Approach for constrained control of the chest-deflection

den Hamer, A.J.

Published: 01/01/2003

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

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

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

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 21. Dec. 2018
Approach for constrained control of the chest-deflection

Arjen den Hamer
Report No. DCT 2003.70

Report of external traineeship

Coach TUE:  ir. R. Hesseling
Coach BMW:  dr. S. Link
Supervisor:  prof.dr.ir. M. Steinbuch

Technische Universiteit Eindhoven (TU/e)
Department of Mechanical Engineering
Control Systems Technology Group

Eindhoven, July 10, 2003
Abstract

In car-industry, crash-tests are a simple but important measure to assess the safety of vehicles. The maximum chest-deflection is one of the criteria in the european crash test, denoting the occupant injuries. For this reason it is interesting to have an approach, to obtain the specifications of the safety belt resulting in minimizing the maximum chest-deflection. It is obvious that crash-tests cannot be used for this purposes, and instead numerical experiments with multi-body and finite element models are used.

During previous research, feed-back control using loop shaping appeared to be a suited approach to find the required belt-force that minimizes the maximum chest-deflection. However the resulting characteristics of the belt-force, using this approach, are not very realistic. In order to make these characteristics more realistic, a new control approach is chosen which is able to take into account constraints.

This new approach is Model Predictive Control (MPC) which uses optimization techniques together with simplified models of the controlled system, to find the optimal controller output. The theory of MPC is discussed for the general case and a method is developed to obtain the necessary models. MPC is implemented using three cases with each another objective criterium. The results are compared. It is concluded that a model predictive controller based on a reference obtained by the standard bisection method, appeared to be the best approach. Finally, a sensitivity analysis is performed which shows beside the sensitivity, that the approach satisfyingly works for a broad band of possible values for the constraints.
Contents

1 Introduction  ........................................  3
  1.1 Overview ........................................  3
  1.2 Goal ........................................  4
  1.3 Outline ........................................  5

2 Terminology ........................................  6

3 Control approaches ...................................  7
  3.1 Robust-control ..................................  7
  3.2 MPC-control ....................................  7
  3.3 Discussion ....................................  8

4 Model Predictive Control in detail ....................  9
  4.1 Mathematics of MPC ................................  10
  4.2 Cost-function ...................................  10
    4.2.1 Receding horizon principle ....................  13
  4.3 Optimization ..................................  13
    4.3.1 Constraints ..................................  13
    4.3.2 Optimization problem .........................  15
    4.3.3 Convexity of cost-function .................  16
    4.3.4 Convexity of the constraints ..........  16
  4.4 Relation with LQG control ......................  17

5 Modelling ..........................................  18
  5.1 Model for the chest-deflection ...............  19
    5.1.1 Model mismatch and disturbances ..........  22
  5.2 Model for dummy-displacement ..................  23
  5.3 State-space representation ....................  24

6 Three choices for the cost-function ..................  26
  6.1 MPC using a reference ..........................  27
  6.2 MPC without a reference .......................  29
  6.3 MPC using support-points .....................  30
  6.4 Discussion ....................................  32
  6.5 Bi-section method ................................  32
    6.5.1 Implementation ............................  33
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Sensitivity Analysis</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>Influence of $F_{\text{max}}$</td>
<td>36</td>
</tr>
<tr>
<td>7.2</td>
<td>Influence of $\dot{F}_{\text{max}}$</td>
<td>38</td>
</tr>
<tr>
<td>7.3</td>
<td>Influence of $t_{\text{dead}}$</td>
<td>38</td>
</tr>
<tr>
<td>8</td>
<td>Conclusions</td>
<td>41</td>
</tr>
<tr>
<td>9</td>
<td>Recommendations</td>
<td>42</td>
</tr>
<tr>
<td>A</td>
<td>Discrete-time state-space models</td>
<td>44</td>
</tr>
<tr>
<td>B</td>
<td>Implementation of MPC</td>
<td>45</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

During the last decades, car safety is becoming more and more an important topic in car-
industry. Star-ratings of crash-tests are a simple but important measure for car-safety which
is seen by the customers during their buy decisions. An important car-safety test in Europe
is the EURO-NCAP test. The rating of this test is based on a weighted penalty system for
several injury-criteria in different parts of the body. Two injuries which are relative highly
weighted in the EURO-NCAP test are the maximum value of the chest-deflection and the
HIC.

For this reason, it is interesting for BMW to have a general approach which determines
the most appropriate belt-force and airbag characteristics that reduce these injuries. The
resulting characteristics will be used in first instance for the design of the real-world passive
restraints system. In a later stadium, the found approach can also be used for real-time
implementation where the inputs for belt- and airbag-actuators are calculated during the
-crash.

The approach will be developed and tested using multi-body modelling of the crash-test
in MADYMO. For consistency with other work, a model of the US-NCAP test will be used
to develop and test the approach.

1.1 Overview

During previous research, it is chosen to start the development of such an approach by using a
PD-feedback controller to following an optimal trajectory for to one certain injury. The PD-
controller of Van der Zalm [1] is applied successfully for the case of minimizing the maximum
chest-deflection using the belt-force as only input for the system while omitting the airbag.

It appeared to be possible to control the chest-deflection with a PD-controller, however
the resulting characteristics of the belt-force are not realistic. The maximum belt-force and
it’s rate are too high and no attention is payed to dead-time. Secondly, the control strategy
is only focussed on one type of injury, the chest-deflection. Using a standard PD-controller,
will not allow to control more types of injuries.

During real-life application, passive belt-restraints systems or belt-force actuators will not
be able to put every arbitrary force on the belt. And even if the actuator was able to do
so, it is often not desirable to put for example high forces on the system because of other
injury-criteria simultaneously.

In order to make the controlled force more realistic, the following constraints are taken
into account: $F_{\text{max}}$, $F'_{\text{max}}$ and $t_{\text{dead}}$. In a later stadium also more injuries, such as head-accelerations, will be minimized using both the belt-force and the airbag as actuators. Therefore the new control approach must also be able to take into account more types of injuries while using possibly more controller outputs.

### 1.2 Goal

The main focus of this report is the design of a control approach which meets the requirements of handling constraints and Multi-Input-Multi-Outputs (MIMO) systems. The approach will be started for a simple case in such a way that it can be extended later. Constraints on the input have priority above taking into account more injuries. So first will be focussed on the simple case were we take the chest-deflection as system output and the belt-force as system input, the airbag will be taken out of the model. Omitting the airbag is justified because the coupling between the chest-deflection and the head-acceleration is low. This means that adding the airbag in a later stadium to control the HIC, will not change the dynamic behaviour of the chest-deflection significantly.

The objective is defined as:

**Design a flexible and easy to use control-strategy to obtain the required belt-force that minimizes the maximum chest-deflection while taking into account constraints on the belt-force, preferably within one simulation. The constraint are: $F_{\text{max}}$, $F'_{\text{max}}$ and $t_{\text{dead}}$.**

A strong requirement for the design of the control strategy is flexibility. This means that it must be possible to satisfyingly use the same control approach for different crash-tests, dummies or vehicles. Secondly, it must be able to control more injury-criteria in a later stadium.

It is desirable that the controller can be used without being a control expert or having knowledge of the control algorithm. In the ideal case, the controller has to be used as a black box where some data, representing the control demands, are inputs and the belt-force is calculated as output of the controller.

It is preferable that the results of the approach can be obtained within one simulation. This makes the approach suitable for future implementation in a vehicle and prevents iterative procedures which cost a lot of "computational" time.

Because the calculated belt-forces will not be used for real-time implementation in the near future, computational speed of the control algorithm is no issue at the moment.

In order to get an idea what the influence of the constraints is on the maximum chest-deflection, it will be interesting to investigate the sensitivity of the constraints to the maximum chest-deflection.

Note that besides the main constraints on the controller output: $F_{\text{max}}$, $F'_{\text{max}}$ and $t_{\text{dead}}$ the are obviously also other constraints such as not hitting the steering wheel or the maximum speed of the dummy when the dummy is moving back into the seat. These constraint are can be satisfied by the chosen reference and are therefore not always implemented as constraints.
1.3 Outline

Some terminology will be discussed in chapter 2. Two suited control strategies which meet the requirements, are compared in chapter 3. In chapter 4, the general theory of the most suited strategy will be worked out and discussed from a mathematical point of view. A method to obtain the required models for this approach is developed in chapter 5. The results of three different applications of the general theory are discussed in chapter 6. The best approach is used for the sensitivity analysis of chapter 7. Chapter 7 is also meant to test the designed approach in a wide range of possible constraints. In the last chapters, conclusions are drawn and recommendations are given.
Chapter 2

Terminology

The next chapters contain a lot terminology with respect to inputs and outputs. In order to prevent confusion, these terms will be defined and discussed shortly in this chapter. This will be done using the usual setup for feedback control, as it will also be used in this report. This setup is depicted in figure 2.1.

Where: $\delta$, $\dot{\delta}$, $d$ and $\dot{d}$ are defined respectively as: the chest-deflection, the rate of the chest-deflection, the dummy displacement and the dummy velocity. $F_{\text{belt}}$ is the calculated belt-force.

The input of the closed-loop system is defined as input which contains references and definitions of constraints. As a consequence, output is defined as the output of the closed-loop system which is in our case the resulting chest-deflection. The belt-force $F_{\text{max}}$ is calculated as controller-output of the controller $C$ which has to be designed. The controller-output is used as input for the actuators modelled in MADYMO. Based on the calculated $F_{\text{belt}}$, MADYMO simulates the next time-step of the crash. A number of signals are measured and sent back to the controller for feedback. The measured signals are: $\delta$, delta, $d$ and $\dot{d}$. These signals are together with the closed-loop inputs defined as the controller-inputs.
Chapter 3

Control approaches

In this chapter two control strategies which meet the requirements, are discussed and compared to decide which method will be used to develop the control approach in the next chapters.

Although robust-control was already discussed by Van der Zalm [1], it is shortly discussed again to make the comparison more clear. For further information about robust-control is refereed to [1].

3.1 Robust-control

Robust-control is a control strategy widely used since the eighties. Above the ability to handle constraints and MIMO-systems, robust-control is able to prove a certain performance with given model-uncertainties and constraints. Another advantage is that even when the controller is applied on large MIMO systems, the controller is still pretty fast which makes it very suitable for real-time implementation on mechanical systems.

A disadvantage is the fact that uncertainties, constraints and the desired performance are defined using frequency domain filters. This makes it hard define hard constraints on inputs or outputs in the time domain because the frequency content of these signals is not exactly known. Also appropriate tuning of the filters, needed for maximum performance, demands a lot of control experience.

Because robust-control makes no use of prediction of the system behaviour, the approach is not able to react at the current time on possible violations of constraints in the future. This contributes to the fact that robust-control is less suited for handling constraints.

3.2 MPC-control

Model Predictive Control, in short MPC, is a control strategy widely used in process industry. Due to it’s large computational times, it is only since the latest years that it is also used for faster mechanical systems.

In fact MPC translates the control problem into an optimization problem. During this optimization problem, a number of future controller outputs is chosen such that the prediction of a cost-function is minimized. This cost-function represents the control objective and can be a function of plant inputs, states and outputs.
Using simplified models of the system behaviour, this cost-function is optimized for a number of future samples. At the end of the optimization process, only the controller output for the next sample is put on the system. During the next sample, the whole process is repeated again based on the new states of the system.

MPC-control is very suited for handling constraints. This is due to the fact that the control algorithm predicts future plant inputs, states and outputs which makes it possible to react at the current time sample on possible violations of the constraints at future samples.

Using the time domain makes the interpretation of the results more easy and demands less control knowledge when constraints are changed, as in case of tuning the filters of a robust controller. On the other hand, the implementation itself is more complex because the control problem has to be translated into an optimization problem based on simplified models of the system. Another drawback of MPC with respect to robust-control is that MPC can not guarantee a certain performance for given model uncertainties.

### 3.3 Discussion

In general, a robust-controller is faster than a MPC-control algorithm. This is due to the computational-time expensive optimization procedure which is performed every sample again. Because we are not working with real-time implementation, this is not an important argument.

The difference between the discrete-time domain and the frequency domain approach is more important. Because it is likely that the constraints will be varied very often to investigate sensitivity, it is easier to spend once more time on the implementation instead of changing the filters every time a constrain is changed.

A MPC-control algorithm is able to prevent future violations of constraints because it looks forward in the future (predicts) using simplified models. This is not possible in case of a robust-controller because it uses no prediction of future states, inputs and outputs so it doesn't know that constraints will be possibly violated in the future. This makes that MPC is better suited to handle constraints.

Overall, it can be concluded that the easiness of changing constrains, better handling of constraints and easy interpretation due to the time-domain approach are more important arguments than the high computational costs and harder implementation of MPC. These reasons makes that it is chosen to further develop the control approach using the MPC strategy.
Chapter 4

Model Predictive Control in detail

This chapter will give a detailed explanation about the working and mathematical background of the MPC-algorithm on a general level. First will be started with a more intuitive explanation of MPC using the analogy of a chess-play. Using this analogy, the general idea of MPC becomes clear so that the mathematical derivations of the next sections will be easier to understand.

Idea of MPC vs. chess-play

Using the analogy of a chess-play, it can be seen how MPC uses prediction to calculate the optimal output for a number of future samples and only apply the output for the next sample. It can also be seen that errors in the prediction will lead to controller outputs which are not optimal.

Imagine a chest-play. When you play the game, you think of a number of moves for future turns based on the current state on the chess-board. During your move you may only apply the first step of this set of moves. Then, the opponent has it’s turn and reacts on your move. Based on the new state on the chess-board, you rethink/recalculate a new set of moves and again apply the first move during your turn.

During the play, your moves are based on the expected reaction/output of your opponent. When your opponent is exactly doing what you expect, you can predict a large number of moves without change of previous planned moves, this gives good performance. If the
opponent moves are different than expected, your planned moves have to be changed every
new turn and you will lose performance.

The MPC-control algorithm uses the same approach. It predicts the process and calculates
the optimal controller output for a number of future samples using simplified models and the
current states of the plant. Then only the first element of the vector with optimal future
controller outputs is applied to the system during the next sample. During the next sample,
the plant reacts on the current input by changing the states and consequently also its outputs.
The new outputs of the system are measured during the next sample and again the whole
optimization process is repeated. Again only the first element of a number of future controller
outputs is put on the system.

4.1 Mathematics of MPC

As already mentioned, MPC writes the control problem as an optimization problem. In
order to do so, the control objective is written as a cost-function which is minimized. This
cost-function contains several criteria which can be a function of: plant-inputs, -states and
-outputs. One of the criteria can for example be the distance to a reference for one or more
outputs but also the change in the controller output.

First, the approach is explained for the current sample $k$ without using constraints. After
that, the approach is extended for more than one sample, leading to the receding horizon
principle. And at last constraints are added and some general topics about optimization are
discussed.

4.2 Cost-function

The cost-function $J$ reflects the control objective. It is not absolutely necessary but the
optimization criterium is often written as a quadratic criterium. This has two advantages:

- positive and negative values cannot be averaged.
- only in case of a square criterium convexity can be proved easily which means that only
  one minimum exists.

A typical cost-function looks like:

$$J(k) = \sum_{i=1}^{p} Q_i (\hat{y}(k+i|k) - y_{ref}(k+i))^2 + \sum_{j=1}^{m} R_j \Delta \hat{u}^2(k+j|k)$$

With the goal to:

$$\min_{\Delta u(k+1|k)} \min_{\Delta u(k+2|k)} \cdots \min_{\Delta u(k+m|k)} J(k)$$

(4.2)

where $p$ is the prediction horizon, $m$ is the control horizon and $m \leq p$

$Q$ and $R$ are vectors with the weighting for the criteria $(\hat{y} - y_{ref})^2$ and $\Delta \hat{u}^2$
$\hat{u}$ and $\hat{y}$ are respectively the in- and output for samples $k + j$ and $k + i$ estimated at sample
The criterium of equation (4.1) is a weighting of two parts. The first term reflects the tracking performance, the second term reflects the control effort. The desired behaviour, which is good performance, is found when the first term is weighted much heavier that the second term.

Notice that the units of $y$ and $\Delta u$ plays an important role in choosing the value of the weighting factors $Q$ and $R$. When, for example, the output $y$ is measured in millimeters instead of meters, $Q$ has to be changed with a factor of $1 \cdot 10^6$ to keep the same objective function.

In equation (4.1), $p$ and $m$ describe the length of respectively the prediction-horizon and the control-horizon. The controller output $\Delta u$ is calculated and weighted by $R$ over $m$ samples in the future. The error to the reference $(y_i - y_{ref})$ is weighted by $Q$ and minimized over $p$ samples. This proces is depicted in figure (4.2). Not only the length of these horizons influences the performance, but also their mutual ratio is important. Note that the prediction horizon is always longer or equal to the control horizon.

Using simple models, $\hat{y}_i$ in equation (4.1) can be approximated by a function of $\Delta \hat{u}_i$. This leads to a new cost-function which is completely formulated in terms of $\Delta \hat{u}$ and constants. This makes the cost-function suited for an optimization algorithm. Obviously, the performance of the controller depends on the models used to describe $\hat{y}_i$ in terms of $\Delta \hat{u}_i$.

Simple models, generally Linear Time Invariant (LTI) models, are used to describe $\hat{y}_i$ in terms of $\Delta \hat{u}_i$ because of advantages during the optimization.

A common representation of LTI models is:

\[
x(k+1|k) = Ax(k) + Bu(k) \quad (4.3)
\]
\[
y(k+1|k) = Cx(k+1|k) + Du(k+1|k) \quad (4.4)
\]
Assuming that $D$ equals zero, which is the case for the most mechanical systems, it can be derived that:

$$
\begin{align*}
    y(k+1|k) &= C[Ax(k) + Bu(k)] \\
    y(k+2|k) &= C[A^2x(k) + (AB + B)u(k) + B\Delta u(k+1)] \\
    &\vdots
\end{align*}
$$

where $A$, $B$ and $C$ are constant matrices.

In this way, all the future outputs within the prediction horizon can be written as function of future inputs.

$$
\begin{bmatrix}
    y(k+1|k) \\
    y(k+2|k) \\
    \vdots \\
    y(k+p|k)
\end{bmatrix} = \begin{bmatrix}
    C \\
    CA \\
    \vdots \\
    CA^{p-1}
\end{bmatrix} x(k) + \begin{bmatrix}
    CB \\
    CABA \\
    \vdots \\
    CABA^{p-1}
\end{bmatrix} u(k) + \begin{bmatrix}
    CB \\
    CABA \\
    \vdots \\
    CABA^{p-1}
\end{bmatrix} \Delta u(k+1|k)
$$

(4.5)

The first part of the equation is defined as the $\alpha$-response. The second part of equation (4.6) is called the $\beta$-response of the system. The $\alpha$-response at $t = k$ is not changed by future controller outputs so this part is represented as a constant vector during the optimization. The $\beta$-response is a function of future controller outputs $\Delta u$ and can be written as:

$$
y_\beta = Y\Delta u
$$

(4.7)

This matrix $Y$ plays a crucial role in the optimization process. Note that the number of columns of $Y$ is equal to the number of samples $m$ in the control horizon and the number of rows of $Y$ equals the number of samples in the prediction horizon represented by $p$. The length of the control horizon, $m$, represents the number of degrees of freedom in the optimization and represents therefore the dimension of the optimization space. This space will be represented in this report by $\Delta u \in \mathbb{R}^m$.

Now, the cost-function can be written as a function of the variable $\Delta u$ and a number of constants:

$$
J = \sum_{i=1}^{p} [y_{\Delta u=0i} + (Y\Delta u)_i - y_{refi}]^2 Q_i + \sum_{j=1}^{m} \Delta u_j^2 R_j
$$

(4.8)

$^1$Note that this part of the response is not equal to the usual terminology of forced-response.
Rewriting, using matrix notation and omitting constant terms, which are not relevant to determine the minimum, gives:

\[
J = \frac{1}{2} \Delta u^T[Y^T Q Y + R] \Delta u_i + Y^T Q (y_{i \text{a}} - y_{\text{ref}}) \Delta u
\]  

(4.9)

Note that in this equation \( Q \) and \( R \) are matrices with the weighting on their diagonal.

The objective \( J \) in equation (4.9) is only a function of \( \Delta u \) and describes therefore the cost at every point in the space \( \Delta u \in \mathbb{R}^m \). Finding the minimum in this space is based on Karush Kuhn Tukker (KKT) conditions [2] which states that the a minimum is found at a point where the gradient equals zero and the Hessian is larger equal zero. The gradient and Hessian can be found by derivation of equation (4.9). This gives:

\[
\nabla J = Y^T Q (y_{\Delta u=0_{\text{opt}}} - y_{\text{ref}}) \\
\nabla^2 J = Y^T Q Y + R
\]

(4.10)

In this representation, \( \nabla J \) and \( \nabla^2 J \), represents respectively the gradient and the Hessian of the cost-function. This gradient and Hessian will be used in section 4.3.2 as input for the optimization algorithm.

### 4.2.1 Receding horizon principle

The previous section describes the procedure of the MPC-control algorithm for only one sample \( k \). Now we want to extend the approach for a chain of samples. Thinking of the chess-play analogy, a number of optimal controller outputs is calculated at sample \( k \) depending on the state of the system. But only the first step was applied to the system. The next sample this procedure is repeated for a new set of controller outputs based on the new states of the plant. In other words the prediction and control horizons shift one sample every time a new sample is started. This is called the receding horizon principle of MPC.

### 4.3 Optimization

#### 4.3.1 Constraints

The advantage of MPC is the ability to take into account constraints. These constrains can be seen as hyperplanes in the optimization space \( \Delta u \in \mathbb{R}^m \). The optimal point, which is called the optimizer, may not lie at the wrong side of the hyperplanes in case of inequality constraints or the solution has to be on the hyperplane in case of equality constraints. In optimization terms, this is called the feasible domain.

In order to describe the constraint surfaces in the optimization space \( \Delta u \in \mathbb{R}^m \), the constraints has to be written as a function of \( \Delta u \). How this is done for several types of constraints will be discussed in the next subsections. Constraints on the change of input \( \Delta u \) can directly be given as input of the used optimization algorithm and will therefore not be discussed.

Notice that constraints are only taken into account for the samples which are predicted/optimized. This makes that constraints on the states or outputs are only seen for \( p \) samples in the future. Therefore certain constraints can have a large influence in the choice of the prediction horizons. This effect will be further discussed during the implementation of MPC in chapter 6.
Constraints on controller output $u$

The controller output $u$ is written as:

$$
\begin{align*}
  u(k+1|k) &= u(k) + \Delta u(k+1|k) \\
  u(k+2|k) &= u(k) + \Delta u(k+1|k) + \Delta u(k+2|k) \\
  & \vdots \\
  u(k+m|k) &= u(k) + \sum_{i=1}^{m} u(k+m|k)
\end{align*}
$$

In matrix form:

$$
\begin{bmatrix}
  u(k+1|k) \\
  u(k+2|k) \\
  \vdots \\
  u(k+m|k)
\end{bmatrix} = \begin{bmatrix}
  u(k) \\
  u(k) \\
  \vdots \\
  u(k)
\end{bmatrix} + \begin{bmatrix}
  1 & 0 & \ldots & 0 \\
  1 & 1 & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  1 & 1 & \ldots & 1
\end{bmatrix} \begin{bmatrix}
  \Delta u(k+1|k) \\
  \Delta u(k+2|k) \\
  \vdots \\
  \Delta u(k+m|k)
\end{bmatrix}
$$

Constraints on the controller output reflecting maxima are written as:

$$
\begin{align*}
  u(k) & \leq u_{\text{max}} \\
  u_k + \Phi \Delta u(k) & \leq u_{\text{max}}
\end{align*}
$$

The constraints on the controller output are now written as a function of the optimization space $\Delta u(k) \in \mathbb{R}^m$. Every row in equation (4.13) represents a hyperplane in the optimization space.

Usually inequality constraints are written in the more common form: $Ax \leq b$. Where $A$ and $b$ can be chosen and $x$ represents the vector with optimization variables, in our case $\Delta u$. Equation (4.12) is written in the common form. For clarity the index $(k)$ is omitted in the following equations.

$$
\begin{align*}
  u_k + \Phi \Delta u & \leq u_{\text{max}} \\
  \Phi \Delta u & \leq \frac{u_{\text{max}} - u_k}{b}
\end{align*}
$$

In the same way, an expression for the set of constraints on the minimum input can be found.

$$
\begin{align*}
  -\Phi \Delta u & \leq \frac{u_k - u_{\text{min}}}{b}
\end{align*}
$$

Constraints on state $x$ and outputs $y$

For constraints on the outputs, again the constraints are written in the common form $Ax \leq b$. The constraints are:

$$
\begin{align*}
  y_k & \leq y_{\text{max}} \\
  y_k & \geq y_{\text{min}}
\end{align*}
$$
According to section 4.2, the output $y$ can be rewritten as a part which depends on $\Delta u$ and a constant part $y_\alpha$. For the constraints on the output representing maxima, this can be reformulated to:

$$
Y \Delta u + y_\alpha \leq y_{\text{max}}
$$

(4.17)

Constraints representing minima:

$$
- Y \Delta u \leq - y_{\text{min}} - y_\alpha
$$

(4.18)

In the same way that these equations describe constraints on the outputs, constraints on the states can be taken into account. As already mentioned, the matrix $Y$ describes how future inputs influence future outputs. Constraints on the states can be taken into account by making the constrained states of the simplified models also outputs. From equation (4.4), it can be seen that this is achieved when the matrix $C$ is extended with extra rows for the desired states. The extended $Y$-matrix can be found with equation (4.7). Using again equation (4.17) and (4.18) gives the constraint matrices $A$, $x$ and $b$ for constrained states and outputs.

### 4.3.2 Optimization problem

Because our optimization criterium is quadratic and the constraints are linear in $\Delta u$, the optimization problem is called Linear Quadratic. Good algorithms are available for solving such a Linear Quadratic Programming (LQP) problem. An example of such an algorithm is the optimization algorithm quadprog from the optimization toolbox of Matlab 5.3 which is used for optimization in this report.

The quadprog algorithm transforms the constrained problem into a new unconstrained problem with the use of Lagrange Multipliers [2]. For this new unconstrained problem, the optimum is found when the set of KKT conditions is solved. In order to do so gradient and Hessian information is needed. Using the gradient and Hessian as described in equation (4.10), the cost-function and the constraints can be written as:

$$
\min_{\Delta u} \frac{1}{2} \Delta u^T H \Delta u + f' \Delta u
$$

subjected to:

$$
A \Delta u \leq b
$$

(4.19)

Where $H$ and $f$ represents respectively the Hessian and the Gradient of the objective function as defined in equation (4.1). $A$ and $b$ are the matrices containing all the constraints.

The total constraint-matrices $A$ and $b$ are constructed by putting all separated constraint matrices in one large matrix:

$$
A = \begin{bmatrix}
\Phi \\
-\Phi \\
Y_{\text{extended}} \\
- Y_{\text{extended}}
\end{bmatrix}
$$

$$
b = \begin{bmatrix}
u_{\text{max}} - u(k) \\
u_{\text{min}} + u(k) \\
y_{\text{max}} - y_{\alpha - \text{response}} \\
y_{\text{min}} + y_{\alpha - \text{response}}
\end{bmatrix}
$$

(4.20)
4.3.3 Convexity of cost-function

A quadratic function is chosen because for such a function it can be easily proved that the found minimum is equal to the global minimum. In optimization terms, this is called convexity of the problem consisting of convexity of the objective function and convexity of the feasible domain.

If there exists a mathematical expression for the control-objective, derivative information can generally be calculated. An optimum is found at the place where the KKT conditions hold: first derivative equals zero and the second derivative is larger than zero. It can be understood that only one optimum exist in case of a second order cost-function. In case of a fourth order cost-function, the found minimum doesn’t have to be the global optimum because there exist more points that satisfy the KKT conditions. If it is assured that a found optimum is the global optimum, the cost-function is called convex.

Let us see which conditions are to be satisfied for convexity of a quadratic cost-function. Therefore the gradient and Hessian of the cost-function are derived:

\[ J = \Delta u^T Y^T Q Y \Delta u + \Delta u^T R \Delta u \]
\[ \nabla J = \frac{\partial J}{\partial \Delta u_1}, \frac{\partial J}{\partial \Delta u_2}, \ldots, \frac{\partial J}{\partial \Delta u_m} \]
\[ \nabla \nabla^T J = \begin{bmatrix}
\frac{\partial^2 J}{\partial \Delta u_1^2} & \frac{\partial^2 J}{\partial \Delta u_1 \Delta u_2} & \cdots & \frac{\partial^2 J}{\partial \Delta u_1 \Delta u_m} \\
\frac{\partial^2 J}{\partial \Delta u_2 \Delta u_1} & \frac{\partial^2 J}{\partial \Delta u_2^2} & \cdots & \frac{\partial^2 J}{\partial \Delta u_2 \Delta u_m} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 J}{\partial \Delta u_m \Delta u_1} & \frac{\partial^2 J}{\partial \Delta u_m \Delta u_2} & \cdots & \frac{\partial^2 J}{\partial \Delta u_m \Delta u_m}
\end{bmatrix} \]

When the Hessian is positive definite for every \( \Delta u \), the second derivative is positive in every direction. This assures that the gradient has one zero-point and as a consequence there exists only one minimum. It can be derived that this is the case if \( Q \) and \( R \) are diagonal matrices which contain only positive elements.

If a higher degree is chosen for the cost-function, the optimization variable \( \Delta u \) appear in the Hessian such that it is much harder to prove convexity.

4.3.4 Convexity of the constraints

In case of a constrained problem, also convexity of the feasible domain has to be proved in order to guarantee that the found optimum is the global optimum.

Although there exist more mathematically based expressions for the determination of convexity of the feasible domain, an easier expression is based on graphical intuition:

*The feasible domain is said to be convex if every point on the domain can be connected with an other arbitrarily point without crossing the constrained-surfaces [2].*

It can be seen that this condition is always fulfilled in case of linear constrains. If convexity of the cost-function and of the feasible domain is proved, as a consequence the optimization problem itself is also convex.
4.4 Relation with LQG control

For the unconstrained case, the controller output \( u(k+1|k) = u(k) + \Delta u \) can be calculated analytically. Therefore \( y(k+n) \) of equation (4.9) is set equal to \( y_{ref}(k+n) \) and the KKT conditions are used to find the optimal controller output:

\[
\frac{\partial f(\Delta u)}{\partial \Delta u} = H\Delta u + f = 0
\]

rewriting gives:

\[
\Delta u = -H^{-1}f
\]  

writing out this equation gives the following control-law:

\[
\Delta u = -[Y^TQY + R]^{-1}Y^TQ (y_{ref} - y_{\Delta u=0}) = K_{MPC}(y_{ref} - y_{\Delta u=0})
\]

The matrix \( K_{MPC} \) is a constant matrix which is only a function of the chosen weighting of \( Q \) and \( R \) and can therefore be calculated off-line. This equation is the same as the equation used for LQG-control. It can therefore be stated that a MPC-controller without constraints behaves equal as an LQG-controller.
Chapter 5

Modelling

As described in chapter 4, simple models play a crucial role in the optimization process of finding the optimal controller output such that the prediction of the cost-function is minimized.

It is obvious that simple models are needed to rewrite the outputs of the system as function of the input $\Delta u$. Less obvious is the fact that also models are needed for the states/outputs which are only constrained but not minimized. These models are needed to predict possible future violations of the constraints. The needed models are:

- **the dynamic I/O relation from belt-force $F$ to chest-deflection $\delta$:** This model is needed to minimize the distance to the reference or to minimize the total chest-deflection itself.

- **the dynamic I/O relation from belt-force $F$ to dummy displacement $d$:** This model is needed for the constraint which prevents that the dummy hits the steering-wheel and to limit the dummy-velocity when the dummy is moving back into the seat.

Note that the constraints which are working at the input can be implemented without a model.

In this chapter, a general method will be described to obtain LTI models of the dynamic behaviour from the belt-force $F(t)$ to both chest-deflection $\delta$ and dummy displacement $d$. As described in the report of Van der Zalm [1], such a dynamic model of the local system behavior for a certain I/O relation can be obtained using a modified approximate minimal realization algorithm. This algorithm uses normalized step-responses to estimate a discrete-time model.

A big advantage of this modified approximate realization algorithm is that insight in the relevance of a higher order approximation can be gained by analysis of the singular values. Good results were obtained with a second order model based on this method during the tuning of the PD-controller. However, there are a few problems when this algorithm is applied for the estimation of a model which is used during MPC:

- the model is fitted on normalized step-response data based on steps added to the nominal input, so the model describes local behavior around some path. It is therefore not assured that the global behavior of the MADYMO model is accurately described by the resulting model.

- The states of the fitted state-space model are not known. This means in our case that the states in the obtained models are a certain combination of position and velocity.
Using MPC, the state $x(k)$ has to be updated every sample (see equation 4.6), meaning that the physical interpretation of the states of the model has to be known.

There exist a certain transformation which transforms the states into position and velocity. This transformation can be found when we make use of the fact that only in the case of position and velocity, the following relation holds:

$$\frac{d}{dt} x_1 = x_2 \tag{5.1}$$

where $x_1$ represents the position and $x_2$ represents the velocity.

Using state-space notation, the desired transformation is found when the first row of the state-space $A$-matrix is transformed into:

$$A = \begin{bmatrix} 0 & 1 \\ \cdot & \cdot \end{bmatrix} \tag{5.2}$$

This transformation is different for every new approximation of the I/O-transfer and is therefore not very flexible when the models has to be updated.

To overcome these problems, a method is proposed based on the knowledge from the modified approximate realization algorithm, resulting in an approximation of the global behaviour.

### 5.1 Model for the chest-deflection

By Van der Zalm [1] it is shown that the dynamic behaviour from belt-force $F$ to chest-deflection $\delta$ can be well approximated by a linear second order LTI-model of the following form:

$$M \ddot{\delta} + D \dot{\delta} + K \delta = \Sigma F + w \tag{5.3}$$

where:

- $M$, $D$, $K$ are the parameters representing respectively inertia, damping and stiffness $^1$.
- $\delta$ is the chest-deflection and $w$ is the misfit between the non-linear MADYMO model and the 2nd order LTI-model.

The chest-deflections $\delta$, velocities $\dot{\delta}$, accelerations $\ddot{\delta}$ and forces $\Sigma F$ at sample $i$ are obtained from the MADYMO-simulation of the PD-controlled closed-loop system as designed by Van der Zalm. Using the trajectory designed by Van der Zalm, makes that the fitted parameters describe the behaviour close to the path which will also be followed by the MPC-control algorithm.

Most information about inertial and damping parameters will be found in the first part till 10 ms, after this part the chest-deflection becomes nearly constant as can be seen in figure 5.1. It is therefore chosen to estimate inertia and damping parameters only in the first part and the stiffness parameter during the whole crash. In this way it is guaranteed that
the steady-state response of the fit is the same as the MADYMO-model. This is important because the chest-deflection will be nearly constant during the crash.

The parameters $M_{chest}$, $D_{chest}$ and $K_{chest}$ are fitted using least-squares techniques minimizing $w$ of equation (5.3) over the first 10 ms.

$$\min_{M,D,K} \sum_{i=1}^{n} w^2 = \sum_{i=1}^{n} [M\ddot{\delta}_i + D\dot{\delta}_i + K\delta_i - \Sigma F_i]^2$$

(5.4)

$w$ can be written in matrix notation as:

$$w = \begin{bmatrix} \ddot{\delta}_1 & \dot{\delta}_1 & \delta_1 \\ \ddot{\delta}_2 & \dot{\delta}_2 & \delta_2 \\ \vdots & \vdots & \vdots \\ \ddot{\delta}_n & \dot{\delta}_n & \delta_n \end{bmatrix} \begin{bmatrix} M \\ D \\ K \end{bmatrix} - \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}$$

(5.5)

$$w = A \begin{bmatrix} M \\ D \\ K \end{bmatrix} - F$$

(5.6)

The unknown parameters can be found when the criterium of equation (5.4) is minimized. Using the KKT-conditions, it can be derived that the position of the minimum is given by:

$$\begin{bmatrix} M \\ D \\ K \end{bmatrix} = [A^T A]^{-1} A^T F$$

(5.7)

The found parameters for the first 10 ms are:

$$M_{chest} = 2.4 \, kg$$
$$D_{chest} = 1 \cdot 10^3 \, N/\text{ms}$$
$$K_{chest} = 2.4 \cdot 10^6 \, N/m$$

(5.8)

1Contrarily to the common representation of these parameters, capitals are used to prevent confusion with parameters in the next chapter.
As can be seen in figure 5.1, the dynamic behaviour is operating point dependent because a constant output requires a continuous changing input. Because the chest-deflection is almost constant after 10 milliseconds, the variation in the system behaviour can not be described by variations in inertial or damping parameters. The operating point-dependent behaviour is therefore modelled by variations of the stiffness as function of time. This can be justified because the PD-controller of Van der Zalm and the MPC-control algorithm will almost follow C1.

The stiffness parameter can be found using equation (5.3). When \( \ddot{\delta} \approx 0 \) and \( \delta \approx 0 \), equation (5.3) can be written as:

\[
K_{\text{chest}} = \frac{\Sigma F}{\delta}
\]  

(5.9)

Using equation (5.9), the time-dependent behavior can roughly be split into 5 parts with stiffness as can be seen from figure 5.1:

\[
\begin{align*}
0 \leq t \leq 35 : \quad K_{1,\text{chest}} &= 2.4 \cdot 10^5 \text{ N/m} \\
35 \leq t \leq 50 : \quad K_{2,\text{chest}} &= 1.7 \cdot 10^5 \text{ N/m} \\
50 \leq t \leq 80 : \quad K_{3,\text{chest}} &= 1.4 \cdot 10^5 \text{ N/m} \\
80 \leq t \leq 90 : \quad K_{4,\text{chest}} &= 2.2 \cdot 10^5 \text{ N/m} \\
90 \leq t : \quad K_{5,\text{chest}} &= 3.0 \cdot 10^5 \text{ N/m}
\end{align*}
\]  

(5.10)

where \( t \) is the time in milliseconds.

In figure 5.2 the least-square fit, applied on the overall behavior, is compared with the output from MADYMO and the fitting results of Van der Zalm. It can be seen that the least-squares fit with constant parameters is not significantly better than the modified approximate realization fit. But the fit with changing stiffness approximate the MADYMO response better.

![Figure 5.2: compare different models](image)

22
5.1.1 Model mismatch and disturbances

The operating point dependent behaviour of the stiffness can be represented by a time-variant model. However it appeared to be easier to see the change of the stiffness parameters as a disturbance acting on the system. This makes that the stiffness parameter $k_{\text{chest}}$ can be used in the model during the whole crash.

A second problem is that the LTI model can not perfectly describe the non-linear behaviour of the MADYMO-model. This may lead to performance degradation or even instability. A way to handle with these model uncertainties/mismatches is to take them into account via disturbance estimation.

![Diagram of uncertainty as addition to the states.](image)

The disturbances don’t necessary have to be taken into account at the output but can also be taken at the input or a combination. In industrial applications, estimating the disturbance at the output is very popular because of easy implementation. In our case, the disturbance acts as a force so it is more likely to model the disturbance at the input. However estimation of disturbance at the input leads to more complicated algorithms. Therefore it’s chosen to use output disturbances first to keep the algorithm as simple as possible. The output disturbance caused by model uncertainties is represented by the $\Delta$ as depicted in figure 5.3.

Using constant output disturbance estimation assumes a constant offset at the output, which is not always true in case of model-uncertainties.

For the output disturbance, we make use of a very common estimation law:

$$d(k+1) = d(k) + L(y - \hat{y})$$

(5.11)

where:

$\hat{y}$ is the predicted value of the MPC-control algorithm for sample $k$. $y$ is the measured output at sample $k$. The gain $L$ is obtained empirically.

Inertial and damping properties causes dynamic behaviour, which makes that the influence of the changing stiffness does not directly appear at the output. Therefore several samples are used to estimate the output disturbance. The disturbance estimation for the chest-deflection
is estimated over five samples:

\[ d_{y_1}(k+1|k) = d(k) + \frac{\sum_{n=1}^{5} L_n [y(k|k) - \hat{y}(k|k-n)]}{5} \]  

(5.12)

The used gains are: \( L_{y_1} = 0.5, L_{y_2} = 0.5, L_{y_3} = 0.5, L_{y_4} = 0.5 \) and again \( L_{y_4} = 0.5 \).

The calculated disturbance is added to the free response as:

\[
\begin{align*}
    y_{1,free}(k+n|k) &:= y_{1,free}(k+n|k) + d_{y_1}(k) + d_{y_2}(k)n_{ts} \\
    y_{2,free}(k+n|k) &:= y_{2,free}(k+n|k) + d_{y_2}(k) \\
    y_{3,free}(k+n|k) &:= y_{3,free}(k+n|k) + d_{y_3}(k) + d_{y_4}(k)n_{ts} \\
    y_{4,free}(k+n|k) &:= y_{4,free}(k+n|k) + d_{y_4}(k)
\end{align*}
\]

(5.13)

Where \( t_s \) represents the sample-time and:

- \( y_1 \): chest-deflection
- \( y_2 \): chest-deflection speed
- \( y_3 \): dummy position
- \( y_4 \): dummy speed

Summarizing, the control algorithm starts with the right value for stiffness \( k_{1,\text{chest}} \) and after the first part, the disturbance estimation correct the changing stiffness which is seen as disturbances at the output.

### 5.2 Model for dummy-displacement

At first sight, it seems that the dummy can be modelled by a pure single body with a mass equal to mass of the dummy. This way of modelling doesn’t give the right results because the belt-force is applied to the end of the belt and don’t have to be equal to the force which is working on the chest due to transmission and friction effects.

Because this transmission depends on the position of the dummy, it is tried describe this effect by an extra stiffness. This stiffness does not have a physical background relation however satisfying results are obtained using this model as shown in the next chapter. In order to let the stiffness work right, the position is normalized to zero by subtracting the begin position.

Using the same approach as used in section 5.1 a 2nd order model can be fitted for the displacement of the dummy. Damping is not taken into account because it is expected that such a parameters is not needed to describe the behaviour of the dummy displacement and velocity with respect to the belt-force. Using equation (5.5) and equation (5.7) gives:

\[ M_{\text{dummy}} = 50 \, \text{kg} \]  
\[ K_{\text{dummy}} = 1 \cdot 10^4 \, \text{N/m} \]  

(5.14)  
\[ (5.15) \]
5.3 State-space representation

Now that the unknown parameters of the continues 2\textsuperscript{nd} order LTI-models are known, the models have to be written in discrete-time state-space representation for the MPC-algorithm. Therefore the second order differential equation (5.3) is rewritten as\(^2\):

\[
\ddot{\delta} = -\frac{D}{M} \dot{\delta} - \frac{K}{M} \delta + F
\]  
(5.16)

Splitting the equation into two first order equations results in:

\[
\begin{bmatrix}
\dot{\delta}
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
-K/M & -D/M
\end{bmatrix}
\begin{bmatrix}
\delta \\
\dot{\delta}
\end{bmatrix} +
\begin{bmatrix}
0 \\
1
\end{bmatrix} F
\]  
(5.17)

The position \(\delta\) and the velocity \(\dot{\delta}\) are not only the states but also the outputs of the system so:

\[
\begin{bmatrix}
y_1 \\
y_2
\end{bmatrix} =
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
\delta \\
\dot{\delta}
\end{bmatrix} +
\begin{bmatrix}
0 \\
0
\end{bmatrix} F
\]  
(5.18)

Where: \(y_1\) and \(y_2\) are the outputs of the state-space model.

Equation (5.17) and (5.18) can be written as:

\[
\begin{align*}
\dot{q} &= Aq + Bu \\
y &= Cq + Du 
\end{align*}
\]  
(5.19)

Where:

\[
A_{\text{chest}} =
\begin{bmatrix}
0 & 1 \\
-D/M & -K/M
\end{bmatrix} \\
B_{\text{chest}} =
\begin{bmatrix}
0 \\
1
\end{bmatrix}
\]

\[
C_{\text{chest}} =
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \\
D_{\text{chest}} =
\begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

\(q\) and \(y\) represent the vectors \([\delta, \dot{\delta}]^T\) and \([y_1, y_2]^T\) which are respectively the states and the outputs. The belt-force \(u\) is the input of the model.

This representation of a system is called the continuous state-space, it describes the states continues over time. Using the same approach, a continues state-space model can be derived for the dynamic behaviour between the belt-force \(F\) and the dummy displacement \(d\) resulting in \(A_{\text{dummy}}, B_{\text{dummy}}, C_{\text{dummy}}\) and \(D_{\text{dummy}}\).

MPC is a discrete-time technique and makes use of discrete models, therefore the models are to be transformed into discrete-time models. In general, a discrete-time model representation has the following form:

\[
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) \\
y(k) &= Cx(k) + Du(k)
\end{align*}
\]  
(5.20)

\(5.21\)

\(^2\)For convenience, the subscript \(\text{chest}\) of the parameters \(M_{\text{chest}}, D_{\text{chest}}\) and \(K_{\text{chest}}\) is left away.
It’s obvious that the matrices $A, B, C, D$ in this equation are not the same as in equation (5.19). For the transformation from a continues to a discrete-time system representation we make use of the Matlab commando \texttt{c2d}.

Because the difference in orders is quite large between the different continues-time models, it is chosen to transform each system apart and join them after that to avoid numerical problems. The two obtained discrete-time models for the chest-deflection and the dummy position can be joined into one large discrete-time state-space model in the following way:

$$A_{\text{total}} = \begin{bmatrix} A_{\text{chest}} & 0 \\ 0 & A_{\text{dummy}} \end{bmatrix}, \quad B_{\text{total}} = \begin{bmatrix} B_{\text{chest}} \\ B_{\text{dummy}} \end{bmatrix}, \quad C_{\text{total}} = \begin{bmatrix} C_{\text{chest}} & 0 \\ 0 & C_{\text{dummy}} \end{bmatrix}, \quad D_{\text{total}} = \begin{bmatrix} 0 \end{bmatrix}$$ (5.22)

$$x = \begin{bmatrix} \delta \\ \dot{\delta} \\ d \\ \dot{d} \end{bmatrix}$$ (5.23)

The numerical values for these matrices can be found in appendix A.
Chapter 6

Three choices for the cost-function

Now the necessary models are obtained, the general theory of the chapter 4 can be used for the implementation of a MPC-control algorithm. The general formulation of the objective function is now replaced by criteria using the chest-deflection \( \delta \) and dummy-displacement \( d \). Three choices for the cost-function are discussed together with their motivation. These cost-function are further explained in the corresponding sections of this chapter.

1. First will be started minimizing the distance to a reference. The results of this tracking controller will be compared with the results of the PD-controller of Van der Zalm [1]. The used cost-function is:

\[
J(k) = \sum_{i=1}^{p} Q_i \left( \delta(k+i|k) - \delta_{ref}(k+i) \right)^2 + \sum_{j=1}^{m} R_j \Delta \dot{u}^2(k+j|k)
\]  

2. Then, the total chest-deflection is minimized, due to problems finding a suitable reference.

\[
J(k) = \sum_{i=1}^{p} Q_i \left( \delta(k+i|k) \right)^2 + \sum_{j=1}^{m} R_j \Delta \dot{u}^2(k+j|k)
\]

3. Finally, support-points for the end-constraints are added to reduce the prediction horizon.

\[
J = \sum_{i=1}^{m} Q \dot{y}_i + \sum_{j=1}^{m} R_j \Delta u_j^2 + W_{d1}(d_p - d_{support}) + W_{d2}(d_p - d_{support})
\]  

The constraints used for the input are:

- the maximum belt-force: \( F_{belt} \leq F_{max} \)
- the minimum belt-force: \( F_{belt} \geq F_{min} \)
- the dead-time: \( F = 0 \) for \( t < t_{dead} \)
- the maximum rate of the belt-force: \( \dot{F}_{max} \)
Additional constraints on the states or outputs are:

- the maximum dummy-velocity \( \dot{d} \) when the dummy is moving back into the seat.
- because the second choice for the cost-function uses no reference, an additional constraints is needed which limits the displacement of the dummy such that the steering wheel is not hit.

The constraints on \( F_{max}, \dot{F}_{max} \) and \( t_{dead} \) are used to make the belt-force characteristics more realistic as described in the objective. The constraint on \( F_{min} \) is needed because the belt-force can not be negative and a constraint on \( \dot{d} \) is needed to force the belt-force back to zero after the crash.

The implementation will be done in Matlab Simulink using a s-function block. The MPC-control algorithm, implemented in a s-function block, calculates the controller output based on the controller-inputs including current measurements of MADYMO. This s-function can be evaluated every time-step of Simulink but it can also be evaluated at a lower frequency which is very useful in our case because solving the optimization problem cost relative much computational time.

6.1 MPC using a reference

First the implementation of a MPC-control algorithm using a reference will be discussed. The results of MPC using a reference can be compared with the results of the PD-controller as proposed in Van der Zalm [1]. However an appropriate reference as described by Van der Zalm is complex and not suited for MPC which makes that he results are not fully comparable. However the results still gives an good idea of the performance of the MPC-control algorithm with respect to a PD-controller.

The reason that the reference of the PD-controller is not used for MPC is in the first place, because it has to be determined iteratively. Secondly, the constraints on the maximum rate can prevent the controller from following the steep parts of the reference. Therefore the reference is chosen as a constant of 30 mm. The constraint on the maximum rate of the belt-force gives the desired smoothing at the begin of the crash. Bringing the belt-force back to zero when the dummy velocity is zero, is achieved by adding a constraint on the velocity of the dummy.

Due to definition of the coordinate frame in the MADYMO model, the dummy speed becomes positive when the dummy is moving back into the seat. Using a constraint which limits the dummy velocity \( \dot{d} \) on a certain value \( v_{\text{max}} \) will compel the belt-force back to zero.

\[
v_{\text{dummy}} \leq v_{\text{max}}
\]

(6.4)

Leaving the reference in this case is possible because constraints have "priority" above following the reference which is weighted in the cost-function. This smoothing of the chest-deflection due to constraints can be seen in figure 6.8.

Following the reference is claimed by minimizing the distance to this reference using the cost-function:

\[
J = \sum_{i=1}^{p} Q_i(\delta - \delta_{\text{ref}})^2 + \sum_{j=1}^{m} R_j \Delta u^2
\]

(6.5)

The control parameters are chosen as:
\[
W_Q = [1 \cdot 10^7] \quad m = 20 = 20 \text{ ms} \quad F_{\text{max}} = \infty
\]
\[
W_R = [1 \cdot 10^{-4}] \quad p = 20 = 20 \text{ ms} \quad \dot{F}_{\text{max}} = 1 \text{ kN/ms}
\]
\[
y_{\text{ref}} = 30 \text{ mm} \quad t_r = 1 \text{ ms} \quad \text{dead-time} = 0 \text{ ms}
\]

Table 6.1: controller settings using a reference

To handle the constraints on the dummy displacement, the dynamic behaviour between the belt-force \( F \) and the dummy displacement \( d \) is also needed. This dynamic behaviour is represented by the extended \( Y \) matrix as explained in section 4.3.1.

Because the weighting \( W_Q \) and \( W_R \) are the same for every sample in the prediction horizon, the vectors \( Q \) and \( R \) are defined as:

\[
Q = [W_{Q_1}, W_{Q_2}, \ldots, W_{Q_p}]
\]
\[
R = [W_{R_1}, R_{Q_2}, \ldots, R_{Q_m}]
\]

The results are depicted in figure 6.1. From figures 5.1 and 6.1, it can be seen that the belt-force calculated by the MPC-controller and the PD-controller are similar. Note the discrete-time characteristic of the MPC-controller in contradiction to the continuous PD-controller. This discrete behaviour can be made more fluent by taking a higher sample frequency for the optimization but this leads obviously to higher computational costs.

Although the control results are satisfying, a disadvantage of using a reference is that the value for the reference has to be found iteratively for every setting of the constraints. A number of simulations is needed to determine the reference such that the full space between the seat and the steering-wheel is used without hitting the steering-wheel. Using the maximum space between seat and steering-wheel assures that: the acceleration, the forces and the chest-deflection are as low as possible.

The reference is varied in order to use the maximum allowable space. This means that the reference can also be replaced by a constraint of not hitting the steering-wheel so that the problem of finding the reference is solved.

Figure 6.1: MPC-controller with reference
Another reason that the use of a reference is not preferred, is that tracking a reference minimizes the corresponding type of injury, but doesn’t necessarily minimize other injuries. Even if a cost-function is chosen which minimized the distance to several references for several injury criteria, this don’t necessarily results in minimal occurring injuries because the interconnection structure between the different injuries is not known.

These arguments makes that is attractive to minimize the injury itself instead of minimizing the distance to a reference. This is implemented in the next section.

6.2 MPC without a reference

Minimizing the chest-deflection itself is reflected by the cost-function:

\[ J = \sum_{i=1}^{p} Q_i \delta_i^2 + \sum_{j=1}^{m} R_j \Delta u_j^2 \]  

(6.7)

As mentioned before, a new constraint has to be added in order to prevent hitting the steering-wheel. This new constraint claims that the displacement of the dummy relative to the car does not exceed the initial distance between the dummy and the steering-wheel \( l_0 \):

\[ d_{\text{dummy}} \leq d_{\text{car}} + l_0 \]  

(6.8)

Where \( d_{\text{dummy}} \) and \( d_{\text{car}} \) represent respectively the dummy displacement and the car displacement measured in the inertial coordinates. \( l_0 \) is the initial relative distance between the dummy in the steering wheel.

Because the dummy consist of several bodies which can move relative to each other, defining the dummy displacement has to be more specific. The chest is the body of the dummy-model which hits the steering-wheel first, so the displacement of this body is measured and used for locate the position of the dummy.

The results are depicted in figure 6.2. The following parameters are used:

\[
\begin{align*}
W_Q &= [1 \cdot 10^7] & m &= 25 \equiv 100 \text{ ms} & F_{\text{max}} &= \infty \\
W_R &= [1 \cdot 10^{-4}] & p &= 25 \equiv 100 \text{ ms} & \dot{F}_{\text{max}} &= 500 \text{ kN/ms} \\
d_{\text{max}} &= 0.6 \text{ m} & t_s &= 4 \text{ ms} & \text{dead-time} &= 0 \text{ ms}
\end{align*}
\]

Where \( d \) represents the normalized position of the dummy which starts at zero for \( t = 0 \). This normalization of the position is needed because also stiffness is used to model the displacement of the dummy as discussed in section 5.2. \( d \) is defined as:

\[ d = d_{\text{measured}} - d_{\text{start}} \]  

(6.9)

Notice that the time-span of the prediction horizon has to be chosen much longer than in the case of using a reference. The reason is that the controller must be able to see all the constraints in the prediction horizon. Because hitting the steering-wheel normally occurs at the end of the crash, the prediction has to be also that long. Due to this long prediction horizon, the sample-frequency of the MPC-control algorithm is chosen lower to prevent excessive computational costs.
The influence of model errors over such a time-span is high, so problems with infeasibility occur easily because, based on simple LTI models, it is not possible to predict possible violation of constraints accurately enough for the whole accident. In combination with constraints on the maximum control output, this can lead to an infeasible/unsolvable optimization problem.

A new idea is proposed to overcome this problem. Using receding "support points" at the end of prediction horizon, the prediction horizon can be chosen shorter. This idea is further worked out in the next section.

![Figure 6.2: MPC-controller without reference](image)

#### 6.3 MPC using support-points

The constraints on the maximum displacement and velocity of the dummy require a long prediction horizon, so support-points will be used which reflect these two constraints. These support-points are prescribed at the end of the prediction horizon and recede together with the horizon.

Implementing the support-points as constraints can lead to infeasibility since a constraint can prevent that the support-point is reached. Therefore it is better to take the distance to these points in the cost-function by:

\[
J = \sum_{i=1}^{p} Q \delta_i + \sum_{j=1}^{m} R_j \Delta u_j^2 + W_d (d_p - d_{support_p}) + W_{ds} (d_p - \dot{d}_{support_p})
\]

(6.10)

where the scalars \(d_p - d_{support_p}\) and \(d_p - \dot{d}_{support_p}\) represent the distance to the support-points predicted for the last sample only, i.e., sample \(k + p\).

The support-points only reflect the position \(d\) and speed \(\dot{d}\) during the last sample of the prediction horizon. Therefore it is chosen to weight \(\delta\) with the vector \(Q\) and weight the distance to the support-points separately in an extra part of the cost-function (equation 6.10).

The following values are used for the weighting:

\[
W_Q = \begin{bmatrix} 1 \cdot 10^4 \end{bmatrix}
\]

(6.11)

\[
W_d = \begin{bmatrix} 1 \cdot 10^5 \\ 1 \cdot 10^5 \end{bmatrix}
\]

(6.12)
The values of table 6.1 are used for the other controller settings. Again $Q$ is defined as:

$$Q = \begin{bmatrix} W_Q & W_Q \\ W_Q & \vdots \\ W_Q \end{bmatrix}$$ \hspace{1cm} (6.13)

The value of $W_2$ to weight the distance to the support-points is chosen such that the end-constraint are reached and a proper weighing with respect to the chest-deflection is achieved.

The values of the support-points, needed for a constant chest-deflection, are not known. During simulation, the support-points are chosen based on a constant acceleration of 170 m/s$^2$. This leads to a time/sample dependent expression for the velocity and the position.

$$d_{\text{support}}(k+p|k) = d_0 + d_0(k+p)t_s + \frac{1}{2} a [(k+p)t_s]^2$$ \hspace{1cm} (6.14)

$$\dot{d}_{\text{support}}(k+p|k) = \dot{d}_0 + a (k+p)t_s$$ \hspace{1cm} (6.15)

With: $a = -170$ m/s$^2$ and sample-time $t_s = 1$ ms

Note that the support-points represent a position or velocity in the future, therefore the corresponding time-sample in the future, $(k+p)$, has to be used.

The results of implementation in the closed-loop with MADYMO are depicted in figure (6.3). It can be seen that the resulting chest-deflection is not constant, which means that it is assumed that the maximum chest-deflection can be lowered. The results detonate in case of active constraints on the maximum controller-output because the support-points can not be reached anymore.

When the support-points can not be reached anymore due to constraints, this leads to larger distances to the support-points. These larger errors are weighted against the chest-deflection in the cost-function of equation (6.10). Consequently this leads to peaks in the chest-deflection when the constraint is not active anymore as can be seen in figure 6.3 for $F_{\text{max}}$. The solution to make the chest-deflection constant seems to be changing the support point till the chest-deflection is more or less constant.

Figure 6.3: MPC-controller with use support-points
6.4 Discussion

The last two approaches intended to solve the problem of finding a reference iteratively. But instead of solving the problem, it became much more complex and still the support points has to be found interactively depending on the value of the constraints. So, this is no solution for the problem of finding a reference.

It can be said that using a reference is the best option with respect to simplicity and performance. However the method has still one disadvantage. For every new set of constraints, the reference has to be determined iteratively.

It would make life easier if the iterative procedure of finding an appropriate reference can be automated. Besides that, an automatic procedure leads to a more structural method with known accuracy. This is desirable during the sensitivity analysis.

6.5 Bi-section method

The bi-section method is a universal method which is often used to search zero-points of mathematical expressions. An advantage is that the method only uses of the sign of the input. This makes the method suited for our case of finding a value for the reference because the only information needed is hitting/not hitting the steering-wheel. The method works as follows:

1. Define a minimum and maximum for the input variable, in our case the niveau of the reference. This value is mostly known from empirical knowledge but can chosen arbitrarily.

2. Using a reference in the middle of this range, a simulation is done to determine whether the steering-wheel is hit or not.

3. If the dummy hits the steering-wheel, the reference has to be higher. The best reference will then be found in the right half of the range as can be seen in figure 6.6. The lower bound is shifted to the current reference.

4. Again the same procedure is repeated for the new range which is half as large as the previous range. The new reference is again chosen in the middle of the new bounds and a simulation is done.

In this way the region where the optimal reference lies is halved every step. This process is depicted in figure (6.6).

It can be derived that the size of the range is a simple function of the number of iterations and the starting size of the range.

\[
\delta_{\text{max}} - \delta_{\text{min}} = \frac{\delta_{\text{max}_0} - \delta_{\text{min}_0}}{2^n}
\]

(6.16)

where \( n \) represents the number of iterations and \( \delta_{\text{max}_0} \) and \( \delta_{\text{min}_0} \) respectively the upper- and lower-bound for the reference during the first iteration.
6.5.1 Implementation

For numerical implementation, several simulations has to be started one by one. Based on reference information of the previous simulation, and information about hitting/not hitting the steering-wheel during the previous simulation, a new reference is calculated.

A numerical expression for hitting/not hitting the steering-wheel including a small space can be found in the measurement of the displacements of the chest and the steer-wheel:

If the following equation holds:

$$\min(d_{\text{steer}} - d) \geq -0.032m$$  \hspace{1cm} (6.17)

the range is changed by:

$$y_{\text{high}}(n + 1) = y_{\text{high}}(n)$$  \hspace{1cm} (6.18)

$$y_{\text{low}}(n + 1) = y_{\text{ref}}(n)$$  \hspace{1cm} (6.19)

if not, the range is changed by:

$$y_{\text{high}}(n + 1) = y_{\text{ref}}(n)$$  \hspace{1cm} (6.20)

$$y_{\text{low}}(n + 1) = y_{\text{low}}(n)$$  \hspace{1cm} (6.21)

The new reference becomes:

$$y_{\text{ref}}(n + 1) = \frac{y_{\text{high}}(n + 1) + y_{\text{low}}(n + 1)}{2}$$  \hspace{1cm} (6.22)

$d$ and $d_{\text{steer}}$ in equation (6.17) are respectively the displacement of the dummy and the displacement of the steering-wheel measured in the minus x-direction of the inertial coordinates.

Using this iterative procedure, the upper- and lower-bound come closer to each other every iteration.

The value of $0.032 \, m$, used in equation (6.17), has to be found once by iteration but can be used for every set of constraints as long as the same combination of vehicle and dummy with corresponding start position is used.

The flow-scheme of this iteration process is depicted in figure 6.5 The results of this method are depicted in figure 6.6, 6.7 and 6.8. The small space between the steering-wheel and chest shows that the value of $0.032 \, m$ in inequality (6.17) is well chosen.
Figure 6.5: Flow scheme of bi-section method
Figure 6.6: bi-section method

Figure 6.7: simulation based on bi-section method

Figure 6.8: $d, \dot{d}$ and $F$
Chapter 7

Sensitivity Analysis

In previous chapters, a control approach is discussed which meets the requirements set in the introduction, however the approach was only tested one set of constraint values. In this chapter, the developed approach will be tested in a wide band of possible values for the constraint values. Besides testing the approach, the sensitivity is also investigated to see whether or not the chest-deflection significantly benefits from improvements corresponding to \( F_{\text{max}} \), \( \dot{F} \) and \( t_{\text{dead}} \).

During the sensitivity analysis, the constraints of the belt-actuator are varied separately namely: \( F_{\text{max}} \), \( \dot{F}_{\text{max}} \) and the \( t_{\text{dead}} \). The controller settings are:

\[
\begin{align*}
W_Q &= [1 \cdot 10^7] & m &= 20 \equiv 20 \text{ ms} & F_{\text{max}} &= 7000 \text{ N} \\
W_R &= [1 \cdot 10^{-4}] & p &= 20 \equiv 20 \text{ ms} & \dot{F}_{\text{max}} &= 1 \text{ kN/ms} \\
t_s &= 1 \text{ ms} & \text{dead-time} &= 12 \text{ ms}
\end{align*}
\]

Table 7.1: Controller settings during sensitivity analysis

The corresponding reference is determined with the bi-section method as discussed in section 6.4. The number of simulations and the starting region are chosen in such a way that the accuracy of the found reference is smaller than 0.25 mm (see equation 6.16).

7.1 Influence of \( F_{\text{max}} \)

It is interesting to see what the influence of \( F_{\text{max}} \) is with respect to the maximum chest-deflection. Simulations are done for different settings of \( F_{\text{max}} \) while keeping the original settings of table 7 for the other parameters. The result are depicted in figure 7.1. The corresponding maximum values for the chest-deflection for the chosen value of \( F_{\text{max}} \) are depicted in figure 7.2. Note that the accuracy in the determination of the reference, as described by equation (6.16), also leads to non-constant minimal distances to the steering-wheel for each simulation which may be interpreted as an uncertainty in the maximum chest-deflection.
The dummy hits the steering-wheel if the maximum allowable belt-force is less than 5500 N. The maximum the chest-deflection reduces very fast in the region of 5500 N till 6500 N. Choosing the $F_{\text{max}}$ higher than 7 kN does barely improves the chest-deflection but can have worse influence on other injury criteria so it would be better to eliminated these peaks to reduce other injuries.

The sensitivity of the maximum chest-deflection with respect to the maximum belt-force can be approximated in the region of interest by following function.

$$\frac{\delta_{\text{max}}}{F_{\text{max}}} = 8.4 \cdot 10^{-4} F_{\text{max}}^2 - 6.8 \cdot 10^{-3} \text{ m/kN}$$ (7.1)

With:

$$F_{\text{max}} \in [5.5, 7] \text{ kN}$$ (7.2)
7.2 Influence of $\dot{F}_{\text{max}}$

Due the mechanical characteristics of the actuators and sensors it is not possible to achieve every desired rate of the belt-force. Therefore it's investigated how further improvements of the rate influences the chest-deflection.

The results are depicted in figure 7.3 and the maximum values of the chest-deflection are depicted in figure 7.4. It can be seen that the rate has to be at least 500 N/ms in order to get an acceptable value for the maximum chest-deflection. A rate higher than 1 kN/ms doesn't influence the maximum chest-deflection significantly. Although very subjective, a reasonable value for the rate seems to be around 750 N/ms.

Again the sensitivity in the region of interest can be approximated by a function:

$$\frac{\delta_{\text{max}}}{\dot{F}_{\text{max}}} = 2.4 \cdot 10^{-2} \dot{F}_{\text{max}}^2 - 2.5 \cdot 10^{-2} \text{ ms/kN}$$

With:

$$\dot{F}_{\text{max}} \in [250, 1000] \text{ N/s}$$

7.3 Influence of $t_{\text{dead}}$

Dead-time is the time needed for the sensors and actuators to detect the crash respectively trigger the restraint system. A typical value for the dead-time, given in literature, is 12 ms [3]. The results for other settings of the dead-time are depicted in figure 7.5 and figure 7.6.

As can be seen in figure 7.5 and 7.6 there seems to be some linear relation between the chest-deflection and the dead-time. This relation seems to hold even on a larger bound. For a dead-time equal to zero, a maximum chest-deflection is found of 28 mm which is consistent with Van der Zalm [1]. An estimation for this linear relation is:

$$\frac{\delta_{\text{max}}}{t_{\text{dead}}} = 0.47 \text{ m/s}$$
Figure 7.4: $\delta_{\text{max}}$ for different $F_{\text{max}}$

Figure 7.5: Sensitivity to dead-time
Figure 7.6: $\delta_{\text{max}}$ for different dead-times
Chapter 8

Conclusions

An approach based on model predictive control appears to meet the objective of minimizing the chest-deflection while handling constraints. Additional requirements of flexibility and easy interpretation are satisfied by using state-space models and the use of a time-domain approach.

Using Least-square techniques in combination with knowledge obtained by a modified approximate realization algorithm, state-space models can be fitted on the non-linear input-output relations of the MADYMO model. Two models are needed to minimize the chest-deflection: the dynamic behaviour between belt-force and chest-deflection and the dynamic behaviour between the belt-force and dummy displacement. These LTI models in combination with output disturbance estimation gave satisfying results during the tracking of a reference.

Three different choices for the cost-function are tested: using a reference, using no reference and using support-points. The best option with respect to minimizing the maximum chest-deflection appeared to be using a reference. A drawback of this method is that the chest-deflection has to be determined iteratively and can therefore not be found within one simulation.

This iterative procedure of finding a reference can be automated using a bi-section method. Using this bi-section method also leads to a more structural approach because the accuracy is known from the number of iteration and a starting range.

A sensitivity analysis is done for three main characteristics of the belt-force actuator: $F_{\text{max}}$, $\dot{F}_{\text{max}}$ and $t_{\text{dead}}$. The sensitivity gives an idea about the benefits of improvements of the restraint-system with respect to the maximum chest-deflection. High values for the constraints barely influence the maximum chest-deflection but can have negative effects on other injury criteria. Using the sensitivity analysis, the developed approach is successfully tested in a wide band of possible constraints.

The relation between the maximum chest-deflection and the maximum belt-force or the belt-force rate can be well approximated by square functions as given in equation (7.1) and (7.3). The relation between maximum chest-deflection and dead-time is nearly linear as described in equation (7.5) which leads to a sensitivity of $0.47 \text{ mm/ms}$ for $t_{\text{dead}}$. 
Chapter 9

Recommendations

- Now that good results are obtained with MPC to reduce the chest-deflection while using the belt-force as actuator, the approach can be extended for multi inputs/outputs. Because the HIC is a heavily weighted criterium in the EURO-NCAP crash-test, minimizing the HIC as extra output while using the airbag as extra actuator seems to be a useful extension of the controller.

- In order to test the compatibility of the developed approach, it is interesting to use the approach for other occupants, crash-tests or vehicles. The compatibility will mainly depend on the quality of the needed models which can be estimated with the developed methods based on least-squares techniques.

- The use of other dummies, crash-tests or vehicles lead to other model-uncertainties. In this case it could be possible that output disturbance estimation is not able to compensate the model-errors. The benefits of more sophisticated techniques such as robust MPC can be investigated with respect to handling model-errors.

- Using more simulations, the investigating of the sensitivity could be extended in the region of interest. This reduces the uncertainties because more simulation results are obtained. A smaller spacing between the chosen constraint values also gives a better idea of the slope of the sensitivity.
Bibliography


Appendix A

Discrete-time state-space models

These discrete state-space matrices are used for during the prediction and optimization of the MPC-controller. The matrices are based on a sample time of 1 ms. Note that the matrices has to be updated when the sample-time is changed.

\[
A = \begin{bmatrix}
0.9566 \cdot 10^{-1} & 8.04 \cdot 10^{-4} & 0 & 0 \\
-8.043 \cdot 10^{2} & 6.215 \cdot 10^{-1} & 0 & 0 \\
0 & 9.979 \cdot 10^{-1} & 9.993 \cdot 10^{-4} \\
0 & 0 & -4.163 & 9.979 \cdot 10^{-1}
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0 \\
1 \\
0 \\
1
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

With the state chosen as:

\[
x = \begin{bmatrix}
\delta \\
\dot{\delta} \\
d \\
\dot{d}
\end{bmatrix}
\]
Appendix B

Implementation of MPC

The MPC-controller is implemented in Matlab Simulink using the s-function blocks. The script below is used for tracking a reference as described in section 6.1.

function [sys,x0,str,ts] = MPC_sfun_simple(t,x,u,flag)

switch flag,
    case 0
        [sys,x0,str,ts] = mdlInitializeSizes
    case 3
        sys = mdlOutputs(t,x,u)
    case { 1 2 3 9}
        sys=[];
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes

sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumInputs = 5;  % states of system and last calculated u
sizes.NumOutputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes)
str = [];
x0 = 0;
global ts_opt
   ts = [ts_opt 0];
function sys = mdlOutputs(t,state,u);

% This file calculates the optimal belt-force in order to minimize chest-deflection.
% input: x1 : chest-deflection
% x2 : chest-deflection speed
% x3 : sternum position
% x4 : sternum speed
% input: t needed for determination of stiffness
% input: uk needed to calculate free response
% input: state, x1, x2, x3 ,x4 needed to calculate free response
% constraints can be given for the outputs and the change of belt-force per sample-time.
% Weighting can be given for outputs and the change of the belt-force.
% !!!! remember that systems has to be build again when sample-time is changed !!!!
%

x1 = u(1);
x2 = u(2);
x3 = u(3)-1.5;  \% normalized to zeros for stiffness
x4 = u(4);
xk = [x1, x2, x3, x4]';

uk = u(5);
t

% load parameters
% global t1 t2 t3 t4
% global m_init p_init t_end ts_opt
% global Qoutput R_output
% global y_max y_min Lb du Ub du y_ref_value F_max F_min end_constraints_max
% global dis_gain_pos dis_gain_vel dis_gain_dstern dis_gain_vstern dead_time

% end load parameters

% prediction horizon and control horizon
m_end = round((t_end-t)/ts_opt);

47
p_end = m_end;

if m_end>m_init
    m = m_init;
    p = p_init;
else
    m = m_end;
    p = p_end;
end

if m<l
    m=1
end
if p<l
    p=1
end

% loading system matrices
global A1 A2 A3 A4 A5 B C Q R

C_opt = [];
for n = 1:length(Qoutput);
    if Qoutput(n) == 0;
        C_opt = [C_opt; C(n,:)];
    end
end

C_con = C;

nu_con = size(B,2);
nu_opt = size(B,2);
y_con = size(C_con,1);
y_opt = size(C_opt,1);

% filling rows till m=p
for h=0:m-1;
    % filling columns
    X = B;
    for g=h:p-1;

        Y_con(g*ny_con+1:(g+1)*ny_con,h*nu_con+1:(h+1)*nu_con) = C_con*X;
    end
end
if g\textasciitilde\text{ts\_opt} < t1-t
  X = A1\textasciitilde X+B;
else if g\textasciitilde\text{ts\_opt} < t2-t
  X = A2\textasciitilde X+B;
else if g\textasciitilde\text{ts\_opt} < t3-t
  X = A3\textasciitilde X+B;
else if g\textasciitilde\text{ts\_opt} < t4-t
  X = A4\textasciitilde X+B;
else
  X = A5\textasciitilde X+B;
end
end

Y\_opt = \text{zeros}(p\textasciitilde\text{size}(C\_opt,1),m\textasciitilde(nu\_opt));

for h=0:m-1
  \text{\% filling columns}
  X = B;
  for g=h:p-1
    Y\_opt(g\textasciitilde ny\_opt+1:(g+1)\textasciitilde ny\_opt,h\textasciitilde nu\_opt+1:(h+1)\textasciitilde nu\_opt) = C\_opt\textasciitilde X;
    if g\textasciitilde\text{ts\_opt} < t1-t
      X = A1\textasciitilde X+B;
    elseif g\textasciitilde\text{ts\_opt} < t2-t
      X = A2\textasciitilde X+B;
    elseif g\textasciitilde\text{ts\_opt} < t3-t
      X = A3\textasciitilde X+B;
    elseif g\textasciitilde\text{ts\_opt} < t4-t
      X = A4\textasciitilde X+B;
    else
      X = A5\textasciitilde X+B;
    end
  end
end

% filling of weighting matrix %
Qdiag_opt = [];
Qoutput_opt = [];
for n = 1:length(Qoutput)
    if Qoutput(n) ~= 0
        Qoutput_opt = [Qoutput_opt, Qoutput(n)];  % only weighted outputs
        % are taken into account
    end
end
for n=1:p
    Qdiag_opt = [Qdiag_opt, Qoutput_opt];
end
Q = diag(Qdiag_opt);
% for changing of input
Rdiag = [];
for n=1:m;
    Rdiag = [Rdiag, R_output];
end
R = diag(Rdiag);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% estimation for disturbances  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% y1_pred1 = predicted value (k+1|k)
% y1_pred2 = predicted value (k|k)

global start y1_pred  y2_pred y3_pred y4_pred  d_pos_old d_vel_old d_dstern_old d_vstern_old
global        PRED1
if start == 1
    d_pos = 0;
    d_vel = 0;
    d_dstern = 0;
    d_vstern = 0;
    PRED1 = zeros(4,4);
    % at end of file the predicted values are saved
    % start is put to zero at the end of the file
else

50
\[
d_{\text{pos}} = d_{\text{pos-old}} + \text{dis\_gain\_pos}\ast((x_1-PRED(1,1))+(x_1-PRED(2,1))+(x_1-PRED(3,1))+(x_1-PRED(4,1))); \\
d_{\text{vel}} = d_{\text{vel-old}} + \text{dis\_gain\_vel}\ast(x_2-y_2_{\text{pred}}); \\
d_{\text{dstern}} = d_{\text{dstern-old}} + \text{dis\_gain\_dstern}\ast(x_3-y_3_{\text{pred}}); \\
d_{\text{vstern}} = d_{\text{vstern-old}} + \text{dis\_gain\_vstern}\ast(x_4-y_4_{\text{pred}}); \\
\]

end

% for next sample, \( y_{1_{\text{pred}}} \) is calculated later
\[
d_{\text{pos-old}} = d_{\text{pos}}; \\
d_{\text{vel-old}} = d_{\text{vel}}; \\
d_{\text{dstern-old}} = d_{\text{dstern}}; \\
d_{\text{vstern-old}} = d_{\text{vstern}}; \\
\]

\%%% calculate free response %
\%
\% prediction start at \( (k+1|k) \)
\[
y_{\text{free_opt}} = []; \\
y_{\text{free_con}} = []; \\
y_{\text{ref}} = []; \\
d_{\text{vel_int}} = 0; \\
d_{\text{vstern_int}} = 0; \\
x_{k_{\text{old}}} = x_{k}; \\
for \( n=1:p \)
    if \( p\ast\text{ts\_opt} < t_{1-t} \)
        \( x_{k_{\text{new}}} = A_1\ast x_{k_{\text{old}}} + B\ast u_{k}; \)
    elseif \( p\ast\text{ts\_opt} < t_{2-t} \)
        \( x_{k_{\text{new}}} = A_2\ast x_{k_{\text{old}}} + B\ast u_{k}; \)
    elseif \( p\ast\text{ts\_opt} < t_{3-t} \)
        \( x_{k_{\text{new}}} = A_3\ast x_{k_{\text{old}}} + B\ast u_{k}; \)
    elseif \( p\ast\text{ts\_opt} < t_{4-t} \)
        \( x_{k_{\text{new}}} = A_4\ast x_{k_{\text{old}}} + B\ast u_{k}; \)
    else
        \( x_{k_{\text{new}}} = A_5\ast x_{k_{\text{old}}} + B\ast u_{k}; \)
    end
\]
\[
y_{\text{new_opt}} = C_{\text{opt}}\ast x_{k_{\text{new}}}; \\
y_{\text{new_con}} = C_{\text{con}}\ast x_{k_{\text{new}}}; \\
d_{\text{vel\_int}} = d_{\text{vel\_int}} + d_{\text{vel}}\ast \text{ts\_opt}; \\
d_{\text{vstern\_int}} = d_{\text{vstern\_int}} + d_{\text{vstern\_int}}\ast \text{ts\_opt}; \\
d_{\text{vel\_t\_con}} = [d_{\text{vel\_int}} + d_{\text{pos}}; d_{\text{vel}}; d_{\text{vstern\_int}} + d_{\text{dstern}}; d_{\text{vstern}}]; \\
d_{\text{vel\_t\_opt}} = d_{\text{vel\_int}}+d_{\text{pos}}; \\
\]

51
y_free_opt = [y_free_opt; y_new_opt + d_vel_t_opt];
y_free_con = [y_free_con; y_new_con + d_vel_t_con];
y_ref = [y_ref; y_ref_value];
xk_old = xk_new;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% making Y-constraints: Ax <=b
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y_max_vec = [];
y_min_vec = [];
for n=1:p
    y_max_vec = [y_max_vec; y_max];
y_min_vec = [y_min_vec; y_min];
end

y_con_max_b = y_max_vec - y_free_con;
y_con_max_A = Y_con;
y_con_min_b = -y_min_vec + y_free_con;
y_con_min_A = -Y_con;

% final state-constraint
if t>t_end - p*ts_opt;
    y_con_max_b(end-3:end) = end_constraints_max - y_free_con(end-3:end);
end

y_con_A = [y_con_max_A;
        y_con_min_A];
y_con_b = [y_con_max_b;
        y_con_min_b];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% making constraints on force
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
J = zeros(m,m);

for i = 1:m
    for j = 1:i
        J(i,j) = 1;
end
end

f_con_A = J;
f_con_b = (F_max - uk)*ones(m,1);

f_min_con_A = -J;
f_min_con_b = -(F_min - uk)*ones(m,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% optimization %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Lb = Lb_du*ones(m,1);
Ub = Ub_du*ones(m,1);

global duoptqp
%% for next optimization run
if start == 1
    duoptqp = ones(nu_con*m,1);
else
    duoptqp = duoptqp(1:m);
end

Hqp=Y_opt'*Q*Y_opt + R;
fpq=Y_opt'*Q*(y_free_opt - y_ref);

A = [y_con_A; f_con_A; f_min_con_A];
b = [y_con_b; f_con_b; f_min_con_b];

OPTIONS = optimset('Largescale','off','Display','iter');

%% dead-time

disp('deadtime')
if t<dead_time
    sys = [0 1];
else
    disp('Optimization begins')
    [duoptqp,FVAL,exitflag,output,LAGDA] = quadprog(Hqp,fpq,A,b,[],[],[],[],duoptqp,OPTIONS);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% information about active constraints %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

53
max_1_active = chooseoutput(LAMDA.ineqlin(1:4*p),4,1);
max_2_active = chooseoutput(LAMDA.ineqlin(1:4*p),4,2);
max_3_active = chooseoutput(LAMDA.ineqlin(1:4*p),4,3);
max_4_active = chooseoutput(LAMDA.ineqlin(1:4*p),4,4);
min_1_active = chooseoutput(LAMDA.ineqlin(4*p+1:8*p),4,1);
min_2_active = chooseoutput(LAMDA.ineqlin(4*p+1:8*p),4,2);
min_3_active = chooseoutput(LAMDA.ineqlin(4*p+1:8*p),4,3);
min_4_active = chooseoutput(LAMDA.ineqlin(4*p+1:8*p),4,4);
max_force   = LAMDA.ineqlin(8*p+1:8*p+m);
min_force   = LAMDA.ineqlin(8*p+m+1:8*p+2*m);

if size(find(max_1_active))~=0
    disp('maximum constraints on output 1 is active')
end
if size(find(max_2_active))~=0
    disp('maximum constraints on output 2 is active')
end
if size(find(max_3_active))~=0
    disp('maximum constraints on output 3 is active')
end
if size(find(max_4_active))~=0
    disp('maximum constraints on output 4 is active')
end
if size(find(min_1_active))~=0
    disp('minimum constraints on output 1 is active')
end
if size(find(min_2_active))~=0
    disp('minimum constraints on output 2 is active')
end
if size(find(min_3_active))~=0
    disp('minimum constraints on output 3 is active')
end
if size(find(min_4_active))~=0
    disp('minimum constraints on output 4 is active')
end
if size(find(max_force))~=0
    disp('constraints on maximum force is active')
end
if size(find(min_force))~=0
    disp('constraints on minimum force is active')
end
\[
\% \text{ calculating total input} \quad \%
\]

\[
\text{start} = 0;
\]

\[
\text{THETA} = \text{zeros(m,m)};
\]

\[
\text{for } j = 1:m
\]

\[
\quad \text{for } i = 1:m
\]

\[
\quad \quad \text{if } j >= i
\]

\[
\quad \quad \text{THETA}(j,i) = 1;
\]

\[
\quad \end
\]

\[
\end
\]

\[
u = u_k*\text{THETA}^\ast\text{duoptqp};
\]

\[
\text{sys} = [u(1),\text{exitflag}];
\]

\[
\text{output} = \text{Y}\_\text{con}\ast\text{duoptqp} + \text{y}\_\text{free}\_\text{con};
\]

\[
\text{y}\_\text{not}\_\text{free} = \text{Y}\_\text{con}\ast\text{duoptqp};
\]

\[
\text{output1} = \text{chooseoutput(output,4,1)};
\]

\[
\text{output2} = \text{chooseoutput(output,4,2)};
\]

\[
\text{output3} = \text{chooseoutput(output,4,3)};
\]

\[
\text{output4} = \text{chooseoutput(output,4,4)};
\]

\[
y_1\_\text{pred} = \text{output1}(1); \% \text{ predicts for next sample and not for sample at this moment}
\]

\[
y_2\_\text{pred} = \text{output2}(1);
\]

\[
y_3\_\text{pred} = \text{output3}(1);
\]

\[
y_4\_\text{pred} = \text{output4}(1);
\]

\[
\text{if } t<t\_\text{end}\_p\_ts\_opt
\]

\[
\quad \text{PRED1} = [\text{output1}(1:4)'; [\text{PRED1}(1:3,2:4), \text{zeros}(3,1)]]
\]

\[
\text{else}
\]

\[
\quad \text{PRED1} = \text{PRED1}
\]

\[
\end
\]

\[
\% y_1\_\text{pred}_{\text{kplus}1} = \text{output1}(2);
\]

\[
\% y_2\_\text{pred}_{\text{kplus}1} = \text{output2}(2);
\]

\[
\% y_3\_\text{pred}_{\text{kplus}1} = \text{output3}(2);
\]

\[
\% y_4\_\text{pred}_{\text{kplus}1} = \text{output4}(2);
\]

\[
\% y_1\_\text{pred}_{\text{kmini}1\_\text{old}} = y_1\_\text{pred}_{\text{kmini}}
\]

\[
\% y_2\_\text{pred}_{\text{kmini}1\_\text{old}} = y_2\_\text{pred}_{\text{kmini}}
\]

\[
\% y_3\_\text{pred}_{\text{kmini}1\_\text{old}} = y_3\_\text{pred}_{\text{kmini}}
\]

\[
\% y_4\_\text{pred}_{\text{kmini}1\_\text{old}} = y_4\_\text{pred}_{\text{kmini}}
\]

\[
\% !!! \text{ remind that output is not the measured output but the output with disturbance added}
\]
eval(['save output_ num2str(t) '.mat output1 '])
eval(['save output_ num2str(t) '.mat output2 -append'])
eval(['save output_ num2str(t) '.mat output3 -append'])
eval(['save output_ num2str(t) '.mat output4 -append'])
eval(['save output_ num2str(t) '.mat u -append'])
eval(['save output_ num2str(t) '.mat uk -append'])
eval(['save output_ num2str(t) '.mat xk -append'])
eval(['save output_ num2str(t) '.mat d_pos -append'])
eval(['save output_ num2str(t) '.mat d_vel -append'])
eval(['save output_ num2str(t) '.mat y_free_con -append'])
eval(['save output_ num2str(t) '.mat duoptqp -append'])
eval(['save output_ num2str(t) '.mat y_not_free -append'])
eval(['save output_ num2str(t) '.mat y_ref -append'])
end