Model Predictive Control of a Turbocharged Engine

IDA KRISTOFFERSSON

KTH Electrical Engineering

Masters' Degree Project
Stockholm, Sweden 2006

XR-EE-RT 2006:005

Model Predictive Control of a Turbocharged Engine

Master of Science Thesis
Performed at **S3**
for **General Motors Powertrain**
by
Ida Kristoffersson

Supervisors: **Richard Backman and Eric Wiklund**
General Motors Powertrain
Oscar Flärdh and Krister Jacobsson
Kungliga Tekniska Högskolan

Examiner: **Mikael Johansson**
Kungliga Tekniska Högskolan

Stockholm, April 28, 2006

Abstract

Engine control becomes increasingly important in newer cars. It is therefore interesting to investigate if a relatively new control method as Model Predictive Control (MPC) can be useful in engine control in the future. One of the advantages of MPC is that it can handle contraints explicitly.

In this thesis basics on turbocharged engines and the underlying theory of MPC is presented. Based on a nonlinear mean value engine model, linearized at multiple operating points, we then implement both a linear and a nonlinear MPC strategy and highlight implementation issues.

The implemented MPC controllers calculate optimal wastegate position in order to track a requested torque curve and still make sure that the constraints on turbocharger speed and minimum and maximum opening of the wastegate are fulfilled.

Keywords: Model Predictive Control, Turbocharged Engine, Mean Value Engine Modeling, Wastegate Control

Acknowledgement

This thesis is performed at General Motors Powetrain in Södertälje in cooperation with the Institute for signals, sensors and systems (S3) at KTH.

I would like to thank my supervisors at GM Powertrain, Richard Backman and Eric Wiklund, for support throughout the thesis work. Thanks also to everyone else at the department for engine algoritm development at GM Powertrain for a really nice time.

I also thank my examiner at KTH, Mikael Johansson, and the PhD students Oscar Flärdh and Krister Jacobsson for your time and good advises.

Contents

List of Figures

Abbreviations

BSR	Blade speed ratio
ECU	Electronic control unit
FMEP	Friction mean effective pressure
IMEP	Indicated mean effective pressure
MD	Measured disturbances
MIMO-system	Multi input multi output system
MPC	Model predictive control
mp-Qp	Multi-parametric quadratic programming
MO	Measured outputs
MV	Manipulated variables
MVEM	Mean value engine model
NMPC	Nonlinear model predictive control
PMEP	Pump mean effective pressure
rpm	Revolutions per minute
rps	Revolutions per second
TC-SI-ENGINE	Turbocharged spark ignited engine
UD	Unmeasured disturbances

Nomenclature

Variable/parameter	Explanation
u	Manipulated variable
u_{MD}	Measured disturbances
w	Unmeasured disturbances
ξ	Measurment noise
x	State vector
y	Output vector
J	Cost function
U	Optimization vector
$Hyst$	Hysteresis deviation
N_p	Prediction horizon
N_u	Control horizon
Q_u	Input weight
Q_x	State weight
Q_y	Output weight
S	Terminal weight
T	Sample time
t_f	Simulation time

Physical variable	Explanation	Unit
$(\frac{A}{F})_s$	Stoichiometric air/fuel ratio	-
E	Energy	J
$\dot{m}$	Mass flow	kg/s
M_e	Engine torque	Nm
N	Engine speed	rpm
N_s	Engine speed	rps
n_r	Engine revolutions per cycle	-
p	Pressure	Pa
q_{HV}	Heating value	J/kg
R	Gas constant	$J/(kg \cdot K)$
T	Temperature	K
V	Volume	m^3
W	Work	Nm
w_{tc}	Turbocharger speed	rad/s
α	Throttle area	mm^2
η	Efficiency	-
λ	Air/fuel ratio	-

Chapter 1

Introduction

1.1 Background

Engine control is very important in today's cars. Legal requirements on emissions result in complex engine designs that need to be controlled, partly to limit emissions but also to increase drivability and reliability. The rapid development of computer technology has allowed a dramatic increase in computer power in the cars during the last decade. All evidence suggests that the field of engine control will continue to increase in importance. Of particular interest is control based on physical models, since the use of physical models has the potential to reduce today's tedious and time-consuming tuning of look-up tables, drastically shortening the design cycle.

The combustion engine competes with other types of engines when it comes to fuel consumption and emissions. One way to increase the efficiency of a combustion engine is to add a turbocharger. The turbocharger connects a turbine placed before the exhaust system to a compressor placed after the air filter. This way the energy in the exhaust gases is utilized to compress the incoming air more and the engine can produce more torque. With a turbocharger the size of the engine can be reduced without a loss in the amount of torque produced (downsizing). A smaller engine is better from a fuel consumption point of view. Mounted on the turbocharger is a valve called the wastegate. If it is open the exhaust gases flow through the wastegate instead of through the turbine. The wastegate is opened if the turbine is close to rotating too fast.

Model Predictive Control (MPC) is a control method that uses a model to predict future behavior of a system. In each time step an optimization problem is solved for a certain number of samples ahead called the prediction horizon. The solution to the optimization problem is a vector of input signals to the system that minimizes a chosen performance criterion without violating any of the constraints on the system. An advantage of MPC is that constraints can be accounted for already when the input signal is computed. So far MPC has been used mostly in the petrochemical industry. In this sector it is not a big problem that it takes time to solve the optimization problem, since the chemical processes are slow.

MPC has in this thesis been applied to a turbocharged engine. The criterion minimized when solving the optimization problem consists of expressions for fuel consumption and deviation from requested torque. A physical model of the engine is utilized to predict future behavior. The model of the turbocharged engine has several input signals, but the MPC controller only calculates optimal wastegate position. The calculation of the optimal wastegate position is done considering constraints on how much the wastegate can be opened and the constraint that the turbine is not allowed to rotate too fast.

1.2 Objective

The objective of this thesis is to develop an MPC controller that controls the trade-off between requested torque tracking performance and fuel consumption of a turbocharged engine and to prepare for a real time implementation. The MPC controller shall calculate optimal wastegate position and thereby give a hint about if it is beneficial to open the wastegate when a step is made in requested torque.

Since very little work has been done in implementing MPC-controllers for large and fast systems that switch between different operating conditions, the objective of this thesis is more to make a thorough investigation than to give better control performance than already available by other controllers today. Infact, when dealing with changing operating conditions one is in the field of nonlinear MPC, which is not as thoroughly investigated as linear MPC.

1.3 Prior Work

In the master thesis (Wa03) an MPC-controller for a turbocharged diesel engine is developed. The controller is however not implemented in real-time in a car. A cruise controller for heavy vehicles is developed with MPC in the master thesis (Ax03). This controller is also not tested in real time. In her master thesis (Ed04) Camilla Edberg investigates the possibilities of Model Predictive Control and a special method called explicit MPC, which aims at reducing the on-line computations. (Ma02) gives a thorough descripion of MPC, the extentions possible and issues as stability. For short introductions to MPC (Gl02) and (Gl03) are recommended.

1.4 Thesis Method

To get the required knowledge about MPC and turbocharged engines, the thesis work started with literature studies on these subjects. The literature consisted of course literature from continuation courses in control, earlier master thesises and books about vehicular systems. When implementing the MPC controller, the tutorial coming along with MPC toolbox was of great assistance.

An MPC controller needs a model to predict future response of the system. The model used in this thesis was developed by Per Andersson in his doctoral thesis (An05). One part of the thesis work was to get acquianted with this model.

The implementations of the MPC controller were made in Matlab and in Simulink. In Matlab the controllers were implemented by extending a simple MPC algorithm from (S3), whereas the MPC block from the MPC toolbox was used in the Simulink implementations.

1.5 Thesis Outline

Chapter 2 describes the basics of a turbocharged engine and the positive and negative effects of adding a turbocharger to an engine. In Chapter 3 the theoretical fundations of model predictive control are presented. Here the reader finds the general MPC problem formulation and information about tuning parameters. After the background in the first two chapters the specific problem is introduced in Chapter 4. In this chapter the nonlinear and linear models of the engine are presented and it is shown how the cost function and constraints are formulated. Chapter 5 and 6 describes the implementation in Matlab and Simulink respectively. The results are presented in Chapter 7 and finally in Chapter 8 conclusions are drawn and future work discussed.

Chapter 2

Basics of a TC-SI-Engine

In this chapter the main sources are the course literature (Er05), the webpage (HSW) and the master thesis (Fo05).

2.1 Spark Ignited Combustion Engines

The idea of a combustion engine is to produce power (torque) from the energy produced when a mixture of air and fuel burns. Unfortunately, power is not the only thing produced, the engine also generates emissions. One type of emissions is unburned hydrocarbons from the fuel. It is therefore important that the ratio between air and fuel is right. For a complete combustion of 1 kg gasoline, about 15 kg air is needed. This is the so called stoichiometric air/fuel ratio $\left(\frac{A}{F}\right)_s$. With the measure $\lambda = \left(\frac{A}{F}\right) / \left(\frac{A}{F}\right)_s$ the present air/fuel ratio is compared to the stoichiometric ratio. If $\lambda > 1$ there is excess air in the mixture and it is said to be lean. If, on the other hand, $\lambda < 1$ there is excess fuel in the mixture and it is said to be rich.

In a combustion engine the same course of events takes place over and over again. It is therefore enough to study one cycle. In each cycle one combustion occurs. The most common cycle is the four stroke cycle, which can be devided in the parts *intake*, *compression*, *expansion* and *exhaust*. During the intake stroke air and fuel flow into the cylinder from the intake manifold. In a spark ignited engine a spark ignites the air/fuel mixture during the compression stroke. The combustion goes on a bit into the expansion stroke. During the exhaust stroke the fluid in the cylinder is pushed out into the exhaust system.

2.2 Different Superchargers

In the literature different meanings of supercharging and turbocharging occur. Here supercharging will be used as the collected name for all methods that charges extra air into the cylinder. Extra air in the cylinder makes an increase in fuel possible, while still keeping the same air/fuel ratio. More fuel implies more power from each combustion in the cylinder. This way a supercharger increases the power-to-weight ratio of the engine.

With the terms used as above, turbocharging is one of the methods to supercharge. Another method is mechanical supercharging, where a separate compressor or pump, driven by the engines crank shaft, is used to compress the air. Mechanical supercharging leads to good performance, but increases the fuel consumption.

With turbocharging the energy in the exhaust gas is utilized by a turbine. The turbine drives a compressor which compresses the air. Except for increased pumping losses the turbocharger does not directly consume engine power, but it is not able to deliver extra power at low engine speeds.

2.3 How a Turbocharged Spark Ignited Engine is Build Up

In a turbocharged engine a turbine and a compressor is added to the engine, see Figure 2.1. They are connected by a shaft and the turbine, shaft and compressor are together called a turbocharger. The most common turbocharger is build up by a centrifugal compressor and a radial turbine. In a radial turbine the exhaust gases leave in a direction perpendicular to the entering direction. A centrifugal compressor has blades that are orthogonal to the entering air. The turbine wheel is driven by waste exhaust gases. Because of the connecting shaft the turbine and compressor wheels rotate simultaneously. Air is drawn into the compressor by rotation of the compressor wheel.

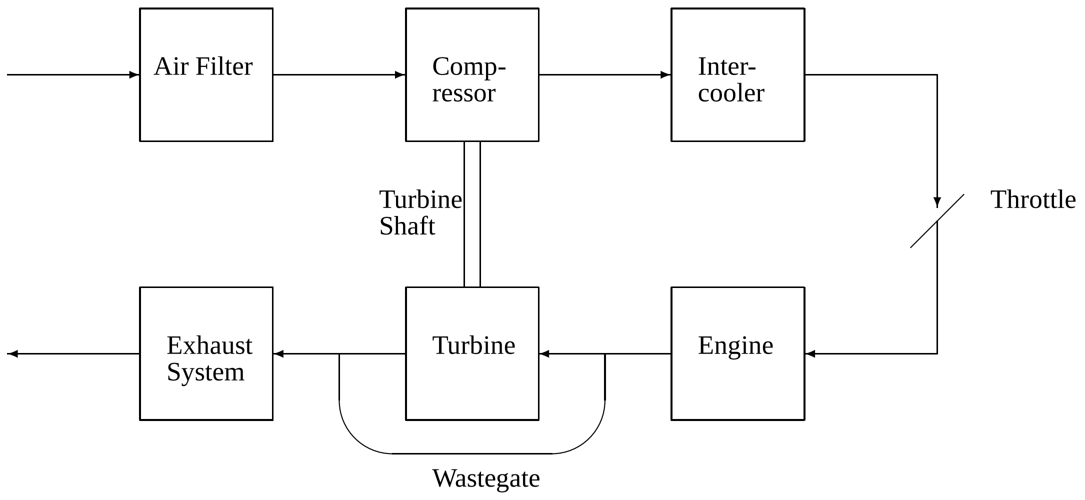

Figure 2.1: Schematic figure of a turbocharged engine

The arrows in the figure show the air mass flow, $\dot{m}_{air}$, through all engine parts. The air flows in through the air filter and is compressed in the compressor. The pressure is now higher than the ambient pressure and is called the boost pressure. When the air is compressed, not only the pressure rises but also the temperature. The air is therefore cooled in an intercooler. The throttle then decides how much air is blown into the engine. The intake and exhaust manifolds are not shown in the figure. They are placed before and after the actual engine (the cylinders with pistons and the crankshaft) respectively. The pressure in intake and exhaust manifold, p_{im} and p_{em}, decide to a great extent the resulting engine torque and fuel consumption.

The fuel is injected between the intake manifold and the cylinder, a so called port injection. Some times the fuel is injected directly into the cylinder next to where the spark plug is located, a so called direct injection. The amount of fuel is implicitly decided by the throttle to achieve a correct air/fuel ratio. The air/fuel mixture is combusted according to section 2.1. Then the exhaust gases flow out through the exhaust manifold and, if the wastegate is closed, into the turbine which starts to rotate through the energy in the exhaust gases. The turbine drives the compressor through a shaft. Finally the exhaust gases flow through an exhaust system (a catalyst) and out in the air.

The Wastegate

The *wastegate* is a valve that bypasses the turbine. Figure 2.1 shows that when the wastegate is opened the gases do not flow through the turbine. This results in less power to the turbine and the turbine speed decreases, which in its turn results in a lowered intake manifold pressure, p_{im}. In the long run this influences the torque, but the effect is delayed and there is no direct connection between the wastegate and the torque produced.

One way the wastegate is used is to avoid that the turbine spins to fast. Another purpose of the wastegate is to avoid a phenomenon called knock. Knock occurs when the air/fuel mixture auto ignites and burns before the flame front, which can cause damages to the engine. There is a larger risk for knock when a turbocharger is used because of the boost pressure.

Today the wastegate is a mechanical valve opened by the differance in compressor pressure and ambient pressure. Therefore, the wastegate can only be opened if this pressure difference is large enough, about 10kPa, which is the case for engine torque values of around 130 Nm or more. If an electrical valve would be used instead this would not be an issue.

2.4 Properties of the Turbocharger

The word turbo comes from latin and means ”that which spins”. The turbine in a car engine spins at speeds up to 180,000 rotations per minute (rpm). Most car engines rotate 30 times slower. The temperature in the turbine is very high, since warm exhaust gases flow through it. The high speeds and temperatures result in demanding design and material, which in its turn makes the turbocharger expensive. For example a fluid bearing is needed to support the turbine shaft at high speeds.

There are both advantages and disadvantages with a turbocharger. A negative effect important for this thesis is the so called turbo lag. With turbo lag is meant the extra time it takes from when the driver steps on the gas to when the driver feels the acceleration compared with a naturally aspirated engine (i.e. air flows through an air filter and directly into the cylinder). The reason for turbo lag is that it takes some time before the turbocharger speed is build up. When the turbo gets moving the car lunges ahead. One way to decrease the turbo lag is to use a smaller turbine with less inertia. With a smaller turbine there is a danger that the turbine spins to fast at high engine speeds. To avoid this the wastegate is opened.

Since it is popular to add a turbocharger to an engine there must also exist some positive effects. These are among others:

- 30 to 40 percent extra power compared with naturally aspirated engines of the same size,
- lower fuel consumption compared with naturally aspirated engines that deliver the same torque due to the higher effiency of a smaller engine,
- less exhaust noise and emissions and
- less reduction of power on high altitudes because thinner air is easier for the turbocharger to compress.

Chapter 3

Introduction to Model Predictive Control

3.1 Why Use MPC?

Normally PID-contol is the control method used in industry. Of the more advanced control methods, the one which is most widely used is Model Predictive Control (MPC). The reasons for this are that it can handle safety constraints and that the idea behind the method is easy to understand. Another reason for its success is that systems with more than one input and output (MIMO-systems) can be handled. MPC also makes it possible to operate closer to constraints than conventional control, which often implies a larger profit.

3.2 General MPC Setup

MPC is a generic term for computer control algorithms that utilize an explicit process model to predict future response of the plant.

An optimal input is computed by solving an open-loop optimal control problem over a finite time horizon, i.e. for a finite number of future samples. The number of samples one looks ahead is called the prediction horizon N_p. In some MPC formulations a difference is made between prediction horizon and control horizon N_u. The control horizon is then the number of samples that the optimal input is calculated for. With a shorter control horizon than prediction horizon the complexity of the problem can be reduced.

From the calculated input signal only the first element is applied to the system. This is done at every time step. The idea is thus to go one step at a time and check further and further ahead. The method can be described as ”repeated open-loop optimal control in feedback fashion” (Fi04).
In an MPC-algorithm there are three important elements:

- the **model** used for predictions,
- the **cost function** and
- the **constraints**.

The MPC-algorithm can be applied to two different types of control problems:

- **regulator problem**, i.e. to maintain the process at a desired steady state and
- **servo problem**, i.e. to move rapidly from one operating point to another.

Model

The model is in most MPC-formulations today given on discrete time state space form:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + B_{MD}u_{MD}(k), \\ y(k) &= Cx(k). \end{aligned} \tag{3.1}$$

Here $x(k)$ is the state vector, $u(k)$ the input vector, u_{MD} is called the vector of measured disturbances, i.e. input signals that are not calculated by the controller, and $y(k)$ is the output vector.

As can be seen, the model is linear. Nonlinear models can also be handeled by MPC, but the computational effort needed is then even larger. For such fast processes as automotive ones it is appropriate to start with a linearized model.

It is important to note that even though the model is linear the MPC-controller will be nonlinear since it accounts for the constraints.

Cost Function

The cost function is designed differently depending on what to minimize. Common is a quadratic cost function which penalizes both deviation from a state reference and rapid changes in the control signal:

$$\min\, (x_{N_p} - x_{ref,N_p})^T S (x_{N_p} - x_{ref,N_p}) + \sum_{i=0}^{N_p-1} \left[(x_i - x_{ref,i})^T Q_x (x_i - x_{ref,i}) + \Delta u_i^T Q_u \Delta u_i \right] \tag{3.2}$$

where $\Delta u_i = u(k+i) - u(k+i-1)$. A penalty on Δu_i punishes rapid changes in the input signal, which can be used to reduce oscillations. According to (Wa03) a penalty on rapid changes in the input signal also introduces integral action. However, this is coupled to the prediction horizon - stationary errors can appear even if integral action is introduced via a penalty on Δu_i if the prediction horizon is not long enough.

In the cost function stated above, the prediction horizon (N_p) and the control horizon (N_u) are the same. Instead of a shorter control horizon there is a penalty on x_{N_p}, which plays a similar role. MPC-toolbox, which will be used for implementation in Simulink, uses the formulation where N_u are distinct from N_p. The matrices S, Q_x and Q_u are weight matrices who decide the penalty on each term in the cost function. Most effort is put on minimizing the term with largest penalty. The cost function of this thesis is set up in Section 4.5.

Constraints

Desired to easily solve the MPC problem is a so called Quadratic optimization Problem (QP). A QP is a convex problem, i.e. if a solution is found uniqueness is guaranteed. To get a QP the constraints need to be linear. They are thus on the form:

$$\begin{aligned} x0 &= given, \\ u_{min} \leq u_i \leq u_{max},\ i &= 0, ..., N_p - 1, \\ y_{min} \leq W x_i \leq y_{max},\ i &= 1, ..., N_p, \end{aligned} \tag{3.3}$$

where the matrix W forms the output from the states.

Optimization Problem and Algorithm

The optimization vector is $U = [u^T(k), ..., u^T(k + N_p - 1)]^T$. If a shorter control horizon than prediction horizon is used, it is assumed that $u(k + i) = u(k + N_u - 1)$ for all $i \geq N_u$. The problem now needs to be rewritten in terms of U only. This is in principle straightforward since:

$$x(k) = A^k x(0) + \sum_{j=0}^{k-1} A^{k-1-j}(Bu(j) + B_{MD}u_{MD}(j)), \tag{3.4}$$

but requires extensive matrix manipulations, see (S3) and Section 5.3. Finally the optimization problem becomes:

$$\begin{array}{cc} \overset{min}{U} & U^T HU + h^T U, \\ \text{s.t.} & LU \leq b, \end{array} \tag{3.5}$$

where the matrices H and L and the vectors h and b are build up by $x_{ref}, u_{ref}, x_0, Q_x, Q_u, S, A, B$ and C. The MPC-algorithm can now be summarized as:

1. Measure the current state $x(k)$ or estimate it using an observer.
2. Solve the k:th optimization problem to obtain $U = [u^T(k), ..., u^T(k + N_p - 1)]^T$.
3. Apply $u(k)$ to the system.
4. Update time $k = k + 1$ and repeat from step 1.

Observer

As mentioned above, an observer might be needed to estimate the states. (Ma02) describes an observer as a copy of the system with a feedback from the measured system output $y(k)$, via a gain matrix L, to get a better value on the estimated $\hat{x}(k)$. The equations for the observer are thus:

$$\begin{aligned} \hat{x}(k|k) &= \hat{x}(k|k-1) + \tilde{L}(y(k) - \hat{y}(k|k-1)), \\ \hat{x}(k+1|k) &= A\hat{x}(k|k) + Bu(k), \\ \hat{y}(k|k-1) &= C\hat{x}(k|k-1). \end{aligned} \tag{3.6}$$

Substituting the third equation into the first and then the first into the second leads to:

$$\begin{aligned} \hat{x}(k+1|k) &= (A - LC)\hat{x}(k|k-1) + Bu(k) + Ly(k), \\ L &= A\tilde{L}. \end{aligned} \tag{3.7}$$

An observer to the model used in this thesis has been developed by Per Andersson in (An05). If the MPC controller is implemented in real time, this observer will be used to get the output signals needed from the model.

3.3 Tuning Parameters

The tuning parameters that exist in the MPC-controller is the prediction horizon - N_p, the control horizon N_u or the penalty matrix S and the weighting matrices Q_x and Q_u.

Since a reduction of the prediction horizon reduces the computational time, it is of interest to keep N_p as short as possible. The risk of a small value on N_p is that a short prediction horizon makes the controller aggressive, which can lead to large oscillations. The system can then be stabilized by a large value on the terminal state weight S, but it is here important to be careful since a change in size of the step height can ruin the stability again. A longer prediction horizon and no penalty on the terminal state is therefore better if it is possible.

If the optimization problem does not have a solution that satisfies all constraints the problem becomes infeasible and the computations break down. This is more likely to occur with a short prediction horizon because of bad planning. There is no risk of an infeasible problem if the constraints are softened. Soft constraints means that small violations are accepted if necessary. A big penalty on the constraint violations in the cost function ensures that no unnecessary violations occur.

To reduce computational time the control horizon can be chosen shorter than the predictive horizon. A short N_u reduces the number of free variables, which makes the computational time shorter. However, a controller with a long N_u is closer to the optimal infinite-horizon controller.

If a certain output performance is desired the weight is Q_y. The state weight Q_x is then obtained by:

$$Q_x = C^T Q_y C. \tag{3.8}$$

How much effort that should be put on good performance and input minimization respectively is adjusted with the relative difference in size of Q_y and Q_u. If there are several outputs that should follow references, Q_y consists of diagonal elements q_p and the relative sizes of these elements decide which output performance is most important. A large value on Q_u penalizes rapid changes in the control signal alot.

3.4 Explicit MPC

Explicit MPC can be used to make MPC applicable not only to slow systems. The on-line optimization is replaced by a simple function evaluation. The computations needed for implementation of the MPC-controller are thus moved and made off-line, but all characteristics of the controller are preserved (Be05).

The main part of linear MPC is the solution of a quadratic optimization problem to determine future control action. The quadratic optimization problem is often called a quadratic program (QP). In the quadratic program of MPC the cost function and the right hand side of the constraints depend linearly on the states. This makes it possible to see the QP as a multi parametric quadratic program (mp-QP). The number of states, reference signals, previous control inputs and measured disturbances are seen as a vector of parameters θ. In (Be00) it is shown that the solution to an mp-QP is a piecewise affine function of this vector of parameters:

$$z(\theta) = F_i\theta + G_i, \tag{3.9}$$

where i is the region index.

When computing an explicit controller the mp-QP is solved for different regions in the state space. This results in one piecewise affine control law for each region as can be seen in Equation 3.9. It is therefore important to keep the number of regions as small as possible. The number of regions depends on:

- the dimension of the parameter vector,
- the control horizon,
- the number of constraints and
- the size of the range within which the partition is made.

The regions that the state space is divided into are so called polytopes. A polytope is defined as a bounded intersection of a finite number of hyperplanes. A polytope can be described through the H-representation, i.e. providing the intersecting hyperplanes, or through the extreme points (vertices) by computing the convex hull of them. An example of a partition into polytopes is shown in Figure 3.1.

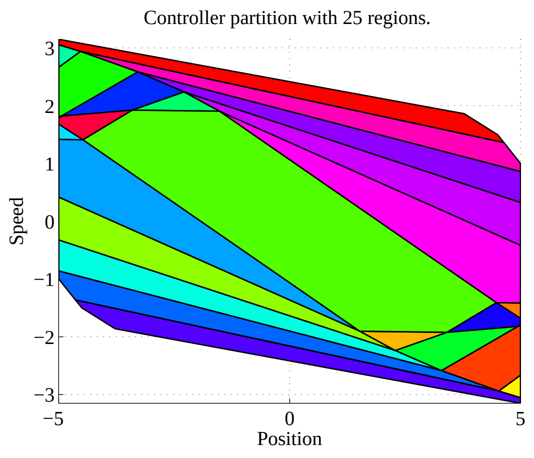

Figure 3.1: Partition into polytopes

It is appropriate to first tune the ordinary MPC-controller. Then, if the explicit formulation has too many regions, the MPC-design is changed, for example the control horizon can be shortened.

A drawback with explicit MPC is that when one has many states, measured disturbances and referens signals the complexity of the solution grows very much.

3.5 Nonlinear MPC

In this section the main source is (Ca01).

Nonlinear model predictive control (NMPC) is more complex than linear MPC discussed above. Since the engine processes are so fast, already linear MPC is computationally difficult to implement in real time. The control problem of this thesis is one of those problems where NMPC is motivated - it fits under the second item below.

NMPC is motivated for:

1. regulator control problems with a highly nonlinear process and large frequent disturbances
2. servo control problems where operating points change often and span a wide range of nonlinear dynamics.

The definition of NMPC is that a nonlinear model is utilised to predict future outputs. The model is on the form:

$$\begin{aligned} x(k+1) &= f\left(x(k), u(k), u_{MD}(k), w(k)\right), \\ y(k) &= g(x(k)) + \xi(k), \end{aligned} \tag{3.10}$$

where $w(k)$ is an unmeasured disturbance and $\xi(k)$ is measurement noise.

A nonlinear model will not be used for prediction in this thesis, but the MPC controller will switch between several linear models in different operating points. In this way the MPC strategy is nonlinear.

One problem with NMPC is that introduction of a nonlinear model leads to loss of convexity, i.e there exists several local optima and one can not be sure that the optimum found is the global optimum. In this thesis the MPC controller uses a linear model in the present operating point to calculate optimal input signal. The algorithm is guaranteed to find a local optima valid for this operating point. When the algorithm switches between operating points stability is not guaranteed. In the case of linear MPC a prediction horizon long enough guarantees stability. This is not the case when several linear models are used. A too long prediction horizon might predict wrong because it looks into a field of another operating point.

The major limiting factors in existing NMPC applications are speed and assurance of a reliable solution in real time. Due to the computational complexity of the NMPC algorithms the size of NMPC applications are typically much smaller than those of linear MPC. The NMPC products sold today do not have a simple user interface and are said to be as complex to the user as linear MPC products was in the mid 1980s.

Chapter 4

Wastegate Control via MPC

4.1 Idea

The goal of this thesis is to develop an MPC controller that calculates the optimal wastegate position, considering the trade-off between torque reference tracking and keeping the fuel consumption down. Today the wastegate is closed during a transient. If it is useful to open the wastegate or not during a transient will be investigated in this thesis.

To open the wastegate at the same time as the throttle opens (at a transient) could have positive effects on the engine torque since it reduces the pumping losses in the engine. Opening the wastegate also makes it easier for the turbine to speed up, because of the reduced exhaust manifold pressure. Another reason for that opening the wastegate could have positive effects is that the efficiency of the turbine depends on the so called blade speed ratio (BSR). In this ratio the turbine wheel tip speed is compared to the speed of the fluid entering. If the turbine wheel moves much faster or slower than the fluid the kinetic energy in the fluid can not be utilized in an optimal way. In a performance plot for the turbine the efficiency is plotted versus the blade speed ratio:

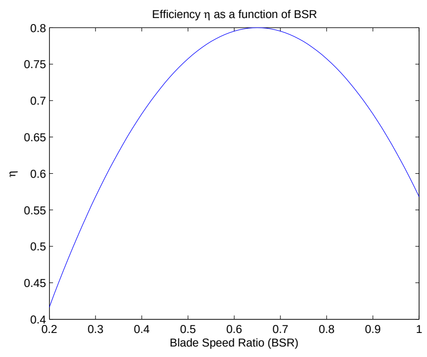

Figure 4.1: Efficiency is dependent on blade speed ratio

As can be seen in the figure the efficiency is optimized for a BSR of about 0.6.
However, the energy is dependent on both the efficiency and the mass flow through the turbine:

$$E = \dot{m}_t \eta \Delta T. \tag{4.1}$$

Since the mass flow through the turbine decreases when opening the wastegate it is not certain that the energy will increase even if the efficiency increases. As said above this will be investigated.

4.2 Nonlinear Predictive Model

The model used in this thesis is a so called mean value engine model (MVEM) developed by Andersson in (An05). See also Figure 4.2. In a mean value engine model the signals, parameters and variables are averaged over one or several cycles. In the model developed by Andersson the physical structure of the engine is preserved, since the model consists of blocks representing the engine components.

The model follows a simple pattern where a control volume is placed after a restriction. Examples of restrictions are air filter, intercooler and throttle. These components all give rise to pressure losses. Often a static relation is enough to model a restriction, but for example the throttle motion needs to be modeled as a differential equation. There are two types of restrictions: the ones that model incompressible flows and the ones that model compressible flows. Components where the gas velocity is high, for example the throttle, has to be modeled as compressible restrictions. The incompressible flow can be divided in the subcategories laminar and turbulent flow. Most flows occuring in the pipes of an engine are turbulent.

The intake and exhaust manifolds are examples of components that can be modeled as control volumes. A control volume is characterized by its volume and its filling and emptying dynamics. The dynamics, i.e. the differential equations, can be derived from conservation of mass and energy. Since mass and energy are difficult to measure, the differential equations are rewritten in pressure and temperature which is much easier to measure. When rewriting the assumption of ideal gas is usually made:

$$pV = mRT. \tag{4.2}$$

The MVEM model used in this thesis has thirteen states and six input signals, see Table 4.1.

States x		Inputs u	
$x_1 = p_{af}$	Air Filter Pressure $[Pa]$	$u_1 = \alpha$	Throttle Area $[mm^2]$
$x_2 = T_{af}$	Air Filter Temperature $[K]$	$u_2 = N$	Engine Speed $[rpm]$
$x_3 = p_{comp}$	Compressor Pressure $[Pa]$	$u_3 = u_{wg}$	Wastegate Position $[-]$
$x_4 = T_{comp}$	Compressor Temperature $[K]$	$u_4 = \lambda$	Air/fuel ratio $[-]$
$x_5 = p_{ic}$	Intercooler Pressure $[Pa]$	$u_5 = p_{amb}$	Ambient Pressure $[Pa]$
$x_6 = T_{ic}$	Intercooler Temperature $[K]$	$u_6 = T_{amb}$	Ambient Temperature $[K]$
$x_7 = p_{im}$	Intake Manifold Pressure $[Pa]$		
$x_8 = T_{im}$	Intake Manifold Temperature $[K]$		
$x_9 = p_{em}$	Exhaust Manifold Pressure $[Pa]$		
$x_{10} = T_{em}$	Exhaust Manifold Temperature $[K]$		
$x_{11} = p_t$	Turbocharger Pressure $[Pa]$		
$x_{12} = T_t$	Turbocharger Temperature $[K]$		
$x_{13} = w_{tc}$	Turbocharger Speed $[rad/s]$		

Table 4.1: States and inputs of the predictive model

The only input signal calculated by the MPC controller will be the wastegate position $u_3 = u_{wg}$. The reason is to keep the size of the already big MPC problem down. The other input signals are seen as measured disturbances. Since the MPC controller only can effect the wastegate and the effect of the wastegate on the torque is limited (see Section 2.3), the conditions for control are not optimal.

The intake manifold pressure, the exhaust manifold pressure and the turbocharger speed, x_7, x_9 and x_{13}, are the most important states for this thesis. In Section 4.5 the intake manifold pressure and the exhaust manifold pressure will be used to formulate expressions for fuel consumption and engine torque. The limit on the turbocharger speed will be the most important constraint in Section 4.6.

The intake manifold temperature, x_8, is defined as constant in Sec. 4.5 to get a linear cost function. This approximation is a bit rough. The intake manifold temperature does normally not vary much, but during a transient it has a peak. The reason for this peak is that the gas in the intake system is compressed if the throttle is opened fast.

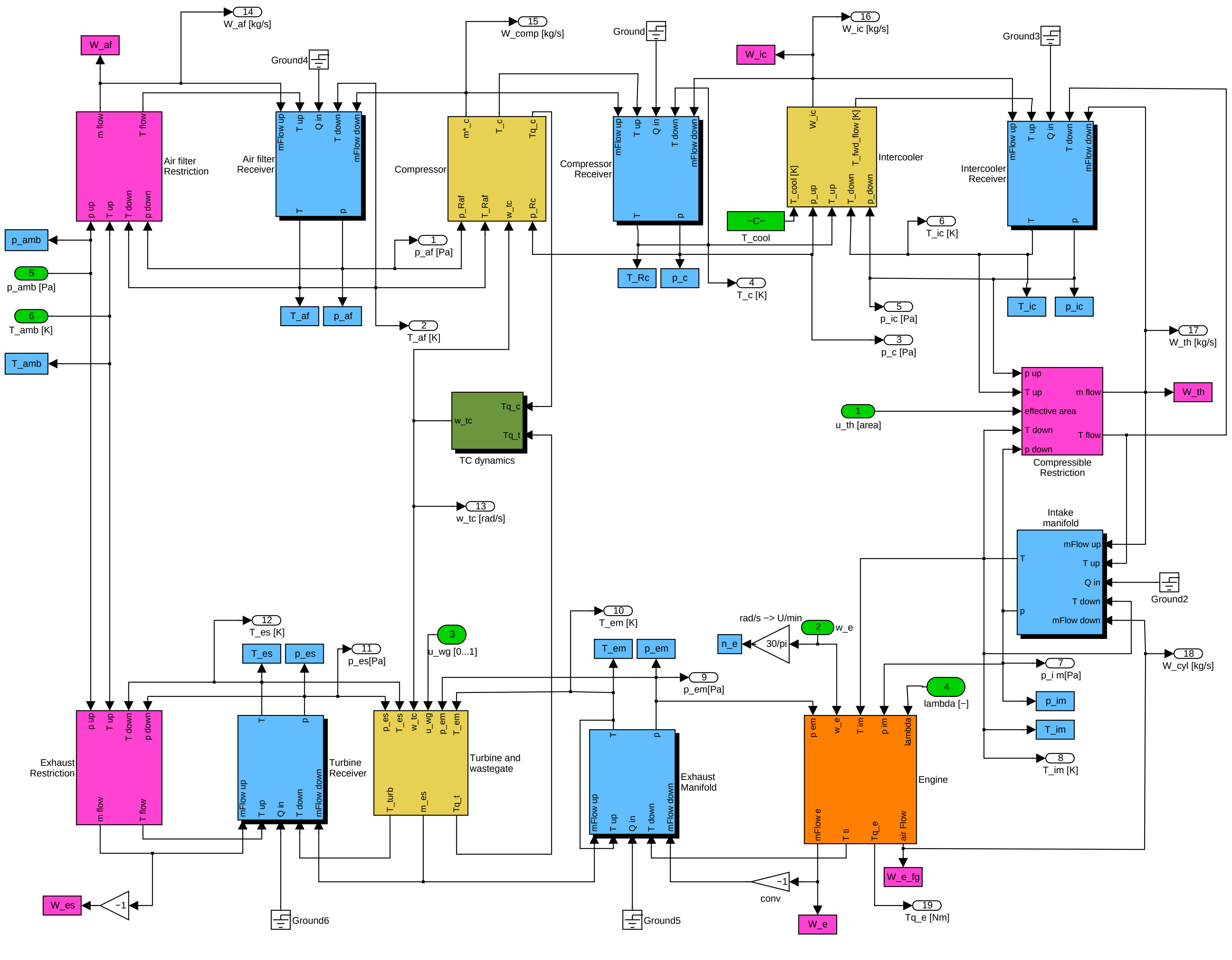

Figure 4.2: MVEM-model of a turbocharged SI-engine

4.3 Linear Predictive Model

The model in Section 4.2 is nonlinear and implemented in Simulink. Also developed by Andersson is a linearization of the model in 45 operating points. The linear models are on the form:

$$\begin{aligned} \dot{\tilde{x}} &= A_{num}\tilde{x} + B_{num}\tilde{u}, \\ y &= C_{num}\tilde{x} + D_{num}\tilde{u}, \end{aligned} \tag{4.3}$$

where $\tilde{x} = x - x_{stat}$, $\tilde{u} = u - u_{stat}$ and num is the operating point number ranging from 1 to 45. The operating points cover engine speeds in the range of $1200\ rpm$ to $4700\ rpm$. As can be seen in the tabulars below, the wastegate was not always closed when the linear matrices were set up. This leads to some difficulties when controlling the system, see Section 4.4.

In Table 4.2 $p_{im,stat} = x_{stat}(7)$, $\alpha_{stat} = u_{stat}(1)$ and $u_{wg,stat} = u_{stat}(3)$ are presented for the seven operating points with the engine speed $2000\ rpm$.

Operating point number	$\alpha_{stat}\ [mm^2]$	$p_{im,stat}\ [Pa]$	$u_{wg,stat}\ [-]$
12	38.8	37260	0
13	64.5	58600	0
14	91.8	78710	0
15	127.8	100560	0
16	537.9	121800	0.14
17	1096.2	144520	0.05
18	1263.7	156680	0.03

Table 4.2: Stationary conditions for engine speed 2000 rpm

For the operating points in Tab. 4.2 frequency responses are plotted. The C-matrix from Section 5.3 is used as output matrix. Since the models are multi-input-multi-output (MIMO) systems the singular values of the systems are plotted instead of Bode diagrams, see Figure 4.3.

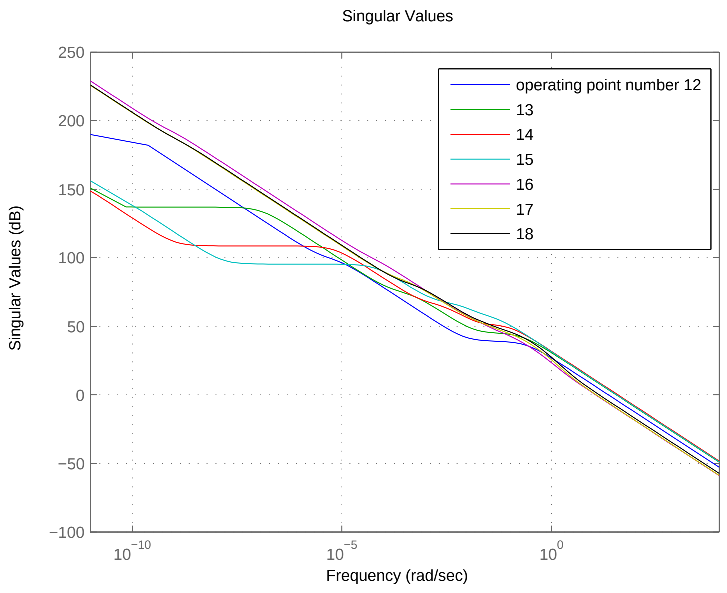

Figure 4.3: Frequency responses of the linear system in operating points 12 to 18

The frequency responses are rather similar. They only differ for low frequencies. Therefore one or two operating points should be enough to cover the dynamics of $2000\ rpm$ if integral action is used.

In Table 4.3 $p_{im,stat} = x_{stat}(7)$, $\alpha_{stat} = u_{stat}(1)$ and $u_{wg,stat} = u_{stat}(3)$ are presented for the seven operating points with the engine speed $2600\ rpm$.

Operating point number	$\alpha_{stat}\ [mm^2]$	$p_{im,stat}\ [Pa]$	$u_{wg,stat}\ [-]$
19	50.9	37810	0
20	78.5	58040	0
21	103.4	79170	0
22	168.0	101300	0.2
23	1206.5	125540	0.35
24	1702.6	146610	0.26
25	1705.0	155430	0.25

Table 4.3: Stationary conditions for engine speed 2600 rpm

For operating points 19 to 25 frequency responses are again plotted, see Figure 4.4.

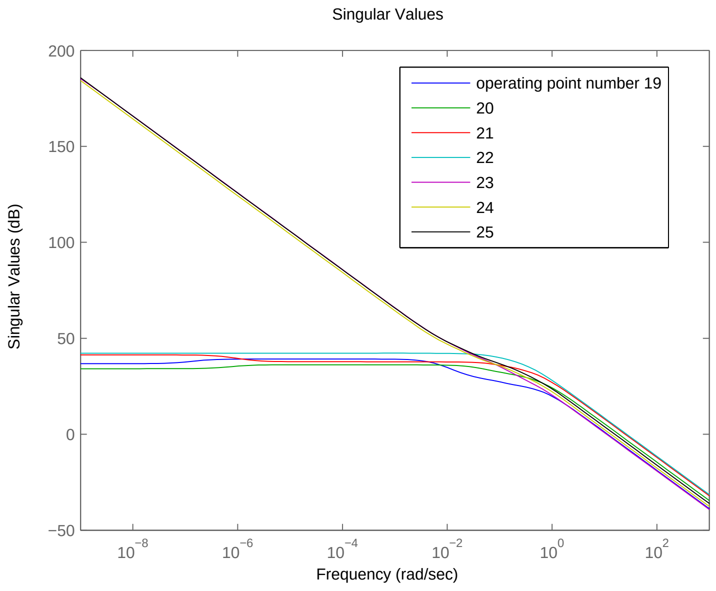

Figure 4.4: Frequency responses of the linear system in operating points 19 to 25

These singular value curves differs more. Reasonable here would be to use two operating points to cover the dynamics of $2600\ rpm$.

4.4 Difference Between Linear and Nonlinear Model

In Table 4.2 and Table 4.3 one can see that the linear systems for operating points at high loads have been derived at an operating condition with open wastegate. The reason is that the wastegate has been used as a tuning parameter to get the desired intake manifold pressure.

This leads to problems when the wastegate is used as manipulated variable for control. To highlight the problem simulations in operating point 18 are carried out. In this operating point the wastegate was somewhat open ($u_{wg} = 0.03$) when the linear system was derived. This results in a difference between the linear and nonlinear model. With the same input signals u=[$\alpha = 1263.7\,mm^2$, $N = 2000\,rpm$, $u_{wg} = 0$, $\lambda = 1$, $p_{amb} = 101300\,Pa$, $T_{amb} = 298\ K$] the two models give the results shown in Figure 4.5. Observe that the same input signals were used and that simulations were carried out without controller.

Note also that the linear system was simulated using `lsim`:

$$[YY\ TT\ \tilde{X}] = \texttt{lsim}(sys,\ \tilde{u},\ tsim,\ x0), \tag{4.4}$$

where $\tilde{u} = u - u_{stat}$ is the vector of input signals linearized around present operating point. After the simulation X_{stat} was added to $\tilde{X}$ and the intake manifold pressure ($p_{im} = X(7)$) was plotted. Hence, even if the results from the simulation of the linear model are corrected with the stationary conditions it still shows a difference compared to the nonlinear model.

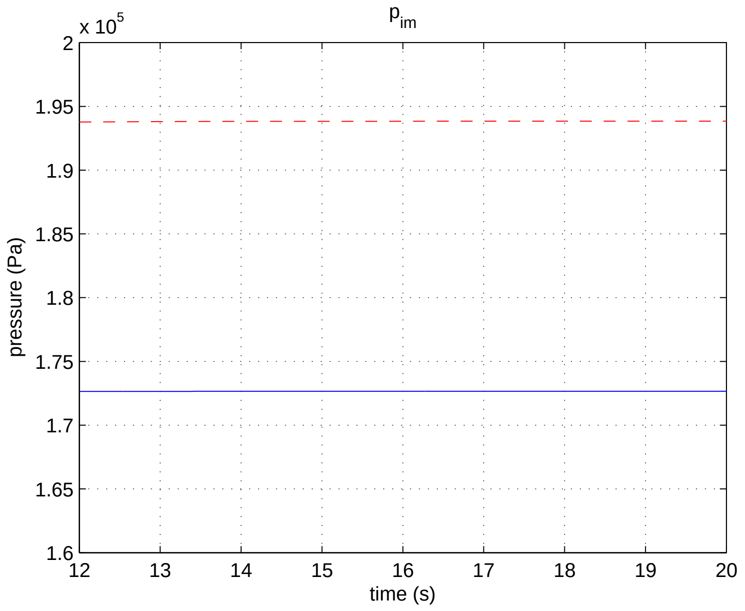

Figure 4.5: Difference in intake manifold pressure between the linear and nonlinear model

The intake manifold pressure differs some between the linear and nonlinear model. The MPC-controller in Simulink has the linear model as an internal model and gets its input signals from the nonlinear model. When the controller calculates future torque values from intake manifold pressure with the linear model, these values are not exactly the same as the torque values that the nonlinear model delivers. This probably works negative on controller performance.

4.5 Cost Function

The objective is to minimize the fuel consumption and the deviation from requested torque. A small deviation from requested torque gives a fast response time. In order to get a quadratic problem (Section 3.2) the expressions for fuel consumption and torque need to be convex functions, for example linear in the states. To introduce integral action, rapid changes in the control signal is also penalized. The cost function will have the following appearance:

$$J = \sum_{i=1}^{N_p} \left[\dot{m}_{fuel,i}^T Q_{fuel} \dot{m}_{fuel,i} + (M_{e,i} - M_{e,ref,i})^T Q_{torque} (M_{e,i} - M_{e,ref,i})\right] + \sum_{i=0}^{N_p-1} \Delta u_{wg,i}^T Q_{wg} \Delta u_{wg,i}, \tag{4.5}$$

where $\Delta u_{wg,i} = u_{wg,i} - u_{wg,i-1}$. In the following parts it will be shown how the expressions for fuel consumption $\dot{m}_{fuel}$ and engine torque M_e are derived.

Fuel Consumption

The fuel mass flow to the cylinder is proportional to the air mass flow (An05):

$$\dot{m}_{fuel} = \frac{\dot{m}_{air}}{\lambda(\frac{A}{F})_s}. \tag{4.6}$$

Here $(\frac{A}{F})_s$ is the stoichiometric air/fuel ratio and λ is the normalized air/fuel ratio. An expression for the air mass flow is:

$$\dot{m}_{air} = (a_0 p_{im} + a_1)\frac{V_d N_s}{RT_{im} n_r}, \tag{4.7}$$

where V_d is the displacement volume, N_s the engine speed in revolutions per second, R is the gas constant and n_r the number of engine revolutions per cycle. The constants a_0 and a_1 are identified in (An05) with the Matlab-function $lscurvefit$ which uses a nonlinear least-squares technique.

If the state T_{im} and the input signal N_s is assumed constant, $\dot{m}_{air}$ can be expressed as a linear function of the state p_{im}:

$$\begin{aligned} \dot{m}_{air} &= k_1 p_{im} + k_2, \\ k_1 &= a_0 \frac{V_d N_s}{RT_{im} n_r}, \\ k_2 &= a_1 \frac{V_d N_s}{RT_{im} n_r}. \end{aligned} \tag{4.8}$$

Engine Torque

How large the torque from the engine is, depends on the work produced and consumed in the engine (Er05):

$$M_e = \frac{W_{ig} - W_p - W_f}{n_r 2\pi}, \tag{4.9}$$

where W_{ig} is the indicated gross work produced, i.e. the mechanical work produced, W_p is the pumping work consumed and W_f is the friction work consumed. There exist different expressions for W_{ig}, W_p and W_f. Often they are modeled as follows:

$$\begin{aligned} W_{ig} &= V_d \cdot IMEP_g = V_d \cdot \dot{m}_{fuel} q_{HV} \frac{2}{Ns} \frac{min(\lambda, 1)}{V_d} \eta_e, \\ W_p &= V_d \cdot PMEP = V_d \cdot (p_{em} - p_{im}), \\ W_f &= V_d \cdot FMEP = V_d \cdot \left[0.97 + 0.15\left(\frac{N}{1000}\right) + 0.05\left(\frac{N}{1000}\right)^2\right]. \end{aligned} \tag{4.10}$$

Here $\dot{m}_{fuel}$ is the fuel mass flow to the cylinder, which is given above, η_e is the engine raw effiency, q_{HV} the heating value (the amount of energy that the combustion of one unit of fuel can release when the fuel is in gaseous phase) and N the engine speed in revolutions per minute.

The indicated gross work is coupled to the energy in the fuel and to operating conditions, which gives the model in the first equation above.

In the second equation the pumping work is modeled as proportional to the difference in exhaust manifold pressure and intake manifold pressure.

The friction model comes from (He88). It accounts for three different kinds of friction: boundary friction (independent of engine speed), hydrodynamic friction (proportional to speed) and turbulent dissipation (proportional to speed squared).

The engine torque can now be expressed as a linear function of the states p_{im} and p_{em}:

$$\begin{aligned} M_e &= k_3 p_{im} + k_4 p_{em} + k_5, \\ k_3 &= \frac{V_d}{n_r 2\pi}\left[k_1 \frac{2}{N_s}\frac{q_{HV}}{\lambda(\frac{A}{F})_s}\frac{min(\lambda,1)}{V_d}\eta_e + 1\right], \\ k_4 &= -\frac{V_d}{n_r 2\pi}, \\ k_5 &= \frac{V_d}{n_r 2\pi}\left[k_2 \frac{2}{N_s}\frac{q_{HV}}{\lambda(\frac{A}{F})_s}\frac{min(\lambda,1)}{V_d}\eta_e - \left(0.97 + 0.15\left(\frac{N}{1000}\right) + 0.05\left(\frac{N}{1000}\right)^2\right)\right]. \end{aligned} \tag{4.11}$$

Here the same assumptions as for the fuel consumption have been made (T_{im}, N_s and N are assumed constant), furthermore it is assumed that the input signal λ does not change. The assumptions are only made for a certain operating point, i.e. the values on T_{im}, N_s, N and λ can change between operating points.

4.6 Constraints

The wastegate position is normalized to be between 0 and 1, where 0 corresponds to totally closed and 1 corresponds to fully open:

$$u_{min} = 0 \leq u_{wg} \leq 1 = u_{max}. \tag{4.12}$$

The constraints on the control signal u_{wg} are hard, whereas the constraints on the states are soft, i.e. small violations can be accepted. Most important is the constraint on maximum turbocharger speed w_{tc}.

$$x_{min} = \begin{pmatrix} 90000 \\ 240 \\ 20000 \\ 273 \\ 20000 \\ 273 \\ 20000 \\ 20000 \\ 273 \\ 20000 \\ 273 \\ 1000 \end{pmatrix} \leq \begin{pmatrix} p_{af} \\ T_{af} \\ p_{comp} \\ T_{comp} \\ p_{ic} \\ T_{ic} \\ p_{im} \\ p_{em} \\ T_{em} \\ p_t \\ T_t \\ w_{tc} \end{pmatrix} \leq \begin{pmatrix} 110000 \\ 350 \\ 300000 \\ 1500 \\ 300000 \\ 1500 \\ 300000 \\ 300000 \\ 1500 \\ 300000 \\ 1500 \\ 18000 \end{pmatrix} = x_{max}$$

In the Simulink implementation, the constraints on u_{wg} and w_{tc} are the only ones implemented. This way the complexity of the problem is reduced. It is checked that the other signals stay at physically reasonable values.

4.7 Model Reduction

Since explicit MPC can not handle systems with too many states, measured disturbances and references, model reduction is necessary. The air filter pressure and temperature, x_1 and x_2, do not vary that much from the ambient pressure and temperature, u_5 and u_6. Therefore, if a formulation with fewer states is preferred, these states can be defined constant. Which other states that can be truncated is tested by looking at controller performance for models containing different many states. The smallest model still showing acceptable controller performance is a system with five states:

States x	
$x_3 = p_{comp}$	Compressor Pressure $[Pa]$
$x_5 = p_{ic}$	Intercooler Pressure $[Pa]$
$x_7 = p_{im}$	Intake Manifold Pressure $[Pa]$
$x_9 = p_{em}$	Exhaust Manifold Pressure $[Pa]$
$x_{13} = w_{tc}$	Turbocharger Speed $[rad/s]$

Table 4.4: The states in the smallest model possible

When simulating in Matlab and Simulink, there is no need to use the reduced model, since computational speed and memory is not an issue.

Chapter 5

Implementation in Matlab

First, the MPC controller is implemented in Matlab. This version uses only *the linear models* in the calculations. The objective is to get a hint about if it is beneficial to open the wastegate at a transient or not. All steps of the MPC-algorithm are in this implementation visible, whereas the matrix formulation is hidden in the MPC-block used in the next chapter. The visible implementation gives a better understanding of MPC, which also motivates the Matlab implementation.

In Matlab two implementations are made: a linear one and a time variant one. The linear controller uses only one operating point throughout the simulation. This is the case where the theory of linear MPC is applicable and stability can be guaranteed. In the time variant implementation the operatíng point is changed from sample to sample. Here one is in the field of nonlinear MPC and the theory is more complicated.

The sample time is choosen to $T = 0.0125s$. The prediction horizon is a tuning parameter that is varied between $N_p = 10 - 40$. This allows the controller to look $0.125 - 0.5s$ ahead. Other tuning parameters are the penalty on rapid changes on the manipulated variable (the wastegate position), Q_{wg}, and for the time variant implementation the amount of hysteresis $Hyst$.

5.1 Inputs not Calculated by the Controller

Only input signal number three, $u_3 = u_{wg}$, is calculated by the controller. Input signals u_2, u_4, u_5 and u_6 are constant during the simulation:

$$\begin{aligned} u_2 &= N = user\ specified, \\ u_4 &= \lambda = 1, \\ u_5 &= p_{amb} = 101300\,Pa, \\ u_6 &= T_{amb} = 298\,K. \end{aligned} \tag{5.1}$$

Input signal number one, $u_1 = \alpha$, vary from sample to sample and its value is in the *linear* case (only matrices corresponding to one operating point is used) taken from a vector representing a step in the throttle opening area. For the time variant implementation, a ramp is used instead to avoid too fast changes. The start and final value of the throttle position α is specified by the user.

Henceforth in this thesis the input signal $u_3 = u_{wg}$ will be refered to just as u, whereas the input signals u_1, u_2, u_4, u_5 and u_6, that are not calculated by the controller, will be refered to as u_{MD}, i.e. a vector of measured disturbances.

5.2 Finding the Operating Point

The linear implementation finds one operating point and then uses it throughout the simulation. The operating point is found by comparing the first α and N to the stationary α and N that the linear systems are linearized around:

$$\begin{aligned} diff &= (N - N_{statvec})^2 + (10 \cdot (\alpha - \alpha_{statvec}))^2, \\ [y\, num] &= \texttt{min}(diff). \end{aligned} \tag{5.2}$$

Here the square is taken elementwise. $Diff$ has 45 elements, one for each operating point, representing the size of the deviation. A weight of 10 is multiplied to α, to get N and α of about the same size. The linear A- and B-matrices with number num and correspondning stationary points x_{stat} and $u_{all,stat}$ are chosen. The vector $u_{all,stat}$ contains stationary conditions for all input signals, i.e. both u and u_{MD}.

Once the A- and B-matrices have been found, the system is simulated using the Matlab function $lsim$. From the simulation initial values on the states are taken when stationary conditions are reached. A mean value of the state T_{im} is also calculated from the simulation to get a reasonable constant value on T_{im}. As mentioned in sec. 4.5 the state T_{im} needs to be constant to get a linear cost function. Rows and columns corresponding to the state T_{im} are taken away in the A- and B-matrices, since this state is assumed constant. To reduce the model more, i.e. assume that more states are constant, just remove rows and columns in the same way. Fewer states of course also affect the length of the constraint vectors x_{min} and x_{max} and the appearance of the C-matrix.

All states, inputs and references are adjusted with the now known stationary points x_{stat} and u_{stat} in the operating point. For example:

$$\tilde{x} = x - x_{stat}. \tag{5.3}$$

In the *time variant* implementation, the first operating point is found and the system is simulated in the same way as in the linear case above. *In each sample* a new search is performed after the optimization to find the present operating point. During the predictions performed in each sample the same matrices are used.

The time variant implementation uses N and the state p_{im} to find the present operating point:

$$\begin{aligned} diff &= (50 \cdot (N - N_{statvec}))^2 + (p_{im} - p_{im,statvec})^2, \\ [y\, num] &= \texttt{min}(diff). \end{aligned} \tag{5.4}$$

The reason for using p_{im} is that it changes slower than α during a transient. This time a weight of 50 is needed to make the engine speed and the intake manifold pressure of about the same size.

When hysteresis is implemented, the search for present operating point is changed. The value on intake manifold pressure at which the operating point switches is here dependent on direction. When coming from a value on the pressure under the original switching point the switching occurs at $Hyst$ pascal more than the original switching point, whereas it occurs at $Hyst$ pascal less than the original switching point when coming from a pressure value above the original switching point, see Fig.5.1. The implementation is done so that the upper and lower $Hyst$ can differ, but this feature is not used here and therefore one collective name is enough.

5.3 Setting Up the Matrices

In this section it is shown how the matrices H and L and the vectors h and b in the optimization problem (Equation 3.5) are obtained. The proceedings in this section is an adjustment of the general instructions in (S3) to the problem of this thesis.

We begin with the C-matrix. This matrix is defined by the cost function, which contains some physical parameters. The values that have been used are:

$$\begin{aligned}\left(\frac{A}{F}\right)_s &= 15.1, \\ q_{HV} &= 44 \cdot 10^6 \; J/kg, \\ \eta_e &= 0.3469, \\ V_d &= 2 \cdot 10^{-3} \; m^3, \\ R &= 287.2 \; J/(kg \cdot K), \\ n_r &= 2.\end{aligned} \tag{5.5}$$

With these parameters, the input signals N and λ and the mean value of the state T_{im} the constants k_1-k_5 are calculated according to sec. 4.5.

$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & k1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k3 & k4 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The two rows of C corresponds to the to output signals y_{fuel} and y_{torque} and the columns of C correspond to the states. The matrix D is a zero matrix of size (2×6).

The contineous state space system is transformed to a discrete state space system with the Matlab functin $c2d$:

$$\begin{aligned} G &= \texttt{ss}(A, B, C, D), \\ Gd &= \texttt{c2d}(G, T), \\ [A, B, C, D] &= \texttt{ssdata}(Gd), \end{aligned} \tag{5.6}$$

where T is the sample time ($T = 0.0125s$).

Now the B-matrix and u_{stat} are divided into two parts - one for the control signal and one for the other input signals:

$$\begin{aligned} B &= B(:, 3), \\ B_{MD} &= [B(:, 1:2) \; B(:, 4:6)], \\ u_{stat} &= u_{all,stat}(:, 3), \\ u_{MD,stat} &= [u_{all,stat}(:, 1:2); u_{all,stat}(:, 4:6)]. \end{aligned} \tag{5.7}$$

References for the two output signals are given for N_p future samples in y_{ref} with size $(2 \times N_p)$. The reference for y_{fuel} (row one in y_{ref}) is always zero and the reference for y_{torque} (row two in y_{ref}) is a step from a lower torque to a higher one. The output reference is converted to a state reference $\tilde{x}_{ref}$ through the pseudoinverse command $pinv$ in Matlab:

$$\tilde{x}_{ref,k} = \texttt{pinv}(C) y_{ref}(:, k) - x_{stat,k}, \; k = 1, ..., N_p. \tag{5.8}$$

Introducing $\tilde{X}_k = \tilde{x}_k - \tilde{x}_{ref,k}$ and using equation 3.4 we get:

$$\tilde{X}_k = A^k \tilde{x}_0 + \sum_{j=0}^{k-1} A^{k-1-j}(B\tilde{u}_j + B_{MD}\tilde{u}_{MD,j}) - \tilde{x}_{ref,k}. \tag{5.9}$$

For $k = 1, ..., N_p$ the matrix form becomes:

$$\underbrace{\begin{pmatrix} \tilde{X}_1 \\ \tilde{X}_2 \\ \vdots \\ \tilde{X}_{N_p} \end{pmatrix}}_{\hat{X}} = \underbrace{\begin{pmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{pmatrix}}_{\hat{A}} \tilde{x}_0 + \underbrace{\begin{pmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{N_p-1}B & A^{N_p-2}B & \cdots & B \end{pmatrix}}_{\hat{B}} \underbrace{\begin{pmatrix} \tilde{u}_0 \\ \tilde{u}_1 \\ \vdots \\ \tilde{u}_{N_p-1} \end{pmatrix}}_{\tilde{U}} +$$

$$\underbrace{\begin{pmatrix} B_{MD} & 0 & \cdots & 0 \\ AB_{MD} & B_{MD} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{N_p-1}B_{MD} & A^{N_p-2}B_{MD} & \cdots & B_{MD} \end{pmatrix}}_{\hat{B}_{MD}} \underbrace{\begin{pmatrix} \tilde{u}_{MD,0} \\ \tilde{u}_{MD,1} \\ \vdots \\ \tilde{u}_{MD,N_p-1} \end{pmatrix}}_{\tilde{U}_{MD}} - \underbrace{\begin{pmatrix} \tilde{x}_{ref,1} \\ \tilde{x}_{ref,2} \\ \vdots \\ \tilde{x}_{ref,N_p} \end{pmatrix}}_{\tilde{X}_{ref}}$$

Objective Function Matrix H and Vector h

The weight matrix for the outputs Q_y is constructed as follows:

$$Q_y = \begin{pmatrix} Q_{fuel} & 0 \\ 0 & Q_{torque} \end{pmatrix}$$

where the relative difference in Q_{fuel} and Q_{torque} decides what the controller should put most effort in minimizing - fuel consumption or deviation from torque.

To get a state weight the following matrix transformation is performed:

$$Q_x = C^T \cdot Q_y \cdot C. \tag{5.10}$$

The terminal weight S_x is constructed in the same way as Q_x:

$$S_x = C^T \cdot S_y \cdot C. \tag{5.11}$$

For N_p time steps the weighting matrices are stacked in one big matrix:

$$\hat{Q}_x = \begin{pmatrix} Q_x & 0 & \cdots & 0 \\ 0 & Q_x & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & S \end{pmatrix}$$

Now H and h^T can be formulated as:

$$\begin{aligned} H &= \hat{B}^T \hat{Q}_x \hat{B} \\ h^T &= (\hat{A}\tilde{x}_0 + \hat{B}_{MD}\tilde{U}_{MD} - \tilde{X}_{ref})\hat{Q}_x \hat{B} \end{aligned} \tag{5.12}$$

Constraint Matrix L and Vector b

To get L and b the constraint matrices also need to be stacked into big matrices. As can be seen in sec.4.6 all states are constrained. The state constraint matrix is therefore set up as the identity matrix by the command $E = eye(n_A)$, where n_A is the number of columns in A. These identity matrices are then stacked into one big matrix:

$$\hat{E} = \begin{pmatrix} E & 0 & \cdots & 0 \\ 0 & E & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & E \end{pmatrix}$$

The constraints on $\tilde{x}$ have to be translated to constraints on $\tilde{U}$. Using $\tilde{x} = \hat{A}\tilde{x}_0 + \hat{B}\tilde{U} + \hat{B}_{MD}\tilde{U}_{MD}$ and x_{min} and x_{max} from sec. 4.6 we get:

$$\underbrace{\hat{E}\hat{B}}_{Lx_{max}}\tilde{U} \leq \underbrace{\begin{pmatrix} \tilde{x}_{max,1} \\ \tilde{x}_{max,2} \\ \vdots \\ \tilde{x}_{max,N_p} \end{pmatrix} - \hat{E}(\hat{A}\tilde{x}_0 + \hat{B}_{MD}\tilde{U}_{MD})}_{bx_{max}}$$

and

$$\underbrace{-\hat{E}\hat{B}}_{Lx_{min}}\tilde{U} \leq \underbrace{-\begin{pmatrix} \tilde{x}_{min,1} \\ \tilde{x}_{min,2} \\ \vdots \\ \tilde{x}_{min,N_p} \end{pmatrix} + \hat{E}(\hat{A}\tilde{x}_0 + \hat{B}_{MD}\tilde{U}_{MD})}_{bx_{min}}$$

where $\tilde{x}_{max,k} = x_{max} - x_{stat,k}$ and $\tilde{x}_{min,k} = x_{min} - x_{stat,k}$.

From sec. 4.6 one can also derive the constraint matrices for the vector of input signals at times $k = 0, \cdots, N_p - 1$:

$$\underbrace{\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{pmatrix}}_{Lu_{max}}\tilde{U} \leq \underbrace{\begin{pmatrix} \tilde{u}_{max,1} \\ \tilde{u}_{max,2} \\ \vdots \\ \tilde{u}_{max,N_p} \end{pmatrix}}_{bu_{max}} \quad and \quad \underbrace{-\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{pmatrix}}_{Lu_{min}}\tilde{U} \leq \underbrace{-\begin{pmatrix} \tilde{u}_{min,1} \\ \tilde{u}_{min,2} \\ \vdots \\ \tilde{u}_{min,N_p} \end{pmatrix}}_{bu_{min}}$$

Now L and b can be build as:

$$L = \begin{pmatrix} Lu_{min} \\ Lu_{max} \\ Lx_{min} \\ Lx_{max} \end{pmatrix} \quad and \quad b = \begin{pmatrix} bu_{min} \\ bu_{max} \\ bx_{min} \\ bx_{max} \end{pmatrix}$$

5.4 Penalty on Rapid Input Changes

A penalty on rapid changes in the input signal is implemented to reduce oscillations. This is especially useful in the time variant case, where changing operating conditions lead to oscillations. According to (Wa03) this penalty also introduces integral action.

In the cost function an extra term is added:

$$\sum_{i=0}^{N_p-1} (\tilde{u}(k+i) - \tilde{u}(k+i-1))^T Q_{wg} (\tilde{u}(k+i) - \tilde{u}(k+i-1)) \tag{5.13}$$

The input signal implemented in last sample $\tilde{u}(k-1)$ is needed for $i = 0$. It is therefore saved from last sample.

The matrix form becomes:

$$\begin{pmatrix} \tilde{u}(k) - \tilde{u}(k-1) \\ \tilde{u}(k+1) - \tilde{u}(k) \\ \vdots \\ \vdots \\ \tilde{u}(k+N_p-1) - \tilde{u}(k+N_p-2) \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}}_{\Phi} \tilde{U} - \underbrace{\begin{pmatrix} \tilde{u}(k-1) \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix}}_{\varphi},$$

where $\tilde{U} = [\tilde{u}(k) \cdots \tilde{u}(k+N_p-1)]^T$ is the optimization vector.

In the cost function this leads to that H and h are extended as follows:

$$\begin{aligned} H &= \hat{B}^T \hat{Q}_x \hat{B} + \Phi^T \hat{Q}_{wg} \Phi, \\ h^T &= (\hat{A}\tilde{x}_0 + \hat{B}_{MD}\tilde{U}_{MD} - \tilde{X}_{ref})\hat{Q}_x \hat{B} - \varphi^T \hat{Q}_{wg} \Phi. \end{aligned} \tag{5.14}$$

The constraint matrices do not have to be extended.

5.5 Softening the Constraints

The state constraints can be soften to ensure stability. This is done by extending the optimization vector with a slack variable ϵ:

$$\tilde{U} = \left[\tilde{U};\ \epsilon\right]. \tag{5.15}$$

The slack variable relaxes the state constraints according to:

$$x \leq x_{max} + \epsilon, \tag{5.16}$$

where x is a vector of states and x_{max} a vector of constraints. The same type of relaxation is made to the lower bound on x. The slack variable has a value other than zero only if it is necessary, i.e. no solution can be found otherwise. In the cost function ϵ is therefore punished with a large weight ρ.

The input constraints has to be adjusted to include constraints on ϵ. This is done by extending the dimensions of the identity matrices Lu_{min} and Lu_{max} by one and by letting:

$$\begin{aligned} bu_{min} &= [bu_{min};\ 0], \\ bu_{max} &= [bu_{max};\ inf]. \end{aligned} \tag{5.17}$$

The slack variabel ϵ is added to all state constraints to make it possible to exceed the constraints if no feasible solution exists:

$$\begin{aligned} Lx_{min} &= [Lx_{min} - ones(mLx, 1)], \\ Lx_{max} &= [Lx_{max} - ones(mLx, 1)]. \end{aligned} \tag{5.18}$$

Here mLx is the number of rows in Lx_{min}. In the objective function H and h must be extended:

$$\begin{aligned} H &= [H \;\; zeros(Np,1); \;\; zeros(1,Np+1)]\,, \\ h &= [h; \;\; \rho]\,. \end{aligned} \tag{5.19}$$

Here ρ punishes the value of ϵ in the cost function so that constraints will not be violated unnecessarily.

5.6 Hysteresis

Hysteresis is only an issue in the time variant case and is therefore only implemented there. It is an alternative to punishing rapid changes in the manipulated variable, when one wants to reduce oscillations due to switching between linear controllers. In Figure 5.1 the idea is illustrated for a transition between two consecutive operating points.

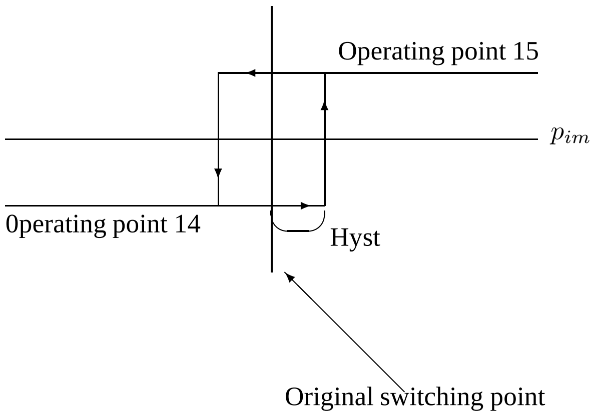

Figure 5.1: The idea of hysteresis

When using hysteresis the switching point depends on the direction. For example, when coming from a lower intake manifold pressure there is no change in operating point until the pressure has past $Hyst$ pascal after the original switching point. This way rapid changes back and forth between opearting points are avoided.

It is assumed that the operating points corresponding to present engine speed are selected. Then two functions are implemented in Matlab. The first finds the new upper and lower limits. These limits depend on the size of $Hyst$. The second function calculates present operating point from last operating point and current pressure, see appendix A.

5.7 Finding the Optimal Input and Applying It

The otimization problem (Equation 3.5) is solved with the Matlab function `quadprog`:

$$\left[\tilde{U},\ J,\ iflag\right] = \texttt{quadprog}(H, h^T, L, b). \tag{5.20}$$

Then the first element of $\tilde{U}$ is applied, the stationary conditions are added to the state, input and output vectors and they are saved for plotting. Finally the initial state is updated:

$$\begin{aligned}
u_{MD} &= [\alpha\ \ N\ \lambda\ \ p_{amb}\ \ T_{amb}]^T, \qquad (5.21)\\
\tilde{u}_{MD} &= u_{MD} - u_{MD,stat},\\
\tilde{u} &= \tilde{U}(1),\\
\tilde{x} &= A\tilde{x}_0 + B\tilde{u} + B_{MD}\tilde{u}_{MD},\\
x &= \tilde{x} + x_{stat},\\
u &= \tilde{u} + u_{stat},\\
UU(:,kk) &= u,\\
XX(:,kk) &= x,\\
YY(:,kk) &= Cx + [k2;\ k5],\\
x_0 &= x,
\end{aligned}$$

where kk is the sample number. The procedure is then repeted for next sample. In the linear implementation the stacked matrices are only set up once, whereas they need to be constructed in each sample in the time variant implementation.

Chapter 6

Implementation in Simulink using MPC-toolbox

In this chapter the MPC-controller is implemented in Simulink. Here the internal model in the MPC-controller is a linear model, whereas the controller gets its inputs from a nonlinear model. Implementation in Simulink is a preperation for C-code generation and real time implementation.

Just as in last chapter, two implementations are done here. The first uses only one MPC-block with an internal model in one operating point. The second one switches between several operating points with one MPC-block for each point.

6.1 The MPC-Block

The MPC-toolbox library contains a block representing the controller, see Figure 6.1. Input signals to the controller are measured outputs (MO), reference signals and measured disturbances (MD). Measured disturbances are independent inputs whose values are measured and used for feedforward compensation. Output signals from the controller are the optimal manipulated variables (MV).

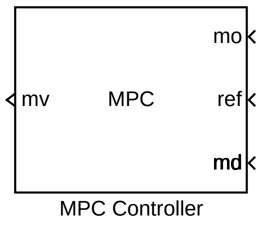

Figure 6.1: The MPC-block used in Simulink

6.2 Creating the Controller

When you double click on the MPC-block a mask appears that lets you define which MPC controller to use. An MPC controller is created with the commands:

$$\begin{gathered}\mathtt{model} = \mathtt{ss}(\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}),\\ \mathtt{model} = \mathtt{setmpcsignals}(\mathtt{model},'\mathtt{MD}',[1\ 2\ 4\ 5\ 6]),\\ \mathtt{controller} = \mathtt{mpc}(\mathtt{model},\mathtt{T_s},\mathtt{N_p},\mathtt{N_u}),\end{gathered}$$

where A,B,C and D-matrices are the same as in Chapter 5, T_s is the sampling time, N_p the prediction horizon and N_u the control horizon. The sampling time is also the same as in last chapter ($T_s = 0.0125$). The prediction horizon and the control horizon are tuned to get best possible performance.

Defualt is that the input signals are manipulated variables, i.e. calculated by the controller. With $setmpcsignals$ the signal type can be changed. Here only input signal number three is a manipulated variable. Therefore the other input signals are changed to measured disturbances. Simulation involving the MPC-block is possible after $controller$ has been exported to Matlabs workspace.

Examples of other changes that can be made to $controller$ are:

$$\begin{gathered}\mathtt{controller.Model.Nominal.U} = \mathtt{u_{stat}}\\ \mathtt{setname}(\mathtt{controller},'\mathtt{input}',1,'\mathtt{alfa}')\\ \mathtt{MV}(1) = \mathtt{struct}('\mathtt{Min}',0,'\mathtt{Max}',1,'\mathtt{RateMin}',-1,'\mathtt{RateMax}',1)\\ \mathtt{OV}(3) = \mathtt{struct}('\mathtt{Min}',0,'\mathtt{Max}',18000)\\ \mathtt{Weights} = \mathtt{struct}('\mathtt{Input}',0,'\mathtt{InputRate}',\mathtt{inputrate},'\mathtt{Output}',[0\ 1\ 0])\\ \mathtt{set}(\mathtt{controller},'\mathtt{Weights}',\mathtt{Weights},'\mathtt{ManipulatedVariables}',\mathtt{MV},'\mathtt{OutputVariables}',\mathtt{OV})\end{gathered}$$

The first line declares the nominal condition. Since the linear model is linearized around u_{stat} this is the nominal condition for the input signals. Nominal conditions for the states X and the output variables Y are defined in the same way. Names of variables can be defined as in line two. Line three sets the constraints on the manipulated variable and line four defines the constraints on the turbine speed. The weights are chosen in the structure *Weights* as can be seen on line five above. The weight on the input signal is zero wheras *inputrate* is a tunable scalar to introduce integral action. In the case above weight is only put on minimizing the deviation from requested torque, which can be seen in the vector for output weight where only the second element differs from zero.

The changes to the controller has above been made using functions. This is practical since the changes are saved in an m-file. This file is then run before the simulation to place the controller in workspace. The m-file is placed under *initFcn* in *Callbacks* under *Model Properties* on the *file menu* in simulink so that it is run automatically when pushing the simulation button.

An alternative to working with functions is to use the graphical user interface available. Here changes in horizons, weights etc. can be made.

6.3 Connecting the Controller to the Model

An important fact is that the controller and the nonlinear model runs on different sampling times. The model is continuous, whereas the MPC-controller is discrete with sample time 0.0125s. The controller is discrete since it looks a certain samples ahead. To be able to run closed loop simulations with a fixed point solver, which is necessary for code generation, rate transition blocks need to be inserted at all places where signals change sampling time. It is very helpful to use the feature *Sample Time Colors*, which can be found on the *Format menu* under *Port/Signal Displays*.

In Fig. 6.2 the closed loop is shown. Continuous signals are black, fastest discrete signals are red and second fastest discrete signals green. This is of course not possible to see in the black and white picture here, but with different colors for different sampling times on the computer screen it is easy to know where rate transition blocks need to be inserted.

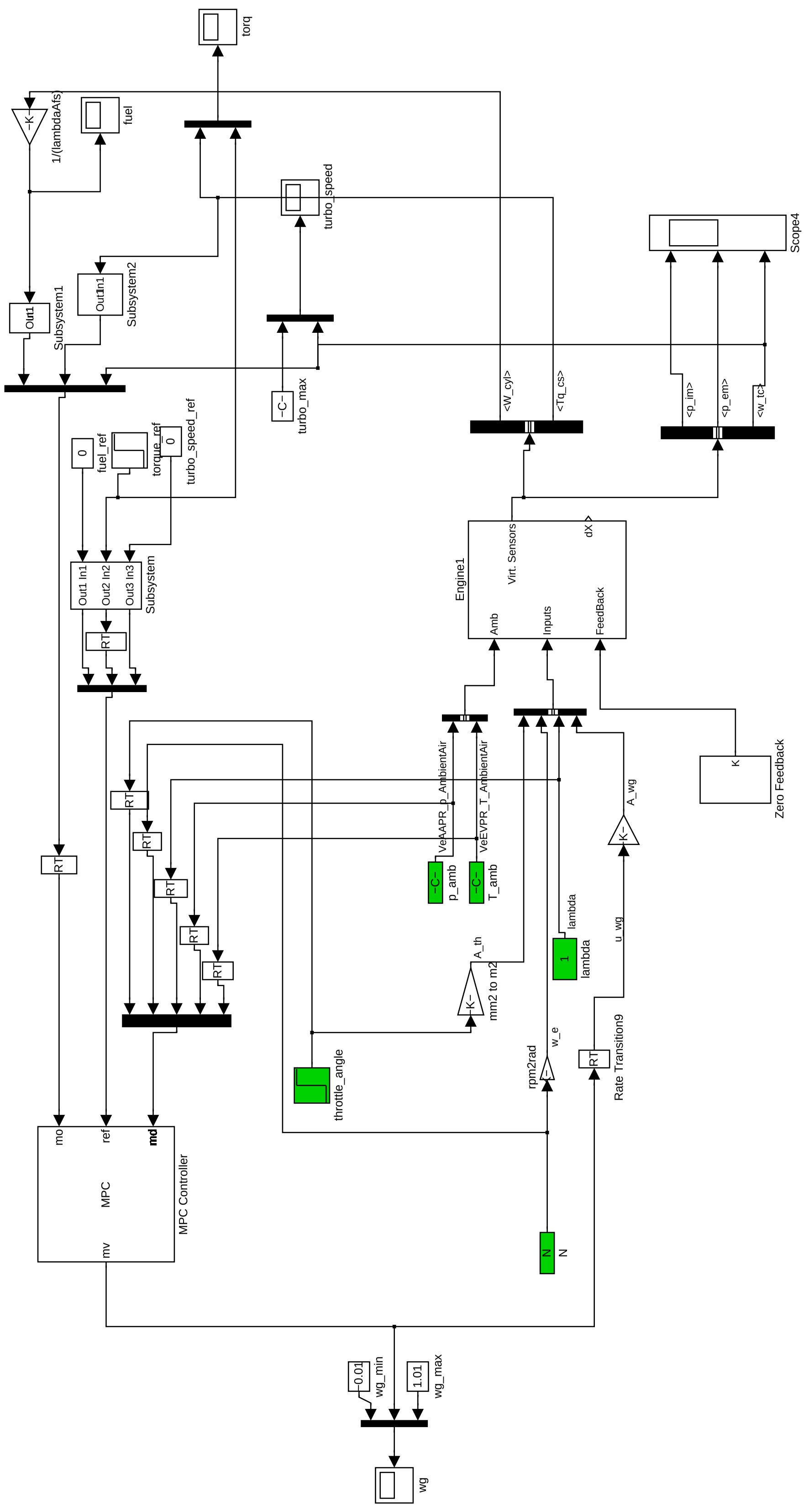

Figure 6.2: MPC-controller connected to the nonlinear model

6.4 Changing Operating Conditions

An MPC-block can only be based on one linear model. To be able to adjust for changing operating conditions it is necessary to switch between several MPC-blocks based on different linear models.

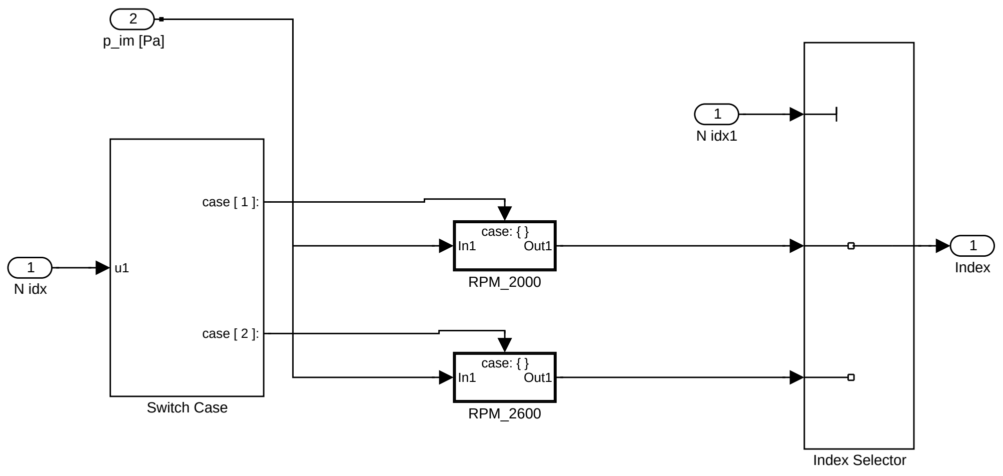

Figure 6.3: How the operating point index is found

First a search is performed to find the present operating point. The search is based on the engine speed N and the intake manifold pressure p_{im} just as in the time variant case in Chapter 5. The search is made using *switch case action subsystems* from the simulink library, see Figure 6.3. In order not to get an enormous model, only four operating points are used - operating point number 14 and 18 for engine speed 2000 rpm and number 21 and 24 for 2600 rpm. The choice of operating points are based on the singular value diagrams, see Figures 4.3 and 4.4.

The resulting signal from the search block is a scalar between 1 and 4 representing the operating points according to [1,2,3,4]=[14,18,21,24]. The index between 1 and 4 is then input signal to another group of switch case action subsystems that contains one MPC-controller each. The switch case action subsystem block with the MPC-controller corresponding to the present index is chosen.

6.5 Applying Explicit MPC

The functions in this section requires the *Hybrid toolbox*. You will also need a C-compiler compatible with Matlab. In the m-file where the MPC-controller was created, see Section 6.2, an extra command is added to convert the controller to explicit form:

$$explicit.controller = \texttt{expcon}(controller, range, options) \tag{6.1}$$

The argument $range$ is optional but useful. It is a structure where one can decide the size within which the partition is going to take place. In $range$ minimum and maximum values for the states, inputs, measured disturbances and references can be set. In $options$ several properties can be altered, for example one can choose which solver to use. If the model contains too many states, measured disturbances and references $expcon$ breaks down. When the controller is created with the mpc-function a disturbance model is automatically created, which adds states to the controller.

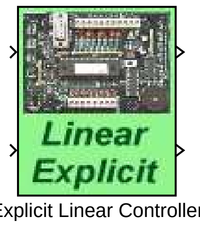

Figure 6.4: The linear explicit controller block from the Simulink library

Just as the MPC-block the Linear Hybrid Controller block, see Figure 6.4, lets you define a controller when you double click on it. The explicit controller created above is exported to workspace and defined in the block mask.

Unfortunately the explicit controller block in Simulink can not handle measured disturbances yet. This feature is to be developed by the creators of Hybrid toolbox. Today measured disturbances can be implemented as constant states:

$$\begin{aligned}\dot{x} &= Ax + Bu + B_{MD}u_{MD} \\ \begin{pmatrix} \dot{x} \\ \dot{u}_{MD} \end{pmatrix} &= \begin{pmatrix} A & B_{MD} \\ 0 & I \end{pmatrix}\begin{pmatrix} x \\ u_{MD} \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u\end{aligned} \tag{6.2}$$

and pass the vector $[x;\ u_{MD}]$ to the simulink block while simulating the controller.

6.6 Real Time Implementation

GM organizes all modeling and control algorithms by placing them in trigged subsystems that are called at different sample time intervals. The reason is, that it is important to control the executing times. The blocks in a triggered subsystem must have an inherited sample time, since it is decided from outside when they are going to be executed. Unfortunately, an MPC controller needs an explicit declared sample time to convert the continous model to discrete time and can therefore not be placed inside a triggered subsystem.

Chapter 7

Results

7.1 Preliminary Work

The controller is supposed to make sure that the engine torque follows a reference curve. To determine if the controller does its job a requirement is defined:

Definition 7.1. *An acceptable stationary error is less than 1% of the reference value.*

Initial Test

First a rough test is made to get a feeling for what happens when the wastegate is opened at the beginning of a transient. A step in the throttle position is simulated without controller. In Figure 7.1 the result of a simulation of the nonlinear model can be seen. The engine speed is constant $2000\ rpm$ and the throttle area is a step from 80 to $1500\ mm^2$.

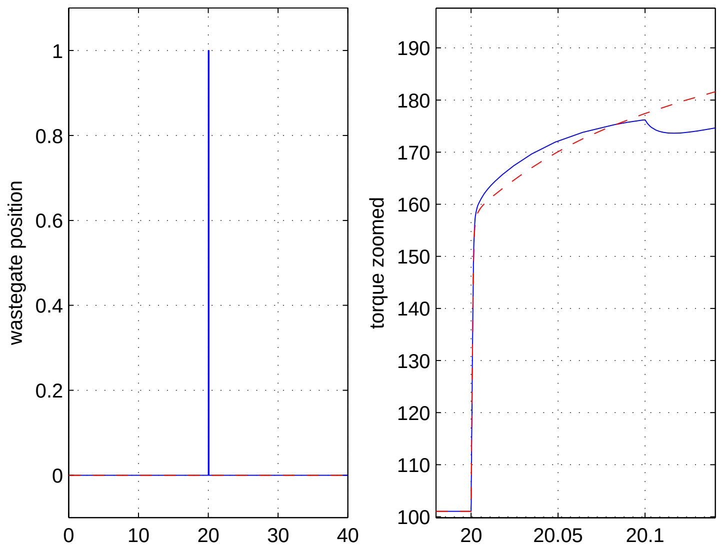

Figure 7.1: Simulation of model without controller

The solid curve shows the result when the wastegate is opened for 0.1s at the beginning of the step, whereas the dashed line is the result of a closed wastegate. There is a small peak in the engine torque when the wastegate is opened at the beginning of a transient. However, after the peak the torque tips down under the curve for the closed wastegate. From this picture it is hard to tell if opening the wastegate is beneficial.

Scenario

The scenario simulated in Figure 7.1 is the following: the car has constant speed $100\ km/h$, which corresponds to an operating condition with engine speed approximately $2000\ rpm$ and torque from crankshaft of about $100\ Nm$. This is the condition of operating point number 14. The driver then gives full gas and the resulting pressure curve goes through the operating points 14 to 18. This scenario will be used more further on.

Restrictions

The implementations have been made very general, except for one case - the implementation of several operating points in Simulink. In the other cases an arbitrary engine speed and positive step/ramp in the throttle position can be chosen. The results in this chapter are however from one engine speed only, $2000\ rpm$. This restriction is made to be able to make a more thorough analysis. The linear models corresponding to engine speed $2000\ rpm$ were already investigated in Section 4.3 and comparisons between the different MPC controller implementations are carried out. For the time variant Simulink case each MPC block needed to have a specified linear model. Therefore, which operating points to use was decided in advance.

7.2 Implementation in Matlab

One Operating Point in Matlab

In this section the linear controller implemented in Matlab is used. This controller uses only one operating point. A transient is simulated for different throttle opening areas. The engine speed is constant $2000\ rpm$. In Figure 7.2 operating point number 12 is used and in Figure 7.3 number 15. In appendix B figures can be found from operating points 13 and 14. Since only one operating point is used in each simulation, too big steps in the throttle area can not be made. The step has to stay in the field where the linear model gives reasonable results. Otherwise the states will take on unreasonable values and the constraints on the states will be violated.

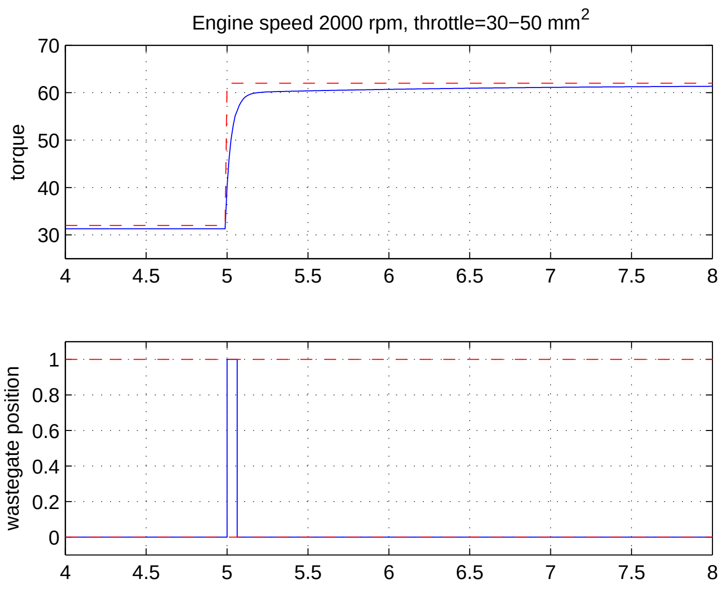

Figure 7.2: Operating point 12: Opening of the wastegate seems to be beneficial

The torque reference is in Figure 7.2 a step between 32 Nm and 62 Nm. These values come from a simulation of the linear model in operating point 12 without controller. The inputs are the same in the simulation with and without controller ($\alpha = 30 - 50\ mm^2$, $N = 2000\ rpm$, $\lambda = 1$, $p_{amb} = 101300\ Pa$ and $T_{amb} = 298\ K$), except for the wastegate which is closed during the simulation without controller. This way comparisons can be made to get a hint about if it is beneficial to open the wastegate during a transient.

Figure 7.2 indicates that it is beneficial to open the wastegate at the beginning of a transient for low torque values. This can be seen from the peak in wastegate position. Right at the beginning of the step the MPC controller calculations result in that it is optimal to fully open the wastegate (wg=1) in order to minimize the deviation from requested torque (fuel consumption is not punished here). The results therefore indicate that a short opening of the wastegate gives a smaller deviation from requested torque comared to keeping the wastegate closed during the step.

Figure 7.3 and the figures in appendix indicate that it is not beneficial to open the wastegate during a transient for higher torque values. In these figures the MPC controller calculations result in a closed wastegate for all times.

That it seems to be beneficial to open the wastegate for low torque values is unfortunate, since the wastegate only can be opened when a certain difference between compressor pressure and ambient pressure exist (see Sec. 2.3 for more information). For torque values under about 130 Nm the difference in pressure is not large enough to open the wastegate. However, if in the future the conventional mechanical design is replaced by a valve controlled by an ECU (electronic control unit), then the wastegate can be opened also for low torque values.

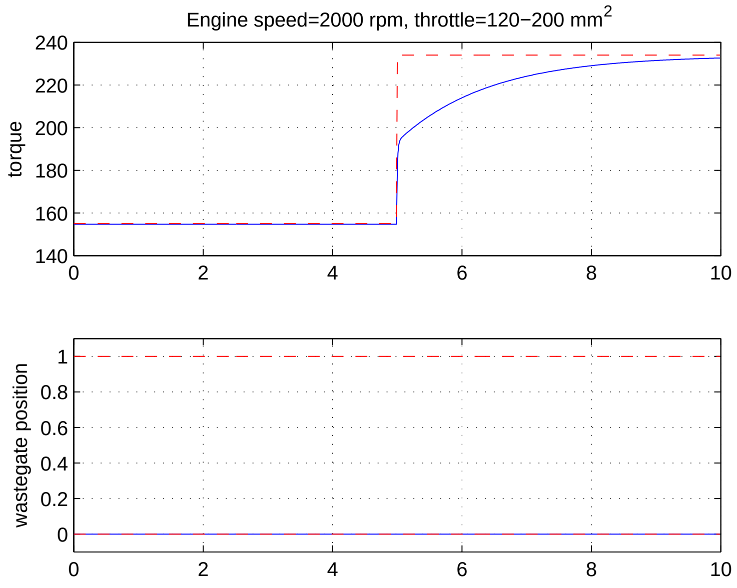

Figure 7.3: Operating point 15: Opening of the wastegate seems not to be beneficial

The torque reference is in Figure 7.3 a step between 155 Nm and 234 Nm. These values come from a simulation of the linear model in operating point 15 without controller. The inputs are the same in the simulation with and without controller ($\alpha = 120 - 200\, mm^2$, $N = 2000\, rpm$, $\lambda = 1$, $p_{amb} = 101300\, Pa$ and $T_{amb} = 298\, K$), except for the wastegate which is closed during the simulation without controller.

In both Figure 7.2 and 7.3 the dashed lines are the reference value for the torque and the constraints on the wastegate position. The solid lines are the resulting torque curve and wastegate position from closed loop simulation in Matlab with the MPC controller that only uses one operating point. Note that the wastegate position stays inside the limits set by the constraints.

The controller performance is tested in Figure 7.4 where the references is lowered to 200 Nm to see how the controller reacts to the change in reference. The controller reacts well to changes in reference and there is no stationary error.

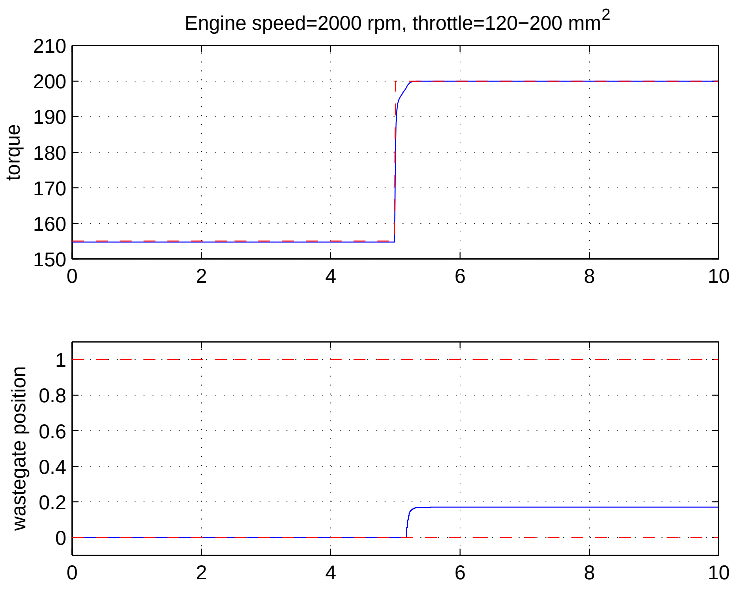

Figure 7.4: Test of the linear Matlab MPC controller

Weights and other tuning parameters are summarized in Table 7.1:

Parameter	Value
T	0.0125
N_p	40
Q_{torque}	1
Q_{fuel}	0
Q_{wg}	0
$Hyst$	0

Table 7.1: Tuning parameters used in simulations shown in Figure 7.2, 7.3, 7.4, B.1 and B.2

The weight on fuel is zero in all figures above, i.e. no effort is put on minimizing the fuel consumption. The reason for this is that the purpose of the simulation was to see what the optimal wastegate position is when the only requirement is to track the torque reference curve as good as possible. If a penalty is put on fuel consumption then the wastegate opens to lower the fuel consumption. The idea here was to see if it is beneficial to open the wastegate only from a torque tracking point of view.

If a weight is put on minimizing the fuel consumption, then a stationary error appears, see Figure 7.5 and the final value on the requested torque is not reached. This is a limitation on the MPC controller. It would be better if it managed to reach the final value but in a longer time. It is difficult to get the time aspect into the implementation of the MPC algoritm. For each sample the MPC controller calculates optimal wastegate position. If there is a larger weight on minimizing fuel consumption then a stationary error in the torque curve will be accepted by the controller.

The weights are in this case $Q_{torque} = 1$ and $Q_{fuel} = 10^6$. The weight on the fuel consumption migth seem unreasonable large, but there is a difference in size between normal torque values and fuel values of about 10^5.

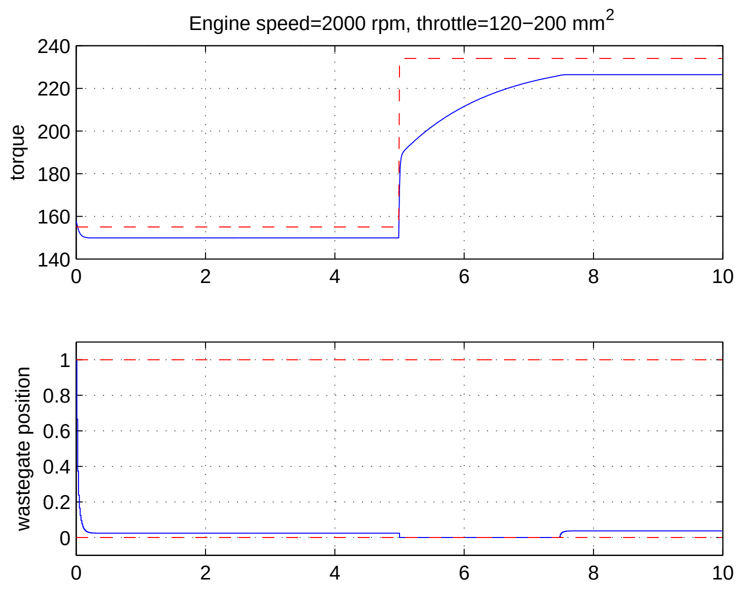

Figure 7.5: Penalizing the fuel consumption gives a stationary error in the torque curve

In Figure 7.4 and Figure 7.5 the dashed lines are still the torque reference and the wastegate constraints and the solid lines the resulting torque and wastegate curves calculated by the MPC controller as the solution to the optimization problem.

Several Operating Points in Matlab

When the controller calculates present operating point in every sample, a ramp in the throttle opening area with much larger distance between initial and final value can be simulated. Since the same operating point is used for predictions in each sample, it is not sure that it is good to use a long prediction horizon. A long prediction can on the other hand lead in the wrong direction.

Now, when the time variant implementation makes it possible to simulate a bigger step in throttle opening area, we go back to the scenario described in Sec. 7.1: the car has a constant speed of $100\ km/h$ and the driver then gives full gas. The corresponding change in throttle position is a ramp from about $80 - 1500\ mm^2$ and a step in requested torque from about $100\ Nm$ to $250\ Nm$. During the transient, the intake manifold pressure curve passes through several operating points: from 14 to 18.

Without any compensation the controller starts jumping between operation points as in Figure 7.6.

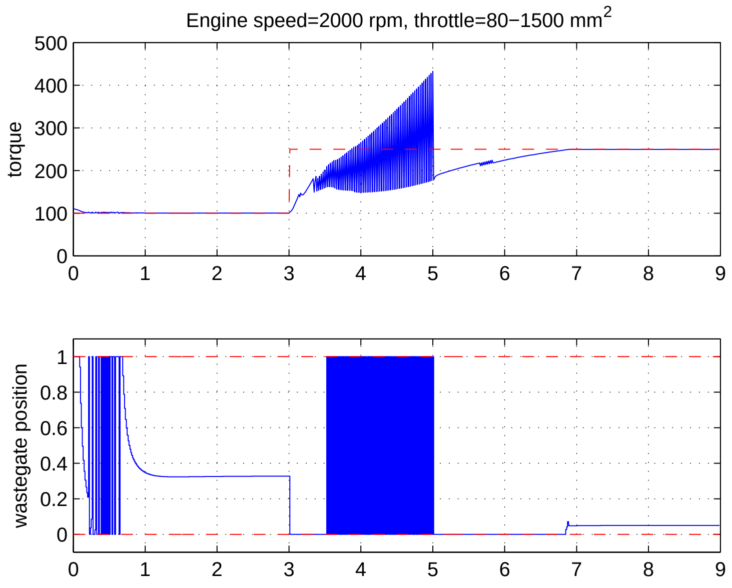

Figure 7.6: Oscillations in torque and wastegate due to changing operating conditions

The oscillations can be made smaller by penalizing rapid changes in the wastegate position, see Section 5.4. This has been done in Figure 7.7, where the weight $Q_{wg} = 10^6$ instead of zero. The oscillations are reduced but not totally surpressed. There is a small stationary error at the lower torque level even though a penalty on rapid input changes should introduce integral action. The reason for this could be that the prediction horizon is not long enough (10 samples = 0.125s). The prediction horizon need to be short since only one operating point is used for predictions in each sample. It is not good to have a large Q_{wg} if one wants to investigate whether it is beneficial to open the wastegate for a short time during the transient. Such a short but large opening will be suppressed by the penalty on rapid changes in the control signal.

From the reasoning above one can conclude that a penalty on rapid changes is in this case not the best solution. It is instead better to solve the cause of the oscillations, which is that the controller jumps between operating points. This is done in by introducing hysteresis, see Sec. 5.6.

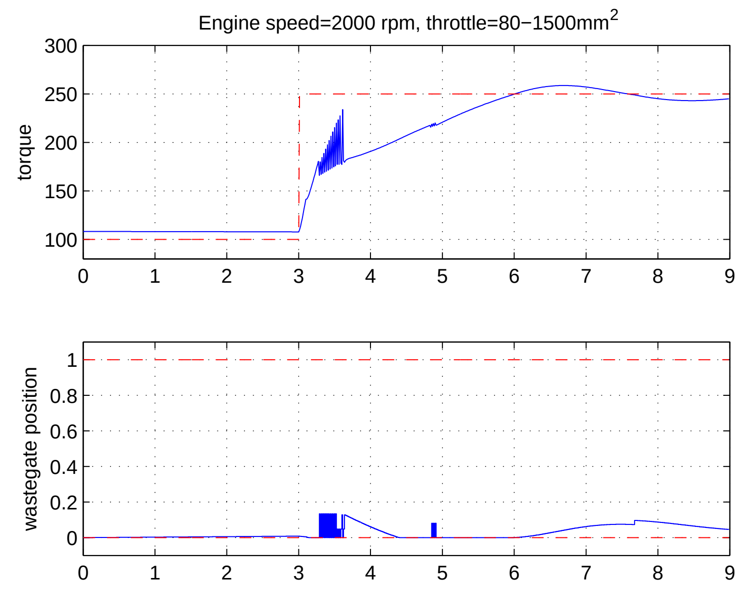

Figure 7.7: Oscillations in torque are reduced by punishing rapid changes in the wastegate

Hysteresis supresses the oscillations and does not lead to any stationary errors, see Figure 7.8. The controller now goes through the operating points 14 to 18 without jumping back and forth. However, to get really good controller performance, a large value on the parameter $hyst$ is needed. The value should be much smaller than the gap between operating points, which is not the case in Figure 7.8, where the $hyst$-value is $10\ kPa$.

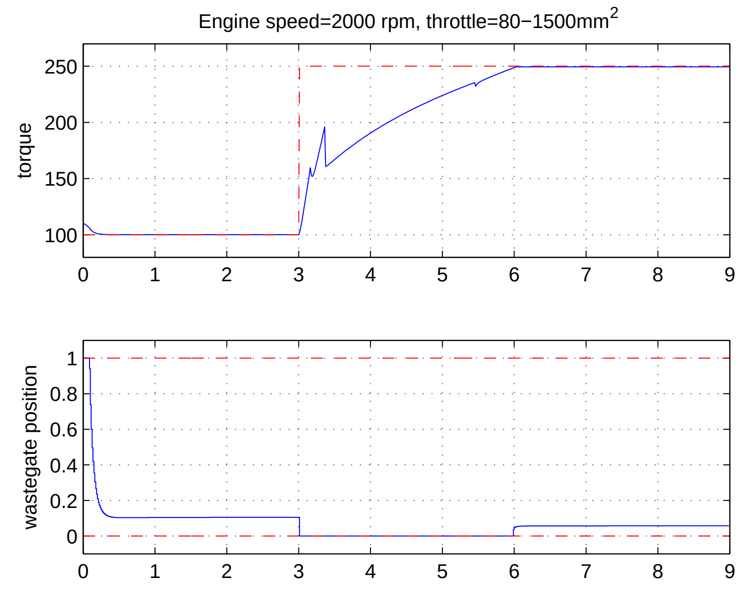

Figure 7.8: Oscillations in torque are reduced by introducing hysteresis

In Figure 7.6, 7.7 and 7.8 the dashed line in the upper picture is the torque reference and dashed line in the lower picture the constraints on wastegate position. The solid lines are resulting torque and wastegate position from closed loop simulation in Matlab carried out with the MPC controller that uses several operating points.

In the Matlab implementation all states are constrained. In Figure 7.9 one can see that all the states stay inside the allowed interval. The states are from the simulation with hysteresis. Most important is the upper constraint on turbocharger speed.

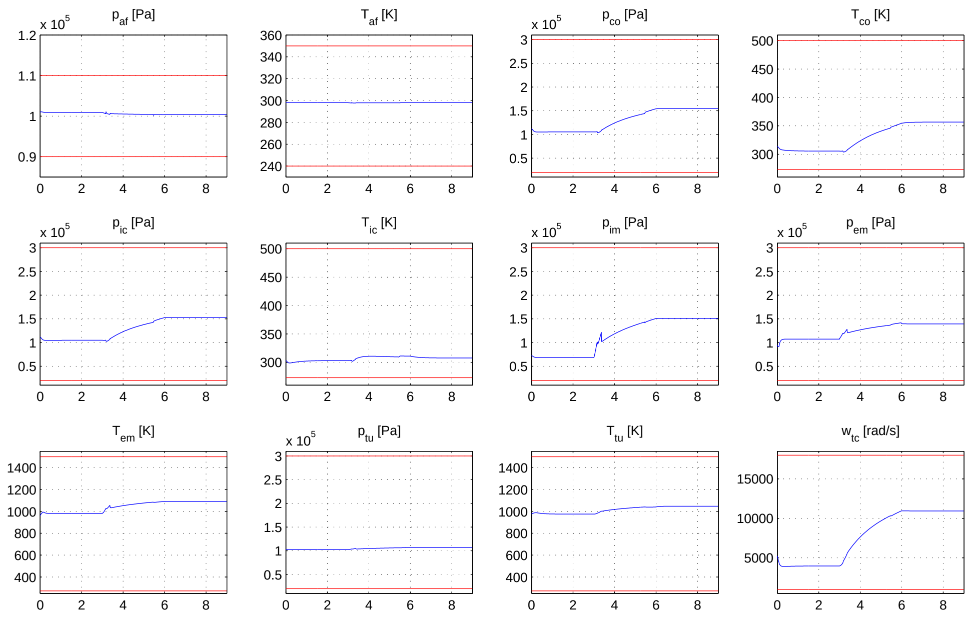

Figure 7.9: The states are inside the constraints

The tuning parameters for the Matlab implementation using several operating points are summarized in Table 7.2:

Parameter	Value
T	0.0125
N_p	10
Q_{torque}	1
Q_{fuel}	0
Q_{wg}	0 (Figure 7.6 and 7.8), 10^6 (Figure 7.7)
$Hyst$	0 (Figure 7.6 and 7.7), 10^4 (Figure 7.8)

Table 7.2: Tuning parameters used in simulations shown in Figure 7.6, 7.7 and 7.8

.

7.3 Implementation in Simulink

The difference in this section compared to Section 7.2 is that the controller is the MPC-block and the input signals are from the *nonlinear model* - not updated as in the implementation in Matlab.

One Operating Point in Simulink

First, only one operating point is used and therefore too big steps are not possible to make. In Figure 7.10 operating point number 12 is used. The engine speed is thereby given to $2000\ rpm$. The throttle opening area is a step from $30 - 50\ mm^2$.

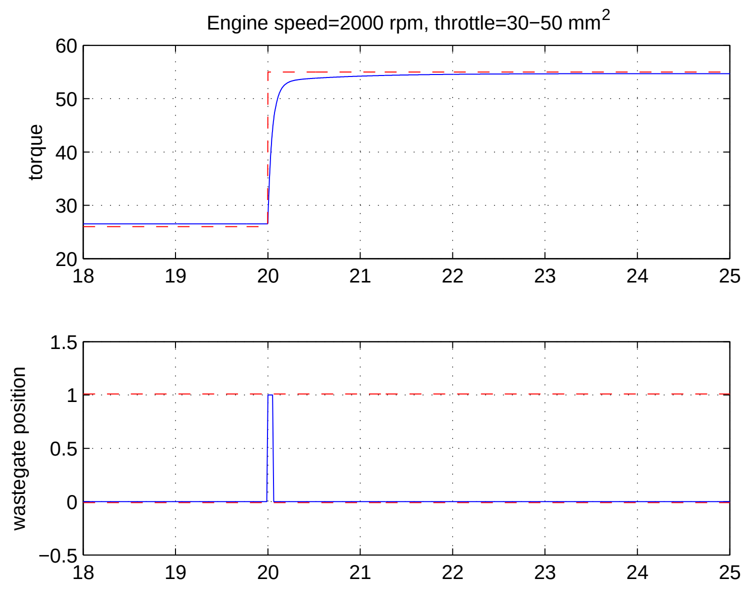

Figure 7.10: Opening of the wastegate seems to be beneficial

The torque reference curve is here a step from $28\ Nm$ to $55\ Nm$. The torque reference values come from a simulation of the nonlinear model without controller. The input signals are the same in the simulations with and without controller ($\alpha = 30 - 50\ mm^2$, $N = 2000\ rpm$, $\lambda = 1$, $p_{amb} = 101300\ Pa$ and $T_{amb} = 298\ K$), except for the wastegate which is closed for all times when the nonlinear model is simulated alone. The peak in the wastegate curve in Figure 7.10 indicates that it seems to be beneficial to open the wastegate for small torque values, just as the results from the implementation in Matlab also showed.

Just as for the implementation in Matlab it does not seem beneficial to open the wastegate at a transient for torque values where it, with todays mechanical design, would be possible to open it, see Figure 7.11. In all three following pictures the operating point number 15 is used. As before no weight is put on minimizing the fuel consumption.

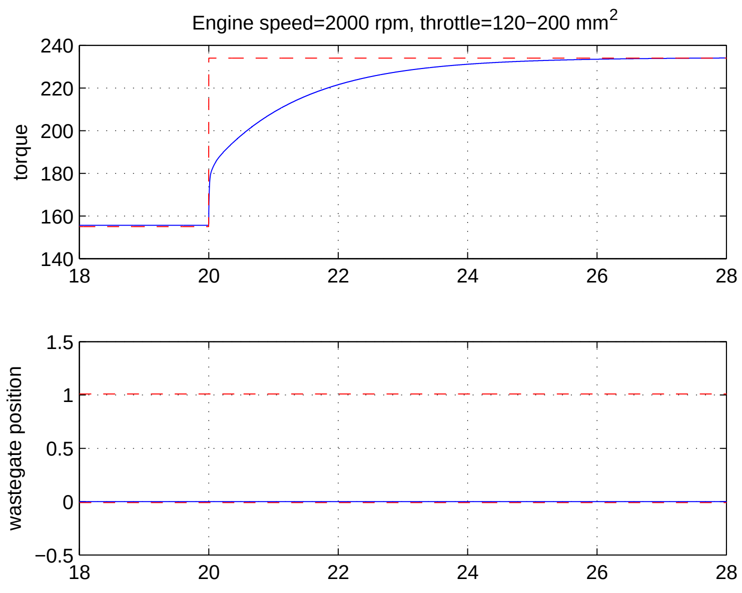

Figure 7.11: Opening of the wastegate seems not to be beneficial

In the next figure the torque reference is lowered to $210\ Nm$ so that the wastegate is open at the upper torque reference. For high torque values it is common that the wastegate is held a bit open to avoid that the turbine spins too fast. This scenario leads to that an oscillation phenomenon appears:

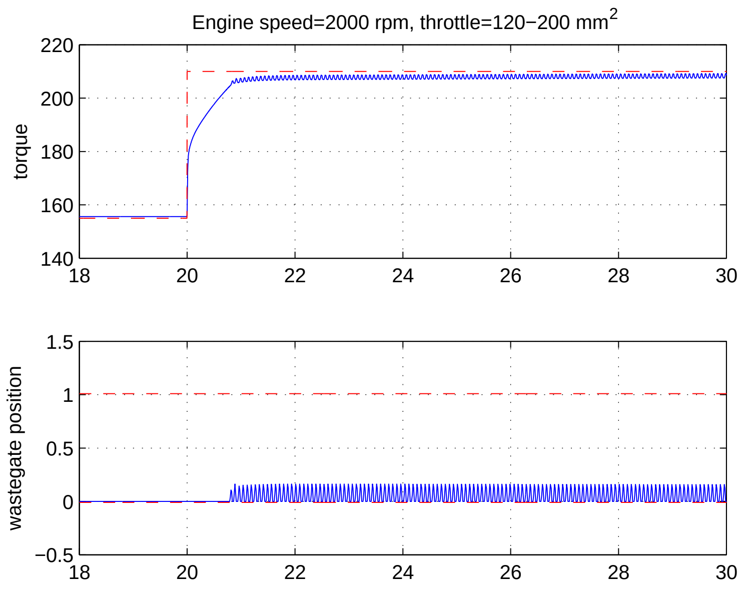

Figure 7.12: Oscillations in the wastegate appear when torque reference is changed

The problem with oscillations can be solved by introducing integral action. The tuning parameter Q_{wg} (or in MPC-toolbox called $InputRate$) was in Figure 7.11 and 7.12 zero. The only change made in the figure below is that $Q_{wg} = 10^3$. The stationary error in Figure 7.13 is however a little bit larger than the requirement in Defenition 7.1. That a stationary error occur even though a penalty on rapid changes in the wastegate position should introduce integral action is unexpected. The reason could be that the prediction horizon is not long enough or that there is a difference between the internal linear model and the external nonlinear model giving input signals to the MPC controller.

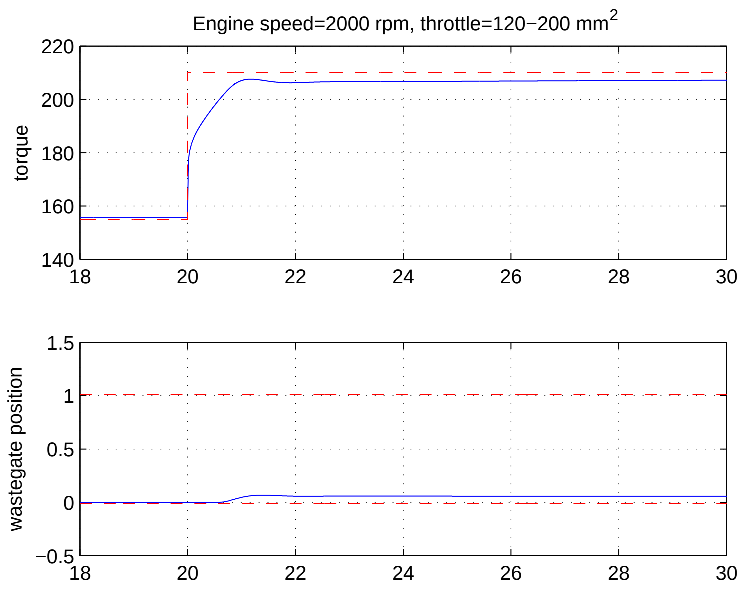

Figure 7.13: Integral action reduces oscillations

In Figure 7.13 we see how punishing rapid changes in the wastegate position eliminates the oscillations, but only reduces oscillations in Figure 7.7. The reason is that the oscillations are of different kinds. In Figure 7.13 only one operating point is used and the oscillations probably come from some fluctuations in the signals from the nonlinear model. These are well surpressed by penalizing rapid changes in the input signal. In Figure 7.7, several operating points are used and the oscillations come from that the controller is unable to decide which operating point to use. This problem is partly solved by penalizing rapid changes in the input signal, but it is better to use hysteresis.

The tuning parameters used can be found in Table 7.3:

Parameter	Value
T	0.0125
N_p	40
Q_{torque}	1
Q_{fuel}	0
Q_{wg}	0 (Figure 7.10, 7.11, 7.12), 10^3 (Figure 7.13)

Table 7.3: Tuning parameters used in simulation shown in Figure 7.10, 7.11, 7.12 and 7.13

In Figure 7.10, 7.11, 7.12 and 7.13 the dashed lines are torque reference and wastegate constraints and the solid lines are resulting torque and wastegate position from closed loop simulation in Simulink of the MPC controller that uses only one operating point.

Several Operating Points in Simulink

Here results are shown from simulations in simulink with changing operating conditions. When the operating point changes, the simulink model switches to another MPC-block with an MPC-controller based on the linear model corresponding to the new operating point. As described in Section 6.4 MPC controllers corresponding to operating point number 14, 18, 21 and 24 have been implemented.

The reason for using several operating points is the possibility to make a big change in throttle opening area and torque reference. It is therefore logical to choose one operating point for low loads and one for high loads. However, the operating points for high loads have the problem that the linear model differs from the nonlinear model as explained in Section 4.4. This difference between the linear internal model of the MPC-block and the nonlinear external model giving input signals to the MPC-block is one reason for that the performance is not that good. Another reason is that it is difficult to tune the controllers. For each operating point there is one controller that needs to be tuned. It is not certain that a long prediction horizon is good, since the predictions in one sample are based on one linear model only.

The controller reacts to changes in requested torque, but there is a stationary error of about $10\ Nm$, which is larger than the requirement in Defenition 7.1, see Figure 7.14.

The simulated scenario in Figure 7.14 is once again that the car has constant speed $100\ km/h$ and that the driver then gives full gas. Requested torque is a step from $100\ Nm$ to $250\ Nm$.

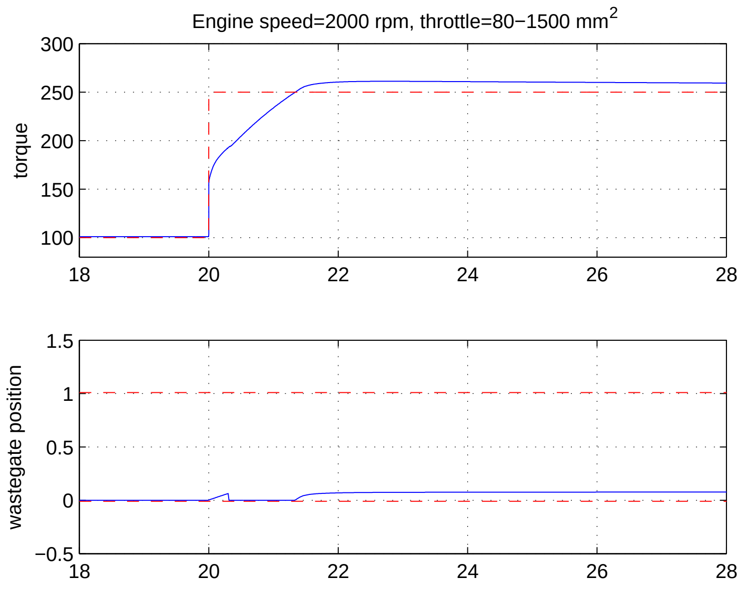

Figure 7.14: Simulation carried out with the Simulink MPC controller which uses several linear models

The dashed lines in Figure 7.14 are the torque reference and the wastegate constraints. The solid lines are the resulting torque and wastegate position from closed loop simulation in Simulink of the MPC controller which switches between operating points.

The parameters used in Figure 7.14 are summarized in Table 7.4:

Parameter	Value
T	0.0125
N_p	5 (oper. point 14), 40 (oper. point 18)
Q_{torque}	1
Q_{fuel}	0
Q_{wg}	10^3 (oper. point 14), 10^2 (oper. point 18)

Table 7.4: Tuning parameters used in simulation shown in Figure 7.14

Hysteresis

Hysteresis is not implemented in Simulink. The reason is that the MPC controller does not jump between operating points in this case. This is because the operating point is determined from the engine speed and intake manifold pressure of the nonlinear model. The jumping phenomenon occur when the present operating point is determined from engine speed and intake manifold pressure of different linear models as in the Matlab case.

Explicit MPC

After alot of work it turned out that even if the model was reduced as much as possible as in Sec.4.7, the resulting MPC controller still contains too many states, measured disturbances and references to be convertable to explicit form with the $expcon$ function.

Real Time Implementation

The communication with the car is done via an autobox. With the program *Control Desk* from *dSpace* it is possible to look at the signals and save data. Even though it is possible to use the MPC toolbox together with Real Time Workshop and generate code, the code generated is not compatible with dSpace. See supported toolboxes at:

```
http://www.dspace.com/ww/en/pub/home/support/supvers/supverscompm/...
supverscompmmlbls/rti53.cfm
```

The MPC controller developed with MPC toolbox can therefore not be tested in a real car.

Chapter 8

Conclusions

8.1 Discussion

In this thesis four MPC controllers have been developed. The problem statement was the same in all four, with the wastegate as manipulated variable and an objective to follow a given torque reference. The difference between the controllers are the models delivering input signals to the controller and the controller implementation.

The first two controllers are implemented in Matlab and the input signals come from the linear models. The first one of the Matlab controllers only uses one linear model. This is the simplest case and the controller performance is very good. The second controller changes between several linear models. This time variant implementation actually belongs to the field of nonlinear MPC. To get good controller performance it is necessary to implement integral action or hysteresis.

The other two controllers are implemented in Simulink. They get their input signals from a nonlinear model. In the first of these only one linear model is used. With some tuning this controller can be made to show acceptable performance. One reason for bad tracking performance is probably the difference between the internal linear model of the MPC controller and the external nonlinear model. Another reason is that to get good performance the controller need to be tuned for each torque reference curve.

The last controller switches between several MPC blocks based on different linear models. This controller is in the field of nonlinear MPC and the theory of linear MPC is not applicable. It is difficult to tune this controller to achieve acceptable performance because each MPC block has several tuning parameters. The tuning parameters most frequently used were the prediction horizon and the penalty on rapid changes in the manipulated variable. Since the controller switches between different linear models it is not sure that a long prediction horizon gives good performance.

One objective of this thesis was to investigate if it is beneficial to open the wastegate during a transient. From the simulations carried out, it seems to be beneficial only for very low torque values. With todays mechanical design, where the wastegate is opened by a pressure difference, it is not possible to open the wastegate for such low torque values. This would require an electronic control unit.

The weight on the fuel consumption has in this thesis mostly been chosen to zero. The reason is that it enables us to see how good the torque tracking performance is. If a penalty is put on fuel consumption, then the tracking performance shows a stationary error. It would be better if penalizing a large fuel consumption would lead to that it takes longer time for the torque to reach its final value, but that the final eventually would be reached. This requires a change in implementation.

In comparison with a PID-controller, which is the control method used in the automotive industry today, the MPC controller can account for constraints and it is easy to extend the implementation to the MIMO case. The problems with MPC in engine control is that so far the theory has been developed mainly for linear systems. For regulator control problems of processes that are not highly nonlinear, MPC has led to excellent results. One has been able to operate closer to constraints, which implies a larger profit. In engine control however, the control problems are of servo type and operating points change often and span a wide range of nonlinear dynamics. For a successful use of MPC in the automotive industry one probably has to wait a couple of years until the theory of nonlinear MPC is further developed and tools customized for the automotive industry exist.

8.2 Future Work

To get better agreement between the linear models and the nonlinear model, new linear models could be calculated in the operating points where the wastegate was held open during the linearization.

In order to implement an MPC controller in real time, it is probably best to try MPC on a system with fewer states and measured disturbances. Explicit MPC can then be used to reduce the computations and memory needed.

To use the MPC tools available today, that are developed to facilitate for the user, they must be adjusted to the needs of the automotive industry. For example the C-code generated with MPC toolbox need to be compatible with dSpace. Before the MPC toolbox is compatible with dSpace another method has to be used for real time implementation. For example the MPC controller could be implemented as an Embedded Matlab Function (EMF) in Simulink. With this method the optimization solver $quadprog$ has to be replaced, since it is not supported by EMF. An explicit code for the optimization problem solver is probably necessary.

Bibliography

Books, Articles and Thesises

[Ak02] M. Åkerblad. *A Second Order Cone Programing Algorithm for Model Predictive Control*. Licensiate Thesis, Royal Institute of Technology, Stockholm 2002.

[An05] P. Andersson. *Air Charge Estimation in Turbocharged SI-Engines*. Disortation, Linköping Institute of Technology, Linköping 2005.

[Ax03] D. Axehill and J. Sjöberg. *Adaptive Cruise Control for Heavy Vehicles*. Master Thesis, Linköping Institute of Technology, Linköping 2003.

[Be00] A. Bemporad. *The explicit linear quadratic regulator for constrained systems*. In Proc. of American Control Conference, pages 872-876, 2000.

[Be05] A. Bemporad. *Hybrid toolbox for Real Time Applications*. User´s Guide, 2005.

[Ca01] M. Cannon et. al. *Nonlinear Predictive Control - Theory and Practice*. IEE Control Engineering Series 61, Chippenham 2001.

[Co05] G. Colin et. al. *Linearized Neural Predictive Control of a Turbocharged SI-Engine Application*. SAE World Congress, Detroit 2005.

[Ed04] C. Edberg. *Control of Systems with Constraints*. Master Thesis, Royal Institute of Technology, Stockholm 2004.

[Er05] L. Eriksson and L. Nielsen. *Vehicular Systems*. Linköping Institute of Technology, Linköping 2005.

[Fi04] R. Findeisen et. al. *Nonlinear Model Predictive Control: From Theory to Application*. University of Stuttgart, Stuttgart 2004.

[Fo05] C. Forssman and E. Wiklund. *Bypass Valve Modeling and Surge Control for Turbocharged SI-Engines*. Master Thesis, Linköping Institute of Technology, Linköping 2005.

[Gl03] T. Glad and L. Ljung. *Control Theory: Multivariable and nonlinear methods*. Lund 2003.

[Gl02] T. Glad et. al. *Digital Control*. Linköping 2002. Language: Swedish.

[He88] J. B. Heywood. *Internal Combustion Engine Fundamentals*. Singapore 1988.

[Kv05] M. Kvasnica et. al. *Multi-Parametric Toolbox*. Users Guide, 2005.

[Ma02] J. M. Maciejowski. *Predictive Control with Constraints*. Essex 2002.

[Wa03] J. Wahlström. *Model Predictive Control of a Diesel Engine with Variable Geometry Turbine and Exhaust Gas Recirculation*. Master Thesis, Linköping Institute of Technology, Linköping 2003. Language: Swedish.

Homepages

[BEM] www.dii.unisi.it/ bemporad, 2005-12-21.

[ETH] http://control.ee.ethz.ch, 2005-12-21.

[HSW] www.howstuffworks.com, 2005-12-07.

[ISY] www.control.isy.liu.se/student/tsrt27/current.html Lecture 8 and 9, 2005-12-15.

[S3] www.s3.kth.se/control/kurser/2E1252/index.shtml Lecture 13, 2005-11-20.

Appendix A

Hysteresis Matlab Implementation

```
function oper_point=hysteresis_calc(lower_limit,upper_limit,...
                    pressure_value,old_oper_point)

activeIndex=old_oper_point;
y=2;
while y˜=0,
    if pressure_value<upper_limit(activeIndex) & ...
       pressure_value>lower_limit(activeIndex)
        y=0;
    end
    if pressure_value>upper_limit(activeIndex)
        y=1;
        activeIndex=activeIndex+1;
    end
    if pressure_value<lower_limit(activeIndex)
        y=-1;
        activeIndex=activeIndex-1;
    end
end

oper_point=activeIndex;
```

Appendix B

Figures to Section 7.2

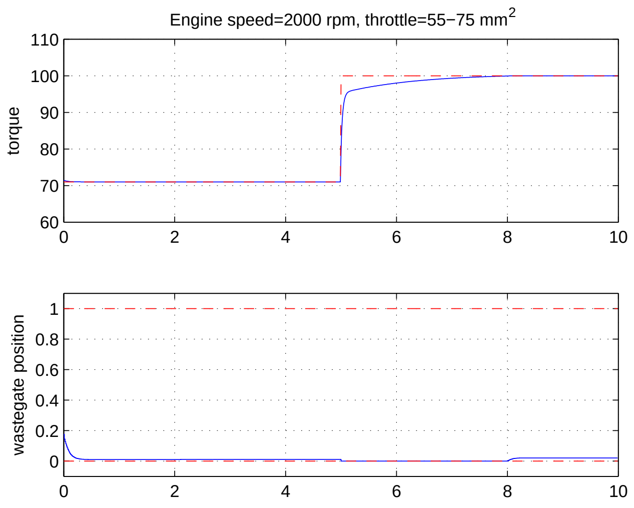

Figure B.1: Operating point 13: Opening of the wastegate seems not to be beneficial

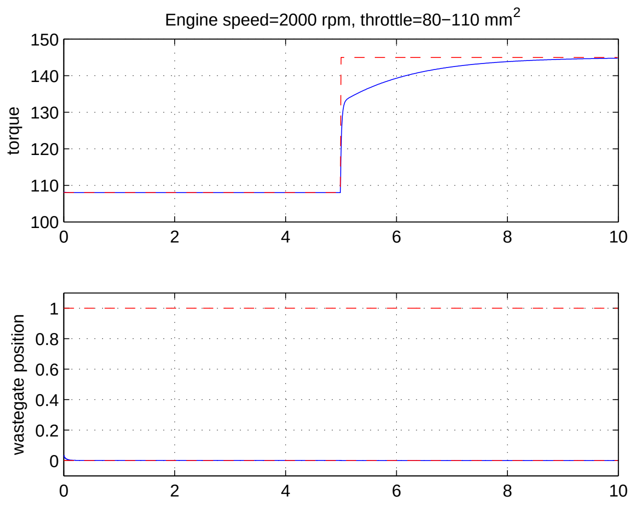

Figure B.2: Operating point 14: Opening of the wastegate seems not to be beneficial