

$$y_{k+1|4} = F_4 u_{k|4} + p_4. \tag{2.70}$$

Example 2.4 (Step response prediction model)
Consider the FIR model in (2.3), i.e.,

$$y_{k+1|4} = F_4 u_{k|4} + p_4. \tag{2.71}$$

where F_4, and p_4 are as in (2.69). A step response model is a function of the control deviation variables (control rate of change) $\Delta u_k = u_k - u_{k-1}$, $\Delta_{k+1} = u_{k+1} - u_k$ and so on. This means that a step response model is a function of the vector $\Delta_{k|L}$, where the prediction horizon is $L = 4$ in this example. An important relationship between $u_{k|4}$ and $\Delta_{k|4}$ can be written in matrix form as follows

$$\overbrace{\begin{bmatrix} u_{k+1} \\ u_{k+2} \\ u_{k+3} \\ u_{k+4} \end{bmatrix}}^{u_{k|4}} = \overbrace{\begin{bmatrix} I_r & 0 & 0 & 0 \\ I_r & I_r & 0 & 0 \\ I_r & I_r & I_r & 0 \\ I_r & I_r & I_r & I_r \end{bmatrix}}^{S} \overbrace{\begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \Delta u_{k+3} \end{bmatrix}}^{\Delta u_{k|4}} + \overbrace{\begin{bmatrix} I_r \\ I_r \\ I_r \\ I_r \end{bmatrix}}^{c} u_{k-1}, \tag{2.72}$$

i.e.,

$$u_{k|4} = S \Delta u_{k|4} + c u_{k-1}. \tag{2.73}$$

Substituting (2.73) into (2.71) gives the PM based on a step response model.

$$y_{k+1|4} = F_4^{\Delta} \Delta u_{k|4} + p_4^{\Delta}, \tag{2.74}$$

where

$$F_4^{\Delta} = F_4 S = \begin{bmatrix} h_1 & 0 & 0 & 0 \\ h_1 + h_2 & h_1 & 0 & 0 \\ h_1 + h_2 + h_3 & h_1 + h_2 & h_1 & 0 \\ h_1 + h_2 + h_3 & h_1 + h_2 + h_3 & h_1 + h_2 & h_1 \end{bmatrix}, \tag{2.75}$$

$$\begin{aligned} p_4^{\Delta} = p_4 + F_4 c u_{k-1} &= \begin{bmatrix} h_3 & h_2 \\ 0 & h_3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{k-2} \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} h_1 \\ h_1 + h_2 \\ h_1 + h_2 + h_3 \\ h_1 + h_2 + h_3 \end{bmatrix} u_{k-1} \\ &= \begin{bmatrix} h_3 & h_1 + h_2 \\ 0 & h_1 + h_2 + h_3 \\ 0 & h_1 + h_2 + h_3 \\ 0 & h_1 + h_2 + h_3 \end{bmatrix} \begin{bmatrix} u_{k-2} \\ u_{k-1} \end{bmatrix}. \end{aligned} \tag{2.76}$$

The PM (2.74) is in so called standard form for use in MPC. One should note that the term p_4^{Δ} is known at each time instant k, i.e., p_4^{Δ} is a function of past and known control inputs, u_{k-1} and u_{k-2}.

Example 2.5 (Alternative step response PM)
Consider the finite impulse response (FIR) model in (2.64) for $k := k+1$, i.e.

$$y_{k+1} = h_1 u_k + h_2 u_{k-1} + h_3 u_{k-2}. \tag{2.77}$$

Subtracting (2.77) from (2.64) gives the step response model

$$y_{k+1} = y_k + h_1 \Delta u_k + h_2 \Delta u_{k-1} + h_3 \Delta u_{k-2}. \tag{2.78}$$

We can write a PM in standard form by writing up the expressions for y_{k+2} and y_{k+3} by using (2.78). This can be written in matrix form. Note that the term p_4^{Δ} takes a slightly different form in this case.

$$p_4^{\Delta} = \begin{bmatrix} I_m \\ I_m \\ I_m \\ I_m \end{bmatrix} y_k + \begin{bmatrix} h_3 & h_2 \\ h_3 & h_2+h_3 \\ h_3 & h_2+h_3 \\ h_3 & h_2+h_3 \end{bmatrix} \begin{bmatrix} \Delta u_{k-2} \\ \Delta u_{k-1} \end{bmatrix}. \tag{2.79}$$

We can now show that (2.79) is identical to the expression in (2.76), by using that y_k is given by the FIR model (2.64). Note however that the last alternative, Equation (2.79), may be preferred because the MPC controller will be off feedback type. To prove equality put (2.79) equal to (2.76) and solve for the y_k term, i.e., show that

$$\begin{bmatrix} I_m \\ I_m \\ I_m \\ I_m \end{bmatrix} y_k = \begin{bmatrix} h_3 & h_1+h_2 \\ 0 & h_1+h_2+h_3 \\ 0 & h_1+h_2+h_3 \\ 0 & h_1+h_2+h_3 \end{bmatrix} \begin{bmatrix} u_{k-2} \\ u_{k-1} \end{bmatrix} - \begin{bmatrix} h_3 & h_2 \\ h_3 & h_2+h_3 \\ h_3 & h_2+h_3 \\ h_3 & h_2+h_3 \end{bmatrix} \begin{bmatrix} \Delta u_{k-2} \\ \Delta u_{k-1} \end{bmatrix}. \tag{2.80}$$

Example 2.6 (PM by solving a Diophantine equation)
Consider the model

$$y_k = -a_1 y_{k-1} + b_0 u_{k-1} + b_1 u_{k-2}. \tag{2.81}$$

where the coefficients are $a_1 = -0.8$, $b_0 = 0.4$ and $b_1 = 0.6$. Hence the $A(z^{-1})$ and $B(z^{-1})$ polynomials in the CARIMA/ARIMAX model (2.39) with delay $d = 0$, are given by

$$A(z^{-1}) = 1 + a_1 z^{-1} = 1 - 0.8 z^{-1}. \tag{2.82}$$
$$B(z^{-1}) = b_0 + b_1 z^{-1} = 0.4 + 0.6 z^{-1}. \tag{2.83}$$

Hence, the $A(z^{-1})$ polynomial is of order $na = 1$ and the $B(z^{-1})$ polynomial is of order $nb = 1$. The noise term in (2.39) is omitted. This will not influence upon the resulting PM when $C = 1$, which is assumed in this example. A prediction horizon $L = 4$ is specified. We now want to solve the Diophantine Equation (2.44) for the unknown polynomials $F_j(z^{-1})$, $E_j(z^{-1})$ and $G_j(z^{-1})$ where $j = 1, 2, 3, 4$. This can be done recursively starting with $E_1 = 1$ and $F_1 = z(1 - \tilde{A})$. We have included a

MATLAB script, **demo_gpcpm.m** *and* **demo_gpcpm2.m**, *for a demonstration of how the coefficients are computed. This gives*

$$\begin{aligned} F_1(z^{-1}) &= f_{10} + f_{11}z^{-1} = 1.8 - 0.8z^{-1}, & (2.84)\\ F_2(z^{-1}) &= f_{20} + f_{21}z^{-1} = 2.44 - 1.44z^{-1}, & (2.85)\\ F_3(z^{-1}) &= f_{30} + f_{31}z^{-1} = 2.952 - 1.9520z^{-1}, & (2.86)\\ F_4(z^{-1}) &= f_{40} + f_{41}z^{-1} = 3.3616 - 2.3616z^{-1}, & (2.87) \end{aligned}$$

$$\begin{aligned} E_1(z^{-1}) &= e_{10} = 1, & (2.88)\\ E_2(z^{-1}) &= e_{20} + e_{21}z^{-1} = 1 + 1.8z^{-1}, & (2.89)\\ E_3(z^{-1}) &= e_{30} + e_{31}z^{-1} + e_{32}z^{-2} = 1 + 1.8z^{-1} + 2.44z^{-2}, & (2.90)\\ E_4(z^{-1}) &= e_{40} + e_{41}z^{-1} + e_{42}z^{-2} + e_{43}z^{-3} & \\ &= 1 + 1.8z^{-1} + 2.44z^{-2} + 2.952z^{-3}, & (2.91) \end{aligned}$$

$$\begin{aligned} G_1(z^{-1}) &= g_{10} + g_{11}z^{-1} = 0.4 + 0.6z^{-1}, & (2.92)\\ G_2(z^{-1}) &= g_{20} + g_{21}z^{-1} + g_{22}z^{-2} = 0.4 + 1.32z^{-1} + 1.08z^{-2}, & (2.93)\\ G_3(z^{-1}) &= g_{30} + g_{31}z^{-1} + g_{32}z^{-2} + g_{33}z^{-3} & \\ &= 0.4 + 1.32z^{-1} + 2.056z^{-2} + 1.464z^{-3}, & (2.94)\\ G_4(z^{-1}) &= g_{40} + g_{41}z^{-1} + g_{42}z^{-2} + g_{43}z^{-3} + g_{44}z^{-4} & \\ &= 0.4 + 1.32z^{-1} + 2.056z^{-2} + 2.6448z^{-3} + 1.7712z^{-4}. & (2.95) \end{aligned}$$

Only the F_j and G_j polynomials are used to form the PM. The F_j polynomials are used in the vector p_4^Δ. The G_j polynomials are used in both p_4^Δ and F_L^Δ. The PM is given by

$$y_{k+1|4} = F_4^\Delta \Delta u_{k|4} + p_4^\Delta, \tag{2.96}$$

where

$$y_{k+1|4} = \begin{bmatrix} y_{k+1}\\ y_{k+2}\\ y_{k+3}\\ y_{k+4} \end{bmatrix}, \Delta u_{k|4} = \begin{bmatrix} \Delta u_k\\ \Delta u_{k+1}\\ \Delta u_{k+2}\\ \Delta u_{k+3} \end{bmatrix}, F_4^\Delta = \begin{bmatrix} 0.4 & 0 & 0 & 0\\ 1.32 & 0.4 & 0 & 0\\ 2.056 & 1.32 & 0.4 & 0\\ 2.6448 & 2.056 & 1.32 & 0.4 \end{bmatrix} \tag{2.97}$$

and

$$p_4^\Delta = \begin{bmatrix} 0.6\\ 1.08\\ 1.464\\ 1.7712 \end{bmatrix} \Delta u_{k-1} + \begin{bmatrix} 1.8\\ 2.44\\ 2.952\\ 3.3616 \end{bmatrix} y_k + \begin{bmatrix} -0.8\\ -1.44\\ -1.952\\ -2.3616 \end{bmatrix} y_{k-1}. \tag{2.98}$$

This can also be written as, i.e. by using that $\Delta u_{k-1} = u_{k-1} - u_{k-2}$

$$p_4^\Delta = \begin{bmatrix} -0.8 & 1.8 & -0.6 & 0.6\\ -1.44 & 2.44 & -1.08 & 1.08\\ -1.952 & 2.952 & -1.464 & 1.464\\ -2.3616 & 3.3616 & -1.7712 & 1.7712 \end{bmatrix} \begin{bmatrix} y_{k-1}\\ y_k\\ u_{k-2}\\ u_{k-1} \end{bmatrix}. \tag{2.99}$$

Example 2.7 (PM by using Proposition 3.4 in Chapter 6)
Consider the state space model matrices

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -a_1 \end{bmatrix}, \; B = \begin{bmatrix} b_0 \\ b_1 - a_1 b_0 \end{bmatrix}, \; D = \begin{bmatrix} 1 & 0 \end{bmatrix}, \tag{2.100}$$

where the coefficients are $a_1 = -0.8$, $b_0 = 0.4$ and $b_1 = 0.6$. This is the state space equivalent to the polynomial model (2.81). A PM is now constructed by first obtaining the ESSM matrices $\tilde{A}_L$ and $\tilde{B}_L$ with $L = 4$ and $J = 2$ as described in Chapter 6. The PM is then constructed by using Proposition 3.4. The results are

$$y_{k+1|4} = F_4^{\Delta} \Delta u_{k|4} + p_4^{\Delta}, \tag{2.101}$$

where

$$y_{k+1|4} = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ y_{k+4} \end{bmatrix}, \Delta u_{k|4} = \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \Delta u_{k+3} \end{bmatrix}, F_4^{\Delta} = \begin{bmatrix} 0.4 & 0 & 0 & 0 \\ 1.32 & 0.4 & 0 & 0 \\ 2.056 & 1.32 & 0.4 & 0 \\ 2.6448 & 2.056 & 1.32 & 0.4 \end{bmatrix} \tag{2.102}$$

and

$$p_4^{\Delta} = y_{k-3|4} + \begin{bmatrix} 0 & 1 & 1 & 1.8 \\ 0 & 0 & 1 & 2.44 \\ 0 & 0 & 0 & 2.952 \\ 0 & 0 & 0 & 2.3616 \end{bmatrix} \Delta y_{k-3|4} + \begin{bmatrix} 0 & 0 & 0.6 \\ 0 & 0 & 1.08 \\ 0 & 0 & 1.464 \\ 0 & 0 & 1.7712 \end{bmatrix} \Delta u_{k-3|3}. \tag{2.103}$$

The term p_4^{Δ} is identical to those obtained by solving the Diophantine equation as in the GPC method, i.e., as given in Example 2.7 and equations (2.98) and (2.99). To see this we write Equation (2.103) as

$$p_4^{\Delta} = y_{k-3|4} + \begin{bmatrix} 1 & 1 & 1.8 \\ 0 & 1 & 2.44 \\ 0 & 0 & 2.952 \\ 0 & 0 & 2.3616 \end{bmatrix} \overbrace{\begin{bmatrix} y_{k-2} - y_{k-3} \\ y_{k-1} - y_{k-2} \\ y_k - y_{k-1} \end{bmatrix}}^{\Delta y_{k-2|3}} + \begin{bmatrix} 0.6 \\ 1.08 \\ 1.464 \\ 1.7712 \end{bmatrix} \Delta u_{k-1}. \tag{2.104}$$

which is identical to (2.98). Note that by choosing $2 < J \leq 4$ gives a different expression for the term p_4^{Δ}, i.e., the parameters in (2.103) may be different. However, the structure is the same. Note that the formulation of p_4^{Δ} as in (2.103) is attractive for steady state analysis, where we must have that $p_4^{\Delta} = y_{k-3|4}$. This example is implemented in the MATLAB script **demo_gpcpm2.m**.

2.4 Computing the MPC control

We will in this section derive the unconstrained MPC control. This is with the presented notation sometimes referred to as unconstrained EMPC control.

2.4.1 Computing actual control variables

In order to compute the MPC control for FIR models we can consider the Prediction model (PM) in Example (2.3) with the following control objective

$$J_k = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + u_{k|L}^T P u_{k|L}. \tag{2.105}$$

Substituting the PM (2.70) into (2.105) gives

$$\frac{\partial J_k}{\partial u_{k|L}} = 2F_L^T Q (y_{k+1|L} - r_{k+1|L}) + 2P u_{k|L} = 0. \tag{2.106}$$

This gives the future optimal controls

$$u_{k|L}^* = (F_L^T Q F_L + P)^{-1} F_L^T Q (r_{k+1|L} - p_L). \tag{2.107}$$

Only the first control vector, u_k, in the vector $u_{k|L}^*$ is used for control.

Note that the control objective (2.105) can be written in the convenient standard form

$$J_k = u_{k|L}^T H u_{k|L} + 2f^T u_{k|L} + J_0, \tag{2.108}$$

where

$$H = F_L^T Q F_L + P, \tag{2.109}$$
$$f = F_L^T Q (p_L - r_{k+1|L}), \tag{2.110}$$
$$J_0 = (p_L - r_{k+1|L})^T Q (p_L - r_{k+1|L}). \tag{2.111}$$

Note that the term p_L is known, and for state space models given as in (2.10). Hence, (2.106) and (2.107)) can simply be expressed as

$$\frac{\partial J_k}{\partial u_{k|L}} = 2H u_{k|L} + 2f = 0, \tag{2.112}$$
$$u_{k|L}^* = -H^{-1} f. \tag{2.113}$$

This algorithm in connection with the state estimate (2.13) and the use of the common constraints is sometimes referred to as the EMPC$_1$ algorithm. Constraints are discussed later in this report.

2.4.2 Computing control deviation variables

In order to e.g., compute the MPC control for step response models we can consider the Prediction model (PM) in Example (2.4) or (2.5) with the following control objective

$$J_k = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + u_{k|L}^T P u_{k|L} + \Delta u_{k|L}^T R \Delta u_{k|L} \tag{2.114}$$

The problem is to minimize (2.114) with respect to $\Delta u_{k|L}$. Recall that the PM is given as, and that $u_{k|L}$ and $\Delta u_{k|L}$ are related as

$$y_{k+1|L} = F_L^{\Delta} \Delta u_{k|L} + p_L^{\Delta}, \tag{2.115}$$
$$u_{k|L} = S \Delta u_{k|L} + c u_{k-1}. \tag{2.116}$$

Substituting the PM (2.74) into (2.114) gives

$$\frac{\partial J_k}{\partial u_{k|L}} = 2F_L^{\Delta T} Q(F_L^{\Delta T} \Delta u_{k+1|L} + p_L^{\Delta} - r_{k+1|L}) + S^T P(S\Delta u_{k|L} + cu_{k-1}) + 2R\Delta u_{k|L} = 0. \tag{2.117}$$

This gives the future optimal controls

$$\Delta u_{k|L}^* = (F_L^{\Delta T} Q F_L^{\Delta} + R + S^T P S)^{-1} [F_L^{\Delta T} Q(r_{k+1|L} - p_L^{\Delta}) + S^T P c u_{k-1}]. \tag{2.118}$$

Only the first control change, Δu_k, in the vector $\Delta u_{k|L}^*$ is used for control. The actual control to the process is taken as $u_k = \Delta u_k + u_{k-1}$.

Note that the control objective (2.114) can be written in the convenient standard form

$$J_k = \Delta u_{k|L}^T H \Delta u_{k|L} + 2f^T \Delta u_{k|L} + J_0, \tag{2.119}$$

where

$$H = F_L^{\Delta T} Q F_L^{\Delta} + R + S^T P S, \tag{2.120}$$

$$f = F_L^{\Delta T} Q(p_L^{\Delta} - r_{k+1|L}) + S^T P c u_{k-1}, \tag{2.121}$$

$$J_0 = (p_L^{\Delta} - r_{k+1|L})^T Q(p_L^{\Delta} - r_{k+1|L}) + u_{k-1}^T c^T P c u_{k-1}. \tag{2.122}$$

Note that the term p_L^{Δ} is known, and for state space models given as in (2.17). Hence, (2.117) and (2.118)) can simply be expressed as

$$\frac{\partial J_k}{\partial \Delta u_{k|L}} = 2H\Delta u_{k|L} + 2f = 0, \tag{2.123}$$

$$\Delta u_{k|L}^* = -H^{-1} f. \tag{2.124}$$

One motivation of computing control deviation variables $\Delta u_{k|L}$ is to obtain offset-free control in a simple manner, i.e., to obtain integral action such that $y = r$ in steady state. The control $u_k = \Delta u_k + u_{k-1}$ with Δu_k computed as in (2.118)) gives offset-free control if the control weighting matrix $P = 0$. Another important advantage of computing $\Delta u_{k|L}$ and choosing $P = 0$ is to avoid practical problems with non-zero mean variables. This algorithm in connection with the state estimate (2.13) and the use of the common constraints is sometimes referred to as the EMPC$_2$ algorithm. Constraints are discussed later in this report.

2.5 Discussion

There exist a large number of MPC algorithms. However, as far as we now, all MPC methods which are based on linear models and linear quadratic objectives fit into the EMPC framework as special cases. We will give a discussion of some differences and properties with MPC algorithms below.

- MPC methods may be different because different linear models are used to compute the predictions of the future outputs. Some algorithms are using

FIR models. Other algorithms are using step response models, ARMAX models, CARIMA models and so on. The more general approach is to use state space models, as in the EMPC method. Any linear dynamic model can be transformed to a state space model.

- Most algorithms are using quadratic control objective functions, J_k. However, different methods may be different because they are using different quadratic functions with different weights.

- All MPC methods are in one or another way dependent upon the present state, x_k. MPC methods are usually different because different state observers are used. The default choice in the EMPC algorithm is the state estimate given in (2.13). It is interesting to note that the prediction model used by the GPC method is obtained by putting J equal to the minimum value. Some algorithms are using very simple observers as in the DMC method which is based upon step response models. See Example 2.5 and the PM (2.79). Other algorithms are based upon a Kalman filter for estimating the state, e.g., the method by Muske and Rawlings (1993). Note that these MPC algorithms are equivalent to LQ/LQG control in the unconstrained case. A promising choice is to use the Kalman filter identified by the DSR method.

- Some MPC methods are computing the actual control variables, u_k, and other are computing control deviation variables, Δu_k. A lot of practical problems due to non-zero offset values on the variables may be avoided by computing deviation variables, Δu_k. This is also a simple way of obtaining offset-free control, i.e., so that $y = r$ in steady state and where r is a specified reference.

- Some algorithms can not be used to control unstable systems or systems which contains integrators. This is the case for the algorithms which are based on FIR and step response models. The reason for this is because FIR models cannot represent unstable systems. An algorithm which is based on a state space model can be used to control unstable systems. The EMPC algorithm works similarly on stable as well as on unstable systems. However, it may be important to tune the prediction horizon and the weighting matrices properly.

- Like it or not. We have that the MPC controller in the unconstrained case are identical to the classical LQ and LQG controllers. The only difference is in the way the controller "gain" are computed. Like MPC, the LQ and LQG controllers are perfect to control multivariable systems with cross-couplings.

- There are however some differences between MPC and LQ/LQG controllers. There is one difference between the MPC method and the LQ/LQG method in the way the computations are done. Furthermore, in the unconstrained case, the LQ/LQG method is more efficient when the prediction horizon is large, e.g., infinite. An infinite horizon MPC problem may however be formulated as a finite horizon problem, by using a special weighting matrix for the final deviation $y_{k+1+L} - r_{k+1+L}$. However, this weighting matrix is a solution to an infinite horizon LQ problem. Infinite prediction horizon problems are important in order to control unstable processes.

- The main advantage and the motivation of using an MPC controller is the ability to cope with constraints on the process variables in a simple and efficient way. Linear models, linear constraints an a linear quadratic objective results in a QP-problem, which can be efficiently solved. See Section 3 for details.
- Another application of the MPC algorithm is to use it for operator support. The MPC algorithms can be used to only compute and to propose the future process controls. The operators can then use the control, u_k^*, which is proposed by the MPC algorithm, as valuable information for operating the process. This is not possible by conventional controllers, e.g., like the PID-controller.
- The DSR algorithm in Di Ruscio (1996), (1997f) and older work, is available for building both the prediction model and the state observer to be used in MPC algorithms. The DSR algorithm are in use with some commercial MPC controllers. The use of the DSR method for MPC will improve effectiveness considerably. First of all due to the time which is spend on model building may be reduced considerably.
- Furthermore, the DSR theory and notation is valuable in its own right in order to formulate the solution to a relatively general MPC problem.
- Furthermore, the subspace matrices $\tilde{A}_L$ and $\tilde{B}_L$ identified by DSR can be used directly to construct a PM as presented in Section 6. Hence, the model (A, B, D) is not necessarily needed. However, we recommend to use the model (A, B, D).
- The unconstrained MPC technique results in general in a control, u_k, which consists of a feedback from the present state vector, x_k, and feed-forward from the specified future references, $r_{k+1|L}$. This is exactly the same properties which results from the Linear Quadratic (LQ) tracking regulator. It can be shown that they indeed are equivalent in the unconstrained case. The feed-forward properties of the control means that the actual control can start to move before actual changes in the reference find place.

References

Di Ruscio, D. and B. Foss (1998). On Model Based Predictive Control. *The 5th IFAC Symposium on Dynamics and Control of Process Systems*, Corfu, Greece, June 8-10,1998.

Di Ruscio, D. (1997). Model Based Predictive Control: An extended state space approach. In proceedings of *the 36th Conference on Decision and Control 1997*, San Diego, California, December 6-14.

Di Ruscio, D. (1997b). Model Predictive Control and Identification: A linear state space model approach. In proceedings of *the 36th Conference on Decision and Control 1997*, San Diego, California, December 6-14.

Di Ruscio, D. (1997c). Model Predictive Control and Identification of a Thermo Mechanical Pulping Plant: A linear state space model approach. *Control 1997*, Sydney Australia, October 20-22, 1997.

Di Ruscio, D. (1997d). Model Based Predictive Control: An extended state space approach. *Control 1997*, Sydney Australia, October 20-22, 1997.

Di Ruscio, D. (1997e). A Method for Identification of Combined Deterministic Stochastic Systems. In *Applications of Computer Aided Time Series Modeling*. Editor: Aoki. M. and A. Hevenner, Lecture Notes in Statistics Series, Springer Verlag, 1997, pp. 181-235.

Di Ruscio, D. (1996). Combined Deterministic and Stochastic System Identification and Realization: DSR-a subspace approach based on observations. *Modeling, Identification and Control*, vol. 17, no.3.

Muske, K. R. and J. B. Rawlings (1993). Model predictive control with linear models. *AIChE Journal*, Vol. 39, No. 2, pp. 262-287.

Chapter 3

Unconstrained and constrained optimization

3.1 Unconstrained optimization

3.1.1 Basic unconstrained MPC solution

The prediction models which are used in connection with linear MPC can be written in a so called standard form. Consider this standard prediction model formulation, i.e.,

$$y = Fu + p. \tag{3.1}$$

Similarly, the control objective which is used in linear MPC can be written in matrix form. Consider here the following control objective

$$J = (y - r)^T Q (y - r) + u^T P u. \tag{3.2}$$

The derivative of J with respect to u is given by

$$\begin{aligned} \frac{\partial J}{\partial u} &= 2F^T Q (Fu + p - r) + 2Pu \\ &= 2(F^T Q F + P)u + 2F^T Q (p - r). \end{aligned} \tag{3.3}$$

Solving $\frac{\partial J}{\partial u} = 0$ gives the solution

$$u^* = -(F^T Q F + P)^{-1} F^T Q (p - r). \tag{3.4}$$

The weighting matrices Q and P must be chosen to ensure a positive definite Hessian matrix. The Hessian is given from the second derivative of the control objective, J, with respect to u, i.e.,

$$H = \frac{\partial^2 J}{\partial u^T \partial u} = \frac{\partial^2 J}{\partial u^2} = F^T Q F + P > 0, \tag{3.5}$$

in order to ensure a unique minimum. Note that the second derivative (3.5) is defined as the Hessian matrix, which is symmetric.

3.1.2 Line search and conjugate gradient methods

Consider the general unconstrained optimization problem

$$x^* = \arg \min_{x \in \mathbb{R}^n} f(x). \tag{3.6}$$

This is the same problem one deals with in system identification and in particular the prediction error methods. Textbooks on the topic often describing a Newton method for searching for the minimum, in order to give valuable insight into the problem. The Newton's method for solving the general non-linear unconstrained optimization problem (3.6) is discussed in Section 12, and is not reviewed in detail here.

We will in this section concentrate on a method which only is dependent on functional evaluations $f = f(x)$. A very basic iterative process for solving (3.6) which includes many methods as special cases are

$$x_{i+1} = x_i - \alpha H_i^{-1} g_i, \tag{3.7}$$

where $g_i = \frac{\partial f(x)}{\partial x} \mid_{x_i} \in \mathbb{R}^n$ is the gradient and α is a line search parameter chosen to minimize $f(x_{i+1})$ with respect to α. If $H_k = \frac{\partial^2 f}{\partial x^T \partial u} \mid_{x_i}$ is the Hessian then we obtain Newton's method, which have 2nd order convergence properties near the minimum. If also $\alpha = 1$ then we have the Newton-Raphson's method. The problem of computing the Hessian and the inverse at each iteration is time consuming and may be ineffective when only functional evaluations $f = f(x)$ is available in order to obtain numerical approximations to the Hessian. A simple choice is to put $H_i = I$. This is equivalent to the steepest descent method, which (only) have 1st order convergence near a minimum. Hence, it would be of interests to obtain a method whit approximately the same convergence properties as Newton's method but which at the same time does not depend upon the inverse of the Hessian. Consider the iterative process

$$x_{i+1} = x_i + \alpha d_i, \tag{3.8}$$

where d_i is the search direction. Hence, for the steepest descent method we have $d_i = -g_i$ and for Newtons method we have $d_i = -H_i^{-1} g_i$.

A very promising method for choosing the search direction,d_i, is the Conjugate Gradient Method (CGM). This method is discussed in detail Section 13 for quadratic problems and linear least squares problems. The CGM can be used for non-quadratic NP problems. The key is to use a line search strategy for α_k instead of the "analytical" expression which is used for quadratic and least squares problems. The quadratic case is discussed in detail in Section 13. See also Luenberger (1984) Ch. 8.6.

3.2 Constrained optimization

Details regarding optimization of general non-linear functions subject to general possibly non-linear inequality and equality constraints are not the scope of this report. However, some definitions are useful in connection with linear MPC which leads to QP problems. For this reason, these definitions have to be pointed out.

3.2.1 Basic definitions

Consider the optimization (Nonlinear Programming (NP)) problem

$$\begin{aligned} &\min_x J(x), \\ &\text{subject to} \;\; h(x) = 0, \\ &\qquad\qquad\;\;\, g(x) \leq 0, \\ &\qquad\qquad\;\;\, x \in \Omega, \end{aligned} \tag{3.9}$$

where $J(x)$ is a scalar objective, $g(x) \leq 0 \in \mathbb{R}^p$ is the inequality constraint and $h(x) = 0 \in \mathbb{R}^m$ is the equality constraint, and in general $x \in \mathbb{R}^n$. The constraints $g(x) \leq 0$ and $h(x) = 0$ are referred to as **functional constraints** while the constraint $x \in \Omega$ is a **set constraint**. The set constraint is used to restrict the solution, x, to be in the interior of Ω, which is a part of $\mathbb{R}^n$. In practice, usually this is equivalent to specifying lower and upper bounds on x, i.e.,

$$x_{LOW} \leq x \leq x_{UP}. \tag{3.10}$$

Lower and upper bounds as in (3.10) can be treated as linear inequality constraints and written in the standard form, $Ax \leq b$.

Definition 3.1 (Feasible (mulig) solution)
A point/vector $x \in \Omega$ which satisfies all the functional constraints $g(x) \leq 0$ and $h(x) = 0$ is said to be a **feasible** *solution.*

Definition 3.2 (Active and inactive constraints)
An inequality constraint $g_i(x) \leq 0$ is said to be active at a feasibile point x if $g_i(x) = 0$, and inactive at x if $g_i(x) \leq 0$.

Note that in studying the optimization problem (3.9) and the properties of a (local) minimum point x we only have to take attention to the active constraints. This is important in connection with inequality functional constraints.

There exist several reliable software algorithms for solving the constrained NP problem (3.9). We will here especially mention the **nlpq** FORTRAN subroutine by Schittkowski (1984) which is found to work on a wide range of problems. This is a so called Sequential Quadratic Programming (SQP) algorithm. At each iteration (or some of the iterations), the function, $f(x)$, is approximated by a quadratic function and the constraints, $h(x)$ and $g(x)$, are approximated with linearized functions. Hence, a Quadratic Programming (QP) problem is solved at each step. This is referred to as *recursive quadratic programming* or to as an SQP method. QP problems is not only importaint for solving NP problems, but most practical MPC methods are using a QP method. The QP solver (subroutine **ql0001**) in **nlpq** is found very promising compared to the MATLAB function **quadprog**. Hence, the QP method needs a closer discussion. This is the topic of the next chapter.

3.2.2 Lagrange problem and first order necessary conditions

We start this section with a definition.

Definition 3.3 (regular point) *Let x be a feasibile solution, i.e., a point such that all the constraints is satisfied, i.e.,*

$$h(x) = 0 \text{ and } g(x) \leq 0. \tag{3.11}$$

Furthermore let J be the set of active conditions in $g(x) \leq 0$, i.e., J is the set of indices j such that $g_j(x) = 0$. Then x is said to be a **regular point** *of the constraints (3.11) if all the gradients $\frac{\partial h_i(x)}{\partial x}$ $\forall\ i = 1, \ldots, m$ and $\frac{\partial g_j(x)}{\partial x}$ $\forall\ j \in J$ are linearly independent.*

This means that a regular point, x, is a point such that all the gradients of the active constraints are linearly independent. We does not concentrate on inactive inequality constraints.

Corresponding to the NP problem (3.9) we have the so called lagrange function

$$L(x) = J(x) + \lambda^T h(x) + \mu^T g(x), \tag{3.12}$$

where $\lambda \in \mathbb{R}^p$, and $\mu \in \mathbb{R}^m$ are denoted Lagrange vectors. First order necessary conditions (Kuhn-Tucker conditions) for a minimum is given in the following lemma.

Lemma 3.1 (Kuhn-Tucker conditions for optimality) *Assume that x is a (local) minimum point for the NP problem (3.9), and that x is a* **regular point** *for the constraints. Then there exists vectors $\lambda \in \mathbb{R}^m$, and $\mu \geq 0 \in \mathbb{R}^p$ such that*

$$\begin{aligned} \frac{\partial L(x)}{\partial x} &= \frac{\partial J(x)}{\partial x} + (\frac{\partial h^T(x)}{\partial x})\lambda + (\frac{\partial g^T(x)}{\partial x})\mu \\ &= \frac{\partial J(x)}{\partial x} + (\frac{\partial h(x)}{\partial x})\lambda + (\frac{\partial g(x)}{\partial x})\mu = 0, \end{aligned} \tag{3.13}$$

$$\frac{\partial L(x)}{\partial \lambda} = h(x) = 0, \tag{3.14}$$

$$\mu \frac{\partial L(x)}{\partial \mu} = \mu^T g(x) = 0. \tag{3.15}$$

Since $\mu \geq 0$ only active constraints, $g_i = 0$ for some indices i, will influence on the conditions. Remark that the Lagrange multipliers, μ_i, $\forall\ i = 1, \ldots, p$ in μ should be constructed/computed such that $\mu_i = 0$ if $g_i(x) \leq 0$ (inactive constraint) and $\mu_i > 0$ if $g_i(x) = 0$ (active constraint). In other words, (3.13) holds if $\mu_i = 0$ when $g_i(x) \neq 0$.

Example 3.1 (Kuhn-Tucker conditions for QP problem)
Consider the QP problem, i.e., minimize

$$J(x) = x^T H x + 2f^T x, \tag{3.16}$$

with respect to x, and subject to the equality and inequality constraints

$$h(x) = A_1 x - b_1 = 0 \text{ and } g(x) = A_2 x - b_2 \leq 0. \tag{3.17}$$

The Lagrange function is

$$L(x) = x^T H x + 2f^T x + \lambda^T (A_1 x - b_1) + \mu^T (A_2 x - b_2). \tag{3.18}$$

The 1st order (Kuhn-Tucker) conditions for a minimum is

$$\frac{\partial L(x)}{\partial x} = 2Hx + 2f + A_1^T\lambda + A_2^T\mu = 0, \tag{3.19}$$

$$\frac{\partial L(x)}{\partial \lambda} = A_1x - b_1 = 0, \tag{3.20}$$

$$\mu^T\frac{\partial L(x)}{\partial \mu} = \mu^T(A_2x - b_2) = 0. \tag{3.21}$$

Here μ is (as usual) constructed to only affect constraints in $g(x) = A_2x - b_2 \leq 0$ which are active.

3.3 Quadratic programming

Quadratic Programming (QP) problems usually assumes that the objective to be minimized is written in a standard form (for QP). The control objective (3.2) with the PM (3.1) can be written in this standard form as follows

$$J = u^THu + 2f^Tu + J_0, \tag{3.22}$$

where

$$H = F^TQF + P, \tag{3.23}$$

$$f = F^TQ(p - r), \tag{3.24}$$

$$J_0 = (p - r)^TQ(p - r). \tag{3.25}$$

One advantage of (3.22) is that it is very simple to derive the gradient (3.3) and the solution (3.4). We have

$$\frac{\partial J}{\partial u} = 2Hu + 2f, \tag{3.26}$$

and the solution

$$u^* = -H^{-1}f. \tag{3.27}$$

Note that the constant term, J_0, in (3.25) is not needed in the solution.

One of the main advantages and motivation behind MPC is to treat process constraints in a simple and efficient way. The common constraints as control amplitude constraints, control rate of change constraints and process output constraints can all be represented with a single linear inequality, i.e.,

$$Au \leq b, \tag{3.28}$$

where A and b are matrices derived from the process constraints. The problem of minimizing a quadratic objective (3.22) with respect to u subject to inequality constraints as in (3.28) is a QP problem.

The QP problem can be solved in MATLAB by using the **quadprog.m** function in the optimization toolbox. The syntax is as follows

```
>> u=quadprog(H,f,A,b);
```

3.3.1 Equality constraints

We will in this section study the QP problem with equality constraints in general. Given the objective

$$J = u^T H u + 2f^T u, \tag{3.29}$$

where $H \in \mathbb{R}^{n\times n}$, $f \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$. Consider the QP problem

$$\begin{aligned} &\min_u J(u), \\ &\text{subject to } \; Au = b, \end{aligned} \tag{3.30}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. This problem can be formulated as an equivalent optimization problem without constraints by the method of Lagrange multipliers. Consider the Lagrange function

$$L(u) = u^T H u + 2f^T u + \lambda^T (Au - b). \tag{3.31}$$

The problem (3.29) is now equivalent to minimize (3.31) with respect to u and λ. The vector λ is a vector with Lagrange multipliers. The first order necessary conditions for this problem is given by putting the gradients of the problem equal to zero, i.e.,

$$\frac{\partial J}{\partial u} = 2Hu + 2f + A^T\lambda = 0, \tag{3.32}$$

$$\frac{\partial J}{\partial \lambda} = Au - b = 0. \tag{3.33}$$

Note that this can be written as a system of linear equations in the unknown vectors u and $\bar{\lambda} = \frac{1}{2}\lambda$. We have

$$\begin{bmatrix} H & A^T \\ A & 0_{m\times m} \end{bmatrix} \begin{bmatrix} u \\ \bar{\lambda} \end{bmatrix} = \begin{bmatrix} -f \\ b \end{bmatrix}. \tag{3.34}$$

It can be proved, Luenberger (1984), that the matrix

$$M = \begin{bmatrix} H & A^T \\ A & 0_{m\times m} \end{bmatrix}, \tag{3.35}$$

is non-singular if $A \in \mathbb{R}^{m\times n}$ has rank m and $H \in \mathbb{R}^{n\times n}$ is positive definite. Hence, (3.34) has a unique solution in this case. An efficient factorization method (such as LU decomposition which utilize the structure of M) is recommended to solve the set of linear equations. Another promising technique is the Conjugate Gradient Method (CGM). Note that the system of linear equations (3.34) simply can be solved by the CGM (PLS1) algorithm in Chapter 13. However, for theorectical considerations note that (3.34) can be solved analytically. We have,

$$\bar{\lambda} = -(AH^{-1}A^T)^{-1}(AH^{-1}f + b), \tag{3.36}$$

$$u = -H^{-1}(f + A^T\bar{\lambda}). \tag{3.37}$$

This can be proved by first solving (3.32) for u, which gives (3.37). Second substitute u given by (3.37) into (3.33) and solve this equation for $\bar{\lambda}$. the resulting equation for the Lagrange multipliers is then given in (3.36).

Note that the number of equality constraints, m, must be less or equal to the number of parameters, n, in u. Hence, we restrict the discussion to equality constraints where $m \leq n$. If $m > n$ then we can work with the normal equations, i.e., $A^TAu \leq A^Tb$ instead of $Au \leq b$.

3.3.2 Inequality constraints

Most linear MPC methods leads to a QP (or LP) problem with inequality constraints. The QP problem with inequality constraints $Au \leq b$ is solved iteratively. However, one should note that the number of iterations are finite and that the solution is guaranteed to converge if the QP problem is well formulated and a solution is feasible.

Consider the QP problem with inequality constraints

$$\begin{array}{l} \min_u J(u), \\ \text{subject to } \ Au \leq b, \end{array} \tag{3.38}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Note that we in this case can allow the number of inequality constraints, m, to be greater than the number of parameters, n, in x.

The QP problem (3.38) is usually always solved by an active set method. The procedure is described in the following. Define

$$A = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{bmatrix}, \ b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}. \tag{3.39}$$

Hence, we have m inequalities of the form

$$a_i^T u \leq b_i \ \forall \ i = 1, \dots, m. \tag{3.40}$$

The inequality (3.40) is treated as active and the corresponding QP problem with an equality constraint is solved by e.g., the Lagrange method. Consider the Lagrange function

$$L(u, \lambda_i) = u^T H u + 2f^T u + \lambda_i^T (a_i^T u - b_i). \tag{3.41}$$

Notice that λ_i is a scalar. The 1st order conditions for a minimum is

$$\frac{\partial L(u, \lambda_i)}{\partial u} = 2Hu + 2f + a_i \lambda_i = 0, \tag{3.42}$$

$$\frac{\partial L(u, \lambda_i)}{\partial \lambda_i} = a_i^T u - b_i = 0, \tag{3.43}$$

Solving (3.42) for u gives $u = -H^{-1}(f + a_i \frac{1}{2} \lambda_i)$. Substituting this into (3.43) and solving this for the Lagrange multiplier, λ_i, gives

$$\bar{\lambda}_i = -\frac{a_i^T H^{-1} f + b_i}{a_i^T H^{-1} a_i}, \tag{3.44}$$

where $\bar{\lambda}_i = \frac{1}{2}\lambda_i$. Assume that the Lagrange multiplier of this problem is computed to be negative, i.e., $\lambda_i < 0$. It can then be shown that the inequality must be inactive and $a_i^T u \leq b$ can be dropped from the set of m inequalities. We are therefore putting

$\lambda_i = 0$ for this inactive constraints. If the Lagrange multiplier is found to be positive, $\lambda_i \geq 0$, then the inequality is active. This procedure leads to the vector

$$\lambda^* = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_i \\ \vdots \\ \lambda_m \end{bmatrix}, \tag{3.45}$$

which corresponds to the Lagrange problem

$$L(u) = u^T H u + 2f^T u + \lambda^T (Au - b). \tag{3.46}$$

The point is now that the vector, λ, of Lagrange multipliers in (3.46) is known. Hence, positive Lagrange multipliers (elements) in λ^* corresponds to active inequality constraints and Lagrange multipliers which are zero corresponds to inactive inequality constraints. We have in (3.46) treated all the inequality constraints as active, but, note that $\lambda^* = \lambda$ is known at this stage, and that inequalities which are inactive (and treated as active) does not influence upon the problem because the corresponding Lagrange multiplier is zero. Hence, the solution is given by

$$u^* = -H^{-1}(f + A^T \frac{1}{2}\lambda^*) = -H^{-1}(f + A^T \bar{\lambda}^*) \tag{3.47}$$

where the Lagrange multipliers in $\bar{\lambda}^* = \frac{1}{2}\lambda^*$ is computed by an active set procedure as described above. Hence, each element $\bar{\lambda}_i$ in the vector $\bar{\lambda}^* \in \mathbb{R}^n$ is found by solving a QP problem with one equality constraint ($a_i^T u = b_i$). The reason for which the Lagrange multipliers of the inequalities must be positive, i.e., $\bar{\lambda}^* = \frac{1}{2}\lambda^* \geq 0$, follows from the Kuhn-Tucker optimality conditions in Lemma 3.1. The above active set procedure is illustrated in the MATLAB function **qpsol1.m**.

A more reliable routine is the FORTRAN **ql0001** subroutine which is a part of the **nlpq** sowtware by Schittkowski (19984). A C-language function, **qpsol**, is build on this and is found very promising in connection with MPC.

3.3.3 Examples

Example 3.2 (Scalar QP problem)
Consider the objective

$$J = \frac{1}{2}(u-1)^2 = \frac{1}{2}u^2 - u + \frac{1}{2}, \tag{3.48}$$

and the inequality constraint

$$u \leq \frac{3}{4}. \tag{3.49}$$

This problem is a standard QP problem with $J_0 = \frac{1}{2}$, $H = \frac{1}{2}$, $f = -\frac{1}{2}$, $A = 1$ and $b = \frac{3}{4}$. The minimum of the unconstrained problem is clearly $u = 1$. The minimum of J with respect to u subject to the constraint is therefore simply $u^* = \frac{3}{4}$.

As an illustration, let us now solve the problem with the active set procedure. Treating the inequality as an equality condition we obtain the Lagrange function

$$L(u,\lambda) = \frac{1}{2}u^2 - u + \lambda(u - \frac{3}{4}). \tag{3.50}$$

The 1st order necessary conditions for a minimum are

$$\frac{\partial L(u,\lambda)}{\partial u} = u - 1 + \lambda = 0, \tag{3.51}$$

$$\frac{\partial L(u,\lambda)}{\partial \lambda} = u - \frac{3}{4} = 0. \tag{3.52}$$

Solving (3.52) gives $u = \frac{3}{4}$*. Substituting this into (3.51) gives* $\lambda = 1 - u = \frac{1}{4}$*. Since the Lagrange multiplier is positive (*$\lambda \geq 0$*), we conclude that the minimum in fact is* $u^* = \frac{3}{4}$. $\square$

References

Luenberger, David G. (1984) *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company.

Schittkowski, K. (1984) *NLPQ: Design, implementation, and test of a nonlinear programming algorithm.* Institut fuer informatik, Universitaet Stuttgart.

Chapter 4

Introductory examples

Example 4.1 (EMPC$_1$: A simple example)
Consider the SISO 1st order process given by

$$y_{k+1} = ay_k + bu_k, \tag{4.1}$$

and the control problem

$$\min_{u_k} J_1 = (y_{k+1} - r_{k+1})^T Q (y_{k+1} - r_{k+1}) + u_k^T P u_k. \tag{4.2}$$

The process model can in this case be used directly as prediction model. The optimal control input is

$$u_k = -\frac{qb}{p + qb^2}(ay_k - r_{k+1}). \tag{4.3}$$

The closed loop system is described by

$$y_{k+1} = a_{cl} y_k + \frac{qb^2}{p + qb^2} r_{k+1}, \tag{4.4}$$

where the closed loop pole is given by

$$a_{cl} = a(1 - \frac{qb^2}{p + qb^2}) = a\frac{1}{1 + \frac{q}{p}b^2}. \tag{4.5}$$

In steady state we have

$$y = \frac{1}{1 + \frac{p}{q}\frac{1-a}{b^2}} r. \tag{4.6}$$

Hence we have zero steady state offset only in certain circumstances

$$y = r \ \textit{when} \ \begin{cases} a = 1 & (\textit{integrator}) \\ b \to \infty & (\textit{infinite gain}) \\ p = 0 & (\textit{free control}) \\ q \to \infty & \\ r = 0 & \end{cases} \tag{4.7}$$

Note that the ratio $\frac{q}{p}$ can be "tuned" by specifying the closed loop pole a_{cl}. From (4.5) we have

$$\frac{q}{p}b^2 = \frac{a}{a_{cl}} - 1. \tag{4.8}$$

□

Example 4.2 (EMPC$_2$: A simple example)
Consider the SISO 1st order process given by

$$y_{k+1} = ay_k + bu_k + v, \tag{4.9}$$

where v is an unknown constant disturbance, and the control problem

$$\begin{aligned}\min_{u_k} J_2 &= (y_{k+1} - r_{k+1})^T Q(y_{k+1} - r_{k+1}) + \Delta u_k^T P \Delta u_k \\ &= q(y_{k+1} - r_{k+1})^2 + p\Delta u_k^2. \end{aligned} \tag{4.10}$$

where $\Delta u_k = u_k - u_{k-1}$. A prediction model in terms of process input change variables is given by

$$y_{k+1} = p_1(k) + b\Delta u_k, \tag{4.11}$$
$$p_1(k) = y_k + a\Delta y_k, \tag{4.12}$$

where $\Delta y_k = y_k - y_{k-1}$. Note that the PM is independent of the unknown constant disturbance, v, in the state equation. The optimal control input change is given by

$$\Delta u_k^* = -\frac{qb}{p + qb^2}(y_k + a\Delta y_k - r_{k+1}). \tag{4.13}$$

In this case we have zero steady state offset. An argumentation is as follows. Assume that the closed loop system is stable and that both the reference and possibly disturbances are stationary variables. Then, the change variables Δu_k and Δy_k are zero in steady state. Then we also have from the above that $y = r$. It is assumed that $\frac{qb}{p+qb^2} \neq 0$.

The closed loop system is described by

$$\Delta y_{k+1} = a(1 - \frac{qb^2}{p + qb^2})\Delta y_k - \frac{qb^2}{p + qb^2} y_k + \frac{qb^2}{p + qb^2} r_{k+1}, \tag{4.14}$$

which can be written as the following difference equation

$$y_{k+1} = c_1 y_k + c_2 y_{k-1} + c_3 r_{k+1}, \tag{4.15}$$
$$c_1 = 1 + a(1 - \frac{qb^2}{p + qb^2}) - \frac{qb^2}{p + qb^2} = (1 + a)\frac{1}{1 + \frac{q}{p}b^2}, \tag{4.16}$$
$$c_2 = -a(1 - \frac{qb^2}{p + qb^2}) = -a\frac{1}{1 + \frac{q}{p}b^2}, \tag{4.17}$$
$$c_3 = \frac{qb^2}{p + qb^2}, \tag{4.18}$$

or as the state space model

$$\begin{bmatrix} y_k \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ c_2 & c_1 \end{bmatrix} \begin{bmatrix} y_{k-1} \\ y_k \end{bmatrix} + \begin{bmatrix} 0 \\ c_3 \end{bmatrix} r_{k+1}. \tag{4.19}$$

A stability analysis of the closed loop system is given in Example 4.3. □

Example 4.3 (Stability analysis)
Consider the control in Example 4.2. The closed loop system is stable if the roots of the characteristic polynomial

$$\left| zI - \begin{bmatrix} 0 & 1 \\ c_2 & c_1 \end{bmatrix} \right| = z^2 - c_1 z - c_2 = 0, \tag{4.20}$$

are located inside the unit circle in the complex plane. The bi-linear transformation

$$z = \frac{1+s}{1-s}, \tag{4.21}$$

will map the interior of the unit circle into the left complex plane. The stability analysis is now equivalent to check whether the polynomial

$$(1 + c_1 - c_2)s^2 + 2(1 + c_2)s + 1 - c_1 - c_2 = 0, \tag{4.22}$$

or equivalently

$$(2 + 2a(1 - c_3) - c_3)s^2 + 2(1 - a(1 - c_3))s + c_3 = 0, \tag{4.23}$$

has all roots in the left hand side of the complex plane. It can be shown, e.g. by Routh's stability criterion, that the system is stable if the polynomial coefficients are positive. The coefficient $c_3 = \frac{qb^2}{p+qb^2}$ is always positive when $q > 0$ and $p \geq 0$. Hence, we have to ensure that the two other coefficients to be positive.

$$a_0 = 2 + 2a(1 - c_3) - c_3 = \frac{qb^2 + 2(1 + a)p}{qb^2 + p} > 0. \tag{4.24}$$

Hence, $a_0 > 0$ when $a \geq 0$, which is the case for real physical systems. Furthermore, we must ensure that

$$a_1 = 2(1 - a(1 - c_3)) = \frac{2qb^2 + 2(1 - a)p}{qb^2 + p} = 2 - 2a\frac{p}{qb^2 + p} > 0. \tag{4.25}$$

After some algebraic manipulation we find that the closed loop system is stable if

$$\frac{a-1}{a} < \frac{qb^2}{p + qb^2} < \frac{2(1+a)}{2a+1}. \tag{4.26}$$

Note that the lower bound is found from (4.25) and that it is only active when the system is unstable (i.e. when $a > 1$). Furthermore we have from (4.25) that the system is stable if

$$\frac{a-1}{b^2} < \frac{q}{p}. \tag{4.27}$$

Finally we present the eigenvalues of the closed loop system

$$\lambda = \frac{a + 1 \pm \sqrt{a^2 - 2a + 1 - 4ab^2\frac{q}{p}}}{2(1 + b^2\frac{q}{p})}. \tag{4.28}$$

□

Example 4.4 (Unconstrained MPC leads to P-control)
Given a system described by

$$y_{k+1} = y_k + u_k. \tag{4.29}$$

Such a model frequently arise in angular positioning control systems where y_k is the angle of a positioning system. Consider the cost function (control objective)

$$J_k = \sum_{i=1}^{2}((y_{k+i} - r_{k+i})^2 + u_{k+i-1}^2). \tag{4.30}$$

This cost function represents a compromise between a desire to have y_k close to the reference r_k but at the same time a desire to have u_k close to zero, i.e. recognizing that the control u_k is costly.

This problem is equivalent to an unconstrained optimization problem in the unknown controls u_k and u_{k+1}. We can express (4.30) as

$$J_k = (y_k + u_k - r_{k+1})^2 + (y_k + u_k + u_{k+1} - r_{k+2})^2 + u_k^2 + u_{k+1}^2. \tag{4.31}$$

The first order necessary conditions for a minimum is

$$\frac{\partial J_k}{\partial u_k} = 2(y_k + u_k - r_{k+1}) + 2(y_k + u_k + u_{k+1} - r_{k+2}) + 2u_k = 0. \tag{4.32}$$

$$\frac{\partial J_k}{\partial u_{k+1}} = 2(y_k + u_k + u_{k+1} - r_{k+2}) + 2u_{k+1} = 0. \tag{4.33}$$

Condition (4.33) gives

$$u_{k+1} = \frac{1}{2}(r_{k+2} - (y_k + u_k)). \tag{4.34}$$

Substituting (4.34) into condition (4.32) and solving for u_k gives

$$u_k^* = -\frac{3}{5}y_k + \frac{2}{5}r_{k+1} + \frac{1}{5}r_{k+2}. \tag{4.35}$$

Substituting (4.35) into (4.34) gives

$$u_{k+1}^* = -\frac{1}{5}y_k - \frac{1}{5}r_{k+1} + \frac{2}{5}r_{k+2}. \tag{4.36}$$

Hence the optimal (MPC) control, u_k^*, consist of a constant feedback from the output y_k and feed-forward from the specified future reference signals. □

Example 4.5 (Unconstrained MPC leads to P-control)
Consider the same problem as in Example 4.4. We will here derive the solution by using the EMPC (matrix) theory. Given

$$y_{k+1} = y_k + u_k. \tag{4.37}$$

The cost function (control objective) (4.30) can be written in matrix form, i.e.,

$$\begin{aligned} J_k &= \sum_{i=1}^{2}((y_{k+i} - r_{k+i})^2 + u_{k+i-1}^2) \\ &= (y_{k+1|2} - r_{k+1|2})^T Q (y_{k+1|2} - r_{k+1|2}) + u_{k|2}^T P u_{k|2}, \\ &= u_{k|2}^T H u_{k|2} + 2f^T u_{k|2} + J_0 \end{aligned} \tag{4.38}$$

where

$$H = F_2^T Q F_2 + P, \tag{4.39}$$

$$f = F_2^T Q (p_2 - r_{k+1|2}), \tag{4.40}$$

$$p_2 = O_2 A x_k. \tag{4.41}$$

where the weighting matrices are specified as $Q = I_2$ and $P = I_2$. I_2 is the 2×2 identity matrix. The PM is of the form $y_{k+1|2} = F_2 u_{k|2} + p_2$ where

$$F_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \; p_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} y_k. \tag{4.42}$$

Note that $x_k = y_k$ in this example. The EMPC control is obtained from

$$\begin{aligned} u_{k|2}^* &= -H^{-1} f = -(F_2^T Q F_2 + P)^{-1} F_2^T Q (p_2 - r_{k+1|2}) \\ &= -\frac{1}{5} \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} y_k - r_{k+1} \\ y_k - r_{k+2} \end{bmatrix}. \end{aligned} \tag{4.43}$$

This gives

$$u_k^* = -\frac{2}{5}(y_k - r_{k+1}) - \frac{1}{5}(y_k - r_{k+2}) = -\frac{3}{5} y_k + \frac{2}{5} r_{k+1} + \frac{1}{5} r_{k+2}. \tag{4.44}$$

$$u_{k+1}^* = \frac{1}{5}(y_k - r_{k+1}) - \frac{2}{5}(y_k - r_{k+2}) = -\frac{1}{5} y_k - \frac{1}{5} r_{k+1} + \frac{2}{5} r_{k+2} \tag{4.45}$$

Hence, the result is the same as in Example (4.4). Note that the computed control u_{k+2}^* is not used for control purposes. However, the future controls may be used in order to support the process operators. Note that the control u_k^* gives off-set free control, i.e., $y = r$ in steady state because the process (4.37) consists of an integrator. Show this by analyzing the closed loop system, $y_{k+1} = y_k + u_k^*$. This example is implemented in the MATLAB script **main_ex45.m**. *See also Figure 4.1 for simulation results.*

□

Example 4.6 (Constrained EMPC control)
Consider the same problem as in Example 4.4. The system is described by the state space model "matrices", $A = 1$, $B = 1$ and $D = 1$. In order to compute the EMPC (actual) control variables we need the matrices F_L, Q, P and O_L. With prediction horizon, $L = 2$, we have

$$F_2 = \begin{bmatrix} DB & 0 \\ DAB & DB \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \; O_2 = \begin{bmatrix} D \\ DA \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; Q = P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.46}$$

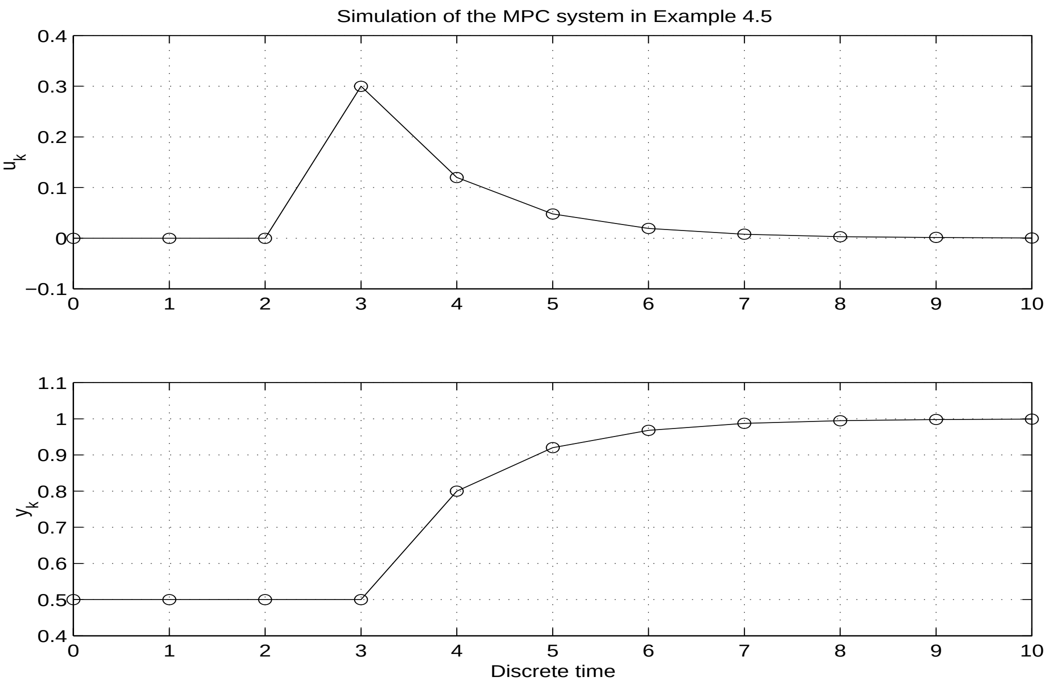

Figure 4.1: Simulation of the MPC system in Example 4.5. This figure was generated with the MATLAB file **main_ex45.m**.

The basic EMPC control are then using the matrices $H = F_2^T Q F_2 + P$, $f = F_2^T Q(p_2 - r_{k+1|2}$ where $p_2 = O_2 A x_k$ as well as an inequality describing constaints. This is specified in the following. We will here add an inequality constraint (lower and upper bound) on the control, i.e.,

$$0 \leq u_k \leq 0.2. \tag{4.47}$$

Hence, we can write (4.47) as the standard form inequality constraint

$$\mathcal{A} u_{k|2} \leq b, \tag{4.48}$$

where

$$\mathcal{A} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}, \; b = \begin{bmatrix} 0.2 \\ 0 \end{bmatrix}. \tag{4.49}$$

This is a QP problem. The example is implemented in the MATLAB script **main_ex46.m**. *See Figure 4.2 for simulation results. As we see from the figure there exist a feasible control satisfying the constraints. Compare with the uncinstrained MPC control in Example 4.5 and Figure 4.2.*

□

Example 4.7 (EMPC control of unstable process)
Some methods for MPC can not be used on unstable processes, usually due to limitations in the process model used. We will therefore illustrate MPC of an unstable

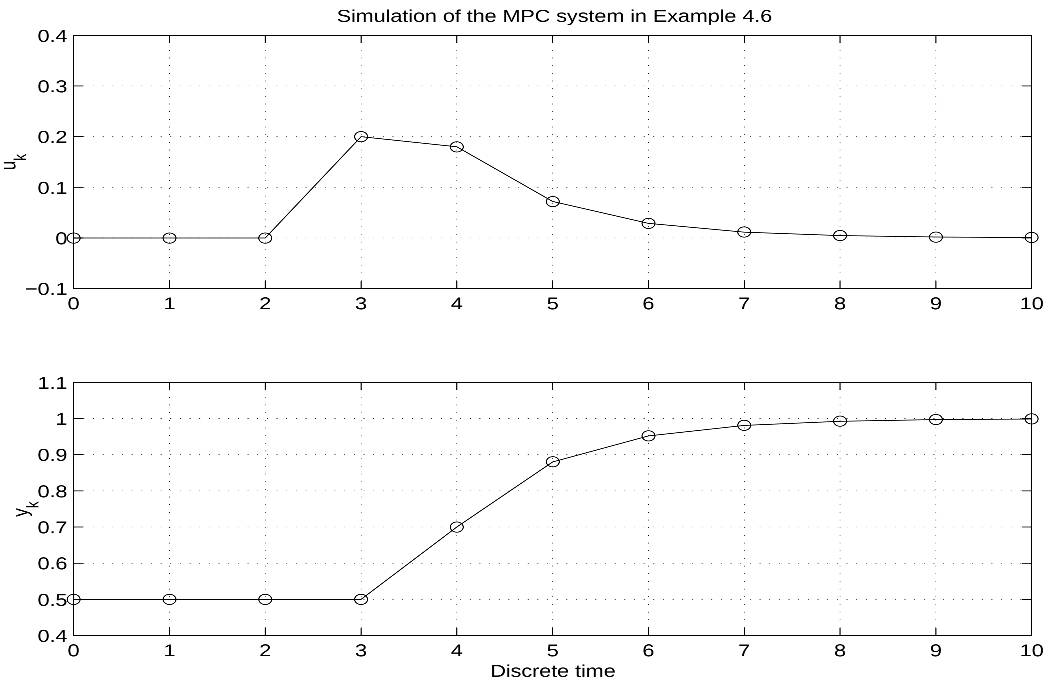

Figure 4.2: Simulation of the EMPC system in Example 4.6. This figure was generated with the MATLAB file **main_ex46.m**.

process. Consider an inverted pendulum on a cart. The equation of motion (model) of the process is

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} q_2 \\ a_{21}\sin(q_1) + b_{21}\cos(q_1)u \end{bmatrix}, \tag{4.50}$$

where q_1 [rad] is the angle from the vertical line, $q_2 = \dot{q}_1$ $[\frac{rad}{s}]$ is the angular velocity and u is the velocity of the cart assumed here to be a control variable. The model parameters are $a_{21} = 14.493$ and $b_{21} = -1.4774$. The coefficients are related to physical parameters as $b_{21} = -\frac{ml_G}{J_t + ml_G^2}$ and $a_{21} = -gb_{21}$, where $m = 0.244$ is the mass of the pendulum, $l_G = 0.6369$ is the length from the cart to the center of gravity of the pendulum, $J_t = 0.0062$ is the moment of inertia about the center of gravity and $g = 9.81$. Linearizing (4.50) around $q_1 = 0$ and $q_2 = 0$ gives the linearized continuous time model

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ a_{21} & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} 0 \\ b_{21} \end{bmatrix} u. \tag{4.51}$$

A discrete time model (A, B) is obtained by using the zero order hold method with sampling interval $\Delta t = \frac{1}{40} = 0.025$ [sec]. The control problem is to stabilize the system, i.e., to control both states, $x_1 = q_1$ and $x_2 = q_2$, to zero for non-zero initial values on the angle $q_1(t = 0)$ and the angular velocity $q_2(t = 0)$. Hence, the reference signals for the EMPC control is simply $r_{k+1|L} = 0$. The output equation for this system is $y_k = Dx_k$ with

$$D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.52}$$

An LQ objective

$$J_k = \sum_{i=1}^{L}((y_{k+i} - r_{k+i})^T Q_i (y_{k+i} - r_{k+i}) + u_{k+i-1}^T P_i u_{k+i-1}), \tag{4.53}$$

with the weighting matrices

$$Q_i = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}, \quad P_i = 1, \tag{4.54}$$

are used in the EMPC method. Unconstrained MPC is identical to LQ and/or LQG control. An infinite horizon LQ and/or LQG controller will stabilize any linear process. Hence, it make sense to chose a large prediction horizon in the EMPC algorithm in order to ensure stability. Trial and error shows that $20 \leq L$ *results in stabilizing EMPC control. Choosing a prediction horizon,* $L = 100$, *gives the EMPC unconstrained control*

$$u_k^{\mathrm{EMPC}} = \begin{bmatrix} 18.8745 & 5.3254 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}_k. \tag{4.55}$$

In order to compare, we present the simple infinite horizon LQ controller which is

$$u_k^{\mathrm{LQ}} = \begin{bmatrix} 18.8747 & 5.3254 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}_k. \tag{4.56}$$

As we see, the EMPC and the LQ controllers are (almost) identical. The EMPC controller will converge to the LQ controller when L increases towards infinity. There is a slow convergence for this example because the system is open loop unstable, and because the sampling interval is relatively small ($\Delta t = 0.025$)*. The smaller the sampling interval is, the larger must the prediction horizon, L, (in number of samples) be chosen. If the states,* x_k*, is not measured, then, an an state observer for estimating* x_k *can be used instead. The result is then an LQG controller. A demo of this pendulum example is given in the MATLAB file* **main_empc_pendel.m**. *See Figure 4.3 for simulation results. The initial value for the angle between the pendulum and the upright position was* $q_1(t = 0) = 0.2618$ [*rad*]*, which is equivalent to* $q_1(t = 0) = 15°$*. The initial value for the angular velocity is,* $q_2(t = 0) = 0$*. As we see, the EMPC/LQ controller stabilizes the pendulum within a few seconds. We have used the linear and unconstrained EMPC algorithm to control the nonlinear pendulum model (4.50). Dear reader ! Try to stabilize the pendulum with a PD-controller.*

□

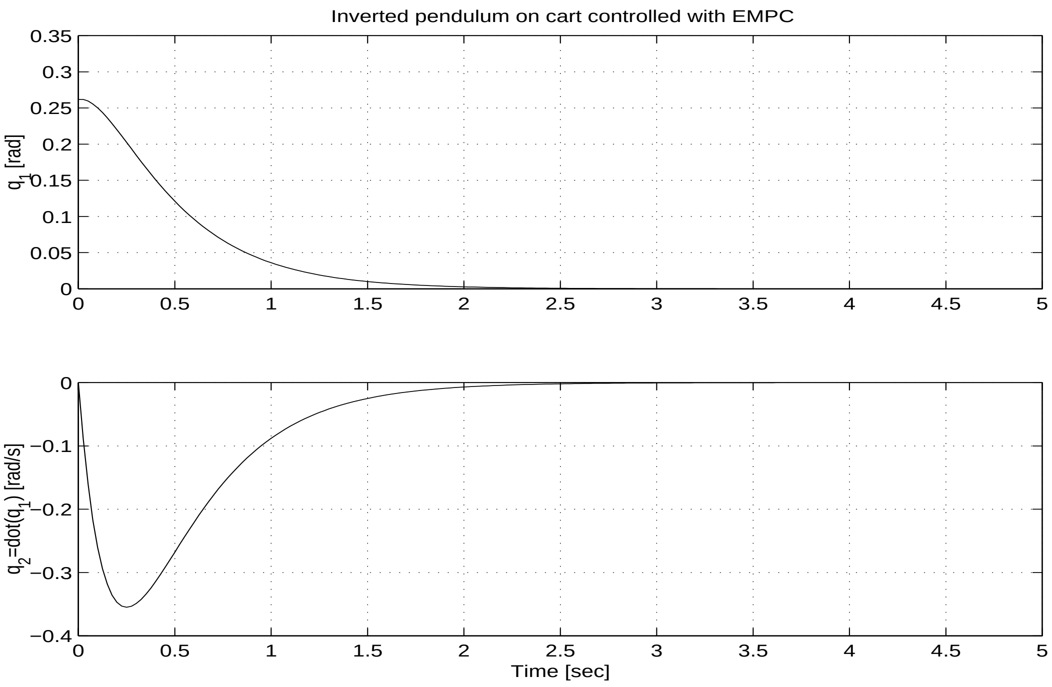

Figure 4.3: Simulation of the EMPC pendulum control problem in Example 4.7. This figure was generated with the MATLAB file **main_empc_pendel.m**.

Example 4.8 (Stable EMPC$_2$)
Consider the process

$$x_{k+1} = Ax_k + Bu_k + v, \tag{4.57}$$
$$y_k = Dx_k + y^0, \tag{4.58}$$

where v and y^0 are constant and unknown disturbances. Consider the control objective

$$J_k = \Delta x_{k+1}^T S \Delta x_{k+1} + (y_{k+1} - r_{k+1})^T Q (y_{k+1} - r_{k+1}) + \Delta u_k^T P \Delta u_k, \tag{4.59}$$

where S is a weighting for the final state deviation, $\Delta x_{k+1} = x_{k+1} - x_k$. The idea with this is to obtain an MPC/LQ controller with guaranteed stability for all choices $Q > 0$ and $P > 0$ if S is chosen properly. In order to ensure stability we chose $S = R$ where R is the solution to the discrete time algebraic Riccati equation. In order to minimize (4.59) we need an PM for y_{k+1} and an expression for Δx_{k+1} in terms of the unknown Δu_k. We have

$$p_1 = y_k + A\Delta x_k = y_k + AD^{-1}\Delta y_k, \tag{4.60}$$
$$y_{k+1} = p_1 + DB\Delta u_k, \tag{4.61}$$
$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k = AD^{-1}\Delta y_k + B\Delta u_k, \tag{4.62}$$

where D is non-singular in order to illustrate how Δx_k can be computed. Assume that D is non-singular, then, without loss of generality we can work with a transformed state space model in which $D = I$. Note, this can be generalized to observable systems. Remark that the PM (4.61) and (4.62) are independent of the constant *disturbances v and y^0. This is one advantage of working with deviation control variables. Substituting into (4.59) gives*

$$\begin{aligned} J_k = (A\Delta x_k + B\Delta u_k)^T S(A\Delta x_k + B\Delta u_k) + \Delta u_k^T P \Delta u_k \\ +(p_1 + DB\Delta u_k - r_{k+1})^T Q(p_1 + DB\Delta u_k - r_{k+1}) = \Delta u_k^T H \Delta u_k + 2f_k^T \Delta u_k, \end{aligned} \quad (4.63)$$

where

$$H = B^T SB + (DB)^T QDB + P, \quad (4.64)$$

$$f_k = B^T SA\Delta x_k + (DB)^T Q(p_1 - r_{k+1}). \quad (4.65)$$

This can be proved by expanding (4.63) or from the first order necessary condition for a minimum, i.e.,

$$\begin{aligned} \frac{\partial J_k}{\partial \Delta u_k} &= 2B^T S(A\Delta x_k + B\Delta u_k) + 2(DB)^T Q(p_1 + DB\Delta u_k - r_{k+1}) + 2P\Delta u_k \\ &= 2(B^T SB + (DB)^T QDB + P)\Delta u_k + 2(B^T SA\Delta x_k + (DB)^T Q(p_1 - r_{k+1})) = 0. \end{aligned}$$

Solving with respect to Δu_k gives

$$\Delta u_k^* = -H^{-1} f_k = -(B^T SB + (DB)^T QDB + P)^{-1}(B^T SA\Delta x_k + (DB)^T Q(p_1 - r_{k+1})).$$

Consider now a scalar system in which $A = a$, $B = b$, $D = d$, $Q = q$, $P = p$ and $S = s$. This gives

$$\Delta u_k^* = -\frac{sab\Delta x_k + qdb(p_1 - r_{k+1})}{sb^2 + q(db)^2 + p} = -\frac{sabd^{-1}\Delta y_k + qdb(p_1 - r_{k+1})}{sb^2 + q(db)^2 + p}. \quad (4.66)$$

It is important to note that the control $u_k = \Delta u_k^* + u_{k-1}$ is independent of v and y^0. This means that the control also is independent of constant terms x^s, u^s, y^s, v^s in a model

$$x_{k+1} - x^s = A(x_k - x^s) + B(u_k - u^s) + Cv^s, \quad (4.67)$$

$$y_k = D(x_k - x^s) + y^s, \quad (4.68)$$

because (4.67) and (4.68) can be written in a form similar to (4.57) and (4.58) with constant disturbances

$$v = x^s - (Ax^s + Bu^s) + Cv^s, \quad (4.69)$$

$$y^0 = y^s - Dx^s. \quad (4.70)$$

□

Example 4.9 (Stability analysis EMPC$_2$)
Consider the process in Example 4.8 and a scalar system where $d = 1$. The control, Equation (4.66), is

$$\Delta u_k^* = -\frac{sab\Delta x_k + qb(y_k + a\Delta x_k - r_{k+1})}{(s + q)b^2 + p}. \quad (4.71)$$

The closed loop system, recognizing that $\Delta x_{k+1} = a\Delta x_k + b\Delta u_k$, *is*

$$\Delta x_{k+1} = a(1 - \frac{(s+q)b^2}{(s+q)b^2+p})\Delta x_k - \frac{qb^2}{(s+q)b^2+p}x_k - c_3 y^0 + c_3 r_{k+1}, \quad (4.72)$$

where

$$c_3 = \frac{qb^2}{(s+q)b^2+p}. \quad (4.73)$$

This can be written as a difference equation

$$x_{k+1} = c_1 x_k + c_2 x_{k-1} - c_3 y^0 + c_3 r_{k+1}, \quad (4.74)$$

where

$$c_1 = 1 + a(1 - \frac{s+q}{q}c_3) - c_3, \quad (4.75)$$

$$c_2 = -a(1 - \frac{s+q}{q}c_3). \quad (4.76)$$

Hence, the closed loop system is described by

$$\begin{bmatrix} x_{k+1} \\ x_{k+2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ c_2 & c_1 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ c_3 & -c_3 \end{bmatrix} \begin{bmatrix} r_{k+2} \\ y^0 \end{bmatrix}, \quad (4.77)$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} + \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} r_{k+2} \\ y^0 \end{bmatrix}. \quad (4.78)$$

In order to analyze the stability of the closed loop system we can put all external signals equal to zero, i.e., $r_{k+1} = 0$ *and* $y^0 = 0$. *We can now analyze the stability similarly as in Example 4.3. Using* $z = \frac{1+w}{1-w}$ *in the characteristic polynomial* $z^2 - c_1 z - c_2 = 0$ *gives the polynomial*

$$(1 + c_1 - c_2)w^2 + 2(1 + c_2)w + c_3 = 0. \quad (4.79)$$

The polynomial coefficients must be positive in order for the closed loop system to be stable. We must ensure that

$$a_0 = 1 + c_1 - c_2 = 2 + 2a(1 - \frac{s+q}{q}c_3) - c_3 = \frac{(2s+q)b^2 + 2(1+a)p}{(s+q)b^2+p} > 0, \quad (4.80)$$

$$a_1 = 2(1 + c_1) = 2 - 2a(1 - \frac{s+q}{q}c_3) = \frac{2(s+q)b^2 + 2(1-a)p}{(s+q)b^2+p} > 0, \quad (4.81)$$

since $c_3 > 0$. *Hence,* a_0 *is always positive when* $a \geq 0$. *Requiring* $a_1 > 0$ *gives the simple expression*

$$\frac{a-1}{b^2} < \frac{s+q}{p}. \quad (4.82)$$

This expression is valid for physical systems in which $a \geq 0$. *Letting* $s = 0$ *gives the same result as in Example 4.3.*

□

Chapter 5

Extension of the control objective

5.1 Linear model predictive control

We will in this section simply discuss an extension of the MPC theory presented so far. The main point is to extend the control objective with so called target variables. Instead of weighting the controls $u_{k|L}$ in the objective we often want to weight the difference $u_{k|L} - u_0$ where u_0 is a vector of specified target variables for the control.

Given a linear process model

$$x_{k+1} = Ax_k + Bu_k + Cv_k, \tag{5.1}$$

$$y_k = Dx_k + Fv_k, \tag{5.2}$$

$$z_k = Hx_k + Mv_k, \tag{5.3}$$

where z_k is a vector of property variables.

A relatively general objective function for MPC is the following LQ index.

$$\mathcal{J}_k = \sum_{i=1}^{L} \mathcal{L}(u_{k+i-1}), \tag{5.4}$$

where

$$\begin{aligned}\mathcal{L}(u_{k+i-1}) = (y_{k+i} - r_{k+i})^T Q_{k+i}(y_{k+i} - r_{k+i}) + \Delta u_{k+i-1}^T R_{k+i-1} \Delta u_{k+i-1} \\ + u_{k+i-1}^T P_{k+i-1} u_{k+i-1} + (u_{k+i-1} - u_{k+i-1}^0)^T P_{k+i-1}^2 (u_{k+i-1} - u_{k+i-1}^0) \\ + (z_{k+i} - z_{k+i}^0)^T Q_{k+i}^z (z_{k+i} - z_{k+i}^0) + (x_{k+i} - x_{k+i}^0)^T Q_{k+i}^x (x_{k+i} - x_{k+i}^0).\end{aligned} \tag{5.5}$$

This can for simplicity of notation be written in matrix form. Note that we are using the vector and matrix notation which are common in subspace system identification.

$$\mathcal{J}_k = \mathcal{L}(u_{k|L}), \tag{5.6}$$

where

$$\begin{aligned}\mathcal{L}(u_{k|L}) = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + \Delta u_{k|L}^T R \Delta u_{k|L} \\ + u_{k|L}^T \mathcal{P} u_{k|L} + (u_{k|L} - u_0)^T \mathcal{P}_2 (u_{k|L} - u_0) \\ + (z_{k+1|L} - z_0)^T Q_z (z_{k+1|L} - z_0) + (x_{k+1|L} - x_0)^T Q_x (x_{k+1|L} - x_0).\end{aligned} \tag{5.7}$$

The time dependence of the weighting matrices and the target vectors are omitted for simplicity of notation. The performance index can be written in terms of the unknown control deviation variables as follows

$$\mathcal{J}_k = \Delta u_{k|L}^T \mathcal{H} \Delta u_{k|L} + 2 f_k^T \Delta u_{k|L} + \mathcal{J}_0(k), \tag{5.8}$$

where $\mathcal{H} \in \mathbb{R}^{Lr \times Lr}$ is a constant matrix, $f_k \in \mathbb{R}^{Lr}$ is a time varying vector but independent of the unknown control deviation variables, $\mathcal{J}_0(k) \in \mathbb{R}$ is time varying but independent of the optimization problem. Note that it will be assumed that $\mathcal{H}$ is non-singular. Furthermore, we have

$$\mathcal{H} = F_L^T Q F_L + R + S^T \mathcal{P} S + S^T \mathcal{P}_2 S + F_z^T Q_z F_z + F_x^T Q_x F_x, \tag{5.9}$$

$$\begin{aligned}f_k = F_L^T Q (p_L - r_{k+1|L}) + S^T \mathcal{P} c u_{k-1} \\ + S^T \mathcal{P}_2 (c u_{k-1} - u_0) + F_z^T Q_z (p_z - z_0) + F_x^T Q_x (p_x - x_0),\end{aligned} \tag{5.10}$$

where F_L, F_z and F_x are lower triangular toepliz matrices of the form $F_L = F_L(A, B, D)$, $F_z = F_L(A, B, H)$, $F_x = F_L(A, B, I_{n \times n})$.

Furthermore, p_L, p_z and p_x represents the autonomeous system responses for y, z and x, respectively.

$$p_L = O_L(D, A) A x_k + F_L(A, B, D) c u_{k-1}, \tag{5.11}$$

$$p_z = O_L(H, A) A x_k + F_L(A, B, H) c u_{k-1}, \tag{5.12}$$

$$p_x = O_L(I, A) A x_k + F_L(A, B, I) c u_{k-1}. \tag{5.13}$$

If x_k is not measured, then, x_k can be taken from a traditional state estimator (e.g. Kalman filter) or computed from the model and a sequence of old inputs and outputs. If only the measured output variables are used for identification of x_k we have (for the EMPC algorithm)

$$\hat{x}_k = A^{J-1} O_J^\dagger y_{k-J+1|J} + (\mathcal{C}_{J-1} - A^{J-1} O_J^\dagger \mathcal{H}_J^d) u_{k-J+1|J-1}, \tag{5.14}$$

where $O_J = O_J(D, A)$, $\mathcal{C}_J = \mathcal{C}_J(A, B, D)$, $\mathcal{H}_L^d = \mathcal{H}_L^d(A, B, D)$. J is the identification horizon, i.e. the number of past outputs used to compute x_k.

Chapter 6

DYCOPS5 paper: On model based predictive control

An input and output model is used for the development of a model based predictive control framework for linear model structures. Different MPC algorithms which are based on linear state space models or linear polynomial models fit into this framework. A new identification horizon is introduced in order to represent the past.

6.1 Introduction

A method and framework for linear Model Predictive Control (MPC) is presented. The method is based on a linear state space model or a linear polynomial model. There are two time horizons in the method: one horizon L for prediction of future outputs (*prediction horizon*) and a second horizon J into the past (*identification horizon*). The new horizon J defines one sequence of past outputs and one sequence of past inputs, which may be used to estimate the model or only the present state of the process. This eliminates the need for an observer. This method is refereed to as *Extended state space Model Predictive Control* (EMPC). A second idea of this method is to obtain a framework to incorporate different linear MPC algorithms.

The EMPC method can be shown to give identical results as the Generalized Predictive Control (GPC) algorithm (see Ordys and Clarke (1994) and the references therein) provided the minimal *identification horizon J* is chosen. A larger identification horizon is found from experiments to have a positive effect upon noise filtering. Note that the EMPC method may be defined in terms of some matrices to be defined lather. This can simplify the computations compared to the traditional formulation of the GPC which is based on the solution of a polynomial matrix Diophantine equation. The EMPC algorithm is a simple alternative approach which avoids explicit state reconstruction. The EMPC algorithm does not require solutions of Diophantine equations (as in GPC) or non linear matrix Riccati equations (as in Linear Quadratic (LQ) optimal control). The EMPC method is based on results from a Subspace IDentification (SID) method for linear systems presented in, e.g. Di Ruscio (1997a). Other SID methods are presented in Van Overschee and De Moor (1996). An SID method can with advantage be used as a modeling tool for MPC.

The contributions herein are: The inclusion of a new identification horizon for MPC. The derivation of a new input and output relationship between the past and the future data which e.g. generalizes Proposition 3.1 in Albertos and Ortega (1989). The framework for the derivation of prediction models from any linear model. The main results in this paper are organized as follows. A matrix equation is presented in Section 6.2. Two prediction models are proposed in Sections 6.3.1 and 6.3.2. The prediction model is analyzed in Section 6.3.3. The MPC strategy is discussed in Section 6.4.

6.2 Preliminaries

6.2.1 System description

Consider a process described by a linear, discrete time invariant state space model (SSM)

$$x_{k+1} = Ax_k + Bu_k + Cv_k, \tag{6.1}$$

$$y_k = Dx_k + Eu_k + Fv_k, \tag{6.2}$$

where k is discrete time, $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^r$ is the control input, $v_k \in \mathbb{R}^l$ is an external input and $y_k \in \mathbb{R}^m$ is the output. (A, B, C, D, E, F) are of appropriate dimensions. In continuous time systems E is usually zero. This is not the case in discrete time due to sampling. However, for the sake of simplicity in notation, only strictly proper systems with $E = 0$ are discussed. The only proper case in which $E \neq 0$ is straightforward because we will treat the case with external inputs v_k and $F \neq 0$. There is structurally no difference between inputs u_k or external inputs v_k in the SSM, (6.1) and (6.2). It is assumed that (D, A) is observable and (A, B) is controllable.

An alternative is to use a polynomial model (e.g. ARMAX, CARIMA, FIR, step response models etc.), which in discrete time can be written as

$$y_{k+1} = \mathcal{A}y_{k-n_a+1|n_a} + \mathcal{B}u_{k-n_b+1|n_b} + \mathcal{C}v_{k-n_c+1|n_c}, \tag{6.3}$$

where we have used the definition

$$y_{i|j} \overset{\text{def}}{=} \begin{bmatrix} y_i^T \; y_{i+1}^T \; \cdots \; y_{i+j-1}^T \end{bmatrix}^T \;\in\; \mathbb{R}^{im}, \tag{6.4}$$

which is refereed to as an extended (output) vector. $\mathcal{A} \in \mathbb{R}^{m\times n_a m}$, $\mathcal{B} \in \mathbb{R}^{m\times n_b r}$ and $\mathcal{C} \in \mathbb{R}^{m\times n_c l}$ are constant matrices. Note that all linear polynomial models, represented by the difference model (6.3), can be transferred to (6.1) and (6.2). Only deterministic systems are considered in this paper.

6.2.2 Definitions

Given the SSM, (6.1) and (6.2). The *extended observability* matrix $\mathcal{O}_i$ for the pair (D, A) is defined as

$$\mathcal{O}_i \overset{\text{def}}{=} \begin{bmatrix} D \\ DA \\ \vdots \\ DA^{i-1} \end{bmatrix} \in \mathbb{R}^{im\times n}, \tag{6.5}$$

where i denotes the number of block rows.

The *reversed extended controllability* matrix $\mathcal{C}_i^d$ for the pair (A, B) is defined as

$$\mathcal{C}_i^d \overset{\text{def}}{=} \begin{bmatrix} A^{i-1}B & A^{i-2} & \cdots & B \end{bmatrix} \in \mathbb{R}^{n\times ir}, \tag{6.6}$$

where i denotes the number of block columns.
A matrix $\mathcal{C}_i^s$ for the pair (A, C) is defined similar, i.e., with C substituted for B in (6.6).

The *lower block triangular Toeplitz* matrix $\mathcal{H}_i^d \in \mathbb{R}^{im\times(i-1)r}$ for the triple (D, A, B)

$$\mathcal{H}_i^d \overset{\text{def}}{=} \begin{bmatrix} 0 & 0 & \cdots & 0 \\ DB & 0 & \cdots & 0 \\ DAB & DB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ DA^{i-2}B & DA^{i-3}B & \cdots & DB \end{bmatrix}. \tag{6.7}$$

A *lower block triangular Toeplitz* matrix $\mathcal{H}_i^s \in \mathbb{R}^{im\times il}$ for the quadruple (D, A, C, F) is defined similarly. See e.g., Di Ruscio (1997a).

Define $\Delta u_k = u_k - u_{k-1}$. Using (6.4) we have

$$u_{k|L} = S\Delta u_{k|L} + cu_{k-1} \tag{6.8}$$

where $S \in \mathbb{R}^{Lr\times Lr}$ and $c \in \mathbb{R}^{Lr\times r}$ are given by

$$S = \begin{bmatrix} I_r & 0_r & \cdots & 0_r \\ I_r & I_r & \cdots & 0_r \\ \vdots & \vdots & \ddots & \vdots \\ I_r & I_r & \cdots & I_r \end{bmatrix}, \quad c = \begin{bmatrix} I_r \\ I_r \\ \vdots \\ I_r \end{bmatrix}, \tag{6.9}$$

where I_r is the $r \times r$ identity matrix and 0_r is the $r \times r$ matrix of zeroes.

6.2.3 Extended state space model

Proposition 2.1 A linear model ((6.1) and (6.2), or (6.3)) can be described with an equivalent extended state space model (ESSM)

$$y_{k+1|L} = \tilde{A}_L y_{k|L} + \tilde{B}_L u_{k|L} + \tilde{C}_L v_{k|L+1}. \tag{6.10}$$

L is defined as the prediction horizon such that $L \geq L_{\min}$ where L_{min}, sufficient for the ESSM to exist, is in case of (6.1) and (6.2) defined by

$$L_{\min} \overset{\text{def}}{=} \begin{cases} n - \text{rank}(D) + 1 & \text{when } \text{rank}(D) < n \\ 1 & \text{when } \text{rank}(D) \geq n \end{cases}.$$

The extended output or state vector $y_{k|L}$, the extended input vectors $u_{k|L}$ and $v_{k|L+1}$, follows from the definition in (6.4). The matrices in (6.10) are in case of an SSM ((6.1) and (6.2)) given as

$$\tilde{A}_L \overset{\text{def}}{=} \mathcal{O}_L A (\mathcal{O}_L^T \mathcal{O}_L)^{-1} \mathcal{O}_L^T, \tag{6.11}$$

$$\tilde{B}_L \overset{\text{def}}{=} \begin{bmatrix} \mathcal{O}_L B & \mathcal{H}_L^d \end{bmatrix} - \tilde{A}_L \begin{bmatrix} \mathcal{H}_L^d & 0_{Lm \times r} \end{bmatrix}, \tag{6.12}$$

$$\tilde{C}_L \overset{\text{def}}{=} \begin{bmatrix} \mathcal{O}_L C & \mathcal{H}_L^s \end{bmatrix} - \tilde{A}_L \begin{bmatrix} \mathcal{H}_L^s & 0_{Lm \times l} \end{bmatrix}, \tag{6.13}$$

where $\tilde{A}_L \in \mathbb{R}^{Lm \times Lm}$, $\tilde{B}_L \in \mathbb{R}^{Lm \times Lr}$ and $\tilde{C}_L \in \mathbb{R}^{Lm \times (L+1)l}$.
The polynomial model (6.3), can be formulated directly as (6.10) for $L \geq \max(n_a, n_b, n_c)$ by the use of (6.14)-(6.17).

Proof. See Di Ruscio (1997a). □

The importance of the ESSM for model predictive control is that it facilitates a simple and general method for building a *prediction model* from a linear process model. Note, it is in general not sufficient to choose $L_{\min}$ as the ceiling function $\lceil \frac{n}{m} \rceil$. The ESSM transition matrix $\tilde{A}_L$ has the same n eigenvalues as the matrix A and $Lm - n$ eigenvalues equal to zero. This follows from similarity transformation on (6.11).

In the SID algorithm, Di Ruscio (1997a), it is shown that the ESSM can be identified from a sliding window of known data. However, when the inputs used for identification are poor from a persistent excitation point of view, it is better to identify only the minimal order ESSM. Note also that $L_{\min}$ coincides with the demand for persistent excitation, i.e., the inputs must at least be persistently exciting of order $L_{\min}$ in order to identify the SSM from known input and output data.

Consider an ESSM of order J defined as follows

$$y_{k+1|J} = \tilde{A}_J y_{k|J} + \tilde{B}_J u_{k|J} + \tilde{C}_J v_{k|J+1}. \tag{6.14}$$

Then, an ESSM of order L is constructed as

$$\tilde{A}_L = \begin{bmatrix} 0_{(L-J)m \times m} & I_{(L-J)m \times (L-1)m} \\ 0_{Jm \times (L-J)m} & \tilde{A}_J \end{bmatrix}, \tag{6.15}$$

$$\tilde{B}_L = \begin{bmatrix} 0_{(L-J)m \times Lr} & \\ 0_{Jm \times (L-J)r} & \tilde{B}_J \end{bmatrix}, \tag{6.16}$$

$$\tilde{C}_L = \begin{bmatrix} 0_{(L-J)m \times (L+1)l} & \\ 0_{Jm \times (L-J)l} & \tilde{C}_J \end{bmatrix}, \tag{6.17}$$

where $L_{min} \leq J < L$. J is defined as the *identification horizon*. This last formulation of the ESSM matrices is attractive from an identification point of view. It will also have some advantages with respect to noise retention compared to using the matrices defined by (6.11) to (6.13) directly (i.e. putting $J = L$) in the MPC method which will be presented in Sections 6.3 and 6.4.

6.3 Prediction models

6.3.1 Prediction model in terms of process variables

Define the following *Prediction Model* (PM)

$$y_{k+1|L} = p_L(k) + F_L u_{k|L}. \tag{6.18}$$

Proposition 3.1 Consider the SSM, (6.1) and (6.2). The term $p_L(k)$ is given by

$$\begin{aligned} p_L(k) &= \mathcal{O}_L A^J \mathcal{O}_J^{\dagger} y_{k-J+1|J} \\ &+\beta_L^s v_{k-J+1|J+L} + P_L u_{k-J+1|J-1}, \end{aligned} \tag{6.19}$$

where $\mathcal{O}_J^{\dagger} = (\mathcal{O}_J^T \mathcal{O}_J)^{-1} \mathcal{O}_J^T$ is the Moore-Penrose pseudo-inverse of $\mathcal{O}_J$, (6.5). The term $p_L(k)$ is completely known, i.e., it depends upon the process model, known past and present process output variables, known past process control input variables and past, present and future external input variables (assumed to be known). $p_L(k)$ can be interpreted as the autonomous process response, i.e. putting future inputs equal to zero in (6.18).

Given the quintuple matrices (A, B, C, D, F). The matrices in (6.18) and (6.19) can be computed as follows. We have one matrix P_L which is related to past control inputs and one matrix F_L which is related to present and future control inputs.

$$\begin{aligned} F_L &= \begin{bmatrix} \mathcal{O}_L B \ \mathcal{H}_L^d \end{bmatrix} & &\in \ \mathbb{R}^{Lm \times Lr}, \\ P_L &= \mathcal{O}_L A \mathcal{C}_{J-1}^d - \mathcal{O}_L A^J \mathcal{O}_J^{\dagger} \mathcal{H}_J^d & &\in \ \mathbb{R}^{Lm \times (J-1)r}. \end{aligned} \tag{6.20}$$

The matrix $\beta_L^s \in \mathbb{R}^{Lm \times J+L}$ in (6.19) is given by

$$\beta_L^s = \begin{bmatrix} \mathcal{O}_L (\mathcal{C}_J^s - A^J \mathcal{O}_J^{\dagger} \mathcal{H}_J^s) \ \mathcal{H}_L^s \end{bmatrix}. \tag{6.21}$$

If desired, we can write $\beta_L^s = \begin{bmatrix} P_L^s \ F_L^s \end{bmatrix}$ where the sub-matrix $P_L^s \in \mathbb{R}^{Lm \times (J-1)l}$ is related to past external inputs and the sub-matrix $F_L^s \in \mathbb{R}^{Lm \times (L+1)l}$ is related to current and future external inputs.

$$\begin{aligned} F_L^s &= \begin{bmatrix} \mathcal{O}_L C - \mathcal{O}_L A^J \mathcal{O}_J^{\dagger} E_J^s \ \mathcal{H}_L^s \end{bmatrix} \\ P_L^s &= \mathcal{O}_L A \mathcal{C}_{J-1}^s - \mathcal{O}_L A^J \mathcal{O}_J^{\dagger} \mathcal{H}_J^s(:, 1:(J-1)l) \end{aligned} \tag{6.22}$$

and where E_J^s is the last J-th block column of matrix $\mathcal{H}_J^s$. Note that $E_J^s = 0$ when $F = 0$ in (6.2).

Proof. See Section 6.3.3. □

Proposition 3.2 Consider the ESSM, (6.10), (6.14)- (6.17). The term $p_L(k)$ is given by

$$\begin{aligned} p_L(k) &= \tilde{A}_L^L y_{k-L+1|L} \\ &+\beta_L^s v_{k-L+1|2L} + P_L u_{k-L+1|L-1}. \end{aligned} \tag{6.23}$$

Given the ESSM double matrices $(\tilde{A}_L, \tilde{B}_L)$. A procedure for defining the *prediction model* matrices F_L and P_L from the ESSM matrices is given as follows. Define the impulse response matrices

$$H_i \overset{\text{def}}{=} \tilde{A}_L^{i-1} \tilde{B}_L \ \in \ \mathbb{R}^{Lm \times Lr}. \tag{6.24}$$

The prediction of future outputs can be expressed in terms of sub-matrices of the impulse response matrices (6.24). Define a (one block column) Hankel matrix from the impulse response matrices H_i in (6.24) for $i = 1, \ldots, L$ as follows

$$\begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_L \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1L} \\ H_{21} & H_{22} & \cdots & H_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ H_{L1} & H_{L2} & \cdots & H_{LL} \end{bmatrix}, \tag{6.25}$$

where the sub-matrices $H_{ij} \in \mathbb{R}^{Lm \times r}$. Matrix F_L is computed from the upper right part of (6.25), including the main block diagonal, of the Hankel matrix. The first block column (sub-matrix) in F_L is equal to the sum of the sub-matrices on the main block diagonal of the Hankel matrix. The second block column (sub-matrix) is equal to the sum of the sub-matrices on the block diagonal above the main block diagonal, and so on. That is, the i-th block column in matrix F_L is given by

$$F_{1,i} = \sum_{j=1}^{L-i+1} H_{j,j+i-1} \quad \forall\ i = 1, \cdots, L. \tag{6.26}$$

The block columns in P_L are given from the lower left part of the Hankel matrix (6.25). The first block column is given by the lower left sub-matrix in (6.25), i.e., H_{L1}. The i-th block column is given by

$$P_{1,i} = \sum_{j=1}^{i} H_{j+L-i,j} \quad \forall\ i = 1, \cdots, L-1. \tag{6.27}$$

Given the ESSM matrices $(\tilde{A}_L, \tilde{C}_L)$. The matrix β_L^s in (6.23) is computed as follows. Define

$$\Lambda_i \overset{\text{def}}{=} \tilde{A}_L^{i-1} \tilde{C}_L \ \in\ \mathbb{R}^{Lm \times (L+1)l}, \tag{6.28}$$

and the following (one block column) Hankel matrix by using Λ_i (6.28) for $i = 1, \ldots, L$,

$$\begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \vdots \\ \Lambda_L \end{bmatrix} = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} & \cdots & \Lambda_{1L} & \Lambda_{1,L+1} \\ \Lambda_{21} & \Lambda_{22} & \cdots & \Lambda_{2L} & \Lambda_{2,L+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Lambda_{L1} & \Lambda_{L2} & \cdots & \Lambda_{LL} & \Lambda_{L,L+1} \end{bmatrix}. \tag{6.29}$$

Then we have

$$\beta_L^s = \begin{bmatrix} P_L^s & F_L^s \end{bmatrix}, \tag{6.30}$$

$$F_L^s = \begin{bmatrix} F_{1,1}^s & \cdots & F_{1,i}^s & \cdots & F_{1,L}^s & F_{1,L+1}^s \end{bmatrix}, \tag{6.31}$$

$$P_L^s = \begin{bmatrix} P_{1,1}^s & \cdots & P_{1,i}^s & \cdots & P_{1,L-1}^s \end{bmatrix}, \tag{6.32}$$

where the i-th block columns in F_L^s and P_L^s are given by

$$F_{1,i}^s = \sum_{j=1}^{L-i+1} \Lambda_{j,j+i-1} \quad \forall\ i = 1, \cdots, L+1, \tag{6.33}$$

$$P_{1,i}^s = \sum_{j=1}^{i} \Lambda_{j+L-i,j} \quad \forall\ i = 1, \cdots, L-1. \tag{6.34}$$

Proof (outline). In order to express the right hand side of (6.10) in terms of future control inputs we write (6.10) as an L-step ahead predictor.

$$\begin{aligned} y_{k+1|L} = \tilde{A}_L^L y_{k-L+1|L} \\ + \textstyle\sum_{i=1}^L \Lambda_i v_{k-i+1|L+1} + \sum_{i=1}^L H_i u_{k-i+1|L}, \end{aligned} \tag{6.35}$$

where we have used (6.24) and (6.28). The first term on the right hand side of (6.35) depends on known past and present output data $(y_{k-L+1}, \cdots, y_k)$. The second term depends on known past, present and future external inputs $(v_{k-L+1}, \cdots, v_k, \cdots, v_{k+L})$, e.g. measured noise variables, reference variables etc. The third term depends on past, present and future control input variables, i.e. $(u_{k-L+1}, \cdots, u_k, \cdots, u_{k+L-1})$. The right hand side of (6.35) can be separated into two terms, one term which is completely described by the model and the known process variables and one term which depends on the, at this stage, unknown future control input variables, Hence, the PM follows. □

We have illustrated a method for extracting the impulse response matrices DA^iB for the system from the ESSM matrices $\tilde{A}_L, \tilde{B}_L$. Hence, this side result can be used to compute a minimal SSM realization for e.g. a polynomial model.

6.3.2 Prediction model in terms of process deviation variables

Define the following *prediction model*

$$y_{k+1|L} = p_L^{\Delta}(k) + F_L^{\Delta} \Delta u_{k|L}, \tag{6.36}$$

where F_L^{Δ} is lower triangular. $p_L^{\Delta}(k)$ can be interpreted as the autonomous process response, i.e. putting future input deviation variables equal to zero in (6.36).

Proposition 3.3 Given the SSM (6.1) and (6.2), then, $p_L^{\Delta}(k) = p_L(k) + F_L c u_{k-1}$ and $F_L^{\Delta} = F_L S$.

Proof. Substitute (6.8) into (6.18). □

Proposition 3.4 Consider the ESSM, (6.10), (6.14)- (6.17). Define

$$\begin{aligned} p_L^{\Delta}(k) = y_{k-L+1|L} + (\textstyle\sum_{i=1}^L \tilde{A}_L^i) \Delta y_{k-L+1|L} \\ + \beta_L^s \Delta v_{k-L+1|2L} + P_L \Delta u_{k-L+1|L-1}. \end{aligned} \tag{6.37}$$

Given the ESSM matrices $(\tilde{A}_L, \tilde{B}_L)$. Define

$$H_i \overset{\text{def}}{=} \sum_{j=1}^{i} \tilde{A}_L^{j-1} \tilde{B}_L \ \in \ \mathbb{R}^{Lm \times Lr}. \tag{6.38}$$

The procedure for computing F_L^{Δ} and P_L is the same as that presented in Proposition 3.2, (6.25)-(6.27), but with H_i in (6.25) as defined in (6.38). Hence, the i-th block column in F_L^{Δ} is given by (6.26).

Given the ESSM matrices $(\tilde{A}_L, \tilde{C}_L)$. Define

$$\Lambda_i \overset{\text{def}}{=} \sum_{j=1}^{i} \tilde{A}_L^{j-1} \tilde{C}_L \ \in \ \mathbb{R}^{Lm \times (L+1)l}. \tag{6.39}$$

The matrix β_L^s in (6.37) is computed from (6.29)-(6.34) but with Λ_i as defined in (6.39).

Proof (outline). The ESSM (6.10) can be written in terms of control changes as follows

$$\begin{aligned} y_{k+1|L} &= y_{k-L+1|L} + (\textstyle\sum_{i=1}^{L} \tilde{A}_L^i)\Delta y_{k-L+1|L} \\ &+ \textstyle\sum_{i=1}^{L} \Lambda_i \Delta v_{k-i+1|L+1} + \sum_{i=1}^{L} H_i \Delta u_{k-i+1|L}, \end{aligned} \tag{6.40}$$

where we have used (6.39) and (6.38). The PM (6.36) and the algorithm follows. □

The PM (6.36) is attractive in order to give offset free MPC. The PM used by GPC can be derived from Proposition 3.4 with $J = L_{\min}$ and ESSM matrices (6.15), (6.16) and (6.17).

6.3.3 Analyzing the predictor

Consider in the following the SSM in (6.1) and (6.2). There is no explicit expression for the present state estimate $\hat{x}_k$ in the predictor (6.18), (6.19) and (6.20). As an alternative the predictions can be computed by the SSM provided an estimate of the present state is available. In the following we will show that these two prediction strategies are equal provided that the current state equals a least squares estimate. This also illustrates the proof of the PM in (6.18), (6.19) and (6.20).

For simplification purposes we omit the external input v in the following derivations. Further, we only consider the case in 6.3.1. Including the external input v and deviation variables, respectively, can be done in a straightforward manner.

Proposition 3.5 The predictions $y_{k+1|L}$ using (6.1) and (6.2) equal the predictions using (6.18), (6.19) and (6.20) if the current state estimate is computed by

$$\hat{x}_k = \arg\min_{x_k} V, \tag{6.41}$$

where

$$V = \frac{1}{2} \parallel y_{k-J+1|J} - \hat{y}_{k-J+1|J} \parallel_F^2, \tag{6.42}$$

and where $J \geq L_{\min}$ and $\hat{y}_{k-J+1|J}$ denotes the past and current model output data.

Proof. The proof is divided into 3 parts.

Part 1: Deriving an equivalence condition
We first derive an expression for the predictor $y_{k+1|L}$. From (6.1) we have the M-step ahead predictor

$$x_{k+M} = A^M x_k + \mathcal{C}_M^d u_{k|M}. \tag{6.43}$$

Using (6.2), (6.5), (6.20) and (6.43) for $M = 1, \ldots, L$ we have

$$y_{k+1|L} = \mathcal{O}_L A x_k + F_L u_{k|L}. \tag{6.44}$$

By (6.44) and (6.18) the predictions $y_{k+1|L}$ are equal if and only if $\mathcal{O}_L A x_k = p_L(k)$. Inserting for p_L, (6.19) and (6.20), we obtain the following condition for equivalence of the two predictors.

$$\begin{aligned} \mathcal{O}_L A x_k = {} & \mathcal{O}_L A^J O_J^\dagger y_{k-J+1|J} \\ & +(\mathcal{O}_L A \mathcal{C}_{J-1}^d - \mathcal{O}_L A^J O_J^\dagger \mathcal{H}_J^d) u_{k-J+1|J-1}. \end{aligned} \tag{6.45}$$

Part 2: Computing the least squares estimate
By (6.1), (6.2), (6.5) and (6.7), we obtain

$$y_{k-J+1|J} = \mathcal{O}_J x_{k-J+1} + \mathcal{H}_J^d u_{k-J+1|J-1}. \tag{6.46}$$

Putting $y_{k-J+1|J} = \hat{y}_{k-J+1|J}$ defines the least squares problem and objective (6.42). The least squares estimate of x_{k-J+1} is given by

$$\hat{x}_{k-J+1} = O_J^\dagger (y_{k-J+1|J} - \mathcal{H}_J^d u_{k-J+1|J-1}). \tag{6.47}$$

The estimate $\hat{x}_{k-J+1}$ can be transferred to the current estimate $\hat{x}_k$ by the SSM. First, similar to (6.43) we have

$$\hat{x}_k = A^{J-1} \hat{x}_{k-J+1} + \mathcal{C}_{J-1}^d u_{k-J+1|J-1}.$$

Using (6.47) gives

$$\begin{aligned} \hat{x}_k = {} & A^{J-1} O_J^\dagger y_{k-J+1|J} \\ + {} & (\mathcal{C}_{J-1}^d - A^{J-1} O_J^\dagger \mathcal{H}_J^d) u_{k-J+1|J-1}. \end{aligned} \tag{6.48}$$

The least squares estimate is clearly a minimum since $\frac{d^2 V}{dx_{k-J+1}^2} = \mathcal{O}_J^T \mathcal{O}_J$ is a positive matrix.

Part 3: Using the least squares estimate for $\hat{x}_k$, i.e. (6.48), guarantees that (6.45) hold. □

6.4 Basic MPC algorithm

6.4.1 The control objective

A discrete time LQ objective can be written in matrix form as follows

$$\begin{aligned} \mathcal{J}_k = {} & (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) \\ & + \Delta u_{k|L}^T R \Delta u_{k|L} + u_{k|L}^T \mathcal{P} u_{k|L}, \end{aligned} \tag{6.49}$$

where $r_{k+1|L}$ is a vector of future references, $y_{k+1|L}$ is a vector of future outputs, $\Delta u_{k|L}$ is a vector of future input changes, and $u_{k|L}$ is a vector of future inputs. Q, R and $\mathcal{P}$ are (usually) block diagonal, time varying, weighting matrices.

6.4.2 Computing optimal control variables

Consider the problem of minimizing the objective (6.49) with respect to the future control variables $u_{k|L}$, subject to a linear model and linear output/state and input constraints. The objective $\mathcal{J}_k$ may be written in terms of $u_{k|L}$. The future outputs $y_{k+1|L}$ can be eliminated from the objective by using the PM (6.18). The deviation variables $\Delta u_{k|L}$ may be eliminated from $\mathcal{J}_k$ by using

$$\Delta u_{k|L} = S^{-1} u_{k|L} - S^{-1} c u_{k-1}, \tag{6.50}$$

where S and c are given in (6.9. Substituting (6.50) and the PM (6.18) into the objective (6.49) gives a QP problem with solution

$$u^*_{k|L} = \arg\min_{Z_2 u_{k|L} \leq b^2_k} \mathcal{J}_k, \tag{6.51}$$

where $Z_2 = ZS^{-1}$, $b^2_k = b_k + ZS^{-1} c u_{k-1}$, Z and b_k as in (6.53). For unconstrained control a closed form solution exists. A *control horizon* N_u can be included by the use of selection matrix in the PM and a modified $\mathcal{J}_k$. The optimal control $u^*_{k|N_u}$ is obtained similarly as (6.51) but with a truncated matrix F_L.

6.4.3 Computing optimal control deviation variables

Consider the problem of minimizing (6.49) with respect to the future control deviation variables $\Delta u_{k|L}$, subject to a linear model and linear constraints. The future outputs $y_{k+1|L}$ may be eliminated from $\mathcal{J}_k$ by using the PM (6.36). $u_{k|L}$ may be eliminated from (6.49) by using (6.8). The minimizing solution to the linear MPC problem is

$$\Delta u^*_{k|L} = \arg\min_{Z \Delta u_{k|L} \leq b_k} \mathcal{J}_k, \tag{6.52}$$

where the matrices Z and b_k may be defined as

$$Z = \begin{bmatrix} S \\ -S \\ I_{Lr} \\ -I_{Lr} \\ F^\Delta_L \\ -F^\Delta_L \end{bmatrix}, \quad b_k = \begin{bmatrix} u^{\max}_{k|L} - c u_{k-1} \\ -u^{\min}_{k|L} + c u_{k-1} \\ \Delta u^{\max}_{k|L} \\ -\Delta u^{\min}_{k|L} \\ y^{\max}_{k+1|L} - p^\Delta_L(k) \\ -y^{\min}_{k+1|L} + p^\Delta_L(k) \end{bmatrix}. \tag{6.53}$$

For unconstrained optimal control we have

$$\begin{aligned} \Delta u^*_{k|L} &= -(R + F^{\Delta T}_L Q F^\Delta_L + S^T \mathcal{P} S)^{-1} \\ &[F^{\Delta T}_L Q (p^\Delta_L(k) - r_{k+1|L}) + S^T \mathcal{P} c u_{k-1}], \end{aligned} \tag{6.54}$$

where usually $F^\Delta_L = F_L S$. Assume $\mathcal{P} = 0$. The optimal control Δu^*_k gives offset free control in steady state. This can be argued as follows: assume that the closed loop system is stable. In steady state we have from (6.37) that $p^\Delta_L(k) = y$ and $\Delta u_{k|L} = 0$. From the expression for the optimal control change $\Delta u^*_{k|L} = 0$ we must have that $y = r$, i.e. zero steady state offset. It is assumed that $(R + F^T_L Q F_L)^{-1} F^T_L Q$ is of full column rank, and that r_k and v_k are bounded as $k \to \infty$. If r_k and v_k are not bounded, they should be modeled and the model included in the process model.

6.5 Conclusions

A framework for computing MPC controllers is presented. The algorithm in this paper (EMPC), the GPC algorithm and the predictive *dynamic matrix control* (DMC) algorithm, etc., fit into this framework. Different linear MPC algorithms are based on different linear process models. However, these algorithms are using a prediction model (PM) with the same structure. The only difference in the PM used by the different algorithms is the autonomous response term $p_L(k)$ in (6.18) or p_L^Δ in (6.36).

An algorithm for computing a PM from different linear models is presented. The EMPC algorithm can be derived directly from the SSM or from any linear model via the ESSM matrices $\tilde{A}_L$, $\tilde{B}_L$ and $\tilde{C}_L$. These matrices can be constructed directly from an linear SSM or an input and output polynomial model. Hence, the ESSM can be viewed as a unifying model for linear model based predictive control.

A new identification horizon is introduced in the MPC concept. The input and output relationship between the past and the future (6.18)-(6.21) which is introduced for MPC in this paper is an improvement with respect to existing literature.

References

Albertos, P. and R. Ortega (1987). On Generalized Predictive Control: Two alternative formulations. *Automatica*, Vol. 25, No.5, pp. 753-755.

Di Ruscio, D. (1997a). A method for identification of combined deterministic stochastic systems. In: *Applications of Computer Aided Time Series Modeling*, Lecture Notes in Statistics 119, Eds. M. Aoki and A. M. Havenner, Springer Verlag.

Di Ruscio, D. (1997b). Model Predictive Control and Identification: A linear state space model approach. Proc. of the *36th IEEE Conference on Decision and Control*, December 10-12, San Diego, USA.

Ordys, A. W. and D. W. Clarke (1993). A state-space description for GPC controllers. *Int. J. Systems Sci.*, Vol. 24, No.9, pp. 1727-1744.

Van Overschee, P. and B. De Moor (1996). *Subspace Identification for Linear Systems*, Kluwer Academic Publishers.

Chapter 7

Extended state space model based predictive control

Abstract

An extended state space (ESS) model, familiar in subspace identification theory, is used for the development of a model based predictive control algorithm for linear model structures. In the ESS model, the state vector consists of system outputs, which eliminates the need for a state estimator. A framework for model based predictive control is presented. Both general linear state space model structures and finite impulse response models fit into this framework.

7.1 Introduction

Two *extended state space model predictive control* (EMPC) algorithms are proposed in Sections 7.3 and 7.4. The first one, EMPC_1, is presented in Sections 7.3.1 and 7.4.1. The second one, EMPC_2, is presented in Sections 7.3.2 and 7.4.2.

There are basically two time horizons involved in the EMPC algorithms: one horizon for prediction of future outputs (*prediction horizon*) and a second horizon into the past (*identification horizon*). The horizon into the past defines one sequence of past outputs and one sequence of past inputs. The sequences of known past inputs and outputs are used to reconstruct (identify) the present state of the process. This eliminates the need for an observer (e.g. Kalman filter). Due to observability there is a minimal *identification* horizon which is necessary to reconstruct (observe) the present state of the plant.

The EMPC_2 algorithm has some similarities with the Generalized Predictive Control (GPC) algorithm presented in Clarke, Mohtadi and Tuffs (1987). See also Albertos and Ortega (1987). The GPC is derived from an input-output transfer matrix representation of the plant. A recursion of Diophantine equations is carried out. Similarities between GPC and LQ/LQG control are discussed in e.g. Bitmead, Gevers and Wertz (1990).

The EMPC algorithms are based on a state space model of the plant. The EMPC$_2$ algorithm can be shown to give identical results as the GPC algorithm when the minimal *identification horizon* is used. A larger identification horizon is found from experiments to have effect upon noise filtering. Note that the EMPC algorithms are exactly defined in terms of the ESS model matrices, something which can simplify the computations compared to the traditional formulation of the GPC which is based on the solution of a polynomial matrix Diophantine equation. The EMPC algorithm is a simple alternative approach which avoid explicit state reconstruction.

The EMPC algorithms do not require iterations and recursions of Diophantine equations (as in GPC) or non linear matrix Riccati equations (as in traditional linear quadratic optimal control).

The EMPC method is based on results from a subspace identification method presented in Di Ruscio (1994), (1995) and (1997). Other subspace methods are presented in Larimore (1983), (1990), Verhagen (1994), Van Overschee and De Moor (1994), (1995) and Van Overschee (1995). See also Viberg (1995) for a survy.

One example that highlights the differences and similarities between thje EMPC and GPC algorithms is presented. We have also investigated how the EMPC algorithms can be used for operator support. Known closed loop input and output time series from a real world industrial process are given. These time series are given as inputs to the EMPC algorithm and the control-inputs are predicted. The idea is that the predicted control-inputs are usful, not only for control but also for operator support.

7.2 Preliminaries

7.2.1 State space model

Consider a process which can be described by the following linear, discrete time invariant state space model (SSM)

$$x_{k+1} = Ax_k + Bu_k + Cv_k \tag{7.1}$$

$$y_k = Dx_k + Eu_k + Fv_k \tag{7.2}$$

where k is discrete time, $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^r$ is the control input vector, $v_k \in \mathbb{R}^l$ is an external input vector and $y_k \in \mathbb{R}^m$ is the output vector. The constant matrices in the SSM are of appropriate dimensions. A is the *state transition matrix*, B is the *control input* matrix, C is the *external input* matrix, D is the *output* matrix, E is the direct *control input to output* matrix, and F is the direct *external input to output* matrix.

Define the integer parameter g as

$$g \overset{\text{def}}{=} \begin{cases} 1 \text{ when } E \neq 0_{m\times r} & (\textit{proper} \text{ system}) \\ 0 \text{ when } E = 0_{m\times r} & (\textit{strictly proper} \text{ system}) \end{cases}$$

In continuous time systems the matrix E is usually zero. This is not the case in discrete time systems due to sampling. The following assumptions are made:

- The pair (D, A) is observable.

- The pair (A, B) is controllable.

7.2.2 Definitions

Associated with the SSM, Equations (7.1) and (7.2), we make the following definitions:

- The *extended observability* matrix $(\mathcal{O}_i)$ for the pair (D, A) is defined as

$$\mathcal{O}_i \overset{\text{def}}{=} \begin{bmatrix} D \\ DA \\ \vdots \\ DA^{i-1} \end{bmatrix} \in \mathbb{R}^{im\times n} \tag{7.3}$$

  where subscript i denotes the number of block rows.

- The *reversed extended controllability* matrix $(\mathcal{C}_i^d)$ for the pair (A, B) is defined as

$$\mathcal{C}_i^d \overset{\text{def}}{=} \begin{bmatrix} A^{i-1}B & A^{i-2}B & \cdots & B \end{bmatrix} \in \mathbb{R}^{n\times ir} \tag{7.4}$$

  where subscript i denotes the number of block columns.

  A matrix $\mathcal{C}_i^s$ for the pair (A, C) is defined similar to Equation (7.4), i.e., with C substituted for B in the above definition.

- The *lower block triangular Toeplitz* matrix $(\mathcal{H}_i^d) \in \mathbb{R}^{im\times(i-1)r}$ for the triple (D, A, B)

$$\mathcal{H}_i^d \overset{\text{def}}{=} \begin{bmatrix} 0 & 0 & \cdots & 0 \\ DB & 0 & \cdots & 0 \\ DAB & DB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ DA^{i-2}B & DA^{i-3}B & \cdots & DB \end{bmatrix} \in \mathbb{R}^{im\times(i-1)r} \tag{7.5}$$

  where subscript i denotes the number of block rows and the number of block columns is $i - 1$.

  A *lower block triangular Toeplitz* matrix $\mathcal{H}_i^s \in \mathbb{R}^{im\times il}$ for the quadruple (D, A, C, F) is defined as

$$\mathcal{H}_i^s \overset{\text{def}}{=} \begin{bmatrix} F & 0 & \cdots & 0 & 0 \\ DC & F & \cdots & 0 & 0 \\ DAC & DC & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ DA^{i-2}C & DA^{i-3}C & \cdots & DC & F \end{bmatrix} \in \mathbb{R}^{im\times il} \tag{7.6}$$

7.2.3 Extended state space model

An *extended state space model* (ESSM) will be defined in this section. The importance of the ESSM for model predictive control is that it facilitates a simple and general method for building a *prediction model.*

Theorem 7.2.1 *The SSM, Equations (7.1) and (7.2), can be described with an equivalent extended state space model (ESSM)*

$$y_{k+1|L} = \tilde{A}_L y_{k|L} + \tilde{B}_L u_{k|L} + \tilde{C}_L v_{k|L+1} \tag{7.7}$$

where L is the number of block rows in the extended state vector $y_{k+1|L}$. L is defined as the prediction horizon chosen such that $L \geq L_{\min}$ where the minimal number L_{min}, sufficient for the ESSM to exist, is defined by

$$L_{\min} \overset{\text{def}}{=} \begin{cases} n - rank(D) + 1 & when\ rank(D) < n \\ 1 & when\ rank(D) \geq n \end{cases} \tag{7.8}$$

The extended output or state vector is defined as,

$$y_{k|L} \overset{\text{def}}{=} \begin{bmatrix} y_k^T & y_{k+1}^T & \cdots & y_{k+L-1}^T \end{bmatrix}^T \tag{7.9}$$

The extended input vectors are defined similarly, i.e.,

$$u_{k|L} \overset{\text{def}}{=} \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+L-1} \end{bmatrix} \quad v_{k|L+1} \overset{\text{def}}{=} \begin{bmatrix} v_k \\ v_{k+1} \\ \vdots \\ v_{k+L-1} \\ v_{k+L} \end{bmatrix} \tag{7.10}$$

The matrices in the ESSM are given as

$$\tilde{A}_L \overset{\text{def}}{=} \mathcal{O}_L A (\mathcal{O}_L^T \mathcal{O}_L)^{-1} \mathcal{O}_L^T \quad \in \mathbb{R}^{Lm \times Lm} \tag{7.11}$$

$$\tilde{B}_L \overset{\text{def}}{=} \begin{bmatrix} \mathcal{O}_L B & \mathcal{H}_L^d \end{bmatrix} - \tilde{A}_L \begin{bmatrix} \mathcal{H}_L^d & 0_{Lm\times r} \end{bmatrix} \quad \in \mathbb{R}^{Lm \times Lr} \tag{7.12}$$

$$\tilde{C}_L \overset{\text{def}}{=} \begin{bmatrix} \mathcal{O}_L C & \mathcal{H}_L^s \end{bmatrix} - \tilde{A}_L \begin{bmatrix} \mathcal{H}_L^s & 0_{Lm\times l} \end{bmatrix} \quad \in \mathbb{R}^{Lm \times (L+1)l} \tag{7.13}$$

where $\tilde{A}_L \in \mathbb{R}^{Lm\times Lm}$, $\tilde{B}_L \in \mathbb{R}^{Lm\times Lr}$ and $\tilde{C}_L \in \mathbb{R}^{Lm\times (L+1)l}$.
$\triangle$

Proof: See Di Ruscio (1994,1996). Equation (7.12) is presented for $g = 0$ for the sake of simplicity. The reason is definition (7.5) which should be defined similarly to Equation (7.6) when $g = 1$.

The ESSM has some properties:

1. **Minimal ESSM order**. $L_{\min}$, defined in Equation (7.8), defines the minimal *prediction horizon*. However the minimal L is the ceiling function $\lceil \frac{n}{m} \rceil$.
Proof: Same as for proving observability, in this case $(\mathcal{O}_{L_{\min}}^T \mathcal{O}_{L_{\min}})^{-1}$ is non-singular and the existence of the ESSM can be proved.

2. **Uniqueness**. Define M as a non-singular matrix which transforms the state vector x to a new coordinate system. Given two system realizations, (A, B, C, D, F) and $(M^{-1}AM, M^{-1}B, M^{-1}C, DM, F)$. The two realizations give the same ESSM matrices for all $L \geq L_{\min}$.
Proof: The ESSM matrices are invariant under state (coordinate) transformations in the SSM.

3. **Eigenvalues**. The ESSM transition matrix $\tilde{A}_L$ has the same n eigenvalues as the SSM transition matrix A and $Lm - n$ eigenvalues equal to zero.
Proof: From similarity, Equation 7.11.

In the subspace identification algorithm, Di Ruscio (1995,1996), it is shown that the ESSM can be identified directly from a sliding window of known data. However, when the inputs used for identification are poor from a persistent excitation point of view, it is better to identify only the minimal order ESSM as explained in Di Ruscio (1996). Note also that $L_{\min}$ coincides with the demand for persistent excitation of the inputs, i.e., the inputs must at least be persistently exciting of order $L_{\min}$ in order to recover the SSM from known input and output data.

Assume that an ESSM of order J is given and defined as follows

$$y_{k+1|J} = \tilde{A}_J y_{k|J} + \tilde{B}_J u_{k|J+g} + \tilde{C}_J v_{k|J+1} \tag{7.14}$$

If the *prediction horizon* L is chosen greater than J it is simple to construct the following ESSM from (7.14).

$$\tilde{A}_L = \begin{bmatrix} 0_{(L-J)m\times m} & I_{(L-J)m\times(L-1)m} \\ 0_{Jm\times(L-J)m} & \tilde{A}_J \end{bmatrix} \tag{7.15}$$

$$\tilde{B}_L = \begin{bmatrix} 0_{(L-J)m\times(L+g)r} & \\ 0_{Jm\times(L-J)r} & \tilde{B}_J \end{bmatrix} \tag{7.16}$$

$$\tilde{C}_L = \begin{bmatrix} 0_{(L-J)m\times(L+1)l} & \\ 0_{Jm\times(L-J)l} & \tilde{C}_J \end{bmatrix} \tag{7.17}$$

where $L_{min} \leq J < L$. J is defined as the *identification horizon*. This last formulation of the ESSM matrices is attractive from an identification point of view. It will also have some advantages with respect to noise retention compared to using the matrices defined by Equations (7.11) to (7.13) directly (i.e. putting $J = L$) in the model predictive control algorithms which will be presented in Sections 7.3 and 7.4. Note also that a polynomial model, e.g., an ARMAX model, can be formulated directly as an ESSM.

7.2.4 A usefull identity

The following identity will be usefull througout the paper

$$y_{k-L+1|L} = \begin{bmatrix} y_k \\ \vdots \\ y_k \end{bmatrix} - T\Delta y_{k-L+1|L} \quad \in \; \mathbb{R}^{Lm} \tag{7.18}$$

where

$$\Delta y_{k-L+1|L} = y_{k-L+1|L} - y_{k-L|L} \tag{7.19}$$

and the matrix T has zeroes on and below the main diagonal and otherwise ones, i.e.,

$$T = \begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & & & 1 \\ 0 & 0 & \cdots & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{Lm \times Lm}. \tag{7.20}$$

7.3 Basic model based predictive control algorithm

7.3.1 Computing optimal control variables

Prediction model for the future outputs

$$y_{k+1|L} = p_L(k) + F_L u_{k|L+g} \tag{7.21}$$

Scalar control objective criterion

$$\begin{aligned} J_1 = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) \\ + u_{k|L+g}^T R u_{k|L+g} \end{aligned} \tag{7.22}$$

where $r_{k+1|L}$ is a stacked vector of (L) future references. Q and R are (usually) block diagonal weighting matrices.

This is an optimization problem in case of constraints on the control variables. In the case of unconstrained control a closed form solution exists. The (present and future) optimal controls which minimize the control objective are given by

$$u_{k|L+g} = -(R + F_L{}^T Q F_L)^{-1} F_L{}^T Q (p_L(k) - r_{k+1|L}) \tag{7.23}$$

Note that usually only the first control signal vector u_k (in the optimal control vector $u_{k|L+g}$) is computed and used for control, i.e., receding horizon control. It is also possible to include a *control horizon* by including a selection matrix. A similar expression for the optimal control is obtained but with a truncated matrix F_L.

The problem is to find a *prediction model* of the form specified by Equation 7.21). One of our points is that the term $p_L(k)$ representing the past can be computed directly from known past inputs and outputs. This will be shown in Section 7.4.1.

7.3.2 Computing optimal control deviation variables

Prediction model for the future outputs

$$y_{k+1|L} = p_L(k) + F_L \Delta u_{k|L+g} \tag{7.24}$$

Control objective criterion

$$J_2 = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) \\ +\Delta u_{k|L+g}^T R \Delta u_{k|L+g} \tag{7.25}$$

The unconstrained (present and future) optimal control changes which minimize the control objective are given by

$$\Delta u_{k|L+g} = -(R + {F_L}^T Q F_L)^{-1} {F_L}^T Q (p_L(k) - r_{k+1|L}) \tag{7.26}$$

Note that the optimal control deviation (Δu_k) gives offset free control in steady state. This can be argued as follows: assume that the closed loop system is stable. In steady state we have $p_L(k) = y$ and $\Delta u_{k|L} = 0$. From the expression for the optimal control change ($\Delta u_{k|L} = 0$) we must have that $y = \tilde{r}$, i.e. zero steady state offset. (The matrix $(R + F_L^T Q F_L)^{-1} F_L^T Q$ should be of full row rank, r_k and v_k are assumed to be bounded as $k \to \infty$.)

7.4 Extended state space modeling for predictive control

Two *prediction models* are proposed. Both models are dependent upon a known *extended state space* vector. Hence, there is no need for a state estimator. The first model, presented in Section (7.4.1), is in addition dependent on actual control input variables. This model has the same form as (7.21). The second model, presented in Section (7.4.2), is in addition dependent on control input deviation variables. This model has the same form as (7.24).

For the simplicity of notation we will assume that $g = 0$ in this section, i.e. a strictly proper system. The only proper case in which $g = 1$ will be evident because we will treat the case with external inputs v_k, see the SSM Equations (7.1) and (7.2). There is notationally no difference between inputs u_k or external inputs v_k.

7.4.1 Prediction model in terms of process variables

In order to express the right hand side of Equation (7.7) in terms of future inputs (control variables) we write

$$y_{k+1|L} = \tilde{A}^L y_{k-L+1|L} + \sum_{i=1}^{L} \Lambda_i v_{k-i+1|L+1} + \sum_{i=1}^{L} H_i u_{k-i+1|L} \tag{7.27}$$

where

$$\Lambda_i \overset{\text{def}}{=} \tilde{A}_L^{i-1} \tilde{C}_L \quad \text{and} \quad H_i \overset{\text{def}}{=} \tilde{A}_L^{i-1} \tilde{B}_L \tag{7.28}$$

The first term on the right hand side of Equation (7.27) depends on known past and present output data vectors $(y_{k-L+1}, \cdots, y_k)$. The second term depends on known past, present and future external input variables $(v_{k-L+1}, \cdots, v_k, \cdots, v_{k+L})$, e.g. measured noise variables, reference variables etc. The third term depends on past, present and future control input variables, i.e. $(u_{k-L+1}, \cdots, u_k, \cdots, u_{k+L-1})$.

We can separate the right hand side of Equation (7.27) into two terms, one term which is completely described by the model and the known process variables and one term which depends on the, at this stage, unknown future control input variables, i.e., we can define the following *prediction model*

$$y_{k+1|L} = p_L(k) + F_L u_{k|L} \tag{7.29}$$

where the known vector $p_L(k)$ is given by

$$p_L(k) = \tilde{A}_L^L y_{k-L+1|L} + \beta_L^s v_{k-L+1|2L} + P_L u_{k-L+1|L-1} \tag{7.30}$$

The term $p_L(k)$ is completely known, e.g., it depends upon the process model, known past and present process output variables, known past process control input variables and past, present and future external input variables (assumed to be known). $p_L(k)$ can be interpreted as the autonomous process response, i.e. putting future inputs equal to zero in (7.30).

Given the quintuple matrices (A, B, C, D, F). The *prediction model matrices* can be computed as follows. We have one matrix which is proportional to past control inputs (P_L) and one matrix which is proportional to present and future control inputs (F_L).

$$\begin{array}{ll} F_L = \begin{bmatrix} \mathcal{O}_L B \ \mathcal{H}_L^d \end{bmatrix} & \in \ \mathbb{R}^{Lm \times Lr} \\ P_L = \mathcal{O}_L A \mathcal{C}_{L-1}^d - \tilde{A}^L \mathcal{H}_L^d & \in \ \mathbb{R}^{Lm \times (L-1)r} \end{array} \tag{7.31}$$

The *prediction model* matrix β_L^s can be divided into two sub-matrices, one sub-matrix which is related to past external inputs (P_L^s) and one related to present and future external inputs (F_L^s).

$$\begin{array}{ll} \beta_L^s = \begin{bmatrix} P_L^s \ F_L^s \end{bmatrix} & \in \ \mathbb{R}^{Lm \times 2Ll} \\ F_L^s = \begin{bmatrix} \mathcal{O}_L C - \tilde{A}^L E_L^s \ \mathcal{H}_L^s \end{bmatrix} & \in \ \mathbb{R}^{Lm \times (L+1)l} \\ P_L^s = \mathcal{O}_L A \mathcal{C}_{L-1}^s - \tilde{A}^L \mathcal{H}_L^s(:, 1:(L-1)l) & \in \ \mathbb{R}^{Lm \times (L-1)l} \end{array}$$

where E_L^s is the last L-th block column of matrix $\mathcal{H}_L^s$.

Given the ESSM double matrices $(\tilde{A}_L, \tilde{B}_L)$. A procedure for defining the *prediction model* matrices F_L and P_L from the ESSM matrices is given in the following. The matrix β_L^s is computed by a similar procedure. The prediction of future outputs can be expressed in terms of sub-matrices of the impulse response matrices for the extended state space model, Equation (7.28). First, define

$$H_i = \begin{bmatrix} H_{i1} \ H_{i2} \cdots H_{ii} \end{bmatrix} \quad \forall \quad i = 1, \cdots, L \tag{7.32}$$

Then, make a (one block column) Hankel matrix from the impulse response matrices for the ESSM.

$$\begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_L \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1L} \\ H_{21} & H_{22} & \cdots & H_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ H_{L1} & H_{L2} & \cdots & H_{LL} \end{bmatrix} \tag{7.33}$$

Matrix F_L is computed from the upper right part, including the main block diagonal, of the Hankel matrix. The first block column (sub-matrix) in F_L is equal to the sum of the sub-matrices on the main block diagonal of the Hankel matrix. The second block column (sub-matrix) is equal to the sum of the sub-matrices on the block diagonal above the main block diagonal, and so on. That is, the i-th block column in matrix F_L is given by

$$F_{1,i} = \sum_{j=1}^{L-i+1} H_{j,j+i-1} \quad \forall\ i = 1, \cdots, L \tag{7.34}$$

The block columns in P_L are given from the lower left part of the Hankel matrix. The first block column is given by the lower left sub-matrix in the Hankel matrix, i.e., H_{L1}. The i-th block column is given by

$$P_{1,i} = \sum_{j=1}^{i} H_{j+L-i,j} \quad \forall\ i = 1, \cdots, L-1 \tag{7.35}$$

We have illustrated a method for extracting the characteristic matrices of the system, e.g., impulse response matrices, from the ESSM matrices. The ESSM matrices can be identified directly from known input and output data vectors by a subspace identification method for combined deterministic and stochastic systems, see Di Ruscio (1995).

The *prediction model*, given by Equations (7.29) and (7.30), depends upon the known *extended state space* vector and actual process input variables. This *prediction model* used for model predictive control (MPC) is referred to as the *extended state space model predictive control* algorithm number one, i.e., (EMPC$_1$).

The process variables which are necessary for defining the *prediction model* can be pointed out as follows. The *prediction model* in Section (7.4.1) depends upon both past controls and present and past outputs. The L unknown future output vectors $y_{k+1|L}$ can be expressed in terms of the L unknown present and future control vectors $u_{k|L}$, the L known past and present output vectors $y_{k-L+1|L}$ and the $L-1$ known past control vectors $u_{k-L+1|L-1}$.

7.4.2 Prediction model in terms of process deviation variables

The extended state space model can be written in terms of control changes as follows

$$\begin{aligned} y_{k+1|L} &= y_{k|L} + \tilde{A}_L \Delta y_{k|L} + \tilde{B}_L \Delta u_{k|L} + \tilde{C}_L \Delta v_{k|L} \\ \Delta y_{k+1|L} &= \tilde{A}_L \Delta y_{k|L} + \tilde{B}_L \Delta u_{k|L} + \tilde{C}_L \Delta v_{k|L} \end{aligned} \tag{7.36}$$

which can be written

$$\begin{aligned} y_{k+1|L} = y_{k-L+1|L} + (\textstyle\sum_{i=1}^{L} \tilde{A}_L^i)\Delta y_{k-L+1|L} \\ + \textstyle\sum_{i=1}^{L} \Lambda_i \Delta v_{k-i+1|L+1} + \textstyle\sum_{i=1}^{L} H_i \Delta u_{k-i+1|L} \end{aligned} \tag{7.37}$$

where

$$\begin{aligned} \Lambda_i &\overset{\text{def}}{=} \textstyle\sum_{j=1}^{i} \tilde{A}_L^{j-1} \tilde{C}_L & \in\ \mathbb{R}^{Lm\times(L+1)l} \\ H_i &\overset{\text{def}}{=} \textstyle\sum_{j=1}^{i} \tilde{A}_L^{j-1} \tilde{B}_L & \in\ \mathbb{R}^{Lm\times Lr} \end{aligned} \tag{7.38}$$

We have the following *prediction model*

$$y_{k+1|L} = p_L(k) + F_L \Delta u_{k|L} \tag{7.39}$$

where

$$\begin{aligned} p_L(k) = y_{k-L+1|L} &+ (\textstyle\sum_{i=1}^{L} \tilde{A}_L^i)\Delta y_{k-L+1|L} \\ &+ \beta_L^s \Delta v_{k-L+1|2L} + P_L \Delta u_{k-L+1|L-1} \end{aligned} \tag{7.40}$$

$p_L(k)$ can be interpreted as the autonomous process response, i.e. putting future input deviation variables equal to zero.

In the following a procedure for computing the matrix β_L^s is given. The procedure for computing F_L and P_L is the same as that presented in Section 7.4.1. Define the (one block column) Hankel matrix

$$\begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \vdots \\ \Lambda_L \end{bmatrix} = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} & \cdots & \Lambda_{1L} & \Lambda_{1,L+1} \\ \Lambda_{21} & \Lambda_{22} & \cdots & \Lambda_{2L} & \Lambda_{2,L+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Lambda_{L1} & \Lambda_{L2} & \cdots & \Lambda_{LL} & \Lambda_{L,L+1} \end{bmatrix} \tag{7.41}$$

then we have

$$\beta_L^s = \begin{bmatrix} P_L^s & F_L^s \end{bmatrix} \tag{7.42}$$

$$F_L^s = \begin{bmatrix} F_{1,1}^s & \cdots & F_{1,i}^s & \cdots & F_{1,L}^s & F_{1,L+1}^s \end{bmatrix} \tag{7.43}$$

$$P_L^s = \begin{bmatrix} P_{1,1}^s & \cdots & P_{1,i}^s & \cdots & P_{1,L-1}^s \end{bmatrix} \tag{7.44}$$

where the i-th block columns in F_L^s and P_L^s are given by

$$F_{1,i}^s = \sum_{j=1}^{L-i+1} \Lambda_{j,j+i-1} \quad \forall\ i = 1, \cdots, L+1 \tag{7.45}$$

$$P_{1,i}^s = \sum_{j=1}^{i} \Lambda_{j+L-i,j} \quad \forall\ i = 1, \cdots, L-1 \tag{7.46}$$

The first term on the right hand side of $p_L(k)$, Equation (7.40), is equal to the *extended state space* vector $y_{k-L+1|L}$ and the other terms are functions of deviation variables. This observation is useful for analyzing the steady state behavior of the final controlled system, i.e., we can prove zero steady state off-set in the final EMPC$_2$ algorithm, provided the disturbances and set-points are bounded as time approaches infinity. If the disturbances are not bounded, they should be modeled and the model included in the SSM.

7.4.3 Prediction model in terms of the present state vector

The predictions generated from a general linear state space model can be written as

$$\begin{aligned} y_{k+1|L} = \mathcal{O}_L A x_k &+ \begin{bmatrix} \mathcal{O}_L C & \mathcal{H}_L^s \end{bmatrix} v_{k|L+1} \\ &+ \begin{bmatrix} \mathcal{O}_L B & \mathcal{H}_L^d \end{bmatrix} u_{k|L} \end{aligned} \tag{7.47}$$

which fits into the standard *prediction model* form when x_k is known or computed by a state estimator. This is referred to as *state space model predictive control* (SSMPC).

Note that the state vector x_k can be expressed in terms of past and present process outputs and past process inputs. We have then

$$x_k = A^{L-1}x_{k-L+1} + \mathcal{C}^s_{L-1}v_{k-L+1|L-1} + \mathcal{C}^d_{L-1}u_{k-L+1|L-1} \tag{7.48}$$

The state vector in the past, x_{k-L+1}, can be computed from

$$y_{k-L+1|L} = \mathcal{O}_L x_{k-L+1} + \mathcal{H}^s_L v_{k-L+1|L} + \mathcal{H}^d_L u_{k-L+1|L-1} \tag{7.49}$$

when observability is assumed. This gives the same result as in Section 7.4.1, and illustrates the proof of the *prediction model* matrices given in (7.31) and (7.32).

7.5 Comparison and connection with existing algorithms

7.5.1 Generalized predictive control

The EMPC$_2$ algorithm presented in Sections 7.4.2 and 7.3.2 has a strong connection to the generalized predictive control algorithm proposed by Clarke *et al.* (1987). The past *identification horizon* J was included in in the EMPC strategy in order to estimate the present state. If J is chosen equal to the minimal horizon L_{min} then the EMPC strategy becomes similar to the GPC strategy. By choosing $L_{min} < J \leq L$ we have more degrees in freedom for tuning the final controller.

7.5.2 Finite impulse response modeling for predictive control

The prediction models discussed in this section have their origin from the following formulation of the SSM, Equations (7.1) and (7.2).

$$\begin{aligned} y_{k+1} = DA^L x_{k-M+1} + D\mathcal{C}^s_M v_{k-M+1|M} \\ + D\mathcal{C}^d_M u_{k-M+1|M} \end{aligned} \tag{7.50}$$

For the sake of simplicity, assume in the following next two sections that the external inputs are zero.

Modeling in terms of process variables

Assume that the transition matrix A has all eigenvalues strictly inside the unit circle in the complex plane. In this case it makes sense to assume that $A^M \approx 0$ for some scalar $M \geq 1$. Hence, neglecting the first term in Equation (7.50) gives

$$y_{k+1} \approx D\mathcal{C}^d_M u_{k-M+1|M} \tag{7.51}$$

where M is defined as the *model horizon*. The prediction model is then given by

$$y_{k+1|L} \approx P_L u_{k-M+1|M-1} + F_L u_{k|L} \tag{7.52}$$

where

$$F_L = \left[\mathcal{O}_L B \ \mathcal{H}_L^d \right] \tag{7.53}$$

$$P_L = \begin{bmatrix} DA^{M-1}B & DA^{M-2}B & \cdots & DAB \\ 0 & DA^{M-1}B & \cdots & DA^2B \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & DA^{M-1}B \end{bmatrix}$$

where $F_L \in \mathbb{R}^{Lm \times Lr}$ and $P_L \in \mathbb{R}^{Lm \times (M-1)r}$.

Modeling in terms of process deviation variables

Neglecting the first term on the right hand side of Equation (7.50) and taking the difference between y_{k+1} and y_k gives the finite impulse response model in terms of process control deviation variables

$$y_{k+1} \approx y_k + D\mathcal{C}_M^d \Delta u_{k-M+1|M} \tag{7.54}$$

where M is defined as the *model horizon*. The L step ahead predictor is given by

$$y_{k+L} \approx y_k + D\mathcal{C}_M^d \sum_{i=1}^{L} \Delta u_{k+i-M|M} \tag{7.55}$$

It is in this case straightforward to put the predictions into standard form

$$y_{k+1|L} \approx p_L(k) + F_L \Delta u_{k|L} \tag{7.56}$$

where

$$p_L(k) = \left[y_k^T \ \cdots \ y_k^T \right]^T + P_L \Delta u_{k-M+1|M-1} \tag{7.57}$$

The matrices F_L and P_L can be computed from a strategy similar to the one in Section 7.4.2. This type of prediction model is used in the DMC strategy.

7.6 Constraints, stability and time delay

Constraints are included in the EMPC strategy in the traditional way, e.g. by solving a QP problem. This is an appealing and practical strategy because the number of flops is bounded (this may not be the case for non-linear MPC when using SQP).

Stability of the receding horizon strategy is not adressed in this section. However, by using classical stability results from the infinite horizon linear quadratic optimal control theory, it is expected that closed loop stability is ensured by chosing the prediction horizon large. Stability of MPC is adressed in e.g., Clarke and Scattolini (1991) and Mosca and Zhang (1992).

In the case of time delay an SSM for the delay can be augmented into the process model and the resulting SSM used directly in the EMPC strategy.

7.7 Conclusions

The EMPC algorithm depends on the ESSM matrices $\tilde{A}_L$ and $\tilde{B}_L$. These matrices can be computed directly from the model, either a state space model or a polynomial model. Both the state space model and the extended state space model can be identified directly from a sliding window of past input and output data vectors, e.g., by the subspace algorithm for combined Deterministic and Stochastic system Realization and identification (DSR), Di Ruscio (1995).

The size of the prediction horizon L must be greater than or equal to the minimal observability index L_{min} in order to construct the ESSM. It is also a lower bound for the *prediction horizon* in order to ensure stability of the unconstrained EMPC algorithms. L_{min} is equal to the necessary number of block rows in the observability matrix $\mathcal{O}_L$ in order to ensure that $\mathcal{O}_L^T \mathcal{O}_L$ is non-singular.

A framework for computing model based predictive control is presented. The EMPC algorithms, the *generalized predictive control* (GPC) algorithm and the predictive *dynamic matrix control* (DMC) algorithm fit into this framework. Different *prediction models* give different methods for model based predictive control.

The DMC algorithm depends on past inputs and the present output. The EMPC algorithms depend on both past inputs and past and present outputs.

References

Albertos, P. and R. Ortega (1987). On Generalized Predictive Control: Two alternative formulations. *Automatica*, Vol. 25, No.5, pp. 753-755.

Bitmead, R. R., Gevers, M. and V. Wertz (1990). *Adaptive Optimal Control: The Thinking Man's GPC.* Prentice Hall.

Camacho, E. F. and C. Bordons (1995). *Model predictive control in the process industry.* Springer-Verlag, Berlin.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987). Generalized Predictive Control -Part I. The Basic Algorithm. *Automatica*, Vol. 23, No.2, pp. 137-148.

Clarke, D. W. and R. Scattolini (1991). Constrained receding horizon predictive control. *Proc. IEE Part D*, vol. 138, pp. 347-354.

Di Ruscio, D. (1994). Methods for the identification of state space models from input and output measurements. *SYSID 94, The 10th IFAC Symposium on System Identification*, Copenhagen, July 4 - 6.

Di Ruscio, D. (1995). A Method for Identification of Combined Deterministic and Stochastic Systems: Robust implementation. *Third European Control Conference ECC95*, September 5-8, 1995.

Di Ruscio, D. (1997). A method for identification of combined deterministic stochastic systems. In: *Applications of Computer Aided Time Series Modeling*, Lecture Notes in Statistics 119, Eds. M. Aoki and A. M. Havenner, Springer Verlag, ISBN 0-387-94751-5.

Larimore, W. E. (1983). System identification, reduced order filtering and modeling via canonical variate analysis. *Proc. of the American Control Conference*, San Francisco, USA, pp. 445-451.

Larimore, W. E. (1990). Canonical Variate Analysis in Identification, Filtering and Adaptive Control. *Proc. of the 29th Conference on Decision and Control*, Honolulu, Hawaii, December 1990, pp. 596-604.

Mosca, E. and J. Zhang (1992). Stable redesign of predictive control. *Automatica*, Vol. 28, No.2, pp. 1229-1233.

Van Overschee, P. and B. De Moor (1994). N4SID: Subspace Algorithms for the Identification of Combined Deterministic Stochastic Systems. *Automatica*, vol. 30, No. 1, pp.75-94.

Van Overschee, P. (1995). *Subspace Identification: theory-implementation-application.* PhD thesis, Katholieke Universiteit Leuven, Belgium.

Van Overschee, P. and B. De Moor (1995). A Unifying Theorem for Three Subspace System Identification Algorithms. *Automatica*, vol. 31, No. 12, pp. 1853-1864.

Verhagen, M. (1994). Identification of the deterministic part of MIMO state space models given on innovations form from input output data. *Automatica*, vol. 30, No. 1, pp. 61-74.

Viberg, M. (1995). Subspace-Based Methods for the Identification of Linear Time-invariant Systems. *Automatica*, vol. 31, No. 12, pp. 1835-1851.

7.8 Numerical examples

7.8.1 Example 1

Given a SSM. The first step in the GPC algorithm is to transform the SSM to a polynomial model. There are no reasons for doing this when the EMPC algorithms are applied. However, if (one does, or) a Polynomial Model (PM) is given, we have the following solution

Given the polynomial model, Camacho and Bordons (1995), p. 27:

$$y_k = a_1 y_{k-1} + b_0 u_{k-1} + b_1 u_{k-2} \tag{7.58}$$

Model parameters $a_1 = 0.8$, $b_0 = 0.4$ and $b_1 = 0.6$.

A prediction horizon of $L = 3$ is chosen. The transformation from the PM to the ESSM is simple, first make

$$y_{k+1} = y_{k+1} \tag{7.59}$$

$$y_{k+2} = y_{k+2} \tag{7.60}$$

$$y_{k+3} = a_0 y_{k+2} + b_0 u_{k+2} + b_1 u_{k+1} \tag{7.61}$$

which gives the ESSM

$$\overbrace{\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \end{bmatrix}}^{y_{k+1|3}} = \overbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & a_1 \end{bmatrix}}^{\tilde{A}_3} \overbrace{\begin{bmatrix} y_k \\ y_{k+1} \\ y_{k+2} \end{bmatrix}}^{y_{k|3}} + \overbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & b_1 & b_0 \end{bmatrix}}^{\tilde{B}_3} \overbrace{\begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \end{bmatrix}}^{u_{k|3}}$$

Using the algorithm in Section 7.4.2 we get the following *prediction model*

$$\overbrace{\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \end{bmatrix}}^{y_{k+1|3}} = p_3(k) + \overbrace{\begin{bmatrix} 0.4 & 0 & 0 \\ 1.32 & 0.4 & 0 \\ 2.056 & 1.32 & 0.4 \end{bmatrix}}^{F_3} \overbrace{\begin{bmatrix} u_k - u_{k-1} \\ u_{k+1} - u_k \\ u_{k+2} - u_{k+1} \end{bmatrix}}^{\Delta u_{k|3}} \tag{7.62}$$

The vector p_3 which is dependent upon the *prediction model* and known past and present process variables is given by

$$p_3(k) = \overbrace{\begin{bmatrix} y_{k-2} \\ y_{k-1} \\ y_k \end{bmatrix}}^{y_{k-2|3}} + \overbrace{\begin{bmatrix} 0 & 1 & 1.8 \\ 0 & 0 & 2.44 \\ 0 & 0 & 1.952 \end{bmatrix}}^{\tilde{A}_3 + \tilde{A}_3^2 + \tilde{A}_3^3} \overbrace{\begin{bmatrix} y_{k-2} - y_{k-3} \\ y_{k-1} - y_{k-2} \\ y_k - y_{k-1} \end{bmatrix}}^{\Delta y_{k-2|3}}$$

$$+\overbrace{\begin{bmatrix} 0 & 0.6 \\ 0 & 1.08 \\ 0 & 1.464 \end{bmatrix}}^{P_3}\overbrace{\begin{bmatrix} u_{k-2}-u_{k-3} \\ u_{k-1}-u_{k-2} \end{bmatrix}}^{\Delta u_{k-2|2}} \tag{7.63}$$

This prediction model is identical to the prediction model obtained when applying the GPC algorithm, see Camacho and Bordon (1995), pp. 28. A critical step in the GPC algorithm is the solution of a Diophantine equation. The EMPC algorithm proposes an alternative and direct derivation of the *prediction model*, either from a SSM or a polynomial model.

7.8.2 Example 2

The polynomial model, Equation (7.58), has the following SSM equivalent

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ 0 & a_1 \end{bmatrix} \quad B = \begin{bmatrix} b_0 \\ b_1 + a_1 b_0 \end{bmatrix} \\ D &= \begin{bmatrix} 1 & 0 \end{bmatrix} \quad E = 0 \end{aligned} \tag{7.64}$$

Model parameters $a_1 = 0.8$, $b_0 = 0.4$ and $b_1 = 0.6$.

If the ESSM matrices given by Equations (7.15) and (7.16) with $L = 3$ and $J = 2$ are used to formulate the *prediction model* in Section 7.4.2, then we get identical results as with the GPC algorithm.

Equations (7.11) and (7.12) with $L = 3$ give the following ESSM

$$\overbrace{\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \end{bmatrix}}^{y_{k+1|3}} = \overbrace{\begin{bmatrix} 0 & 0.6098 & 0.4878 \\ 0 & 0.4878 & 0.3902 \\ 0 & 0.3902 & 0.3122 \end{bmatrix}}^{\tilde{A}_3}\overbrace{\begin{bmatrix} y_k \\ y_{k+1} \\ y_{k+2} \end{bmatrix}}^{y_{k|3}}$$

$$+\overbrace{\begin{bmatrix} -0.2927 & -0.1951 & 0 \\ 0.3659 & 0.2439 & 0 \\ 0.2927 & 0.7951 & 0.4 \end{bmatrix}}^{\tilde{B}_3}\overbrace{\begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \end{bmatrix}}^{u_{k|3}}$$

Using the algorithm in Section 7.4.2 we get the following *prediction model*

$$\overbrace{\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \end{bmatrix}}^{y_{k+1|3}} = p_3(k)$$

$$+\overbrace{\begin{bmatrix} 0.4 & 0 & 0 \\ 1.32 & 0.4 & 0 \\ 2.056 & 1.32 & 0.4 \end{bmatrix}}^{F_3}\overbrace{\begin{bmatrix} u_k - u_{k-1} \\ u_{k+1} - u_k \\ u_{k+2} - u_{k+1} \end{bmatrix}}^{\Delta u_{k|3}} \tag{7.65}$$

The vector p_3 which is dependent upon the *prediction model* and known past and present process variables is given by

$$p_3(k) = \overbrace{\begin{bmatrix} y_{k-2} \\ y_{k-1} \\ y_k \end{bmatrix}}^{y_{k-2|3}}$$
$$+ \overbrace{\begin{bmatrix} 0 & 1.4878 & 1.1902 \\ 0 & 1.1902 & 0.9522 \\ 0 & 0.9522 & 0.7618 \end{bmatrix}}^{\tilde{A}_3+\tilde{A}_3^2+\tilde{A}_3^3} \overbrace{\begin{bmatrix} y_{k-2} - y_{k-3} \\ y_{k-1} - y_{k-2} \\ y_k - y_{k-1} \end{bmatrix}}^{\Delta y_{k-2|3}}$$
$$+ \overbrace{\begin{bmatrix} 0.3659 & 0.8439 \\ 0.8927 & 1.6751 \\ 0.7141 & 1.9401 \end{bmatrix}}^{P_3} \overbrace{\begin{bmatrix} u_{k-2} - u_{k-3} \\ u_{k-1} - u_{k-2} \end{bmatrix}}^{\Delta u_{k-2|2}} \tag{7.66}$$

$p_3(k)$ is defined from known past variables. Putting (7.65) into (7.26) gives the optimal unconstrained control deviations $\Delta u_{k|3}$.

7.8.3 Example 3

This example shows how a finite impulse response model fits into the theory presented in this paper. This also illustrate the fact that the DMC strategy is a special case of the theory which are presented in the paper.

Considder the following truncated impulse response model

$$y_{k+1} = \sum_{i=1}^{M} h_i u_{k-i+1} = h_1 u_k + h_2 u_{k-1} + h_3 u_{k-2} \tag{7.67}$$

with $M = 3$, $h_1 = 0.4$, $h_2 = 0.92$ and $h_3 = 0.416$.

The prediction horizon is specified to be $L = 5$. Using (7.67) with $L - 1$ artifical states $y_{k+1} = y_{k+1} \ldots$, $y_{k+4} = y_{k+4}$ we have the following ESSM model

$$\overbrace{\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ y_{k+4} \\ y_{k+5} \end{bmatrix}}^{y_{k+1|5}} = \overbrace{\begin{bmatrix} 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \\ 0&0&0&0&0 \end{bmatrix}}^{\tilde{A}_5} \overbrace{\begin{bmatrix} y_k \\ y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ y_{k+4} \end{bmatrix}}^{y_{k|5}} + \overbrace{\begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&h_1&h_2&h_3 \end{bmatrix}}^{\tilde{B}_5} \overbrace{\begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ u_{k+3} \\ u_{k+4} \end{bmatrix}}^{u_{k|5}} \tag{7.68}$$

Using the algorithm presented in Section 7.4.2 with $L = 5$ and $\tilde{A}_5$ and $\tilde{B}_5$ as specified above, then we get the following prediction model

$$y_{k+1|5} = p_5(k) + \overbrace{\begin{bmatrix} 0.4 & 0 & 0 & 0 & 0 \\ 1.32 & 0.4 & 0 & 0 & 0 \\ 1.736 & 1.32 & 0.4 & 0 & 0 \\ 1.736 & 1.736 & 1.32 & 0.4 & 0 \\ 1.736 & 1.736 & 1.736 & 1.32 & 0.4 \end{bmatrix}}^{F_5} \overbrace{\begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \Delta u_{k+3} \\ \Delta u_{k+4} \end{bmatrix}}^{\Delta u_{k|5}} \tag{7.69}$$

where the vector $p_5(k)$ is given by

$$p_5(k) = \begin{bmatrix} y_k \\ y_k \\ y_k \\ y_k \\ y_k \end{bmatrix} + \overbrace{\begin{bmatrix} 0.416 & 0.920 \\ 0.416 & 1.336 \\ 0.416 & 1.336 \\ 0.416 & 1.336 \\ 0.416 & 1.336 \end{bmatrix}}^{P_5} \overbrace{\begin{bmatrix} \Delta u_{k-2} \\ \Delta u_{k-1} \end{bmatrix}}^{\Delta u_{k-M+1|M-1}} \tag{7.70}$$

Finaly, it should be noted that if a *control horizon* $C \le L$, say $C = 2$, is specified, then the above prediction model reduces to

$$y_{k+1|5} = p_5(k) + \overbrace{\begin{bmatrix} 0.4 & 0 \\ 1.32 & 0.4 \\ 1.736 & 1.32 \\ 1.736 & 1.736 \\ 1.736 & 1.736 \end{bmatrix}}^{F_5} \overbrace{\begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \end{bmatrix}}^{\Delta u_{k|2}} \tag{7.71}$$

However, note that the prediction horizon L must be choosen such that $L \geq M$. The control horizon is then bounded by $1 \geq C \leq L$. Note also that Cutler (1982) suggest setting $L = M + C$ for the DMC strategy. Note also that the truncated impulse response model horizon M usually must be choosen large, typically in the range of 20 to 70.

Chapter 8

Constraints for Model Predictive Control

8.1 Constraints of the type $Au_{k|L} \leq b$

Common types of constraints are

1. System input amplitude constraints.
2. System input rate constraints.
3. System output constraints.

These constraints can be written as the following linear inequality

$$Au_{k|L} \leq b, \tag{8.1}$$

where the matrix A and the vector b are discussed below.

8.1.1 System Input Amplitude Constraints

It is common with amplitude constraints on the input signal to almost any practical control system. Such constraints can be formulated as

$$u_{\min} \leq u_{k|L} \leq u_{\max}. \tag{8.2}$$

These linear constraints can be formulated more convenient (standard form constraints for QP problems) as

$$\begin{bmatrix} I \\ -I \end{bmatrix} u_{k|L} \leq \begin{bmatrix} u_{\max} \\ -u_{\min} \end{bmatrix}. \tag{8.3}$$

8.1.2 System Input Rate Constraints

$$\Delta u_{\min} \leq \Delta u_{k|L} \leq \Delta u_{\max}. \tag{8.4}$$

Note that $\Delta u_{k|L} = u_{k|L} - u_{k-1|L}$. This relationship is not sufficient to formulate a matrix inequality describing the constraints, because the term $u_{k-1|L}$ on the right hand side, consists of dependent variables when $L > 1$. This is however solved by introducing the relationship

$$u_{k|L} = S\Delta u_{k|L} + cu_{k-1}, \tag{8.5}$$

which gives

$$\Delta u_{k|L} = S^{-1}u_{k|L} - S^{-1}cu_{k-1}. \tag{8.6}$$

Substituting (8.6) into the linear constraints (8.5) can then be formulated as the following matrix inequality

$$\begin{bmatrix} I \\ -I \end{bmatrix} u_{k|L} \leq \begin{bmatrix} \Delta u_{\max} + S^{-1}cu_{k-1} \\ -\Delta u_{\min} - S^{-1}cu_{k-1} \end{bmatrix}. \tag{8.7}$$

8.1.3 System Output Amplitude Constraints

System output constraints are defined as follows

$$y_{\min} \leq y_{k+1|L} \leq y_{\max}. \tag{8.8}$$

Combination with the prediction model in terms of control change variables

$$y_{k+1|L} = p_L(k) + F_L\Delta u_{k|L}, \tag{8.9}$$

or simply

$$y_{k+1|L} = p_L(k) + F_L u_{k|L}, \tag{8.10}$$

gives

$$\begin{bmatrix} F_L \\ -F_L \end{bmatrix} u_{k|L} \leq \begin{bmatrix} y_{\max} - p_L(k) \\ -y_{\min} + p_L(k) \end{bmatrix}. \tag{8.11}$$

8.1.4 Input Amplitude, Input Change and Output Constraints

The three types of constraints in (8.3), (8.7) and (8.11) can be combined and described by the following linear inequality

$$\overbrace{\begin{bmatrix} I \\ -I \\ I \\ -I \\ F_L \\ -F_L \end{bmatrix}}^{A} u_{k|L} \leq \overbrace{\begin{bmatrix} u_{\max} \\ -u_{\min} \\ \Delta u_{\max} + S^{-1}cu_{k-1} \\ -\Delta u_{\min} - S^{-1}cu_{k-1} \\ y_{\max} - p_L(k) \\ -y_{\min} + p_L(k) \end{bmatrix}}^{b}. \tag{8.12}$$

One should note that state constraints can be included similarly.

8.2 Solution by Quadratic Programming

Given the control objective criterion

$$J_2 = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + \Delta u_{k|L}^T P \Delta u_{k|L} \tag{8.13}$$

and the prediction model

$$y_{k+1|L} = p_L(k) + F_L \Delta u_{k|L} \tag{8.14}$$

Substituting the prediction model into J_2 gives

$$J_2 = \Delta u_{k|L}^T H \Delta u_{k|L} + 2 f_2^T \Delta u_{k|L} + J_{2,0} \tag{8.15}$$

where

$$\begin{aligned} H &= P + F_L^T Q F_L && (8.16)\\ f_2 &= F_L^T Q (p_L(k) - r_{k+1|L}) && (8.17)\\ J_{2,0} &= (p_L(k) - r_{k+1|L})^T Q (p_L(k) - r_{k+1|L}) && (8.18) \end{aligned}$$

This objective functional is expressed in terms of input change variables. The relationship between $\Delta u_{k|L}$ and $u_{k|L}$ can be used in order to express J_2 in terms of actual input variables.

Substituting the vector of control change variables

$$\Delta u_{k|L} = u_{k|L} - u_{k-1|L} \tag{8.19}$$

into the control objective criterion J_2 gives

$$J_2 = u_{k|L}^T H u_{k|L} + 2 f^T u_{k|L} + J_0 \tag{8.20}$$

where

$$\begin{aligned} H &= P + F_L^T Q F_L && (8.21)\\ f &= F_L^T Q (p_L(k) - r_{k+1|L}) - H u_{k-1|L} && (8.22)\\ J_0 &= (p_L(k) - r_{k+1|L})^T Q (p_L(k) - r_{k+1|L}) + u_{k-1|L}^T H u_{k-1|L} \\ &- 2 F_L^T Q (p_L(k) - r_{k+1|L}) u_{k-1|L} && (8.23) \end{aligned}$$

or equivalently

$$\begin{aligned} H &= P + F_L^T Q F_L && (8.24)\\ f &= f_2 - H u_{k-1|L} && (8.25)\\ J_0 &= J_{2,0} + u_{k-1|L}^T H u_{k-1|L} - 2 f_2^T u_{k-1|L} && (8.26) \end{aligned}$$

We have the following QP problem

$$\min_{u_{k|L}} (u_{k|L}^T H u_{k|L} + 2 f^T u_{k|L}) \tag{8.27}$$

$$\begin{gathered} \text{subject to:} \\ A u_{k|L} \le b \end{gathered} \tag{8.28}$$

which e.g. can be solved in MATLAB by the Optimization toolbox function QP, i.e.

$$u_{k|L} = \text{qp}(H, f, A, b) \tag{8.29}$$

In order to compute $u_{k|L}$ from the QP problem $u_{k-1|L}$ must be known. Initially $u_{k-1|L}$ must be specified in order to start the control strategy. This means that the QP problem must be initialized with unknown future control input signals. The simplest solution is to initially putting $u_{k-1|L} = [u_{k-1}^T \cdots u_{k-1}^T]^T$. However, this can be a drawback with this strategy.

The above QP problem can be reformulated in order to overcome this drawback. The relationship between $\Delta u_{k|L}$ and $u_{k|L}$ can be written as

$$\Delta u_{k|L} = S_2 u_{k|L} - c_2 u_{k-1} \tag{8.30}$$

S_2 and c_2 ar matrices with ones and zeroes. Substituting into the objective functional Equation (8.15) gives

$$J_2 = u_{k|L}^T H u_{k|L} + 2f^T u_{k|L} + J_0 \tag{8.31}$$

where

$$H = S_2^T H S_2 \tag{8.32}$$
$$f = f_2 - S_2^T H c_2 u_{k-1} \tag{8.33}$$
$$J_0 = J_{2,0} + (c_2 u_{k-1})^T H c_2 u_{k-1} - 2 f_2^T c_2 u_{k-1} \tag{8.34}$$

At time instant k the vector f and the scalar J_0 are determined in terms of known process variables.

Chapter 9

More on constraints and Model Predictive Control

9.1 The Control Problem

Consider a linear quadratic (LQ) objective functional

$$J = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + \Delta u_{k|L}^T R \Delta u_{k|L} + u_{k|L}^T R_2 u_{k|L} \quad (9.1)$$

We will study the problem of minimizing J with respect to the future control inputs, subject to input and output constraints.

This problem can be formulated as follows:

$$\min_{\Delta u_{k|L}} J \quad (9.2)$$

subject to:

$$\begin{array}{rcccl} u_{k|L}^{\min} & \leq & u_{k|L} & \leq & u_{k|L}^{\max} \text{ (input amplitude constraints)} \\ \Delta u_{k|L}^{\min} & \leq & \Delta u_{k|L} & \leq & \Delta u_{k|L}^{\max} \text{ (input change constraints)} \\ y_{k+1|L}^{\min} & \leq & y_{k+1|L} & \leq & y_{k+1|L}^{\max} \text{ (output constraints)} \end{array} \quad (9.3)$$

9.2 Prediction Model

The prediction model is assumed to be of the form

$$y_{k+1|L} = p_L(k) + F_L \Delta u_{k|L} \quad (9.4)$$

where $p_L(k)$ is defined in terms of known past inputs and outputs. F_L is a constant lower triangular matrix.

9.3 Constraints

The constraints (9.3) can be written as an equivalent linear inequality:

$$A \Delta u_{k|L} \leq b \quad (9.5)$$

where

$$A = \begin{bmatrix} S \\ -S \\ I \\ -I \\ F_L \\ -F_L \end{bmatrix}, \quad b = \begin{bmatrix} u_{k|L}^{\max} - cu_{k-1} \\ u_{k|L}^{\min} + cu_{k-1} \\ \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L}^{\min} \\ y_{k+1|L}^{\max} - p_L(k) \\ -y_{k+1|L}^{\min} + p_L(k) \end{bmatrix} \tag{9.6}$$

This will be prooved in the next subsections.

9.3.1 Relationship between $\Delta u_{k|L}$ and $u_{k|L}$

It is convenient to find the relationship between $\Delta u_{k|L}$ and $u_{k|L}$ in order to formulate the constraints (9.3) in terms of future deviation variables $\Delta u_{k|L}$.

We have

$$u_{k|L} = S\Delta u_{k|L} + cu_{k-1} \tag{9.7}$$

where

$$S = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ I & I & 0 & \cdots & 0 \\ I & I & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ I & I & I & \cdots & I \end{bmatrix}, \quad c = \begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix} \tag{9.8}$$

The following alternative relation is also useful:

$$\Delta u_{k|L} = S_2 u_{k|L} - c_2 u_{k-1} \tag{9.9}$$

where

$$S_2 = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ -I & I & 0 & \cdots & 0 \\ 0 & -I & I & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & I \end{bmatrix}, \quad c_2 = \begin{bmatrix} I \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{9.10}$$

Note that $S_2 = S^{-1}$ and $c_2 = S^{-1}c$.

Note also that:

$$u_{k-1|L} = (I - S_2)u_{k|L} + c_2 u_{k-1}. \tag{9.11}$$

9.3.2 Input amplitude constraints

The constraints

$$u_{k|L}^{\min} \le u_{k|L} \le u_{k|L}^{\max} \tag{9.12}$$

is equivalent to

$$\begin{aligned} S\Delta u_{k|L} &\le u_{k|L}^{\max} - cu_{k-1} \\ -S\Delta u_{k|L} &\le -u_{k|L}^{\min} + cu_{k-1} \end{aligned} \tag{9.13}$$

9.3.3 Input change constraints

The constraints

$$\Delta u_{k|L}^{\min} \le u_{k|L} \le \Delta u_{k|L}^{\max} \tag{9.14}$$

is equivalent to

$$\begin{aligned} \Delta u_{k|L} &\le \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L} &\le -\Delta u_{k|L}^{\min} \end{aligned} \tag{9.15}$$

9.3.4 Output constraints

The constraints

$$y_{k+1|L}^{\min} \le y_{k+1|L} \le y_{k+1|L}^{\max} \tag{9.16}$$

By using the prediction model, Equation (9.4), we have the equivalent constraints:

$$\begin{aligned} F_L \Delta u_{k|L} &\le y_{k+1|L}^{\max} - p_L(k) \\ -F_L \Delta u_{k|L} &\le -y_{k+1|L}^{\min} + p_L(k) \end{aligned} \tag{9.17}$$

9.4 Solution by Quadratic Programming

The LQ objective functional Equation (9.1) can be written as:

$$J = \Delta u_{k|L}^T H \Delta u_{k|L} + 2f^T \Delta u_{k|L} + J_0 \tag{9.18}$$

where

$$H = R + F_L^T Q F_L + S^T R_2 S \tag{9.19}$$

$$f = F_L^T Q(p_L(k) - r_{k+1|L}) + S^T R_2 c u_{k-1} \tag{9.20}$$

$$J_0 = (p_L(k) - r_{k+1|L})^T Q(p_L(k) - r_{k+1|L}) + u_{k-1}^T c^T R_2 c u_{k-1} \tag{9.21}$$

H is a constant matrix which is referred to as the Hessian matrix.. It is assumed that H is positive definit (i.e. $H > 0$). f is a vector which is independent of the unknown present and future inputs. f is defined by the model, a sequence of known past inputs and outputs (inkluding y_k). Similarly J_0 is a known scalar.

The problem can be solved by the following QP problem

$$\min_{\Delta u_{L|L}} (\Delta u_{k|L}^T H \Delta u_{k|L} + 2f^T \Delta u_{k|L}) \tag{9.22}$$

$$\begin{aligned} &\text{subject to:} \\ &A\Delta u_{k|L} \le b \end{aligned} \tag{9.23}$$

The QP problem can e.g. be solved in MATLAB by the Optimization toolbox function QP, i.e.

$$\Delta u_{k|L} = \mathrm{qp}(H, f, A, b) \tag{9.24}$$

Chapter 10

MODEL PREDICTIVE CONTROL AND IDENTIFICATION: A Linear State Space Model Approach

10.1 Introduction

A recursive subspace algorithm (RDSR) for the identification of state space model matrices are presented in [2]. The state space model matrices which are identified by the RDSR algorithm can be used for model based predictive control. A state space Model based Predictive Control (MPC) algorithm which is based on the DSR subspace identification algorithm is presented in [1]. However, due to space limitations in this paper, only unconstrained predictive control is considered.

In this work we will show that input and output constraints can be incorporated into the state space model based control algorithm.

The properties of the MPC algorithm as well as comparisons with other MPC algorithms are presented in the paper, [1]. A short review of the predictive control algorithm which is extended to handle constraints, is presented in this paper.

The MPC algorithm will be demonstrated on a three input, two output MIMO process. The process is a Thermo Mechanical Pulping (TMP) refiner.

10.2 Model Predictive Control

10.2.1 State space process model

Consider a process which can be described by the following linear, discrete time invariant state space model (SSM)

$$x_{k+1} = Ax_k + Bu_k + Cv_k \tag{10.1}$$

$$y_k = Dx_k + Eu_k + Fv_k \tag{10.2}$$

where the integer $k \geq 0$ is discrete time, $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^r$ is the control input vector, $v_k \in \mathbb{R}^l$ is an external input vector and $y_k \in \mathbb{R}^m$ is the output vector. The constant matrices in the SSM are of appropriate dimensions. A is the *state transition matrix*, B is the *control input* matrix, C is the *external input* matrix, D is the *output* matrix, E is the direct *control input to output* (feed-through) matrix, and F is the direct *external input to output* matrix.

In continuous time systems the matrix E is usually zero. This is not the case in discrete time systems due to sampling.

The following assumptions are made: The pair (D, A) is observable. The pair (A, B) is controllable. When deling with the MPC algorithm we will assume that the vector of external input signals is known.

10.2.2 The control objective:

A standard discrete time Linear Quadratic (LQ) objective functional is used. The LQ objective can for convenience be written in compact matrix form as follows:

$$\mathcal{J}_k = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + \Delta u_{k|L}^T R \Delta u_{k|L} + u_{k|L}^T \mathcal{P} u_{k|L}, \tag{10.3}$$

where $r_{k+1|L}$ is a vector of future references, $y_{k+1|L}$ is a vector of future outputs, $\Delta u_{k|L}$ is a vector of future input changes, and $u_{k|L}$ is a vector of future inputs.

$$r_{k+1|L} = \begin{bmatrix} r_{k+1}^T & r_{k+2}^T & \cdots & r_{k+L}^T \end{bmatrix}^T \in \mathbb{R}^{Lm}, \tag{10.4}$$

$$y_{k+1|L} = \begin{bmatrix} y_{k+1}^T & y_{k+2}^T & \cdots & y_{k+L}^T \end{bmatrix}^T \in \mathbb{R}^{Lm}, \tag{10.5}$$

$$\Delta u_{k|L} = \begin{bmatrix} \Delta u_k^T & \Delta u_{k+1}^T & \cdots & \Delta u_{k+L-1}^T \end{bmatrix}^T \in \mathbb{R}^{Lr}, \tag{10.6}$$

$$u_{k|L} = \begin{bmatrix} u_k^T & u_{k+1}^T & \cdots & u_{k+L-1}^T \end{bmatrix}^T \in \mathbb{R}^{Lr}. \tag{10.7}$$

The weighting matrices are defined as follows:

$$Q = \begin{bmatrix} Q_k & 0 & \cdots & 0 \\ 0 & Q_{k+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_{k+L-1} \end{bmatrix} \in \mathbb{R}^{Lm \times Lm}, \tag{10.8}$$

$$R = \begin{bmatrix} R_k & 0 & \cdots & 0 \\ 0 & R_{k+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{k+L-1} \end{bmatrix} \in \mathbb{R}^{Lr \times Lr}, \tag{10.9}$$

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_k & 0 & \cdots & 0 \\ 0 & \mathcal{P}_{k+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{P}_{k+L-1} \end{bmatrix} \in \mathbb{R}^{Lr \times Lr} \quad . \tag{10.10}$$

Note that both the actual control input vector $u_{k|L}$ and the control input change vector $\Delta u_{k|L}$ can be weighted in the control objective. Note also that the objective-weighting matrices Q, P and R can in general be time varying.

We will study the problem of minimizing $\mathcal{J}_k$ with respect to the future control input change vector $(\Delta u_{k|L})$, subject to linear constraints on the input change, the input amplitude and the output.

Problem 10.1 (MPC problem)

$$\min_{\Delta u_{k|L}} \mathcal{J}_k \tag{10.11}$$

with respect to linear constraints on u_k, Δu_k and y_k.

To solve this problem we

- need a prediction model for (prediction of) $y_{k+1|L}$:

$$y_{k+1|L} = p_L(k) + F_L \Delta u_{k|L} \tag{10.12}$$

- need a relationship between $u_{k|L}$ and $\Delta u_{k|L}$:

$$u_{k|L} = S\Delta u_{k|L} + c u_{k-1} \tag{10.13}$$

  where

$$S = \begin{bmatrix} I_r & 0_r & 0_r & \cdots & 0_r \\ I_r & I_r & 0_r & \cdots & 0_r \\ I_r & I_r & I_r & \cdots & 0_r \\ \vdots & \vdots & \vdots & \ddots & 0_r \\ I_r & I_r & I_r & \cdots & I_r \end{bmatrix} \in \mathbb{R}^{Lr \times Lr}, \quad c = \begin{bmatrix} I_r \\ I_r \\ I_r \\ \vdots \\ I_r \end{bmatrix} \in \mathbb{R}^{Lr \times r} \tag{10.14}$$

  where I_r is the $r \times r$ identity matrix and 0_r is the $r \times r$ matrix of zeroes.

- need a linear inequality of the form

$$\mathcal{A}\Delta u_{k|L} \leq b_k \tag{10.15}$$

  describing the linear constraints.

When $\Delta u_{k|L}$ is computed we can

- apply $u_k = u_{k-1} + \Delta u_k$ as the control input to the process (Note that only the first change in $\Delta u_{k|L}$ is used, i.e., a receding horizon strategy).

- or use $u_k = u_{k-1} + \Delta u_k$ for operating support.

10.2.3 Prediction Model:

The MPC algorithm is based on the observation that the general linear state space model, Equations (10.1) and (10.2), can be transformed into a model which is convenient for prediction and predictive control. This prediction model is independent of the state vector x_k. Hence, there is no need for a state observer.

There are two horizons involved in the MPC algorithm.

$$\text{Horizons in the MPC algorithm} \quad \begin{cases} \text{Identification horizon: } J \\ \text{Prediction horizon: } \quad L \end{cases}$$

An identification horizon J into the past is defined in order to construct an estimate of the state vector. Known past input and output vectors defined over the identification horizon is used to eliminate the state vector from the state space model. The resulting model (prediction model) can be written as

$$y_{k+1|L} = p_L(k) + F_L \Delta u_{k|L} \tag{10.16}$$

where L is the prediction horizon, $y_{k+1|L}$ is a vector of future output predictions, $\Delta u_{k|L}$ is a vector of future input change vectors.

$$y_{k+1|L} = \begin{bmatrix} y_{k+1}^T \; y_{k+2}^T \; \cdots \; y_{k+L}^T \end{bmatrix}^T \; \in \; \mathbb{R}^{Lm} \tag{10.17}$$

$$\Delta u_{k|L} = \begin{bmatrix} \Delta u_k^T \; \Delta u_{k+1}^T \; \cdots \; \Delta u_{k+L-1}^T \end{bmatrix}^T \; \in \; \mathbb{R}^{Lr} \tag{10.18}$$

Note that $\Delta u_k = u_k - u_{k-1}$. F_L is a constant lower triangular matrix which is a function of the known state space model matrices. $p_L(k)$ represents the information of the past which is used to predict the future. $p_L(k)$ is a known vector. This vector is a function of a number J known past input and output vectors, and the state space model matrices. A simple algorithm for constructing F_L and $p_L(k)$ from the known state space model matrices and the past inputs and outputs are proposed.

- A simple algorithm to compute $p_L(k)$ from the SSM matrices (A, B, C, D, E, F) and known past inputs and outputs is presented in [1].

10.2.4 Constraints:

The following constraints are considered,

$$\begin{array}{rcccl} u_{k|L}^{\min} & \leq & u_{k|L} & \leq & u_{k|L}^{\max} \text{ (input amplitude constraints)} \\ \Delta u_{k|L}^{\min} & \leq & \Delta u_{k|L} & \leq & \Delta u_{k|L}^{\max} \text{ (input change constraints)} \\ y_{k+1|L}^{\min} & \leq & y_{k+1|L} & \leq & y_{k+1|L}^{\max} \text{ (output constraints)} \end{array} \tag{10.19}$$

The constraints (10.19) can be written as an equivalent linear inequality:

$$\mathcal{A} \Delta u_{k|L} \leq b_k. \tag{10.20}$$

where $\mathcal{A}$ is a constant matrix and b_k is a vector which is defined in terms of the specified constraints. See Appendix D for details.

10.2.5 Solution by Quadratic Programming:

The control problem is to minimize the control objective given by Equation (10.3) subject to the constraints given by the linear inequality, Equation (10.20). Substituting the prediction model, Equation (10.16), into the control objective, Equation (10.3), we have the following alternative formulation of the control objective

$$\mathcal{J}_k = \Delta u_{k|L}^T H \Delta u_{k|L} + 2 f_k^T \Delta u_{k|L} + \mathcal{J}_k^0 \tag{10.21}$$

which is on a form suitable for computing the future control input change vectors $\Delta u_{k|L}$. See Appendix D1 and E for details.

The control problem can now be formulated as follows:

$$\min_{\Delta u_{k|L}} \mathcal{J}_k \tag{10.22}$$

subject to constraints:

$$\mathcal{A} \Delta u_{k|L} \leq b_k. \tag{10.23}$$

where $\mathcal{J}_k$ is given by (10.21), $\mathcal{A}$ and b_k is defined in the appendix.

This can be solved as a Quadratic Programming (QP) problem. The QP problem can be solved in MATLAB by the Optimization toolbox function qp. We have

$$\Delta u_{k|L} = \mathrm{qp}(H, f_k, \mathcal{A}, b_k) \tag{10.24}$$

Remark 10.1 The MPC algorithm is based on a general linear state space model. However, there is no need for a state observer (e.g. Kalman filter).

An estimate of the state vector is constructed from a number of past inputs and outputs. This state estimate is used to eliminate the state vector from the model.

Remark 10.2 The predictive control algorithm can be shown to give offset free control in the unconstrained case, when $\mathcal{P}$ in the control objective (10.3).

Remark 10.3 Usually a receding horizon strategy is used, i.e. only the first control change $\Delta u_k = u_k - u_{k-1}$ in $\Delta u_{k|L}$ is used (and computed). The control at time instant k is then $u_k = \Delta u_k + u_{k-1}$ where u_{k-1} is known from the previous iteration.

Remark 10.4 The MPC algorithm can be formulated so that the actual future control input variables $u_{k|L}$ are directly computed, instead of computing the control change variables $\Delta u_{k|L}$ as shown in this work.

Remark 10.5 Unconstrained control is given by

$$\Delta u_{k|L} = -H^{-1} f_k \tag{10.25}$$

Equivalently

$$\Delta u_{k|L} = -(R + F_L^T Q F_L + S^T \mathcal{P} S)^{-1} [F_L^T Q (p_L(k) - r_{k+1|L}) + S^T \mathcal{P} c u_{k-1}] \tag{10.26}$$

See the appendix for details.

10.3 MPC of a thermo mechanical pulping plant

The process considered in this section is a Thermo Mechanical Pulping (TMP) plant at Union Co, Skien, Norway. A key part in this process is a Sunds Defibrator double (rotating) disk RGP 68 refiner.

Identification and model predictive control of a TMP refiner are addressed. The process variables considered for identification and control is the refiner specific energy and the refiner consistency.

In TMP plants both the specific energy and the consistency are usually subject to setpoint control. It is common to use two single PI control loops. The process input used to control the specific energy is the plate gap. The process input used to control the consistency is the dilution water.

In this section, the MPC algorithm will be demonstrated on the TMP refiner (simulation results only). The refiner is modeled by a MIMO (3-input, 2-output and 6-state) state space model. Model predictive control of a TMP refiner has to our knowledge not yet been demonstrated.

10.3.1 Description of the process variables

Input and output time series from a TMP refiner are presented in Figures 10.1 and 10.2. The time series is the result of a statistical experimental design .

The manipulable input variables

$$\text{Refiner input variables} \begin{cases} u_1 : \text{Plug screw speed, } [rpm] \\ u_2 : \text{Flow of dilution water, } [\frac{kg}{s}] \\ u_3 : \text{Plate gap, } [mm] \end{cases}$$

The following vector of input variables is defined

$$u_k = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}_k \in \mathbb{R}^3. \tag{10.27}$$

The output variables
The process outputs used for identification is defined as follows

$$\text{Refiner output variables} \begin{cases} y_1 : \text{Refiner specific energy, } [\frac{MWh}{1000kg}] \\ y_2 : \text{Refiner consistency, } [\%] \end{cases}$$

The following vector of process output variables is defined

$$y_k = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_k \in \mathbb{R}^2. \tag{10.28}$$

10.3.2 Subspace identification

Input and output time series from a TMP refiner are presented in Figures 10.1 and 10.2. The time series is the result of a statistical experimental design .

The problem is to identify a state space model, including the system order (n), for both the deterministic part and the stochastic part of the system, i.e., the quadruple matrices (A, B, D, E) and the double matrices (C, F), respectively, directly from known system input and output data vectors (or time series). The SSM is assumed to be on innovations form (Kalman filter).

The known process input and output data vectors from the TMP process can be defined as follows

$$\left.\begin{array}{l} u_k \ \forall \ k = 1, \ldots, N \\ y_k \ \forall \ k = 1, \ldots, N \end{array}\right\} \textbf{Known data}$$

For the TMP refiner example we have used:

$$\begin{aligned} N = 1500 \ [samples] & \text{ for model identification} \\ 860 \ [samples] & \text{ for model validation} \end{aligned}$$

Given sequences with process input and output raw data. The first step in a data modeling procedure is usually to analyze the data for trends and time delays. Trends should preferably be removed from the data and the time series should be adjusted for time delays. The trend of a time series can often be estimated as the sample mean which represents some working point. Data preprocessing is not necessary but it often increase the accuracy of the estimated model.

The following constant trends (working points) are removed from the refiner input and output data.

$$u^0 = \begin{bmatrix} 52.3 \\ 7.0 \\ 0.58 \end{bmatrix}, \quad y^0 = \begin{bmatrix} 1.59 \\ 34.3 \end{bmatrix} \tag{10.29}$$

The simplicity of the subspace method [6] will be illustrated in the following. Assume that the data is adjusted for trends and time delays. Organize the process output and input data vectors y_k and u_k as follows

$$\overbrace{Y = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_N^T \end{bmatrix} \in \mathbb{R}^{N \times m}}^{\textbf{Known}\text{ data matrix of output variables}} \tag{10.30}$$

$$\overbrace{U = \begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_N^T \end{bmatrix} \in \mathbb{R}^{N \times r}}^{\textbf{Known}\text{ data matrix of input variables}} \tag{10.31}$$

The problem of identifying a complete (usually) dynamic model for the process can be illustrated by the following function (similar to a matlab function).

$$\begin{bmatrix} A, \ B, \ C, \ D, \ E, \ F \end{bmatrix} = \mathrm{DSR}(Y, U, \mathcal{L}) \tag{10.32}$$

where the sixfold matrices (A, B, C, D, E, F) are the state space model matrices in Equations (10.1) and (10.2). The algorithm name **DSR** stands for Deterministic and Stochastic model Realization and identification, see [1]- [7] for details. $\mathcal{L}$ is a positive integer parameter which should be specified by the user. The parameter $\mathcal{L}$ defines an upper limit for the unknown system order n. The user must chose the system order by inspection of a plot with Singular Values (SV) or Principal Angles (PA). The system order n is identified as the number of "non-zero" SV's or "non-zero" PA's. Note that the Kalman filter gain is given by $K = CF^{-1}$ and that the covariance matrix of the noise innovation process is given by $\mathrm{E}(v_k v_k^T) = FF^T$.

$\mathcal{L} = 3$ and $n = 6$ where chosen in this example.

10.3.3 Model predictive control

The following weighting matrices are used in the control objective, Equation (10.3):

$$Q_i = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}, \quad R_i = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \quad \mathcal{P}_i = 0_r, \ \forall \ i = 1, \cdots, L. \tag{10.33}$$

The following horizons are used

$$\text{Horizons in MPC algorithm} \begin{cases} L = 6, & \text{the prediction horizon} \\ J = 6, & \text{the identification horizon} \end{cases}$$

The following constraints are specified

$$u_k^{\text{min}} = \begin{bmatrix} 50.0 \\ 6.0 \\ 0.5 \end{bmatrix}, \quad u_k^{\text{max}} = \begin{bmatrix} 54.0 \\ 8.0 \\ 0.7 \end{bmatrix} \ \forall \ k > 0 \tag{10.34}$$

$$\Delta u_k^{\text{min}} = \begin{bmatrix} -0.5 \\ -0.15 \\ -0.05 \end{bmatrix}, \quad \Delta u_k^{\text{max}} = \begin{bmatrix} 0.5 \\ 0.15 \\ 0.05 \end{bmatrix} \ \forall \ k > 0 \tag{10.35}$$

$$y_k^{\text{min}} = \begin{bmatrix} 1.5 \\ 33 \end{bmatrix}, \quad y_k^{\text{max}} = \begin{bmatrix} 1.7 \\ 40 \end{bmatrix} \ \forall \ k > 0 \tag{10.36}$$

Simulation results are illustrated in Figures 10.3 and 10.4.

Bibliography

[1] D. Di Ruscio, " Extended state space model based predictive control," Submitted to the ECC 1997, 1-4 July, Brussels, Belgium.

[2] D. Di Ruscio, "Recursive implementation of a subspace identification algorithm, RDSR," Submitted to the *11th IFAC Symposium on System Identification*, July 8-11, Fukouka, Japan.

[3] D. Di Ruscio, "A Method for Identification of Combined Deterministic and Stochastic Systems", In: *Applications of Computer Aided Time Series Modeling*, Eds: Aoki, M. and A. Hevenner, Lecture Notes in Statistics Series, Springer Verlag, 1997, pp. 181-235.

[4] D. Di Ruscio, "A Method for identification of combined deterministic and stochastic system," *In Proceedings of the European Control Conference, ECC95*, Roma, Italy, September 5-8, 1995.

[5] D. Di Ruscio, "Methods for the identification of state space models from input and output measurements," *SYSID 94, The 10th IFAC Symposium on System Identification*, Copenhagen, July 4 - 6, 1994. Also in: *Modeling, Identification and Control*, Vol. 16, no. 3, 1995.

[6] "**DSR**: Algorithm for combined deterministic and stochastic system identification and realization of state space models from observations," *Windows program commercial available by Fantoft Process AS, Box 306, N-1301 Sandvika.*

[7] "**DSR Toolbox for Matlab**: Algorithm for combined deterministic and stochastic system identification and realization of state space models from observations," *Commercial available by Fantoft Process AS, Box 306, N-1301 Sandvika.*

[8] K. R. Muske and J. B. Rawlings, "Model Predictive Control with Linear Models," *AIChE Journal*, Vol. 39, No. 2, 1993, pp. 262-287.

10.4 Appendix: Some details of the MPC algorithm

10.5 The Control Problem

Consider a linear quadratic (LQ) objective functional

$$\mathcal{J}_k = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + \Delta u_{k|L}^T R \Delta u_{k|L} + u_{k|L}^T \mathcal{P} u_{k|L} \tag{10.37}$$

We will study the problem of minimizing $\mathcal{J}_k$ with respect to the future control inputs, subject to input and output constraints.

This problem can be formulated as follows:

$$\min_{\Delta u_{k|L}} \mathcal{J}_k \tag{10.38}$$

subject to:

$$\begin{array}{rcccll} u_{k|L}^{\min} & \leq & u_{k|L} & \leq & u_{k|L}^{\max} & \text{(input amplitude constraints)} \\ \Delta u_{k|L}^{\min} & \leq & \Delta u_{k|L} & \leq & \Delta u_{k|L}^{\max} & \text{(input change constraints)} \\ y_{k+1|L}^{\min} & \leq & y_{k+1|L} & \leq & y_{k+1|L}^{\max} & \text{(output constraints)} \end{array} \tag{10.39}$$

10.6 Prediction Model

The prediction model is assumed to be of the form

$$y_{k+1|L} = p_L(k) + F_L \Delta u_{k|L} \tag{10.40}$$

where $p_L(k)$ is defined in terms of the known state space model matrices (A, B, C, D, F) and known past inputs (both control inputs and external inputs) and outputs. F_L is a constant lower triangular matrix.

10.7 Constraints

The constraints (10.39) can be written as an equivalent linear inequality:

$$\mathcal{A} \Delta u_{k|L} \leq b_k \tag{10.41}$$

where

$$\mathcal{A} = \begin{bmatrix} S \\ -S \\ I_{Lr} \\ -I_{Lr} \\ F_L \\ -F_L \end{bmatrix}, \quad b_k = \begin{bmatrix} u_{k|L}^{\max} - c u_{k-1} \\ -u_{k|L}^{\min} + c u_{k-1} \\ \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L}^{\min} \\ y_{k+1|L}^{\max} - p_L(k) \\ -y_{k+1|L}^{\min} + p_L(k) \end{bmatrix} \tag{10.42}$$

This will be proved in the next subsections.

10.7.1 Relationship between $\Delta u_{k|L}$ and $u_{k|L}$

It is convenient to find the relationship between $\Delta u_{k|L}$ and $u_{k|L}$ in order to formulate the constraints (10.39) in terms of future deviation variables $\Delta u_{k|L}$.

We have

$$u_{k|L} = S\Delta u_{k|L} + c u_{k-1} \tag{10.43}$$

where

$$S = \begin{bmatrix} I_r & 0_r & 0_r & \cdots & 0_r \\ I_r & I_r & 0_r & \cdots & 0_r \\ I_r & I_r & I_r & \cdots & 0_r \\ \vdots & \vdots & \vdots & \ddots & 0_r \\ I_r & I_r & I_r & \cdots & I_r \end{bmatrix} \in \mathbb{R}^{Lr\times Lr}, \quad c = \begin{bmatrix} I_r \\ I_r \\ I_r \\ \vdots \\ I_r \end{bmatrix} \in \mathbb{R}^{Lr\times r} \tag{10.44}$$

where L_r is the $r \times r$ identity matrix and 0_r is the $r \times r$ matrix of zeroes.

The following alternative relation is also useful:

$$\Delta u_{k|L} = S_2 u_{k|L} - c_2 u_{k-1} \tag{10.45}$$

where

$$S_2 = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ -I & I & 0 & \cdots & 0 \\ 0 & -I & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & I \end{bmatrix}, \quad c_2 = \begin{bmatrix} I \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{10.46}$$

Note that $S_2 = S^{-1}$ and $c_2 = S^{-1}c$.

Note also that:

$$u_{k-1|L} = (I - S_2)u_{k|L} + c_2 u_{k-1}. \tag{10.47}$$

10.7.2 Input amplitude constraints

The constraints

$$u_{k|L}^{\min} \leq u_{k|L} \leq u_{k|L}^{\max} \tag{10.48}$$

is equivalent to

$$\begin{aligned} S\Delta u_{k|L} &\leq u_{k|L}^{\max} - c u_{k-1} \\ -S\Delta u_{k|L} &\leq -u_{k|L}^{\min} + c u_{k-1} \end{aligned} \tag{10.49}$$

10.7.3 Input change constraints

The constraints

$$\Delta u_{k|L}^{\min} \leq u_{k|L} \leq \Delta u_{k|L}^{\max} \tag{10.50}$$

is equivalent to

$$\begin{aligned} \Delta u_{k|L} &\leq \quad \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L} &\leq -\Delta u_{k|L}^{\min} \end{aligned} \tag{10.51}$$

10.7.4 Output constraints

The constraints

$$y_{k+1|L}^{\min} \leq y_{k+1|L} \leq y_{k+1|L}^{\max} \tag{10.52}$$

By using the prediction model, Equation (10.40), we have the equivalent constraints:

$$\begin{aligned} F_L \Delta u_{k|L} &\leq \quad y_{k+1|L}^{\max} - p_L(k) \\ -F_L \Delta u_{k|L} &\leq -y_{k+1|L}^{\min} + p_L(k) \end{aligned} \tag{10.53}$$

10.8 Solution by Quadratic Programming

The LQ objective functional Equation (10.37) can be written as:

$$\mathcal{J}_k = \Delta u_{k|L}^T H \Delta u_{k|L} + 2 f_k^T \Delta u_{k|L} + \mathcal{J}_k^0 \tag{10.54}$$

where

$$H = R + F_L^T Q F_L + S^T \mathcal{P} S \tag{10.55}$$

$$f_k = F_L^T Q (p_L(k) - r_{k+1|L}) + S^T \mathcal{P} c u_{k-1} \tag{10.56}$$

$$\mathcal{J}_k^0 = (p_L(k) - r_{k+1|L})^T Q (p_L(k) - r_{k+1|L}) + u_{k-1}^T c^T \mathcal{P} c u_{k-1} \tag{10.57}$$

H is a constant matrix which is referred to as the Hessian matrix. It is assumed that H is positive definite (i.e. $H > 0$). f_k is a vector which is independent of the unknown present and future control inputs. f_k is defined by the SSM model matrices, a sequence of known past inputs and outputs (including y_k). Similarly $\mathcal{J}_k^0$ is a known scalar.

The problem can be solved by the following QP problem

$$\min_{\Delta u_{L|L}} (\Delta u_{k|L}^T H \Delta u_{k|L} + 2 f_k^T \Delta u_{k|L}) \tag{10.58}$$

$$\begin{aligned} &\text{subject to:} \\ &\mathcal{A} \Delta u_{k|L} \leq b_k \end{aligned} \tag{10.59}$$

The QP problem can e.g. be solved in MATLAB by the Optimization toolbox function QP, i.e.

$$\Delta u_{k|L} = \text{qp}(H, f_k, \mathcal{A}, b_k) \tag{10.60}$$

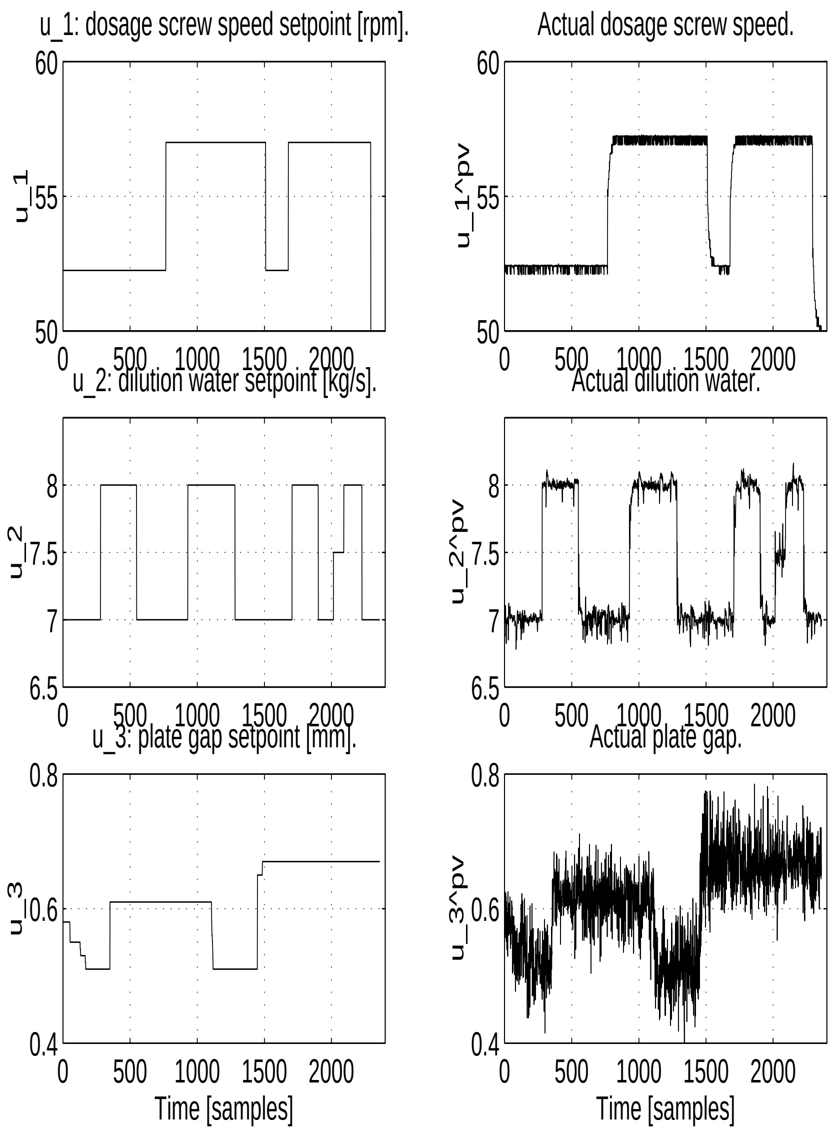

Figure 10.1: Input time series from a TMP plant at Union Bruk. The inputs are from an experimental design. The manipulable input variables are u_1, u_2 and u_3 These inputs are setpoints to local input controllers. The outputs from the local controllers (controlled variables) are shown to the left and denoted u_1^{pv}, u_2^{pv} and u_3^{pv}.

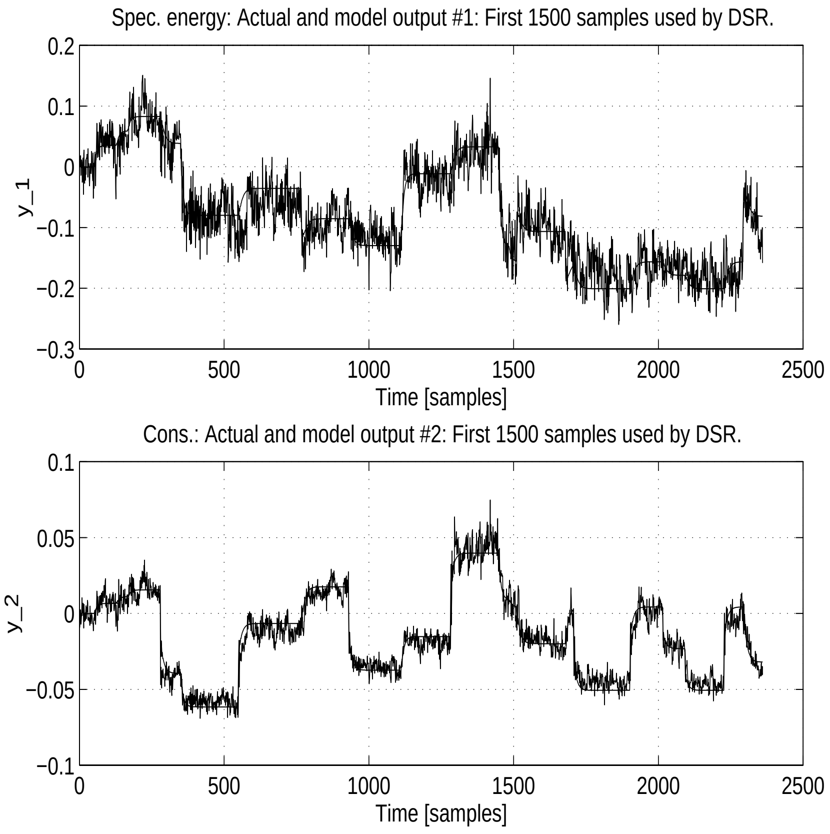

Figure 10.2: Actual and model output time series. The actual output time series is from a TMP plant at Union Bruk. The corresponding input variables are shown in Figure 10.1.

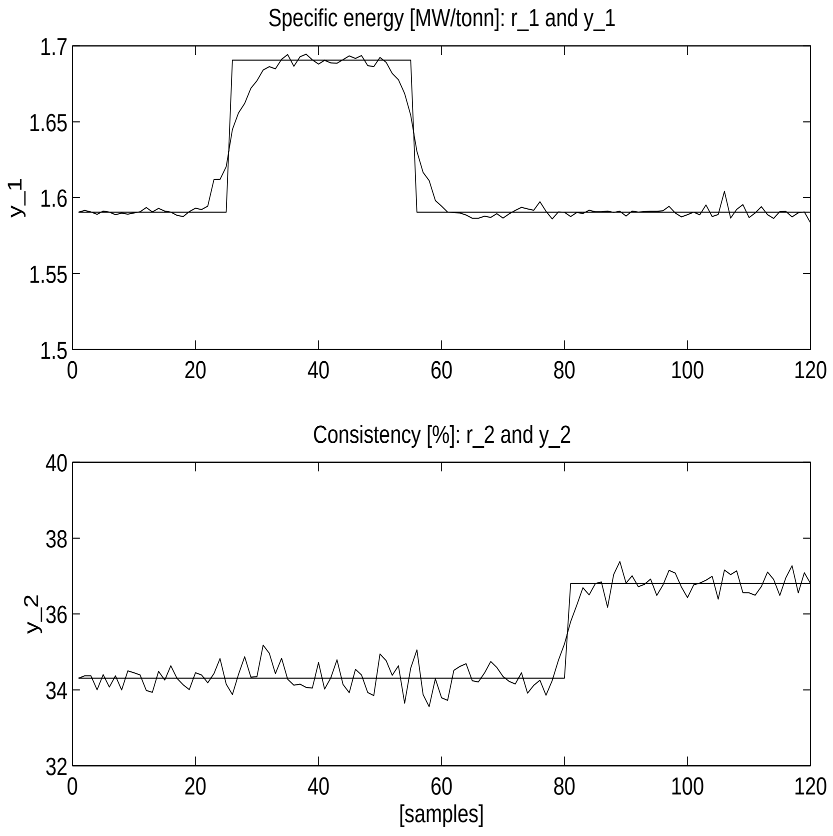

Figure 10.3: Simulation results of the MPC algorithm applied on a TMP refiner. The known references and process outputs are shown.

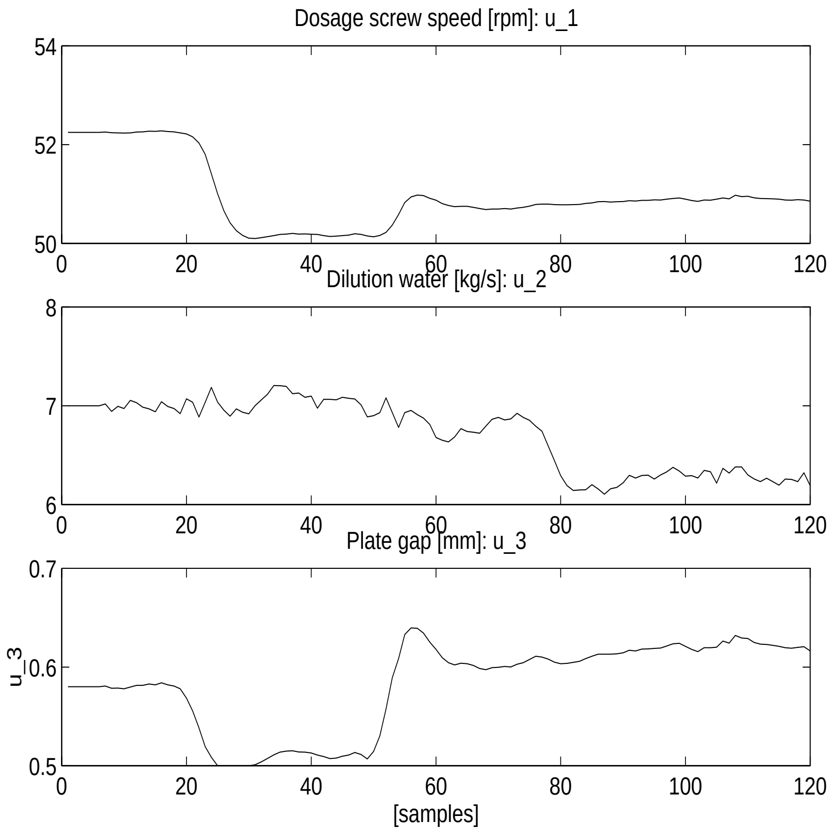

Figure 10.4: Simulation results of the MPC algorithm applied on a TMP refiner. The optimal control inputs are shown.

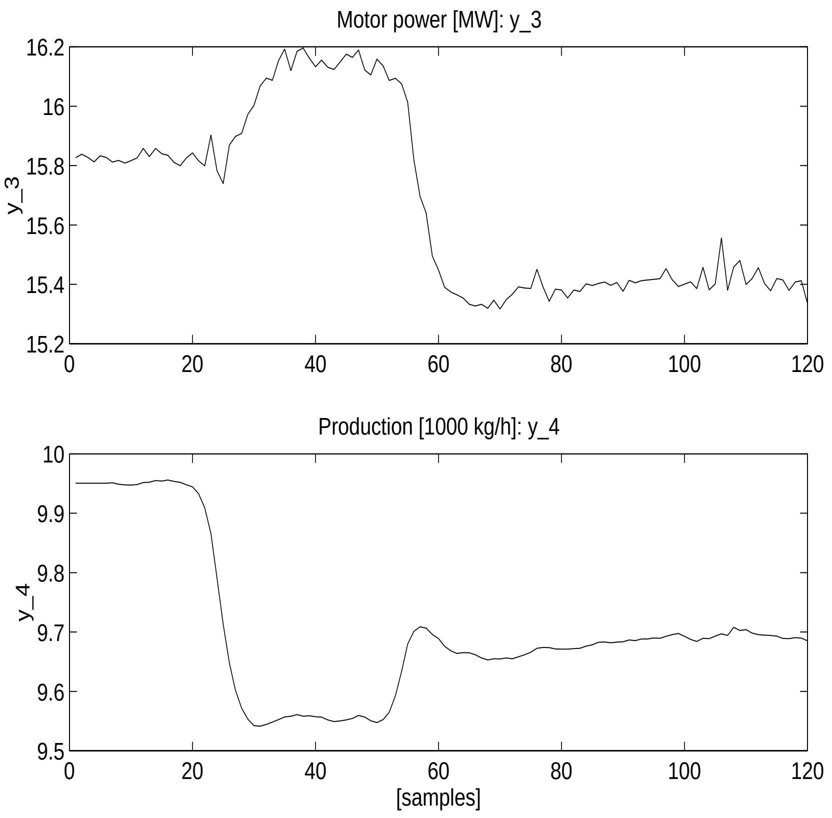

Figure 10.5: Simulation results of the MPC algorithm applied on a TMP refiner.

Chapter 11

EMPC: The case with a direct feed trough term in the output equation

11.1 Introduction

The MPC algorithms presented in the literature are as far as we know, usually only considering the case with a strictly proper state space model (SSM), i.e., an SSM with output equation $y_k = Dx_k$. In this note we describe how strictly proper systems, i.e., an SSM with output equations $y_k = Dx_k + Eu_k$, are implemented in the EMPC algorithm. This problem has many solutions and it is not self-evident which of these are the best.

This paper is organized as follows: The Linear Quadratic (LQ) objective in case of a direct feed through term in the output equation is discussed in Section 11.3. The EMPC algorithm in case of only proper systems is presented in Section 11.4. The equivalence between the solution to the unconstrained MPC problem and the solution to the Linear Quadratic (LQ) optimal control problem and the corresponding Riccati equation is presented in Section 11.5. Furthermore, the possibility of incorporating a final state weighting in the objective is presented. This can be used to ensure nominal stability of the controlled system. Finally, a new and explicit non-iterative solution to the discrete time Riccati equation is presented. Finally some concluding remarks follows in Section 11.7.

11.2 Problem and basic definitions

We assume a linear discrete time invariant process model of the form

$$x_{k+1} = Ax_k + Bu_k + Cr_k, \tag{11.1}$$

$$y_k = Dx_k + Eu_k, \tag{11.2}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^r$ and $y_k \in \mathbb{R}^m$ are the control input and output vectors, respectively. $r_k \in \mathbb{R}^m$ is a known reference vector. The quintuple matrices (A, B, C, D, E) are of appropriate dimensions. The reason for the reference term in the state equation is that this is the result if an integrator $z_{k+1} = z_k + r_k - y_k$ is augmented into a standard state equation $x_{k+1} = Ax_k + Bu_k$, in order to have offset-free steady state output control. The traditional case is obtained by putting $C = 0$.

Definition 11.1 (basic matrix definitions)
The extended observability matrix, O_i, for the pair (D, A) is defined as

$$O_i \overset{\text{def}}{=} \begin{bmatrix} D \\ DA \\ \vdots \\ DA^{i-1} \end{bmatrix} \in \mathbb{R}^{im\times n}, \tag{11.3}$$

where the subscript i denotes the number of block rows.

The reversed extended controllability matrix, C_i^d, for the pair (A, B) is defined as

$$C_i^d \overset{\text{def}}{=} \begin{bmatrix} A^{i-1}B \; A^{i-2}B \cdots B \end{bmatrix} \in \mathbb{R}^{n\times ir}, \tag{11.4}$$

where the subscript i denotes the number of block columns. The lower block triangular Toeplitz matrix, H_i^d, for the quadruple matrices (D, A, B, E)

$$H_i^d \overset{\text{def}}{=} \begin{bmatrix} E & 0_{m\times r} & 0_{m\times r} & \cdots & 0_{m\times r} \\ DB & E & 0_{m\times r} & \cdots & 0_{m\times r} \\ DAB & DB & E & \cdots & 0_{m\times r} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ DA^{i-2}B & DA^{i-3}B & DA^{i-4}B & \cdots & E \end{bmatrix} \in \mathbb{R}^{im\times(i+g-1)r}, \tag{11.5}$$

where the subscript i denotes the number of block rows and $i + g - 1$ is the number of block columns. Where $0_{m\times r}$ denotes the $m \times r$ matrix with zeroes.

The following extended L-block diagonal weighting matrices are defined for later use:

$$Q = \begin{bmatrix} Q_k & 0_{m\times m} & \cdots & 0_{m\times m} \\ 0_{m\times m} & Q_{k+1} & \cdots & 0_{m\times m} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{m\times m} & 0_{m\times m} & \cdots & Q_{k+L-1} \end{bmatrix} \in \mathbb{R}^{Lm\times Lm}, \tag{11.6}$$

$$R = \begin{bmatrix} R_k & 0_{r\times r} & \cdots & 0_{r\times r} \\ 0_{r\times r} & R_{k+1} & \cdots & 0_{r\times r} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{r\times r} & 0_{r\times r} & \cdots & R_{k+L-1} \end{bmatrix} \in \mathbb{R}^{Lr\times Lr}, \tag{11.7}$$

$$P = \begin{bmatrix} P_k & 0_{r\times r} & \cdots & 0_{r\times r} \\ 0_{r\times r} & P_{k+1} & \cdots & 0_{r\times r} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{r\times r} & 0_{r\times r} & \cdots & P_{k+L-1} \end{bmatrix} \in \mathbb{R}^{Lr\times Lr} \quad . \tag{11.8}$$

Given a number j of m-dimensional vectors, say, y_i, y_{i+1}, $\ldots$, y_{i+j-1}. Then the extended vector $y_{i|j}$ are defined as

$$y_{i|j} = \begin{bmatrix} y_i \\ y_{i+1} \\ \vdots \\ y_{i+j-1} \end{bmatrix} \in \mathbb{R}^{jm}. \tag{11.9}$$

11.3 The EMPC objective function

Consider the following LQ objective

$$\begin{aligned} J_k &= (y_{k+g-1|L+g} - r_{k+g-1|L+g})^T Q (y_{k+g-1|L+g} - r_{k+g-1|L+g}) \\ &+ \Delta u_{k|L+g}^T R \Delta u_{k|L+g} + (u_{k|L+g} - u_0)^T P (u_{k|L+g} - u_0), \end{aligned} \tag{11.10}$$

where the parameter g is either $g = 0$ or $g = 1$. The strictly proper case where $E = 0$ is parameterized with $g = 0$. The possibly non-zero nominal values for the inputs, u_0, will for the sake of simplicity be set to zero in the following discussion. Hence, we have the LQ objective on matrix form which was used in Di Ruscio (1997c)

$$J_{k,0} = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + \Delta u_{k|L}^T R \Delta u_{k|L} + u_{k|L}^T P u_{k|L} \tag{11.11}$$

The term y_k is not weighted in the objective because y_k is not influenced by u_k for strictly proper systems.

Consider now the only proper case where $E \neq 0$. This is parameterized with $g = 1$. Hence, we have the modified LQ objective

$$J_{k,1} = (y_{k|L+1} - r_{k|L+1})^T Q (y_{k|L+1} - r_{k|L+1}) + \Delta u_{k|L+1}^T R \Delta u_{k|L+1} + u_{k|L+1}^T P u_{k|L+1} \tag{11.12}$$

It is important to note that the prediction horizon has increased by one, i.e., from L to $L+1$ and that the output y_k is weighted in the objective since y_k is influenced by u_k for proper systems. Let us illustrate this with a simple example.

Example 11.1 *The standard control objective which is used in connection with models $x_{k+1} = Ax_k + Bu_k$ and $y_k = Dx_k$ can be illustrated with the following LQ objective with prediction horizon $L = 1$*

$$J_{k,0} = q(y_{k+1} - r_{k+1})^2 + pu_k^2. \tag{11.13}$$

Note that this objective is consistent with the model. I.e., the model states that u_k influences upon y_{k+1}, and both u_k and y_{k+1} are weighted in the objective.

Assume now that we instead have an output equation $y_k = Dx_k + Eu_k$. The above objective is in this case not general enough and usually not sufficient, because u_k will influence upon y_k and y_{k+1} is dependent of u_{k+1}. Both y_k and u_{k+1} are not weighted in the objective (11.13). A more general objective is therefore

$$J_{k,1} = q_{11}(y_k - r_k)^2 + q_{22}(y_{k+1} - r_{k+1})^2 + p_{11}u_k^2 + p_{22}u_{k+1}^2. \tag{11.14}$$

Hence, the prediction horizon in this case is $L + 1 = 2$.

11.4 EMPC with direct feed through term in SS model

We will in this section illustrate how the MPC problem for the strictly proper case, i.e., with a direct feed through term in the output equation, can be properly solved. We will focus on the EMPC method in Di Ruscio (1997b), (1997c) and Di Ruscio and Foss (1998). This is a variant of the traditional MPC in witch the present state is expressed in terms of past inputs and outputs as well as extended matrices witch results from the subspace identification algorithm, Di Ruscio (1996) and (1997d).

11.4.1 The LQ and QP objective

In the following we are putting $L =: L + 1$ for simplicity of notation. Hence, we consider the following LQ objective

$$J_k = (y_{k|L} - r_{k|L})^T Q (y_{k|L} - r_{k|L}) + \Delta u_{k|L}^T R \Delta u_{k|L} + u_{k|L}^T P u_{k|L}. \tag{11.15}$$

The prediction of $y_{k|L}$ is given by

$$y_{k|L} = O_L x_k + H_L^d u_{k|L}, \tag{11.16}$$

$$u_{k|L} = S \Delta u_{k|L} + c u_{k-1}, \tag{11.17}$$

which gives

$$y_{k|L} = F_L^{\Delta} \Delta u_{k|L} + p_L^{\Delta}, \tag{11.18}$$

$$p_L^{\Delta} = O_L x_k + H_L^d c u_{k-1}, \tag{11.19}$$

$$F_L^{\Delta} = H_L^d S. \tag{11.20}$$

x_k is taken as a state estimate based on J past inputs and outputs. J is defined as the past identification horizon The LQ objective can then be written in a form suited for Quadratic Programming (QP), i.e.,

$$J_k = \Delta u_{k|L}^T H \Delta u_{k|L} + 2 f_k^T \Delta u_{k|L}, \tag{11.21}$$

where

$$H = F_L^{\Delta T} Q F_L^{\Delta} + R + S^T P S, \tag{11.22}$$

$$f_k = F_L^{\Delta T} Q (p_L^{\Delta} - r_{k|L}) + S^T P c u_{k-1}. \tag{11.23}$$

11.4.2 The EMPC control

The EMPC control is given by the following QP problem:

$$\Delta u_{k|L}^* = \arg \min_{Z \Delta u_{k|L} \leq b_k} J_k = \arg \min_{Z \Delta u_{k|L} \leq b_k} (\Delta u_{k|L}^T H \Delta u_{k|L} + 2 f_k^T \Delta u_{k|L}). \tag{11.24}$$

The matrices in the inequality for the constraints, $Z \Delta u_{k|L} \leq b_k$, is presented in Di Ruscio (1997c). If the problem is unconstrained, then we have the explicit solution

$$\Delta u_{k|L}^* = -H^{-1} f_k \tag{11.25}$$

$$= (F_L^{\Delta T} Q F_L^{\Delta} + R + S^T P S)^{-1} (F_L^{\Delta T} Q (p_L^{\Delta} - r_{k|L}) + S^T P c u_{k-1}), \tag{11.26}$$

if the inverse exists.

11.4.3 Computing the state estimate when $g = 1$

The state can be expressed in terms of the known past inputs and outputs as follows

$$x_k = A^J O_J^{\dagger} y_{k-J|J} + (C_J - A^J O_J^{\dagger} H_J^d) u_{k-J|J}, \tag{11.27}$$

where $u_{k-J|J}$ and $y_{k-J|J}$ is defined from the known past inputs and outputs, respectively.

$$y_{k-J|J} = \begin{bmatrix} y_{k-J} \\ y_{k-J+1} \\ \vdots \\ y_{k-1} \end{bmatrix} \in \mathbb{R}^{Jm}, \quad u_{k-J|J} = \begin{bmatrix} u_{k-J} \\ u_{k-J+1} \\ \vdots \\ u_{k-1} \end{bmatrix} \in \mathbb{R}^{Jr}, \tag{11.28}$$

One should in this case note that y_{k-1} only is influenced by u_{k-1}. However, the most important remark is that y_k is not used in order to compute the state estimate. The reason for this is that y_k is influenced by the present input u_k which is not known. Note that u_k is to be computed by the control algorithm. If y_k is available before u_k, then, then it make sense to use an approach with $E = 0$.

11.4.4 Computing the state estimate when $g = 0$

In order to compare with the strictly proper case, the when $g = 0$, we present how the state is computed in this case:

$$x_k = A^{J-1} O_J^{\dagger} y_{k-J+1|J} + (C_{J-1} - A^{J-1} O_J^{\dagger} H_J^d) u_{k-J+1|J-1}, \tag{11.29}$$

where $u_{k-J+1|J-1}$ and $y_{k-J+1|J}$ is defined from the known past inputs and outputs, respectively.

$$y_{k-J+1|J} = \begin{bmatrix} y_{k-J+1} \\ y_{k-J+2} \\ \vdots \\ y_{k-1} \\ y_k \end{bmatrix} \in \mathbb{R}^{Jm}, \quad u_{k-J+1|J-1} = \begin{bmatrix} u_{k-J+1} \\ u_{k-J+2} \\ \vdots \\ u_{k-1} \end{bmatrix} \in \mathbb{R}^{(J-1)r} \tag{11.30}$$

One should in this case note that y_k only is influenced by u_{k-1}, and hence, y_k can be used for the state computation.

11.5 EMPC with final state weighting and the Riccati equation

Consider for simplicity the case where we do not have external signals. The LQ objective is given by

$$J_k = x_{k+L}^T S_{k+L} x_{k+L} + \sum_{i=0}^{L-1} (y_{k+i}^T Q_{k+i} y_{k+i} + u_{k+i}^T P_{k+i} u_{k+i}), \tag{11.31}$$

where S_{k+L} is a final state weighting matrix. The weighting of the final state is incorporated in order to obtain nominal stability of the controlled system, even for finite prediction horizons L. This will be discussed later.

11.5.1 EMPC with final state weighting

The objective can be written in matrix form as follows

$$J_k = x_{k+L}^T S x_{k+L} + y_{k|L}^T Q y_{k|L} + u_{k|L}^T P u_{k|L}, \tag{11.32}$$

where we for simplicity of notation have defined $S = S_{k+L}$.

Lemma 11.1 *The future optimal controls which minimizes the LQ objective (11.32) can be computed as*

$$u_{k|L}^* = -(P + H_L^{dT} Q H_L^d + C_L^{dT} S C_L^d)^{-1}(H_L^{dT} Q O_L + C_L^{dT} S A^L) x_k, \tag{11.33}$$

if the inverse exists.

Proof 11.1 *The future outputs, $y_{k|L}$, and the final state, x_{k+L}, can be expressed in terms of the unknown inputs, $u_{k|L}$, as:*

$$y_{k|L} = O_L x_k + H_L^d u_{k|L}, \tag{11.34}$$

$$x_{k+L} = A^L x_k + C_L^d u_{k|L}. \tag{11.35}$$

Substituting into the LQ objective (11.32) gives

$$\begin{aligned} J_k = {} & (A^L x_k + C_L^d u_{k|L})^T S (A^L x_k + C_L^d u_{k|L}) \\ & + (O_L x_k + H_L^d u_{k|L})^T Q (O_L x_k + H_L^d u_{k|L}) + u_{k|L}^T P u_{k|L}. \end{aligned} \tag{11.36}$$

The gradient with respect to $u_{k|L}$ is

$$\frac{\partial J_k}{\partial u_{k|L}} = 2 C_L^{dT} S (A^L x_k + C_L^d u_{k|L}) + 2 H_L^{dT} Q (O_L x_k + H_L^d u_{k|L}) + 2 P u_{k|L} \tag{11.37}$$

$$= 2(P + H_L^{dT} Q H_L^d + C_L^{dT} S C_L^d) u_{k|L} + 2 H_L^{dT} Q O_L x_k + C_L^{dT} S A^L x_k. \tag{11.38}$$

Solving $\frac{\partial J_k}{\partial u_{k|L}} = 0$ gives (11.33). □

It is worth to note that when the prediction horizon, L, is large (or infinity) and A is stable, then the final state weighting have small influence upon the optimal control since the term $A^L \approx 0$ in (11.33), in this case. This is also a well known property for the classical infinite horizon LQ solution which is independent of the final state weighting. The connection and equivalence between the explicit MPC solution (11.33) and the classical LQ solution is presented in the following section.

11.5.2 Connection to the Riccati equation

The optimal control can also be computed in the traditional way, i.e., by solving a Riccati equation. However, traditionally the solution to the LQ problem is in the classical literature, as far as we now, usually only presented for strictly proper systems. Here we will present the solution for only proper systems, i.e., with an output equation $y_k = D x_k + E u_k$ and the LQ objective in (11.32). Furthermore, we will present a new and non-iterative solution to the Riccati equation.

Lemma 11.2 *The LQ optimal control for the system* $x_{k+1} = Ax_k + Bu_k$ *and* $y_k = Dx_k + Eu_k$ *subject to the LQ objective (11.31) or equivalently (11.32), can be computed as*

$$u_k^* = -(P_k + B^T R_{k+1} B + E^T Q_k E)^{-1}(B^T R_{k+1} A + E^T Q_k D)x_k, \tag{11.39}$$

where R_{k+1} *is a solution to the Riccati equation*

$$\begin{aligned} R_k &= D^T Q_k D + A^T R_{k+1} A \\ &-(B^T R_{k+1} A + E^T Q_k D)^T (P_k + B^T R_{k+1} B + E^T Q_k E)^{-1}(B^T R_{k+1} A + E^T Q_k D), \end{aligned} \tag{11.40}$$

with final value condition

$$R_L = S_{k+L}. \tag{11.41}$$

Finally, the LQ objective (11.31) is equivalent with the following standard LQ objective

$$\begin{aligned} J_k &= x_{k+L}^T S_{k+L} x_{k+L} \\ &+ \textstyle\sum_{i=k}^{k+L-1}(x_i^T D^T Q_i D x_i + 2x_i^T D^T Q_i E u_i + u_i^T (P_i + E^T Q_i E)u_k), \end{aligned} \tag{11.42}$$

with cross-term weightings between x_k *and* u_k*.*

Proof 11.2 *See Appendix 11.9.* □

Note that the only proper case gives rise to cross-terms in the LQ objective. The solution to this is of course well known. The Riccati equation is iterated backwards from the final time, $k + L$, to obtain the solution R_{k+1} which is used in order to define the optimal control (11.39). Note that R_{k+1} is constant for unconstrained receding horizon control, provided the weighting matrices S_{k+L}, Q and P are time invariant weighting matrices.

We will now show that the solution to the Riccati equation can be expressed directly in terms of the extended observability matrix, O_L, and the Toepliz matrix, H_L^d, of impulse response matrices.

Lemma 11.3 *Consider for simplicity the two cases where the final state weighting is either* $S_{k+L} = 0$ *or* $S_{k+L} = S$*. An explicit and non-iterative solution to the Riccati equation is given as:*

Case with $S_{k+L} = 0$

$$\begin{aligned} R_k &= (O_L + H_L^d G O_L)^T Q (O_L + H_L^d G O_L) + (G O_L)^T P G O_L \\ &= O_L^T [(I_{Lm} + H_L^d G)^T Q (I_{Lm} + H_L^d G) + G^T P G] O_L, \end{aligned} \tag{11.43}$$

where

$$G = -(P + H_L^{dT} Q H_L^d)^{-1} H_L^{dT} Q. \tag{11.44}$$

Case with $S_{k+L} = S$

$$
\begin{aligned}
R_k = O_L^T (I_{Lm} + H_L^d G)^T Q (I_{Lm} + H_L^d G) + G^T P G] O_L \\
+ O_L^T [(A^L O_L^\dagger + C_L^d G)^T S (A^L O_L^\dagger + C_L^d G) O_L,
\end{aligned} \tag{11.45}
$$

where

$$
G = -(P + H_L^{dT} Q H_L^d + C_L^{dT} S C_L^d)^{-1} (H_L^{dT} Q + C_L^d S A^L O_L^\dagger), \tag{11.46}
$$

and $O_L^\dagger = (O_L O_L^T)^{-1} O_L^T$.

This result was proved in Di Ruscio (1997a). However, for completeness, the prof is presented below.

Proof 11.3 *Substituting the optimal control (11.33) into the objective (11.36) gives the minimum objective*

$$
J_k^* = x_k^T \overbrace{[(O_L + H_L^d G O_L)^T Q (O_L + H_L^d G O_L) + (G O_L)^T P G O_L]}^{R_k} x_k. \tag{11.47}
$$

From the classical theory of LQ optimal control we know that the minimum objective corresponding to the LQ solution in Lemma 11.2 is given by

$$
J_k^* = x_k^T R_k x_k. \tag{11.48}
$$

Comparing (11.47) and (11.48) proves the lemma. □

Proposition 11.1 *Since the EMPC control given by Lemma 11.1 and the LQ optimal control given by Lemma 11.2 gives the same minimum for the objective function, they are equivalent.*

Proof 11.4 *The prof follows from the proof of Lemma 11.39.* □

It is of interest to note that the solution to the discrete Riccati equation is defined in terms of the extended observability matrix O_L and the Toepliz matrix H_L^d. These matrices can be computed directly by the subspace identification algorithms, e.g., the DSR algorithm, provided the identification signal u_k is rich enough, i.e. persistently exciting of sufficiently high order. However, in many cases it may be better to redefine O_L and H_L^d from the system matrices D, A, B.

11.5.3 Final state weighting to ensure nominal stability

It is well known from classical LQ theory that,under some detectability and stabilizability conditions, the solution to the infinite time LQ problem results in a stable closed loop system.

Lemma 11.4 *Consider the problem of minimizing the LQ objective (11.31) and (11.32) subject to $x_{k+1} = Ax_k + Bu_k$ and $y_k = Dx_k + Eu_k$. The LQ optimal solution yields a stable nominal system if the final state weighting matrix is chosen as*

$$S_{k+L} = R, \tag{11.49}$$

where R is the positive definite solution to the discrete algebraic Riccati equation (11.40), where we have assumed that the pair $(A, \sqrt{D^T QD})$ is detectable and the pair (A, B) is stabilizable.

Proof 11.5 *The infinite objective*

$$J_k = \sum_{i=0}^{\infty}(y_{k+i}^T Q_{k+i} y_{k+i} + u_{k+i}^T P_{k+i} u_{k+i}). \tag{11.50}$$

can be splitted into two parts

$$J_k = J_L + \sum_{i=0}^{L-1}(y_{k+i}^T Q_{k+i} y_{k+i} + u_{k+i}^T P_{k+i} u_{k+i}). \tag{11.51}$$

where

$$J_L = \sum_{i=L}^{\infty}(y_{k+i}^T Q_{k+i} y_{k+i} + u_{k+i}^T P_{k+i} u_{k+i}). \tag{11.52}$$

Using the principle of optimality and that the minimum of the term J_L with infinite horizon is

$$J_L = x_{k+L}^T R x_{k+L}, \tag{11.53}$$

where R is a solution to the DARE. □

One point with all this is that we can re-formulate the infinite LQ objective into a finite programming problem with nominal stability for all choices of the prediction horizon, L.

$$J_k = u_{k|L}^T H u_{k|L} + 2f_k^T u_{k|L} \tag{11.54}$$

where

$$H = P + H_L^{dT} Q H_L^d + C_L^{dT} S C_L^d, \tag{11.55}$$

$$f_k = (H_L^{dT} Q O_L + C_L^{dT} S A^L) x_k, \tag{11.56}$$

$$S = R. \tag{11.57}$$

Hence, we have the QP problem

$$u_{k|L}^* = \arg \min_{Z u_{k|L} \leq b_k} J_k \tag{11.58}$$

With nominal stability we mean stability for the case where there are no modeling errors and that the inputs are not constrained.

Example 11.2 *Consider the LQ objective with the shortest possible prediction horizon, i.e., $L = 1$.*

$$J_k = x_{k+1}^T R x_{k+1} + y_k^T Q y_k + u_k^T P u_k, \tag{11.59}$$

where R is the positive solution to the Discrete time Algebraic Riccati Equation (DARE). The control input which minimizes the LQ objective is obtained from

$$\frac{\partial J_k}{\partial u_k} = 2B^T R(Ax_k + Bu_k) + 2E^T Q(Dx_k + Eu_k) + 2Pu_k = 0, \tag{11.60}$$

where we have used that $x_{k+1} = Ax_k + Bu_k$ and $y_k = Dx_k + Eu_k$ in (11.59). This gives

$$u_k^* = -(P + B^T RB + E^T QE)^{-1}(B^T RA + E^T QD)x_k, \tag{11.61}$$

which is exactly (11.39) with $R_{k+1} = R$. Hence, the closed loop system is stable since (11.61) is the solution to the infinite horizon LQ problem.

A promising and simple infinite horizon LQ controller (LQ regulator) with constraints is then given by the QP problem with inequality constraints

$$\begin{array}{ll} \min_{u_k} & J_k = x_{k+1}^T R x_{k+1} + y_k^T Q y_k + u_k^T P u_k, \\ \text{subject to } Zu_k \leq b. & \end{array} \tag{11.62}$$

Using the process model we write this in standard QP form as follows

Lemma 11.5 (Infinite horizon LQ controller with constraints)
Given a discrete time linear model with matrices (A, B, D, E), and weighting matrices Q and P for the LQ objective

$$J_k = \sum_{i=0}^{\infty} (y_{k+i}^T Q y_{k+i} + u_{k+i}^T P u_{k+i}). \tag{11.63}$$

A solution to the infinite horizon LQ problem with constraints is simply given by the QP problem (here) with inequality constraints

$$\begin{array}{ll} \min_{u_k} & J_k = u_k^T H u_k + 2 f_k^T u_k, \\ \text{subject to } Zu_k \leq b. & \end{array} \tag{11.64}$$

where

$$H = P + B^T RB + E^T QE, \tag{11.65}$$

$$f = (B^T RA + E^T QD)x_k, \tag{11.66}$$

and R is the solution to the discrete time algebraic (form of the) Riccati equation (11.40).

Proof 11.6 *Consider the case where the constraints are not active, i.e., consider first the unconstrained case. The solution is then*

$$u_k^* = -H^{-1} f, \tag{11.67}$$

which is identical to the LQ solution as presented in (11.61).

11.6 Linear models with non-zero offset

A linear model are often valid around some steady state values, x^s, for the states and u^s for the inputs. This is usually the case when the linear model is obtained by linearizing a non-linear model and when a linear model is identified from data which are adjusted for constant trends. Hence,

$$\begin{aligned} x_{k+1}^p - x^s &= A_p(x_k^p - x^s) + B_p(u_k - u^s), && (11.68)\\ y_k &= D_p x_k^p + y^0 \\ &= D_p(x_k^p - x^s) + y^s, && (11.69) \end{aligned}$$

where $y^0 = y^s - Dx^s$ and the initial state is x_0^p. If the model is obtained by linearizing a non-linear model, then we usually have that $y^0 = y^s - Dx^s = 0$. However, this is usually not the case when the model is obtained from identification and when the identification input and output data is adjusted for some constant trends u^s and y^s, e.g. the sample mean. In order to directly use this model, many of the algorithms which are presented in the literature, has to be modified in order to properly handle the offset values. Instead on should note that there exists an equivalent state space model of the form

Lemma 11.6 (Linear model without constant terms) *The linear model (11.68) and (11.69) is equivalent with*

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, && (11.70)\\ y_k &= Dx_k, && (11.71) \end{aligned}$$

where A, B and D are given by

$$A = \begin{bmatrix} A_p & 0_{n\times 1} \\ 0_{1\times n} & 1 \end{bmatrix}, \; B = \begin{bmatrix} B_p \\ 0_{1\times r} \end{bmatrix}, \; D = \begin{bmatrix} D_p & D_z \end{bmatrix}, \qquad (11.72)$$

and where the vector D_z and the initial state, x_0, are given by

$$D_z = D_p(x^s - (I_n - A_p)^{-1}B_p u^s) + y^0 = -D_p(I_n - A_p)^{-1}B_p u^s + y^s, \qquad (11.73)$$

$$x_0 = \begin{bmatrix} x_0^p - x^s + (I_n - A_p)^{-1}B_p u^s \\ 1 \end{bmatrix}. \qquad (11.74)$$

Proof 11.7 *The model (11.68) and (11.69) can be separated into*

$$\begin{aligned} x_{k+1}^p &= A_p x_k^p + B_p u_k, && (11.75)\\ x^s &= Ax^s + Bu^s, && (11.76)\\ y_k &= D_p x_k^p - D_p x^s + y^s. && (11.77) \end{aligned}$$

The states in (11.76) are time-invariant. Hence, we can solve for x^s, i.e.,

$$x^s = (I_n - A_p)^{-1}B_p u^s. \qquad (11.78)$$

Substituting (11.78) into (11.77) gives

$$\begin{aligned} y_k &= D_p x_k^p + y^0, && (11.79)\\ y^0 &= -D_p x^s + y^s. && (11.80) \end{aligned}$$

Using (11.75), (11.79) and an integrator $z_{k+1} = z_k$ in order to handle the off-set y^0 in (11.79), gives

$$\begin{bmatrix} x_{k+1}^p \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} A_p & 0_{n\times 1} \\ 0_{1\times n} & 1 \end{bmatrix} \begin{bmatrix} x_k^p \\ z_k \end{bmatrix} + \begin{bmatrix} B_p \\ 0_{1\times r} \end{bmatrix} u_k, \tag{11.81}$$

$$y_k = \begin{bmatrix} D_p & y^0 \end{bmatrix} \begin{bmatrix} x_k^p \\ z_k \end{bmatrix}. \tag{11.82}$$

The initial states can be found by comparing (11.69) with (11.82) at time $k = 0$, i.e.,

$$D_p x_0^p + y^0 z_0 = D(x_0^p - x^s) + y^s. \tag{11.83}$$

Choosing $z_0 = 1$ gives

$$x_0^p = (x_0^p - x^s) + x^s. \tag{11.84}$$

□

The number of states in the model (11.70) and (11.71) has increased by one compared to the number of states in the model (11.68) and (11.69), in order to handle the non-zero offset. Moreover, the matrix A will have a unit eigenvalue representing an uncontrollable integrator, which is used to handle the offset. Lemma 11.7 is important because it implies that theory which is based on a linear state space models without non-zero mean constant values still can be used. However, the theory must be able to handle the integrator. This is not always the case.

In practice we could very well work with non-zero mean values on the output. Hence, we have the following lemma

Lemma 11.7 (Linear model without constant state and input terms) *The linear model (11.68) and (11.69) is equivalent with*

$$\begin{aligned} x_{k+1}^p &= A_p x_k^p + B_p u_k, && (11.85)\\ y_k &= D_p x_k^p + y^0, && (11.86) \end{aligned}$$

where the initial state is

$$\begin{aligned} x_0^p &= (x_0^p - x^s) + \tilde{x}^s, && (11.87)\\ \tilde{x}^s &= (I - A_p)^{-1} B_p u^s, && (11.88) \end{aligned}$$

end where the off-set y^0 is given by

$$y^0 = y^s - D_p \tilde{x}^s. \tag{11.89}$$

Proof 11.8 *The proof follows from the proof of Lemma 11.6, and in particular Equations (11.75), (11.79), (11.80), (11.78) and (11.84).*

Hence, one only need to take properly care of non-zero mean constant values on the output.

Another simple method for constructing a model without offset variables is to first simulate the model (11.68) and (11.69) in order to generate identification data and then to use a subspace identification algorithm, e.g. the DSR-method. The correct order for the state in the equivalent model is then identified by the subspace algorithm.

11.7 Conclusion

The problem of MPC of systems which is only proper, i.e., systems with a direct feed through term in the output equation, is addressed and a direct matrix based solution is proposed. The solution can be expressed in terms of some extended matrices from subspace identification. These matrices may be identified directly or formed from any linear model.

A final state weighting is incorporated in the objective in order to ensure stability of the nominal and unconstrained closed loop system.

The equivalence between the solution to the classical discrete time LQ optimal control problem and the solution to the unconstrained MPC problem is proved.

Furthermore, a new explicit and non-iterative solution to the discrete Riccati equation is presented. This solution is a function of two matrices which can be computed directly from the DSR subspace identification method.

11.8 references

Di Ruscio, D. and B. Foss (1998). On Model Based Predictive Control. *The 5th IFAC Symposium on Dynamics and Control of Process Systems*, Corfu, Greece, June 8-10,1998.

Di Ruscio, D. (1997a). Optimal model based control: System analysis and design. Lecture notes. Report, HiT, TF.

Di Ruscio, D. (1997b). Model Based Predictive Control: An extended state space approach. In proceedings of *the 36th Conference on Decision and Control 1997*, San Diego, California, December 6-14.

Di Ruscio, D. (1997c). Model Predictive Control and Identification: A linear state space model approach. In proceedings of *the 36th Conference on Decision and Control 1997*, San Diego, California, December 6-14.

Di Ruscio, D. (1997d). A Method for Identification of Combined Deterministic Stochastic Systems. In *Applications of Computer Aided Time Series Modeling.*

Editor: Aoki. M. and A. Hevenner, Lecture Notes in Statistics Series, Springer Verlag, 1997, pp. 181-235.

Di Ruscio, D. (1996). Combined Deterministic and Stochastic System Identification and Realization: DSR-a subspace approach based on observations. *Modeling, Identification and Control*, vol. 17, no.3.

11.9 Appendix: Proof of Lemma 11.2

We are going to use the Maximum principle for the proof. The Hamilton function is

$$H_k = \frac{1}{2}((Dx_k + Eu_k)^T Q_k (Dx_k + Eu_k) + u_k^T P_k u_k) + p_{k+1}^T((A-I)x_k + Bu_k)), \tag{11.90}$$

where p_k is the co-state vector.

11.9.1 The LQ optimal control

An expression for the optimal control is found from the gradient

$$\frac{\partial H_k}{\partial u_k} = E^T Q_k (Dx_k + Eu_k) + P_k u_k + B^T p_{k+1} = 0, \tag{11.91}$$

which gives

$$(P_k + E^T Q_k E)u_k = -B^T p_{k+1} - E^T Q_k D x_k. \tag{11.92}$$

It can be shown from the state and co-state equations that

$$p_k = R_k x_k. \tag{11.93}$$

See Section 11.9.3 for a proof. Using this in (11.92) gives

$$(P_k + E^T Q_k E)u_k = -B^T R_{k+1}(Ax_k + Bu_k) - E^T Q_k D x_k. \tag{11.94}$$

Hence, we have the following expression for the optimal control

$$u_k = (P_k + E^T Q_k E + B^T R_{k+1} B)^{-1}(B^T R_{k+1} A + E^T Q_k D)x_k, \tag{11.95}$$

which is the same as in Lemma 11.2. □

11.9.2 The Riccati equation

An expression for the co-state vector are determined from

$$p_{k+1} - p_k = -\frac{\partial H_k}{\partial x_k} = -(D^T Q_k (Dx_k + Eu_k) + (A-I)^T p_{k+1}), \tag{11.96}$$

which gives

$$p_k = D^T Q_k D x_k + D^T Q_k E u_k + A^T p_{k+1}, \tag{11.97}$$

Using (11.93) and the state equation gives

$$R_k x_k = D^T Q_k D x_k + D^T Q_k E u_k + A^T R_{k+1}(A x_k + B u_k). \tag{11.98}$$

This can be simplified to

$$R_k x_k = D^T Q_k D x_k + (B^T R_{k+1} A + E^T Q_k D)^T u_k + A^T R_{k+1} A x_k. \tag{11.99}$$

Substituting the optimal control (11.95) into (11.99)gives

$$\begin{gathered} R_k x_k = D^T Q_k D x_k + A^T R_{k+1} A x_k \\ (B^T R_{k+1} A + E^T Q_k D)^T (P_k + E^T Q_k E + B^T R_{k+1} B)^{-1} (B^T R_{k+1} A + E^T Q_k D) x_k \end{gathered} \tag{11.100}$$

This equation must hold for all $x_k \neq 0$, and the Riccati Equation (11.40) follows. □

11.9.3 The state and co-state equations

Substituting the optimal control given by (11.92), which we have written as $u_k = G_1 x_k + G_2 p_{k+1}$, into the co-state equation (11.97) and the state equation gives the state and co-state system

$$\begin{aligned} x_{k+1} &= (A + B G_1) x_k + B G_2 p_{k+1}, && (11.101) \\ p_k &= (D^T Q_k D + D^T Q_k E G_1) x_k + (A^T p + D^T Q_k E G_2) p_{k+1}. && (11.102) \end{aligned}$$

Starting with the final condition for the co-state, i.e., $p_{k+L} = S_{k+L} x_{k+L}$, and using induction shows the linear relationship

$$p_k = R_k x_k, \tag{11.103}$$

between the co-state and the state. □

11.10 Appendix: On the EMPC objective with final state weighting

In the general case we have to incorporate external signals into the problem. For the sake of completeness, this is discussed in this section

$$\begin{aligned} J_k = (x_{k+L} - x_r)^T S_{k+L} (x_{k+L} - x_r) + (y_{k|L} - r_{k|L})^T Q (y_{k|L} - r_{k|L}) \\ + \Delta u_{k|L}^T R \Delta u_{k|L} + (u_{k|L} - u_0)^T P (u_{k|L} - u_0), \end{aligned} \tag{11.104}$$

where x_r is a reference for the final state weighting and u_0 is a target vector for the future control inputs

Part II

Optimization and system identification

Chapter 12

Prediction error methods

12.1 Overview of the prediction error methods

Given the state space model in innovations form.

$$\bar{x}_{k+1} = A\bar{x}_k + Bu_k + Ke_k, \tag{12.1}$$

$$y_k = \underbrace{D\bar{x}_k + Eu_k}_{\bar{y}_k} + e_k, \tag{12.2}$$

where $\Delta = \mathrm{E}(e_k e_k^T)$ is the covariance matrix of the innovations process and $\bar{x}_1$ is the initial predicted state. Suppose now that the model is parameterized, i.e. so that the free parameters in the model matrices (A, B, K, D, E, x_1) are organized into a parameter vector θ. The problem is to identify the "best" parameter vector from known output and input data matrices (Y, U). The optimal predictor, i.e. the optimal prediction, $\bar{y}_k$, for the output y_k, is then of the form

$$\bar{x}_{k+1} = (A - KD)\bar{x}_k + (B - KE)u_k + Ky_k, \tag{12.3}$$

$$\bar{y}_k(\theta) = D\bar{x}_k + Eu_k, \tag{12.4}$$

with initial predicted state $\bar{x}_1$. $\bar{y}_k(\theta)$ is the prediction of the output y_k given inputs u up to time k, outputs y up to time $k-1$ and the parameter vector θ. Note that if $E = 0_{m\times r}$ then inputs u only up to only $k-1$ are needed. The free parameters in the system matrices are mapped into the parameter vector (or visa versa). Note that the predictor $\bar{y}_k(\theta)$ is (only) optimal for the parameter vector θ which minimize some specified criterion. This criterion is usually a function of the *prediction errors*. Note also that it is common to use the notation $\bar{y}_{k|\theta}$ for the prediction. Hence, $\bar{y}_{k|\theta} = \bar{y}_k(\theta)$.

Define the Prediction Error (PE)

$$\epsilon_k(\theta) = y_k - \bar{y}_k(\theta). \tag{12.5}$$

A good model is a model for which the model parameters θ results in a "small" PE. Hence, it make sense to use a PE criterion which measure the size of the PE. A PE criterion is usually always in one ore another way defined as a scalar function of the following important expression and definition

$$R_\epsilon(\theta) = \frac{1}{N}\sum_{k=1}^{N}\epsilon_k(\theta)\epsilon_k(\theta)^T \ \in \ \mathbb{R}^{m\times m}, \tag{12.6}$$

which is the sample covariance matrix of the PE. This means that we want to find the parameter vector which make R_ϵ as small as possible. Hence, it make sense to use a PE criterion which measure the size of the sample covariance matrix of the PE. A common scalar PE criterion for multivariable output systems is thus

$$V_N(\theta) = \mathrm{tr}(\frac{1}{N}\sum_{k=1}^{N}\epsilon_k(\theta)\epsilon_k(\theta)^T) = \frac{1}{N}\sum_{k=1}^{N}\epsilon_k(\theta)^T\epsilon_k(\theta). \tag{12.7}$$

Note that the trace of a matrix is equal to the sum of its diagonal elements and that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ of two matrices A and B of appropriate dimensions. We will in the following give a discussion of the PE criterion as well as some variants of it. Define a PE criterion as follows

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N}\ell(\epsilon_k(\theta)) \tag{12.8}$$

where $\ell(\cdot)$ is a scalar valued function, e.g. the Euclidean l_2 norm, i.e.

$$\ell(\epsilon_k(\theta)) = \parallel \epsilon_k(\theta) \parallel_2^2 = \epsilon_k(\theta)^T\epsilon_k(\theta), \tag{12.9}$$

or a quadratic function

$$\ell(\epsilon_k(\theta)) = \epsilon_k(\theta)^T\Lambda\epsilon_k(\theta) = \mathrm{tr}(\Lambda\epsilon_k(\theta)\epsilon_k(\theta)^T), \tag{12.10}$$

for some weighting matrix Λ. A common criterion for multiple output systems (with weights) is thus also

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N}\epsilon_k(\theta)^T\Lambda\epsilon_k(\theta) = \mathrm{tr}(\Lambda(\frac{1}{N}\sum_{k=1}^{N}\epsilon_k(\theta)\epsilon_k(\theta)^T)), \tag{12.11}$$

where we usually simply are using $\Lambda = I$. The weighting matrix Λ is usually a diagonal and positive matrix. One should note that there exist an optimal weighting matrix Λ, but that this matrix is difficult to define a-priori. The optimal weighting (when the number of observations N is large) is defined from the knowledge of the innovations noise covariance matrix of the system, i.e., $\Lambda = \Delta^{-1}$ where $\Delta = \mathrm{E}(e_k e_k^T)$.

The sample covariance matrix of the PE is positive semi-definite, i.e. $R_\epsilon \geq 0$. The PE may be zero for deterministic systems however for combined deterministic and stochastic systems we usually have that $R_\epsilon > 0$, i.e. positive definite. This means off-course in any case that R_ϵ is a symmetric matrix. The eigenvalues of a symmetric matrix are all real. Define $\lambda_1, \ldots, \lambda_m$ as the eigenvalues of $R_\epsilon(\theta)$ for use in the following discussion.

Hence, a good parameter vector, θ, is such that the sample covariance matrix R_ϵ is small. The trace operator in (12.11), i.e. the PE criterion

$$V_N(\theta) = \mathrm{tr}(R_\epsilon(\theta)) = \lambda_1 + \ldots + \lambda_m, \tag{12.12}$$

is a measure of the size of the matrix $R_\epsilon(\theta)$. Hence, the trace of a matrix is equal to the sum of the diagonal elements of the matrix, but the trace is also equal to the sum of the eigenvalues of a symmetric matrix.

An alternative PE criterion which often is used is the determinant, i.e.,

$$V_N(\theta) = \det(R_\epsilon(\theta)) = \lambda_1\lambda_2\dots\lambda_m. \tag{12.13}$$

Hence, the determinant of the matrix is equal to the product of its eigenvalues. This lead us to a third alternative, which is to use the maximum eigenvalue of $R_\epsilon(\theta)$ as a measure of its size, i.e., we may use the following PE criterion

$$V_N(\theta) = \lambda_{\max}(R_\epsilon(\theta)), \tag{12.14}$$

where $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of a symmetric matrix.

Note the special case for a single output system, then we have

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N}\epsilon_k(\theta)^2 = \frac{1}{N}\sum_{k=1}^{N}(y_k - \bar{y}_k(\theta))^2. \tag{12.15}$$

The minimizing parameter vector is defined by

$$\hat{\theta}_N = \arg\min_{\theta\in D_{\mathcal{M}}} V_N(\theta) \tag{12.16}$$

where arg min denotes the operator which returns the argument that minimizes the function. The subscript N is often omitted, hence $\hat{\theta} = \hat{\theta}_N$ and $V(\theta) = V_N(\theta)$. The definition (12.16) is a standard optimization problem. A simple solution is then to use a software optimization algorithm which is dependent only on function evaluations, i.e., where the user only have to define the PE criterion $V(\theta)$.

We will in the following give a discussion of how this may be done. The minimizing parameter vector $\hat{\theta} = \hat{\theta}_N \in \mathbb{R}^p$ has to be searched for in the parameter space $D_{\mathcal{M}}$ by some iterative non-linear optimization method. Optimization methods are usually constructed as variations of the Gauss-Newton method and the Newton-Raphson method, i.e.,

$$\theta_{i+1} = \theta_i - \alpha H_i^{-1}(\theta_i)g_i(\theta_i) \tag{12.17}$$

where α is defined as a line search (ore step length) scalar parameter chosen to ensure convergence (i.e. chosen to ensure that $V(\theta_{i+1} < V(\theta_i))$, and where the gradient, $g_i(\theta_i)$, is

$$g_i(\theta_i) = \frac{dV(\theta_i)}{d\theta_i} \ \in \ \mathbb{R}^p, \tag{12.18}$$

and

$$H_i(\theta_i) = \frac{dg_i(\theta_i)}{d\theta_i^T} = \frac{d}{d\theta_i^T}(\frac{dV(\theta_i)}{d\theta_i}) = \frac{d^2V(\theta_i)}{d\theta_i^T d\theta_i} \ \in \ \mathbb{R}^{p\times p}, \tag{12.19}$$

is the Hessian matrix. Remark that the Hessian is a symmetric matrix and that it is positive definite in the minimum, i.e.,

$$H(\hat{\theta}) = \frac{d^2V(\hat{\theta})}{d\hat{\theta}^T d\hat{\theta}} > 0, \tag{12.20}$$

where $\hat{\theta} = \hat{\theta}_N$ is the minimizing parameter vector. Note that the iteration scheme (12.17) is identical to the Newton-Raphson method when $\alpha = 1$. In practice one often have to use a variable step length parameter α, both in order to stabilize the algorithm and to improve the rate of convergence far from the minimum. Once the gradient, $g_i(\theta_i)$ and the Hessian matrix $H_i(\theta_i)$ (or an approximation of the Hessian) have been computed, we can chose the line search parameter, α, as

$$\alpha = \arg\min_{\alpha} V_N(\theta_{i+1}(\alpha)) = \arg\min_{\alpha}(\theta_i - \alpha H_i^{-1}(\theta_i) g_i(\theta_i)). \tag{12.21}$$

Equation (12.21) is an optimization problem for the line search parameter α. Once α have been determined from the scalar optimization problem (12.21), the new parameter vector θ_{i+1} is determined from (12.17).

The iteration process (12.17) must be initialized with an initial parameter vector, θ_1. This was earlier (before the subspace methods) a problem. However, a good solution is to use the parameters from a subspace identification method. Equation (12.17) can then be implemented in a **while** or **for** loop. That is to iterate Equation (12.17) for $i = 1, \ldots,$ until convergence, i.e., until the gradient is sufficiently zero, i.e., until $g_i(\theta_i) \approx 0$ for some $i \geq 1$. This, and only such a parameter vector is our estimate, i.e. $\hat{\theta} = \hat{\theta}_N = \theta_i$ when $g(\theta_i) \approx 0$.

The iteration equation (12.17) can be deduced from the fact that in the minimum we have that the gradient, g, is zero, i.e.,

$$g(\hat{\theta}) = \frac{dV(\hat{\theta})}{d\hat{\theta}} = 0. \tag{12.22}$$

We can now use the Newton-Raphson method, which can be deduced as follows. An expression of $g(\hat{\theta})$ can be defined from a Taylor series expansion of $g(\theta)$ around θ, i.e.,

$$0 = g(\hat{\theta}) \approx g(\theta) + \left.\tfrac{dg(\theta)}{d\theta^T}\right|_{\theta} (\hat{\theta} - \theta). \tag{12.23}$$

Using this and that $g(\hat{\theta}) = 0$ gives

$$\hat{\theta} = \theta - (\left.\tfrac{dg(\theta)}{d\theta^T}\right|_{\theta})^{-1} g(\theta). \tag{12.24}$$

This equation is the background for the iteration scheme (12.17), i.e., putting $\theta := \theta_i$ and $\hat{\theta} := \theta_{i+1}$. Hence,

$$\theta_{i+1} = \theta_i - (\left.\tfrac{dg(\theta)}{d\theta^T}\right|_{\theta_i})^{-1} g(\theta_i). \tag{12.25}$$

Note that we in (12.17) has used the shorthand notation

$$\frac{dg(\theta_i)}{d\theta_i^T} = \left.\tfrac{dg(\theta)}{d\theta^T}\right|_{\theta_i}. \tag{12.26}$$

Note that the parameter vector, θ, the gradient, g, and the Hessian matrix have structures as follows

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix} \in \mathbb{R}^p, \tag{12.27}$$

$$g(\theta) = \frac{dV(\theta)}{d\theta} = \begin{bmatrix} g_1 \\ \vdots \\ g_p \end{bmatrix} = \begin{bmatrix} \frac{dV}{d\theta_1} \\ \vdots \\ \frac{dV}{d\theta_p} \end{bmatrix} \in \mathbb{R}^p, \tag{12.28}$$

$$H = \frac{dg(\theta)}{d\theta^T} = \frac{d^2V(\theta)}{d\theta^T d\theta} = \begin{bmatrix} \frac{dg_1}{d\theta_1} & \cdots & \frac{dg_1}{d\theta_p} \\ \vdots & \ddots & \vdots \\ \frac{dg_p}{d\theta_1} & \cdots & \frac{dg_p}{d\theta_p} \end{bmatrix} = \begin{bmatrix} \frac{d^2V}{d\theta_1^2} & \cdots & \frac{d^2V}{d\theta_p d\theta_1} \\ \vdots & \ddots & \vdots \\ \frac{d^2V}{d\theta_1 d\theta_p} & \cdots & \frac{d^2V}{d\theta_p^2} \end{bmatrix} \in \mathbb{R}^{p\times p} \tag{12.29}$$

The gradient and the Hessian can be computed numerically. However, it is usually more efficient by an analytically computation if possible. Remark that a common notation of the Hessian matrix is

$$H = \frac{d^2V(\theta)}{d\theta^2}, \tag{12.30}$$

and that the elements in the Hessian is given by

$$h_{ij} = \frac{\partial^2 V(\theta)}{\partial\theta_i \partial\theta_j}, \tag{12.31}$$

where θ_i and θ_j are parameter number i and j, respectively, in the parameter vector θ.

The iteration process (12.17) is guaranteed to converge to a local minimum, at least theoretically. There may exist many local minima. However, our experience is that the parameters from an estimated model from a subspace method is very close to the minimum. Hence, this initial choice for θ_1 should always be considered first. For systems with many outputs, many inputs and many states there may be a huge number of parameters. The optimization problem may in some circumstances be so complicated that the process (12.17) diverges even when the initial parameter θ_1 is close to minimum, due to numerical problems. Another problem with PEM is the model parameterization (canonical form) for systems with many outputs. One need to specify a canonical form of the state space model, i.e. a state space realization where there are as few free parameters as possibile in the model matrices (A, B, D, E, K, x_1). A problem for multiple output systems is that there may not even exist such a canonical form. Another problem is that the PE criterion, $V_N(\theta)$, may be almost in-sensitive to perturbations in the parameter vector, θ, for the specified canonical form. Hence, the optimization problem may be ill-conditioned.

An advantage of the PE methods is that it does not matter if the data (Y, U) is collected from closed loop or open loop process operation. It is however necessary that for open loop experiments, that the inputs, u_k, are rich enough for the specified model structure (usually the model order, n). For closed loop experiments we must typically require that the feedback is not to simple, e.g. not a simple proportional controller $u_k = K_p y_k$. However, feedback of the type $u_k = K_p(r_k - y_k)$ where the reference, r_k, is perturbed gives data which are informative enough.

12.1.1 Further remarks on the PEM

Note also that in the multivariable case we have that the parameter estimate

$$\hat{\theta}_N = \arg\min_{\theta} \operatorname{tr}(\Lambda R_\epsilon(\theta)), \tag{12.32}$$

with $\Lambda = \Delta^{-1}$ where $\Delta = (\mathrm{E}(e_k e_k^T))$, has the same asymptotic covariance matrix as the parameter estimate

$$\hat{\theta}_N = \arg\min_{\theta} \det(R_\epsilon(\theta)). \tag{12.33}$$

It can also be shown that the PEM parameter estimate for Gaussian distributed disturbances, e_k, (12.33), or equivalently the PEM estimate (12.32) with the optimal weighting, is identical to the Maximum Likelihood (ML) parameter estimate. The PEM estimates (12.32) or (12.33) are both statistically optimal. The only drawback by using (12.33), i.e., minimizing the determinant PE criterion $V_N(\theta) = \det(R_\epsilon(\theta))$, is that it requires more numerical calculations than the trace criterion. However, on the other side the evaluation of the PE criterion (12.32) with the optimal weighting matrix $\Lambda = \Delta^{-1}$ is not (directly) realistic since the exact covariance matrix Δ is not known in advance. However, note that an estimate of Δ can be built up during the iteration (optimization) process.

The parameter estimate $\hat{\theta}_N$ is a random vector, i.e., Gaussian distributed when e_k is Gaussian. This means that $\hat{\theta}_N$ has a mean and a variance. We want the mean to be as close to the true parameter vector as possibile and the variance to be as small as possible. We can show that the PEM estimate $\hat{\theta}_N$ is consistent, i.e., the mean of the parameter estimate, $\hat{\theta}_N$, converges to the true parameter vector, θ_0, as N tends to infinity. In other words we have that $\mathrm{E}(\hat{\theta}_N) = \theta_0$. Consider $g(\hat{\theta}_N) = 0$ expressed as a Taylor series expansion of $g(\theta_0)$ around the true parameter vector θ_0, i.e.

$$0 = g(\hat{\theta}_N) \approx g(\theta_0) + H(\theta_0)(\hat{\theta}_N - \theta_0). \tag{12.34}$$

where the Hessian matrix

$$H(\theta_0) = \left. \tfrac{dg(\theta)}{d\theta} \right|_{\theta_0}, \tag{12.35}$$

is a deterministic (constant) matrix. However, the gradient, $g(\theta_0)$, is a random vector with zero mean and covariance matrix P_0. From this we have that the difference between our estimate, $\hat{\theta}_N$ and the true vector θ_0 is given by

$$\hat{\theta}_N - \theta_0 = -H^{-1}(\theta_0) g(\theta_0). \tag{12.36}$$

Hence,

$$\mathrm{E}(\hat{\theta}_N - \theta_0) = -H^{-1}(\theta_0)\mathrm{E}(g(\theta_0)) = 0. \tag{12.37}$$

This shows consistency of the parameter estimate, i.e.

$$\mathrm{E}(\hat{\theta}_N) = \theta_0, \tag{12.38}$$

because θ_0 is deterministic.

The parameter estimates (12.33) and (12.32) with optimal weighting are efficient, i.e., they ensures that the parameter covariance matrix

$$P = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = H^{-1}(\theta_0)\mathrm{E}(g(\theta_0)g(\theta_0)^T)H^{-1}(\theta_0) \tag{12.39}$$

is minimized. The covariance matrix P may be estimated from the data by evaluating (12.39) numerically by using the approximation $\theta_0 \approx \hat{\theta}_N$.

For single output systems ($m = 1$) we have that the parameter covariance matrix, P, can be expressed as

$$P = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = \Delta(\mathrm{E}(\psi_k(\theta_0)\psi_k^T(\theta_0)))^{-1}, \tag{12.40}$$

where $\Delta = \mathrm{E}(e_k e_k^T) = \mathrm{E}(e_k^2)$ in the single output case, and with

$$\psi_k(\theta_0) = \frac{d\bar{y}_k(\theta)}{d\theta} |_{\theta_0} \in \mathbb{R}^{p\times m} \tag{12.41}$$

Loosely spoken, Equation (12.40) states that the variance, P, of the parameter estimate, $\hat{\theta}_N$, is "small" if the covariance matrix of $\psi_k(\theta_0)$ is large. This covariance matrix is large if the predictor $\bar{y}_k(\theta)$ is "very" sensitive to (perturbations in) the parameter vector θ.

12.1.2 Derivatives of the prediction error criterion

Consider the PE criterion (12.8) with (12.10), i.e.,

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N} \ell(\epsilon_k(\theta)) = \frac{1}{N}\sum_{k=1}^{N} \overbrace{\epsilon_k^T(\theta)\Lambda\epsilon_k(\theta)}^{\ell(\epsilon_k(\theta))}. \tag{12.42}$$

The derivative of the scalar valued function $\ell = \ell(\epsilon_k(\theta))$ with respect to the parameter vector θ can be expressed from the kernel rule

$$\frac{\partial \ell(\epsilon_k(\theta))}{\partial\theta} = \frac{\partial\epsilon_k}{\partial\theta}\frac{\partial\ell}{\partial\epsilon_k} = -\frac{\partial\bar{y}_k(\theta)}{\partial\theta}2\Lambda\epsilon_k. \tag{12.43}$$

Define the gradient matrix of the predictor $\bar{y}_k \in \mathbb{R}^{\times m}$ with respect to the parameter vector $\theta \in \mathbb{R}^{\times p}$ as

$$\psi_k(\theta) = \frac{\partial\bar{y}_k(\theta)}{\partial\theta} \in \mathbb{R}^{p\times m}. \tag{12.44}$$

This gives the following general expression for the gradient, i.e.,

$$g(\theta) = \frac{\partial V_N(\theta)}{\partial\theta} = \frac{1}{N}\sum_{k=1}^{N}\frac{\partial\ell(\epsilon_k(\theta))}{\partial\theta} = -2\frac{1}{N}\sum_{k=1}^{N}\psi_k(\theta)\Lambda\epsilon_k \tag{12.45}$$

12.1.3 Least Squares and the prediction error method

The optimal predictor, $\bar{y}_k(\theta)$, is generally a non-linear function of of the unknown parameter vector, θ. The PEM estimate can in this case in generally not be solved analytically. However, in some simple and special cases we have that the predictor is a linear function of the parameter vector. This problem have an analytical solution and the corresponding PEM is known as the Least Squares (LS) method. we will in this section give a short description of the solution to this problem.

Linear regression models

Consider the simple special case of the general linear system (12.1) and (12.2) described by

$$y_k = Eu_k + e_k, \tag{12.46}$$

where $y_k \in \mathbb{R}^m$, $E \in \mathbb{R}^{m\times r}$, $u_k \in \mathbb{R}^r$ and $e_k \in \mathbb{R}^m$ is white with covariance matrix $\Delta = \mathrm{E}(e_k e_k^T)$. The model (12.46) can be written as a linear regression

$$y_k = \varphi_k^T \theta + e_k, \tag{12.47}$$

where

$$\varphi_k^T = u_k^T \otimes I_m \ \in \ \mathbb{R}^{m\times rm} \tag{12.48}$$

and the true parameters in the system, θ_0, is related to those in E as

$$\theta = \mathrm{vec}(E) \ \in \ \mathbb{R}^{mr}. \tag{12.49}$$

Note that the number of parameters in this case is $p = mr$. Furthermore, note that (12.47) is a standard notation of a linear regression equation used in the identification literature. In order to deduce (12.47) from (12.46) we have used that $\mathrm{vec}(AXB) = (B^T \otimes A)\mathrm{vec}(X)$. Using this and the fact that $Eu_k = I_m E u_k$ gives (12.47).

Also note that dynamic systems described by ARX models can be written as a linear regression of the form (12.47).

The least squares method

Given a linear regression of the form

$$y_k = \varphi_k^T \theta_0 + e_k, \tag{12.50}$$

where $y_k \in \mathbb{R}^m$, $\varphi_k \in \mathbb{R}^{p\times m}$, $e_k \in \mathbb{R}^m$ is white with covariance matrix $\Delta = \mathrm{E}(e_k e_k^T)$ $\in \mathbb{R}^{m\times m}$ and where $\theta_0 \in \mathbb{R}^p$ is the true parameter vector. A natural predictor is as usual

$$\bar{y}_k(\theta) = \varphi_k^T \theta. \tag{12.51}$$

We will in the following find the parameter estimate, $\hat{\theta}_N$, which minimizes the PE criterion

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N} \epsilon_k^T(\theta)\Lambda\epsilon_k(\theta). \tag{12.52}$$

The gradient matrix, $\psi_k(\theta)$, defined in (12.44) is in this case given by

$$\psi_k(\theta) = \frac{\partial \bar{y}_k(\theta)}{\partial \theta} = \varphi_k \ \in \ \mathbb{R}^{p\times m}. \tag{12.53}$$

Using this and the expression (12.45) for the gradient, $g(\theta)$, of the PE criterion, $V_N(\theta)$, gives

$$g(\theta) = \frac{\partial V_N(\theta)}{\partial \theta} = -2\frac{1}{N}\sum_{k=1}^{N} \varphi_k \Lambda (y_k - \varphi_k^T \theta) = -2\frac{1}{N}\sum_{k=1}^{N} (\varphi_k \Lambda y_k - \varphi_k \Lambda \varphi_k^T \theta) \quad (12.54)$$

The OLS and the PEM estimate is here simply given by solving $g(\theta) = 0$, i.e.,

$$\hat{\theta}_N = (\sum_{k=1}^{N} \varphi_k \Lambda \varphi_k^T)^{-1} \sum_{k=1}^{N} \varphi_k \Lambda y_k. \quad (12.55)$$

if the indicated inverse (of the Hessian) exists. Note that the Hessian matrix in this case is simply given by

$$H(\theta) = \frac{\partial g(\theta)}{\partial \theta^T} = 2\frac{1}{N}\sum_{k=1}^{N} \varphi_k \Lambda \varphi_k^T. \quad (12.56)$$

Matrix derivation of the least squares method

For practical reasons when computing the least squares solution as well as for the purpose of analyzing the statistical properties of the estimate it may be convenient to write the linear regression (12.47) in vector/matrix form as follows

$$Y = \Phi \theta_0 + e, \quad (12.57)$$

where $Y \in \mathbb{R}^{mN}$, $\Phi \in \mathbb{R}^{mN\times p}$, $\theta_0 \in \mathbb{R}^p$, $p = mr$ and $e \in \mathbb{R}^{mN}$ and given by

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^{mN}, \ \Phi = \begin{bmatrix} \varphi_1^T \\ \varphi_2^T \\ \vdots \\ \varphi_N^T \end{bmatrix} \in \mathbb{R}^{mN\times p}, \ e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix} \in \mathbb{R}^{mN}. \quad (12.58)$$

Furthermore, e, is zero mean with covariance matrix $\tilde{\Delta} = \mathrm{E}(ee^T)$. This covariance matrix will be discussed later in this section.. Define

$$\varepsilon = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_k \\ \vdots \\ \epsilon_N \end{bmatrix} = Y - \Phi\theta, \quad (12.59)$$

as the matrix of prediction errors. Hence we have that the PE criterion is given by

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N} \epsilon_k^T(\theta)\Lambda\epsilon_k(\theta) = \frac{1}{N}\varepsilon^T\tilde{\Lambda}\varepsilon, \quad (12.60)$$

where $\tilde{\Lambda}$ is a block diagonal matrix with Λ on the block diagonals, i.e.,

$$\tilde{\Lambda} = \begin{bmatrix} \Lambda & 0 & \dots & 0 \\ 0 & \Lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Lambda \end{bmatrix} \in \mathbb{R}^{mN \times mN}. \tag{12.61}$$

The nice thing about (12.60) is that the summation is not present in the last expression of the PE criterion. Hence we can obtain a more direct derivation of the parameter estimate. The gradient in (12.54) can in this case simply be expressed as

$$g(\theta) = \frac{\partial V_N(\theta)}{\partial \theta} = \frac{\partial \epsilon}{\partial \theta}\frac{\partial V_N(\theta)}{\partial \epsilon} = \frac{1}{N}(-\Phi^T)2\tilde{\Lambda}(Y - \Phi\theta). \tag{12.62}$$

The Hessian is given by

$$H(\theta) = \frac{\partial g(\theta)}{\partial \theta^T} = 2\frac{1}{N}\Phi^T\tilde{\Lambda}\Phi. \tag{12.63}$$

Solving $g(\theta) = 0$ gives the following expression for the estimate

$$\hat{\theta}_N = (\Phi^T\tilde{\Lambda}\Phi)^{-1}\Phi^T\tilde{\Lambda}Y, \tag{12.64}$$

which is identical to (12.55). It can be shown that the optimal weighting is given by $\Lambda = \Delta^{-1}$. The solution (12.55) or (12.64) with the optimal weighting matrix $\Lambda = \Delta^{-1}$ is known in the literature as the Best Linear Unbiased Estimate (BLUE). Choosing $\Lambda = I_m$ in (12.55) or (12.64) gives the OLS solution.

A short analysis of the parameter estimate is given in the following. Substituting (12.57) into (12.64) gives an expression of the difference between the parameter estimate and the true parameter vector, i.e.,

$$\hat{\theta}_N - \theta_0 = (\Phi^T\tilde{\Lambda}\Phi)^{-1}\Phi^T\tilde{\Lambda}e. \tag{12.65}$$

The parameter estimate (12.64) is an unbiased estimate since

$$\mathrm{E}(\hat{\theta}_N) = \theta_0 + (\Phi^T\tilde{\Lambda}\Phi)^{-1}\Phi^T\tilde{\Lambda}\mathrm{E}(e) = \theta_0. \tag{12.66}$$

The covariance matrix of the parameter estimate is given by

$$P = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = (\Phi^T\tilde{\Lambda}\Phi)^{-1}\Phi^T\tilde{\Lambda}\mathrm{E}(ee^T)\tilde{\Lambda}\Phi(\Phi^T\tilde{\Lambda}\Phi)^{-1}. \tag{12.67}$$

Suppose first that we are choosing the weighting matrix $\tilde{\Lambda} = \tilde{\Delta}^{-1}$ where $\tilde{\Delta} = \mathrm{E}(ee^T)$. It should be noted that $\tilde{\Delta}$ also is a block diagonal matrix with $\Delta = \mathrm{E}(e_i e_i^T)$ on the block diagonals. Then we have

$$P_{\text{BLUE}} = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = \sum_{k=1}^{N}\varphi_k\Delta^{-1}\varphi_k^T = (\Phi^T\tilde{\Delta}^{-1}\Phi)^{-1}. \tag{12.68}$$

The Ordinary Least squares (OLS) estimate is obtained by choosing $\Lambda = I_m$, i.e., no weighting. For the OLS estimate we have that

$$P_{\text{OLS}} = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{E}(ee^T)\Phi(\Phi^T\Phi)^{-1}. \tag{12.69}$$

In the single output case we have that $\tilde{\Delta} = \mathrm{E}(ee^T) = \delta_0 I_N$. Using this in (12.69) gives the standard result for univariate ($m = 1$) data which is presented in the literature, i.e.,

$$P_{\mathrm{OLS}} = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = \delta_0(\Phi^T\Phi)^{-1}. \tag{12.70}$$

An important result is that

$$P_{\mathrm{BLUE}} \leq P_{\mathrm{OLS}}. \tag{12.71}$$

In fact, all other symmetric and positive definite weighting matrices gives a larger parameter covariance matrix than the covariance matrix of the BLUE estimate.

Alternative matrix derivation of the least squares method

Another linear regression model formulation which is frequently used, e.g., in the Chemometrics literature, is given by

$$Y = XB + E, \tag{12.72}$$

where $Y \in \mathbb{R}^{N\times m}$ is a matrix of dependent variables (outputs), $X \in \mathbb{R}^{N\times r}$ is a matrix of the independent variables (regressors or inputs), $B \in \mathbb{R}^{m\times r}$ is a matrix of system parameters (regression coefficients). $E \in \mathbb{R}^{N\times m}$ is a matrix of white noise.

The OLS solution is simply obtained by solving the normal equations $X^TY = X^TXB$ for B, i.e,

$$B_{\mathrm{OLS}} = (X^TX)^{-1}X^TY, \tag{12.73}$$

it X^TX is non-singular.

12.2 Input and output model structures

Input and output discrete time model structures are frequently used in connection with the prediction error methods and software. We will in this section give a description of these model structures and the connection with the general state space model structure.

12.2.1 ARMAX model structure

An ARMAX model structure is defined by the (polynomial) model

$$A(q)y_k = B(q)u_k + C(q)e_k. \tag{12.74}$$

The acronym ARMAX comes from the fact that the term $A(q)y_k$ is defined as an Auto Regressive (AR) part, the term $C(q)e_k$ is defined as an Moving Average (MA) part and that the part $B(q)u_k$ represents eXogenous (X) inputs. Note in connection with this that an Auto Regressive (AR) model is of the form $A(q)y_k = e_k$, i.e.

with additive equation noise. The noise term, $C(q)e_k$, in a so called ARMA model $A(q)y_k = C(q)e_k$ represents a moving average of the white noise e_k.

The ARMAX model structure can be deduced from a relatively general state space model. This will be illustrated in the following example.

Example 12.1 (Estimator canonical form to polynomial form) *Consider the following single input and single output discrete time state space model on estimator canonical form*

$$x_{k+1} = \begin{bmatrix} -a_1 & 1 \\ -a_0 & 0 \end{bmatrix} x_k + \begin{bmatrix} b_1 \\ b_0 \end{bmatrix} u_k + \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} v_k \tag{12.75}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + e u_k + f v_k \tag{12.76}$$

where u_k is a known deterministic input signal, v_k is an unknown white noise process and $x_k^T = \begin{bmatrix} x_{k+1}^1 & x_{k+1}^2 \end{bmatrix}$ is the state vector.

An input output model formulation can be derived as follows

$$x_{k+1}^1 = -a_1 x_k^1 + x_k^2 + b_1 u_k + c_1 v_k \tag{12.77}$$
$$x_{k+1}^2 = -a_0 x_k^1 + b_0 u_k + c_0 v_k \tag{12.78}$$
$$y_k = x_k^1 + e u_k + f v_k \tag{12.79}$$

Express Equation (12.77) with $k =: k+1$ and substitute for x_{k+1}^2 defined by Equation (12.78). This gives an equation in terms of the 1st state x_k^1. Finaly, eliminate x_k^1 by using Equation (12.79). This gives

$$\begin{aligned} &\begin{bmatrix} 1 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} y_k \\ y_{k-1} \\ y_{k-2} \end{bmatrix} \\ &= \begin{bmatrix} e & b_1 + a_1 e & b_0 + a_0 e \end{bmatrix} \begin{bmatrix} u_k \\ u_{k-1} \\ u_{k-2} \end{bmatrix} + \begin{bmatrix} f & c_1 + a_1 f & c_0 + a_0 f \end{bmatrix} \begin{bmatrix} v_k \\ v_{k-1} \\ v_{k-2} \end{bmatrix} \end{aligned} \tag{12.80}$$

Let us introduce the backward shift operator q^{-1} such that $q^{-1}u_k = u_{k-1}$. This gives the following polynomial (or transfer function) model

$$\begin{aligned} &\overbrace{(1 + a_1 q^{-1} + a_0 q^{-2})}^{A(q)} y_k \\ &= \overbrace{(e + (b_1 + a_1 e) q^{-1} + (b_0 + a_0 e) q^{-2})}^{B(q)} u_k + \overbrace{(f + (c_1 + a_1 f) q^{-1} + (c_0 + a_0 f) q^{-2})}^{C(q)} v_k \end{aligned} \tag{12.81}$$

which is equal to an ARMAX model structure.

Example 12.2 (Observability canonical form to polynomial form) *Consider the following single input and single output discrete time state space model on observability canonical form*

$$x_{k+1} = \begin{bmatrix} 0 & 1 \\ -a_1 & -a_0 \end{bmatrix} x_k + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} u_k + \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} v_k, \tag{12.82}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + eu_k + fv_k, \tag{12.83}$$

where u_k is a known deterministic input signal, v_k is an unknown white noise process and $x_k^T = \begin{bmatrix} x_{k+1}^1 & x_{k+1}^2 \end{bmatrix}$ is the state vector. An input and output model can be derived by eliminating the states in (12.82) and (12.83). We have

$$x_{k+1}^1 = x_k^2 + b_0 u_k + c_0 v_k, \tag{12.84}$$
$$x_{k+1}^2 = -a_1 x_k^1 - a_0 x_k^2 + b_1 u_k + c_1 v_k, \tag{12.85}$$
$$y_k = x_k^1 + eu_k + fv_k. \tag{12.86}$$

Solve (12.84) for x_k^2 and substitute into (12.85). This gives an equation in x_k^1, i.e.,

$$x_{k+2}^1 - b_0 u_{k+1} - c_0 v_{k+1} = -a_1 x_k^1 - a_0(x_{k+1}^1 - b_0 u_k - c_0 v_k) + b_1 u_k + c_1 v_k. \tag{12.87}$$

Solve (12.86) for x_k^1 and substitute into (12.87). This gives

$$\begin{aligned} y_{k+2} - eu_{k+2} - fv_{k+2} - b_0 u_{k+1} - c_0 v_{k+1} = -a_1(y_k - eu_k - fv_k) \\ -a_0(y_{k+1} - eu_{k+1} - fv_{k+1}) + a_0 b_0 u_k + a_0 c_0 v_k + b_1 u_k + c_1 v_k. \end{aligned} \tag{12.88}$$

This gives the input and output equation

$$\begin{aligned} \begin{bmatrix} 1 & a_0 & a_1 \end{bmatrix} \begin{bmatrix} y_{k+2} \\ y_{k+1} \\ y_k \end{bmatrix} &= \begin{bmatrix} e & b_0 + a_0 e & b_1 + a_0 b_0 + a_1 e \end{bmatrix} \begin{bmatrix} u_{k+2} \\ u_{k+1} \\ u_k \end{bmatrix} \\ &+ \begin{bmatrix} f & c_0 + a_0 f & c_1 + a_0 c_0 + a_1 f \end{bmatrix} \begin{bmatrix} v_{k+2} \\ v_{k+1} \\ v_k \end{bmatrix}. \end{aligned} \tag{12.89}$$

Hence, we can write (12.89) as an ARMAX polynomial model

$$A(q)y_k = B(q)u_k + C(q)v_k, \tag{12.90}$$
$$A(q) = 1 + a_0 q^{-1} + a_1 q^{-2}, \tag{12.91}$$
$$B(q) = e + (b_0 + a_0 e)q^{-1} + (b_1 + a_0 b_0 + a_1 e)q^{-2}, \tag{12.92}$$
$$C(q) = f + (c_0 + a_0 f)q^{-1} + (c_1 + a_0 c_0 + a_1 f)q^{-2}. \tag{12.93}$$

12.2.2 ARX model structure

An Auto Regression (AR) model $A(q)y_k = e_k$ with eXtra (or eXogenous) inputs (ARX) model can be expressed as follows

$$A(q)y_k = B(q)u_k + e_k \tag{12.94}$$

where the term $A(q)y_k$ is the Auto Regressive (AR) part and the term $B(q)u_k$ represents the part with the eXtra (X) inputs. The eXtra variables u_k are called eXogenous in econometrics. Note also that the ARX model also is known as an *equation error model* because of the additive error or white noise term e_k.

It is important to note that the parameters in an ARX model, e.g. as defined in Equation (12.94) where e_k is a white noise process, can be identified directly by the Ordinary Least Squares (OLS) method, if the inputs, u_k, are informative enough. A problem with the ARX structure is off-course that the noise model (additive noise) is to simple in many practical cases.

Example 12.3 (ARX and State Space model structure) *Comparing the ARX model structure (12.94) with the more general model structure given by (12.81) we find that the state space model (12.75) and (12.76) is equivalent to an ARX model if*

$$c_1 = -a_1 f,\ c_0 = -a_0 f,\ e_k = f v_k \tag{12.95}$$

This gives the following ARX model

$$\overbrace{(1 + a_1 q^{-1} + a_0 q^{-2})}^{A(q)} y_k = \overbrace{(e + (b_1 + a_1 e) q^{-1} + (b_0 + a_0 e) q^{-2})}^{B(q)} u_k + f v_k \tag{12.96}$$

which, from the above discussion, have the following state space model equivalent

$$x_{k+1} = \begin{bmatrix} -a_1 & 1 \\ -a_0 & 0 \end{bmatrix} x_k + \begin{bmatrix} b_1 \\ b_0 \end{bmatrix} u_k + \begin{bmatrix} -a_1 \\ -a_0 \end{bmatrix} f v_k \tag{12.97}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + e u_k + f v_k \tag{12.98}$$

It is at first sight not easy to see that this state space model is equivalent to an ARX model. It is also important to note that the ARX model has a state space model equivalent. The noise term $e_k = f v_k$ is white noise if v_k is withe noise. The noise e_k appears as both process (state equation) noise and measurements (output equation) noise. The noise is therefore filtred throug the process dynamics.

12.2.3 OE model structure

An Output Error (OE) model structure can be represented as the following polynomial model

$$A(q) y_k = B(q) u_k + A(q) e_k \tag{12.99}$$

or equivalently for single output plants

$$y_k = \frac{B(q)}{A(q)} u_k + e_k \tag{12.100}$$

Example 12.4 (OE and State Space model structure) *From (12.81) we find that the state space model (12.75) and (12.76) is equivalent to an OE model if*

$$c_1 = 0,\ c_0 = 0,\ f = 1,\ v_k = e_k \tag{12.101}$$

Hence, the following OE model

$$\begin{aligned} \overbrace{(1 + a_1 q^{-1} + a_0 q^{-2})}^{A(q)} y_k &= \overbrace{(e + (b_1 + a_1 e) q^{-1} + (b_0 + a_0 e) q^{-2})}^{B(q)} u_k \\ &+ \overbrace{(1 + a_1 q^{-1} + a_0 q^{-2})}^{A(q)} e_k \end{aligned} \tag{12.102}$$

have the following state space model equivalent

$$x_{k+1} = \begin{bmatrix} -a_1 & 1 \\ -a_0 & 0 \end{bmatrix} x_k + \begin{bmatrix} b_1 \\ b_0 \end{bmatrix} u_k \tag{12.103}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + e u_k + e_k \tag{12.104}$$

Note that the noise term e_k appears in the state space model as an equivalent measurements noise term or output error term.

12.2.4 BJ model structure

In an ARMAX model structure the dynamics in the path from the inputs u_k to the output y_k is the same as the dynamics in the path from the process noise e_k to the output. In practice it is quite realistic that some or all of the dynamics in the two paths are different. This can be represented by the Box Jenkins (BJ) model structure

$$F(q)D(q)y_k = D(q)B(q)u_k + F(q)C(q)e_k \tag{12.105}$$

or for single output systems

$$y_k = \frac{B(q)}{F(q)} u_k + \frac{C(q)}{D(q)} e_k \tag{12.106}$$

In the BJ model the moving average noise (coloured noise) term $C(q)e_k$ is filtred throug the dynamics represented by the polynomial $D(q)$. Similarly, the dynamics in the path from the inputs u_k are represented by the polynomial $A(q)$.

However, it is important to note that the BJ model structure can be represented by an equivalent state space model structure.

12.2.5 Summary

The family of polynomial model structures

$$\begin{array}{ll} BJ & : \text{Box Jenkins} \\ ARMAX & : \text{Auto Regressive Moving Average with eXtra inputs} \\ ARX & : \text{Auto Regressive with eXtra inputs} \\ ARIMAX & : \text{Auto Regressive Integrating Moving Average with eXtra inputs} \\ OE & : \text{Output Error} \end{array} \tag{12.107}$$

can all be represented by a state space model of the form

$$y_k = Dx_k + Eu_k + Fe_k, \tag{12.108}$$

$$x_{k+1} = Ax_k + Bu_k + Ce_k. \tag{12.109}$$

When using the prediction error methods for system identification the model structure and the order of the polynomial need to be specified. It is important that the prediction error which is to be minimized is a function of as few unknown parameters as possible. Note also that all of the model structures discussed above are linear models. However, the optimization problem of computing the unknown parameters are highly non-linear. Hence, we can run into numerical problems. This is especially the case for multiple output and MIMO systems.

The subspace identification methods, e.g. **DSR**, is based on the general state space model defined by (12.108) and (12.109. The subspace identification methods is therefore flexibile enough to identify systems described by all the polynomial model structures (12.107). Note also that not even the system order n need to be specified beforehand when using the subspace identification method.

Remark 12.1 (Other notations frequently used)
Another notation for the backward shift operator q^{-1} (defined such that $q^{-1}y_k = y_{k-1}$) is the $z-$ operator, (z^{-1} such that $z^{-1}y_k = y_{k-1}$. The notation $A(q)$, $B(q)$ and so on are used by Ljung (1999). In Söderström and Stoica (1989) the notation $A(q^{-1})$, $B(q^{-1})$ are used for the same polynomials.

12.3 Optimal one-step-ahead predictions

12.3.1 State Space Model

Consider the innovations model

$$\bar{x}_{k+1} = A\bar{x}_k + Bu_k + Ke_k, \tag{12.110}$$

$$y_k = D\bar{x}_k + Eu_k + e_k, \tag{12.111}$$

where K is the Kalman filter gain matrix. A predictor $\bar{y}_k$ for y_k can be defined as the two first terms on the right hand side of (12.111), i.e.

$$\bar{y}_k = D\bar{x}_k + Eu_k. \tag{12.112}$$

The equations for the (optimal) predictor (Kalman filter) is therefore given by

$$\bar{x}_{k+1} = A\bar{x}_k + Bu_k + K(y_k - \bar{y}_k), \tag{12.113}$$

$$\bar{y}_k = D\bar{x}_k + Eu_k, \tag{12.114}$$

which can be written as

$$\bar{x}_{k+1} = (A - KD)\bar{x}_k + (B - KE)u_k + Ky_k, \tag{12.115}$$

$$\bar{y}_k = D\bar{x}_k + Eu_k. \tag{12.116}$$

Hence, the optimal prediction, $\bar{y}_k$, of the output y_k can simply be obtained by simulating (12.115) and (12.116) with a specified initial predicted state $\bar{x}_1$. The results is known as the one-step ahead predictions. The name one-step ahead predictor comes from the fact that the prediction of y_{k+1} is based upon all outputs up to time k as well as all relevant inputs. This can be seen by writing (12.115) and (12.116) as

$$\bar{y}_{k+1} = D(A - KD)\bar{x}_k + D(B - KE)u_k + Eu_{k+1} + DKy_k, \tag{12.117}$$

which is the one-step ahead prediction.

Note that the optimal prediction (12.115) and (12.116) can be written as the following transfer function model

$$\bar{y}_k(\theta) = H_e^d(q)u_k + H_e^s(q)y_k, \tag{12.118}$$

where

$$H_e^d(q) = D(qI - (A - KD))^{-1}(B - KE) + E, \tag{12.119}$$
$$H_e^s(q) = D(qI - (A - KD))^{-1}K. \tag{12.120}$$

The derivation of the one-step-ahead predictor for polynomial models is further discussed in the next section.

12.3.2 Input-output model

The linear system can be expressed as the following input and output polynomial model

$$y_k = G(q)u_k + H(q)e_k. \tag{12.121}$$

The noise term e_k can be expressed as

$$H^{-1}(q)y_k = H^{-1}(q)G(q)u_k + e_k. \tag{12.122}$$

Adding y_k on both sides gives

$$y_k = (I - H^{-1}(q))y_k + H^{-1}(q)G(q)u_k + e_k. \tag{12.123}$$

The prediction of the output is given by the first two terms on the right hand side of (12.123) since e_k is white and therefore cannot be predicted, i.e.,

$$\bar{y}_k(\theta) = (I - H^{-1}(q))y_k + H^{-1}(q)G(q)u_k. \tag{12.124}$$

Loosely spoken, the optimal prediction of y_k is given by the predictor, $\bar{y}_k$, so that a measure of the difference $e_k = y_k - \bar{y}_k(\theta)$, e.g., the variance, is minimized with respect to the parameter vector θ, over the data horizon. We also assume that a sufficient model structure is used, and that the unknown parameters is parameterized in θ. Ideally, the prediction error e_k will be white.

12.4 Optimal M-step-ahead prediction

12.4.1 State space models

The aim of this section is to develop the optimal j-step-ahead predictions $\bar{y}_{k+j}\ \forall\ j = 1, ..., M$. The optimal one-steap-ahead predictor, i.e., for $j = 1$, is defined in (12.117). We will in the following derivation assume that only outputs up to time k is available. Furthermore, it is assumed that all inputs which are needed in the derivation are available. This is realistic in practice. Only, past outputs $\ldots, y_{k-2}, y_{k-1}, y_k$ are known. Note that we can assume values for the future inputs, or the future inputs can be computed in an optimization strategy as in Model Predictive Control (MPC). The Kalman filter on innovations form is

$$\bar{x}_{k+1} = A\bar{x}_k + Bu_k + Ke_k, \tag{12.125}$$

$$y_k = D\bar{x}_k + Eu_k + e_k. \tag{12.126}$$

The prediction for $j = 1$, i.e., the one-step-ahead prediction, is given in (12.117). The other predictions are derived as follows. The output at time $k := k + 2$ is then defined by

$$\bar{x}_{k+2} = A\bar{x}_{k+1} + Bu_{k+1} + Ke_{k+1}, \tag{12.127}$$

$$y_{k+2} = D\bar{x}_{k+2} + Eu_{k+2} + e_{k+2}. \tag{12.128}$$

The (white) noise vectors, e_{k+1} and e_{k+2} are all in the future, and they can not be predicted because they are white. The best prediction of y_{k+2} must then be to put $e_{k+1} = 0$ and $e_{k+2} = 0$. Hence, the predictor for $j = 2$ is

$$x_{k+1} = \bar{x}_{k+1}, \tag{12.129}$$

$$x_{k+2} = Ax_{k+1} + Bu_{k+1}, \tag{12.130}$$

$$\bar{y}_{k+2} = Dx_{k+2} + Eu_{k+2}. \tag{12.131}$$

This can simply be generalized for $j > 2$ as presented in the following lemma.

Lemma 12.1 (Optimal j-step-ahead predictions)
The optimal j-step-ahead predictions, $j = 1, 2, \ldots, M$, can be obtained by a pure simulation of the system (A, B, D, E) with the optimal predicted Kalman filter state $\bar{x}_{k+1}$ as the initial state.

$$x_{k+j+1} = Ax_{k+j} + Bu_{k+j}, \tag{12.132}$$

$$y_{k+j} = Dx_{k+j} + Eu_{k+j}, \ \forall\ j = 1, \ldots, M \tag{12.133}$$

where the initial state $x_{k+1} = \bar{x}_{k+1}$ is given from the Kalman filter state equation

$$\bar{x}_{k+1} = (A - KD)\bar{x}_k + (B - KE)u_k + Ky_k, \tag{12.134}$$

where the initial state $\bar{x}_1$ is known and specified. This means that all outputs up to time, k, and all relevant inputs, are used to predict the output at time $k + 1$, ..., $k + M$.

Example 12.5 (Prediction model on matrix form (proper system))
Given the Kalman filter matrices (A, B, D, E, K) of an only proper process, and the initial predicted state $\bar{x}_k$. The predictions $\bar{y}_{k+1}$, $\bar{y}_{k+2}$ and $\bar{y}_{k+3}$ can be written in compact form as follows

$$\begin{bmatrix} \bar{y}_{k+1} \\ \bar{y}_{k+2} \\ \bar{y}_{k+3} \end{bmatrix} = \begin{bmatrix} D \\ DA \\ DA^2 \end{bmatrix} (A - KD)\bar{x}_k + \begin{bmatrix} D \\ DA \\ DA^2 \end{bmatrix} Ky_k$$

$$+ \begin{bmatrix} D(B-KE) & E & 0 & 0 \\ DA(B-KE) & DB & E & 0 \\ DA^2(B-KE) & DAB & DB & E \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ u_{k+3} \end{bmatrix}. \tag{12.135}$$

This formulation may be useful in e.g., model predictive control.

Example 12.6 (Prediction model on matrix form (strictly proper system))

Given the Kalman filter matrices (A, B, D, K) of an strictly proper system, and the initial predicted state $\bar{x}_k$. The predictions $\bar{y}_{k+1}$, $\bar{y}_{k+2}$ and $\bar{y}_{k+3}$ can be written in compact form as follows

$$\begin{aligned}\begin{bmatrix}\bar{y}_{k+1}\\ \bar{y}_{k+2}\\ \bar{y}_{k+3}\end{bmatrix} &= \begin{bmatrix}D\\ DA\\ DA^2\end{bmatrix}(A-KD)\bar{x}_k + \begin{bmatrix}D\\ DA\\ DA^2\end{bmatrix}Ky_k\\ &+ \begin{bmatrix}DB & 0 & 0\\ DAB & DB & 0\\ DA^2B & DAB & DB\end{bmatrix}\begin{bmatrix}u_k\\ u_{k+1}\\ u_{k+2}\end{bmatrix}.\end{aligned} \tag{12.136}$$

This formulation is more realistic for control considerations, where we almost always have a delay of one sample between the input and the output.

12.5 Matlab implementation

We will in this section illustrate a simple but general MATLAB implementation of a Prediction Error Method (PEM).

12.5.1 Tutorial: SS-PEM Toolbox for MATLAB

The MATLAB files in this work is built up as a small toolbox, i.e., the SS-PEM toolbox for MATLAB. This toolbox should be used in connection with the D-SR Toolbox for MATLAB. An overview of the functions in the toolbox is given by the command

```
>> help ss-pem
```

The main prediction error method is implemented in the function **sspem.m**. An initial parameter vector, θ_1, is computed by first using the DSR algorithm to compute an initial state space Kalman filter model, i.e. the corresponding matrices $(A, B, D, E, K, x1)$. This model is then transformed to observability canonical form by the D-SR Toolbox function **ss2cf.m**. The initial parameter vector, θ_1, is then taken as the free parameters in these canonical state space model matrices. The minimizing parameter vector, $\hat{\theta}$, is then computed by the Optimization toolbox function **fminunc.m**. Other optimization algorithms can off-course be used.

A tutorial and overview is given in the following. Assume that output and input data matrices, $Y \in \mathbb{R}^{N\times m}$ and $U \in \mathbb{R}^{N\times r}$ are given. A state space model (Kalman filter) can then simply be identified by the PEM function **sspem.m**, in the MATLAB command window as follows.

```
>> [A,B,D,E,K,x1]=sspem(Y,U,n);
```

A typical drawback with PEMs is that the system order, n, has to be specified beforehand. A good solution is to analyze and identify the system order, n, by the subspace algorithm DSR. The identified model is represented in observability canonical form. That is a state space realization with as few free parameters as possibile. This realization can be illustrated as follows. Consider a system with $m = 2$ outputs, $r = 2$ inputs and $n = 3$ states, which can be represented by a linear model. The resulting model from **sspem.m** will be on Kalman filter innovations form, i.e.,

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ke_k, && (12.137)\\ y_k &= Dx_k + Eu_k + e_k, && (12.138)\end{aligned}$$

with initial predicted state x_1 given, or on prediction form, i.e.,

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + K(y_k - \bar{y}_k), && (12.139)\\ \bar{y}_k(\theta) &= Dx_k + Eu_k, && (12.140)\end{aligned}$$

and where the model matrices (A, B, D, E, K) and the initial predicted state x_1 are given by

$$A = \begin{bmatrix} 0 & 0 & 1 \\ \theta_1 & \theta_3 & \theta_5 \\ \theta_2 & \theta_4 & \theta_6 \end{bmatrix}, \; B = \begin{bmatrix} \theta_7 & \theta_{10} \\ \theta_8 & \theta_{11} \\ \theta_9 & \theta_{12} \end{bmatrix}, \; K = \begin{bmatrix} \theta_{13} & \theta_{16} \\ \theta_{14} & \theta_{17} \\ \theta_{15} & \theta_{18} \end{bmatrix}, \qquad (12.141)$$

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \; E = \begin{bmatrix} \theta_{19} & \theta_{21} \\ \theta_{20} & \theta_{22} \end{bmatrix}, \; x_1 \begin{bmatrix} \theta_{23} \\ \theta_{24} \\ \theta_{25} \end{bmatrix}. \qquad (12.142)$$

This model is on a so called observability canonical form, i.e. the third order model has as few free parameters as possibile. The number of free parameters are in this example 25. The observability canonical form is such that the D matrix is filled with ones and zeros. We have that $D = \text{ones}(m, n)$. Furthermore, we have that a concatenation of the D matrix and the first $n-m$ rows of the A matrix is the identity matrix, i.e. $\begin{bmatrix} D \\ A(1:n-m,:) \end{bmatrix} = I_n$. This may be used as a rule when writing up the $n - m$ first rows of the A matrix. The rest of the A matrix as well as the matrices B, K, E and x_1 is filled with parameters. However, the user does not have to deal with the construction of the canonical form, when using the SSPEM toolbox.

The total number of free parameters in a linear state space model and in the parameter vector $\theta \in \mathbb{R}^p$ are in general

$$p = mn + nr + nm + mr + n = (2m + r + 1)n + mr \qquad (12.143)$$

An existing DSR model can also be refined as follows. Suppose now that a state space model, i.e. the matrices $(A, B, D, E, K, x1)$ are identified by the DSR algorithm as follows

```
>> [A,B,D,E,C,F,x1]=dsr(Y,U,L);
>> K=C*inv(F);
```

This model can be transformed to observability canonical form and the free parameters in this model can be mapped into the parameter vector θ_1 by the function **ss2thp.m**, as follows

```
>> th_1=ss2thp(A,B,D,E,K,x1);
```

The model parameter vector θ_1 can be further refined by using these parameters as initial values to the PEM method **sspem.m**. E.g. as follows

```
>> [A,B,D,E,K,x1,V,th]=sspem(Y,U,n,th_1);
```

The value of the prediction error criterion, $V(\theta)$, is returned in V. The new and possibly better (more optimal) parameter vector is returned in th.

Note that for a given parameter vector, then the state space model matrices can be constructed by

```
>> (A,B,D,E,K,x1)=thp2ss(th,n,m,r);
```

It is also worth to note that the value, $V(\theta)$, of the prediction error criterion is evaluated by the MATLAB function **vfun_mo.m**. The data matrices Y and U and the system order, n, must first be defined as global variables, i.e.

```
>> global Y U n
```

The PE criterion is then evaluated as

```
>> V=vfun_mo(th);
```

where th is the parameter vector. Note that the parameter vector must be of length p as explained above.

See also the function **ss2cf.m** in the D-SR Toolbox for MATLAB which returns an observability form state space realization from a state space model.

```
function [A,B,D,E,K,x0,V,th,th_0]=sspem(y,u,ns,th_0)
% SSPEM State Space Prediction error Method.
% Syntax:
% [A,B,D,E,K,x0,V,th,th_0]=sspem(Y,U,n)
% [A,B,D,E,K,x0,V,th,th_0]=sspem(Y,U,n,th_0)
% Purpose:
% Computing the Kalman filter model matrices by a prediction
% error method (PEM). The initial parameters are computed by
% DSR. FMINUNC is used for the optimization.
% ON INPUT:
% Y, U- Data matrices of size (N x m) and (N x r) respectively.
% n   - The system order.
% Functions called: vfun_mo, thp2ss

% Written by: DDIR, 09.03.2001
if nargin==3; th_0=0; end
global Y U n
% Define some local parameters.
Y=y; U=u; n=ns;
[Ny,ny]=size(Y);
[Nu,nu]=size(U);
if nargin==3           % Use DSR to estimate initial parameters.
% Using DSR to estimate initial state space model.
 L=ceil(n/ny); J=L; g=1;
 [a,b,d,e,c,f,x0]=dsr(Y,U,L,g,J,1,n);
 k=c*pinv(f);
% Transform DSR model to canonical form.
% and construct the inital parameter vector, th_0, from this model.
 th_0=ss2thp(a,b,d,e,k,x0);
elseif nargin==4       % Use initial parameter vector provided by the user.
   p=(2*ny+nu+1)*n+ny*nu;
   pth_0=length(th_0);
   if pth_0 ~= p
      disp('ERROR: wrong number of parameters');
      disp('Number of parameters in th_0 is:'); pth_0
      disp('Number of parameters should be: '); p
   end
end
% Compute optimal parameters if convergence
th=fminunc('vfun_mo',th_0);
V=vfun_mo(th);
% Transform parameter vector to state space model matrices.
[A,B,D,E,K,x0]=thp2ss(th,n,ny,nu);
% END SSPEM
```

```
function V=vfun_mo(th);
% VFUN_MO
% Syntax: V=vfun_mo(th);
% Purpose: Computing the prediction error criterion from
% the parameter vector.
% Functions called: thp2ss, dlsim

% Written by: DDIR, 09.03.2001
global Y U n
% Define some local parameters.
[Ny,ny]=size(Y);
[Nu,nu]=size(U);
N=min(Ny,Nu);
% Construct matrices from parameter vector
[A,B,D,E,K,x0]=thp2ss(th,n,ny,nu);
% Compute the value of the prediction error criterion.
Yp=dlsim(A-K*D,[B-K*E K],D,[E zeros(ny,ny)],[U Y],x0);
pe=Y-Yp;
R=pe'*pe/N;
ipec=1;
if ipec==1
 V=trace(R);
elseif ipec==2
 V=det(R);
end
% END VFUN_MO
```

```
function [A,B,D,E,K,x0]=thp2ss(th,n,ny,nu)
% THP2SS
% Syntax: [A,B,D,E,K,x0]=thp2ss(th,n,ny,nu)
% Purpose: Construct the state space Kalman filter
% model matrices from the parameter vector, theta.
% It is assumed that the parameters are stored in order
% as follows, th=[A(:);B(:);K(:);E(:);x0(:)].

% Written by: DDIR, 09.03.2001
% Make A matrix
th_a=th(1:ny*n); th_a=reshape(th_a,ny,n);
A=[rot90(rot90(eye(n-ny,n)));th_a];
% Make B matrix
th_b=th(ny*n+1:ny*n+n*nu);
B=reshape(th_b,n,nu);
% Make K matrix
th_k=th(ny*n+n*nu+1:ny*n+n*nu+n*ny);
K=reshape(th_k,n,ny);
% Make E matrix
th_e=th(ny*n+n*nu+n*ny+1:ny*n+n*nu+n*ny+ny*nu);
E=reshape(th_e,ny,nu);
% Make x0
th_x0=th(ny*n+n*nu+n*ny+ny*nu+1:ny*n+n*nu+n*ny+ny*nu+n);
x0=reshape(th_x0,n,1);
% Make D matrix with fixed parameters
D=eye(ny,n);
% END THP2SS
```

```
function th=ss2thp(a,b,d,e,k,x0)
% SS2THP
% Syntax: th=ss2thp(A,B,D,E,K,x0)
% Purpose: Given a state space model matrices (A,B,D,E,K,x0).
% ss2thp transform this model to observability canonical form
% and returns the free parameters in the parameter vector, th.

% Written by: DDIR, 09.03.2001
[ny,nu]=size(e);
n=size(a,1);
% Transform DSR model to canonical form.
[ac,bc,dc]=ss2cf(a,[b k],d);
kc=bc(:,nu+1:ny+nu);
bc=bc(:,1:nu);
% Construct inital parameter vector, theta, from DSR model
% in canonical form.
th_a=ac(n-ny+1:n,:);
th=[th_a(:);bc(:);kc(:);e(:);x0(:)];
% END SS2THP
```

12.6 Recursive least squares method

The OLS estimate (12.55) can be written as

$$\hat{\theta}_t = (\sum_{k=1}^{t} \varphi_k \Lambda \varphi_k^T)^{-1} \sum_{k=1}^{t} \varphi_k \Lambda y_k. \tag{12.144}$$

We have simply replaced N in (12.55) by t in order to stress the dependence of $\hat{\theta}$ on time, t. Let us define the indicated inverse in (12.144) by

$$P_t = (\sum_{k=1}^{t} \varphi_k \Lambda \varphi_k^T)^{-1}. \tag{12.145}$$

From this definition we have that

$$P_t^{-1} = P_{t-1}^{-1} + \varphi_t \Lambda \varphi_t^T \tag{12.146}$$

The idea is now to write (12.144) as

$$\begin{aligned}
\hat{\theta}_t &= P_t \sum_{k=1}^{t} \varphi_k \Lambda y_k && (12.147)\\
&= P_t(\sum_{k=1}^{t-1} \varphi_k \Lambda y_k + \varphi_t \Lambda y_t) && (12.148)\\
&= P_t(P_{t-1}^{-1}\hat{\theta}_{t-1} + \varphi_t \Lambda y_t) && (12.149)
\end{aligned}$$

From (12.144) we have that $\hat{\theta}_{t-1}$ is given by

$$\hat{\theta}_{t-1} = P_{t-1} \sum_{k=1}^{t-1} \varphi_k \Lambda y_k \tag{12.150}$$

Substituting this into (12.149) gives

$$\hat{\theta}_t = P_t(\sum_{k=1}^{t-1} \varphi_k \Lambda y_k + \varphi_t \Lambda y_t) \tag{12.151}$$

$$= P_t(P_{t-1}^{-1} \hat{\theta}_{t-1} + \varphi_t \Lambda y_t) \tag{12.152}$$

From (12.146) we have that

$$P_{t-1}^{-1} = P_t^{-1} - \varphi_t \Lambda \varphi_t^T \tag{12.153}$$

Substituting this into (12.149) gives

$$\hat{\theta}_t = P_t((P_t^{-1} - \varphi_t \Lambda \varphi_t^T) \hat{\theta}_{t-1} + \varphi_t \Lambda y_t) \tag{12.154}$$

Chapter 13

The conjugate gradient method and PLS

13.1 Introduction

The PLS method for univariate data (usually denoted PLS1) is optimal in a prediction error sense, Di Ruscio (1998). Unfortunately (or not), the PLS algorithm for multivariate data (usually denoted PLS2 in the literature) is not optimal in the same way as PLS1. It is often judged that PLS has good predictive properties. This statement is based on experience and numerous applications. This may well be true. Intuitively, we believe that the predictive performance of PLS1 and PLS2 is different. The reason for this is that PLS1 is designed to be optimal on the identification data while PLS2 is not. If the model structure is correct and time invariant, then, PLS1 would also be good for prediction.

We will in this note propose a new method which combines the predictive properties of PLS1 and PLS2. We are seeking a method which has better prediction properties than both PLS1 and PLS2.

13.2 Definitions and preliminaries

Definition 13.1 (Krylov matrix)
Given data matrices $X \in \mathbb{R}^{N\times r}$ and $Y \in \mathbb{R}^{N\times m}$. The ith component Krylov matrix $K_i \in \mathbb{R}^{r\times im}$ for the pair (X^TX, X^TY) is defined as

$$K_i = \left[\, X^TY \;\, X^TXX^TY \;\, (X^TX)^2X^TY \;\cdots\; (X^TX)^{i-1})X^TY \,\right], \tag{13.1}$$

where i denotes the number of m-block columns.

13.3 The method

Proposition 13.1
Given data matrices $X \in \mathbb{R}^{N\times r}$ and $Y \in \mathbb{R}^{N\times m}$. Define the ath component Krylov

matrix $K_a \in \mathbb{R}^{r\times am}$ of the pair (X^TX, X^TY) from Definition 13.1. Compute the SVD of the compound matrix

$$\begin{bmatrix} X^TXK_a & X^TY \end{bmatrix} = USV^T, \tag{13.2}$$

where $U \in \mathbb{R}^{r\times r}$, $S \in \mathbb{R}^{r\times(a+1)m}$ and $V \in \mathbb{R}^{(a+1)m\times(a+1)m}$.

Chose the integer number $1 \leq n \leq r$ to define the number of singular values to represent the signal part X^TXK_a, and partition the matrices S and V such that

$$\begin{bmatrix} X^TXK_a & X^TY \end{bmatrix} = U \begin{bmatrix} S_1 & S_2 \end{bmatrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}^T, \tag{13.3}$$

where $S_1 \in \mathbb{R}^{r\times n}$, $S_2 \in \mathbb{R}^{r\times(a+1)m-n}$, $V_{11} \in \mathbb{R}^{am\times n}$, $V_{21} \in \mathbb{R}^{m\times n}$, $V_{12} \in \mathbb{R}^{am\times(a+1)m-n}$ and $V_{22} \in \mathbb{R}^{m\times(a+1)m-n}$.

The NTPLS solution is then computed as

$$B_{\text{NTPLS}} = K_a p^\star, \tag{13.4}$$

where $p^\star \in \mathbb{R}^{am\times m}$ is defined as

$$p^\star = -V_{12}V_{22}^\dagger, \tag{13.5}$$

or equivalently

$$p^\star = (V_{11}^T)^\dagger V_{21}^T. \tag{13.6}$$

Furthermore, the residual $X^TY - X^TXB_{\text{NTPLS}}$ is determined by

$$\mathcal{E}_{\text{NTPLS}} = US_2V_{22}^\dagger. \tag{13.7}$$

Proof 13.1 *Proof of (13.5) and (13.7). We have from (13.3) that*

$$\begin{bmatrix} X^TXK_a & X^TY \end{bmatrix} = US_1 \begin{bmatrix} V_{11}^T & V_{21}^T \end{bmatrix} + US_2 \begin{bmatrix} V_{12}^T & V_{22}^T \end{bmatrix}. \tag{13.8}$$

Postmultiplication with $\begin{bmatrix} V_{12} \\ V_{22} \end{bmatrix}$ gives

$$\begin{bmatrix} X^TXK_a & X^TY \end{bmatrix} \begin{bmatrix} V_{12} \\ V_{22} \end{bmatrix} = US_2, \tag{13.9}$$

because V is orthogonal. Postmultiplying with $V_{22}^\dagger$ gives

$$\begin{bmatrix} X^TXK_a & X^TY \end{bmatrix} \begin{bmatrix} V_{12}V_{22}^\dagger \\ I_m \end{bmatrix} = US_2V_{22}^\dagger \tag{13.10}$$

where we have assumed that $V_{22}V_{22}^T \in \mathbb{R}^{m\times m}$ is non-singular. Comparing this with the normal equations shows that $-V_{12}V_{22}^\dagger$ is a solution and that $US_2V_{22}^\dagger$ is the residual.

Proof of (13.6). From the signal part we have that

$$US_1V_{21}^T = US_1V_{11}^Tp. \tag{13.11}$$

Solving for p in a LS optimal sense gives (13.6) provided $V_{11}V_{11}^T \in \mathbb{R}^{n\times n}$ is non-singular. hence, the n singular values in S_1 has to be non-zero. □

Hence, The n large singular values in S_1 are representing the signal. Similarly, the $r-n$ smaller singular values in S_2 are representing the residual. Furthermore, when $m=1$ and $a=n=r$, the algorithm results in the OLS solution. The algorithm can equivalently be computed as presented in the following proposition

Proposition 13.2
Given data matrices $X \in \mathbb{R}^{N\times r}$ and $Y \in \mathbb{R}^{N\times m}$. Furthermore, assume given the number of components a and the number n of singular values. Define the Krylov matrix $K_{a+1} \in \mathbb{R}^{r\times(a+1)m}$ of the pair (X^TX, X^TY) from Definition 13.1. Compute the SVD of K_{a+1}, i.e.

$$K_{a+1} = US\bar{V}^T, \tag{13.12}$$

where $U \in \mathbb{R}^{r\times r}$, $S \in \mathbb{R}^{r\times(a+1)m}$ and $V \in \mathbb{R}^{(a+1)m\times(a+1)m}$. Partition the matrix of left singular vectors as

$$\bar{V} = \begin{bmatrix} \bar{V}_{11} & \bar{V}_{12} \\ \bar{V}_{21} & \bar{V}_{22} \end{bmatrix}^T, \tag{13.13}$$

$\bar{V}_{11} \in \mathbb{R}^{m\times n}$, $\bar{V}_{21} \in \mathbb{R}^{am\times n}$, $\bar{V}_{12} \in \mathbb{R}^{m\times(a+1)m-n}$ and $\bar{V}_{22} \in \mathbb{R}^{am\times(a+1)m-n}$. The NTLS solution is then computed as

$$B_{\text{NTPLS}} = K_a p^\star, \tag{13.14}$$

where $p_\star \in \mathbb{R}^{am\times m}$ is defined as

$$p^\star = -\bar{V}_{22}\bar{V}_{12}^\dagger, \tag{13.15}$$

or equivalently

$$p^\star = (\bar{V}_{21}^T)^\dagger \bar{V}_{11}^T. \tag{13.16}$$

Furthermore, the residual $X^TY - X^TXB_{\text{NTPLS}}$ is determined by

$$\mathcal{E}_{\text{NTPLS}} = US_2\bar{V}_{12}^\dagger. \tag{13.17}$$

13.4 Truncated total least squares

PLS may give a bias on the parameter estimates in case of an errors-in-variables model, i.e. in the case when X is corrupted with measurements noise. Note also that OLS and PCR gives bias in this case. An interesting solution to the errors-in-variables problem is the Total Least Squares (TLS), Van Huffel and Vandevalle (1991), and the Truncated Total Least Squares (TTLS) solution, De Moor *et al* (1996), Fierro *et al* (1997) and Hansen (1992). The TTLS solution can be computed as $B_{\text{TTLS}} = -V_{12}V_{22}^\dagger$ where $V_{12} \in \mathbb{R}^{r\times r+m-a}$ and $V_{22} \in \mathbb{R}^{m\times r+m-a}$ are taken from the SVD of the compound matrix $\begin{bmatrix} X & Y \end{bmatrix} = USV^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}^T$. In MATLAB notation, $V_{12} := V(1:r, a+1:r+m)$ and $V_{22} := V(r+1:r+m, a+1:r+m)$. This is the solution to the problem of minimizing $\| \begin{bmatrix} X & Y \end{bmatrix} - \begin{bmatrix} X_{\text{TTLS}} & Y_{\text{TTLS}} \end{bmatrix} \|_F^2 = \| X - X_{\text{TTLS}} \|_F^2 + \| Y - Y_{\text{TTLS}} \|_F^2$ with respect to X_{TTLS} and Y_{TTLS} where $Y_{\text{TTLS}} = X_{\text{TTLS}}B_{\text{TTLS}}$ is the TTLS prediction.

13.5 CPLS: alternative formulation

The CPLS solution can be defined as in the following Lemma.

Lemma 13.1 (CPLS: Controllability PLS solution)
Given output data from a multivariate system, $Y = \begin{bmatrix} y_1 \; y_2 \; \ldots \; y_m \end{bmatrix} \in \mathbb{R}^{N\times m}$, and input (predictor) data, $X \in \mathbb{R}^{N\times r}$. The CPLS solution to the multivariate model $Y = XB + E$ where E is noise is given by

$$B_{CPLS} = \begin{bmatrix} K_a^1 p \; K_a^2 p \; \cdots \; K_a^m p \end{bmatrix} \;\in\; \mathbb{R}^{r\times m} \tag{13.18}$$

where $K_a^i \in \mathbb{R}^{r\times a}$ is the Krylov (reduced controllability) matrix for the pair (X^TX, X^Ty_i), i.e.,

$$K_a^i = \begin{bmatrix} X^Ty_i \; X^TXX^Ty_i \; (X^TX)^2X^Ty_i \; \cdots \; (X^TX)^{a-1}X^Ty_i \end{bmatrix}, \tag{13.19}$$

and the coefficient vector $p \in \mathbb{R}^a$ is defined from the single linear equation

$$(\sum_{i=1}^{m}(K_a^i)^TX^TXK_a^i)p = \sum_{i=1}^{m}(K_a^i)^TX^Ty_i. \tag{13.20}$$

The CPLS solution is defined as a function of a single coefficient vector $p \in \mathbb{R}^a$. the coefficient vector is found as the minimizing solution to the prediction error. In order to illustrate the difference between the CPLS solution and the common strategy of using PLS1 on each of the m outputs we give the following Lemma.

Lemma 13.2 (PLS1 used on multivariate data)
Given output data from a multivariate system, $Y = \begin{bmatrix} y_1 \; y_2 \; \ldots \; y_m \end{bmatrix} \in \mathbb{R}^{N\times m}$, and input (predictor) data, $X \in \mathbb{R}^{N\times r}$. The PLS1 solution to the multivariate model $Y = XB + E$ where E is noise is given by

$$B_{PLS1} = \begin{bmatrix} K_a^1 p_1 \; K_a^2 p_2 \; \cdots \; K_a^m p_m \end{bmatrix} \;\in\; \mathbb{R}^{r\times m} \tag{13.21}$$

where $K_a^i \in \mathbb{R}^{r\times a}$ is the Krylov (reduced controllability) matrix for the pair (X^TX, X^Ty_i) as in (13.19) and the coefficient vectors $p_i \in \mathbb{R}^a$ are defined from the m linear equations

$$((K_a^i)^TX^TXK_a^i)p_i = (K_a^i)^TX^Ty_i \;\forall\; i = 1, \cdots, m. \tag{13.22}$$

13.6 Efficient implementation of CPLS

The bi-linear, possibly multivariate, model $Y = XB + E$ has the univariate eqvivalent

$$y = \mathcal{X}b + e, \tag{13.23}$$

where

$$
\begin{aligned}
y &= \mathrm{vec}(Y) \in \mathbb{R}^{Nm}, && (13.24)\\
\mathcal{X} &= (I_m \otimes X) \in \mathbb{R}^{Nm\times rm}, && (13.25)\\
b &= \mathrm{vec}(B) \in \mathbb{R}^{rm}, && (13.26)\\
e &= \mathrm{vec}(E) \in \mathbb{R}^{Nm}. && (13.27)
\end{aligned}
$$

The univariate model (13.23) can be solved for $b = \mathrm{vec}(B)$ by, e.g. the PLS1 algorithm, and $B \in \mathbb{R}^{r\times m}$ can be reshaped from $b \in \mathbb{R}^{rm}$ afterhand. The PLS1 algorithm can be used directly. However, this strategy is time consuming due to the size of $\mathcal{X}$. It is a lot more efficient to utilize the structure in $\mathcal{X}$ in the computations. As we will show, this strategy is even faster than the fastest PLS2 algorithm, i.e. when $N > r$ the Kernal algorithm. One reason for this is that the CPLS method does not need eigenvalue or singular value problem computations, which is needed in PLS2, in order to compute the solution.

The normal equations for the model (13.23) are

$$
\mathcal{X}^T y = \mathcal{X}^T \mathcal{X} b. \qquad (13.28)
$$

The symmetric matrix $\mathcal{X}^T\mathcal{X} \in \mathbb{R}^{rm\times rm}$ and the vector $\mathcal{X}^T y \in \mathbb{R}^{rm}$ are of central importance in the PLS1 algorithm. Using that $x^T = (I_m \otimes X)^T = (I_m \otimes X^T)$ is a m-block diagonal matrix with X^T on the diagonal, (i.e., there are m blocks on the diagonal, each block is equal to X^T), we have that

$$
\mathcal{X}^T y = (I_m \otimes X^T) y = \overbrace{\begin{bmatrix} X^T & 0 & \cdots & 0 \\ 0 & X^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X^T \end{bmatrix}}^{(I_m \otimes X^T)} \overbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}^{y} = \begin{bmatrix} X^T y_1 \\ X^T y_2 \\ \vdots \\ X^T y_m \end{bmatrix}. \qquad (13.29)
$$

Note that the zeroes in (13.29) denotes $(r \times N)$ zero matrices. Hence, a lot of multiplications are saved by taking the structure of $\mathcal{X}$ into account when computing the correlation $\mathcal{X}^T y$. The term $\mathcal{X}^T y$ given by Equation (13.29) is a basis for the first PLS1 weighting vector. We can choose $w_1 = \mathcal{X}^T y$ directly as the first PLS1 weighting vector or we can choose a normalized version, e.g., $w_1 = \mathcal{X}^T y / \|\mathcal{X}^T y\|_F$.

The matrix product $\mathcal{X}^T\mathcal{X}$ is also a m-block diagonal matrix, but in this case with m diagonal blocks equal to $X^T X$, i.e.,

$$
\mathcal{X}^T\mathcal{X} = (I_m \otimes X^T)(I_m \otimes X) = (I_m \otimes X^T X) = \begin{bmatrix} X^T X & 0 & \cdots & 0 \\ 0 & X^T X & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X^T X \end{bmatrix} (13.30)
$$

In order to compute the next weighting vector we have to compute $\mathcal{X}^T\mathcal{X} w_1$. In general, we have to compute terms of the type, $\mathcal{X}^T\mathcal{X} w_i$ or $\mathcal{X}^T\mathcal{X} d_i$ in the CGM, during each iteration. We have that

$$\mathcal{X}^T\mathcal{X}d_i = (I_m \otimes X^T X)d_i = \overbrace{\begin{bmatrix} X^T X & 0 & \cdots & 0 \\ 0 & X^T X & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X^T X \end{bmatrix}}^{\mathcal{X}^T\mathcal{X}=(I_m\otimes X^T X)} \overbrace{\begin{bmatrix} d_{i1} \\ d_{i2} \\ \vdots \\ d_{im} \end{bmatrix}}^{d_i} = \begin{bmatrix} X^T X d_{i1} \\ X^T X d_{i2} \\ \vdots \\ X^T X d_{im} \end{bmatrix} \tag{13.31}$$

where the *search direction* vector is

$$d_i = \begin{bmatrix} d_{i1} \\ d_{i2} \\ \vdots \\ d_{im} \end{bmatrix} \in \mathbb{R}^{rm}, \tag{13.32}$$

and where $d_{ij} \in \mathbb{R}^r$.

The PLS1 algorithm can now be effectively modified by using (13.29) and (13.31).

13.7 Conjugate gradient method of optimization and PLS1

The equivalence between the PLS1 algorithm and the Conjugate Gradient Method (CGM) by Hestenes and Stiefel (1952) used for calculating generalized inverses is presented in Wold *et al* (1984). Each step in the conjugate gradient method gives the corresponding PLS1 solution. However, the strong connection between the PLS1 algorithm and the CGM can be clarified further. The conjugate gradient method by Hestenes and Stiefel (1952) is also known as an important iterative method for solving univariate least squares problems, solving linear equations, as well as an iterative optimization method for finding the minimum of a quadratic and non-quadratic functions.

We will show that the the method of Hestenes and Stiefel (1952) and further analyzed, among others, in Golub (1983), (1996) for iterativeley solving univariate LS problems gives identical results as the PLS1 algorithm. Furthermore, we will show that the conjugate gradient optimization method presented and analyzed in Luenberger (1968) §10.8 for solving quadratic minimization problems is identical to the PLS1 algorithm for solving the possibly rank deficient LS problem. See also Luenberger (1984) §8.3 and 8.4.

By iterative, we mean as in Luenberger (1984), that the algorithm generates a series of points, each point being calculated on the basis of the points preceding it. Ideally, the sequence of points generated by the iterative algorithm in this way converges in a finite or infinite number of steps to a solution of the original problem. The conjugate gradient method can be regarded as being somewhat intermediate between the *Stepest Descent Method* (SDM) and Newtons' method. Newtons method converges in one step for quadratic problems if the solution exists, and this solution is identical to the OLS solution to the corresponding normal equations. The CGM converges to the OLS solution in at most r, where r is the number of regressor variables.

The SDM converges to the OLS solution in a finite number of steps, but the rate of convergence may be slow. For ill-posed and ill-conditioned problems where the OLS solution is unreliable or does not exist, then, a number of iterations, $a < r$, of the CGM can be evaluated in order to obtain a regularized and stabilized solution. Hence, an optimization and LS algorithm may be iterative even if it converges in a finite number of steps. The definition of iterative does not depend on the number of iterations.

Proposition 13.3 (LS and quadratic minimization problems)
The LS problem

$$B^* = \arg\min_{B \in \mathcal{B}} V(B), \tag{13.33}$$

where $\mathcal{B}$ is the parameter space and where

$$V(B) = \frac{1}{2}\|Y - XB\|_F^2, \tag{13.34}$$

is similar to the quadratic minimization problem

$$B^* = \arg\min_{B \in \mathcal{B}} V_Q(B), \tag{13.35}$$

where

$$V_Q(B) = \frac{1}{2}B^T X^T XB - B^T X^T Y. \tag{13.36}$$

Furthermore, with the definitions $Q = X^T X$ and $b = X^T Y$ we have the normal equations

$$QB = b. \tag{13.37}$$

Proof 13.2 *In LS problems the usual criterion to be minimized can be written as*

$$V = \frac{1}{2}\|Y - XB\|_F^2 = \frac{1}{2}(Y - XB)^T(Y - XB) = \frac{1}{2}Y^T Y + \frac{1}{2}B^T X^T XB - B^T X^T Y. \tag{13.38}$$

The 1st term on the right hand side is independent of the solution. Hence, a linear LS problem is equivalent to a quadratic minimization problem, i.e., minimizing $\|Y - XB\|_F^2$ with respect to B is equivalent to minimizing $B^T X^T XB - 2B^T X^T Y$ with respect to B. $\square$.

Note that the parameter space which gives the PLS1 solution coincide with the column (range) space of the Krylov matrix, i.e., $\mathcal{B} \in R(K_a)$. A standard recursion for the solution B^* which is used in minimization methods, e.g., as in Newtons method, the stepest descent (gradient direction) method and in conjugate direction methods, is defined as follows

Definition 13.2 (Recursive equation for the solution)
Define k as the iteration index. Define the solution update equation as

$$B_{k+1} = B_k + \alpha_k d_k, \tag{13.39}$$

where $d_k \in \mathbb{R}^r$ is the search direction and α_k is a line search scalar parameter.

In the stepest descent algorithm the search direction vector d_k at a given point B_k is taken to be the negative of the gradient of V_Q at B_k. In order to speed up the convergence of the stepest descent algorithm, and avoiding evaluation and inversion of the Hessian matrix as in Newtons method, the use of *conjugate direction methods* can be motivated. A version of this is the conjugate gradient method (CGM). The CGM and the connection with PLS is developed in the following. However, in order to proceede properly the gradient is first defined and presented in the following proposition with proof.

Proposition 13.4 (The gradient)
The gradient of V_Q at the point B_k is defined by

$$g_k = QB_k - b. \tag{13.40}$$

An alternative recursion scheme for the gradient is

$$g_{k+1} = g_k + \alpha_k Q d_k. \tag{13.41}$$

Proof 13.3 *The gradient of V_Q with respect to B_k is given by*

$$\frac{\partial V_Q}{\partial B_k} = QB_k - b. \tag{13.42}$$

Hence, we have definition (13.40).

Evaluate the expression $g_{k+1} - g_k$ and using $B_{k+1} = B_k + \alpha_k d_k$, i.e.,

$$B_{k+1} - B_k = QB_{k+1} - QB_k = Q(B_k + \alpha_k d_k) - QB_k = \alpha_k Q d_k. \tag{13.43}$$

This gives the alternative gradient update (13.41). □

Note that the residual of the normal equations for a solution B_k is identical to the negative gradient. This is clarified with the following definition.

Definition 13.3 (Residual of the normal equations)
The residual of the normal equations evaluated with a point B_k is defined as

$$w_k \overset{\text{def}}{=} b - QB_k = -g_k. \tag{13.44}$$

Hence, in the SDM the search for a minimum is done in the direction of the residual of the normal equations. The scalar line search parameter in the solution update Equation (13.39), will now defined in the following proposition.

Proposition 13.5 (Line search parameter)
The line search scalar parameter α_k in (13.39) is given by

$$\alpha_k = -\frac{d_k^T g_k}{d_k^T Q d_k} \tag{13.45}$$

when $d_k^T Q d_k > 0$.

Proof 13.4 *The parameter α_k is taken as the value which analytically is minimizing the quadratic function $V_Q(B_{k+1})$ in the search direction, d_k. Hence,*

$$\begin{aligned}\frac{\partial V_Q(B_{k+1})}{\partial \alpha_k} &= \frac{\partial}{\partial \alpha_k}[\frac{1}{2}(B_k+\alpha_k d_k)^T Q(B_k+\alpha_k d_k) - (B_k+\alpha_k d_k)^T b] \\ &= d_k^T Q(B_k+\alpha_k d_k) - d_k^T b = d_k^T \overbrace{(QB_k - b)}^{g_k} + \alpha_k d_k^T Q d_k. \quad (13.46)\end{aligned}$$

Solving $\frac{\partial V_Q(B_{k+1})}{\partial \alpha_k} = d_k^T g_k + \alpha_k d_k^T Q d_k = 0$ for α_k gives (13.45). Furthermore, the minimum is unique when $\frac{\partial^2 V_Q(B_{k+1})}{\partial \alpha_k^2} = d_k^T Q d_k > 0$. □

We will now discuss and present the search direction which is used in the CGM. The first search direction in the CGM is equal to the negative gradient direction, i.e., $d_1 = -g_1$. Hence, for one component (or one iteration) the CGM is identical to the SDM. A new point B_2 is then evaluated from the solution update equation. The next search direction is choosen as a linear combination af the new negative gradient, $-g_2$, and the old direction d_1. Hence, $d_2 = -g_2 + \beta_1 d_1$ where β_1 is a scalar parameter. The parameter β_1 is choosen such that d_2 is Q-orthogonal (or conjugate) to d_1, i.e., so that $d_1^T Q d_2 = d_2^T Q d_1 = 0$. Hence, d_2 is in the space spanned by $-g_2$ and $d_1 = -g_1$ but Q-orthogonal to d_1. In general, the search direction , d_{k+1}, in the CGM is a simple linear combination of the current residual, $w_{k+1} = -g_{k+1}$ and its predecessor d_k.

Proposition 13.6 (CGM search direction)
The search direction in CGM is initialized with $d_1 = -g_1 = b = X^T Y$. The CGM search direction update equation is given by

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \quad (13.47)$$

where

$$\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}. \quad (13.48)$$

Proof 13.5 *The search directions d_{k+1} and d_k are Q-orthogonal, or conjugate with respect to Q, if $d_{k+1}^T Q d_k = 0$. Hence,*

$$d_{k+1}^T Q d_k = -g_{k+1}^T Q d_k + \beta_k d_k^T Q d_k = 0. \quad (13.49)$$

Solving this for β_k gives (13.48). □

Remark 13.1 *There are two other expressions for α_k and β_k which may be usefull when implementing the CGM. The alternatives are (Luenberger (1984) p. 245)*

$$\alpha_k = \frac{g_k^T g_k}{d_k^T Q d_k}, \quad (13.50)$$

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}. \quad (13.51)$$

Hence, a number of multiplications can be saved by using the alternatives.

Based on the theory developed so far we state the following implementation of the CGM.

Algorithm 13.7.1 (PLS1: Conjugate gradient implementation)
Given $X \in \mathbb{R}^{N \times r}$ and $Y \in \mathbb{R}^N$ and possibly an initial guess $B_0 \in \mathbb{R}^r$ for the solution. Note that $B_0 = 0$ gives PLS1. We have

```
$b = X^T Y$                                              Vector and right hand side matrix
$Q = X^T X$                                              in the normal equations, $X^T Y = X^T X B$.
$g_1 = Q B_0 - b$                                        The gradient (1st weight vector)
$d_1 = -g_1$                                             The 1st search direction
for    $k = 1, a$                                        Loop for all components $1 \le a \le r$.
       $G_k$      $=$    $\begin{bmatrix} g_1 & \cdots & g_k \end{bmatrix}$   Store present gradient vector.
       $D_k$      $=$    $\begin{bmatrix} d_1 & \cdots & d_k \end{bmatrix}$   Store present direction vector.
       $\alpha_k$ $=$    $-\frac{g_k^T d_k}{d_k^T Q d_k}$      Coeff. for bidiagonal matrix.
       $B_k$      $:=$   $B_{k-1} + \alpha_k d_k$            The PLS1 solution at step $k$.
       if     $k < a$
       $g_{k+1}$  $=$    $g_k + \alpha_k Q d_k$              The gradient, (weighting vector).
       $\beta_k$  $=$    $\frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$   Coeff. for bidiagonal matrix.
       $d_{k+1}$  $=$    $-g_{k+1} + \beta_k d_k$            Update the search direction.
       end
end
```

This results, after a iterations, in the LS solution B_a, in the gradient or weighting matrix $G \in \mathbb{R}^{r \times a}$, and a direction matrix $D \in \mathbb{R}^{r \times a}$. The matrices are defined as follows

$$G_a = \begin{bmatrix} g_1 & \cdots & g_a \end{bmatrix}, D_a = \begin{bmatrix} d_1 & \cdots & d_a \end{bmatrix}. \qquad (13.52)$$

△

Note that we have used the alternative solution update equation $B_k = B_{k-1} + \alpha_k d_k$ instead of (13.39) in order for the actual implementation in Algorithm (13.7.1 to be more readable. The first search direction, d_1, in the algorithm is equal to the negative gradient, $-g_1 = X^T Y$. Hence, the first step is a stepest descent step, and the first weighting vector, w_1, in the PLS1 algorithm is identical to the first search direction, i.e., $w_1 = d_1 = -g_1$. This shows that the choice $w_1 = X^T Y$ is higely motivated. This is in contrast to the statement in Helland (1988) p. 588. The algorithm may be implemented with a test on the gradient which is equal to the negative residual, i.e., the algorithm should be terminated when $\|g_k\| = \|w_k\|$ is less than a tolerance for zero.

Theorem 13.7.1 (Equivalence between PLS1 and CGM)
From Algorithm 13.7.1 we obtain

$$B_k \in B_0 + R(K_k(X^T X, d_1)) \qquad (13.53)$$

$$D_k \in G_k \in R(K_k(X^T X, d_1)). \qquad (13.54)$$

where $d_1 = b - QB_0 = X^TY - X^TXB_0$.

Furthermore, if the initial guess for the solution is zero, i.e., $B_0 = 0$, *then the* k*-component PLS1 solution is given by*

$$B_{\mathrm{PLS}} = B_k \;\in\; R(K_k(X^TX, X^TY)). \tag{13.55}$$

Proof 13.6 *The fact that the CGM solution,* B_k, *is the minimizing LS solution in the subspace spanned by* $\mathcal{B} = B_0 + R(K_k(X^TX, d_1))$ *is prooved in Luenberger (1984) §8.2 and §8.4.. Se also Golub and Van Loan (1984) pp. 366-367 and Golub and Van Loan (1996) §10.2.*

The fact that $B_{\mathrm{PLS}} \in R(K_k(X^TX, X^TY))$ *is prooved in Helland (1988), Di Ruscio (1998).*

Since both the PLS1 solution and the CGM solution (with $B_0 = 0$*) is the minimizing LS solution of the same subspace, i.e. the column (range) space of the Krylov matrix* $K_k(X^TX, X^TY)$. $\square$

Based on the above theory we state the following new results concerning the PLS1 solution.

Theorem 13.7.2 (PLS1: alternative expression for the solution)
The PLS1 solution for a *components is identical to the CGM solution obtained after* a *iterations. The solution can be expressed as*

$$B_{\mathrm{PLS}} = D_a(D_a^TX^TXD_a)^{-1}D_a^TX^TY, \tag{13.56}$$

where it is important to recognize that $D_a^TX^TXD_a \in \mathbb{R}^{a\times a}$ *is diagonal. Furthermore,*

$$B_{\mathrm{PLS}} = G_a(G_a^TX^TXG_a)^{-1}G_a^TX^TY, \tag{13.57}$$

where we have that $G_a^TX^TXG_a \in \mathbb{R}^{a\times a}$ *is tri-diagonal.*

Proof 13.7 *A parameterized candidate solution which belong to the Krylov subspace is*

$$B_a = D_ap. \tag{13.58}$$

Furthermore, the LS solution

$$p^* = Y - X\overbrace{D_ap}^{B_a}{}_F^2 \tag{13.59}$$

gives

$$p^* = (D_a^TX^TXD_a)^{-1}D_a^TX^TY. \tag{13.60}$$

Using p^* *instead of* p *in the parametreized solution gives the PLS solution (13.56).*

The fact that $D_a^TQD_a$ *follows from the Q-orthogonal (or Q-conjugate) properties of the CGM.*

13.8 References

Luenberger, D. G. (1968). Optimization by vector space methods. John Wiley and Sons, Inc.

.1 1st effective implementation of CPLS

```
% Effektiv implementering av CPLS for m=3 utganger og a=2 komponenter.
% Benytter PLS1 algoritme i Automatica paper pp. 6-7.
% Given a multivariate system
% Y = XB+E
% This is equivalent to the univariate system
% Y(:)=kron(I_m,X)*B(:).

m=3;
[N,r]=size(X);

y1=Y(:,1); y2=Y(:,2); y3=Y(:,3);                  % Y=[y1 y2 y3]

w11=X'*y1; w12= X'*y2; w13=X'*y3;
w1=[w11;w12;w13];                                  % 1st weight vector

xx=X'*X;
xxw1=[xx*w11;xx*w12;xx*w13];                       % x'*x*w1 where x=kron(I_m,X)

w2=w1-xxw1*inv(w1'*xxw1)*w1'*w1;                   % 2nd weight vector

w21=w2(1:r); w22=w2(r+1:2*3); w23=w2(2*r+1:3*r);
xxw2=[xx*w21;xx*w22;xx*w23];

W=[w1 w2];


B=W*inv(W'*[xxw1 xxw2])*W'*w1
B=reshape(B,r,m)
```

.2 2nd effective implementation of CPLS

```
function [B,W,Xxw]=cpls_i1(X,Y,a);
% CPLS_I1
% B=cpls_i1(X,Y,a);
%
% Effektiv implementering av CPLS for m=3 utganger og a=2 komponenter.
% Benytter PLS1 algoritme i Automatica paper pp. 6-7.
% Given a multivariate system
% Y = XB+E
% This is equivalent to the univariate system
% Y(:)=kron(I_m,X)*B(:).

[Nx,r]=size(X); [Ny,m]=size(Y);
xx=X'*X;

w=[]; W=[]; xxw=[];
for i=1:m
  wij=X'*Y(:,i);
  w=[w;wij];                                % 1st weight vector
  xxw=[xxw;xx*wij];                         % x'*x*w1 where x=kron(I_m,X)
end

Xxw=[];
for i=1:a
 Xxw=[Xxw xxw];
 W  =[W w];
 if i < a
  w=w-Xxw*inv(W'*Xxw)*W'*w;                 % 2nd weight vector

  xxw=[];
  for j=1:m
    wij=w((j-1)*r+1:j*r);
    xxw=[xxw;xx*wij];
  end
 end
end

B=W*inv(W'*Xxw)*W'*W(:,1);
B=reshape(B,r,m);
```

.3 3nd effective implementation of CPLS

```
function [B,W,Xxw]=cpls_i2(X,Y,a);
% CPLS_I2, Same as CPLS_I1 but with orthonormal weighting vectors
% B=cpls_i1(X,Y,a);
%
% Effektiv implementering av CPLS for m=3 utganger og a=2 komponenter.
% Benytter PLS1 algoritme i Automatica paper pp. 6-7.
% Given a multivariate system
% Y = XB+E
% This is equivalent to the univariate system
% Y(:)=kron(I_m,X)*B(:).

[Nx,r]=size(X); [Ny,m]=size(Y);
xx=X'*X;

w=[]; W=[]; xxw=[];
for i=1:m
  wij=X'*Y(:,i);
  w=[w;wij];                              % 1st weight vector
end
xy=w;
w=w/norm(w);
for j=1:m
  wij=w((j-1)*r+1:j*r);
  xxw=[xxw;xx*wij];                       % x'*x*w1 where x=kron(I_m,X)
end

Xxw=[];
for i=1:a
 Xxw=[Xxw xxw];
 W  =[W w];
 if i < a
  w=w-Xxw*inv(W'*Xxw)*W'*w;               % 2nd weight vector
  w=w/norm(w);
  xxw=[];
  for j=1:m
    wij=w((j-1)*r+1:j*r);
    xxw=[xxw;xx*wij];
  end
 end
end

B=W*inv(W'*Xxw)*W'*xy;
B=reshape(B,r,m);
```

.4 4th effective implementation of CPLS

```
function [B,W,Xxw]=cpls_i3(X,Y,a);
% CPLS_I3, Same as CPLS_I1 but with orthonormal weighting vectors
%
% B=cpls_i1(X,Y,a);
%
% Effektiv implementering av CPLS for m=3 utganger og a=2 komponenter.
% Benytter PLS1 algoritme i Automatica paper pp. 6-7.
% Given a multivariate system
% Y = XB+E
% This is equivalent to the univariate system
% Y(:)=kron(I_m,X)*B(:).

[Nx,r]=size(X); [Ny,m]=size(Y);
xx=X'*X;

w=[]; W=[]; xxw=[];
for i=1:m
  wij=X'*Y(:,i);
  w=[w;wij];                                % 1st weight vector
end
xy=w;
w=w/norm(w);
for j=1:m
  wij=w((j-1)*r+1:j*r);
  xxw=[xxw;xx*wij];                         % x'*x*w1 where x=kron(I_m,X)
end

Xxw=[];
for i=1:a
 z(i,i)=w'*xxw;                        % diagonal of W'*xx*W, alpha's
 if i>1;                               % subdiagonal of W'*xx*W, beta's
  z(i,i-1)=w'*Xxw(:,i-1);
  z(i-1,i)=z(i,i-1);
 end

 Xxw=[Xxw xxw];
 W  =[W w];
 if i < a
%  w=w-Xxw*inv(W'*Xxw)*W'*w;                 % ith weight vector
  w=w-Xxw*inv(z)*flipud(eye(i,1));           % ith weight vector

  w=w/norm(w);
  xxw=[];
  for j=1:m
    wij=w((j-1)*r+1:j*r);
    xxw=[xxw;xx*wij];
  end
 end
end

%B=W*inv(W'*Xxw)*W'*xy;
%B=W*inv(z)*W'*xy;
B=W*inv(z)*[W(:,1)'*xy;zeros(a-1,1)];

B=reshape(B,r,m);
```

.5 Conjugate Gradient Algorithm: Luenberger (1984) page 244

```
function [x,G,D]=cgrad(X,Y,a);
% CGRAD
% B=CGRAD(X,Y,A);
% PURPOSE
% Implementation of the Conjugate Gradient algorithm in
% Luenberger (1984) page 244 to solve LS problems.
% The algorithm is identical to the PLS1 algorithm

b=X'*Y; Q=X'*X;
r=length(b);

x=zeros(r,1);                         % x_0 gives PLS

d=b - Q*x;                            % d_0
g=-d;                                 % g_0

G=zeros(r,a); D=G;
for k=1:a
  G(:,k)=g; D(:,k)=d;
  alpha=-g'*d/(d'*Q*d);               % alpha_k

  x    = x + alpha*d;                 % x_(k+1)
  if k<a
   g    =Q*x - b                      % g_(k+1)
   beta =g'*Q*d/(d'*Q*d)              % beta_k
   d    =-g+beta*d                    % d_k+1
  end
end
```

.6 Conjugate Gradient Algorithm II: Luenberger (1984) page 244

```
function [x,G,D,Beta]=cgrad2(X,Y,a,x);
% CGRAD
% [B,G,D,Beta]=CGRAD(X,Y,A);
% PURPOSE
% Implementation of the Conjugate Gradient algorithm in
% Luenberger (1984) page 244 to solve LS problems.
% The algorithm is identical to the PLS1 algorithm.
% Faster implementation than cgrad.m.
% NOTE
% - Normalized G is identical to W_pls1.
% - x=D*inv(D'*X'*X*D)*D'*X'*Y where D'*X'*X*D is diagonal
% - x=G*inv(G'*X'*X*G)*G'*X'*Y where G'*X'*X*G is tri-diagonal

b=X'*Y; Q=X'*X;
r=length(b);

if nargin==3
 x=zeros(r,1);                       % x_0 gives PLS
 g=-b;
else
 g=Q*x-b;                            % d_0
end
d=-g;                                % 1st search direction=stepest descent

G=zeros(r,a); D=G; Beta=zeros(a-1,1);
for k=1:a
  D(:,k)=d; G(:,k)=g/norm(g);

  work1=Q*d;                        %
  work2=d'*work1;
  alpha=-g'*d/work2;                % alpha_k

  x    = x + alpha*d;               % x_(k+1)
  if k<a
%   g    =Q*x - b;                   % g_(k+1)
   g    =g+alpha*work1;             % g_(k+1)
   beta =g'*work1/work2;            % beta_k
   d    =-g+beta*d;                 % d_k+1
   Beta(k)=beta;
  end
end
```

.7 Conjugate Gradient Algorithm II: Golub (1984) page 370

```
function [B,R,P]=galg10213(x,y,a);
% Conjugate gradient algorithm for PLS1.
% Nearly the same as Algorithm 10.2.13 in
% Golub and Van Loan (1996).

b=x'*y;
A=x'*x;
[nx,ny]=size(b);

B=zeros(nx,1); R=zeros(nx,a); P=zeros(nx,a);
r=b;
for k=1:a
   rr=r'*r;
   if k==1
    beta=0; p=r;
   else
    beta=rr/(r_old'*r_old);
    p=r+beta*p
   end
   R(:,k)=r; P(:,k)=p;
   z=A*p;
   alpha=rr/(p'*z)
   B=B+alpha*p
   r_old=r;
   r=r-alpha*z
end
```

.8 CGM implementation of CPLS

```
function [x,G,D]=cg_cpls(X,Y,a);
% CG_CPLS
% [B,G,D]=CG_CPLS(X,Y,A);
% PURPOSE
% Compute CPLS solution by the conjugate gradient method.
% Implementation of the Conjugate Gradient algorithm in
% Luenberger (1984) page 244 to solve multivariate LS problems.

[Nx,r]=size(X); [Ny,m]=size(Y);

Q=X'*X;

x=zeros(r*m,1);                      % Initial guess is zero.

d=[];                                % d_1=chi'*Y(:)=kron(I,X')*Y(:)
for i=1:m                            % The first direction.
  di=X'*Y(:,i);
  d =[d;di];
end
g=-d;                                % Stepest descent 1st step.

for k=1:a

  work1=[];                          % Qt d_k, Qt=kron(I,X'*X)=chi'*chi.
  for i=1:m
    work1=[work1;Q*d((i-1)*r+1:i*r)];
  end

  work2=d'*work1;                    % d_k^T*Qt*d_k
  alpha=-g'*d/work2;                 % alpha_k

  x    = x + alpha*d;                % x_(k+1)
  if k<a
   g    =g+alpha*work1;              % g_(k+1)
   beta =g'*work1/work2;             % beta_k
   d    =-g+beta*d;                  % d_k+1
  end
end
x=reshape(x,r,m);
G=g; D=d;
```

Appendix A

Linear Algebra and Matrix Calculus

A.1 Trace of a matrix

The trace of a $n \times m$ matrix A is defined as the sum of the diagonal elements of the matrix, i.e.

$$tr(A) = \sum_{i=1}^{\min(n,m)} a_{ii} \tag{A.1}$$

We have the following trace operations on two matrices A and B of apropriate dimensions

$$tr(A^T) = tr(A) \tag{A.2}$$

$$tr(AB^T) = tr(A^T B) = tr(B^T A) = tr(BA^T) \tag{A.3}$$

$$tr(AB) = tr(BA) = tr(B^T A^T) = tr(A^T B^T) \tag{A.4}$$

$$tr(A \pm B) = tr(A) \pm tr(B) \tag{A.5}$$

The eigenvalues, $\lambda_1, \dots, \lambda_n$ of a square symmetric matrix, $A \in \mathbb{R}^{n \times n}$, are all real. The trace of A is then equal to the sum of the eigenvalues

$$tr(A) = \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i. \tag{A.6}$$

A.2 Gradient matrices

$$\frac{\partial}{\partial X}\ tr[X] = I \tag{A.7}$$

$$\frac{\partial}{\partial X}\ tr[AX] = A^T \tag{A.8}$$

$$\frac{\partial}{\partial X}\ tr[AX^T] = A \tag{A.9}$$

$$\frac{\partial}{\partial X}\ tr[AXB] = A^TB^T \tag{A.10}$$
$$\frac{\partial}{\partial X}\ tr[AX^TB] = BA \tag{A.11}$$
$$\frac{\partial}{\partial X}\ tr[XX] = 2X^T \tag{A.12}$$
$$\frac{\partial}{\partial X}\ tr[XX^T] = 2X \tag{A.13}$$
$$\frac{\partial}{\partial X}\ tr[X^n] = n(X^{n-1})^T \tag{A.14}$$
$$\frac{\partial}{\partial X}\ tr[AXBX] = A^TX^TB^T + B^TX^TA^T \tag{A.15}$$
$$\frac{\partial}{\partial X}\ tr[e^{AXB}] = (Be^{AXB}A)^T \tag{A.16}$$

$$\frac{\partial}{\partial X} tr[XAX^T] = 2XA,\ \text{ if } A = A^T \tag{A.17}$$

$$\frac{\partial}{\partial X^T}\ tr[AX] = A \tag{A.18}$$
$$\frac{\partial}{\partial X^T}\ tr[AX^T] = A^T \tag{A.19}$$
$$\frac{\partial}{\partial X^T}\ tr[AXB] = BA \tag{A.20}$$
$$\frac{\partial}{\partial X^T}\ tr[AX^TB] = A^TB^T \tag{A.21}$$
$$\frac{\partial}{\partial X^T}\ tr[e^{AXB}] = Be^{AXB}A \tag{A.22}$$

A.3 Derivatives of vector and quadratic form

The derivative of a vector with respect to a vector is a matrix. We have the following identities:

$$\frac{\partial x}{\partial x^T} = I \tag{A.23}$$
$$\frac{\partial}{\partial x}\ (x^TQ) = Q \tag{A.24}$$
$$\frac{\partial}{\partial x}\ (Qx) = Q^T \tag{A.25}$$

The derivative of a scalar with respect to a vector is a vector. We have the following identities:

$$\frac{\partial}{\partial x}\ (y^Tx) = y \tag{A.26}$$
$$\frac{\partial}{\partial x}\ (x^Tx) = 2x \tag{A.27}$$
$$\frac{\partial}{\partial x}\ (x^TQx) = Qx + Q^Tx \tag{A.28}$$
$$\frac{\partial}{\partial x}\ (y^TQx) = Q^Ty \tag{A.29}$$

Note that if Q is symmetric then

$$\frac{\partial}{\partial x}\ (x^TQx) = Qx + Q^Tx = 2Qx. \tag{A.30}$$

A.4 More on vector derivatives

We will illustrate the kernel rule for the derivative of a scalar function $f(u(x))$ with respect to a vector x. Consider a scalar function

$$f(u(x)) = u^T Qu, \tag{A.31}$$

where $u = u(x)$ is a vector

$$u = r - Dx. \tag{A.32}$$

Then,

$$\frac{\partial f}{\partial x} = \frac{\partial u}{\partial x}\frac{\partial f}{\partial u} = (-D)^T 2Qu = -2D^T Q(r - Dx), \tag{A.33}$$

where we have used that

$$\frac{\partial u}{\partial x} = -D^T, \tag{A.34}$$

and

$$\frac{\partial f}{\partial u} = 2Qu. \tag{A.35}$$

A.5 Matrix norms

The trace of the matrix product $A^T A$ is related to the Frobenius norm of A as follows

$$\|A\|_F^2 = \mathrm{tr}(A^T A), \tag{A.36}$$

where $A \in \mathbb{R}^{N\times m}$.

A.6 Linearization

Given a vector function $f(x) \in \mathbb{R}^m$ where $x \in \mathbb{R}^n$. The derivative of the vector f with respect to the row vector x^T is defined as

$$\frac{\partial f}{\partial x^T} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m\times n} \tag{A.37}$$

Given a non-linear differentiable state space model

$$\dot{x} = f(x, u), \tag{A.38}$$

$$y = g(x). \tag{A.39}$$

A linearized model around the stationary points x_0 and u_0 is

$$\dot{\delta x} = A\delta x + B\delta u, \tag{A.40}$$

$$\delta y = D\delta x, \tag{A.41}$$

where

$$A = \frac{\partial f}{\partial x^T}\,|_{x_0,u_0}, \tag{A.42}$$

$$B = \frac{\partial f}{\partial u^T}\,|_{x_0,u_0}, \tag{A.43}$$

$$D = \frac{\partial g}{\partial x^T}\,|_{x_0,u_0}, \tag{A.44}$$

and where

$$\delta x = x - x_0, \tag{A.45}$$

$$\delta u = u - u_0 \tag{A.46}$$

$$\delta y = y - y_0 \tag{A.47}$$

$$y_0 = g(x_0). \tag{A.48}$$

This is deduced as follows. A Taylor series expansion of $f(x,u)$ around the stationary points x_0 and u_0 gives

$$f(x,u) \approx f(x_0,u_0) + \frac{\partial f}{\partial x^T}\,|_{x_0,u_0}\,(x - x_0) + \frac{\partial f}{\partial u^T}\,|_{x_0,u_0}\,(u - u_0). \tag{A.49}$$

Since x_0 and u_0 are stationary points such that $\dot{x}_0 = f(x_0,u_0) = 0$. This gives also that $\dot{\delta x} = \dot{x} - \dot{x}_0 = \dot{x}$. Using this and inserting (A.49) into (A.38) gives (A.40). A Taylor series expansion of $g(x)$ around x_0 gives

$$g(x) \approx g(x_0) + \frac{\partial g}{\partial x^T}\,|_{x_0}\,(x - x_0). \tag{A.50}$$

Inserting (A.50) into (A.39) and defining $y_0 = g(x_0)$ and $\delta y = y - y_0$ gives (A.41).

Note also that if the points x_0, u_0 are not stationary points then the linearized state equation is given by

$$\dot{\delta x} = A\delta x + B\delta u + f(x_0,u_0) - \dot{x}_0. \tag{A.51}$$

A.7 Kronecer product matrices

Given a matrix $X \in \mathbb{R}^{N\times r}$. Let I_m be the $(m\times m)$ identity matrix. Then

$$(X \otimes I_m)^T = X^T \otimes I_m \ \in \ \mathbb{R}^{rm\times Nm}, \tag{A.52}$$

$$(I_m \otimes X)^T = I_m \otimes X^T \ \in \ \mathbb{R}^{rm\times Nm}. \tag{A.53}$$

Given a matrix $A \in \mathbb{R}^{n\times m}$. The vec$(\cdot)$ operator is defined as

$$\text{vec}(A) = \text{vec}(\begin{bmatrix} a_1 & a_2 & \dots & a_m \end{bmatrix}) = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \in \mathbb{R}^{nm}, \tag{A.54}$$

where $a_i \ \forall \ i = 1, \dots, m$ is column number i in A. I.e. vec(A) is an nm dimensional vector formed by stacking the columns of the matrix A on each other.

Given three matrices $A \in \mathbb{R}^{n\times m}$, $X \in \mathbb{R}^{m\times p}$ and $B \in \mathbb{R}^{p\times k}$, i.e, of appropriate dimensions. The vec$(\cdot)$ operation on the product AXB is given by

$$\text{vec}(AXB) = (B^T \otimes A)\text{vec}(X) \ \in \ \mathbb{R}^{kn}. \tag{A.55}$$

Note also that

$$(B^T \otimes A) \ \in \ \mathbb{R}^{kn\times pm}, \tag{A.56}$$

and

$$\text{vec}(X) \ \in \ \mathbb{R}^{pm}. \tag{A.57}$$

A.8 The matrix exponential

By definition, the matrix exponential of At where $A \in \mathbb{R}^{n\times n}$ and t is a scalar is determined as the infinite power (Taylor) series

$$e^{At} = I_n + At + \frac{1}{2}A^2t^2 + \frac{1}{3!}A^3t^3 + \dots = I_n + \sum_{k=1}^{\infty} \frac{1}{k!}A^k t^k \tag{A.58}$$

This definition should usually not be used for numerical computation of the matrix exponential due to rounding-off errors in computing the matrix powers. See Moler and van Loan (1978) for effective methods in computing the matrix exponential. An numerically stable and reliable algorithm for the computation of e^A is via the real Schur decomposition of A. The real Schur decomposition of A is given by

$$A = UTU^T, \tag{A.59}$$

where T is a block upper triangular matrix with the eigenvalues of A in one-by-one or two-by-two blocks on the diagonal of T. U is a matrix with Schur eigenvectors. We then have that

$$e^A = Ue^TU^T. \tag{A.60}$$

The problem is now reduced to the problem to compute the matrix exponential, $F = e^T$, of a block upper triangular matrix, T. Since T is block upper triangular, then also $F = e^T$ is block upper triangular. In case of real eigenvalues (one-by-one blocks on the diagonal) we have that the diagonal elements of F is given by

$$f_{ii} = e^{t_{ii}} \ \forall \ i = 1, \dots, n. \tag{A.61}$$

The other elements in the upper triangular part of F can be found from the fact that the matrices F and T commutes, i.e. from the property that,

$$FT = TF. \tag{A.62}$$

Equating the coefficients in (A.62) and using (A.61) gives a formula for the off-diagonal elements of F. In case of distinct eigenvalues of A we get the formula

$$f_{ij}(t_{jj} - t_{ii}) = t_{ij}(f_{jj} - f_{ii}) + \sum_{k=j+1}^{i-1} (t_{ik} f_{kj} - f_{ik} t_{kj}), \tag{A.63}$$

which can be used to compute the f_{ij} elements. In the confluent case, i.e., when $t_{ii} = t_{jj}$, this can be modified by using that

$$\lim_{t_{jj} \to t_{ii}} \frac{f_{jj} - f_{ii}}{t_{jj} - t_{ii}} = \lim_{t_{jj} \to t_{ii}} \frac{e^{t_{jj}} - e^{t_{ii}}}{t_{jj} - t_{ii}} = e^{\frac{t_{ii}+t_{jj}}{2}} = e^{t_{ii}} = f_{ii}. \tag{A.64}$$

This is referred to as Parlets method, Parlet (1976).

A.9 Recursive computations

The mean of some vector variable, say $y_k \ \forall \ k = 1, \ldots, N$, can be computed recursively for e.g. online use. The mean is defined as

$$\bar{y}_N = \frac{1}{N} \sum_{k=1}^{N} y_k. \tag{A.65}$$

We can write (A.65) as

$$\begin{aligned} \bar{y}_N &= \frac{N-1}{N} \frac{1}{N-1} \sum_{k=1}^{N-1} y_k + \frac{1}{N} y_N, \\ &= \frac{N-1}{N} \bar{y}_{N-1} + \frac{1}{N} y_N, \end{aligned} \tag{A.66}$$

Hence we have the following recursive scheme for computing the mean

$$\begin{aligned} \bar{y}_k &= \frac{k-1}{k} \bar{y}_{k-1} + \frac{1}{k} y_k \\ &= \bar{y}_{k-1} + \frac{1}{k}(y_k - \bar{y}_{k-1}). \end{aligned} \tag{A.67}$$

where we use the initialization at $k = 1$ as $\bar{y}_0 = 0$. Equation (A.67) is computing the mean $\bar{y}_k$ at time instant k by updating the mean from the previous time instant $\bar{y}_{k-1}$ with the new information in y_k.

A.10 References

Moler, C. and C. Van Loan (1978). Nineteen dubious ways to compute the exponential of a matrix. *SIAM review*, vol. 20, no. 4, pp. 801-836.

Parlet, B. N. (1976) A recurrence among the elements of functions of triangular matrices. *Linear algebra and its applications*, vol. 14, pp. 117-121.