MODEL PREDICTIVE CONTROL OF DIESEL ENGINE AIR PATH WITH ACTUATOR DELAYS

by

Betul Kekik

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2019

MODEL PREDICTIVE CONTROL OF DIESEL ENGINE AIR PATH WITH ACTUATOR DELAYS

APPROVED BY:

Prof. Mehmet Akar .....
(Thesis Supervisor)

Prof. Yağmur Denizhan .....

Assist. Prof. Yeşim Öniz .....

DATE OF APPROVAL: 25.12.2018

ACKNOWLEDGEMENTS

Foremost, I would like to thank my dear supervisor, Professor Mehmet Akar for his invaluable guidance and enlightening contributions. He taught me how to do research and present an academic work, which was an invaluable experience for me.

I also would like to express my special thanks to Professor Yağmur Denizhan and Assist. Prof. Yeşim Öniz for their invaluable support and being a member of my thesis jury.

I am sincerely grateful to my colleagues Uğur Yavaş, Fatih Kendir, and Tevhide Dayıoğlu for their great support and assistance during this work. I would also like to thank AVL Turkey for supporting me during my master education.

I should thank all of my friends Nefise Yağlıkçı, Şeyda Çetintaş, Merve Kaştan, Emre Koray Karpuz, and Fatih Ünal for their continuous support and encouragement. Everything would be much harder without them.

Lastly, this thesis is dedicated to my mother Dilek and my sister Büşra. I am really grateful to them and my uncles Kudret and Fikret Ak for supporting me not only for this research, but also throughout my life.

ABSTRACT

MODEL PREDICTIVE CONTROL OF DIESEL ENGINE AIR PATH WITH ACTUATOR DELAYS

In recent years, emission regulations of the countries have been tightened with the increase in the number of cars. Nevertheless, reducing the emissions of diesel engines is a difficult technical issue. Since the Diesel Engine Air Path (DEAP) is a MIMO system with 2 actuators, EGR valve and VGT valve, and 2 outputs, manifold absolute pressure (MAP) and air mass flow (MAF), control of this system with SISO PID controllers requires an iterative fine-tuning process for controller parameters due to coupled effect of VGT and EGR on MAP and MAF.

The main objective of this thesis is the Model Predictive Control (MPC) of MIMO Diesel Engine Air Path system. MPC is a well-known technique in the literature with its many applications on MIMO systems. Existing results show that MPC can satify the desired settling time, zero steady-state error, and overshoot criteria for both outputs MAP and MAF. However, the effect of actuator delay that considerably affects system performance, is not addressed sufficiently. In this thesis, MPC of DEAP is extended with a delay term added to actuators EGR and VGT on the plant model. A linear state-space model of the plant is obtained by using the System Identification techniques and the states of the identified model are extended due to delay term. It is shown that MPC performs better when the delay is taken into account in the algorithm. Another contribution of this thesis is that SISO PID controllers are optimized by Particle Swarm Optimization (PSO) method. The PID gains found by Ziegler-Nichols (ZN) method are taken as the initial points of the PSO and it is shown that PSO improves the PID controller performance for the MIMO system.

ÖZET

EYLEYİCİ GECİKMELİ DİZEL MOTOR HAVA YOLUNUN MODEL ÖNGÖRÜLÜ KONTROLÜ

Son yıllarda, otomobillerin sayısındaki artışla ülkelerin emisyon düzenlemeleri sıkılaştırılmıştır. Bununla birlikte, dizel motorların emisyonlarının azaltılması zor bir teknik konudur. Dizel Motor Hava Yolu (DMHY), 2 eyleyici, EGR valfi ve VGT vanası, ve 2 çıkışı, manifold mutlak basıncı ve hava kütle akışı, olan çok giriş çok çıkışlı (ÇGÇÇ) bir sistem olduğundan, tek giriş tek çıkışlı (TGTÇ) PID denetleyiciler ile bu sistemin kontrolü, çıkışlar üzerinde eyleyicilerin birleşik etkisi nedeniyle kontrol parametreleri için yinelenen ince ayar işlemi gerektirir.

Bu tezin temel amacı, ÇGÇÇ DMHY sisteminin Model Öngörülü Kontrolüdür. Model Öngörülü Kontrol, literatürde ÇGÇÇ sistemler ile ilgili birçok uygulama ile iyi bilinen bir tekniktir. Mevcut sonuçlar, Model Öngörülü Kontrolün istenen çökme süresini, sıfır sabit durum hatasını ve sistem çıkışlarının her ikisinin de aşma kriterlerini karşılayabildiğini göstermektedir. Bununla birlikte, sistem performansını önemli ölçüde etkileyen eyleyici gecikmesinin etkisi yeterince ele alınmamıştır. Bu tezde, DMHY'nun Model Öngürülü Kontrolü, sistem modelinde EGR ve VGT eyleyicilerine eklenen gecikme süresiyle genişletilmiştir. Sistem tanımlama teknikleri kullanılarak, sistemin doğrusal bir durum-uzay modeli elde edilmiş ve belirlenen modelin durumları gecikme süresi nedeniyle genişletilmiştir. Algoritmada gecikme göz önüne alındığında Model Öngörülü denetleyicinin daha iyi performans gösterdiği gösterilmiştir. Bu tezin bir başka katkısı ise TGTÇ PID denetleyicilerinin Parçacık Sürü Optimizasyonu (PSO) yöntemi ile optimize edilmiş olmasıdır. Ziegler-Nichols (ZN) yöntemi ile bulunan PID kazanımları PSO'nun başlangıç noktaları olarak alınmıştır ve PSO'nun, ÇKÇÇ sistem için PID denetleyici performansını iyileştirdiği gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A_p	State transition matrix of plant model
A_r	Effective area of EGR valve
B_p	Input matrix of plant model
C_p	Output matrix of plant model
D_p	Direct transition matrix of plant model
J_{dr}	Driveline inertia
J_e	Engine inertia
J_t	Turbocharger inertia
K_d	Derivative gain of PID controller
K_i	Integral gain of PID controller
K_p	Proportional gain of PID controller
N	Crankshaft speed
N_c	Control horizon
N_p	Prediction horizon
N_t	Turbocharger speed
p_a	Compressor inlet pressure
p_c	Compressor outlet pressure
P_c	Compressor power
P_t	Turbine power
T_a	Compressor inlet temperature
T_c	Compressor downstream temperature
T_t	Turbine outlet temperature
V_d	Displacement volume
W_{ci}	Compressor air mass flow
W_{ie}	Air mass flow from intake manifold to cylinders
W_{xt}	Turbine air mass flow
W_{xi}	Air mass flow through the EGR valve
x_n	State vector of measurement noise model

x_{od}	State vector of output disturbance model
x_p	State vector of plant model
y_n	Output vector of measurement noise model
y_{od}	Output vector of output disturbance model
y_p	Output vector of plant model
ΔU	Input change rate
ϵ	Slack variable
η_c	Isentropic efficiency
η_v	Volumetric efficiency

LIST OF ACRONYMS/ABBREVIATIONS

DEAP	Diesel Engine Air Path
DPF	Diesel Particulate Filter
EGR	Exhaust Gas Recirculation
IAE	Integral of Absolute Error
ISE	Integral of Square Error
ITAE	Integral of Time-Weighted Absolute Error
MAF	Mass Air Flow
MAP	Manifold Absolute Pressure
MIMO	Multi-Input Multi-Output
MPC	Model Predictive Control
NOx	Nitrogen Oxids
PEM	Prediction Error Method
PID	Proportional Integral Derivative
PM	Particulate Matter
PRBS	Pseudo Random Binary Sequence
PSO	Particle Swarm Optimization
QP	Quadratic Programming
SCR	Selective Catalytic Reduction
SISO	Single-Input Single-Output
VGT	Variable Geometry Turbocharger
ZN	Ziegler-Nichols

1. INTRODUCTION

In recent years, vehicles powered by diesel engines are preferred due to their advantages over gasoline engines in terms of fuel consumption, torque output, and carbon oxides (CO, CO2) emissions. However, because of their high nitrogen oxide (NOx) and particulate matter (PM) generation, emission regulations have tightened that makes diesel engine production a challenging task for the suppliers in terms of development, control, and calibration processes.

To obey the emission legislation, some components are added to the engine air path such as Exhaust Gas Recirculation (EGR) valve, Diesel Particulate Filter (DPF), and Selective Catalytic Reduction (SCR). On the other hand, high performance is achieved from the engine with the addition of a turbocharger system which is a novel technology that makes the engine size smaller and increases the power output with the same fuel consumption compared to normally aspirated engine.

In automotive, using separate single input single output (SISO) control loops with proportional-integral-derivative (PID) controllers is still the most popular technique to control the diesel engine air path (DEAP). However, with the increase in the number of components added to it, traditional methods become time consuming and complex due to increasing number of control parameters. Model Predictive Control (MPC) is a promising technique for such complex and multi-input multi-output (MIMO) systems.

1.1. Related Work

Control of the diesel engine air path with MPC is a recent popular research topic. There are several studies in literature to deal with this problem. In [1], instead of using two separate single-input single output (SISO) control loops as in the production engines, explicit MPC approach is adopted to get good tracking results for both mass air flow (MAF), and intake manifold pressure (MAP). First of all, since air path of diesel engine is a highly nonlinear system, it is modeled as a combination of linear systems.

Although clustering is recommended to divide the operating range into regions, it is divided into 12 regions empirically. Engine speed and fuel injection are the quantities describing the operating range. Then, locally linear models are used to define each region. Prediction error method (PEM) with 2nd order models is chosen to obtain good identification results. As a control strategy, EGR and VGT positions are taken as manipulated variables, while engine speed and injected fuel are taken into account as measured disturbances because they are the dominant parameters determining the warm engine dynamics. Subsequently, control problem is formulated as an optimal problem with constraints. Due to the computational effort of online QP solution at each step, the cost function is treated as a multiparametric quadratic program (mp-QP). Therefore, explicit MPC becomes useful for high-speed applications. The proposed approach is tested on a BMW M47D engine. An improvement of NOx-PM tradeoff is obtained.

Another study on the implementation of explicit MPC approach to diesel engine air path is [2]. The proposed algorithm for the air path regulation in turbocharged diesel engines allows tracking of the time-varying set point values generated by the supervisory level controller while satisfying the actuator constraints. To obey the emission standards, EGR and VGT actuators should be well tuned for regulating the intake mass flow for combustion with the desired burnt gas fraction to minimize NOx, without violating the air-fuel ratio associated with the particulate matter (PM) generation. However, since burnt gas fraction and air-fuel ratio are unmeasurable quantities using normal sensors, two intermediate variables, compressor air mass flow rate and intake manifold pressure are introduced as the new controlled variables, which are closely related with the previous ones. To solve this complex problem, Explicit MPC is preferred as one of the most promising strategies in industrial applications. Firstly, the operation range of the engine is divided into sub-regions and each sub-region is modelled by a proper linear system ranging from 2nd order to 3rd order by using a system identification method based on the data. Then, an augmented explicit MPC (EMPC) design is carried out and implemented on a heavy-duty off-highway engine. As a result, compared to traditional MPC, EMPC provides a time-saving way in real-time applications, while maintaining the identical performance as MPC. Implementation of

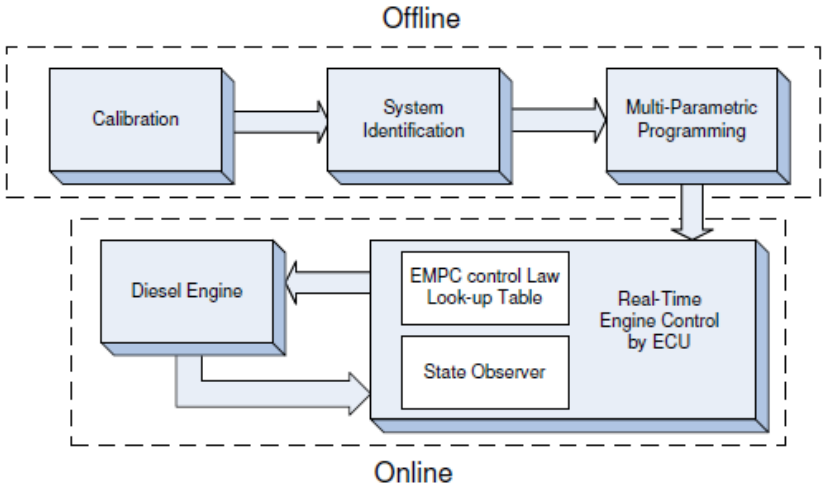

Figure 1.1. Implementation Procedure of EMPC on the Diesel Engine [2].

the EMPC procedure is shown in Figure 1.1.

Because of the nonlinear dynamics of the air-path system and constraints on inputs and process variables, achieving desired performance specifications is a challenging task. For this reason, instead of linear model predictive control techniques, nonlinear model predictive control is applied to the turbocharged diesel engine in [3]. Firstly, a third order nonlinear model of diesel engine is described with the assumption of constant intake and exhaust manifold temperatures. Later, closed loop stability of the NMPC approach is shown. The objective of the control is to track the set points of the intake manifold pressure, exhaust manifold pressure, compressor power, and effective areas of EGR and turbine valves. Lastly, optimal control problem is solved at each sample in finite horizon and applied to the system as in the conventional MPC approach. Although simulation results show that a better transient performance is achieved, NMPC is not suitable for fast systems due to its high computation time.

Nonlinear Model Predictive Control (NMPC) of a diesel engine air path is studied in [4]. Instead of modeling the air path by a multilinear approach, a data based Linear Parameter Varying (LPV) model is used to obtain a superior tracking result. Another reason for choosing LPV model is to avoid the huge parametrization work of a mean

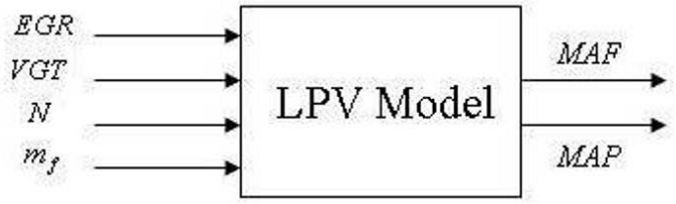

Figure 1.2. Inputs and Outputs of LPV Model [4].

value model. Two MISO models are used to identify MAF and MAP independently according to inputs EGR positon, VGT positon, engine speed, and fuel flow that are shown in Figure 1.2. To calculate the MAF and MAP, first order transfer functions with different parameters are defined. VGT position and engine speed N are taken as the scheduling parameters of the transfer functions. To identify the model parameters, data-based Autoregression with Exogenous Input (ARX) method is used. After the identification step, the nonlinear model is substituted into the MPC formulation. The obtained quadratic problem is solved by using the active set strategy method. To test the performance of NMPC, LPV model is linearized and a linear MPC is designed. In simulations, it is shown that LPV NMPC performs better than a linear MPC since it involves the exact model.

In [5], a fixed geometry turbo and a variable valve actuation (VVA) system are controlled by using the MPC technique with intake valve hold (IVH), intake valve close (IVC), advanced angle, and needle opening pressure (NOP) as the manipulating variables. The aim of this study is to minimize the pumping losses to decrease the fuel consumption while CO2 based burned gas fraction (BGF) tracks a desired set point and air-fuel ratio stays greater than a limit. The reason of selecting BGF as the control objective is its strong relationship to NOx emissions. Because of the nonlinear characteristic of the engine and limited memory of the electronic control unit (ECU), multiple linear models are used to cover the operating range of the engine. Linear Kalman filters are used to estimate the state variables of the linear models. A simplified implicit MPC approach is proposed as a control strategy. Simulation results obtained

from a six cylinder heavy-duty Volvo Diesel engine exhibit the good performance of MPC with regards to fuel consumption and emission rates.

As a difference from [5], instead of VVA, EGR valve and VGT are controlled by multi-linear MPC in [6]. 192 linear models are used to describe the air path. The objective functions are to keep NOx emissions under a certain limit by tracking the set points of BGF and to keep air-fuel ratio greater than a reference level while minimizing fuel consumption. Implicit MPC performed on a production engine is shown to improve transient response while decreasing the fuel consumption and emissions.

1.2. Motivation of the Thesis

As discussed above, there are several studies in the literature that deal with the DEAP control problem by using the EMPC and Nonlinear MPC methods, and linear or LPV models of the system. However, control of input delayed DEAP system has not been addressed sufficiently. In this thesis, MPC of a MIMO system with and without input delay is studied. Also, SISO PID controllers are tuned by Ziegler-Nichols (ZN) Method and Particle Swarm Optimization (PSO). The main motivation of the thesis is to cope with the tight emission regulations and get rid of the tuning effort of standard gain-scheduled PID controllers. The main objectives are listed below:

- System identification is performed for different operation regions of a nonlinear system.
- MPC method is practiced on a MIMO DEAP system. Input, input rate, and output constraints are handled in the optimization problem. All of the functions and algorithm are developed in Matlab environment to make use of the Quadratic Optimization Problem solver.
- Input delay effect on control system design is examined. Deterioration of the control performance due to the dead time is prevented by the force of Model Predictive Control.
- The performance of the standard ZN-PID controllers is increased by using the PSO method for MIMO system.

1.3. Organization of the Thesis

The organization of the rest of the thesis is as follows:

In Chapter 2, the main components of the diesel engine air path are introduced. Firstly, physical equations of the components used in the Mean Value Modeling (MVM) approach are described. Then, Linear Parameter Varying (LPV) modeling method is defined to use in model base control algorithms. Lastly, AVL Boost RT model layout is presented.

In Chapter 3, Model Predictive Control technique is explained with all steps for systems with linear state-space models. Firstly, the original model is extended with disturbance models in order to deal with the model-plant mismatch. Secondly, state estimation step is detailed for the output prediction. Then, the optimization problem with its quadratic cost function and constraints are established.

In Chapter 4, the system identification study of the Boost RT model is presented. Also, Model Predictive Control of the system is described with its tuning parameters. The results of the controller are given when the operation region and set points of the system are changed.

In Chapter 5, the actuator delay is added to the plant. The model used in the MPC algorithm is redefined to deal with the input delay effect. The performance of the MPC controller is evaluated in the case of actuator delays.

In Chapter 6, standard SISO PID controllers are designed for the control of the MIMO system with and without actuator delay. Tuning of the PID controllers are done by the ZN Method and improved by the PSO method. Finally, the results of the 3 control methods are compared.

In Chapter 7, the thesis concludes with a brief summary and future recommendations.

2. DIESEL ENGINE AIR PATH

A basic physical model layout of the diesel engine is as below:

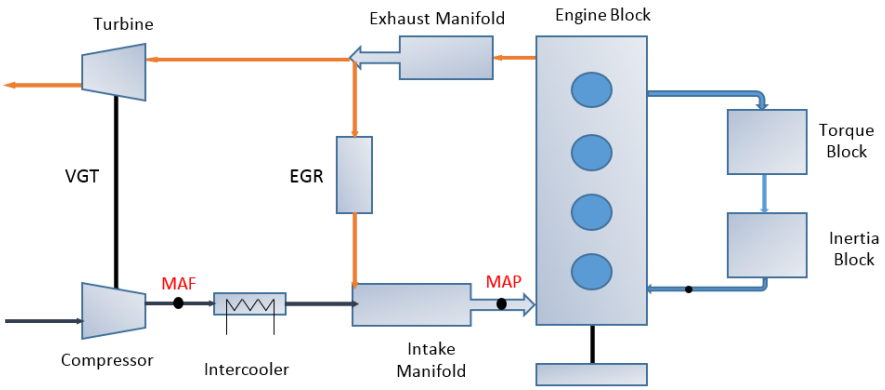

Figure 2.1. Diesel Engine Air Path.

Variable Geometry Turbocharger (VGT) and Exhaust Gas Recirculation (EGR) valve are strongly coupled MIMO system in the air path of the engine, since they are both driven by the exhaust gas. The working principle of the VGT is to use the power of waste gasses produced during combustion in order to rotate the turbine wheel and to compress the air transmitted to intake manifold by compressor which is connected to turbine with a shaft. VGT controls Mass Air Flow (MAF) by changing the blade angles or vane in the turbine. On the other hand, EGR valve provides reduced NOx emissions by recirculating some of the exhaust gas. In this way, temperature in the cycliders are reduced due to mixture of fresh air with recirculated burnt gas that has a poor portion of oxygen. As a consequence, reduction in the temperature decreases the NOx formation rate during the chemical reactions. Manifold Absolute Pressure (MAP) is the other parameter to be controlled and it is directly affected by the relationship between EGR valve position and VGT valve position. However in the industry, multiple SISO control loops with PID controllers are used to manage this MIMO system.

2.1. Modelling Diesel Engine Air Path

Since MPC requires accurate models for prediction, a sufficient model of the diesel engine air path is the first step for the implementation of MPC. One of the most popular ways to model the diesel engine air path is the Mean Value Modeling (MVM) approach that is based on the physical equations. An 8th order model is derived in [7] with the equations in Sections 2.1.1, 2.1.2, and 2.1.3:

2.1.1. Turbocharger

Turbocharger block contains compressor and turbine blocks that are connected via shaft. The rotational speed of this turbocharger shaft is expressed as,

$$\dot{N}_t = \left(\frac{60}{2\pi}\right)^2 \frac{P_t - P_c}{J_t N_t} \tag{2.1}$$

where N_t is the turbocharger speed, J_t is the turbocharger inertia, P_t and P_c are the power of turbine and compressor, respectively. The following expressions describe the dynamics of the compressor side:

$$\left(\frac{T_{c,is}}{T_a}\right) = \left(\frac{p_c}{p_a}\right)^{\frac{\gamma-1}{\gamma}} \tag{2.2}$$

$$\eta_c = \frac{T_{c,is} - T_a}{T_c - T_a} \tag{2.3}$$

By combining (2.2) and (2.3), one obtains

$$T_c = T_a + \frac{1}{\eta_c} T_a \left(\left(\frac{p_c}{p_a}\right)^{\frac{\gamma}{\gamma-1}} - 1\right) \tag{2.4}$$

Equation (2.4) gives the downstream temperature of the compressor where T_a and p_a are the temperature and pressure at the inlet, p_c is the pressure at the outlet, γ is the specific heat ratio, η_c is the isentropic efficiency that is calculated by using the relation

between theoretical temperature ($T_{c,is}$) and the actual temperature (T_c). By using the thermodynamics law, multiplication of compressor mass flow (W_{ci}) and enthalpy change yields the compressor power:

$$\begin{aligned} P_c &= W_{ci} c_p (T_c - T_a) \\ &= W_{ci} c_p T_a \frac{1}{\eta_c} \left(\left(\frac{p_c}{p_a} \right)^{\frac{\gamma}{\gamma - 1}} - 1 \right) \end{aligned} \tag{2.5}$$

The dynamics of the turbine are similar to those of the compressor and are described by

$$T_t = T_x - \eta_t T_x \left(1 - \left(\frac{p_t}{p_x} \right)^{\frac{\gamma}{\gamma - 1}} \right) \tag{2.6}$$

$$P_t = W_{xt} c_p T_x \eta_t \left(1 - \left(\frac{p_t}{p_x} \right)^{\frac{\gamma}{\gamma - 1}} \right) \tag{2.7}$$

where T_t and p_t are the turbine outlet temperature and pressure, P_t is the turbine power, and W_{xt} is the turbine flow.

2.1.2. Engine

By defining J_e as engine inertia, J_{dr} as driveline inertia, T_l as the load torque, T_b as the difference between torque obtained from the cylinders and friction torque, the crankshaft speed is derived as

$$\dot{N} = \frac{60}{2\pi} \frac{T_b - T_l}{J_e - J_{dr}} \tag{2.8}$$

Equation (2.9) gives the mass flow rate from intake manifold to cylinders where η_v is the volumetric efficiency and V_d is the displacement volume.

$$W_{ie} = \eta_v \frac{m_i}{V_i} \frac{N}{60} \frac{V_d}{2} \tag{2.9}$$

2.1.3. Manifolds

Intake and Exhaust manifolds are the other components to be modeled in order to complete the diesel engine air path cycle. Conservation of Energy, Conservation of Mass, and Ideal Gas laws give the manifold pressures ($\dot{p}_i$ and $\dot{p}_x$), accumulation rate of mass in manifolds ($\dot{m}_i$ and $\dot{m}_x$), and manifold temperatures (T_i and T_x):

$$\begin{aligned} \dot{p}_i &= \frac{\gamma R}{V_i}\left(T_{ic}W_{ci} + T_r W_{xi} - T_i W_{ie}\right) \\ \dot{p}_x &= \frac{\gamma R}{V_x}\left(T_e W_{ex} - T_x(W_{xi} + W_{xt})\right) \end{aligned} \tag{2.10}$$

$$\begin{aligned} \dot{m}_i &= W_{ci} + W_{xi} - W_{ie} \\ \dot{m}_x &= W_{ex} - W_{xi} - W_{xt} \end{aligned} \tag{2.11}$$

$$\begin{aligned} T_i &= \frac{V_i}{R m_i} p_i \\ T_x &= \frac{V_x}{R m_x} p_x \end{aligned} \tag{2.12}$$

Standard orifice equation is used to calculate the mass flow through the EGR valve (W_{xi}) as follows:

$$W_{xi} = \frac{A_r(x_r) p_x}{\sqrt{R T_x}} \sqrt{\frac{2\gamma}{\gamma - 1}\left[p_r^{\frac{2}{\gamma}} - p_r^{\frac{\gamma+1}{\gamma}}\right]} \tag{2.13}$$

where

$$p_r = max\left(\frac{p_i}{p_x}, \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma}{\gamma - 1}}\right) \tag{2.14}$$

Lastly, the effect of intercooler is added to the system. Then, the downstream temperatures of EGR and intercooler (T_{down}) are calculated by using the heat exchanger effectiveness ($\eta_{h.e.}$), the upstream temperature (T_{up}), and the coolant temperature (T_{cool})

as follows:

$$T_{down} = \eta_{h.e.} T_{cool} + (1 - \eta_{h.e.}) T_{up} \tag{2.15}$$

Using such a high order model makes the control design complex and difficult to implement. For this reason, in [8], the higher order model is simplified to a third order nonlinear model by differentiating the ideal gas law and the following linear parameter varying (LPV) form of the model is obtained:

$$\begin{aligned} \dot{p}_i &= \frac{RT_i}{V_i}(W_{ci} + W_{xi} - W_{ie}) + \frac{\dot{T}_i}{T_i} p_i \\ \dot{p}_x &= \frac{RT_x}{V_x}(W_{ie} + W_f - W_{xi} + W_{xt}) + \frac{\dot{T}_x}{T_x} p_x \\ \dot{P}_c &= \frac{1}{\tau}(-P_c + \eta_m P_t) \end{aligned} \tag{2.16}$$

where τ is a time constant, η_m is the turbocharger mechanical efficiency. To further simplify the model, temperature changes ($\dot{T}_i$ and $\dot{T}_x$) are neglected and n_m is assumed to be equal to 1. The compressor flow is given by

$$W_{ci} = \frac{\eta_c}{c_p T_a} \frac{P_c}{\left(\frac{p_i}{p_a}\right)^{\mu} - 1}, \quad \mu = \frac{\gamma - 1}{\gamma} = 0.286 \tag{2.17}$$

The EGR flow is also defined as:

$$W_{xi} = \frac{A_r(x_r) p_x}{\sqrt{RT_x}} \sqrt{2 \frac{p_i}{p_x} \left(1 - \frac{p_i}{p_x}\right)} \tag{2.18}$$

where A_r is the effective area of EGR valve that depends on the valve lift x_r. Finally, mass flow from the intake manifold to cylinders W_{ie}, turbine flow W_{xt}, and turbine power P_t are defined by taking the reference temperature $T_{ref} = 298K$ and pressure $p_{ref} = 101.3kPa$ and by linearizing the effective area as a function of VGT position:

$$W_{ie} = \eta_v \frac{p_i}{T_i R} \frac{N}{60} \frac{V_d}{2} \tag{2.19}$$

$$W_{xt} = (ax_v + b)\left(c\left(\frac{p_x}{p_a} - 1\right) + d\right)\frac{p_x}{p_{ref}}\sqrt{\frac{T_{ref}}{T_x}\sqrt{2\frac{p_a}{p_x}\left(1 - \frac{p_a}{p_x}\right)}}, \tag{2.20}$$
$$a = 490.4, b = 633.7, c = 0.4, d = 0.6$$

$$P_t = W_{xt} c_p T_x \eta_t \left(1 - \left(\frac{p_a}{p_x}\right)^{\mu}\right) \tag{2.21}$$

By combining (2.16)-(2.21), we obtain

$$\begin{aligned}
\dot{p}_i &= \frac{RT_i}{V_i}\frac{n_c}{c_p T_a}\frac{\mathbf{P_c}}{\left(\frac{p_i}{p_a}\right)^{\mu} - 1} - \frac{p_i}{V_i}\frac{\mu_v V_d}{2\cdot 60}\mathbf{N} + \frac{RT_i}{V_i}\frac{p_x}{\sqrt{RT_x}}\sqrt{2\frac{p_i}{p_x}\left(1 - \frac{p_i}{p_x}\right)}\mathbf{A_r} \\
\dot{p}_x &= \frac{T_x}{T_i}\frac{p_i}{V_x}\frac{\eta_v V_d}{2\cdot 60}\mathbf{N} - \frac{RT_x}{V_x}\frac{p_x}{\sqrt{RT_x}}\sqrt{2\frac{p_i}{p_x}\left(1 - \frac{p_i}{p_x}\right)}\mathbf{A_r} \\
&\quad - \frac{RT_x}{V_x}a\left(c\left(\frac{p_x}{p_a}\right) + d\right)\frac{p_x}{p_{ref}}\sqrt{2\frac{p_a}{p_x}\left(1 - \frac{p_a}{p_x}\right)}\sqrt{\frac{T_{ref}}{T_x}}\mathbf{x_v} \\
&\quad - \frac{RT_x}{V_x}b\left(c\left(\frac{p_x}{p_a}\right) + d\right)\frac{\mathbf{p_x}}{p_{ref}}\sqrt{2\frac{p_a}{p_x}\left(1 - \frac{p_a}{p_x}\right)}\sqrt{\frac{T_{ref}}{T_x}} \\
&\quad + \frac{RT_x}{V_x}\mathbf{W_f} \\
\dot{P}_c &= -\frac{\mathbf{P_c}}{\tau} \\
&\quad + \frac{\eta_t c_p T_x}{\tau}\left(1 - \left(\frac{p_a}{p_x}\right)^{\mu}\right)\frac{p_x}{p_{ref}}\sqrt{\frac{T_{ref}}{T_x}}a\left(c\left(\frac{p_x}{p_a} - 1\right) + d\right)\sqrt{2\frac{p_a}{p_x}\left(1 - \frac{p_a}{p_x}\right)}\mathbf{x_v} \\
&\quad + \frac{\eta_t c_p T_x}{\tau}\left(1 - \left(\frac{p_a}{p_x}\right)^{\mu}\right)\frac{\mathbf{p_x}}{p_{ref}}\sqrt{\frac{T_{ref}}{T_x}}b\left(c\left(\frac{p_x}{p_a} - 1\right) + d\right)\sqrt{2\frac{p_a}{p_x}\left(1 - \frac{p_a}{p_x}\right)}
\end{aligned} \tag{2.22}$$

LPV systems are described by models with matrices depending on a time varying parameter $\rho(t)$. A general form of an LPV system is:

$$\begin{aligned}
\dot{x} &= A(\rho(t))x + B(\rho(t))u \\
y &= C(\rho(t))x + D(\rho(t))u
\end{aligned} \tag{2.23}$$

For the case of the air path model, (2.22) can be written in the form of (2.23) by taking the intake manifold pressure p_i, exhaust manifold pressure p_x and compressor power P_c as states; effective area of EGR valve A_r and VGT position x_v as the manipulated variables; and engine speed N and fuel flow W_f as the external disturbances. Since the scheduling parameter contains the states, a quasi-LPV model is obtained as follows:

$$\begin{bmatrix} \dot{p}_i \\ \dot{p}_x \\ \dot{P}_c \end{bmatrix} = A(\rho(t)) \begin{bmatrix} p_i \\ p_x \\ P_c \end{bmatrix} + B(\rho(t)) \begin{bmatrix} A_r \\ x_v \\ N \\ W_f \end{bmatrix} \tag{2.24}$$

where $\rho(t)$ is a time varying scheduling parameter.

2.2. Boost RT Model

For control oriented purposes, development of a diesel engine air path model with Mean Value Modeling approach is a time consuming and complex process due to its highly nonlinear dynamics. Therefore, a model designed in the AVL Boost RT software is used in this study. AVL Boost RT is a tool that allows component based model development. Each component works according to physical equations and only requires the parameters specific to the designed model. Moreover, the software enables to import the developed model to Matlab Simulink for controller simulations [9].

The overview of the diesel engine model imported to Simulink is shown in Figure 2.2. The model has five inputs namely engine speed (`n_sp`), fuel (`mf_tot`), VGT position (`ang_vgt`), EGR position (`ang_egr`), and start of injection setpoint which is determined by the 2D map depending on the engine speed and fuel. Depending on these inputs, only two outputs MAF and MAP are used for control purposes.

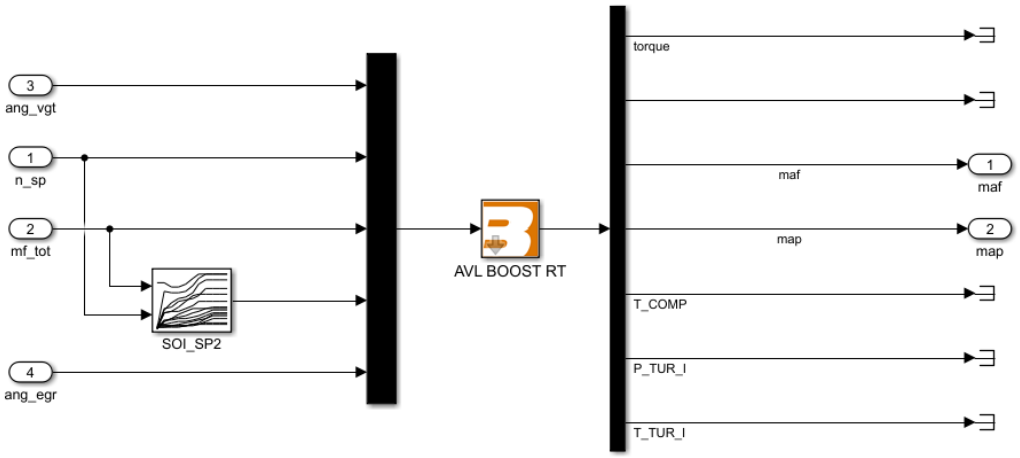

Figure 2.2. Simulink Interface of Boost RT Model

2.3. Summary of the Chapter

This chapter includes the physical equations of the components in the DEAP, i.e., turbocharger, engine, manifolds, and intercooler. Additionally, LPV modeling has been described for DEAP modeling that is used in example MPC applications in the literature. Lastly, AVL Boost RT plant model has been presented which will be used in this thesis for control purpose.

3. MODEL PREDICTIVE CONTROL

Model Predictive Control (MPC) is a trending strategy for the control of multi input-multi output systems. The objective of MPC is to calculate the future manipulated inputs denoted by u to optimize the future behaviour of the plant output denoted by y [2]. It determines the optimal control sequence that minimizes the objective function within a limited optimization window at each instant and then, uses the first element of this control sequence to apply to the system in the next step.

Figure 3.1 shows the general MPC scheme at time k. N_p and N_c symbols denote the control horizon and prediction horizon, respectively. To clarify the terms, prediction horizon (N_p) can be described as the number of future samples for prediction of plant output and control horizon (N_c) is the number of samples within the prediction horizon that are used to capture the control action. Note that, N_c can be less than or equal to N_p. Since the lengths of these parameters directly affect the complexity of the problem and system performance, the number of samples should be selected to provide the optimal solution.

The key elements of the MPC algorithm can be summarized in 4 headings:

(i) Model of the controlled system to predict the future outputs.

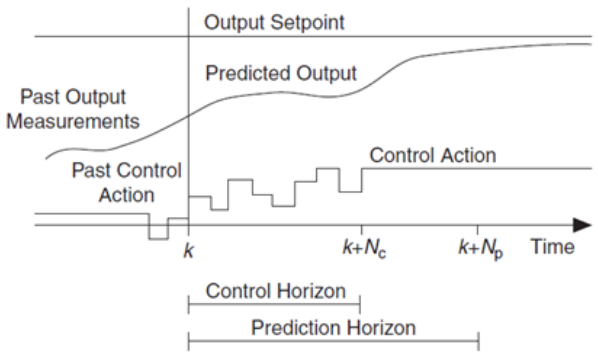

Figure 3.1. General MPC methodology [10].

(ii) Prediction of Outputs.

(iii) Cost function.

(iv) Constraints.

3.1. Model

As the MPC approach is based on the prediction of the future outputs of the plant, a state space model of the plant is required to generate the desired control signal that allows the system to track the desired trajectory. To develop an MPC algorithm, the steps explained in the MPC Toolbox of Matlab are followed. All formula and derivations in this section can be found in [11].

3.1.1. Plant Model

A discrete time state space model of the system used in MPC calculations is given by:

$$\begin{aligned} x_p(k+1) &= A_p x_p(k) + B_p u_p(k) \\ y_p(k) &= C_p x_p(k) + D_p u_p(k) \end{aligned} \tag{3.1}$$

where A_p, B_p,C_p,D_p matrices are the model matrices obtained by System Identification; x_p and u_p are the system's states and inputs vectors. Since the model includes the measured disturbance inputs, (3.1) is rearranged by separating the B_p matrix columns into B_{pu} that corresponds to manipulated inputs $u(k)$ and B_{pv} that corresponds to measured disturbance inputs $v(k)$. The following form is obtained:

$$\begin{aligned} x_p(k+1) &= A_p x_p(k) + B_{pu} u(k) + B_{pv} v(k) \\ y_p(k) &= C_p x_p(k) + D_p u_p(k) \end{aligned} \tag{3.2}$$

3.1.2. Disturbance Models

MPC solves the optimization problem based on the identified plant model. Therefore, the accuracy of the plant model directly affects the calculation of the desired inputs. In order to account for the model-plant mismatch errors and to guarantee steady-state offset-free reference tracking, an output disturbance model is directly added to plant model outputs. An integrator driven by white noise w_{od} with zero mean and unit variance is added for each measured output:

$$\begin{aligned} x_{od}(k+1) &= A_{od}x_{od}(k) + B_{od}w_{od}(k) \\ y_{od}(k) &= C_{od}x_{od}(k) + D_{od}w_{od}(k) \end{aligned} \tag{3.3}$$

where A_{od}, B_{od}, C_{od}, and D_{od} are constant state space matrices, x_{od} is the vector of output disturbance states, and y_{od} is the vector of output disturbances to be added to plant outputs.

Beside the output disturbance model (3.3), to cope with the undesired measurement noise on the measured outputs, a measurement noise model is added as:

$$\begin{aligned} x_n(k+1) &= A_n x_n(k) + B_n w_n(k) \\ y_n(k) &= C_n x_n(k) + D_n w_n(k) \end{aligned} \tag{3.4}$$

where w_n is white noise with zero mean and unit variance, x_n is the vector of noise model states, and y_n is the vector of noise signals to be added to measured plant outputs.

The overall augmented system model to use for output prediction can be combined as:

$$\begin{aligned} x_c(k+1) &= Ax_c(k) + Bu_0(k) \\ y_m(k) &= Cx_c(k) + Du_0(k) \end{aligned} \tag{3.5}$$

where the augmented form of system states, input vector, and system matrices are given by:

$$x_c(k) = [x_p^T(k) \quad x_{od}^T(k) \quad x_n^T(k)]^T,$$
$$u_0(k) = [u^T(k) \quad v^T(k) \quad w_{od}^T(k) \quad w_n^T(k)]^T,$$

$$A = \begin{bmatrix} A_p & 0 & 0 \\ 0 & A_{od} & 0 \\ 0 & 0 & A_n \end{bmatrix}, \quad B = \begin{bmatrix} B_{pu} & B_{pv} & 0 & 0 \\ 0 & 0 & B_{od} & 0 \\ 0 & 0 & 0 & B_n \end{bmatrix},$$
$$C = \begin{bmatrix} C_p & C_{od} & C_n \end{bmatrix}, \quad D = \begin{bmatrix} D_p & D_{od} & D_n \end{bmatrix}.$$

It is important to note that, controllability of the augmented model should be checked to achieve closed-loop control performance as discussed in [12].

3.2. Prediction of Outputs

3.2.1. State Estimation

In general, when system identification-based model derivation is performed, system states are not related to a physical variable and are not measurable. To predict the future outputs based on the system model, the first step is the estimation of these unknown states. For a successful state estimation, observability of the augmented system matrices (A, C) is a pre-condition. The necessary and sufficient condition for the observability is that the observability matrix must have full rank.

Kalman filter is a popular state observer used in most control applications. It is a technique to estimate the unknown states based on a state space model, measured data, and noise covariance data. The basic diagram of the Kalman State Estimation is below:

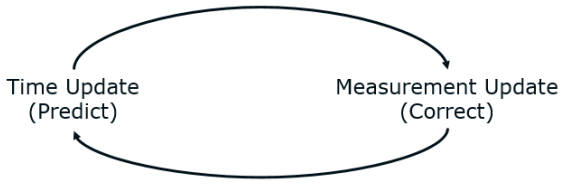

Figure 3.2. Kalman State Estimation Cycle [13].

For a state space system with white noise process and measurement disturbances, state estimation process via Kalman filter is as follows:

(i) Innovation variable $e(k)$ is calculated based on the current measured plant output $y_m(k)$ and the state estimate calculated from previous step $x_c(k|k-1)$ as:

$$e(k) = y_m(k) - Cx_c(k|k-1)$$

(ii) A more accurate estimate of the state variable is calculated by using the new measured information. This new state is:

$$x_c(k|k) = x_c(k|k-1) + Me(k)$$

(iii) After optimum input is calculated, future state estimate is computed as follows in order to use in the next control interval:

$$x_c(k+1|k) = Ax_c(k|k-1) + B_u u_{opt}(k) + B_v v(k) + Le(k)$$

where B_u and B_v are the columns of B corresponding to u and v vectors, respectively.

The second step above is referred to as 'Measurement Update' phase in the Kalman State Estimation Process where the third step is called the 'Time Update' phase. In here, M and L are the Kalman innovation and estimator gain matrices that

are calculated by using the *kalman* function in Matlab. The inputs to the function are the observer model (3.5) and noise covariance matrices that are calculated as below:

$$Q = BB^T, \quad R = DD^T, \quad N = BD^T$$

The steady-state Kalman filter gain L and innovation gain M are calculated as:

$$L = PC^T R^{-1}, \quad M = PC^T(CPC^T + R)^{-1}$$

to minimize the steady-state error covariance and the covariance matrix P is the solution to the Algebraic Riccati Equation:

$$AP + PA^T + BQB^T - PC^T R^{-1} CP = 0$$

More detailed information can be obtained from [13] and [14].

3.2.2. Output Prediction

Prediction of future states at time k and within the optimization window N_p are calculated by:

$$\begin{aligned}
&x_c(k+1|k) = Ax_c(k|k) + B_u u(k|k) + B_v v(k) \\
&x_c(k+2|k) = Ax_c(k+1|k) + B_u u(k+1|k) + B_v(k+1|k) \\
&\cdots \\
&x_c(k+N_p|k) = Ax_c(k+N_p|k) + B_u u(k+N_p|k) + B_v v(k+N_p|k)
\end{aligned}$$

and future predicted outputs at time k and within the optimization window N_p are:

$$\begin{aligned}
y(k+1|k) &= Cx_c(k+1|k) + D_v v(k+1|k) \\
y(k+2|k) &= Cx_c(k+2|k) + D_v v(k+2|k) \\
&\cdots \\
y(k+N_p|k) &= Cx_c(k+N_p|k) + D_v v(k+N_p|k)
\end{aligned}$$

Denote the columns of D matrix corresponding to manipulated inputs u and measured disturbances v in the overall output model (3.5) as D_u and D_v, respectively. Although there is a D_u term, no direct feedthrough is assumed from u to y since the current plant output is required for both prediction and control. Therefore, D_u term is assumed as zero matrix. On the other hand, specifically on the diesel engine model identified in this study, D_v term is found as zero matrix; therefore D term will not be used in the remaining calculations. However, it will be shown in equations. To simplify the calculations in the next sections, Δu term is introduced as input change rate:

$$\Delta u(k) = u(k) - u(k-1)$$

After defining the vectors:

$$\begin{aligned}
Y &= [y(k+1|k) \quad y(k+2|k) \quad y(k+3|k) \quad \dots \quad y(k+N_p|k)]^T \\
\Delta U &= [\Delta u(k) \quad \Delta u(k+1) \quad \Delta u(k+2) \quad \dots \quad \Delta u(k+N_c-1)]^T \\
V &= [v(k|k) + v(k+1|k) + \quad \dots \quad v(k+N_p|k)]^T
\end{aligned}$$

and putting together the predicted state equations, the output vector is given by:

$$Y = S_x x_c(k|k) + S_{u1} u(k-1) + S_u \Delta U + H_v V \tag{3.6}$$

where

$$S_x = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix}; \quad S_{u1} = \begin{bmatrix} CB_u \\ CB_u + CAB_u \\ \vdots \\ \sum_{h=0}^{N_p-1} CA^h B_u \end{bmatrix};$$

$$S_u = \begin{bmatrix} CB_u & 0 & \dots & 0 \\ CB_u + CAB_u & CB_u & \dots & 0 \\ \vdots & & & \\ \sum_{h=0}^{N_p-1} CA^h B_u & \sum_{h=0}^{N_p-2} CA^h B_u & \dots & \sum_{h=0}^{N_p-N_c} CA^h B_u \end{bmatrix};$$

$$H_v = \begin{bmatrix} CB_v & D_v & 0 & \dots & 0 \\ CAB_v & CB_v & D_v & \dots & 0 \\ \vdots & & & & \\ CA^{N_p-1}B_v & CA^{N_p-2}B_v & CA^{N_p-3}B_v & \dots & D_v \end{bmatrix}$$

3.3. Optimization Problem

The general purpose of the MPC control design is to obtain a predicted output that follows the desired trajectory within a prediction horizon. To find such a control input that provides the goal, generally a quadratic cost function (J) and constraints are defined. A general form of a QP is as follows:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} z^T \bar{H} z + \bar{f}^T z \\ subject \quad to \quad & A_{in} z \leq b_{in} \\ & A_{eq} z = b_{eq} \end{aligned} \tag{3.7}$$

Then, Quadratic Optimization Problem (QP) is solved by well known QP solvers [15].

3.3.1. Cost Function

The general cost function used in MPC applications contains 4 cost functions that are namely output reference tracking cost (J_y), manipulated variable reference tracking cost (J_u), manipulated variable change cost ($J_{\Delta u}$), and a slack variable cost (J_ϵ) which is used to quantify the soft constraint violations. Each cost function contains weighting factor within itself for prioritization. The overall cost function is:

$$J(z) = J_y(z) + J_u(z) + J_{\Delta u}(z) + J_\epsilon(z)$$

In this study, manipulated variables do not need to track a reference value, but they should remain in a desired range during the control process. Instead of including $J_{\Delta u}$ in the cost function, it is added to the constraint part of the optimization problem.

The detailed form of cost functions are given below:

$$J_y = (Y - Y_{ref})^T Q (Y - Y_{ref}) \tag{3.8}$$

$$J_{\Delta u} = \Delta U^T R \Delta U \tag{3.9}$$

$$J_\epsilon = \rho_\epsilon \epsilon^2 \tag{3.10}$$

where Y_{ref} is the desired trajectory vector, Q is the output reference tracking weighting value, R is the weight matrix to pay attention to the size of ΔU, and ρ_ϵ is the constraint violation penalty.

To form an optimization problem similar to (3.7), firstly, the cost function (3.8) is rearranged. By substituting (3.6) into (3.8), the following equation is obtained:

$$\begin{aligned} J_u &= (S_u \Delta U + c_y)^T Q (S_u \Delta U + c_y) \\ &= \Delta U^T S_u^T Q S_u \Delta U + 2 c_y^T Q S_u \Delta U + c_y^T Q C_y \\ &= \Delta U^T S_a \Delta U + 2 c_y^T Q S_u \Delta U + c_y^T Q C_y \end{aligned} \tag{3.11}$$

where

$$c_y = S_x x_c(k|k) + S_{u1} u(k-1) + H_v V - Y_{ref}, \quad S_a = S_u^T Q S_u$$

Due to c_y term depending only on prediction state, previous step control input, measured disturbances, and output reference values, it does not affect the solution of the optimization problem; hence $c_y^T Q C_y$ part is ignored in the cost function. By putting together all cost function terms together, we obtain

$$\begin{aligned} J &= \Delta U^T S_a \Delta U + 2 c_y^T Q S_u \Delta U + \Delta U^T R \Delta U + \rho_\epsilon \epsilon^2 \\ &= \Delta U^T \underbrace{(S_a + R)}_{\mathrm{H}} \Delta U + \underbrace{2 c_y^T Q S_u}_{f^T} \Delta U + \rho_\epsilon \epsilon^2 \end{aligned} \tag{3.12}$$

Let the QP decision variable z be

$$z = [\Delta U \quad \epsilon]^T$$

Then we can rewrite the cost function as follows:

$$\begin{aligned} J &= \frac{1}{2} \begin{bmatrix} \Delta U \\ \epsilon \end{bmatrix}^T \underbrace{\begin{bmatrix} 2H & 0 \\ 0 & 2\rho_\epsilon \end{bmatrix}}_{\bar{H}} \begin{bmatrix} \Delta U \\ \epsilon \end{bmatrix} + \underbrace{\begin{bmatrix} f \\ 0 \end{bmatrix}^T}_{\bar{f}^T} \begin{bmatrix} \Delta U \\ \epsilon \end{bmatrix} \\ &= \frac{1}{2} z^T \bar{H} z + \bar{f}^T z \end{aligned} \tag{3.13}$$

3.3.2. Constraints

Another important part of the MPC design is to take into account the constraints that are generally imposed on the outputs, inputs, and rate of change of the inputs.

3.3.2.1. Output Constraints. The output constraints inequalities are written with the minimum and maximum output limits y_{min} and y_{max} and the constraint softness values v^y_{min} and v^y_{max} as follows:

$$\underbrace{\begin{bmatrix} y_{min}(k+1) \\ \vdots \\ y_{min}(k+N_p) \end{bmatrix}}_{Y_{min}} - \epsilon \underbrace{\begin{bmatrix} v^y_{min}(k+1) \\ \vdots \\ v^y_{min}(k+N_p) \end{bmatrix}}_{V^y_{min}} \leq \underbrace{\begin{bmatrix} y(k+1|k) \\ \vdots \\ y(k+N_p|k) \end{bmatrix}}_{\mathrm{Y}}$$

$$\leq \underbrace{\begin{bmatrix} y_{max}(k+1) \\ \vdots \\ y_{max}(k+N_p) \end{bmatrix}}_{Y_{max}} + \epsilon \underbrace{\begin{bmatrix} v^y_{max}(k+1) \\ \vdots \\ v^y_{max}(k+N_p) \end{bmatrix}}_{V^y_{max}}$$

The constraint softness and output limit values are assumed as constant during the prediction horizon N_p, i.e., we have

$$Y_{min} - \epsilon V^y_{min} \leq Y \leq Y_{max} + \epsilon V^y_{max} \tag{3.14}$$

Equation (3.14) can be transformed into the inequality matrices form as in (3.7):

$$-Y - \epsilon V^y_{min} \leq -Y_{min}$$
$$Y - \epsilon V^y_{max} \leq Y_{max}$$

by putting Y as in (3.6),

$$-S_u \Delta U - \epsilon V^y_{min} \leq -Y_{min} + S_x x_c(k|k) + S_{u1} u(k-1) + H_v V$$
$$S_u \Delta U - \epsilon V^y_{max} \leq Y_{max} - S_x x_c(k|k) - S_{u1} u(k-1) - H_v V$$

$$\underbrace{\begin{bmatrix} -S_u & -V^y_{min} \\ S_u & -V^y_{max} \end{bmatrix}}_{M_1} \underbrace{\begin{bmatrix} \Delta U \\ \epsilon \end{bmatrix}}_{\mathrm{z}} \leq \underbrace{\begin{bmatrix} -Y_{min} \\ Y_{max} \end{bmatrix}}_{N_{1c}} + \underbrace{\begin{bmatrix} S_x \\ -S_x \end{bmatrix}}_{N_{1x}} x_c(k|k) + \underbrace{\begin{bmatrix} S_{u1} \\ -S_{u1} \end{bmatrix}}_{N_{1u}} u(k-1) + \underbrace{\begin{bmatrix} H_v \\ -H_v \end{bmatrix}}_{N_{1v}} V$$

$$M_1 z \leq N_1 \tag{3.15}$$

where $N_1 = N_{1c} + N_{1x}x_c(k|k) + N_{1u}u(k-1) + N_{1v}V$

3.3.2.2. Input Constraints. The same procedure as in 3.3.2.1 is followed to get the input inequality constraints in QP format where u_{min} and u_{max} are the minimum and maximum input limits and v^u_{min} and v^u_{max} are the constraint softening values, i.e., the input constraints are represented as follows:

$$\underbrace{\begin{bmatrix} u_{min}(k) \\ \vdots \\ u_{min}(k+N_c-1) \end{bmatrix}}_{U_{min}} - \epsilon \underbrace{\begin{bmatrix} v^u_{min}(k) \\ \vdots \\ v^u_{min}(k+N_c-1) \end{bmatrix}}_{V^u_{min}} \leq \underbrace{\begin{bmatrix} u(k|k) \\ \vdots \\ u(k+N_c-1|k) \end{bmatrix}}_{\mathrm{U}}$$

$$\leq \underbrace{\begin{bmatrix} u_{max}(k) \\ \vdots \\ u_{max}(k+N_c-1) \end{bmatrix}}_{U_{max}} + \epsilon \underbrace{\begin{bmatrix} v^u_{max}(k) \\ \vdots \\ v^u_{max}(k+N_c-1) \end{bmatrix}}_{V^u_{max}}$$

$$U_{min} - \epsilon V^u_{min} \leq U \leq U_{max} + \epsilon V^u_{max} \tag{3.16}$$

Then, the inequalities in (3.16) are rearranged and the following inequalities are obtained:

$$\begin{aligned} -U - \epsilon V^u_{min} &\leq -U_{min} \\ U - \epsilon V^u_{max} &\leq U_{max} \end{aligned} \tag{3.17}$$

The next step is writing the U matrix in terms of the optimization variable ΔU. The relation between ΔU and U is as follows:

$$\Delta U = \underbrace{\begin{bmatrix} I & 0 & 0 & \dots & 0 & 0 \\ -I & I & 0 & \dots & 0 & 0 \\ 0 & -I & I & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & -I & I \end{bmatrix}}_{T_1} U - \underbrace{\begin{bmatrix} I \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{T_2} u(k-1)$$

Then, U can be written as:

$$U = T_1^{-1}\Delta U + T_1^{-1}T_2 u(k-1)$$

After substituting U into (3.17), the input constraint matrices are transformed into the inequality matrices' form as in (3.7) by the following equations:

$$-T_1^{-1}\Delta U - \epsilon V_{min}^u \leq -U_{min} + T_1^{-1}T_2 u(k-1)$$
$$T_1^{-1}\Delta U - \epsilon V_{max}^u \leq U_{max} - T_1^{-1}T_2 u(k-1)$$

$$\underbrace{\begin{bmatrix} -T_1^{-1} & -V_{min}^u \\ T_1^{-1} & -V_{max}^u \end{bmatrix}}_{M_2} \underbrace{\begin{bmatrix} \Delta U \\ \epsilon \end{bmatrix}}_{\text{z}} \leq \underbrace{\begin{bmatrix} -U_{min} \\ U_{max} \end{bmatrix}}_{N_{2c}} + \underbrace{\begin{bmatrix} T_1^{-1}T_2 \\ -T_1^{-1}T_2 \end{bmatrix}}_{N_{2u}} u(k-1)$$

$$M_2 z \leq N_2 \tag{3.18}$$

where $N_2 = N_{2c} + N_{2u}u(k-1)$

3.3.2.3. Input Rate Constraints. It is also possible to limit the input rate of change with the minimum and maximum Δu_{min} and Δu_{max} values. As in the previous steps,

$v_{min}^{\Delta u}$ and $v_{max}^{\Delta u}$ are the constraint softening values and constraint inequalities can be written as follows:

$$\underbrace{\begin{bmatrix} \Delta u_{min}(k) \\ \vdots \\ \Delta u_{min}(k+N_c-1) \end{bmatrix}}_{\Delta U_{min}} - \epsilon \underbrace{\begin{bmatrix} v_{min}^{\Delta u}(k) \\ \vdots \\ v_{min}^{\Delta u}(k+N_c-1) \end{bmatrix}}_{V_{min}^{\Delta u}} \leq \underbrace{\begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_c-1|k) \end{bmatrix}}_{\Delta U}$$

$$\leq \underbrace{\begin{bmatrix} \Delta u_{max}(k) \\ \vdots \\ \Delta u_{max}(k+N_c-1) \end{bmatrix}}_{\Delta U_{max}} + \epsilon \underbrace{\begin{bmatrix} v_{max}^{\Delta u}(k) \\ \vdots \\ v_{max}^{\Delta u}(k+N_c-1) \end{bmatrix}}_{V_{max}^{\Delta u}}$$

$$\Delta U_{min} - \epsilon V_{min}^{\Delta u} \leq \Delta U \leq \Delta U_{max} + \epsilon V_{max}^{\Delta u} \tag{3.19}$$

Next, (3.19) is rearranged as follows:

$$-\Delta U - \epsilon V_{min}^{\Delta u} \leq -\Delta U_{min}$$
$$\Delta U - \epsilon V_{max}^{\Delta u} \leq \Delta U_{max}$$

$$\underbrace{\begin{bmatrix} -I & -V_{min}^{\Delta u} \\ I & -V_{max}^{\Delta u} \end{bmatrix}}_{M_3} \underbrace{\begin{bmatrix} \Delta U \\ \epsilon \end{bmatrix}}_{\text{z}} \leq \underbrace{\begin{bmatrix} -\Delta U_{min} \\ \Delta U_{max} \end{bmatrix}}_{N_3}$$

$$M_3 z \leq N_3 \tag{3.20}$$

As a last step, an extra inequality constraint is added to keep the slack variable ϵ higher than 0:

$$\underbrace{\begin{bmatrix} 0 & -1 \end{bmatrix}}_{M_4} \underbrace{\begin{bmatrix} \Delta U \\ \epsilon \end{bmatrix}}_{\mathrm{z}} \leq \underbrace{0}_{N_4}$$

$$M_4 z \leq N_4 \tag{3.21}$$

Then, all inequality constraints (3.15), (3.17), (3.20), and (3.21) are put together to obtain overall inequality matrices A_{in} and B_{in}:

$$A_{in} = \begin{bmatrix} M_1 & M_2 & M_3 & M_4 \end{bmatrix}^T$$
$$B_{in} = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix}^T$$

In summary, all constraints are arranged in the form of (3.7) so that QP solver can be used.

3.4. Quadratic Problem Solver

For solving a QP (3.7), there are commonly used methods in the literature namely Active Set Methods [16] and Interior Point Methods [17]. QP solution methods are not detailed in this study. Matlab *mpcqpsolver* function is used to calculate the optimum z variable in each time step. The first element of z corresponds to rate of change of inputs in the first prediction step $\Delta u(k|k)$. Then, the optimum control signal $u(k|k)$ that is applied to the system is obtained by adding the $z(1)$ to previous control input as follows:

$$u(k|k) = u(k-1) + z(1) \tag{3.22}$$

where $z(1) = \Delta u(k|k)$.

3.5. Summary of the Chapter

In this chapter, the Model Predictive Control procedure for linear state-space models has been explained. Model extension technique to deal with the model-plant mismatch and state estimation technique to predict the unknown states have been detailed. Subsequently, quadratic optimization problem with constraints has been established. In the following chapter, the MPC technique is applied to the MIMO DEAP problem.

4. SYSTEM IDENTIFICATION AND MPC OF DEAP

To control a nonlinear system, defining multiple linear models for different operation regions and designing MPC for each region, and then switching the controllers depending on the operation region is a common method. This methodology is known as Gain-Scheduled MPC.

In this chapter, system identification process of DEAP model is described. Also, Gain-Scheduled MPC simulation results are presented. All simulations are performed in the Matlab/Simulink environment.

4.1. System Identification of DEAP

Diesel engine air path is a system with highly nonlinear dynamics. However, on the bright side, a nonlinear system can be approximated as a linear system around a specific operating point.

System identification proceeds mainly in 4 stages [18]:

- Experiment Design and Data Collection.
- Model Selection
- Parameter Estimation
- Validation.

Experiment Design part is an essential identification step to obtain a best possible excitation of the system. Input/output data should be maximally informative to catch the important system dynamics. Gaussian White Noise, Random Binary Sequence (RBS), Pseudo Random Binary Sequence (PRBS), Multisine inputs are the commonly used excitation signals in literature.

The second step of the identification is choosing an appropriate model structure. State-space models are good candidates to represent MIMO systems and have simpler structure for model based control design. When there is no physical interpretation consideration of the system (black-box modeling), a linear state-space model with a suitable order (order is the number of states in the state space model case) can be viewed as a way to achieve input/output data fit. A general discrete time state space model structure is given by:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{4.1}$$

where A, B, C, D are the system matrices to be identified, $x(k)$ is the system state, $u(k)$ is the system input, and $y(k)$ is the output. Note that, when the black-box modeling is performed, system states do not have to correspond to physical states of the system directly. Therefore, state estimation process is necessary.

Prediction Error Method (PEM) and Subspace-Based State-Space Identification (4SID) are two well-known techniques for state-space model estimation. The basic idea behind the PEM method is to minimize the error between the measured output data and the predicted output of the model. On the other hand, 4SID methods are developed based on the information about the column space of the extended observability matrix or row space of state sequence matrix of a state-space system to identify the parameters of the system. For more information about the PEM and 4SID, [18] and [19] can be checked. Morever, N4SID method is a popular subspace identification algorithm that uses the estimates of state sequence matrix to find to A, B, C and D matrices of the model [20], [21]. Matlab System Identification Toolbox allows for using PEM and N4SID techniques just by defining the input-output data and model structure.

There are several approaches to validate the estimated model. Plotting the estimated model output and measured output provides inside into how estimated model imitates the plant. A validation data set different from estimation data can be designated to avoid over-fitting of the model. This process is called cross-validation. Corre-

lation analysis on the prediction errors can also be performed to check the validity of the estimated model.

In this study, AVL Boost Rt diesel engine air path model is used as the plant model. The plant has 4 inputs namely engine speed, fuel rate, VGT position, EGR position and 2 outputs MAF and MAP. Engine speed and fuel rate are the measured disturbances of the system whereas VGT and EGR positions are the manipulated variables that are run by the controller. Operation region of the engine is divided into sub-zones according to different engine speed and fuel rate in order to get sufficiently precise local linear models.

To predict the linear model around 2000 rpm engine speed and 8 mg/st fuel operating points (called as region 1), the system was excited with the signals shown in Figure 4.1. Random noise signals were superposed to operating points. On the other hand, PRBS signals were chosen for the manipulated variables VGT and EGR position to catch the system dynamics. Figure 4.2 shows the output signals of the plant that correspond to inputs in Figure 4.1.

By using Matlab System Identification toolbox, PEM and N4SID identification methods were tested with the state space models of order from 2 to 8. After the trials with validation data different from estimation data, a 4th order state space model derived with the N4SID method performed the best fit to the system output. The fitness value is calculated by the fit formula based on Normalized Root Mean Square Error (NRMSE) between estimation data y and model output $\hat{y}$ as shown in (4.2)

$$fit = 100\left(1 - \frac{||y - \hat{y}||}{||y - mean(y)||}\right) \tag{4.2}$$

where $||.||$ indicates the 2-norm. The calculated plant model equation is the following:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + Ke(k) \\ y(k) &= Cx(k) + Du(k) + e(k) \end{aligned} \tag{4.3}$$

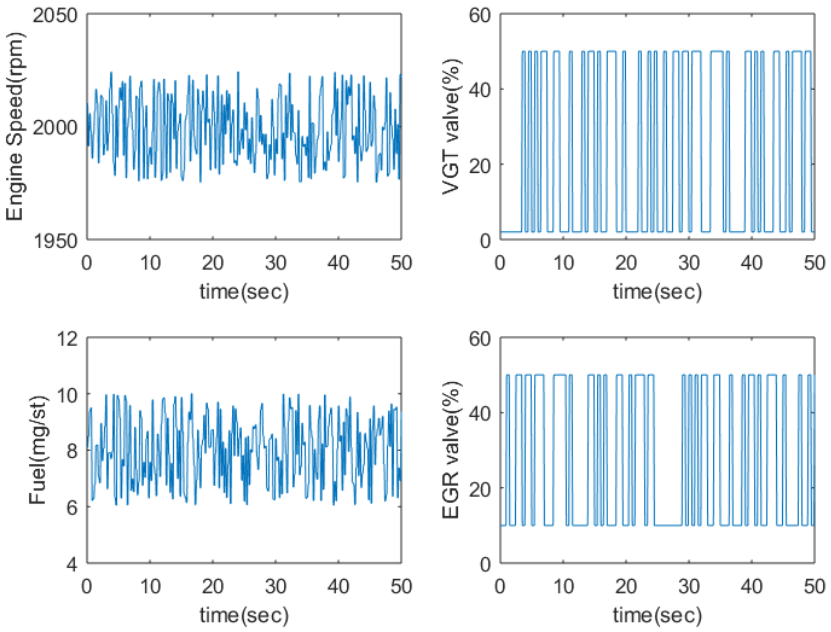

Figure 4.1. Identification input signals for region 1.

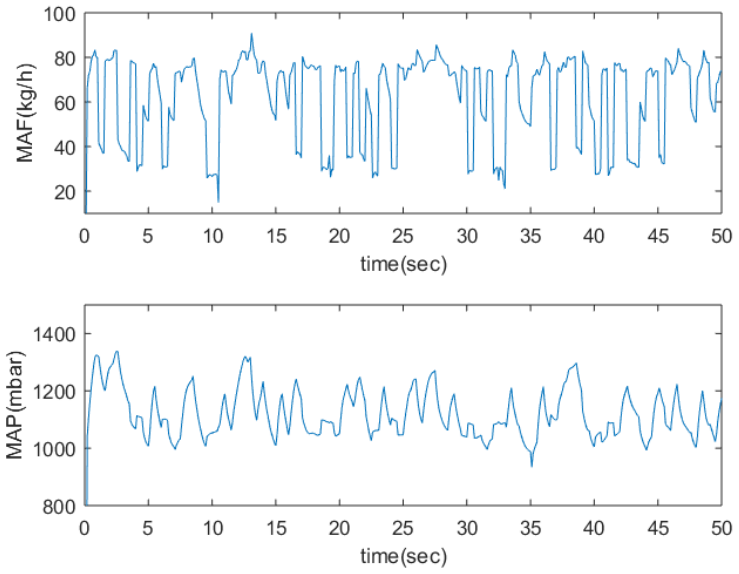

Figure 4.2. Identification output signals for region 1.

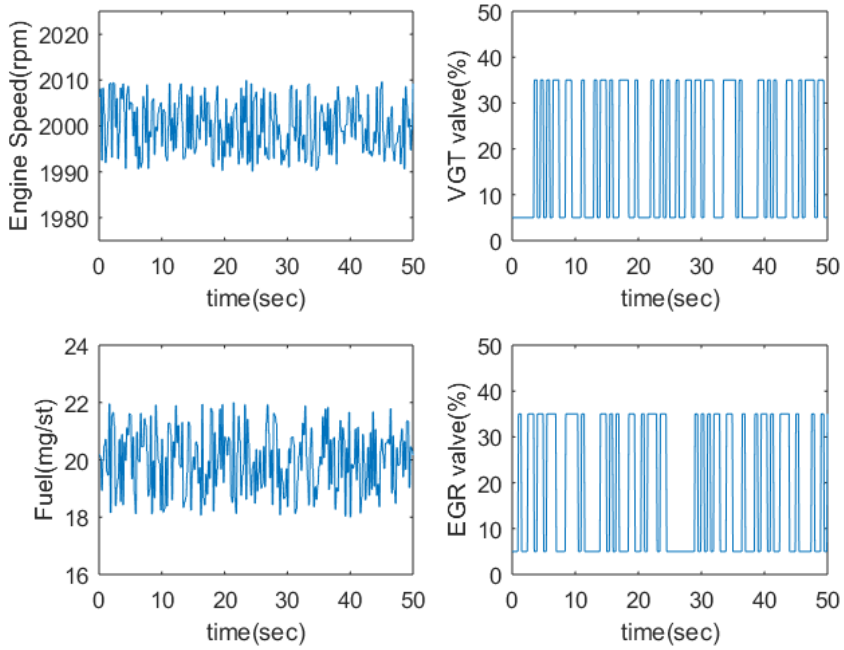

Figure 4.3. Identification input signals for region 2.

where A and B are 4×4, C and D are 2×4 and K is 4×2 system matrices, $x(k)$ is the system state at time k, $u(k)$ is the vector of system inputs consisting of engine speed, fuel, VGT position, and EGR position and $y(k)$ is the vector of outputs MAF and MAP. Discretization step size is 0.1 second. The model outputs MAF and MAP fit to the estimation data 77.83% and 93.69% respectively.

A second operation region is chosen in this study for gain-scheduling control purpose. While keeping the engine speed around 2000 rpm, 20 mg/st fuel region is taken as the second operation region that represents the high fuel zone and results more boost pressure than region 1. The excitation input signals and corresponding outputs are shown in Figures 4.3 and 4.4, respectively. Moreover, a 4th order linear state space model as in (4.3) is obtained with the N4SID method with the fitness values 75.66% for MAF and 93.19% for MAP. System matrices obtained by System Identification for operation region 1 and operation 2 are given in Appendix.

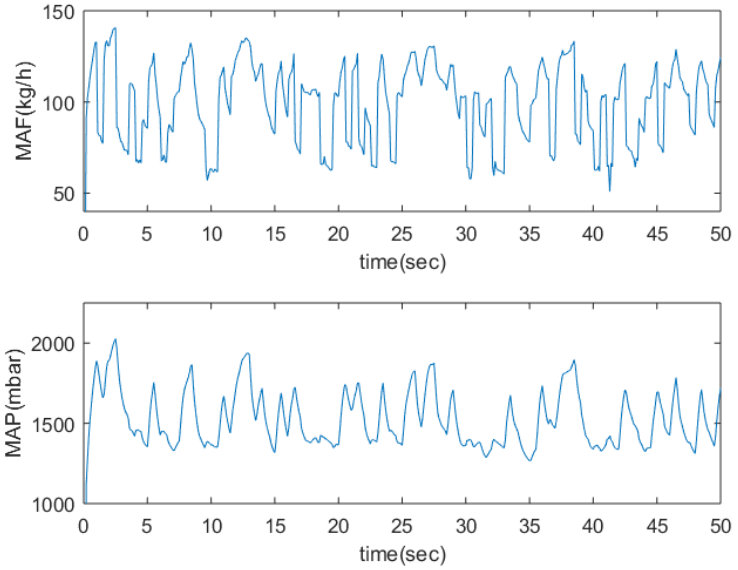

Figure 4.4. Identification output signals for region 2.

4.2. MPC Design Parameters

The main design parameters of the MPC controller are listed below:

(i) *Prediction and Control Horizons:* Prediction horizon (N_p) is chosen as 15 samples while control horizon (N_c) is setting 3.

(ii) *Constraint Parameters:* Input, output, and input change rate constraints are determined according to real engine operation limits.

- Output constraints: The physical unit of MAF is kg/h and MAP is in terms of mbar.

$$\begin{bmatrix} 40 \\ 250 \end{bmatrix} \leq \begin{bmatrix} y_{maf} \\ y_{map} \end{bmatrix} \leq \begin{bmatrix} 1000 \\ 2500 \end{bmatrix}$$

- Input constraints: Actuator limits are the input constraints.

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} u_{vgt} \\ u_{egr} \end{bmatrix} \leq \begin{bmatrix} 60 \\ 50 \end{bmatrix}$$

- Input rate constraints: This constraint limits are related to movement speed of the actuator in terms of [%/ms].

$$\begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \leq \begin{bmatrix} \delta u_{vgt} \\ \delta u_{egr} \end{bmatrix} \leq \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

In addition to the constraint limits given above, the slack variable described in Section 3.3.2 is included in the constraint equations with constraint softness values. Maximum and minimum constraint softness values are set as 0 for input and input rate to prevent violations while they are set to 1 for output constraints.

(iii) *Weighting Matrices:* The weighting factors mentioned in Section 3.3.1 are tuned to give the same priority for the reference tracking performance of both outputs MAP and MAF considering their dimensions. Another important point in the multiobjective cost tuning is that the slack variable weight should be too high compared to others to avoid constrained violations. In simulations, the following weighting matrices are used:

$$Q_{maf} = 15, \quad Q_{map} = 1, \quad R_{vgt} = 15, \quad R_{egr} = 1.5, \quad \rho_\epsilon = 10^8$$

where Q_{maf} and Q_{map} are the reference tracking weighting values of MAF and MAP outputs; R_{vgt} and R_{egr} are the weighting values to pay attention to the sizes of change of VGT input and change of EGR input, respectively; and ρ_ϵ is the constraint violation penalty.

(iv) *Disturbance Model Parameters (Integral action):* To eliminate model-plant mismatch and steady-state error, output disturbance model is added to identified system model as mentioned in Section 3.1.2. The model parameters are tuned as follows:

$$A_{od} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B_{od} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.7 \end{bmatrix},$$
$$C_{od} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D_{od} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

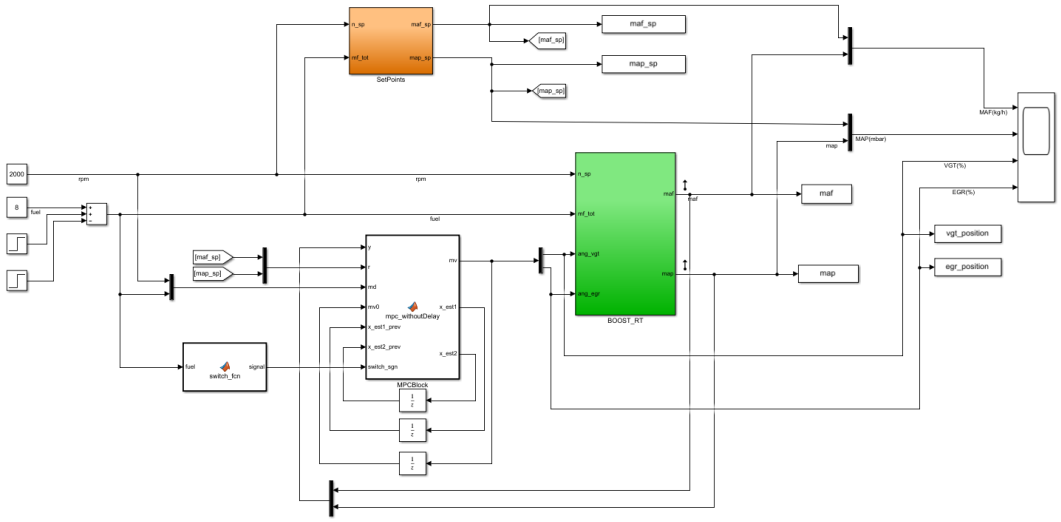

Figure 4.5. Simulink Diagram of Input Delay Free MPC System.

(v) *State Estimator Gain:* State estimator gain matrices are calculated using 'kalman' function of Matlab.

(vi) *Sample Time (T_s):* Sample time of the simulations is 10ms.

4.3. MPC of DEAP without Actuator Delay Model

Actuator time delay (or input delay) is a general problem to be considered in control phase for most of the real-time systems. In this study, first of all, the plant model BOOST-RT is designed as input delay free which takes into account the dynamic relations between the injected fuel, engine rpm, VGT position, EGR position, and MAF and MAP outputs. Then, system identification process is applied as described in Section 4.1. Lastly, linear MPC is computed for the delay free model.

Figure 4.5 depicts the block diagram of the delay free control system. Switch function is used to control the model switching when the operation points is changed. MPCBlock calculates the VGT and EGR inputs described in Chapter 3 and with the design parameters in Chapter 4.2. The codes written in the control block are given in the Appendix.

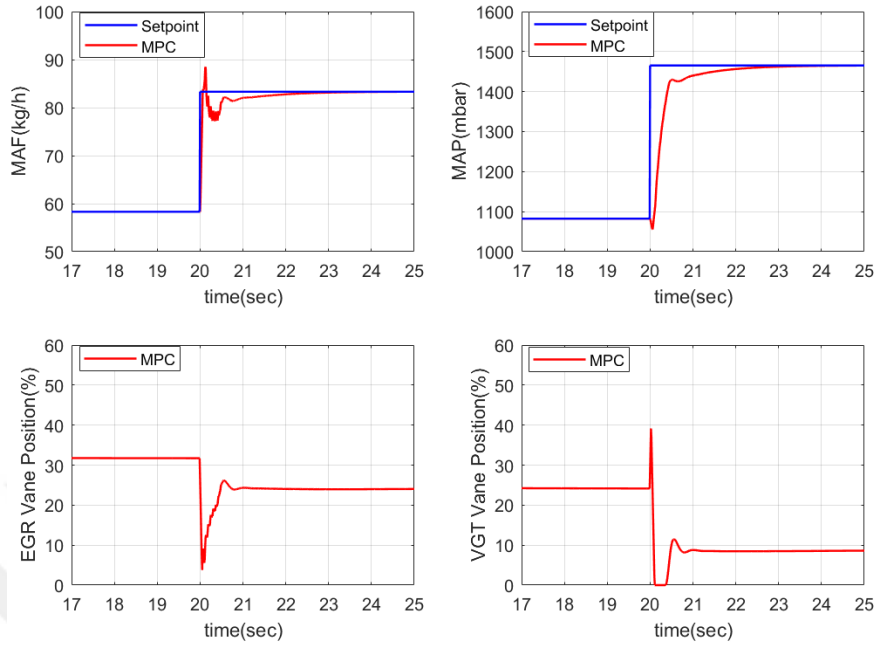

Figure 4.6. Simulation Results of MPC on Input Delay Free Model - Rising Step Setpoint Change.

In order to check the control performance, two cases are examined on the system that are setpoint increase case and setpoint decrease case. 2 operation points are chosen and the system is linearized at these points for the MPC methodology as described in Section 4.1. Simulation results of MPC of DEAP are given in Figures 4.6 and 4.7 with MAF and MAP output values, corresponding setpoints, and EGR and VGT control inputs. In Figure 4.6, operation point is changed from region-1 to region-2 whereas in Figure 4.7, it comes back to former state region-1.

From Figures 4.6 and 4.7 it is noted that MPC exhibits good performance in terms of settling time and steady-state error. It satisfies the expected settling times which are observed from the open loop response of the system as approximately 1.5 sec for MAF and 2 sec for MAP. Additionally, MPC obeys the actuator position change rate constraints even when there is a sudden change in setpoint.

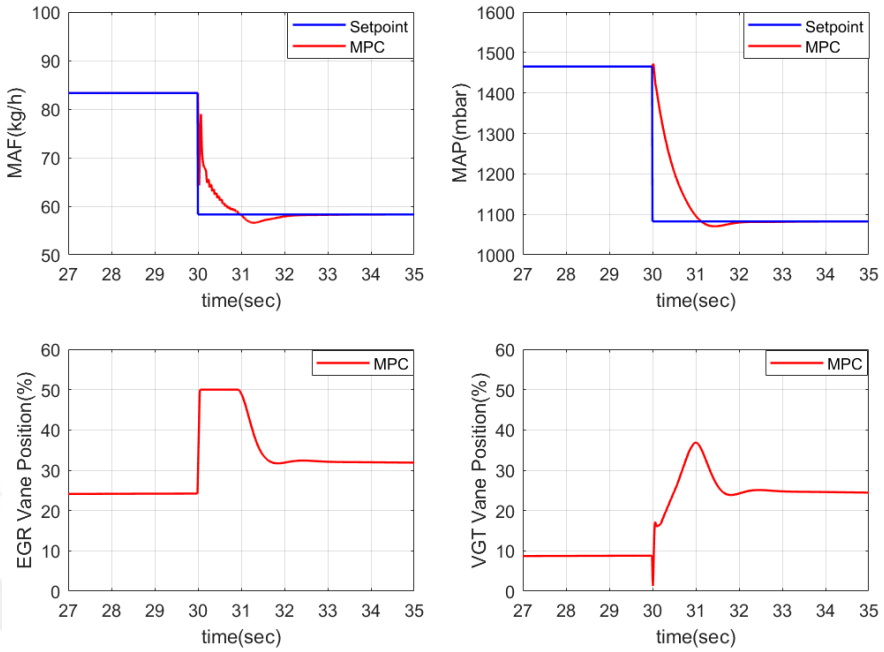

Figure 4.7. Simulation Results of MPC on Input Delay Free Model - Falling Step Setpoint Change.

4.4. Summary of the Chapter

In this chapter, system identification has been performed on AVL Boost RT DEAP model. Further, MPC technique has been applied to control the model. It is shown that MPC has a good control performance in terms of rise time, overshoot, and zero steady-state error when the operation region and set points of the system are changed. The identified model and MPC codes derived in this chapter will form the basis of the MPC of the actuator-delayed MIMO system that will be studied in the next chapter.

5. MPC OF DEAP WITH ACTUATOR DELAY MODEL

Actuator time delay is a physical limitation for the control of real-time systems. In most of control applications, although time delays are ignored for simplicity, it deteriorates the control performance of the system. In this chapter, actuator time delays with dead time are explicitly embedded to the VGT and EGR inputs in the model. 70 ms is experimentally estimated dead time of the actuators. Therefore, delay length is set to 7 samples since the model step time T_s is 10 ms. The new control diagram is shown in Figure 5.1.

5.1. Control of Input Delayed System without Delay Consideration in MPC

In this section of the study, system response with MPC is examined when the actuator dead time is added to the system.

Simulation results are given in Figure 5.2. Since MPC makes the predictions based on delay-free model, the performance of the controller is deteriorated and thus making the system oscillate, especially in operation region-2.

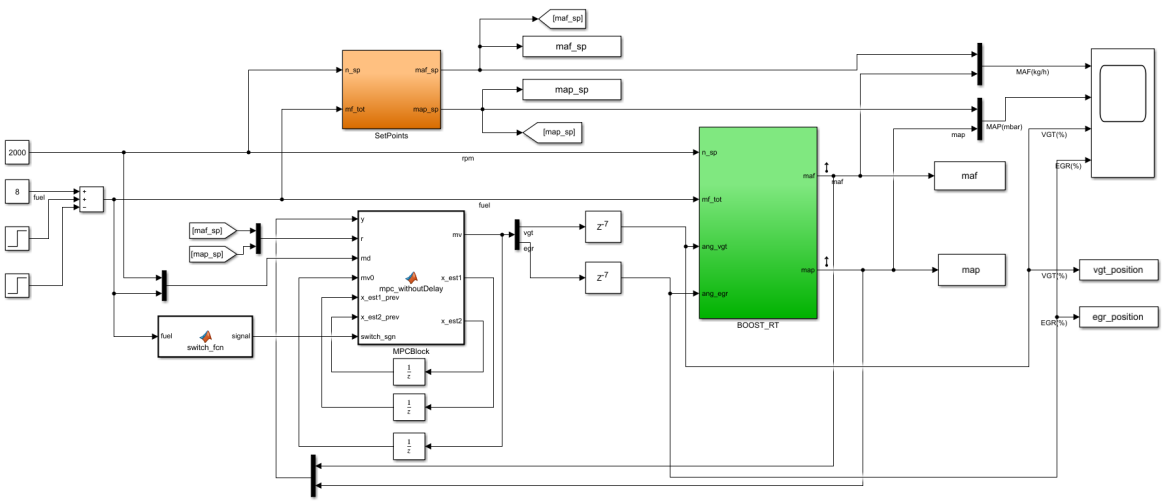

Figure 5.1. Simulink Diagram of the Delayed System.

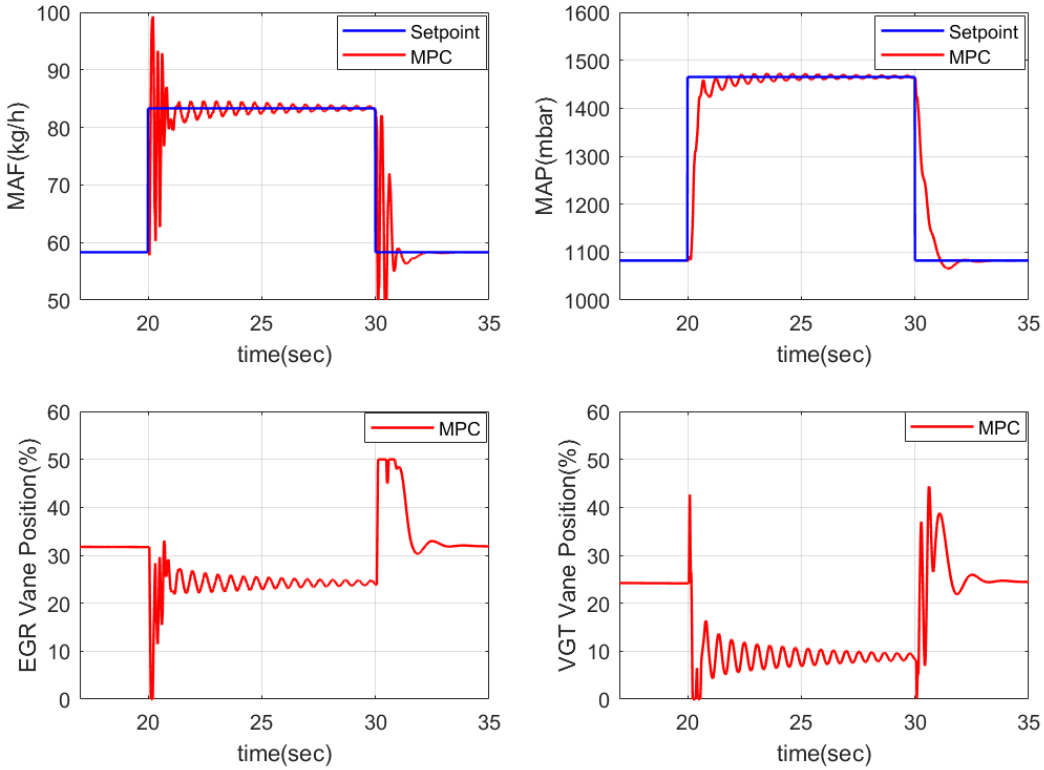

Figure 5.2. Simulation Results of MPC Input Delayed System.

5.2. Control of Input Delayed System with Delay Consideration in MPC

Model predictive control (MPC) is well known control methodology for its ability to deal with a system with time delay [22]. The presence of time delay in a system causes state dimension expansion, which does not represent the system dynamics. Therefore, observer is employed to estimate the state of the system while the delay is embedded to the input of the plant. Augmentation of the plant model described in (3.2) is shown in Figure 5.3 and delay model is implemented to the state-space equations by the following steps:

(i) Write u_{p_new} in terms of u_p,

$$u(k) = u_{p\_new}(k - N) \longrightarrow u_{p\_new}(k) = u(k + N)$$

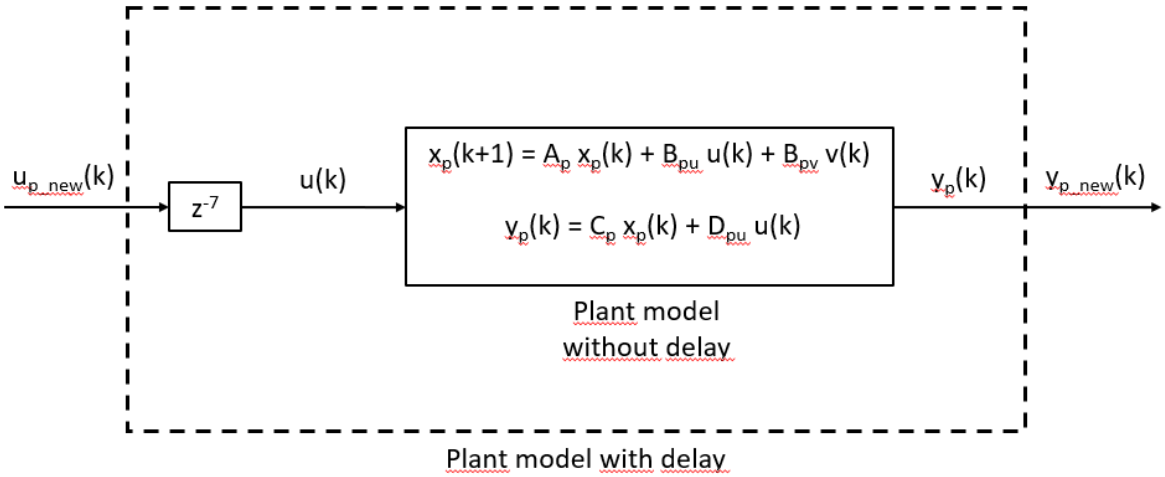

Figure 5.3. Augmentation of the Plant with Delayed Input.

where N is the delay sample which is set to 7 for this system.

(ii) Define new states,

$$x_u(k) = \begin{bmatrix} x_{u1}(k) \\ x_{u2}(k) \\ \vdots \\ x_{uN}(k) \end{bmatrix} = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{bmatrix}$$

,

$$x_u(k+1) = \underbrace{\begin{bmatrix} 0 & I & 0 & 0 & \dots & 0 \\ 0 & 0 & I & 0 & \dots & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}}_{\mathbf{O}} \begin{bmatrix} x_{u1}(k) \\ x_{u2}(k) \\ \dots \\ x_{uN}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ I \end{bmatrix} u_{p\_new}(k)$$

(iii) Combine state equations and rewrite the plant model,

$$x_{p\_new}(k+1) = \begin{bmatrix} x_p(k+1) \\ x_u(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} A_p & B_{pu} & 0 & \dots & 0 \\ 0 & & O & & \end{bmatrix}}_{A_{p\_new}} \begin{bmatrix} x_p(k) \\ x_u(k) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ I \end{bmatrix}}_{B_{pu\_new}} u_{p\_new}(k) + \underbrace{\begin{bmatrix} B_{pv} \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{B_{pv\_new}} v(k),$$

$$y_{p\_new}(k) = y_p(k) = \underbrace{\begin{bmatrix} C_p & D_{pu} & 0 & \dots & 0 \end{bmatrix}}_{C_{p\_new}} \begin{bmatrix} x_p(k) \\ x_u(k) \end{bmatrix}$$

Each element x_{ui} in new x_u state variable is 2×1 matrices because the actuator delay is added to both VGT and EGR inputs. At the same time, corresponding identity (I) and zero (0) elements in O matrix are 2×2 matrices for the same reason. Overall plant equation is:

$$\begin{aligned} x_{p\_new}(k+1) &= A_{p\_new}x_{p\_new}(k) + B_{pu\_new}u_{p\_new}(k) + B_{pv\_new}v(k), \\ y_{p\_new}(k) &= C_{p\_new}x_{p\_new}(k) \end{aligned} \tag{5.1}$$

5.2.1. Numerical Analysis

In the remaining part of the study of delayed MPC, new plant model is augmented with the disturbance models as discussed in Section 3.1.2. All states are estimated with Kalman filter and then, output prediction and cost optimization steps proceed respectively by sticking to related sections in Chapter 3. Afterwards, prediction horizon N_p and control horizon N_c are adjusted to 30 and 7 respectively to increase the controller performance.

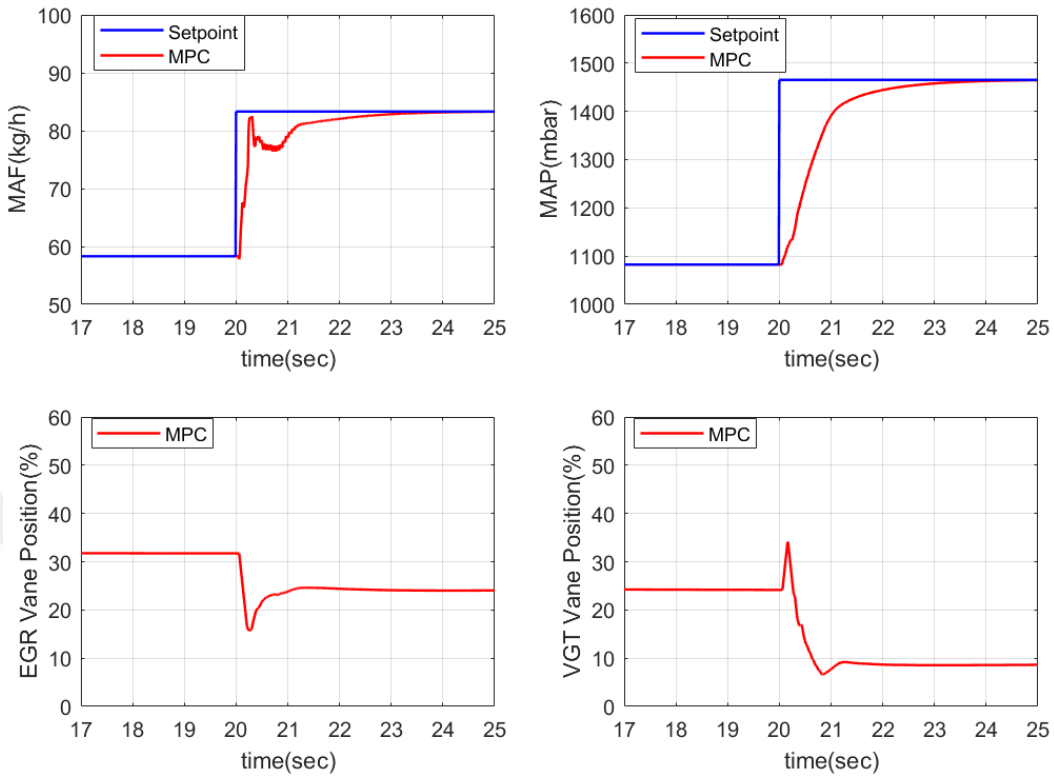

Figure 5.4. Simulation Results of MPC on Input Delayed Model - Rising Step Setpoint Change.

In simulation results depicted in Figures 5.4 and 5.5, MPC improves its performance when the model used in MPC is updated with the input delay. This shows that MPC performs better when the model used in it for predictions is closer to the actual plant model.

5.3. Summary of the Chapter

In this chapter, MPC for the AVL Boost RT DEAP model with actuator delays has been studied. The linear state-space model that is used in MPC has been extended with the new states due to delay effect to get a better control performance. It is shown that MPC performs better when the delay-case is taken into account in the plant model. In the following chapter, MPC performance will be compared to standard PID controllers.

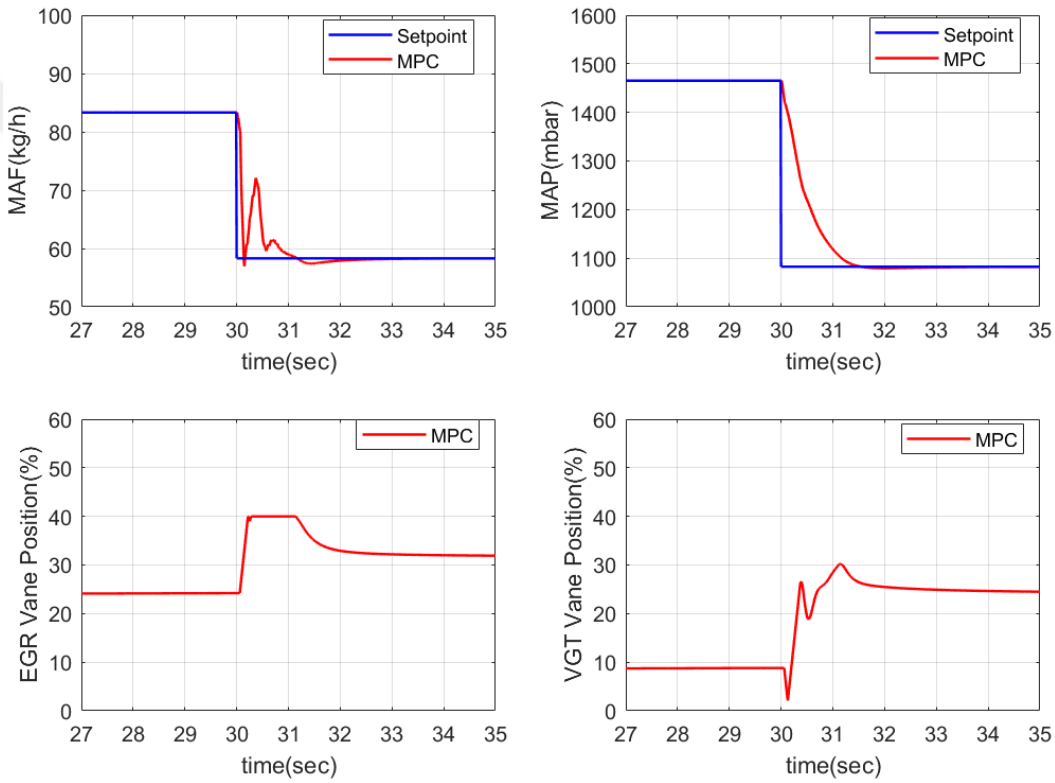

Figure 5.5. Simulation Results of MPC on Input Delayed Model - Falling Step Setpoint Change.

6. OPTIMIZED MIMO PID CONTROL OF DEAP

Map-based gain-scheduled PI-PID controllers are commonly used in industrial automotive control applications [23, 24]. The general formula of a PID controller is given by

$$\frac{u(s)}{e(s)} = K_p + K_i\frac{1}{s} + K_d s \tag{6.1}$$

where $u(s)$ is the control signal, $e(s)$ is the error between the reference signal and measured output, K_p is the proportional gain, K_i is the integral action gain, and K_d is the derivative term gain. Proportional control affects the system responsiveness to the error, the integral term is used to eliminate steady-state errors, and the derivative action is used to add damping to the system. The controller parameters are thus chosen in order to achieve prescribed performance criteria in terms of rise and settling times, overshoot, and steady-state error.

The differentiation is sensitive to the noise. Therefore, derivative action is combined with a first order filter in practical applications [25]. Matlab Simulink PID controller block is designed with this filter in derivative term and it is shown as:

$$\frac{u(s)}{e(s)} = K_p + K_i\frac{1}{s} + \frac{K_d s}{1 + \frac{1}{N}s} \tag{6.2}$$

where N is the filter coefficient.

In the air path control problem, SISO PID control loops are designed where VGT position regulates the MAP level and EGR position regulates the MAF level. Simulink overview of the closed-loop PID control system is shown in Figure 6.1. SetPoints block contains the maps that output the MAF and MAP reference values according the fuel injection and engine rpm operation points. VGT and EGR positions are regulated with PID controllers designed in PID Controller VGT and PID Controller EGR blocks. The parameters of the PID controllers are scheduled with maps in the controller blocks

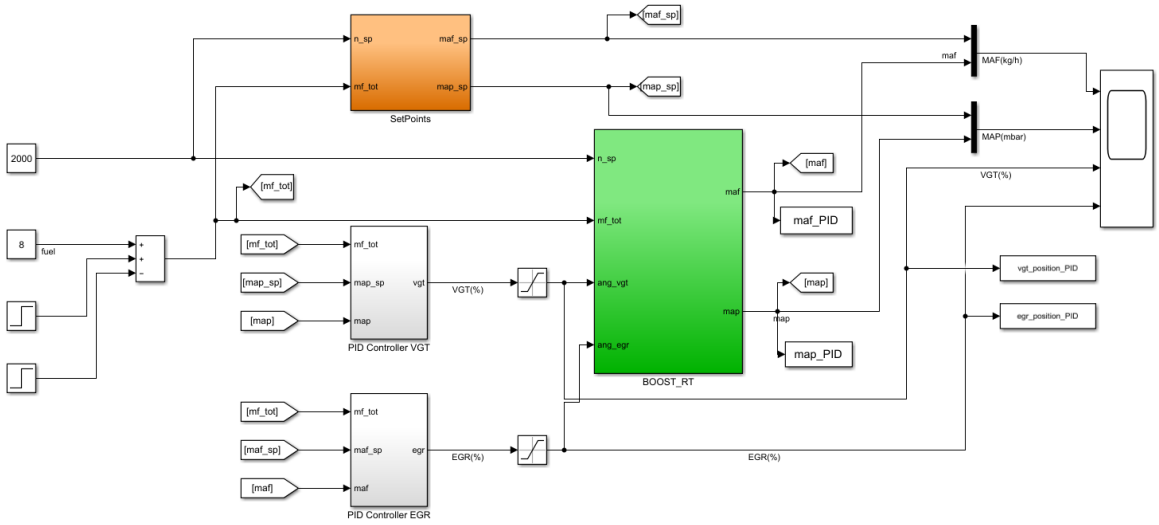

Figure 6.1. Simulink Diagram of the PID Control System.

with respect to fuel injection operation points. The inside of the PID Controller VGT block is shown in Figure 6.2. PID Controller EGR block is designed with the same principles. Also, the saturation blocks are used to keep the VGT and EGR inputs within the physical limits between 0 and 60.

The tuning of PID controller gains is a challenging task to get good performance in terms of closed loop stability and reference tracking. There are several methods in the literature for PID Tuning in a systematic way such as Ziegler-Nichols (ZN) and Cohen-Coon methods. On the other hand, optimization-based algorithms are the recent techniques to find controller gains for optimum performance, especially for systems with multiple controllers [26]. Genetic Algorithm [27], Simulating Annealing [28], and Particle Swarm Optimization [29] are examples of these techniques that are suitable for multi-parameter PID tuning.

In this chapter, firstly Ziegler-Nichols method is used to tune the PID controllers based on SISO approach. Then, Particle Swarm Optimization (PSO) technique is applied to Diesel Engine Air Path Control problem by taking the results of ZN method as a starting point in the optimization.

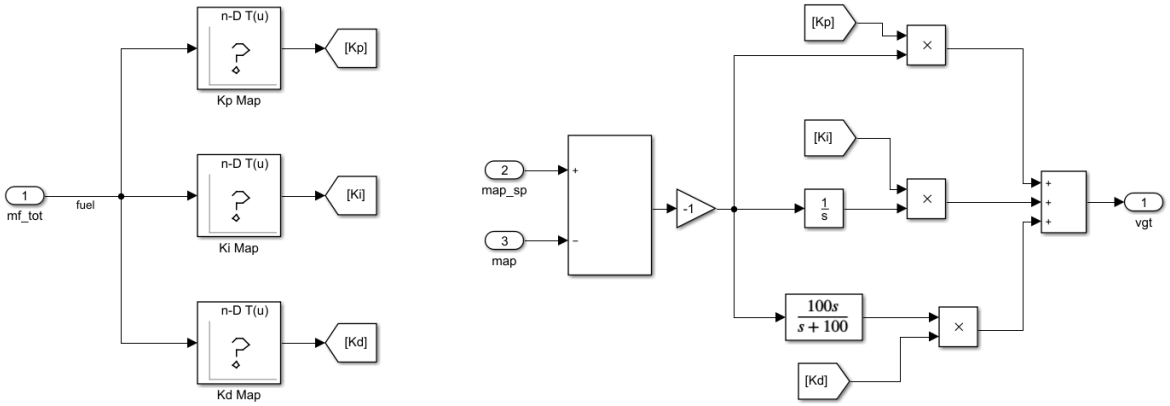

Figure 6.2. PID Controller VGT block.

6.1. Zeigler-Nichols Tuning Method

Zeigler-Nichols technique was proposed by Zeigler and Nichols in 1940s for the purpose of tuning the SISO conventional controllers and accepted as the standard in practical control applications [30]. Controller gains are calculated according to Table 6.1.

To obtain the parameters K_{cr} and P_{cr}, the steps below are followed:

(i) Set Integral and derivative gains to zero.

(ii) Increase proportional gain until critical oscillations occur.

(iii) Record proportional gain K_{cr} where the oscillations occur and period of oscillations P_{cr} in seconds.

Table 6.1. Ziegler-Nichols Tuning Rules.

Controller Type\Gain	K_p	K_i	K_d
P	$0.5K_{cr}$	0	0
PI	$0.45K_{cr}$	$1.2K_p/P_{cr}$	0
PID	$0.6K_{cr}$	$2K_p/P_{cr}$	$K_pP_{cr}/8$

(iv) Calculate the controller gains according to Table 6.1.

6.2. Particle Swarm Optimization Method

Particle Swarm Optimization (PSO) is an evolutionary computation-based optimization technique introduced by Kennedy & Eberhart in 1995. It is a population based algorithm, inspired by behaviours of flocks of birds or fish [26], [31]. In PSO, each particle has an initial position and velocity at the beginning. At each iteration, the objective function of each particle is calculated and the velocity and position of each particle are updated with the experience of finding the best objective. The velocity and position update equations are as follows:

$$V_i(k+1) = w(k)V_i(k) + c_1 rand_1(Pbest_i(k) - X_i(k)) + c_2 rand_2(Gbest(k) - X_i(k)) \tag{6.3}$$

$$X_i(k+1) = X_i(k) + V_i(k+1) \tag{6.4}$$

where $V_i(k)$ is the velocity of the ith particle at the kth iteration; w is the inertia weight; c_1 and c_2 are the acceleration factors; $rand_1$ and $rand_2$ are the uniformly distributed random variables between 0 and 1; $X_i(k)$ is the position of the ith particle at iteration k; $Pbest_i(k)$ is the best position of the ith particle at iteration k; and $Gbest(k)$ is the best position of the group until iteration k [26]. The iterations are repeated until the calculation reaches a stopping criteria such as maximum number of iterations or a relative change in the best cost function. The general flow chart of the PSO algorithm is given in Figure 6.3.

PSO has the ability to solve derivative-free unconstrained optimization problems or optimization problems with bounds. In this study, *particleswarm* function of *Matlab* is used by determining the cost function and lower and upper bounds of the variables. To define the cost function, there are some basic performance indicators and commonly used ones are the integral of the square error (ISE), integral of the absolute

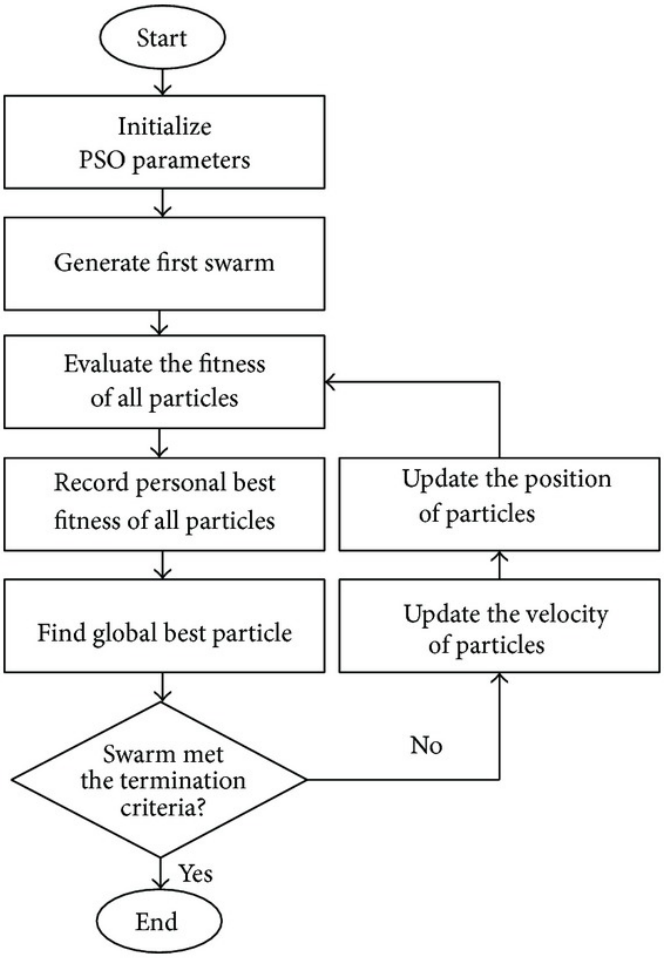

Figure 6.3. General Flow Chart of PSO [32].

value of the error (IAE), and integral of the time-weighted absolute error (ITAE). In this study, IAE minimization is chosen as the cost function. IAE is calculated by the formula:

$$IAE = \int_0^T \mid e(t) \mid dt \tag{6.5}$$

where T is the duration of simulation.

6.3. Numerical Analysis

In this part of the study, PID controller parameters are tuned for plant model with and without input delay by using the Ziegler-Nichols Tuning Method and Particle Swarm Optimization. These scenarios were discussed in Sections 4.3 and 5.2 under MPC methodology and are numerically evaluated for 3 control strategy in this section.

For the tuning of SISO PID controllers by ZN Method, firstly, EGR valve is closed and a PID controller is tuned for the MAP control. Then, a P controller is designed for the MAF control. We have opted for a P controller in regulating MAF as we have observed that PI and PID with suggested ZN parameters may lead to instability in the system.

After determining the initial PID and P gains by ZN technique, PSO algorithm is used to find the optimum PID parameters for MAP and MAF outputs. The cost function chosen to minimize the algorithm is the summation of the normalized IAE of the two outputs given as follows

$$IAE = \int_{T_1}^{T_2} \left(\frac{\mid y_{maf} - y_{maf\_ref} \mid}{y_{maf\_ref}} + \frac{\mid y_{map} - y_{map\_ref} \mid}{y_{map\_ref}} \right) dt \tag{6.6}$$

where y_{maf} and y_{map} are the measured outputs MAP and MAF; y_{maf_ref} and y_{map_ref} are the reference values for tracking of MAF and MAP outputs; T_1 is the second where the setpoint is changed and T_2 is chosen as $T_1 + 3$ second because 3 sec is enough time for offset-free tracking in the system.

The inertia weight and acceleration factors used in the PSO algorithm are chosen as the default settings of the *Matlab* 2017*b* *particleswarm* function. Two parameters, swarm size and maximum iteration number, are set to 20 as a termination criteria.

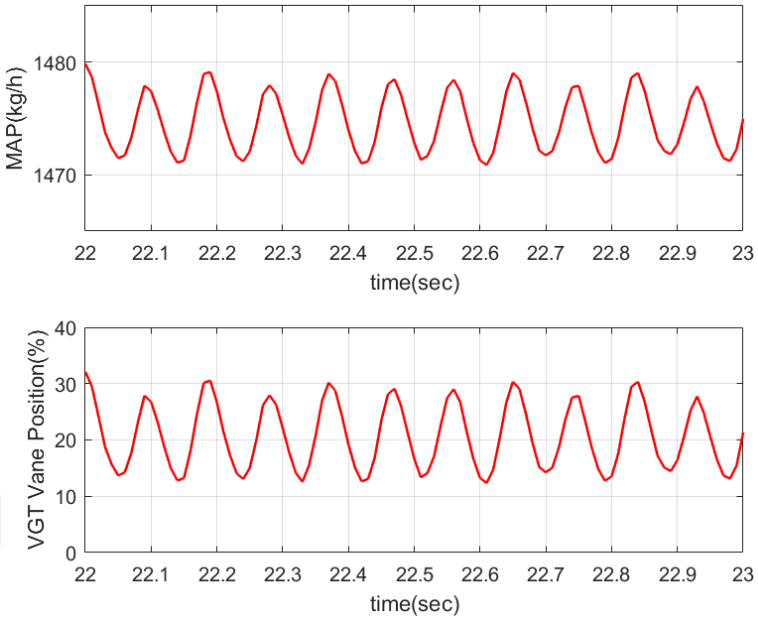

Figure 6.4. Critical Oscillations on the MAP.

6.3.1. PID Tuning for Model Without Actuator Delay

In this part of the study, firstly, the MAP control loop is tuned for operation region-2 for rising step change scenario by adjusting the gains of the VGT controller. The EGR valve is closed and then, the derivative and integral terms of the MAP controller are set to 0 and proportional gain is adjusted until critical oscillations begin in the MAP output. At this point, the value of the critical gain and the period of oscillations are noted. The critical oscillations on the MAP can be seen in Figure 6.4. Then, PID controller gains are calculated according to Ziegler-Nichols method given in Table 6.1.

MAF control is calibrated in the same manner as in the MAP control. This time, the obtained VGT controller gains are kept constant. Then, EGR controller proportional gain is increased until critical oscillations occur on the MAF output. K_{cr} and P_{cr} values are recorded. Lastly, P controller is designed by ZN method. The K_{cr}, P_{cr}, and calculated gain values for both MAP and MAF loops are given in Table 6.2.

Table 6.2. Critical Oscillation and PID Parameters for Region-2.

	K_{cr}	P_{cr}	K_p	K_i	K_d
MAP	1.4	0.12	0.84	14	0.0126
MAF	1.1	0.02	0.55	0	0

Table 6.3. PID Optimization Parameters for Region-2.

	MAF			**MAP**		
	K_p	K_i	K_d	K_p	K_i	K_d
Initial Value	0.55	0	0	0.84	14	0.0126
Lower Bound	0.1	0	0	0.1	1	0
Upper Bound	1	50	0.05	2	20	0.05
PSO Result	0.6493	38.9192	0	0.4538	1.0606	0.0066

Next, PSO algoritm is used to tune the PID parameters for MAP and MAF control. Initially, ZN parameters are given to the optimization. Although K_i and K_d terms in ZN-PID controller causes oscillations, they are kept as free parameters in the optimization by assigning lower and upper bounds. Initial conditions, upper and lower bounds given to optimization and parameters chosen by PSO are summarized in Table 6.3. In addition, average simulation time for 20 iteration is 5.5 hour.

Simulation results of ZN-PID controller, PSO-PID controller, and MPC for rising step change are shown in Figure 6.5. Also, Table 6.4 shows the MAP and MAF tracking performance of 3 controllers in terms of IAE of MAF and MAP outputs and their normalized summation.

When the results of the 3 controllers are compared in the rising step setpoint change case, MPC exhibits better performance in terms of overshoot in the system response for both output. Additionally, MPC obeys the actuator position change

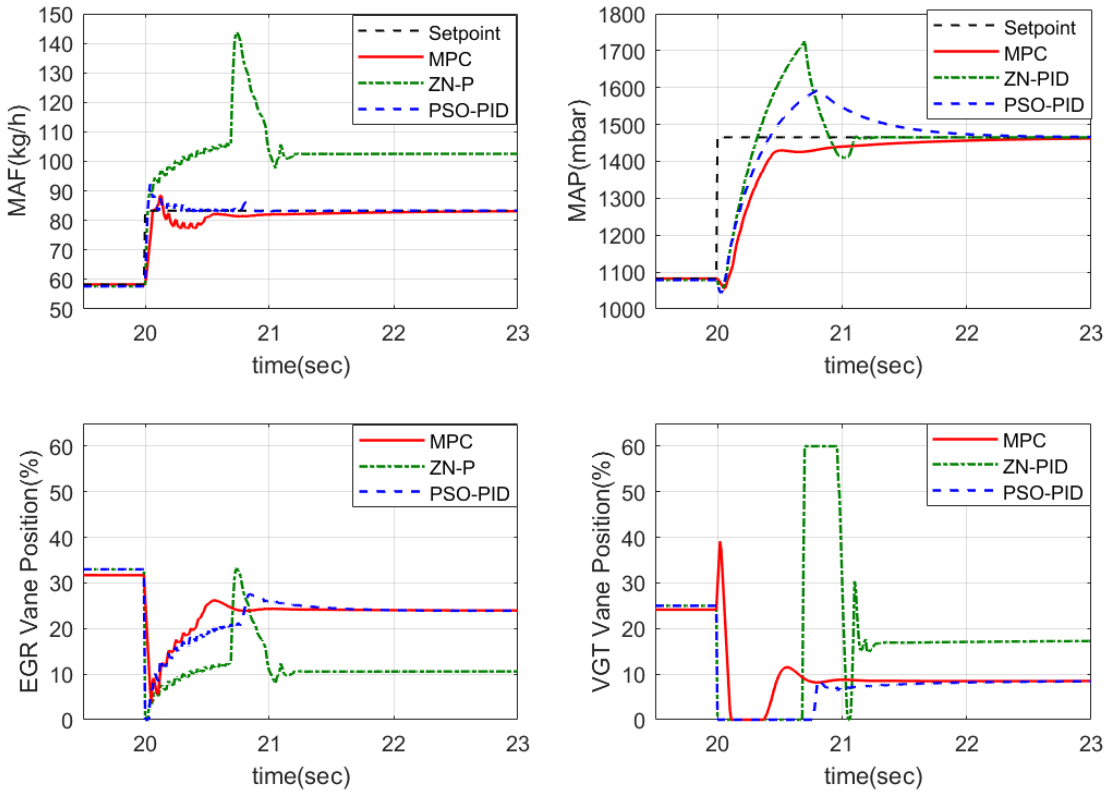

Figure 6.5. Simulation Results of 3 Controller on Input Delay Free Model - Rising Step Setpoint Change.

rate constraints even if there is a sudden change in the setpoint. Actuator position change rate could not be handled by the PID controller. On the other hand, PSO-PID improves the performance of the ZN-PID controller and gives the minimum cost in terms of normalized IAE sum of two outputs. However, it results in overshoot in MAP.

The second case is the falling step change case for the delay free model. A PID controller is designed for the operation region-1 as well. A similar way to rising step change case is followed to find the PID parameters. After K_{cr} and P_{cr} are determined, calculated ZN-PID and ZN-P parameters are used as initial points of the PSO. Parameters and the result of the PSO algorithm are summarized in Table 6.5 and closed loop responses of the 3 controllers are shown in Figure 6.6. Additionally, Table 6.6 gives the IAE values of 2 outputs and their normalized sum to compare the performances of the 3 controllers.

Table 6.4. Output Performance of 3 Controllers for Region-2.

Controller Type\Cost	IAE (MAF)	IAE (MAP)	IAE (Normalized Sum)
MPC	4.4243	138.3123	0.1475
ZN-PID	62.7335	153.5726	0.8578
PSO-PID	1.2116	162.4726	0.1254

Table 6.5. PID Optimization Parameters for Region-1.

	MAF			**MAP**		
	K_p	K_i	K_d	K_p	K_i	K_d
Initial Value	1.05	0	0	3.96	79.2	0.0495
Lower Bound	0.1	0	0	0.1	10	0
Upper Bound	2	10	0.01	5	100	0.1
PSO Result	2	1.6087	0.0009	3.0614	10	0.0074

As seen from Figure 6.6, ZN-P control results in a steady-state error and overshoot on MAF output. Although PSO-PID removes the steady-state error, there is still an overshoot on the MAF output. On the other hand, MPC and PSO-PID satisfy the expected settling times which are approximately 1.5 sec for MAF and 2 sec for MAP. Further, MPC removes the steady-state error and the overshoot in the system and it has the minimum cost compared to PID controllers.

Table 6.6. Output Performance of 3 Controllers for Region-1.

Controller Type\Cost	IAE (MAF)	IAE (MAP)	IAE (Normalized Sum)
MPC	5.7768	156.9380	0.2241
ZN-PID	52.5121	150.8728	1.0399
PSO-PID	21.1654	146.1289	0.4980

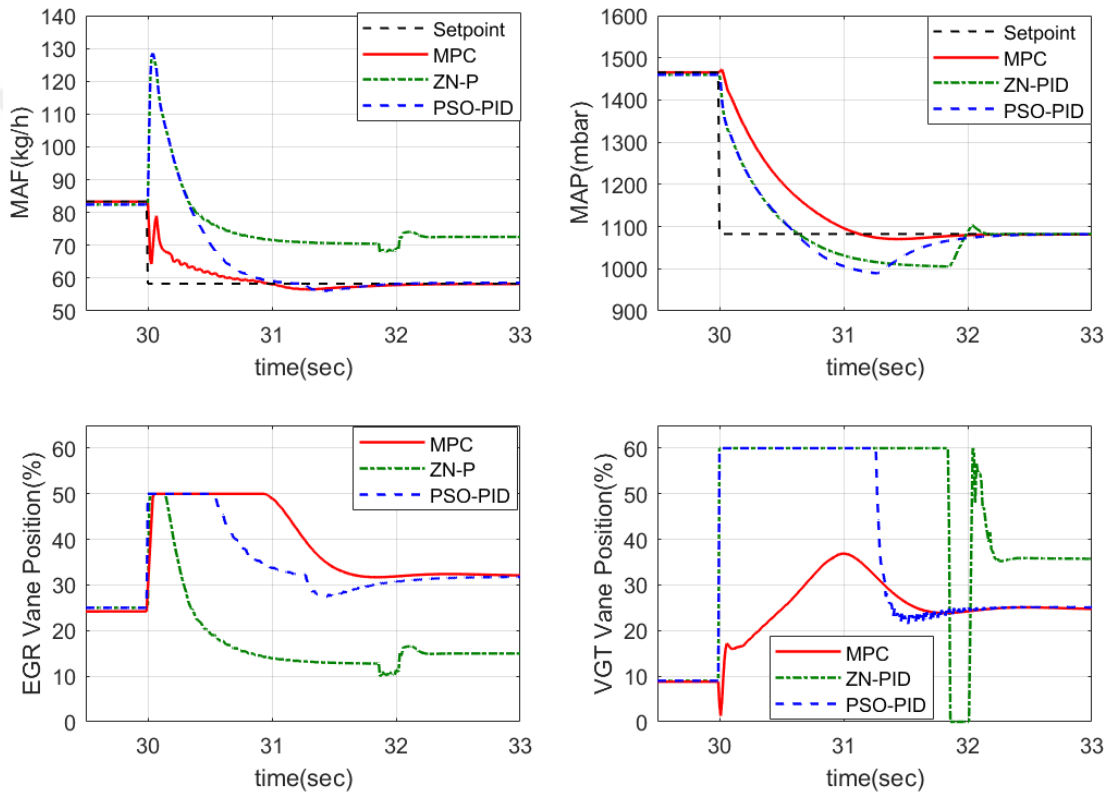

Figure 6.6. Simulation Results of 3 Controller on Input Delay Free Model - Falling Step Setpoint Change.

Table 6.7. PID Optimization Parameters for Input Delayed Model Region-2.

	MAF			**MAP**		
	K_p	K_i	K_d	K_p	K_i	K_d
Initial Value	0.375	0	0	0.21	1.1667	0.0095
Lower Bound	0.1	0	0	0.1	0.1	0
Upper Bound	1	10	0.05	1	10	0.05
PSO Result	0.2344	6.3045	0.0004	0.2261	0.3007	0.0137

6.3.2. PID Tuning for Model With Actuator Delay

The dynamics of the system change if an input delay is added. This case has been examined in Chapter 5 and the ability of MPC to handle delay has been demonstrated. In this section, ZN-PID and PSO-PID controllers are designed for the rising-step change case and the falling step change case for the input-delayed system.

For the rising step change case, a ZN-PID controller is designed for the MAP control whereas a ZN-P controller is used for the MAF control to avoid the oscillations in the system. Then, these parameters are given to the optimization as starting points. Table 6.7 summarizes the controller parameters and Figure 6.7 shows the simulation results. Furthermore, Table 6.8 gives the IAE performances of the 3 controllers.

Simulation results in Figure 6.7 show that PID controllers tuned by ZN and PSO result in oscillations in MAF and overshoot in both MAF and MAP outputs. On the other hand, MPC can handle the delay effect and performs better than PID controllers in terms of overshoot and oscillations. However, it has a slower MAP response than PID controllers due to the fact that it obeys the actuator position change rate constraints. This results in higher IAE than PSO-PID algorithm.

Lastly, a PID controller and a P controller are tuned for operation region-1 by ZN method when there is an input delay in the system. As discussed before, PSO

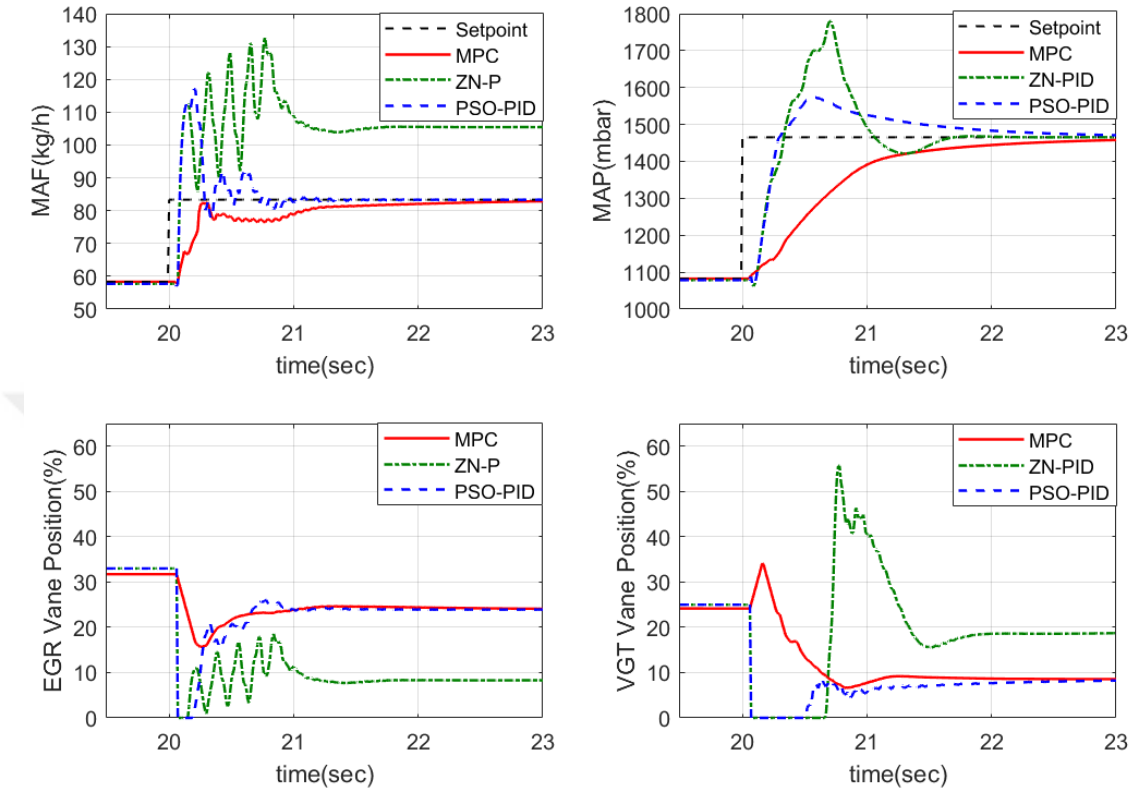

Figure 6.7. Simulation Results of 3 Controller on Input Delayed Model - Rising Step Setpoint Change.

Table 6.8. Output Performance of 3 Controllers for Delayed Model Region-2.

Controller Type\Cost	IAE (MAF)	IAE (MAP)	IAE (Normalized Sum)
MPC	11.0442	276.4756	0.332
ZN-PID	71.2192	204.0266	0.9941
PSO-PID	8.9791	171.770	0.2250

Table 6.9. PID Optimization Parameters for Input Delayed Model Region-1.

	MAF			**MAP**		
	K_p	K_i	K_d	K_p	K_i	K_d
Initial Value	0.47	0	0	0.84	4.8	0.0367
Lower Bound	0.1	0	0	0.1	1	0
Upper Bound	1	10	0.05	2	10	0.05
PSO Result	0.2896	1.6323	0.0036	0.6623	1	0

Table 6.10. Output Performance of 3 Controllers for Delayed Model Region-1.

Controller Type\Cost	IAE (MAF)	IAE (MAP)	IAE (Normalized Sum)
MPC	6.7171	174.1308	0.2761
ZN-PID	61.1368	166.9334	1.2026
PSO-PID	19.9565	145.2457	0.4764

algorithm is run and the parameters in Table 6.9 are obtained. Simulation results can be seen in Figure 6.8 and discussed according to Table 6.10.

In the last example, as in the other examples, it is seen that PSO has improved the performance of ZN controller. However, MPC is superior to others in terms of overshoot, settling-time, and steady-state error due to its predictive and multi-objective structure.

6.4. Summary of the Chapter

In this chapter, DEAP model with and without actuator delays has been controlled by SISO PID controllers. For the tuning of the PID controllers, ZN method and PSO method with the initialization of ZN parameters have been introduced. It has been shown that PSO has improved the control performance of the PID controllers.

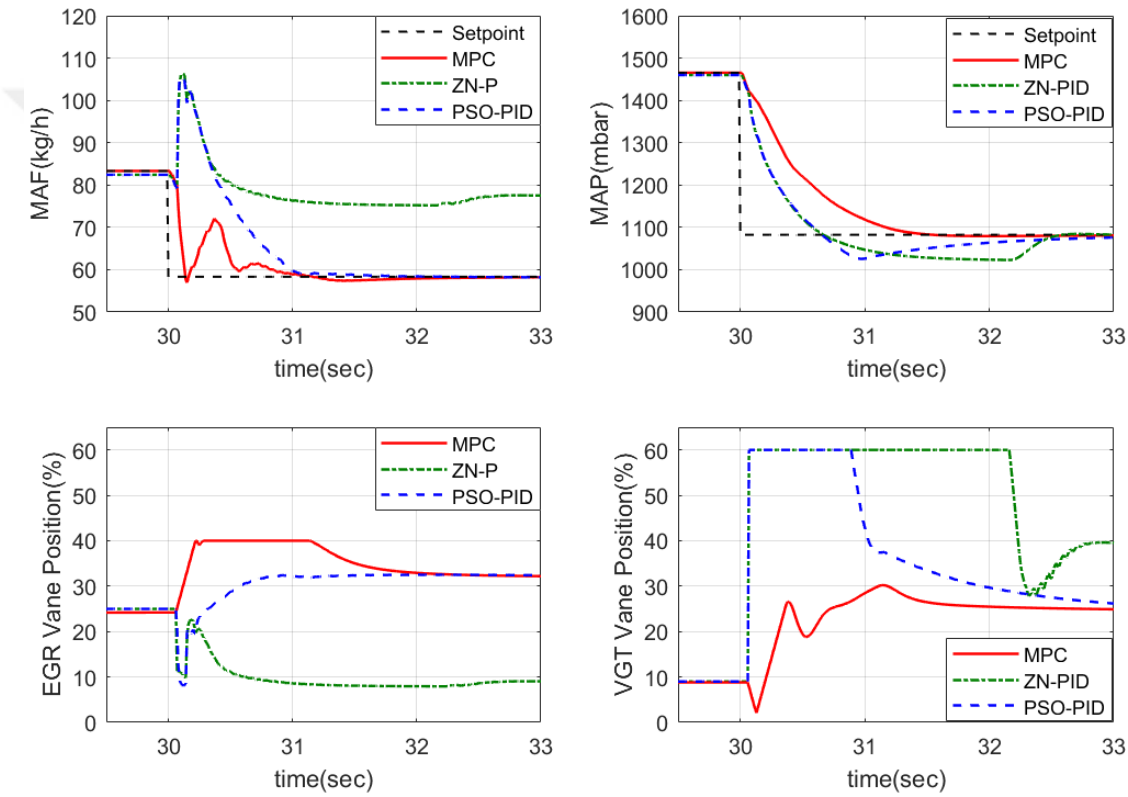

Figure 6.8. Simulation Results of 3 Controller on Input Delayed Model - Falling Step Setpoint Change.

Furthermore, the results of the 3 control methods have been compared, and it has been observed that MPC is superior to others in terms of overshoot, settling-time, and steady-state error when the setpoint decreases. When the setpoint increases, PSO-PID has the minimum normalized IAE sum of the two outputs. However, it results in overshoots in the system outputs.

7. CONCLUSION

Control of MIMO Diesel Engine Air Path is a challenging task for car manufacturers due to tightening emission regulations. In industry, control of this system is generally done by SISO PID controllers. However, working with PID controllers requires high calibration effort. On the other hand, it is difficult the fulfill the tightening emission standards without taking into account the decoupling between actuators.

The first objective of this thesis has been to apply MPC technique on Diesel Engine Air Path system. For this reason, system identification has been performed on 2 operation points of the system. The performance of the MPC controller has been evaluated when there is a change in the system operation point and setpoints of the outputs increase/decrease. It has been shown that MPC can handle the settling time, overshoot, and zero steady-state criteria for both outputs even if there is a coupling between 2 actuators and 2 outputs.

Actuator delays are commonly present in real-time systems and they deteriorate the control performance. In Chapter 5, a time delay has been added to the system actuators and model in the MPC algorithm has been extended with this time delay. It has been shown that MPC performs better when the delay is taken into account in the algorithm.

Although model based control algorithms are the mostly studied techniques in literature, SISO PID controllers are still the most common control method in automotive industry. Calibration of these SISO PID controllers are done by calibration engineers starting with Ziegler-Nichols constants and then tuned manually until a desired performance is achieved. To improve the performance of PID controllers and tune the parameters in a systematic way, PSO algorithm has been used in this study. Two PID controllers for MAF and MAP control have been tuned starting with the ZN calibration parameters. It has been seen in numerical analysis that PSO improves the control performance compared to ZN tuning.

As a conclusion, performances of the MPC, ZN-PID, and PSO-PID controllers have been compared in terms of IAE of MAF and MAP outputs and their normalized summation. It has been observed that MPC has the minimum tracking cost when the setpoint is decreased. On the other hand, if the setpoint is increased, the cost of PSO-PID controller is the lowest compared to others. However, it results in overshoot in MAP output and leads oscillations in MAF output when there is an input delay in the system. MPC can handle the delay effect without causing oscillations on the system outputs. The final conclusion of this work is about the computation time of the PSO-PID controllers. For each scenario discussed in this thesis, PSO algorithm has been run about 5.5 hour for the calibration of PID controllers. In real time systems, it requires a big offline calibration effort for each predefined operation point. Since the MPC is an online algorithm, it requires less offline preparation time than calibrating the PID controllers.

Lastly, there are some open subjects for further improvements of this study. One of them is improvement of the plant model used in MPC control. Instead of black-box modeling, a more accurate model can be developed such as by using Neural Network, Machine Learning, or Fuzzy Logic. A second improvement is in the PID tuning part. The PSO can be started from more proper initial conditions to reach a global optimal solution. On the other hand, cost function IAE of the PSO algorithm can also be extended to reduce the overshoot.

REFERENCES

1. Ortner, P. and L. D. Re, "Predictive Control of a Diesel Engine Air Path", *IEEE Transactions on Control Systems Technology*, Vol. 15, 2007.

2. Zhao, D., C. Liu, R. Stobart, J. Deng and E. Winward, "Explicit Model Predictive Control on the Air Path of Turbocharged Diesel Engines", *American Control Conference (ACC)*, 2013.

3. Herceg, M., T. Raff, R. Findeisen and F. Allgöwer, "Nonlinear Model Predictive Control of a Turbocharged Diesel Engine", *International Conference on Control Applications*, 2006.

4. Herceg, M., T. Raff, R. Findeisen and F. Allgöwer, "Nonlinear Model Predictive Control of a Diesel Engine Airpath", *IFAC Proceedings Volumes*, Vol. 42, 2009.

5. Gelso, E. R. and J. Lindberg, "Airpath Model Predictive Control of a Heavy-Duty Diesel Engine with Variable Valve Actuation", *IFAC Proceedings Volumes*, Vol. 47, 2014.

6. Gelso, E. R. and J. Dahl, "Air-Path Control of a Heavy-Duty EGR-VGT Diesel Engine", *IFAC-PapersOnLine*, Vol. 49, 2016.

7. Jung, M., R. Ford, K. Glover, N. Collings, U. Christen and M. Watts, "Parameterization and Transient Validation of a Variable Geometry Turbocharger for Mean-Value Modeling at Low and Medium Speed-Load Points", *SAE Technical Paper*, 2002.

8. Jung, M. and K. Glover, "Control-Oriented Linear Parameter-Varying Modelling of a Turbocharged Diesel Engine", *Proceedings of 2003 IEEE Conference*, Vol. 1, 2003.

9. Yavas, U. and M. Gokasan, "Model Predictive Control of Turbocharged Diesel Engine with Exhaust Gas Recirculation", *World Academy of Science, Engineering and Technology, International Journal of Industrial and Manufacturing Engineering*, Vol. 2, No. 11, 2015.

10. Halvorsen, H. P., *Model Predictive Control in LabVIEW*, 2016, `http://home.hit.no/ hansha/documents/labview/training/Model Predictive Control in LabVIEW/Model Predictive Control in LabVIEW.pdf`, accessed at September 2017.

11. Bemporad, A., M. Morari and N. L. Ricker, *Model Predictive Control Toolbox 3 User's Guide*, The mathworks, 2010.

12. Wang, L., *Model predictive control system design and implementation using MATLAB®*, Springer Science & Business Media., London, 2009.

13. Bishop, G., G. Welch *et al.*, "An introduction to the Kalman filter", *Proc of SIGGRAPH, Course*, Vol. 8, No. 27599-23175, p. 41, 2001.

14. Aliyu, B. K., C. A. Osheku, L. M. Adetoro and A. A. Funmilayo, "Optimal solution to matrix Riccati equation–for Kalman filter implementation", *MATLAB-A Fundamental Tool for Scientific Computing and Engineering Applications-Volume 3*, InTech, 2012.

15. Bemporad, A. and P. Patrinos, "Simple and Certifiable Quadratic Programming Algorithms for Embedded Linear Model Predictive Control", *Proc. 4th IFAC Nonlinear Model Predictive Control Conference*, pp. 14–20, 2012.

16. Ferreau, H. J., H. G. Bock and M. Diehl, "An Online Active Set Strategy to Overcome the Limitations of Explicit MPC", *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, Vol. 18, No. 8, pp. 816–830, 2008.

17. Wang, Y. and S. Boyd, "Fast Model Predictive Control Using Online Optimiza-

tion", *IEEE Transactions on Control Systems Technology*, Vol. 18, No. 2, p. 267, 2010.

18. Ljung, L., *Sytem Identification Theory for the User*, Prentice Hal., Englewood Cliffs, New Jersey, 1999.

19. Viberg, M., "Subspace-based Methods for the Identification of Linear Time-invariant Systems", *Automatica*, Vol. 32, pp. 1835–1851, 1995.

20. Overschee, P. V. and B. D. Moor, "N4SID: Subspace Algorithms for the Identification of Combined Deterministic-Stochastic Systems", *Automatica*, Vol. 30, No. 1, pp. 75–93, 1994.

21. Favoreel, W., B. D. Moor and P. V. Overschee, "Subspace State Space System Identification for Industrial Processes", *Journal of Process Control*, Vol. 10, No. 2-3, pp. 149–155, 2000.

22. Choomkasien, P., P. Chomphooyod and D. Banjerdpongchai, "Design of Model Predictive Control for Industrial Process with Input Time Delay", *Control, Automation and Systems (ICCAS), 2017 17th International Conference on*, pp. 625–630, 2017.

23. Howell, M. N. and M. C. Best, "On-line PID Tuning for Engine Idle-speed Control Using Continuous Action Reinforcement Learning Automata", *Control Engineering Practice*, Vol. 8, No. 2, pp. 147–154, 2000.

24. Mayr, C. H., N. Euler-Rolle, M. Kozek, C. Hametner and S. Jakubek, "Engine control unit PID controller calibration by means of local model networks", *Control Engineering Practice*, Vol. 33, pp. 125–135, 2014.

25. Karl Johan Åström, "Control system design lecture notes for me 155a", *Department of Mechanical and Environmental Engineering University of California Santa Barbara*, p. 333, 2002.

26. Iruthayarajan, M. W. and S. Baskar, "Evolutionary Algorithms Based Design of Multivariable PID Controller", *Expert Systems with Applications*, Vol. 36, No. 5, pp. 9159–9167, 2009.

27. Chang, W.-D., "A Multi-crossover Genetic Approach to Multivariable PID Controllers Tuning", *Expert Systems with Applications*, Vol. 33, No. 3, pp. 620–626, 2007.

28. Mary, A., L. P. Suresh and S. K. Veni, "Comparative Performance Analysis of Different Controllers for a Nonlinear Multivariable System", *Emerging Technological Trends (ICETT), International Conference on*, pp. 1–6, IEEE, 2016.

29. El-Gammal, A. A. and A. A. El-Samahy, "A Modified Design of PID Controller for DC Motor Drives Using Particle Swarm Optimization PSO", *Power Engineering, Energy and Electrical Drives, 2009. POWERENG'09. International Conference on*, pp. 419–424, IEEE, 2009.

30. Copeland, B. R., *The Design of PID Controllers Using Ziegler Nichols Tuning*, 2008, `http://educypedia.karadimov.info/library/Ziegler_Nichols.pdf`, accessed at September 2018.

31. Queen, A. A. and D. J. Auxillia, "Simplified Discrete Binary PSO Tuned Multivariable PID Controller for Binary Distillation Column Plant", *Circuits, Power and Computing Technologies (ICCPCT), 2013 International Conference on*, pp. 667–672, IEEE, 2013.

32. Kalatehjari, R., A. Rashid, N. Ali and M. Hajihassani, "The Contribution of Particle Swarm Optimization to Three-Dimensional Slope Stability Analysis", *The Scientific World Journal*, Vol. 2014, 2014.

APPENDIX A: SYSTEM IDENTIFICATION MATRICES/MATLAB CODES

In this part, the plant model matrices obtained by System Identification for 2 operation regions and the codes written in Matlab environment for MPC control are given.

A.1. System Identification Matrices

System matrices of operation region 1 are obtained as follows:

$$A_p = \begin{bmatrix} 0.837 & -0.155 & 0.062 & -0.067 \\ -0.015 & 0.087 & -0.383 & -0.133 \\ -0.049 & -0.129 & 0.510 & -0.011 \\ 0.029 & -0.069 & 0.034 & 0.633 \end{bmatrix}, \quad B_p = \begin{bmatrix} 0 & 0.002 & -0.001 & 0 \\ 0 & -0.001 & -0.002 & 0.003 \\ 0 & 0.001 & -0.001 & 0.001 \\ 0 & -0.001 & 0 & 0.001 \end{bmatrix},$$
$$C_p = \begin{bmatrix} 111.095 & -245.572 & 49.291 & 10.431 \\ 1571.988 & -26.136 & 19.525 & -36.703 \end{bmatrix}, \quad D_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{A.1}$$

System matrices of operation region 2 are given below:

$$A_p = \begin{bmatrix} 0.851 & -0.105 & 0.115 & -0.040 \\ 0.094 & 0.157 & -0.382 & -0.273 \\ -0.041 & -0.198 & 0.389 & -0.173 \\ -0.028 & -0.007 & 0.282 & 0.739 \end{bmatrix}, \quad B_p = \begin{bmatrix} 0 & 0.002 & -0.001 & 0 \\ 0 & -0.001 & -0.002 & 0.004 \\ 0 & 0.001 & -0.001 & 0.001 \\ 0 & -0.003 & 0 & 0 \end{bmatrix},$$
$$C_p = \begin{bmatrix} 227.113 & -236.058 & 52.452 & 2.918 \\ 3257.899 & -170.959 & 127.491 & -65.065 \end{bmatrix}, \quad D_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{A.2}$$

A.2. Matlab Codes for MPC

At the beginning, the user defined model parameters, disturbance parameters, input, input change, output constraints, controller sample time, prediction and control horizons, cost function weightings are defined for 2 operation region.

```
load('ss_8fuel.mat');
load('ss_20fuel.mat');
load('ALL_map.mat');

PH = 10; % prediction horizon
CH = 3; %Control Horizon

%%%%%prepare offline matrices for linearized region 1
mpcModel1.plantModel=evalin('base','ss_8fuel');
mpcModel1.Ts = 0.01;
mpcModel1.PH = PH; % prediction horizon
mpcModel1.CH = CH; %Control Horizon

% manipulated variable interval
mpcModel1.uLowLimit = [0;0];
mpcModel1.uHighLimit = [60;50];
mpcModel1.uMinECR = [0;0];
mpcModel1.uMaxECR = [0;0];

% manipulated variable rate interval
mpcModel1.uRateLowLimit = [-5;-5];
mpcModel1.uRateHighLimit = [5;5];
mpcModel1.uRateMinECR = [0;0];
mpcModel1.uRateMaxECR = [0;0];

% control interval
mpcModel1.yLowLimit = [30;800];
mpcModel1.yHighLimit = [100;1500];
mpcModel1.yMinECR = [1;1];
mpcModel1.yMaxECR = [1;1];

mpcModel1.Q_maf = 15;
mpcModel1.Q_map = 1;
mpcModel1.R_vgt = 15;
mpcModel1.R_egr = 1.5;
mpcModel1.ro_epsilon = 1e8;

%output disturbance matrices
mpcModel1.outputDistModel.A_od = [1 0;0 1];
mpcModel1.outputDistModel.B_od = [0.1 0;0 0.7];
mpcModel1.outputDistModel.C_od = [1 0;0 1];
mpcModel1.outputDistModel.D_od = [0 0;0 0];

%meas noise matrices
mpcModel1.measurementNoiseModel.A_n = [];
mpcModel1.measurementNoiseModel.B_n = [];
mpcModel1.measurementNoiseModel.C_n = [];
mpcModel1.measurementNoiseModel.D_n = [1 0;0 1];

clear OfflineCalcMatrices_m1;
OfflineCalcMatrices_m1 = calculateOfflineMatrices(mpcModel1);
```

Figure A.1. m File of Parameter Definitions.

```
%%%%%prepare offline matrices for linearized region 2
mpcModel2.plantModel=evalin('base','ss_20fuel');
mpcModel2.Ts = 0.01;
mpcModel2.PH = PH; % prediction horizon
mpcModel2.CH = CH; %Control Horizon

% manipulated variable interval
mpcModel2.uLowLimit = [0;0];
mpcModel2.uHighLimit = [60;50];
mpcModel2.uMinECR = [0;0];
mpcModel2.uMaxECR = [0;0];

% manipulated variable rate interval
mpcModel2.uRateLowLimit=[-5;-5];
mpcModel2.uRateHighLimit=[5;5];
mpcModel2.uRateMinECR = [0;0];
mpcModel2.uRateMaxECR = [0;0];

% control interval
mpcModel2.yLowLimit=[40;1000];
mpcModel2.yHighLimit=[250;2500];
mpcModel2.yMinECR = [1;1];
mpcModel2.yMaxECR = [1;1];

mpcModel2.Q_maf = 50;
mpcModel2.Q_map = 1;
mpcModel2.R_vgt = 15;
mpcModel2.R_egr = 1.5;
mpcModel2.ro_epsilon = 1e8;

%output disturbance matrices
mpcModel2.outputDistModel.A_od=[1 0;0 1];
mpcModel2.outputDistModel.B_od=[0.1 0;0 0.7];
mpcModel2.outputDistModel.C_od=[1 0;0 1];
mpcModel2.outputDistModel.D_od=[0 0;0 0];

%meas noise matrices
mpcModel2.measurementNoiseModel.A_n=[];
mpcModel2.measurementNoiseModel.B_n=[];
mpcModel2.measurementNoiseModel.C_n=[];
mpcModel2.measurementNoiseModel.D_n=[1 0;0 1];

clear OfflineCalcMatrices_m2;
OfflineCalcMatrices_m2 = calculateOfflineMatrices(mpcModel2);
```

Figure A.1. m File of Parameter Definitions (cont.).

Also, all matrices mentioned in Chapter 3 are calculated with the calculateOfflineMatrices function given below. This function is called from the above m file before the simulation is started.

```
function OfflineCalcMatrices = calculateOfflineMatrices(mpcModel)

Ts = mpcModel.Ts;
PH = mpcModel.PH; % prediction horizon
CH = mpcModel.CH; %Control Horizon
% manipulated variable interval
uLowLimit = mpcModel.uLowLimit;
uHighLimit = mpcModel.uHighLimit;
uMinECR = mpcModel.uMinECR;
uMaxECR = mpcModel.uMaxECR;

% manipulated variable rate interval
uRateLowLimit = mpcModel.uRateLowLimit;
uRateHighLimit = mpcModel.uRateHighLimit;
uRateMinECR = mpcModel.uRateMinECR;
uRateMaxECR = mpcModel.uRateMaxECR;

% control interval
yLowLimit = mpcModel.yLowLimit;
yHighLimit = mpcModel.yHighLimit;
yMinECR = mpcModel.yMinECR;
yMaxECR = mpcModel.yMaxECR;

Q_maf = mpcModel.Q_maf;
Q_map = mpcModel.Q_map;
R_vgt = mpcModel.R_vgt;
R_egr = mpcModel.R_egr;
ro_epsilon = mpcModel.ro_epsilon;

OfflineCalcMatrices.Ts = Ts;
OfflineCalcMatrices.PH = PH;
OfflineCalcMatrices.CH = CH;
OfflineCalcMatrices.ro_epsilon = ro_epsilon;

%resample identified plant model
plantModel_resamp = d2d(mpcModel.plantModel,Ts);

plantModel_withoutDelay = absorbDelay(plantModel_resamp);

Ap=plantModel_withoutDelay.A ;
Bp=plantModel_withoutDelay.B;
Cp=plantModel_withoutDelay.C;
Dp=plantModel_withoutDelay.D;

Bp_u = Bp(:,3:4); %manipulated variables
Bp_v = Bp(:,1:2); %measured disturbances

%output disturbance matrices
A_od = mpcModel.outputDistModel.A_od;
B_od = mpcModel.outputDistModel.B_od;
C_od = mpcModel.outputDistModel.C_od;
D_od = mpcModel.outputDistModel.D_od;

%meas noise matrices
A_n = mpcModel.measurementNoiseModel.A_n;
B_n = mpcModel.measurementNoiseModel.B_n;
C_n = mpcModel.measurementNoiseModel.C_n;
D_n = mpcModel.measurementNoiseModel.D_n;
```

Figure A.2. calculateOfflineMatrices Function.

```
%State Observer
[m1,n1]=size(Ap);
[m2,n2]=size(A_od);
A=eye(m1+m2,n1+n2);
A(1:m1,1:n1)=Ap;
A(m1+1:end,n1+1:end)=A_od;

[m1,n1]=size(Bp_u);
[m2,n2]=size(Bp_v);
[m3,n3]=size(B_od);
[m4,n4]=size(D_n);

B=zeros(m1+m3,n1+n2+n3+n4);
B(1:m1,1:n1)=Bp_u;
B(1:m1,n1+1:n1+n2)=Bp_v;
B(m1+1:end,n1+n2+1:n1+n2+n3)=B_od;

B_u = B(:,1:2); %for manipulated variables
B_v = B(:,3:4); %for measured disturbances

C=[Cp C_od];
D = [Dp D_od D_n];

OfflineCalcMatrices.A = A;
OfflineCalcMatrices.B = B;
OfflineCalcMatrices.B_u = B_u;
OfflineCalcMatrices.B_v = B_v;
OfflineCalcMatrices.C = C;

%augmented plant with the output disturbance model
B_kal = eye(size(B,1));
D_kal = zeros(size(D,1),size(B,1));
Plant_Kalman = ss(A,B_kal,C,D_kal,Ts);

%State Estimation
Qn = B*B';
Rn = D*D';
Nn = B*D';
[~,L,~,M] = kalman(Plant_Kalman,Qn,Rn,Nn);

OfflineCalcMatrices.L = L;
OfflineCalcMatrices.M = M;

% Q matrix penalizes tracking error for performance.
Q_out = diag([Q_maf^2 Q_map^2]);
Q = Q_out;
for i=1:PH-1
    Q = blkdiag(Q,Q_out);
end


% R matrix penalizes MV rate of change for robustness.
R_u = diag([R_vgt^2 R_egr^2]);
R = R_u;
for i=1:CH-1
    R = blkdiag(R,R_u);
end
```

Figure A.2. calculateOfflineMatrices Function (cont.).

```
%calculate matrices for J
[r1,c1] = size(A) ;
[r2,c2] = size(B_u) ;
[r3,c3] = size(B_v) ;
[r4,c4] = size(C) ;
Sx = ones(PH*r4,c1);
for i=1:PH
    j = i-1 ;
   Sx((j*r4+1):(i*r4),:) = C*(A^i) ;
end

Su_1 = zeros(PH*r4,c2);
for i=1:PH
    if i == 1
       Su_1(1:r4,:) = C*B_u ;
    else
       j = i-1 ;
       Su_1((j*r4+1):(i*r4),:) = Su_1(((j-1)*r4+1):(j*r4),:) + C*(A^j)*B_u ;
    end
end

Su = zeros(r4*PH,CH*c2) ;
for k =1:CH
    l= k-1 ;
    for i = 1:PH
        j = i-1 ;
        if(k>i)
          Su((j*r4+1):(i*r4),(l*c2+1):(k*c2)) =  0 ;
        else
          if i-k == 0
              Su((j*r4+1):(i*r4),(l*c2+1):(k*c2)) =  C*B_u ;
          else
              Su((j*r4+1):(i*r4),(l*c2+1):(k*c2)) =...
              Su(((j-1)*r4+1):(j*r4),(l*c2+1):(k*c2)) + C*(A^(i-k))*B_u ;
          end
        end
    end
end

Hv = zeros(r4*PH,c3*PH) ;
for k =1:PH
    l= k-1 ;
    for i = 1:PH
        j = i-1 ;
        if(k>i)
          Hv((j*r4+1):(i*r4),(l*c3+1):(k*c3)) =  0 ;
        else
          Hv((j*r4+1):(i*r4),(l*c3+1):(k*c3)) =  C*(A^(i-k))*B_v ;
        end
    end
end
```

Figure A.2. calculateOfflineMatrices Function (cont.).

```
Hv = zeros(r4*PH,c3*PH) ;
for k =1:PH
    l= k-1 ;
    for i = 1:PH
        j = i-1 ;
        if(k>i)
          Hv((j*r4+1):(i*r4),(l*c3+1):(k*c3)) =  0 ;
        else
          Hv((j*r4+1):(i*r4),(l*c3+1):(k*c3)) =  C*(A^(i-k))*B_v ;
        end
    end
end


%matrices to convert U to deltaU
T1_ini = eye(c2,c2);
T1 = zeros(c2*CH,c2*CH) ;
for k =1:CH %column
    l= k-1 ;
    for i = 1:CH %row
        j = i-1 ;
        if(k==i)
          T1((j*c2+1):(i*c2),(l*c2+1):(k*c2)) =  T1_ini ;
        elseif (i-k==1)
          T1((j*c2+1):(i*c2),(l*c2+1):(k*c2)) =  -T1_ini;
        end
    end
end

T2 = zeros(c2*CH,c2);
T2(1:c2,1:c2) = eye(c2,c2);

%calculate matrices for J_deltaU
Hu = R;

%calculate matrices for Jy
Sa = Su'*Q*Su;
Hy = Sa;

Sb = 2*Su'*Q';
gy_1 = Sb*Su_1; %multiplied by u(-1)
gy_2 = Sb*Sx;
gy_3 = Sb*Hv;
gy_4 = -1*Sb;


%matrices for QP
H = 2*(Hu + Hy);
OfflineCalcMatrices.H = H;

OfflineCalcMatrices.g1 = gy_1; %multiplied part by u(-1)
OfflineCalcMatrices.g2 = gy_2; %multiplied part by x(0)
OfflineCalcMatrices.g3 = gy_3; %multiplied part by v
OfflineCalcMatrices.g4 = gy_4; %multiplied part by W
```

Figure A.2. calculateOfflineMatrices Function (cont.).

```
%Calculate matrices for inequality constraints

%matrices for input constraints
Vmin_U = repmat(uMinECR,CH,1);
Vmax_U = repmat(uMaxECR,CH,1);
Umin = repmat(uLowLimit,CH,1);
Umax = repmat(uHighLimit,CH,1);

M1 = [-1*(inv(T1)) -1*Vmin_U; inv(T1) -1*Vmax_U];
N1_c = [-1*Umin;Umax]; %%constant part
N1_u = [inv(T1)*T2;-1*(inv(T1)*T2)]; % multiplied by u(-1)

%matrices for input rate constraints
Vmin_deltaU = repmat(uRateMinECR,CH,1);
Vmax_deltaU = repmat(uRateMaxECR,CH,1);
deltaUmin = repmat(uRateLowLimit,CH,1);
deltaUmax = repmat(uRateHighLimit,CH,1);

M2 = [-1*eye(c2*CH) -1*Vmin_deltaU; eye(c2*CH) -1*Vmax_deltaU];
N2 = [-1*deltaUmin;deltaUmax];

%matrices for output constraints
Vmin_y = repmat(yMinECR,PH,1);
Vmax_y = repmat(yMaxECR,PH,1);
Ymin = repmat(yLowLimit,PH,1);
Ymax = repmat(yHighLimit,PH,1);

M3 = [-1*Su -1*Vmin_y; Su -1*Vmax_y];

N3_c = [-1*Ymin; Ymax]; %constant part
N3_x = [Sx; -1*Sx]; % multiplied by xp
N3_u = [Su_1; -1*Su_1]; % multiplied by u(-1)
N3_v = [Hv; -1*Hv]; % multiplied by v

%matrices for slack variable constraint
M4 = [zeros(1,c2*CH) -1];
N4 = 0;

OfflineCalcMatrices.M1 = M1;
OfflineCalcMatrices.M2 = M2;
OfflineCalcMatrices.M3 = M3;
OfflineCalcMatrices.M4 = M4;
OfflineCalcMatrices.N1_c = N1_c;
OfflineCalcMatrices.N1_u = N1_u;
OfflineCalcMatrices.N2 = N2;
OfflineCalcMatrices.N3_c = N3_c;
OfflineCalcMatrices.N3_x = N3_x;
OfflineCalcMatrices.N3_u = N3_u;
OfflineCalcMatrices.N3_v = N3_v;
OfflineCalcMatrices.N4 = N4;
```

Figure A.2. calculateOfflineMatrices Function (cont.).

Lastly, calculateOnlineMatrices function is called from the Simulink MPCBlock at each sample of the simulation to calculate optimum manipulated variables.

```
function [mv,x_est1,x_est2] = calculateOnlineMatrices(y,yref,u_prev,...
                                x_est1_prev,x_est2_prev,md,switch_sgn)
%%get offline calculated matrices for current region
if switch_sgn == 1
    OfflineCalcMatrices_curr=evalin('base','OfflineCalcMatrices_m1');
    x_est0 = x_est1_prev;
else
    OfflineCalcMatrices_curr=evalin('base','OfflineCalcMatrices_m2');
    x_est0 = x_est2_prev;
end

OfflineCalcMatrices_m1 = evalin('base','OfflineCalcMatrices_m1');
OfflineCalcMatrices_m2 = evalin('base','OfflineCalcMatrices_m2');

PH = OfflineCalcMatrices_curr.PH;
CH = OfflineCalcMatrices_curr.CH;
ro_epsilon = OfflineCalcMatrices_curr.ro_epsilon;

L = OfflineCalcMatrices_curr.L;
M = OfflineCalcMatrices_curr.M;
A = OfflineCalcMatrices_curr.A;
B_u = OfflineCalcMatrices_curr.B_u;
B_v = OfflineCalcMatrices_curr.B_v;
C = OfflineCalcMatrices_curr.C;
H = OfflineCalcMatrices_curr.H;
g1 = OfflineCalcMatrices_curr.g1; %multiplied part by u(-1)
g2 = OfflineCalcMatrices_curr.g2; %multiplied part by x(0)
g3 = OfflineCalcMatrices_curr.g3; %multiplied part by v
g4 = OfflineCalcMatrices_curr.g4; %multiplied part by W
M1 = OfflineCalcMatrices_curr.M1;
M2 = OfflineCalcMatrices_curr.M2;
M3 = OfflineCalcMatrices_curr.M3;
M4 = OfflineCalcMatrices_curr.M4;
N1_c = OfflineCalcMatrices_curr.N1_c;
N1_u = OfflineCalcMatrices_curr.N1_u;
N2 = OfflineCalcMatrices_curr.N2;
N3_c = OfflineCalcMatrices_curr.N3_c;
N3_x = OfflineCalcMatrices_curr.N3_x;
N3_u = OfflineCalcMatrices_curr.N3_u;
N3_v = OfflineCalcMatrices_curr.N3_v;
N4 = OfflineCalcMatrices_curr.N4;

%estimate state x
y_est = C*x_est0;
e = y - y_est;
xp = x_est0 + M*e;

W = repmat(yref,PH,1);

md_aug = repmat(md,PH,1);
g = g1*u_prev + g2*xp + g3*md_aug + g4*W;
fq = g;

%augment cost function with slack variable
[r_H,c_H] = size(H);
Hbar = [H zeros(r_H,1);zeros(1,c_H) 2*ro_epsilon];
fbar = [fq;0];
```

Figure A.3. calculateOnlineMatrices Function.

```
%matrices for ineq constraints
N1 = N1_c + N1_u*u_prev;
N3 = N3_c + N3_x*xp + N3_u*u_prev + N3_v*md_aug;

M_in = [M1;M2;M3;M4];
N_in = [N1;N2;N3;N4];


[L_i,p_i] = chol(Hbar,'lower');
Linv = L_i\eye(size(Hbar,1));
A_in = -1*M_in;
B_in = -1*N_in;
A_eq = [];
B_eq = zeros(0,1);
opt = mpcqpsolverOptions;
iA0 = false(size(B_in));
[zbar_opt,status] = mpcqpsolver(Linv,fbar,A_in,B_in,A_eq,B_eq,iA0,opt);
% U_opt = quadprog(Hbar,fbar,A_in,B_in,A_eq,B_eq);
% U_opt = quadprog(H,fq);
% U_opt = quadprog(Hbar,fbar);

[r1,c1] = size(B_u);
if isempty(zbar_opt)
   zbar_opt = [zeros(c1*CH,1) 0];
end

% optimal move
mv = u_prev + zbar_opt(1:c1,1);

x_est1 = estimateStates(y,x_est1_prev,md,mv,OfflineCalcMatrices_m1);
x_est2 = estimateStates(y,x_est2_prev,md,mv,OfflineCalcMatrices_m2);
end
```

Figure A.3. calculateOnlineMatrices Function (cont.).

State estimation function called from calculateOnlineMatrices function is:

```
function x_est = estimateStates(y,x_est0,md,mv,OfflineCalcMatrices)

L = OfflineCalcMatrices.L;
A = OfflineCalcMatrices.A;
B_u = OfflineCalcMatrices.B_u;
B_v = OfflineCalcMatrices.B_v;
C = OfflineCalcMatrices.C;

%estimate state x
y_est = C*x_est0;
e = y - y_est;

x_est = A*x_est0 + B_u*mv + B_v*md + L*e;
end
```

Figure A.4. estimateStates Function.