

FMH606 Master's Thesis 2021

Industrial IT and Automation

Advanced model-based control of B36:45 LNG engines based on data driven models using machine learning tools

Svein Roar Kvåle

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

Course: FMH606 Master's Thesis, 2021

Title: Advanced model-based control of B36:45 LNG engines based on data driven models using machine learning tools

Number of pages: 212

Keywords: MPC, Lagrange, DMPC, Fmincon, LNG, qpOASES

Student: Svein Roar Kvåle

Supervisor: Associate Professor Roshan Sharma

External partner: Bergen Engines AS

Summary:

This work is about developing a data driven MPC control for optimization of fuel consumption by minimizing the heat rate of an LNG gas engine from Bergen Engines AS.

The model is based on real life data from an installed B36:45 gas engine in a power plant. The process data from the engine was used to develop a state space model of the process consisting of 2 controllable inputs, 3 measure disturbances and 6 measured outputs.

The goal was to use the global ignition timing and the charge air pressure set point as controllable outputs to minimize the heat rate while considering constraints on the measured outputs.

Several MPC concepts has been tested, including qpOASES; Quadprog, Fmincon and DMPC with Laguerre functions; all of which has their pros and cons, and which produced different results.

Mostly the Fmincon and the DMPC gave the most promising results, DMPC with speed and ease of implantation but lacked successful results on output constraints. Fmincon produced some usable results but often got into trouble handling the output constraints and was computational heavy to use.

The University of South-Eastern Norway takes no responsibility for the results and conclusions in this student report.

Preface

This master's thesis has been written for the completion of the Master's degree in Industrial Automation and Communication at the University of South-East Norway (USN) Porsgrunn, Norway.

My work in this project presents Model Predictive Control for an optimization control set points for charge air pressure and global ignition timing for an LNG in order to minimize heat rate. MATLAB was mainly used for performing simulations using m-files scripts.

The work was initiated and supported by Bergen Engines AS and the Research and Development department there. The work builds on the data driven models developed in the Master project during the fall of 2020.

I would especially like to thank my supervisor, Associate Professor Roshan Sharma for his guidance and supervision during the course of this project. His support, guidance and encouragement has been vital for the research and testing done during this project.

I would also like to thank Bergen Engines for giving me the opportunity to do this project and use valuable time to work on it. It has been a challenging year in many ways and time spent on this project has impacted on other deliveries. I'm grateful for the support from my employer and the trust that the time spent is valuable for the future development of Bergen Engines control systems.

Lastly, and not least, I would like to thank my children, girlfriend and family for the support throughout these 4 years of study. This project and master's study would not have been possible without them and the encouragement during the difficult times has been paramount. We'll make up for the lost weekends and vacations in the upcoming years.

Bergen, 18.05.2021

Svein Roar Kvåle

Contents

Preface	3
Contents.....	4
Abbreviations	6
Symbols	7
1 Introduction	8
1.1 Main objective	8
1.2 Background	8
1.3 Operational philosophy.....	13
1.3.1 <i>Lean burn gas engine - Otto cycle</i>	13
1.3.2 <i>Main control loops</i>	15
1.3.3 <i>Engine control system structure</i>	25
1.3.4 <i>Main engine controller</i>	25
2 Model predictive control.....	28
2.1 Brief history	28
2.2 Main objective	29
2.3 Cost function.....	31
2.4 Receding horizon.....	32
2.5 Constraints	33
2.6 Grouping.....	34
2.7 System model.....	35
2.8 Integral action	36
2.9 Previous work	37
2.10 Set up.....	37
3 Modelling	38
3.1 Data collection	38
3.1.1 <i>Sampling</i>	38
3.1.2 <i>Pre-processing and analysis</i>	39
3.2 System identification.....	40
3.2.1 <i>Controllability</i>	42
3.2.2 <i>Limitations</i>	43
3.2.3 <i>Open loop simulation</i>	43
4 MPC	63
4.1 Controllable inputs	63
4.1.1 <i>Charge air pressure</i>	63
4.1.2 <i>Global ignition timing</i>	64
4.2 Measured disturbances.....	65
4.2.1 <i>Suction air temperature</i>	65
4.2.2 <i>Charge air temperature</i>	65
4.2.3 <i>IMEP</i>	65
4.3 Measured outputs	66
4.3.1 <i>Heat rate</i>	66
4.3.2 <i>Knock level</i>	66
4.3.3 <i>Peak pressure</i>	66
4.3.4 <i>NOx</i>	67

4.3.5 O ₂	67
4.3.6 Exhaust temperature	67
4.4 Constraints	69
4.5 qpOASES	70
4.5.1 Set point tracking with bounds on control value.....	70
4.5.2 Code for qpOASES	74
4.5.3 Status of qpOASES	75
4.6 Quadprog.....	76
4.6.1 Set point tracking with bounds on control value.....	77
4.6.2 Code for Quadprog	79
4.6.3 Status of Quadprog.....	79
4.7 Fmincon	80
4.7.1 Set point tracking – unconstrained.....	80
4.7.2 Set point tracking with constraints on Δu_m	93
4.7.3 Set point tracking with constraint on outputs	101
4.7.4 Minimize heat rate with constraints	116
4.7.5 Grouping of control inputs.....	129
4.7.6 Prediction horizon.....	131
4.8 Laguerre based Discrete Model Predictive Control	134
4.8.1 Classical DMPC	134
4.8.2 DMPC with Laguerre functions.....	141
5 Discussion	163
5.1 Modelling	163
5.2 qpOASES and Quadprog	163
5.3 Fmincon	164
5.4 DMPC with Laguerre functions	165
5.5 Future work	165
6 Conclusion	167
References.....	168
Appendices.....	170
Appendix A.....	171
Appendix B.....	173
Appendix C.....	176
Appendix D.....	177
Appendix E	179
Appendix F	181
Appendix G.....	184
Appendix H.....	191
Appendix I	201

Abbreviations

<i>degCA:</i>	Degree crank angle
<i>TDC:</i>	Top dead centre
<i>PID:</i>	Proportional Integral Differential
<i>DMC:</i>	Dynamic matrix control
<i>GPC:</i>	Generalized predictive control
<i>LQR:</i>	Linear quadratic regulators
<i>QP:</i>	Quadratic programming
<i>LQ:</i>	Linear Quadratic
<i>MATLAB:</i>	Matrix Laboratory
<i>MIMO:</i>	Multiple Input and Multiple Output
<i>SISO:</i>	Single Input and Single Output
<i>CV:</i>	Control value
<i>MPC:</i>	Model Predictive Control
<i>LNG:</i>	Liquefied Natural Gas
<i>CPM:</i>	Cylinder Pressure Monitoring
<i>CFD:</i>	Computational fluid dynamics
<i>ppm:</i>	Parts per million

Symbols

A :	State matrix in state space model
B :	Input to state matrix in state space model
C :	State to output matrix in state space model
N_p :	Prediction horizon
N_c :	Control horizon
ΔU :	Vector of control input signals
$\Delta u(k_i)$:	incremental control signal at current time instance k_i
$\Delta u(k_i + m)$:	future incremental control signal at time instance m
Δu^{max} :	Max limit on control signal
Δu^{min} :	Minimum limit on control signal
ΔY :	Vector of predicted output signals
y :	Output signal
Δy^{max} :	Max limit on output signal
Δy^{min} :	Minimum limit on output signal
$x(\cdot)$:	State variable
$x(k_i + m k_i)$:	Predicted state at sample time m given current state $x(k_i)$
J :	Optimization variable in cost function
Q :	Weight matrices in cost function
R :	Weight matrices in cost function
Ω, Ψ :	Matrices in cost function - $J = \eta^T \Omega \eta + 2\eta^T \Psi x(\cdot)$
a :	Scaling factor for discrete time Laguerre functions
l_i :	The i th discrete Laguerre function
$L(\cdot)$:	Laguerre function as vector
λ :	Lagrange multiplier
η :	Laguerre parameter vector
M, γ :	Matrix and vector in inequality constraints ($Mx \leq \gamma$)

1 Introduction

This section contains an introduction to the problem domain and background information for the objective of the project.

1.1 Main objective

This project is a continuation from the Master project FM4017 – “Data driven modelling using Machine Learning tools for control and analysis of charge air pressure and global ignition timing control on B36:45 LNG engines” from the fall 2020. In the FM4017 project a data driven model of the combustion process was developed which will form the basis for the a MPC driven optimizer developed in this project.

The purpose of this project is to develop a controller which will, through optimization, find optimal set points to give improved fuel efficiency of an LNG single fuel internal combustion engine. This optimizer will mainly be used for land based power plants where fuel economy is of utmost importance.

Based on measured input/output data the optimizer will adjust two controllable inputs such that the fuel consumption is optimized while keeping other engine conditions within given constraints.

The controllable inputs are the general engine global ignition timing point and the charge air (boost air) pressure set points. Lower level standard PID controllers will use these inputs as set point for local control.

The findings in the Master project showed that a data driven model with 2 controllable inputs, 3 measured disturbances and 6 measured outputs would represent the system to be controlled.

1.2 Background

Bergen Engines AS is a developer and producer of gas and diesel engines for the marine and land-based power market. The factory is located just north of Bergen on the west coast of Norway and have been since it moved from the city centre of Bergen in 1965. Bergen Engines was part to the Ulstein group from mid-eighties until 1999 from which it has been a part of Rolls-Royce. Figure 1-1 shows an illustration of the factory area.

Currently there are approx. 1100 employees in Bergen Engines and about 650 of these are situated in the factory outside Bergen.



Figure 1-1: Factory illustration

The latest LNG fuel engine type developed is the B36:45 engine family, and a graphical representation of the engine is shown in Figure 1-2. This engine type has been in commercial operation since late 2018 and now forms the most sales for the land-based power generation market. During the development of this engine type the overall engine control system has been re-designed and new embedded controllers has replaced the traditional PLC's. In addition, the instrumentation and diagnostic capabilities has increased due to the feasibility to do in-cylinder pressure monitoring for each cylinder on cycle-to-cycle basis. This gives large amount for valuable data back to the control system to utilize for optimization.

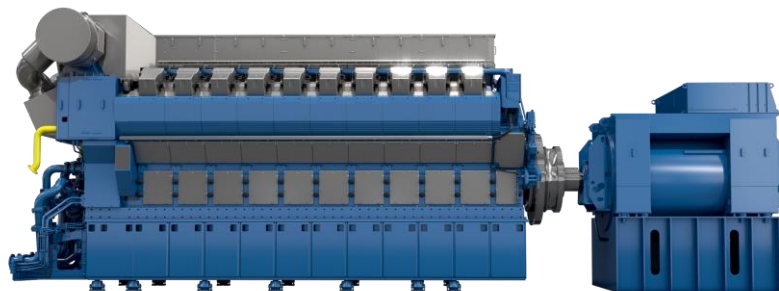


Figure 1-2: 20 cylinder B36:45 LNG gas engine with generator

The B36:45 engine is often sold for single or multi engine installation for land-based power plants. An example of such an installation is shown in Figure 1-3. For most of these installations the engines run at steady conditions for long periods of time and often at maximum, or close to maximum, power output. For these engines the efficiency and power output are what is sold on the electrical grid. A slight increase in

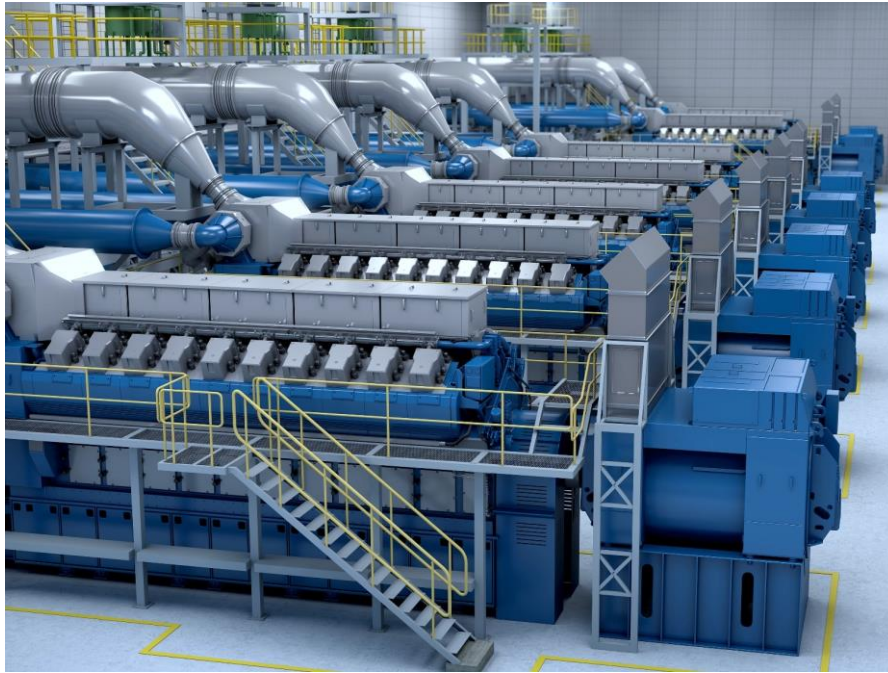


Figure 1-3: Multi engine installation power plant

efficiency will hence generate a large amount of income increase for the owner. The engines are therefore tuned to run as close as possible to its limits, but within the design margins.

Figure 1-4 shows how the energy input to a gas engine is distributed. A lot the energy can be lost to the exhaust and the heat emission via the cooling fluids and lubrication fluid. The exhaust gas temperature is often reused by extracting the heat via boiler systems or heat exchangers and then used to drive other power trains. For greenhouses the exhaust if often cleaned and then the CO₂ is injected into the greenhouse as nutrition to the plants. The cooling fluids are also used to heat water which is used to heat up the greenhouse.

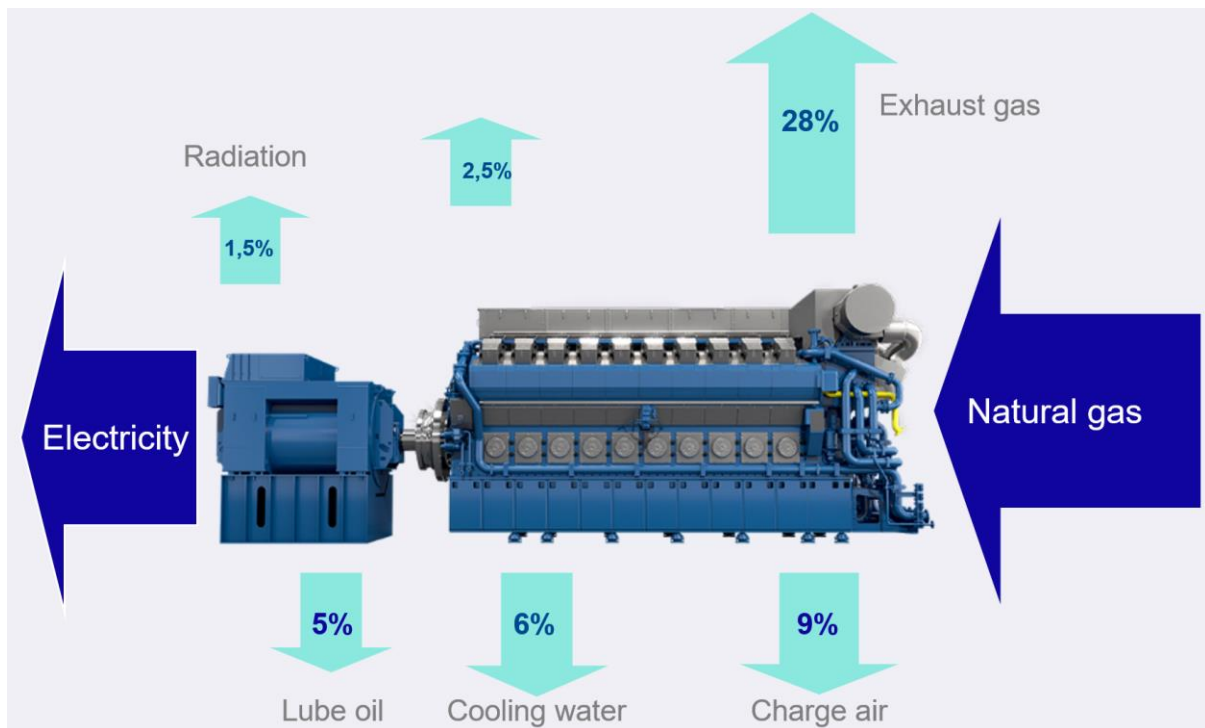


Figure 1-4: Gas engine energy balance

These “optimal” operational set points are set during the commissioning phase such that the engine can operate within specification, but with some safety margin to the design limitations of the engine. These set points are mostly static maps which needs to be recalibrated at given intervals to remain optimal as the engine condition changes both by time and ambient conditions.

For an LNG gas engine the engine efficiency is in general given as in Figure 1-5 for a given power output. We note that the efficiency is not very good below 20 [%] and rises slowly from approx. 30 [%] power output to 100 [%] at which the efficiency is close to 50 [%]. The distribution of the efficiency losses can be seen in Figure 1-6. As shown in Figure 1-4, most of the losses are to the exhaust gas [1].

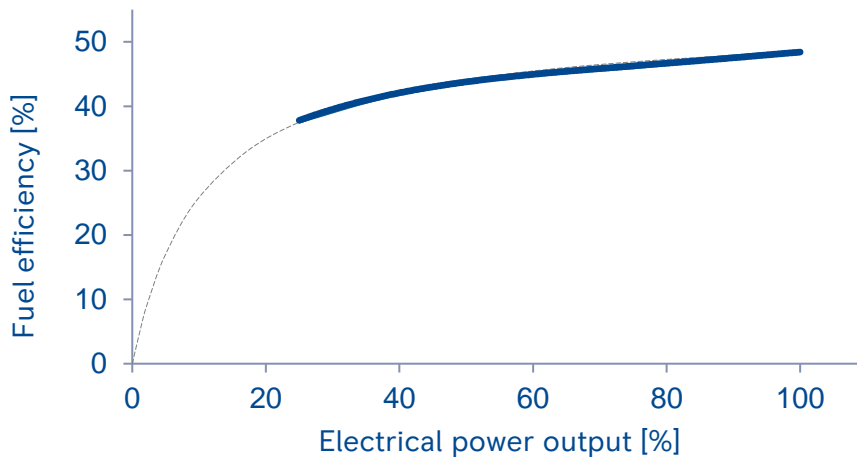


Figure 1-5: Fuel efficiency for and LNG engine

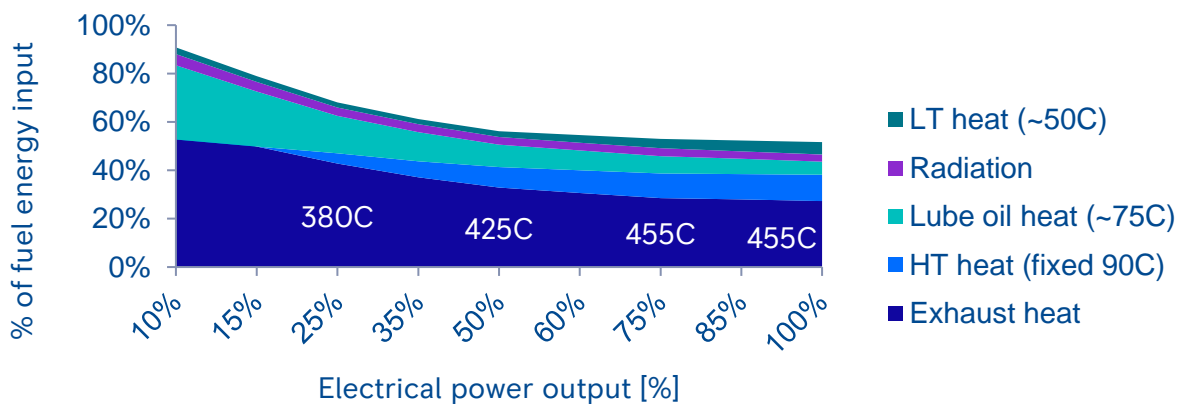


Figure 1-6: Distribution of losses for an LNG gas engine

In the search for higher efficiency, more complex logic, which takes into account more of the information available, is constantly developed. This has resulted in a large increase in parameters and maps that interact with each other, which makes the engine tuning phase a complex and time-consuming job. This project will therefore be used to let a MPC search for the optimal set points based on measured states of the engine and known disturbances.

This is set to be the first step towards more data driven, self-optimizing algorithms that can use the large amount of data produced. The ultimate goal here is to use a reduced set of parameters which can be used to prioritize the different possibilities such that a project optimal goal is reached. For instance on a given project the most important goal is to reach a given NOx set point and secondary fuel efficiency, while it for another project might be the fuel economy which is the most important while keeping the NOx within given constraints.

1.3 Operational philosophy

The B36:45 engine family is a medium speed lean-burn single fuel spark ignited internal combustion engine. It mainly uses LNG as fuel source and runs at 720/750 [rpm] for 60- and 50 [Hz] applications respectively.

It is turbocharged and has a 2-stage water cooled charge air cooler.

In most land-based power plants it is connected to a generator which is often connected to either a small local grid or a large national grid. The engine's nominal power output is 600 [kW] per cylinder mechanically and comes in both in inline and Vee configurations. The smallest is an inline 6-cylinder engine and the largest is a Vee 20-cylinder engine.

1.3.1 Lean burn gas engine - Otto cycle

A lean-burn gas engine runs with a high air to fuel ratio compared to the required air for a stoichiometric combustion. This lowers the combustion temperature and hence reduces NO_x emissions. The B36:45 engine is a lambda 2 engine, indicating that it runs with twice the required amount of air for a stoichiometric combustion. This lean mixture is difficult to ignite and hence a pre-combustion chamber is mounted in the cylinder head. A rich mixture is here ignited by a spark plug and the resulting flames will propagate out and into the main chamber where it will ignite the lean mixture. Figure 1-7 shows an illustration of the combustion chamber with the spark plug, pre-chamber and main chamber.

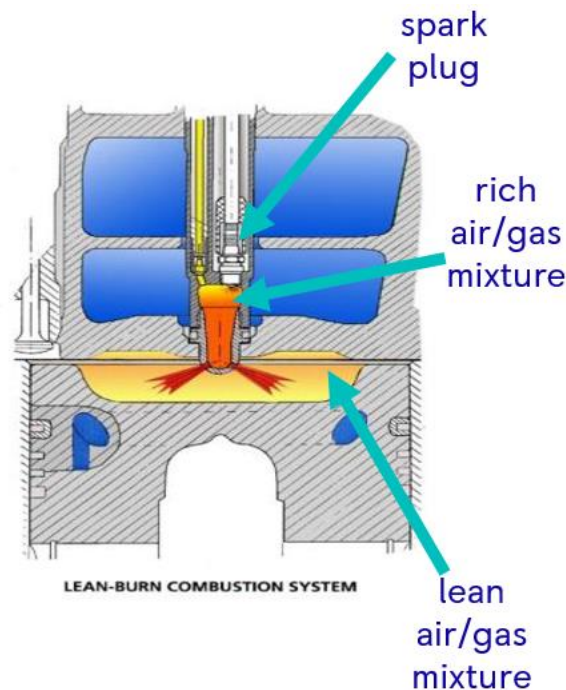


Figure 1-7: Combustion chamber illustration

The engine is a 4-stroke (also known as the Otto cycle) which means that there are 4 distinct phases for the combustion process. The different phases can be seen in the illustration in Figure 1-8.

During the first phase the air/fuel mixture is drawn into the combustion chamber during the downward movement of the piston in the cylinder. This is the phase where the piston moves from the top dead centre (TDC) to the bottom dead centre (BDC).

The second phase is the compression stroke. This is the upward motion of the piston at which the air/fuel mixture is compressed. Here the piston moves from the bottom dead centre to the top dead centre. During the last phase of the compression a spark plug is ignited in order to ignite the air/fuel mixture.

The third phase is the expansion phase, also known as the power phase. After the ignition of the air/fuel mixture the temperature rises rapidly and hence the pressure. This pushes the piston downwards. This forced movement is used to rotate the crankshaft driving the generator.

Once the piston reaches the bottom the final phase starts. This phase is the exhaust phase at which the exhaust from the combustion is pushed out of the combustion chamber as the piston moves upward to the top dead centre again. Once back at the top the cycle starts over again with the first phase.

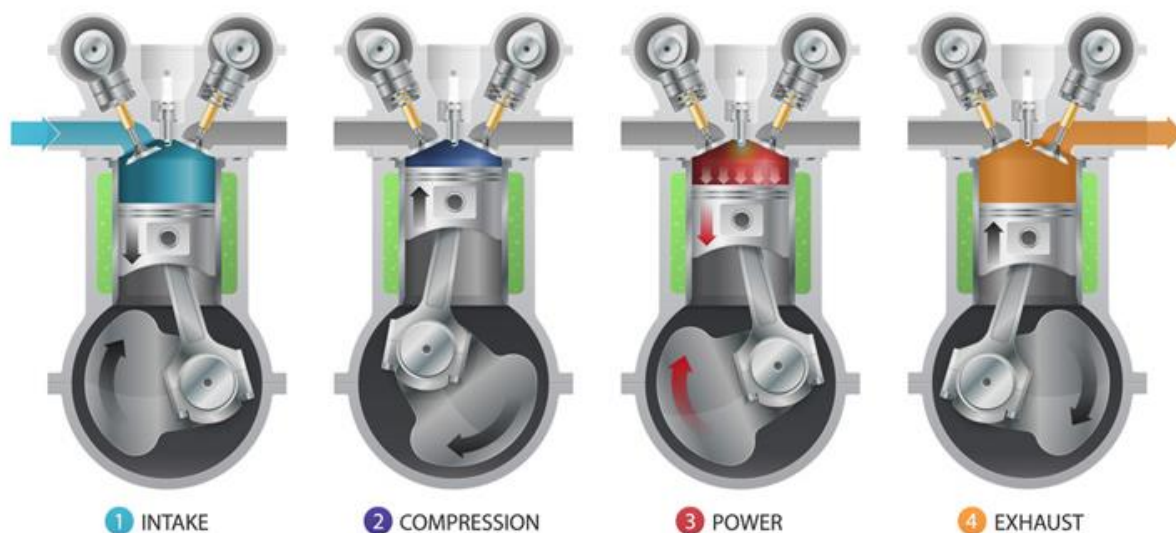


Figure 1-8: Illustration of the 4-phases of the combustion process – illustration from <https://cartreatments.com/internal-combustion-engine-fundamental/>

1.3.2 Main control loops

The following sections shortly describes the most common control loops for the combustion process which are controlled by the engine control system today. There some other control loops as well, but the major once are described here.

1.3.2.1 Speed control

The base function in the engine controller is the speed control loop which is a PID control loop with measured speed as feedback. This engine controller is often called the speed governor as for traditional marine applications with diesel engines its main purpose is to maintain engine speed at a variable set point. But for engine used in power generation the speed setpoint is usually fixed and hence the controller is more a power controller.

The PID controller controls the flow of fuel admission in order to keep the engine speed at set point. When connected to a large grid with fixed frequency the speed control loop is used to control engine power output according to a set point. Increasing the speed set point will result in the speed controller to increase the fuel admission by increasing the fuel flow control which will result in an increase of engine power output as the frequency cannot change.

The output from the speed control PID controller is a control signal to the main fuel actuator on the engine. The fuel actuator transforms the control signal in a mechanical control movement which is used to adjust the fuel flow control valves; one for each cylinder. The controlled flow of fuel is then led to the fuel admission valve which is mechanically linked to the inlet valves. The inlet valves are mechanically linked to the camshaft which controls the valve opening timing. Once in the correct cycle the inlet valve will open and hence the fuel admission valve will open. During the period the inlet valve is open the fuel is led into the air flow and mixed through a mixing port. The inlet channel design allows for a given swirl which will further dilute the air with the fuel which will lead to a close to perfect mixing in the combustion chamber.

Some of the fuel gas is internally in the cylinder head guided to the prechamber valve and, once open, fed into the pre-chamber where the spark plug is located.

The operational principle is simple and can for a diesel engine be controlled by a pure mechanical governor. For a gas engine this is not the case as the fuel is both flow controlled as well as pressure controlled; but the theory is the same. In case the measured engine speed (or power output for the power generating engines) the fuel flow control valve is opened, and hence more fuel is admitted into the combustion process allowing for an increase in speed (or power output). If above the set point the fuel flow is reduced.

1.3.2.2 Air pressure control / AFR control

The AFR, or air pressure control, is a control loop which main purpose if to control the charge air pressure in the air receiver to a given set point. The set point is based on a map with engine power output and engine speed as inputs. The set point map is derived based on numerous of test runs at the test bed by skilled engineers. Since this is a static map the set point must be biased to a certain degree based on operational conditions. The most notable is the NO_x control which is an outer control loop in its own right which will adjust the set point for air pressure control in order to get the NO_x emissions to a given set point. Figure 1-9 shows an illustration of the control loop structure.

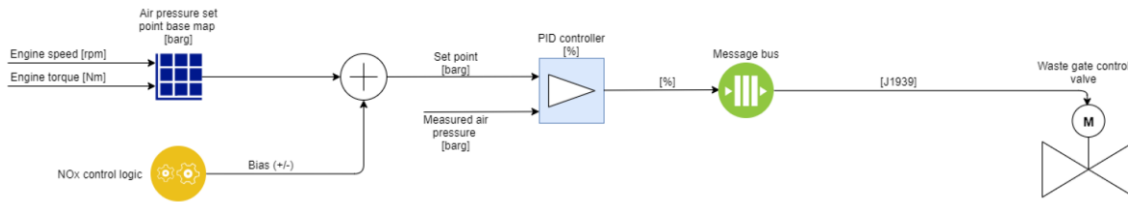


Figure 1-9: Charge air pressure control loop illustration

In Figure 1-10 an indication of how the set point currently is set is shown. It indicates the values in the map structure and shows some rate limitations on the set point changes and the biases to the set point. The output from this structure is the set point to the PID controller.

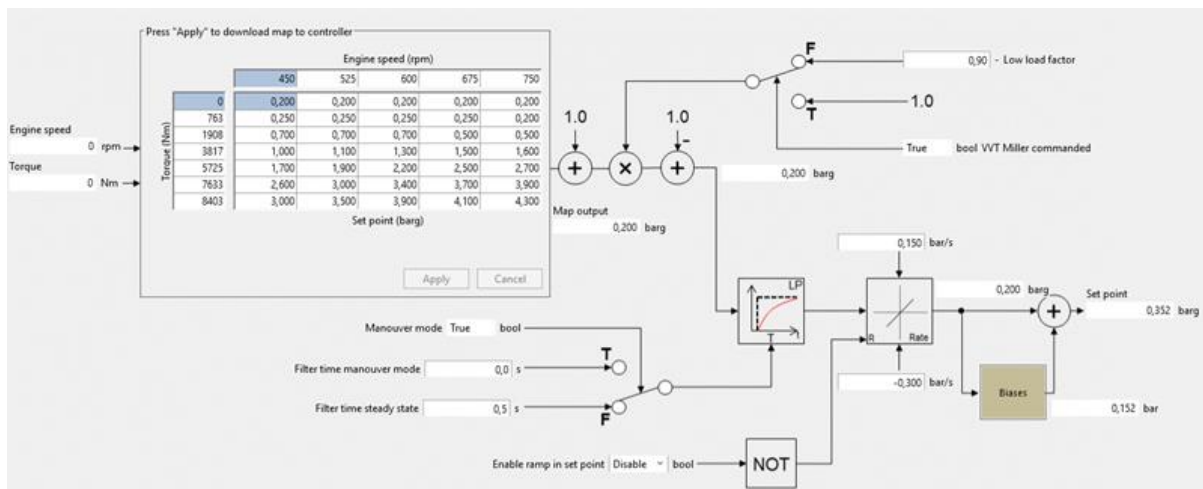


Figure 1-10: Air pressure control – current set point design

The air pressure control is the most active and influents of all control loops. It dictates most of the engine behaviour as it directly controls the air/fuel ratio under all operational conditions.

The output from the air pressure control is a position control signal to a waste gate actuator. The waste gate actuator will control the amount of exhaust by-passing the turbine part of the turbocharger(s) and hence the energy used to increase the air pressure.

The feedback to the air pressure control is the measured air pressure in the air receiver which then forms a closed loop control system.

One of the goals of this project is to find the optimum charge air pressure to maximize engine efficiency.

1.3.2.3 Air temperature control

In order to further control the air/fuel ratio the temperature of the combustion air should be kept at given values. A lower air temperature will increase the density of the air, and hence larger amounts of air and fuel can be added to the combustion process, increasing the power output from the engine. But the air temperature must not be so cold that water is condensed which can be potential damaging to the engine. The air temperature is often controlled by a

low level PID controller which will control the air temperature to a given set point based on operational ambient conditions.

The control signal from the PID controller is used to control a 3-way valve which will direct, or bypass, water to a 2-stage charge air cooler. By increasing the amount of water going through cooler the temperature can be reduced, and vice-versa.

The air temperature after the turbocharger compressor is too high for the combustion process and hence the charge air cooler is located between the outlet from turbocharger compressor and the engine air receiver. The air receiver is a common volume from where the inlet channels to each cylinder originate.

In Figure 1-11 the charge air cooler (1), the air bends (2,3) between the compressor outlet of the turbochargers (4,5) are shown graphically for a Vee engine.

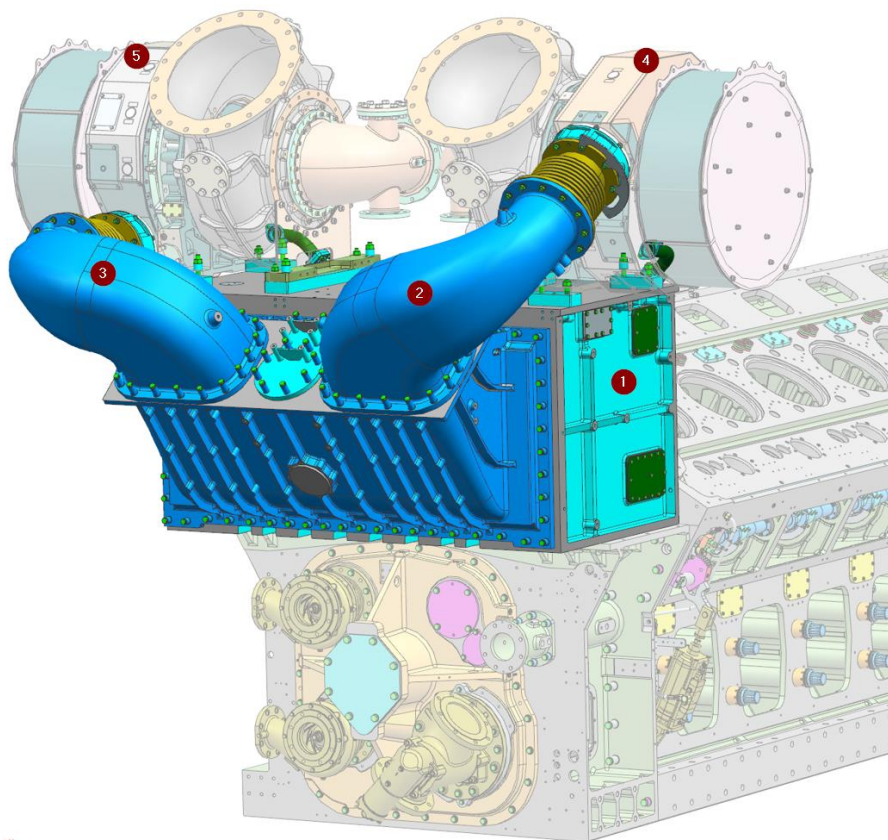


Figure 1-11: Graphical representation of upper front end of a Vee engine

An increase in temperature will lead to an increase in NO_x due to lower air mass added to the combustion process and due to increase in temperature. This will also result in the engine operating closed to the ignition knocking limit.

1.3.2.4 NO_x control

The engine is equipped with dual NO_x sensors in the exhaust outlets. These sensors measure both the amount of NO_x in the exhaust [ppm] and the amount of Oxygen [%]. The NO_x level is fed back as process value to an outer PID controller which will control the NO_x level such

that it is according to a given set point. Figure 1-12 shows a simplified illustration of NOx control structure and Figure 1-13 shows how the implementation looks like today.

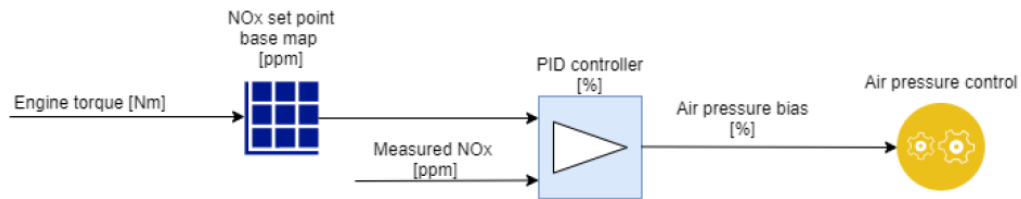


Figure 1-12: NOx control loop

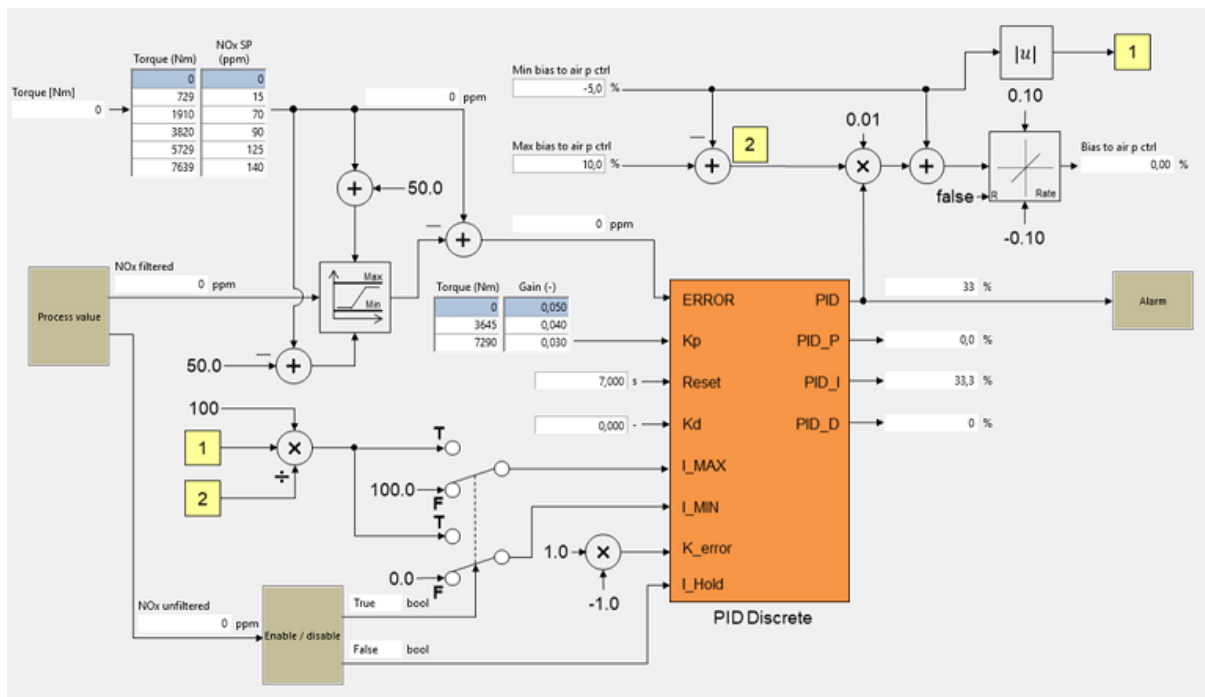


Figure 1-13: NOx control loop implementation

The engines have traditionally been sold to run at TA Luft¹ or ½ TA Luft which is equivalent to 500 [mg/Nm³] NOx at 5 [%] O₂ or 250 [mg/Nm³] NOx at 5 [%] O₂.

The NOx control is a process which is not acting very fast and is only active during steady state operations. The set point for the NOx control is given in ppm and hence the limit is calculated from the dry 500/250 [mg/Nm³] value back to the approx. 143/108 [ppm] set point at nominal power output.

The NOx controller will bias the air pressure base set point between +10 [%] and -5 [%] in order to minimize the error between the measured NOx and the set point.

In Figure 1-14 we see the operational principle of the LNG gas engine and how the various aspects of the combustion process influences each other. The main take from this is how the

¹ TA Luft – is a term defined by the german air pollution control regulation titled “Technical Instructions on air Quality Control” (*Technische Anleitung zur Reinhaltung der Luft*) [11]

air excess ratio operational window narrows as the power output increases and the risk of either knocking or misfire increases. At the same time the increase in efficiency while still keeping NOx within limits and the UHC low enough is important.

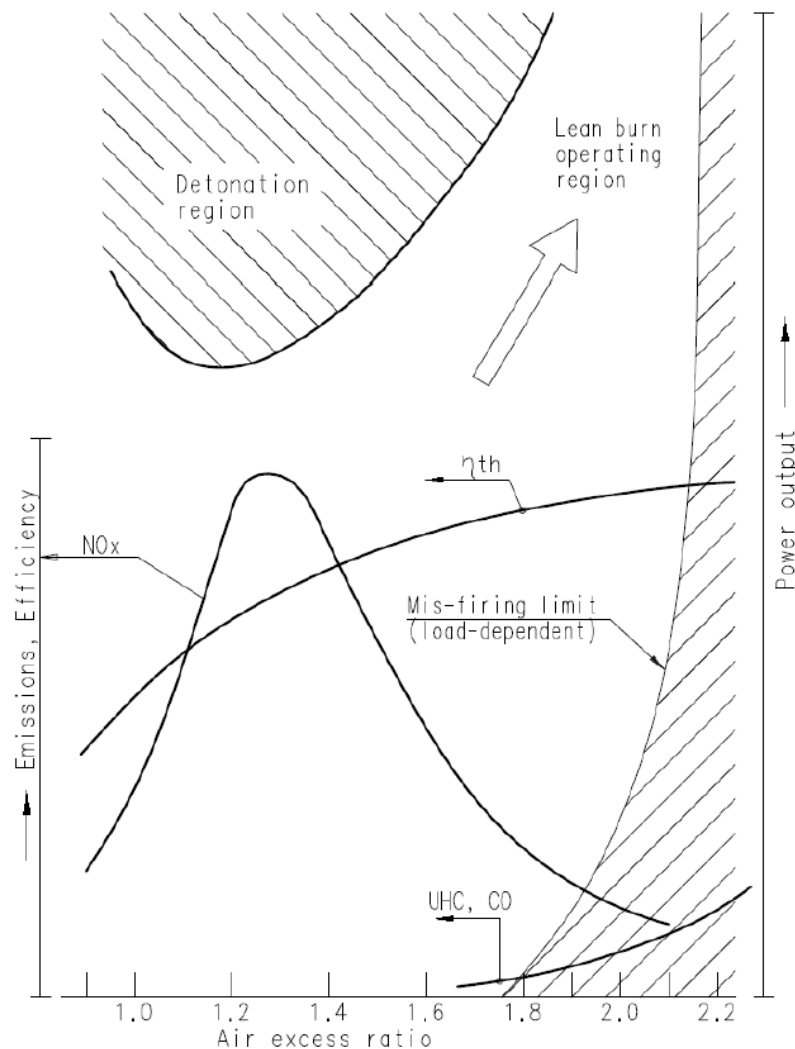


Figure 1-14: Engine performance trends as function of air excess ratio

1.3.2.5 Global ignition timing control

The ignition timing is the time in crank angle degrees at which the cylinder individual spark plug is ignited in the pre combustion chamber. This location is firstly set to a base set point for all cylinders. Then each individual cylinder can adjust its own ignition timing between ± 3 [degCA] in order to balance the power output and peak pressure for each cylinder.

But firstly, the base timing is set in a map based on testing on the testbed by engineers. This base timing is adjusted such that a good margin to ignition knocking while maintaining high level of efficiency is achieved. The global timing is adjusted so that the maximum pressure in the cylinder is occurring around 13-15 [degCA] after TDC. This will give rise to best performance.

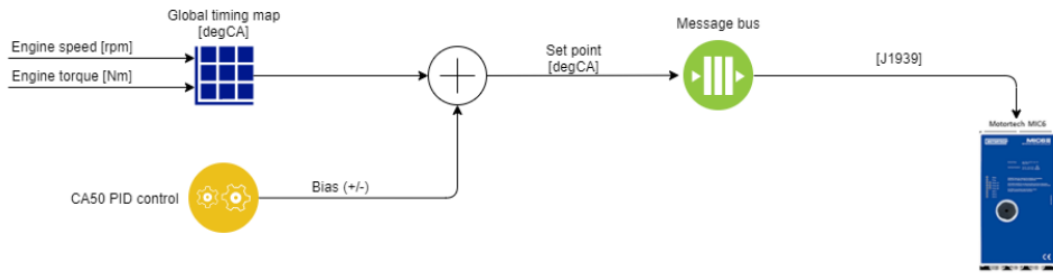


Figure 1-15: Global ignition timing control loop illustration

In order to achieve this the engine control system will measure the location for each cylinder at which 50 [%] of the fuel has been burned (CoC²) by monitoring the pressure increase curve during the combustion. The engine average value of this location is then used as feedback to the engine control system such that this location is kept at a given set point. In addition, there are multiple manipulators to this set point based on operational conditions. This makes the control of the timing location complex and difficult to maintain. Figure 1-16 shows the naming convention for the ignition timing of the crank angle degrees and some guidance to real world numbers.

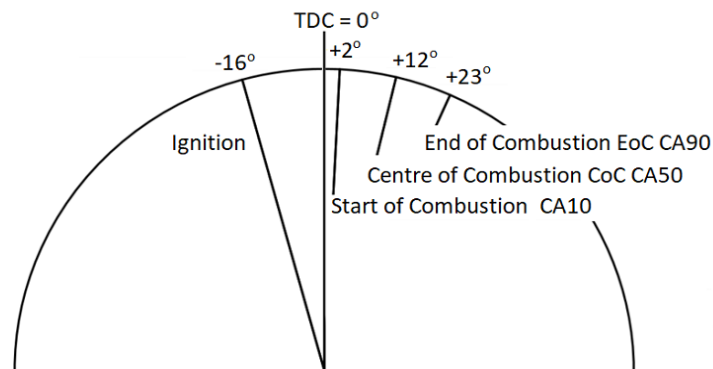


Figure 1-16: Naming convention for ignition timing

The CA10, CA50 and CA90 names indicate the locations at which 10 [%], 50 [%] and 90 [%] fractions of the fuel have been burned. These values are found based on the measured pressure curve and the gradient of it. Figure 1-17 shows an illustration of the pressure curve from a combustion. The red curve is during a power stroke and the green curve is the motored³ curve.

² CoC – Centre of Combustion, also known as CA50

³ Motored curve – This is the pressure curve as a result on compression only, that is a cycle where no combustion takes place

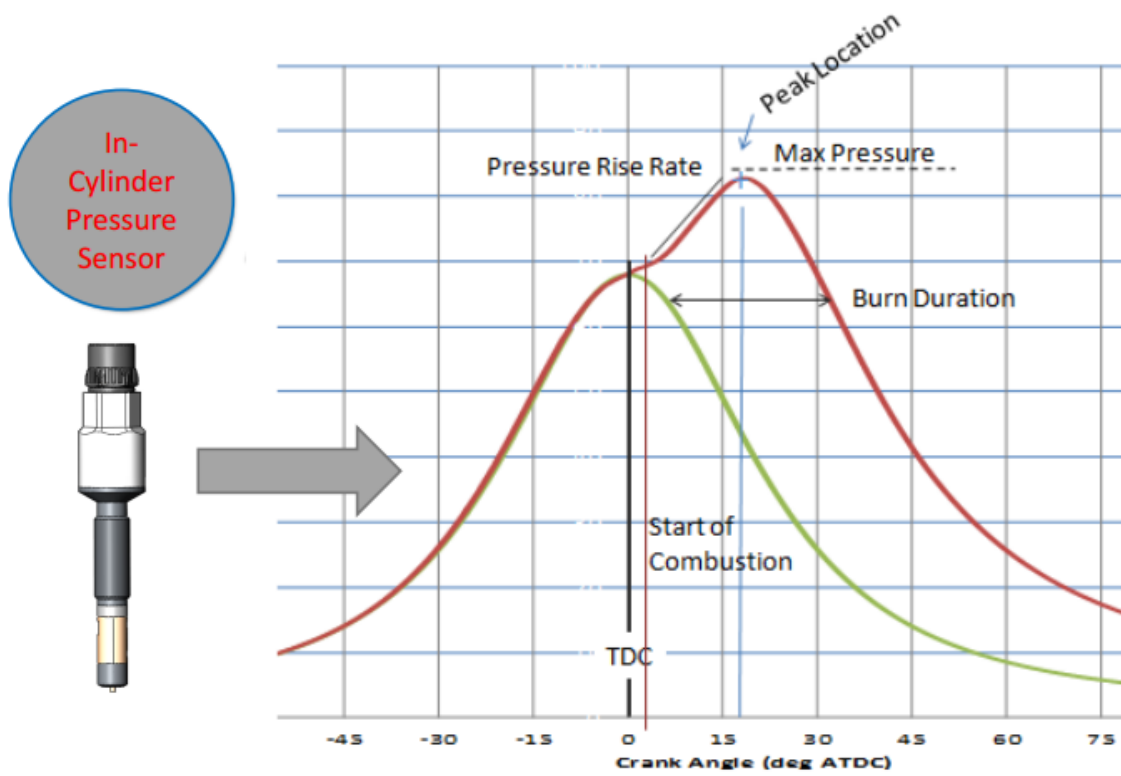


Figure 1-17: Pressure curve and locations

In Figure 1-18 the heat release curve is shown with the locations for CA10, C50 and C90. The accumulated heat release average over a number of combustion cycles are used together with the measured engine load to calculate the heat rate of the engine which in turn are the property we want to minimize. The goal is to have the relationship between the power output and the accumulated heat release to be as small as possible, that is the least amount of fuel used per power unit output.

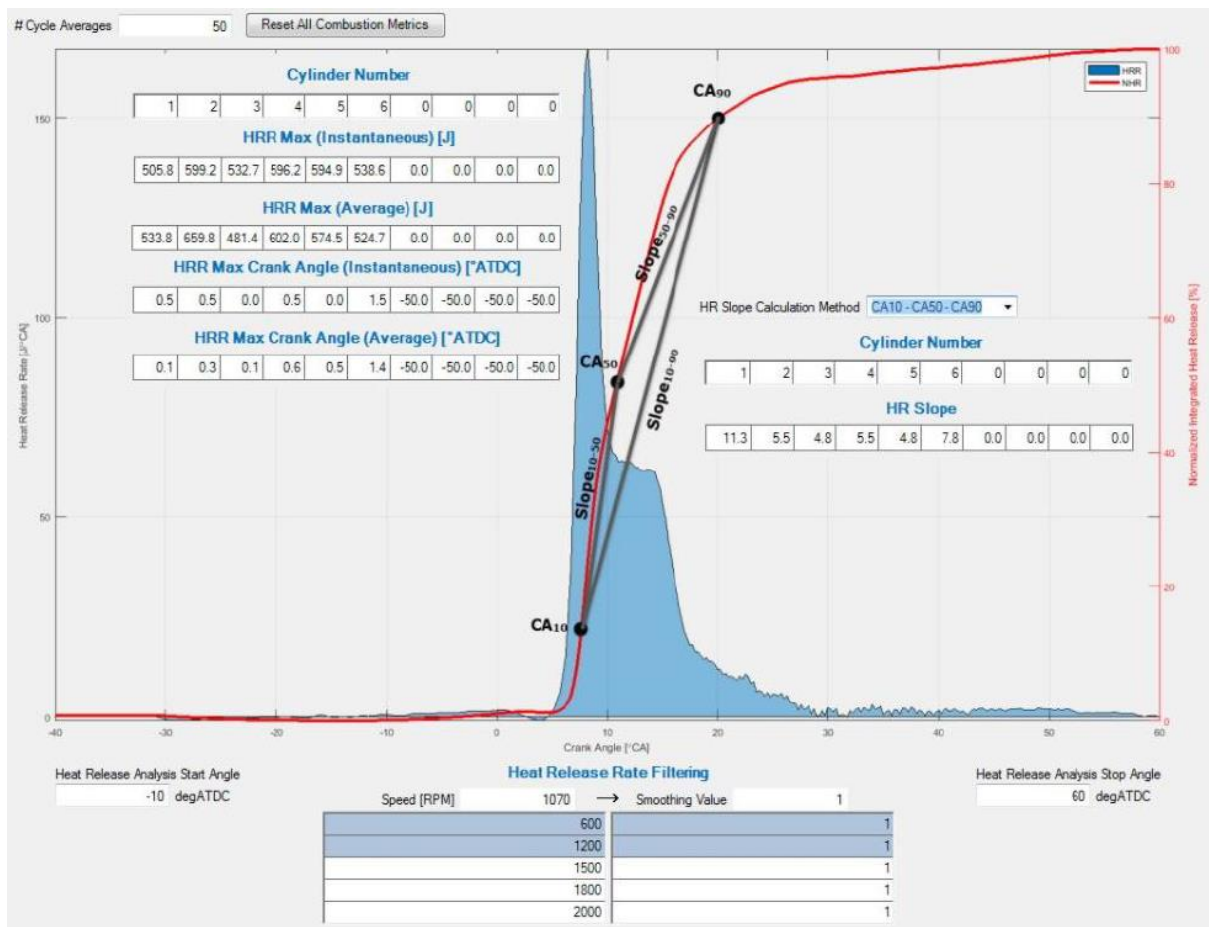


Figure 1-18: Heat release curve

This project will try to reduce this complexity by allowing the MPC to find the optimum timing set points given a set of constraints to protect the engine from running in to dangerous operational points.

In Figure 1-19 we see an indication of the relationship between global ignition timing and peak pressure and efficiency. These curves are based on data from tests performed on the previous version of the Bergen LNG gas engine, the B35:40. This engine operates at lower BMEP than the B36:45 engine and with lower peak pressures. The indicated relationship however shows similar behaviour. Earlier ignition timing, that is ignition before top dead centre (BTDC) will increase the peak pressure in the cylinder, but it will also increase the efficiency of the engine. There are however, as indicated, a mechanical limit in the construction of the engine how high the peak pressure can become before there is a risk of mechanical breakdown. The engine control system therefore monitors the peak pressure for all cylinders and in case of too high pressure the engine will shut down. This pressure will also play a role in the MPC as it must be used as a constraint to avoid too high pressure.

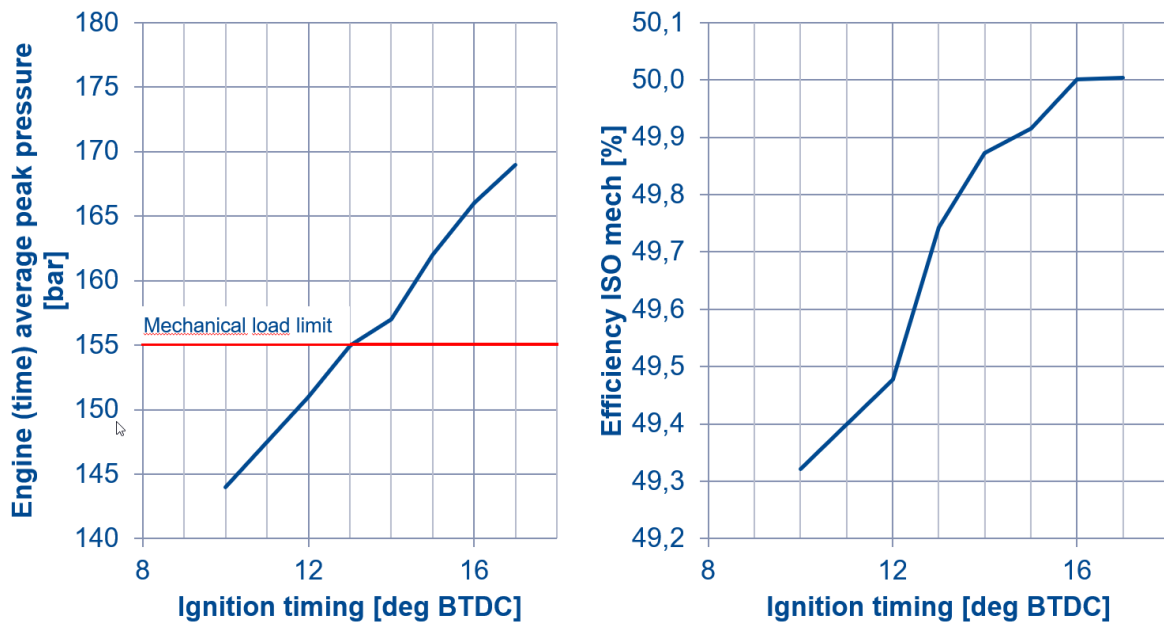


Figure 1-19: Global timing influence on efficiency and cylinder pressure

1.3.2.6 Cylinder individual timing control

The cylinder individual timing control is used to automatically balance the engine based on peak pressure measurements. The peak pressure is dimensioning factor when constructing and running the LNG engine. Too high pressure during the combustion process will damage the crankshaft and cylinder liner. The pressures for each combustion cycle are monitored and the average calculate for engine. The cylinders producing pressure above the engine average will have their ignition timing retarded while those running below the average will ha the timing advanced. This will keep the pressures and hence power output contribution from each cylinder more equal and hence no cylinder will need to run in an overload condition.

Previously this was taken care of by monitoring the exhaust temperatures and manually control the fuel amount to each cylinder mechanically. This manual process was error prone and the was both time consuming and influenced by ambient conditions. There nor a guarantee that it actually produced equal power output.

In Figure 1-20 an indication of how the cylinder peak pressures deviates during traditional operation at the left hand side (CPM⁴ off) and how the distribution is once the controller automatically adjust the ignition timing continuously to the right (CPM on). This automatic balancing of the cylinders only became possible a few years ago when the cylinder pressure sensors became affordable. Now all LNG gas engines are supplied with these sensors mounted.

⁴ CPM – Cylinder Pressure Monitoring

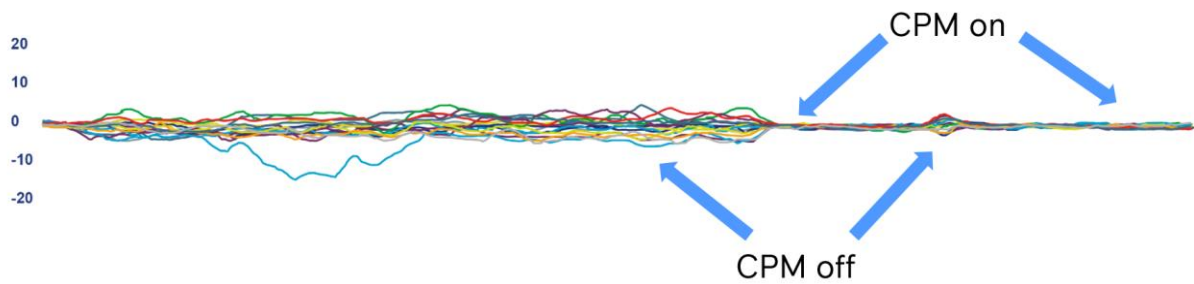


Figure 1-20: Cylinder individual timing biases on and off

1.3.2.7 Fuel gas pressure control

The fuel gas pressure has in recent years changed from a map set point based open loop control to a simple open loop control.

The set point is now directly given by the measured charge air pressure including a speed dependent bias with a separate pressure control during engine start up. This has simplified the control greatly.

In the past there was also separate control structures for the main fuel gas pressure used for the main combustion and the fuel gas which was used for the pre-chamber. The engine was then started with only gas supplied to the main combustion chamber and only after the engine had started firing and accelerated was the fuel to the prechamber enabled. This period was difficult to control and the starting phase was sometimes troublesome.

The simplification done over the recent years based on more accurate knowledge from the combustion process and CFD analysis resulting in optimisation of the fuel control valve as results in a more robust starting sequence. Today the pre-chamber and the main combustion chamber are fuelled by the same pressure and at the same time. Separate internal routings of the pre-chamber gas have also resulted in fewer components and less control structures in the control system.

1.3.3 Engine control system structure

The engine control system consists of several main units with their own dedicated purpose.

The main components and the main tasks are listed below:

- Engine safety unit
 - Separate PLC mounted on engine which monitors selected critical process values and will shut down the engine in case of breach of associated operational limits.
 - The unit will independently mechanically force the fuel control valves to zero fuel position and cut the fuel gas supply by tripping the main fuel gas valves.
- IPC PLC
 - Separate off-engine mounted PLC which will interface supervisory systems and handle active power control. Measures active power output of generator and adjusts engine speed bias in order to maintain power output at set point.
 - Handles most process value associated alarms
 - Controls the engine in primary, secondary and tertiary control.
- Engine controller
 - On-engine mounted embedded controller which controls engine control loops.
 - Interfaces IPC PLC for speed bias signals
 - Responsible for air/fuel ratio control, ignition timing, combustion process value measurements and engine limitation handling.
 - Encoder position control and cylinder pressure measurements and diagnostics
- Ignition controller
 - Receives set points from engine controller and adjusts both engine wide base ignition timing and cylinder individual timing accordingly
 - Controls spark duration and spark intensity based on set points from the engine controller.

1.3.4 Main engine controller

The main engine controller currently used is an embedded controller from Woodward Inc. The LECM (Large Engine Control Module) is a purpose-built controller with suitable hardware for interfacing large industrial engines. The software for the controller is developed and built in-house at Bergen Engines and hence gives a large flexibility in custom made control algorithms. The purpose of this project is to add the MPC control structure to this controller to optimize engine fuel optimization.

Figure 1-21 shows a graphical representation of the engine controller.

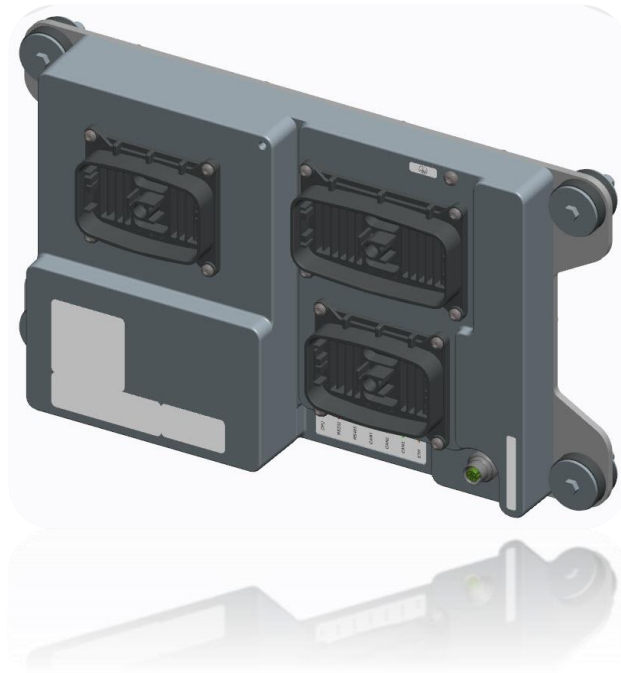


Figure 1-21: Engine controller - LECM

The control software used in the LECM is developed in MATLAB Simulink with a proprietary library for hardware access to the actual controller from MotoHawk. The MotoHawk library is a rapid programming development tool which allows engineers to quickly develop control software in Simulink to run on a MotoHawk enabled control module; like the LECM. It is model based and allows for simulation and testing within the Simulink environment during the development. This methodology also simplifies verification as already well-proven and tested functionality blocks are available.

MotoHawk also allows for support of online parameter adjustments and calibration during runtime; and includes ready-made commonly used control structures for engine control. The calibration tool is also built and maintained in-house at Bergen Engines. It is via this tool the user needs to control and adjust the control algorithms and set up the parameters. Figure 1-22 shows an example view of one of the pages on the calibration tool.

2 Model predictive control

This chapter gives a short introduction to the model predictive control (MPC in short) and the application areas it has been utilized and the general principles.

2.1 Brief history

Model predictive control was originally developed for chemical processes to control transients of dynamic systems with hundreds of inputs and outputs which was also under constraints [2]. It can be dated back to the 1960's where Kalman worked on the optimal linear control system.

During the 1970's various industrial applications using MPC were developed including MPPC by Richalet; later renamed to Model Algorithm Control (MAC); and Dynamix Matrix Control by Cutler and Ramaker [2]. Both of these were based on predicting future output of a system based on a model of the system and knowledge of the control inputs. This was done by minimizing the error between the predicted output and the measured output given some constraints. Stability in these early versions of MPC was achieved with long prediction horizon.

In the 1980's the 2nd generation of MPC was introduced, Quadratic Dynamic Matrix Control; or QDMC in short which was described by Cutler in the 1983 AIChE conference. [3]. In QDMC quadratic programming is used to solve an open loop problem with a linear system and quadratic objective function including linear inequality constraints.

Later in the 80's and early 90's; Shell engineers are among those driving further the development of MPC algorithms which also uses State Space models and Kalman filter for state estimation to overcome the problem with infeasibility by using several layers of constraints, not solved in the second generation. IDCOM-M, RMPC, PCT and SMOC were among those algorithms that were developed. These were the 3rd generation of MPC's [2]. Generalized Predictive Control (GPC) was also developed by Clarke et al. in 1987 [4] which is still widely used.

In the 90's the RMPC by Honeywell and PCT by Profimatics was merged to RMPCT, Robust Model Predictive Control Technology; and together with DMC-Plus (development of DMC and SMCA) was introduced as the 4th generation MPC.

MPC algorithms are still developed further today and the search for stability for finite horizon prediction. It has in recent years also gained a lot of interest and found applications in power system balancing as these move more toward dynamic control of smaller grid systems with large variety of power producer, including renewable sources [5].

2.2 Main objective

The main general objective of model predictive control is estimate future trajectory of a control variable u such as to optimize the future behaviour of a system output y . The optimal condition is given as a cost function for which the MPC algorithm will try to minimize for each iteration. The system here is the system or plant which are controlled by an application which implements the MPC algorithm. The MPC optimization within a limited time window is known as the *prediction horizon* based on previous states of the system. These states are the initial input to the MPC.

The MPC uses a model of the system and the cost function to estimate the future values of u which will give the best possible control outputs to achieve the goal.

In most practical applications there needs to be some limitations to this action as the optimizer can usually not take *any* value. These limitations are given to the MPC controller in forms of constraints which needs to be fulfilled. These constraints forms a specific region which is also knows as the feasible region, for which the optimal solution needs to lie within [6].

Most common are constraints on the control values, that is the output from the optimizer. The control signals used in an industrial application is usually limited by some physical constraints, for instance the amplitude of the voltage on the control signals and/or how fast the output can change. This can be a valve which cannot change instantaneously as this could potentially wear out the valve. This is something the MPC needs to be aware of and take into considerations when giving the optimal output.

$$J_k = \sum_{i=1}^N (e_{k+i}^T Q_i e_{k+i} + u_{k+i-1}^T P_i u_{k+i-1} + \Delta u_{k+i-1}^T R_i \Delta u_{k+i-1}) \quad (2-1)$$

$$e_{k+i} = (y_{k+i} - r_{k+i})$$

Equation (2-1) shows the general objective function; or cost function; for an MPC where J is a scalar value at time instance k . N is defined as the prediction horizon, $Q_i \in \mathbb{R}^{m \times m}$, $P_i \in \mathbb{R}^{r \times r}$ and $R_i \in \mathbb{R}^{r \times r}$ are symmetric positive semi-definite weighting matrices for the operator/user to specify. P can often be set to zero in order to obtain offset free control [7]. The weighting matrices are usually time invariant and hence do not change during the prediction horizon as will be the case in this project.

The general quadratic programming representation of the equation in (2-1) is given in (2-2).

$$J_k = e_{k+1}^T Q e_{k+1} + u_k^T P u_k + \Delta u_k^T R \Delta u_k \quad (2-2)$$

$$e_{k+1} = (y_{k+1} - r_{k+1})$$

Note again that P can be set to zero to get offset free control, given that no constraints are active.

Some of the advantages of the MPC over traditionally methods like PID control includes:

- No dynamic parameters that needs to be tuned
- Can be used from simple to complex systems for both SISO⁵ to MIMO⁶ systems
- Can control systems with long delay times
- It introduces feed forward control inherently
- Easy integration of constraints

On the other hand, there are also some drawbacks of using a MPC Vs the more traditionally PID controller; some of these are listed below

- For more complex problems the computational effort can be too high to be done every time step
- When constraints are considered the computational effort is even higher
- The availability of a system model and the results will depend on the discrepancies between the model and the real process.

⁵ SISO – Single Input Single Output

⁶ MIMO – Multiple Input Multiple Output

2.3 Cost function

The cost function is a scalar used to determine when the optimal solution is found. For a set point tracker this scalar will indicate the difference between the future output and a given reference but at the same time take into account that a change of the output comes with a cost. The cost is summed over the prediction horizon. The goal is to minimize this cost over this prediction horizon with respect to future control outputs. The cost function is given as a general equation in Equation (2-1). The cost function can be expanded to take in any number of elements that we want to play a role in the minimization and the user uses the weight matrices to determine which cost that would carry the highest impact to violate.

The weight matrix $Q \in R^{m \times m}$ where m is the number of set point objectives to achieve.

The weight matrix $P \in R^{r \times r}$ where r is the number of control inputs.

The weight matrix $R \in R^{r \times r}$ where r is the number of control inputs.

The matrixes are symmetric and positive semi-definite.

2.4 Receding horizon

The output from the optimizer is a prediction of optimal future control movements for the whole prediction horizon. In an industrial MPC we usually only utilize the first element of these optimal control outputs and apply it to the system under control. We then measure the response from the system and its states and feed that back to the optimizer and we do it all over again. This is called the moving horizon or receding horizon control law. Figure 2-1 shows the principle of the MPC operation where we for each sample time produces a new optimal control output sequence for the next $k + i$ time steps (the prediction horizon).

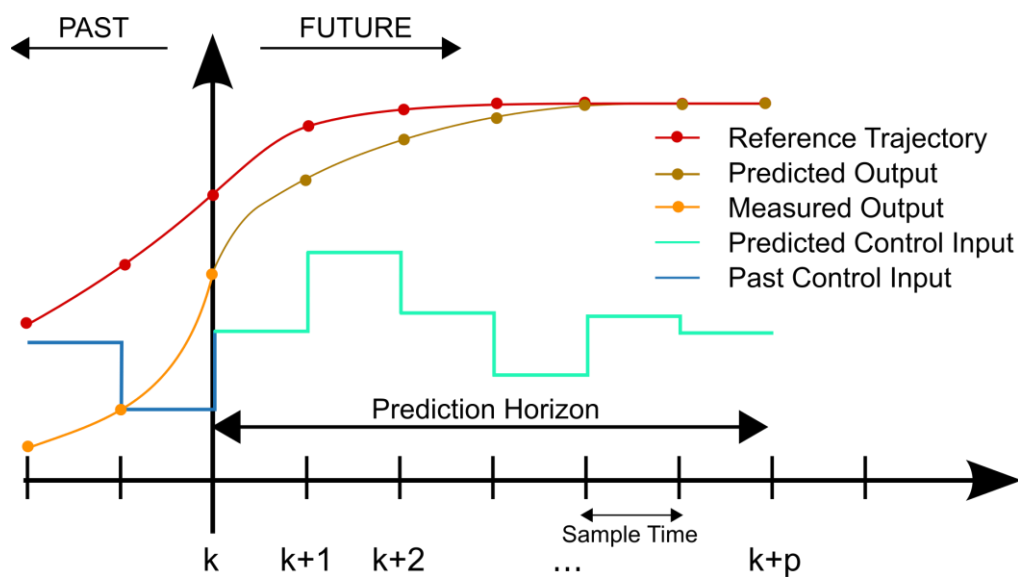


Figure 2-1: MPC principle [5]

One of the, possibly, disadvantages of this method is the computational effort involved in predicting a larger number of future control moves for each iteration where only the first move is used. Depending on the size of the problem and the number of constraints and number of parameters to optimize this might take a substantial large finite time which in turn only allows for slower update times. The sample time needs to be larger than the time it takes to do the optimization.

For the problem at hand in this project the updates for process values are usually in the range of 100 [ms] for slower algorithms and 10ms for the faster controls. This optimization might however possibly run at slower rates as this is an optimization for long term fuel efficiency.

2.5 Constraints

One of the benefits of MPC is the handling of constraints and especially on the control signal. Both the amplitude and rate of change of the control signal can be added to the problem formulation and hence is taken into account when the optimal solution is calculated. Both inequality and equality constraints can be added.

Constraints can be both hard and soft constraints where the difference is that the hard constraints must be obeyed at all time instances while soft constraints can be exceeded in case infeasibility occurs. Hard constraints are typically on the control output (both amplitude and rate of change) which cannot result in infeasibility. Soft constraints can for instance be on both measured outputs of the system and on internal states. Here the slack variables can be used to control how each constraint that can be violated the most and to what expense [6]. The slack variables are therefore added to the cost function so that the amount of violation is reduced.

The constraints on the control variable can be written as inequality constraints such that

$$\begin{aligned} A\Delta u &\leq \gamma \\ Au &\leq \gamma \end{aligned} \tag{2-3}$$

where A is a matrix and γ is a vector.

The constraint on the amplitude of the control signal can be written as

$$u^{min} \leq u \leq u^{max}$$

Which can be rearranged to

$$\begin{aligned} u &\leq u^{max} \\ -u &\leq -u^{min} \end{aligned}$$

Most linear MPC controllers are in the form of a QP problem with inequality constraints as given in (2-3). These are then solved iteratively by the optimizer. The number of iterations is however limited and there is no guarantee that a feasible solution can be found within a finite number of iterations.

The QP problem is usually always solved by an active set method [7]. The method is further explained in section 4.8.1.2.

2.6 Grouping

One way of reducing the size of the optimal control problem is by introducing grouping of control signals [6]. The idea here is that the prediction horizon can be divided into several blocks in which the control signal is kept constant. The groups can be both equal in size or with different sizes. By keeping the control signals constant within each group, we will only need to consider that signal once within that group instead of for each sample. The size of unknowns which the optimizer than need to estimate is then reduced to the number of groups times the number of control signals.

For example, if the prediction horizon is 100 samples and it is split into 5 groups and the number of control signals is 2; then the number of unknowns prior to grouping is 200 while after the grouping it is only 10.

Depending on the number of control signals and prediction horizon, this reduction method could drastically reduce computational effort.

2.7 System model

The system model used for prediction can be considered as the biggest drawback for an MPC. The model and how it predicts the outputs based on inputs can be difficult to develop for highly dynamic systems, at least if these are to be based on conservation laws. Usually these models are data driven linear models based on measured input/output data. As for this project, the data is used in system identification to get a linear model representation of the system given as a state space model.

The system model in an MPC is used as a prediction model, that is a model that will predict future outputs and states based on known inputs over the prediction horizon.

The type of model can vary depending on the application and area of use. One popular model type is the Truncated Impulse Response Model which is based on passed impulse responses to the system. It does however need a large number of parameters for estimation and are not feasible for open loop or unstable systems [8].

Another popular choice is the Step response model which is very much the same as the impulse response model, and only differ in the excitation signal used on the system.

The state space model is perhaps one of the most widely used in the academic/research community. The derivation can be very simple from measured input/output data and even for the multivariable systems. The state space version allows for the use of well know thermos and theory and applies to both linear and non-linear cases. Well proven system identification techniques exist based on measured data already included in the MATLAB toolboxes. Tools for analysis of stability and other properties can be utilized based on the state space model.

A transfer function model is also very popular in by researcher and academic alike. It is somewhat more difficult to derive but it often required few parameters. It handles dead time inherently and is commonly used in by industry engineers to explain and model systems. This very much applies to the Bergen Engines as well where the engine simulators are based on numerous of transfer functions.

2.8 Integral action

The MPC algorithms are based on the accuracy of the system model to represent the actual system under control. There exist errors between the model and the real system in all applications. One of the results from this mismatch is the probability of offset error during steady state. That is there exist an error between the predicted output from the model and the measured output from system under control.

In order to counteract this offset error, there are several methods that can be used to achieve offset free control by adding integral action to the MPC. According to Åkesson [9] a disturbance observer can be used to obtain offset free control by adding integral action to an output feedback controller. This solution was generalized for the cases where the number of outputs exceeds the number of control inputs.

One approach discussed by Rawling [10] is the addition of integrating disturbance to the process model which shows that offset free steady state can be achieved with open loop stable, integrating and unstable systems. This is done by adding a number of integrator disturbance terms equal to the number of measured variables.

The method used in [11] by Wang is to use embedded integrators by using the augmented state space model of the system which includes the Δu in the state space formulation and the Δx in the state variable vector.

2.9 Previous work

MPC's has been successfully used in a range of different areas and applications. Mostly interesting for this project is use in the industry as this control problems carries some synergies. MPC's has proven in use for instance in steam generators and servos as illustrated in [12]. In later years, as the era of autonomy evolves, the MPC is (and can be) used in a large area of applications within systems for auto pilot for both the automobile and marine areas.

In the Marine area the auto pilot for ships and DP⁷ operation use the benefits from MPC controller. Withing Rolls-Royce the use of MPC has been introduced in the engine control system for high speed MTU engines where the newly developed Hyperspace controller uses MPC controllers to find the optimal set points for the operation, based a given goal for the engine. That goal could be optimal fuel efficiency, minimum NOx emissions or disturbance rejection, all individual application dependent.

In the automotive industry the MPC principle is widely used in adaptive cruise control, obstacle avoidance, lane keeping assist etc. [13]. Separate toolboxes for MATLAB have been developed to serve the automotive industry with tailer made solutions for MPC control within this area.

2.10 Set up

The simulation results and testing has been done using MATLAB 2019b together with SIMULINK. The host computer on which this project has been done is a HP ZBook 15 G5 with an Intel Core i7-8850H CPU with 16GB or RAM. The host computer runs Windows 10 Enterprise build 17763.

⁷ DP - Dynamic positioning

3 Modelling

The models for this project was developed using the System Identification toolbox in MATLAB. During the master project several models based on different realization techniques was developed. Based on the conclusion in the report the selected model structure was the polynomial model from the System Identification toolbox.

3.1 Data collection

The dataset used as basis for the development of the MPC model is measured input/output data from a commercial operational engine. The engine is operated on power plant in the city of Tabor in the Czech Republic connected to the national grid. The engine primarily works on a base load operation, but often operate in Secondary regulation. It is these data sets which are used to modelling as these contains the most dynamic behaviour end exits the system the most.

Figure 3-1 shows an image of a part of the C-Energy power plant.



Figure 3-1: C-Energy power plant

3.1.1 Sampling

The data is sampled at 10 [Hz] by the engine controller and transmitted to an edge computer located on the power plant premises. Most process values available as well as internal engine controller states and statuses are collected and transmitted to this edge device. The currently used edge computer is shown in Figure 3-2 and is a Siemens IOT 2040. On this unit it runs several Docker containers which handles various aspects of the data collection and transmission to the cloud service.

On the edge computer the data is stored in a local Influx time series database for local access by the service engineers and customers through Grafana. The data is then stored in .csv files which each contains 30 minutes of process data. The .csv files are transmitted to an Azure cloud storage location at regular intervals. In the cloud there are several process running which utilizes these data for both machine learning, abnormally detection and for historic time series

visualization. An Influx database in the cloud stores the time series data at different resolutions for different length. For instance, the 10 [Hz] data is stored for some months while 1 [Hz] and 0,5 [Hz] variants are stored for longer.



Figure 3-2: Edge computer - Siemens IOT2040

From this storage several datasets were retrieved and analysed during the Master project.

3.1.2 Pre-processing and analysis

The datasets which was retrieved from the cloud storage was first “washed” by engineers by reducing the number of useful variables from several hundred down to about 20 which would be of interest. The dataset was then processed further by removing data where the engine was not running and not producing power. A soft sensor was added as the heat rate is not directly given by the dataset, but as a direct calculation by others.

The engine control system calculates the engine heat release cumulation based on the pressure rise rate during the combustion. This heat release is given in J/cycle. To get to the heat rate which is the one we would like to minimize we divide the heat release with the IMEP.

$$heat\ rate = \frac{heat\ release}{IMEP}$$

PCA analysis was performed to find co-linear variables and which variables that had the most contribution to the dynamics of interest. During the analysis and discussion afterwards, a decision was taken on the number of inputs, outputs and disturbances. The model structure was then given by on 5 inputs variables, whereas 2 are the controllable variables and 3 are measured disturbances, and 6 measured outputs.

3.2 System identification

During the modelling part of the Master project several different models based on different principles was developed. Models was developed using Neural Networks, DMDC, the DSR toolbox and the System Identification Toolbox.

Based on the findings and conclusions it was decided to go on with a simple polynomial model from the System Identification Toolbox. This model has a simple input/output structure where there is a simple relation between the model and the physical world.

The polynomial model found with the System Identification toolbox is transferred to a state space model during the initial phase of the MPC development as the Simulink testing did not allow for the polynomial model to be used. The state space model derived from the polynomial model was used for a substantial amount of time during the testing of different MPC structures. This model initial gave promising results but when adding constraints to the output it did not behave as expected.

During the simulation is was then decided to re-do the testing with a new model based on the same data set as the original polynomial model, but this time the model was developed as a State Space model from the start. The same training data set and validation data set was used to develop the new model.

The results presented in this paper are mostly from the State Space model. Most of the already performed simulations had to be redone but the results looked more promising with the new model.

The discrete state space model is given by the general form

$$\begin{aligned}x(k + 1) &= Ax(k) + Bu(k) + B_d u_d(k) \\ y(k) &= Cx(k)\end{aligned}\tag{3-1}$$

where x is the state vector, u is the control signal vector, u_d is the measured disturbances and y are the measured outputs. The sampling interval is denoted by k .

From the model we find that the state matrix A is 25×25 ; The input matrix B is a 25×2 and the disturbance matrix B_d is 25×3 . The output matrix C is 6×25 .

There are no direct feed trough terms in the model and hence no D matrix.

Contrary to the polynomial model which had a identity matrix in the C matrix of the state space formulation, this state space model has elements in the output matrix. For the polynomial that meant that the measured outputs where available as the first 6 elements of the state vector. This is not the case for the new state space model and hence the physical “meaning” of the states is lost.

The model order for the polynomial was $n = 30$ while the order is chosen as $n = 25$ for the state space model. The model order is based on an inspection of the residual error plot of the system identification toolbox where the logarithmic error flattens out.

In Figure 3-3 we note the tracking of the simulated heat rate from the model and the measured heat rate from the validation data set. The fit here is about 79 [%].

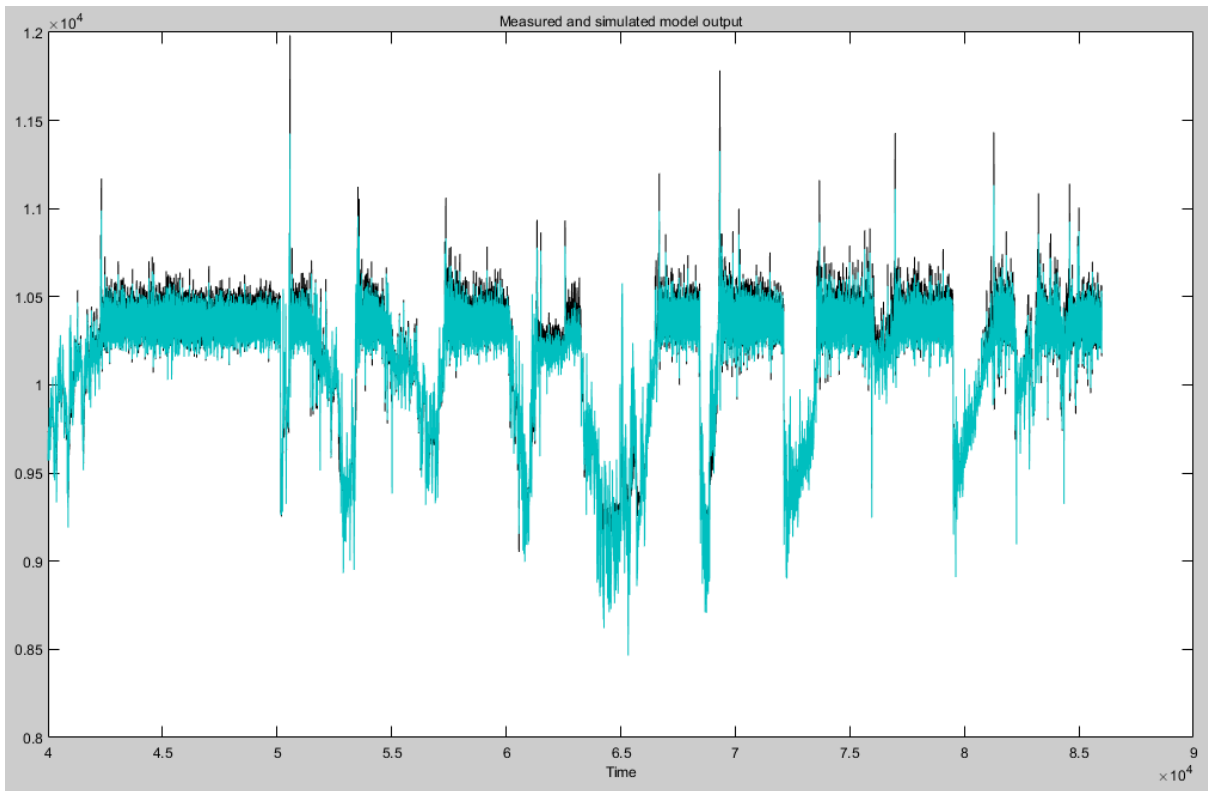


Figure 3-3: Output from System Identification - Measure heat rate Vs simulated

Another example is shown in Figure 3-4 where the *O2* from the validation data set is compared to the simulated.

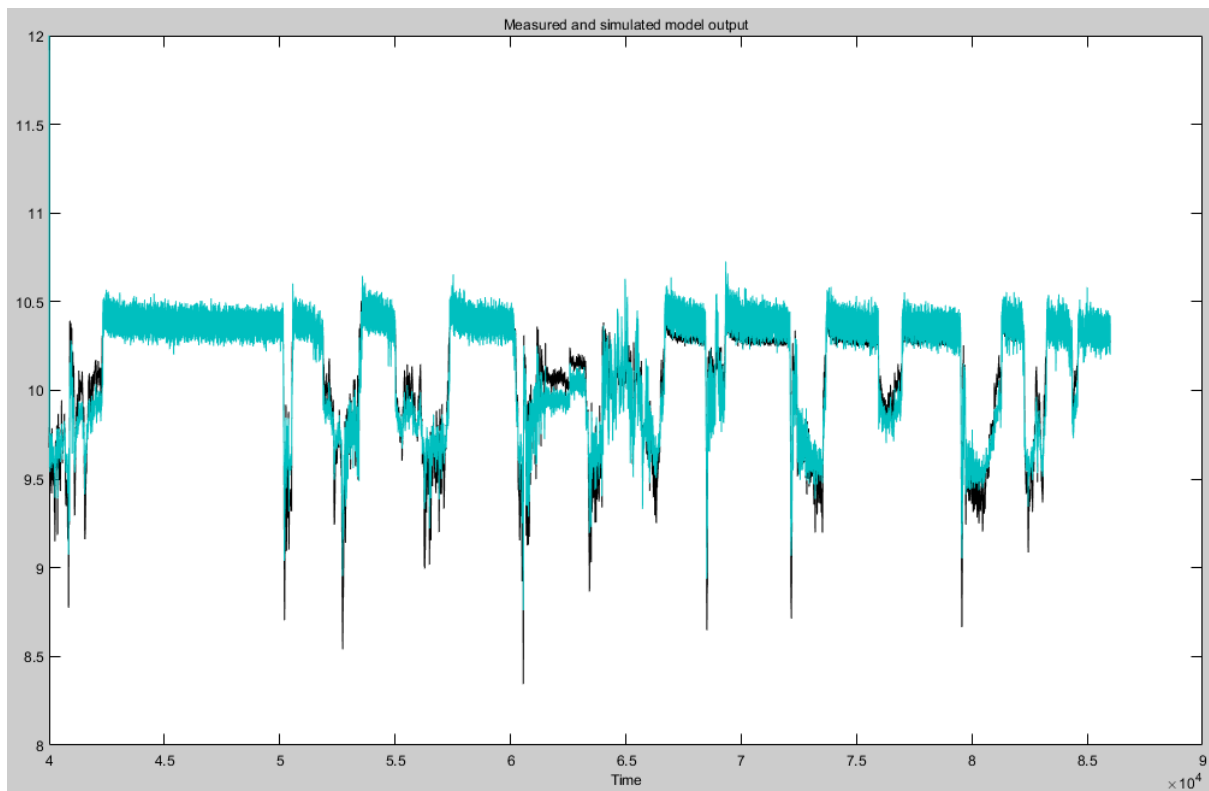


Figure 3-4: Output from System Identification - Measure O2 percentage Vs simulated

The code from the MATLAB for the model estimation

```
% State space model estimation
```

```
Options = n4sidOptions;
```

```
Options.Display = 'on';
```

```
Options.EstimateCovariance = false;
```

```
Options.Focus = 'simulation';
```

```
Options.EnforceStability = 1;
```

```
Options.N4Weight = 'CVA';
```

```
Options.N4Horizon = [75 0 87];
```

```
ss1 = n4sid(TrainingMean20200902Red, 25, 'DisturbanceModel', 'none', Options)
```

3.2.1 Controllability

After the development of the state space model a simple check was done to test the controllability of the model. In the MATLAB the function `ctrb(sys.A, sys.B)` was used to get the controllability matrix and then checking the rank of this matrix has full rank n .

Since the state space model has rank $n = 25$ we check if the controllability matrix has the same rank.

The rank of the controllability matrix is also $n = 25$ and hence there are possibility that the system is controllable since there are not direct uncontrollable states. Howere there are limitations to this method of detecting controllability and the function is sensitive to round off errors and errors in the data.

Testing also with the MATLAB function $[Abar, Bbar, Cbar, T, k] = ctrbf(sys.A, sys.B, sys.C)$ and looking at the length of $sum(k)$ which is the number of controllable states we also end up with 25 which is the same as the rank of the A matrix.

3.2.2 Limitations

The model used for the development of the MPC is, as described in the previous sections, based on real life operational data for an installation. It is known and well understood that this model has limitations. Even though the data availability is large the data is based on a real installation running with control structures which are kept withing operational limits. These control structure keeps the engine at the operational point to the best of their capability.

This results in the data not being exited with control signals to its maximum and minimum and hence the representation of the data is within these limitations.

When the data series was selected it was from areas where the dynamics where present as much as possible, so some excitation is experienced.

In order to improve on the data quality the control software of the engine controller was updated with additional logic which will be used to exited the control outputs to such an extent that a larger operational area of the measured data can be captured. A test program for testing on a real engine has been defined and a production engine was selected for the experiment. The test program was approved by the Bergen Engines management. Unfortunate due to circumstances and delay in the production line resulted in the selected engine testing being delayed into the easter holyday. This resulted in reduced availability of test engineers and due to delivery obligations, the test was postponed and needs to be done on another engine at a later stage.

The test is still scheduled to be done at the next available time slot.

The limitations in the model do however not interfere with the proof of concept as the basic idea and testing would still go on with the model at hand. This model would still give a good understanding of the various implementations of the MPC algorithms and the overall structure could be tested.

3.2.3 Open loop simulation

This section shows some of the initial test of open loop simulation of the model.

The purpose of the open loop simulations was to check the model was stable if all inputs was kept at constant values and how the model reacted to changes in either of the two controllable inputs.

From the definition of the original problem the controllable inputs were defined as the “charge air pressure” and “global timing”. There are three measured input disturbances: the “IMEP”, “Charge air temperature” and “Suction air temperature”.

Before any of the tests were performed the initial state was found using MATLAB built in function for “findstate”. The model was also run for several iterations with real life data to have the states updated with real life data before starting.

The code for the open loop simulation is found in

3.2.3.1 Open loop simulation with no change on controllable inputs

The first test is open loop simulation with steady state input data with an initial state. The test is performed to verify that the model is stable over time. The test was run with several simulation times and the results for testing with 15000 samples (1500 seconds) are shown in Figure 3-5, Figure 3-6, Figure 3-7 and Figure 3-8.

As we can see from Figure 3-7 and Figure 3-8 the output stabilizes to a steady state value as expected with no changes to the inputs.

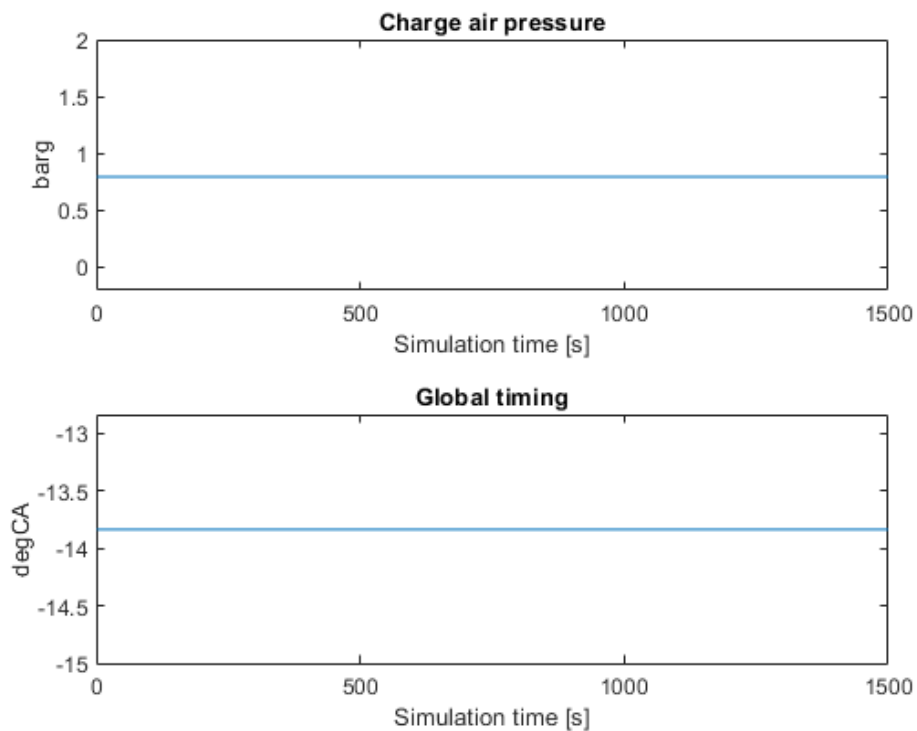


Figure 3-5: Open loop simulation with fixed input values – controllable inputs

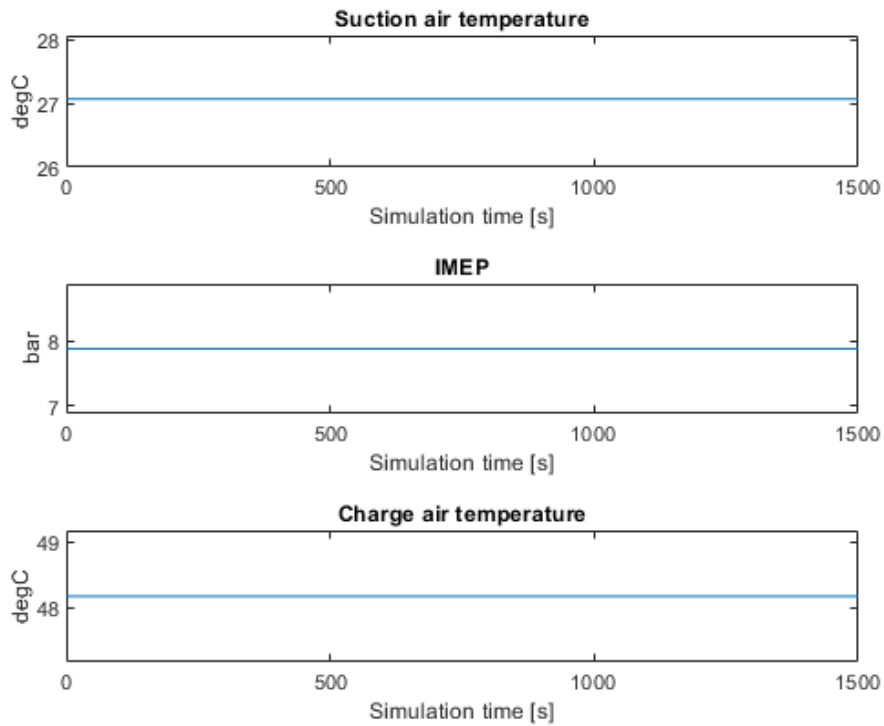


Figure 3-6: Open loop simulation with fixed input values – measured disturbances

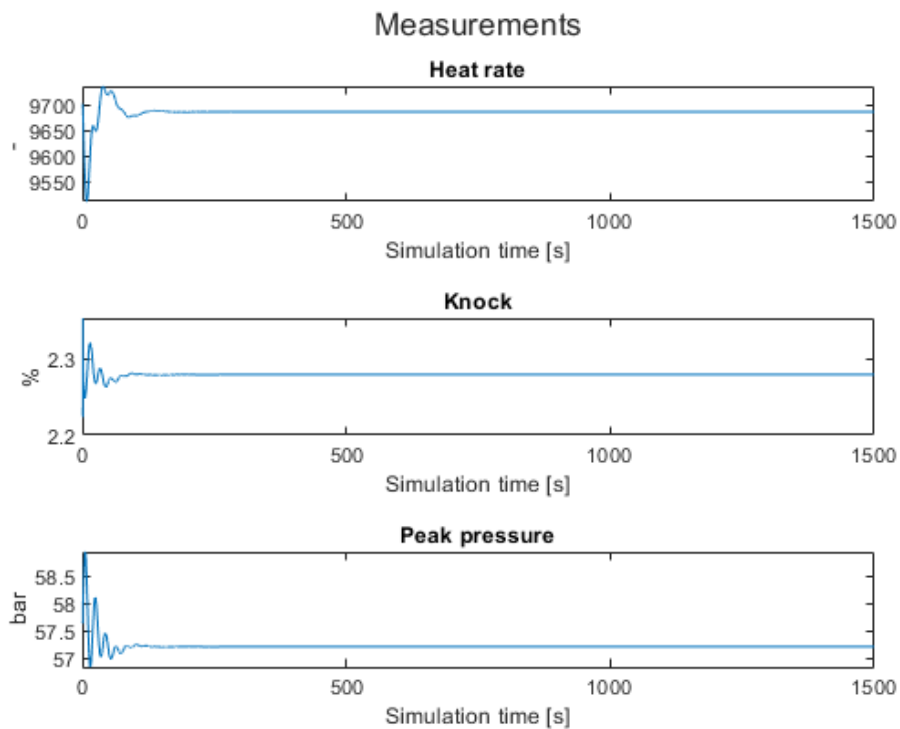


Figure 3-7: Open loop simulation with fixed input values - measured outputs from the model

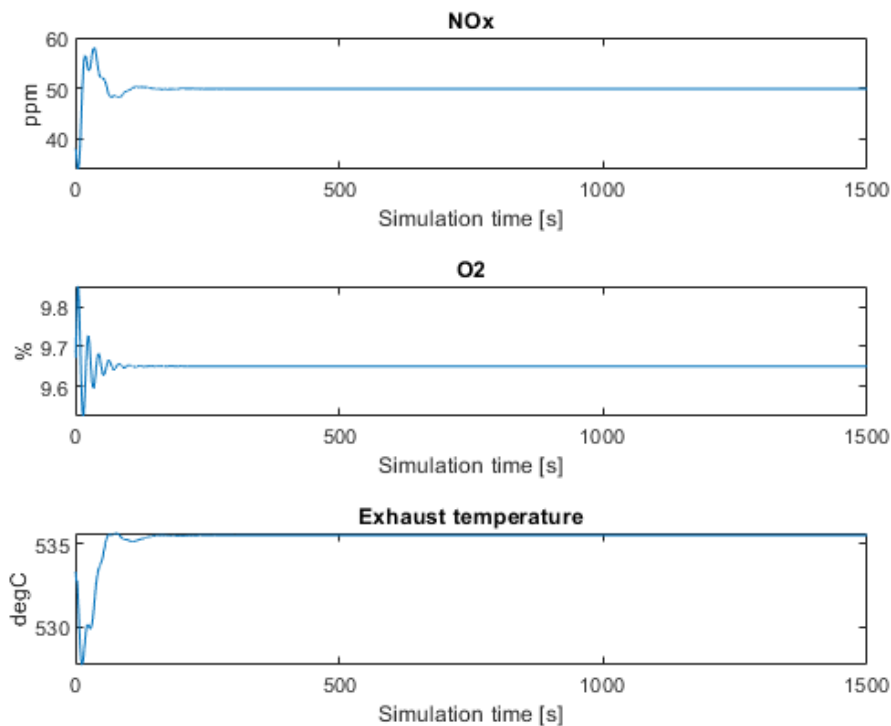


Figure 3-8: Open loop simulation with fixed input values - measured outputs from model

3.2.3.2 Open loop simulations with positive step change on charge air pressure

This test is done by manually adding a step change to the charge air pressure control in the positive direction. That is to add extra air to the combustion process. As for the test in 3.2.3.1 the measured disturbances are kept constant and are not shown in the following figures.

It should be noted that a step change is a not possible in real life as the pressure needs to build up over time nor is it possible to build up high pressure at lower engine power outputs as the turbocharger is not able to produce higher pressure due to lower energy in the exhaust.

There is also an upper limit to the boost pressure as it must not become too high as this could lead to too lean mixture and misfire as well as turbocharger stall.

The step input is shown in Figure 3-9 and the output responses are shown in Figure 3-10 and Figure 3-11.

As expected, the increase in air pressure to the combustion process reduces the exhaust gas temperature due to the increase in air excess, the O₂ level increases as expected and the NO_x decrease. The increase in peak pressure mostly indicate that the charge air pressure is closely related to IMEP (engine load) during closed loop real life running and hence the model interprets this as a potential load increase and rises the pressure reading.

The heat rate shows a sudden initial drop before stabilizing at a slightly lower value than before the step change.

With the step change added here the model outputs do not exceed any “normal” operating values.

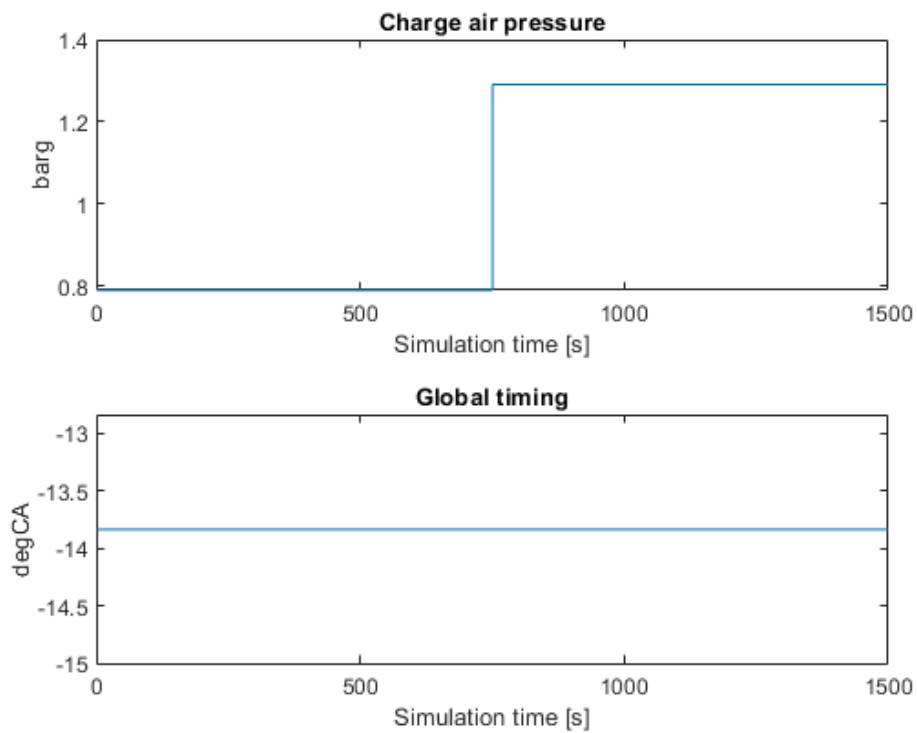


Figure 3-9: Open loop simulation with step change in positive direction on charge air pressure

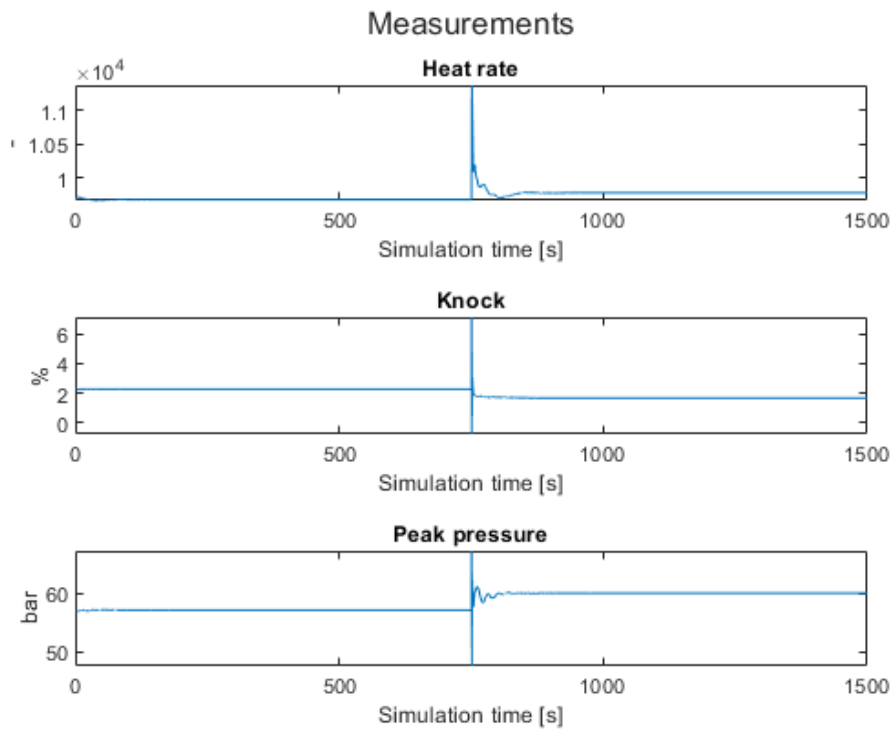


Figure 3-10: Open loop simulation with step change on charge air pressure - model outputs

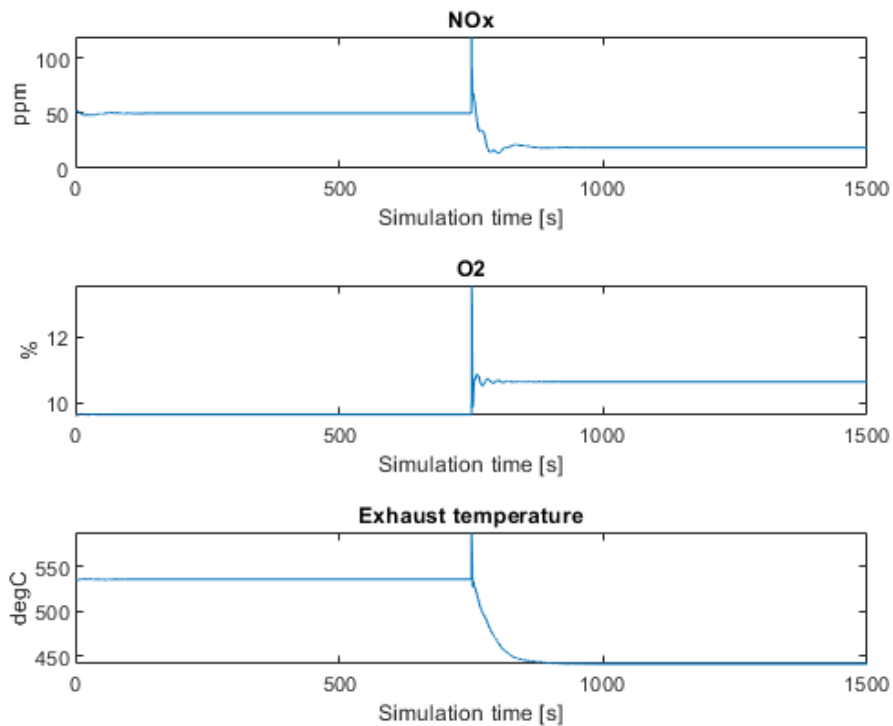


Figure 3-11: Open loop simulation with step change on charge air pressure - model outputs

3.2.3.3 Open loop simulations with negative step change on charge air pressure

This test is done by manually adding a step change to the charge air pressure control in the negative direction. That is to reduce the air to the combustion process. As for the test in 3.2.3.1 the measured disturbances are kept constant and are not shown in the following figures.

It should be noted that a step change is a not possible in real life as some time will pass before the pressure is effectively removed.

The step input is shown in Figure 3-12 and the output responses are shown in Figure 3-13 and Figure 3-14.

Here the results are expected to be the opposite of the test in 3.2.3.2 for most of the outputs of the model. Indeed, the exhaust temperature is increased as expected and the O2 percentage is reduced. The NOx level increases slightly as expected.

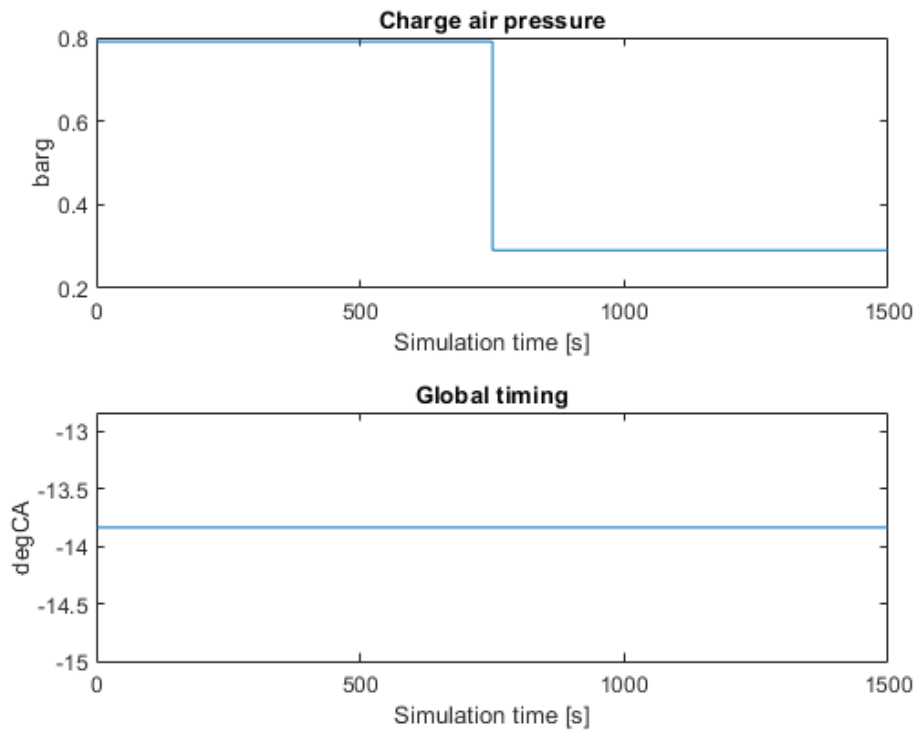


Figure 3-12: Open loop simulation with a negative step change in charge air pressure

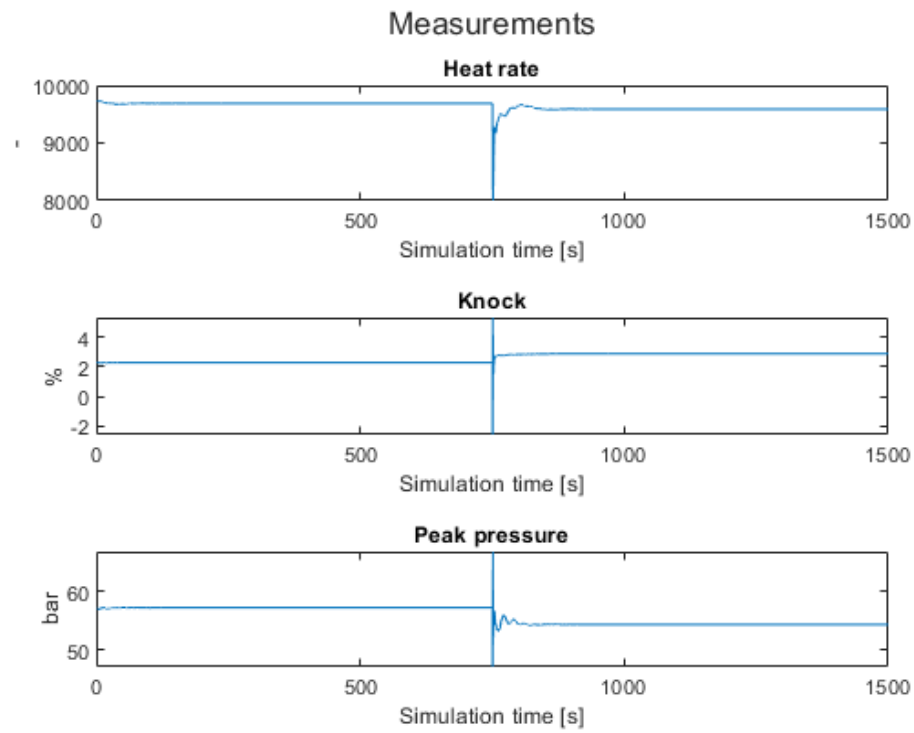


Figure 3-13: Open loop simulation with negative step change on charge air pressure – model outputs

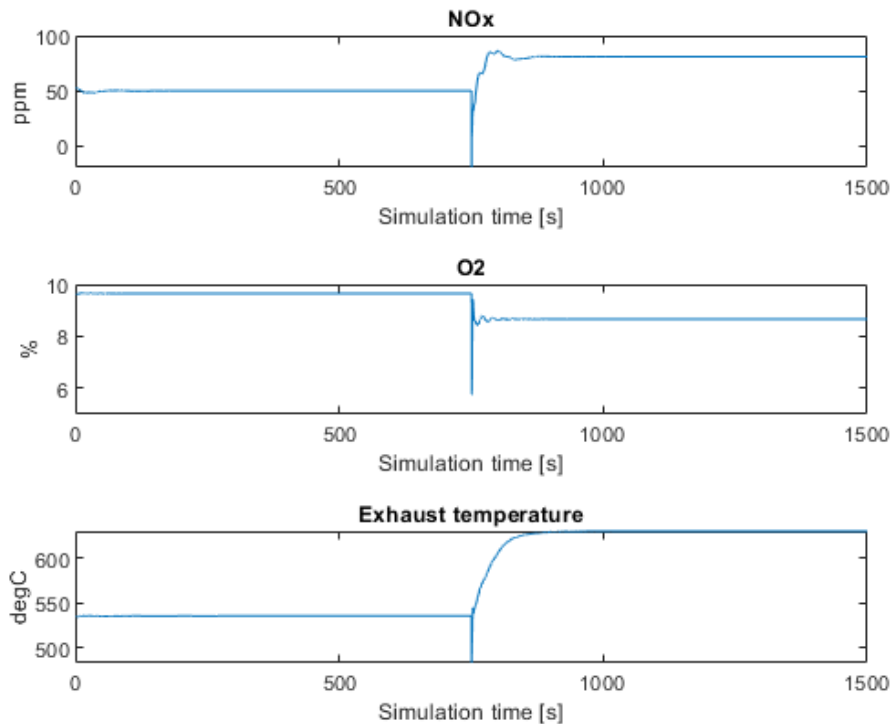


Figure 3-14: Open loop simulation with negative step change on charge air pressure – model outputs

3.2.3.4 Open loop simulation with step change in global timing - advanced timing

This test is performed by keeping the charge air pressure control input fixed and add a step change on global timing. The step change is in the *advance* direction which means earlier ignition timing. The timing location is given as a negative number indicating before TDC⁸.

The step change in the control signal is shown in Figure 3-15 and the model outputs is shown in Figure 3-16 and Figure 3-17.

From the results we get some of the expected results based on system knowledge. The earlier ignition timing increased the peak pressure and increases the NOx level. We also see a slight change in O2 level, but this might be due to the model expecting higher NOx to influence the O2 level. The exhaust temperature is reduced as is normal with earlier ignition timing resulting in faster combustion. The heat rate is reduced to indicate higher efficiency.

⁸ TDC – Top dead center. The absolute crank angle is given as 0-degree crank angle when a cylinder is at TDC in the compression stroke. The total number of degrees during a combustion process is 720 degrees, i.e. 2 revolutions of the crankshaft.

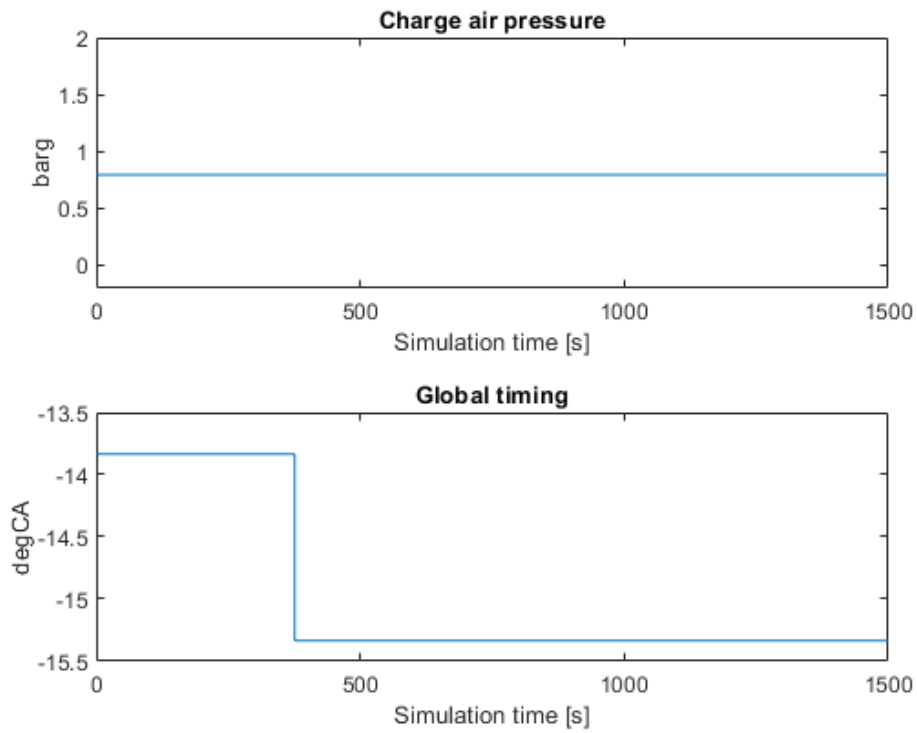


Figure 3-15: Open loop simulation with step change on global timing – earlier timing

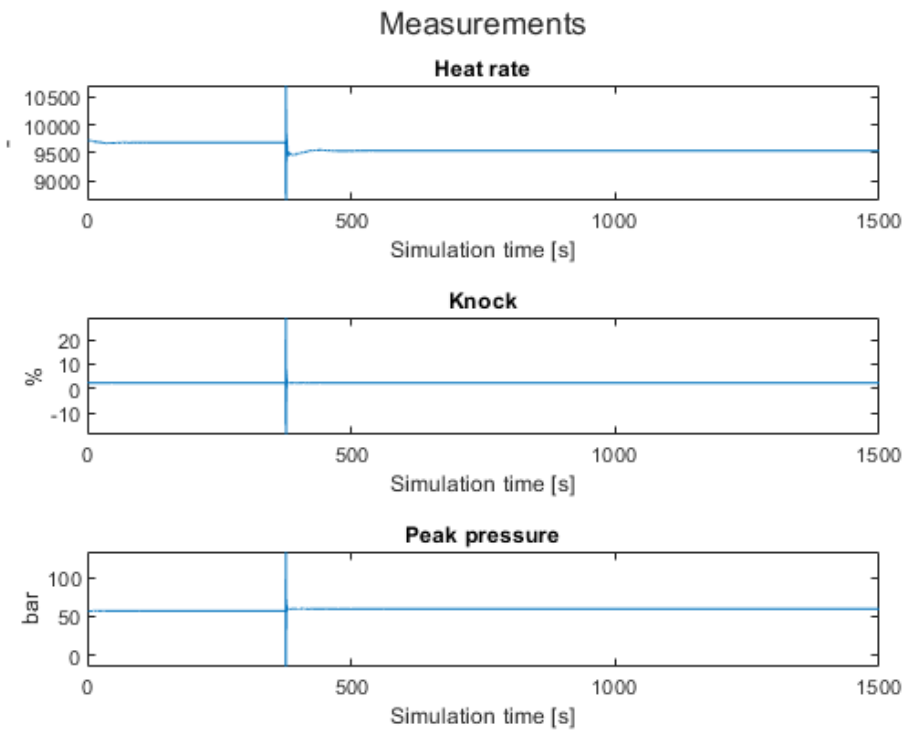


Figure 3-16: Open loop simulation with step change on global timing - earlier timing - model output

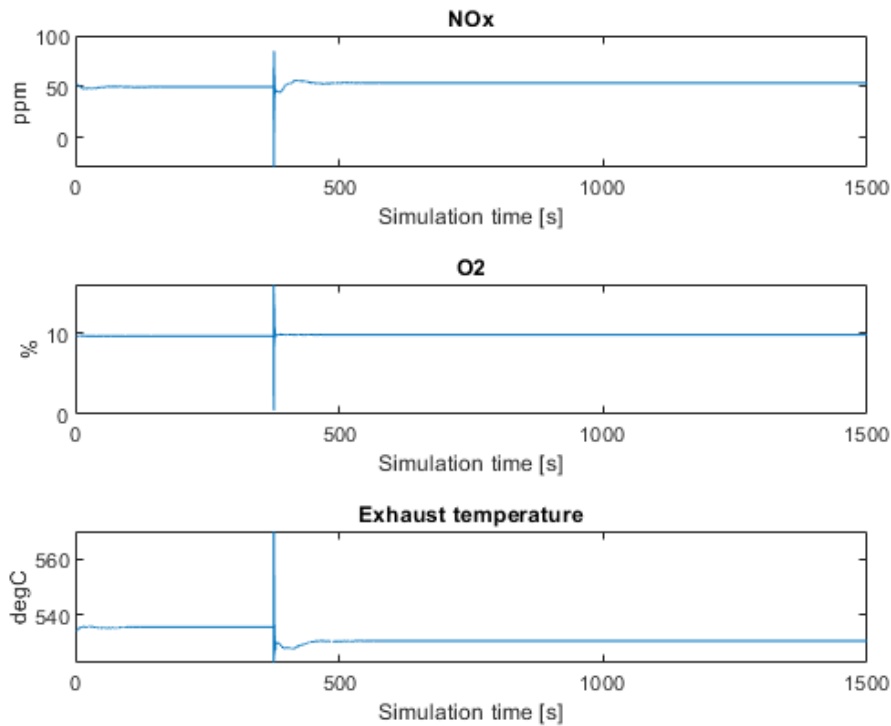


Figure 3-17: Open loop simulation with step change on global timing - earlier timing - model output

3.2.3.5 Open loop simulation with step change on global timing – retard direction

This test is performed by keeping the charge air pressure control input fixed and add a step change on global timing. The step change is in the *retard* direction which means later ignition timing. The timing location is given as a negative number indicating before TDC.

The step change in the control signal is shown in Figure 3-18 and the model outputs is shown in Figure 3-19 and Figure 3-20.

From the results we get some of the expected results based on system knowledge. The later ignition timing decreases the peak pressure and lowers the NOx level. We also see a slight change in O2 level as for the output from the test in 3.2.3.4. The exhaust temperature is increased as expected. The heat rate increases also slightly to indicate less efficiency.

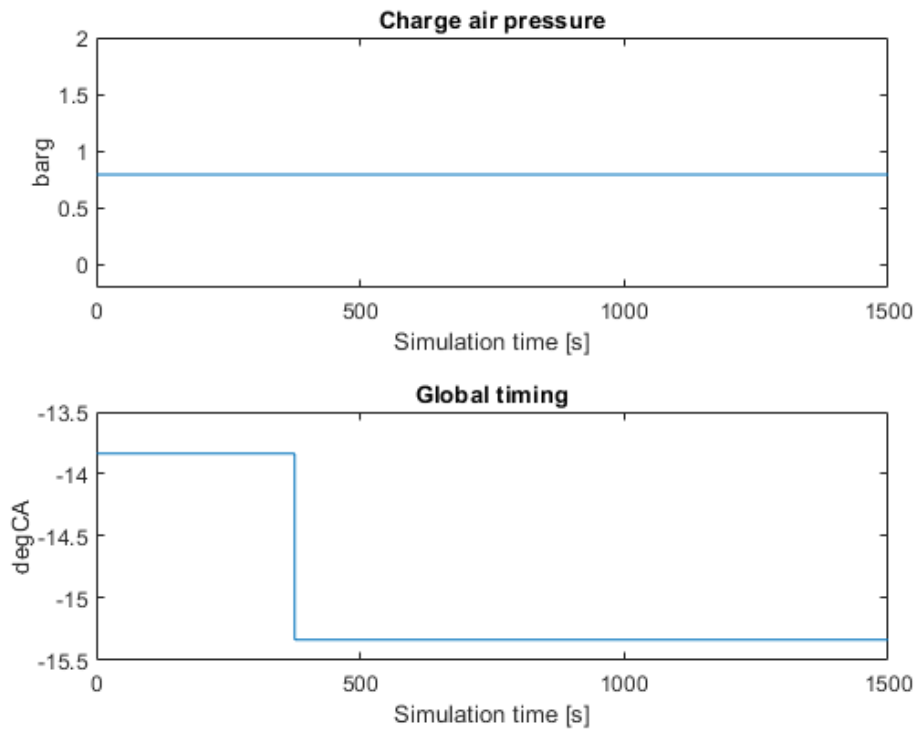


Figure 3-18: Open loop simulation with step change on global timing – earlier timing

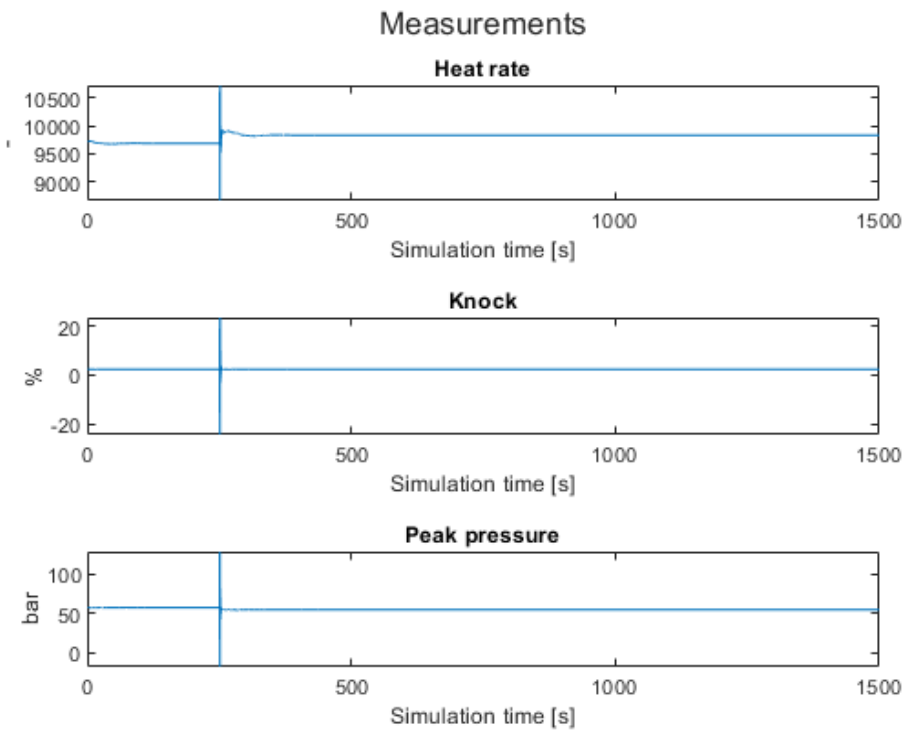


Figure 3-19: Open loop simulation with step change on global timing - later timing - model output

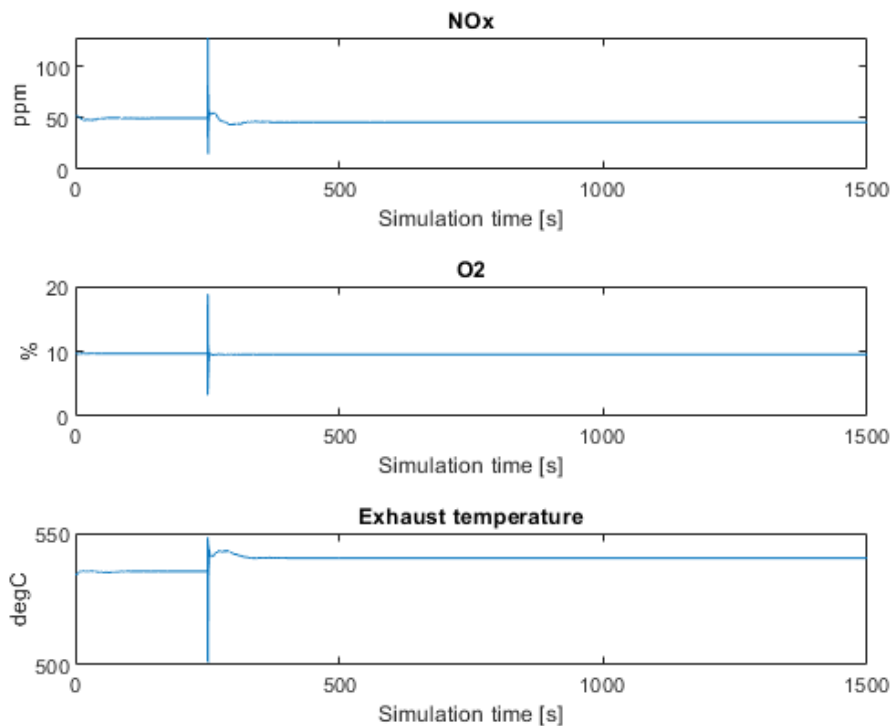


Figure 3-20: Open loop simulation with step change on global timing - later timing - model output

3.2.3.6 Open loop simulation with gradually increase in charge air pressure

This test is performed to show the resultant output of the model in case of a gradually increasing control signal with constant disturbances. The change is kept within the max/min limits of the control signal but given the disturbances the control signal end up outside normal operating range.

In this first test the charge air pressure is increased gradually from the initial value upwards with 1,5 [barg]. The control signal change can be seen in Figure 3-21.

The model outputs is shown in Figure 3-22 and Figure 3-23. The same applies here as for the step change, the expected O2 percentage increases by adding air to the process and the exhaust temperature drops. Here it drops to the nonphysical value close to 0 [°C] but the control signal reaches here nonphysical values compared to the input disturbances. The NOx value drops negative.

The model however does not go off to extreme values in any case as long as the charge air pressure control signal remains within expected working value.

The heat rate increases indicating poorer fuel efficiency when increasing the air excess and hence a leaner mixture is given.

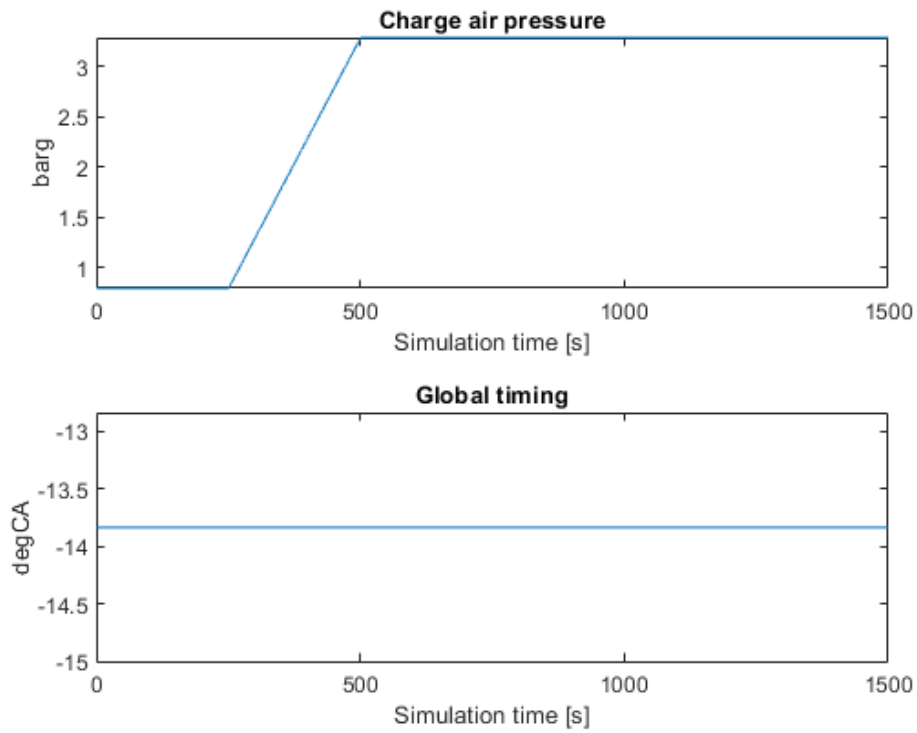


Figure 3-21: Open loop simulation with gradually increasing charge air pressure control signal

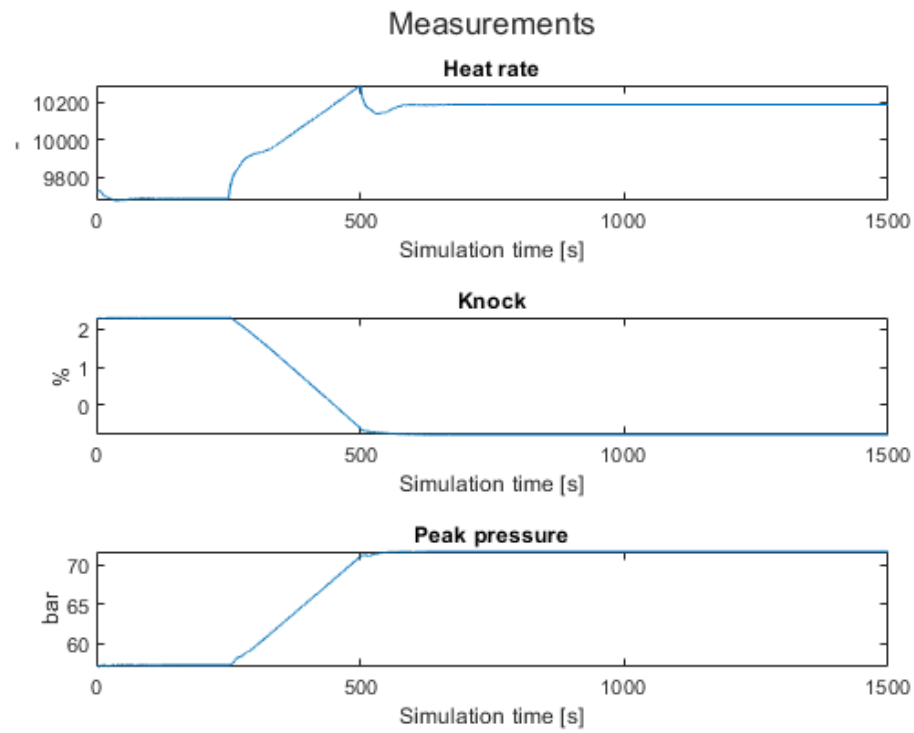


Figure 3-22: Open loop simulation with gradually increasing charge air pressure - model output

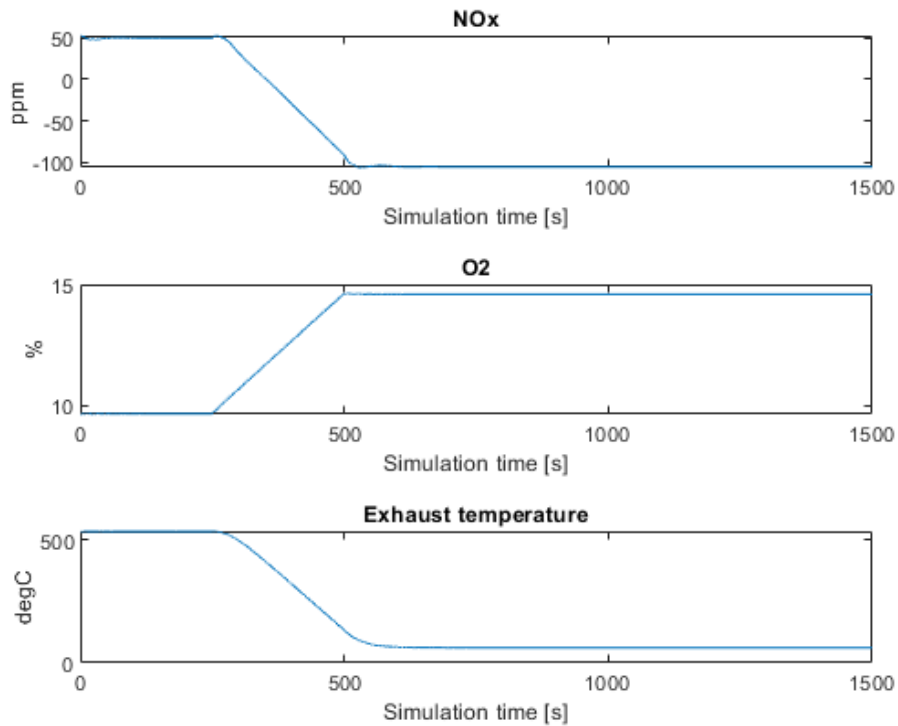


Figure 3-23: Open loop simulation with gradually increasing charge air pressure - model output

3.2.3.7 Open loop simulation with gradually decrease in charge air pressure

This test is performed to verify that the model does not converge to extreme values for any outputs given a gradual reduction in the charge air control signal. To keep things within physical meaning the charge air pressure is not allowed to go to a negative value.

The control signal is seen in Figure 3-24. The model outputs can be viewed Figure 3-25 and Figure 3-26.

The exhaust temperature increases as expected and the O2 level is reduced. The NOx is increases as expected with less lean mixture.

The effect on the heat rate is noted as it is reduced compared to the initial value – hence a less lean mixture improves the fuel efficiency.

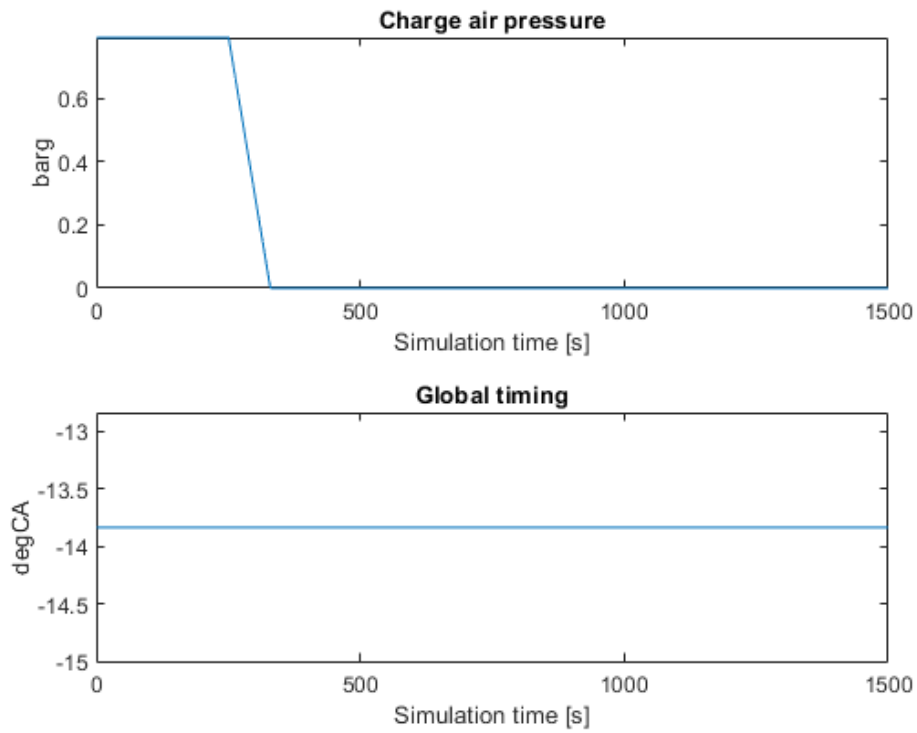


Figure 3-24: Open loop simulation with gradually decreasing charge air pressure control signal

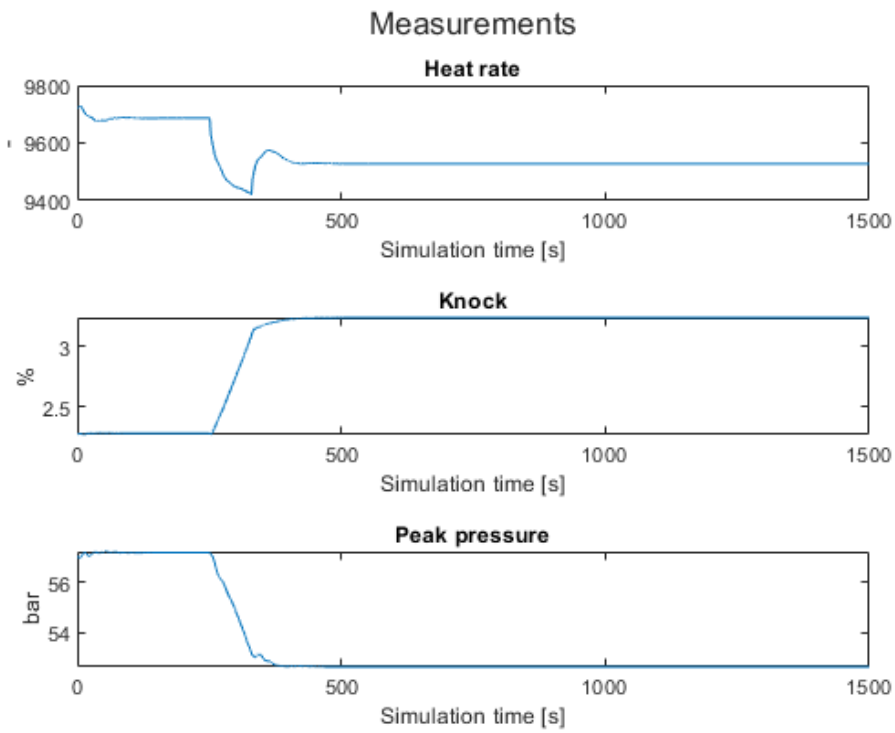


Figure 3-25: Open loop simulation with gradually decreasing charge air pressure - model output

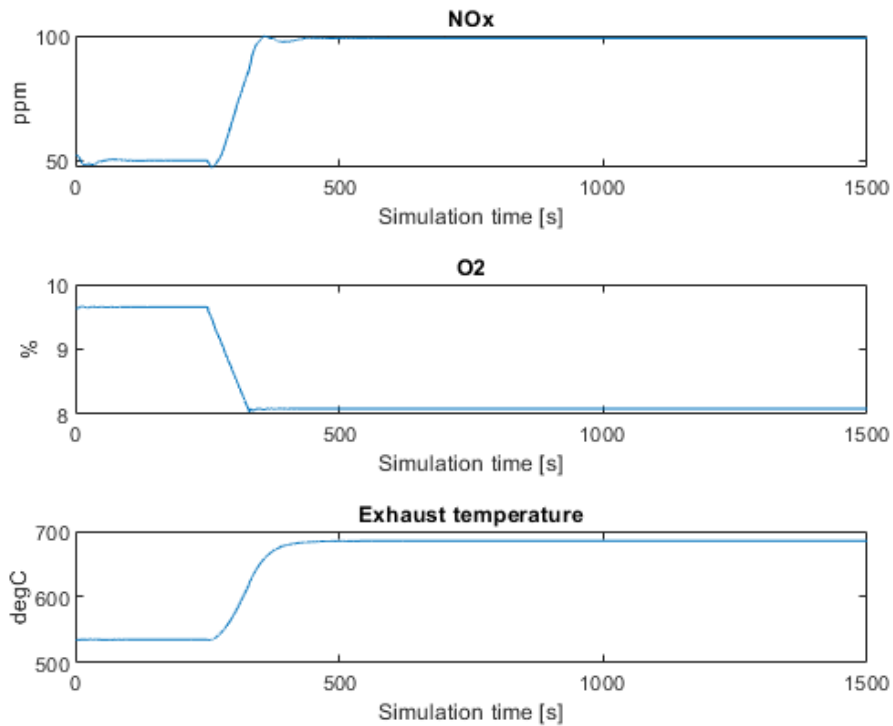


Figure 3-26: Open loop simulation with gradually decreasing charge air pressure - model output

3.2.3.8 Open loop simulation with gradually changing global timing – advance

Basically, the same test as done in section 3.2.3.4 but now with gradually earlier timing until 18 [degCA] before TDC. This time the timing is advanced further than during the step change and hence larger changes is seen on the outputs. The NOx increases and the peak pressure increases. The exhaust temperature is reduced as for the step response. The heat rate is also reduced indicating better efficiency.

The gradually changed control values is show in Figure 3-27 and the model output in Figure 3-28 and Figure 3-29.

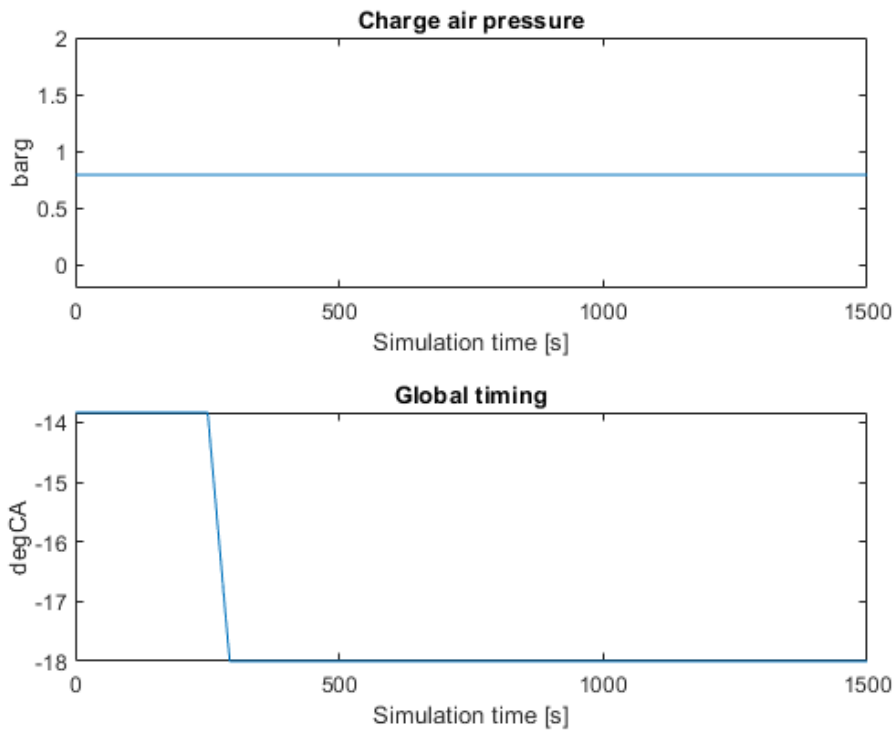


Figure 3-27: Open loop simulation with gradually changing the ignition timing to earlier timing

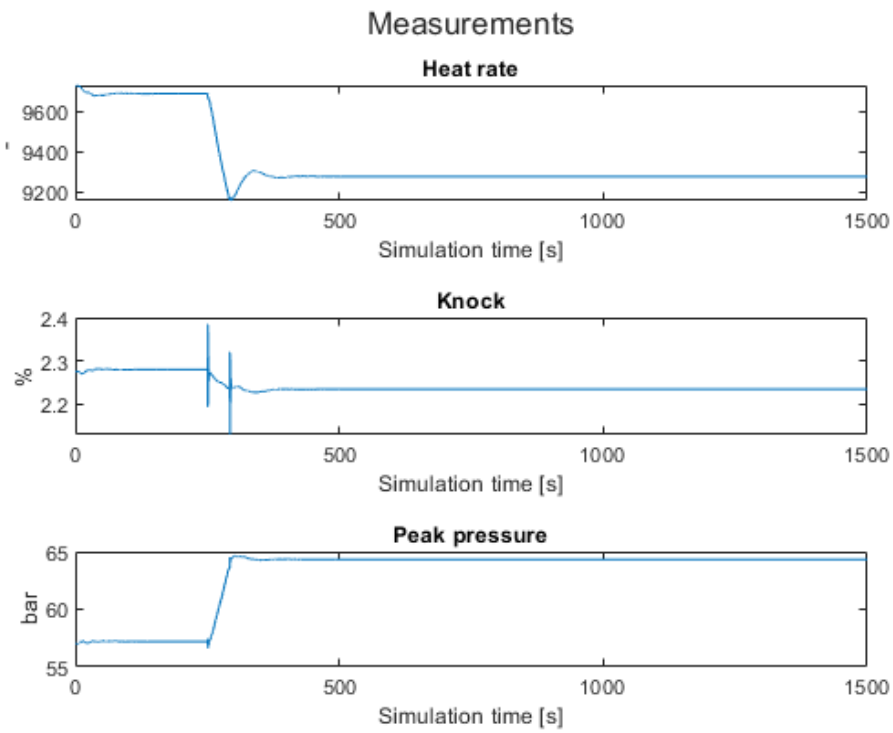


Figure 3-28: Open loop simulation with gradually changing ignition timing – earlier timing – model output

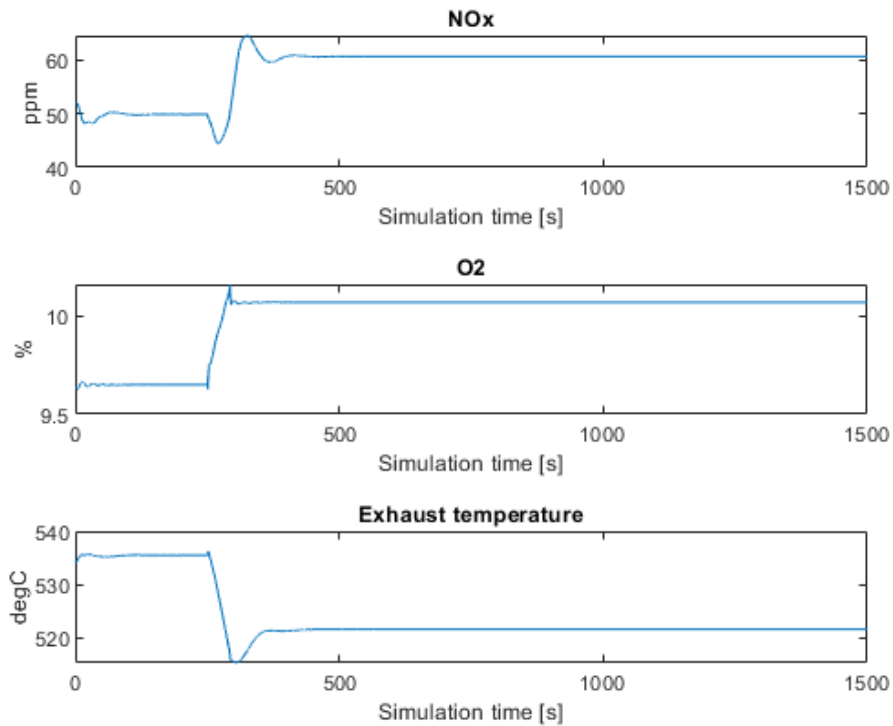


Figure 3-29: Open loop simulation with gradually changing ignition timing – earlier timing – model output

3.2.3.9 Open loop simulation with gradually changing global timing – retard

Test related to the test done in 3.2.3.5 but this time there is no step change in the control signal but rather a gradually changing signal. The change is here kept within normal operating ranges and hence later timing than 8 [degCA] before TDC is not performed.

The control signal change is shown in Figure 3-30. The outputs from the model is shown in Figure 3-31 and in Figure 3-32.

We here notice the same results as with the step change. Heat rate is increased, and the exhaust temperature increases. The NOx level is decreased. The results here are opposite to the results from the test in section 3.2.3.8 as expected.

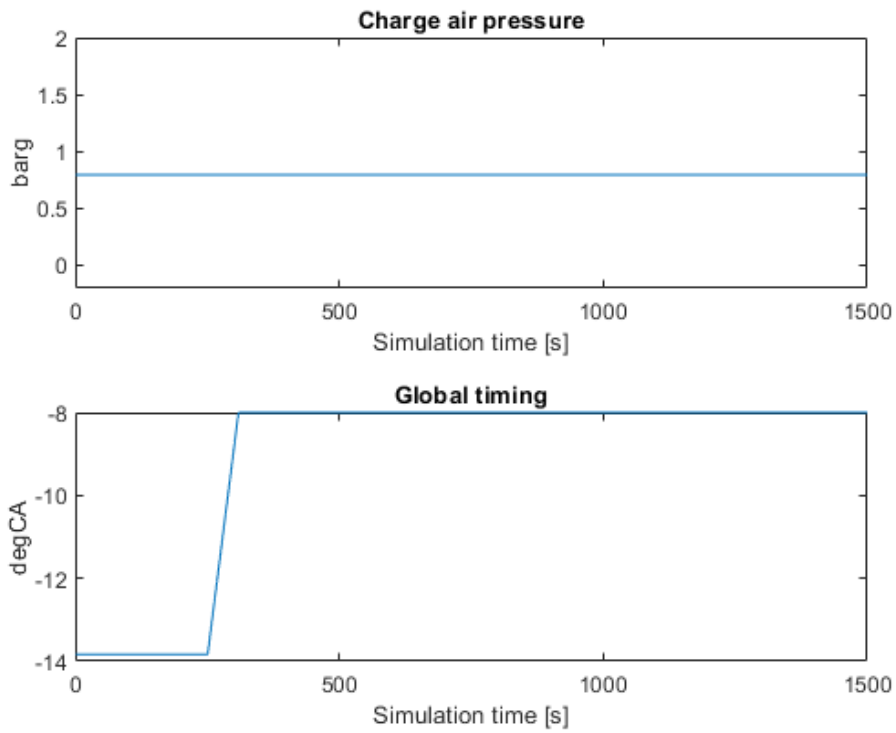


Figure 3-30: Open loop simulation with gradually changing the ignition timing to later timing

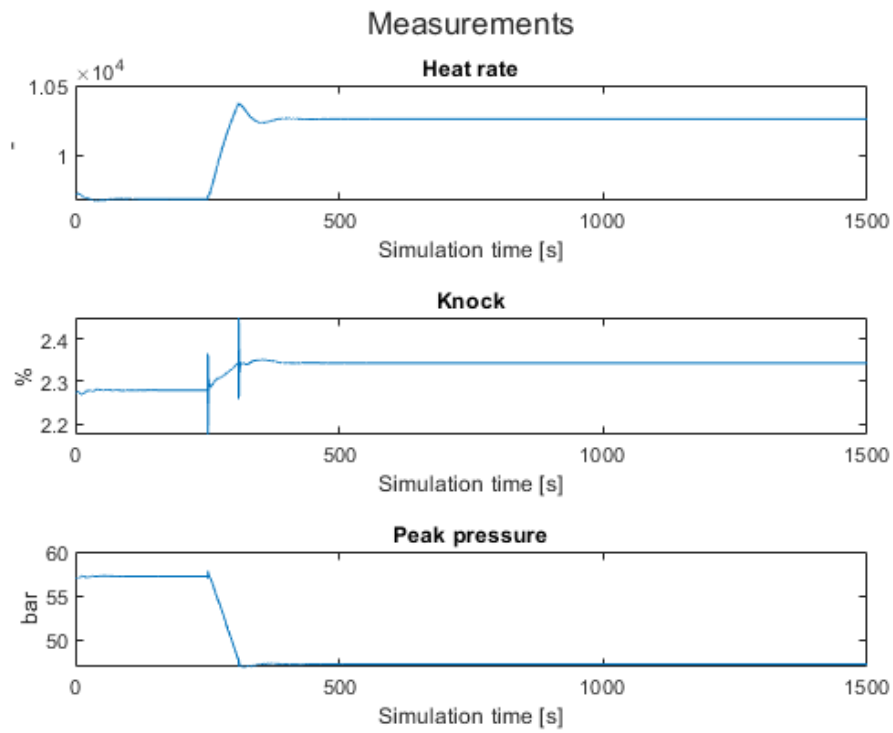


Figure 3-31: Open loop simulation with gradually changing ignition timing – later timing – model output

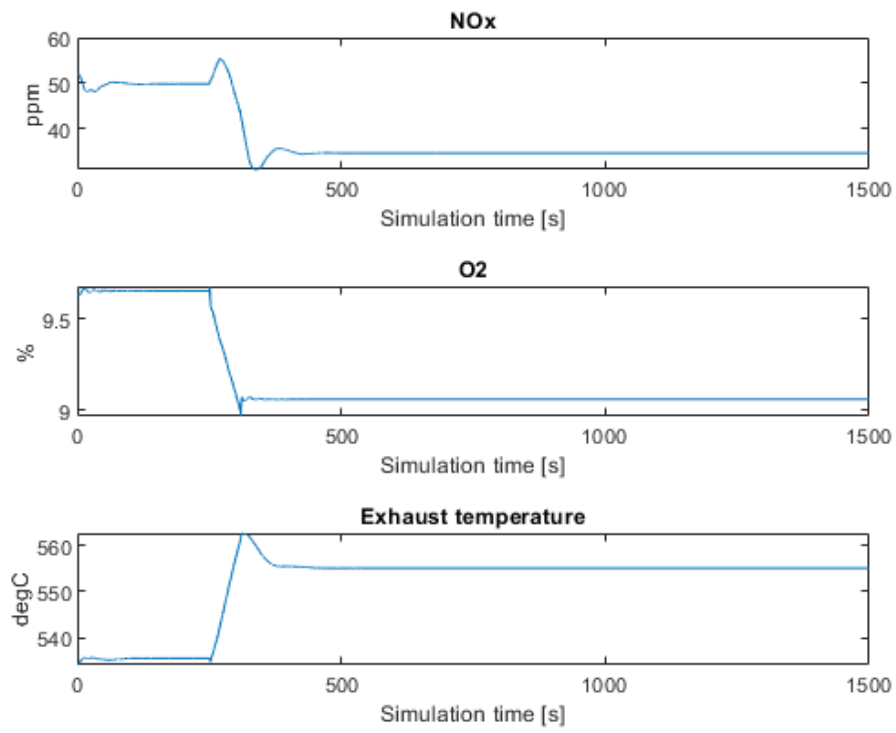


Figure 3-32: Open loop simulation with gradually changing ignition timing – later timing – model output

4 MPC

This section describes the development phase of the MPC controller, and the steps taken and results from several tests and analysis.

Firstly, a section describing the variables involved and their physical meaning are given. Then a section regarding the measurements and the various constraints are discussed.

4.1 Controllable inputs

There are defined two controllable signals in the project based on the project definition and the results from the system identification part done during the Mater Project. These signals will be used to adjust the process in order to achieve the optimization. The signals are defined to be *charge air pressure* and *global ignition timing*. These signals will not directly control the process values but be the optimal set points for the lower level PID controllers which in turn will adjust accordingly to achieve the optimal operation.

Since these two signals do have physical meaning there are some limitations that needs to be fulfilled for the system to have physical meaning.

The overall purpose of this controller is to optimize fuel efficiency over time and hence it is not the short-term benefit that is the most importance here. There is no optimal set point defined for the heat rate. The goal is to minimize the heat rate as much as possible by still staying within the constraints defined.

4.1.1 Charge air pressure

The charge air pressure is the pressure of the combustion air entering the combustion chamber from the air receiver. This pressure is controlled by adjusting the waste gate bypass valve such that the pressure is according to set point. Traditionally these set points are found based on testing on a real engine where the emissions are measured and the distance to the knocking limit is observed. In addition, the pressure is mapped towards the turbocharger to prevent any stalling or crossing the surge limit.

The charge air pressure set point is usually a map where the set point is based on the engine speed and the power output but biased from several sources to make it adaptable to ambient conditions and ageing. This charge air pressure set point is traditionally highly driven by the engine power output in an almost linear relation. Nominal charge air pressure at 100 [%] power output is approximately 4,2 [barg].

The physical constraints for this pressure will be imposed as bounds to the optimizer. By a defined upper limit for what is physical possible and at the same time set a lower bound close to 0 [barg]. Since the optimizer will not be working on 0 [barg] air pressure the lower bound will be set to 0,3 [barg] and the upper bound to 4.5 [barg] to give some regulation margin.

The charge air pressure cannot instantly change from one pressure to the next and hence the optimizer is limited based on rate of change of the control value such that it cannot change instantly. Nor can the waste gate valve change instantly and hence such limitations makes sense.

4.1.2 Global ignition timing

The global ignition timing command is used to set the base ignition timing for the engine. Each individual cylinder will adjust this base timing within a window of ± 3 [degCA] to balance the peak pressure off the cylinders.

The global base timing will influence the efficiency of the engine but also has an impact on the peak pressures, NOx generation, knocking and exhaust temperature.

By advancing the global timing the peak pressure increases and this needs to be within the design limit of the engine to prevent mechanical damage to it. If the ignition is retarded the exhaust temperature increases and the NOx emissions decrease as the combustion air temperature increases due to longer burn duration. By retarding the ignition timing the heat rate increases to indicate less efficiency.

Finding the optimal balance for the global timing is not straightforward as one would like to advance timing as far as possible, but not risking knocking to take place. The ignition timing is also dependent on the charge air pressure and the charge air temperature as these will influence the combustion air temperature.

The ignition base timing is traditionally found during testing and running the engine close to the knocking limit during controlled environments. It is however not given that the same conditions will be applicable on every project and hence margin must be added on the set point to take into account different fuel compositions and ambient conditions. Ageing is also a factor here.

To counteract these changing conditions several set point modifiers are in place which will bias the set point if a change in ignition knocking is detected or if exhaust temperature is increased. In addition, the location of the centre of combustion is measured based on the heat release curve from the combustion process. This location is used as a set point on a second level PID controller which biases the base set point so that this location is kept on set point as well. But none of these measures are there to optimize the fuel consumption over time and to take into account all these constraints as an MPC controller can do.

There are some physical known limitations that should be obeyed. The global ignition timing is seldom, if at all, below -8.5 [degCA] for a power plant running connected to the grid and producing power at nominal speed. It will also not be possible to advance the timing more than to -20 [degCA]. Nominal global ignition timing is usually in range of -12 [degCA] to -16 [degCA].

4.2 Measured disturbances

There are defined 3 main disturbances which are considered to have an impact on the system to such an extent that they need to be included. All 3 disturbances are measured directly by the control system. The most dominant are the engine power output. The measured value here is the IMEP which is the measured cylinder work done over the combustion cycle utilizing cylinder pressure sensors.

These disturbances are set as inputs to the model of the system under consideration.

4.2.1 Suction air temperature

This is slow varying input to the system which has the least impact. The suction air temperature is the air temperature measured at the inlet of the compressor part of the turbocharger. This disturbance will inform the system about the ambient conditions under which the engine is currently operating. The ambient temperature, and hence the suction air temperature will vary over a year for a given installation location. This varies might be small or large depending on the location. It might therefore have an impact in some cases and hence it is added here.

4.2.2 Charge air temperature

The charge air temperature is measured in the air receiver and is the temperature of the combustion air fed into the combustion chamber during the opening time of the inlet valve.

This temperature is in some projects actively regulated by a PID controller, while it in some installations are mechanically adjusted at max power output to give a certain temperature. Normal operational temperature here is around 50-55 [°C]. This might however change if the humid conditions are such that condensation might occur at this temperature.

The temperature of the charge air influences the air mass which is available to the combustion process and hence any change here will impact both NO_x emissions and the resilience towards knocking.

4.2.3 IMEP

IMEP (Indicated Mean Effective Pressure) is measured directly by the cylinder pressure sensors. The highest and lowest values are removed and the average over the number of cylinders is taken and fed into a moving average filter over 100 cycles. This final value indicates the current loading (power output) of the engine. This value is used by the model as the major measured disturbance. Most engine dynamics can be explained by this single signal.

4.3 Measured outputs

The following list summarizes and shortly explains the measured outputs which are used by optimizer as constraints.

4.3.1 Heat rate

This is the value that is to be minimized. It is indicative of the relation between the power output and the fuel consumption estimation. The heat rate is given as the relation between the IMEP and the total heat release. The IMEP is measured by the cylinder pressure sensors as well as the total cumulative heat release. The heat release is given as [kJ/cycle] and is estimated based on the pressure rise curve measured by the cylinder pressure sensors.

During the set point tracking testing this value is used as the feedback from the process.

4.3.2 Knock level

Knock level is also known as engine detonation and is when the combustion takes place prematurely in part of the compressed air fuel mixture in the cylinder. This knocking can cause severe damage to the engine if not responded to early because of high frequency pressure waves causing very high cylinder pressures potentially above the design limit of the engine.

The engines are constantly pushed towards the knocking limit as this area produces better fuel efficiency at higher power outputs. Knocking might occur if the air fuel mixture is not correct or substances such as oil leaks into the combustion chamber causing changes to the burn rate of the air/fuel mixture.

Each cylinder is monitored for knock level and any increase in knocking results in that cylinder ignition timing being retarded for some time. If several cylinders experience knocking the global timing point is usually retarded to avoid any further increase into non-operational areas.

The knock value is measured by looking at the ripples on the cylinder pressure curve after the ignition location. This value indicates the level of knocking for each cylinder but is averaged for all cylinders here. This will in general only pickup up globally severe knocking. Knocking can also be reduced by lowering engine power output or increasing the amount of air in the air/fuel mixture resulting in a leaner mixture. That will however impact efficiency.

The value here is included as constraint at it is a limiting factor for advancing the timing too much or reducing the charge air pressure too much. Typically, the engine is shutdown with a value over 30 [%]. This will be initially kept as constraint in this application.

4.3.3 Peak pressure

The peak pressures are measured from cycle to cycle and is the highest measured pressure in the combustion chamber of the combustion cycle. The peak pressure is a value which must be limited as there are design limitations on the engine for how high pressures the internal components can withstand without damage.

The nominal peak pressures during 100 [%] power output at nominal speed is usually around 175 [bar]. The control system has alarming and shutdown conditions if sustain operation

around 200 [bar] is experienced and hence the optimizer should avoid operation above 200 [bar], and preferably limit operation to 180 [bar] but with some slack.

4.3.4 NO_x

The NO_x emissions are measured in the exhaust outlet after the turbocharger. The emissions are measured with sensor from Continental most commonly used on trucks and cars. This sensor gives the wet NO_x values in ppm directly and is used in a closed loop regulation for controlling the level of NO_x to a given set point.

All power plant engines are sold with a given NO_x emission set point at MCR⁹. More information about the NO_x values are given in section 1.3.2.4.

The NO_x values are good indications of how rich or lean the fuel mixture in the combustion is. A high NO_x value indicates a rich mixture and vice versa. The NO_x value is very sensitive to these variations and will rapidly increase in case of the charge air pressure is reduced.

It should however be noted that the NO_x values should rarely be seen drifting high during steady operation. During transients a change in NO_x value is expected as the engine increases the air pressure during the transient to get better margins to the knock limit.

The NO_x value would be used as a constraint during the optimization phase such that it can stabilize below at least 150 [ppm].

4.3.5 O₂

The same sensors that measures the NO_x level in exhaust will also measure the O₂ level. In the traditional engine controller, the O₂ percentage is used actively for engine limitation. That is if the O₂ level becomes too low, which indicates a too rich mixture, the engine will limit the fuel admittance and hence reduce power output.

This value is rather critical and hence strict constraints on the low level is too be used.

On the higher level the engine risk misfire in case the value becomes too high indicating lean mixture. A softer constraint here should be imposed preventing the optimizer from going into too lean conditions.

During normal operational conditions the O₂ percentage is somewhere between 8,5 [%] and 12,5 [%], with nominal condition a approx. 9.5 [%]. This lower limit is also handled externally to the MPC as it is a critical condition. The optimization should be rather strict on the lower limit while the upper limit can be broken during given condition. The aim should however be to stay within the limits of 8,5 [%] and 12,5 [%].

4.3.6 Exhaust temperature

Traditionally the exhaust temperature outlet from each cylinder has been used to balance the engine power output from each cylinder. Before the cylinder pressure sensor era the only

⁹ MCR is Maximum continues rating of the engine, that is the max power output.

possibility to check how much each cylinder contributed to the power output was by looking at the deviation in exhaust temperature between the cylinders.

The exhaust temperature used here is the temperature measured in the collecting pipe just prior to the turbine part of the turbocharger. This exhaust will therefore be an indicative of the all cylinders on that pipe collectively.

There are limitations from the turbocharger supplier on the max inlet temperature of turbine and hence these needs to be obeyed. The temperature will increase in case the power output increases and mixture becomes too rich. It is therefore very dependent on the charge air pressure, but also the ignition timing. In case the ignition timing is retarded the temperature will increase and hence this needs to be handled.

The constraints can here be set based on normal operational conditions where the exhaust temperature before the turbocharger turbine should not exceed 600 [°C].

4.4 Constraints

This section looks into the definition of the constraints what we want to impose on the MPC controller based on the engineering knowledge of the valid operational window. These are here given in a table format. They are more discussed in section 0.

In Table 1 the constraints on the measured outputs are given. For “Heat rate” no limits are given as this the signal to optimize.

Table 1: Output constraints

SIGNAL	UNIT	LOWER LIMIT	UPPER LIMIT
Peak pressure	bar	0	180
Knock level	%	0	30
Heat rate	-		
Exhaust temperature before turbine	°C	0	600
NOx	ppm	0	150
O2	%	8,5	12,5

In Table 2 the constraints for the controllable input signals are shown. There are constraints both on the absolute value and the rate of change.

Table 2: Input constraints

SIGNAL	UNIT	LOWER LIMIT	UPPER LIMIT
Charge air pressure	barg	1	4.5
Global timing	degCA	-20	-8.5
Charge air pressure	barg/s	-0.2	0.2
Global timing	degCA/s	-0.3	0.5

4.5 qpOASES

The first optimizer that is tested is the qpOASES. The qpOASES is an open source implementation of the online active set strategy [14]. It is inspired by parametric quadratic programming and is based on the assumption that the optimal active set does not change much from one quadratic program to the next. The qpOASES solves the QP of the following form,

$$\begin{aligned}
 &\text{Objective function} && \min_x \quad J = \frac{1}{2}x^T Hx + c^T x \\
 & && \text{s. t.} \\
 &\text{Linear inequality/equality constraints} && b_L \leq A_e x \leq b_U \\
 &\text{Bounds} && x_L \leq x \leq x_U
 \end{aligned} \tag{4-1}$$

where the Hessian matrix is symmetric and positive semi-definite. Both the linear equality and inequality constraints uses the same notation. For inequality constraints the b_U and b_L is different while for equality constraints they are equal. So for equality constraints we have that

$$\begin{aligned}
 b_L &= b_U = b_e \\
 A_e x &= b_e
 \end{aligned} \tag{4-2}$$

4.5.1 Set point tracking with bounds on control value

For the set point tracking with fixed disturbances and upper and lower bounds on the control value the decision variable is defined as

$$z = \begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix} \tag{4-3}$$

Based on a prediction horizon of N_p the total number unknowns are given as

$$n_z = N_p(n_u + n_x + n_y + n_e) \tag{4-4}$$

The standard QP objective function can be set up as:

$$J = \frac{1}{2} \begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix}^T \begin{bmatrix} H_{11} & 0 & 0 & 0 \\ 0 & H_{22} & 0 & 0 \\ 0 & 0 & H_{33} & 0 \\ 0 & 0 & 0 & H_{44} \end{bmatrix} \begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}^T \begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix} \tag{4-5}$$

We note that each element in the decision variable is in itself a vector of elements equal to the prediction horizon N_p such that for example for u the vector looks like

$$u = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_p-1} \end{bmatrix} \quad (4-6)$$

Each block in the Hessian matrix is now the weighting matrix for each variable in the decision vector where we set the element for each step of the prediction horizon equal to each other such that for the decision variable u we have that

$$H_{11} = \begin{bmatrix} P_0 & 0 & \dots & 0 \\ 0 & P_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P_{N_p-1} \end{bmatrix} \quad (4-7)$$

where $P_0 = P_1 \dots \dots = P_{N_p-1}$

In this project there are 2 controllable outputs and 1 set point to track and there are 3 measured disturbances and 5 additional outputs to be used as constraints. So, in this case the decision variable u contains to values for each time step in the prediction horizon as

$$u = \begin{bmatrix} u_0^1 \\ u_0^2 \\ u_1^1 \\ u_1^2 \\ \vdots \\ u_{N_p-1}^1 \\ u_{N_p-1}^2 \end{bmatrix} \quad (4-8)$$

and

$$H_{11} = I_{N_p} \otimes P_{n_u \times n_u}$$

where \otimes is the Kronecker product.

A simple test is performed with a set point tracking case. In this test the set point is firstly changed in a step, then there is a step change in the disturbance *IMEP*.

The control output response is shown in Figure 4-1. This time the *Charge air pressure* stay well below the upper bound of 4,5 [barg] while the *Global timing* goes straight to the upper bound of -8.5 [degCA].

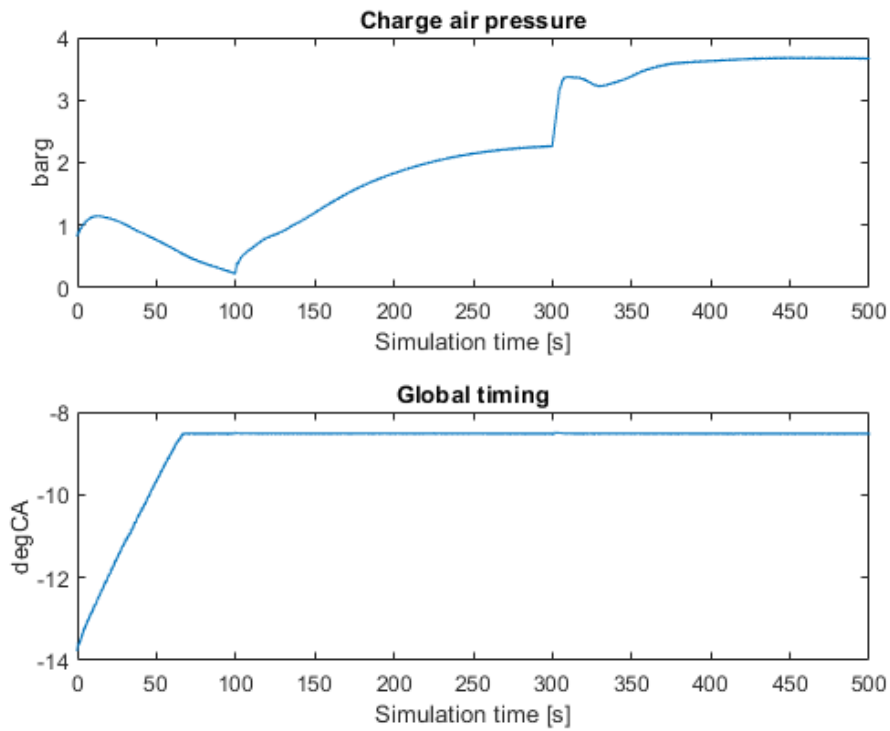


Figure 4-1: Set point tracking - qpOASES - control outputs

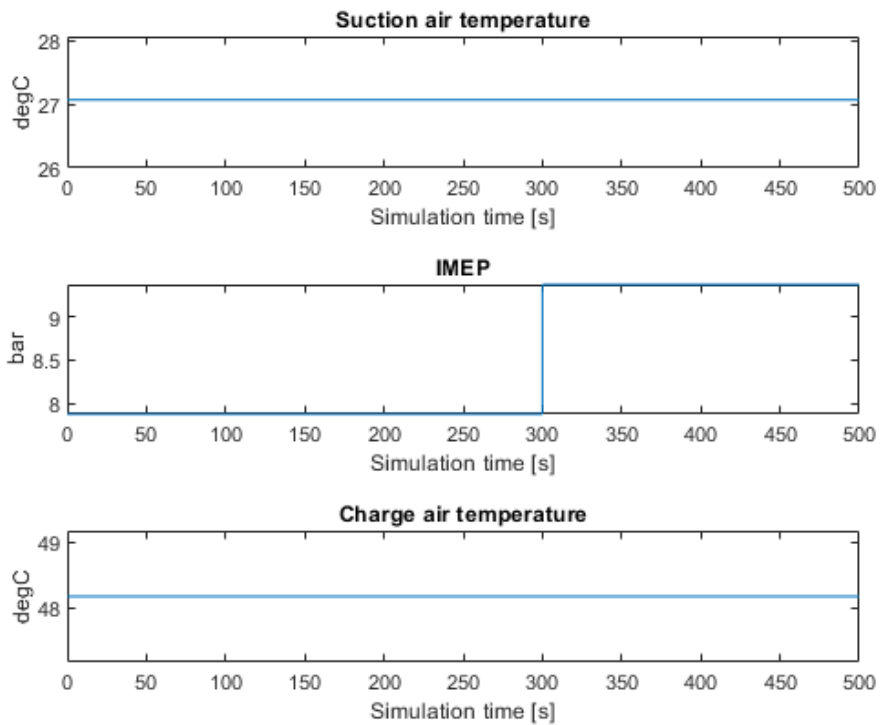


Figure 4-2: Set point tracking - qpOASES - disturbances

Figure 4-2 shows the disturbances which are kept constant except for the step in *IMEP* at 300 seconds. The response in the *Charge air pressure* is clearly visible at this point from Figure 4-1.

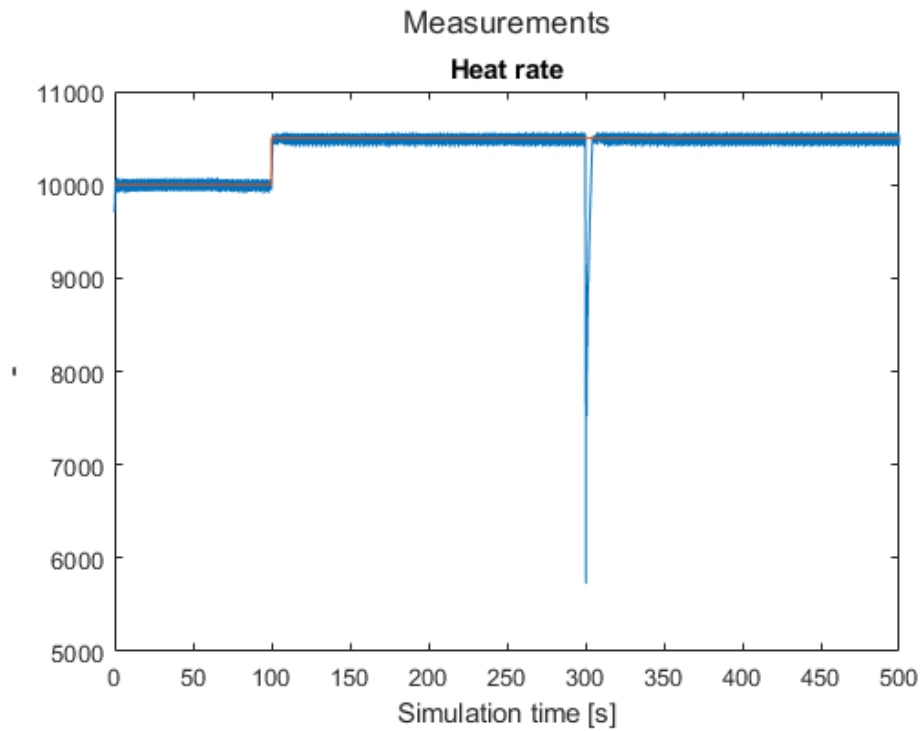


Figure 4-3: Set point tracking - qpOASES - heat rate and set point

In Figure 4-3 the heat rate and the step change in the set point at 100 seconds is shown. The heat rate tracks the set point well and returns quickly to the set point after the disturbance to the *IMEP*.

4.5.3 Status of qpOASES

The qpOASES was tested with a set up in SIMULINK with MATLAB functions for declaring and incorporating the functions. Running the simulations is a time-consuming task and internally in the project it was suggested quickly to try the Quadprog and FMINCON approaches. Quadprog might be a quicker optimizer and hence should be tested. On the other hand, if non-linear model could be the outcome it was suggested to start to test with FMINCON as it is much more versatile in the structure it accepts, and it also accepts non-linear models and constraints.

The qpOASES was therefore quickly discontinued for further development at this stage and the focus was shifted to the FMINCON for further development.

4.6 Quadprog

During the initial phases of this project the Quadprog in MATLAB was also tested to solve the QP problem. This is a built-in function in MATLAB from the optimization toolbox which can use different routines to solve a QP problem. The Quadprog accepts the standard form of QP problem as used for the qpOASES in section 4.5.

$$\begin{array}{ll} \text{Objective function} & \min_x J = \frac{1}{2}x^T Hx + c^T x \\ & s. t. \\ \text{Linear equality constraints} & A_e x = b_e \\ \text{Linear inequality constraints} & A_i x \leq b_i \\ \text{Bounds} & x_L \leq x \leq x_U \end{array} \quad (4-9)$$

During the testing the set point tracking problem was used to test the model. Bounds on the upper and lower limit of the control signal was in place. Much of the same code as was used for the qpOASES was re-used for this testing as the structure is basically the same. It was also initially tested in SIMULINK, but this was quickly revoked to be done internally in MATLAB to try to speed up the simulation a bit and not needing to declare the function as extrinsic.

4.6.1 Set point tracking with bounds on control value

A short set point tracking test with the Quadprog is shown in Figure 4-5 - Figure 4-7. The control outputs in Figure 4-5 shows that the optimization is not as “clean” as it was for the qpOASES and the control outputs goes to its outer constraints rather quickly. Just before the 70 second mark it fails completely.

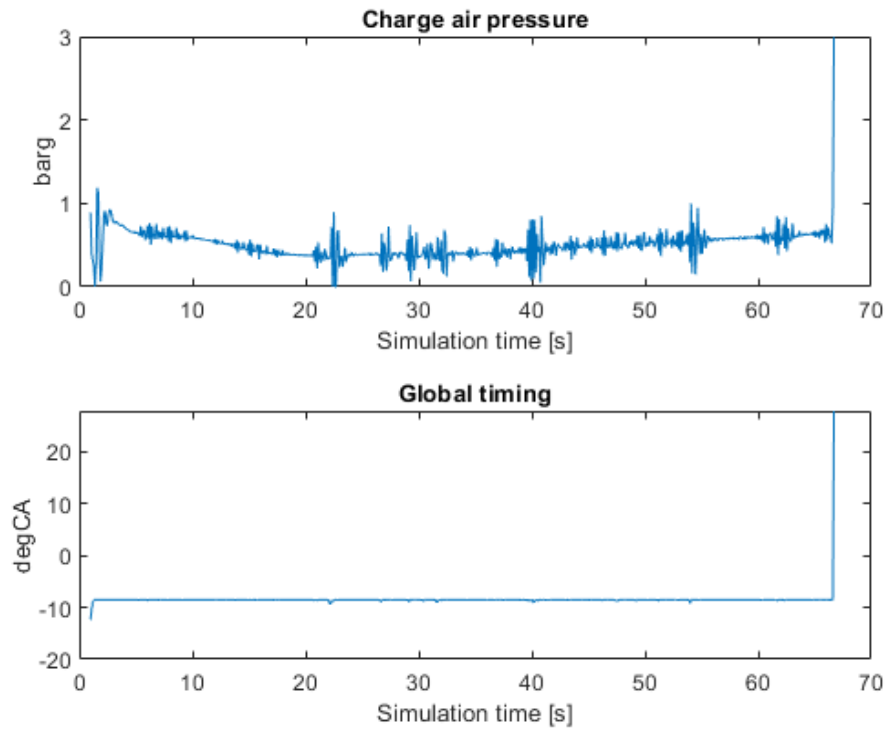


Figure 4-5: Set point tracking with Quadprog - Control inputs

The measurements and the set point is shown in Figure 4-6 and Figure 4-7. These shows that the set point is tracked but there are some sudden changes to the outputs at sporadic times.

The set point tracking here was tested with the “interior point” method as it contained both bounds and inequality constraints.

The solving of the optimization problem was not timed, but it took 6+ hours to solve for 500 samples which is far more than the qpOASES used for the a much larger problem.

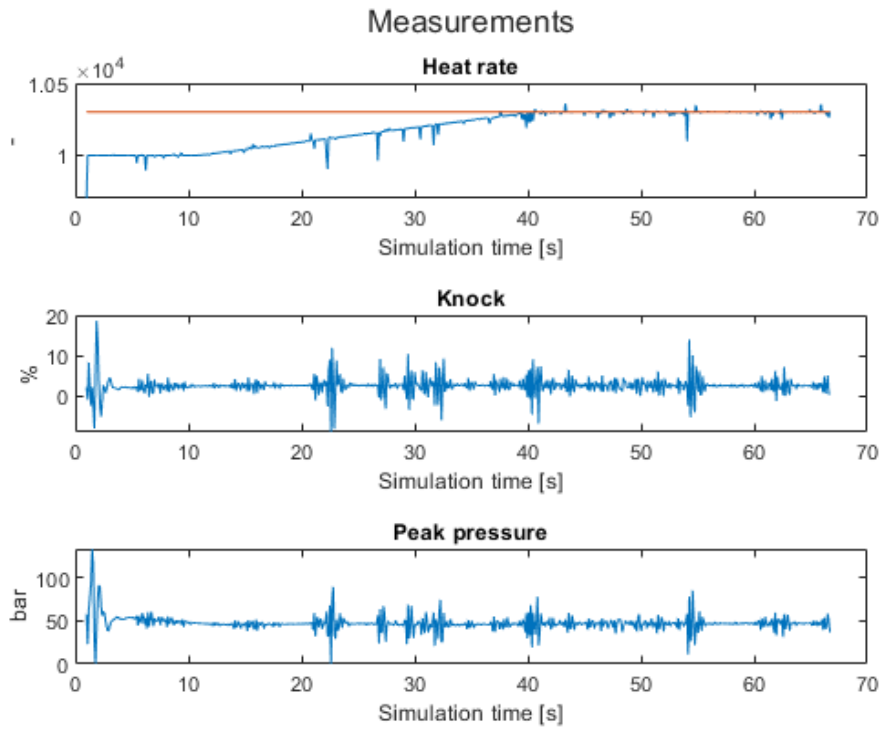


Figure 4-6: Set point tracking with Quadprog – Measured outputs

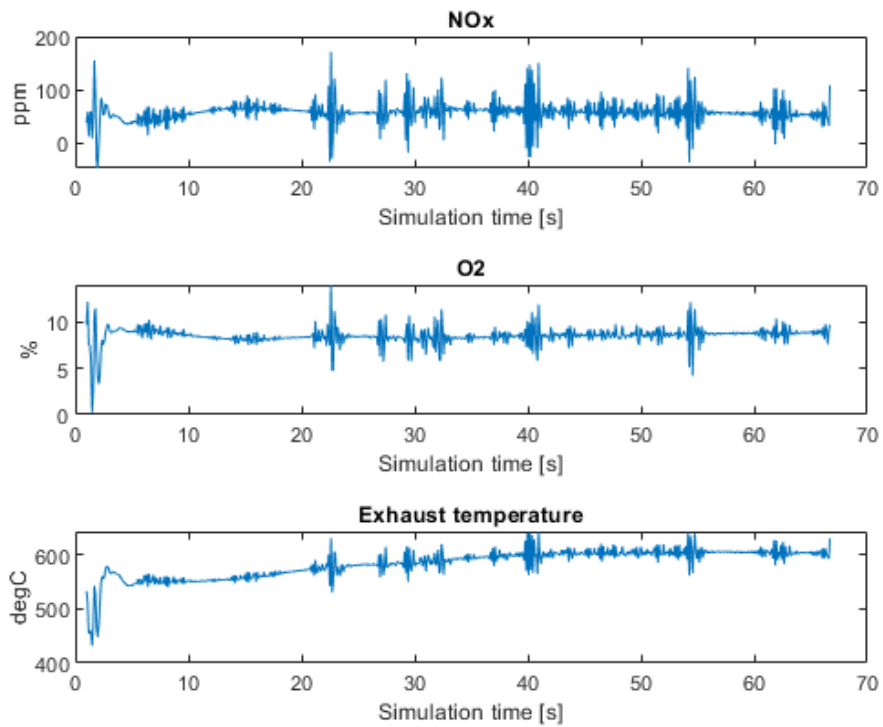


Figure 4-7: Set point tracking with Quadprog – Measured outputs

4.6.2 Code for Quadprog

The code used with the Quadprog is very much the same as used for the qpOASES in section 4.5, but this time all the simulation is done outside of the SIMULINK environment any only MATLAB scripts is used. Some minor changes to the setup of the matrices as the Quadprog accepts for instance the Hessian as a matrix and not only in vector form as the qpOASES.

The code can be viewed the Appendix G

4.6.3 Status of Quadprog

The testing with the Quadprog became very limited as the shift in focus towards the Fmincon was established at the same time as the Quadprog was tested. During the first testing phases infeasibility was often the case during simulation. The first set of testing was done using the polynomial model converted to a state space model, but even with the updated state space model the results were not always feasible even with only constraints on the control inputs signals.

The simulation time running the Quadprog was even for short simulation time (500 samples) and prediction horizon of 100 samples a very time-consuming task. This could have been improved by minimizing the decision variables for example by grouping. This was not prioritized at the current stage and the Quadprog solution was not explored further.

4.7 Fmincon

The Fmincon is a function from the MATLAB Optimization Toolbox which can be used to solve nonlinear constrained multivariable optimization problems. This function has been used for testing as it can handle both non-linear and linear functions. Most of the work in this project is done using the Fmincon and substantial time has been spent testing various variations of settings and set ups. Most of the results are presented in the following sections. It should be noted that many more simulations have been tested which has failed during testing as parameter extremes has been tested. The simulation part with Fmincon is also a time consuming task, but not the same extent as qpOASES and Quadprog, but still hours and hours has been spent tweaking the set up.

4.7.1 Set point tracking – unconstrained

These tests are to verify the MPC based on Fmincon to work as a set point tracker without any constraints on any outputs. The control inputs have their upper and lower bounds set so that large changes are possible and hence the physical limitations are not imposed on the control signals. The set point is set on the heat rate as this is the value of interest for the project. The disturbances used are based on measured real-life data in some of the tests and on fixed values during some tests. This is specified for each test.

The objective function is set to minimize the error on between the set point and the measured value and to minimize the control output. The general formula is given in equation (4-10).

$$\min_{(u)} J = \frac{1}{2} \sum_{k=1}^N (e_k^T Q_k e_k + \Delta u_k^T P_k \Delta u_k) \quad (4-10)$$

In this example e is the error between the set point and the reference and value and the minimization is also in the rate of change of the control values.

4.7.1.1 Set point tracking with step change in set point

The first tests are to check the response to step change in set point. Firstly, the initial set point is set to 11000 as this is slightly above the initial value of the heat rate. We can then see if the controller stabilizes before doing the actual set point change after 300 seconds. This will be a step change which is nonphysical operation but will still show if the set point tracking feature of the MPC works in theory.

We see in Figure 4-8 the two control values from the set point tracking. These shows that the global timing control value quickly goes to its maximum constraint value while the charge air pressure stays close to its minimum constraint until the step change at 300 seconds.

The set point is firstly set to 11000 and then changed to 12000 after 300 seconds. During this test the prediction horizon was set to 50 samples and the simulation time to 500 seconds. The optimizer algorithm used was the “interior point” with max 1000 iterations and max 1000 function calls.

The bounds on the control signals are set wide such that

$0,3 \leq \text{charge air pressure} \leq 8$
 $-20 \leq \text{global timing} \leq 0$

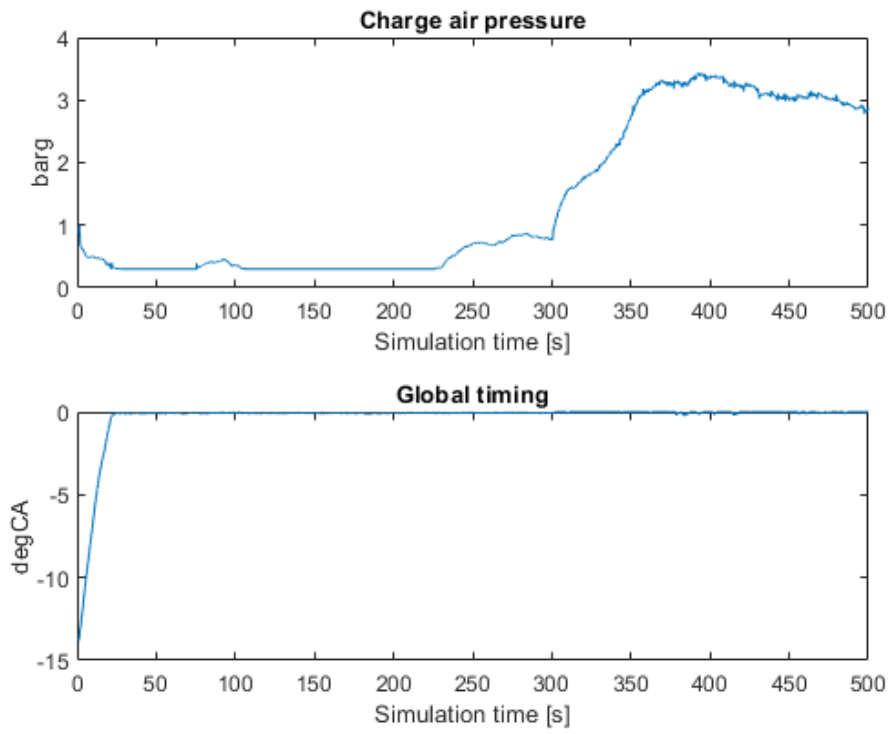


Figure 4-8: Set point tracking - control values

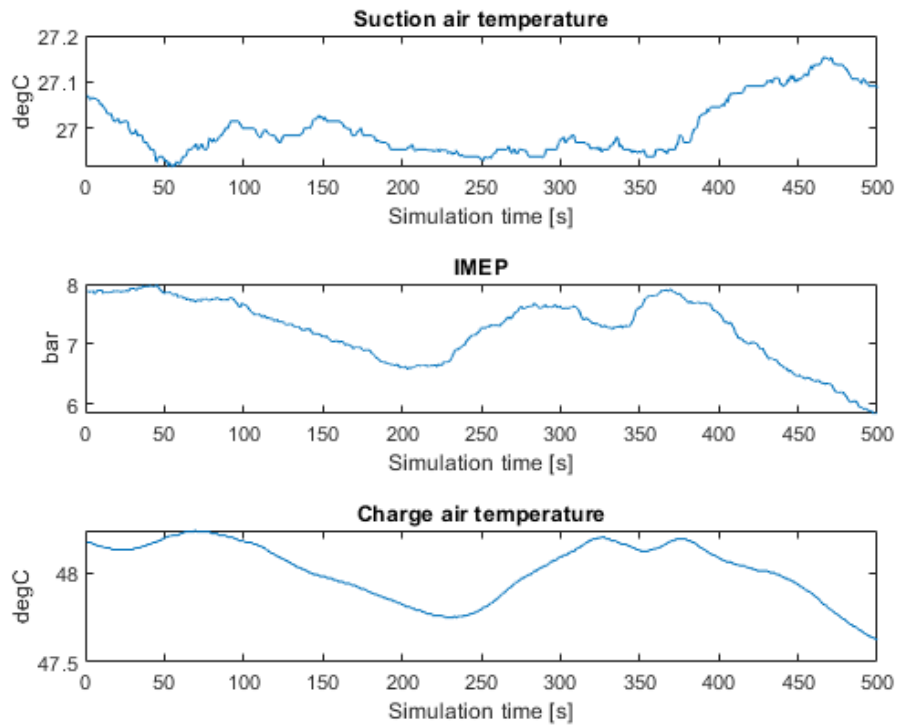


Figure 4-9: Set point tracking – disturbances

In Figure 4-9 we see the 3 disturbances over the 500 second period. Based on the physical nature these are only slightly varying over this period. For the suction air temperature, the change is about 0,2 [°C], the IMEP is varying with about 2 bar and the charge air temperature about 1 [°C].

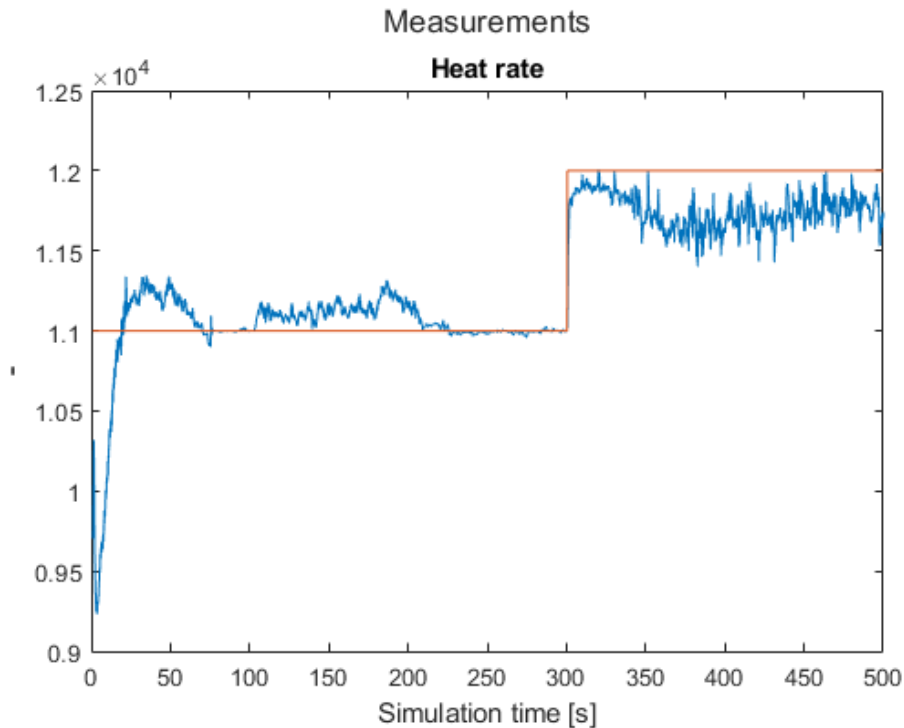


Figure 4-10: Set point tracking - heat rate output

We can see from the output plot in Figure 4-10 that the heat rate quickly gets to the set point of 11000 and quickly changes towards 12000 when the set point changes at 300 second. We observe that the control value *global timing* goes to upper bound quickly and stay there in Figure 4-8. This value is now abnormally high and only allowed to go to this point in order to show the set point tracking feature. But even with these outer limitations we did not get completely to the set point for 12000 for the heat rate. We also note that the heat rate signal is not clean. To exclude the disturbances as the cause of this “noise” the simulation is repeated with only the set point changing.

With fixed disturbances and only a set point change the model produces the results shown in Figure 4-12 and Figure 4-11. Here we see that now the set point tracking is completely stable until 300 seconds and tracks well during the step change.

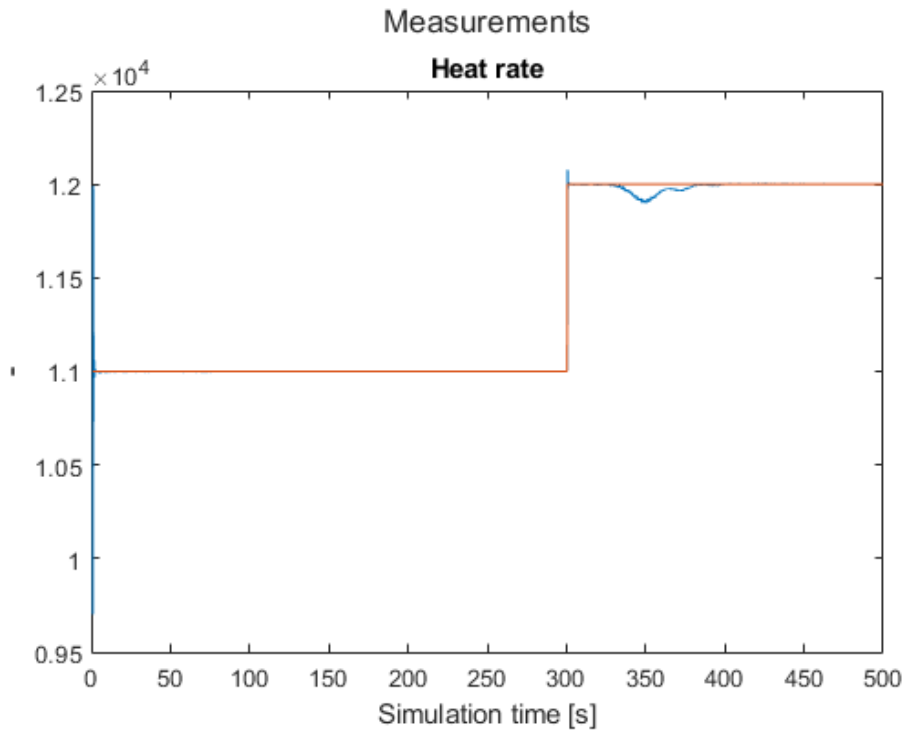


Figure 4-11: Set point tracking with fixed disturbances and set point change – heat rate and set point

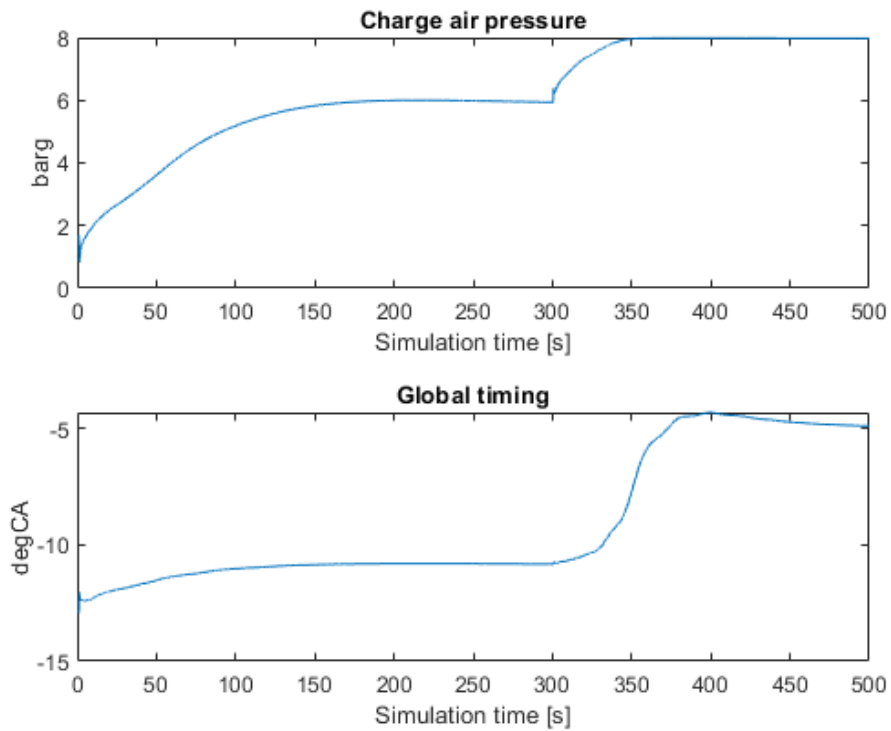


Figure 4-12: Set point tracking with fixed disturbances and set point change – control inputs

During this test we reach the limit on the charge air control signal after the step change resulting in the dip in heat rate. Here we see that the global timing is then used to get the heat rate back to set point. These values are not outside what we consider normal operating window and hence the choice of set point might not have been ideal in this case.

4.7.1.2 Set point tracking with fixed set point and disturbance rejection

If we look at the impact of a major disturbance on the engine power output, but keep the set point stable and the other two disturbances stable we get the results as in Figure 4-13, Figure 4-14 and Figure 4-15. Here we see that a step change is added to the IMEP at 300 seconds. The heat rate has a quick error before it stabilizes back to set point. Both control outputs adjust rapidly to compensate. The control outputs do however not physically have the possibility to adjust at such speed and hence limitations on the control value rate of change will need to be added.

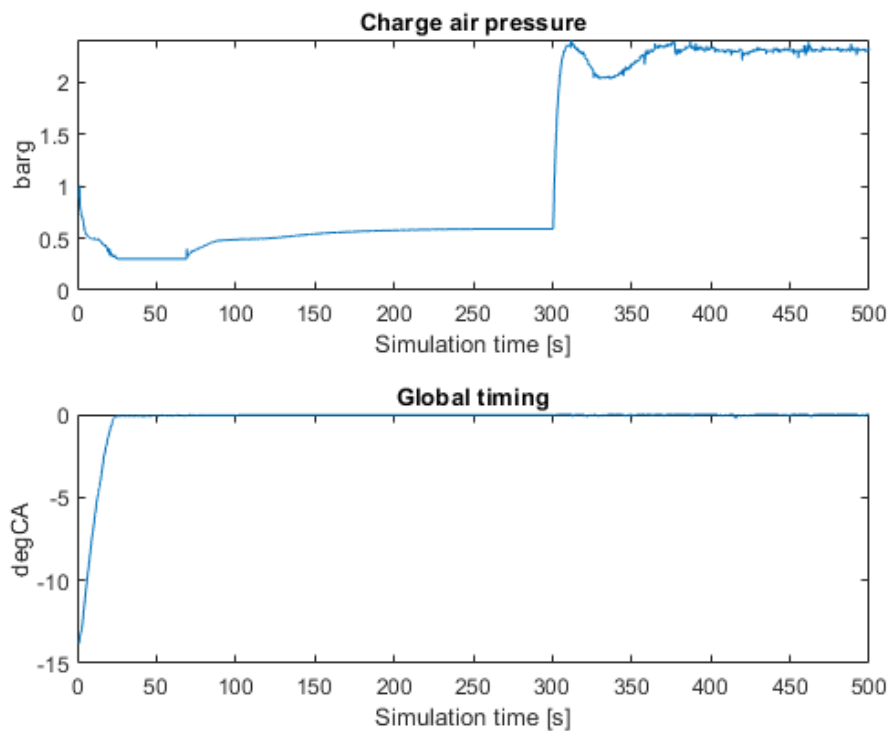


Figure 4-13: Set point tracking with disturbance rejection - control inputs

We note from Figure 4-13 that the control signal of global timing quickly reaches it the maximum value while the charge air pressure stays within bounds.

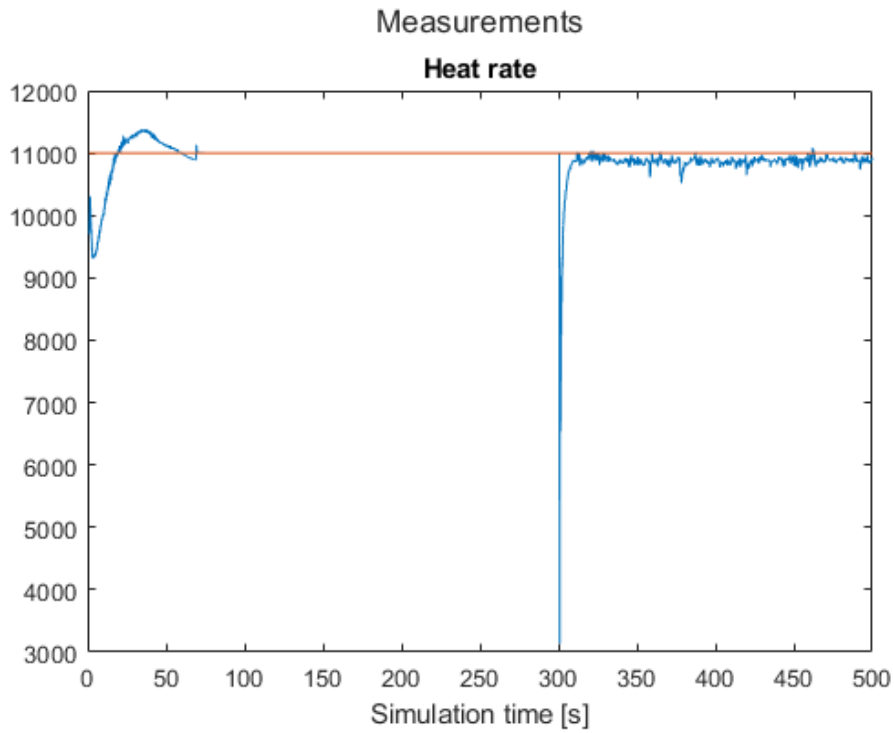


Figure 4-14: Set point tracking with disturbance rejection - measured output

We note that once again we are not able to keep the set point after the step change with current values even though the charge air pressure control signal doesn't reach its maximum value.

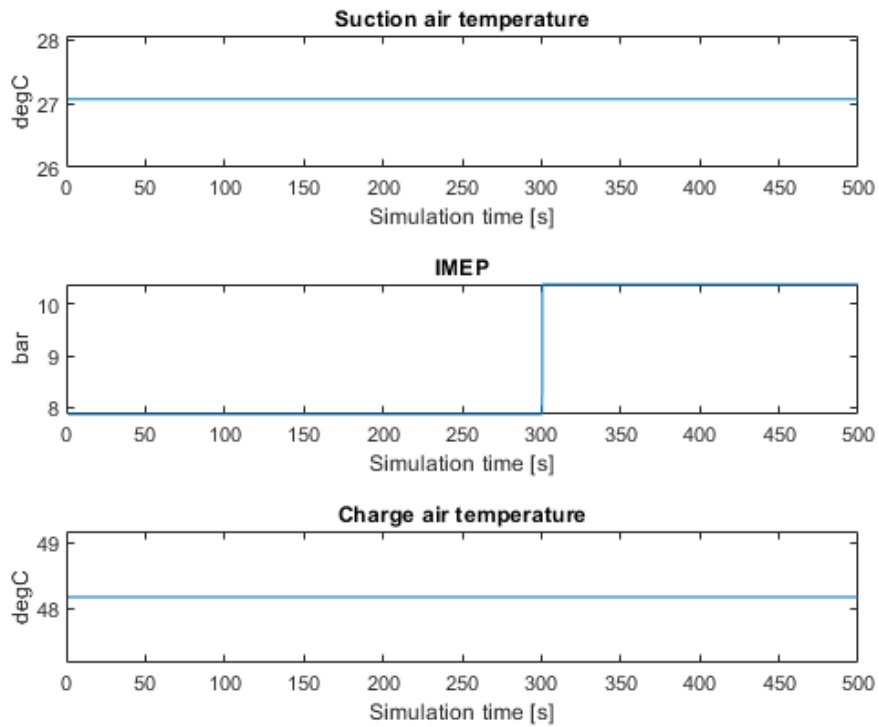


Figure 4-15: Set point tracking with disturbance rejection - disturbances

During this test the disturbances are kept fixed to avoid any added noise, except for the step change in IMEP.

4.7.1.3 Set point tracking with ramp in set point

In this section the set point is gradually increased from an initial value of 10500 and up to 12000 over a period of 1 unit per 100 [ms]. The change in set point starts at 150 seconds.

Note that the set point is not known to the controller before the change and is kept constant for the whole prediction horizon.

The prediction horizon is still set to 50 samples and the upper and lower bounds on the control signals amplitude is given as:

$$\begin{aligned} 0,3 &\leq \text{charge air pressure} \leq 8 \\ -20 &\leq \text{global timing} \leq 0 \end{aligned}$$

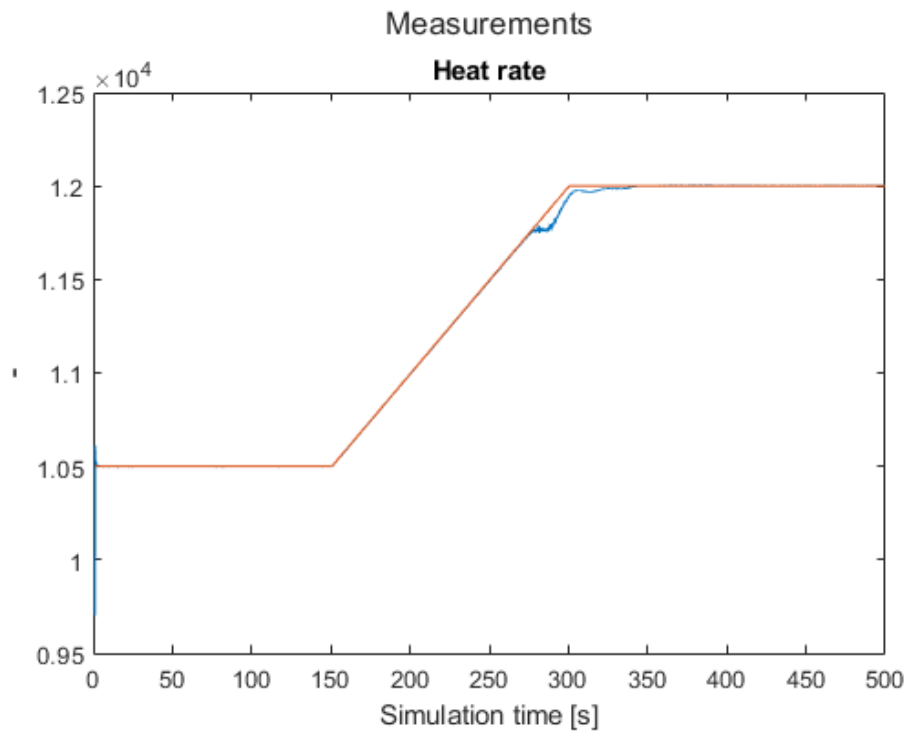


Figure 4-16: Set point tracking with gradually increasing set point – Heat rate

From Figure 4-16 that the set point tracks very well until just before the max set point is reached. It is noted that this period is the period in which the charge air pressure reaches its maximum allowed value and hence the global timing is used to reduce the error and get the heat rate back on set point.

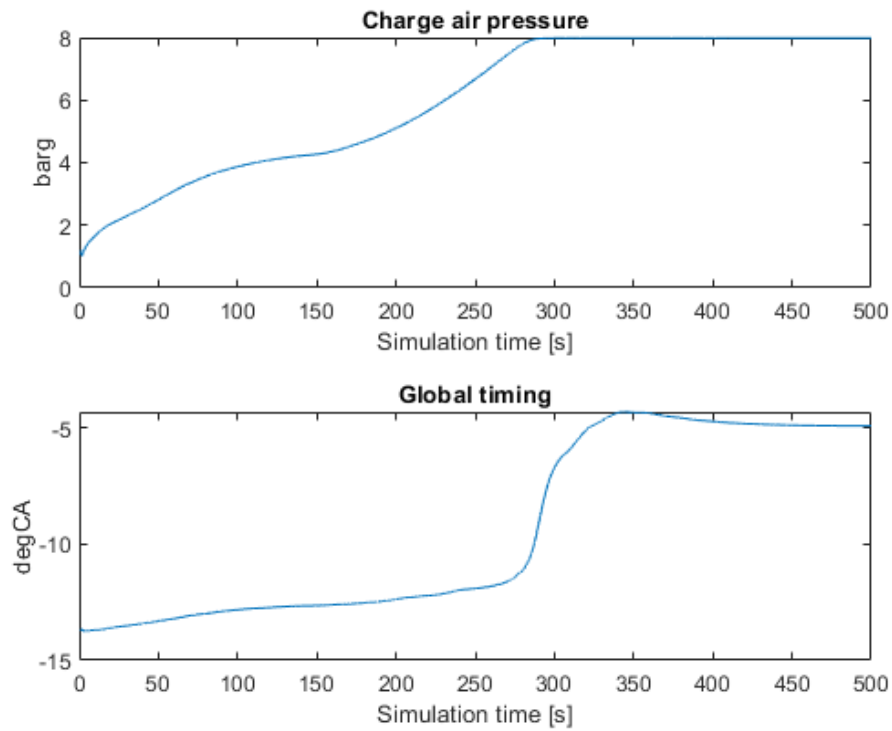


Figure 4-17: Set point tracking with gradually increasing set point – control inputs

From Figure 4-17 we can see that the charge air pressure is saturated at 8 [barg] while the global timing is still within its constraints.

By increasing the prediction horizon from 50 samples to 150 samples the response is as indicated in Figure 4-18 on the heat rate. We still get an error when the set point gets close to 12000 but this time it has a different shape.

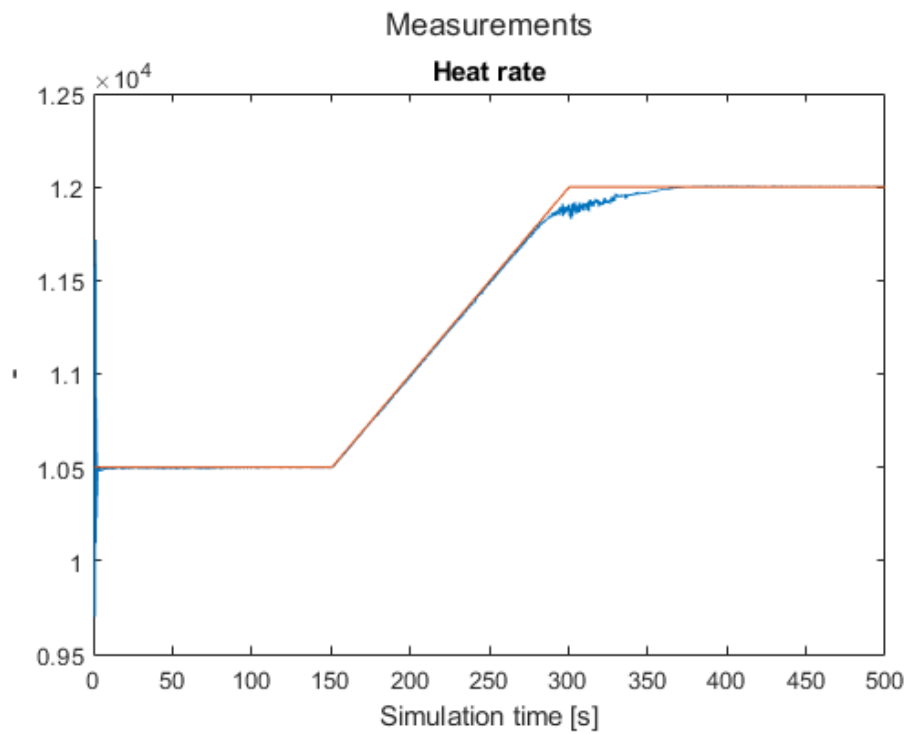


Figure 4-18: Set point tracking with gradually increasing set point – Heat rate and prediction horizon of 150 samples

Looking at the control outputs again in Figure 4-19 it is noted that the charge air pressure this time is limited to just below the max limit and the gradient of the ignition timing is less steep and predicts higher global timing value earlier than with the prediction horizon on 50 samples.

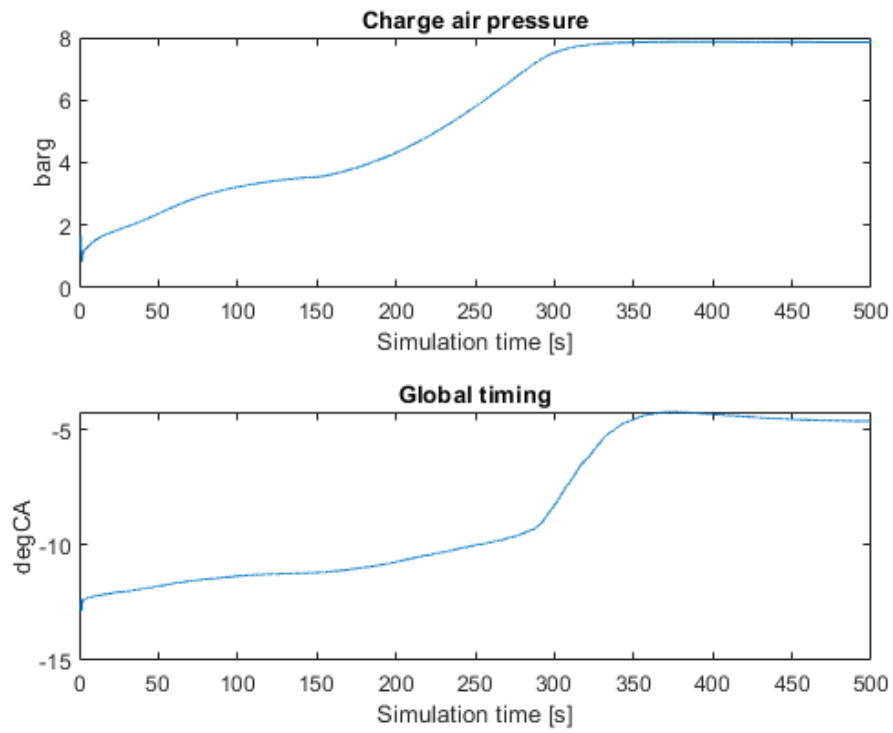


Figure 4-19: Set point tracking with gradually increasing set point – control outputs and prediction horizon of 150 samples

4.7.2 Set point tracking with constraints on Δu_m

By adding constraints on the rate of change of the control variable we get the results as in Figure 4-20 and Figure 4-21. This time we see a different result as the MPC takes into account the rate of change limitations of the control values. In this example the disturbance is identical to the one in Figure 4-15, that is a step change of 2,5 [bar].

The constraints are

$$\begin{aligned} -0,2 &\leq \Delta u_1 \leq 0,2 \\ -0,3 &\leq \Delta u_2 \leq 0,5 \\ 0,3 &\leq u_1 \leq 4 \\ -20 &\leq u_2 \leq -8 \end{aligned}$$

where u_1 is charge air pressure and u_2 is global ignition timing. The Δu_m is given as unit per seconds. Note that the amplitude of the control signals is added as bounds as in the earlier sections and hence are interpreted as hard constraints. The rate of change constraints are added as nonlinear constraints even though they are not nonlinear. The reason is the rate of change is calculated withing the cost function and also is one element in the cost function to minimize.

4.7.2.1 Set point tracking with load rejection

The set point is set to 10300 for the heat rate. There are no constraints on the outputs.

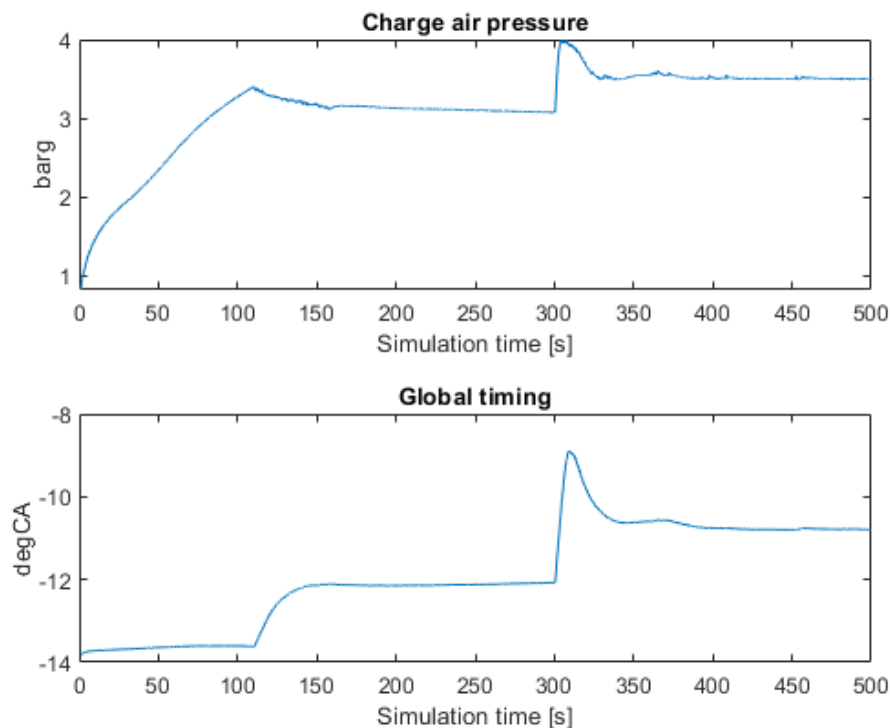


Figure 4-20: Set point tracking with disturbance rejection - control values with rate of change constraint

From Figure 4-20 it can be seen that the charge air pressure control output slightly touches the upper bound on the control output, but for the most parts stay within bounds. The global timing control output is within the operational window at all times.

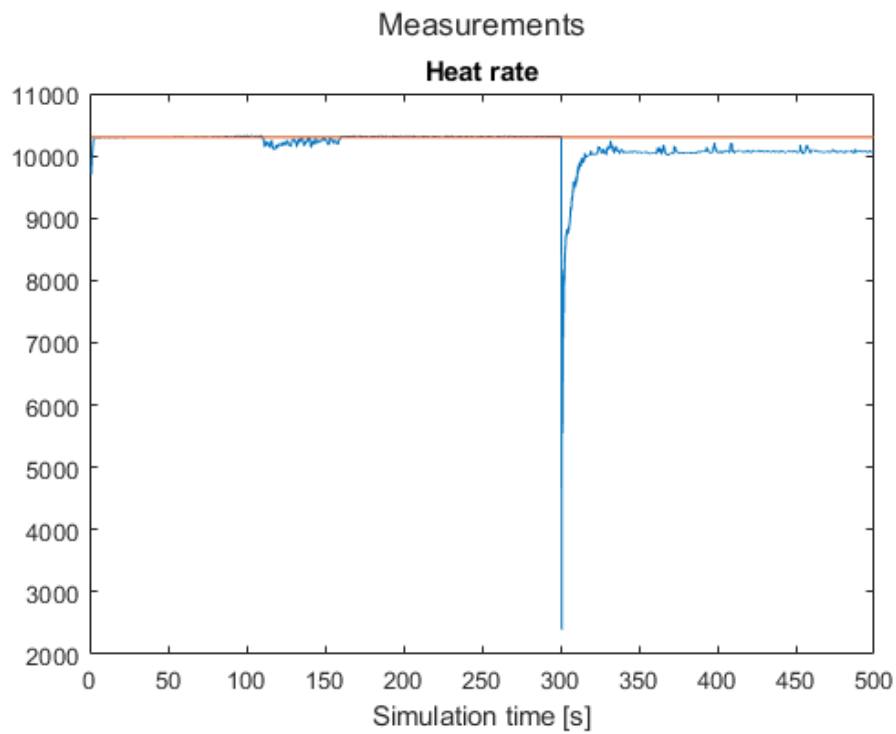


Figure 4-21: Set point tracking with disturbance rejection - measured outputs with constraint on rate of change of control values

The heat rate output tracks the set point rather well before the step disturbance as can be seen in Figure 4-21. After the disturbance it never gets back to the set point and a constant offset error is seen. Adding either internal integral terms or external should remove the constant error term.

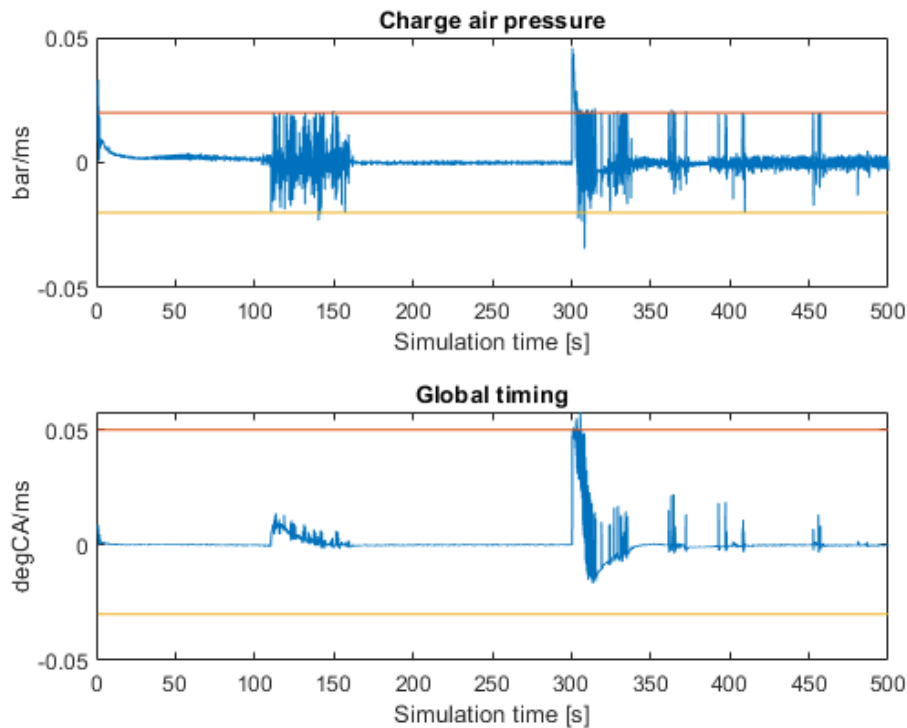


Figure 4-22: Set point tracking with disturbance rejection – rate of change on control values with constraints

The Δu control signals can be seen in Figure 4-22. Just after the disturbance at 300 seconds, we can see some constraints being broken by the charge air pressure but overall the constraints are honoured well. We have some issues after about 100 seconds where the charge air pressure control signals go from a positive direction to a negative direction and the ignition timing is increased to keep the set point. It should be noted that at this point we are physically outside normal operating window as an IMEP of 7,8 [bar] does not produce a charge air pressure of 3,5 [barg] during normal physical operation.

4.7.2.2 Set point tracking with ramp in set point

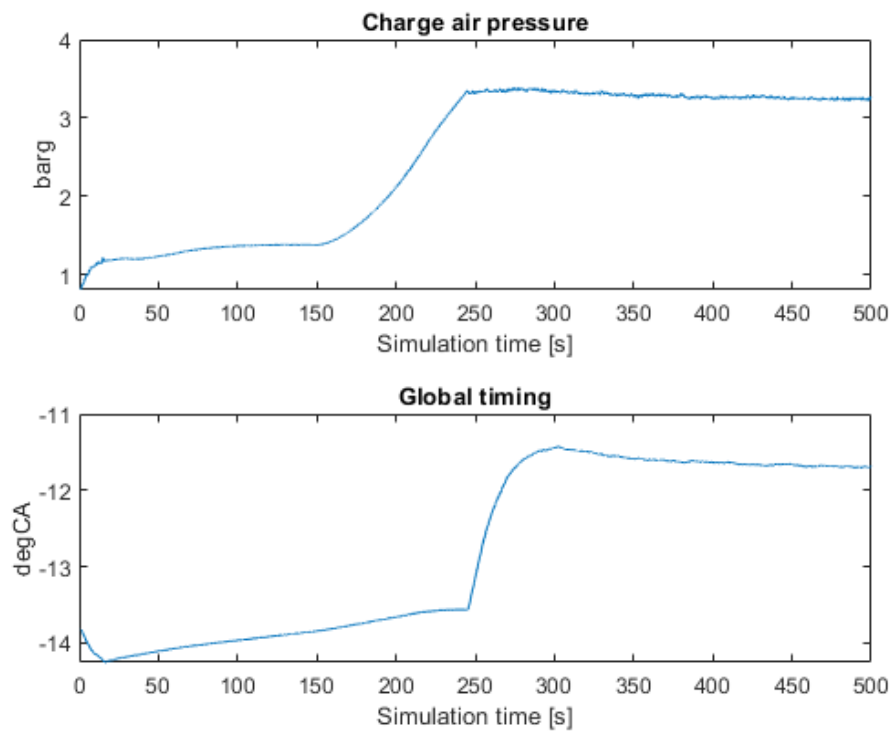


Figure 4-23: Set point tracking with ramp in set point - control values with rate of change constraint

In Figure 4-23 we observe the control output during a ramp change in set point while keeping the disturbances fixed. The set point is ramped between 9800 to 10500 as seen in Figure 4-24. We note that neither the global timing nor the charge air pressure reaches any bounds during this simulation.

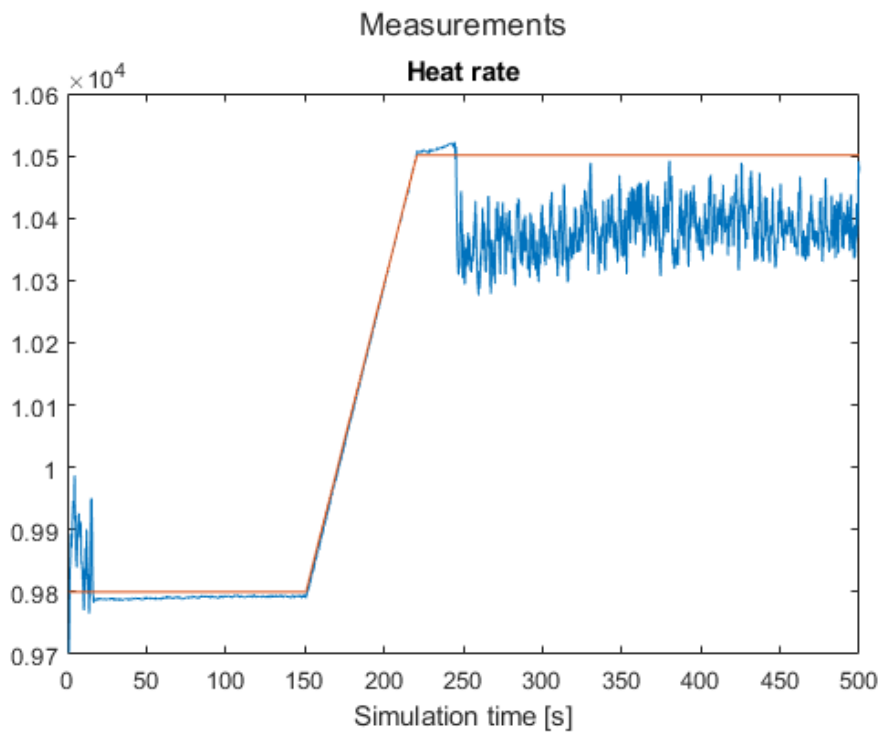


Figure 4-24: Set point tracking with ramp in set point - heat rate output with constraint on rate of change of control values

The set point is here in Figure 4-24 clearly not kept after the change in set point. After a successful ramp up of set point and ok tracking for the first few seconds after the final point is reach the heart rate drops below set point for the rest of the simulation.

The rate of change on the control outputs can be seen in Figure 4-25 as is clear the control output is not at all stable after the 250 seconds mark.

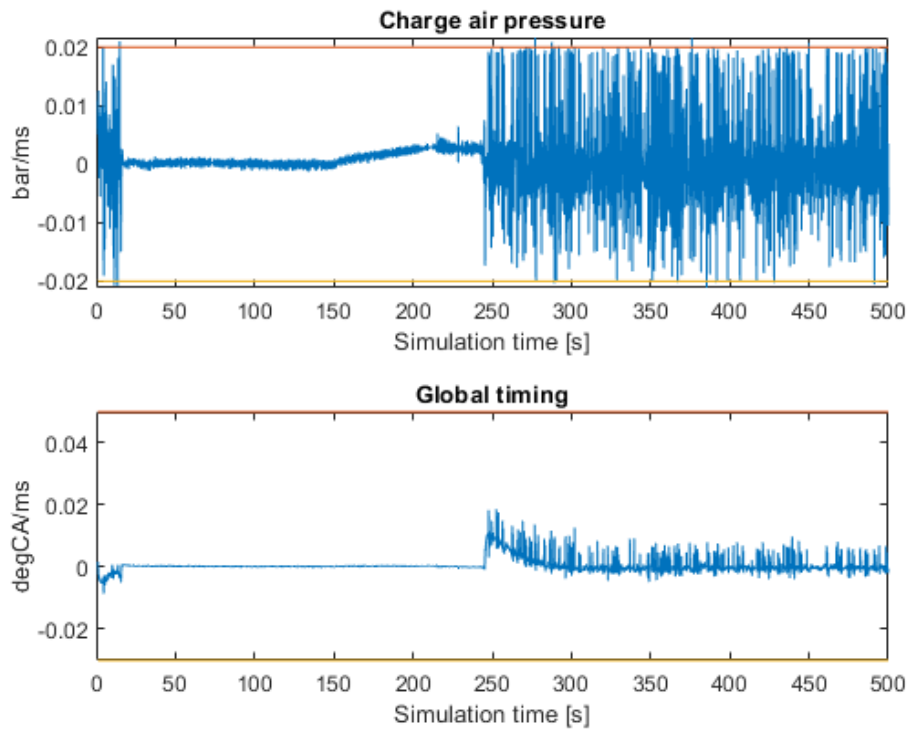


Figure 4-25: Set point tracking with ramp in set point – rate of change on control outputs

By increasing the weighting matrix on the error term $Q = 10$ and adding the u variable to the cost function the results change. The control outputs have the same shape but the absolute values changes. The updated control values is seen in Figure 4-26 and can be compared to those in Figure 4-23.

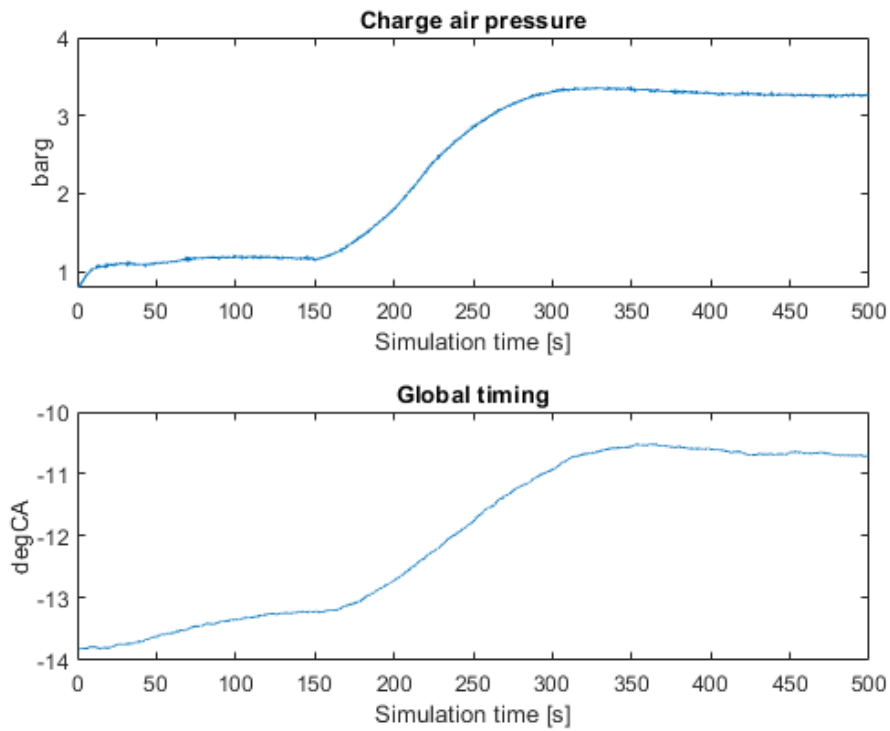


Figure 4-26: Set point tracking with ramp in set point - control values with rate of change constraint and increased weight on error

From Figure 4-27 it can be clearly seen that the set point is now tracked but the signal looks very noisy compared to the previous simulations. This comes after added weight on the error term in the cost function compared to the weight on the Δu and u .

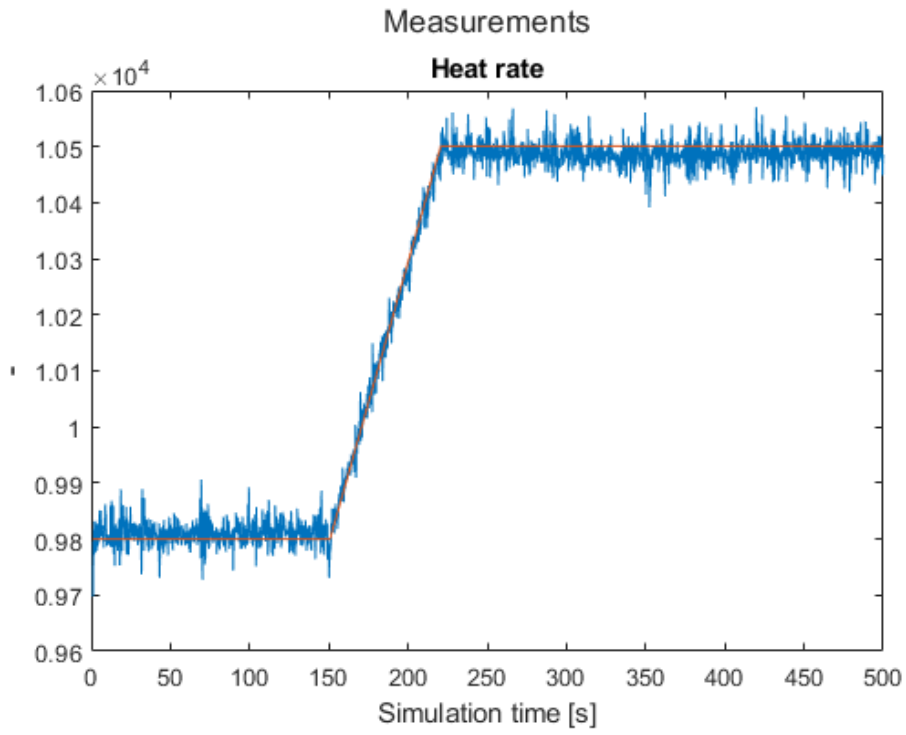


Figure 4-27: Set point tracking with ramp in set point - heat rate output with constraint on rate of change of control values and increased weight on error

4.7.3 Set point tracking with constraint on outputs

This section will impose constraints on the measured outputs in addition to the control signals amplitude and rate of change. The constraints are added as inequality constraints to the Fmincon algorithm. The constraints are internally by the Fmincon handled as soft constraints. The amplitude of the control signals is still imposed as upper and lower bounds and hence are treated as hard constraints.

In the following test there is added constraint on exhaust temperature at 600 [°C], that is the temperature needs to be below this level. At the same time there is added constraint on the measured O₂ percentage to keep it within 8,5 [%] (lower constraint) and 12,5 [%] (upper constraint).

The test is done by running with fixed set point and adding a step change in the *IMEP* (engine power output) of 2 [bar]. Set point is set to 10 300.

The results can be seen in Figure 4-28, Figure 4-29, Figure 4-30 and Figure 4-31.

The constraints in the measured outputs shown in Figure 4-31 is throughout the simulation. The constraints on the O₂ percentage is indicated with a yellow and red line and for the exhaust temperature with a red line.

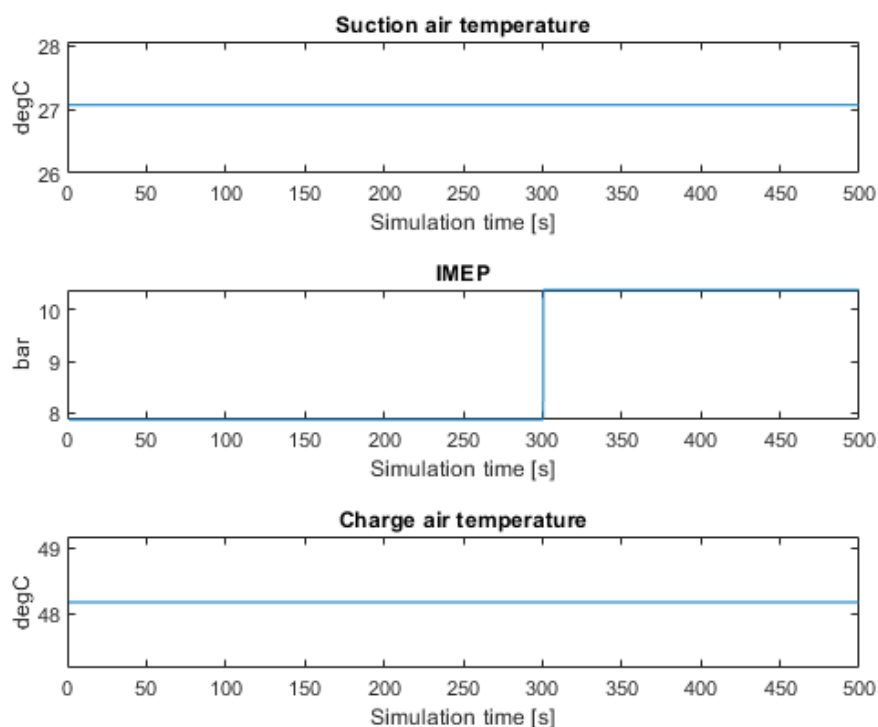


Figure 4-28: Set point tracking with constraints and disturbance rejection - step change in IMEP

From Figure 4-28 it noted that the disturbances other than the step change in the IMEP is kept constant. Looking at the control signals in Figure 4-29 it is clear that the system is unstable. The control value of charge air pressure is kept withing the upper and lower bounds while the global timing goes in and out of the lower bound after about 150 seconds.

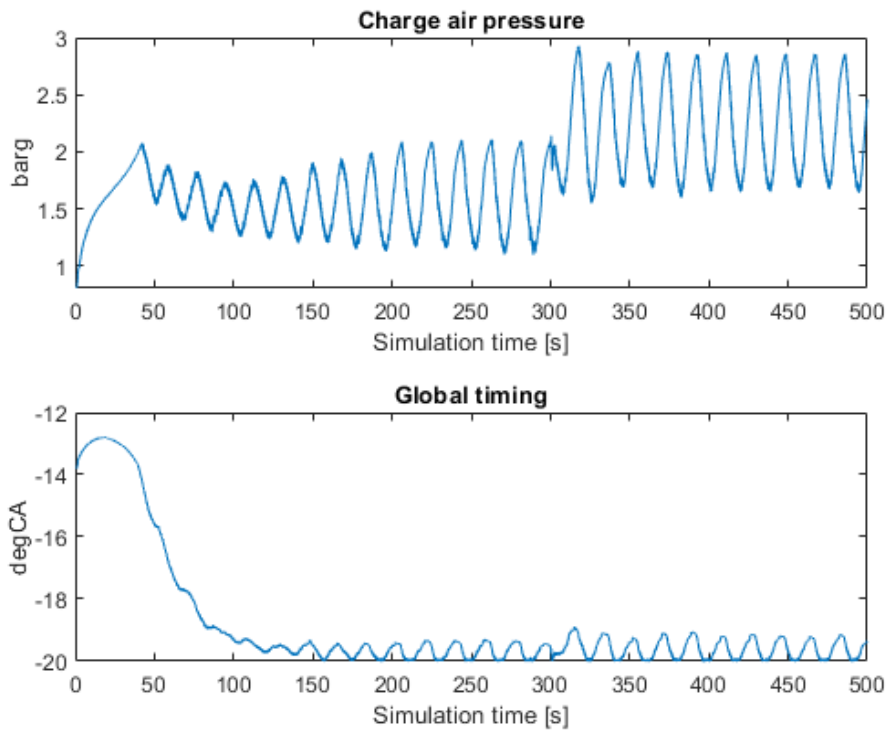


Figure 4-29: Set point tracking with constraints and disturbance rejection – control outputs

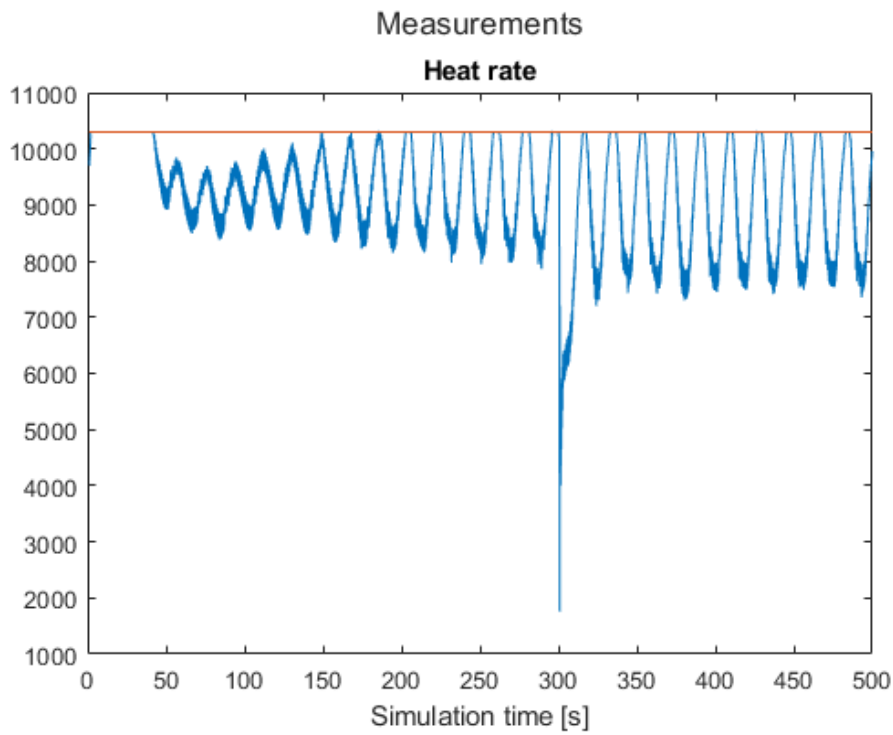


Figure 4-30: Set point tracking with constraints and disturbance rejection – heat rate

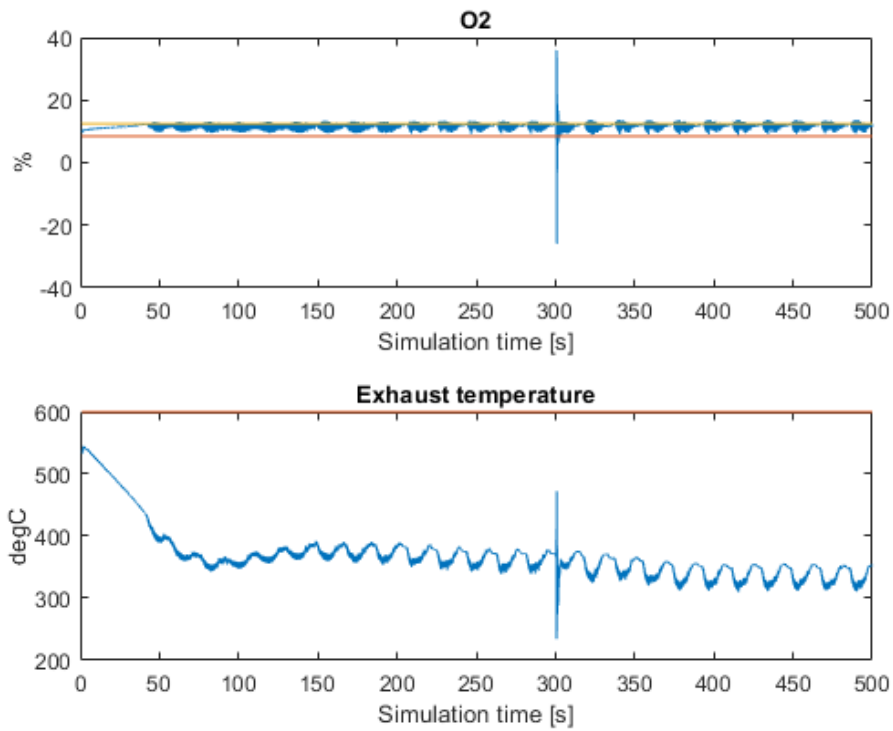


Figure 4-31: Set point tracking with constraints and disturbance rejection – measured values with constraints

By increasing the weight on the error term in the optimization the results as show in Figure 4-32, Figure 4-33, Figure 4-34 and Figure 4-35 are produced. No other parameters where changed. We see here a much “nosier” signal than previously, but still the overall trends are the same and the constraint on the *O2* is kept. We still get a between the set point and the heat rate.

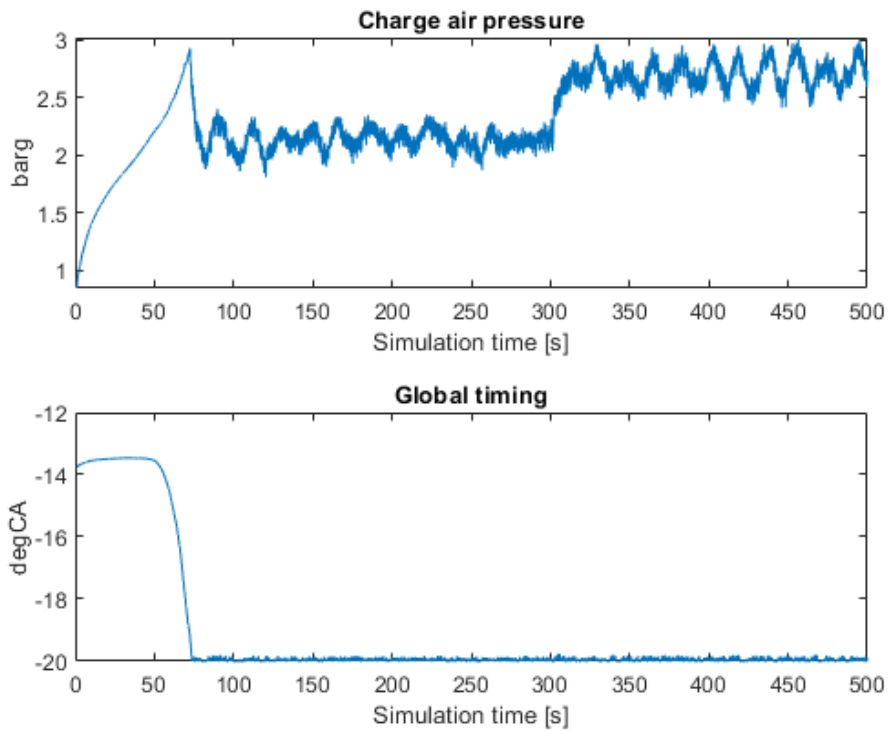


Figure 4-32: Set point tracking with constraints and disturbance rejection - control values – increased weight on error

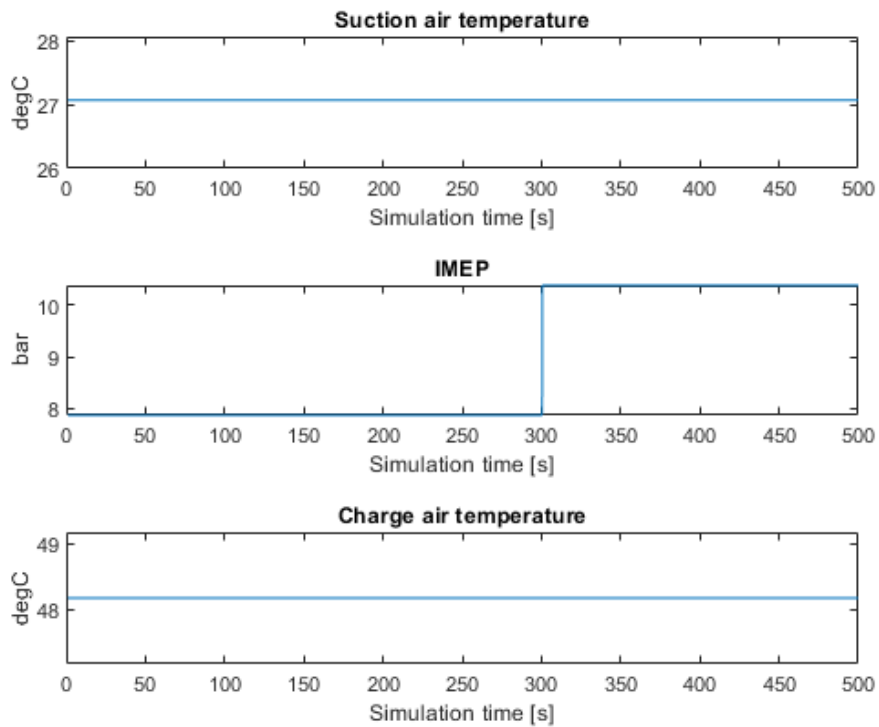


Figure 4-33: Set point tracking with constraints and disturbance rejection - disturbances - increased weight on error

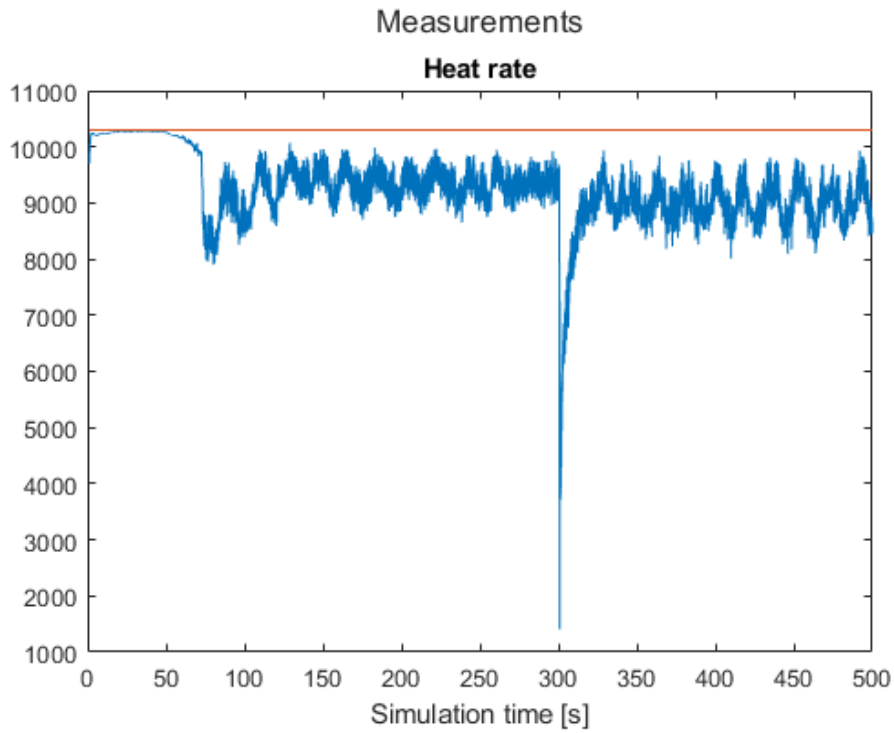


Figure 4-34: Set point tracking with constraints and disturbance rejection – heat rate – increased weight on error

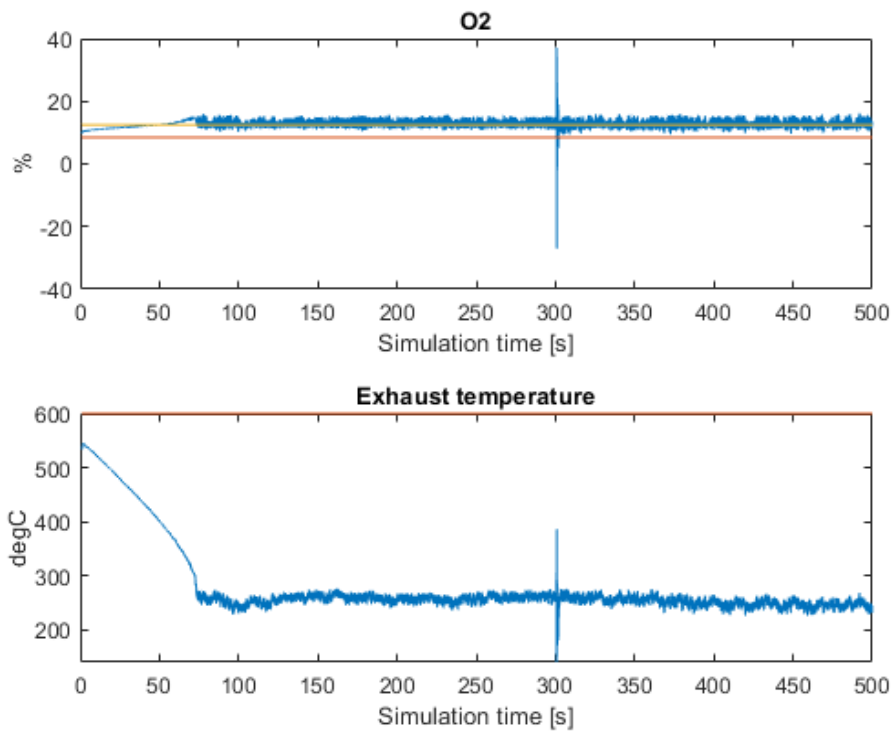


Figure 4-35: Set point tracking with constraints and disturbance rejection - measured values with constraints – increased weight on error

By increasing the prediction horizon from 70 to 150 we get the results in Figure 4-36, Figure 4-37 and Figure 4-38. The disturbance change remains the same as in the previous attempts. We now get a slight larger initial break of the constraint on the O_2 and a slightly higher charge air pressure set point around 100 seconds. The heat rate is now less noisy.

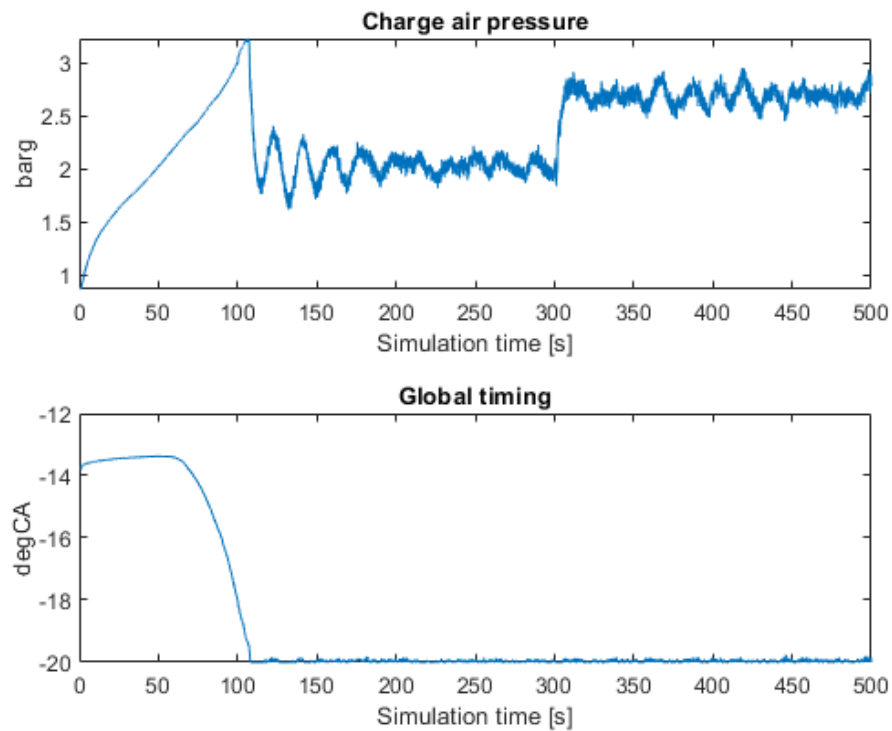


Figure 4-36: Set point tracking with constraints and disturbance rejection - control values – 150 steps in prediction horizon

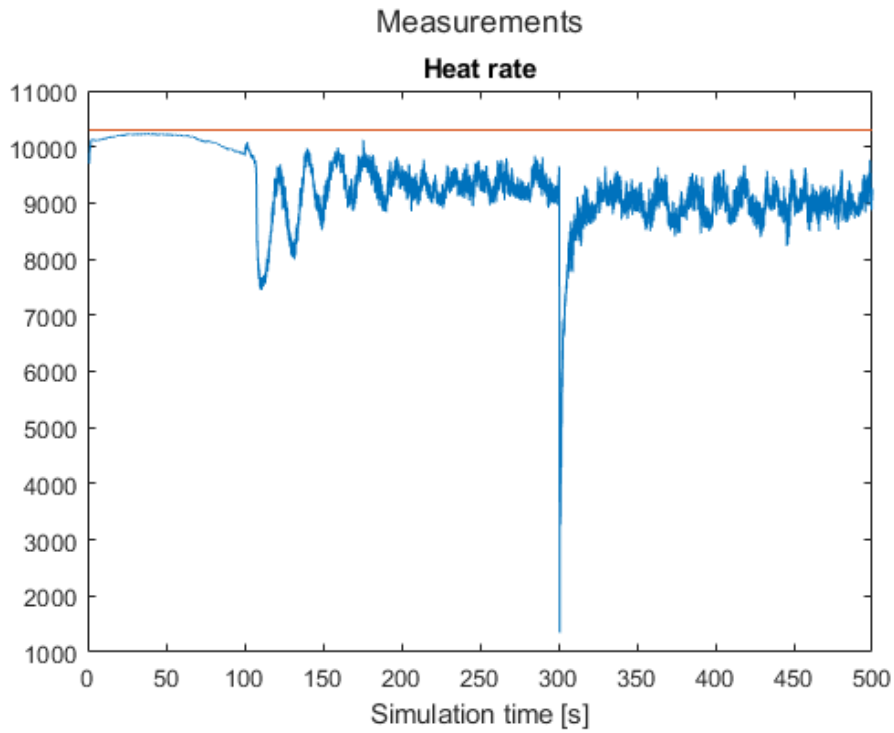


Figure 4-37: Set point tracking with constraints and disturbance rejection – heat rate – 150 steps in prediction horizon

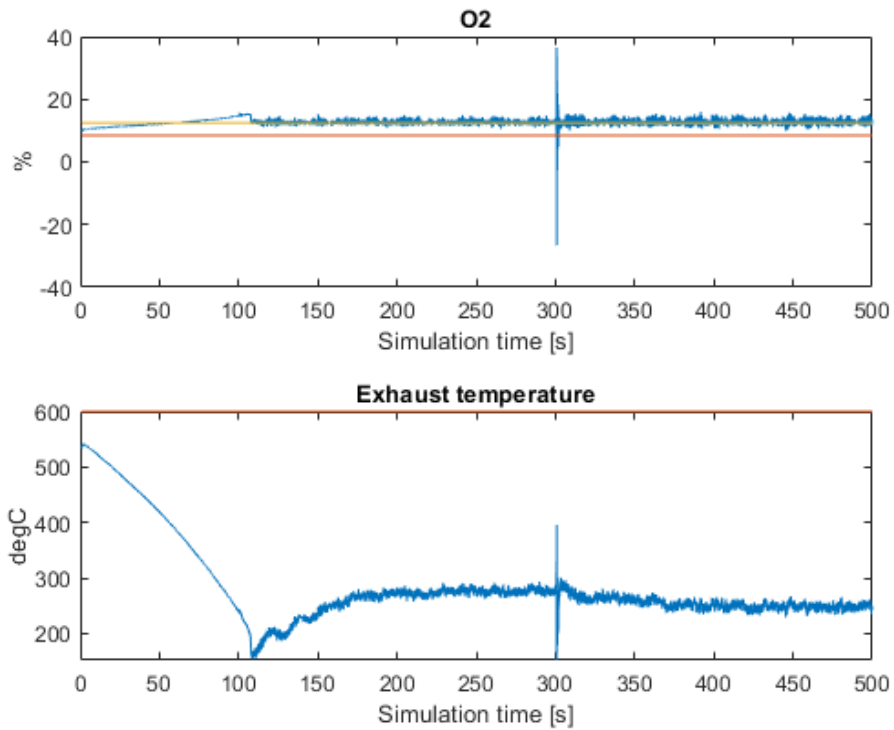


Figure 4-38: Set point tracking with constraints and disturbance rejection - measured values with constraints – 150 steps in prediction horizon

Looking more into the “noise” on the signals a test is done where the disturbances is kept constant and ramp on the set point is added. Prior to the test the starting set point and the finishing set point is tested such that the no limit is broken while at steady state.

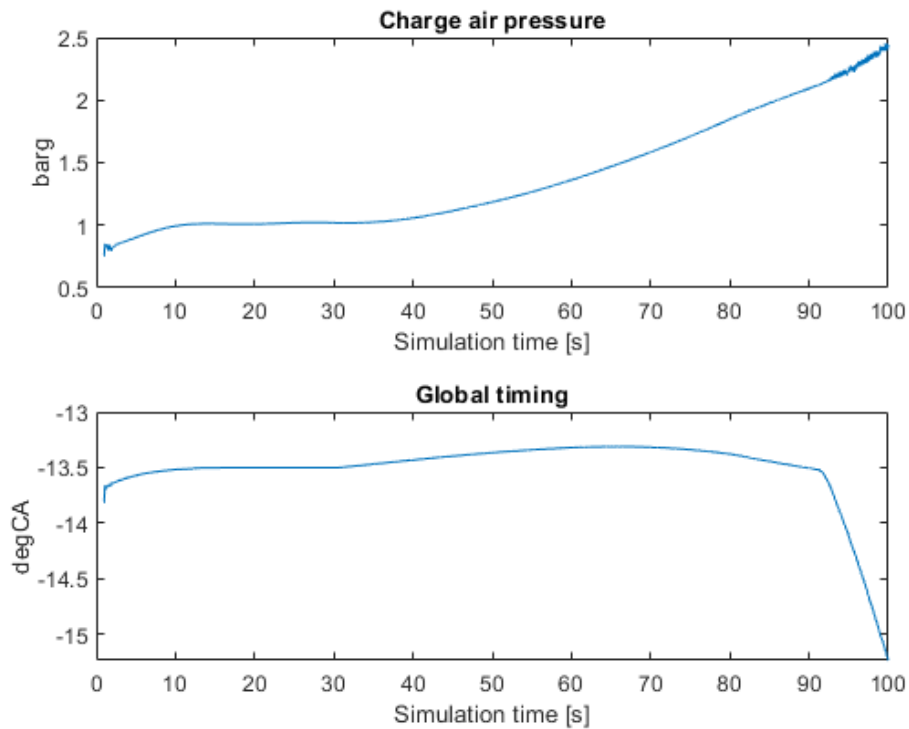


Figure 4-39: Set point tracking with constraint on output – control output

From Figure 4-39 it is seen that the control values are well within our well established upper and lower bounds even though a steep gradient on the global timing is seen. From Figure 4-40 it we note that the set point and process output tracks as it should between the original set point and the final set point, but after some time at the upper set point the Heat rate looks unstable and this is at the same time as the global timing reduces quickly.

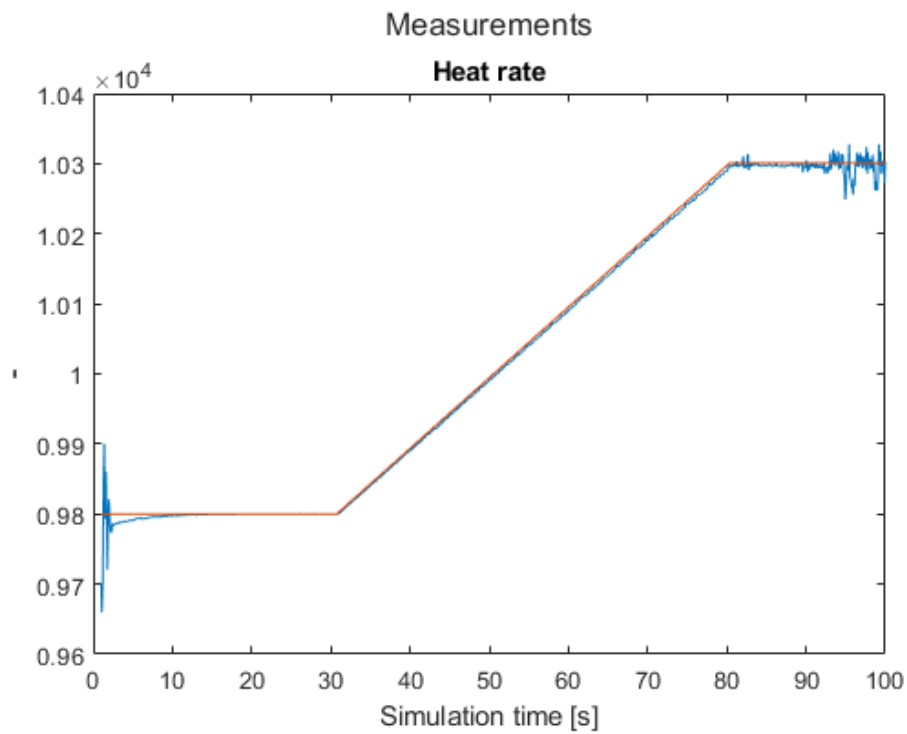


Figure 4-40: Set point tracking with constraint on output – Heat rate and set point

From Figure 4-41 it is clear that it is the O_2 limit breach which is causing the unstable control. The Fmincon is not able to keep the O_2 within the constraint and is resulting in unstable control once the constraint is broken.

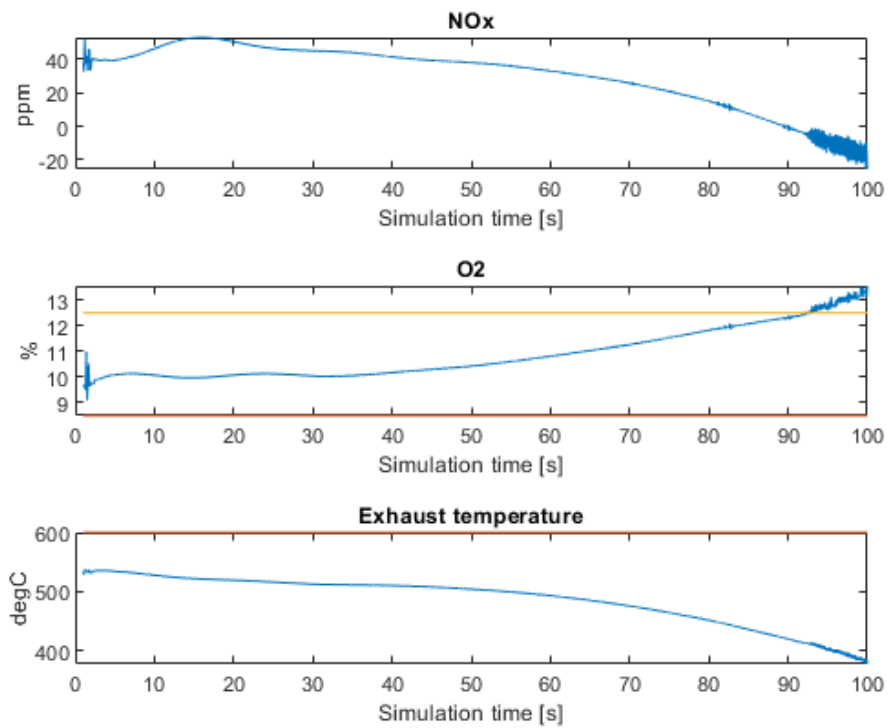


Figure 4-41: Set point tracking with constraint on output – Outputs

Moving the set point down slightly makes the control signals stay closer to their initial value. Still having the same absolute increase in set point of 500, but this time from 9600 to 10100 the set point tracking is ok with no noise.

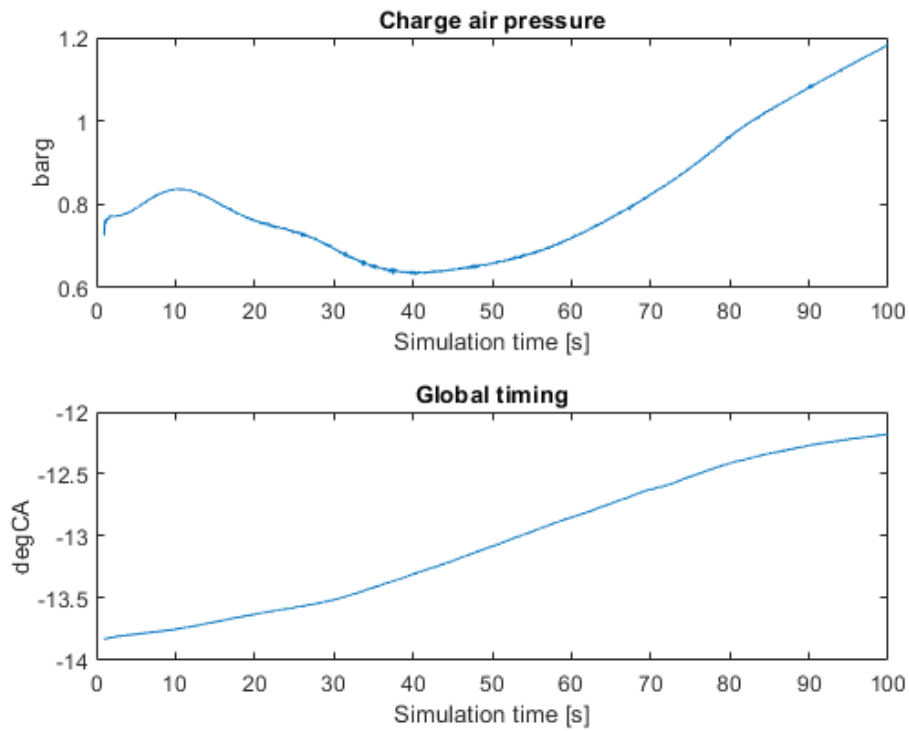


Figure 4-42: Set point tracking - ramp in set point - control values

The control value outputs are shown in Figure 4-42 and they stay well within their bounds. This time there is no sudden drop in the global timing as we note from Figure 4-44 that the outputs do not get close the constraint limits. The heat rate tracks therefor the set point as indicated in Figure 4-49 without any problems.

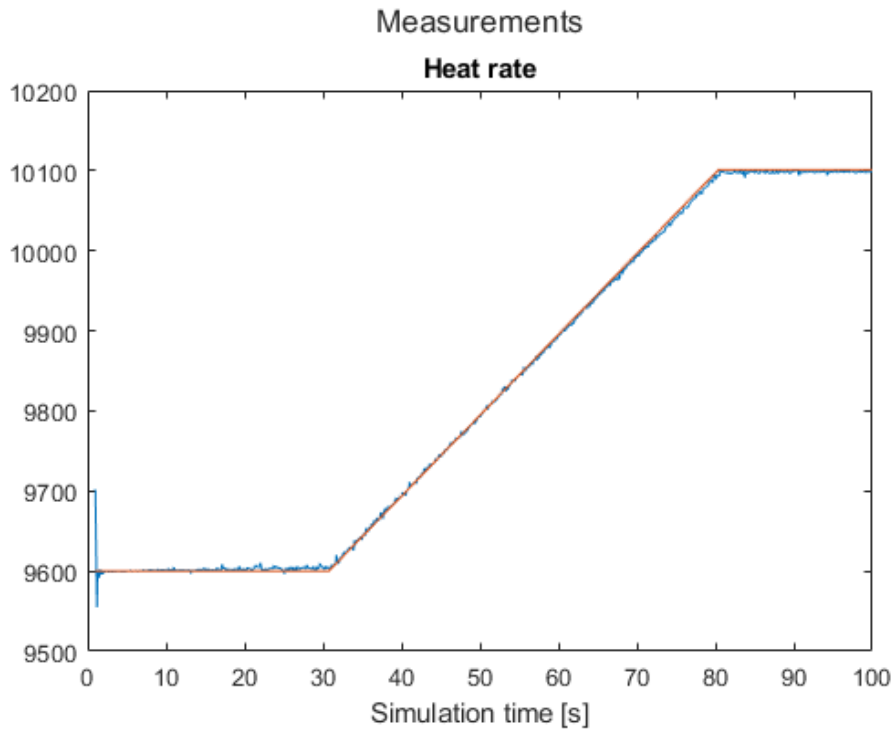


Figure 4-43: Set point tracking - ramp in set point - heat rate and set point

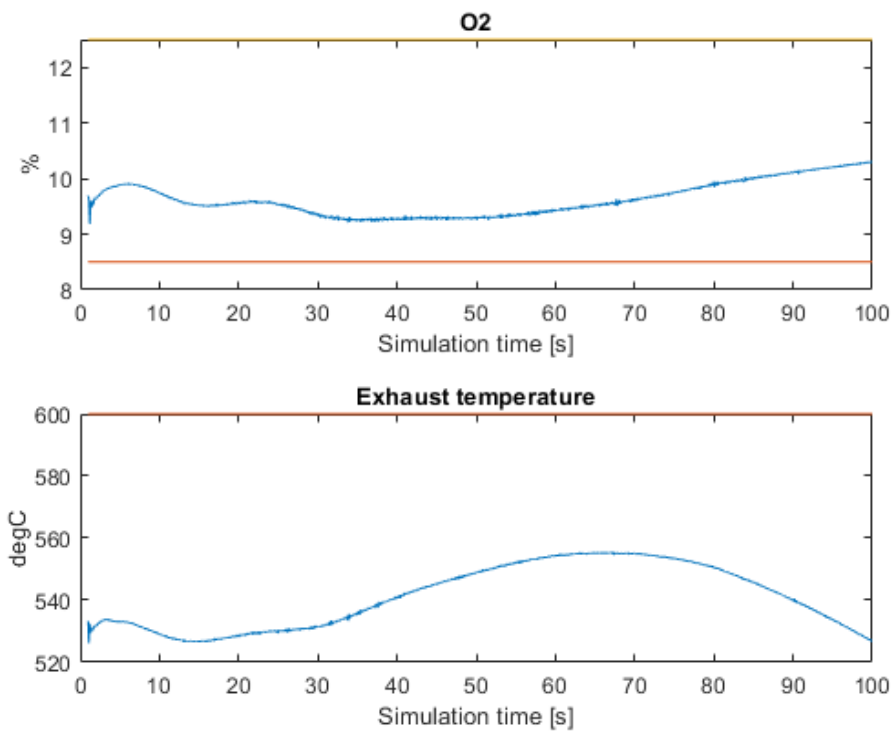


Figure 4-44: Set point tracking - ramp in set point - outputs

It is clear from several tests that getting into the constraints with the Fmincon will result in the a more noise results. Several attempts are done on moving the set point and the constraints to see if there are particular constraints that are more of a problem than others.

One important aspect here is that the modification of the set point and the addition of the disturbance might not be within the boundaries of the model at all. The heat rate is usually a result of the disturbance and keeping it to a given set point is not a physical thing to do. The heat rate will reduce as the *IMEP* disturbance increases as the engine become more efficient. The final optimizer here will also not have a set point and hence it will try to minimize the heat rate as much as possible but still stay within the constraints; and hence the constraints will most likely be active at all times.

In a test the weight on the change on control signal was increased from 0,01 to 1 in order to reduce the change in control signal more Vs the error.

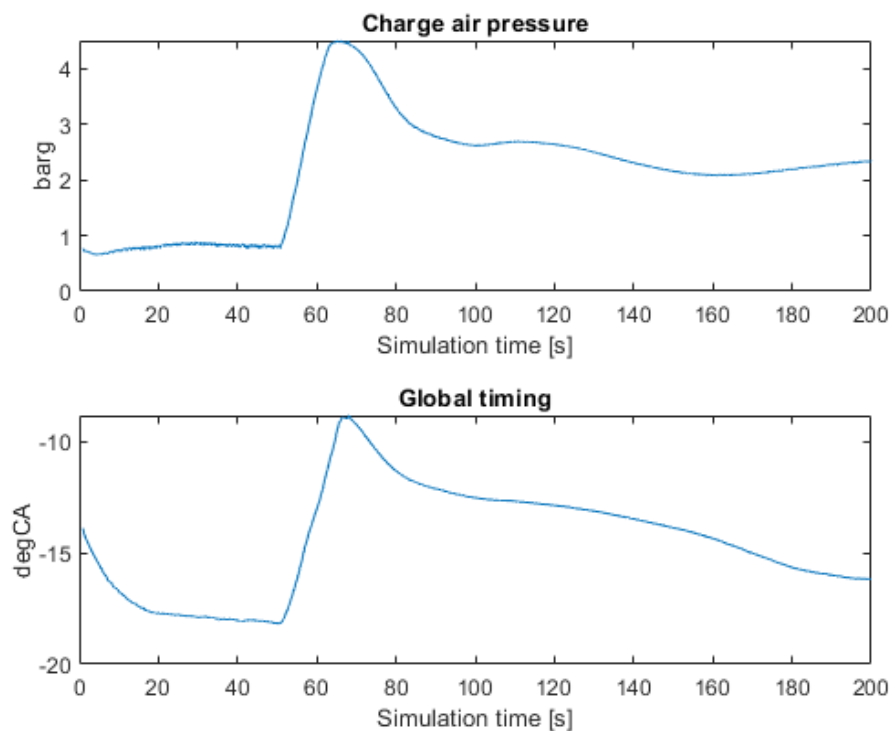


Figure 4-45: Set point tracking with ramp in disturbance - constraint on Exhaust - control outputs

In Figure 4-45 we see that the control signal momentarily hits the upper bound of 4,5 [barg] at 70 seconds causing. At the same time from Figure 4-48 it is noted that the heat rate drops during this period before recovering. At the same time, in Figure 4-46, the disturbance reaches its maximum value around 65 seconds.

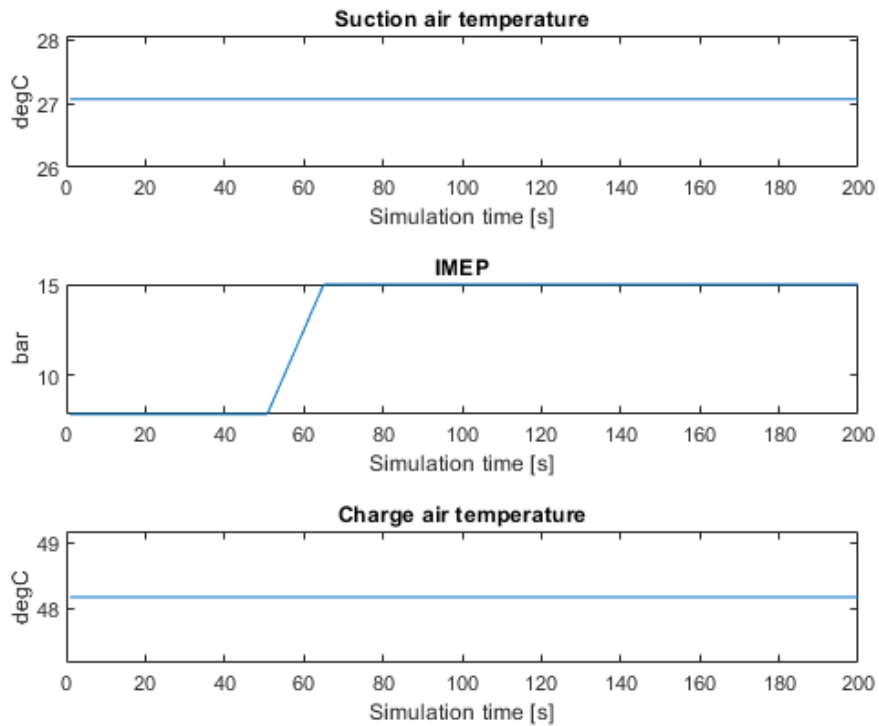


Figure 4-46: Set point tracking with ramp in disturbance - constraint on Exhaust – disturbances
 Only the IMEP disturbance is changed and the other two kept constant.

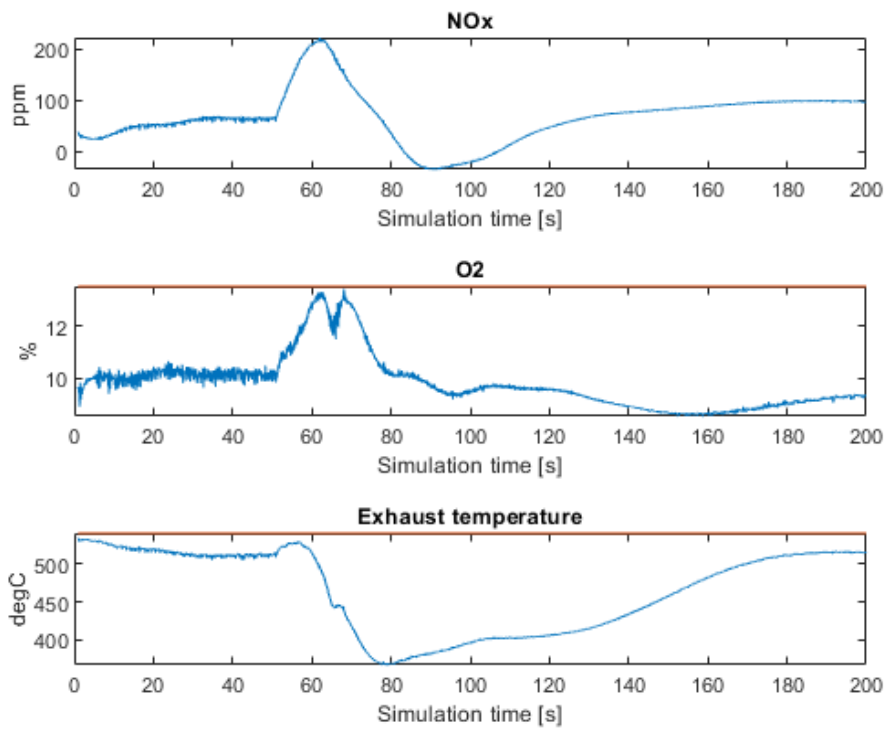


Figure 4-47: Set point tracking with ramp in disturbance - constraint on Exhaust – outputs

During the set point tracking of the first stable phase – no change in either set point nor disturbance there is a lot of noise on the signal as the set point is not reached. This comes from the optimizer and the fact that the weight matrix of the error term was reduced from 100 to 1. This is definitely causing problems. During the change of set point on the heat rate and the disturbance change there are some noise, see Figure 4-47, until it stabilizes after the changes stopes. From Figure 4-47 it is noted that the *Exhaust temperature* is close to its constraint of 540 [°C] just before 60 seconds as well as the *O2* getting close into its constraint.

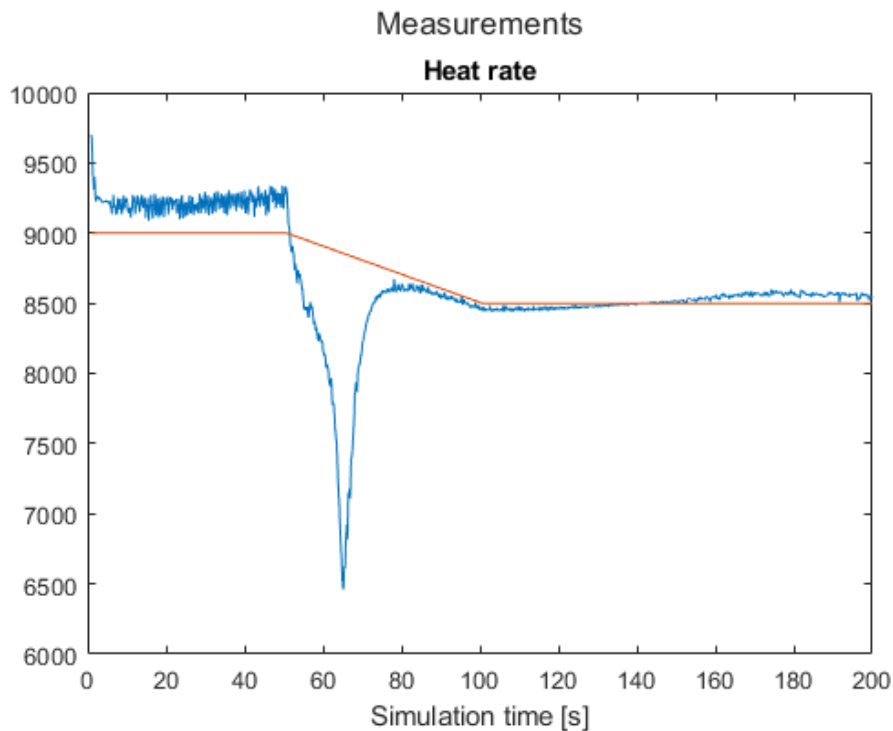


Figure 4-48: Set point tracking with ramp in disturbance - constraint on Exhaust – Heat rate

4.7.4 Minimize heat rate with constraints

In the following test the heat rate is minimized separately. That is; there is no set point tracking, but the heat rate is to be minimized while still honouring the constraints. Since the value for the heat rate is so large, i.e. not normalized, care must be taken as the Fmincon uses the constraint weight and the objective function weight to decide which to follow in case of not both being fully achievable. In our real-life case we want to minimize the heat rate but stay within the limits.

In the test below the IMEP is ramped from initial condition and up to ~18 bar with constraints on O₂ percentage and exhaust temperature.

The results can be viewed in Figure 4-49, Figure 4-50, Figure 4-51 and Figure 4-52. We see that the control value for Global timing reaches its upper bounds which gives a non-practical condition, but the Fmincon manages to keep the O₂ percentage above the 8,5 [%] orange line in Figure 4-52 and still keep the *exhaust temperature* below 600 [°C].

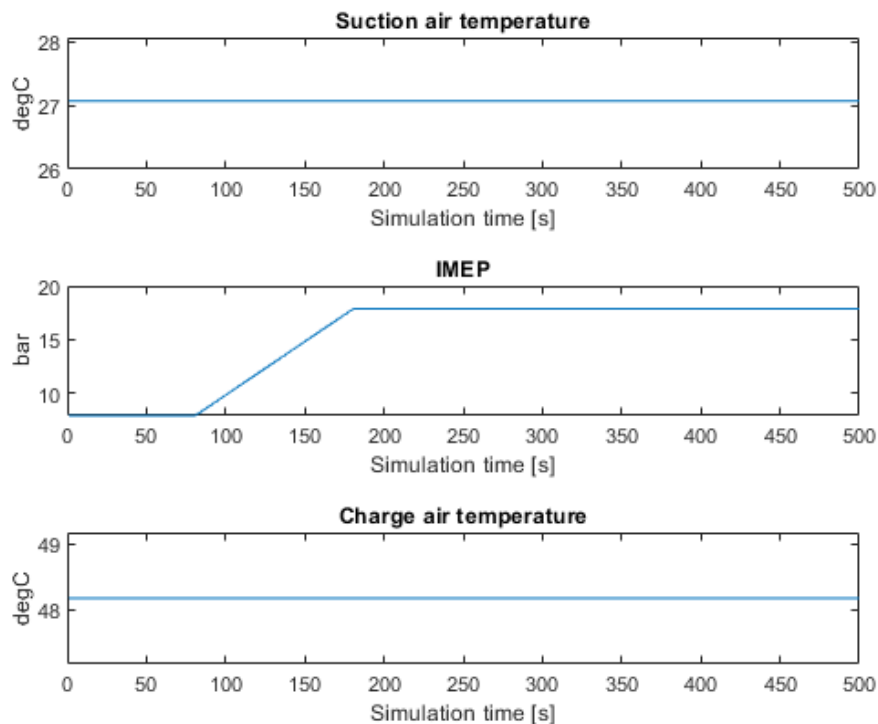


Figure 4-49: Minimized heat rate with constraints and disturbance rejection – disturbances

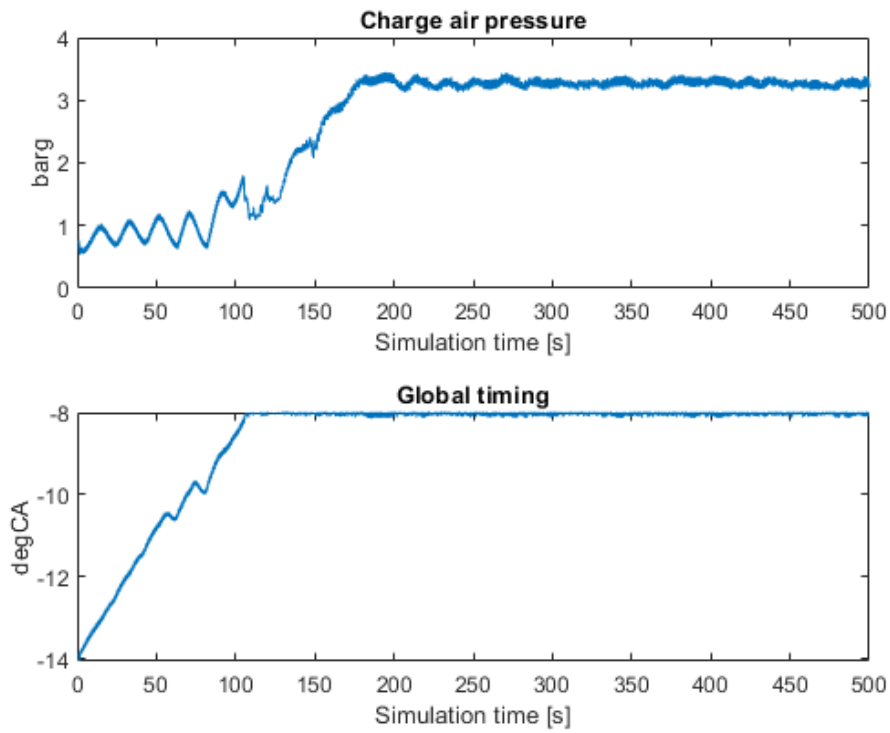


Figure 4-50: Minimized heat rate with constraints and disturbance rejection - control inputs

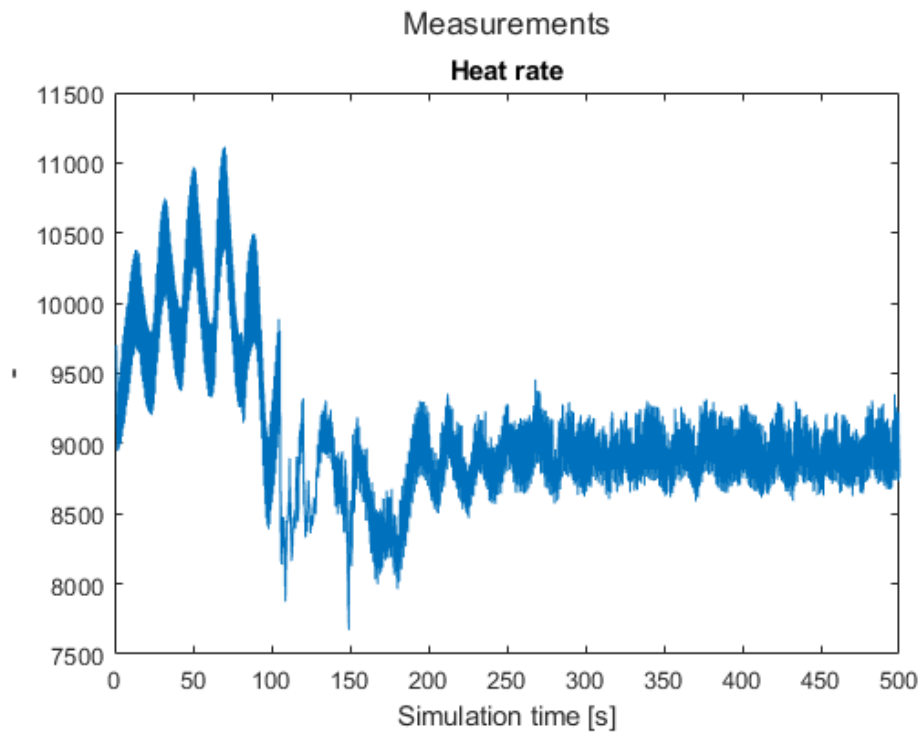


Figure 4-51: Minimized heat rate with constraints and disturbance rejection – heat rate

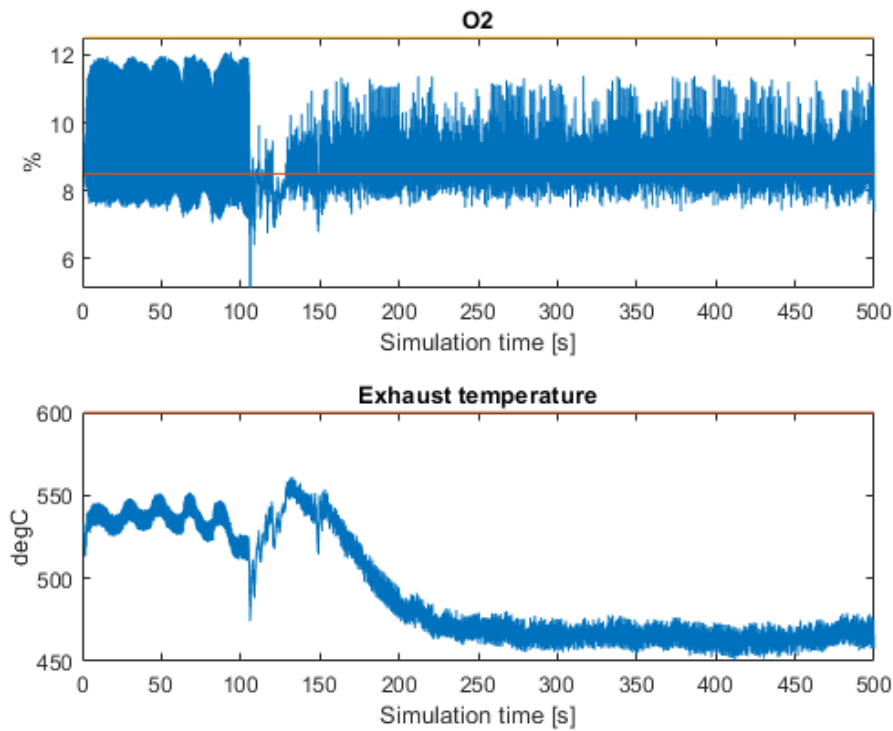


Figure 4-52: Minimized heat rate with constraints and disturbance rejection – process measurements

A second attempt is done by reducing the weight on the heat rate minimization and increase the weight on the rate of change of the output to see if it would stabilize. All other conditions remain the same. We see in Figure 4-53, Figure 4-54 and Figure 4-55 that the trajectory is the same as before but this time it stabilises and the noise is largely reduced. At least this applies to the period after 150 seconds point. The heat rate settles at the about same location as in the previous attempt.

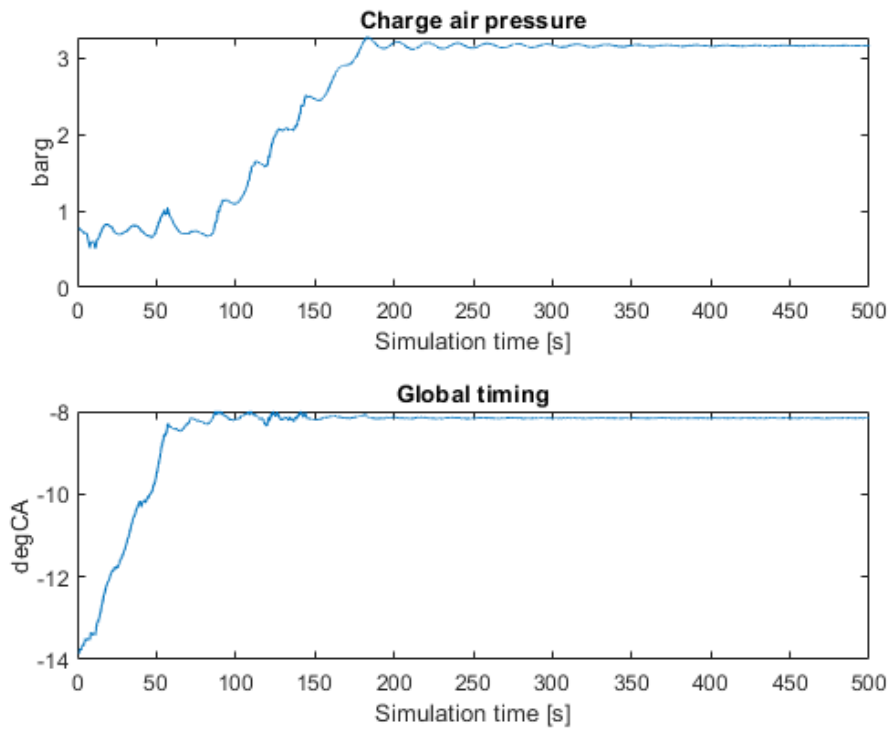


Figure 4-53: Minimized heat rate with constraints and disturbance rejection - control inputs – increased weight on control signal change

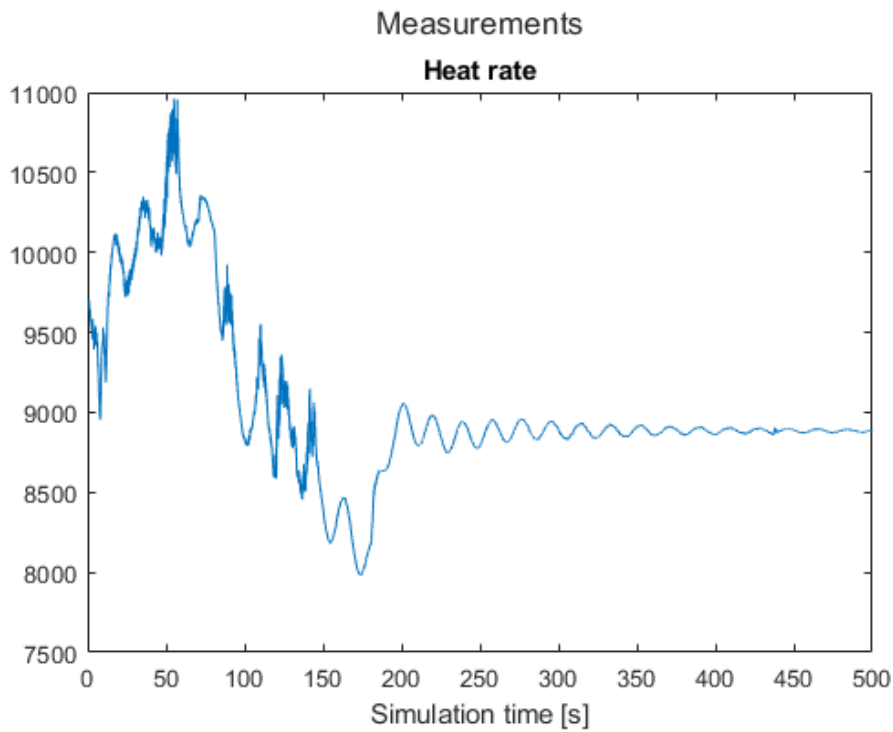


Figure 4-54: Minimized heat rate with constraints and disturbance rejection – heat rate – increased weight on control signal change

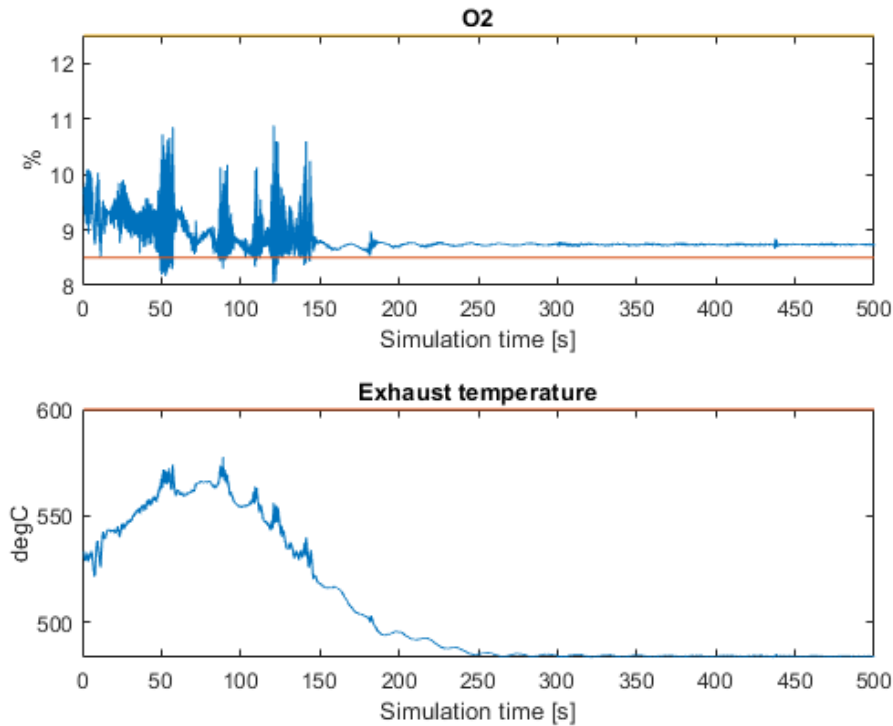


Figure 4-55: Minimized heat rate with constraints and disturbance rejection - outputs – increased weight on control signal change

Several simulations are performed to improve the behaviour of the controller with a fixed ramped disturbance. The results in Figure 4-56, Figure 4-57, Figure 4-58 and Figure 4-67 shows some of the results with a slightly modified set up of the inequality constraints. This time the constraints are estimated based on the initial state estimation for each iteration over the whole prediction horizon given as

$$Y = Fx(k_i) + \phi_u U + \phi_{u_d} U_d \quad (4-11)$$

where Y is the estimated outputs over the whole prediction horizon and $x(k_i)$ is states at the current initial time of the iteration and U and U_d is the control inputs signal and disturbances over the prediction horizon respectively.

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix} \quad (4-12)$$

$$\phi_u = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \dots & CA^{N_p-N_c}B \end{bmatrix} \quad (4-13)$$

$$\phi_{u_d} = \begin{bmatrix} CB_d & 0 & 0 & \dots & 0 \\ CAB_d & CB_d & 0 & \dots & 0 \\ CA^2B_d & CAB_d & CB_d & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B_d & CA^{N_p-2}B_d & CA^{N_p-3}B_d & \dots & CA^{N_p-N_c}B_d \end{bmatrix} \quad (4-14)$$

The source code for these tests are given in Appendix I

The control output signals in Figure 4-56 now shows very little noise on the control signals during this simulation.

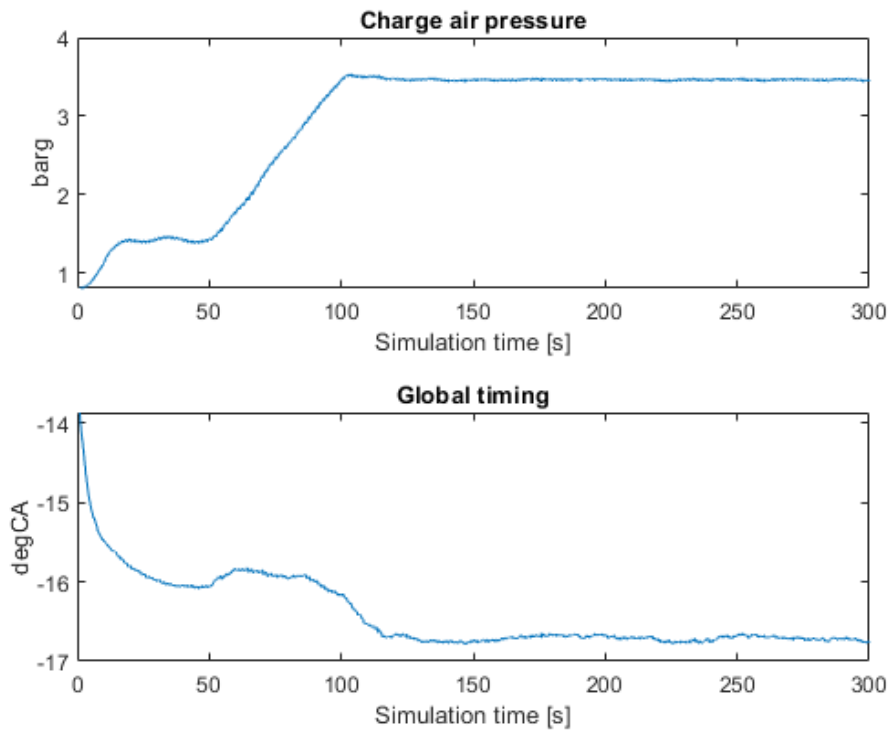


Figure 4-56: Minimized heat rate with constraints and disturbance rejection - control inputs

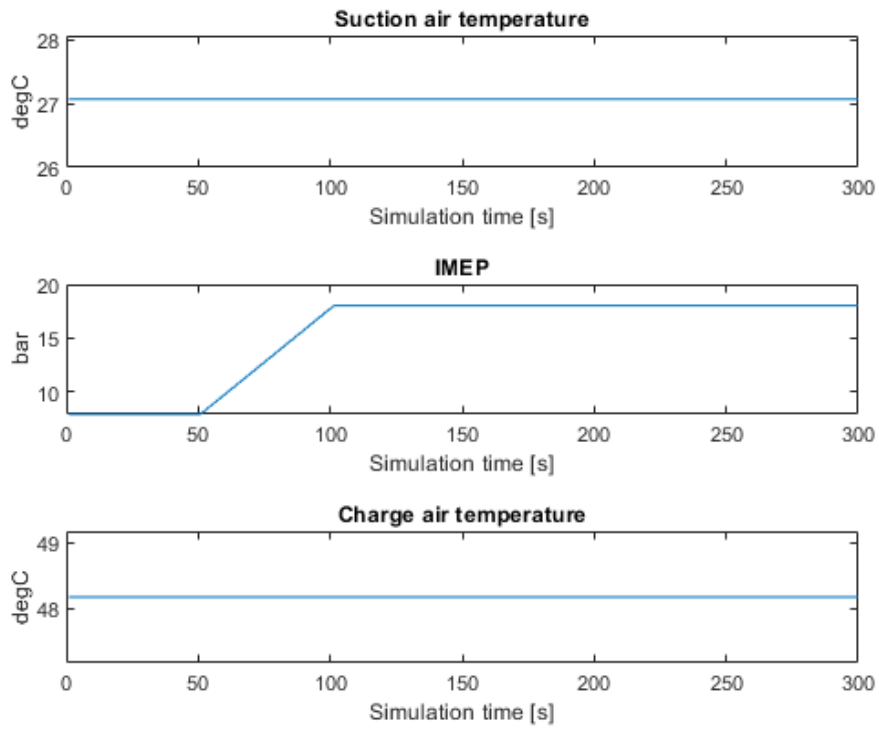


Figure 4-57: Minimized heat rate with constraints and disturbance rejection - disturbance
 The disturbance signal is still the same with a ramp to about 18 [bar] on the *IMEP* disturbance.

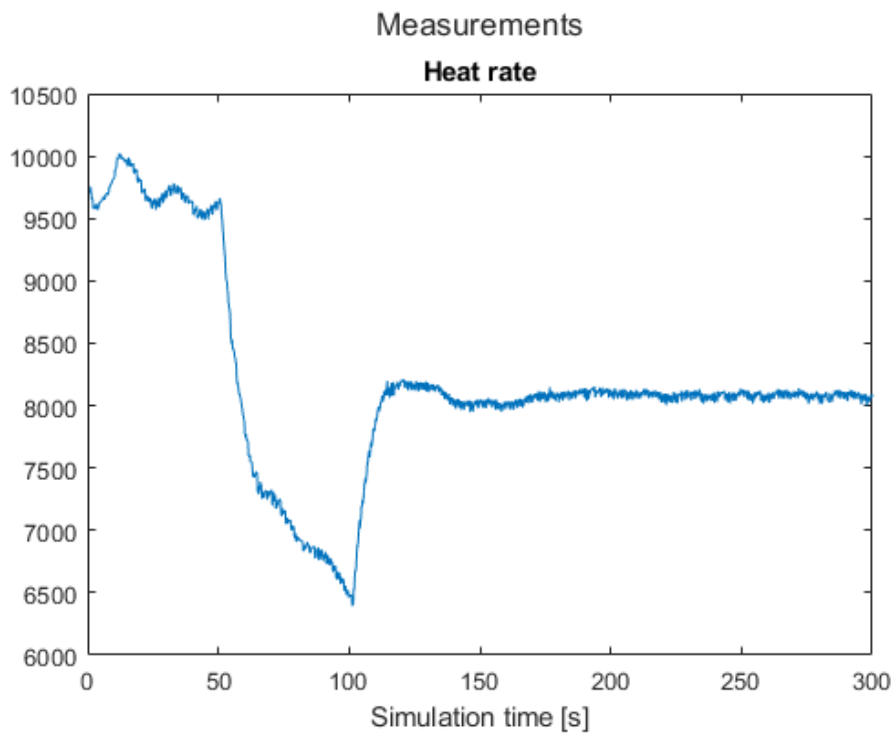
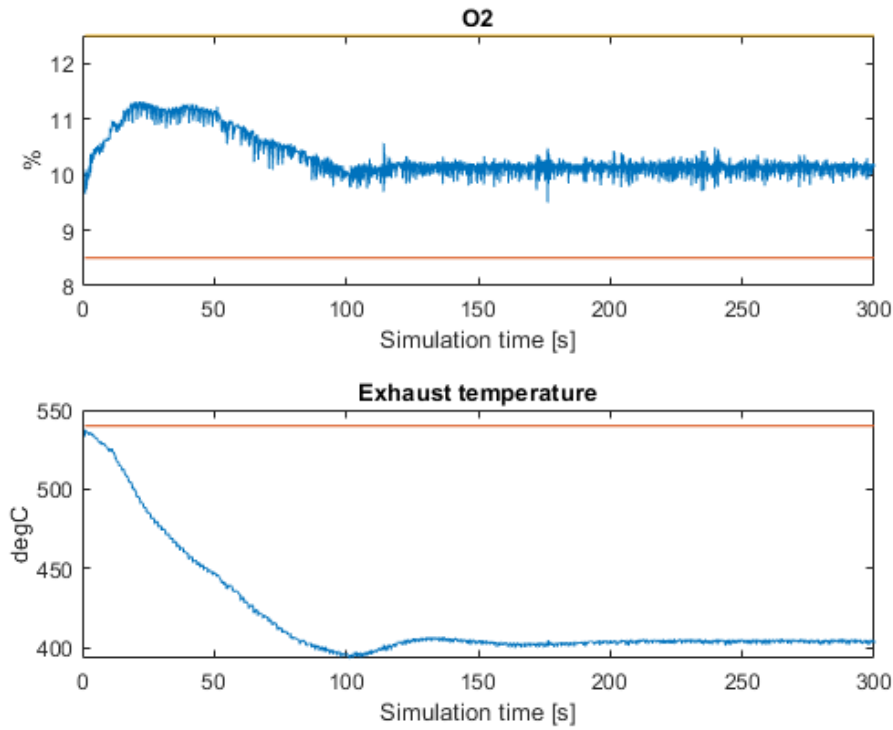


Figure 4-58: Minimized heat rate with constraints and disturbance rejection – heat rate

The shape of the heat rate is similar to the previous simulations but this time the oscillations are no longer present.



The outputs here stay well within the constraint limits even though the limit on the exhaust temperature has been lowered to 540 [°C] to have closer to the limit.

By adding real-life disturbances on the other two, *Charge air temperature* and *Suction air temperature* we will see the effects of the non-fixed values on the optimization output. From the simulation results presented in Figure 4-59 to Figure 4-62 there differences are small as the variations in the disturbances are limited. There are however some subtle differences in the response.

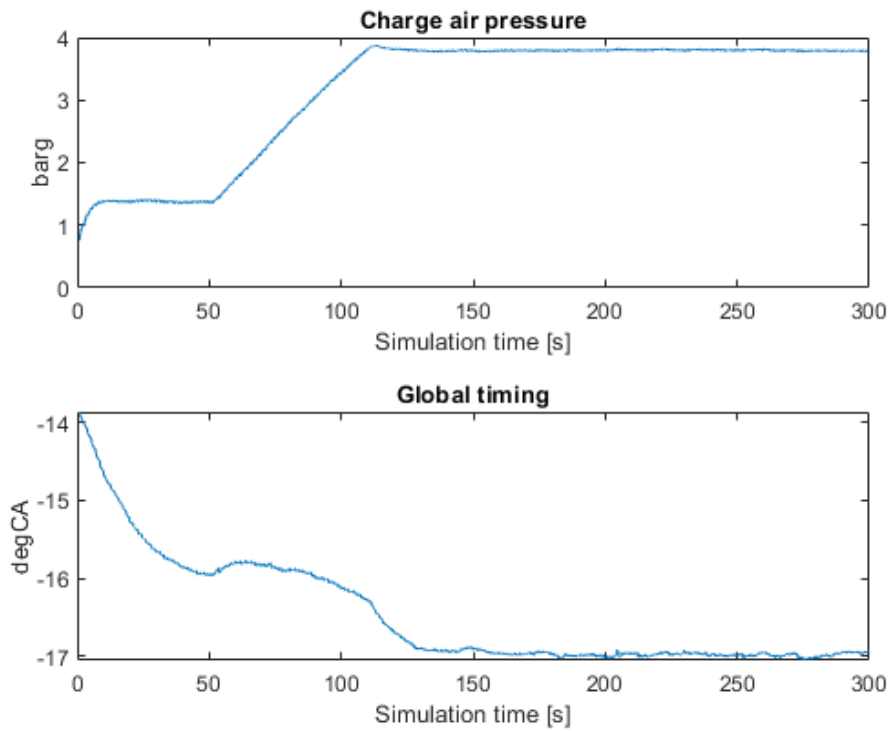


Figure 4-59: Minimize heat rate with constraint on outputs and real life disturbance - control inputs

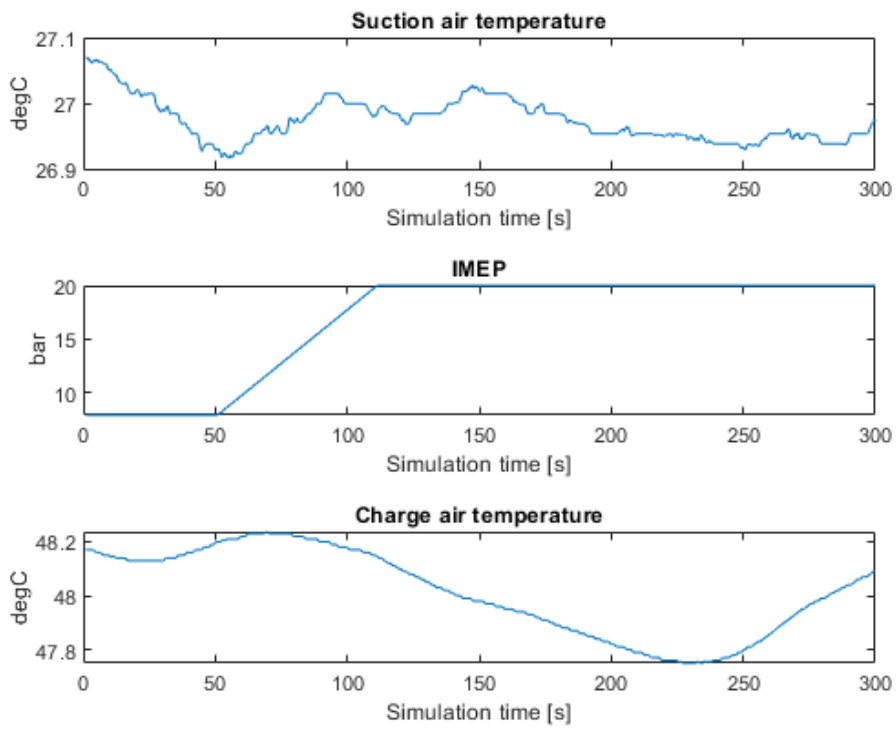


Figure 4-60: Minimize heat rate with constraint on outputs and real life disturbance - disturbances

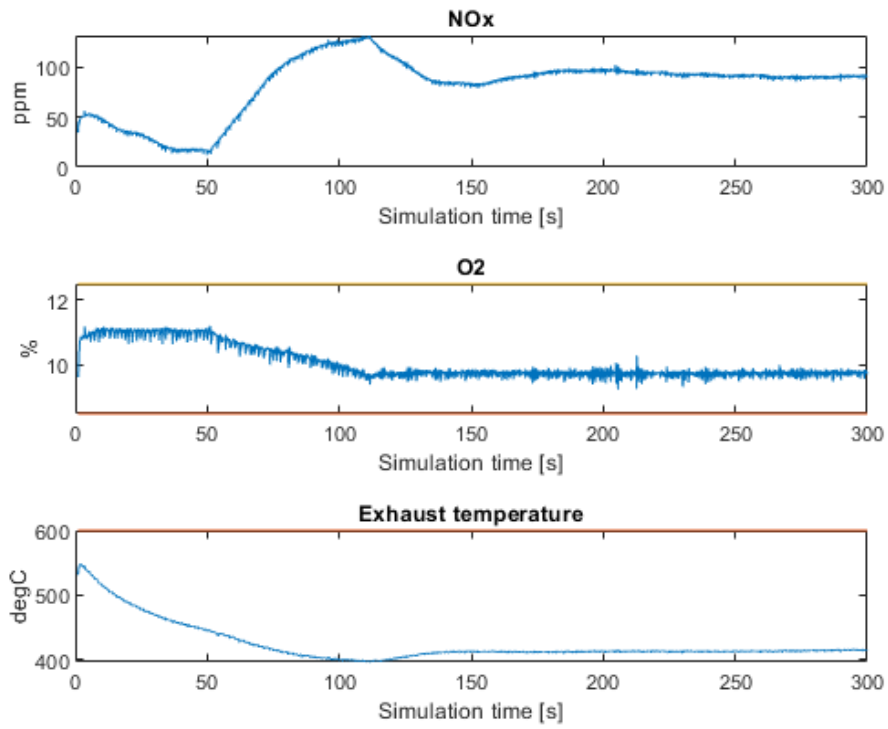


Figure 4-61: Minimize heat rate with constraint on outputs and real life disturbance - outputs

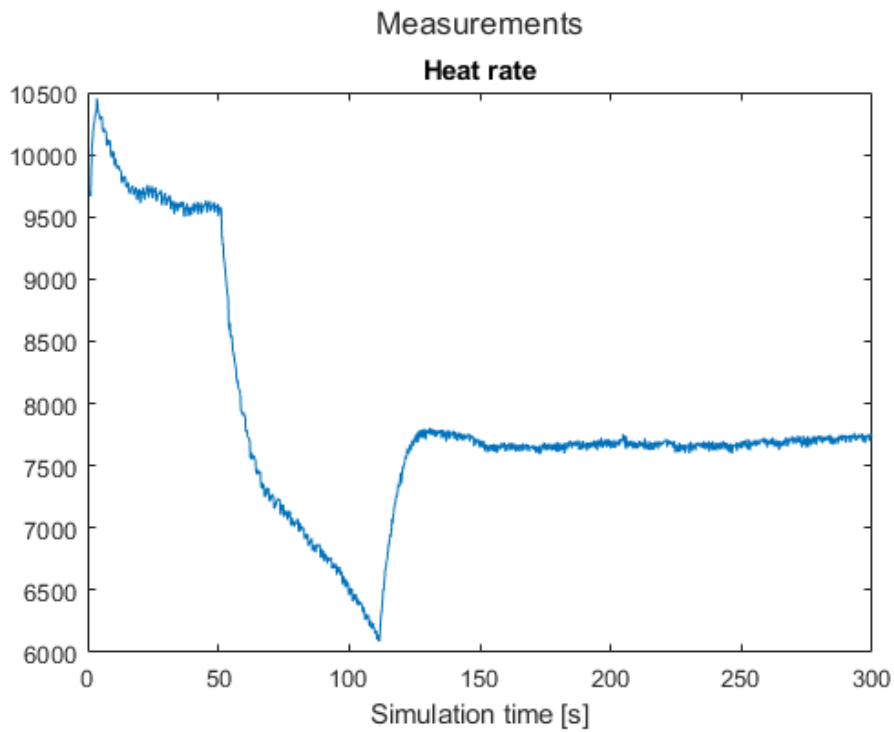


Figure 4-62: Minimize heat rate with constraint on outputs and real life disturbance – heat rate

As a test about 37000 samples of real process data with some large variation in the disturbance input was sent through the optimizer to view the outcome.

From Figure 4-63 we note the variation in the control variables as the disturbance's changes. The disturbances are given in Figure 4-64 and in particular the IMEP covers a range of operational windows from lower values to a peak at full nominal power at 1200-1500 seconds before reducing back down to low level again.

In addition to the blue curves in Figure 4-63 which shows the output from the optimizer the added red curves shows the control output from the real engine controller from period. We note the separation and differences in particular related to the engine global timing which the optimizer advances more than the current controller which should improve the fuel efficiency in theory. The charge air pressure is also slightly increased to compensate for the increase in burn rate due to the advanced timing.

It should be noted, as indicated in pervious sections, that an unlimited possibility of I increasing the charge air pressure might not be feasible due to capacity of the turbocharger and a variable upper bound in this value should be established to avoid unrealistic optimistic behavior of the optimizer.

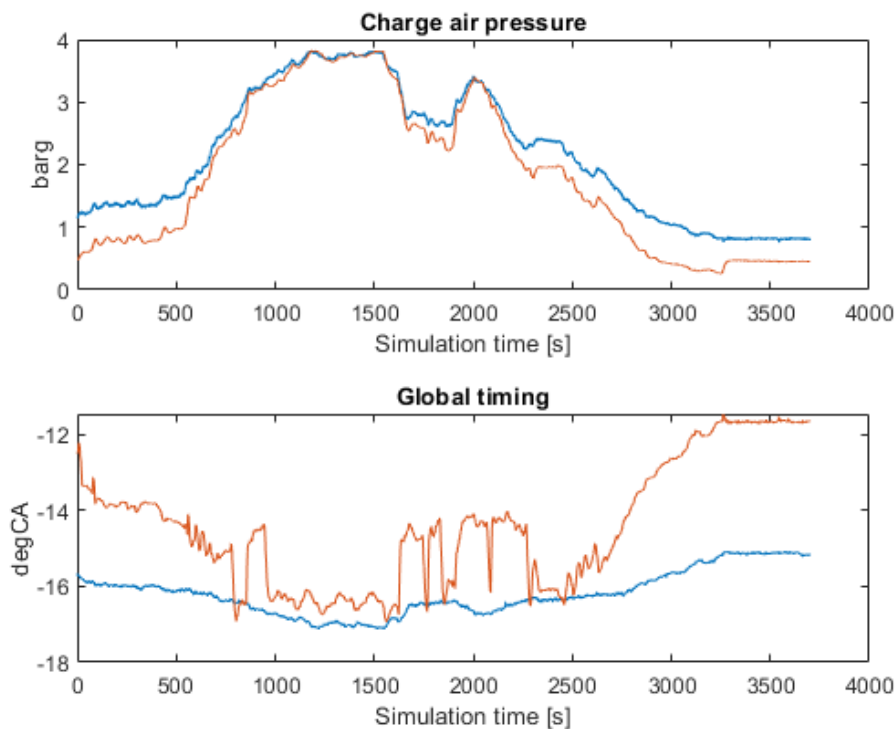


Figure 4-63: : Minimize heat rate with constraint on outputs and real life disturbance – control outputs from optimizer (blue) and real life control inputs (blue)

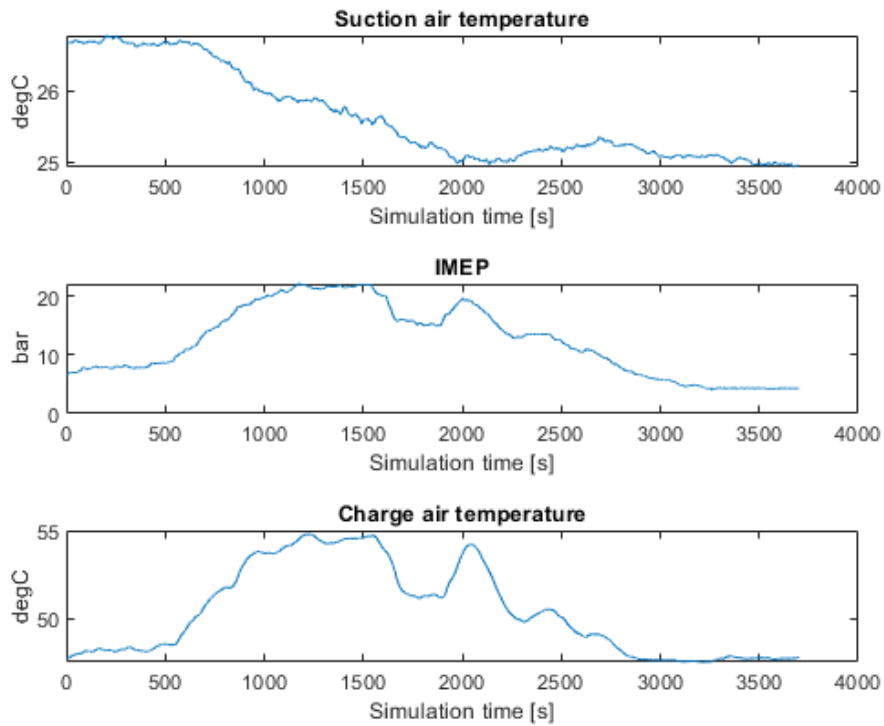


Figure 4-64: : Minimize heat rate with constraint on outputs and real life disturbance – disturbances

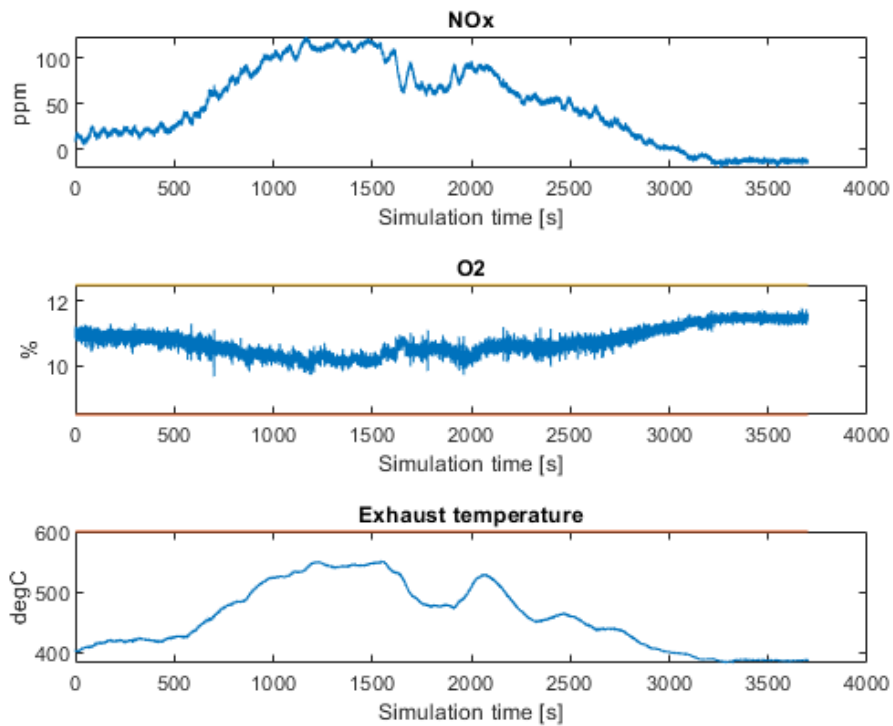


Figure 4-65: Minimize heat rate with constraint on outputs and real life disturbance – measurements

Through the simulation the outputs O2 and Exhaust temperature are kept within the constraints and hence not problem for the optimizer in that regard as are illustrated in Figure 4-65.

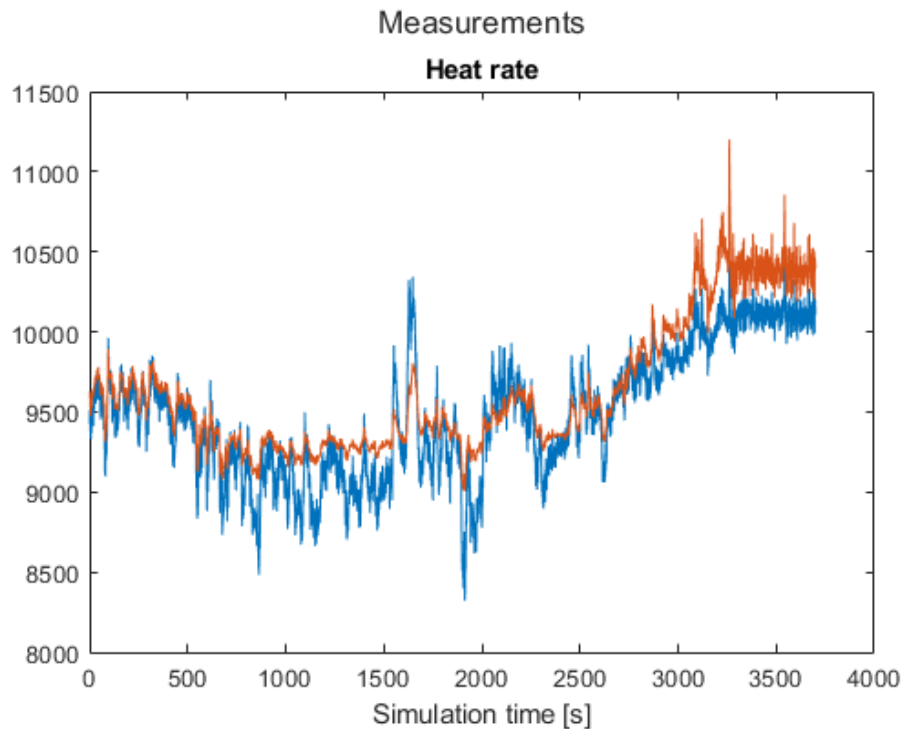


Figure 4-66: : Minimize heat rate with constraint on outputs and real life disturbance – heat rate (blue) versus measured real heat rate (red) from the same period

The heat rate output is here minimized by the optimizer and the result is shown in “blue” in Figure 4-66. In addition, the measure heat rate from the field is shown in red for the same disturbances. We note that values are closely related and share the same form. For the major part of the simulation the optimized heat rate is also lower than the measured from the installed engine running traditional control. There is a period around 2250 second mark where the estimated optimize is slightly higher than the measured.

4.7.5 Grouping of control inputs

This section will look into the possibilities for grouping the control inputs into groups in order to reduce the computational efforts by reducing the number of decision variables.

The theory behind grouping is explained in section 2.6. This section will only look at the results on reducing the computational effort by reducing the number of decision variables. In the first optimization shown below the time for the simulation of 5000 steps with a prediction horizon of 150 samples has been shown. The number of control signals are the same as before, *Charge air pressure* and *global ignition timing*.

The number of decision variables are then $150 \times 2 = 300$

The optimization is the same as the results shown in Figure 4-53 - Figure 4-55 are from.

The output from the MATLAB prompt over the simulation time of 500 seconds is given as:

```
Elapsed time is 2159.803144 seconds.
```

It is clear for this result that the MPC controller is not able to optimize the problem within the physical time of the system on the computer hardware available for the “interior point” algorithm.

In the second attempt the control signals are grouped into 5 groups of unequal length.

The first group is 5 samples, the second group is 10 samples, the third group is 20 samples, the fourth is 40 samples and the last is 75 samples. The number of decision variables then becomes $5 \times 2 = 10$.

This time the output from MATLAB prompt over the 5000 samples is given as:

```
Elapsed time is 1187.139346 seconds.
```

We see that the time it takes to do the simulation is greatly reduced while still being very high.

But what we also note is that the optimization fails completely in staying within the constraints. Figure 4-67 shows that both the *O2* and the *exhaust temperature* constraints are broken.

Several simulations were tested with grouping of control inputs, but all suffer from worse handling of constraints due to the grouping. Most of them also tended to become unstable. It might be that the implementation is not done correctly on how the problem is structured when using grouping with *Fmincon*. More testing needs to be performed here, but due to the limit time on the project, more testing at this stage was not possible.

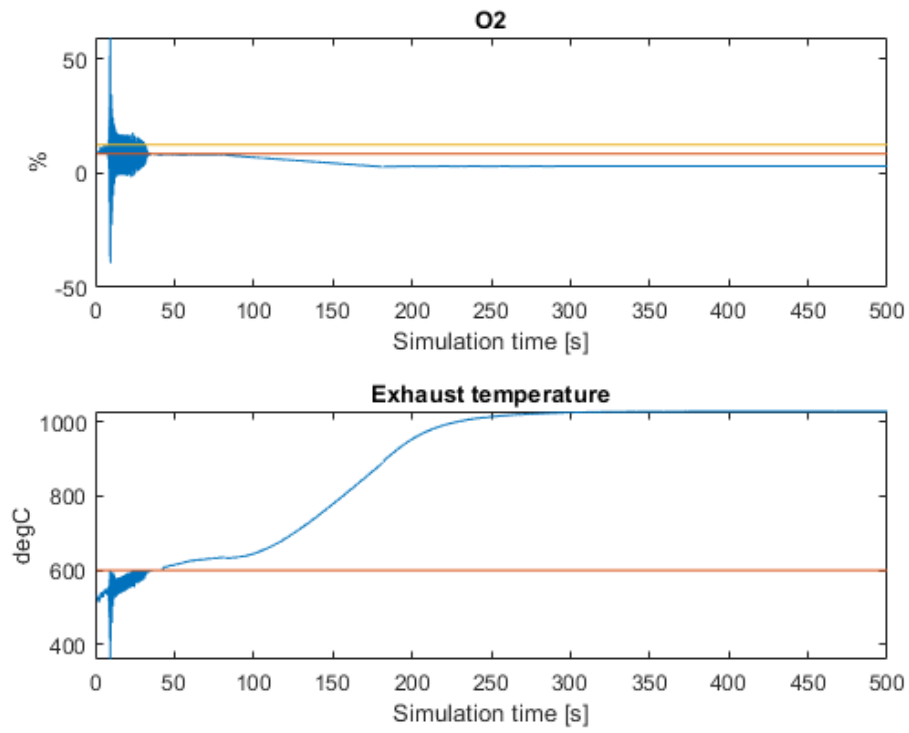


Figure 4-67: Minimized heat rate with constraints and disturbance rejection - outputs – grouping active

4.7.6 Prediction horizon

This section shows the effect of different prediction horizons on the optimization. The objective is to minimize the heat rate with constraints on the amplitude of the control signal and the measurements *O2* and *Exhaust temperature*. The disturbances are real life data and the simulation is for 15 000 steps which is equal to 1500 [s] of process data with a resolution of 0.1 [s]. The simulation time is given in Table 3 and visualized in Figure 4-68.

Table 3: Simulation time for various prediction horizons

PREDICTION HORIZON	TIME [s]
50	2950
75	3951,9
100	4808,4
125	5613
150	7359,8
175	7716,4
200	9748,5

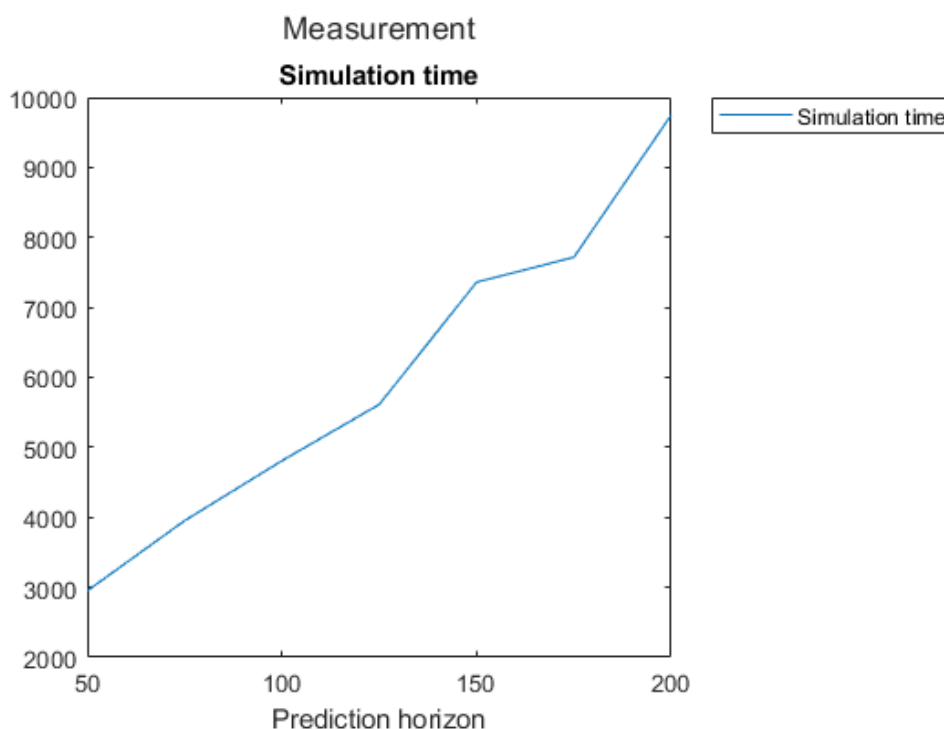


Figure 4-68: Simulation time with 15 000 steps for different prediction horizons

We note that the simulation time increases gradually until 125 where there is a steeper section until 150 steps. There might have been other processes active on the computer during the test of 150 prediction horizon so not too much should be put into. The general trend is that the prediction horizon has a big influence on the simulation time

In Figure 4-69 the two control values are given for the different prediction horizons. It is noted that from a prediction horizon of 125 the values are very much the same.

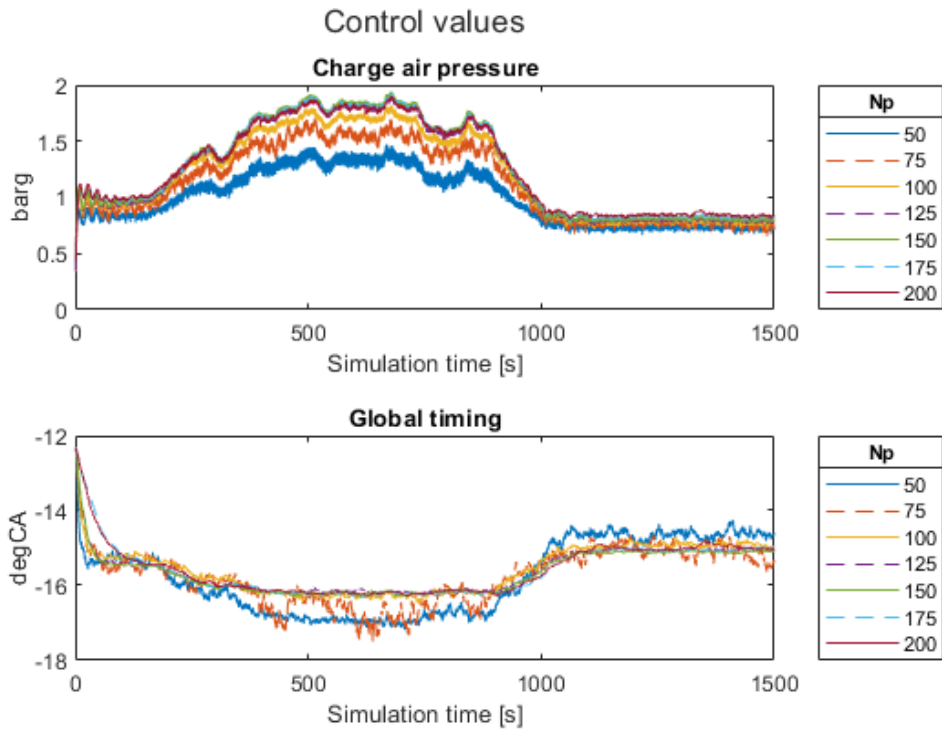


Figure 4-69: Control value outputs for different prediction horizons

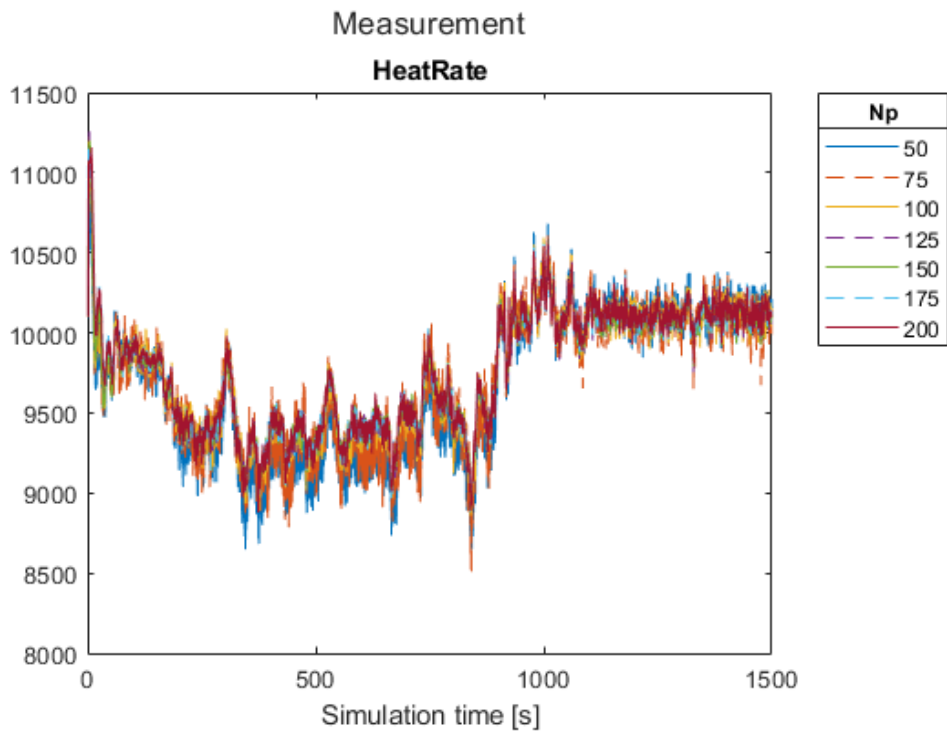


Figure 4-70: Heat rate for different prediction horizons

From Figure 4-70 it is difficult to tell if there is any big difference in the heat rate depending on the prediction horizon. If anything, it appears that the lower prediction horizon gives a lower

heart rate but in general it is difficult to tell. The average value for a prediction horizon of 50 is 9696 and for a prediction horizon of 200 the average is 9770, so basically the same value.

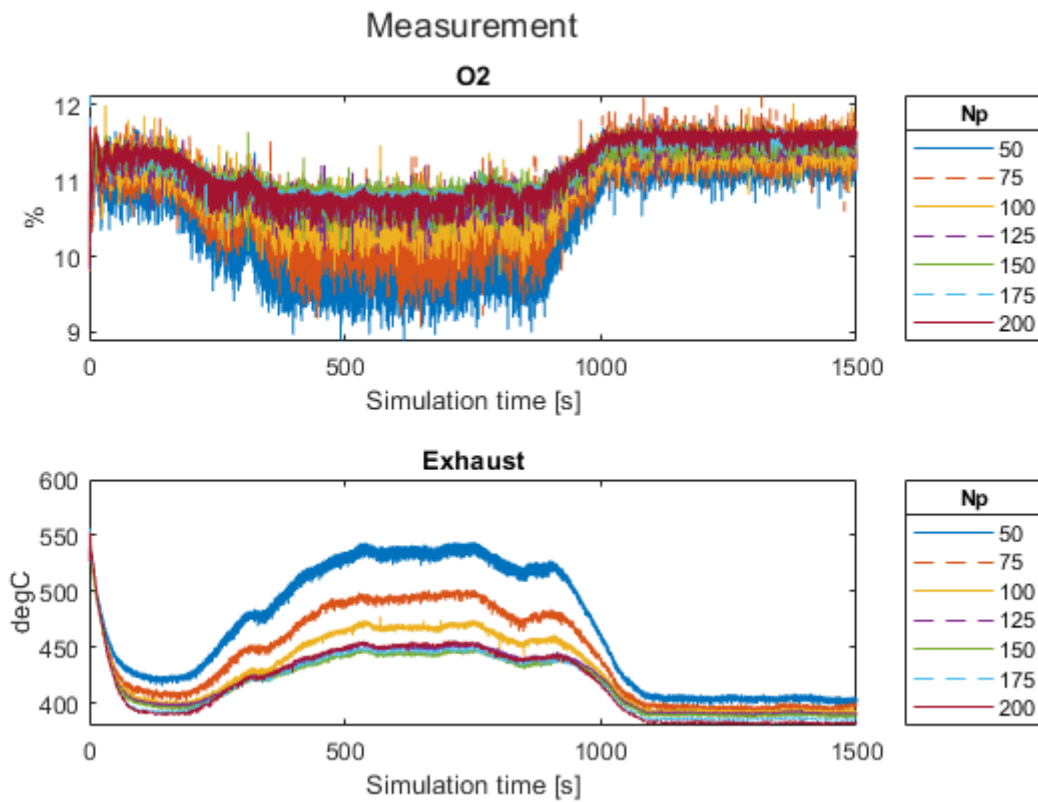


Figure 4-71: Measurements for different prediction horizons

For the two measurements mostly looked at throughout the testing, the *O2* and *Exhaust temperature* Figure 4-71 shows that these differs in the same way as the control values in Figure 4-69. From a prediction horizon of 125 and upwards they are basically the same, while for the lower prediction horizons there are larger variations.

4.8 Laguerre based Discrete Model Predictive Control

This section will look into the attempt to produce a Laguerre based DMPC. This version is based on the traditionally DMPC but where the Laguerre multipliers are used to simplify the structure.

During the study of previously MPC controllers this version emerged as a practical implementation of an industrial MPC where computational effort is reduced by reformulating the MPC state space model using Laguerre functions.

The practical implementation presented by Wang [11] looked instating and substantial amount of time was spent looking into this approach. The theory and testing are presented in the following sections.

The theory presented here also applies the traditional MPC as used in the previous chapters but are here presented in the context of the Laguerre based DMPC.

4.8.1 Classical DMPC

The classical DMPC can be formulated using a standard state space model given as in equation (4-15) where A, B, C, D represents the state space matrices, u is the control input, y is the measured output and x is the state variable vector.

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k) \\y(k) &= Cx(k) + Du(k)\end{aligned}\tag{4-15}$$

Since we are using receding horizon control, we can assume that the input $u(k)$ cannot directly affect the $y(k)$ at the same time and hence $D = 0$.

If we redefine the system to the augmented state space model, including the integrator term for offset free operation have.

$$\begin{aligned}x(k+1) &= Ax(k) + B\Delta u(k) \\y(k) &= Cx(k)\end{aligned}\tag{4-16}$$

Where the terms are defines as

$$\begin{aligned}\begin{bmatrix} \overbrace{\Delta x(k+1)}^{x(k+1)} \\ \overbrace{y(k+1)}^{y(k+1)} \end{bmatrix} &= \begin{bmatrix} \overbrace{A}^{A_e} & 0 \\ \overbrace{CA} & I \end{bmatrix} \begin{bmatrix} \overbrace{\Delta x(k)}^{x(k)} \\ \overbrace{y(k)}^{y(k)} \end{bmatrix} + \begin{bmatrix} \overbrace{B}^{B_e} \\ \overbrace{CB} \end{bmatrix} \Delta u(k) \\y(k) &= \begin{bmatrix} \overbrace{0}^{C_e} & I \end{bmatrix} \begin{bmatrix} \overbrace{\Delta x(k)}^{x(k)} \\ \overbrace{y(k)}^{y(k)} \end{bmatrix}\end{aligned}\tag{4-17}$$

If we define the current sampling instant as k_i , the prediction horizon as N_p and the control horizon as N_c , the future state variables are given in equation (4-18). For simplicity the subscript e on the matrices for the augmented version has been omitted.

$$\begin{aligned}
x(k_i + 1|k_i) &= Ax(k_i) + B\Delta u(k_i) \\
x(k_i + 2|k_i) &= Ax(k_i + 1|k_i) + B\Delta u(k_i + 1) \\
&= A^2x(k_i) + AB\Delta u(k_i) + B\Delta u(k_i + 1) \\
&\vdots \\
x(k_i + N_p|k_i) &= A^{N_p}x(k_i) + A^{N_p-1}B\Delta u(k_i) + A^{N_p-2}B\Delta u(k_i + 1) + \dots \\
&\quad + A^{N_p-N_c}B\Delta u(k_i + N_c - 1)
\end{aligned} \tag{4-18}$$

The predicted outputs are given as in equation (4-19).

$$\begin{aligned}
y(k_i + 1|k_i) &= CAx(k_i) + CB\Delta u(k_i) \\
y(k_i + 2|k_i) &= CA^2x(k_i) + CAB\Delta u(k_i) + CB\Delta u(k_i + 1) \\
y(k_i + 3|k_i) &= CA^3x(k_i) + CA^2B\Delta u(k_i) + CAB\Delta u(k_i + 1) \\
&\quad + CB\Delta u(k_i + 2) \\
&\vdots \\
y(k_i + N_p|k_i) &= CA^{N_p}x(k_i) + CA^{N_p-1}B\Delta u(k_i) + CA^{N_p-2}B\Delta u(k_i + 1) + \dots \\
&\quad + CA^{N_p-N_c}B\Delta u(k_i + N_c - 1)
\end{aligned} \tag{4-19}$$

The output prediction is now dependent on current state information and future control movement.

Defining the output vector Y and future control vector ΔU as:

$$\begin{aligned}
Y &= [y(k_i + 1|k_i) \ y(k_i + 2|k_i) \ y(k_i + 3|k_i) \ \dots \ y(k_i + N_p|k_i)]^T \\
\Delta U &= [\Delta u(k_i) \ \Delta u(k_i + 1) \ \Delta u(k_i + 2) \ \dots \ \Delta u(k_i + N_c - 1)]^T
\end{aligned} \tag{4-20}$$

Given in the compact form we have:

$$Y = Fx(k_i) + \phi\Delta U$$

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}$$

$$\phi = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \cdots & CA^{N_p-N_c}B \end{bmatrix} \quad (4-21)$$

Looking at the classical set point tracking use of the MPC, the goal of the optimizer is to minimize the error between the set point and the predictive output. Within the optimization window we keep the set point constant and let the optimizer find the ‘best’ control signals ΔU such that the error is minimized.

Defining the set point vector as constant over the prediction horizon N_p

$$R_s^T = \overbrace{[1 \ 1 \ 1 \ \dots \ 1]}^{N_p} r(k_i) \quad (4-22)$$

we get the following cost function

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U \quad (4-23)$$

This cost function will in the first term minimize the error between the set point and the estimated output and in the second term reduce the effort on the control signal needed to achieve the goal. \bar{R} is diagonal matrix with the dimension equal to the number of control signals. The elements on the diagonal is ≥ 0 and is used as tuning parameters of how much “weight” the cost function should put on minimizing the control signal in order to minimize the error between the set point and the predictive output. Setting the weighting to zero means that it does not considered the change in control signal at all. Using a large value will let the optimizer carefully change the control signal in order to reduce the error.

Substituting from (4-21) into (4-23) we get

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \phi^T (R_s - Fx(k_i)) + \Delta U^T (\phi^T \phi + \bar{R}) \Delta U \quad (4-24)$$

Taking the first derivative of J with respect to ΔU gives the optimal solution ΔU when the partial derivative is zero.

This results in the optimal solution equation

$$\Delta U = (\phi^T \phi + \bar{R})^{-1} \phi^T (R_s - Fx(k_i)) \quad (4-25)$$

The first term $(\phi^T \phi + \bar{R})^{-1}$ is also known as the Hessian matrix and R_s is set point vector.

The ΔU vector now contains all the optimal control signals over the prediction horizon. In the receding horizon control we only use the first element of the control vector before doing the optimization again using the most up to date measurements of the state and outputs.

4.8.1.1 Constraints

In many practical implementations of MPC there will be some constraints present. These constraints can be physical limitations that the system under control needs to operate within. Constraints can be on the control value; both on amplitude and rate of change; as well as on the measured outputs.

We make distinction between hard and soft constraints. Hard constraints are those that needs to be obeyed at all times. These are usually on the control signal in the form of max/min amplitude and rate of change.

The soft constraints are constraints that should not be broken if a feasible solution can be found within the constraints. If a situation occurs where a solution is not found the soft constraints can be broken; but it should be as gentle as possible. In these cases, a slack variable is often added to constraint. This slack variable is then used as part of the cost function so that the amount of slack on the soft constraint is minimized.

The constraint on the amplitude of the control signal can be formulated as

$$u^{min} \leq u(k) \leq u^{max} \quad (4-26)$$

and constraint on the rate of change

$$\Delta u^{min} \leq \Delta u(k) \leq \Delta u^{max} \quad (4-27)$$

Sometime there is also output constraints as

$$y^{min} \leq y(k) \leq y^{max} \quad (4-28)$$

and with the slack variable s this becomes

$$y^{min} - s \leq y(k) \leq y^{max} + s \quad (4-29)$$

In MIMO¹⁰ systems the constraints are defined for each control signal and output signal individually.

¹⁰ MIMO – Multiple Input Multiple Output

To add the constraints to the optimal control problem they are formulated into inequalities with respect to the parameter vector ΔU ; also known as the decision variable vector.

The constraints are formulated for the entire prediction horizon and applied at each sample interval in the receding horizon strategy. They can be changed and updated between each optimization.

If the computational load is high, they could be imposed only on a subset of the prediction horizon as it is only the first element of the control vector that is used.

To formulate the rate of change on the control signals as part of the control problem we get

$$\begin{aligned} \Delta U^{min} &\leq \Delta U \leq \Delta U^{max} \\ &= \\ -\Delta U &\leq -\Delta U^{min} \\ \Delta U &\leq \Delta U^{max} \end{aligned} \quad (4-30)$$

and in matrix form

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U = \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix} \quad (4-31)$$

The same applies to all amplitudes of the control signal. From (4-26)

$$U^{min} \leq U \leq U^{max} \quad (4-32)$$

If we defined the vector U as a function of ΔU

$$u(k+l) = u(k-1) + \sum_{i=0}^l \Delta u(k+i) \quad (4-33)$$

We get

$$\begin{bmatrix} U(k) \\ U(k+1) \\ U(k+2) \\ \vdots \\ U(k+N_c-1) \end{bmatrix} = \begin{bmatrix} I_{N_c} \\ I_{N_c} \\ I_{N_c} \\ \vdots \\ I_{N_c} \end{bmatrix} U(k-1) + \begin{bmatrix} I_{N_c} & 0 & \cdots & 0 \\ I_{N_c} & I_{N_c} & \cdots & 0 \\ I_{N_c} & I_{N_c} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I_{N_c} & I_{N_c} & \cdots & I_{N_c} \end{bmatrix} \begin{bmatrix} \Delta U(k) \\ \Delta U(k+1) \\ \Delta U(k+2) \\ \vdots \\ \Delta U(k+N_c-1) \end{bmatrix} \quad (4-34)$$

We can rewrite (4-34) in compact form as where C_1 and C_2 corresponding to the appropriate matrices

$$\begin{bmatrix} -C_2 \\ C_2 \end{bmatrix} \Delta U \leq \begin{bmatrix} -U^{min} + -C_1 u(k_i - 1) \\ U^{max} - C_1 u(k_i - 1) \end{bmatrix} \quad (4-35)$$

The output constraints are given here following from (4-28) and (4-24)

$$Y^{min} \leq Fx(k) + \phi \Delta U \leq Y^{max} \quad (4-36)$$

Reformulating into matrix form

$$\begin{bmatrix} -\phi \\ \phi \end{bmatrix} \Delta U \leq \begin{bmatrix} -Y^{min} + Fx(k) \\ Y^{max} - Fx(k) \end{bmatrix} \quad (4-37)$$

Adding all the inequality constraints together

$$\begin{bmatrix} -I \\ I \\ -C_2 \\ C_2 \\ -\phi \\ \phi \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \\ -U^{min} + -C_1 u(k_i - 1) \\ U^{max} - C_1 u(k_i - 1) \\ -Y^{min} + Fx(k) \\ Y^{max} - Fx(k) \end{bmatrix} \quad (4-38)$$

In short, we can write this as

$$M \Delta U \leq \gamma \quad (4-39)$$

In quadratic programming the optimization problem is given as

$$J = \frac{1}{2} x^T E x + x^T F \quad (4-40)$$

$$M x \leq \gamma$$

Where the decision variable is given as x and not ΔU .

We say that an inequality constraint becomes active if $Mx = \gamma$. The constraint is inactive if $Mx < \gamma$. Often the number of inequality constraints can be larger than the number of decision variables; M is then a matrix with rows equal to number of constraints and γ is then a column vector with length equal to the number of inequality constraints.

4.8.1.2 Lagrange multipliers

The Lagrange multipliers is a strategy in optimization theory to find the local maxima or minima of a function subject to equality constraints. It is widely used to solve complicated constrained problems and by using the Kuhn-Tucker conditions it is extended to the inequality cases [15].

In general, for the equality constraint case ($Mx - \gamma = 0$) the following objective function is given with the Lagrange expression

$$J = \frac{1}{2} x^T E x + x^T F + \lambda^T (Mx - \gamma) \quad (4-41)$$

To minimize this function using partial derivative we get

$$\frac{\partial J}{\partial x} = Ex + F + M^T \lambda = 0 \quad (4-42)$$

$$\frac{\partial J}{\partial \lambda} = Mx - \gamma = 0$$

The optimal λ and x are then found as

$$\begin{aligned} \lambda &= -(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}F) \\ x &= -E^{-1}(M^T \lambda + F) \end{aligned} \quad (4-43)$$

Here λ are the Lagrange multipliers.

For the inequality constraints using the Kuhn-Tucker conditions

$$\begin{aligned} Ex + F + M^T \lambda &= 0 \\ Mx - \gamma &\leq 0 \\ \lambda^T (Mx - \gamma) &= 0 \\ \lambda &\geq 0 \end{aligned} \quad (4-44)$$

Using the definition of inactive and active constraints we can define (4-44) with the help of the Lagrange multipliers. Here i denotes the inequality number in a multiple constraint problem

$$\begin{aligned} Ex + F + \sum_{i \in S_{act}} \lambda_i M_i^T &= 0 \\ M_i x - \gamma_i &= 0 \text{ Active constraint} \\ M_i x - \gamma_i &< 0 \text{ Constraint satisfied} \\ \lambda_i &\geq 0 \text{ Active constraint} \\ \lambda_i &= 0 \text{ Inactive constraint} \end{aligned} \quad (4-45)$$

If the active constraints are given, then the optimal solution is also given

$$\begin{aligned} \lambda &= -(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}F) \\ x &= -E^{-1}(F + M^T \lambda) \end{aligned} \quad (4-46)$$

The **active set** method used in the QP problem defines a subset of the constraints for each step in the algorithm that is active at the current step. This set is also known as the working set. The solution is started off with a subset that is feasible. The algorithm then moves on the surface defined by the working set to an improved point. At each step an equality constraint is solved. If all the Lagrange multipliers are greater or equal to zero ($\lambda_i \geq 0$) then the point is a local solution. If the Lagrange multiplier is less than zero, then the objective function can be decreased by relaxing the constraint i , i.e. removing it from the constraint equation.

During the optimization all constraints must be monitored in order not to break any of them. If a new constraint is detected it needs to be added to the working set.

In the active set methods, the active constraints need to be identified as well as the optimal decision variables. It is clear that if there are many constraints the computational load increases. The Lagrange multipliers are here used to detect the constraints which are not active and hence can be removed from the solution. The Lagrange multipliers are called the dual variables [11].

4.8.2 DMPC with Laguerre functions

From (4-20) we had that

$$\Delta U = [\Delta u(k_i) \Delta u(k_i + 1) \Delta u(k_i + 2) \dots \Delta u(k_i + N_c - 1)]^T \quad (4-47)$$

where N_c is the control horizon. At any discrete time k_i we can define an element in ΔU as a function of ΔU using the pulse operator such that

$$\Delta u(k_i + i) = [\delta(i) \delta(i - 1) \dots \delta(i - N_c + 1)] \Delta U \quad (4-48)$$

What is proposed in [11] is to use a set of Laguerre functions to approximate the sequence $\Delta u(k_i) \Delta u(k_i + 1) \Delta u(k_i + 2) \dots \Delta u(k_i + N_c - 1)$. The reason for this is given as the number of parameters required to approximate the control signal Δu is far less than the number traditionally used in MPC design for instance using the pulse operator. Using Laguerre polynomial reduces the number of parameters and hence the computational load which might be critical for processes with complicated dynamics and/or high demand on closed loop performance.

The discrete Laguerre functions are given as

$$L(k + 1) = A_l L(k) \quad (4-49)$$

Here A_l is a matrix consisting of the parameters a and β for the case of $N = 4$

$$A_l = \begin{bmatrix} a & 0 & 0 & 0 \\ \beta & a & 0 & 0 \\ -a\beta & \beta & a & 0 \\ a^2\beta & -a\beta & \beta & a \end{bmatrix} \quad (4-50)$$

The initial condition $L(0)$ is given as

$$L(0) = \sqrt{\beta} \begin{bmatrix} 1 \\ -a \\ a^2 \\ -a^3 \end{bmatrix} \quad (4-51)$$

At a sample instance k with initial time k_i

$$\Delta u(k_i + k) = \sum_{j=1}^N c_j(k_i) l_j(k) \quad (4-52)$$

where N is the number of terms in the expansion and c_j is the coefficients which are function of the initial time of the moving horizon window, k_i . l_j are here the Laguerre functions. Equation (4-52) can also be written in vector form as:

$$\Delta u(k_i + k) = L(k)^T \eta \quad (4-53)$$

where $\eta = [c_1 \ c_2 \ \dots \ c_N]^T$, the Laguerre functions, and $L(k)^T$ is the transpose Laguerre function vector in (4-49). The control horizon, N_c , is no longer used and replaced by N and a . Here a large value on a will give a long control horizon. If we set $a = 0$ then $N = N_c$ as traditionally.

We can now add the Laguerre functions to the augmented state space formulation where the initial state is k_i as:

$$\begin{aligned} x(k_i + m|k_i) &= A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta \\ y(k_i + m|k_i) &= C A^m x(k_i) + \sum_{i=0}^{m-1} C A^{m-i-1} B L(i)^T \eta \end{aligned} \quad (4-54)$$

Here the Δu is now longer present and hence it is the coefficient vector η that needs to be optimized. We therefor rewrite the cost function accordingly

$$J = \sum_{m=1}^{N_p} (r(k_i) - y(k_i + m|k_i))^T (r(k_i) - y(k_i + m|k_i)) + \eta^T R_L \eta \quad (4-55)$$

R_L is the weighting matrix which is a diagonal matrix of size $N \times N$ and with diagonal elements which is ≥ 0 . $r(k_i)$ is the set point at time instance k_i .

In [11] this traditional cost function is reformulated in order to form at link to DLQR¹¹ where the objective is to find the optimal coefficient vector η that minimizes the cost. The reason behind this reformulation is explained as numerous DLQR classical results can be utilized for analysis, tuning and design of the MPC.

The new cost function is then given as:

$$J = \sum_{m=1}^{N_p} x(k_i + m|k_i)^T Q x(k_i + m|k_i) + \eta^T R_L \eta \quad (4-56)$$

¹¹ DLQR – Discrete time Linear Quadratic regulators.

Now in order to include the set point in the cost function the state variable needs to be re-defined.

In the augmented state space model given in (4-17) we defined the output matrix, C as $[0 \ 0 \ \dots \ 0 \ I]$ and the state vector as $[\Delta x(k)^T \ y(k)^T]$. If we now define a new vector with the reference signal (which is unchanged over the prediction horizon) as $x_r(k_i) = [0 \ 0 \ \dots \ 0 \ r(k_i)]^T$. This vector has the number of zeroes equal to the dimensions of the state vector which hence leads to $r(k_i) = Cx_r(k_i)$. The new state vector is now:

$$x(k_i + m|k_i) = [\Delta x_m(k_i + m|k_i)^T \ y(k_i + m|k_i) - r(k_i)]^T \quad (4-57)$$

We can from (4-54) simplify the next state equation by letting

$$\phi(m)^T = \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \quad (4-58)$$

We then have

$$x(k_i + m|k_i) = A^m x(k_i) + \phi(m)^T \eta \quad (4-59)$$

Since we now have the new state equation in the cost function we would also need to update that accordingly

$$\begin{aligned} J = \eta^T & \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \eta + 2\eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i) \\ & + \sum_{m=1}^{N_p} x(k_i)^T (A^T)^m Q A^m x(k_i) \end{aligned} \quad (4-60)$$

We now define two new variables

$$\begin{aligned} \Omega &= \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \\ \Psi &= \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) \end{aligned} \quad (4-61)$$

Taking the partial derivative of (4-60) with respect to the optimal parameter vector η we get

$$\eta = -\Omega^{-1} \Psi x(k_i) \quad (4-62)$$

Finding the minimum of the cost function J based on (4-60) now gives us

$$J = (\eta + \Omega^{-1}\Psi x(k_i))^T \Omega (\eta + \Omega^{-1}\Psi x(k_i)) - x(k_i)^T \Psi^T \Omega^{-1} \Psi x(k_i) + \sum_{m=1}^{Np} x(k_i)^T (A^T)^m Q A^m x(k_i) \quad (4-63)$$

Since the optimal parameter vector is given when as in (4-62) we get

$$J = x(k_i)^T \left(\sum_{m=1}^{Np} (A^T)^m Q A^m - \Psi^T \Omega^{-1} \Psi \right) x(k_i) = x(k_i)^T P_{dmpc} x(k_i) \quad (4-64)$$

Here P_{dmpc} is the content of the large brackets.

With the formulation in (4-54) we noted that the Δu was replaced by the coefficient vector η of the Laguerre network and in the following sections we showed how this was done by adding it to the cost function. Now in order to predict the future x we need to solve the convolution sum of

$$S_c(m) = \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \quad (4-65)$$

From (4-49) we had that $L(k+1) = A_l L(k)$

Studying the convolution sum we finally get for $m = 2, 3, 4 \dots N_p$

$$S_c(m) = A S_c(m-1) + S_c(1) (A_l^{m-1})^T \quad (4-66)$$

For $m = 1$ we have $S_c(1) = B L(0)^T$

Now for the receding horizon strategy we end up with

$$\Delta u(k_i) = L(0)^T \eta \quad (4-67)$$

For the multiple input and multiple output case the Laguerre tuning parameters (a and N) are given separately for each control signal. The input matrix B is separated into m columns; one for each input.

The control signal increment is given as

$$\Delta u_i(k) = L_i(0)^T \eta_i \quad (4-68)$$

where m denotes the i^{th} control input.

$$L_i(k)^T = [l_1^i(k) l_n^i(k) \dots l_{N_i}^i(k)] \quad (4-69)$$

For the simplified state equation in (4-59) we now have

$$\phi(m)^T = \sum_{i=0}^{m-1} A^{m-i-1} [B_1 L_1(i)^T \ B_2 L_2(i)^T \ \dots \ B_m L_m(i)^T] \quad (4-70)$$

For each “block” in the $\phi(m)^T$ matrix the structure is identical to the SISO case for the convolution sum in (4-66) and the cost function in (4-60) remains the same.

In Matrix form this is given as:

$$\Delta u(k_i) = \begin{bmatrix} L_1(0)^T & 0_2^T & \cdots & 0_m^T \\ 0_1^T & L_2(0)^T & \cdots & 0_m^T \\ \vdots & \vdots & \ddots & \vdots \\ 0_1^T & 0_2^T & \cdots & L_m(0)^T \end{bmatrix} \eta \quad (4-71)$$

In Appendix B

the MATLAB function for calculating the Ω and Ψ is given. These are used in the cost function $J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i)$; based on (4-60). The code here is based on the code given in [11].

The inputs to the function are:

- A – State matrix – extended if augmented state space is used
- B – Input matrix – extended if augmented state space is used
- a – vector of Laguerre pole location for each input
- N – The number of terms in the Laguerre function for each input
- N_p – Prediction horizon
- Q – Weighting matrix for the states
- R – Weighting matrix for the input signals

4.8.2.1 Set point tracking without constraints

A test function is written in order to test function. The model is based on the State Space model used throughout this report.

Firstly, the augmented state space model is formulated

```
%State matrix
A = ss1.A;

%Inputs
B = ss1.B;

%outputs
C = ss1.C;

%Extract the sizes of the matrices
[y_n,n_n] = size(C);
[n_n,u_n] = size(B);
```

Create the augmented matrices with integrator terms

The new matrices then becomes $A_e = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix}$, $B_e = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix}$, and $C_e = [0 \quad I]$

```
A_e=eye(n_n+y_n,n_n+y_n);
A_e(1:n_n,1:n_n) = A;
A_e(n_n+1:n_n+y_n,1:n_n) = C*A;

B_e=zeros(n_n+y_n,u_n);
B_e(1:n_n,:)=B;
B_e(n_n+1:n_n+y_n,:)=C*B;

C_e(:,n_n+1:n_n+y_n)=eye(y_n,y_n);
```

We then set up the control variables, Q and R weighting matrices and find the Ω and Ψ with the dmPC function. The source code for this function is found in

In this section we also set up the Laguerre vector and number of terms in the Laguerre function.

```
Q = C_e'*C_e;

R = 0.1*eye(u_n,u_n);

a = [0.5 0.5 0.5 0.5 0.5];
N = [15 15 15 15 15];

Np = 100; %Preciction horizon
[Omega,Psi] = dmPC(A_e,B_e,a,N,Np,Q,R);
```

We set the prediction horizon to 200.

The next part of the function is to find the A_l matrix and L_m for the Laguerre functions.

```
[A1,L0] = lagd(a(1),N(1));

L_m=zeros(u_n,sum(N));
L_m(1,1:N(1)) = L0';

In_s = 1;

for jj =2:u_n
    [A1,L0] = lagd(a(jj),N(jj));
    In_s = N(jj-1)+In_s;
    In_e = In_s+N(jj)-1;
    L_m(jj,In_s:In_e) = L0';
end
```

We then defined the initial condition based on real life data so that we use valid operational data as a starting point. Firstly, we let MATLAB find the initial value from a dataset based on 30 samples. Then we take known 10 control inputs and run through the model with the firstly know initial state so that we are sure that the starting point is a hot starting point.

We define the first state vector on deviation form since we are using the augmented state space model. We also add the error for the first output as this is what we want to track and minimize. The other outputs are added as pure measurements.

```
X0_SS1 =
findstates(ss1,iddata(TrainingMean20200903Red.OutputData(161:190,1:6),Trainin
gMean20200903Red.InputData(161:190,1:5),0.1));

state_ini_values = X0_SS1;

y = zeros(y_n,1);
y = TrainingMean20200903Red.OutputData(201,1:6);
y = y';

u = zeros(u_n,1);

u = [TrainingMean20200903Red.InputData(201,1); ...
    TrainingMean20200903Red.InputData(201,2); ...
    TrainingMean20200903Red.InputData(201,3); ...
    TrainingMean20200903Red.InputData(201,4); ...
    TrainingMean20200903Red.InputData(201,5)];

N_sim = 1000;
```

```

sp = 10000*ones(1,N_sim+10);

for ii = 1:10
    state_ini_values_old = state_ini_values;
    state_ini_values =
ss1.A*state_ini_values+ss1.B*TrainingMean20200903Red.InputData(190+ii,1:5)';
end

xm = [state_ini_values-state_ini_values_old;y(1,1)-
sp(1,1);y(2,1);y(3,1);y(4,1);y(5,1);y(6,1)];

```

The simulation part of the unconstrained test is shown below

```

for kk = 1:N_sim

    u = [TrainingMean20200903Red.InputData(200+kk,1); ...
        TrainingMean20200903Red.InputData(200+kk,2); ...
        u(3); ...
        TrainingMean20200903Red.InputData(200+kk,4); ...
        u(5)];

    eta = -(Omega\Psi)*xm;
    deltau = L_m*eta;

    u(3) = u(3)+deltau(3);
    u(5) = u(5)+deltau(5);

    %save values
    deltau1(:,kk) = deltau;
    sp1(1,kk) = sp(1,1);
    u1(1:u_n,kk) = u;
    y1(1:y_n,kk) = y;
    state_ini_values1(1:n_n,kk) = state_ini_values;

    %Plant simulation
    yp = y;
    state_ini_values_old = state_ini_values;
    state_ini_values = ss1.A*state_ini_values+ss1.B*u;
    y = ss1.C*state_ini_values_old;

    xm = [state_ini_values-state_ini_values_old;y(1,1)-
sp(1,1);y(2,1);y(3,1);y(4,1);y(5,1);y(6,1)];
end

```

In Figure 4-72, Figure 4-73 and Figure 4-74 we see the output from the test function for unconstrained set point tracking. We see that it tracks set point good after some initial oscillations on the heat rate

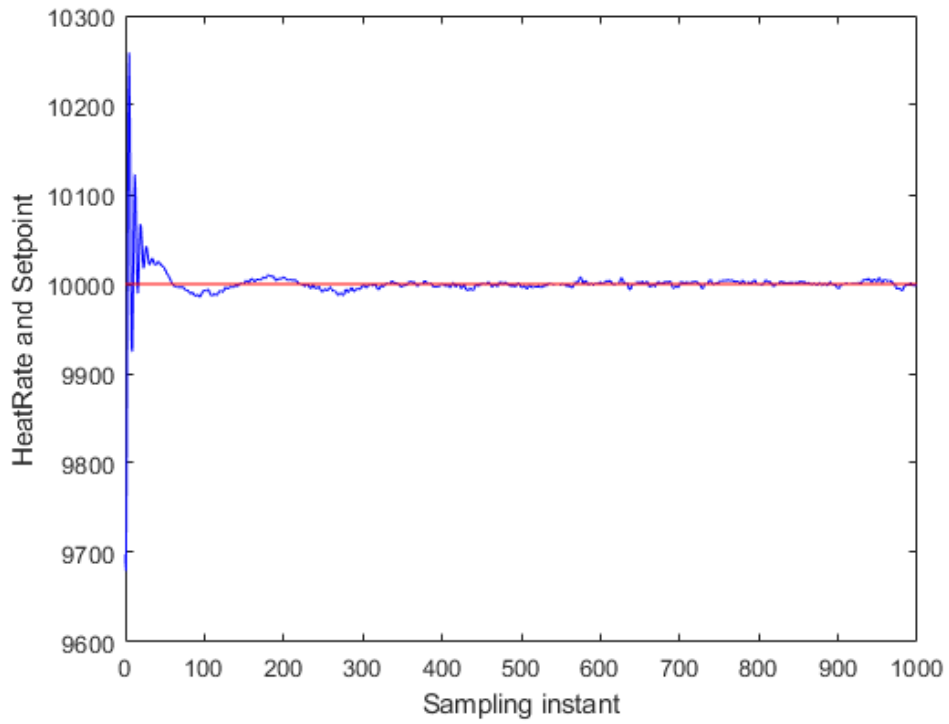


Figure 4-72: Unconstrained set point tracking with DMPC and Laguerre functions – Heat rate and set point

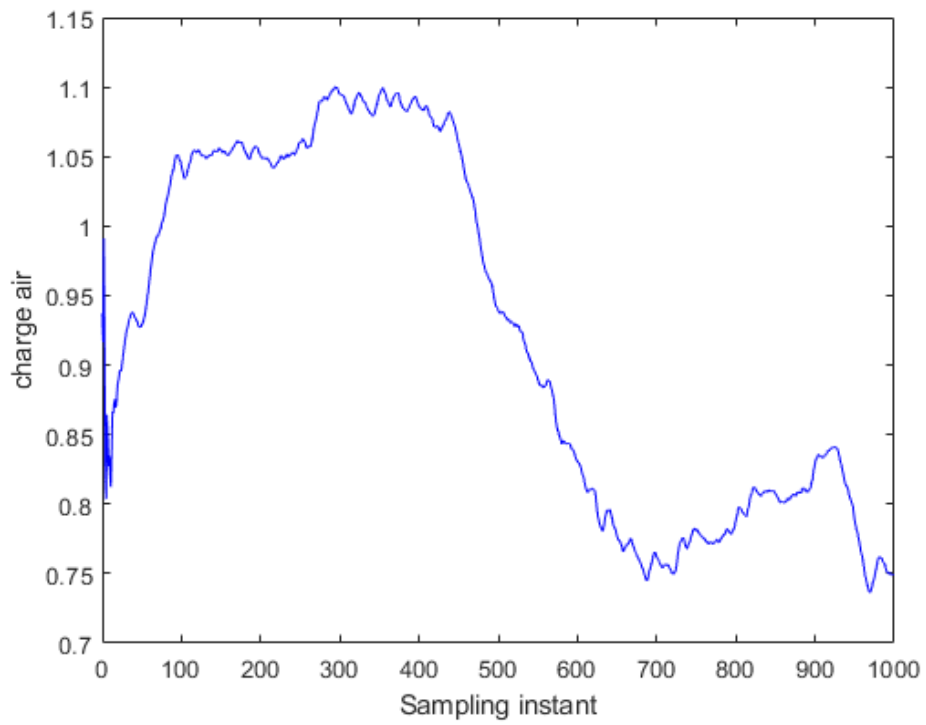


Figure 4-73: Unconstrained set point tracking with DMPC and Laguerre functions - charge air control output

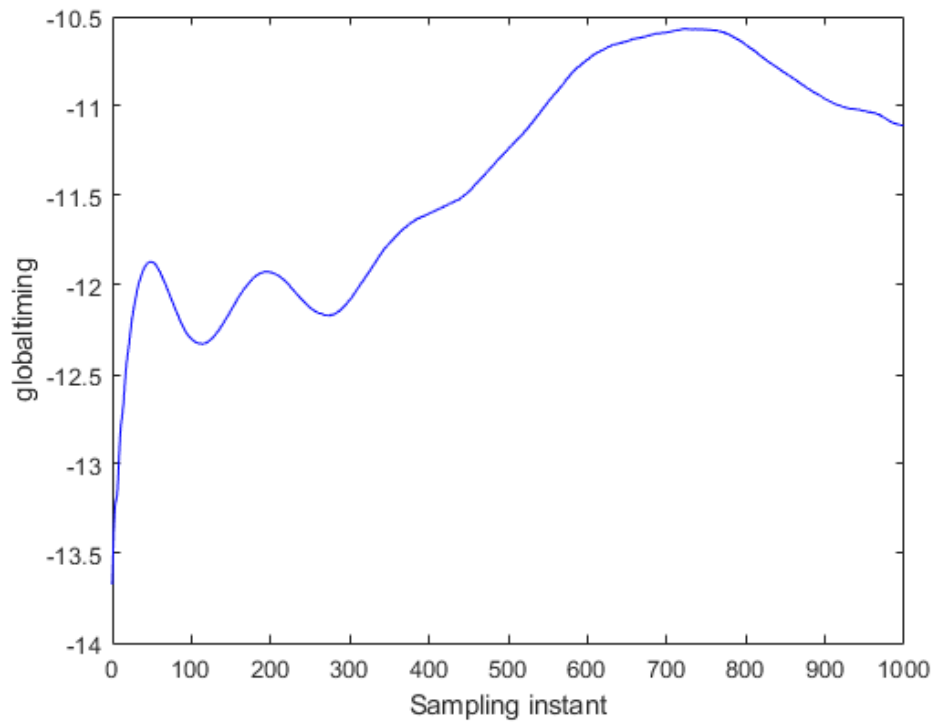


Figure 4-74: Unconstrained set point tracking with DMPC and Laguerre functions - global timing control signal

By adding a step change on the heat rate set point we get the results as shown in Figure 4-75. Still we see that the set point tracks good with some initial oscillations after set point change.

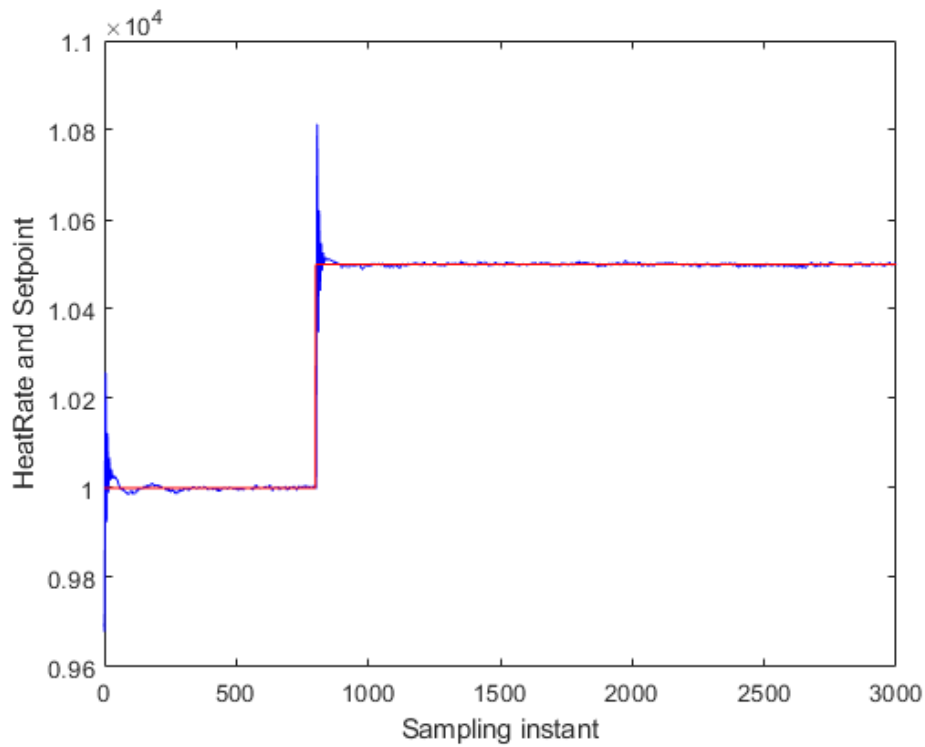


Figure 4-75: Unconstrained set point tracking with step change in set point - Heat rate

To test disturbance rejection, we change the IMEP values as we did in test in section 4.7.1. During this test we keep the disturbances fixed during the simulation except for the step change on IMEP at 100 seconds. We see from Figure 4-76, Figure 4-77, Figure 4-78 and Figure 4-79 that the sudden change results in a short time change in heat rate output before quickly settle down on set point again. The outputs adjust immediately and keeps the value on set point.

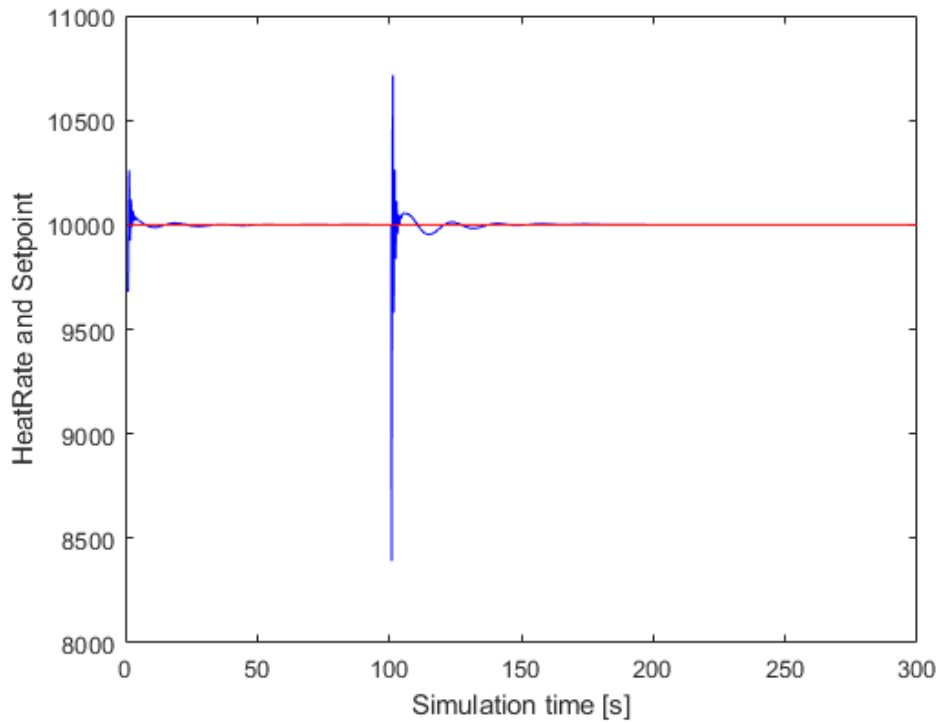


Figure 4-76: Unconstrained set point tracking with disturbance rejection - heat rate

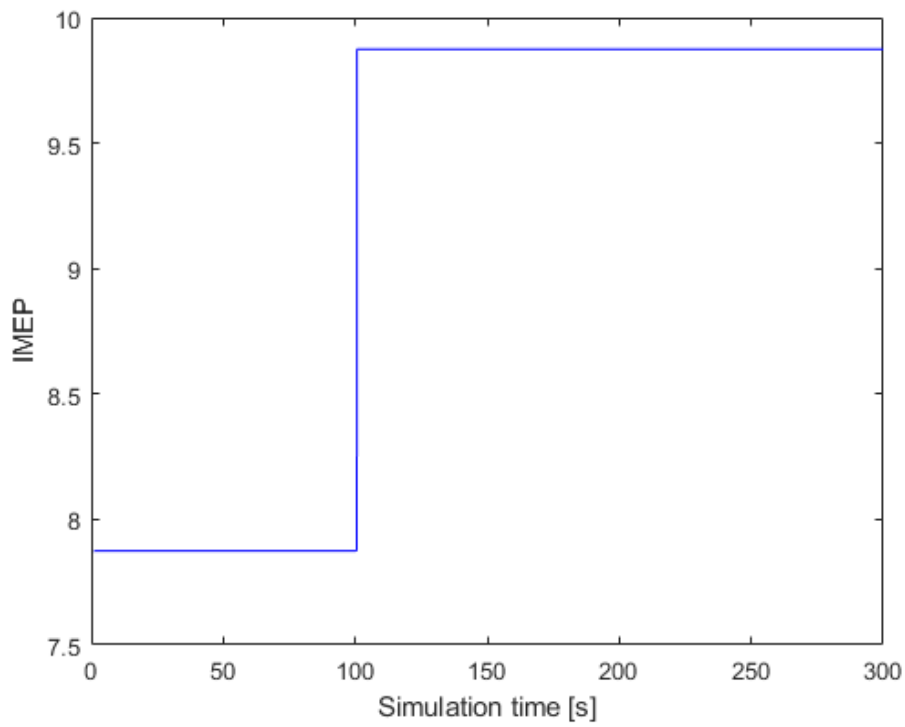


Figure 4-77: Unconstrained set point tracking with disturbance rejection - IMEP with step change

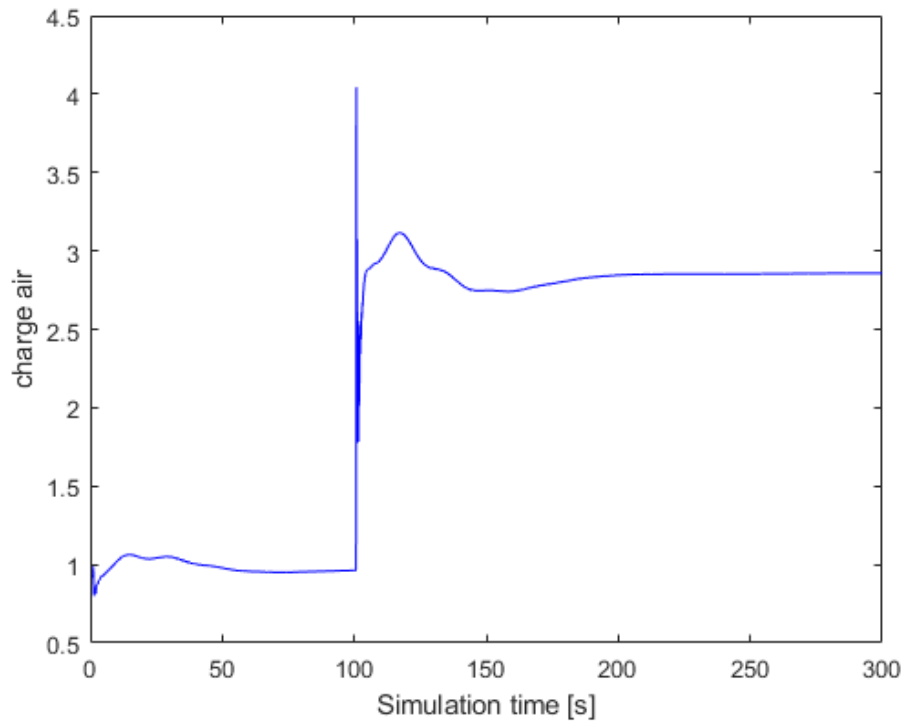


Figure 4-78: Unconstrained set point tracking with disturbance rejection - Charge air control signal

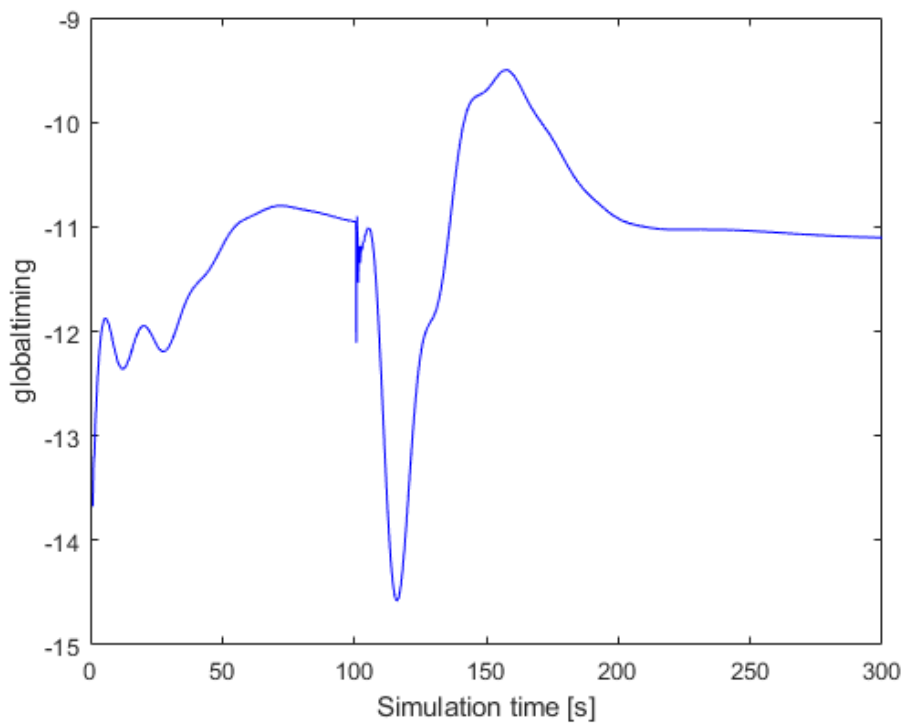


Figure 4-79: Unconstrained set point tracking with disturbance rejection - Global timing control signal

One last test is done on the unconstrained control by adjusting the IMEP gradually from initial value to initial value + 10 bar over the course of 100 seconds.

From Figure 4-80, Figure 4-81, Figure 4-82 and Figure 4-83 we see that during the change in disturbance we note a slight error between the set point and the measured output. After the change in disturbance is done the output settles back to set point.

We note that it is not physical possible to keep heat rate at 10000 while increasing the IMEP to 18 bar and hence the charge air pressure increases to abnormal high values.

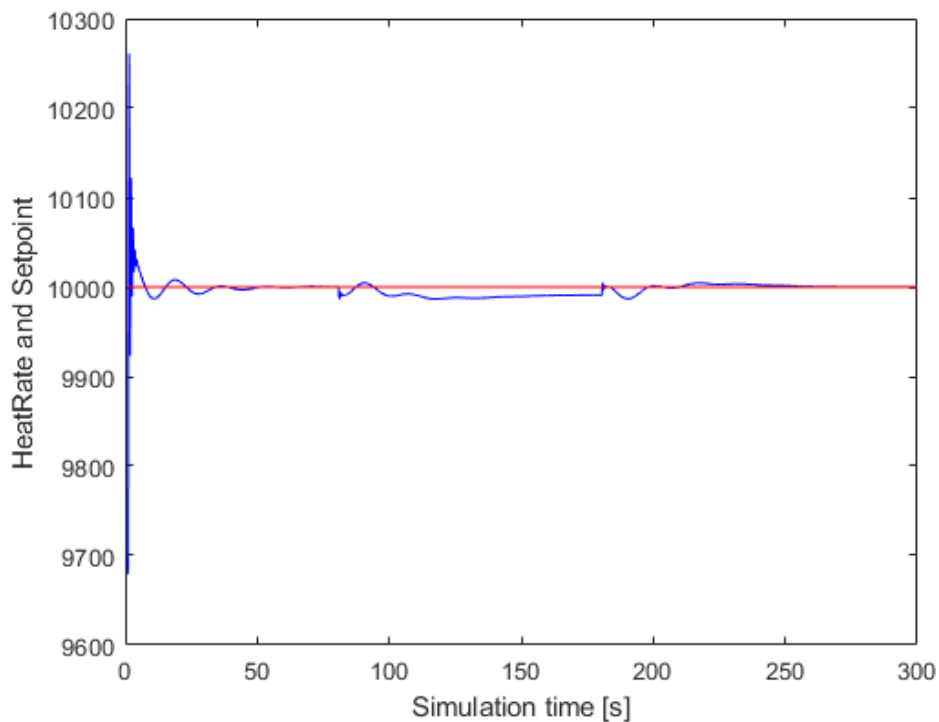


Figure 4-80: Unconstrained set point tracking with disturbance rejection – ramp in disturbance - Heat rate

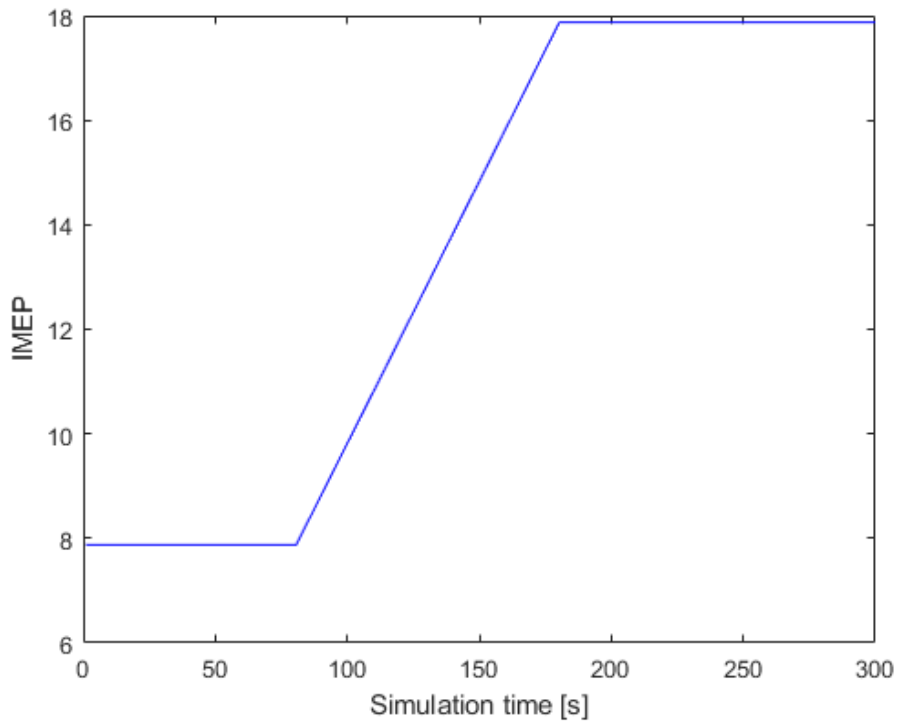


Figure 4-81: Unconstrained set point tracking with disturbance rejection – ramp in disturbance – IMEP disturbance

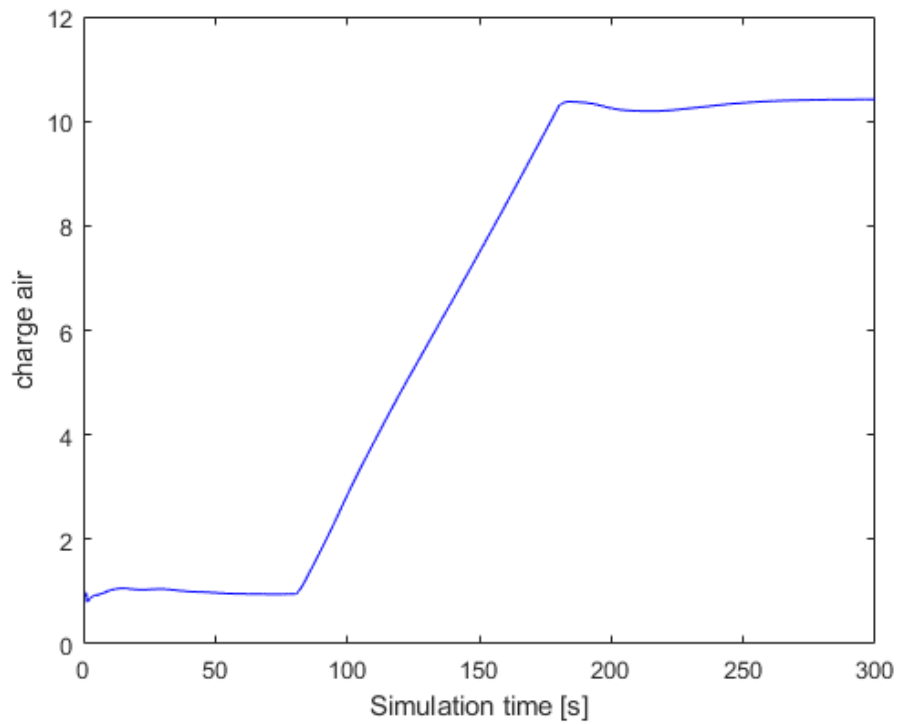


Figure 4-82: Unconstrained set point tracking with disturbance rejection – ramp in disturbance – charge air control output

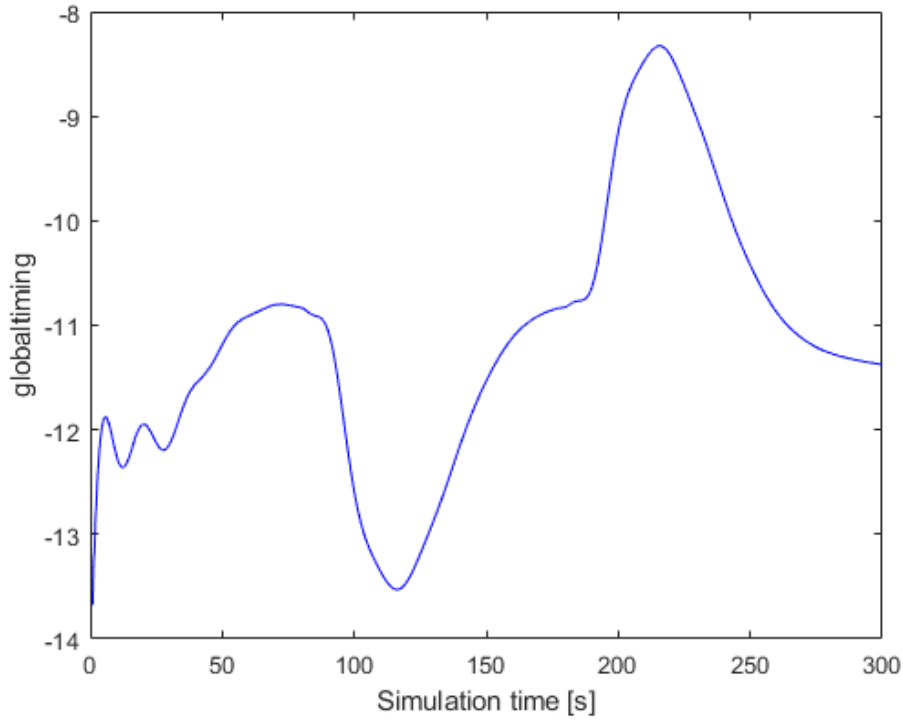


Figure 4-83: Unconstrained set point tracking with disturbance rejection – ramp in disturbance – global timing control output

4.8.2.2 DMPC with constraints on control signal

This section looks into how we can add constraints to the DMPC algorithm. Adding constraints on the control signal difference we define:

$$\Delta u^{min} \leq \Delta u(k_i + m) \leq \Delta u^{max} \quad (4-72)$$

We know from (4-53) that

$$\Delta u(k_i + k) = L(k)^T \eta \quad (4-73)$$

For MIMO system we have Δu are vectors with limits for each input.

We can arrange this in matrix form as

$$\Delta u^{min} \leq \begin{bmatrix} L_1(m)^T & 0_2^T & \dots & 0_m^T \\ 0_1^T & L_2(m)^T & \dots & 0_m^T \\ \vdots & \vdots & \ddots & \vdots \\ 0_1^T & 0_2^T & \dots & L_m(m)^T \end{bmatrix} \eta \leq \Delta u^{max} \quad (4-74)$$

In Appendix D

the MATLAB code for the function which returns matrix M is given.

The inequality constraints can then be formulated as:

$$\begin{aligned} M\eta &\leq \Delta U^{max} \\ -M\eta &\leq -\Delta U^{min} \end{aligned} \quad (4-75)$$

For constraint on amplitude of control signal we have the following conditions:

$$u(k) = \sum_{i=0}^{k-1} \Delta u(i) \quad (4-76)$$

We can then construct this in matrix form

$$u^{min} \leq \begin{bmatrix} \sum_{i=0}^{k-1} L_1(i)^T & 0_2^T & \dots & 0_m^T \\ 0_1^T & \sum_{i=0}^{k-1} L_2(i)^T & \dots & 0_m^T \\ \vdots & \vdots & \ddots & \vdots \\ 0_1^T & 0_2^T & \dots & \sum_{i=0}^{k-1} L_m(i)^T \end{bmatrix} \eta + u(k_i - 1) \leq u^{max} \quad (4-77)$$

The inequality constrain then becomes

$$\begin{aligned} M\eta &\leq U^{max} - u(k_i - 1) \\ -M\eta &\leq -U^{min} + u(k_i - 1) \end{aligned} \quad (4-78)$$

In Appendix E

the MATLAB code for the function which returns matrix M is given.

By adding limits to the control outputs and rate of change of the control signal as:

$$\begin{aligned} 0.3 \text{ barg} &\leq \text{ChargeAirPressure} \leq 4.5 \text{ barg} \\ -20 \text{ degCA} &\leq \text{GlobalTiming} \leq -8.5 \text{ degCA} \\ -0.1 \text{ barg/s} &\leq \Delta \text{ChargeAirPressure} \leq 0.1 \text{ barg/s} \\ -0.3 \text{ degCA/s} &\leq \Delta \text{GlobalTiming} \leq 0.3 \text{ degCA/s} \end{aligned} \quad (4-79)$$

By trial and error the prediction horizon is set to 200 and the constraint horizon to 15.

For the *heat rate* the set point is set artificially low so that it will continuously minimize the error.

We see from Figure 4-84, Figure 4-85, Figure 4-86 and Figure 4-87 that the heat rate is minimized while keeping the control values within the constraints. We can see the lower constraints in red and the upper constraints in yellow. There are some oscillations on the heat rate and charge air pressure control signal until the disturbance starts. It is difficult to control the constraints as these are not imposed as hard constraints on the optimization. They will be broken if in conflict with the objective and hence an outer protection loop will need to be in place in order to guarantee that the control signals do not go out of bounds if the objective is not met. In this case there are now such outer limitations and it can be seen that the constraint on the global timing is broken for some time after the disturbance has reach its maximum. In order to control it the weight on the minimization of the heat rate has been reduced to 0,0001. Since the weight is over a prediction horizon of 200 and with an error in the range around 3500 this still adds up to substantial amount.

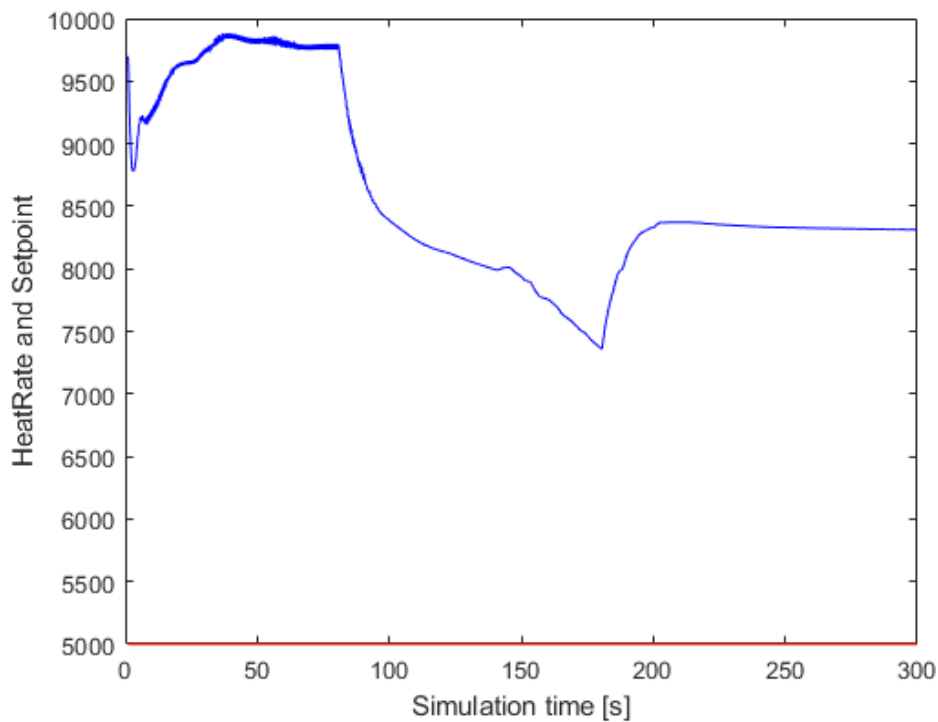


Figure 4-84: Constrained minimization of heat rate with disturbance - Heat rate

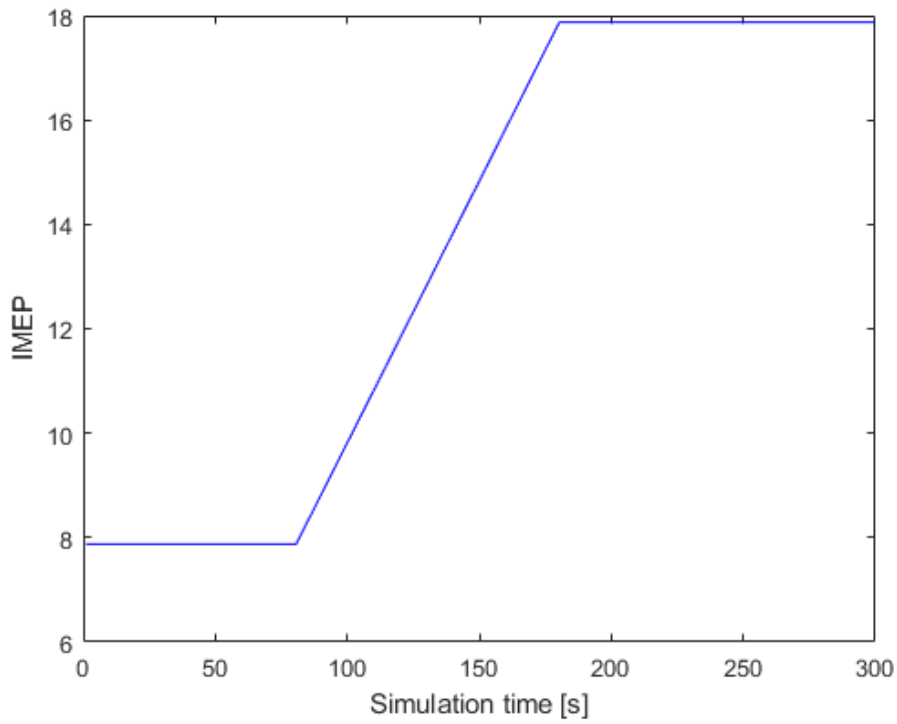


Figure 4-85: Constrained minimization of heat rate with disturbance – IMEP disturbance

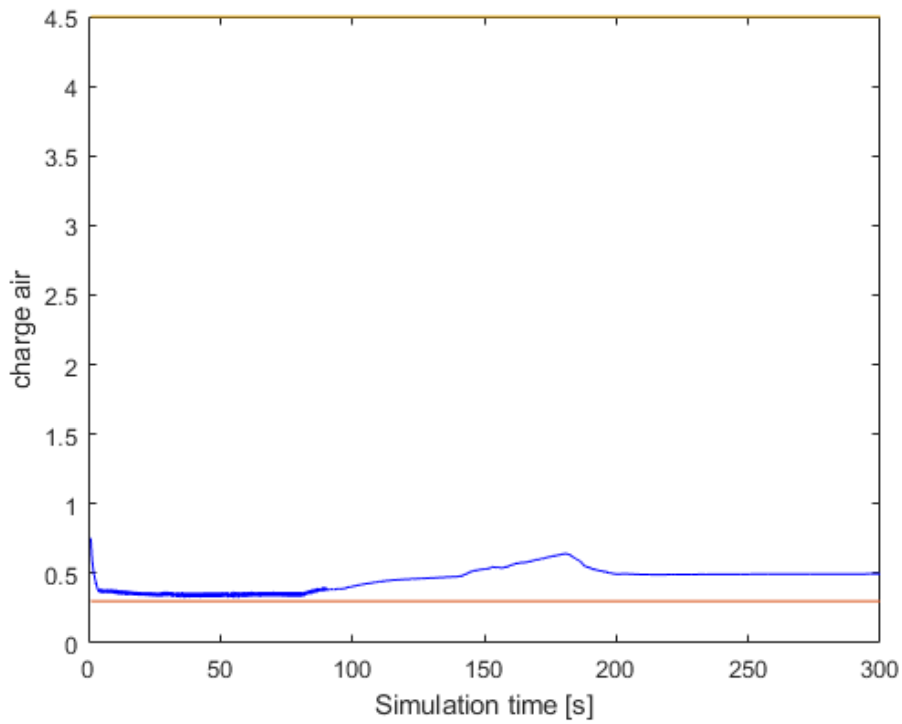


Figure 4-86: Constrained minimization of heat rate with disturbance – control output for charge air pressure

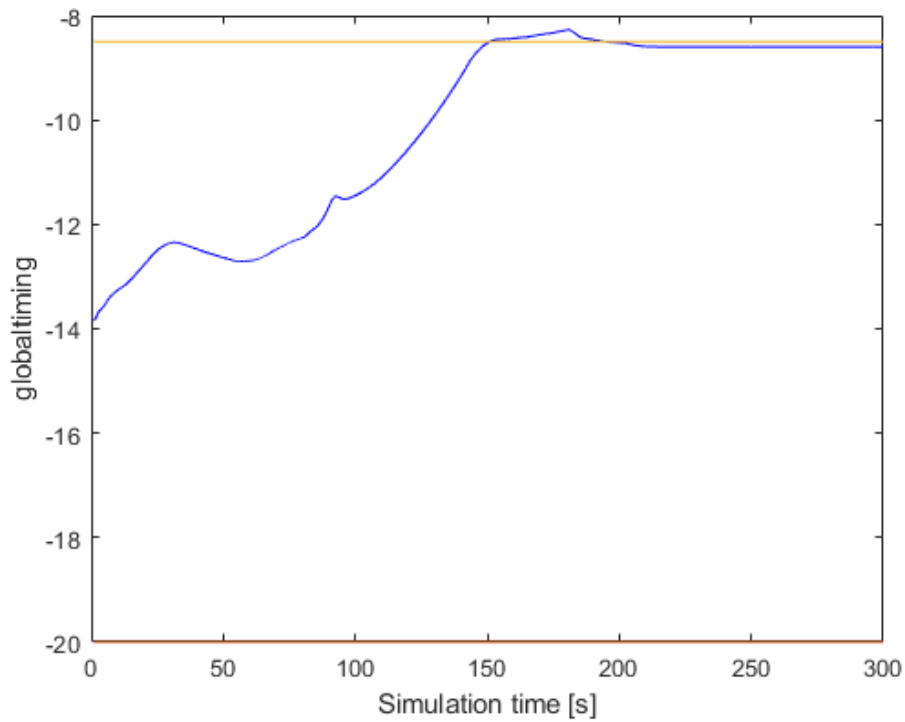


Figure 4-87: Constrained minimization of heat rate with disturbance – control output for global timing

By changing the constraint on global ignition timing to max -9.5 [degCA] we get the result as in Figure 4-88. We see that the constraint is still maintained with the same amount of overshoot initially.

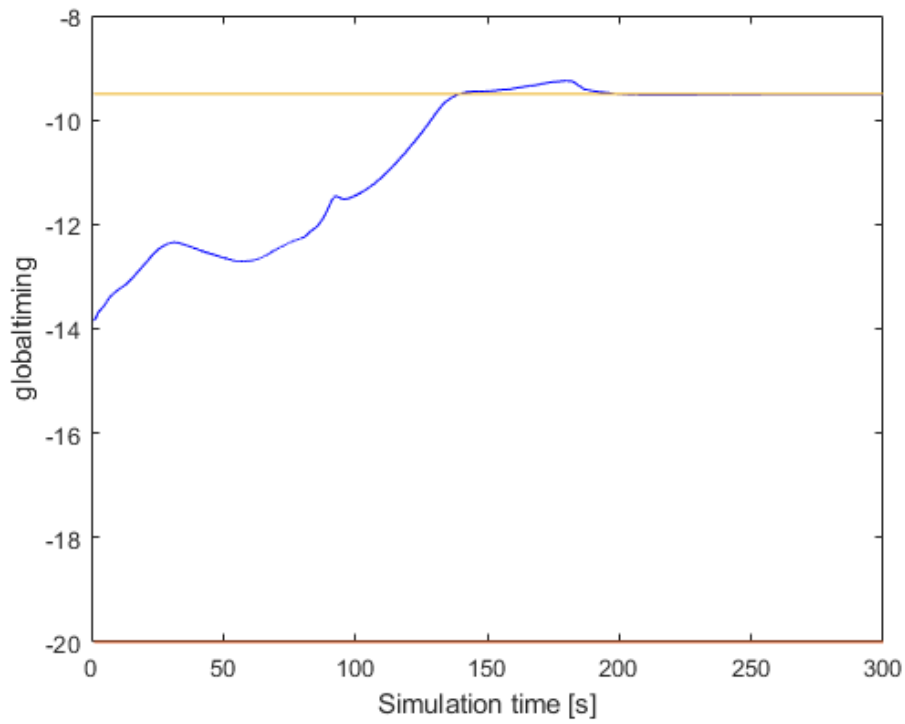


Figure 4-88: Constrained minimization of heat rate with disturbance – control output for global timing with reduced max limit

4.8.2.3 DMPC with constraints on output signals

We can look into how constraints on the outputs might be introduced. The theory here is from the work of Wang [11].

For constraints on the output we define:

$$y^{min} \leq y(k) \leq y^{max} \quad (4-80)$$

We know from (4-54) that the following equation applies

$$\begin{aligned} x(k_i + 1|k_i) &= Ax(k_i) + BL(0)^T \eta \\ y(k_i + 1|k_i) &= CAx(k_i) + CBL(0)^T \eta \end{aligned} \quad (4-81)$$

We know that if $y^{min} \leq y(k_i + 1|k_i) \leq y^{max}$ the optimal solution is where $\eta = -\Omega^{-1}\Psi x(k_i)$

If the lower limit constraint is broken, then we need to find a solution where the cost function is minimized while still satisfy the constraint. Here we defined

$$-CBL(0)^T \eta \leq -y^{min} + CAx(k_i) \quad (4-82)$$

Denoting $M_{act} = -CBL(0)^T$ the Lagrange multiplier λ_{act} is given as

$$\lambda_{act} = -(M_{act}\Omega^{-1}M_{act}^T)^{-1}(-y^{min} + CAx(k_i) + M_{act}\Omega^{-1}\Psi x(k_i)) \quad (4-83)$$

The optimal solution is now given as

$$\eta = -\Omega^{-1}(\Psi x(k_i) + M_{act}^T \lambda_{act}) \quad (4-84)$$

Looking at the upper constraint limit this is almost the same as for the lower limit

$$CBL(0)^T \eta \leq y^{max} - CAx(k_i) \quad (4-85)$$

$$\lambda_{act} = -(M_{act}\Omega^{-1}M_{act}^T)^{-1}(y^{max} - CAx(k_i) + M_{act}\Omega^{-1}\Psi x(k_i)) \quad (4-86)$$

The optimal solution is the same as in (4-84).

Several attempts on getting the output constraint to work in the MIMO system with several outputs imposing constraints on the augmented state space model has been made. It within the available time frame not succeeded in making both the output constraints and the constraints on the control signal to work simultaneously. Additional work in order to modify the logic to work with this particular case would need to be tested.

The DMPC with Laguerre functions has so far worked well and works good for the constraints on the control signal and with substantial less computational effort than the Fmincon from MATLAB.

5 Discussion

The following sections will discuss the project results and the situations which arose during the project as well as suggest some future work. The chapter is divided into separate sections, each dealing with separate parts of the project.

5.1 Modelling

During the development of the MPC controller for this project various aspects has been looked upon. During the initial phase of the development a lot of time was spent using the polynomial model developed in the Masters project during the fall of 2020, but after several failed attempts in getting a reliable model to work, it was decided to swap the model with an new state space model. There are several reasons for the update of model, but it was mostly driven by the lack of results when using the qpOASES and the Quadprog optimizers. Even though the step response of the open loop simulation gave indications that the model had a good representation of the system, which was also backed up by good simulation fit with the validation data set during the system identification. The initial problem with the qpOASES and the Quadprog was also related to problem of infeasibility and challenges in adding constraints on outputs into the structure resulting in a very large optimization problem.

Another aspect was that the polynomial model needed to be transferred into a state space model representation to work with the set up in SIMULINK where the initial benefits of having all parts in the polynomial model with a physical meaning got lost anyway.

The model development based on measured input/outputs from a real engine in the field has some limitations which are potential critical to problems observed throughout the testing. The model has a good representation of the system at which the data has been collected but lacks the dimensions outside the normal operating range as the control signals was not exhausted in either direction as it was running in closed loop control. Updated control software for the engine controller has been developed and prepared for further testing, but due to lack of testing possibilities at the production test bed in Bergen the tests has been postponed for now.

The next goal is therefor to get a better model representation of the system by running the control signals outside the normal operating range.

5.2 qpOASES and Quadprog

The results from the qpOASES and the Quadprog was limited and the possibilities was not explored to their full potential as the decision within the project to focus on Fmincon was taken after a few weeks. These potentials could have shown that there where underlying problems in the implementation which was not discovered during the code walkthrough and the study of the results. However, some testing was still performed, mainly related to set point tracking. The results are briefly presented. The simulation time however was clearly a limiting factor. Even short simulation times required several hours of running on the development computer.

The results from Quadprog was the least promising as these as pr usual run in to feasibility issues at an early stage.

5.3 Fmincon

The focus has largely been on the Fmincon during most of the time in this project. The pre-knowledge of Fmincon was limited to references to it in the lecture notes of IIA 4117 and the examples presented there. The possibilities are large with the use of Fmincon, and a large portion of the time has been spent exploring various settings and numerous of simulations in MATLAB. Some of those simulations results are presented in this paper, but most them are trial and error and failed attempts.

The basic structure of setting up the problem is clearly understood and the setup of the cost function is straight forward as the goal here is to minimize the *Heat rate* will keeping the outputs within boundaries. Several weeks was spent on testing various ways of adding constraints and bounds and look at how the Fmincon handles this.

During the first test all constraints was added as non-linear constraints, which according to the MATLAB forum was not very efficient given that the constraints are not non-linear adding unnecessary computational load. The bounds were also added as non-linear constraints during the first tests. Later the amplitude constraints on the control signal was added as bounds which are treated as hard constraints by the Fmincon. The constraints on the measurements were added as inequality constraints. The constraint on the Δu remained as non-linear constraints during the final testing as these are calculated and updated for each iteration in with the cost function.

The results from the testing shown in section 4.7 shows that several problems related to how the Fmincon handles constraints causing “noisy” measurements if activated, particular when testing set point tracking. This is however not how the MPC problem will be set up as the cost function is based on minimizing the heat rate and not error minimization.

The Fmincon balances the integer number from the cost function versus the output from the constraint function and will weigh the absolute value the highest if in conflict. The output from the cost function must therefore not be of an amplitude higher than the breach of the constraints will produce. The *Heat rate* which has a high absolute value where therefor scaled in the cost function and the weight is reduced compared the Δu weight so that the control signals is carefully adjusted.

One problem that appeared during some of the test was that the optimal, or minimal value of the *Heat rate*, was found (local optimum) while the constraints still where broken. The Fmincon then aborted before trying to adjust the measurements within the constraints resulting in an escalating breach of the constraints. This problem arises during some of the testing but was not easy to reproduce afterwards.

There are still needed more verification on how the Fmincon activates the constraints on the measurements as this seems to be an issue which could potentially cause problems.

Also for the Fmincon there are long simulation times, and more effort needs to be added look into the different methods of reducing the computational load. Here grouping was briefly tested but caused problems. Using Lagrange multipliers could be used or QR factorization.

5.4 DMPC with Laguerre functions

During the theory study for the project one of the theories that seem to be well documented was the DMPC with Laguerre function as explained by Wang [11]. The theory and approach were appealing, and an attempt was made to apply the concepts and theory of this approach to the problem at hand.

A lot of time was spent trying to get the implementation to work with the state space model identified for this problem with measured disturbances. The unconstrained problem worked well and as a set point tracking it produced results which were promising.

The implantation of constraint handling on the control output worked well but when adding constraint to the measurements some the results was not as expected. Numerous attempts with various setups with how to handle the constraints in combination with the control signal constraints was tested without success. The use of augmented state space model was primarily used but also variants without the augmented version was tested without getting around the MIMO case with multiple constraints on the outputs.

The structure and theory are promising and with some more work it might work well, and for a simpler set point tracking without constraints on the measured outputs it would most likely work very well. It might be a good test to using on the cooling water control which currently is using a standard PID controller with set point biasing to handle measured disturbances and gain scheduling.

5.5 Future work

Based on the findings so far there is still a lot of work that needs to be done to find a working version of the MPC. It is suggested to look into other industrial implementations of the MPC controller other than the Fmincon and possibly the CasADi might be a way forward. This already proved well in various industrial applications [16] for MIMO systems and using multiple shooting.

The testing on lab engines do get more dynamic data is however first on the list of next steps to be performed. This is expected to be done in the near future and will give rise to a better model which can be tested on the implantation already at hand.

The development of alternative fuels for the Bergen B36:45 is already ongoing where MPC is expected to play a vital role when implementing Hydrogen mixing into the LNG during operation such as to optimize the mixing process.

A step in between might be to test a simpler structure as for example to replace our current cooling water temperature PID controller with an MPC controller. The PID controller used today could be optimized to some extent by adding future control moves based on already known measure disturbances. Today this is a simple feedback loop which do not utilize this information and has a poor set point scheduling algorithm to handle known disturbances. This could be a good proof of concept for an easier set point tracking function implementation in our current controller where the computational effort is far less as there will not be any constraints on several measured outputs. The implementation in the SIMULINK code already utilized by Bergen is still an open topic and hence this test could help in discovering any limitations in the software structure as well as the resources on the hardware platform.

6 Conclusion

The project has proven to be more challenging than expected when it first was defined in cooperation with Bergen Engines. The model on which the MPC is based does not contain all the dynamics one would have liked and the extended testing that was planned but not done due to the delays in the production line has affected the work to some degree. The lack of access to the factory and the resources during the ongoing pandemic has limited the progress to some extent but the access to data has still be good.

During the earlier phases of the project the contact person at Bergen Engines, who defined the problem in the first place, also left the company. The results of this was the loss of a highly competent person to discuss the results and step further with. The discussions during the project has therefor relied on the weekly meetings with the supervisor. Without those this project would have become even more challenging and progress would have been far less.

Some results are found and numerous of hours has been spent in testing and simulations. This is a time-consuming task and has at times caused a lot frustration when it fails after several hours of testing.

The project was first thought to be completed with an implementation in Simulink the environment of which the engine controller resides and finally with testing on a real engine. The current status is that we are not there yet and substantial more exploration of alternative implementations are needed as discussed in section 5.5.

The learning curve has been steep due limited prior knowledge in the field of model predictive control. Even though the field of MPC is not new it is still very much still being developed as more powerful computers and controllers can handle more complex and large optimization problems. The optimization theory subject is large, and one is not expected to learn it all during such a project. Future study is required before the theory can be put into practise. The CasADi framework looks promising will be studied further so see if there is a fit with the future projects at Bergen Engines.

Based on the testing done so far it is the Fmincon which has the most promising results as it is able to minimize the heat rate with a wide range of *IMEP* input values, ranging from low values of 4-5 bar and up to full power output at around 21-22 bar. Testing on the qpOASES and Quadprog is not likely to be continued in the near future, while the DMPC with Laguerre functions is definitely on the list for testing on the cooling water set point tracking MPC.

It is clear that the MPC currently being looked at are a computational demanding task and with the Fmincon implementation there are still open issues regarding the feasibility of being implemented on the current hardware structure.

References

- [1] Bergen Engines AS, *Master presentation*, Bergen Engines AS, 2020.
- [2] N. R. Ruchika, “Model Predictive Control: History and Development,” *International Journal of Engineering Trends and Technology (IJETT) - Volume 4 Issue 6*, pp. 2600-01, 2013.
- [3] C. M. A. & H. J. Cutler, “An industrial perspective on advanced control,” in *AICHE annual meeting*, Washington DC, 1983.
- [4] C. M. P. T. D.W. Clarke, “Generalized predictive control—Part I. The basic algorithm,” *Automatica Volume 23, Issue 2*, pp. 137-148, 1987.
- [5] Wikipedia contributors, “Model predictive control,” 09 04 2021. [Online]. Available: https://en.wikipedia.org/wiki/Model_predictive_control.
- [6] R. Sharma, *Lecture notes for the course IIA 4117: Model Predictive Control*, Porsgrunn: University of South-Eastern Norway, 2019.
- [7] D. D. Ruscio, *Model predictive control and optimization, Lecture notes Model Predictive Control*, Porsgrunn: University of South-Eastern Norway, 2019.
- [8] N. Shah, “Simulation of model predictive control using dynamic matrix control algorithm,” ProQuest LLC, Ann Arbor, 2015.
- [9] P. H. Johan Åkesson, “Integral action - A disturbance observer approach,” in *European Control Conference*, Cambridge, 2003.
- [10] G. P. James B. Rawlings, “Disturbance Models for Offset-Free Model-Predictive Control,” *AICHE Journal*, vol. 49, no. 2, pp. 426-437, 2003.
- [11] L. Wang, *Model Predictive System Design and Implementation Using MATLAB*, Melbourne: Springer, 2009.
- [12] J. Richalet, “Industrial applications of model based predictive control,” *Automatica*, vol.29, no. 5, pp. 1251-1274, 1993.
- [13] MathWorks, “Automated driving using model predictive control,” 10 04 2021. [Online]. Available: <https://www.mathworks.com/help/mpc/ug/automated-driving-using-model-predictive-control.html>.
- [14] H. J. Ferreau, A. Potschka and C. Kirches, “qpOASES User's Manual,” April 2007-2017. [Online]. Available: <https://github.com/coin-or/qpOASES>. [Accessed 12 05 2021].
- [15] Wikipedia contributors, “Lagrange multiplier,” 17 04 2021. [Online]. Available: https://en.wikipedia.org/wiki/Lagrange_multiplier.

- [16] J. A. Andersson, J. Gillis, G. Horn, B. J. Rawlings and M. Diehl, “CasADi - A software framework for nonlinear optimization,” *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1-36, 2019.
- [17] DieselNet, “Emission Standards,” 30 03 2021. [Online]. Available: <https://dieselnet.com/standards/de/taluft.php>.

Appendices

Appendix A – Master thesis task description

Appendix B – DMPC.mlx code

Appendix C – lagd.mlx code

Appendix D – Mdu.mlx code

Appendix E – Mu.mlx code

Appendix F – QP_formulation_SPTracking2.mlx

Appendix G – SetPointTracking_SS1_Quadprog_1.mlx

Appendix H – OpenLoopTesting.mlx

Appendix I – Minimise_HeatRate_SS1_OutputConstraint.mlx

Appendix A



Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Advanced model based control of B36:45 LNG engines based on data driven models using machine learning tools

USN supervisor: Associate Professor Roshan Sharma (USN)

External partner: Bergen Engines AS

Task background:

The B36:45 LNG is a marine engine designed by Rolls-Royce with a philosophy of a common platform for all fuel options and applications, and can be configured to meet specific engine requirements of customers. The power output is at 600 kW per cylinder at 750 rpm.

For this engine, current control philosophy for air pressure control is based on static 3d pre-set map with biases from various process values, most dominant by NO_x measurement, as set point to a single PID controller with measure air pressure as feedback.

Global ignition timing is based on a pre-set 3d map fixed set point with biases from measured burn duration, peak pressure measurement and ignition knocking.

It is a time consuming task to find and maintain traditional control structure with many set points and dimensions. The traditional control structures provide non-optimal solutions and static set point and thus are not ideal to handle constantly changing running conditions.

In this project, the focus is on the development of an optimal controller to control the compression system, or else improve the already existing controller using optimal control structures based on data driven models with state of the art tools in machine learning tools.

Aim:

It is of interest to study the possibilities of replacing the traditional control structure with more advanced model based controller such as the model predictive controllers (MPC). In order to develop a model based controller, the first aim is to develop a suitable dynamic model that can capture the necessary/major dynamics of the system. The model should be suitable for designing control systems for charge air pressure and global ignition timing control on B36:45 LNG engines.

For a B36:45 LNG engine, mechanistic models using physical laws will be too complex to develop, tune/calibrate and to update as operational conditions change. The task for this thesis is to develop data driven model(s) using available data sets. The model should be relatively easier to update in order to adapt to changing running conditions, ageing and errors. It should also be possible to incorporate safety functions and fallback solutions in case of sensor and controller failures.

The second aim of this thesis is to make use of this data driven model and develop an advanced model based controller to control the operation of the engine for increased efficiency and performance.

Task description:

The following are the main tasks:

- (i) Literature review on:
 - a. Dynamic models of LNG engines suitable for control purposes
 - b. Classical and/or advanced control of LNG engines
- (ii) Use the collected data sets to develop a data driven model that can capture the major dynamics of the system. The data driven model should be used as a "prediction model" for designing MPC. In particular, the model should be able to perform multi-time steps predictions. The focus should be on the use of machine learning algorithms (neural networks, deep learning or other system identification methods including statistical methods). The model should be relatively easier to adapt/calibrate to changing operational conditions.
- (iii) Model verification using real field data should be performed to ensure its correctness.
- (iv) Develop an advanced model based controller (preferably MPC) based on the data driven model of (ii) for optimal operation of the B36:45 LNG engine for increased efficiency and performance.
- (v) If possible (and if time permits), test the controller on a real plant or on a high fidelity simulator of the B36:45 LNG engines.
- (vi) Document the work in a report. The report should be technically sound. Presentation of the work.

Student category: This thesis is reserved for online IIA student Svein Roar Kvåle, and thus is not open for other students.

The task is suitable for online students (not present at the campus): Yes


Practical arrangements:

This project will be performed in close cooperation with Bergen Engines AS. Data sets will be made available by Bergen Engines AS. During verification, various tests may have to be performed at test facility of Bergen Engines AS.

Supervision:

As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature): 01.02.2021 

Student (write clearly in all capitalized letters): SVEIN ROAR KVÅLE

Student (date and signature): Svein Roar Kvåle 25/01-2021

Appendix B

DMPC.mlx code

This function is based on the code given by Liuping Wang in the book Model Predictive control system design and implementation using MATLAB.

Svein Roar Kvåle, USN, 2021

This function will calculate the Omega and Psi variables given the state matrix, input matrix and the Laguerre functions. This function will handle the MIMO case

The inputs to the function are:

- A_e – State matrix – extended if augmented state space is used
- B_e – Input matrix – extended if augmented state space is used
- a – vector of Laguerre pole location for each input
- N – The number of terms in the Laguerre function for each input
- N_p – Prediction horizon
- Q – Weighting matrix for the states
- R – Weighting matrix for the inputs

The outputs are:

- $\Omega = \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right)$
- $\Psi = \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right)$

```
function [Omega,Psi] = dmPC(A_e,B_e,a,N,Np,Q,R)
```

First we get in the number of inputs control signals based on the size of the B matrix

```
[n,n_in] = size(B_e);
```

Then we find the dimension of η and set up the sizes of the Ω , Ψ and R_L

```
N_pa = sum(N);  
Omega = zeros(N_pa,N_pa);  
Psi = zeros(N_pa,n);  
R_para = zeros(N_pa,N_pa);
```

Now take the number of terms for each input in N and put the weights from the R input matrix on the diagonal for each term

```

n0=1;
ne=N(1);

for i=1:n_in-1
    R_para(n0:ne,n0:ne)=R(i,i)*eye(N(i),N(i));
    n0 = n0+N(i);
    ne=ne+N(i+1);
end

R_para(n0:N_pa,n0:N_pa) = R(n_in,n_in)*eye(N(n_in),N(n_in));

```

Initialise the convolution sum and calculate the case for $i = 1$

$$S_c(m) = AS_c(m-1) + S_c(1)(A_l^{m-1})^T$$

With $S_c(1) = BL(0)^T$

Here the *lagd.m* function is used to calculate the initial condition $L(0)$ and the state space matrix A_l

$$A_l = \begin{bmatrix} a & 0 & 0 & 0 \\ \beta & a & 0 & 0 \\ -a\beta & \beta & a & 0 \\ a^2\beta & -a\beta & \beta & a \end{bmatrix} \text{ and } L(0) = \sqrt{\beta} \begin{bmatrix} 1 \\ -a \\ a^2 \\ -a^3 \end{bmatrix}$$

```

S_in = zeros(n,N_pa);
[Al,L0] = lagd(a(1),N(1));
S_in(:,1:N(1)) = B_e(:,1)*L0';

In_s = 1;
for jj=2:n_in
    [Al,L0] = lagd(a(jj),N(jj));
    In_s = N(jj-1)+In_s;
    In_e = In_s+N(jj)-1;
    S_in(:,In_s:In_e) = B_e(:,jj)*L0';
end

S_sum = S_in;
phi = S_in;
Omega = (phi)'*Q*(phi);
Psi = phi'*Q*A_e;

```

In the section below we iterate through the prediction horizon N_p for each of the inputs

For each iteration of the prediction horizon we calculate the A_l and $L(0)$

$$S_c(m) = AS_c(m-1) + S_c(1)(A_l^{m-1})^T$$

```

for i=2:Np
    Eae=A_e^i;

    [Al,L0] = lagd(a(1),N(1));
    S_sum(:,1:N(1)) = A_e*S_sum(:,1:N(1))+S_in(:,1:N(1))*(Al^(i-1))';
    In_s = 1;
    for kk=2:n_in
        [Al,L0] = lagd(a(kk),N(kk));
        In_s = N(kk-1)+In_s;
        In_e = In_s+N(kk)-1;
        S_sum(:,In_s:In_e) = A_e*S_sum(:,In_s:In_e) + ...
            S_in(:,In_s:In_e)*(Al^(i-1))';
    end
    phi = S_sum;
    Omega=Omega+phi'*Q*phi;
    Psi=Psi+phi'*Q*Eae;
end
Omega = Omega+R_para;
end

```

Appendix C

lagd.mlx

This function will return the initial condition of the Laguerre function $L(0)$ and the state space system matrix A_l

$$A_l = \begin{bmatrix} a & 0 & 0 & 0 \\ \beta & a & 0 & 0 \\ -a\beta & \beta & a & 0 \\ a^2\beta & -a\beta & \beta & a \end{bmatrix}, L(0) = \sqrt{\beta} \begin{bmatrix} 1 \\ -a \\ a^2 \\ -a^3 \end{bmatrix} \text{ where } \beta = (1 - a^2)$$

```
function [A,L0] = lagd(a,N)
v(1,1) = a;
L0(1,1) = 1;

for k = 2:N
    v(k,1) = (-a).^(k-2)*(1-a*a);
    L0(k,1) = (-a).^(k-1);
end
L0=sqrt((1-a*a))*L0;
A(:,1)=v;

for i = 2:N
    A(:,i) = [zeros(i-1,1);v(1:N-i+1,1)];
end

end
```


Appendix D

Mdu.mlx

This function will limit the difference constraint of the control signals

$$\Delta u^{\min} \leq \Delta u(k_i + m) \leq \Delta u^{\max}$$

$$\Delta u(k_i + m) = L(m)^T \eta$$

This function generates the matrix M.

$$\Delta u^{\min} \leq \begin{bmatrix} \sum_{i=0}^{k-1} L_1(m)^T & 0_2^T & \dots & 0_m^T \\ 0_1^T & \sum_{i=0}^{k-1} L_2(m)^T & \dots & 0_m^T \\ \vdots & \vdots & \vdots & \vdots \\ 0_1^T & 0_2^T & \dots & \sum_{i=0}^{k-1} L_m(m)^T \end{bmatrix} \eta \leq \Delta u^{\max}$$

$m = 0, 1, \dots$ denotes future time instants to which the constraint is imposed on

$$M\eta \leq \Delta U^{\max}$$

$$-M\eta \leq -\Delta U^{\min}$$

We remember that

$$A_l = \begin{bmatrix} a & 0 & 0 & 0 \\ \beta & a & 0 & 0 \\ -a\beta & \beta & a & 0 \\ a^2\beta & -a\beta & \beta & a \end{bmatrix}, L(0) = \sqrt{\beta} \begin{bmatrix} 1 \\ -a \\ a^2 \\ -a^3 \end{bmatrix} \text{ where } \beta = (1 - a^2)$$

and that

$$L(k+1) = A_l L(k)$$

```
function [M,Lzerot] = Mdu(a,N,n_in,Nc)

%Lzerot is used for creating the control signal

N_pa=sum(N);
M=zeros(n_in,N_pa);

M_du1=zeros(n_in,N_pa);
```

```

k0=1;
[A1,L0] = lagd(a(k0),N(k0));

M_du1(1,1:N(1))=L0';
cc=N(1);

for k0=2:n_in
    [A1,L0] = lagd(a(k0),N(k0));
    M_du1(k0,cc+1:cc+N(k0))=L0';
    cc=cc+N(k0);
end
Lzerot=M_du1;
M=M_du1;

for kk=2:Nc
    k0=1;
    [A1,L0] = lagd(a(k0),N(k0));
    L=A1^(kk-1)*L0;
    M_du1(1,1:N(1))=L';
    cc=N(1);
    for k0=2:n_in
        [A1,L0] = lagd(a(k0),N(k0));
        L=A1^(kk-1)*L0;
        M_du1(k0,cc+1:cc+N(k0))=L';
        cc=cc+N(k0);
    end
    M=[M;M_du1];
end

end

```

Appendix E

Mu.mlx

This function will limit the amplitude constraint of the control signals

This function generates the matrix M

$$u^{\min} \leq \begin{bmatrix} \sum_{i=0}^{k-1} L_1(i)^T & 0_2^T & \dots & 0_m^T \\ 0_1^T & \sum_{i=0}^{k-1} L_2(i)^T & \dots & 0_m^T \\ \vdots & \vdots & \vdots & \vdots \\ 0_1^T & 0_2^T & \dots & \sum_{i=0}^{k-1} L_m(i)^T \end{bmatrix} \eta + u(k-1) \leq u^{\max}$$

$$\begin{aligned} M\eta &\leq U^{\max} - \bar{u}(k_i - 1) \\ -M\eta &\leq -U^{\min} + \bar{u}(k_i - 1) \end{aligned} \quad \bar{u}(k_i - 1) \text{ is a vector of past } u \text{ values}$$

```
function M = Mu(a,N,n_in,Nc)

%a = Languerre scaling factor
%N = Number of Languerre coeficients
%n_in = Number of inputs
%Nc = Number of future samples to impose4 constraints on

N_pa=sum(N);
M=zeros(n_in,N_pa);

M_du1=zeros(n_in,N_pa);
k0=1;
[A1,L0]=lagd(a(k0),N(k0));
M_du1(1,1:N(1))=L0';

cc=N(1);

for k0=2:n_in
    [A1,L0] = lagd(a(k0),N(k0));
    M_du1(k0,cc+1:cc+N(k0))=L0';
    cc=cc+N(k0);
end
```

```

M = M_du1;
Ms = M_du1;

for kk=2:Nc
    k0=1;
    [A1,L0] = lagd(a(k0),N(k0));
    L = A1^(kk-1)*L0;
    M_du1(1,1:N(1))=L';
    cc=N(1);

    for k0=2:n_in
        [A1,L0] = lagd(a(k0),N(k0));
        L = A1^(kk-1)*L0;
        M_du1(k0,cc+1:cc+N(k0))=L';
        cc=cc+N(k0);
    end
    Ms = Ms+M_du1;
    M=[M;Ms];
end
end

```

Appendix F

QP_formulation_SPTracking2.mlx

This code presented below gives the qpOASES QP formulation for set point tracking.

```
function [H,c, Ae,zL,zU,be] = QP_formulation_SPTracking2(dx0, ud, dr,
u_prev, A, B, Bd, C1)

%=====
%Svein Roar Kvåle
%USN - spring 2021
%Set point tracking -> using qpOASES

N=20;      %Prediction horizon
nx = 25;   %number of states
nu = 2;    %number of controlable outputs
nd = 3;    %number of known disturbances
ny = 1;    %feedback signal for setpoint tracking

%number of decision variables
nz = N * (nu+nx+ny+ny+nu);

dr = (ones(1,N).*dr(1)); %Keep set point fixed throught the prediction
horizon

Q=diag(0.8); %error weighting matrix for set point tracking
P=diag([0.01, 0.01]); %Weighting matrix for control signals
S=diag([0.1, 0.1]);%Weighting matrix for delta u

H11 = kron(eye(N),P); %Inputs
H22 = zeros(N*nx,N*nx); %states
H33 = kron(eye(N),Q); %error
H44 = zeros(N*ny,N*ny); %Output
H55 = kron(eye(N),S); %Delta u

H_mat = blkdiag(H11,H22,H33,H44,H55);
H = H_mat(:);

c = zeros(nz,1);

%=====
%Equality constarintns

%Xk+1 = A*Xk + B*Uk + Bd*Udk
Ae1u = -kron(eye(N),B);
Ae1x = eye(N*nx)-kron(diag(ones(N-abs(-1),1),-1),A);
```

```

Ae1e = zeros(N*nx,N*ny);
Ae1y = zeros(N*nx,N*ny);
Ae1du = zeros(N*nx,N*nu);
be1 = [A*dx0 + Bd*ud; kron(ones(N-1,1),Bd*ud)];

%Y measured signal used for set point tracking
%Yk = C1*Xk
Ae2u = zeros(N*ny,N*nu);
Ae2x = -kron(eye(N),C1);
Ae2e = zeros(N*ny, N*ny);
Ae2y = eye(N*ny,N*ny);
Ae2du = zeros(N*ny,N*nu);
be2 = zeros(N*ny,1);

%Error signal used fot set point tracking
%Ek = Rk-Yk
Ae4u = zeros(N*ny, N*nu);
Ae4x = zeros(N*ny,N*nx);
Ae4e = eye(N*ny);
Ae4y = eye(N*ny,N*ny);
Ae4du = zeros(N*ny,N*nu);
be4 = reshape(dr,N*ny,1);

%Delta u on control signals -
%Used to limit rate of change of control outputs
%dUk = Uk-Uk-1
Ae5u = -eye(N*nu)+kron(diag(ones(N-abs(-1),1),-1),eye(nu));
Ae5x = zeros(N*nu,N*nx);
Ae5e = zeros(N*nu,N*ny);
Ae5y = zeros(N*nu,N*ny);
Ae5du = eye(N*nu);
be5 = [-u_prev;zeros((N-1)*nu,1)];

Ae_mat = [Ae1u, Ae1x, Ae1e, Ae1y, Ae1du;...
          Ae2u, Ae2x, Ae2e, Ae2y, Ae2du;...
          Ae4u, Ae4x, Ae4e, Ae4y, Ae4du;...
          Ae5u, Ae5x, Ae5e, Ae5y, Ae5du];
Ae = Ae_mat(:);
be = [be1;be2;be4;be5];

%=====
%Bounds

%Charge air pressure = 0-4.5barg
%Global timing = -20 - -8.5degCA
uMin = [ones(1,N).*0;...           %10CA control output minimum value
        ones(1,N).*-20];         %Global timing minimum value (degCA
before TDC)

```

```

%Delta charge air pressure = -0,1 - 0,1 barg/s
%Delta global timing = -0,1 - 0,1 degCA/s
deltaUMin = [ones(1,N).*-0.02;... %10CA set point minimum rate of
change
            ones(1,N).*-0.03]; %Global timing minimum rate of
change

zL = [reshape(uMin,N*nu,1);... % [u]
      -Inf*ones(N*(nx+ny+ny),1);... % [x + e + y]
      reshape(deltaUMin,N*nu,1)]; % [u]

%Charge air pressure = 0-4.5barg
%Global timing = -20 - -8.5degCA
uMax = [ones(1,N).*4.5;... %10Ca control output max value
        ones(1,N).*-8.5]; %Global timing maximum value

%Delta charge air pressure = -0,1 - 0,1 barg/s
%Delta global timing = -0,1 - 0,1 degCA/s
deltaUMax = [ones(1,N).*0.02;... %10CA set point maximum rate of change
            ones(1,N).*0.05]; %Global timing maximum rate of change

zU = [reshape(uMax,N*nu,1);...
      Inf*ones(N*(nx+ny+ny),1);...
      reshape(deltaUMax,N*nu,1)];

```

Appendix G

SetPointTracking_SS1_Quadprog_1.mlx

The code presented below is the Quadprog code used during the initial testing phase.

Set point tracking with Quadprog

Svein Roar Kvåle

Master thesis spring 2021 @ USN

Table of Contents

Load in simulation data

```
clear;
load ss1.mat
load TrainingMean20200903Red.mat
```

Set up initial values by reading in data from real life training data

Find the initial states by using the "findstates" function. Then run the model for 30 samples with the real-life data to get updated initial states. Use this initial states further and use the next real-life controllable inputs and measured disturbances as initial values for the testing next

```
u_ini_s = TrainingMean20200903Red.InputData(201,:);

X0 =
findstates(ss1,iddata(TrainingMean20200903Red.OutputData(140:170,1:6),Trainin
gMean20200903Red.InputData(140:170,1:5),TrainingMean20200903Red.Ts));

for jj=1:30
    u_ini_s = TrainingMean20200903Red.InputData(170+jj,:);
    X_n = ss1.A*X0 + ss1.B*u_ini_s;
    X0 = X_n;
end

u1 = TrainingMean20200903Red.InputData(201,1) %SuctionAirTemp
u2 = TrainingMean20200903Red.InputData(201,2) %IMEP
u3 = TrainingMean20200903Red.InputData(201,3) %ChargeAirPressure
u4 = TrainingMean20200903Red.InputData(201,4) %ChargeAirTemperature
u5 = TrainingMean20200903Red.InputData(201,5) %GlobalTiming

%Receive u_previous measurements
u_k_prev(1) = TrainingMean20200903Red.InputData(200,3);
u_k_prev(2) = TrainingMean20200903Red.InputData(200,5);
```



```

%State matrix
A = ss1.A;

%Controllable inputs B matrix
B(:,1) = ss1.B(:,3);
B(:,2) = ss1.B(:,5);

%Measured input disturbance Bd Matrix
Bd(:,1) = ss1.B(:,1);
Bd(:,2) = ss1.B(:,2);
Bd(:,3) = ss1.B(:,4);

C1 = ss1.C(1,:);
C2 = ss1.C(2:end,:);
D = ss1.D;

Sim = 500;

```

Set up system for simulation

This part sets up the constants prior to running the actual simulation

```

N=100;      %Prediction horizon
[nx,~] = size(A); %number of states
[~,nu] = size(B); %number of controllable outputs
[~,nd] = size(Bd); %number of known disturbances
[ny,~] = size(C1); %feedback signal for setpoint tracking

%number of decision variables
nz = N * (nu+nx+ny+ny+nu);
dr = (ones(1,N).*10000); %Keep set point fixed throught the prediction
horizon

%Set up disturbances
ud = [u1;u2;u4];

Q=diag(0.8); %error weighting matrix for set point tracking
P=diag([0.01, 0.01]); %Weighting matrix for control signals
S=diag([0.1, 0.1]);%Weighting matrix for delta u

H11 = kron(eye(N),P); %Inputs
H22 = zeros(N*nx,N*nx); %states
H33 = kron(eye(N),Q); %error
H44 = zeros(N*ny,N*ny); %Output
H55 = kron(eye(N),S); %Delta u

H = blkdiag(H11,H22,H33,H44,H55);

```

```

%Set up of linear term
c = zeros(nz,1);

%=====
%Bounds

%Charge air pressure = 0-4.5barg
%Global timing = -20 - -8.5degCA
uMin = [ones(1,N).*0; ...           %10CA control output minimum value
        ones(1,N).*-20];          %Global timing minimum value (degCA
before TDC)

%Delta charge air pressure = -0,1 - 0,1 barg/s
%Delta global timing = -0,1 - 0,1 degCA/s
deltaUMin = [ones(1,N).*-10.02;... %10CA set point minimum rate of
change
             ones(1,N).*-10.03];   %Global timing minimum rate of
change

zL = [reshape(uMin,N*nu,1);...      %[u]
      -Inf*ones(N*(nx+ny+ny),1);... %[x + e + y]
      reshape(deltaUMin,N*nu,1)];  %[u]

%Charge air pressure = 0-4.5barg
%Global timing = -20 - -8.5degCA
uMax = [ones(1,N).*4.5;...         %10Ca control output max value
        ones(1,N).*-8.5];         %Global timing maximum value

%Delta charge air pressure = -0,1 - 0,1 barg/s
%Delta global timing = -0,1 - 0,1 degCA/s
deltaUMax = [ones(1,N).*10.02;... %10CA set point maximum rate of change
             ones(1,N).*10.05];   %Global timing maximum rate of change

zU = [reshape(uMax,N*nu,1);...
      Inf*ones(N*(nx+ny+ny),1);...
      reshape(deltaUMax,N*nu,1)];

```

Simulation

Run the simulation for the defined time and prediction horizon

```

for i=1:Sim
%Stat vector update
dx0 = X0;

```

```

%Transpose the
u_prev = u_k_prev';

%=====
%Equality constraints

%Xk+1 = A*Xk + B*Uk + Bd*Udk
Ae1u = -kron(eye(N),B);
Ae1x = eye(N*nx)-kron(diag(ones(N-abs(-1),1),-1),A);
Ae1e = zeros(N*nx,N*ny);
Ae1y = zeros(N*nx,N*ny);
Ae1du = zeros(N*nx,N*nu);
be1 = [A*dx0 + Bd*ud; kron(ones(N-1,1),Bd*ud)];

%Y measured signal used for set point tracking
%Yk = C1*Xk
Ae2u = zeros(N*ny,N*nu);
Ae2x = -kron(eye(N),C1);
Ae2e = zeros(N*ny, N*ny);
Ae2y = eye(N*ny,N*ny);
Ae2du = zeros(N*ny,N*nu);
be2 = zeros(N*ny,1);

%Error signal used fot set point tracking
%Ek = Rk-Yk
Ae4u = zeros(N*ny, N*nu);
Ae4x = zeros(N*ny,N*nx);
Ae4e = eye(N*ny);
Ae4y = eye(N*ny,N*ny);
Ae4du = zeros(N*ny,N*nu);
be4 = reshape(dr,N*ny,1);

%Delta u on control signals -
%Used to limit rate of change of control outputs
%dUk = Uk-Uk-1
Ae5u = -eye(N*nu)+kron(diag(ones(N-abs(-1),1),-1),eye(nu));
Ae5x = zeros(N*nu,N*nx);
Ae5e = zeros(N*nu,N*ny);
Ae5y = zeros(N*nu,N*ny);
Ae5du = eye(N*nu);
be5 = [-u_prev;zeros((N-1)*nu,1)];

Ae = [Ae1u, Ae1x, Ae1e, Ae1y, Ae1du;...
      Ae2u, Ae2x, Ae2e, Ae2y, Ae2du;...
      Ae4u, Ae4x, Ae4e, Ae4y, Ae4du;...
      Ae5u, Ae5x, Ae5e, Ae5y, Ae5du];

be = [be1;be2;be4;be5];

```

```

f = [];
Aq = [];
b = [];
x0 = dx0;
Aeq = Ae;
beq = be;
lb = zL;
ub = zU;

%Call the optimiser quadprog
x_opt = quadprog(H,f,Aq,b,Aeq,beq,lb,ub,x0);

%Retrieve the two first control moves
du_opt = x_opt(1:2);

%Update the state variable
X0 = A*dx0+B*du_opt+Bd*ud;

%Store the outputs from the plant model
Y(i,:) = ss1.C*dx0;

%Store the control moves
U(i,:) = du_opt';

%Update previous control moves to the latest
u_k_prev = du_opt';

end

```

Plot function

```

fig_1 = figure("Name", 'Control variables');
x = linspace(1,Sim/10,Sim);
y1 = U(:,1);
y2 = U(:,2);
tiledlayout(2,1);

ax1 = nexttile;
plot(ax1,x,y1);
title(ax1, 'Charge air pressure');
ylabel(ax1, 'barg');
xlabel('Simulation time [s]')

ax2 = nexttile;
plot(ax2,x,y2);
title(ax2, 'Global timing');

```

```
ylabel(ax2,"degCA");
xlabel('Simulation time [s]')
```

The figure below shows the model outputs over the simulation time

```
fig_1 = figure("Name", 'Measurements');
y1 = Y(:,1);
y2 = Y(:,2);
y3 = Y(:,3);
y4 = Y(:,4);
y5 = Y(:,5);
y6 = Y(:,6);
y7 = Y(:,1);

t = tiledlayout(3,1, 'TileSpacing', "normal");
title(t, 'Measurements')
ref = dr(1,1)*ones(Sim,1);
ax1 = nexttile;
plot(ax1,x,y1,x,ref);
title(ax1, 'Heat rate');
ylabel(ax1, '-');
xlabel('Simulation time [s]')

ax2 = nexttile;
plot(ax2,x,y2);
title(ax2, 'Knock');
ylabel(ax2, '%');
xlabel('Simulation time [s]')

ax3 = nexttile;
plot(ax3,x,y3);
title(ax3, 'Peak pressure');
ylabel(ax3, 'bar');
xlabel('Simulation time [s]')

fig_1 = figure("Name", 'Measurements');
tt = tiledlayout(3,1, 'TileSpacing', "normal");
ax4 = nexttile;
plot(ax4,x,y4);
title(ax4, 'NOx');
ylabel(ax4, 'ppm');
xlabel('Simulation time [s]')

ax5 = nexttile;
plot(ax5,x,y5);
title(ax5, 'O2');
ylabel(ax5, '%');
xlabel('Simulation time [s]')
```

```
ax6 = nexttile;
plot(ax6,x,y6);
title(ax6,'Exhaust temperature');
ylabel(ax6,'degC');
xlabel('Simulation time [s]')
fig_1 = figure("Name", 'Measurements');
y1 = Y(:,1);
t = tiledlayout(1,1,'TileSpacing', "normal");
title(t, 'Measurements')

ax1 = nexttile;
plot(ax1,x,y1,x,ref);
title(ax1,'Heat rate');
ylabel(ax1,'-');
xlabel('Simulation time [s]')
```

Appendix H

OpenLoopTesting.mlx

Svein Roar Kvåle

Master thesis spring 2021 @ USN

This file contains the testing done on the open loop of the state space model.

Load in simulation data

```
clear;
load ss1.mat
load TrainingMean20200903Red.mat
```

Set up initial values by reading in data from real life training data

Find the initial states by using the "findstates" function. Then run the model for 30 samples with the real life data to get updated initial states. Use this initial states further and use the next real life controllable inputs and measured disturbances as initial values for the testing next

```
u1 = TrainingMean20200903Red.InputData(201,1); %SuctionAirTemp
u2 = TrainingMean20200903Red.InputData(201,2); %IMEP
u3 = TrainingMean20200903Red.InputData(201,3); %ChargeAirPressure
u4 = TrainingMean20200903Red.InputData(201,4); %ChargeAirTemperature
u5 = TrainingMean20200903Red.InputData(201,5); %GlobalTiming

u_ini = [u1;u2;u3;u4;u5];

X0 =
findstates(ss1,iddata(TrainingMean20200903Red.OutputData(140:170,1:6),TrainingMean20200903Red.InputData(140:170,1:5),0.1));

for jj=1:30
    u1 = TrainingMean20200903Red.InputData(170+jj,1); %SuctionAirTemp
    u2 = TrainingMean20200903Red.InputData(170+jj,2); %IMEP
    u3 = TrainingMean20200903Red.InputData(170+jj,3); %ChargeAirPressure
    u4 = TrainingMean20200903Red.InputData(170+jj,4); %ChargeAirTemperature
    u5 = TrainingMean20200903Red.InputData(170+jj,5); %GlobalTiming

    u_ini = [u1;u2;u3;u4;u5];

    X_n = ss1.A*X0 + ss1.B*u_ini;
    X0 = X_n;
end
```

```

u1 = TrainingMean20200903Red.InputData(201,1) %SuctionAirTemp
u2 = TrainingMean20200903Red.InputData(201,2) %IMEP
u3 = TrainingMean20200903Red.InputData(201,3) %ChargeAirPressure
u4 = TrainingMean20200903Red.InputData(201,4) %ChargeAirTemperature
u5 = TrainingMean20200903Red.InputData(201,5) %GlobalTiming

u_ini = [u1;u2;u3;u4;u5];

```

Simulate with fixed values

The following results is found by not changing the controlable values nor the disturbances but keeping them constant based on the initial conditions

```

Sim = 15000;
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;
for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    Pu(i,:) = u(:);
    Py(i,:) = y(:);

    xm = X_next;
end

Plotting(Pu,Py,Sim)

```

Set state values to steady state

Run model such that it is in steady state before changes to controlable inputs

```

Sim = 500;
xm = X0;
u = u_ini;
for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    xm = X_next;
end

X0 = xm;

```


Simulate with positive step change on charge air pressure

The following shows the resultant output of the model based on a step change on the charge air pressure controllable input

```
Sim = 15000;
Step = 0.5
InitialValue = u(3,1)
StepTime = Sim/2
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;
for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    Pu(i,:) = u(:);
    Py(i,:) = y(:);

    xm = X_next;

    if i==(StepTime)
        u(3,1) = u(3,1)+Step;
    end
end

Plotting(Pu,Py,Sim)
```

Simulation with negative step change on charge air pressure

```
Sim = 15000;
Step = 0.5
InitialValue = u(3,1)
StepTime = Sim/2
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;
for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    Pu(i,:) = u(:);
    Py(i,:) = y(:);
```

```

xm = X_next;

if i==(StepTime)
    u(3,1) = u(3,1)-Step;
end

end

Plotting(Pu,Py,Sim)

```

Simulation with negative step change on ignition timing

```

Sim = 15000;
Step = 1.5
InitialValue = u(5,1)
StepTime = Sim/2
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;
for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    Pu(i,:) = u(:);
    Py(i,:) = y(:);

    xm = X_next;

    if i==(StepTime/2)
        u(5,1) = u(5,1)-Step;
    end

end

Plotting(Pu,Py,Sim)

```

Simulation with negative step change on ignition timing

```

Sim = 15000;
Step = 1.5
InitialValue = u(5,1)
StepTime = 2500
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;

```

```

for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    Pu(i,:) = u(:);
    Py(i,:) = y(:);

    xm = X_next;

    if i==(StepTime)
        u(5,1) = u(5,1)+Step;
    end

end

Plotting(Pu,Py,Sim)

```

Simulation with gradually increase on charger air pressure - up 2,5 barg

```

Sim = 15000;
Rate = 0.001;
RatePrSec = 0.001/10 %pr second
InitialValue = u(5,1)
ChangeStartTime = 2500
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;

for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    Pu(i,:) = u(:);
    Py(i,:) = y(:);

    xm = X_next;

    %After stable time - increase with 0,01barg pr 100ms until 2,5 barg
    %increase is reached
    if i>(ChangeStartTime)
        if u(3,1) < (u_ini(3,1) + 2.5)
            u(3,1) = u(3,1)+Rate;
        end
    end

end

```

```
end
```

```
Plotting(Pu,Py,Sim)
```

Simulation with gradually decrease on charger air pressure - down to 0,0 barg

```
Sim = 15000;
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;
for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    Pu(i,:) = u(:);
    Py(i,:) = y(:);

    xm = X_next;

    %After stable time - increase with 0,01barg pr 100ms until 2,5 barg
    %increase is reached
    if i>(2500)
        if u(3,1) > 0
            u(3,1) = u(3,1)-0.001;
        end
    end

end

end

Plotting(Pu,Py,Sim)
```

Simulation with gradually increase on global timing - down to -18degCA

```
Sim = 15000;
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;
for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
```

```

y = ss1.C*xm;

Pu(i,:) = u(:);
Py(i,:) = y(:);

xm = X_next;

%After stable time - increase with 0,01degCA pr 100ms until -18degCA
%increase is reached
if i>(2500)
    if u(5,1) > -18
        u(5,1) = u(5,1)-0.01;
    end
end
end
end

Plotting(Pu,Py,Sim)

```

Simulation with gradually decrease on global timing - up to -8degCA

```

Sim = 15000;
Py = zeros(Sim,6);
Pu = zeros(Sim,5);
xm = X0;
u = u_ini;
for i=1:Sim
    X_next = ss1.A*xm + ss1.B*u;
    y = ss1.C*xm;

    Pu(i,:) = u(:);
    Py(i,:) = y(:);

    xm = X_next;

    %After stable time - increase with 0,01degCA pr 100ms until -8degCA
    %increase is reached
    if i>(2500)
        if u(5,1) < -8
            u(5,1) = u(5,1)+0.01;
        end
    end
end
end

Plotting(Pu,Py,Sim)

```

Plot function

```
function [] = Plotting(Pu,Py,Sim)
fig_1 = figure("Name", 'Control variables');
x = linspace(1,Sim/10,Sim);
y1 = (Pu(:,3));
y2 = (Pu(:,5));
tiledlayout(2,1);

ax1 = nexttile;
plot(ax1,x,y1);
title(ax1, 'Charge air pressure');
ylabel(ax1, 'barg');
xlabel('Simulation time [s]')

ax2 = nexttile;
plot(ax2,x,y2);
title(ax2, 'Global timing');
ylabel(ax2, "degCA");
xlabel('Simulation time [s]')
```

The figure below shows the disturbances over the simulation time

```
fig_2 = figure("Name", 'Disturbances');
x = linspace(1,Sim/10,Sim);
y1 = (Pu(:,1));
y2 = (Pu(:,2));
y3 = (Pu(:,4));
tiledlayout(3,1);

ax1 = nexttile;
plot(ax1,x,y1);
title(ax1, 'Suction air temperature');
ylabel(ax1, 'degC');
xlabel('Simulation time [s]')

ax2 = nexttile;
plot(ax2,x,y2);
title(ax2, 'IMEP');
ylabel(ax2, "bar");
xlabel('Simulation time [s]')

ax3 = nexttile;
plot(ax3,x,y3);
title(ax3, 'Charge air temperature');
ylabel(ax3, "degC");
```

```
xlabel('Simulation time [s]')
```

The figure below shows the model outputs over the simulation time

```
fig_1 = figure("Name", 'Measurements');
x = linspace(1, Sim/10, Sim);
y1 = (Py(:,1));
y2 = (Py(:,2));
y3 = (Py(:,3));
y4 = (Py(:,4));
y5 = (Py(:,5));
y6 = (Py(:,6));
t = tiledlayout(3,1, 'TileSpacing', "normal");
title(t, 'Measurements')

ax1 = nexttile;
plot(ax1,x,y1);
title(ax1, 'Heat rate');
ylabel(ax1, '-');
xlabel('Simulation time [s]')

ax2 = nexttile;
plot(ax2,x,y2);
title(ax2, 'Knock');
ylabel(ax2, '%');
xlabel('Simulation time [s]')

ax3 = nexttile;
plot(ax3,x,y3);
title(ax3, 'Peak pressure');
ylabel(ax3, 'bar');
xlabel('Simulation time [s]')

fig_1 = figure("Name", 'Measurements');
tt = tiledlayout(3,1, 'TileSpacing', "normal");
ax4 = nexttile;
plot(ax4,x,y4);
title(ax4, 'NOx');
ylabel(ax4, 'ppm');
xlabel('Simulation time [s]')

ax5 = nexttile;
plot(ax5,x,y5);
title(ax5, 'O2');
ylabel(ax5, '%');
xlabel('Simulation time [s]')

ax6 = nexttile;
```

```
plot(ax6,x,y6);  
title(ax6,'Exhaust temperature');  
ylabel(ax6,'degC');  
xlabel('Simulation time [s]')  
end
```


Appendix I

Minimise_HeatRate_SS1_OutputConstraint.mlx

Svein Roar Kvåle

Master thesis spring 2021 @ USN

Software structure is based on the lecture notes in IIA 4117 [6] – Model predictive control - 2019

This program will try to minimize the heat rate with bounds on control inputs and constraints on the measured outputs

Disturbance is changed at various test

Various modifications to the set point and the disturbances is added while constraints are active on the control input and the measured outputs.

Load in simulation data and model

```
clear;
load ss1.mat
load TrainingMean20200903Red.mat
```

Set up initial values by reading in data from real life training data

Find the initial states by using the "findstates" function. Then run the model for 30 samples with the real-life data to get updated initial states. Use this initial states further and use the next real life controllable inputs and measured disturbances as initial values for the testing next

```
StartLoc = 100000;
u_ini_s = TrainingMean20200903Red.InputData(StartLoc+201,:);

X0 =
findstates(ss1,iddata(TrainingMean20200903Red.OutputData(StartLoc+140:StartLoc+170,1:6),TrainingMean20200903Red.InputData(StartLoc+140:StartLoc+170,1:5),TrainingMean20200903Red.Ts));

for jj=1:30
    u_ini_s = TrainingMean20200903Red.InputData(StartLoc+170+jj,:);
    X_n = ss1.A*X0 + ss1.B*u_ini_s;
    X0 = X_n;
end

u1 = TrainingMean20200903Red.InputData(StartLoc+201,1) %SuctionAirTemp
u2 = TrainingMean20200903Red.InputData(StartLoc+201,2) %IMEP
u3 = TrainingMean20200903Red.InputData(StartLoc+201,3) %ChargeAirPressure
u4 = TrainingMean20200903Red.InputData(StartLoc+201,4) %ChargeAirTemperature
u5 = TrainingMean20200903Red.InputData(StartLoc+201,5) %GlobalTiming
```

```

%Receive u_previous measurements
u_k_prev(1) = TrainingMean20200903Red.InputData(StartLoc+200,3);
u_k_prev(2) = TrainingMean20200903Red.InputData(StartLoc+200,5);

%Controllable inputs B matrix
B(:,1) = ss1.B(:,3);
B(:,2) = ss1.B(:,5);

%Measured input disturbance Bd Matrix
Bd(:,1) = ss1.B(:,1);
Bd(:,2) = ss1.B(:,2);
Bd(:,3) = ss1.B(:,4);

```

Simulate with constraints on inputs and outputs

The following results is found by set point tracking with a gradient change in the IMEP disturbance with the other disturbances fixed. The set point is ramped also

This time we limit the control values to max/min and add constraint to the outputs

```

Sim =5000;
dt = 0.1;
Np = 150;

[y_n, c_n] = size(ss1.C);
[~, u_n] = size(B);
[~, d_n] = size(Bd);
[~, n_n] = size(ss1.A);

Py = zeros(Sim,y_n);
Pu = zeros(Sim,u_n);
Pdu = zeros(Sim,u_n);
Pud = zeros(Sim,d_n);
Pc = zeros(Sim,n_n);

state_ini_values = X0;

u_ini = [u3*ones(Np,1); ... %Charge air pressure
         u5*ones(Np,1)];   %Global timing

ud_ini = [u1*ones(Np,1); ...%Suction air temperature
          u2*ones(Np,1); ... %IMEP
          u4*ones(Np,1)];  %Charge air temperature

for i=1:Sim
    %Save values
    Pc(i,:) = state_ini_values;

```

```

    %Call the optimization
    u_k_ast =
optimization(u_ini,ud_ini,state_ini_values,Np,ss1.A,B,Bd,ss1.C,u_k_prev);

    %Pick the first two control moves
    u_k(1,1) = u_k_ast(1,1); %First control input - charge air pressure
    u_k(2,1) = u_k_ast(Np+1,1); %First control input - global timing

    u_ini = u_k_ast;

    %Save control inputs
    Pu(i,:) = u_k';
    Pdu(i,1)= u_k(1)-u_k_prev(1);
    Pdu(i,2)= u_k(2)-u_k_prev(2);
    u_k_prev = u_k;

    %Retrieve the disturbances
    ud_k(1,1) = ud_ini(1);      %First disturbance - Suction air temperature
    ud_k(2,1) = ud_ini(Np+1);  %First disturbance - IMEP
    ud_k(3,1) = ud_ini(2*Np+1); %First disturbance - charge air temperature

    %Save disturbance
    Pud(i,:) = ud_k';

    %Simulate process
    X_next = ss1.A*state_ini_values + B*u_k + Bd*ud_k;
    Py(i,:) = ss1.C*state_ini_values;

    state_ini_values = X_next;

    %Update the disturbances with real life data for iteration, but keep it
    %fixed for the duration of the prediction horizon
    ud_ini(1:Np) =
TrainingMean20200903Red.InputData(StartLoc+201+i,1)*ones(Np,1); ...%Suction
air temperature
    ud_ini(Np+1:2*Np) =
TrainingMean20200903Red.InputData(StartLoc+201+i,2)*ones(Np,1); ...%Suction
air temperature
    ud_ini(2*Np+1:3*Np) =
TrainingMean20200903Red.InputData(StartLoc+201+i,4)*ones(Np,1); ...%Suction
air temperature

end

Plotting(Pu,Pud,Py,Pdu,Sim)
%Plotting(Pu,Pud,Py,Pdu,Sim);

```

Pred function

This function takes the matrices and prediction horizon and calculate the F and ϕ matrices such that

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}, \phi_u = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \dots & CA^{N_p-N_c}B \end{bmatrix} \text{ and}$$

$$\phi_{u_d} = \begin{bmatrix} CB_d & 0 & 0 & \dots & 0 \\ CAB_d & CB_d & 0 & \dots & 0 \\ CA^2B_d & CAB_d & CB_d & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B_d & CA^{N_p-2}B_d & CA^{N_p-3}B_d & \dots & CA^{N_p-N_c}B_d \end{bmatrix}$$

$$Y = Fx(k_i) + \phi_u U + \phi_{u_d} U_d$$

Y is the predicted outputs where U is the control moves over the prediction horizon and U_d is the disturbance inputs over the prediction horizon (usually constant)

```
function [F, Phi_u, Phi_ud] = Pred(A,B,Bd,C,Np)

%Find the sizes of the matrices and set up their dimensions
[y_n,~] = size(C);
[~,u_n] = size(B);
[n_n,ud_n] = size(Bd);
F = zeros(y_n*Np,n_n);
Phi_u = zeros(y_n*Np,u_n*Np);
Phi_u_temp = zeros(y_n*Np,u_n*Np);
Phi_ud = zeros(y_n*Np,u_n*Np);
Phi_ud_temp = zeros(y_n*Np,ud_n*Np);

%%Find the F matrix
for jj=1:Np
    F((jj-1)*y_n+1:(jj-1)*y_n+y_n,1:n_n) = C*(A^jj);
end

%%Set up the Phi matrix for the controllable input
Phi_u_temp(1:y_n,1:u_n) = C*B;

for jj=1:Np-1
    Phi_u_temp(y_n*jj+1:y_n*jj+y_n,1:u_n) = C*(A^jj)*B;
end
```

```

%%Copy column 1 to the next and move down one row and so on
for jj=1:Np-1
    Phi_u_temp(y_n*jj+1:end,u_n*jj+1:u_n*jj+u_n) = Phi_u_temp(1:end-
jj*y_n,1:u_n);
end
%Rearrange the matrix as the predicted output is organized with all the
%predicitons for a single control output first and then the second control
%output
for jj=1:Np
    Phi_u(:,jj) = Phi_u_temp(:,(jj*u_n)-1);
    Phi_u(:,Np+jj) = Phi_u_temp(:,(jj*u_n));
end

%%Set up the Psi matrix for the measured disturbance

Phi_ud_temp(1:y_n,1:ud_n) = C*Bd; %Set up the first row since that is
without A matrix

%Set up the rest of the rows in the first column
for jj=1:Np-1
    Phi_ud_temp(y_n*jj+1:y_n*jj+y_n,1:ud_n) = C*(A^jj)*Bd;
end

%%Copy column 1 to the next and move down one row and so on
for jj=1:Np-1
    Phi_ud_temp(y_n*jj+1:end,ud_n*jj+1:ud_n*jj+ud_n) = Phi_ud_temp(1:end-
jj*y_n,1:ud_n);

end

%Rearrange the matrix as the predicted output is organized with all the
%predicitons for a single control output first and then the second control
%output
for jj=1:Np
    Phi_ud(:,jj) = Phi_ud_temp(:,(jj*ud_n)-2);
    Phi_ud(:,Np+jj) = Phi_ud_temp(:,(jj*ud_n)-1);
    Phi_ud(:,Np*2+jj) = Phi_ud_temp(:,(jj*ud_n));
end
end

```

Compute both functions

```

function [myJ,myG,myHeq] =
compute_both(u_ini,ud_ini,state_ini_values,Np,A,B,Bd,C,u_k_prev)

du_u1 = zeros(Np,1); %Charge air pressure rate of change place holder

```

```

du_u2 = zeros(Np,1); %Global timing rate of change place holder
[y_n,~] = size(C);
Pc = zeros(Np,y_n);

u_ini = u_ini';

%Weight factor for heat rate
Q = 0.0001;
%Weight matrix for the controllable outputs and
%rate of change
%Pu = diag([0.01, 0.01]);
Pdu = diag([10.00, 10.00]);

%Set objective to zero before the for loop
J = 0;

%we need to calculate the outputs for the whole prediction horizon
for i = 1:Np
    %find out which control input to use for each time step within the
    %prediction horizon
    %Calculate the change between the previous outputs and the next inputs.
    if i == 1
        du_u1(i) = u_ini(i)-u_k_prev(1);
        du_u2(i) = u_ini(Np+i)-u_k_prev(2);
    else
        du_u1(i) = u_ini(i)-u_ini(i-1);
        du_u2(i) = u_ini(Np+i)-u_ini(Np+i-1);
    end

    u_k(1,1) = u_ini(i);      %Charge air pressure
    u_k(2,1) = u_ini(Np+i);   %Global timing

    du_k = [du_u1(i);du_u2(i)];

    %find out which disturbance input to use for each time step within the
    %prediction horizon
    ud_k(1,1) = ud_ini(i);
    ud_k(2,1) = ud_ini(i+Np);
    ud_k(3,1) = ud_ini(i+Np*2);

    % u_k_prev = u_k;

    %Calculate the next states based on the current state, controllable inputs
    %and measured disturbance
    x_next = A*state_ini_values + B*u_k + Bd*ud_k;

    %use the states to calculate the output

```

```

Pc(i,:) = C*state_ini_values;

%Extract the output which is the once we want use as process value for our
%set point tracking
HeatRate = Pc(i,1); %Heat rate

%update the state
state_ini_values = x_next;

%now make the objective function
%J_temp = ((Pc_reg-Ref)/1000)'*Q*((Pc_reg-Ref)/1000) + du_k'*Pdu*du_k +
u_k'*Pu*u_k;
J_temp = HeatRate*Q + du_k'*Pdu*du_k;
J = J+J_temp; %Increment the objective over the whole prediction horizon
end

myJ = J/2;

%if there are equality constraints, it should be listed as a column vector
%here we don't have equality constraints so we use empty matrix
myHeq = [];
myG = [];
% myG =[du_u1-0.02; ...
%       -du_u1-0.02; ...
%       du_u2-0.05; ...
%       -du_u2-0.03];
end

```

Optimization function

```

function u =
optimization(u_ini,ud_ini,state_ini_values,Np,A,B,Bd,C,u_k_prev)

ops = optimset('Algorithm','interior-
point','Display','off','MaxIter',1000,'MaxFunEvals',1000,'AlwaysHonorConstrai
nts','bounds','TolX', 1e-14);

uLast = [];% Last place compute_both was called
myJ = [];% Use for objective at xLast
myG = [];% Use for nonlinear inequality constraint
myHeq = [];% Use for nonlinear equality constraint
[~,u_n] = size(B);
[y_n,~] = size(C);
%Deine lower and upper bounds on control inputs
lb = [ones(1,Np)*0.3, ...
      ones(1,Np)*-20 ...

```

```

];

ub = [ones(1,Np)*4.5, ...
      ones(1,Np)*-8.5 ...
      ];

obj_func = @(u)objfun_engine(u,ud_ini,state_ini_values,Np);
cons_func = @(u)confun_engine(u,ud_ini,state_ini_values,Np);

%Calculate the matrices to be used in the inequality constraints
[F, Phi_u, Phi_ud] = Pred(A,B,Bd,C,Np);

%Set up the constraints and repeat for the horizon
y_max = [Inf;50;200;200;12.5;600];
y_max = repmat(y_max,Np,1);

y_min = [-Inf;0;0;0;8.5;0];
y_min = repmat(y_min,Np,1);

%Calculat all the right hand side of the inequality constraint
YYY = y_max-F*state_ini_values-Phi_ud*ud_ini;
YY = -y_min+F*state_ini_values+Phi_ud*ud_ini;
%Extract only the constraints we are interessted in
Aneq = zeros(2*Np,u_n*Np);
Bneq = zeros(2*Np,1);

%Testing with only high limit on O2 and Exhaust and lower limit on O2
%For more constraints - add them
for ii=1:Np
    Aneq(ii*3-2,:) = Phi_u(ii*y_n-1,:);
    Aneq(ii*3-1,:) = Phi_u(ii*y_n,:);
    Aneq(ii*3,:) = -(Phi_u(ii*y_n-1,:));

    Bneq(ii*3-2,:) = YYY(ii*y_n-1);
    Bneq(ii*3-1,:) = YYY(ii*y_n);
    Bneq(ii*3,:) = YY(ii*y_n-1);
end

[u,fval,exitflag,output,solutions] =
fmincon(obj_func,u_ini,Aneq,Bneq,[],[],lb,ub,cons_func,ops);

function J = objfun_engine(u,ud_ini,state_ini_values,Np)
    if ~isequal(u,uLast) %check if computation is necessary
        [myJ,myG,myHeq] =
compute_both(u,ud_ini,state_ini_values,Np,A,B,Bd,C,u_k_prev);
    end
end

```



```

        uLast = u;
    end

    %now compute objective function
    J = myJ;
end

function [G,Heq] = confun_engine(u,ud_ini,state_ini_values,Np)
    if ~isequal(u,uLast) %check if computation is necessary
        [myJ,myG,myHeq] =
compute_both(u,ud_ini,state_ini_values,Np,A,B,Bd,C,u_k_prev);
        uLast = u;
    end

    G = myG;
    Heq = myHeq;
end
end

```

Plot function

```

function [] = Plotting(Pu,Pud,Py,Pdu,Sim)
fig_1 = figure("Name", 'Control variables');
x = linspace(1,Sim/10,Sim);
y1 = (Pu(:,1));
y2 = (Pu(:,2));
tiledlayout(2,1);

ax1 = nexttile;
plot(ax1,x,y1);
title(ax1,'Charge air pressure');
ylabel(ax1,'barg');
xlabel('Simulation time [s]')

ax2 = nexttile;
plot(ax2,x,y2);
title(ax2,'Global timing');
ylabel(ax2,"degCA");
xlabel('Simulation time [s]')

```

The figure below shows the disturbances over the simulation time

```

fig_2 = figure("Name", 'Disturbances');
x = linspace(1,Sim/10,Sim);
y1 = (Pud(:,1));

```

```

y2 = (Pud(:,2));
y3 = (Pud(:,3));
tiledlayout(3,1);

ax1 = nexttile;
plot(ax1,x,y1);
title(ax1,'Suction air temperature');
ylabel(ax1,'degC');
xlabel('Simulation time [s]')

ax2 = nexttile;
plot(ax2,x,y2);
title(ax2,'IMEP');
ylabel(ax2,"bar");
xlabel('Simulation time [s]')

ax3 = nexttile;
plot(ax3,x,y3);
title(ax3,'Charge air temperature');
ylabel(ax3,"degC");
xlabel('Simulation time [s]')

```

The figure below shows the model outputs over the simulation time

```

fig_3a = figure("Name",'Measurements');
x = linspace(1,Sim/10,Sim);
y1 = (Py(:,1));
y2 = (Py(:,2));
y3 = (Py(:,3));
y4 = (Py(:,4));
y5 = (Py(:,5));
y6 = (Py(:,6));
Lim_exh = 600*ones(Sim,1);
Lim_O2_1 = 8.5*ones(Sim,1);
Lim_O2_2 = 12.5*ones(Sim,1);
t = tiledlayout(3,1,'TileSpacing','normal');
title(t,'Measurements')

ax1 = nexttile;
plot(ax1,x,y1);
title(ax1,'Heat rate');
ylabel(ax1,'-');
xlabel('Simulation time [s]')

ax2 = nexttile;
plot(ax2,x,y2);
title(ax2,'Knock');
ylabel(ax2,'%');

```

```

xlabel('Simulation time [s]')

ax3 = nexttile;
plot(ax3,x,y3);
title(ax3,'Peak pressure');
ylabel(ax3,'bar');
xlabel('Simulation time [s]')

fig_3b = figure("Name",'Measurements');
tt = tiledlayout(3,1,'TileSpacing','normal');
ax4 = nexttile;
plot(ax4,x,y4);
title(ax4,'NOx');
ylabel(ax4,'ppm');
xlabel('Simulation time [s]')

ax5 = nexttile;
plot(ax5,x,y5,x,Lim_O2_1,x,Lim_O2_2);
title(ax5,'O2');
ylabel(ax5,'%');
xlabel('Simulation time [s]')

ax6 = nexttile;
plot(ax6,x,y6,x,Lim_exh);
title(ax6,'Exhaust temperature');
ylabel(ax6,'degC');
xlabel('Simulation time [s]')

fig_3c = figure("Name",'Measurements');
x = linspace(1,Sim/10,Sim);
y1 = (Py(:,1));
t = tiledlayout(1,1,'TileSpacing','normal');
title(t,'Measurements')
ax1 = nexttile;
plot(ax1,x,y1);
title(ax1,'Heat rate');
ylabel(ax1,'-');
xlabel('Simulation time [s]')

fig_3d = figure("Name",'Measurements');
ax5 = nexttile;
plot(ax5,x,y5,x,Lim_O2_1,x,Lim_O2_2);
title(ax5,'O2');
ylabel(ax5,'%');
xlabel('Simulation time [s]')

ax6 = nexttile;
plot(ax6,x,y6,x,Lim_exh);

```

```
title(ax6,'Exhaust temperature');
ylabel(ax6,'degC');
xlabel('Simulation time [s]')
```

Plot Δu for both control signals

```
fig_4 = figure("Name",'delta u');
ax7 = nexttile;
plot(ax7,x,(Pdu(:,1)),x,0.02*ones(Sim,1),x,-0.02*ones(Sim,1));
title(ax7,'Charge air pressure');
ylabel(ax7,'bar/ms');
xlabel('Simulation time [s]')

ax8 = nexttile;
plot(ax8,x,(Pdu(:,2)),x,0.05*ones(Sim,1),x,-0.03*ones(Sim,1));
title(ax8,'Global timing');
ylabel(ax8,'degCA/ms');
xlabel('Simulation time [s]')

end
```