Optimal control of a diesel engine with VGT and EGR

Master's thesis
performed in **Vehicular Systems**

by
Jonas Olsson and Markus Welander

Reg nr: LiTH-ISY-EX-44/3799-SE-2006

June 19, 2006

Optimal control of a diesel engine with VGT and EGR

Master's thesis

performed in **Vehicular Systems**,
Dept. of Electrical Engineering
at **Linköpings universitet**

by **Jonas Olsson and Markus Welander**

Reg nr: LiTH-ISY-EX-44/3799-SE-2006

Supervisor: **Johan Wahlström**

Examiner: **Associate Professor Lars Eriksson**
Linköpings Universitet

Linköping, June 19, 2006

Abstract

To fulfill todays requirements on emissions from engines, SCANIA has developed an engine with EGR (Exhaust Gas Recirculation) and VGT (Variable Geometry Turbine). This gives two extra control signals to take into consideration. Open loop optimal control is used to investigate how these two actuators should be controlled to minimize emissions and fuel consumption. A cost function, consisting of the errors between the most important variables and their set points, has been used in the minimization. The variables are the torque, the EGR mass fraction, the oxygen/fuel ratio and the pumping losses. From studies of the two control signals in different transients in the engine, information of how to control the VGT and EGR in the optimal way is found. The result from the optimal control has been compared with a PID simulation and has showed a better way to control the signals. The mayor reason why the optimal control is better than a PID controller is the ability to use future values from the transients.

Keywords: TOMOC, MATLAB, PID-controller, fmincon

Preface

This master's thesis has been performed at Vehicular System at Linköping University from September 2005 to May 2006, and is a part of PhD student Johan Wahlströms research.

Acknowledgment

We would like to thank our supervisor Johan Wahlström and our examiner Lars Eriksson for helping and supporting us. We also would like to thank Adam Lagerberg for helping us to use and understand TOMOC. A number of friends also deserves to be thanked for their help and support.

Contents

Chapter 1

Introduction

1.1 Background

1.1.1 EGR

Higher and higher environmental requirements are demanded from todays engines which means that the industry constantly have to develop new methods to reduce emissions. One method the engine industry have come up with to lower the emissions is called EGR (Exhaust Gas Recirculation), which means that some of the exhaust gas is recycled back into the engine to cool down the combustion temperature which leads to reduced NO_x emissions.

1.1.2 VGT

VGT (Variable Geometry Turbine) is a variable restriction which controls how much exhaust gas that is lead in to the turbo. The VGT has a set of vanes which can be used to control the turbine inlet pressure. The closing of the VGT vanes leads to an increase in the turbine inlet pressure which makes the turbine to spin faster and generate more boost in the engine. The VGT is also used to build up the needed pressure in the exhaust manifold in order to recirculate the exhaust gases back to the engine.

1.1.3 Objectives

The objective with this master's thesis is to find the optimal way to control the VGT and EGR in order to lower the emissions from the engine. Earlier a PID controller has been used to control the two valves, but since the system of the engine is non linear and cross correlated it is very difficult to see if a PID controller gives the best possible control. Since optimal control can not be used in the reality to control the two valves, it is used as a comparison for other control methods.

1.1.4 Earlier Work

Johan Wahlsröm has built up a mean value model of a diesel engine, using EGR and VGT, in MATLAB/SIMULINK [1] . He has also used a PID controller to control the VGT and EGR valves [7]. The simulation of the PID controller will be used as a comparison to the optimal control. Tomas Johansson has in a master's thesis [5] confirmed that the model describes the engine and its behaviour successfully.

Chapter 2

Engine Model

In this chapter the dynamic of the engine and it's functions will be explained. The equations of the states, control signals and some of the most important control parameters will be described.

2.1 The model

The engine used in this master's thesis is a diesel engine with EGR and VGT. A mean value model of this engine has been created in MATLAB/SIMULINK, by Ph.D. student Johan Wahlström at Linköpings University. The model has in earlier studies [5] been compared with a real engine and has been proofed to be reliable.
The model can according to [1] be written as:

$$\dot{x} = f(x, u) \tag{2.1}$$

where the states are:

$$x = (p_{im}, p_{em}, X_{Oim}, X_{Oem}, \omega_t, \tilde{u}_\delta, \tilde{u}_{egr}, \tilde{u}_{vgt})^T \tag{2.2}$$

states	description	unit
p_{im}	Intake manifold pressure	Pascal
p_{em}	Exhaust manifold pressure	Pascal
X_{Oim}	Fraction oxygen in intake manifold	%
X_{Oem}	Fraction oxygen in exhaust manifold	%
ω_t	turbine speed	rad/s
$\tilde{u}_\delta$	dynamic of the fuel injection actuator	%
$\tilde{u}_{egr}$	dynamic of the EGR actuator	%
$\tilde{u}_{vgt}$	dynamic of the VGT actuator	%

and the control signals are:

$$u = (u_{egr}, u_{vgt})^T$$

Control signals	description	unit
u_{egr}	egr actuator	%
u_{vgt}	vgt actuator	%

Other inputs used in the model are the engine speed,n_e, and the fuel injection actuator,u_δ.

Inputs	description	unit
n_e	engine speed	rad/min
u_δ	fuel injection actuator	mg/cycle

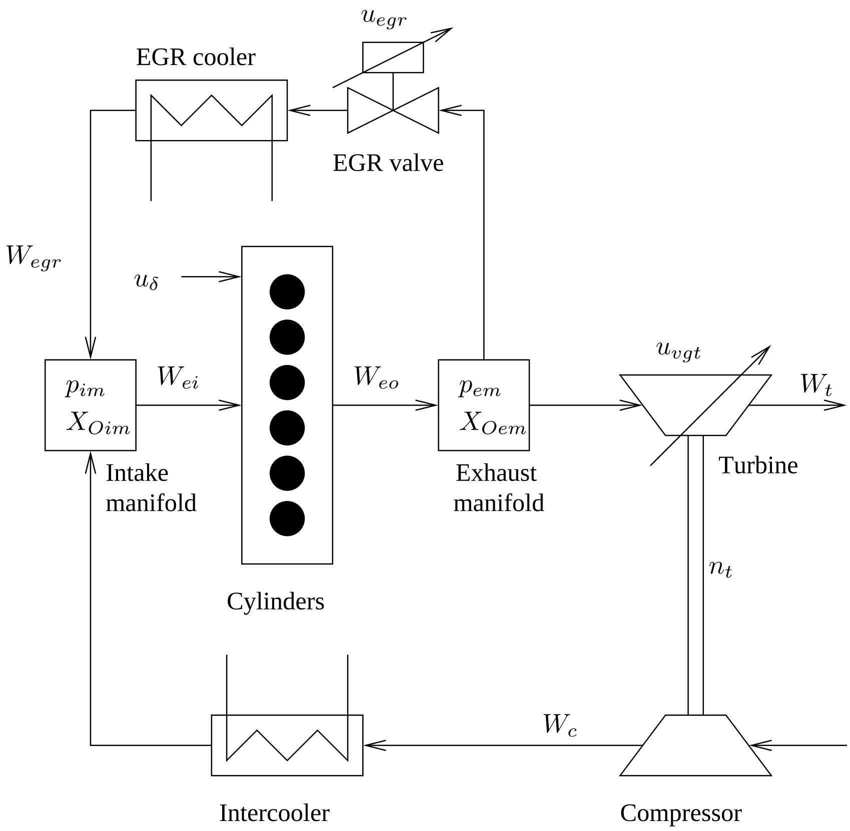

Figure 2.1: The structure of the diesel engine model.

2.2 Engine dynamics

2.2.1 Manifolds

The intake and exhaust manifolds are modeled as a dynamic control volume. The differential equations for the manifold pressures are based on the isothermal consumption,[4].

$$\frac{d}{dt}p_{im}(t) = \frac{R_a T_{im}}{V_{im}}(W_e + W_{egr} - W_{ei}) \tag{2.3}$$

$$\frac{d}{dt}p_{em}(t) = \frac{R_e T_{em}}{V_{em}}(W_{eo} + W_t - W_{egr}) \tag{2.4}$$

The EGR fraction in the intake manifold is based on the inflow.

$$x_{egr} = \frac{W_{egr}}{W_c + W_{egr}} \tag{2.5}$$

The dynamic of X_{Oim}(oxygen concentration in intake manifold) and X_{Oem} (Oxygen fraction in exhaust manifold) can be derived as [6]:

$$\frac{d}{dt}X_{Oim}(t) = \frac{R_a T_{im}}{p_{im} V_{im}}(W_{egr}(X_{Oem} - X_{Oim}) + W_c(X_{Oc} - X_{Oim})) \tag{2.6}$$

$$\frac{d}{dt}X_{Oem}(t) = \frac{R_e T_{em}}{p_{em} V_{em}}(W_{eo}(X_{Oe} - X_{Oem})) \tag{2.7}$$

where X_{Oc} is the oxygen concentration of the gas flow past the compressor i.e. the same as in air, and X_{Oe} the oxygen concentration in the exhaust gases out from the engine cylinders.

Symbol	Description	unit
W	Flow	g/s
T	Temperature	K
R_a	Gas constant for air	$J/(kg \cdot K)$
R_e	Gas constant for exhaust gas	$J/(kg \cdot K)$
V	Volume	m^3

2.2.2 Turbo

The turbo speed is modeled using a dynamic system with an inertia J_t

$$\frac{d}{dt}\omega_t(t) = \frac{M_t - M_c}{J_t} \tag{2.8}$$

M_t is the driving torque from the turbine and M_c is the braking torque from the compressor.

2.2.3 Stoichiometry combustion

In a diesel engine it is very important to keep control on the air/fuel ratio $(A/F) = \frac{m_a}{m_f}$ in order to decrease the smoke emission. A stoichiometric combustion reaction between a general hydrocarbon fuel and air, where the rest product only consists of water and carbon monoxide looks like

$C_aH_b + (a + \frac{b}{4})(O_2 + 3.773N_2) \rightarrow aCO_2 + \frac{b}{2}H_2O + 3.773(a + \frac{b}{4})N_2$

(2.9)

Where a and b represent the number of carbon and hydrogen atoms in one molecule in the fuel that are used in the engine. What is interesting in this equation is the relation between a and b, $y = \frac{a}{b}$ which shows the relative amount carbon in the fuel. It is now possible, using the molecular weights for carbon, hydrogen, oxygen and nitrogen, to write an expression for the stoichiometric mixture.

$(A/F)_s = \frac{m_a}{m_f}_s = \frac{(1+y/4)(32+3.773\cdot 28.16)}{12.011+1.1008y} = \frac{34.56(4+y)}{12.011+1.008y}$

(2.10)

$(A/F)_s$ has usually a value around 14.3.

We can now define λ , the air/fuel equivalence ratio, as:

$$\lambda = \frac{(A/F)}{(A/F)_s} \tag{2.11}$$

If $\lambda > 1$ it is indicate a surplus of air and if $\lambda < 1$ it's indicate a surplus of fuel. In a diesel engine λ must be larger than a minimum limit, e.g to avoid to much smoke emission.

Since it's the oxygen fraction and not the air fraction that is used in the model, λ is modeled as:

$$\lambda = \frac{W_{ei} X_{Oim}}{W_f O_c (A/F)_s} \tag{2.12}$$

Symbol	Description	unit
W_{ei}	Air flow in intake manifold	g/s
W_f	Fuel flow	g/s
O_c	Fraction oxygen in air	%

2.2.4 Mass fraction in EGR, x_{egr}

Introducing exhaust gases in the air-fuel mixture will reduce the combustion temperature and thereby reduce NO_x gases. Both emissions and fuel consumption will decrease with raised EGR ratio. x_{egr} is the mass fraction of exhaust gases in the air-fuel mixture. If x_{egr} is to low there will be to much NO_x emissions and if it is to high there will be to much smoke. The EGR fraction also displaces fresh oxygen, making it less available for combustion and thus reducing the probability of interaction between nitrogen and oxygen atoms even under lean conditions.

Emission limits are formulated as set-points in EGR fraction. It is therefore desirable to minimize the error between the x_{egr} mass fraction and it is set point, in a optimization point of view.

Opening the EGR valve will reduce the exhaust manifold pressure and thereby reduce the turbine speed. Opening the VGT valve will reduce exhaust manifold pressure and also decrease EGR fraction.

2.3 Actuator dynamics

2.3.1 The EGR valve state

$\tilde{u}_{egr}$ describes the EGR actuator dynamic and can be derived as:

$$\frac{d}{dt}\tilde{u}_{egr} = \frac{1}{\tau_{egr}}(u_{egr} - \tilde{u}_{egr}) \tag{2.13}$$

The EGR throttle is open when u_{egr}=100% and closed when u_{egr}=0%

2.3.2 The VGT position state

$\tilde{u}_{vgt}$ describes the VGT actuator dynamic and can be derived as:

$$\frac{d}{dt}\tilde{u}_{vgt} = \frac{1}{\tau_{vgt}}(u_{vgt} - \tilde{u}_{vgt}) \tag{2.14}$$

The VGT position is open when u_{vgt}=100% and closed when u_{vgt}=0%

2.3.3 Fuel injection

The amount of fuel injected $\tilde{u}_{delta}$ can be derived as

$$\frac{d}{dt}\tilde{u}_{delta} = \frac{1}{\tau_{delta}}(u_{delta} - \tilde{u}_{delta}) \tag{2.15}$$

which describes the actuator dynamic.

2.4 Summary

All the necessary equations and constants that are needed to describe the dynamic of the engine are available. The states and control signals are also successfully described.

Chapter 3

Optimization and PID-regulation

This chapter describes the optimization parameters, the control objectives and the cost function. The ETC cycle and the PID structure will also be described.

3.1 Control objectives

To fulfill all necessary requirements, the engine has to have a number of control goals. When values are received from the ETC-cycle it is important to know which parameters to control. In this case the control goals are:

1. Follow the set-point engine torque from the ETC-cycle.
2. Minimize the error between the x_{egr} fraction and its set point. If x_{egr} is to low there will be to much NO_x emissions and if it is to high there will be to much smoke.
3. In order to decrease the smoke, λ is not allowed to go below a certain limit.
4. The turbine speed, ω_t, can not be allowed to exceed a maximum limit, otherwise the turbo charger can be damaged.
5. Minimize pump losses in order to decrease the fuel consumption.

3.1.1 Cost function

The cost function is designed to take into consideration a number of different physical functions in the engine. It is also gives the possibility to weight different functions more than others.

The cost function used in this master's thesis is

$$min \sum_{j=1}^{n} C_1(u_{\delta setp}(j) - u_\delta(j))^2 + C_2(x_{egr}(j) - x_{egrSetp}(j))^2 + \quad (3.1)$$

$$C_3(p_{em}(j) - p_{im}(j)) + C_4(max(\lambda_{Setp}(j) - \lambda(j), 0))^2 +$$

$$C_5(u_{egr}(j) - u_{egr}(j-1))^2 + C_6(u_{vgt}(j) - u_{vgt}(j-1))^2$$

The first term minimizes the quadratic error between the real fuel injection (u_δ) and the desired fuel injection ($u_{\delta setp}$), which is calculated from the ETC cycle by the following equation.

$$u_{\delta_{setp}} = k_1 n_e(t)^2 + k_2 n_e(t) + k_3 M_{eSetp}(t) + k_4(P_{em} - P_{im}) + k_5 \quad (3.2)$$

where $M_{eSetp}(t)$ is the torque, n_e is the engine speed and k_1, k_2, k_3, k_4, k_5 are constants.

The second term is the quadratic error between EGR fraction (x_{egr}) and the reference value $x_{egrSetp}$. In this case $x_{egrSetp}$ has been set to 0.15, that is a value which was obtained from stationary measurements with emissions just below the legislated requirements(see [7]). This term helps control objective 2 in section 3.1 to be fulfilled.

The third term is the pump loses, which is the difference between the intake and exhaust manifold pressures. The reason to not minimize the quadratic error in the pumping loses is that a negative value is possible to attain, which is an advantage when minimizing the fuel consumption but a disadvantage when minimizing the x_{egr}. This term helps to fulfill control objective 5 in section 3.1.

The fourth term minimizes the difference between λ and λ_{Setp}. λ_{Setp} has been obtained in the same way as $x_{egrSetp}$. In this case $x_{egrSetp}$ is 1.8. Since there is no negative with a high λ, the whole term is set to zero if λ is higher than 1.8. This helps control objective 3 in section 3.1 to be fulfilled.
The fifth and sixth term is a way to avoid to much ripple in the control signals. The control signal is compared with an earlier signal and the difference between them is minimized.
C_1, C_2, C_3,C_4,C_5 and C_6 are parameters chosen depending on how much the

different terms are weight to each other. These are set depending on which term in the cost function that is desired to have the most influence. How the different terms effects the result will be described in section 5.

3.1.2 Constraints

In order to reach todays requirement on emissions, some limitations have to be set. It is also necessary to limit some physical values. The constraints in the optimization are

- $\lambda \geq 1.3$ to prevent to much smoke emissions.
- $n_t \leq 1.1 * 10^5$ rpm to make sure that the engine do not get damaged. n_t is the turbine speed
- $u_\delta \geq 0$,the fuel injection can not be negative.
- $0\% \leq u_{egr} \leq 100\%$
- $10\% \leq u_{vgt} \leq 100\%$

The states and control signals are also constrained in order to make the simulation time as short as possible. A higher number of constraints means fewer number of possible solutions which means a shorter simulation time. The constraints have been chosen after what is realistic for an diesel engine of this kind. There are also a number of constraints that have been set on physical parameters to make sure that the model will work in a realistic way (for example temperature T can not be below 0 K), those constraints are the same as in the original simulink model.

3.2 ETC-Cycle

As a reference for the following optimization the ETC-cycle (European Transient Cycle) has been used. This cycle is used for emission certification of heavy-duty diesel engines in Europe, starting in the year 2000. Different driving conditions are represented by three parts of the ETC cycle.

- Part 1 (0-600 s) represents city driving with a maximum speed of 50 km/h, frequent starts, stops and idling.

- Part 2 (600-1200 s) represents rural driving, starting with a steep acceleration segment. The average speed is about 72 km/h.

- Part 3 (1200-1800 s) is motorway driving with average speed of about 88 km/h.

Values from the ETC cycle comes in three parts. Vehicle Speed, Engine speed and Engine Torque as function of time. Where the sample time is one second. In this case the engine speed and torque has been used. In this case the engine speed from the ETC-cycle is the control signal n_e. The signal is taken directly from the cycle and is implemented in TOMOC as a simple bound. By setting the upper bound (UB) equal to the lower bound (LB) n_e will be treated as a fixed value in the optimization. But since the sample time from the ETC-cycle is 1 sec this signal have to be interpolated to get a higher resolution. This is solved by using the interpolation command in MATLAB. The result is a vector from t_1 to t_2 with the desired sample time as length. The desired fuel injection is calculated with values from torque and engine speed from the ETC-cycle.

$$u = C_1 n_e^2 + C_2 n_e + C_3 M_e + C_4(P_{em} - P_{im}) + C_5 \tag{3.3}$$

This value is then a part of the cost function where the goal is to get the real fuel injection to follow the calculated one.

3.3 PID-regulation

In the original simulink model of the engine, a PID controller is used to regulate the control signals. The design objective for the PID controller is to coordinate u_δ, u_{egr} and u_{vgt} in order to achieve the control objectives. In figure 3.1 a simulink schematic of the control structure is shown. The signals needed for the controller are shown as signals in figure 3.1. The set points and limits for the controllers are obtained from stationary measurements with emissions just below the legislated requirements and represented as look up tables as a function of operation condition. In tuning these set points are held constant. The engine torque is controlled to it's set point M_{eSetp} with the control signal u_{delta} using feed forward. The block Delta feed forward in figure 3.1 calculates the set point value for u_{delta}.

$u_{\delta setp} = c_1 n_e(t)^2 + c_2 n_e(t) + c_3 M_{eSetp}(t) + c_4(P_{em} - P_{im}) + c_5$

The parameters C_1 to C_4 are estimated from stationary measurements. λ

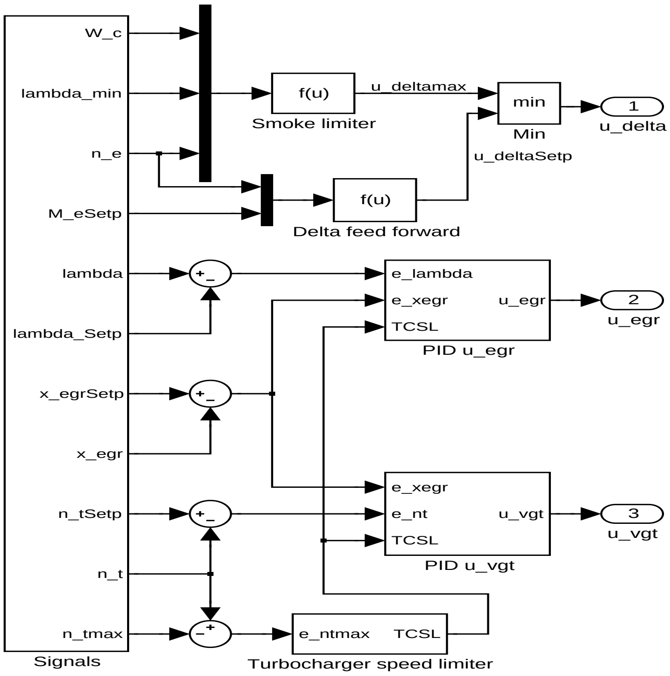

Figure 3.1: Simulink model of the PID structure

is controlled to a set point λ_{Setp} with the control signal u_{egr}. This causes a closing of the EGR valve during a load transient in order to speed up the turbocharger and increase the air mass flow into the engine. The intake manifold EGR-fraction x_{egr} is controlled to it's set point $x_{egrSetp}$ with the control signal u_{vgt}. The smoke limiter and turbocharger speed limiter are safety functions to prevent to much smoke and overshoots in speed. The two PID controllers are set up so that x_{egr} can be controlled both by u_{vgt} and u_{egr}.

The control objectives have been the same and the same engine dynamics have been used.

3.4 Summary

In this chapter optimization and control of the system have been introduced. Eq 3.1 is the cost function and is central in the optimization section and the control objectives in 3.1 are central in the control section. It has been described how a PID controller was used in the original model to fulfill the control objectives.

Chapter 4

TOMOC

In this chapter the engine model and the optimization is put together and a optimal control problem is formulated. The optimization algorithm TOMOC is described and explained.

4.1 Introduction

4.1.1 Background

TOMOC is an optimization algorithm developed by Adam Lagerberg at the School of Engineering Jönköpings University, in his PhD thesis. TOMOC is described in [2]. TOMOC is used for solving optimal control problems. There exists many different methods to solve optimization problems and there are many different software tools to use. But the usual optimization problem do not consider the dynamics, that are used in optimal control problems. TOMOC reforms the optimal control problem into a numerical minimization problem which can be solved in MATLAB. TOMOC was originally designed to use the general NLP-solvers implemented in TOMLAB, but since TOMLAB is not available in this thesis, a special version of TOMOC which uses the MATLAB function `fmincon` which follows with the optimization toolbox has been used.

4.1.2 Function

A general optimal control problem can be formulated as followes [3].

Find the control function $u(t), t \in [t_I, t_F]$ that minimize the cost function.

$$J = \Phi(x(t_I), x(t_F), t_I, t_F) + \int_{t_I}^{t_F} L(x(t), u(t), t)dt \tag{4.1}$$

The first part of the function defines cost related to the final and initial states. $x(t)$ is a vector containing all the states and $u(t)$ is a vector containing the control signals.
The state equations that defines the dynamic are:

$$\dot{x} = f(x(t), u(t), t) \tag{4.2}$$

$$u(t) \in U, 0 \leq t \leq t_F \tag{4.3}$$

$$x(0) = x_0, \Phi(x(t_F)) = 0 \tag{4.4}$$

Simple boundary conditions are:

$$x(t_I) = x_I \tag{4.5}$$

$$x(t_F) = x_F \tag{4.6}$$

Complex boundary conditions are:

$$g_I(x(t_I), u(t_I)) = 0 \tag{4.7}$$

$$g_F(x(t_F), u_F) = 0 \tag{4.8}$$

Simple constraints on state and control variables are:

$$x_{min} \leq x(t) \leq x_{max}, t \in [t_I, t_F] \tag{4.9}$$

$$u_{min} \leq u(t) \leq u_{max}, t \in [t_I, t_F] \tag{4.10}$$

Complex path constraints are:

$$g_L \leq g(x(t), u(t), t) \leq g_U, t \in [t_I, t_F] \tag{4.11}$$

The final time, t_F can be fixed or allowed to vary and be a part of the optimization problem.

The central idea in TOMOC is to implement a transcription formulation of the dynamics equation.The transcription is based on a direct collocation formula which discretizise the differential equations tons to solve the optimal control problem. The basis of this approach is a finite dimensional approximation of control and state variables i.e. a discretization. From this discretization, a constrained optimal control problem is transformed into a finite dimensional nonlinear program which can be solved by a standard NPL solvers. TOMOC transforms the differential equation to a non linear problem, with methods such as Euler or Runge-kutta. The non linear problem can be written as:

$$J = \Phi(x(t_I), x(t_F), t_I, t_F) + \sum_{j=0}^{N} L(x_j, u_j) \tag{4.12}$$

The non linear problem can be solved in MATLAB by fmincon.

4.1.3 fmincon

`fmincon` is a MATLAB function which solves non linear optimization problems. `fmincon` uses a method called SQP (Sequential Quadratic Programming) to find a constrained minimum of a scalar function of several variables starting at an initial estimate. `fmincon` starts at X0 and attempts to find a minimum X to the function described in cost function subject to the linear inequalities. In this case X0 is a vector. The function to be minimized, the cost function, is a function that accepts a vector x and returns a scalar f. There is no way to know if the found minimum is local or global, because `fmincon` always searches locally for decreasing solutions.

4.2 Implementation

To make `fmincon` work we need the following in data.

```
[X] =fmincon(FUN,X0,Aieq,Bieq,Aeq,Beq,LB,
     UB,NONLCON,OPTIONS,P)
```
(4.13)

where

- `FUN` is the cost function.
- `X0` are the initial values.

- Aeq and Beq are inequality constraints.
- Aeq and Beq are equality constraints.
- LB and UB are lower and upper bounds for the states and control signals.
- NONLCON are the linear and nonlinear constraints.
- OPTIONS are parameters for the simulation.

P is a struct in MATLAB that contains user defined options.

With these values fmincon gives a vector X that contains values of the states and control signals that minimize the cost function for a specific time interval.

In the original TOMOC, A and B are matrices that describes the dynamic on the form $\dot{x} = Ax + Bu$. However, it is only possible to have the dynamic on this form if it is linear. In the diesel engine dynamic, that is not the case. Therefore it is necessary to get the dynamic from a separate file. The file Statesmaker uses the engine dynamic and creates the wanted states, and return them on correct form to the start file and fmincon.

4.2.1 Implementation of the engine model in TOMOC

The only way to implement the model in TOMOC is to write down the dynamics in MATLAB code.
To confirm the accuracy of the model a small simulink model was built, which can be seen in figure 4.2. If the model is correct, the resulting step response from a transient, should correspond with the step response in the reference model. The engine model is collected from workspace and has the derivate states as output. The input consists of control signals from the reference model and feedback states. The states was then plotted and compared with plots from the reference model. The plots in figure 4.1 show that the system gives the same solution, after an initial transient.

The figure shows what happens when the VGT valve goes from 90% open to 70% open and then back to 90% again. The reason why the two models not are exactly the same, which would be the case if everything was correct, is that some rounds off have been made and that different initial values have been used, but that difference can be neglected.

The block called MATLAB fcn, in figure 4.2 contains the engine dynamic.

The inputs to the block are the current states x and control signals u. The outputs are the derived states $\dot{x}$.

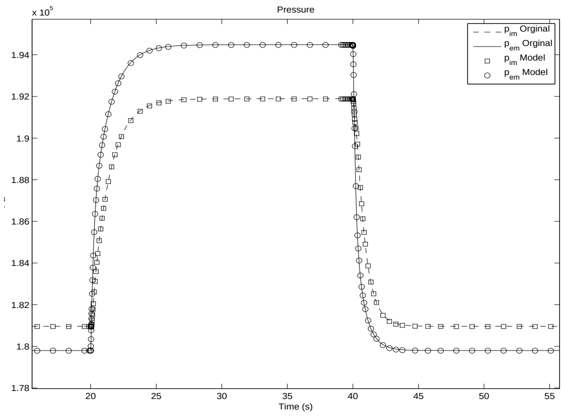

Figure 4.1: Step response in P_{im} and P_{em} for the original simulink model and the model implemented in TOMOC

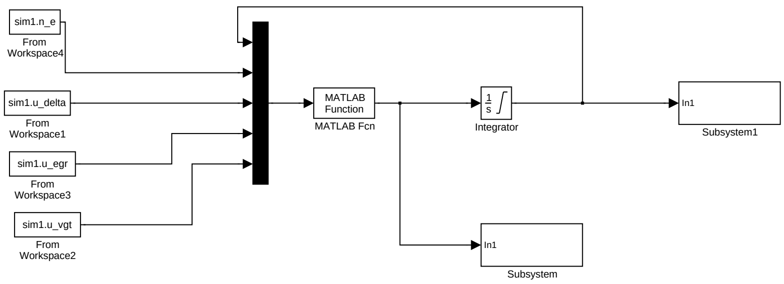

Figure 4.2: Simulink model used for validation of the model implemented in TOMOC

4.3 Summary

The optimization algorithm TOMOC can, with some modifications, be used to solve the wanted optimal control problem for the engine. For more information about TOMOC, see [2] and appendix A. It has been shown that the model implemented in TOMOC is correct since it gives the same step response as the original simulink model.

Chapter 5

Results

In this chapter the results from all simulations will be presented. It will also be described how the different simulations have been implemented. The results will be commentated and analyzed.
To find the optimal control signals of the VGT and EGR, several different simulations with a number of different settings have been done. Not only have the parameters of the model been changed but also the specific method to simulate the model. The simulations have also been compared with earlier simulations, made with a PID controller instead of optimal control with TOMOC. In this way it is possible to see if TOMOC gives a better result than a normal PID. The result has finally been tested in the original simulink model to make sure that they are correct. The simulations have been done with transients from a 20 s long time interval in the ETC-cycle. The transients can be seen in figure 5.1 and 5.2

5.1 Simulations

The simulation consists of five 4 s simulations that has been put together to one 20 sec long simulation. That's because TOMOC has a difficulty with long simulations with a short sample time. To solve that the simulation has been divided up in five 4 sec interval with a 16 segment, i.e. a sample time of 0.25 sec. It would be desirable to have a faster sample time, but the long simulation time makes that impossible.

5.1.1 Initial values

TOMOC is very sensitive for different initial values. It is important to have as correct values as possible or else no solution will be found. To solve this prob-

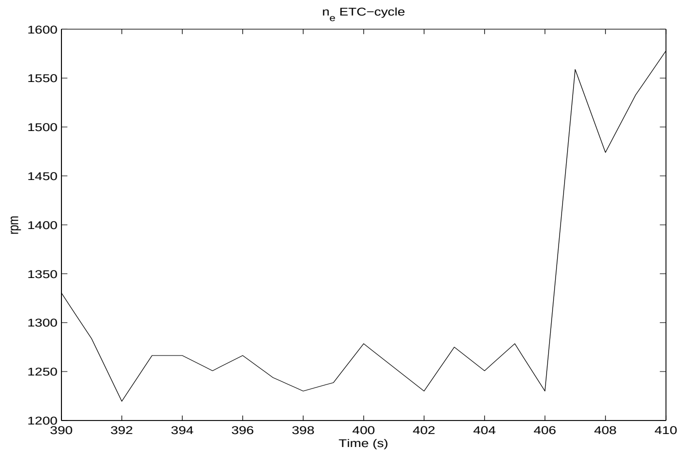

Figure 5.1: The engine speed from the ETC cycle.

lem values from earlier simulations, with a PID regulator, have been used. This can only be done with the assumption that the PID regulator gives a result that is rather close to the optimal result. This means that the result will have the same principal appearance as the PID simulation, but since the regulation objectives are the same, it is quite natural.

5.1.2 ETC cycle

In the simulation, the time interval 390-410 s of the 1800 s long cycle has been used. The engine speed and the torque from the ETC cycle can be seen in figure 5.1 and 5.2.

5.1.3 EGR and VGT

The EGR and VGT optimal way to control the engine can be seen in figure 5.3 and 5.4, and the states $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$ can be seen in figure 5.5.

5.1.4 Mass fraction in EGR

The mass fraction in the EGR, x_{egr}, can be seen in figure 5.6. The x_{egr} has much ripple the first seconds but after 6 sec it converges to its set point on 0.15. The reason that x_{egr} is below 0.15 in the beginning is because the EGR damper is closing in that time interval.That leads to a lower x_{egr} fraction. It shows in figure 5.6 , that this simulation gives a better result on the x_{egr} than a PID regulator.

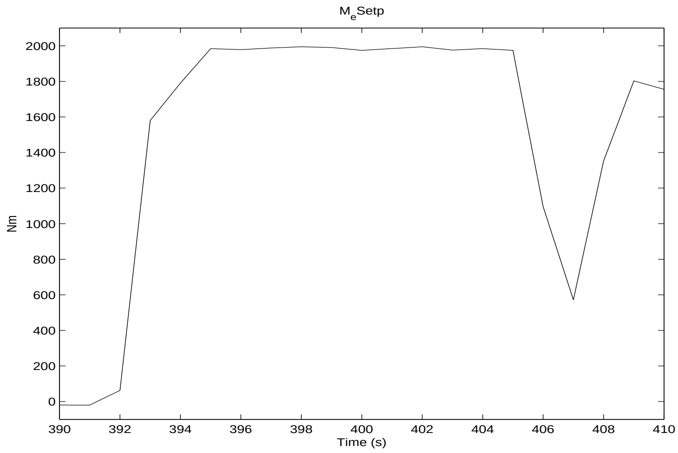

Figure 5.2: The torque from the ETC cycle.

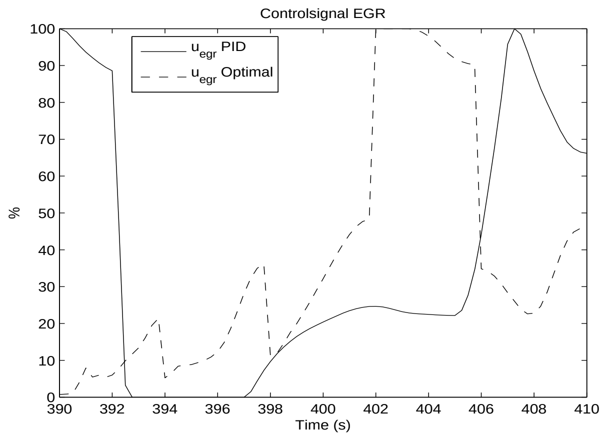

Figure 5.3: The optimal EGR signal and the EGR signal from the PID simulation.

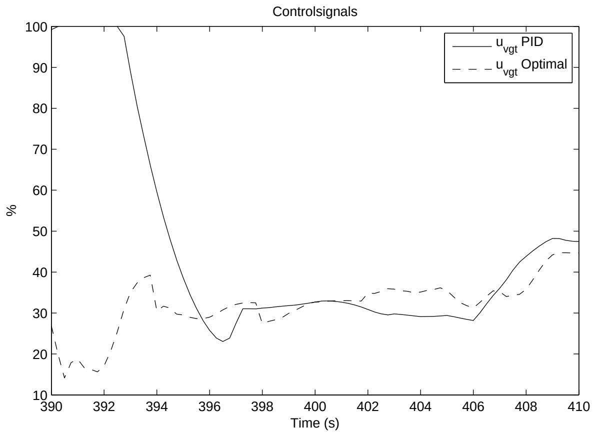

Figure 5.4: The optimal VGT signal and the VGT signal from the PID simulation.

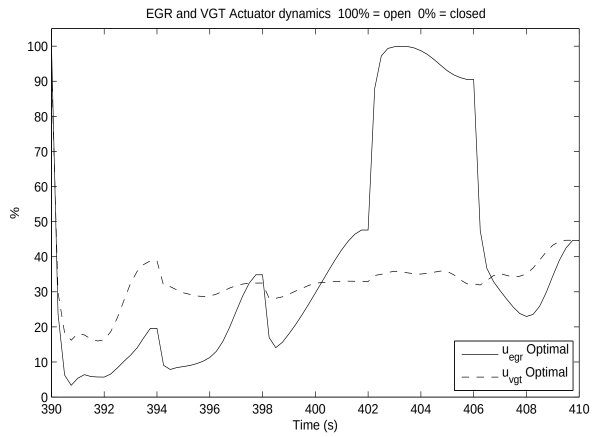

Figure 5.5: The states of the optimal EGR and VGT signals.

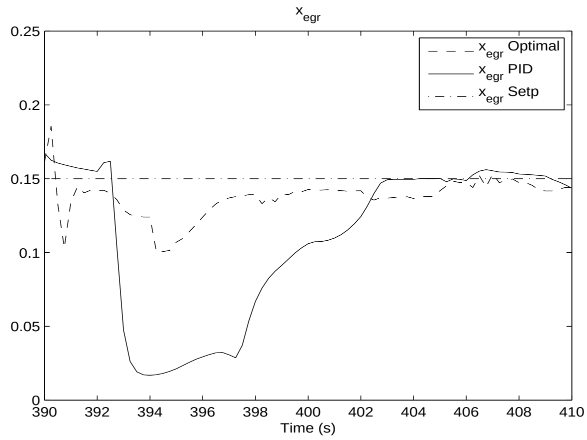

Figure 5.6: The EGR mass fraction from the optimal control signals and from the PID-controller compared to the set point.

5.1.5 Turbine speed

The turbine speed is higher than the PID simulation the first 8 s, this means that it will be a faster response in the transients. The rest of the time, except the two last seconds, the optimal turbine speed is lower than the PID simulation. That means lower fuel consumption and lower wear on the turbine. The reason is because the pressure in the exhaust manifold is not as high here. It can be seen in figure 5.11 that after 6 s the pump losses from our simulation and the PID simulation starts to differ, which leads to a difference in the turbine speed at the same time.

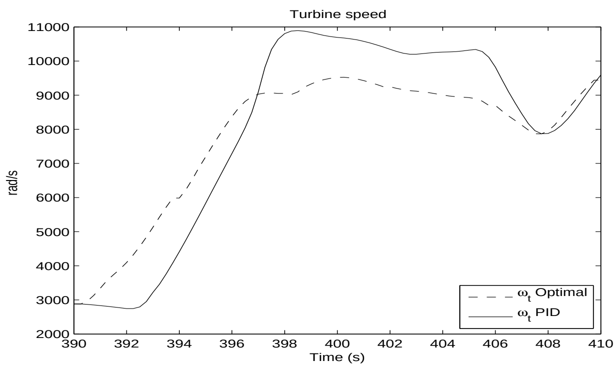

Figure 5.7: Turbine speed from the optimal control signals and from the PID-controller.

5.1.6 Oxygen fraction in intake and exhaust manifold

X_{Oim} lays around 0.2 which is the oxygen fraction in the air. X_{Oem} has the same principal appearance as λ in figure 5.9.

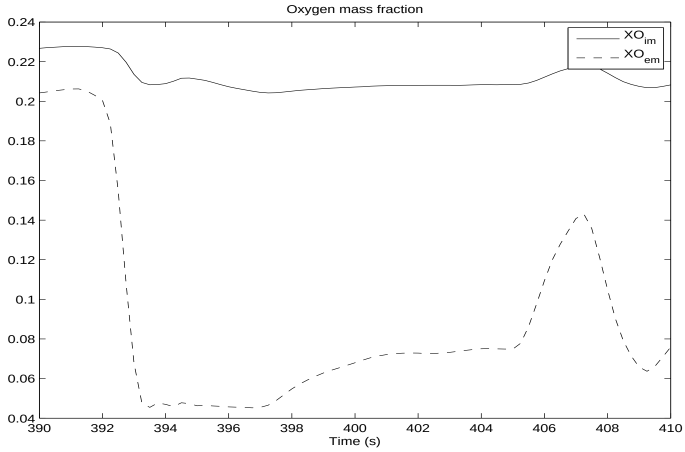

Figure 5.8: X_{Oim} and X_{Oem} from the optimal control signals.

5.1.7 Oxygen to fuel ratio, λ

Figure 5.9 shows that λ, most of the time, is lower than λ_{PID}. A high lambda is to prefer but since λ follows the given restrictions, $\lambda > 1.3$, the requirements are fulfilled.

5.1.8 Torque

To get an understanding of how good u_δ follows its set point compared to other simulations, the torque M_e is the most appropriate parameter to use. Since $u_{\delta Setp}$ is a function of n_e,M_e and the pump losses, it will change for every simulation and not be a constant parameter. M_{eSetp}, on the other hand, is taken directly from the ETC cycle and is the same under all circumstances and it is also the dominating factor in the function for $u_{\delta Setp}$. Thereby u_δ and M_e will have the same principal appearance. This means that it will be easier to see how the set point is followed when it does not change. Figure 5.10 shows that the optimal torque has a faster response than the PID simulation. The reason for this is the higher turbine speed in section 5.1.5.

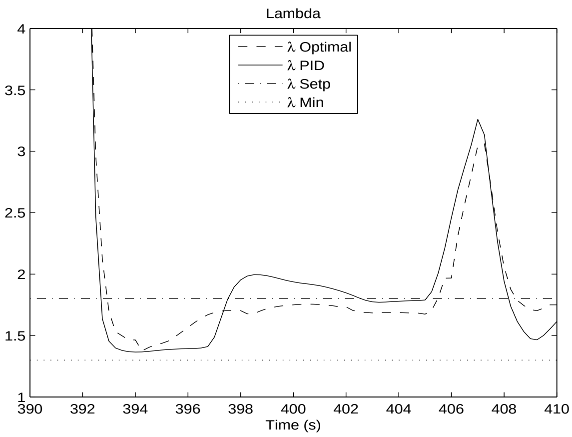

Figure 5.9: λ from the optimal control signals and λ from the PID-controller compared to the set point and to λ_{min}.

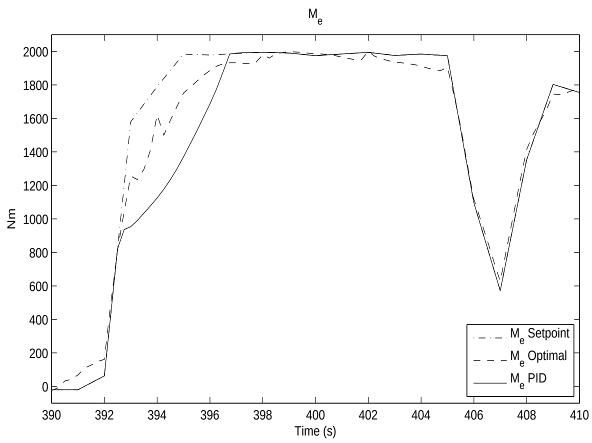

Figure 5.10: The torque from the optimal control signals and from the PID controller.

5.1.9 Intake- and exhaust manifold pressure

Figure 5.11 shows that the difference between p_{im} and p_{em}, the pump loses, are smaller than it is with the PID regulator. The reason why the pumping losses is lower than in the PID simulation can be seen in figure 5.3. We can see that the EGR valve is more open in our simulation compared to the PID simulation. The EGR valve also has a faster opening than the PID simulation. If the EGR valve is open, the pressure can not be built up which leads to lower pumping losses.

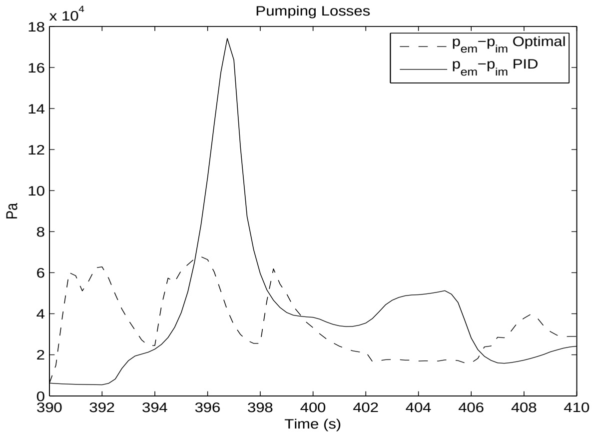

Figure 5.11: Pump losses from the optimal control signals and from the PID controller.

5.1.10 Time mean value

It is interesting to see how much better optimal control is than a PID-controller. The time mean value of the different signals can be calculated by solving the integral in equation(5.1), where y is the value of the signals for each time point. This gives the results that are presented in the following tabular.

$$\overline{y} = \frac{1}{T}\int_{t_0}^{t_0+T} y dt \tag{5.1}$$

Description	Signal	Change from PID to optimal %
EGR error	$\lvert x_{egrSetp} - x_{egr} \rvert$	-68%
Positive λ error	max($\lambda_{Setp} - \lambda$,0)	-14%
Torque error	$\lvert M_{eSetp} - M_e \rvert$	-9%
Pumping losses	$p_{em} - p_{im}$	-14%

All signals that the cost function contains decreases,i.e. follows the set point better, with the optimal control signals. Most obvious is it on the x_{egr} signal which improves with 68% compared to the PID-simulation. The reason why the optimal control is better than the PID controller is because it is closing the EGR and VGT valve before the transient is coming, to build up a pressure before the transient to get a quick response and thereby be able to control the x_{egr} in a better way. This is possible because the optimal control have information about future values which it gets from the X vector in section 4.2.

5.2 Result with different weight parameters

The four terms that are minimized in the cost function can be weighted with the constants C_1 to C_4 depending on which control objective or objectives in 3.1 that is/are considered to be the most important. When different objectives are prioritized , the results will also be different. It also gives an idea of how each term effects the result of the whole cost function.

5.2.1 Low weight on x_{egr}

If the constant C_2 in equation 3.1 is set to zero, the quadratic error between the EGR mass fraction and its set point won't be a part of the cost function. Figure 5.16 shows that x_{egr} doesn't follow the set point as good as before. A low EGR mass fraction means a higher oxygen fraction in the air and results in that more fuel is injected which can be seen on the torque in figure 5.15. This also means that the pump losses can be reduced by closing the EGR and open the VGT. This can be seen in figure 5.14.

The numerical change compared to the normal simulation are:

Description	Signal	Change from normal to $C_2 = 0$ %
EGR error	$\lvert x_{egrSetp} - x_{egr} \rvert$	96%
Positive λ error	max($\lambda_{Setp} - \lambda$,0)	76%
Torque error	$\lvert M_{eSetp} - M_e \rvert$	28%
Pumping losses	$p_{em} - p_{im}$	-112%

All values except the pump losses are worse than the normal simulation. The reason why the pump losses are so much better is because they are negative. That's because the VGT valve is opening up so much.

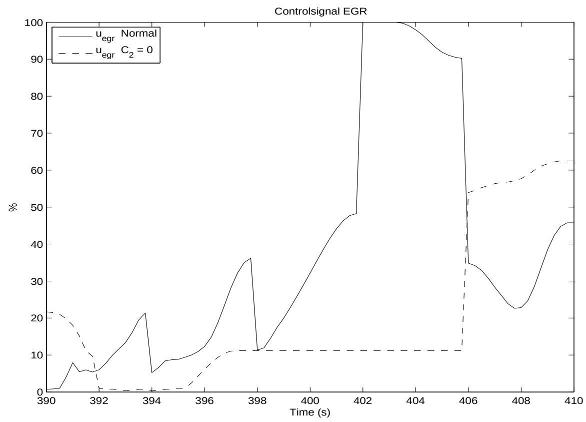

Figure 5.12: EGR when C_2=0, i.e. low weight on x_{egr} compared with the normal weight.

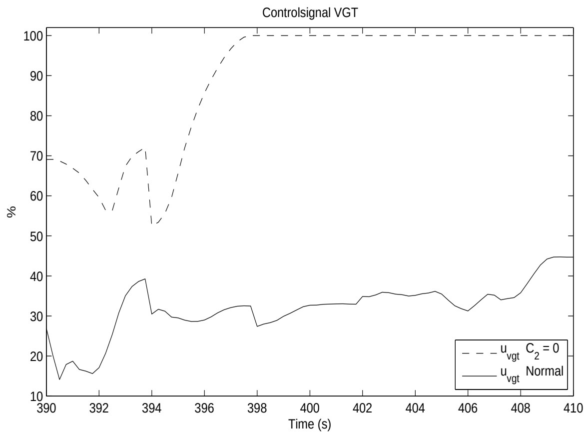

Figure 5.13: VGT when C_2=0, i.e. low weight on x_{egr} compared with the normal weight.

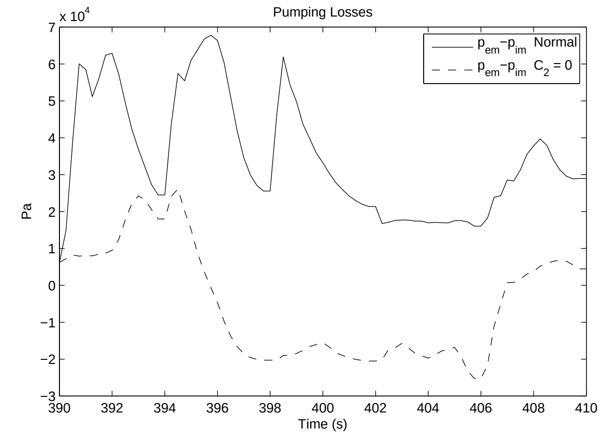

Figure 5.14: Pump losses when C_2=0, i.e. low weight on x_{egr} compared with the normal weight.

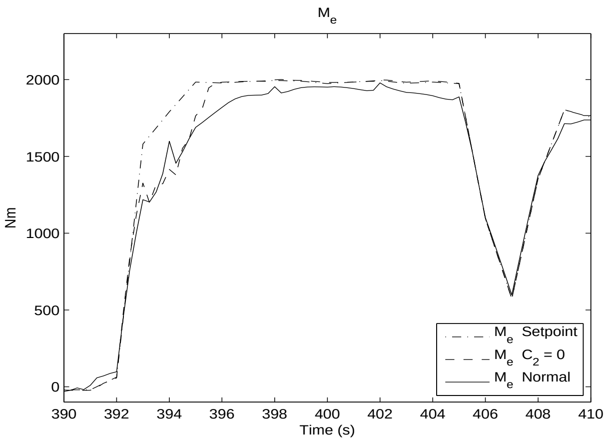

Figure 5.15: Torque when C_2=0, i.e. low weight on x_{egr} compared with the normal weight.

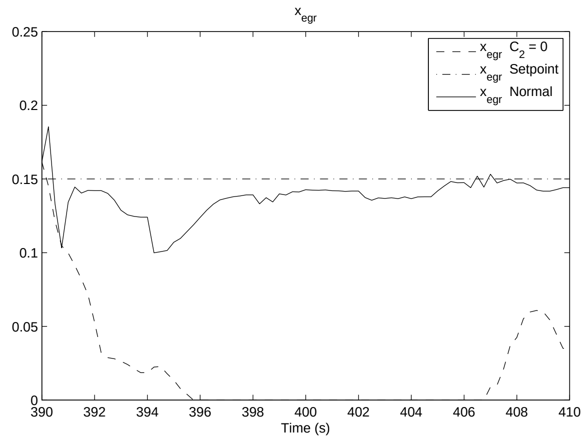

Figure 5.16: x_{egr} when C_2=0, i.e. low weight on x_{egr} compared with the normal weight.

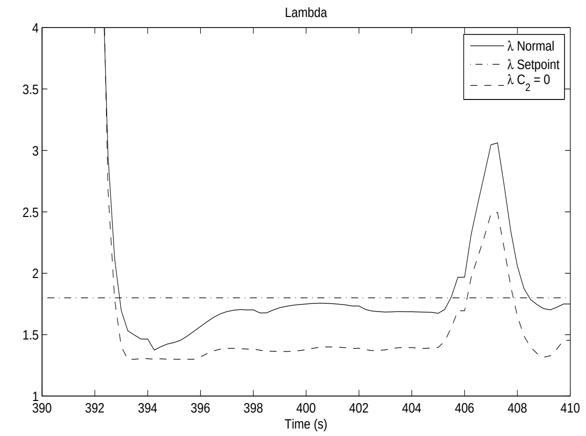

Figure 5.17: λ when C_2=0, i.e. low weight on x_{egr} compared with the normal weight.

5.2.2 Low weight on λ

Low weight on the quadratic error between λ and its set point means lower requirements on the air/fuel ratio and gives the possibility to inject more fuel. This results in that u_δ follows its set point better and slightly better pump losses, which can be seen in figure 5.20 and 5.21.

Description	Signal	Change from normal to $C_4 = 0$ %
EGR error	$\|x_{egrSetp} - x_{egr}\|$	68%
Positive λ error	max($\lambda_{Setp} - \lambda$,0)	13%
Torque error	$\|M_{eSetp} - M_e\|$	9%
Pumping losses	$p_{em} - p_{im}$	14.5%

Low weight on λ don't effect the result so much. The values are almost the same as in section 5.1.10.

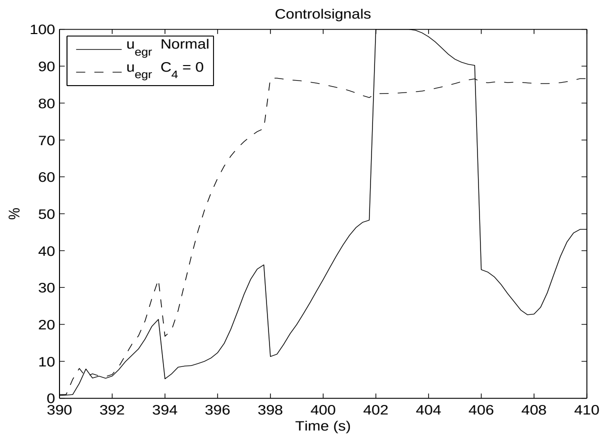

Figure 5.18: EGR when C_4=0, i.e. low weight on λ compared with the normal weight.

5.2.3 Low weight on u_δ

Low weight on the quadratic error between u_δ and its set point results in lower fuel injection, which can bee seen in figure 5.27. This results in that x_{egr} follows its set point better, figure 5.26 since not as much oxygen is needed as before. The pump losses are also lower, figure 5.25.The reason why there is a peak in $p_{em} - p_{im}$ after 4 sec is because the VGT is closing quickly right before, see figure 5.24.

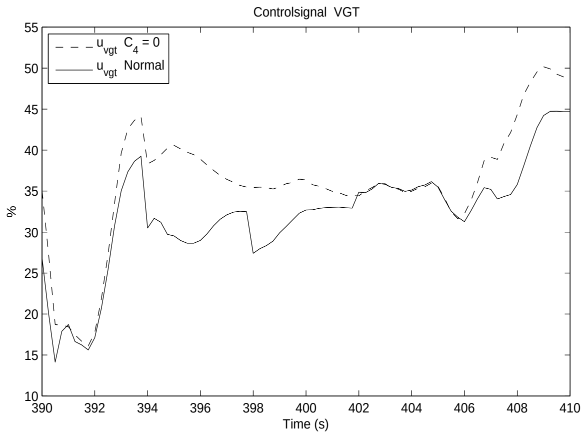

Figure 5.19: VGT when C_4=0, i.e. low weight on λ compared with the normal weight.

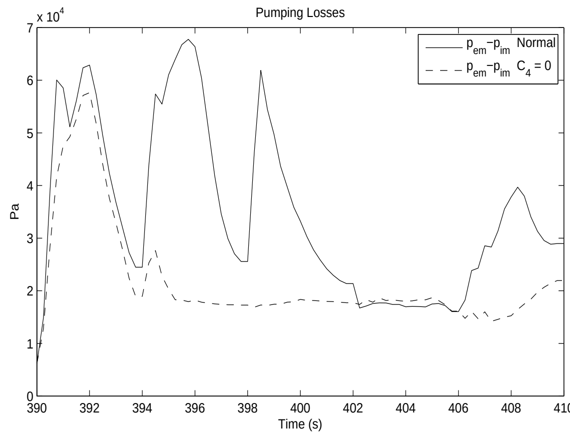

Figure 5.20: Pump losses when C_4=0, i.e. low weight on λ compared with the normal weight.

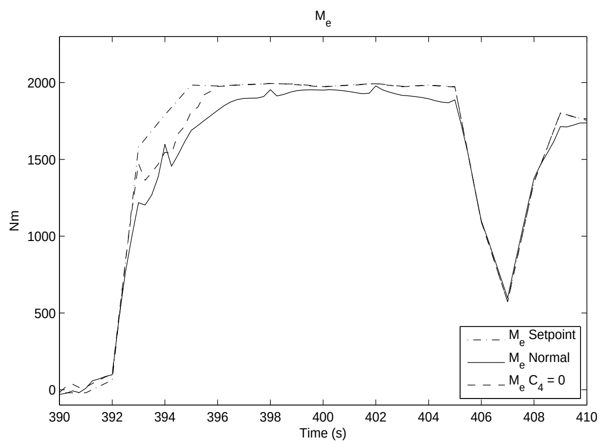

Figure 5.21: Torque when C_4=0, i.e. low weight on λ compared with the normal weight.

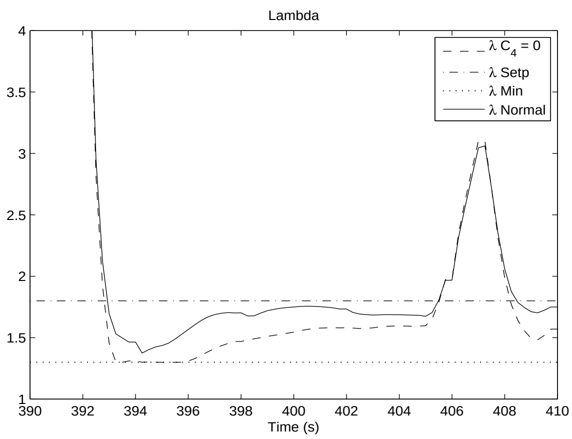

Figure 5.22: λ when C_4=0, i.e. low weight on λ compared with the normal weight.

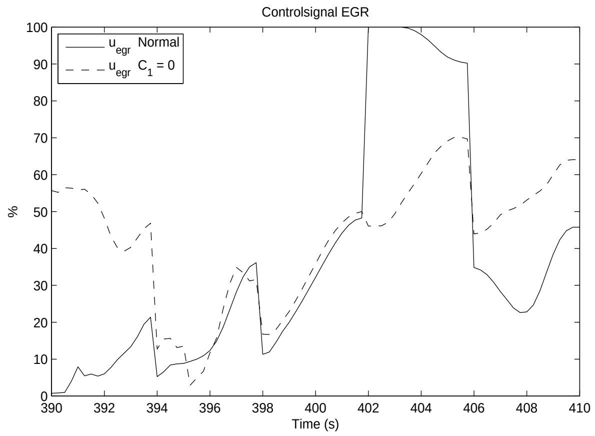

Figure 5.23: EGR when C_1=0, i.e. low weight on u_δ compared with the normal weight.

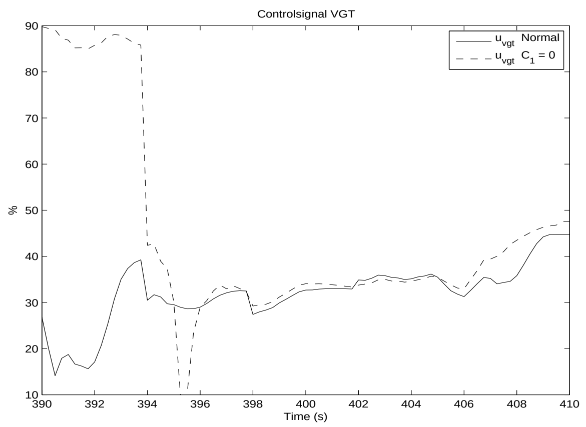

Figure 5.24: VGT when C_1=0, i.e. low weight on u_δ compared with the normal weight.

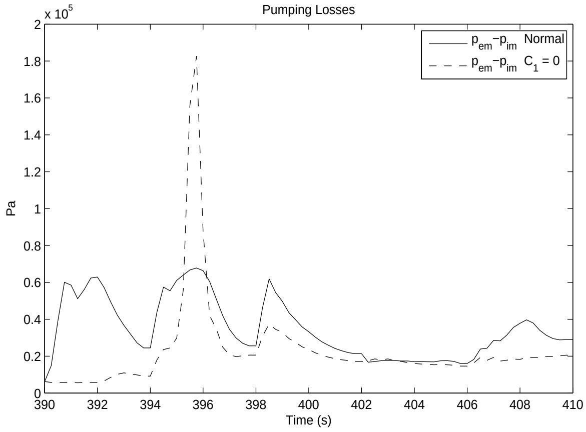

Figure 5.25: Pump losses C_1=0, i.e. low weight on u_δ compared with the normal weight.

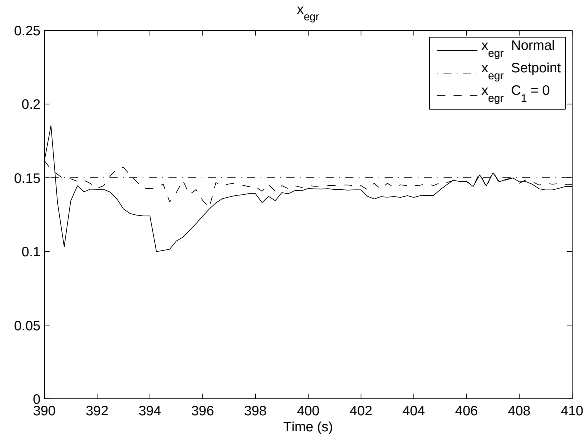

Figure 5.26: x_{egr} when C_1=0, i.e. low weight on u_δ compared with the normal weight.

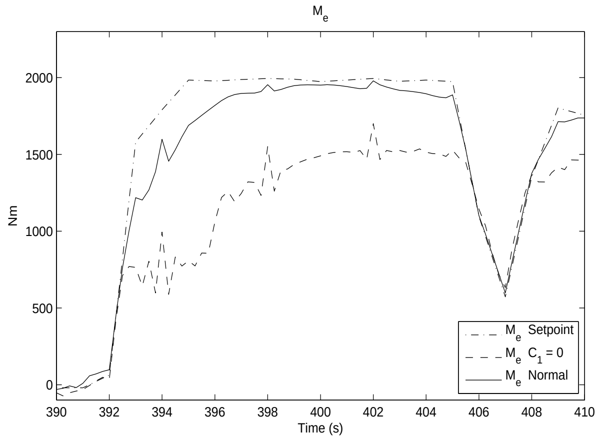

Figure 5.27: Torque when C_1=0, i.e. low weight on u_δ compared with the normal weight.

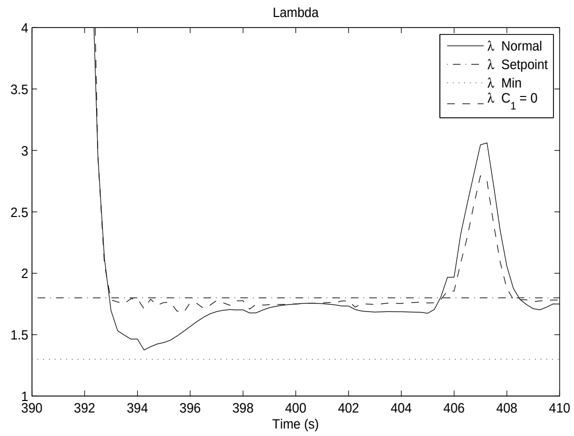

Figure 5.28: λ when C_1=0, i.e. low weight on u_δ compared with the normal weight.

Description	Signal	Change from normal to $C_1 = 0$ %
EGR error	$\|x_{egrSetp} - x_{egr}\|$	-88%
Positive λ error	max($\lambda_{Setp} - \lambda$,0)	-72%
Torque error	$\|M_{eSetp} - M_e\|$	256%
Pumping losses	$p_{em} - p_{im}$	-43%

Low weight on u_δ gives good results for x_{egr}, λ and pump losses, but really bad result for M_e.

5.3 Turbine efficiency

The total turbine efficiency, η_{tm} consists of the turbine efficiency and mechanical efficiency. Measurements show that η_{tm} is dependent of BSR (Blade Speed Ratio) and the turbine speed ω_t according to [1].

$$\eta_{tm} = \eta_{tmmax} - c_m(BSR - BSR_{opt})^2 \tag{5.2}$$

$$BSR = \frac{R_t \omega_t}{\sqrt{2c_{pe}T_{em}(1 - \Pi_t^{1/\gamma_e})}} \tag{5.3}$$

$$c_m = c_{m1}(\omega_t - c_{m2})^{c_{m3}} \tag{5.4}$$

Where η_{tmmax} is the maximal turbine efficiency, BSR_{opt} is the optimal BSR value for maximal turbine efficiency and c_{m1},c_{m2},c_{m3} are parameters for the model for c_m.
To get a view of how important the turbine efficiency is for the optimization, the parameter c_{m1}, which variates the mechanical losses in equation 5.4 can be changed. A high c_{m1} gives a deterioration of the conditions because the losses is assumed to be higher. It is now possible to get an opinion of how important this parameter is for the system. If the efficiency is important for the optimization , the difference between a large and small c_{m1} should be large, i.e. the sensitivity of this parameter is significant.

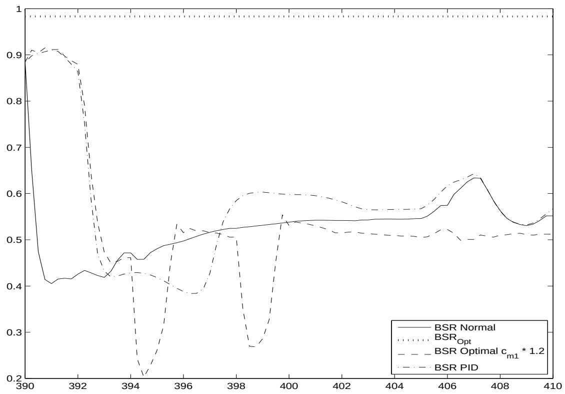

Figure 5.29: BSR for normal and $c_{m1} \cdot 1.2$ compared to BSR for the PID controller.

In figure 5.29, we can see that the difference between the curves is small which implicates that the optimization is not sensitive to a deterioration of the turbine efficiency. But a comparison of the time mean values show a rather big difference.

Description	Signal	Change from normal to $c_{m1} \cdot 1.2$
EGR error	$\lvert x_{egrSetp} - x_{egr} \rvert$	103%
Positive λ error	max($\lambda_{Setp} - \lambda$,0)	147%
Torque error	$\lvert M_{eSetp} - M_e \rvert$	229%
Pumping losses	$p_{em} - p_{im}$	76%

It is clear that it is more difficult for the signals to follow their set point, which can be seen in figure 5.30, 5.31, 5.32 and 5.33.

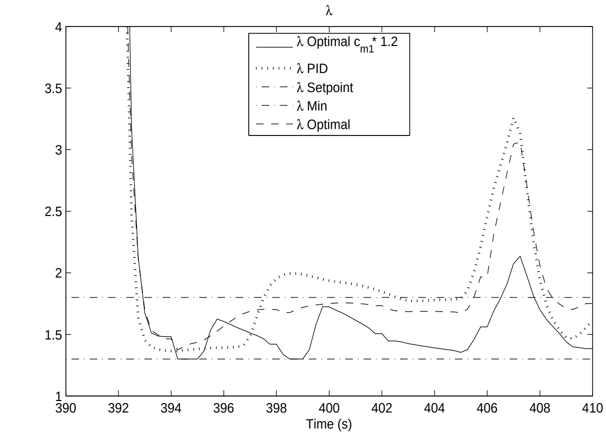

Figure 5.30: λ with $c_{m1} \cdot 1.2$ compared to normal λ and λ for the PID controller.

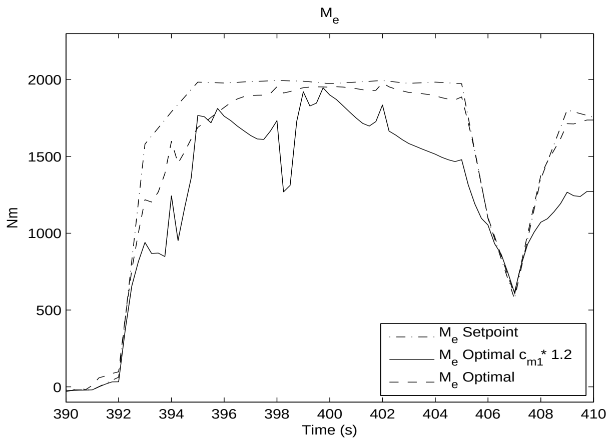

Figure 5.31: M_e with $c_{m1} \cdot 1.2$ compared to normal M_e.

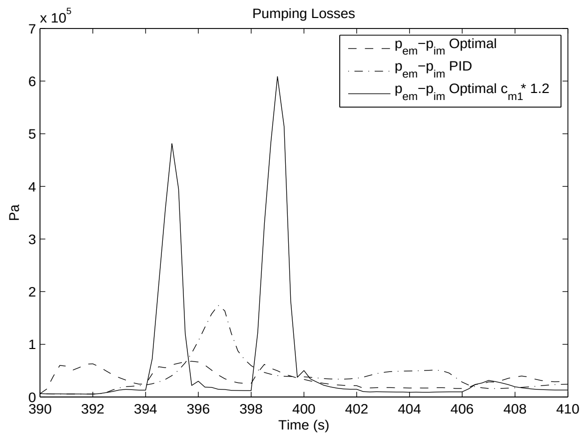

Figure 5.32: Pump losses with $c_{m1} \cdot 1.2$ compared to normal pump losses and pump losses for the PID controller.

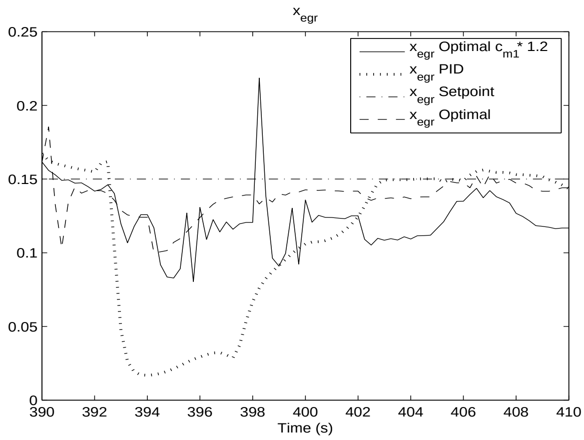

Figure 5.33: x_{egr} with $c_{m1} \cdot 1.2$ compared to normal x_{egr} and x_{egr} for the PID controller.

5.4 Compressor efficiency

In the same way that it is possible to make worse conditions by changing the parameter c_{m1} in the turbine, it is possible to make better conditions by changing the maximal efficiency in the compressor and see if the transient improves. The compressor efficiency,η_c can, according to [1], be described as:

$$\eta_c = \eta_{cmax} - \chi^T Q_c \chi \tag{5.5}$$

χ is a vector that contains the inputs

$$\chi = [W_c - W_{opt}, \pi_c - \pi_{copt}]^T \tag{5.6}$$

and Q_c is a matrix which consists of three parameters.
The transients with a higher maximal efficiency can be seen in figure 5.34, 5.35, 5.36 and 5.37

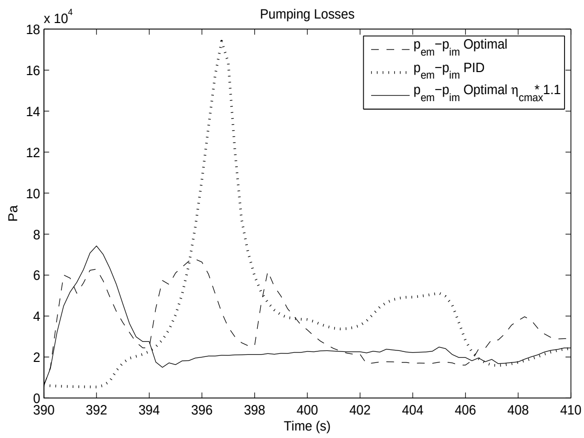

Figure 5.34: Pump losses with a higher maximal efficiency on the compressor compared to the normal efficiency.

And the change in % is

Description	Signal	Change from normal to $\eta_{cmax} \cdot 1.1$
EGR error	$\lvert x_{egrSetp} - x_{egr} \rvert$	-60%
Positive λ error	max($\lambda_{Setp} - \lambda$,0)	-80%
Torque error	$\lvert M_{eSetp} - M_e \rvert$	-26%
Pumping losses	$p_{em} - p_{im}$	-24%

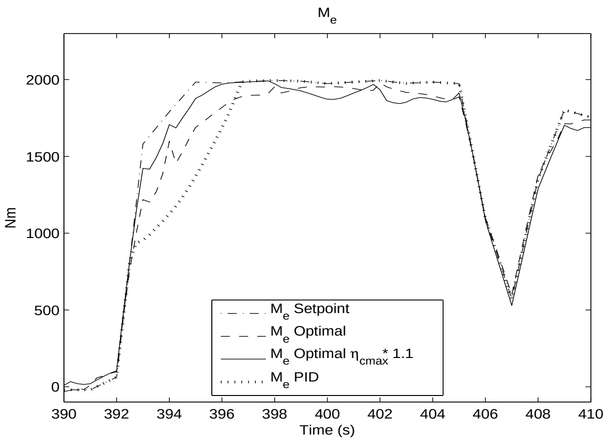

Figure 5.35: M_e with a higher maximal efficiency on the compressor compared to the normal efficiency.

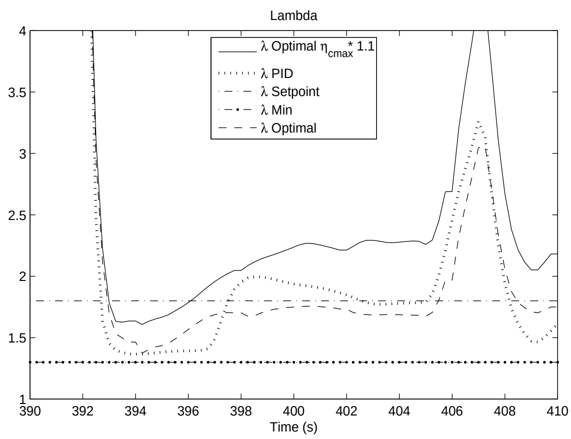

Figure 5.36: λ with a higher maximal efficiency on the compressor compared to the normal efficiency.

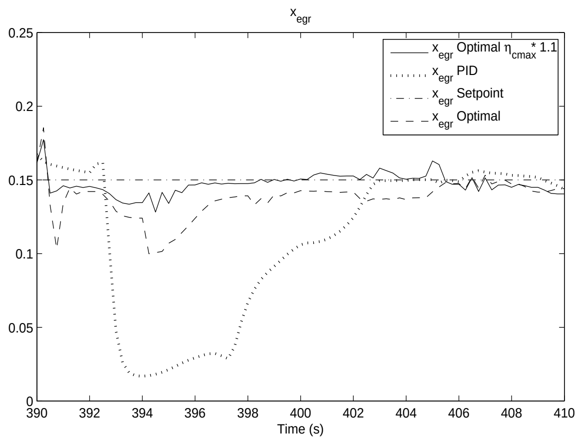

Figure 5.37: x_{egr} with a higher maximal efficiency on the compressor compared to the normal efficiency.

It is obvious that a higher maximal efficiency in the compressor makes it possible for the different signals to follow the set points better.

5.5 Summary

Even if the simulations not where made in the most satisfactory way it is clear that the solution from TOMOC gives a better result,i.e. follows the set points and have lower pump losses, than the PID simulation. The time mean value for the different signals has been calculated and compared to the PID simulation. The most significant improvement can be seen in the x_{egr} which improves with 68% when using the new control signals. This is possible when TOMOC is controlling by closing the VGT and EGR before the transient, in order to build up the pressure. The optimal control signals are plotted in figure 5.3 and 5.4. The time mean values between the different signals and their set point has been calculated and compared to each other. Interesting is also that a rather small change in the efficiency of the compressor and turbine gives a significant change in the result.

Chapter 6

Conclusions

In this chapter an analysis of the results of the different signals and simulations is made.

6.1 Cost function

The assignment has been to find the optimal control signals by minimizing the cost function 3.1. The different parameters have been weighted in different ways and several simulations has been made. The most important parameters in the cost function that have been weighted are:

- The quadratic error between the fuel injection and its set point. When analyzing the results it is better to analyze the error between the torque and its set point instead, see section 5.1.8.
- The quadratic error between the mass fraction in the EGR valve, x_{egr}, and its set point.
- The quadratic error between λ and its set point. Consideration to that it is nothing negative with a large λ has been taken.

6.2 Results

If we study the results in chapter 5, we can come to the conclusion that optimal control with TOMOC gives a better results from than a PID regulator, even though the results are quite similar. The improvement for the time mean values of the different signals can be seen in the tabulars in chapter 5. It is also interesting to see that a rather small change of the parameters c_{m1} and η_{cmax} in section 5.3 and 5.4 gives a significant change in the result. The most significant improvement is that the x_{egr} follows its set point 68% better than the PID simulation, this is because TOMOC can prepare the system for the different transients.

6.3 Control signals

6.3.1 Is this the optimal control?

The optimal control signals that have been calculated in TOMOC can be seen in figure 5.3 and 5.4. But can we be sure that this really are the optimal control way to control the two signals? The answer to that question is no. Since we do not have any possibility to control if the solution is a global or local optimum, we can not exclude that there could be a better way to control the signals. It is also unclear how the use of the initial values have effected the results. We can also see that the big difference between our simulation and the PID simulation is the control of the EGR valve. The VGT signal is almost the same in the two simulations but the EGR shows some interesting differences. It seems to be desirable to open up the EGR valve more and faster in order to lower the exhaust manifold pressure. That leads to a lower turbine speed and a smaller amount of air can be pressed into the engine by the turbo, which leads to a lower pressure in the intake manifold. In that way the pump losses are reduced.

6.4 Simulations

Since there have been problems with long simulation times, and that the simulations crashes if the sample time is to high, the 20 sec long simulations have been divided into five 4 sec long simulations. This should not give so much different result than one 20 sec long simulation. However, in some intervals the control signals have some tops and dips that is a direct result of this.
Why the simulations crash when the sample time is to high, we don't know. Normally a simulation with high sample time would be more accurate than a simulation with low sample time. Probably the simulations do not converge because of some kind of numerical difficulties.

Chapter 7

Future Work

In this chapter further work to this master's thesis is discussed.

7.1 Optimization

The mayor problem in this work has been the long simulation time and it is desirable to short down this time. It would also be desirable to simulate with a better sample time, i.e. simulate with more segments. The accuracy would then be better and the result more precise.
Something that would be interesting is to compare `fmincon` with an other kind of solver. That it is a very important part of the optimization but still difficult to say if `fmincon` is a good choice or not. An other solver may give a different result.
Try to use other methods instead of trapeze method, like Euler which already is implemented in TOMOC, to make the optimization problem discrete.
TOMOC is very sensitive for different initial values. If an observer which could calculate future values was made, these values could then be used as initial values. This would mean that the simulation didn't have to follow the PID simulation at all. There would be a better chance that a global optimum was found.
If TOMLAB is available, it would of course be interesting to see the result when using that.

References

[1] J.Wahlsröm. Modeling of a diesel engine with VGT and EGR. Technical report, Division of Vehicular Systems, Linköpings University, Linköpings Universitet, Linköping, Sweden, 2005.

[2] A. Lagerberg. Open-loop optimal control of a backlash traverse. Technical report, Department of Signals and Systems,Chalmers University of Technologi, School of engineering Jönköping University, 2004.

[3] Lennart Ljung and Torkel Glad. *Reglerteori*. 3 edition, 1997.

[4] Lars Nielsen and Lars Eriksson. *Course material Vehicular System*. 2004.

[5] T.Johansson. System analysis of a diesel engine with VGT and EGR. Master's thesis LiTH-ISY-EX-3714, , Linköpings Universitet, Linköping, Sweden, January 2005.

[6] C. Vigild. *The Internal Combustion Engine Modelling, Estimation and Control Issues*. PhD thesis, Technical University of Denmark, Lyngby, 2001.

[7] Johan Wahlström, Lars Eriksson, Lars Nielsen, and Magnus Pettersson. PID controllers and their tuning for EGR and VGT control in diesel engines. IFAC World Congress, Prague, Czech Republic, 2005.

Notation

Symbols used in the report.

Variables and parameters

states	description	unit
p_{im}	Intake manifold pressure	Pa
p_{em}	Exhaust manifold pressure	Pa
X_{Oim}	Fraction oxygen in intake manifold	$\%$
X_{Oem}	Fraction oxygen in exhaust manifold	$\%$
ω_t	turbine speed	rpm
$\tilde{u}_\delta$	dynamic of the fuel injection actuator	$\%$
$\tilde{u}_{egr}$	dynamic of the egr actuator	$\%$
$\tilde{u}_{vgt}$	dynamic of the vgt actuator	$\%$
n_e	engine speed	rad/min
u_δ	fuel injection actuator	mg/cycle
u_{egr}	egr actuator	$\%$
u_{vgt}	vgt actuator	$\%$
W	Flow	kg/s
T	Temperature	K
R_a	Gas constant for air	$J/(kg \cdot K)$
R_e	Gas constant for exhaust gas	$J/(kg \cdot K)$
V	Volume	m^3
W_{ei}	Total gas flow into the cylinder	kg/s
W_f	Fuel flow	kg/s
O_c	Fraction oxygen in air	$\%$
η	Efficiency	-

Appendix A

TOMOC Manual

A.1 Introduction

This is a introduction to the TOMOC version that was made, by Adam Lagerberg, special for Linköpings university. It is written mainly to give future users a help to get started with the program. We have described how we solved different problems without saying that our method is the only way or the best way to solve the problem.

TOMOC consists of several different m-files which are used together with `fmincon` to solve the optimal control problem.
There are both problem specific and not problem specific files. The problem specific functions,i.e. the functions that must be changed to match the specific problem, are describes in the following sections.

A.2 Tomlkpg1start

The TOMOC template used here is originally made to solve a specific problem so many things have to be changed to make it work for other problems. For the problem in this thesis is not possible to make a setup on matrix form so it has a different setup than the original. This introduction is an effort to explain the changes necessary to get a more complex model to work within the template. `Tomlkpg1start` which is the main script in TOMOC is built around `fmincon` and calls on the other functions. It defines time interval, sample time, simple bounds etc..

A.2.1 Phases

The number of phases is described by `(P.user.nph)` in TOMOC. If the dynamic trajectory of the system differs over time it is possible to divide it into different parts. Each phase will correspond with the specific dynamics and will be treated as a separate systems which are linked together. Also different boundary conditions and constraints can be given for each phase. The advantage with this setup is that it makes it easy to handle different dynamics within the same area problem.

A.2.2 States

The number of states is in TOMOC defined by `(P.user.phph.ny)`. This value is used to decide the size of the vector X and X0.

A.2.3 Control signals

The number of control signals is defined by `(P.user.phph.nu)`.This value is used to decide the size of the vector X and X0. Segments on phase `(P.user.phph.ns)`. This parameter decides how many segments to be calculated with on each phase. Indirectly this will also decide the sample time for the run if the goal is to minimize over a timespan. If the goal is to minimize time this will indicate the accuracy. It is convenient to start with a small value and then iteratively increase it to improve the accuracy of the solution.

A.2.4 Initial values

The initial values are in TOMOC defined by `(P.user.phph.(xi , ui))` and `(P.user.phph.(xf , uf))`. It is important that this value is a good guess if a correct solution will be achieved since the program searches for a minimum around this point. It is not possible to know if this is a global or local minimum. This value is a constraint and will be the fixed value for states and control signal in the first and/or last point of the calculation. It is possible to set (xi , ui) for one or more whole segments in the X vector or pick out specific values in one or more segments. If no points are going to be used set to infinity. This value is used to calculate the X0 vector how are interpolate from start to final, if no final exists the X0 vector will be interpolated from final to final. If nothing is provided zeros are assumed. Initial values are to be set as a transposed vector or matrix.

A.2.5 Time constraints

The time interval for the calculation is set in `P.user.phph.Dtconstr`. The time is a part of the X vector and is a constraint value. Set to infinity if a minimal time calculation is wanted, in this case time will not be used as a constraint. The sample time will be time divided with number of segments. P.user.Dttotconstr is total time for all phases.

A.2.6 Initial guesses

A guess of the initial and final values can be set in `P.user.phph.(xig , uig)` and `P.user.phph.(xfg , ufg)`. It is favorable to use this function if a good guess exists for the whole vector or a part of it. If these values are set they will act as start points for the X vector and they will not be set as constraints. This will reduce the convergence speed since the NPL solution is dependent on a good initial value. If a good guess exists for the whole vector it would be possible to directly replace it with X0 without using TomocInitguess, a constraint initial guess for the first point in the vector would still be necessary since it is used on other places than TomocInitguess. Simple bounds LB and UB restrains the values between two points for each state and control signal. If LB = UB a fixed value is obtained. Bounds can be vectors or scalers, and can be set for every segment. It is e g possible to get the bounds to follow a trajectory by defining a vector and letting LB = UB

X vector

The vector used by the solver is called X and contains time, dynamic states and control signals. The initial X vector is created in `Tomocinitguess` from the initial values defined in `tomlkpgstart`. The values are interpolated from initial to final to fit the number of segments on the phase. The X vector is structured in the following way

$X = [X^{(1)}, X^{(2)}, ..., X^{(n_{ph})}]$

where $X^{(p)}$ is the vector for phase p, $p = 1, 2..., n_{ph}$

$X^{(p)} = [\Delta t, x^1, u^1, x^{2,} u^2, ..., x^{n_s+1}, u^{n_s+1}]$

where

Δt is the time on phase p, $x^k$is the state variable and u^k is the control signal vector in each segment node k,$k = 1, 2, .., n_s + 1$

$x^k = [x_1^k, x_2^k, ..., x_{n_y}^k]$
$u^k = [u_1^k, u_2^k, ..., u_{n_u}^k]$

For example assume:
number of segments = 3
number of phases = 1

number of states = 2
number of control signals = 2

the created vector is linearly interpolated from (x_i, u_i) to (x_f, u_f) so that

$$X^0 = [\Delta t, x_1^1, x_2^1, u_1^1, u_2^1, x_1^2, x_2^2, u_1^2, x_1^3, x_2^3, u_1^3, u_2^3, x_1^4, x_2^4, u_1^4, u_2^4]$$

And if a prior X exists in workspace `Tomocinitguess` will use this as a new starting guess. If the number of segments have been changed since last run a new X0 whit the right length will be interpolated from the prior grid points. This is of importance since the convergence speed of the solution is dependent of a good initial value. It is even possible that no solution at all is found if the initial values are to bad. It also possible to set only a few individual initial values and let the rest be zeros. This is possible by using guessed values. If the (xi, xf) vector is set to infinite, and guessed values provided in (xig, xfg) are used, all values not used in (xi_g, xf_g) will be set to zeros and only the guessed values are used.

Upper-and lower bounds

The states and control signals can be restricted to not exceed or be below a certain value. This will be done for every state and control signal for all segments by controlling the correct element in the X vector.With these bounds it is possible to give a state a specific value during the whole simulation by setting UB=LB.

Structs

The struct P contains data from user defined functions and statemaker.

Optimset

The function optimset creates a structure that pass on a input argument to the fmincon function. This gives the user a possibility to change the conditions for the simulation. For example number of iterations, termination conditions, display options can be set. Also gives the opportunity to choose which numerical method the solver should use.

A.3 tomlkpg1fun

Contains the cost function and all the dynamics that is required to calculate the cost function. The function picks out indexes of control signals and states at each time point. The index correspond to the place in the vector X where the signal is located. This makes it possible to sort out the right signal from

vector X needed to calculate the cost function. The result is a vector corresponding to chosen signal whit length equal to number of segments. If other variables is needed to calculate the cost function it is necessary to estimate them here by using the generated vectors.
Example:
The needed signal is control signal 1 `(u1)` then index u1 is
`iu1=iDt+P.user.ph(ph).ny+1+P.user.ph(ph).nyu`
`*(0:P.user.ph(ph).ns)`
where `iDt` is time which is placed first in the vector for each phase.

`P.user.ph(ph).ny` is number of states on each phase. (1)
`P.user.ph(ph).nyu` is total number of signals on each phase. (2)
`P.user.ph(ph).ns` is number of segments on each phase. (3)
if (1) = 4 , (2) = 2 , (3) = 2 then the result is a vector which contains index for u1 iu1 = [6 12 18] then it is straight forward to pick out the signal from the X vector.

A.4 tomlkpg1dynfun

`tomlkpg1dynfun` contains the dynamics to be used in optimization. If the system is linear and written on the form $\dot{x} = Ax + Bu$ it is possible to implement it on matrix form (see [2]). If it is a non linear problem , which can't be written on matrix form, a separate function which contains the dynamics can be written to solve the problem. In this case dynfun should work as a link between Tomoc and the dynamics. Every time Tomoc calls upon dynfun the information should be passed on to the dynamics function and the answer return the same way. This is very straightforward to implement, just exchange the existing dynamics in template with the new dynamics function so that instead of `f = Ax+Bu` you get `f = dynamicsfunction(x,u).`

It can be a good idea to weight the dynamics so that the values outside the dynamics function are approximately the same size. This is done to prevent that big units has more influence than small units in the handling of constraints. The benefits is a greater convergence speed. The easiest way to do this is to multiply the ingoing values to the dynamics function whit a diagonal matrix containing numbers in the same range as states and control signals. Then multiply outgoing values with the inverse of the same matrix.

Example
The values that are going in and coming out from the function should be approximately the same size, but the values used by the dynamics should have the proper value. Hence the values of inputs to the dynamics should be multi-

plied with a vector (x_0, u_0) and values of the outputs from the dynamics must be divided with x0. An easy way to do this is as following

s = diag(x_0)
s^{-1} = s-1
v = $u_0 \cdot u$
$\dot{x}$ = $s^{-1}\cdot$ dynamics$(s \cdot x, v)$

Since the dynamics function are called upon many times during a simulation it is computably necessary to set the used constants as global, so that they are available for calculations at all times without the need to load them at each call to the function. One way to do this is to run a initialization file were the needed constants are assigned a global value. Any assignment to that variable, in any function, then is available to all the other functions declaring it as global.

A.5 tomlkpg1initpar

Initialization of problem specific parameter values. Since this file uses the states on matrix form, it should be replaced/changed if the problem is non linear.

A.6 tomlkpg1constraints

The file `Tomlkpg1constraints` makes it possible to set problem specific (non)linear constraints.

The problem independent functions are described in the following sections.

A.7 tomocinitguess

The file `Tomocinitguess` constructs an initial guess by linear interpolation between parameters set in start (see X). Even thou the file is said to be problem independent it has to be changed to fit the problem specifics. The template is made to work with two states and one control signal, if this differs it has to be changed. Guesses for u should be rewritten in the same form as for states and the linear interpolation part has to be expanded to fit the new problem.

A.8 tomocconph

Multi-phase constraint definition. This function calls tomoccon (see A.9) once for each phase, and then assembles all constraints into one vector. The state and control variables on each phase are assumed to be the same, and to be continuous over the phase boundaries. TOMLAB uses one vector both for equality and inequality constraints. Special constrains equations are used for this.

A.9 tomoccon

This function is called by `tomocconph` once for each phase and calculates constraints for the boundary values, initial and final time. The function calculate the constraints vector ceq. It is calculated in different parts, some in loops and some in sequence after each other. Every calculation part uses the temporary variable ceq_k. After that the vector is put together gradually with tomocconph.

A.10 tomocpostproc

This function handles the result vector X and reshapes it to a matrix , xr, containing states, control signals and time. This makes it easy to pick out the wanted variables for use in plot functions. The result is also used as initial guess if the workspace is not cleared for a second run.