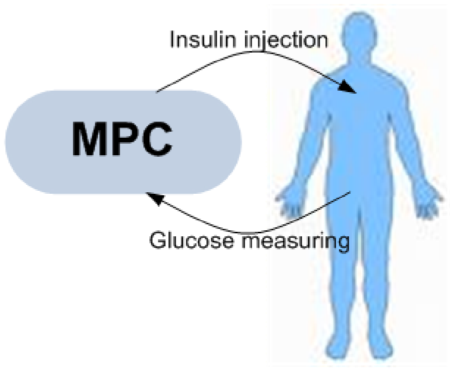
Model Predictive Control
for an artifical pancreas

B.Sc. Thesis

Matias Sørensen og Simon Kristiansen



Technical University of Denmark
Kongens Lyngby 2007

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk
Supervisor: John Bagterp Jørgensen, IMM

Summary

This thesis deals with linear Model Predictive Control, MPC, with the goal of making a controller for an artificial pancreas. A diabetic is simulated by a mathematical model, and based on this model the MPC will compute the optimal insulin input, taking constraints, disturbances and noise into account. Below is a brief description of each chapter.

Chapter 2 describes linearization of differential equations in continuous time. These are converted into a linear state-space model with discrete time representation, which is a requirement for linear MPC.

In Chapter 3, the basic idea for MPC is reviewed. By starting of with MPC on basic form, the control model is extended step by step. Constraints are imposed on the input and on the input rate of movement, which makes sure the input appears in a controlled volume and speed.
Soft constraints are used for the output, to ensure the output are held inside the wanted boundaries, but, if needed, the boundaries can be violated. In some cases it would be impossible to stay within the constraints, and this would make the problem infeasible, if soft constraints wasn't used.
Feedforward and feedback is also described. These two approaches will make the system more robust, and gives a better reaction speed on changes in the process, such as disturbances and noise.

Chapter 4 is the implementation of the MPC problem in MATLAB. This is divided into three different phases: Design, simulation and evaluation. The user calls a file, where the wanted test scenario is initialized. This includes properties such as constraints and meal disturbances. After all the needed parameters are specified, the MPC controller is designed, and the simulation is completed. Finally the simulation is evalutated by various plots.

This implementation is used on a modified version of Bergmans "minimal model" in Chapter 5. This model consists of five differential equations, which simulates a type 1 diabetic patient. The knowledge from the previous chapters is used to transform the model into a linear state space model with discrete time representation, and then use this with MPC. This chapter also discuss the selection of noise and weight matrices. Furthermore simulations are done, to finally test the functionality of the controller.

The thesis concludes that, it is possible that MPC can be used for the purpose of an insulin pump, but severe testing, and a better model would be needed.

Resumé

Formålet med dette projekt er at undersøge om Model Predictive Control, MPC, kan bruges som kontrolanordning til en insulinpumpe, hvilket kan blive brugt til at udvikle en kunstig bugspytkirtel til mennesker som ikke producerer insulin selv.
De basale idéer for lineær MPC vil blive gennemgået og implementeret, og anvendt på en matematisk model for en type 1 diabetiker. Dette inkluderer begrænsninger på insulin input og input-hastigheden, hvilket sørger for der ikke bliver injiceret for meget insulin, og/eller dette foregår for hurtigt.
På systemets output, patientens blodsukker niveau, bruges der "bløde" begrænsninger, hvilket betyder at grænserne kan blive krydset, for at sikre at kontrol problemet ikke bliver uløseligt. Dette betyder at der er en risiko for at patientens blodsukker niveau ikke ligger på et sundt niveau i en kortere periode, men som det vil blive vist, sker dette kun i ekstreme tilfælde.

Kontrolleren håndterer forstyrelser i systemet vha. feedforward og feedback. Feedback får kontrol algoritmen til at evaluere det målte output og dermed basere det kommende insulin input på dette, mens feedforward kan komme fremtidige forstyrrelser, såsom indtagelse af mad og drikke, i forkøbet. Sådan håndtering af forstyrrelser gør at systemet bliver mere robust og hurtigt reagerende.

I simuleringerne bliver patienten udsat for forskellige scenarier med forskellige størrelser måltider. Her bliver kontrollerens ydelse testet, og det viser sig at den håndterer de fleste scnearier godt, dog skal det haves i mente at den fiktive patient er baseret på en noget mangelfuld model.

Contents

Chapter 1

Introduction

In year 2000 there were around 171 mio. people with diabetes in the world. WHO has estimated that this number will be 366 mio. in 2030, i.e. diabetes is a rising problem.
In the USA alone, there is used 130 billion USD a year on diabetes, which is 10% of the health care budget.
There are two types of diabetes, type 1 and type 2. Type 1 is called the insulin dependent diabetes and occur when the cells in the pancreas, which producess insulin, are destroyed.
The human body needs insulin to move glucose from the food into cells, throughout the body, so the energy in the food can be used.
Figure 1.1 illustrates how the pancreas works in a healthy body.
When there isn't produced any insulin, the glucose is accumulated in the blood, which can cause serious damage. Therefore people with diabetes needs to have an external source of insulin, which means injections into the veins.

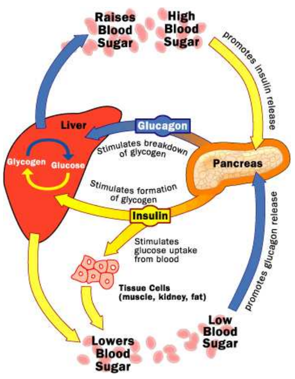

Raises Blood Sugar
High Blood Sugar
promotes insulin release
Liver
Glucagon
Stimulates breakdown of glycogen
Pancreas
Glycogen
Glucose
Stimulates formation of glycogen
Insulin
Stimulates glucose uptake from blood
promotes glucagon release
Tissue Cells (muscle, kidney, fat)
Low Blood Sugar
Lowers Blood Sugar

Figure 1.1: The diabetes-glucose-regulation

These injections are done with a needle several times at day, with the correct proportioning of insulin for the individual. If the patient gets too much insulin he can go into hypoglycemia, and too little can cause hyperglycemia. Hyperglycemia can lead to blindness, kidney failure and other long terms complication, while hypoglycemia can lead to loss of consciousness and coma. A device which could automate the injection of insulin would obviously be a big advantage for the patient.

A comparatively new device for type 1 diabetes is an insulin pump, which makes the injection for the diseased person.
The pump consists essentially of three components: A sensor, which measures the blood glucose concentration in the body, a controller that estimates the needed insulin quantity, and a pump which makes the injection. See Figure 1.2.
This project will deal with a control algorithm for an insulin pump, for which linear Model Predictive Control will be used.

MPC is a controller build on a model for the specific case. The controller calculates the optimal quantity of insulin based on measurements of the blood sugar on the subcutaneous layer. In theory, MPC could be used to make a device, which could act as an artificial pancreas, reducing the impact the disease has on the patients life.

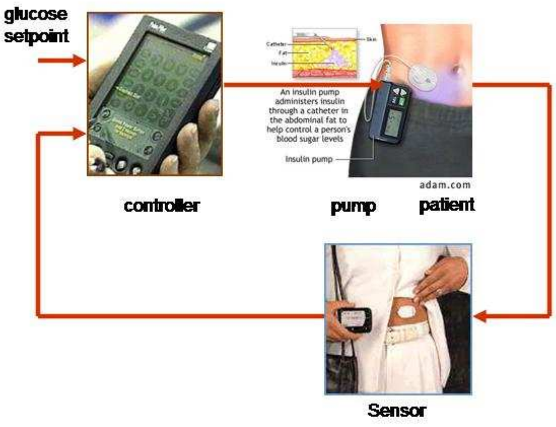

glucose setpoint
Catheter
Fat
Insulin
Skin
An insulin pump administers insulin through a catheter in the abdominal fat to help control a person's blood sugar levels
Insulin pump
adam.com
controller
pump
patient
Sensor

Figure 1.2: The insulin pump

Chapter 2

Linearization

This chapter will show how to convert a model consisting of first order differential equations, into a linear model with discrete time representation, and thereby making it possible to use this model with linear MPC.
Such a system of differential equations in continuous time is denoted by;

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \dot{x}(t) = f(x(t), u(t), d(t)) \tag{2.1}$$

The first step is to identify an equilibrium point, a steady-state, (x_s, u_s, d_s), i.e. $f(x_s, u_s, d_s) = 0$. Making a Taylor expansion around this point yields an approximation to the system (2.1);

$$\begin{aligned}\dot{x}(t) &= f(x(t), u(t), d(t)) \\ &\simeq f(x_s, u_s, d_s) + \left.\frac{\partial f}{\partial x}\right|_{(x_s,u_s,d_s)} (x(t) - x_s) + \left.\frac{\partial f}{\partial u}\right|_{(x_s,u_s,d_s)} (u(t) - u_s) \\ &\qquad + \left.\frac{\partial f}{\partial d}\right|_{(x_s,u_s,d_s)} (d(t) - d_s) \\ &= \bar{A}\,(x(t) - x_s) + \bar{B}\,(u(t) - u_s) + \bar{E}\,(d(t) - d_s)\,, \end{aligned} \tag{2.2}$$

where;

$$\bar{A} = \left.\frac{\partial f}{\partial x}\right|_{(x_s,u_s,d_s)}, \quad \bar{B} = \left.\frac{\partial f}{\partial u}\right|_{(x_s,u_s,d_s)}, \quad \bar{E} = \left.\frac{\partial f}{\partial d}\right|_{(x_s,u_s,d_s)}, \tag{2.3}$$

$\dot{x}(t)$ is transcribed so it containes x_s:

$$\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{\mathrm{d}\left(x(t) - x_s\right)}{\mathrm{d}t} = \frac{\mathrm{d}x(t)}{\mathrm{d}t} - \frac{\mathrm{d}x_s}{\mathrm{d}t} \tag{2.4}$$

Introducing the deviation variables, $X(t) = x(t) - x_s$, $U(t) = u(t) - u_s$ and $D(t) = d(t) - d_s$, gives;

$$\begin{aligned}
&\frac{\mathrm{d}\,(x(t) - x_s)}{\mathrm{d}t} = \bar{A}\,(x(t) - x_s) + \bar{B}\,(u(t) - u_s) + \bar{E}\,(d(t) - d_s) && \Leftrightarrow \\
&\frac{\mathrm{d}X(t)}{\mathrm{d}t} = \bar{A}X(t) + \bar{B}U(t) + \bar{E}D(t) && \Leftrightarrow \\
&\dot{X}(t) = \bar{A}X(t) + \bar{B}U(t) + \bar{E}D(t) && (2.5)
\end{aligned}$$

This is a system of linear time invariant differential equations.
The solution to this system is given in [7];

$$X(t) = e^{\bar{A}(t-t_0)}X(t_0) + \int_{t_0}^{t} e^{\bar{A}(t-s)}\left(\bar{B}U(s) + \bar{E}D(s)\right)\,\mathrm{d}s \tag{2.6}$$

Where e denotes the matrix exponential function.

2.1 Continuous-Discrete Time Conversion

Having the linearized system, the next step is to convert it from continuous time to discrete time. Considering equation (2.6), and let x_k, u_k and d_k still be deviation variables;

$$\begin{aligned}
x_{k+1} &= e^{\bar{A}(t_{k+1}-t_k)}x(t_k) + \int_{t_k}^{t_{k+1}} e^{\bar{A}(t_{k+1}-s)}\left(\bar{B}u(s) + \bar{E}d(s)\right)\,\mathrm{d}s \\
&\Rightarrow e^{\bar{A}T_s}x(t_k) + \int_0^{T_s} e^{\bar{A}(t_{k+1}-s)}\left(\bar{B}u(s) + \bar{E}d(s)\right)\,\mathrm{d}s \\
& && T_s = t_{k+1} - t_k \\
&\Rightarrow e^{\bar{A}T_s}x(t_k) + \int_0^{T_s} e^{\bar{A}\tau}\left(\bar{B}u(\tau) + \bar{E}d(\tau)\right)\,\mathrm{d}\tau \\
& && \tau = t_{k+1} - s \\
&\Rightarrow e^{\bar{A}T_s}x_k + \int_0^{T_s} e^{\bar{A}\tau}\bar{B}\,\mathrm{d}\tau\, u_k + \int_0^{T_s} e^{\bar{A}\tau}\bar{E}\,\mathrm{d}\tau\, d_k \\
&\Rightarrow A\,x_k + B\,u_k + E\,d_k && (2.7)
\end{aligned}$$

where the matrices A, B and E are;

$$A = e^{\bar{A}T_s}, \qquad B = \int_0^{T_s} e^{\bar{A}\tau}\bar{B}\,\mathrm{d}\tau, \qquad E = \int_0^{T_s} e^{\bar{A}\tau}\bar{E}\,\mathrm{d}\tau \tag{2.8}$$

T_s is the sampling time.
A practical way to find A, B and E, is to solve the equation (see [1]);

$$\begin{bmatrix} A & B & E \\ 0 & I & I \end{bmatrix} = e^{M \cdot T_s}, \qquad M = \begin{bmatrix} \bar{A} & \bar{B} & \bar{E} \\ 0 & 0 & 0 \end{bmatrix} \tag{2.9}$$

Let t_k be the time defined as $t_k = t_0 + kT_s$ and let $x(t_k) = x_k$, $u(t_k) = u_k$ and $d(t_k) = d_k$, for $t \leq t_k < t_{k+1}$, be the states at time t_k.
Then the evolution of the states is governed by the difference equation, as above;

$$x(t_{k+1}) = A\,x(t_k) + B\,u(t_k) + E\,d(t_k) \Rightarrow x_{k+1} = A\,x_k + B\,u_k + E\,d_k \tag{2.10}$$

Over a time-interval, from 1 to k, the values of x are;

$$\begin{aligned}
x_1 &= A\,x_s + B\,u_s + E\,d_s \\
x_2 &= A\,x_1 + B\,u_1 + E\,d_1 \\
&= A\,(A\,x_s + B\,u_s + E\,d_s) + B\,u_1 + E\,d_1 \\
&= A^2\,x_s + ABu_s + AEd_s + B\,u_1 + E\,d_1 \\
x_3 &= A\,x_2 + B\,u_2 + E\,d_2 \\
&= A\left(A^2\,x_s + ABu_s + AEd_s + B\,u_1 + E\,d_1\right) + B\,u_2 + E\,d_2 \\
&= A^3 x_s + A^2 Bu_s + A^2 Ed_s + ABu_1 + AEd_1 + Bu_2 + Ed_2 \\
&\vdots \\
x_k &= A^k\,x_s + (A^{k-1}B)u_s + (A^{k-1}E)d_s + \cdots + Bu_{k-1} + Ed_{k-1} \\
&= A^k\,x_s + \sum_{j=0}^{k-1}\left(A^{k-1-j}B\right)u_j + \sum_{j=0}^{k-1}\left(A^{k-1-j}E\right)d_j
\end{aligned} \tag{2.11}$$

By defining the output vector z as the measurable states, the output in k-time is;

$$\begin{aligned}
z_k &= C\,x_k \\
&= C\,A^k\,x_s + \sum_{j=0}^{k-1} C\left(A^{k-1-j}B\right)u_j + \sum_{j=0}^{k-1}\left(A^{k-1-j}E\right)d_j \\
&= C\,A^k\,x_s + \sum_{j=0}^{k-1} H_{k-j}u_j + \sum_{j=0}^{k-1} H_{k-j,d}\,d_j,
\end{aligned} \tag{2.12}$$

H_k is the k'th Markov parameter. H_k and $H_{k,d}$ are denoted below.

$$H_k = \begin{cases} 0, & k = 0 \\ CA^{k-1}B, & k \geq 1 \end{cases}, \qquad H_{k,d} = \begin{cases} 0, & k = 0 \\ CA^{k-1}E, & k \geq 1 \end{cases} \tag{2.13}$$

These parameters will be of use in later sections.

The system can be set up as a linear discrete-time state-space model of the form;

Linearized discrete time state-space model (2.14)

$$x_{k+1} = A\,x_k + B\,u_k + E\,d_k$$
$$z_k = C\,x_k$$

2.2 Summary

In this chapter it has been shown how to convert a system of first order differential equations in continuous time, into a linear model in discrete time. This process is summarized below.

- Linearization of model,
    - Given a system of differential equations, $\dot{x}(t) = f(x(t), u(t), d(t))$, indentify a steady-state point (x_s, u_s, d_s).
    - Calculate $\bar{A} = \frac{\partial f}{\partial x}\Big|_{(x_s,u_s,d_s)}$, $\bar{B} = \frac{\partial f}{\partial u}\Big|_{(x_s,u_s,d_s)}$, and $\bar{E} = \frac{\partial f}{\partial d}\Big|_{(x_s,u_s,d_s)}$
    - The corresponding linear model is $\dot{X}(t) = \bar{A}X(t) + \bar{B}U(t) + \bar{E}D(t)$, with X, U and D as deviation variables.
- Converting to discrete time,
    - Calculate $A = e^{\bar{A}T_s}$, $B = \int_0^{T_s} e^{\bar{A}\tau}\bar{B}\,\mathrm{d}\tau$, and $E = \int_0^{T_s} e^{\bar{A}\tau}\bar{E}\,\mathrm{d}\tau$.
    - Model in discrete time is $x_{k+1} = A\,x_k + B\,u_k + E\,d_k$., where x, u and d are deviation variables.
    - The output is $z_k = C\,x_k$, with C indicating the measureable states.

Chapter 3

Model Predictive Control

Linear Model Predictive Control will now be introduced.
Figur 3.1 illustrates the basic idea for the MPC.

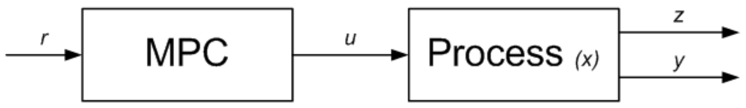


Figure 3.1: The basic MPC

Where z is the actual output and y is the measured output. The controller manipulates the input, u, to achieve an output as close to the setpoint, r, as possible. This basic control model will be expanded step by step in this chapter, for instance by adding disturbance and different sorts of constraints. It will also be shown how these extended control problems can be formulated as a Quadratic Programming (QP) problem, since QP problems are easily solved by known algorithms.

3.1 Unconstrained MPC

In this section focus will be on the basic control model, the unconstrained MPC. The goal of the controller is, as mentioned, to make the difference between the output, z_k, and the reference, r_k, as small as possible. This can be done by using a least squares problem. The weighted 2-norm is used:

$$\frac{1}{2}\sum_{k=0}^{N}||z_k - r_k||_{Q_y}^2 \tag{3.1}$$

Since z_0 can't be influenced, the term $\frac{1}{2}||z_0 - r_0||^2_{Q_z}$ is discarded. This gives the first control problem;

Basic control problem	(3.2)

$$\begin{aligned}
\min \quad & \phi_z = \frac{1}{2}\sum_{k=1}^{N} ||z_k - r_k||^2_{Q_z} \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k, \qquad k = 0, 1, \dots, N-1 \\
& z_k = Cx_k, \qquad k = 0, 1, \dots, N
\end{aligned}$$

From (2.12) it's known that;

$$\begin{aligned}
z_k &= CA^k x_0 + \sum_{j=0}^{k-1} H_{k-j} u_j + \sum_{j=0}^{k-1} H_{k-j,d}\, d_j \\
&= z_{x_0} + z_{u_j} + z_{d_j} \qquad (3.3)
\end{aligned}$$

For now the term z_{d_j} is discarded, such that;

$$z_k = z_{x_0} + z_{u_j} = CA^k x_0 + \sum_{j=0}^{k-1} H_{k-j} u_j \qquad (3.4)$$

This can be written as;

$$z_0 = Cx_0 \qquad (3.5)$$

$$\underbrace{\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_N \end{bmatrix}}_{Z} = \underbrace{\begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^N \end{bmatrix}}_{\Phi} x_0 + \underbrace{\begin{bmatrix} H_1 & 0 & 0 & \cdots & 0 \\ H_2 & H_1 & 0 & \cdots & 0 \\ H_3 & H_2 & H_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_N & H_{N-1} & H_{N-2} & \cdots & H_1 \end{bmatrix}}_{\Gamma} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{U} \qquad (3.6)$$

$$\Rightarrow Z = \Phi x_0 + \Gamma U \qquad (3.7)$$

By introducing R as a vector containing the setpoints

$$R = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix} \qquad (3.8)$$

the objective function can be written as;

$$\phi_z = \frac{1}{2}\sum_{k=1}^{N} ||z_k - r_k||^2_{Q_z} = \frac{1}{2}||Z - R||^2_{\mathcal{Q}_z} \tag{3.9}$$

Where the weight matrix $\mathcal{Q}_z$ is given by;

$$\mathcal{Q}_z = \begin{bmatrix} Q_z & 0 & \cdots & 0 \\ 0 & Q_z & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & Q_z \end{bmatrix} \tag{3.10}$$

Plugging equation (3.7) into (3.9) gives;

$$\begin{aligned} \phi_z &= \frac{1}{2}||Z - R||^2_{\mathcal{Q}_z} && (3.11)\\ &= \frac{1}{2}||\Phi x_0 + \Gamma U - R||^2_{\mathcal{Q}_z} && (3.12)\\ &= \frac{1}{2}||\Gamma U - (R - \Phi x_0)||^2_{\mathcal{Q}_z} && (3.13)\\ &= \frac{1}{2}||\Gamma U - b||^2_{\mathcal{Q}_z}, \qquad b = R - \Phi x_0 && (3.14) \end{aligned}$$

To make this problem easier to solve, it is convenient to express it as a QP problem;

$$\begin{aligned} \phi_z &= \frac{1}{2}||\Gamma U - b||^2_{\mathcal{Q}_z} \\ &= \frac{1}{2}(\Gamma U - b)^{\mathrm{T}}\mathcal{Q}_z(\Gamma U - b), \qquad \text{by definition of weighted norm} \\ &= \frac{1}{2}U^{\mathrm{T}}\Gamma^{\mathrm{T}}\mathcal{Q}_z\Gamma U - (\Gamma^{\mathrm{T}}\mathcal{Q}_z b)^{\mathrm{T}}U + \frac{1}{2}b^{\mathrm{T}}\mathcal{Q}_z b \\ &= \frac{1}{2}U^{\mathrm{T}}H_z U + g_z^{\mathrm{T}}U + \rho_z && (3.15) \end{aligned}$$

where H_z, g_z and ρ_z are given by;

$$\begin{aligned} H_z &= \Gamma^{\mathrm{T}}\mathcal{Q}_z\Gamma && (3.16)\\ g_z &= -\Gamma^{\mathrm{T}}\mathcal{Q}_z b \\ &= -\Gamma^{\mathrm{T}}\mathcal{Q}_z(R - \Phi x_0) \\ &= \Gamma^{\mathrm{T}}\mathcal{Q}_z\Phi x_0 - \Gamma^{\mathrm{T}}\mathcal{Q}_z R \\ &= M_{x_0}x_0 + M_R R && (3.17)\\ \rho_z &= \frac{1}{2}b^{\mathrm{T}}\mathcal{Q}_z b && (3.18) \end{aligned}$$

This is the QP problem equivalent to problem (3.2);

QP formulation of problem (3.2) (3.19)

$$\min_{U} \quad \phi_z = \frac{1}{2}U^{\mathrm{T}}H_zU + g_z^{\mathrm{T}}U$$
$$H_z = \Gamma^{\mathrm{T}}\mathcal{Q}_z\Gamma$$
$$g_z = M_{x_0}x_0 + M_R R$$

ρ_z is discarded since it doesn't influence the solution to the problem. Note that the gradient is dynamic and needs to be updated for every timestep, as opposed to the Hessian, which is static.

3.1.1 Regularization

Regularization is done by introducing a new term, $\phi_{\Delta u}$, in the objective function, where $\Delta u_k = u_k - u_{k-1}$;

Control problem, with regularization (3.20)

$$\min \quad \phi = \phi_z + \phi_{\Delta u} = \frac{1}{2}\sum_{k=1}^{N}||z_k - r_k||_{Q_z}^2 + \frac{1}{2}\sum_{k=0}^{N-1}||\Delta u_k||_S^2$$
$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \qquad k = 0, 1, \ldots, N-1$$
$$z_k = Cx_k, \qquad k = 0, 1, \ldots, N$$

This new term minimizes the difference between two consecutive steps in u, which gives more "smooth" input, ie. tries to ensure that steps in u are either continously decreasing or continously increasing.
Again this should be formulated as a QP problem. Compared to problem (3.2), the only difference is a new term in the objective function, so to formulate as a

QP problem, this new term has to be rewritten;

$$
\begin{aligned}
\phi_{\Delta u} &= \frac{1}{2}\sum_{k=0}^{N-1} ||\Delta u_k||_S^2 \\
&= \frac{1}{2}\sum_{k=0}^{N-1} ||u_k - u_{k-1}||_S^2 \\
&= \frac{1}{2}\sum_{k=0}^{N-1} (u_k - u_{k-1})^{\mathrm{T}} S (u_k - u_{k-1}) \\
&= \frac{1}{2}\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix}^{\mathrm{T}}
\underbrace{\begin{bmatrix} 2S & -S & & & \\ -S & 2S & -S & & \\ & \ddots & \ddots & \ddots & \\ & & -S & 2S & -S \\ & & & -S & S \end{bmatrix}}_{H_S}
\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix} \\
&\qquad + \left(-\underbrace{\begin{bmatrix} S \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{M_{u-1}} u_{-1}^{\mathrm{T}} \right) \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix} + \frac{1}{2} u_{-1} S u_{-1} \\
&= \frac{1}{2} U^{\mathrm{T}} H_S U + \left(M_{u-1} u_{-1}\right)^{\mathrm{T}} U + \frac{1}{2} u_{-1} S u_{-1} \qquad (3.21)
\end{aligned}
$$

This shows, that introducing $\phi_{\Delta u}$ extends the QP problem by the following terms;

$$H_{\Delta u} = H_S \qquad (3.22)$$

$$g_{\Delta u} = M_{u-1} u_{-1} \qquad (3.23)$$

Like with ρ_z, the term $\frac{1}{2}u_{-1}Su_{-1}$ is discarded, because of the lack of influence on the solution to the problem. The new QP problem is;

QP formulation of problem (3.20) (3.24)

$$\min_U \quad \phi = \frac{1}{2} U^{\mathrm{T}} H U + g^{\mathrm{T}} U$$

$$H = H_z + H_{\Delta u} = \Gamma^{\mathrm{T}} \mathcal{Q}_z \Gamma + H_S$$

$$g = g_z + g_{\Delta u} = M_{x_0} x_0 + M_R R + M_{u-1} u_{-1}$$

3.1.2 Disturbance

The MPC problem will now be extended by adding disturbance. Virtually every system has some disturbance. Figure 3.2 illustrates how the disturbance d influences the MPC problem.

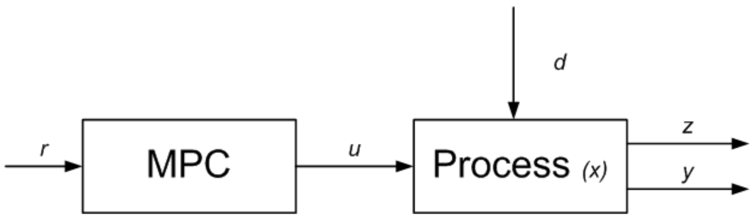


Figure 3.2: MPC with disturbance

The disturbance in this model is relatively easily handled by including the last term, z_{d_j}, from equation (3.3), in the problem;

Control problem with regularization and disturbance		(3.25)
min	$\phi = \phi_z + \phi_{\Delta u} = \frac{1}{2}\sum_{k=1}^{N} \|\|z_k - r_k\|\|_{Q_z}^2 + \frac{1}{2}\sum_{k=0}^{N-1} \|\|\Delta u_k\|\|_S^2$	
s.t.	$x_{k+1} = Ax_k + Bu_k + Ed_k,$	$k = 0, 1, \dots, N-1$
	$z_k = Cx_k,$	$k = 0, 1, \dots, N$

As mentioned in Chapter 2.1, H_i is analogous to $H_{i,d}$, so z_{d_j} must then be analogous to z_{u_j}. Therefore it's easy to see that including the term z_{d_j} yields a new term, analogous to ΓU, in Z. This new term is named $\Gamma_d D$;

$$Z = \Phi x_0 + \Gamma U + \Gamma_d D \tag{3.26}$$

Where;

$$\Gamma_d = \begin{bmatrix} H_{1,d} & 0 & 0 & \cdots & 0 \\ H_{2,d} & H_{1,d} & 0 & \cdots & 0 \\ H_{3,d} & H_{2,d} & H_{1,d} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_{N,d} & H_{N-1,d} & H_{N-2,d} & \cdots & H_{1,d} \end{bmatrix}, \qquad D = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{N-1} \end{bmatrix} \tag{3.27}$$

The introduction of this new term in Z means, that ϕ_z has to be rewritten as a QP;

$$\begin{aligned} \phi_z &= \frac{1}{2}\sum_{k=1}^{N} ||z_k - r_k||^2_{Q_z} \\ &= \frac{1}{2}||Z - R||^2_{\mathcal{Q}_z} \\ &= \frac{1}{2}||\Gamma U - (R - \Phi x_0 - \Gamma_d D)||^2_{\mathcal{Q}_z}, && \text{by using (3.26)} \\ &= \frac{1}{2}||\Gamma U - b||^2_{\mathcal{Q}_z}, && b = R - \Phi x_0 - \Gamma_d D \\ &= \frac{1}{2}(U^{\mathrm{T}}\Gamma^{\mathrm{T}} - b^{\mathrm{T}})\mathcal{Q}_z(\Gamma U - b) \\ &= \frac{1}{2}\left(U^{\mathrm{T}}\Gamma^{\mathrm{T}}\mathcal{Q}_z\Gamma U - b^{\mathrm{T}}\mathcal{Q}_z\Gamma U - U^{\mathrm{T}}\Gamma^{\mathrm{T}}\mathcal{Q}_z b + b^{\mathrm{T}}\mathcal{Q}_z b\right) \\ &= \frac{1}{2}U^{\mathrm{T}}\Gamma^{\mathrm{T}}\mathcal{Q}_z\Gamma U - (\Gamma^{\mathrm{T}}\mathcal{Q}_z b)^{\mathrm{T}}U + \frac{1}{2}b^{\mathrm{T}}\mathcal{Q}_z b \\ &= \frac{1}{2}U^{\mathrm{T}}H_z U + g_z^{\mathrm{T}}U + \rho_z \end{aligned} \tag{3.28}$$

Where;

$$\begin{aligned} H_z &= \Gamma^{\mathrm{T}}\mathcal{Q}_z\Gamma && (3.29) \\ g_z &= -\Gamma^{\mathrm{T}}\mathcal{Q}_z b \\ &= -\Gamma^{\mathrm{T}}\mathcal{Q}_z(R - \Phi x_0 - \Gamma_d D) \\ &= M_{x_0}x_0 + M_R R + M_D D && (3.30) \\ \rho_z &= \frac{1}{2}b^{\mathrm{T}}\mathcal{Q}_z b && (3.31) \end{aligned}$$

The QP problem is summarized below. Compared to (3.24), the problem has been expanded by a new term in the gradient.

QP formulation of problem (3.25) (3.32)

$$\min_{U} \quad \phi = \frac{1}{2}U^{\mathrm{T}}HU + g^{\mathrm{T}}U$$

$$H = \Gamma^{\mathrm{T}}\mathcal{Q}_z\Gamma + H_S$$

$$g = M_{x_0}x_0 + M_R R + M_D D + M_{u-1}u_{-1}$$

3.1.3 Feed-forward/feedback

Two important aspects in a MPC controller is feedforward and feedback. They both react on changes in the process, and can therefore make the controller better in terms of reaction speed and robustness.
The feedback approach is illustrated on Figure 3.3.

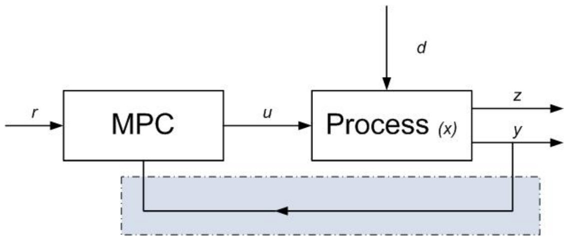


Figure 3.3: MPC with feedback

Feedback is often used to control the dynamic behavior of the system. When providing a system with feedback, the measured output, y, and thereby the potential occurring disturbance, is fed back to the controller.
This ensures that the measured output is approximately the same as the wanted value, the reference, r, which means the system will be more robust. The drawback with feedback is slow reaction speed, since output has to be measured before it can be used.

The feedforward approach is illustrated on Figure 3.4. Feedforward is a loop in the system, which takes some known disturbance and forward it to the controller.

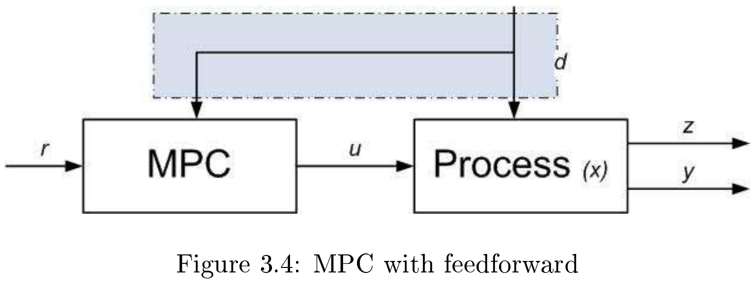


Figure 3.4: MPC with feedforward

When a system exhibits feedforward behavior, it responds to disturbances which are predefined and therefore known.
This means, that the system can respond more quickly to the disturbance and the measured output will be more robust.

But meanwhile there is a relatively big chance that the output isn't comparable to the wanted values, the references, r, since it cannot do much about novel disturbance.

When combining these two appoaches, the system will be aware of both the known and the unmeasured disturbance, so the system in theory will be trust-worthy and fast. Figur 3.5 illustrates a controller where both these approaches are used.

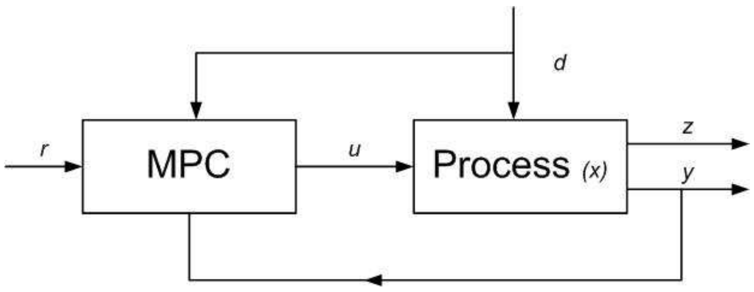


Figure 3.5: The MPC-model with feed-forward/feedback

3.2 Constrained MPC

The MPC will be extended such that it contains constraints.
The MPC problem has to take the limits of the physical system into consideration. It's infrequent that a system has no boundaries. Constraints are imposed on the input quantity, the input rate of movement and on the output. For all constraints it's assumed that, for every timestep k, the boundaries are the same.

3.2.1 Input constraints

The input constraints are limitations to the maximum and minimum input volume.

MPC with input constraint (3.33)

$$\begin{aligned}
\min \quad & \phi = \frac{1}{2}\sum_{k=1}^{N} ||z_k - r_k||_{Q_z}^2 + \frac{1}{2}\sum_{k=0}^{N-1} ||\Delta u_k||_S^2 \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k + Ed_k, \qquad && k = 0,1,\dots,N-1 \\
& z_k = Cx_k, && k = 0,1,\dots,N \\
& u_{\min} \leq u_k \leq u_{\max} && k = 0,1,\dots,N-1
\end{aligned}$$

To formulate this as a QP problem this new constraint is put in vector form;

$$u_{\min} \leq u_k \leq u_{\max} \Leftrightarrow \underbrace{\begin{bmatrix} u_{\min} \\ u_{\min} \\ \vdots \\ u_{\min} \end{bmatrix}}_{\mathring{U}_{\min}} \leq \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \leq \underbrace{\begin{bmatrix} u_{\max} \\ u_{\max} \\ \vdots \\ u_{\max} \end{bmatrix}}_{\mathring{U}_{\max}} \tag{3.34}$$

This yields a constrained QP problem;

QP formulation of problem (3.33) (3.35)

$$\begin{aligned}
\min_U \quad & \phi = \frac{1}{2}U^{\mathrm{T}}HU + g^{\mathrm{T}}U \\
\text{s.t.} \quad & \mathring{U}_{\min} \leq U \leq \mathring{U}_{\max}
\end{aligned}$$

The Hessian and gradient for this problem are the same as in problem (3.32).

3.2.2 Constraints on input rate of movement

It is also possible to have constraints on how fast the input constraints can change. The input rate of movement is the change from k to $k+1$ and is therefore called Δu_k. The control problem is extended again;

MPC with constraints on input and input rate (3.36)

$$\begin{aligned}
\min \quad & \phi = \frac{1}{2}\sum_{k=1}^{N} ||z_k - r_k||_{Q_z}^2 + \frac{1}{2}\sum_{k=0}^{N-1} ||\Delta u_k||_S^2 \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k + Ed_k, && k = 0, 1, \dots, N-1 \\
& z_k = Cx_k, && k = 0, 1, \dots, N \\
& u_{\min} \le u_k \le u_{\max}, && k = 0, 1, \dots, N-1 \\
& \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, && k = 0, 1, \dots, N-1
\end{aligned}$$

This new constraint for Δu_k can be written as;

$$\Delta u_{\min} \le \Delta u_k \le \Delta u_{\max} \Leftrightarrow$$

$$\begin{bmatrix} \Delta u_{\min} \\ \Delta u_{\min} \\ \vdots \\ \Delta u_{\min} \end{bmatrix} \le \begin{bmatrix} u_0 - u_{-1} \\ u_1 - u_0 \\ \vdots \\ u_N - u_{N-1} \end{bmatrix} \le \begin{bmatrix} \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \Leftrightarrow$$

$$\begin{bmatrix} \Delta u_{\min} + u_{-1} \\ \Delta u_{\min} \\ \vdots \\ \Delta u_{\min} \end{bmatrix} \le \begin{bmatrix} I & & & \\ -I & I & & \\ & \ddots & \ddots & \\ & & -I & I \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \le \begin{bmatrix} \Delta u_{\max} + u_{-1} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix}$$

Since the first row contains u_{-1} this can be written as;

$$\Delta u_{\min} + u_{-1} \le u_0 \le \Delta u_{\max} + u_{-1} \quad \wedge \tag{3.37}$$

$$\underbrace{\begin{bmatrix} \Delta u_{\min} \\ \Delta u_{\min} \\ \vdots \\ \Delta u_{\min} \end{bmatrix}}_{\Delta U_{\min}} \le \underbrace{\begin{bmatrix} -I & I & & & \\ & -I & I & & \\ & & \ddots & \ddots & \\ & & & -I & I \end{bmatrix}}_{\Lambda} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix} \le \underbrace{\begin{bmatrix} \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix}}_{\Delta U_{\max}} \tag{3.38}$$

An input constraint for u_0 is already given in (3.34), combining with (3.37) gives;

$$\left.\begin{aligned} u_0 &\leq \Delta u_{\max} + u_{-1} \\ u_0 &\leq u_{\max} \\ u_0 &\geq \Delta u_{\min} + u_{-1} \\ u_0 &\geq u_{\min} \end{aligned}\right\} \Rightarrow \begin{cases} u_0 \leq \min(u_{\max}, \Delta u_{\max} + u_{-1}) \\ u_0 \geq \max(u_{\min}, \Delta u_{\min} + u_{-1}) \end{cases} \tag{3.39}$$

So the input constraints becomes;

$$\underbrace{\begin{bmatrix} \max(u_{\min}, \Delta u_{\min} + u_{-1}) \\ u_{\min} \\ \vdots \\ u_{\min} \end{bmatrix}}_{U_{\min}} \leq \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \leq \underbrace{\begin{bmatrix} \min(u_{\max}, \Delta u_{\max} + u_{-1}) \\ u_{\max} \\ \vdots \\ u_{\max} \end{bmatrix}}_{U_{\max}} \tag{3.40}$$

The QP problem becomes;

QP formulation of problem (3.36) (3.41)

$$\begin{aligned} \min_{U} \quad & \phi = \frac{1}{2} U^{\mathrm{T}} H U + g^{\mathrm{T}} U \\ \text{s.t.} \quad & U_{\min} \leq U \leq U_{\max} \\ & \Delta U_{\min} \leq \Lambda U \leq \Delta U_{\max} \end{aligned}$$

Note that $U_{\min}$ and $U_{\max}$ are dynamic and needs to be re-calculated for every timestep.

3.2.3 Output constraints

Output constraints are analogous to the input constraints, i.e. limitations to the maximum and minimum output.

MPC with output and input constraints (3.42)

$$\begin{aligned}
\min \quad & \phi = \frac{1}{2}\sum_{k=0}^{N} ||z_k - r_k||^2_{Q_z} + \frac{1}{2}\sum_{k=0}^{N-1} ||\Delta u_k||^2_S \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k + Ed_k, && k = 0, 1, \dots, N-1 \\
& z_k = Cx_k, && k = 0, 1, \dots, N \\
& u_{\min} \le u_k \le u_{\max}, && k = 0, 1, \dots, N-1 \\
& \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, && k = 0, 1, \dots, N-1 \\
& z_{\min} \le z_k \le z_{\max}, && k = 1, 2, \dots, N
\end{aligned}$$

The output at k = 0 cannot be affected, so the constraint here is disregarded. This leads to:

$$z_{\min} \le z_k \le z_{\max} \Rightarrow$$
$$\underbrace{\begin{bmatrix} z_{\min} \\ z_{\min} \\ \vdots \\ z_{\min} \end{bmatrix}}_{Z_{\min}} \le \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}}_{Z} \le \underbrace{\begin{bmatrix} z_{\max} \\ z_{\max} \\ \vdots \\ z_{\max} \end{bmatrix}}_{Z_{\max}} \tag{3.43}$$

Using $Z = \Phi x_0 + \Gamma U + \Gamma_d D$ from equation (3.26) yields;

$$\begin{aligned}
& Z_{\min} \le Z \le Z_{\max} \Leftrightarrow \\
& Z_{\min} \le \Phi x_0 + \Gamma U + \Gamma_d D \le Z_{\max} \Leftrightarrow \\
& \underbrace{Z_{\min} - \Phi x_0 - \Gamma_d D}_{\bar{Z}_{\min}} \le \Gamma U \le \underbrace{Z_{\max} - \Phi x_0 - \Gamma_d D}_{\bar{Z}_{\max}}
\end{aligned} \tag{3.44}$$

The problem transforms into;

QP formulation of problem (3.42) (3.45)

$$
\begin{aligned}
\min_{U} \quad & \phi = \frac{1}{2}U^{\mathrm{T}}HU + g^{\mathrm{T}}U \\
\text{s.t.} \quad & U_{\min} \leq U \leq U_{\max} \\
& \Delta U_{\min} \leq \Lambda U \leq \Delta U_{\max} \\
& \bar{Z}_{\min} \leq \Gamma U \leq \bar{Z}_{\max}
\end{aligned}
$$

3.2.4 Soft output constraints

It can occour that the control problem will be infeasible. This is mainly due to the use of constraints greatly complicating the problem. In some cases it can simply be impossible to stay within the boundaries.
A solution to this problem is to soften the constraints, meaning the boundaries can be violated occasionally, if needed. In this case, soft output constraints are used. The easiest way to softening output constraints is to introduce a new slack variable, ψ.

MPC with input and soft output constraints (3.46)

$$
\begin{aligned}
\min \quad & \phi = \phi_z + \phi_\psi + \phi_{\Delta u} \\
& \quad = \sum_{k=1}^{N} \frac{1}{2}||z_k - r_k||_{Q_z}^2 + \frac{1}{2}||\psi_k||_{S_\psi}^2 + s_\psi^{\mathrm{T}}\psi_k + \frac{1}{2}\sum_{k=0}^{N-1} ||\Delta u_k||_S^2 \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k + Ed_k, && k = 0, 1, \ldots, N-1 \\
& z_k = Cx_k, && k = 0, 1, \ldots, N \\
& u_{\min} \leq u_k \leq u_{\max}, && k = 0, 1, \ldots, N-1 \\
& \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max}, && k = 0, 1, \ldots, N-1 \\
& -\infty < z_k - \psi_k \leq z_{\max}, && k = 1, 2, \ldots, N \\
& z_{\min} \leq z_k + \psi_k < \infty, && k = 1, 2, \ldots, N \\
& 0 \leq \psi_k < \infty, && k = 1, 2, \ldots, N
\end{aligned}
$$

If it's not neccesary to violate output constraints, the solution to the problem will simply have $\psi = 0$, and will be equivalent to the solution with hard output constraints. If it's not possible to use hard constraints, ψ is selected as small as possible, so the output constraints are only violated by a minimum.

The approach to formulate (3.46) as a QP problem is a little different than for the previous problems. First Ψ is introduced;

$$\Psi = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_N \end{bmatrix} \tag{3.47}$$

The new term in the objective function, ϕ_ψ, is formulated as QP;

$$\phi_\psi = \frac{1}{2}||\psi_k||^2_{S_\psi} + s_\psi^{\mathrm{T}}\psi_k = \frac{1}{2}\Psi^{\mathrm{T}}\bar{S}_\psi\Psi + \bar{s}_\psi^{\mathrm{T}}\Psi \tag{3.48}$$

where;

$$\bar{S}_\psi = \begin{bmatrix} S_\psi & & \\ & \ddots & \\ & & S_\psi \end{bmatrix}, \quad \bar{s}_\psi = \begin{bmatrix} s_\psi \\ \vdots \\ s_\psi \end{bmatrix} \tag{3.49}$$

For this problem the Hessian is $\bar{S}_\psi$ and the gradient is $\bar{s}_\psi$. The QP problem should be formulated in terms of the vector $\bar{U}$,

$$\bar{U} = \begin{bmatrix} U \\ \Psi \end{bmatrix} \tag{3.50}$$

Which means the objective function for the QP should be $\frac{1}{2}\bar{U}^{\mathrm{T}}\bar{H}\bar{U} + \bar{g}^{\mathrm{T}}\bar{U}$. The Hessian and gradient for this problem are;

$$\bar{H} = \begin{bmatrix} H & 0 \\ 0 & \bar{S}_\psi \end{bmatrix}, \quad \bar{g} = \begin{bmatrix} g \\ \bar{s}_\psi \end{bmatrix} \tag{3.51}$$

The constraints for U and Ψ should then be written as;

$$\underbrace{\begin{bmatrix} U_{\min} \\ 0 \end{bmatrix}}_{\bar{U}_{\min}} \leq \underbrace{\begin{bmatrix} U \\ \Psi \end{bmatrix}}_{\bar{U}} \leq \underbrace{\begin{bmatrix} U_{\max} \\ \infty \end{bmatrix}}_{\bar{U}_{\max}} \tag{3.52}$$

It's obvious that as in the previous QP problem, (3.45), the constraint $\Delta U_{\min} \leq \Lambda U \leq \Delta U_{\max}$ should still be imposed. Formulated in terms of $\bar{U}$ yields;

$$\Delta U_{\min} \leq \begin{bmatrix} \Lambda & 0 \end{bmatrix} \begin{bmatrix} U \\ \Psi \end{bmatrix} \leq \Delta U_{\max} \tag{3.53}$$

The upper bound on z_k needs to be transcribed;

$$\begin{aligned}
&-\infty < z_k - \psi_k \le z_{\max} && && \Leftrightarrow \\
&-\infty < Z - \Psi \le Z_{\max} && && \Leftrightarrow \\
&-\infty < \Phi\, x_0 + \Gamma_D\, D + \Gamma\, U - \Psi \le Z_{\max}, && \text{using (3.26)} && \Leftrightarrow \\
&-\infty < \Gamma\, U - \Psi \le Z_{\max} - (\Phi\, x_0 + \Gamma_D\, D) && && \Leftrightarrow \\
&-\infty < \Gamma\, U - \Psi \le \bar{Z}_{\max} && && \Leftrightarrow \\
&-\infty < \begin{bmatrix} \Gamma & -I \end{bmatrix} \begin{bmatrix} U \\ \Psi \end{bmatrix} \le \bar{Z}_{\max} && && (3.54)
\end{aligned}$$

Equivalently, this can be done for the lower bound, which gives;

$$\bar{Z}_{\min} \le \begin{bmatrix} \Gamma & I \end{bmatrix} \begin{bmatrix} U \\ \Psi \end{bmatrix} < \infty \tag{3.55}$$

Combining equations (3.53), (3.54), and (3.55) in matrix formulation yields;

$$\underbrace{\begin{bmatrix} \Delta U_{\min} \\ -\infty \\ \bar{Z}_{\min} \end{bmatrix}}_{b_{\min}} \le \underbrace{\begin{bmatrix} \Lambda & 0 \\ \Gamma & -I \\ \Gamma & I \end{bmatrix}}_{A} \begin{bmatrix} U \\ \Psi \end{bmatrix} \le \underbrace{\begin{bmatrix} \Delta U_{\max} \\ \bar{Z}_{\max} \\ \infty \end{bmatrix}}_{b_{\max}} \tag{3.56}$$

So the QP problem is;

QP formulation of problem (3.46) (3.57)

$$\begin{aligned}
\min_{U} \quad & \phi = \frac{1}{2}\bar{U}^{\mathrm{T}} \bar{H} \bar{U} + \bar{g}^{\mathrm{T}} \bar{U} \\
\text{s.t.} \quad & \bar{U}_{\min} \le \bar{U} \le \bar{U}_{\max} \\
& b_{\min} \le A\bar{U} \le b_{\max}
\end{aligned}$$

3.3 Kalman filter

The basic ideas behind the Kalman filter will now be introduced. Kalman Filter is used to minimize the impact of noise in the problem. For further and more detailed information, see [8] and [9].
To make the simulations more realistic, noice is assumed on both process and output. This noise is denoted w_k and v_k, respectively;

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3.58}$$
$$y_k = Cx_k + v_k \tag{3.59}$$

Both w_k and v_k are assumed to be white noise and normal distributed, with mean zero and covariance Q_w and R_v, respectively.

$$w_k \sim N(0, Q_w) \tag{3.60}$$
$$v_k \sim N(0, R_v) \tag{3.61}$$

An estimate of x is given by:

$$\begin{aligned}\hat{x}_{k+1|k} &= E\left\{x_{k+1}|y_k, y_{k-1}, \dots, y_0\right\}\\ &= E\left\{x_{k+1}|\mathcal{Y}\right\}\\ &= E\left\{Ax_k + Bu_k + w_k|\mathcal{Y}\right\}\\ &= A\ E\left\{x_k|\mathcal{Y}\right\} + Bu_k + E\left\{w_k|\mathcal{Y}\right\}\\ &= A\hat{x}_{k|k} + Bu_k + \hat{w}_{k|k}\end{aligned} \tag{3.62}$$

Since the mean of w_k is equal to 0, the noise estimate becomes:

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k \tag{3.63}$$

This means that $x_{k|k}$ is normal distibuted with mean $\hat{x}_{k|k}$ and error covariance $P_{k|k}$, i.e:

$$x_{k|k} \sim N\left(\hat{x}_{k|k}, P_{k|k}\right) \tag{3.64}$$

The system can be augmented by adding an integrated disturbance, η_k, to achieve offset-free performance;

$$x_{k+1} = Ax_k + B\left(u_k + \eta_k\right) + w_k \tag{3.65}$$
$$\eta_{k+1} = I\eta_k + \xi \tag{3.66}$$
$$y_k = Cx_k + v_k \tag{3.67}$$

Transformed into matrix form gives;

$$\begin{bmatrix} x_{k+1} \\ \eta_{k+1} \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ \eta_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} w_k \\ \xi \end{bmatrix} \tag{3.68}$$

$$y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \eta_k \end{bmatrix} + v_k \tag{3.69}$$

where η_k is normal distributed with zero mean and covariance $Q_{\eta,k}$ and ξ is disturbance noise and normal distributed with $\xi \sim N(0, Q_\xi)$.

Now the Kalman Filter is used to estimate the state $\hat{x}_{k+1|k}$.
First phase of the Kalman filter is the "time update". It is used to produce an estimate of the current state, using the estimate of the previous state;

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \tag{3.70}$$

$$P_{k|k-1} = AP_{k-1|k-1}A^{\mathrm{T}} + Q_w \tag{3.71}$$

The second phase of the filter, the "measurement update", refines measurement information from the current timestep to get a new, more accurate, estimate;

$$e_k = y_k - C\hat{x}_{k|k-1} \tag{3.72}$$

$$S_k = CP_{k|k-1}C^{\mathrm{T}} + R_v \tag{3.73}$$

$$K_kS_k = P_{k|k-1}C^{\mathrm{T}}, \qquad \text{(optimal Kalman gain, solve for } K_k\text{)} \tag{3.74}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_ke_k \tag{3.75}$$

$$P_{k|k} = (I - K_kC)\,P_{k|k-1} \tag{3.76}$$

An illustration of the Kalman Filter is shown on figure 3.6.

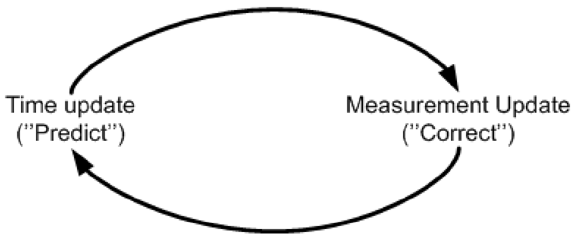


Figure 3.6: The Kalman Filter cycle

3.4 Summary

This chapter derived the MPC problem (3.46) shown on Figure 3.7.

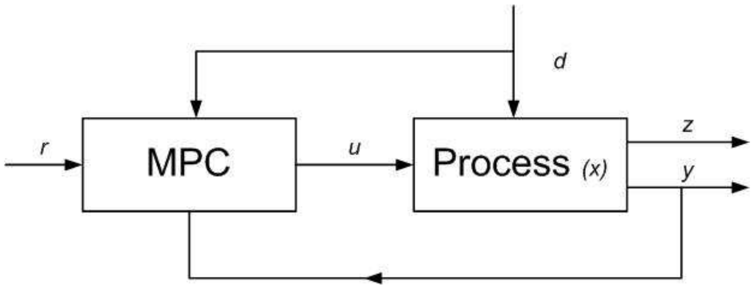


Figure 3.7: The MPC-model

The goal of the MPC controller is to make the output, z, as close to the reference, r, as possible. This is done by calculating the optimal input, u, for each time-step. Feedback is used, such that the controller analyzes the measured output, y. When feedforward is used, the controller analyzes the future disturbances, d. Feedback and feedforward can well be used simultaneously.

The described MPC algorithm also takes care of input constraints, constraints on input rate of movement, and soft output constraints. The soft output constraints keeps us from ending up with an infeasible problem, but can also violate the physical limitations for the output, which may cause the system to malfunction and should therefore be used with concern.
Also the use of the Kalman Filter is described. The filter makes an estimate of x, when there is an addition of noise on the output measurements and/or the process.

Formulating the control problem as a QP problem makes it relatively easy solvable.

CHAPTER 4

Implementation

This chapter will describe the implementation of the MPC problem (3.46) in MATLAB.
The process tree is seen on Figure 4.1.

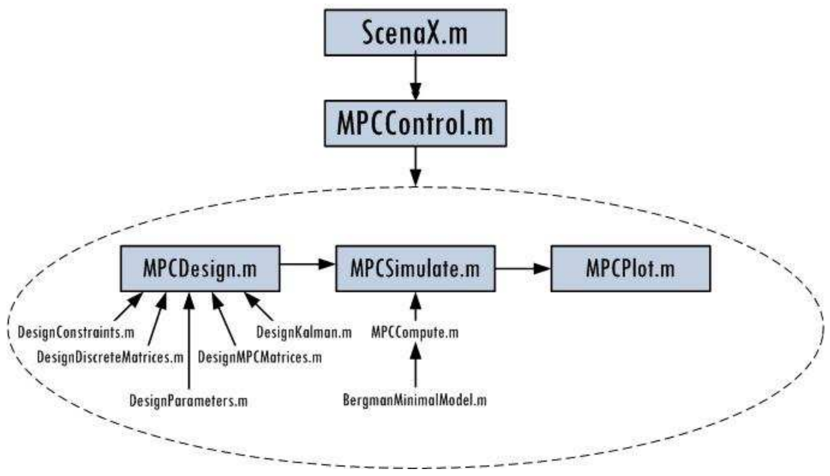


Figure 4.1: MPC tree

The individual MATLAB files will now be described in detail. Names in `typewriter font` nominates MATLAB entries.

4.1 ScenaX.m

SceneX.m is the file which is called by the user. The X specifies the name of the scene. The idea is that, all parameters the user should have influence on, can be adjusted in this file. Table 4.1 shows the basic variables which this file initializes.

Parameters(s)	MATLAB name(s)	Describtion
N	N	Size of prediction horizon.
$u_{\min}$, $u_{\max}$	umin, umax	Bounds on u.
$\Delta u_{\min}$, $\Delta u_{\max}$	udmin, udmax	Bound on rate of change in u.
$z_{\min}$, $z_{\max}$	zmin, zmax	Bounds on z.
Q, S, S_ψ, s_ψ	Q, S, Spsi, spsi	Weight matrices.

Table 4.1: Basic MPC variables initialized in SceneX.m

Additionally, this file introduces the following:

- The timevector `t`, which specifies timesteps in terms of the starttime `t0`, the sampling time `Ts`, and the ending time `tf`.
- A vector `mealsv`, which defines at which points in time a meal is consumed.
- `sigmaw` and `sigmav`, which is used to scale the simulated noise, and the `seed` parameter, which defines what set of random numbers to be used.
- `feed` determines if feedforward should be used. This will be described in details later.

4.2 MPCControl.m

This is a wrapper script, which simply calls the three scripts needed for the different phases of MPC: Design, simulation and evaluation.

4.2.1 MPCDesign.m

In this design file, everything is prepared for the actual simulation. This mainly consists of setting up the many matrices found in Chapter 3.

First the function ParameterDesign.m is called, which assigns the basic model parameters, shown in Table 5.1, to a vector p, to make handling of these easier. Several new variables are initialized;

Parameters(s)	MATLAB name(s)	Describtion
∞	big	Large number which acts as infinity.
x_s, u_s, d_s, z_s	xs, us, ds, zs	Steady-state point for the system.
x_0, u_0	x0, u0	Starting points for the simulation.
R, D	R, D	Vectors of lenght N.
x, u, y, $\bar{U}$	x, u, y, ubar	Initialized to zero.
$\bar{A}$, $\bar{B}$, $\bar{C}$, $\bar{D}$	Ac, Bc, Cc, Ec	System matrices in continous time.

Table 4.2: Basic MPC variables initialized in MPCControl.m

MPCDesign also performs the following operations;

- The constraints for u and z are transformed into deviation constraints.
- The vectors w and v are generated by the randn command and scaled, so they have the desired standard deviation. The seed option is used so it is possible to reproduce the same noise vector.
- DesignConstraints.m assigns the constraints for u, Δu, z, and $\bar{U}$ to vectors, i.e. $U_{\min} = \begin{bmatrix} u_{\min} & u_{\min} & \cdots & u_{\min} \end{bmatrix}^{\mathrm{T}}$, etc.
  See equations (3.38), (3.40), (3.43), and (3.52).
- DesignDiscreteMatrices.m converts the system matrices $\bar{A}$, $\bar{B}$, $\bar{C}$, $\bar{E}$ into the equivalent matrices in discrete time, A, B, C, E. This is done by equation (2.9).
- The Kalman Filter matrices is designed in DesignKalman.m. See section 3.3.
- DesignMPCMatrices.m designs the matrices $\bar{H}$, Γ, Γ_d, Φ, M_{x0}, M_{u-1}, M_R, M_D, Λ, $\bar{s}_\psi$. These are all created by basic MATLAB matrix operations.
- Finally xp is formed, which is the augmented system with integrated disturbance, corresponding to equation (3.68).

4.2.2 MPCSimulate.m

Now the actual simulation is started. This consists of a loop which runs over the timesteps defined in t. First noise is simulated on model x and measurement y, by adding values of vectors w and v. Meals are simulated by adjusting $D_m(t)$, and are assumed to span only one timestep, that is from t_k to t_{k+1}. For instance, a meal at $t = 420$ min (7 hours) is simulated by adding the size of the meal to $D_m(420)$, see Figure 4.2.

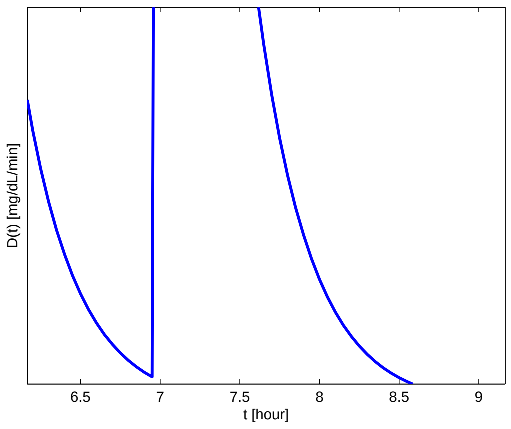
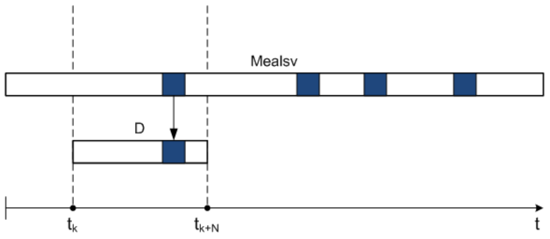
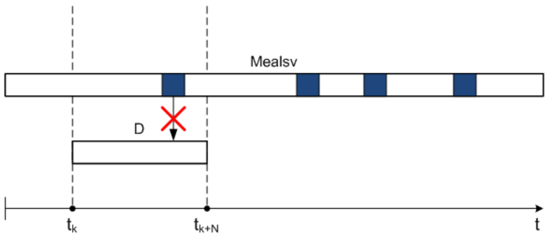


Figure 4.2: Modelling meals, change in D_m

On Figure 4.3 is shown how the implementation treats D when feedforward is used and when it isn't. See Section 3.1.3 for further details of these.



(a) Feedforward in use, D is updated with meals



(b) Feedforward not in use, D is kept to all-zeros at all times

Figure 4.3: Implementation of feedforward

Now comes the actual control step, where the optimal u is calculated for this timestep, this is done in `MPCCompute.m`. Below is the outline of `MPCCompute.m`;

- First y, u_{-1}, R and D are transformed into deviation variables and the current prediction for x is updated with the Kalman gain `Kfx`.
- The gradient g is updated.
- Matrix system (3.56) is formed.
- Starting guess for $\bar{U}$ is made by forming $\bar{U}_{\text{init}} = [u_k \quad u_{k+1} \quad \cdots \quad u_{k+N-1} \quad u_{k+N-1} \quad \cdots \quad \phi_k \quad \phi_{k+1} \quad \cdots \quad \phi_{k+N-1} \quad \phi_{k+N-1}]^{\mathrm{T}}$.
- Then $\bar{U}$ is found by solving the QP problem (3.57) by MATLAB's `quadprog` or John Bagterp Jørgensens `qpsolver.m`, where `qpsolver.m` has shown itself to greatly outperform `quadprog`.
  The first element of $\bar{U}$ is used as u_k. If the solver fails, u_{k-1} is used as u_k.
- `xp` is updated.

- The found u is returned as a physical variable (as opposed to deviation variable).

The program now returns to `MPCSimulate.m`, where x_{k+1} is found by evaluating the Bergman model using the MATLAB Ordinary Differential Equation solver `ode15s`.

4.2.3 `MPCPlot`

After the simulation-loop is finished the results are plotted in convenient ways. The results include the state vector x, the calculated input u and the meal disturbances D_m. Plots are made with x-axis in hours, and constraints are shown where applicable.

Chapter 5

Case study - A “minimal model”

In this case study, the Bergman “minimal model” will be used with the implemented MPC.
Based on measurements of the glucose level in the subcutaneous layer, the controller will calculate the optimal amount of insulin to inject into the patient.
To model the patient, a modified version of Bergmans “minimal model” is used. As the name implies, this model is small, and is mainly used to test the MPC. It would be unrealistic to think of it as an accurate model of a real patient.
The model consists of five differential equations, (see [2] for further details):

$$\frac{\mathrm{d}G}{\mathrm{d}t} = -P_1(G + G_b) - X_r G + D_m(t) \tag{5.1}$$

$$\frac{\mathrm{d}X_r}{\mathrm{d}t} = -P_2 X_r + P_3(I - I_b) \tag{5.2}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = -nI + \frac{U(t)}{V_I} \tag{5.3}$$

$$\frac{\mathrm{d}G_{sc}}{\mathrm{d}t} = \frac{G - G_{sc}}{5} - R_{utln} \tag{5.4}$$

$$\frac{\mathrm{d}D_m}{\mathrm{d}t} = -\alpha D_m(t) \tag{5.5}$$

A very brief description of the five states;

- G (mg/dL): Blood plasma glucose concentration above basal value.
- X_r (mU/L): Insulin in the remote compartment.
- I (mU/L): Plasma insulin concentration above basal value.
- G_{sc} (mg/dL): Glucose concentration on the subcutaneous layer. This state approximates G, and is the one which are measurable.
- D_m (mg/dL/min): Meal glucose disturbance.

and the input is the manipulated insulin infusion rate ($U(t)$, mU/min). The time variable t is measured in minutes. The standard parameters for the model can be found in Table 5.1. $G_b, X_{br}, I_b, G_{bsc}$ and D_m denote the basal values for the system.
The parameters are assumed to be optimal, so they will not be inspected in this project.

Name	Value
P_1	$0.028735\ \text{min}^{-1}$
P_2	$0.028355\ \text{min}^{-1}$
P_3	$5.035 \cdot 10^{-5}$ mU/L
n	$5/54\ \text{min}^{-1}$
V_I	12 L
R_{utln}	0.7400mg/dL/min
α	0.05
G_b	81.3 mg/dL
X_{br}	0
I_b	15mU/L
G_{bsc}	$G_b - 5\,R_{utln}$
D_{bm}	0

Table 5.1: Bergman model parameter values

Boundaries on the blood sugar level is needed to avoid the person going into hyperglycemia or hypoglycemia. These limits corresponds to output constraints. The input constraints restricts how much, and how fast, the insulin can be injected, and ensures the system obeys physiological and physical limits.
The chosen constraints are shown below, respectively;

$$60\ \text{mg/dL} \leq z \leq 180\ \text{mg/dL} \tag{5.6}$$
$$0\ \text{mU/min} \leq u \leq 100\ \text{mU/min} \tag{5.7}$$
$$-16.7\ \text{mU/min} \leq \Delta u \leq 16.7\ \text{mU/min} \tag{5.8}$$

Neither these will be investigated during this project.
To use the equations in MPC, let u and d be the inputs $U(t)$ and $D(t)$, and let x be the system of equations (5.1-5.5);

$$x = \begin{bmatrix} G \\ X_r \\ I \\ G_{sc} \\ D \end{bmatrix} \quad \Rightarrow \quad \dot{x} = \begin{bmatrix} \dot{G} \\ \dot{X}_r \\ \dot{I} \\ \dot{G}_{sc} \\ \dot{D}_m \end{bmatrix} \tag{5.9}$$

A steady-state point for the system is given by;

$$x_s = \begin{bmatrix} G_b \\ X_{br} \\ I_b \\ G_{bsc} \\ D_{bm} \end{bmatrix}, \qquad u_s = n\, I_b\, V_I, \qquad d_s = 0 \tag{5.10}$$

This model will now be used with MPC.
First step is to linearize the model.

5.1 Linearization

By using the procedure shown in Chapter 2, it's clear that this system of differential equations can be set up like this;

$$\dot{X} = \bar{A}X + \bar{B}U + \bar{E}D \tag{5.11}$$

where X, U and D are deviation variables, $X = x - x_s$, $U = u - u_s$ and $D = d - d_s$.
In this case, X is the state vector, U is the input variable for insulin injection, and D is the input variable for meal consumption. The matrices $\bar{A}$, $\bar{B}$ and $\bar{E}$ are the partial derivatives of the model:

$$\bar{A} = \left.\frac{\partial f}{\partial x}\right|_{(x_s,u_s,d_s)}, \quad \bar{B} = \left.\frac{\partial f}{\partial u}\right|_{(x_s,u_s,d_s)}, \quad \bar{E} = \left.\frac{\partial f}{\partial d}\right|_{(x_s,u_s,d_s)} \tag{5.12}$$

which for this system gives;

$$\bar{A} = \begin{bmatrix} -P_1 - X_b & -G_b & 0 & 0 & 1 \\ 0 & -P_2 & P_3 & 0 & 0 \\ 0 & 0 & -n & 0 & 0 \\ 0.2 & 0 & 0 & -0.2 & 0 \\ 0 & 0 & 0 & 0 & -\alpha \end{bmatrix} \tag{5.13}$$

$$\bar{B} = \begin{bmatrix} 0 & 0 & \frac{1}{V_I} & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{5.14}$$

$$\bar{E} = \begin{bmatrix} 1 & 0 & 0 & 0 & -\alpha \end{bmatrix}^{\mathrm{T}} \tag{5.15}$$

Converting to discrete time yields (referring to Section 2.1);

$$X_{k+1} = AX_k + BU_k + ED_k \tag{5.16}$$

By using parameter values from Table 5.1, the matrices A, B and E can be evaluated by equation (2.9). These matrices are not evaluated here since they depend on the used sampling time T_s.

The model is now linearized and converted to discrete time, and ready to be used with the MPC controller.

5.2 MPC

The goal for the controller is to keep the glucose level, for the diabetic, at a healthy level at all times.
The sensor in the insulin pump is able to measure the glucose on the subcutaneous layer, that is, the state G_{sc}. Since this is the only state which is measureable, the vector C is given by $C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix}$. The setpoint for this state is selected as it's basal value, G_{bsc}.
The constraints used are giving in equations (5.7-5.8), and it should be noted that the output constraints are implemented as soft constraints.

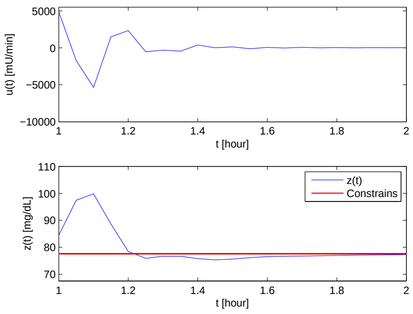
5.2.1 Weight matrices

There are four weigths used in the objective function: Q_z, S_ψ, s_ψ and S. These can be chosen to "tune" the controller.
S_ψ and s_ψ are weights for the slack variable ψ_k, which are property of the soft output constraints, and are kept constant as identity matrices.
Q_z and S matrices are weights for the output and regularization term, respectively. Tuning can be done by keeping one of these constant while varying the other, since it's the ratio of these that actually matters in the objective function. Q_z is selected as identity matrix, and S is varied, see Figures 5.1 and 5.2. The patient is given a meal at $t = 1$ hour.
For the very low values for S, there is close to no regularization, which means that u varies greatly between timesteps. If the value of S is set too high, the controller responds too slow. S is selceted to 10^{-2}, since this values gives some regularization and is still low enough to provide a decent response from the controller in regards of speed.

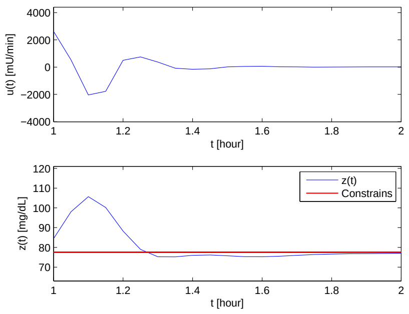
It would also be possible to tune the Kalman Filter, in this thesis the weights are selected as; $Q_w = 10^{-2}$ and $Q_\xi = 10^{-2}$.

The seleted weights are summarized below;

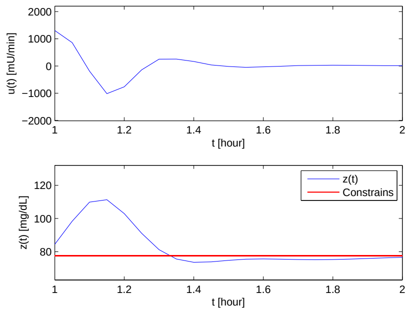
$$Q_z = 1, \quad S = 10^{-2}, \quad S_\psi = 1, \quad s_\psi = 1, \quad Q_w se = 10^{-2}, \quad Q_\xi = 10^{-2} \tag{5.17}$$



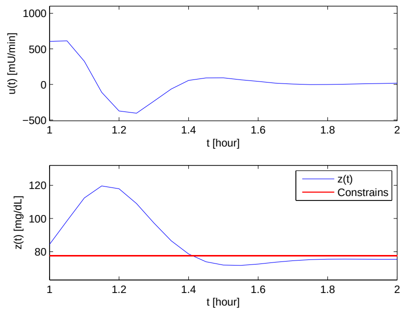
(a) $S = 10^{-5}$



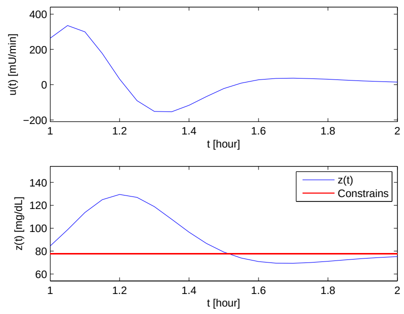
(b) $S = 10^{-4}$



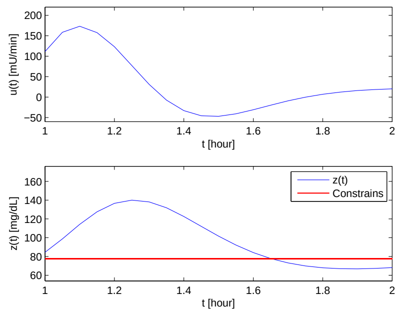
(c) $S = 10^{-3}$



(d) $S = 10^{-2}$

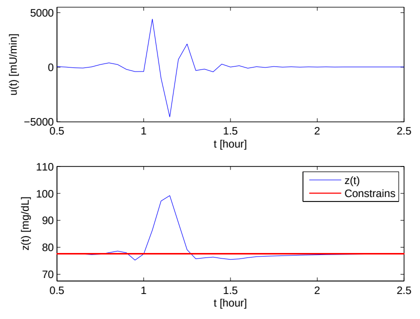


(e) $S = 10^{-1}$

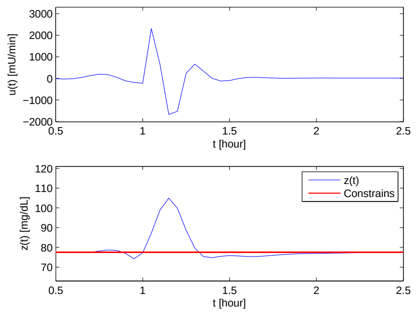


(f) $S = 1$

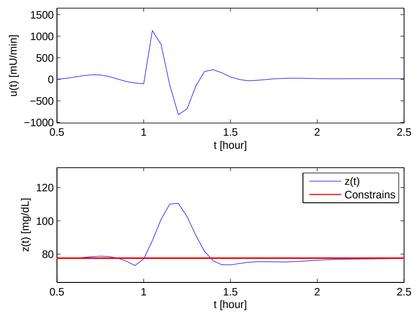
Figure 5.1: Weight matrix S varied, without feedforward



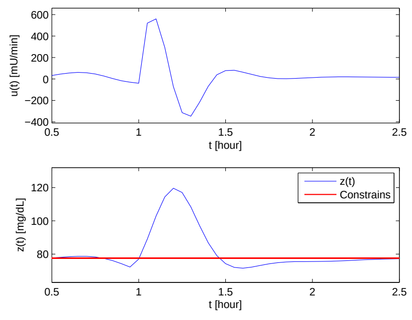
(a) $S = 10^{-5}$



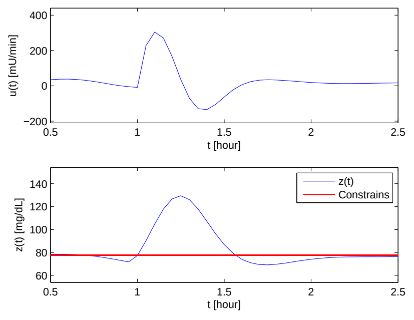
(b) $S = 10^{-4}$



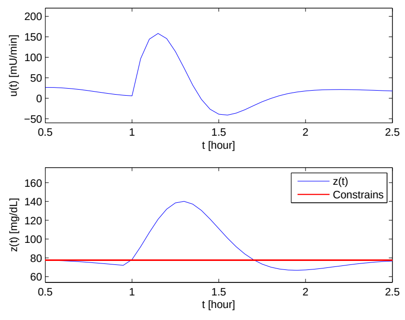
(c) $S = 10^{-3}$



(d) $S = 10^{-2}$



(e) $S = 10^{-1}$



(f) $S = 1$

Figure 5.2: Weight matrix S varied, with feedforward

5.2.2 Horizon and sampling time

An appropriate horizon N, and a sampling time T_s, are now selected. This is done by investigating how a stepchange in the meal-disturbance, D_m, and in $u(t)$ will effect the non-linearized system. See Appendix B.15 for the implementation of this.

Figure 5.3 illustrates what is meant by stepchange.

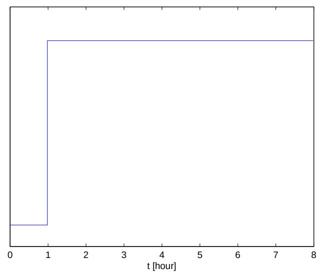


Figure 5.3: Stepchange

The effect of this stepchange on the output z is seen on Figure 5.4. This Figure shows that, for the stepchange in D_m, the system reaches a new steady-state after approximately three hours (180 minutes), which is slightly faster than for the stepchange in u.
A general rule-of-thumb says, that one should use a sampling time of about 180 min/12 samples = 15 min. Since computational speed is not an issue for this problem , the used sampling time is $T_s = 8$ min.
The horizon should be at least three hours, so $N = 25$ is chosen.

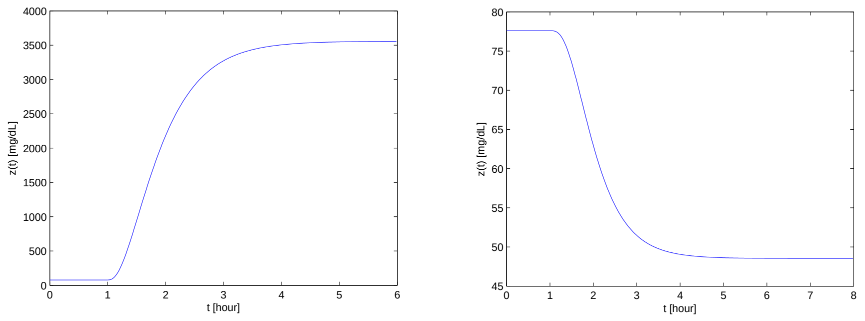


Figure 5.4: Effect of stepchange

5.2.3 Noise

As described in Section 3.3, noise can be simulated on output measurement and/or the model itself;

$$\begin{aligned}&x_{k+1} = Ax_k + Bu_k + Ed_k + w_k \Rightarrow \hat{x}_{k+1} = Ax_k + Bu_k + Ed_k\\&y_k = Cx_k + v_k \Rightarrow \hat{y}_k = Cx_k\end{aligned}$$

where w and v are both assumed to be Gaussian distributed;

$$\begin{aligned}&w \sim N(0, Q_w)\\&v \sim N(0, R_v)\end{aligned}$$

These vectors of noise are created in MATLAB by use of the `randn` function, which is able to create random numbers with mean zero and standard deviation one.
To obtain the desired distribution for the noise, the noise is scaled with σ. Experiments are required to determine the size of σ_w and σ_v. By use of the “seed” parameter for `randn`, it’s possible to reproduce the same “random” vectors of noise, and thereby produce simulations with the same noise vectors scaled differently.

By experiments, σ_w and σ_v are chosen to be;

$$\sigma_w = \begin{bmatrix}0.7 & 10^{-3} & 0.4 & 0.8 & 0\end{bmatrix}, \quad \sigma_v = 0.8 \tag{5.18}$$

These have shown themselfs to give simulations with a reasonable level of noise.

5.3 Simulations

The performance of the controller will now be tested with three different scenarios. In the first test scenario our virtual patient is given three regular meals. This is the main test scenario, and here the effect of feedforward will be investigated, as well as measurement noise and process noise. In the last two scenarios the size of the meals are varied. Scenario II models a patient eating big meals, and in Scenario III the patient ingests alot of meals throughout the day.

The main objective for the controller is to keep the patient within healthy blood sugar limits at all time.

5.3.1 Scenario I

Tabel 5.2 shows at what hours the patient ingests meals;

Time (hours)	$D(t)$ (mg/dL/min)
8	5
12	7
18	10

Table 5.2: Scenario I meal disturbances

This is the basic simulation scenario. This is now tested with and without feedforward.

5.3.1.1 Feedforward

The feedforward approach was described in Section 3.1.3. In this section the controller will be equiped with this approach, which simulates a controller where the patient informs of future incoming meals.
The simulations are done without any process or measurement noise. Figures 5.5 and 5.6 illustrates the evolution of the states and the input/output, respectively, without feedforward, while Figures 5.7 and 5.8 illustrates the case with feedforward.

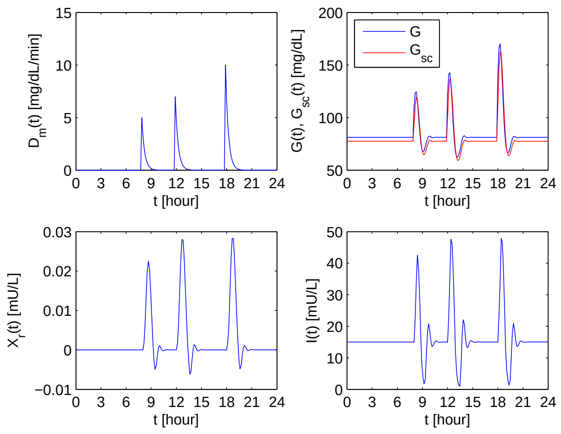


Figure 5.5: The evolution of states without feedforward

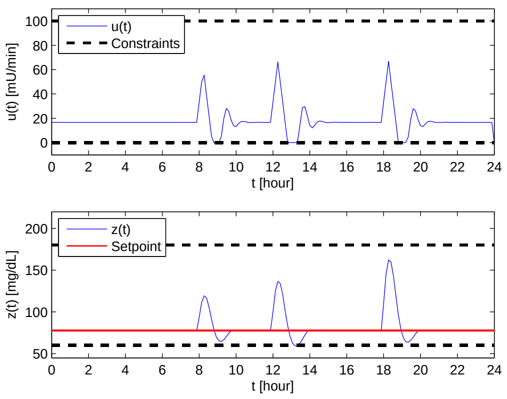


Figure 5.6: Insulin input and output without feedforward

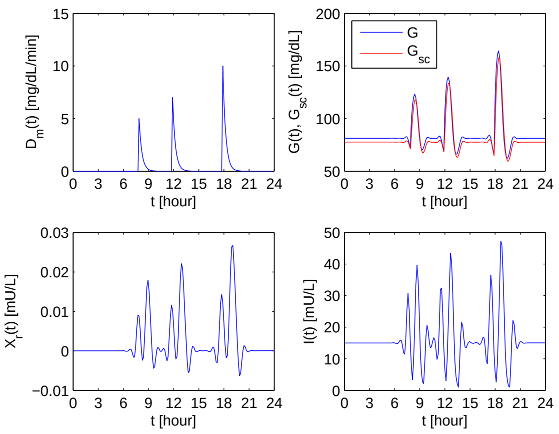


Figure 5.7: The evolution of states with feedforward

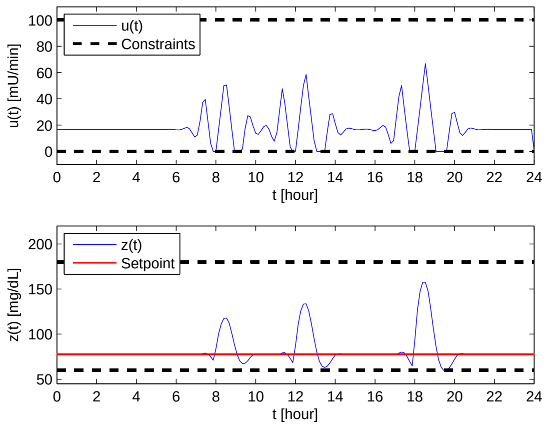


Figure 5.8: Insulin input and output with feedforward

Using feedforward, the controller knows when there is going to be an increase in the glucose value, because of meal consumption. That means that before the consumptions are made, the controller injects some insulin. The glucose level is held on a lower value than without feedforward and therefore it is more likely for the patient to maintain a heathy blood glucose level.

From now, on all simulations are done with feedforward.

5.3.1.2 Measurement noise

As mentioned in Section 5.2.3, the system can be simulated with measurement noise and/or process noise. In this section only the measurement noise, noise on y, will be used .
A way to think of measurement noise, is how well the sensor, that measure the glucose level in the subcutaneous layer, works.
Figure 5.9 illustrates the evolution of the five states for the model, Figure 5.10 the input and the outputs, while 5.11 illustrates the difference between the measured output and the real output.

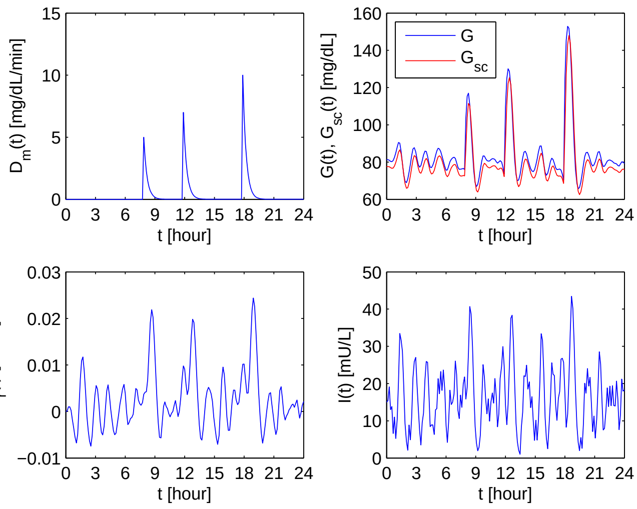


Figure 5.9: The evolution of states with measurement noise

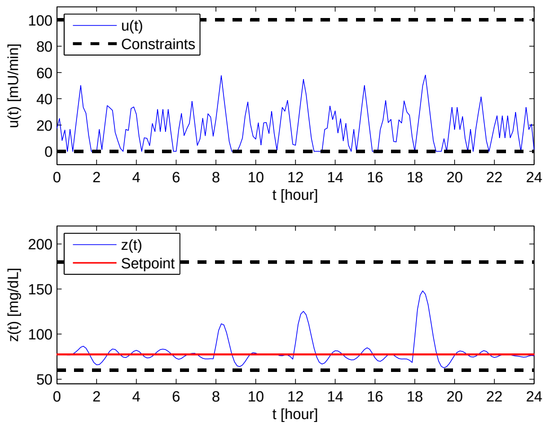


Figure 5.10: Insulin input and output with measurement noise

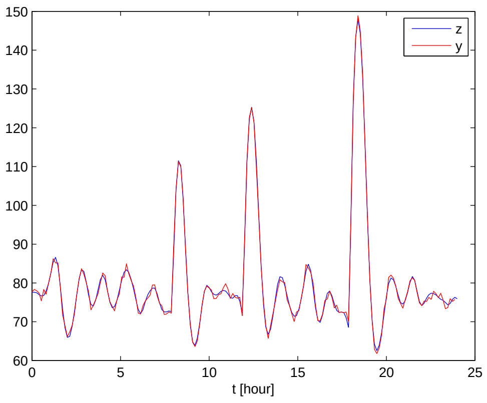


Figure 5.11: The measured and real output with measurement noise

Compared to Figures 5.7 and 5.8, Figures 5.9 and 5.10 does show some oscillations, which means the control problem is harder, though, the controller still performs nicely, as the output is kept within the given blood glucose limits. On Figure 5.11 the measured output y is compared to the real output z. It shows that these two accompanies each other pretty good, even though oscillations are present here too.

5.3.1.3 Process noise

This section will handle the process noise, that is, noise on x.
Process noise could simulate physiological aspects the model doesn't consider.

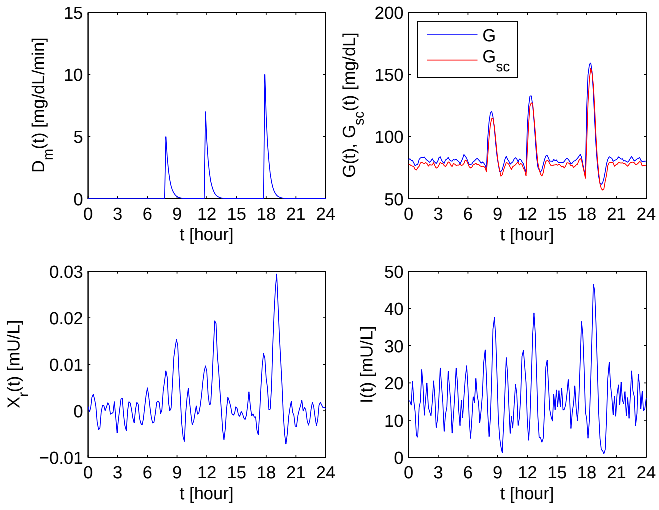


Figure 5.12: The evolution of states with process noise

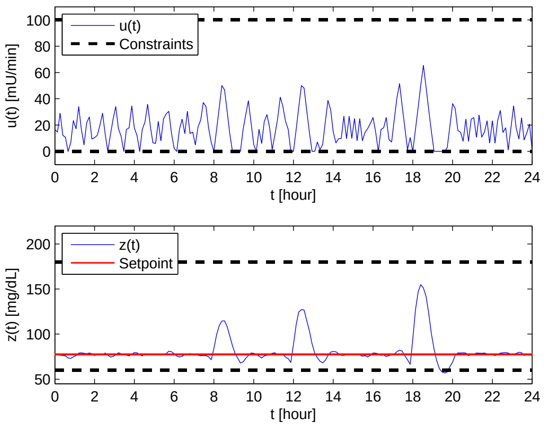


Figure 5.13: Insulin input and output with process noise

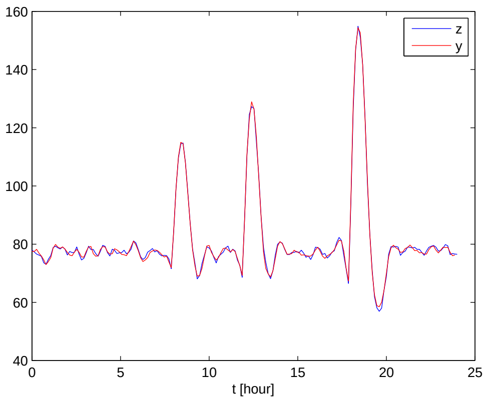


Figure 5.14: The measured and real output with process noise

Again the controller handles the noise nicely.

5.3.1.4 Measurement and process noise

Virtually every system has both measurement noise and process noise, here they are both applied to the scenario. This is expected to be the most realistic simulation, as these types of noise would be present in the physical system.

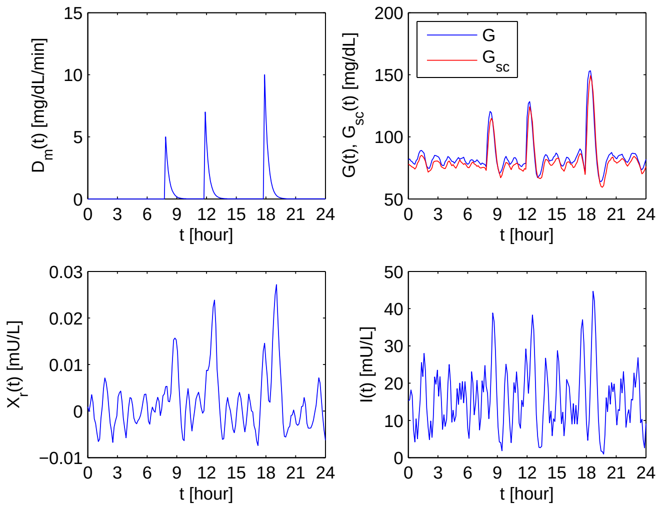


Figure 5.15: The evolution of states with process and measurement noise

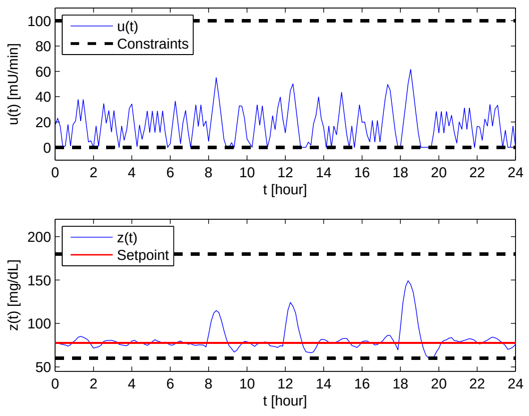


Figure 5.16: Insulin input and output with process and measurement noise

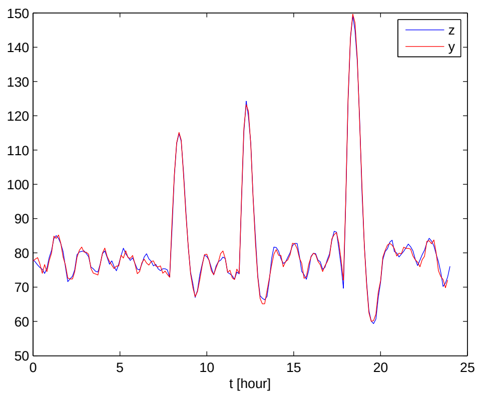


Figure 5.17: The measured and real output with process and measurement noise

5.3.2 Scenario II

In the next two scenarios, the simulations will be done with both measurement and process noise. Here the diabetic eats five big meals.

Time (hours)	$D(t)$ (mg/dL/min)
8	7
12	10
15	5
18	13
22	6

Table 5.3: Scenario II meals disturbances

Figures 5.18 and 5.19 illustrates the states and the input/output respectively.

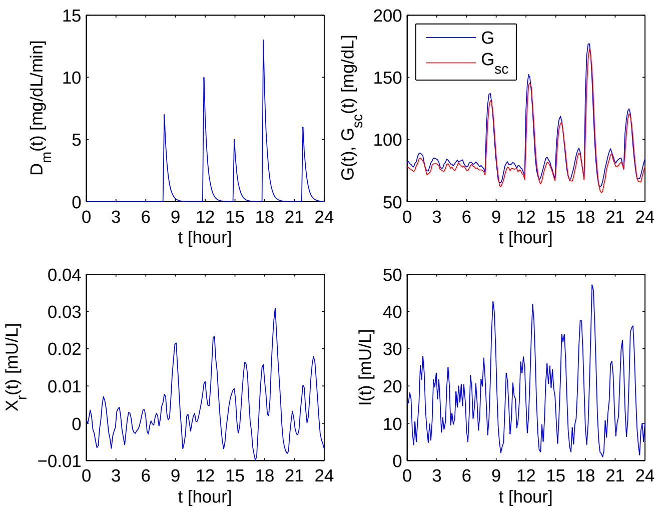


Figure 5.18: The evolution of states with process and measurement noise

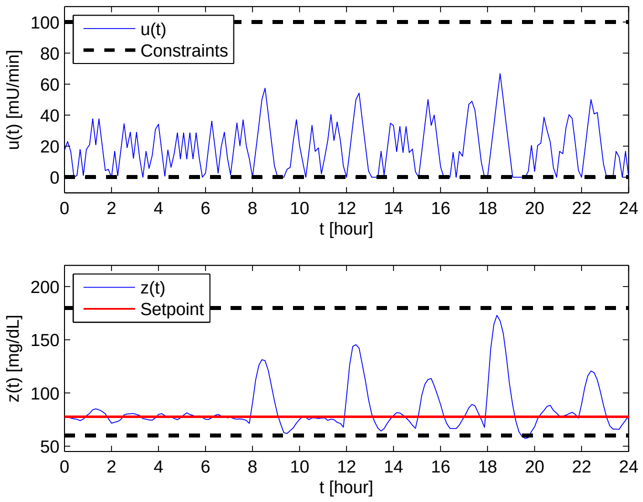


Figure 5.19: Insulin input and output

Even with these big meals, the controller manages to keep the blood sugar at a healthy quantity. Figure 5.19 does show that z comes close to both the upper and lower bound, and at $t \approx 20$ the lower bound is violated for a very short period of time. This is the effect of soft output constraints, and is to be expected when the ingested meals are big.

5.3.3 Scenario III

In this scenario the diabetic eats meals all day long.

Time (hours)	$D(t)$ (mg/dL/min)
8	7
10	5
12	5
13	7
14	4
15	5
16	6
18	13
20	5
21	3
22	4

Table 5.4: Scenario III meals disturbances

Again, Figures 5.20 and 5.21, illustrates the states and the input/output

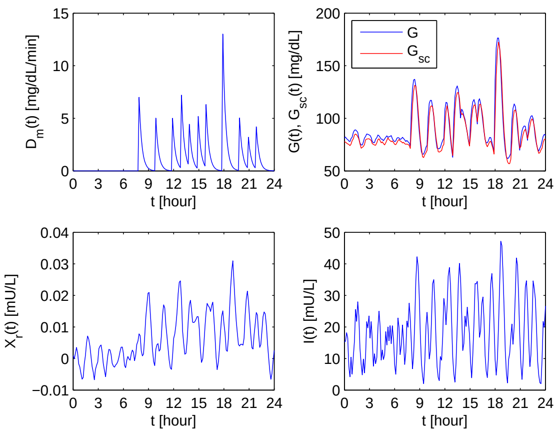


Figure 5.20: The evolution of states with process and measurement noise

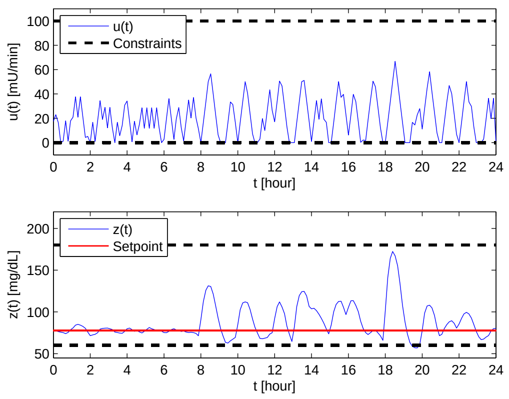


Figure 5.21: Insulin input and output

Again the controller performs good, but, like with Scenario II, the lower boundary on the output is violated briefly.

Chapter 6

Conclusion

What the simulations show, is that the MPC controller, based on Bergmans "minimal model", does a good job keeping the patient inside the blood glucose limits. This is the case for both Scenario I, where meals of ordinary size is simulated, Scenario II, where the patient is assumed to eat relatively big meals, and Scenario III, where big meals combined with many small meals are ingested.

The simulations also show, that the controller handles noise, on both the measurements and on the process, well. This is important since a practical implementation of the system should definitely be expected to contain noise.

Feedforward showed itself to be an advantage, as the controller is able to take precaution of future meals, and thereby making it more likely for the patient to stay within the healthy blood glucose limits. This can be seen by comparing Figures 5.6 and 5.8. Though it should be kept in mind, that if a controller on a real diabetic should use feedforward, it would require the patient to tell the device what hours a day he would eat, and how big the meals would be. Therefore feedforward may not be very useful in practice.

It should be kept in mind that the model used for simulations is a "minimal model", so what can be concluded is that there is a change the MPC would work for the insulin pump case, but the model used is too inadequate to draw any definite conclusion. Another reason why MPC wouldn't be used is seen on Figure 5.19 for Scenario II at $t \approx 20$, where the the effect of soft constraints reveals itself. The amount of blood glucose slightly violates it's lower boundary, which is potentially disastrous for the patient. On the other hand, soft constraints are needed to make sure the control problem stay feasible.

MPC is an interesting tool to use as controller in an insulin pump, but the model of the patient must be improved to make it really useful.

Appendix A

Impulse-response method

The MPC method solves the control problem using a model. Instead of using a linear model, the impuls-response method and the step response could be used [1].
The idea behind these methods, is that the process can apply an impuls at any input, and then measure the response. Thereby the name impuls-response.
Related to the state-space model from the previous chapters:

$$x_{k+1} = A\,x_k + B\,u_k \tag{A.1}$$

$$y_k = C\,x_k \tag{A.2}$$

where y_k is a vector of measured outputs.
$E\,d_k$ is neglected from the equation, due the fact that we only want make an impuls at one input statement, and in this case it is u_k.
Now assuming that $x_0 = 0$ and then applying that there only is a change in the input at, t_0, such that u is an integer, $u_0 \neq 0$ and $u_k = 0$ for $k \geq 1$. I.e the vector u will look like;

$$u = \begin{bmatrix} u_0 & 0 & 0 & \cdots & 0 \end{bmatrix}^{\mathrm{T}}. \tag{A.3}$$

By using these assymptions the following outputs, z_k, at the different states, x_k, is;

$$x_0 = 0 \qquad y_0 = H_0\,u_0 = 0 \tag{A.4}$$

$$x_1 = B\,u_0 \qquad y_1 = H_1\,u_0 = C\,Bu_0 \tag{A.5}$$

$$x_2 = A\,Bu_0 \qquad y_2 = H_2\,u_0 = C\,ABu_0 \tag{A.6}$$

$$\vdots \qquad \vdots \tag{A.7}$$

$$x_k = A^{k-1}Bu_0 \qquad y_k = H_k\,u_0 = C\,A^{k-1}Bu_0 \tag{A.8}$$

Where $H_k = C\,A^{k-1}B$ is called the k'th Markov parameter, as mentioned in the previous chapters, and the impuls response sequence is indicated by H_k for

State sequence	Output sequence	Impulse response sequence	Step response sequence
$x_0 = 0$	$y_0 = 0$	$H_0 = 0$	$S_0 = 0$
$x_1 = Bu_0$	$y_1 = C\,Bu_0$	$H_1 = C\,B$	$S_1 = C\,B$
$x_2 = ABu_0$	$y_2 = C\,ABu_0$	$H_2 = C\,AB$	$S_2 = C\,AB + C\,B$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x_k = A^{k-1}Bu_0$	$y_k = C\,A^{k-1}Bu_0$	$H_k = C\,A^{k-1}B$	$S_k = \sum_{i=0}^{k-1} C\,A^iB$

Table A.1: Relation between the different sequences

$k = 0 \ldots N$.
Step responce is then the summerize of all the values of H_i in the input sequence. I.e. the step response can be defined as;

$$S_k = \sum_{i=0}^{k-1} H_i \tag{A.9}$$

The relation between the sequences is outlined in Table A;

Using these terms, there can be achived an output using a single input at a given time t_s. It is also possible to use the other inputs variable, d_k, as inputs instead of u_k.
B is replaced with E, and u_0 with d_0. The state and the Impulse response sequence would be;

$$x_0 = 0 \qquad H_0 = 0 \tag{A.10}$$
$$x_1 = E\,d_0 \qquad H_1 = C\,B \tag{A.11}$$
$$x_2 = A\,Eu_0 \qquad H_2 = C\,AB \tag{A.12}$$
$$\vdots \qquad \vdots \tag{A.13}$$
$$x_k = A^{k-1}Ed_0 \qquad H_k = C\,A^{k-1}B \tag{A.14}$$

And it's easily to find the sequences for the output, y, and the step response, S, from this.

The impuls-responce method can be used, when the MPC does not have a model to build the controller on. Then the method is used to make the Markov parameter and then solve the problem in an indirect way.

Appendix B

Matlab programs

This appendix contains the complete listing of the MATLAB scripts used.

B.1 Scena1.m

```
%Scena1.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Scenario I. Three normal meals are simulated.
%
%-----------------------------------------------------------------

clear all
close all

%Time
t0 = 0.0;                %[min] Start
Ts = 8;                  %[min] Sampling time
tf = 60*24;              %[min] End
t = (t0:Ts:tf)';         %Time vector

%Size of horizon
N = 25;

%Constraints
umin  = 0;        umax  = 100;
udmin = -16.7;    udmax = 16.7;
zmin  = 60;       zmax  = 180;

%Meals (disturbance)
mealsv = zeros(1,length(t));
mealsv(8*60/Ts) = 5;
mealsv(12*60/Ts) = 7;
mealsv(18*60/Ts) = 10;

%Noise
sigmaw = [0.7 1e-3 0.4 0.8 0];
%sigmaw = [0 0 0 0 0];
sigmav = 0.8;
%sigmav = 0;
seed = 3507;  %Choice for randn

%Use feedforward?
feed = 0; %1 = yes, 0 = no

%Weight matrices
Q = eye(1);
S = eye(1)*1e-2;
Seta = eye(1);
seta = eye(1);

%Start MPC
MPCControl
```

B.2 Scena2.m

```
%Scena2.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Scenario II. Five big meals are simulated.
%
%-------------------------------------------------------------------

clear all
close all

%Time
t0 = 0.0;                 %[min] Start
Ts = 8;                   %[min] Sampling time
tf = 60*24;               %[min] End
t = (t0:Ts:tf)';          %Time vector

%Size of horizon
N = 25;

%Constraints
umin  = 0;         umax  = 100;
udmin = -16.7;     udmax = 16.7;
zmin  = 60;        zmax  = 180;

%Meals (disturbance)
mealsv = zeros(1,length(t));
mealsv(8*60/Ts) = 7;
mealsv(12*60/Ts) = 10;
mealsv(round(15*60/Ts)) = 5;
mealsv(18*60/Ts) = 13;
mealsv(22*60/Ts) = 6;

```

```
%Noise
sigmaw = [0.7 1e-3 0.4 0.8 0];
sigmav = 0.8;
seed = 3507; %Choice for randn

%Use feedforward?
feed = 1; %1 = yes, 0 = no

%Weight matrices
Q = eye(1);
S = eye(1)*1e-2;
Seta = eye(1);
seta = eye(1);

%Start MPC
MPCControl
```

B.3 Scena3.m

```
%Scena3.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Scenario III. Many meals throughout the 24 hour period.
%
%----------------------------------------------------------------

clear all
close all

%Time
t0 = 0.0;              %[min] Start
Ts = 8;                %[min] Sampling time
tf = 60*24;            %[min] End
t = (t0:Ts:tf)';       %Time vector

%Size of horizon
N = 25;

%Constraints (bequette article)
umin  = 0;        umax  = 100;
udmin = -16.7;    udmax = 16.7;
zmin  = 60;       zmax  = 180;

%Meals (disturbance)
mealsv = zeros(1,length(t));
mealsv(round(8*60/Ts)) = 7;
mealsv(round(10*60/Ts)) = 5;
mealsv(round(12*60/Ts)) = 5;
mealsv(round(13*60/Ts)) = 7;
mealsv(round(14*60/Ts)) = 4;
mealsv(round(15*60/Ts)) = 5;
mealsv(round(16*60/Ts)) = 6;
mealsv(round(18*60/Ts)) = 13;
mealsv(round(20*60/Ts)) = 5;
mealsv(round(21*60/Ts)) = 3;
mealsv(round(22*60/Ts)) = 4;

%Noise
sigmaw = [0.7 1e-3 0.4 0.8 0];
sigmav = 0.8;
seed = 3507; %Choice for randn

%Use feedforward?
feed = 1; %1 = yes, 0 = no

%Weight matrices
Q = eye(1);
S = eye(1)*1e-2;
Seta = eye(1);
seta = eye(1);

%Start MPC
MPCControl
```

B.4 MPCControl.m

```
%MPCControl.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Wrapper file for the three MPC phases.
%
%---------------------------------------------------------------

MPCDesign
MPCSimulate
MPCPlot
```

B.5 MPCDesign.m

```
%MPCDesign.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Design phase, initializes the MPC matrices by calling
%various design files.
%
%---------------------------------------------------------------

%Retrieve parameters
p = DesignParameters();
P1 = p(1);
```

```
P2 = p(2);
P3 = p(3);
n = p(4);
VI = p(5);
Rutln = p(6);
alpha = p(7);
Gb = p(8);
Xb = p(9);
Ib = p(10);
Gbsc = p(11);
Db = p(12);

big = 10^10;             %Acts as infinity

%Steady-state values (basal values)
xs = [Gb; Xb; Ib; Gbsc; Db];
us = VI*n*Ib;
ds = Db;
zs = Gbsc;

%Starting points
x0 = xs;
u0 = us;

%Initialize variables
x = zeros(5, length(t));     x(:, 1) = x0;
u = zeros(length(t),1);      u(1) = u0;
ubar = zeros(2*N,1);         ubar(1) = u0;
y = zeros(length(t),1);

%Disturbance
D = zeros(N, 1);

%Set point
R = repmat(Gbsc, N, 1);

%Generate noise
randn('seed', seed);
w = randn(5,length(t));
for k = 1:5
    w(k,:) = w(k,:)*sigmaw(k);
end
randn('seed', seed);
v = randn(length(t),1)*sigmav;


%Transform into deviation variables
umin = umin - us; umax = umax - us;
zmin = zmin - zs; zmax = zmax - zs;

%Set up constraints
[Umin, Umax, Udmin, Udmax, Zmin, Zmax, ubarmin, ubarmax] = ...
    DesignConstraints(umin, umax, udmax, udmin, zmin, zmax, big, N);

%Matrices in continous time
Ac = [  -P1-Xb  -Gb 0   0     1; ...
        0       -P2 P3  0     0; ...
        0       0   -n  0     0; ...
        0.2     0   0   -0.2  0; ...
        0       0   0   0     -alpha];
Bc = [0; 0; 1/VI; 0; 0];
Ec = [1; 0; 0; 0; -alpha];
Cc = [0 0 0 1 0];

%Matrices in discrete time
```

```
[A,B,E,C] = DesignDiscreteMatrices(Ac,Bc,Ec,Cc,Ts);

%Kalman Filter design
[Kfx, A, B, C, E] = DesignKalman(A, B, C, E);

%MPC matrices
[Hbar,Gamma,Gammad,Phi,Mx0,Mum1,MR,MD,Lambda,barseta] = ...
    DesignMPCMatrices(A,B,E,C,Q,S,N,Seta,seta);

xp = [x0-xs; 0];
```

B.6 DesignKalman.m

```
%DesignConstraints.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Designs matrices needed for the Kalman Filter, and extends
%A,B,E and C such they contain the integrated noise.
%
%------------------------------------------------------------

function [Kfx, Ae, Be, Ce, Ee] = DesignKalman(A, B, C, E)

n = size(A,1);
m = size(C,1);

%Kalman design
Qw = eye(n,n)*1e-2;
Qxi = eye(m,m)*1e-2;
Rv = eye(m,m);
Ad = eye(m,m);
Ae = [A B; zeros(m,n) Ad];
Be = [B; zeros(m,m)];
Ee = [E; zeros(m,m)];
Ce = [C zeros(m,m)];
Qe = [Qw zeros(n,m); zeros(m, n) Qxi];

P = dare(Ae', Ce', Qe, Rv);
Re = Ce*P*Ce'+Rv;
Kfx = P*(Ce' / Re);
%keyboard
```

B.7 DesignDiscreteMatrices.m

```
%DesignConstraints.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
```

```
%File description:
%
%Converts matrices A,B,E and C into discrete time.
%
%------------------------------------------------------------

function [Ad,Bd,Ed,Cd] = DesignDiscreteMatrices(Ac, Bc, Ec, Cc, Ts)

M1 = [Ac Bc Ec; zeros(2, 7)];
M2 = expm(M1*Ts);
Ad = M2(1:size(Ac,1), 1:size(Ac,2));
Bd = M2(1:size(Ac,1), size(Ac,2)+1:size(Ac,2)+1);
Ed = M2(1:size(Ec,1), size(Ac,2)+size(Bc,2)+1:end);
Cd = Cc;
```

B.8 DesignParameters.m

```
%DesignParameters.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%----------
%File description:
%
%Contains basic parameters for the Bergman model.
%
%------------------------------------------------------------

function p = DesignParameters()

%Model parameters
P1 = 0.028735;          %[min^(-1)]
P2 = 0.028344;          %[min^(-1)]
P3 = 5.035e-5;          %[mU/L]
Gb = 81.3;              %[mg/dL]
Ib = 15;                %[mU/L]
VI = 12;                %[L]
n = 5/54;               %[min^(-1)]
Rutln = 0.74;           %[mg/dL/min]
Gbsc = Gb-5*Rutln;      %[mg/dL]
alpha = 0.05;           %[]
Xb = 0;                 %[mU/L]
Db = 0;                 %[mg/dL/min]

%Parameter vektor
p = [P1; P2; P3; n; VI; Rutln; alpha; Gb; Xb; Ib; Gbsc; Db];
```

B.9 DesignConstraints.m

```
%DesignConstraints.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
```

```
%----------
%File description:
%
%Assigns constraints for u, Delta_u, z and ubar into vectors;
%
%-----------------------------------------------------------

function [Umin, Umax, Udmin, Udmax, Zmin, Zmax, ubarmin, ubarmax] = ...
            DesignConstraints(umin, umax, udmax, udmin, zmin, zmax, big, N)


Umin = repmat(umin, N, 1);
Umax = repmat(umax, N, 1);

Udmin = repmat(udmin, N-1, 1);
Udmax = repmat(udmax, N-1, 1);

Zmin = repmat(zmin,N,1);
Zmax = repmat(zmax,N,1);

ubarmin = [Umin; zeros(N,1)];
ubarmax = [Umax; ones(N,1)*big];
```

B.10 DesignMPCMatrices.m

```
%DesignMPCMatrices.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%----------
%File description:
%
%Designs various matrices needed for the MPC algorithm.
%Credit: John Bagterp Jørgensen
%
%-----------------------------------------------------------

function [barH,Gamma,Gammad,Phi,Mx0,Mum1,MR,MD,Lambda, barseta] = ...
          DesignMPCMatrices(A,B,E,Cz,Qz,S,N,Seta,seta)

nx = size(A,2);
nu = size(B,2);
nd = size(E,2);
nz = size(Cz,1);

% Form Gamma, Gammad, Phi
Gamma = zeros(N*nz,N*nu);
Gammad = zeros(N*nz,N*nd);
Phi = zeros(N*nz,nx);

T = Cz;
kz = 0;
for k=1:N
    Gamma(kz+1:kz+nz,1:nu) = T*B;
    Gammad(kz+1:kz+nz,1:nd) = T*E;
    T = T*A;
    Phi(kz+1:kz+nz,1:nx) = T;
    kz = kz+nz;
end
```

```

for k=2:N
     Gamma((k-1)*nz+1:end,(k-1)*nu+1:k*nu ) = Gamma(1:(N+1-k)*nz,1:nu);
     Gammad((k-1)*nz+1:end,(k-1)*nd+1:k*nd) = Gammad(1:(N+1-k)*nz,1:nd);
end

% Form QZ
QZ = zeros(N*nz,N*nz);
kz = 0;
for k=1:N
     QZ(kz+1:kz+nz,kz+1:kz+nz) = Qz;
     kz = kz+nz;
end

% Form HS
HS = zeros(N*nu,N*nu);

if N == 1
    HS = S;
else
     k=0;
     HS(1:nu,1:nu) = 2*S;
     HS(1+nu:nu+nu,1:nu) = -S;

     for k=1:N-2
          ku = k*nu;
          HS(ku-nu+1:ku,ku+1:ku+nu) = -S;
          HS(ku+1:ku+nu,ku+1:ku+nu) = 2*S;
          HS(ku+nu+1:ku+2*nu,ku+1:ku+nu) = -S;
     end

     k=N-1;
          ku = k*nu;
          HS(ku-nu+1:ku,ku+1:ku+nu) = -S;
          HS(ku+1:ku+nu,ku+1:ku+nu) = S;
end


% Form Mum1
Mum1 = [-S; zeros((N-1)*nu,nu)];

barSeta = eye(N)*Seta;
barseta = ones(N,1)*seta;

% Form H,Mx0,MR,MD
T = Gamma'*QZ;
H = T*Gamma + HS;
H = (H+H')/2;
Mx0 = T*Phi;
MR = -T;
MD = T*Gammad;

% barH, barg

z = zeros(N,N);
barH = [H z; z barSeta];

% Form Lambda
Lambda = zeros((N-1)*nu,N*nu);
T = [-eye(nu,nu) eye(nu,nu)];

for k=1:N-1
    Lambda((k-1)*nu+1:k*nu,(k-1)*nu+1:(k+1)*nu) = T;
end
```

B.11 MPCSimulate.m

```
%MPCPlot.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Simulation phase, loops over each timestep and finds the optimal
%insulin u, and uses this as input to the Bergman model.
%
%-----------------------------------------------------------------

%Iterate
for k=1:length(t)-1
    display(k)

    %--- Noise ---
    %Measurement noise
    y(k) = x(4,k) + v(k);

    %Model noise
    x(1,k) = x(1,k) + w(1,k);
    x(2,k) = x(2,k) + w(2,k);
    x(3,k) = x(3,k) + w(3,k);
    x(4,k) = x(4,k) + w(4,k);
    %--- Disturbance ---
    %Meal disturbance at this point in time
    x(5,k) = x(5,k) + mealsv(k);

    %Without feedforward
    if feed == 0
        % Do nothing
    %With Feedforward
    elseif feed == 1
        n = length(t)-1-k;
        if N <= n;
            %Assign future meals to D
            D(1:N) = mealsv(k:k+N-1);
        else
            %Simulation is ending, assign zeros to D
            D = zeros(N,1);
            D(1:n) = mealsv(k:k+n-1);
        end
    else
        display('Fejl i feed parameter')
        return;
    end

    %Dummy statement to handle u0
    if k == 1
        un = u0;
    else
        un = u(k-1);
    end

    %MPC step
    [unow,xp,info] = MPCCompute(y(k),un,R,D,Zmin,Zmax, ...
                                zs,zs,us,ds,xp,ubar, ...
                                Hbar,Mx0,Mum1,MR,MD,barseta, ...
                                Phi, Gammad, Gamma, Lambda, big, ...
```

```
                                        umin, umax, udmin, udmax, ...
                                        ubarmin, ubarmax , ...
                                        Udmin, Udmax, ...
                                        Kfx, ...
                                        A, B, E, C, N);


    %Check for error
    if info == 1
        display('Error in QP solver!');
        return
    end

    %Simulate step
    [T,X] = ode15s(@BergmanMinimalModel, [t(k) t(k+1)], x(:, k), ...
                                                                    [], p, unow);

    % Remember variables
    x(:, k+1) = X(end, :);
    u(k) = unow;
end
```

B.12 MPCCompute.m

```
%MPCCompute.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%----------
%File description:
%
%Calculates the insulin input for this timestep, u0.
%
%------------------------------------------------------------

function [u0,xp,info] = MPCCompute(y,um1,R,D,Zmin,Zmax, ...
                                    ys,zs,us,ds,xp,ubar, ...
                                    Hbar,Mx0,Mum1,MR,MD,sbareta, ...
                                    Phi, GammaD, Gamma, Lambda, big, ...
                                    umin, umax, dumin, dumax, ...
                                    ubarmin, ubarmax , ...
                                    dUmin, dUmax, ...
                                    Kfx, ...
                                    A, B, E, C, N)

nu = length(um1);
nd = length(ds);
nz = length(zs);
%Form deviation variables
dy = y - ys;
dum1 = um1-us;
dR = R-repmat(zs, N, 1);
dD = D-repmat(ds, N, 1);
dd = dD(1:nd, 1);

%Kalman Filter
e = dy - C*xp;
x0 = xp + Kfx*e;

```

```
%Update gradient
g = Mx0*x0 + MR*dR + MD*dD + Mum1*dum1;
gbar = [g; sbareta];

%Create bounds for ubar
ubarmin(1:nu,1) = max(umin,dumin+dum1);
ubarmax(1:nu,1) = min(umax,dumax+dum1);

%Create bounds for Z
c = Phi*x0 + GammaD*dD;
Zbarmin = Zmin - c;
Zbarmax = Zmax - c;

%Setup matrix system for boundaries
bmin = [dUmin; -big*ones(N, 1); Zbarmin];
bmax = [dUmax; Zbarmax; big*ones(N,1)];
Abar = [Lambda zeros(N-1, N); Gamma -eye(N,N); Gamma eye(N,N)];

%Create startguess
ubarinit = [ ubar(nu+1:N*nu,1); ubar((N-1)*nu+1:N*nu,1); ...
             ubar(nu*N+nz+1:end,1); ubar(nu*N+(N-1)*nz+1:end) ];

% ---------------------------------------------------------------------
% QPSOLVER
% ---------------------------------------------------------------------
    [ubar,info]=qpsolver(Hbar,gbar,ubarmin,ubarmax,Abar,bmin,bmax,ubarinit);

    if info == 0
        du0 = ubar(1:nu,1);
    else
       du0 = dum1;
    end
% ---------------------------------------------------------------------

% ---------------------------------------------------------------------
% QUADPROG
% ---------------------------------------------------------------------
%     Abar2 = [Abar; -Abar];
%     bbar = [bmax; -bmin];
%
%     [ubar,feval,EXITFLAG] = ...
%             quadprog(Hbar,gbar,Abar2,bbar,[],[],ubarmin,ubarmax,ubarinit);
%
%     if EXITFLAG == 1
%         du0 = ubar(1:nu,1);
%         info = 0;
%     else
%        du0 = dum1;
%        info = 1;
%     end
% ---------------------------------------------------------------------


% Update Kalman Filter
xp = A*x0 + B*du0 + E*dd;

% Form physical variable
u0 = du0 + us;
```

B.13 BergmanMinimalModel.m

```
%BergmanMinimalModel.m
```

```
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Implementation of the Bergman ''minimal model'', which consists of five
%differential equations.
%
%----------------------------------------------------------------------

function xdot = BergmanMinimalModel(t,x,p,u)

xdot = zeros(5,1);

G = x(1,1);
X = x(2,1);
I = x(3,1);
Gsc = x(4,1);
D = x(5,1);

P1 = p(1);
P2 = p(2);
P3 = p(3);
n = p(4);
VI = p(5);
Rutln = p(6);
alpha = p(7);
Gb = p(8);
Xb = p(9);
Ib = p(10);
Gbsc = p(11);
Db = p(12);

xdot(1,1) = -P1*(G-Gb) - X*G + D;
xdot(2,1) = -P2*X + P3*(I-Ib);
xdot(3,1) = -n*I + u/VI;
xdot(4,1) = (G-Gsc)/5 - Rutln;
xdot(5,1) = -alpha*D;
```

B.14 MPCPlot.m

```
%MPCPlot.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Evaluation phase, plots the simulations.
%
%----------------------------------------------------------------------

%t should be in hours
t = t/60;

```

```
%Plot the five states
figure;
subplot(221);
plot(t,x(5,:));
ylabel('D_{m}(t) [mg/dL/min]');
xlabel('t [hour]');
V = axis; axis([0 tf/60 V(3) V(4)])
set(gca, 'XTick', 0:3:24)

subplot(222);
plot(t,x(1,:),'b',t,x(4,:),'r');
legend('G','G_{sc}','Location', 'NorthWest');
ylabel('G(t), G_{sc}(t) [mg/dL]');
xlabel('t [hour]');
V = axis; axis([0 tf/60 V(3) V(4)])
set(gca, 'XTick', 0:3:24)

subplot(223);
plot(t,x(2,:));
ylabel('X_{r}(t) [mU/L]');
xlabel('t [hour]')
V = axis; axis([0 tf/60 V(3) V(4)])
set(gca, 'XTick', 0:3:24)

subplot(224);
plot(t,x(3,:));
ylabel('I(t) [mU/L]');
xlabel('t [hour]')
V = axis; axis([0 tf/60 V(3) V(4)])
set(gca, 'XTick', 0:3:24)

%Plot input and output
figure
subplot(211);
plot(t, u, 'b')
ylabel('u(t) [mU/min]');
xlabel('t [hour]')
hold on
stairs(t, repmat(umin+us, length(t)), '--k', 'linewidth', 2);
stairs(t, repmat(umax+us, length(t)), '--k', 'linewidth', 2);
legend('u(t)','Constraints','Location', 'NorthWest');
V = axis; axis([0 tf/60 V(3)-10 V(4)*1.1])
set(gca, 'XTick', 0:2:24)

subplot(212);
plot(t, x(4, :), 'b');
hold on
plot(t, repmat(Gbsc, length(t)), 'r', 'linewidth', 1);
plot(t, repmat(zmin+zs, length(t)), '--k', 'linewidth', 2);
plot(t, repmat(zmax+zs, length(t)), '--k', 'linewidth', 2);
xlabel('t [hour]')
ylabel('z(t) [mg/dL]');
legend('z(t)', 'Setpoint','Location', 'NorthWest');
V = axis; axis([0 tf/60 V(3)*0.9 V(4)*1.1])
set(gca, 'XTick', 0:2:24)


%For S investigation
% st = 0.5;
% en = 2.5;
% figure
% subplot(211);
% plot(t, u, 'b')
% ylabel('u(t) [mU/min]');
% xlabel('t [hour]')
% hold on
% V = axis; axis([st en V(3)-10 V(4)*1.1])
% %set(gca, 'XTick', 0:1:24)
%
% subplot(212);
% plot(t, x(4, :), 'b');
% hold on
% plot(t, repmat(Gbsc, length(t)), 'r', 'linewidth', 1);
% xlabel('t [hour]')
% ylabel('z(t) [mg/dL]');
% legend('z(t)','Constrains','Location', 'NorthEast');
% V = axis; axis([st en V(3)*0.9 V(4)*1.1])


% y vs z
figure
plot(t, x(4, :),'b')
hold on
plot(t(1:end-1),y(1:end-1),'r')
legend('z','y')
xlabel('t [hour]')
```

B.15 InvestSampling.m

```
%InvestSampling.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
%-----------
%File description:
%
%Scenario where a stepchange in either u or D is specified,
%and the sampling time is investigated by looking at the
%resulting graphs. See thesis for details.
%
%--------------------------------------------------------------

clear all
close all

p = DesignParameters();
P1 = p(1);
P2 = p(2);
P3 = p(3);
n = p(4);
VI = p(5);
Rutln = p(6);
alpha = p(7);
Gb = p(8);
Xb = p(9);
Ib = p(10);
Gbsc = p(11);
Db = p(12);

t0 = 0.0;
Ts = 1;
tf = 60*24;
t = t0:Ts:tf;

```

```
xs = [Gb; 0; Ib; Gbsc; 0];
us = n*Ib*VI;

x0 = xs;
u0 = us;

N = length(t);

u = zeros(1,N);
u(1:end) = us;
u(60:end) = us + 10;

x = zeros(5,N);
x(:, 1) = x0;

mealsv = zeros(length(t))*4;
%mealsv(1:60) = 0;

for k = 1: N-1
    display(k);

    x(5,k) = x(5,k) + mealsv(k);

    [T, X] = ode15s(@BergmanMinimalModel,[t(k) t(k+1)],x(:,k),[],p,u(k));

    x(:, k+1) = X(end, :)';
end

figure
en = 8*60;
t = t /60;

plot(t(1:en), x(4, 1:en), 'b');
ylabel('z(t) [mg/dL]');
xlabel('t [hour]');
hold on
set(gca, 'XTick', 0:1:en/60)

figure
stairs(t(1:en), mealsv(1:en))
xlabel('t [hour]');
ylabel('D(t) [mg/dL/min]');
set(gca, 'XTick', 0:1:24)
set(gca, 'YTick', []);
V = axis; axis([V(1) V(2) V(3)-0.5 V(4)+0.5])

figure
stairs(t(1:en), u(1:en))
xlabel('t [hour]');
set(gca, 'XTick', 0:1:24)
set(gca, 'YTick', []);
V = axis; axis([V(1) V(2) V(3)-0.5 V(4)+0.5])
```

B.16 InvestWeights.m

```
%InvestWeights.m
%
%Matias Sørensen s042300 and Simon Kristiansen s042264
%Technical University of Denmark
%Spring 2007
%B.Sc. Thesis - Model Predictive Control for an Artificial Pancreas
%
```

```
%————————
%File description:
%
%Weight matrix S varied.
%
%———————————————————————————————————————————

clear all
close all

%Time
t0 = 0.0;               %[min] Start
Ts = 3;                 %[min] Sampling time
tf = 60*24;             %[min] End
t = (t0:Ts:tf)';        %Time vector

%Size of horizon
N = 25;

%Constraints (bequette article)
big = 10^10;
% umin  = 0;        umax  = 100;
% udmin = -16.7;    udmax = 16.7;
% zmin  = 60;       zmax  = 180;

umin  = -big;       umax  = big;
udmin = -big;       udmax = big;
zmin  = -big;       zmax  = big;

%Meals (disturbance)
mealsv = zeros(1,length(t));
mealsv(1*60/Ts) = 10;

%Noise
noisefact = 0;  %Standard deviation of noise
seed = 3501;    %Choice for randn

%Use feedforward?
feed = 1; %1 = yes, 0 = no

%Weight matrices
Q = eye(1);
S = eye(1)*1e-1;
Seta = eye(1);
seta = eye(1);

%Start MPC
MPCControl(t, Ts, tf, umin, umax, udmin, udmax, zmin, zmax, ...
            mealsv, noisefact, Q, S, Seta, seta, N, seed, feed)
```

Bibliography

[1] Jan M. Maciejowski: *Predictive Control with Constraints*, Prentice Hall, 2001

[2] Sandra M. Lynch, B. Wayne Bequette: *Model Predictive Control of Blood Glucose in Type 1 Diabetics Using Subcutaneous Glucose Measurements*, Rensselaer Polytechnic Institute, 2002

[3] Cesar C. Palerm *et al.*: *Hypoglycemia Prediction and Detection Using Optimal Estimation*, Diabetes Technology & Therapeutics, vol. 7, num. 1, 2005

[4] B. Wayne Bequette: *A Critical Assessment of Algorithms and Challenges in the Development of a Closed-Loop Artificial Pancreas*, Diabetes Technology & Therapeutics, vol. 7, num. 1, 2005

[5] H.M. Steil *et al.*: *Modeling Insulin Action for Development of a Closed-Loop Artificial Pancreas*, Diabetes Technology & Therapeutics, vol. 7, num. 1, 2005

[6] Nakhle H. Asmar: *Partial Differential Equations*, second edition, Prentice Hall, 2004

[7] John B. Jørgensen, Sten B. Jørgensen: *Model Predictive Control*, Technical University of Denmark, 1998

[8] Greg Welch, Gary Bishop: *An Introduction to the Kalman Filter*, University of North Carolina,2006

[9] Gabriele Pannocchia, James B. Rawlings: *Disturbance Models for Offset-Free Model-Predictive Control*, AIChE Journal Vol.49 No.2, 2003