Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TECHNION
Israel Institute
of Technology

Automatic Control Laboratory

DEPARTMENT OF
AEROSPACE ENGINEERING

**Masterthesis**

# Model Predictive Control of Micro-Gas Turbines

Fabian Ulmer
2nd May, 2023

**Advisors**
Assoc. Prof. Daniel Zelazo
Assoc. Prof. Beni Cukurel
Michael Palman
Prof. Dr. Florian Dörfler

# Abstract

The dynamic performance of micro-gas turbines (MGTs) is of paramount importance in many of their applications, whether they are used as small to medium size power generators or for propulsion of small airplanes and unmanned aerial vehicles (UAVs). For heavy-duty gas turbines, advanced control algorithms have been widely used for more than a decade [13]. The electronic control units (ECUs) available for MGTs however are mostly PID-based and are closed for modification. While there is some literature available about model predictive control (MPC) of MGTs, most of the developed controllers found in literature have only been tested using simulation and not undergone hardware-in-the-loop (HIL) tests [11, 31, 32]. This thesis addresses this gap by designing a testbed for MGT controllers, establishing a methodology to design an MPC controller for MGTs and testing the controller on the AMT Olympus HP MGT.
The testbed is built using a dSPACE MicroLabBox real-time controller and custom circuitry. The controller is implemented with MathWorks Simulink. With this setup, a methodology is developed to design an MPC controller. Using open-loop measurement data from the MGT, linear, time-discrete state-space models are fitted using Numerical algorithms for Subspace State Space System IDentification (N4SID). To manage the non-linearity of the engine, gain scheduling is used. Five linear state-space models are identified for different operating regions and an MPC controller is designed for each.
These controllers are tested and evaluated both in simulation and on the physical MGT. The simulation is compared with the experimental data and performance metrics are presented for both. Some mismatches between simulation and experiments can be identified, but generally a good agreement is found. While the individual controllers perform acceptably within their operating region, the performance of the composite controller for reference trajectories spanning several operating regions exhibits some problems and needs further tuning.

# Preface

This master thesis was completed during an exchange semester at the Technion in Haifa, Israel between October 2022 and May 2023. With great gratitude I look back on how I have been received and welcomed with open arms by the people in the Turbomachinery and Heat Transfer Laboratory, by my host parents Lea and David and countless other friends. You made my stay in Israel worthwhile and pleasant.
A great thank-you goes to Prof. Florian Dörfler for agreeing to supervise this thesis from the side of ETH and thus making this thesis possible. Another great thank-you I owe to Prof. Dan Zelazo, who invited me to the Technion, came up with the project and tutored me throughout the project. The greatest thank-you though has to be attributed to Prof. Beni Cukurel who had to bear with me in his office for half a year.
Also Michael Palman and Dr. Arkady Lichtsinder deserve special mention for helping me with putting up the mechanical test stand. My friends Acar and Felix, I hope for more opportunities to come to celebrate friendship. But without my soon-to-be wife the whole time in Israel would not nearly be as memorable.

Haifa, May 2023

# Contents

# List of sections with practical instructions

**Sections with practical instructions**

Throughout this thesis, a number of boxes in this style are included. They contain practical information, mainly intended for people continuing to work on the project. For quick reference, those sections are listed on this page.

# Nomenclature

### Quantities

| | |
|---|---|
| $f$ | Frequency [Hz] |
| $U$ | Voltage [V] |
| $N_{RPM}$ | Engine rotor speed [revolutions per minute (RPM)] |
| $\widetilde{N}_{RPM}$ | Engine rotor speed normalized as described in Section 3.1.4 (dimensionless) |
| $r$ | Reference setpoint for the engine rotor speed [RPM] |
| $\widetilde{r}$ | Reference setpoint for the engine rotor speed, normalized in the same way as $\widetilde{N}_{RPM}$ as described in Section 3.1.4 (dimensionless) |
| $T_{EGT}$ | exhaust gas temperature [°C] |
| $\widetilde{T}_{EGT}$ | exhaust gas temperature normalized as described in Section 3.1.4 (dimensionless) |
| $u$ | Control input to the fuel pump (dimensionless) |
| $Q$ | Fuel flow rate [L/min] |

### Math

| | |
|---|---|
| $\|\cdot\|$ | Euclidean or $L^2$ norm of a vector |
| $\widetilde{\cdot}$ | Normalized version of a scalar. See Section 3.1.4 for used normalizations. |
| NRMSE | Normalized Root Mean Square Error in percent (see Equation (3.2) for how it is calculated) |
| $\text{NRMSE}_{\text{ref}}$ | Normalized Root Mean Square Error metric for reference tracking in percent (see Equation (3.8) for how it is calculated) |

### MPC terminology

| | |
|---|---|
| $z_k$ | Quadratic programming (QP) decision at time step $k$ with optimal control inputs and slack variable |
| $c$ | Control horizon |
| $p$ | Prediction horizon |
| $J$ | Cost function |
| $J_y$ | Output reference tracking penalization |
| $J_u$ | Control variable penalization |
| $J_{\Delta u}$ | Control variable move penalization |
| $J_\varepsilon$ | Soft constraint violation penalization |
| $w^y$ | Tuning weight for the speed output reference tracking penalization |
| $s^y$ | Scale factor for the speed output |
| $V^y_{min}$, $V^y_{max}$ | Equal concern for relaxation (ECR) for the speed output minimum or maximum softened constraint respectively |
| $V^{EGT}_{max}$ | ECR for the exhaust gas temperature (EGT) maximum softened constraint |
| $w^u$ | Tuning weight for the control variable penalization |
| $w^{\Delta u}$ | Tuning weight for the control variable move penalization |
| $s^u$ | Scale factor for the control variable |
| $\varepsilon$ | Slack variable |
| $\rho_\varepsilon$ | Constraint violation penalty weight |

# Glossary

**A**

**ADRC** active disturbance rejection controler.

**C**

**CFD** computational fluid dynamics.
**CVA** Canonical Variate Algorithm.

**E**

**ECR** equal concern for relaxation.
**ECU** electronic control unit.
**EGT** exhaust gas temperature.
**ETHZ** Eidgenössische Technische Hochschule Zürich (Swiss Federal Institute of Technology Zurich).

**H**

**HIL** hardware-in-the-loop.

**M**

**MGT** micro-gas turbine.
**MPC** model predictive control.

**N**

**N4SID** Numerical algorithms for Subspace State Space System IDentification.
**NAMUR** Normen-Arbeitsgemeinschaft für Mess- und Regeltechnik in der chemischen Industrie (User Association of Automation Technology in Process Industries).
**NRMSE** Normalized Root Mean Square Error.

**O**

**OEM** original equipment manufacturer.

**P**

**PCB** printed circuit board.
**PID** Widely used control loop mechanism that calculates the control signal using proportional, integral and derivative versions of the error signal.
**PWM** pulse-width modulation.

**Q**

**QP** quadratic programming.

**R**

**RES** renewable energy source.
**RMS** Root Mean Square.
**RPM** revolutions per minute.
**RTI** Real-Time Interface.

**S**

**SID** subspace identification (*see glossary*).
**SIL** software-in-the-loop.
**Subspace identification (SID)** A method in control theory, that "aims at identifying linear time invariant (LTI) state space models from input-output data"[30]..

**U**

**UAV** unmanned aerial vehicle.

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Micro-gas turbines (MGTs) are small engines that rely on the same operating principles as larger gas turbines, which have powered aircraft and produced electricity for decades. They are particularly interesting for use in unmanned aerial vehicles (UAVs) and for energy generation [4, 17, 19]. The use of MGTs for electricity and heat production has received particular attention in light of the increasing share of renewable energy sources (RESs) in the power mix [16, 19]. Due to the inherent variability of the output power of these RESs, the conventional power sources are subject to more restrictive dynamic requirements. Gas turbines are a particularly interesting option as a backup power source in this context due to their ability to quickly regulate the electric power output [17].
To optimize the performance of gas turbines, the control system is of paramount importance. It has been demonstrated repeatedly that advanced control mechanisms might improve the performance of gas turbines [11, 13, 17, 31, 32]. According to [13], model predictive control (MPC) is widely used in practice in heavy-duty aircraft gas turbines, because MPC controllers can handle multi-variable constraints conveniently and can remedy disturbances and unmodeled dynamics. For the smaller MGTs however, only little literature is available that evaluates the controller designs not only in simulation but also using hardware-in-the-loop (HIL) tests on real engines.

### Gas turbine modeling

In turbomachinery literature, engine modelling traditionally focuses on performance analysis. With the help of computational fluid dynamics (CFD) or empirical performance models, the static performance of engines is simulated and predicted. An example of this can be found in [10] where the performance predictions of various techniques for the AMT Olympus HP MGT (the same engine which is used in the present project) are evaluated and compared.
In contrast, in literature from the field of control systems (such as the present thesis), models generally refer to dynamic models, which are used to predict the system outputs to given system inputs. These models can be used in control algorithms. Control algorithms aim to calculate system inputs, that optimize the system output w.r.t a specified metric. Usually, one or several outputs are to follow a given reference trajectory, respecting system constraints.
Dynamic models for MGT are found to be derived with the usual techniques such as modeling from first principles or data-driven techniques. In [5], a high-order, non-linear dynamic model is developed based on physical laws and an appropriate PID-based controller is also simulated. The model is used to predict and simulate the power output of a Capstone C65 MGT in off-design and transient conditions. The simulation results are compared to experimental data. In [32], a linear model for a MGT-based combined cooling, heating and power system is identified using a data-driven subspace identification (SID) method. White noise is applied to the inputs

of the system and the output data is recorded. Using the Numerical algorithms for Subspace State Space System IDentification (N4SID) algorithm described in [14], a state-space model is then automatically fitted to this input-output data. The paper reports that simulation results using the identified model match data from open-loop experiments on the MGT accurately.
An extension of the above work is found in [31], where the SID algorithm estimates the system model in an online fashion, using previous input-output data. The approach is validated on a 80 kW MGT simulator.

### Model predictive control for MGT

A pivotal problem control theory tries to solve is how the inputs of a system should be actuated, in order that its outputs follow a given reference. The controllers for MGTs generally aim to regulate the engine speed, the power output or the generated thrust depending on the application, using the system inputs which usually entail a fuel pump, fuel valves and/or additional valves and actuator of peripheral systems, depending on the examined gas turbine. Various control algorithms exist and have been used for this task. Traditionally, PID controllers are widely used to control gas turbines [11, 13]. While these provide comparatively easy tuning using well-established methods and a range of methods that aid the specification of performance guarantees, the relatively few parameters that can be tuned for this algorithm might indicate that not the full performance of the engines is exploited. In fact, it was shown repeatedly that advanced control algorithms outperform PID controllers [11, 17, 32]. Often times, these algorithms make use of a predictive model (as described in the section above), to optimize the input trajectory.
The use of MPC for controlling gas turbines has been suggested repeatedly [13, 17, 31, 32]. In [17], a linear MPC controller for a heavy-duty gas turbine for power generation is designed using a fourth-order linearized model obtained with a first principles approach. The performance of the controller is compared to two PID controllers and an alternative model-based approach (feedback linearization control) found in literature, the MPC controller is found to outperform the other approaches.
In [31, 32], an MPC controller for a simulated, 80 kW, MGT-based combined cooling, heating and power system is designed, based on a model obtained through variations of N4SID. Simulations show a clear benefit of MPC over a reference PID controller.
A project at the Institute for Dynamic Systems and Control at Eidgenössische Technische Hochschule Zürich (Swiss Federal Institute of Technology Zurich) (ETHZ) from 2011 aimed at building a teststand for a power-generating MGT, for which an MPC controller was to be designed. The documentation still available from the institute suggests that the project was abandoned due to frequent hardware failures, no significant results could be identified from the two reports obtained [6, 26].

### Contributions of this thesis

The literature reviewed above shows that several works have used MPC controllers to control MGTs. However, none of these works have validated the results using HIL tests. Reasons for this might be the risk involved in operating gas turbines with custom controllers: A failure might lead to high financial costs and potentially more physical damage. Also, the commercially available gas turbines and controllers are proprietary and closed to modifications.
This thesis addresses this issue. The goals are as follows: A framework shall be developed, which enables development of controllers for MGTs on their testing on physical engines. This includes the test stand, which interfaces the control system with the MGT, a real-time control system which runs the control algorithm and the software framework to implement the control algorithms. With this setup, a method shall be detailed to identify predictive models for MGTs,

evaluate them and design an associated MPC controller. The designed MPC controller is to be tested on the physical engine.
In this project, the following things were achieved:

- A test stand for a 230 N thrust MGT (AMT Olympus HP) is developed using a dSPACE MicroLabBox real-time controller. The developed setup allows for quick design and evaluation of new control strategies on real engines, and has been designed with extensions for other engine types in mind.
- A method how to identify a linear, gain-scheduled state-space model based on open-loop measurements using N4SID is proposed. The identified models of fifth or eleventh order respectively are evaluated.
- Therewith, a gain-scheduled array of five linear MPC controllers is designed.
- The controller is implemented using Matlab Simulink and dSPACE Real-Time Interface (RTI).
- The performance of this controller is evaluated using simulation and HIL testing on the test stand with the AMT Olympus HP engine.

The remaining documentation is structured as follows: In Chapter 2, the setup of the physical test stand and the software framework built in the scope of this thesis is introduced and described, both for documentation and for reference for future use and expansion of the setup. Chapter 3 introduces the theoretical methods used in this thesis and describes the proposed methodology how to identify the engine model from open-loop measurements and therewith designing the MPC controller. In Chapter 4 the resulting models and the performance of the controller in simulation and in HIL tests is presented and discussed, while Chapter 5 summarizes the results of this thesis and discusses possibilities for future work.

# Chapter 2

# Experimental Setup

Developing a hardware-in-the-loop (HIL) test stand for micro-gas turbines (MGTs), which enables to quickly design and test new controllers, is one of the main objectives of this thesis. The resulting setup with the physical parts and the software framework is described in this chapter. It should provide instructions for users of the test stand as well as document it for continued development.

## 2.1 Physical Setup

An overview of the components of the test stand is given in Figure 2.1. The host/control PC is used to issue commands, display measurements and record measurement signals. On the real-time controller, the program with the control algorithm runs. The output signals of the real-time controller control the power circuitry which drive the inputs of the MGT (fuel pump, starter motor, etc.). The signals of the sensors of the MGT (engine rotor speed, temperatures, fuel flow rates and pressure probes) are conditioned and fed back into the real-time controller. The components are further described in the following sections. An overall view of the physical test stand can be seen in Figure 2.2.

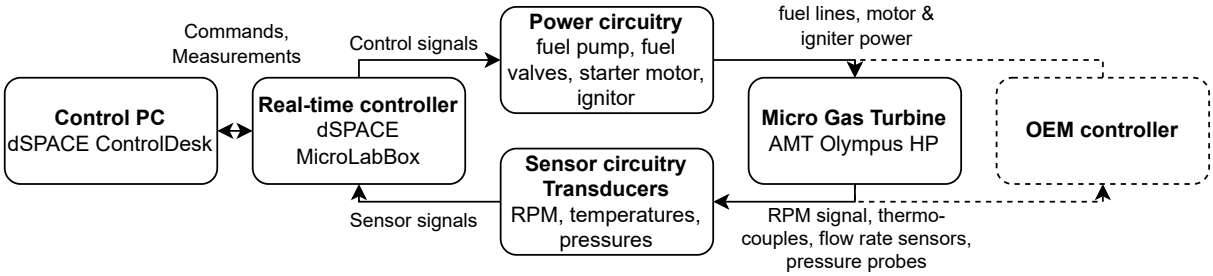



Figure 2.1: Schematic overview of the components of the test setup.

### 2.1.1 Micro-gas turbine AMT Olympus HP

The Olympus HP is a small gas turbine produced by AMT Netherlands. It is used primarily for hobbyist model jets, but has been used in a variety of applications, e.g. as a bringing home device for gliders or as a leaf blower [20]. This engine has a single stage centrifugal compressor with vaned diffusers and a single stage axial turbine mounted on a single shaft, with a standard inlet, an annular combustor and a convergent nozzle [10]. It is also available in a special university layout with additional measuring points, which makes it particularly useful for research and test

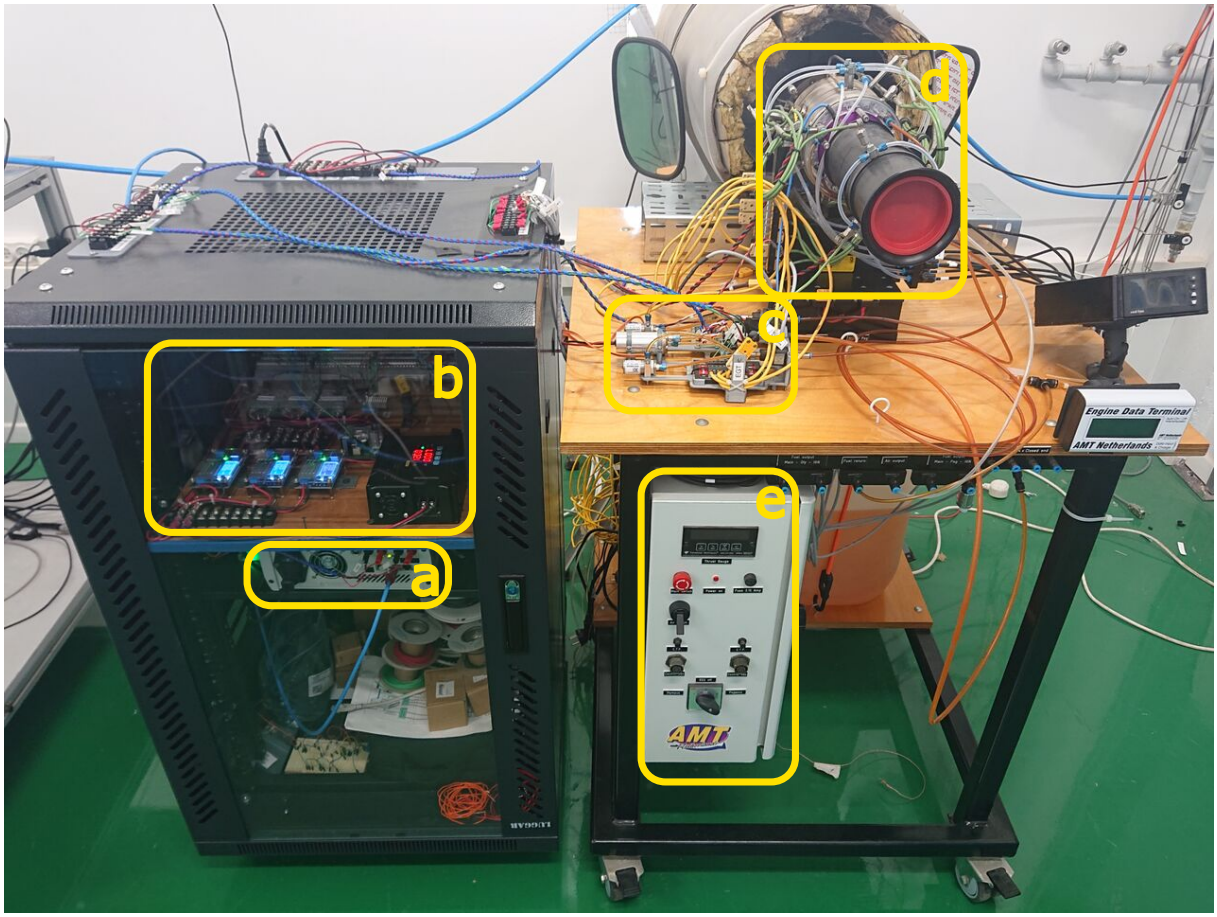



Figure 2.2: Overall view of the test stand.

a: Real-time controller; b: Power circuitry: c: Sensor circuitry and fuel pump; d: micro-gas turbine AMT Olympus HP; e: original equipment manufacturer (OEM) controller from AMT Netherlands

purposes. The engine used in this project is of this university layout. A cutout of the Olympus HP engine can be seen in Figure 2.4, some of its specifications can be seen in Table 2.1.
A simplified explanation of the working principle of a MGT can be found in Figure 2.3.

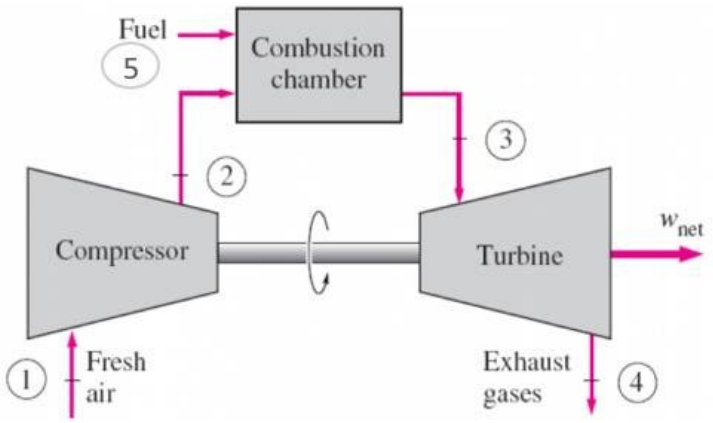



Figure 2.3: Simplified block diagram of the working principle of a gas turbine.

The working principle of a gas turbine can be put into simple terms as follows:
Atmospheric air is fed into the compressor (1) and compressed (2). In the combustion chamber, heat is added through burning fuel (5). The specific volume of the gas increases (3) and it expands through the turbine, where the exhaust gases exit the system (4). The turbine drives the shaft on which also the compressor is mounted, any excess energy can be used according to the application, be it to generate electricity or thrust [29].
Source: [7]

### Inputs and outputs of the Olympus HP

The Olympus HP has four inputs: The main fuel line, the ignition fuel line, a DC starter motor and a glow plug (igniter). During normal operation, only the main fuel line needs to be controlled, the others are used solely during the start-up and shutdown procedures, thus the system can be regarded as a single input system.
The normal version of the Olympus HP has two sensors: A type K thermocouple and an RPM sensor. The RPM sensor is an inductive sensor (Contrinex DW-AD-605-04), which senses the blades of the compressor. In the custom cable fitted to the sensor, some circuitry is inbuilt, which couldn't be identified completely. The sensor will output a signal with a frequency proportional to the speed of the engines rotor. Given that the compressor of the engine has 7 blades, the rotational speed of the engine rotor in revolutions per minute (RPM) (customary unit in the turbo machinery community, also used as the unit for the engine speed in the OEM controller) can be calculated as

$$N_{RPM} = f_{\text{RPM sensor}} \cdot \frac{60\,\text{s/min}}{7}. \tag{2.1}$$

The university layout of the Olympus HP features additional measurement points on five different axial stages (a schematic crosscut of the engine can be seen in Figure 2.5):

- 1: at inlet (temperature Tt1, static pressure Ps1)
- 3: behind diffusor stage (temperature Tt3, static pressure Ps3, total pressure Ps3)
- 4: before nozzle guide vanes (temperature Tt4, total pressure Pt4)

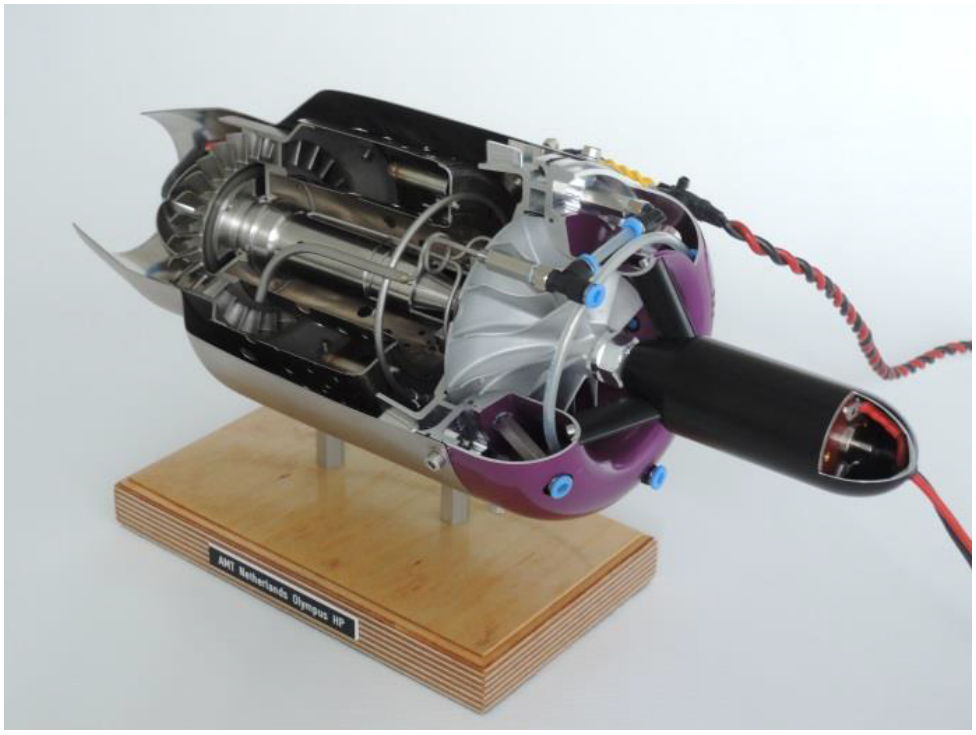



Figure 2.4: Cutout of an Olympus HP engine.

Source: [2] © 2020, AMT Netherlands

| | |
|---|---|
| Diameter | 131 mm |
| Length | 384 mm |
| Turbine weight | 2850 g |
| Thrust @ max. RPM | 230 N |
| Thrust @ min. RPM | 13 N |
| Maximum RPM | 108 500 RPM |
| Idle RPM | 36 000 RPM |
| Pressure ratio @ max. RPM | 3.8:1 |
| Mass flow @ max. RPM | 450 g/s |
| Normal exhaust gas temperature (EGT) | 700 °C |
| Maximum EGT | 750 °C |
| Fuel consumption @ max. RPM | 640 g/min |
| Fuel | JP-4/petroleum/Jet A1 |
| Oil | 4.5 % aeroshell 500 mixed with fuel |
| SN of the engine used throughout the project | NL2415 |

Table 2.1: Specifications of the AMT Olympus HP MGT used in this project [3]

- 5: after turbine wheel (temperature Tt5, standard temperature measurement EGT, total pressure Pt5)
- 6: at exhaust nozzle (temperature Tt6)

The standard EGT measurement is taken after the turbine wheel (at the same axial stage as Tt5).
All of the additional measurements are taken at three separate radial angles A, B and C, yielding a total of 15 different temperature measurement points. The pressure probes at the stages 1 and 3 are joined to one outlet, as their pressure difference is assumed to be small. At the stages 4 and 5, turbulences can lead to considerable pressure differences, thus the pressure lines must not be connected. This yields a total of 9 pressure signals. Signal circuitry and acquisition is explained in Section 2.1.3.
The Olympus HP has an inlet for compressed air, it is connected to a manual valve on the control panel of the OEM control panel and should be used to cool down the engine in case the controller fails or in a power outage situation [1].

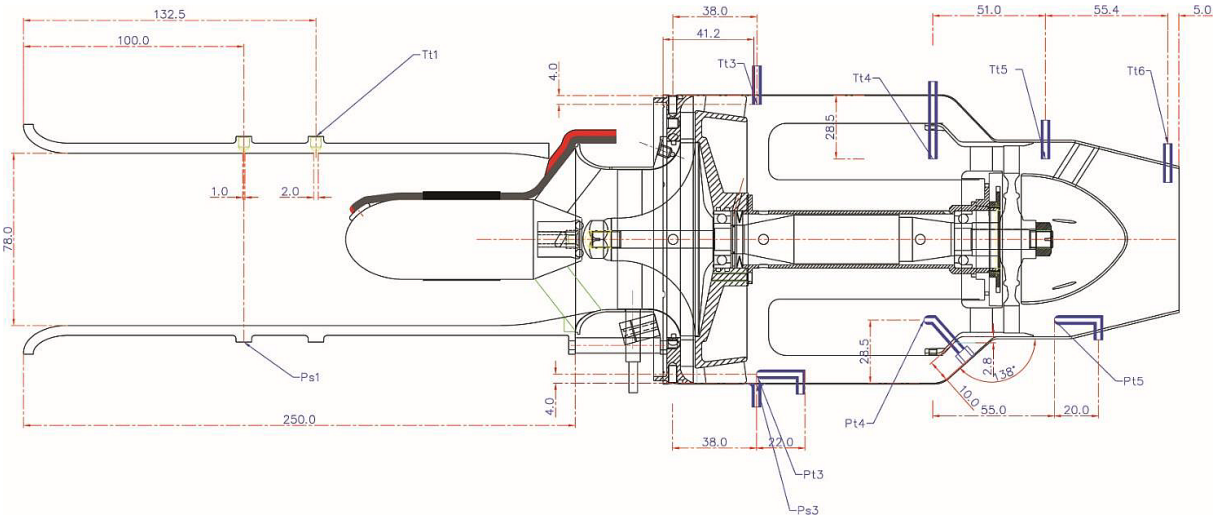



Figure 2.5: Crosscut schematic of the Olympus HP with the axial position of the additional measuring points.

Source: [2] © 2020, AMT Netherlands

### Flameouts

The prevention of flameouts is an important function of control systems for MGTs. Flameouts are extinctions of the flame in the combustor of the engine and can occur due to various reasons, among others due to incorrect air-fuel ratios in the combustion chamber because of sudden or inappropriate changes in the fuel supply rate [28]. Flameouts had occurred often during the experiments for this thesis. In open-loop experiments, the magnitude of changes in the fuel pump control input must be limited to prevent flameouts, for closed-loop control an effective method to prevent flameouts still has to be found.

## 2.1.2 Power circuitry

The power circuitry drives the fuel pump, the fuel valves, the starter motor and the igniter of the engine, using the control signals of the real-time controller as inputs. It is built on top of

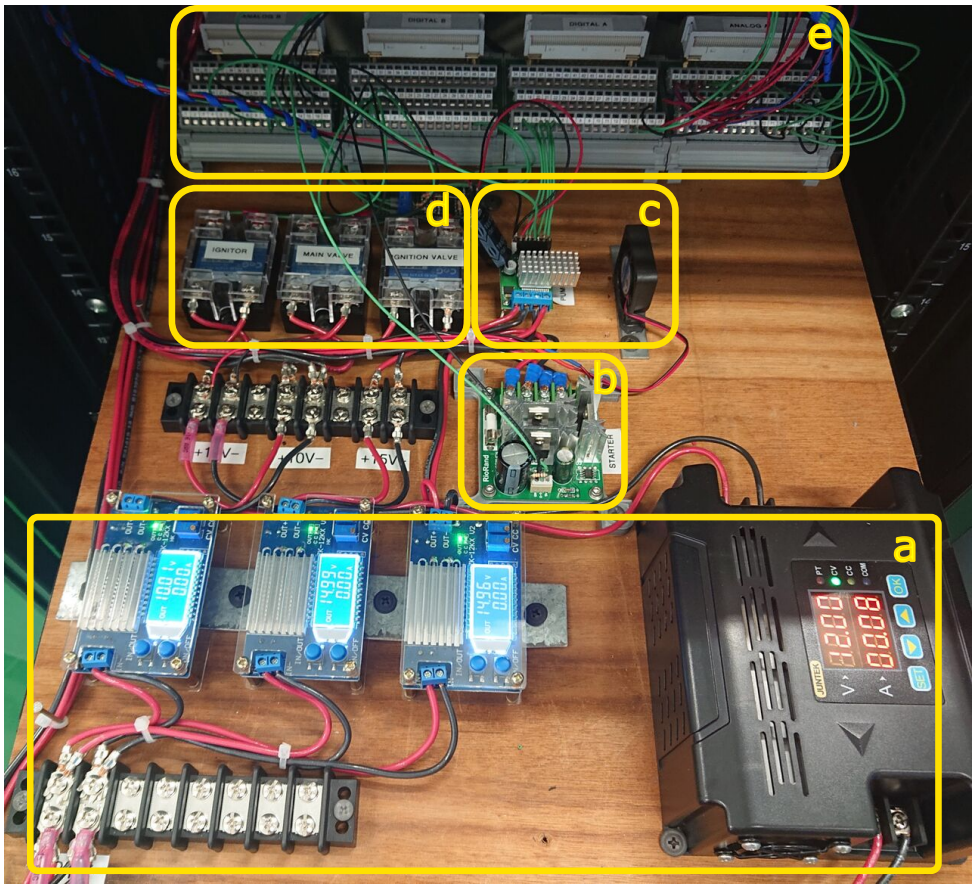



Figure 2.6: Power circuitry section of the test stand.

a: DC-DC converters; b: Starter motor driver: c: Pump driver; d: Solid-state relays for valves and igniter; e: Spring cage connectors of the dSPACE MicroLabBox
The setup is mounted on a wooden pull-out board inside the test rack.

a wooden board inside the test rack and leaves space for future extensions. Figure 2.6 shows a picture of the power circuitry, Figure 2.8 shows the top of the test rack with the connector terminals to the engine. In Figure 2.7 a connection diagram of the power circuitry is depicted, also the used parts are listed and considerations about the nature of the signals given.

### 2.1.3 Sensor circuitry and transducers

The signals of the RPM sensor, the thermocouples, the flow rate sensors and the pressure transducers from the engine need some conditioning to be compatible with the inputs of the dSPACE MicroLabBox. Each signal is shortly described here.
The sensor circuitry is directly powered from the power supply of the MicroLabBox, not by the power circuitry, as having two completely separated power regions for the sensors and the power section reduces the measurement noise.

#### RPM sensor

Reading out the RPM signal reliably turned out to be a major challenge. The RPM sensor (Contrinex DW-AD-605-04) in the Olympus HP is a 2-wire inductive sensor compatible with the NAMUR standard, it increases its resistance whenever a passing blade is sensed, therefore the voltage drops at the input of the comparator circuit, as can be seen in Figure 2.12. This voltage itself is relatively noisy, since the supply current to the sensor (which is on the same line as the signal) oscillates at the operating frequency of the sensor (the data sheet of the sensor indicates an oscillator frequency of 1400 kHz). Since every passing of the engine rotor produces a variation in the output signal of the sensor, it is clear that the frequency of the signal encodes for the engine rotor speed.
However, not only the frequency of the signal changes at different engine speeds, the shape and voltage level of the signal also varies, which can be seen from the two scope images in Figure 2.12, recorded at an engine speed of 45 kRPM and 90 kRPM respectively. This caused initial attempts to read out the RPM sensor to fail, as the circuits would work well at one engine speed but not for others.
Several circuits were tested to reliably get readings over the whole speed range. The circuit that worked well and was used for the experiments in this thesis can be seen in Figure 2.11, it is an inverting comparator with hysteresis. This hysteresis has to be tuned with the two potentiometers, to adjust the voltage levels for switching the output. Refer to practical instructions section 2 for instructions how to tune it. The circuit was realized on a prototype printed circuit board (PCB), it can be seen in Figure 2.9.
The frequency of the output signal is measured by the MicroLabBox and the rotational speed is inferred using Equation (2.1).

**Tuning the hysteresis of the RPM circuit**

Measure the in- and output of the RPM comparator circuit with an oscilloscope. Make the engine turn slowly, e.g. by giving short blows of compressed air and observe the signal. Turn the 100 kΩ potentiometer to its maximum value, i.e. such that there is 100 kΩ resistance between OUT and TP and the hysteresis is minimal (refer to Figure 2.11 to identify the parts). Adjust the 10 kΩ potentiometer s. t. the switching voltage is around 8.4V. Then decrease the value of the 100 kΩ potentiometer and iterate, *until the circuit switches high at an input voltage of* 8.3 V *and switches low at* 8.5 V. Tuned like this, the RPM sensor worked reliably over the duration of the project and didn't need retuning.

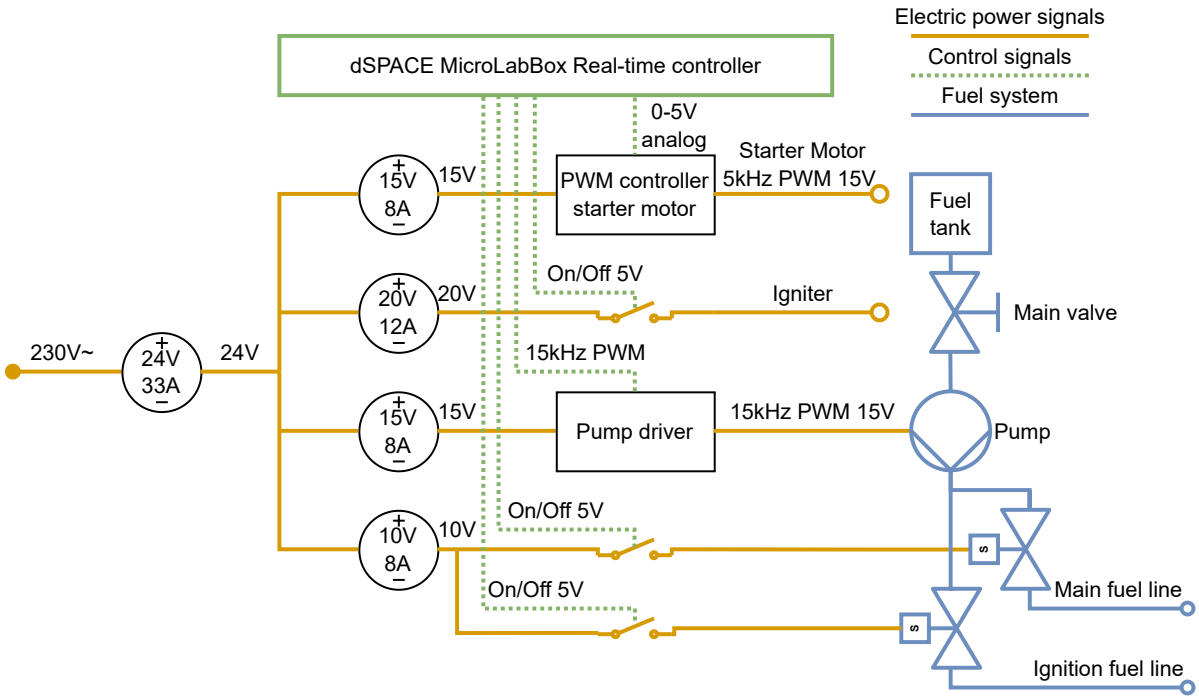



Figure 2.7: Connection diagram of the power section and the fuel system.

Below, the used parts are listed and some considerations are given.

- AC-DC 24V 33A: Meishile S800-24
  Main power supply with enough power reserve
- DC-DC 20V 12A: Juntek DPM8624
  Power supply for the igniter. Initially the igniter draws a very high current, as high as 12A were observed. When powering it through the weaker TKXEC DC-DC converters, the fuel was not ignited, thus a stronger power supply was used.
- DC-DC 15V/10V 8A: TKXEC 12KX-V3
  DC converters for the motor drivers and the solenoid valves. At the top of its operating range the pump uses close to the maximum power one of these DC converters can supply, thus the supply for the pump and the starter motor were separated. The solenoid valves indicate that they can operate in a wide operating range between 6 and 16V, it was observed though that they produce a considerable amount of heat on 15V supply, thus their voltage was reduced to 10V.
- PWM controller starter motor: RioRand 6V-90V 15A DC Motor speed controller
  Cheap general purpose motor controller with an analog 0-5V input, output PWM duty cycle proportional to input voltage, PWM frequency 5kHz. It was observed that the controller limits the slew rate and has a rise time of around 300ms. For the starter motor this controller proved sufficient.
- Pump driver: Polulu VNH5019 Motor Driver Carrier
  DC motor driver with direct PWM input with built-in protection features. Initially, the same RioRand motor controller as for the starter motor was used, but its delay and slew rate limitation quickly showed that a better motor driver was necessary, as it limits the bandwidth of any control loop. For the currently used VNH5019 motor driver a delay in the order of µs between the in- and output was measured, which is negligible. To be able to sustain peak pump power for extended periods, a heat sink and a cooling fan had to be added.
- Pump: AMT fuel pump Olympus HP
  The original OEM fuel pump was used. See Figure 2.9 for an image of the pump in the setup.
- Relays/switches: CG SSR-25DD Solid state relays
  Solid state relays, they switch the igniter and the solenoid valves from a 5V control signal coming from the MicroLabBox
- Solenoid valves: AMT solenoide valve fuel Olympus HP (high flow) for the main fuel line, AMT ignitor solenoide valve for the ignition fuel line respectively
  Also for the fuel valves, OEM parts were used. They can be seen in Figure 2.9.

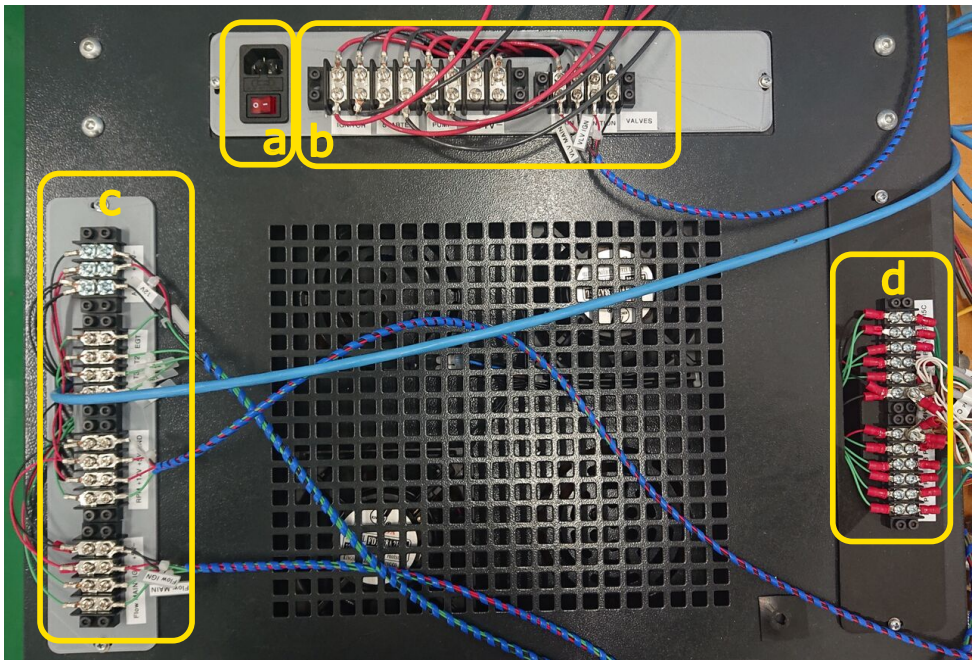



Figure 2.8: Terminal connectors on top of the test rack.

a: Main power connector and switch; b: Output terminals; c: Sensor input terminals; d: Pressure sensor input terminals

### Thermocouples

For reading the thermocouples, a 4-channel EGT-K thermocouple amplifier from REVELTRONICS is used. For a temperature range of 0 °C to 1250 °C it outputs a voltage in the range of 0 V to 5 V. This voltage is read by analog inputs of the MicroLabBox and the temperature is inferred using the following output equation of the thermocouple amplifier, which is found in its data sheet:

$$T_{\circ\mathrm{C}} = 250\,^{\circ}\mathrm{C/V} \cdot U_{\mathrm{out}} \tag{2.2}$$

During this project, the following thermocouples were connected to the channels of the amplifier and recorded by the real-time controller for most of the experiments:

| Amplifier channel | Thermocouple on the Olympus HP | Stage in turbine |
|---|---|---|
| 1 | EGT | after turbine wheel |
| 2 | Tt1A | at inlet |
| 3 | Tt4A | before nozzle guide vanes (after the combustor) |
| 4 | Tt5A | after turbine wheel (additional measurement on same stage as EGT) |

This mapping can be changed by replugging the cables. In the measurement files taken during this project, the mapping is also noted in the comment field.

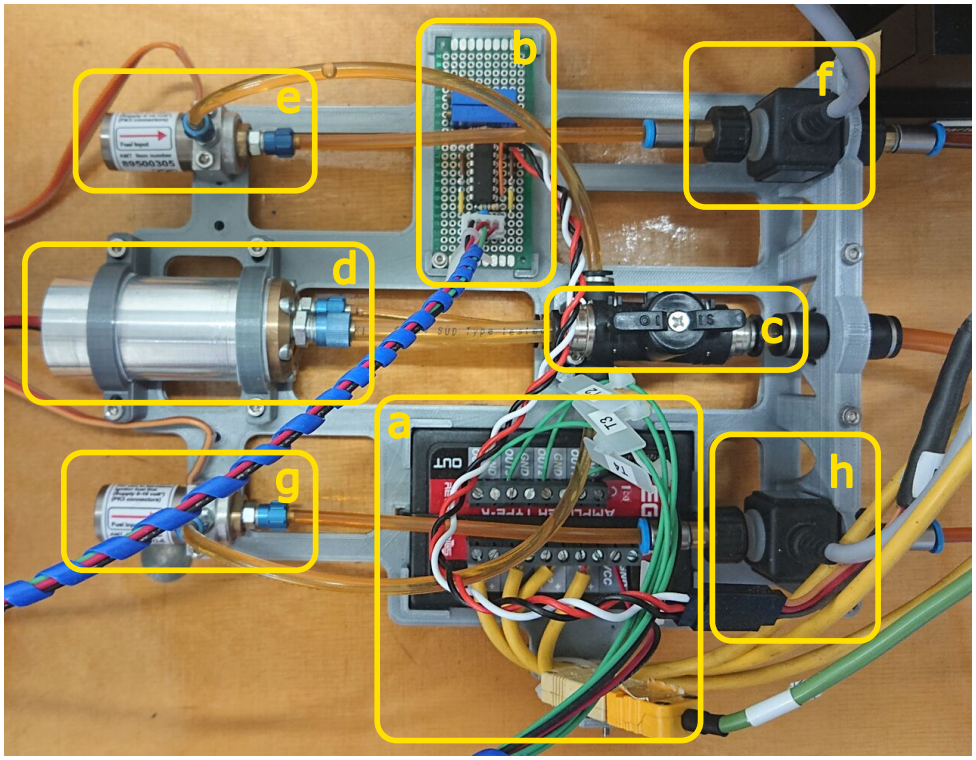



Figure 2.9: View of the sensor circuitry and the fuel system.

a: Thermocouple amplifier; b: Custom RPM sensor comparator circuit; c: Main valve; d: Fuel pump; e: Main fuel solenoid valve; f: Main fuel flow rate sensor; g: Ignition fuel solenoid valve; h: Ignition flow rate sensor
The whole setup is nicely mounted on a 3D-printed fixture, thanks to Michael

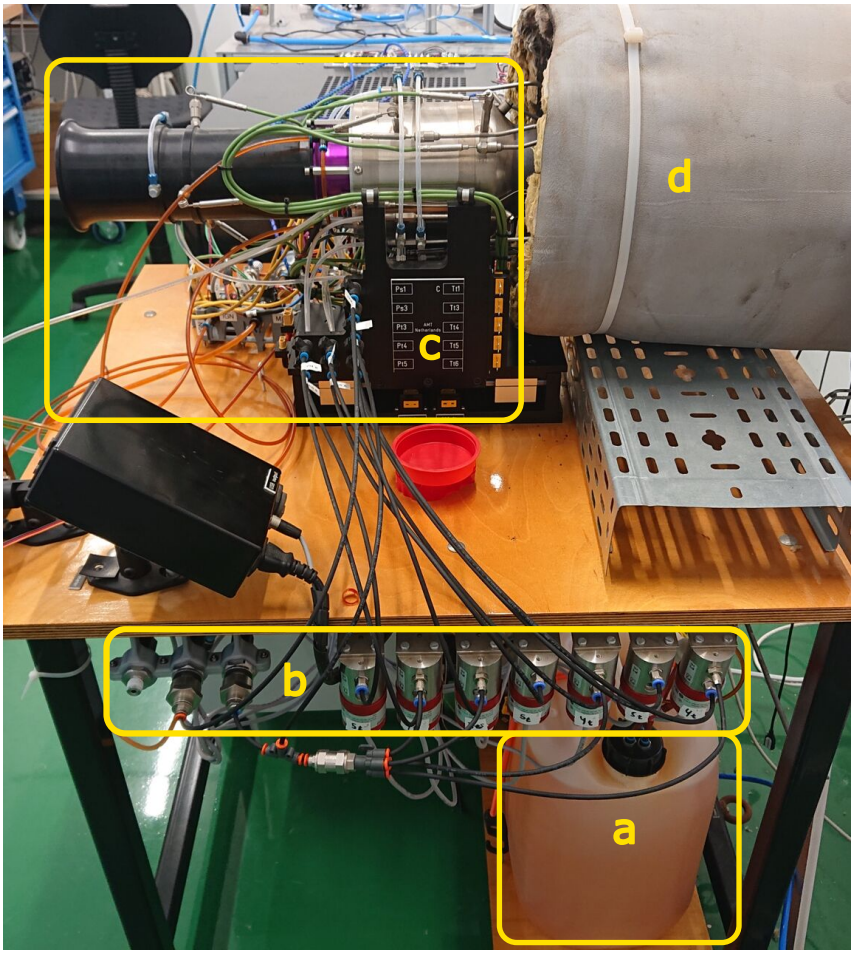



Figure 2.10: Side view of the test stand.

a: Fuel tank; b: Pressure transducers; c: MGT AMT Olympus HP; d: Exhaust duct, leads outside through a hole in the wall

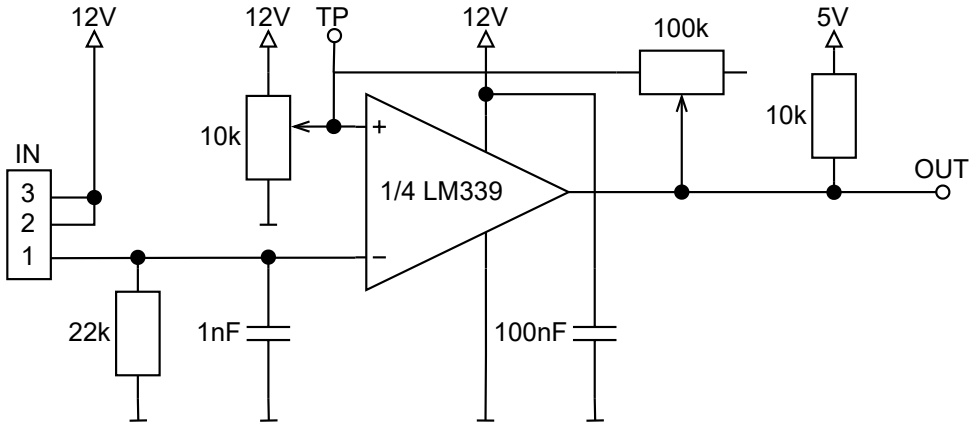
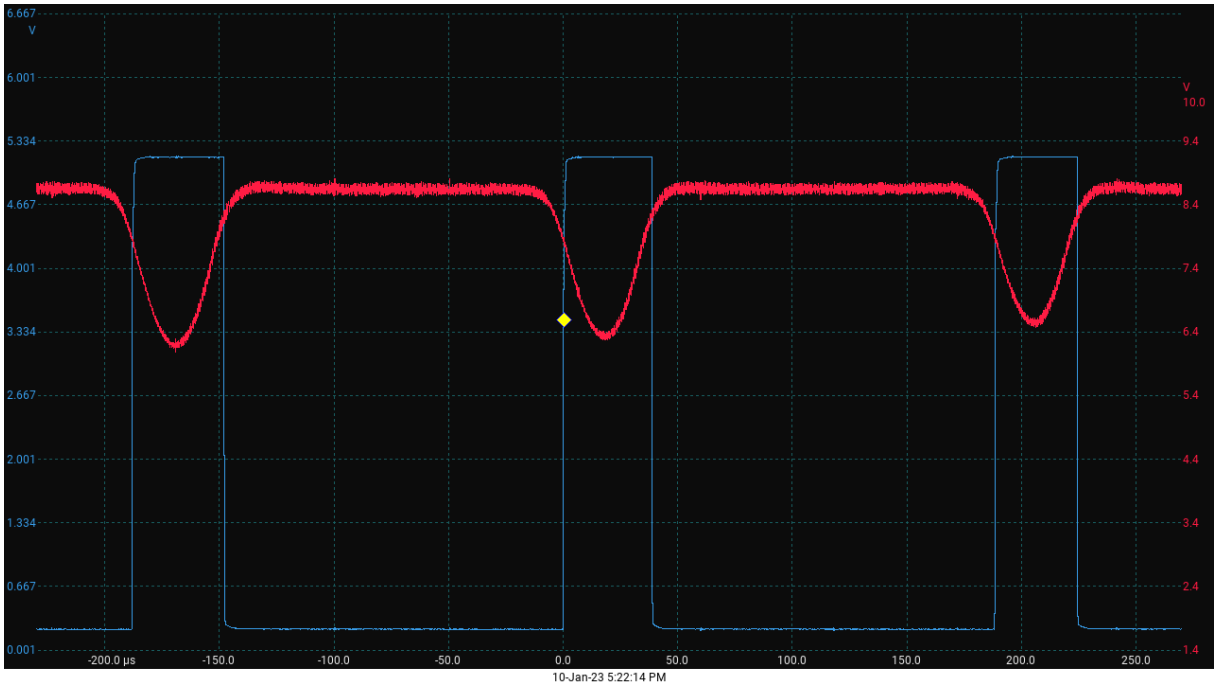
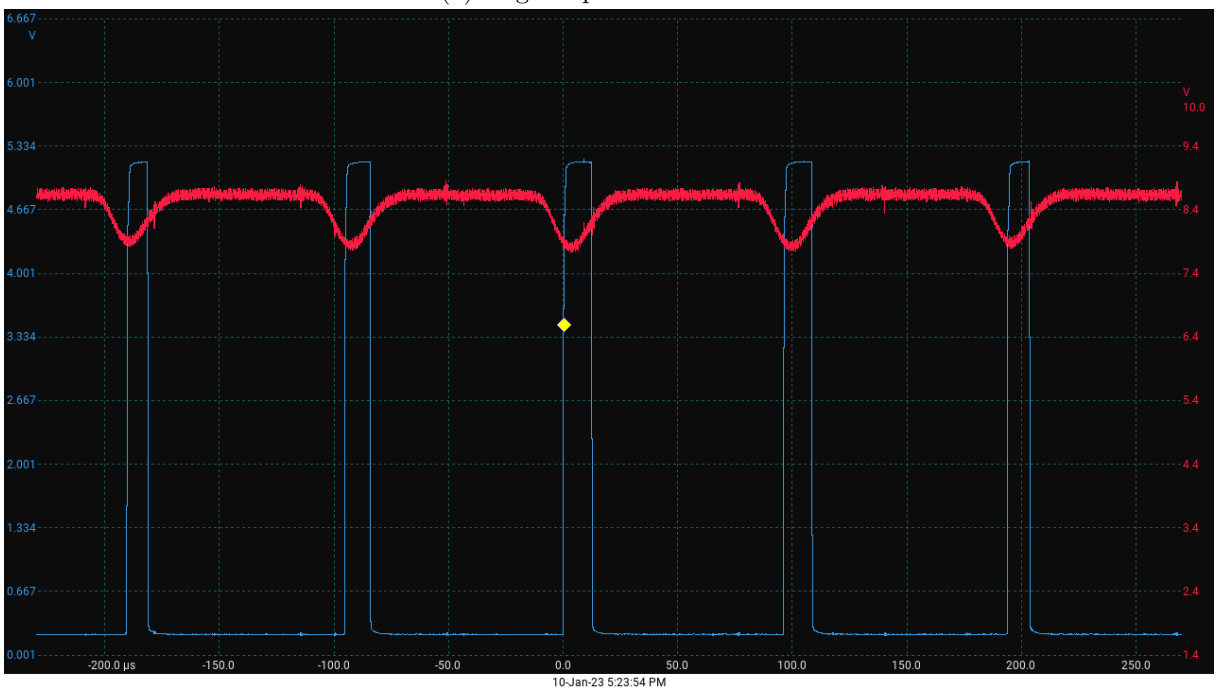



Figure 2.11: Custom circuit for converting the signal from the RPM sensor to a 5V TTL signal.




(a) Engine speed of 45kRPM




(b) Engine speed of 90kRPM

Figure 2.12: Scope images of the RPM signal at different engine speeds.

Channel 2 (red): Signal from the RPM sensor, input to the RPM comparator circuit seen in Figure 2.11
Channel 1 (blue): Output of the RPM comparator circuit
The hysteresis can be seen as the voltages for switching high and low differ.

| Signal | Transducer model | Range |
|---|---|---|
| P0 | 528.812102L311 | 0 bar to 1.6 bar |
| Ps1 - P0 | 528.901108L811 | −1 bar to 0 bar |
| Pt3 - P0 | 528.915108L811 | 0 bar to 4 bar |
| Pt3 - Ps3 | 692.9121010A1W | 0 bar to 0.5 bar |
| Pt3 - Pt4A/B/C | 692.9121010A1W | 0 bar to 0.5 bar |
| Pt5A/B/C - P0 | 692.9161010A1 | 0 bar to 2.5 bar |

Table 2.2: List of the measured pressure signals, the used pressure sensors and their measurement range.

P0 denotes ambient pressure. Refer to Figure 2.5 for a schematic where the respective pressures are taken in the engine.

### Flow rate sensors

To measure the fuel flow rate in the two fuel lines, two IR-Opflow turbine flowmeters from TECFLOW, measurement range 1 are used. They are in the schematic in Figure 2.7 and can be seen in the picture in Figure 2.9. Their measurement range is 0.1 L/min to 2 L/min, they output a square wave with 36 000 pulses/L (duty cycle of 50 %), thus the flow rate can be inferred with the equation

$$Q = f_{\text{Output flow rate sensor}} \cdot \frac{60\,\text{s/min}}{36\,000\,\text{L}^{-1}}. \tag{2.3}$$

The square wave encoded frequency signal needs some special consideration regarding the update frequency: The measured frequency and thus the estimated flow rate in the controller can be updated at most at every edge of the signal. For a flow rate of 0.2 L/min (about the fuel consumption at idle speed of the engine), which corresponds to a frequency of 120 Hz, the controller gets a new value only every $\frac{1}{120\,\text{Hz}\cdot 2} = 4.2\,\text{ms}$ (the factor of two is because the measurement value can be updated both at the rising and the falling edge of the signal). Given that this signal is still quite noisy and should be filtered, this limits the bandwidth of a control loop using the flow rate signal considerably.

### Pressure transducers

The pressures in the engine are measured with Huba Control pressure transmitters. Table 2.2 lists the acquired pressure signals, the used transducers and their measurement range. P0 (ambient pressure) is measured with an absolute pressure sensor, all the other sensors are relative pressure sensors and measure either the pressure w.r.t. ambient pressure or w.r.t to another pressure. All pressure sensors output a proportional signal in the range of 0 V to 10 V, which is then fed into an analog input of the MicroLabBox.
The pressure transducer readings were verified using an Additel 681 pressure gauge and an Additel 912A low pressure test pump. It was found that all transducers except the transducer for Pt5A delivered accurate readings over the whole measurement range with a maximum deviation of ±10 mbar. However, the value for Pt5A was found to deviate considerably from the reference value and the readings of the same sensor models (deviation of −45 mbar at an absolute pressure of 2 bar), this value should thus not be used.
The pressure transmitters are powered from the 24 V main power supply, they can be seen in Figure 2.10.

### 2.1.4 Real-time controller dSPACE Microlabbox

The dSPACE MicroLabBox is a compact prototyping unit featuring a dual-core 2 GHz real-time processor, a dedicated processor for communicating with the host computer, a Xilinx FPGA, 1 GB DRAM memory, Ethernet interfaces for communicating with the host computer or with other real-time units, 32 analog inputs (−10 V to 10 V), 16 analog outputs (−10 V to 10 V), 48 bidirectional digital channels with configurable voltage level and several other interfaces [8]. In this project, the front-panel variant of the MicroLabBox is used, with external I/O connector terminal blocks (they can be seen in Figure 2.6).
Additionally, the MicroLabBox features a 12 V sensor supply output, sourcing a maximum of 3 W/250 mA, and a variable output with an adjustable voltage of 2 V to 20 V (maximum 1 W/200 mA). In the scope of this project, the variable output is set to supply 5 V. These supplies are used to power the sensors and sensor circuitry.
The MicroLabBox can be programmed with the dSPACE Real-Time Interface (RTI), an extension for Simulink Coder, that compiles Simulink models to run on the MicroLabBox. This allows for a quick design cycle, implementing new functionality or controllers in Simulink and running the models seamlessly on the real-time hardware.
The digital channels can output and measure pulse-width modulation (PWM) signals with 10 ns resolution. As the frequency measurement capabilities of the MicroLabBox are very important in the scope of this project, a measurement series was conducted to characterize the response of the frequency measurement. It was found that the MicroLabBox delivers accurate readings after the first complete period of the input signal, over all tested duty cycles.

### 2.1.5 Control PC

The host or control PC is connected to the MicroLabBox via Ethernet cable. From there, variables of the real-time program running on the MicroLabBox can be set/read. In the scope of this project, this can be for instance reading, displaying and recording measurement values as the engine speed or EGT, the operating state of the controller etc., or setting reference values and changing the operating state. It is important to note however, that the MicroLabBox can run independently of the host PC, no data must be exchanged with the host PC. Consequently, control loops implemented in the real-time software are entirely local to the MicroLabBox and don't involve delays introduced through data transmission or processing on the control PC.

### 2.1.6 OEM controller

To be able to reverse-engineer the startup and shutdown sequence of the original controller and to measure the performance of the OEM controller supplied by AMT, wiring, adapters and additional circuitry were made to be able to record the signals the controller outputs to the engine. Refer to practical instructions section 3 for instructions how to use it.

**Record signals of the OEM controller**

The wires and circuits necessary to connect the OEM controller should still be in the test cabinet. Refer to the Simulink model to identify the ports the individual signals need to be connected to if you have to reconnect some cables. The cables are labeled, insert the connectors for the output signals in between the connectors of the engine and the controller. The cables are already equipped with voltage dividers to ensure that the measured signals are within the measurement range of the inputs of the MicroLabBox. The inputs are measured differentially. For the RPM sensor there is a special Y-cable, which allows the RPM signal to be read both by the OEM controller and by the MicroLabBox. You need to retune the RPM comparator circuit though to find a setting that works for the required measurement range in this setup. The EGT thermocouple can't be measured in parallel with the OEM controller, measure any of the Tt5 thermocouples instead, they are at the same axial stage as the EGT probe. The OEM controller doesn't require the flow rate measurement, unplug the cable of the Olympus flow rate sensor and connect it to one of the flow rate sensor lines of the test stand. Nothing needs to be changed for the pressure measurement, the setup is the same as when the engine is run with the MicroLabBox as controller.

## 2.2 Software setup

The software for this project is built using Matlab, Simulink and dSPACE ControlDesk and compiled using the dSPACE RTI compiler (see Appendix C for information which software versions were used). The structure of the software is outlined in this section, also instructions on how to use it and continue development are given.

### 2.2.1 Structure of the main Simulink model

The main model (model `AMT_Olympus_HIL` in folder `uGT_MPC_controller`) was used to run the engine throughout this project. It is built modularly, the engine controller and the engine interface are in a library (folder `custom_libraries`) and can be reused in other models. Also, important parameters can be changed easily from the mask dialog of the subsystem, they are not hard-coded. The software projects are version controlled with git, the exact software state recordings are taken with can be reproduced.

#### Top-level model

The top-level model connects the single components together. There is a range of reference inputs to select from. Apart from the default manual reference (the user sets the reference speed in the ControlDesk interface), predefined reference trajectories can be applied: A triangle signal, staircase signal (series of discrete steps), step signal or an arbitrary reference signal which was generated and saved offline.
This reference is fed into the controller block. This calculates control signals, which are output to the physical engine in the engine interface block. The measurements are fed back to the controller block. Additionally, there is a block that calculates absolute pressure values from the differential pressure measurements and the runtime counter, which logs the time the engine has been running and how much fuel was used.
The top-level model runs at a fixed step-size of 0.5 ms, which means that the measurement signals are sampled at this rate. Other components of the model might run at a different sampling rate.

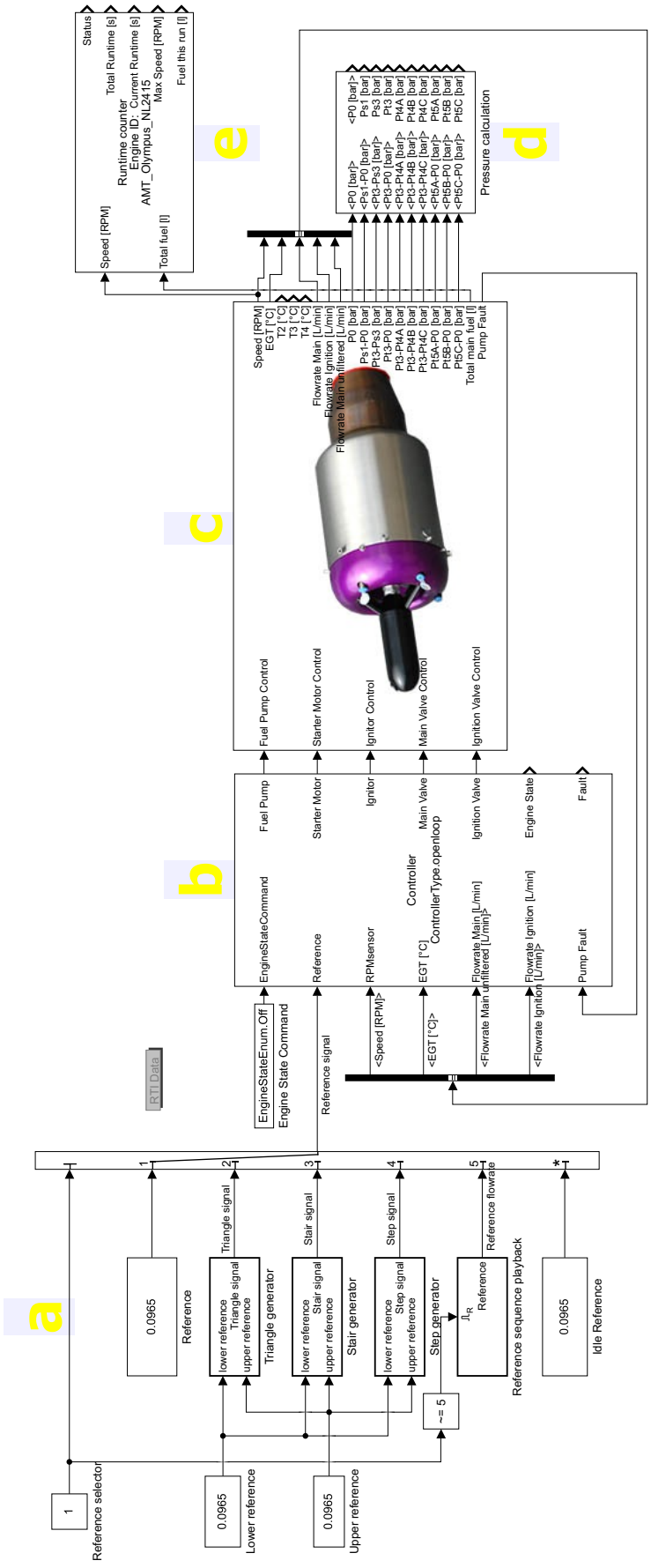



Figure 2.13: Top-level Simulink model.



a: Reference generation and selection; b: Controller block; c: Engine interface block; d: Pressure calculation block; e: Runtime counter

### Controller

The controller is the central component of the model. It has three parts, see Figure 2.14 for an image of the contents of the subsystem. The run logic state machine controls the operating state. During the startup and shutdown sequences it controls the control signals, when the engine is running it yields control to the control block. The control block holds the actual control algorithms. There are several control algorithms that can be switched in the parameters of the block, the current options are open loop, a preliminary PID controller and the main model predictive control (MPC) algorithm. Additionally, there is an independent safety block. If a measurement value exceeds the allowed range, the safety logic trips, it starts a forced shutdown and a corresponding fault message is displayed in the ControlDesk interface. Decoupling this block from the main control logic decreases failure probability and thus increases the safety of the system.
The controller subsystem has many parameters with which e.g. the startup sequence can be tuned or adapted to a different engine. A screenshot of the dialog with some parameters can be seen in Figure 2.15 and the startup and shutdown sequences are described in more detail in Section 2.3.

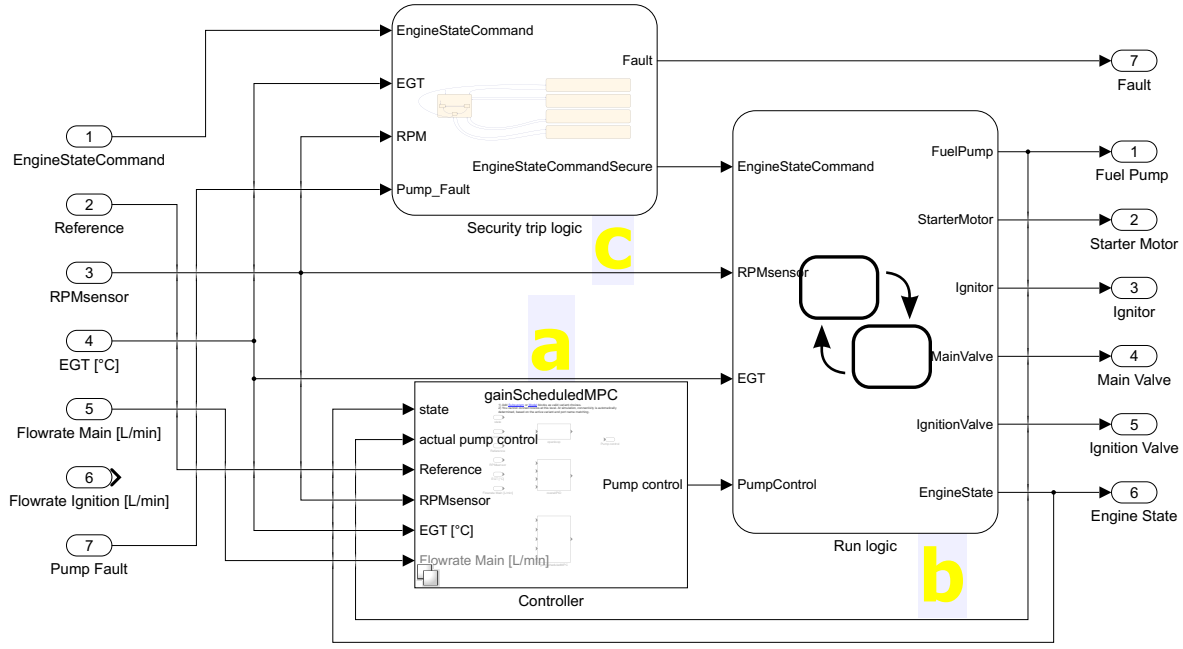



Figure 2.14: Engine controller subsystem.

a: Subsystem with control algorithm variants; b: run logic tate machine controlling operating state; c: Safety trip logic

### Engine interface

The engine interface connects the software to the physical world. It sends the control signals to the physical ports of the MicroLabBox. The measurement signals are acquired, optionally filtered and converted to physical values. The inputs can optionally be lowpass-filtered to reduce measurement noise. The cutoff frequency and the order of the filter can be specified in the parameters of the subsystem, the filter coefficients are then calculated at compile-time. The

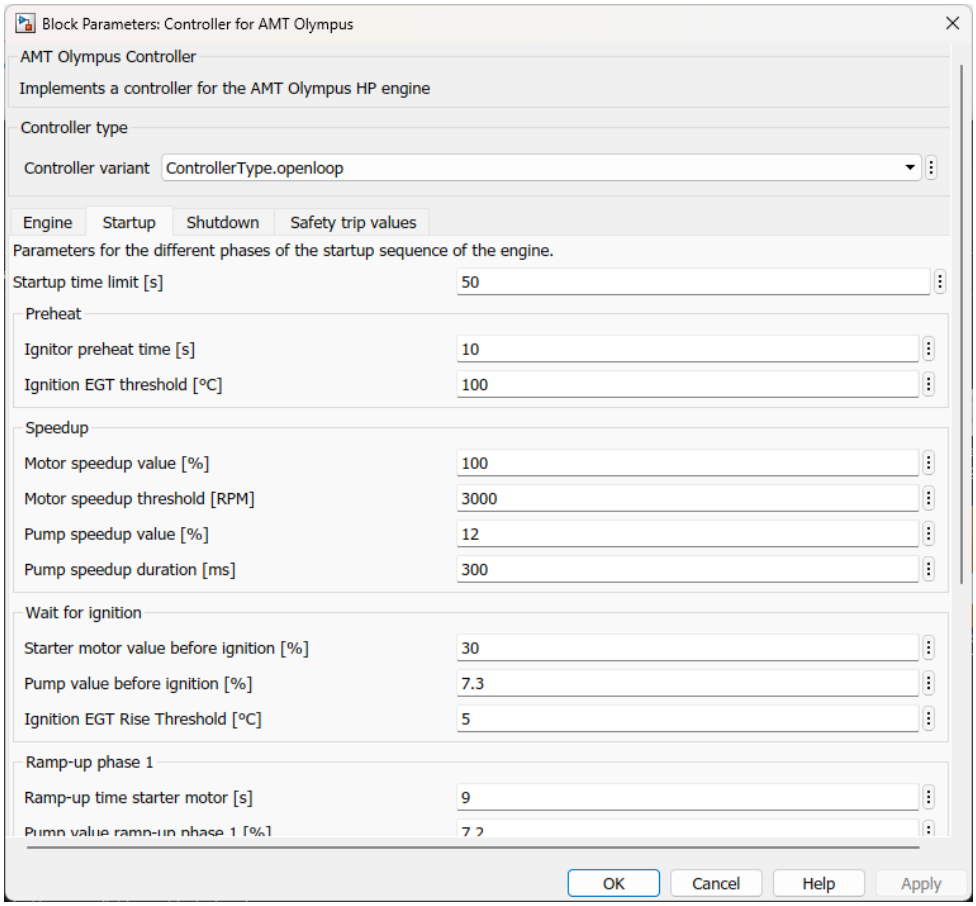



Figure 2.15: Screenshot of part of the parameter dialog of the controller subsystem.

preset for the cutoff frequency of the filter is the Nyquist frequency, i.e. the signal is not filtered.

### 2.2.2 Simulation model

To enable simulations of the generated controllers over the whole operating region and on the same hardware that is used to control the engine, a dedicated Simulink model was designed which has the identical blocks for reference generation and for the controller, while the engine interface block is replaced with a software simulation model (see Section 3.1.6). With this model, software-in-the-loop (SIL) tests can be run both directly in the Simulink environment or compiled and run on the dSPACE MicroLabBox real-time system.

**Guide to the different Simulink models in this project**

In the scope of this project, several Simulink models were developed and used. They are saved on the computer `AEJETLAB17` in the directory `C:\In-house ECU` or in the folder `Models` of the archive of this project.

All the Simulink models/software projects feature the following things:

- They share the subsystem blocks with the engine interface and the controller where applicable, through the shared git submodule `custom_libraries`.
- They contain a dSPACE ControlDesk project folder, which provides the interface through which the user can communicate with the model running on the real-time controller.
- They contain a Matlab project file, which sets up the required project paths.
- They are version controlled through a local git repository.

**Folder `uGT_MPC_controller`** The model `AMT_Olympus_HIL.slx` is the main model used for running the engine, refer to Section 2.2.1 for a description of this model.

The model `AMT_Olympus_simulation.slx` can be used to simulate the controller and the engine (either directly in Simulink or on the MicroLabBox, using the ControlDesk experiment `Simulation control stand` to interface it).

**Folder `manualControl`** Holds the engine interface without the controller. Can be used to test the power circuitry or to prime the fuel system, without actually starting the engine.

**Folder `EngineMeasurements`** Can be used just to record measurement signals from the engine, e.g. when the engine is run with the OEM controller.

**Folder `JetEngine_EndToEndSimulation`** Preliminary attempt at designing an engine controller by Dr. Arkady Lichtsinder, based on work by Liraz Dvorkind. Left there for completeness.

### 2.2.3 Controller design and compilation

The MPC controller is not designed directly in Simulink, but designed offline and saved as Matlab MPC objects in the file `custom_libraries/MPC_controller_objects_Olympus.mat`. This file is referenced by the Simulink models and the objects included in the compiled binary.

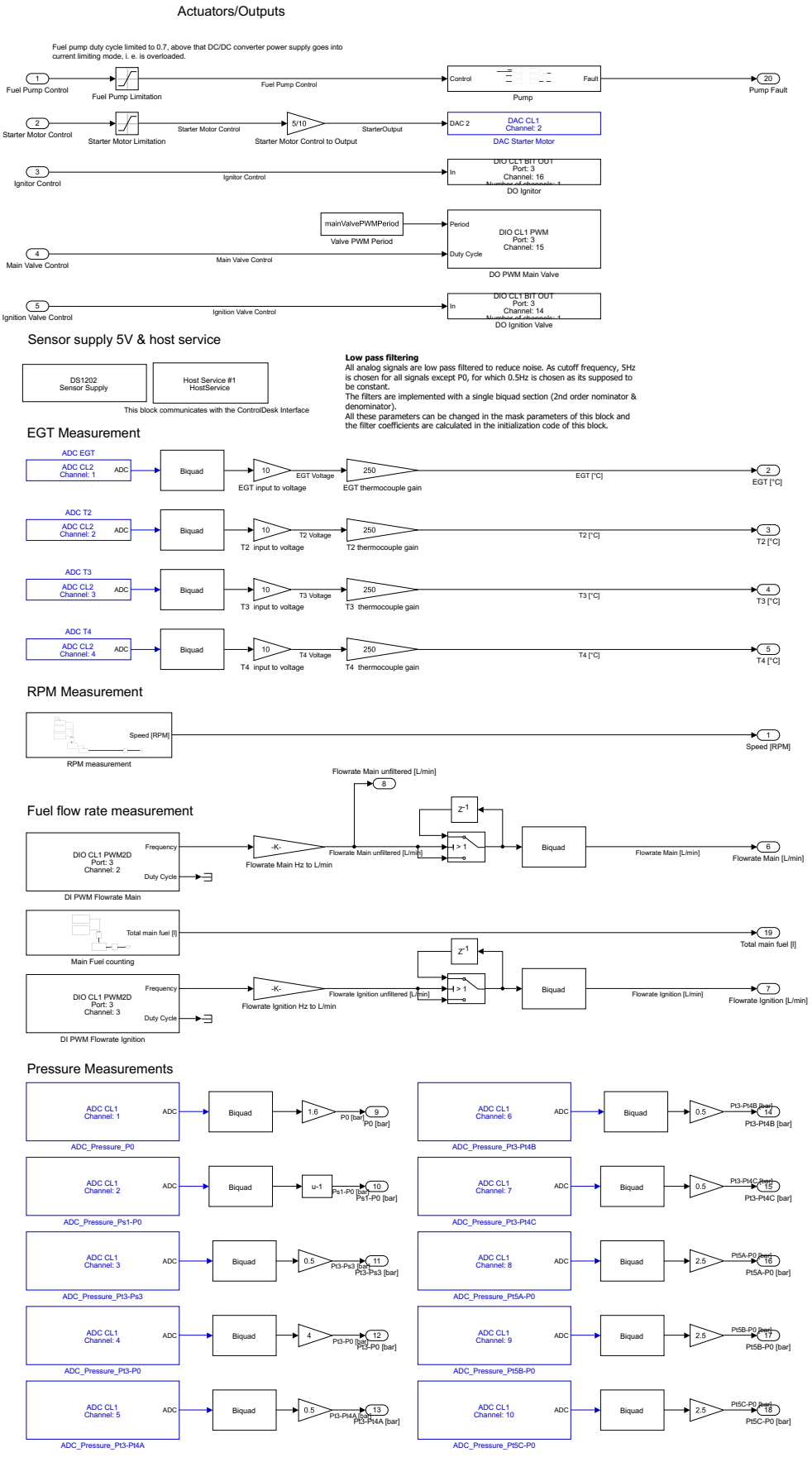



Figure 2.16: Engine interface subsystem.

**5 Compiling, managing and running software models**

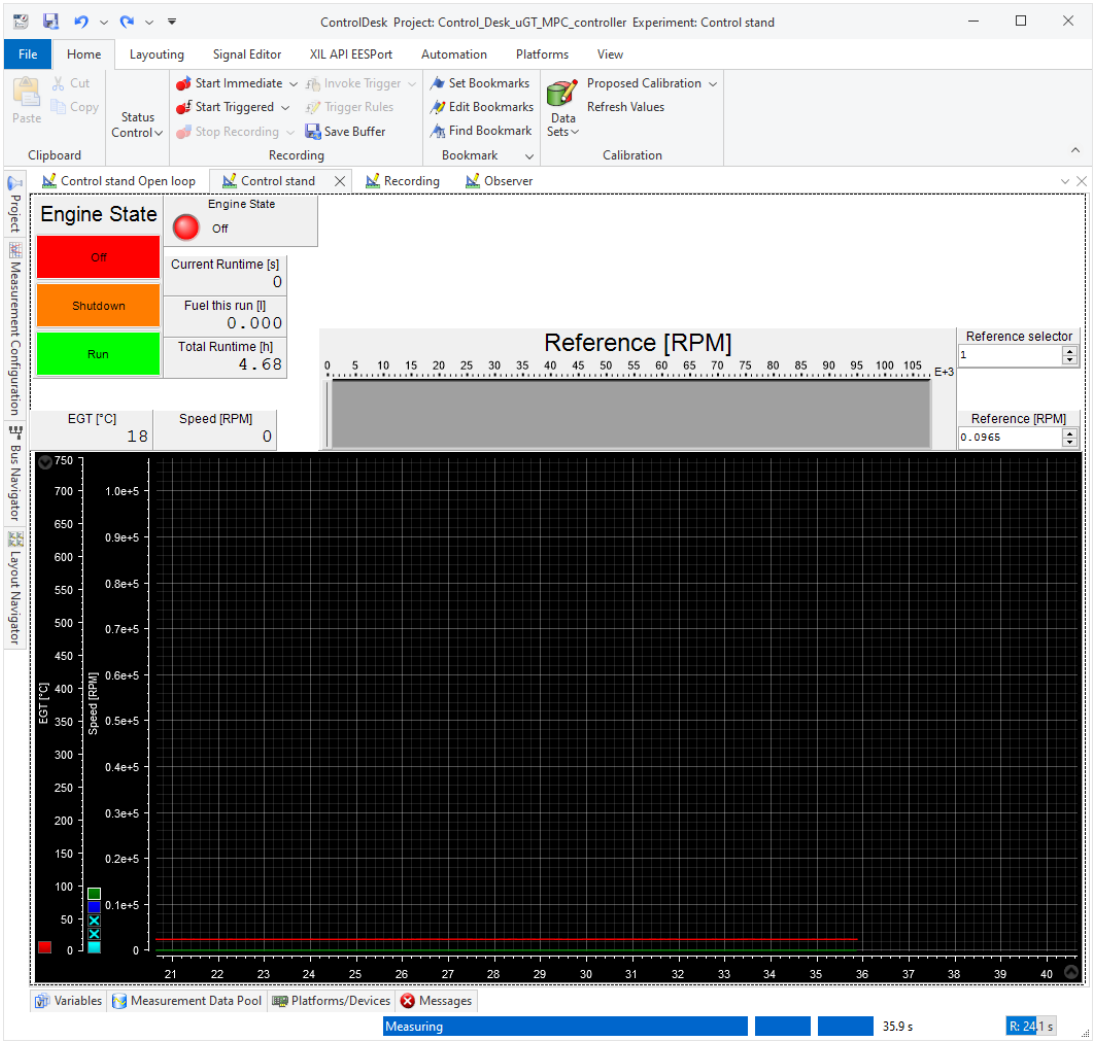
If you want to edit a model and run it on the real-time hardware, follow these steps:

1. Open the Matlab project in this folder (`.prj` file in the same folder)
2. Edit and save the Simulink model.
   Be aware that the file configured as reference trajectory in the reference sequence playback block in the main model and the file `custom_libraries/MPC_controller_objects_Olympus.mat` are also part of the final binary.
3. Ensure that the dSPACE software is installed on your computer and that the license is active (the dSPACE license dongle is plugged in)
4. Ensure that the MicroLabBox is connected to your computer via LAN. The MicroLabBox is configured to have a static IP of 192.168.140.7, you need to manually assign an IP in the same subnet to the interface connected to the MicroLabBox, e.g. 192.168.140.1
5. Open the respective ControlDesk project with dSPACE ControlDesk
6. Hit `Ctrl + B` to start compilation with dSPACE RTI. It compiles the model and loads in to the MicroLabBox
7. In ControlDesk, hit "Go Online" to connect to the running model in the MicroLabBox. You should see the measurements and be able to send commands.
8. If you are done working on a feature, be sure to commit your changes to the version control system. Getting familiar with git is highly recommended. If you did any changes in the `custom_libraries` sub-repository, push them to the shared file repository s.t. other projects can use the updated version of the engine interface or the controller.

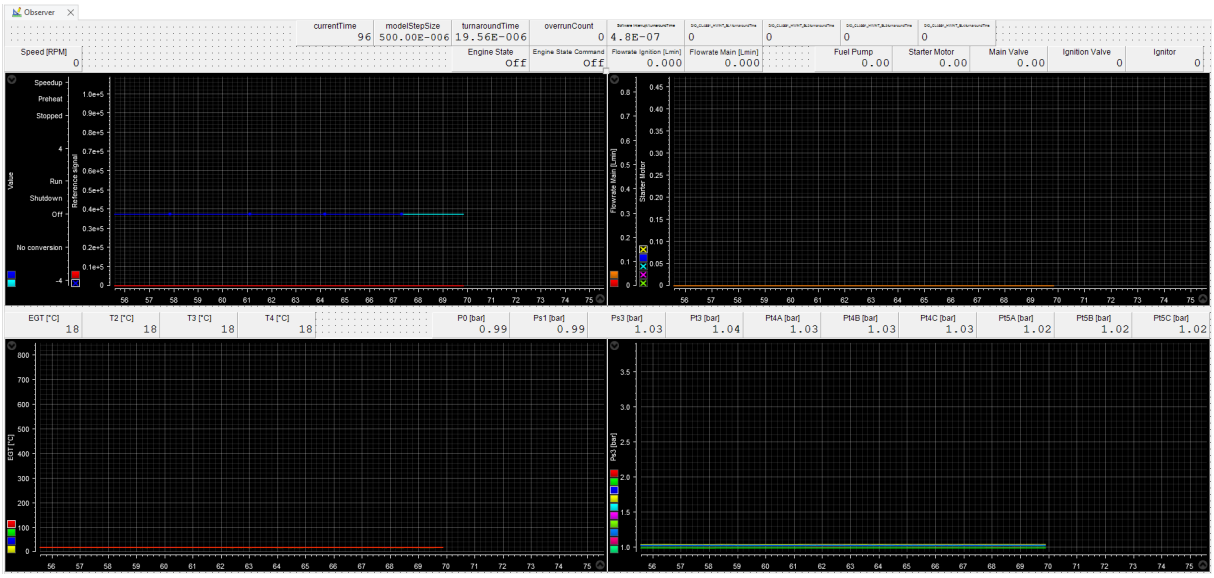
The design process for the MPC objects is automated in the script `Synthesize and save MPC.mlx`. The process model is identified using subspace identification (SID) methods from a specified dataset and preprocessing methods, then the MPC controller is designed with the Matlab toolbox, using the parameters specified in the script. The synthesized MPC objects then can be analyzed and saved to the file `custom_libraries/MPC_controller_objects_Olympus.mat`.

### 2.2.4 Control interface with dSPACE ControlDesk

Once the software is compiled, the ControlDesk project provides a user interface for running the engine and taking measurements. Several interfaces were created during this project to facilitate the use of the software models. Two views of the main interface can be seen in Figure 2.17.




(a) Main display of the control interface with the control inputs and the most important measurement values displayed.



(b) Secondary tab of the control interface displaying all measured signals.

Figure 2.17: Screenshots of the ControlDesk display of the main software project, through which the engine can be controlled.

**6 Recording signals during engine runs**

To analyze and examine the signals of an engine run, dSPACE ControlDesk has a feature to record the signals. In "Measurement Configuration", configure which variables should be recorded, start and stop the recording through the interface in the "Recording" group. The measurement files are saved as `.mf4` files (ASAM measurement data format version 4, a widely used standard file format for measurement data), they can be viewed in ControlDesk or natively read by Matlab with the command `mdf` for analysis and processing.

## 2.3 Startup and shutdown of the engine

A precondition for running the closed-loop control algorithms on the engine is that the engine runs, which requires that proper startup and shutdown procedures are implemented. These are currently implemented in a state machine, as described in Section 2.2.1. An image of the state machine implemented in Matlab Stateflow can be seen in Figures 2.18 and 2.19.
The transition between the states off, starting, running and shutdown is controlled by commands issued by the user through the ControlDesk interface (or if the safety module trips).

### 2.3.1 Startup sequence

The procedure for starting the Olympus HP engine was identified from examining recorded data from running the engine with the OEM controller and from information in the manual of the Olympus HP [1]. It comprises the following steps:

1. Preheat
   For 10 s, the igniter/glow plug is turned on to preheat.

2. Speedup Motor
   The engine rotor is sped up to a speed of 3000 RPM.

3. Wait for ignition
   Open the ignition fuel solenoid valve and start the pump at a very low power of 7.2 %. The pump was observed to have some startup hysteresis, to be sure the pump motor starts and is not stuck, start it with a pulse of 300 ms at a higher power of 12 % (value determined experimentally). The starter motor is kept at 30 % power. Ignition is considered to have taken place when a EGT rise of 5 °C has been observed since opening the ignition fuel solenoid valve.

4. Ramp-up phase 1
   Slowly transition the fuel flow from the ignition fuel line to the main fuel line by applying a PWM signal to the main fuel solenoid valve, at an initial duty cycle of 4 %, increasing it at a rate of 8 %/s. Also, the starter motor is ramped up to full power over a period of 9 s. At a speed of 10 000 RPM transition to the next phase.

5. Ramp-up phase 2
   Close the ignition fuel solenoid valve, turn off the igniter and constantly open the main fuel solenoid valve. Start to raise the fuel flow slowly by increasing the pump power at a rate of 0.2 %/s. At a speed of 20 000 RPM transition to the last phase.

6. Ramp-up phase 3
   Turn off the starter motor, the engine should be able to run now. Continue increasing the pump value until the idle speed of 36 000 RPM is reached. Running condition is reached,

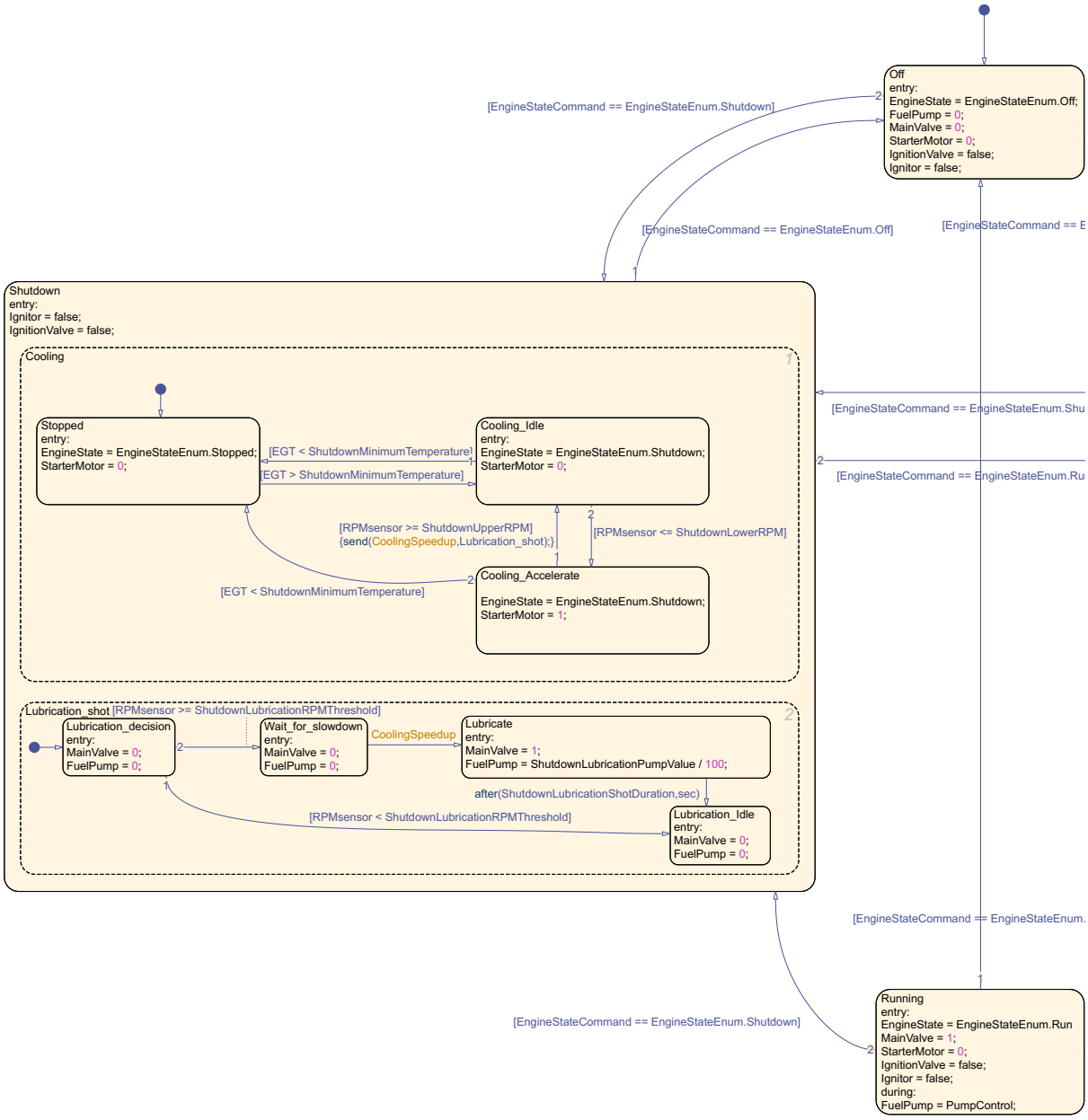



Figure 2.18: States Off (top), Running (bottom) and Shutdown (left, with sub-states, implementing the procedure) of the state machine controlling the operating state of the engine.

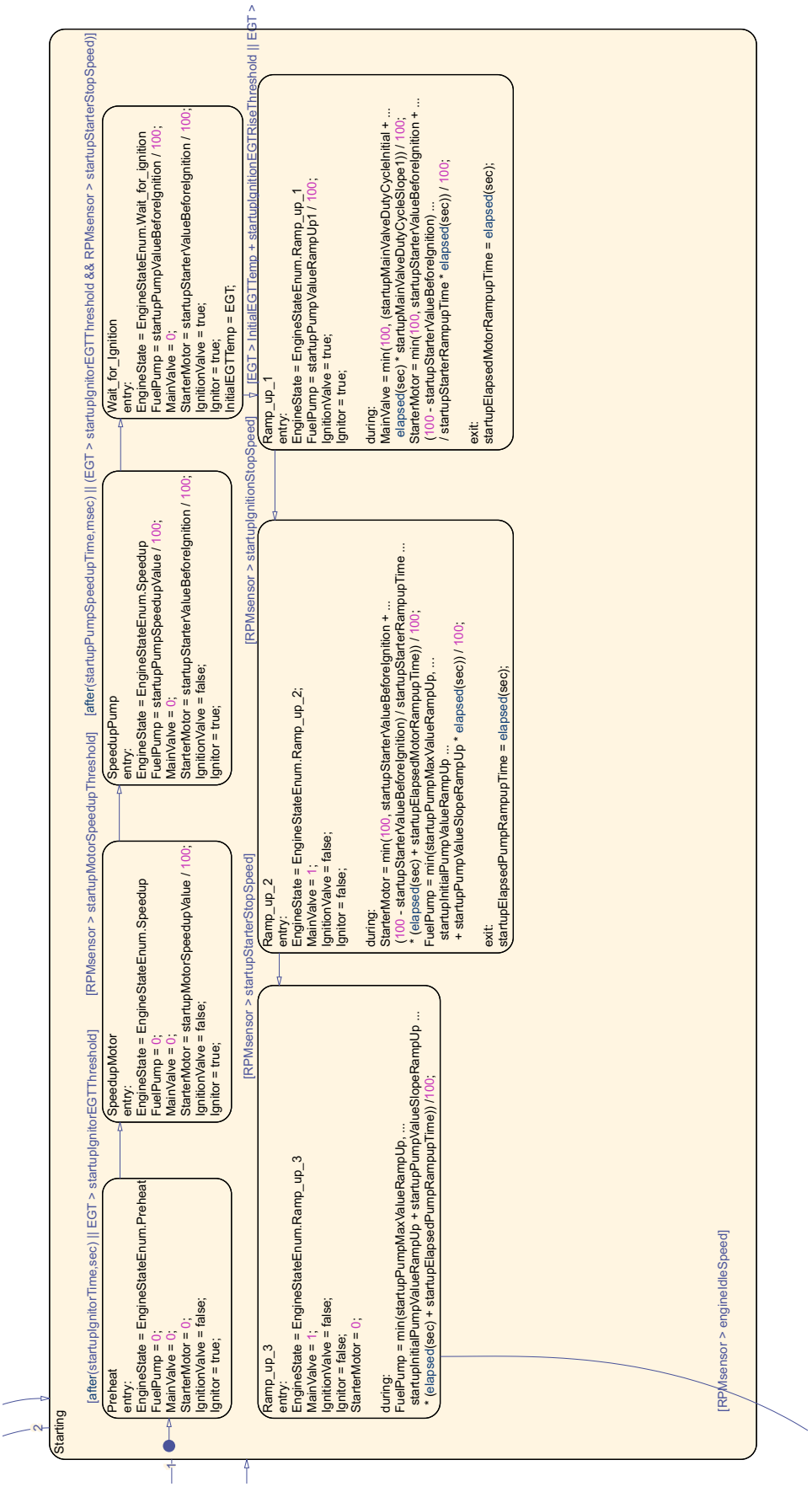



Figure 2.19: State Starting (with sub-states, implementing the procedure) of the state machine controlling the operating state of the engine.

and the fuel pump is henceforth controlled by the control algorithm.
In case a flame-out is experienced during startup, most likely rate at which the pump power is increased should be lowered, ramping it up too fast might cause the ignition to stop, as the combustor has not enough heat to evaporate all incoming fuel.

### 2.3.2 Shutdown sequence

When the engine shuts down (either because the respective command was issued or because a measurement exceeded the safety limits), the pump is turned off and the main fuel line solenoid valve is closed, the engine speed will then reduce within a few seconds. To ensure cooling, the starter motor keeps the engine rotor speed between 1900 RPM and 4500 RPM until a temperature of 88 °C is reached. Additionally, as soon as the engine is cooled down sufficiently, a small amount of fuel is pumped into the engine. This extends the bearing lifetime, as the bearing lubrication happens through the oil mixed into the fuel.

### 7 Instructions and checklist for running the engine

1. Ensure that enough fuel is in the tank for the planned engine run. The engine consumes around 0.2 L/min at idle speed and around 0.7 L/min at top speed, a full tank of 20 L of fuel should be enough for about 30 min runtime, depending on the engine speed. If you need to refuel, afterwars make sure to prime the fuel system afterwards (i.e. remove air bubbles from the system, e.g. by plugging the fuel lines to the fuel return and running the pump with the software model `manualControl`, see practical instructions section 4). Also verify that no air bubbles are entering the fuel system at the fuel filters (they can be seen in the clear fuel tubes when the pump runs at low speeds). This represents a major input disturbance and has caused some problems.

2. Ensure that the engine points through the exhaust pipe, stand is secured with the chain to the wall and that the breaks are fastened.

3. Verify that the compressed air system is connected by shortly opening the air switch on the OEM control panel and verifying that the engine rotor turns. It is used for emergency cooling. For safety, also be sure to know where the fire extinguishers are located, know where the power switch to turn off the electricity in the engine room is in case of an emergency and never run the engine alone!

4. Ensure that the channels of the thermocouple amplifier are hooked to the desired measurement points, that the cables of the engine are connected to the test stand and that all pressure lines are either connected to their pressure transducer or their valve is closed. If the line of a pressure probe is open to ambient air, a high air flow occurs and the heat would destroy the tubing.

5. Open the main fuel valve.

6. Remove the red cap at the engine inlet.

7. In dSPACE ControlDesk, hit “Start measuring”, optionally start the recording and press the green “Run” button. Enjoy the ride.

8. When shutting the engine down, be sure to press the orange “Shutdown” button to ensure a proper cooldown of the engine. The red “Off” button executes an emergency shutdown.

9. When the engine has sufficiently cooled down, re-place the red cap in the engine inlet and turn off the power supply of the test stand.

### 8 Roadmap to adapt test bed for a new engine model

To control a different engine model, following changes in the setup might be necessary, depending on the differences between the models:

1. Adapt the circuitry and sensors to the new engine. If the same in- and outputs are available as for the AMT Olympus HP, only different cabling might be necessary. If other control or sensor signals are required, the appropriate interfaces to the MicroLabBox must be made.

2. Create a dedicated engine interface and controller block for the new engine in the libraries in `custom_libraries`. The blocks for the AMT Olympus HP can be copied and only required things changed.

3. Initial hardware tests to ensure that the in- and outputs are interfaced correctly should be performed with the model `manualControl`.

4. Adjust the safety limits of the controller according to the specifications of the engine. The safety limits are parameters of the controller block.

5. Get the engine running - identifying an appropriate startup sequence is individual for every engine. The control signals of existing engine controllers can be imitated or the startup procedure for the AMT Olympus HP might be adjusted in an iterative process to find suitable parameters.

6. Perform open-loop experiments as described in Section 3.1.3.

7. Using the acquired measurement data, an MPC controller can be designed as described in practical instructions section 9.

# Chapter 3

# Methodology

This chapter describes the methodology and its theoretical foundations, how the dynamic model of the micro-gas turbine (MGT) is identified and based on this, the model predictive control (MPC) controller is designed and tuned. The results are presented and evaluated in Chapter 4.

## 3.1 System identification

There are different approaches to identify a dynamic model for a plant. First principles modelling derives the model from physical laws. This requires thorough knowledge of the plant and its physical processes. In contrast, subspace identification (SID) methods fit a model to input-output data, without requiring much knowledge about the underlying system. This project uses the second approach, since automated system identification methods allow to design controllers for new engine types or for changed operating environments much quicker than manual processes.

### 3.1.1 N4SID

The Numerical algorithms for Subspace State Space System IDentification (N4SID) algorithm used in this thesis is out of a family of SID algorithms which aim at identifying linear time-invariant models to sequences of input-output data. The form of the identified discrete-time state-space models is the well known model structure given by

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

with $x(k)$ being the internal states of the system at time $k$ (the number of states being the order of the system), $u(k)$ being the inputs at time $k$, $y(k)$ the outputs and $A, B, C$ and $D$ the system matrices, which are to be determined by the SID process. N4SID does so by first estimating the state sequence of the Kalman filter through an oblique projection. From this state sequence, the system matrices $A, B, C$ and $D$ are obtained solving a least squares problem [14]. N4SID is always convergent, and since it uses only QR and Singular Value Decompositions it is numerically stable [25].
In this thesis, a variant of the N4SID algorithm known as Canonical Variate Algorithm (CVA) is used. The mathematical details of this algorithm are well explained in [14, 25].

### 3.1.2 Gain scheduling

Doing experiments with the AMT Olympus HP, differences in the dynamic behavior w.r.t the operating point soon become evident. This is illustrated by the non-linear gain in Figure 3.1 and from the varying rise-time in the open-loop step responses in different operating regions in

Figures 4.1 to 4.5. Also literature reports great non-linearities in the system dynamics of MGTs [10, 13].

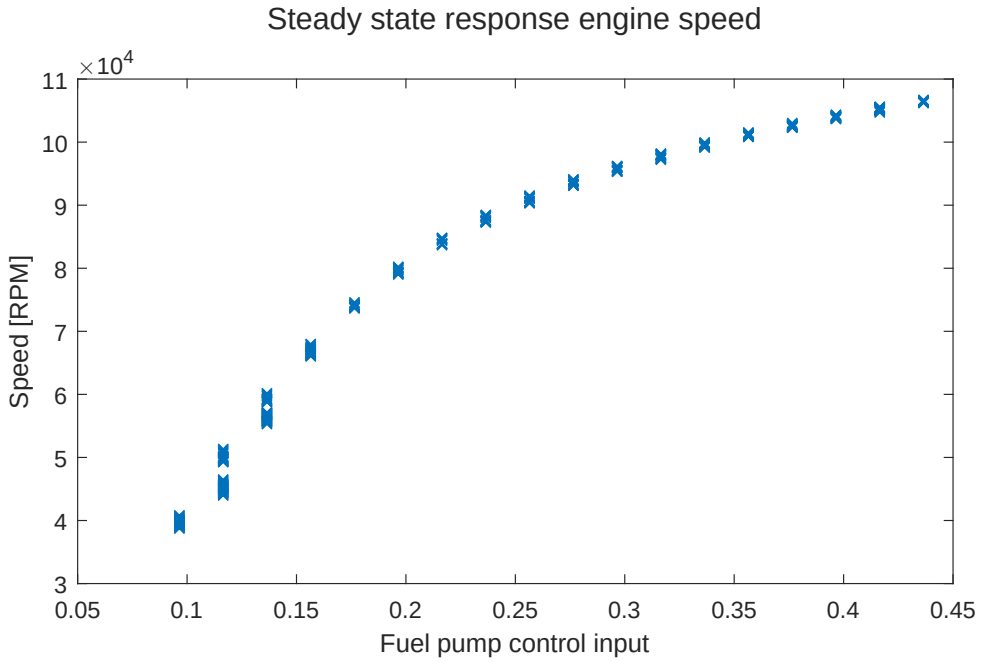



Figure 3.1: Steady state response of the engine speed vs. fuel pump control input.

To reduce the model mismatch when linearizing the plant model, gain scheduling can be used: Instead of fitting a single model to the whole operating range, the operating range is divided into smaller subspaces and a model fitted to each of them. In their respective operating region, these individual models can capture the dynamics more accurately.
In this thesis, the operating range is divided into operating regions by the engine speed. The operating range of 36 000 RPM idle speed to 108 000 RPM full speed is divided into five regions of 14 400 RPM each. To counter oscillations between adjacent operating regions, a hysteresis of 5000 RPM is used.

### 3.1.3 Openloop experiments

To acquire input-output data for the SID algorithm, open-loop experiments are performed. The choice of the excitation signal is very important, since all the system dynamics should be captured in the experiment data. It is reported that generally good results are achieved using white noise as an input signal [15, 32].
The input signal to acquire input-output data for the SID process within a single operating region is designed as follows:

1. The nominal operating point in the speed output is set in the middle of the speed output range covered by the operating region. The corresponding nominal fuel pump control input is identified by using the steady state measurement data plotted in Figure 3.1.

2. A white noise signal with a duration of 88 s and a standard deviation corresponding to 2000 RPM is superimposed to the nominal operating point for the fuel pump control input. The gain to map the standard deviation of 2000 RPM to a standard deviation for the fuel pump control input value is again identified using the data plotted in Figure 3.1. The

step width of the white noise is chosen according to the method described in Section 3.1.5 using the data plotted in Figure A.1.

3. In an attempt to get consistent initial conditions, a period with a constant input signal is prepended to the input sequence.

The input signal sequences for the individual operating regions are concatenated. The whole resulting input trajectory is repeated twice, to get two individual datasets with input-output data for each operating region.
The resulting input sequence and the measurement outputs can be seen in Figure 3.2, Figures 3.3 and 3.4 show input-output data for operating regions 1 and 5 respectively, with the two repetitions of the signal overlayed.

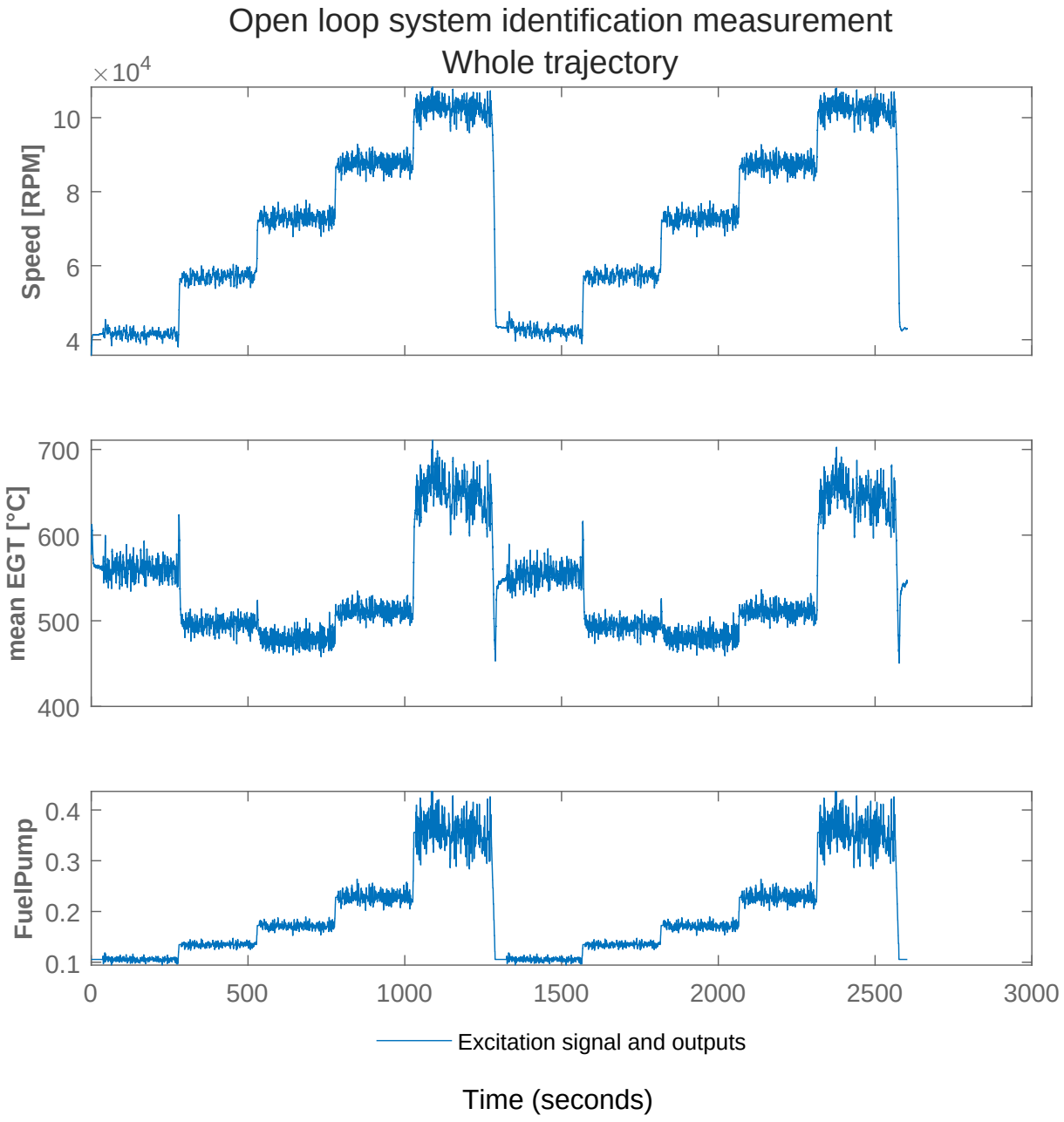



Figure 3.2: Input signal (fuel pump) and output signals of the open loop test run.

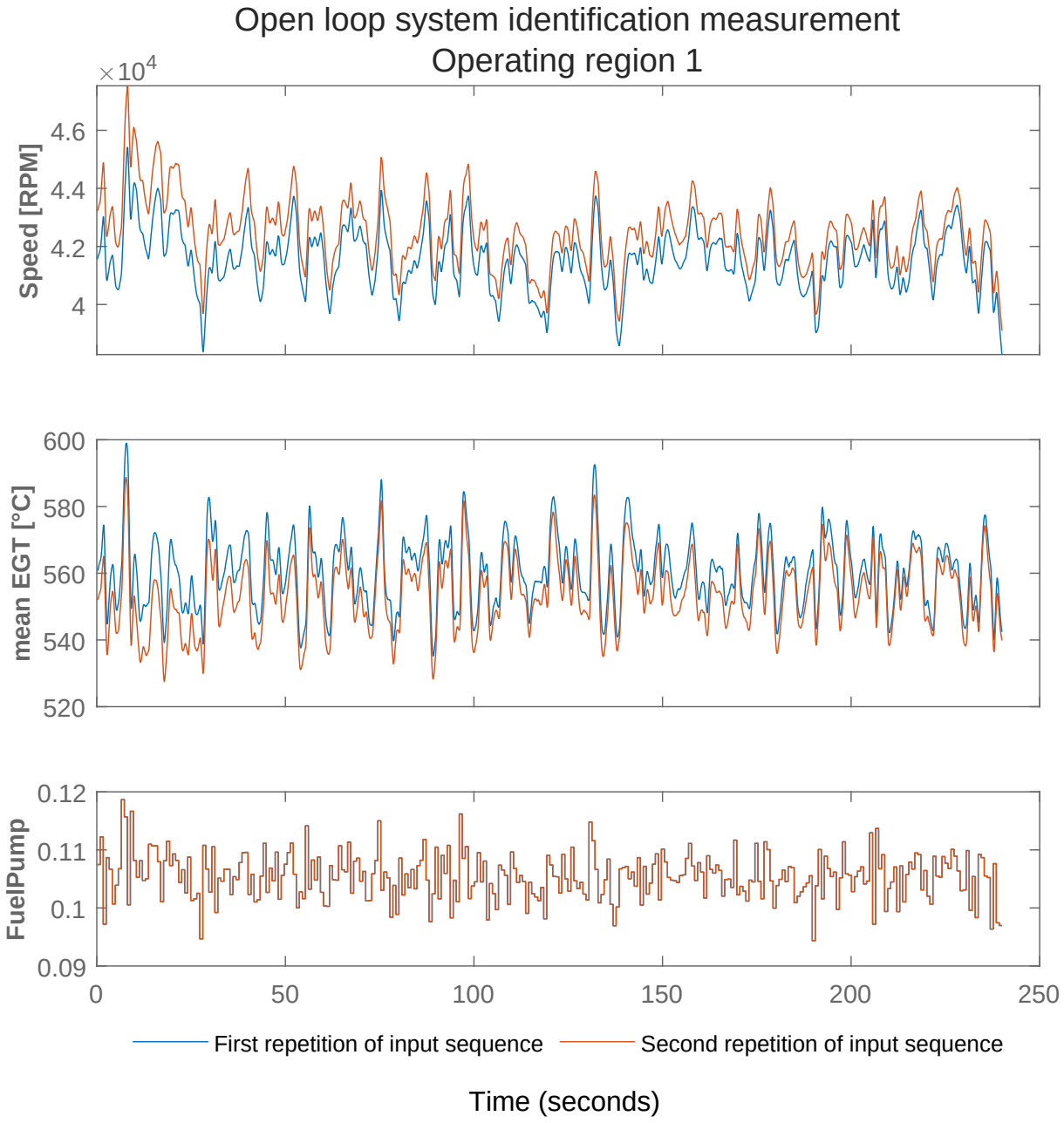



Figure 3.3: Input signal (fuel pump) and output signals of the open loop test run of both repetitions of the input trajectory in operating region 1.

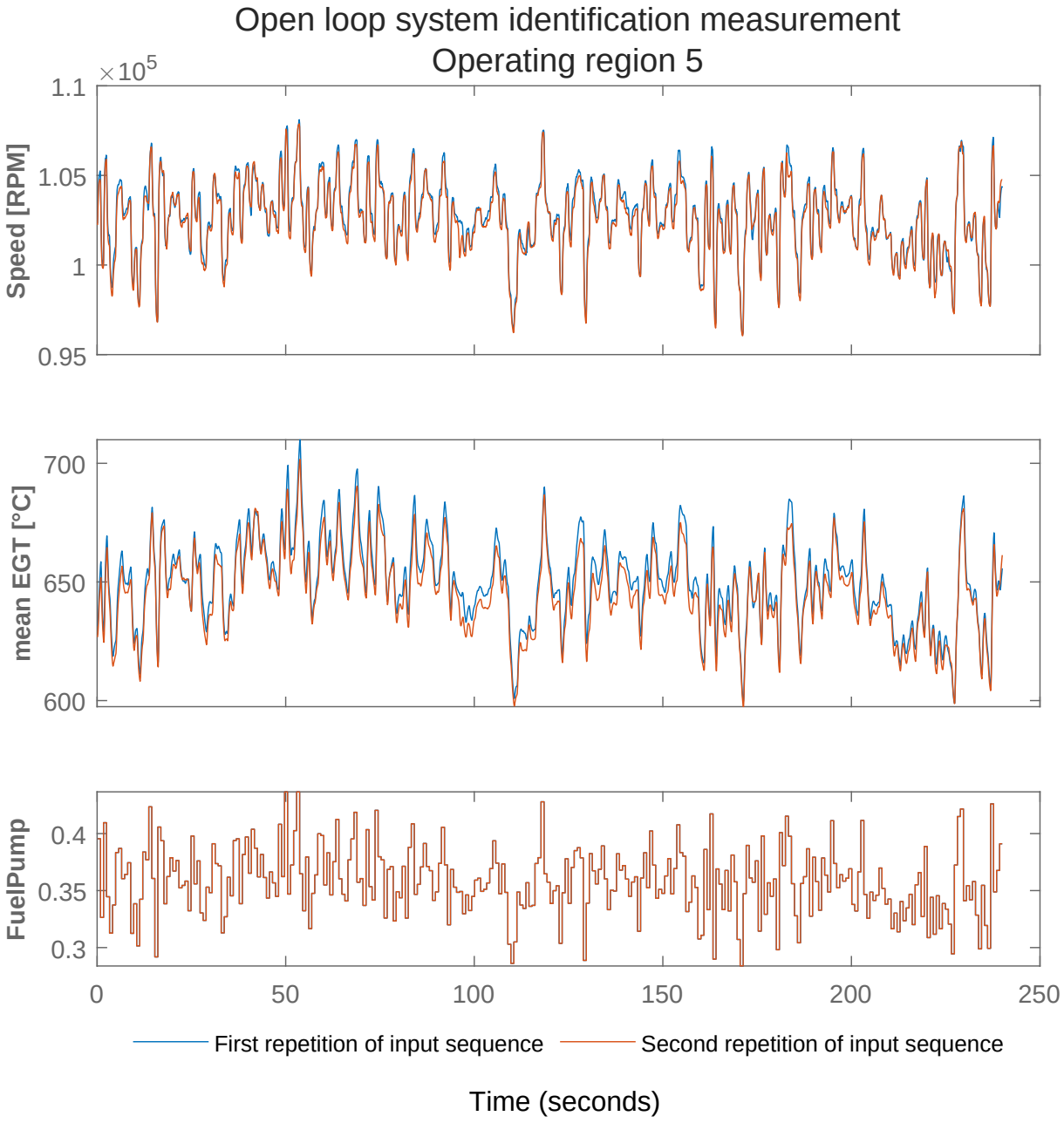



Figure 3.4: Input signal (fuel pump) and output signals of the open loop test run of both repetitions of the input trajectory in operating region 5.

### 3.1.4 Data preprocessing

The data is sampled at a rate of 2 kHz. The following steps are applied to the data, both online on the real-time hardware and offline for system identification and controller design:

1. The data is downsampled to the model step size by averaging all measurement samples in one period.

2. Output transformations are applied. The following set of transformations is used for the results of this thesis (other transformations have been tested, see Section 3.1.5):

   (a) The outputs are normalized. A scale factor of $\frac{1}{108\,000}$ is applied to the speed channels, a factor of $\frac{1}{750}$ to the temperature channels, yielding a value of 1 for the maximum allowed value of the respective channels (refer to Table 2.1 for the specifications of the AMT Olympus HP MGT).

   (b) The temperature measurements EGT and T4 are averaged. These measurements are taken at the same axial stage, see Section 2.1.3.

### 3.1.5 Parameter selection for system identification

To select the optimal parameters for system identification, model are fitted using different sets of parameters and the resulting models are compared.
To evaluate the models, a separate dataset is used with steps of different amplitude in each operating region. The signals and the outputs to it can be seen in the experiment channel in Figures 4.1 to 4.5. The output of the system is simulated using the model to be evaluated, while initial conditions are estimated s.t. the prediction error is minimized. For each output channel separately, the Normalized Root Mean Square Error (NRMSE) in percent is calculated using the formula

$$NRMSE = 100 \cdot \left(1 - \frac{\|y - \hat{y}\|}{\|y - \mathrm{mean}(y)\|}\right), \tag{3.2}$$

where $y$ is the measured output and $\hat{y}$ is the simulated output of the model under evaluation. A value of 100 % indicates that the model follows the output perfectly, while lower values indicate poorer model accuracy. This NRMSE metric is used to compare the model with other models.
For the following parameters for system identification, different parameter settings are compared to identify the best set of parameters for system identification:

**Input noise step width** The most suitable step width for the input noise is picked by fitting models to short sequences of data with different step widths (see Section 3.1.3 for a description of the input signal and Figure A.1 for a plot of the results)

**Temperature unit** The use of the temperature channels in units of °C or K (Figure A.2)

**Output normalization** Three different options for the normalization of the outputs (Figure A.3):

- No normalization
- Normalize by the maximum value in the dataset
- Normalize by the maximum allowed values of the engine (108 000 RPM for the speed output and 750 °C for the temperature output)

**Exhaust gas temperature (EGT) measurement** For the exhaust gas temperature, two measurement channels are available as described in Section 2.1.1. The following options are evaluated (Figure A.4):

- use only the EGT measurement
- use only the T4 measurement
- use the mean of the two measurements

**Exponentiation of the EGT measurement** Raising the EGT measurement to different powers, as it is not a priori clear that no exponentiation yields the best linear models (Figure A.5)

**Exponentiation of the speed measurement** Likewise, raising the speed measurement to different powers (Figure A.6)

**Model step size** Different step sizes for the identified model (Figure A.7)

**Model order** Finally, the order of the identified model (Figure A.8)

For each parameter, the best option is selected by generating models for each possibility and a set of different options for the remaining parameters. The NRMSE for each model is plotted in a boxplot (one box for each evaluated parameter, displaying the distribution of the NRMSE values of the different combinations for the remaining parameters), to examine the influence of the parameter and its optimal setting. The plots and what parameter settings are picked can be found in Section 4.1.2, Table 4.1, and Figures A.1 to A.8.

### 3.1.6 Simulation setup

The engine behavior within a single operating region is simulated with one model, to simulate them no special setup is required. To be able to simulate the engine behavior over the whole operating range, the following setup is used: Five Kalman filters are designed based on the model of each operating region. They estimate the state of the internal model of each operating region using the past control inputs and the simulated outputs. The predicted outputs of the filters are demultiplexed using the same switchover signal as for the controllers (see Section 3.1.2). This model can be run both in the Simulink environment or compiled and executed on the dSPACE MicroLabBox, the same real-time hardware which is used to control the engine.
The plots to evaluate models within a single operating region are generated with the Matlab command `compare` (Figures 4.1 to 4.5), the data for comparing the model with the system response of the physical engine is generated running the simulation on the dSPACE MicroLabBox (Figure 4.6).

## 3.2 MPC controller design

In this section, a short overview of the working principle of MPC controllers is given and the cost function and the constraints used in this project are explained.
MPC is a control algorithm which periodically calculates the optimal control input using a dynamic model of the controlled plant to predict future outputs and respecting imposed constraints. In each step, the current internal states of the system are estimated, using e.g. a Kalman filter as state estimator. Employing the plant model, equations are derived that predict the outputs of the plant during the prediction horizon $p$ as a function of the control inputs during the control horizon $c$ (with $c \leq p$, see Figure 3.5 for an illustration of the control and prediction horizon). Together with the cost function $J$ and optionally constraints for the in- and output variables, this constitutes an optimization problem. Solving it yields the optimal control input sequence to the plant under the current estimated internal state of the system. The cost function $J$ can entail various penalties. Usually, penalties for reference tracking error

and for high variations in the control variable are included, also penalties for deviations from target values of the control variable or for violations of soft constraints (see Section 3.2.1 for an explanation of constraint softening) can be added. If the cost function is designed appropriately, the resulting optimization problem is a quadratic problem, for which efficient solving algorithms exist.
Once the MPC algorithm has determined the optimal control input sequence in the current step, the first input of this sequence is applied to the plant. In the next step of the algorithm, the new output measurement samples are used to re-estimate the internal state of the system and the whole optimization problem is re-evaluated. The MPC control algorithms scheme of re-evaluating the optimal input sequence in every control interval anew for a finite period into the future is also referred to as Receding Horizon Control, since the considered time period is constantly expanded as the start of the prediction and control horizons is always updated to the current control interval.
The cost function $J$ and the constraints for the MPC controller used in this project are explained in the next section.

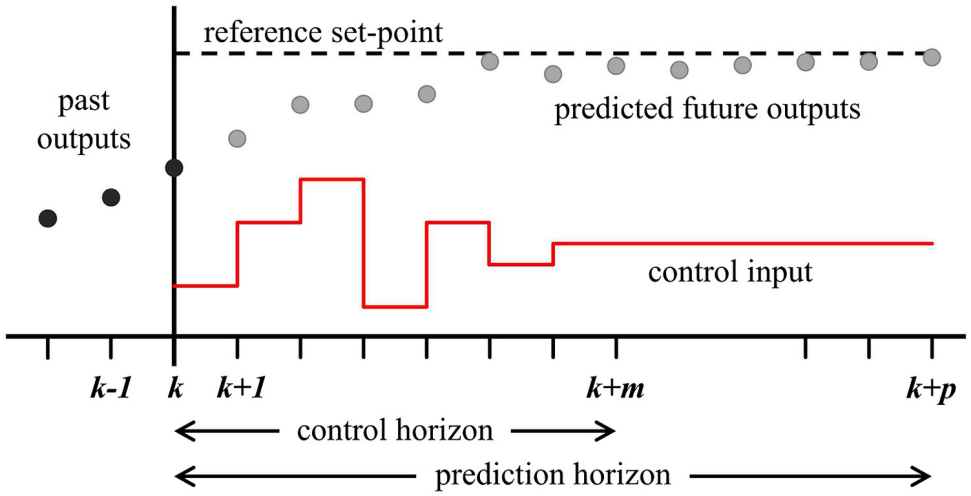



Figure 3.5: Illustration of the working principle of an MPC controller with prediction and control horizons.

Source: [9]

### 3.2.1 Model predictive control

As explained in the section above, the MPC algorithm solves an optimization problem in each control interval. The cost function $J$ which is optimized and the constraints are presented in this section.
The resulting optimization problem (the full optimization problem is reproduced in Equation (4.1)) is quadratic in its free variables and can thus be solved as a quadratic programming (QP) problem. In this thesis, the active-set solver is used for solving the QP problem. This algorithm first obtains a feasible point which respects all constraints, and then iteratively lowers the objective function $J$ while maintaining feasibility (i.e. respecting all hard constraints) in each iteration using the QPKWIK algorithm found in [18]. The active-set solver can provide fast and robust performance for small-scale and medium-scale optimization problems [23]. To reduce the computational burden, the found solution of the previous control interval is used as an initial guess, as the optimal solution in the current interval is usually close to this point. The

detailed description of this algorithm exceeds the scope of this thesis and can be understood with [24].

**Cost function $J$**

The overall cost function is

$$J(z_k) = J_y(z_k) + J_u(z_k) + J_{\Delta u}(z_k) + J_\varepsilon(z_k) \tag{3.3}$$

with

$$\begin{aligned}
J &:= \text{Cost function} \\
z_k &:= [u(k|k)\ u(k+1|k)\ \cdots\ u(k+c-1|k)\ \varepsilon_k]^T \\
&\quad \text{QP decision at time step } k \\
c &:= \text{Control horizon} \\
u(j|k) &:= \text{Optimal control input to the fuel pump at time } j\text{, evaluated at time } k \\
\varepsilon_k &:= \text{Slack variable} \\
J_y &:= \text{Output reference tracking penalization} \\
J_u &:= \text{Fuel pump control variable penalization} \\
J_{\Delta u} &:= \text{Fuel pump control variable move penalization} \\
J_\varepsilon &:= \text{Soft constraint violation penalization.}
\end{aligned}$$

The single terms are defined as follows: The output reference tracking penalization $J_y$ imposes a cost when the predicted output deviates from the given reference. Usually, a high weight is set for this, to ensure that the output follows the reference closely. In the scope of this thesis, only a reference for the engine rotor speed $N_{RPM}$ is set by the user. Both the engine rotor speed and its reference value are normalized as described in Section 3.1.4 and denoted with $\widetilde{N}_{RPM}$ and $\widetilde{r}$ respectively. The reference $\widetilde{r}$ is sampled at time $k$ and is assumed to be constant throughout the whole prediction horizon $p$.
The output reference tracking penalization is calculated with the formula

$$J_y(z_k) = \sum_{i=1}^{p} \left( \frac{w^y}{s^y} \left( \widetilde{r}\,(k) - \widetilde{N}_{RPM}\,(k+i|k) \right) \right)^2, \tag{3.4}$$

where

$$\begin{aligned}
p &:= \text{Prediction horizon} \\
\widetilde{r}\,(k) &:= \text{Normalized reference setpoint for the engine rotor speed} \\
&\quad \text{at time } k \text{ (reference input variable)} \\
\widetilde{N}_{RPM}\,(i|k) &:= \text{Predicted, normalized engine rotor speed for time } i\text{,} \\
&\quad \text{evaluated at time } k \text{ (predicted using the plant model and } z_k\text{)} \\
w^y &:= \text{Tuning weight for the speed output reference tracking penalization} \\
s^y &:= \text{Scale factor for the speed output.}
\end{aligned}$$

While the scaling of the output variables is optional and could be incorporated in the tuning weights, [22] recommends setting scale factors s. t. all variables in the optimization problem have the same amplitude in the range of the expected values. This facilitates comparison of tuning weights. Here, $s^y$ is set the normalized difference between idle speed and full speed, i.e.

$$\frac{108\,000\,\text{RPM} - 36\,000\,\text{RPM}}{108\,000\,\text{RPM}} = \frac{2}{3}.$$

The speed output reference tracking weight $w^y$ is fixed at a value of 1, all weights are tuned relative to this.
The term for control variable penalization $J_u$ is used to keep the control variable close to some specified target value. In the present problem, it is used to keep the fuel pump control input $u$ close to zero, to favor a lower control input and thus lower fuel and energy consumption. The control variable penalization term is calculated by

$$J_u(z_k) = \sum_{i=0}^{c-1} \left( \frac{w^u}{s^u} u\left(k+i|k\right) \right)^2 + (p-c-1) \cdot \left( \frac{w^u}{s^u} u\left(k+c-1|k\right) \right)^2, \tag{3.5}$$

where

$$\begin{aligned} w^u &:= \text{Tuning weight for the control variable penalization} \\ s^u &:= \text{Scale factor for the control variable.} \end{aligned}$$

The fuel pump control inputs during the control horizon $[u(k|k)\ u(k+1|k)\ \cdots\ u(k+c-1|k)]$ are optimization variables and are to be optimized by the QP solving algorithm. After the end of the control horizon, for the evaluation of the cost function the control input is assumed to remain constant throughout the rest of the prediction horizon. $s^u$ is set to the difference of the steady state fuel pump input for idle speed and the maximum expected fuel pump value, $0.5 - 0.09 = 0.41$.
The term for the control variable move penalization $J_{\Delta u}$ imposes a penalty on changes in the fuel pump control input. Having a high weight $w^{\Delta u}$ in this term favors a solution to the optimization problem, where the fuel pump control variable changes as little as possible. This is desirable since it reduces over- and undershoots and favors a steady output. This term is calculated by

$$J_{\Delta u}(z_k) = \sum_{i=0}^{c-1} \left( \frac{w^{\Delta u}}{s^u} \left( u\left(k+i|k\right) - u\left(k+i-1|k\right) \right) \right)^2, \tag{3.6}$$

with

$$w^{\Delta u} := \text{Tuning weight for the control variable move penalization.}$$

The value $u(k-1)$ is the fuel pump control input applied in the previous control interval.
Finally, the cost function $J$ has a term for penalization of soft constraint violations $J_\varepsilon$. To ensure that the optimization is always feasible, constraints can be relaxed. Instead of imposing hard limits, a cost can be attributed to the violation of some constraints, as described in the next section. By incorporating this cost in the cost function and attributing it a high weight, a solution is targeted where all hard and soft constraints are respected. The soft constraint violation penalization term $J_\varepsilon$ is calculated by

$$J_\varepsilon(z_k) = \rho_\varepsilon \cdot \varepsilon_k^2. \tag{3.7}$$

$\varepsilon_k$ is the slack variable at control interval $k$ as defined in the next section, $\rho_\varepsilon$ is the constraint violation penalty weight. $\rho_\varepsilon$ is fixed at a value of 100 000, the individual constraint violations can be weighted individually as explained in the next section.
The value of the tuned parameters for the prediction horizon $p$, the control horizon $c$, the weights for the control variable penalization and control variable move penalization are listed in Table 4.2, while the final cost function is listed in Equation (4.1).

### Constraints

MPC controllers allow specifying constraints on in- and output variables. The constraints imposed in the MPC controller for this project are listed and explained in the following section. The final set of constraints including their values can be seen in Equation (4.1).
On the fuel pump control input $u$, a minimum of 0.071 and a maximum of 0.7 is imposed. At a control input of less than 0.071, it was observed that the pump occasionally stalls. At a control input of more than 0.7, the pump driver might overheat and the pump might get damaged. These constraints are implemented as hard constraints (as opposed to soft constraints, see the next paragraph for an explanation), as they can always be satisfied.
If output constraints are implemented as hard constraints, it is possible that the optimization problem gets infeasible and has no solution [22], as illustrated by the following example: If one of the output variables is outside of the boundaries (e.g. because of a disturbance or because another controller had been active) and it is not possible to bring this output variable within the boundaries in the next step, the optimization problem becomes infeasible. Since the QP solver provides no solution, the MPC controller can not update the control input which leads to an opened control loop. To avoid such a situation, constraint softening is introduced: The slack variable $\varepsilon$ is added as a free variable to the optimization problem, which is a measure of the predicted maximum violation of the constraints throughout the prediction horizon. Since a high cost is associated with the violation of soft constraints as seen in Equation (3.7), the controller will satisfy also the soft constraints if there is any feasible solution with this property. In a situation where there are contradicting soft constraints active, their relative weights can be tuned using the equal concern for relaxation (ECR) values $V^y_{min}$, $V^y_{max}$ and $V^{EGT}_{max}$.
On the speed output $N_{RPM}$, a minimum of 30 000 RPM and a maximum of 108 000 RPM with ECR values $V^y_{min}$ and $V^y_{max}$ of 1 are imposed. The lower limit is set 6000 RPM below the idle speed of the engine to prevent excessive undershoot, the imposed maximum ensures safe operation of the engine (see Table 2.1 for the specifications of the engine).
The maximum for the EGT output $T_{EGT}$ is set depending on the operating region. It was found that the models in the lower operating regions underestimate the temperature overshoot for quick transitions to higher speeds. In an attempt to prevent overshoots over the maximum allowed EGT, which causes the safety module to initiate a forced shutdown, the EGT constraint in lower operating regions is set at a lower temperature than in the operating ranges at higher speeds. The set $T_{EGT\,max}$ for each operating region can be found in Table 4.2. Since the EGT constraint was found to be more critical than the speed constraint, the associated ECR value $V^{EGT}_{max}$ was set to 0.2. There is no lower EGT constraint set.

### Noise models and state estimation

At the beginning of each control interval, the internal states of the system are estimated using a Kalman filter. The Kalman gains in each operating region are estimated using the identified noise models. Both the output disturbance noise (to be corrected for by the controller) and the measurement disturbance (to be ignored) are assumed to be white noise. In each operating region, the standard deviation of the two white noise signals is estimated by examining the output signal in a period which is in steady state or close to steady state. The Root Mean Square (RMS) value of the steady-state output signal filtered with a low-pass filter is taken as the standard deviation of the output disturbance, while the RMS value of the same steady-state output signal filtered with a high-pass filter is taken as the standard deviation of the measurement disturbance, i.e. any high-frequency noise is assumed to be measurement noise. As a cutoff frequency for the filters, 3 Hz is used for the speed output and 0.5 Hz for the EGT output, since the latter shows a much slower rise time than the speed output.
The full process how the Kalman filter is derived can be reproduced with [21].

**Data alignment**

The models identified with the method described in Section 3.1 predict the output response to the control input within the same sampling period. In the real-time implementation of the data-preprocessing method described in Section 3.1.4, the output samples are averaged over one control interval and hence the response to the control input is available only in the next interval. To correct for this, an output delay of one control interval has to be added to the internal models of the MPC controllers. Neglecting this results in increased overshoot and settling time and considerable mismatches between experiment and simulation.

### 3.2.2 Parameter selection for the MPC controller

To evaluate and compare designed MPC controllers, the following metrics are calculated for simulated or measured speed outputs $N_{RPM}$ in response to steps in the reference:

**Settling time** The settling time is calculated as the time between when the signal crosses the mid-reference level between the two steady states until it remains within a 5 % tolerance region around the final state.

**Over-/Undershoot** Over- and undershoot are the difference between the maximum/minimum value respectively and the final steady state value. They are expressed as a percentage of the step height and evaluated separately, while the over-/undershoot metric is the mean of the two values.

**$\mathbf{NRMSE_{ref}}$** The Normalized Root Mean Square Error for reference tracking $\mathrm{NRMSE_{ref}}$ is defined similarly as NRMSE, but normalized with the absolute reference value instead of the step height. It is calculated by

$$NRMSE_{ref} = 100 \cdot \left(1 - \frac{\|r - N_{RPM}\|}{\|r\|}\right), \tag{3.8}$$

where $r$ is the reference and $N_{RPM}$ the speed output over the whole evaluated signal duration.

**Steady state error** The steady state error is the euclidean norm of the error between the speed output and the reference signal over a period of one second before the first step in the signal. The output is assumed to be in steady state during that time. It is expressed as a percentage of the absolute reference during that period.

**Rise/Fall time** The time the output requires from exiting a 10 % region around the initial value until entering a 10 % region around the new steady state value. The parameter is calculated for rising and falling transitions separately, the mean of the two values is presented as rise/fall time.

**Fuel consumption** The fuel consumption is estimated by applying a first order transfer function and mapping the output with a second order polynomial function which maps the values to the flow rate. This flow rate is integrated. The transfer and mapping functions were experimentally determined.

To select the optimal horizons and weights for the MPC controller, the output response to a positive and a negative reference step of ±10 000 RPM around the nominal operating point of the controller is analyzed. Examples of such output responses can be seen in Figures 4.7 to 4.11. To choose the parameters, the above metrics are calculated and plotted in boxplots. The plots used for picking the parameters for can be seen in Figures B.1 to B.10.
The controllers are chosen according to the following figures of merit, in the order of importance:

- To prevent flameouts (see Section 2.1.1 for a discussion of flameouts), a rise/fall time of 0.5 s is targeted, lower values have been found to cause frequent flameouts.
- A steady state error as low as possible is targeted.
- The settling time and over-/undershoot are to be kept low.
- If candidates with equal or almost equal performance in the above metrics are available, lower fuel consumption is to be favored.

While there is no universal set of rules according to which the MPC parameters can be chosen and the process requires some experience and knowledge of the controlled plant, the parameters for the MPC controller for the present project are chosen following this general procedure:

1. All candidates with a rise/fall time outside the range of 0.4 s to 0.6 s are filtered out.
2. To allow the controller to predict exceedances of the maximum allowed EGT early enough, the prediction horizon $p$ is chosen to be at least as long as the maximum observed delay of the peak EGT after a step in the fuel pump control input. This delay is estimated using the data in Figures 4.1 to 4.5.
3. Balancing different performance metrics as described above, a control horizon $c$ is picked from a range of candidate controllers with the prediction horizon identified above, using the boxplots Figures B.1 to B.10.
4. Again using the same plots, the weight for the control variable penalization $u$ is chosen such that the steady state error is minimized.
5. Finally, the control variable move penalization weight $w^{\Delta u}$ is chosen to achieve the most favorable simulated performance results.
6. Using the identified set of parameters, simulations are performed. If the results are not satisfactory, above steps may be repeated.
7. If the simulation results are acceptable, the controller is tested on the hardware-in-the-loop (HIL) test bed and the performance is compared.

### 3.2.3 Evaluation methodology

Candidate controllers are evaluated with a three-step procedure. First, the single controllers are simulated within their operating region and plots as seen in Figures 4.7 to 4.11 (yet only with the simulation data, without the experiments which are plotted together in the mentioned plots) are generated. The initial conditions for these plots are found by simulating the controller for a constant reference at the level of the start of the dataset until steady state is reached.
Second, the gain scheduled controller is compiled and simulated on the real-time target hardware. The simulation data in the plots Figures 4.12 to 4.14 are generated with this procedure. To emulate the MGT response to given system inputs, the same approach as described in Section 3.1.6 is used.
Finally, the controller shall be tested on the physical engine. The results then can be compared to the simulation data and the two datasets plotted alongside. For all the steps, the metrics mentioned in the section above can be calculated and compared.

### How to compile and deploy a new MPC controller

To design a new MPC controller, proceed as follows:

1. In the script `MPC_design.mlx`, specify the desired parameters for each operating condition in the section "MPC parameter selection" and run it. If you want to adjust constraints or other specifications of the controller, this has to be done in the local function `design_MPC_controllers` towards the end of the file.

2. To tune the MPC parameters, the sections "Parameter tuning - Prediction and Control horizon" and "Parameter tuning - Cost function weights" can be used to simulate controllers with different parameters and compare their performance metrics using boxplots.

3. Run the section "Generate Controllers for different operating regions". The controller objects are now generated and stored in the workspace variable `mpc_controllers`.

4. With the section "Analyze generated controllers", simulation plots can be generated and the performance metrics calculated for each operating region. The reference signal for the simulation can be specified at the beginning of the section.

5. If the controllers perform satisfactorily, run the section "Save generated MPC objects for code generation". This saves the controller objects in the file `MPC_controller_objects_Olympus.mat`, which is read by the Simulink models and used for code generation.

6. To simulate the gain-scheduled controller in the whole operating range, open the model `AMT_Olympus_simulation.slx`. The simulation can be run in Simulink or compiled and executed on the dSPACE MicroLabBox.

7. To run the controller on the physical hardware, follow the procedure in practical instructions section 5.

8. To plot experimental data recorded as described in practical instructions section 6 alongside simulation data, use the script `MPC_variousScripts`, section "Plot simulation data from dSPACE simulation runs together with experiment data". There also the performance metrics are calculated from the data.

# Chapter 4

# Results

## 4.1 Micro-gas turbine model

### 4.1.1 Parameter selection for system identification and model evaluation

Using the process described in Section 3.1.5, the parameters specified in Table 4.1 are used for the system identification process. The plots which show the NRMSE for the different parameter sets can be seen in Figures A.1 to A.8.

### 4.1.2 Model evaluation

With the data preprocessing methods as described in Section 3.1.4, models are fitted to the experiment data using Numerical algorithms for Subspace State Space System IDentification (N4SID). Estimating the initial conditions s. t. the prediction error is minimized, the response of the micro-gas turbine (MGT) is simulated using the identified model. In Figures 4.1 to 4.5, the recorded experimental data is compared to the simulated output and the NRMSE metric given. It is concluded that the identified models capture most of the dynamics of the physical system accurately. Closer to the limits of the operating regions, it can be observed that some of the dynamics are not captured in the model anymore, particularly in operating region 4 it can be seen that the overshoot of the EGT is underestimated (see Figure 4.4 at 64 s and 72 s). This supports that using a gain scheduled model is necessary.
Additionally, the individual models for the separate operating regions are used in the overall gain scheduled model (refer to Section 3.1.6 for a description how the simulation is setup). The data for the plot in Figure 4.6 is generated using this software-in-the-loop (SIL) setup, to compare the simulation with the system response over the whole operating region. Also here, non-linearities are visible at the limits of the operating regions in the prediction of the EGT, considerable model

| | |
|---|---|
| Input noise step width | 0.8 s |
| Temperature unit | °C |
| Output normalization | Normalizating by the maximum allowed values |
| Exhaust gas temperature (EGT) measurement | Use the mean of EGT and T4 |
| Exponentiation of EGT | No exponentiation |
| Exponentiation of speed measurement | No exponentiation |
| Model step size | 0.1 s |
| Model order for operating regions 1 to 4 | 7 |
| Model order for operating region 5 | 11 |

Table 4.1: Selected parameter options for the system identification process.

mismatches are visible at the transition from one operating region to another. Please note that the model predictive control (MPC) algorithm can mitigate a lot of model inaccuracies through its scheme of estimating the internal state of the system from the output measurements in every control interval and using these states for prediction of the future outputs.

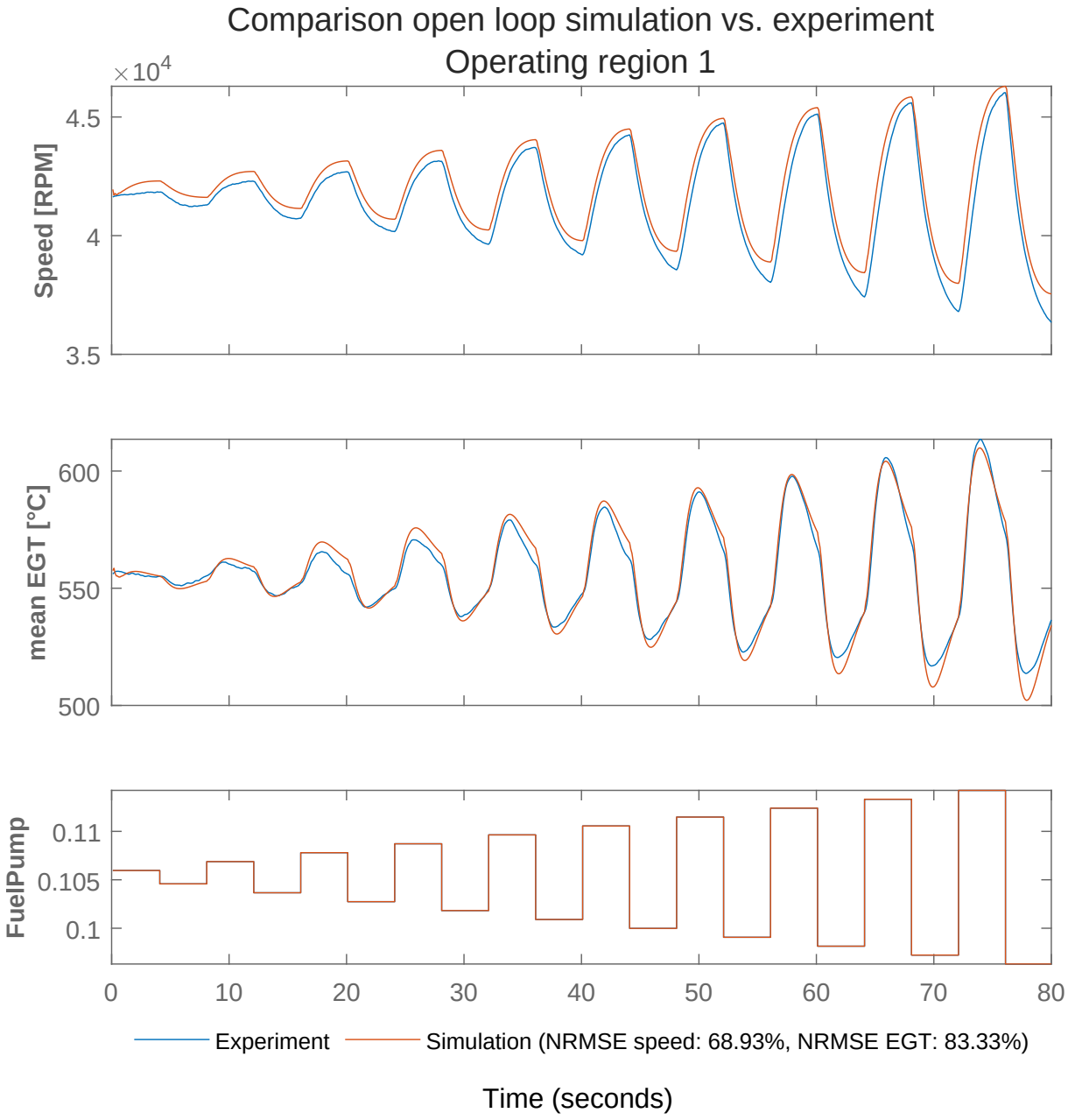



Figure 4.1: Comparison of the simulated open loop responses vs. hardware-in-the-loop (HIL)-experiment in operating region 1 for steps of increasing amplitude.

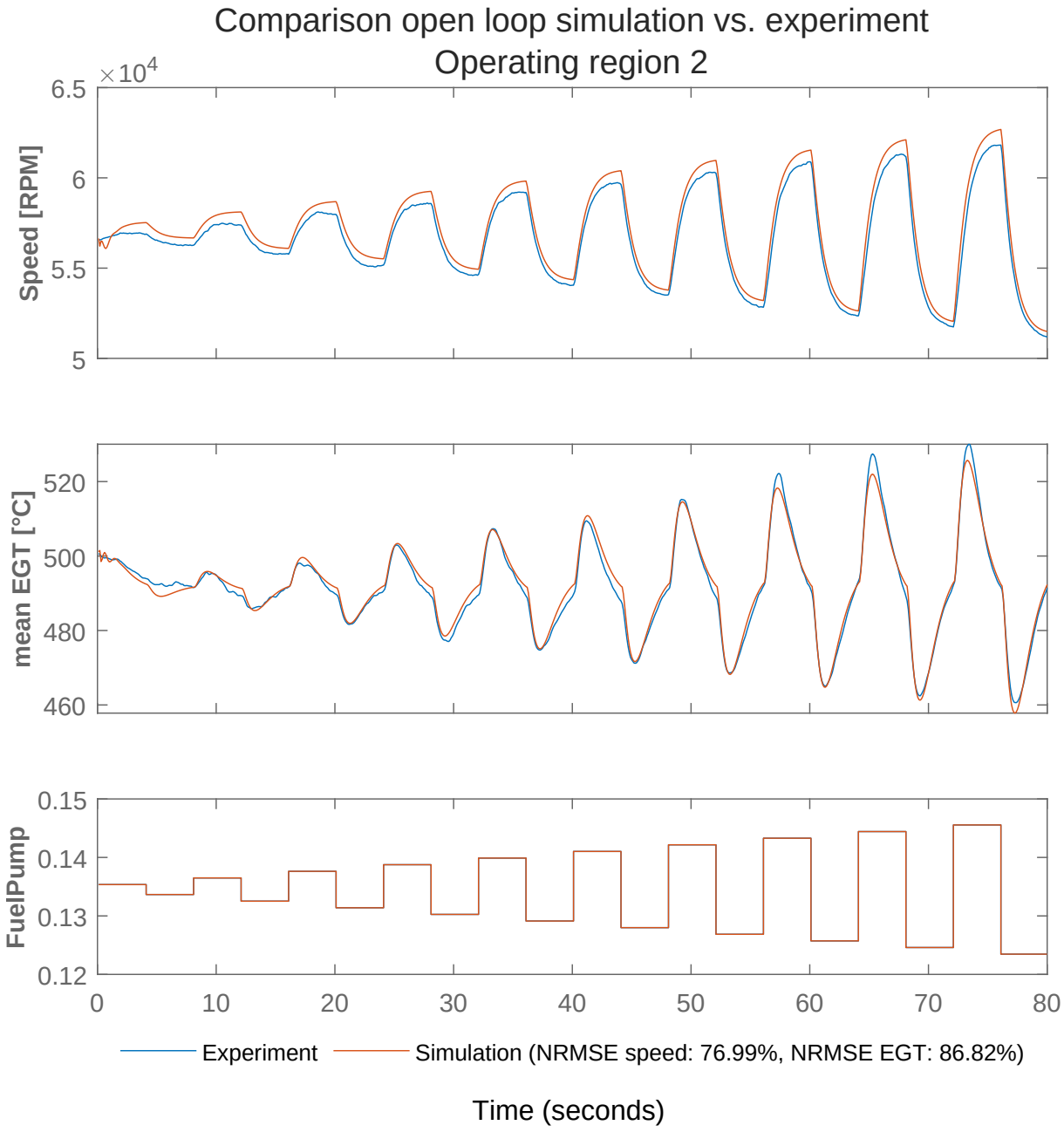



Figure 4.2: Comparison of the simulated open loop responses vs. HIL-experiment in operating region 2 for steps of increasing amplitude.

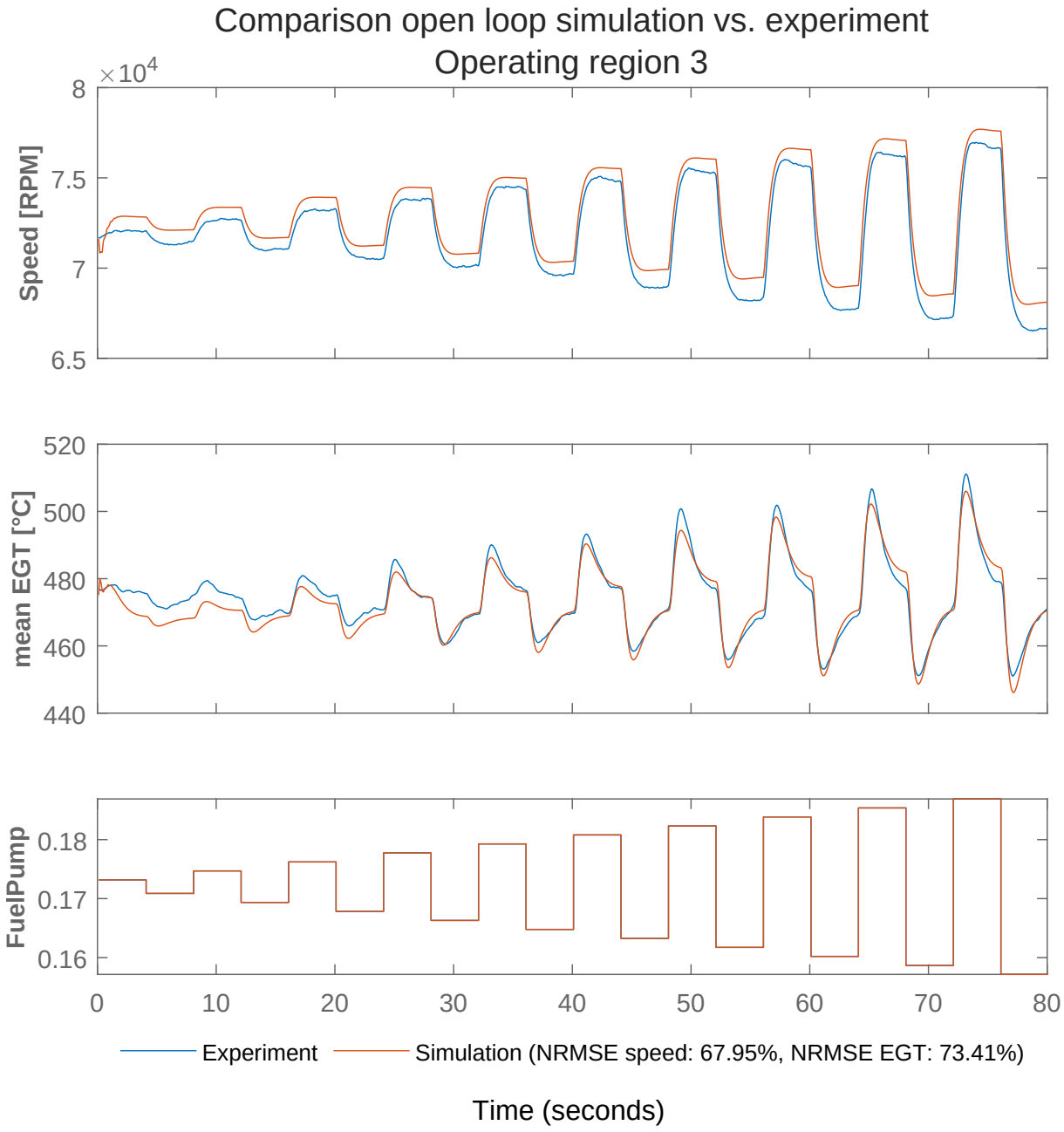



Figure 4.3: Comparison of the simulated open loop responses vs. HIL-experiment in operating region 3 for steps of increasing amplitude.

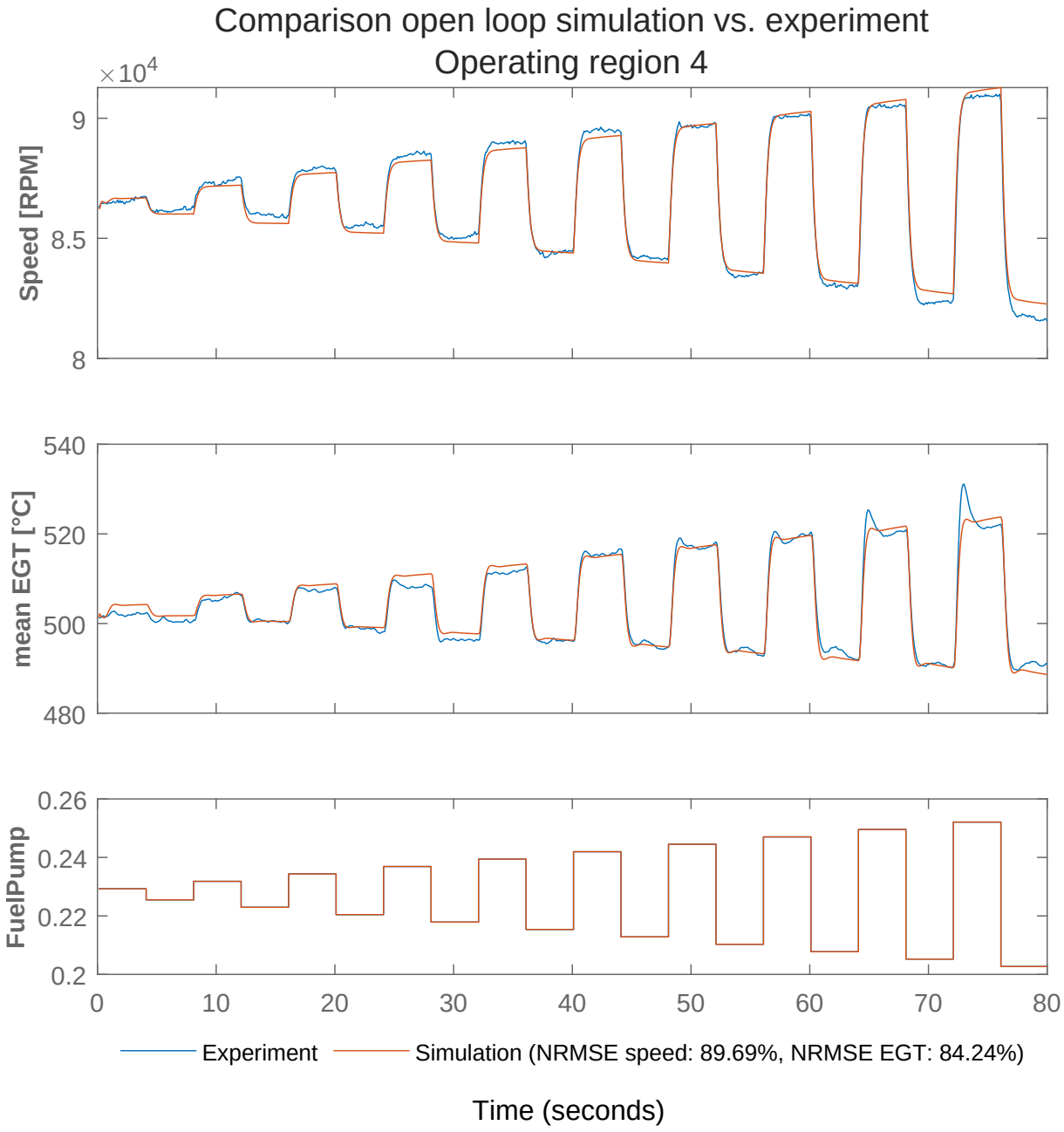



Figure 4.4: Comparison of the simulated open loop responses vs. HIL-experiment in operating region 4 for steps of increasing amplitude.

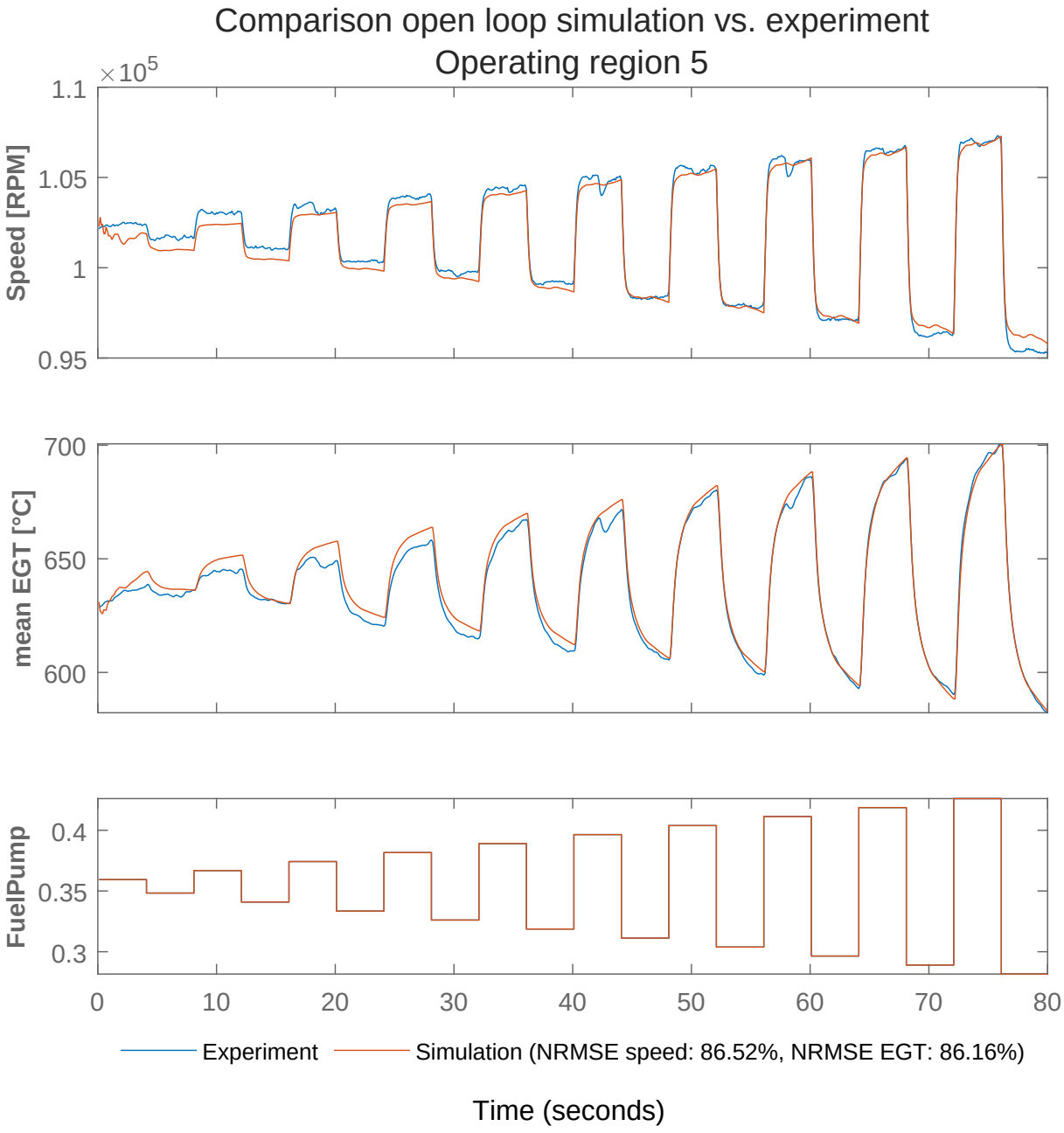



Figure 4.5: Comparison of the simulated open loop responses vs. HIL-experiment in operating region 5 for steps of increasing amplitude.

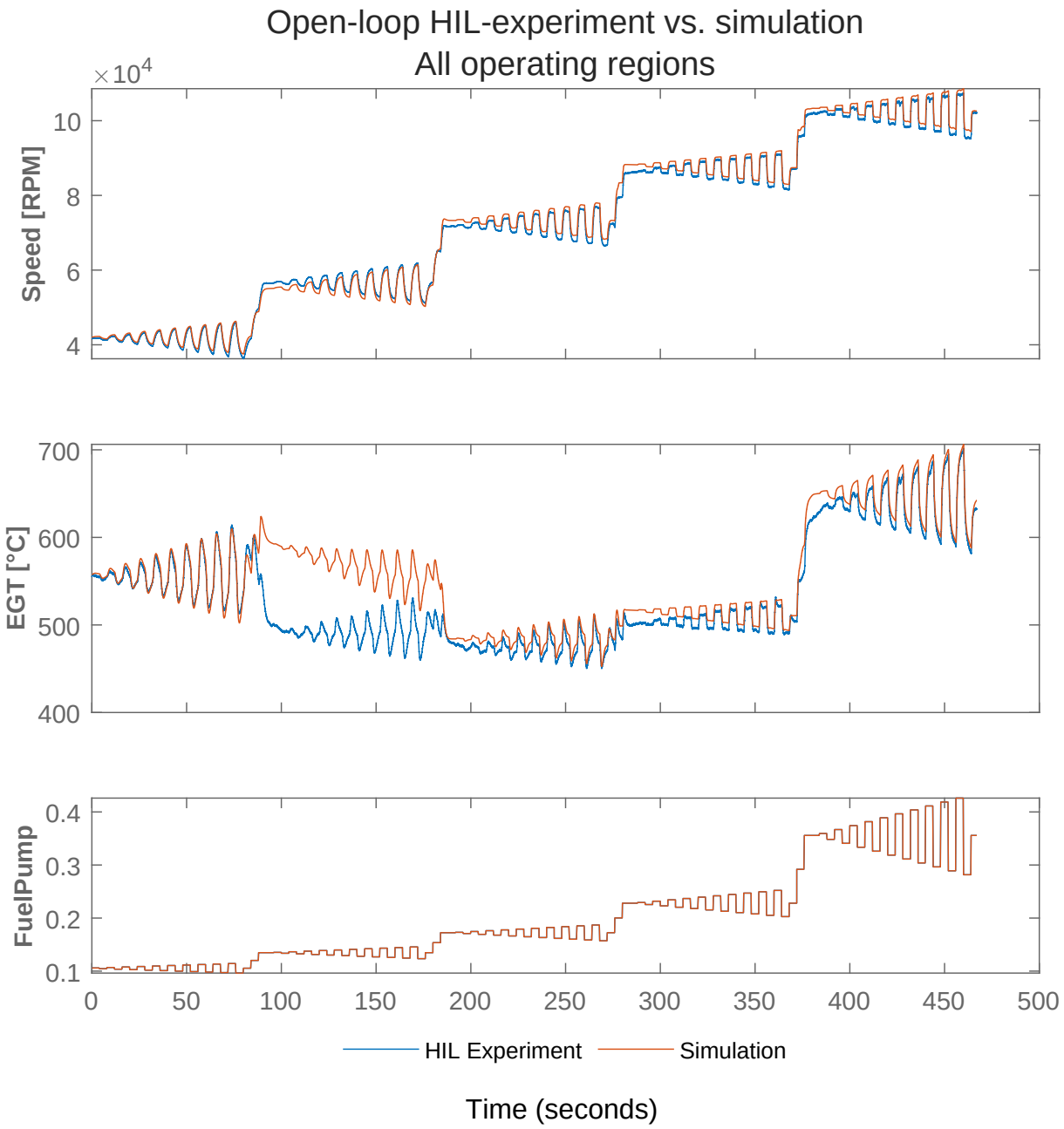



Figure 4.6: Comparison of the simulated open loop responses vs. HIL-experiment across the whole operating range.

In every operating region, steps of increasing amplitude and a width of 4 s are applied to the fuel pump control input.
At the transitions from one operating region to the other, considerable model mismatches can be seen. In the MPC controller, these mismatches are partly mitigated by re-estimating the internal states of the system in every step, using past control inputs and the output measurements.

## 4.2 MPC controller

### 4.2.1 Parameter selection MPC controller

The parameters for the prediction horizon $p$, the control horizon $c$, the fuel pump weight $w^u$ and the fuel pump move weight $w^{\Delta u}$ selected using the method described in Section 3.2.2 and using Figures B.1 to B.10 are listed in Table 4.2.

### 4.2.2 MPC optimization problem

The final optimization problem with its cost function and the constraints reads as follows:

| Operating region | Prediction horizon $p$ | Control horizon $c$ | Weight fuel pump $w^u$ | Weight fuel pump move $w^{\Delta u}$ | Maximum EGT constraint $T_{EGTmax}$ |
|---|---|---|---|---|---|
| 1 | 19 | 5 | 0.1 | 0.4 | 700 °C |
| 2 | 16 | 5 | 0.0 | 0.9 | 710 °C |
| 3 | 12 | 5 | 0.0 | 1.0 | 720 °C |
| 4 | 10 | 4 | 0.0 | 1.3 | 730 °C |
| 5 | 10 | 3 | 0.0 | 0.9 | 740 °C |

Table 4.2: Parameters used in the MPC controller in each operating region.

$$
\begin{aligned}
\min_{z_k} \quad J(z_k) = & \sum_{i=1}^{p} \left( \tfrac{3}{2} \left( \widetilde{r}\,(k) - \widetilde{N}_{RPM}\,(k+i|k) \right) \right)^2 + \dots \\
& + \sum_{i=0}^{c-1} \left( \frac{w^u}{0.41} u\,(k+i|k) \right)^2 + (p-c-1) \cdot \left( \frac{w^u}{0.41} u\,(k+c-1|k) \right)^2 + \dots \\
& + \sum_{i=0}^{c-1} \left( \frac{w^{\Delta u}}{0.41} \left( u\,(k+i|k) - u\,(k+i-1|k) \right) \right)^2 + \dots \\
& + 100\,000 \cdot \varepsilon_k^2
\end{aligned}
$$

subject to

$$
\begin{aligned}
0.071 \leq \quad & u\,(i|k) && \leq 0.7 && , \; i = 0, \dots, c-1 \\
\frac{30\,000\,\mathrm{RPM}}{108\,000\,\mathrm{RPM}} - \varepsilon_k \cdot \tfrac{2}{3} \cdot 1 \leq \quad & \widetilde{N}_{RPM}\,(i|k) && \leq \frac{108\,000\,\mathrm{RPM}}{108\,000\,\mathrm{RPM}} + \varepsilon_k \cdot \tfrac{2}{3} \cdot 1 && , \; i = 1, \dots, p \\
& \widetilde{T}_{EGT}\,(i|k) && \leq \frac{T_{EGT\,max}}{750\,^{\circ}\mathrm{C}} + \varepsilon_k \cdot \tfrac{2}{3} \cdot 0.2 && , \; i = 1, \dots, p \\
0 \leq \quad & \varepsilon_k
\end{aligned}
$$

with

$$
\begin{aligned}
J &:= \text{Cost function} \\
z_k &:= [u(k|k) \; u(k+1|k) \; \cdots \; u(k+c-1|k) \; \varepsilon_k]^T \\
&\quad \text{Quadratic programming (QP) decision at time step } k \text{ (free variable)} \\
u(j|k) &:= \text{Optimal control input to the fuel pump at time } j, \\
&\quad \text{evaluated at time } k \text{ (free variable)} \\
\varepsilon_k &:= \text{Slack variable (free variable)} \\
p &:= \text{Prediction horizon (from Table 4.2)} \\
c &:= \text{Control horizon (from Table 4.2)} \\
\widetilde{r}\,(k) &:= \text{Normalized reference setpoint for the engine rotor speed} \\
&\quad \text{at time } k \text{ (reference input variable)} \\
\widetilde{N}_{RPM}\,(i|k) &:= \text{Predicted, normalized engine rotor speed for time } i, \\
&\quad \text{evaluated at time } k \text{ (predicted using the plant model and } z_k) \\
w^u &:= \text{Tuning weight for the control variable penalization (from Table 4.2)} \\
w^{\Delta u} &:= \text{Tuning weight for the control variable move penalization (from Table 4.2)} \\
\widetilde{T}_{EGT}\,(i|k) &:= \text{Predicted, normalized EGT for time } i, \\
&\quad \text{evaluated at time } k \text{ (predicted using the plant model and } z_k) \\
T_{EGT\,max} &:= \text{Maximum constraint for the EGT (from Table 4.2)}
\end{aligned}
\tag{4.1}
$$

For each of the operating regions, the cost function is built using the weights, horizons and EGT constraint presented in Table 4.2. In every control interval, the active operating region is identified as described in Section 3.1.2 and the corresponding QP problem is solved.

### 4.2.3 MPC controller evaluation

The evaluation of candidate MPC controllers is done according to the steps in Section 3.2.3, the results are presented and discussed in this section.

#### Simulation and experiments of controllers within a single operating region

The plots with the simulation results plotted alongside the experimental data can be seen in Figures 4.7 to 4.11, in Table 4.3 the metrics described in Section 3.2.2 are presented.
The following things are observed:

- The initial condition of the EGT has a considerable deviation between experiment and simulation. This illustrates the observation made in many of the experiments, that the EGT has a high dependency on the previous operation of the engine.

- The steady state error is considerably higher in the experiment compared to the simulation. While a small proportion of it can be attributed to measurement noise, which is not simulated and thus not accounted for in the performance metrics for simulation data, the reason for most of the mismatch is not entirely known and has to be investigated further. The hypothesis however is that the mismatch in the EGT leads to errors in state estimation and thus in the prediction of the outputs over the prediction horizon. This hypothesis is supported by the observed correlation between the amplitudes of the steady state error and the EGT mismatch over several operating regions. To correct for this, the state estimator would have to be redesigned to weight the output channels accordingly.
  The differences in the $\text{NRMSE}_{\text{ref}}$ are due to the mismatches in the steady state of the system.

- The model underestimates the over- and undershoots in both speed and EGT output, which can be seen in the higher amplitude of the transients in the experimental data w.r.t. the transients in the simulation data and in the over-/undershoot metric. Also the settling time is underestimated. This can be attributed to the remaining non-linearity in the system dynamics within single operating regions, which are not accurately captured in the linear model.

- In the data for operating region one seen in Figure 4.7, it can be observed how the EGT exceeds the set EGT constraint of 700 °C for this region. The constraint violation is assumedly not predicted early enough due to the reasons put forth in the previous point. After the constraint violation becomes evident because of the state re-estimations using measurement data, it is corrected for by a drastic decrease in the fuel pump control input, which causes the dip in the speed line.
  The simulation does not expose the same behavior, because the initial EGT value is lower and because the model predicts a lower amplitude of the EGT response.

- While the absolute amplitude of the in- and output signals exposes considerable mismatch between simulation and experiment, the general features and shape of the control action and the system response is well predicted by the simulations. Also, the fuel consumption is predicted with good accordance except for operating region 5.

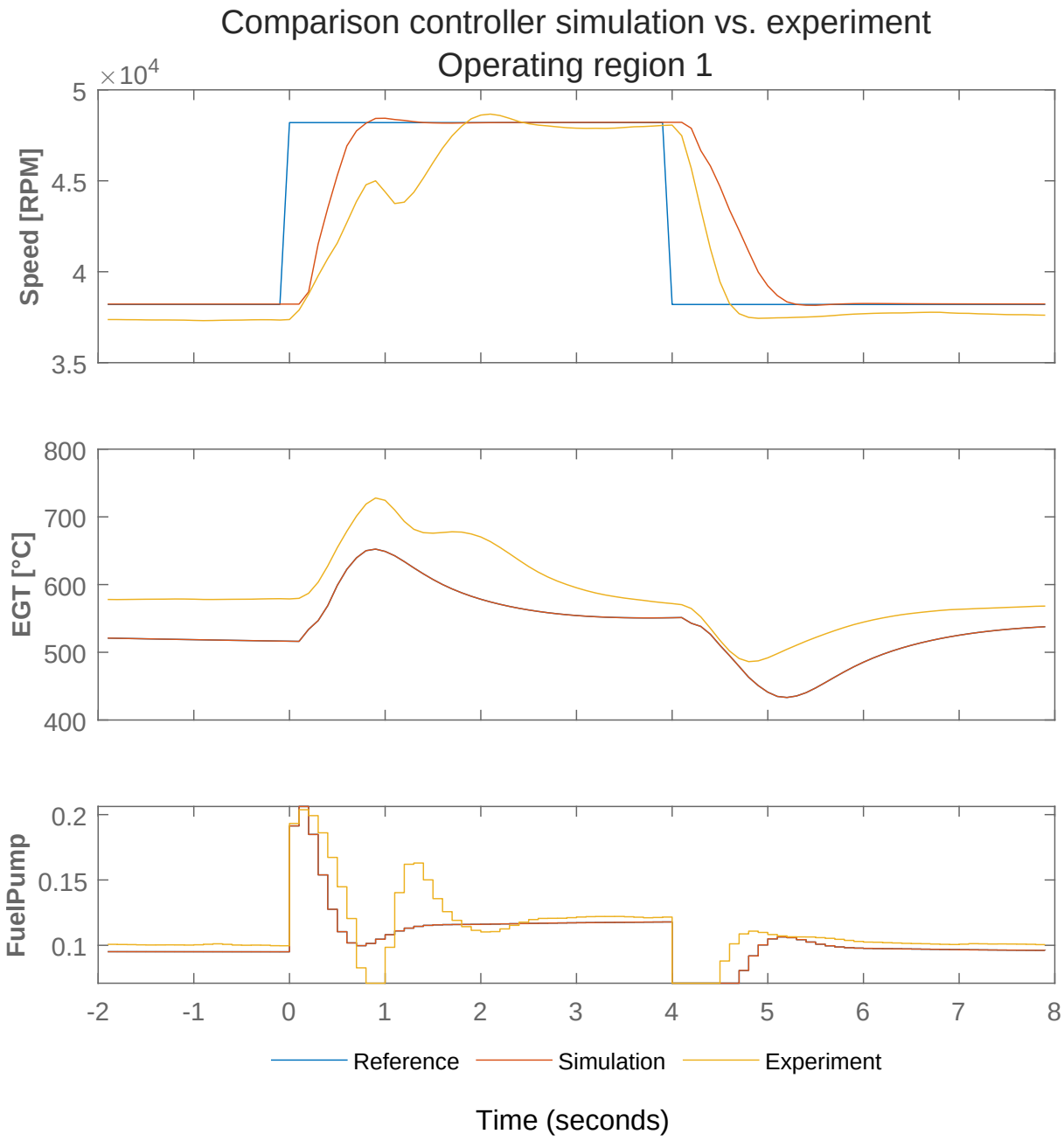



Figure 4.7: Comparison of the simulation and the HIL-experiment in operating region 1 for a step of 10 000RPM.

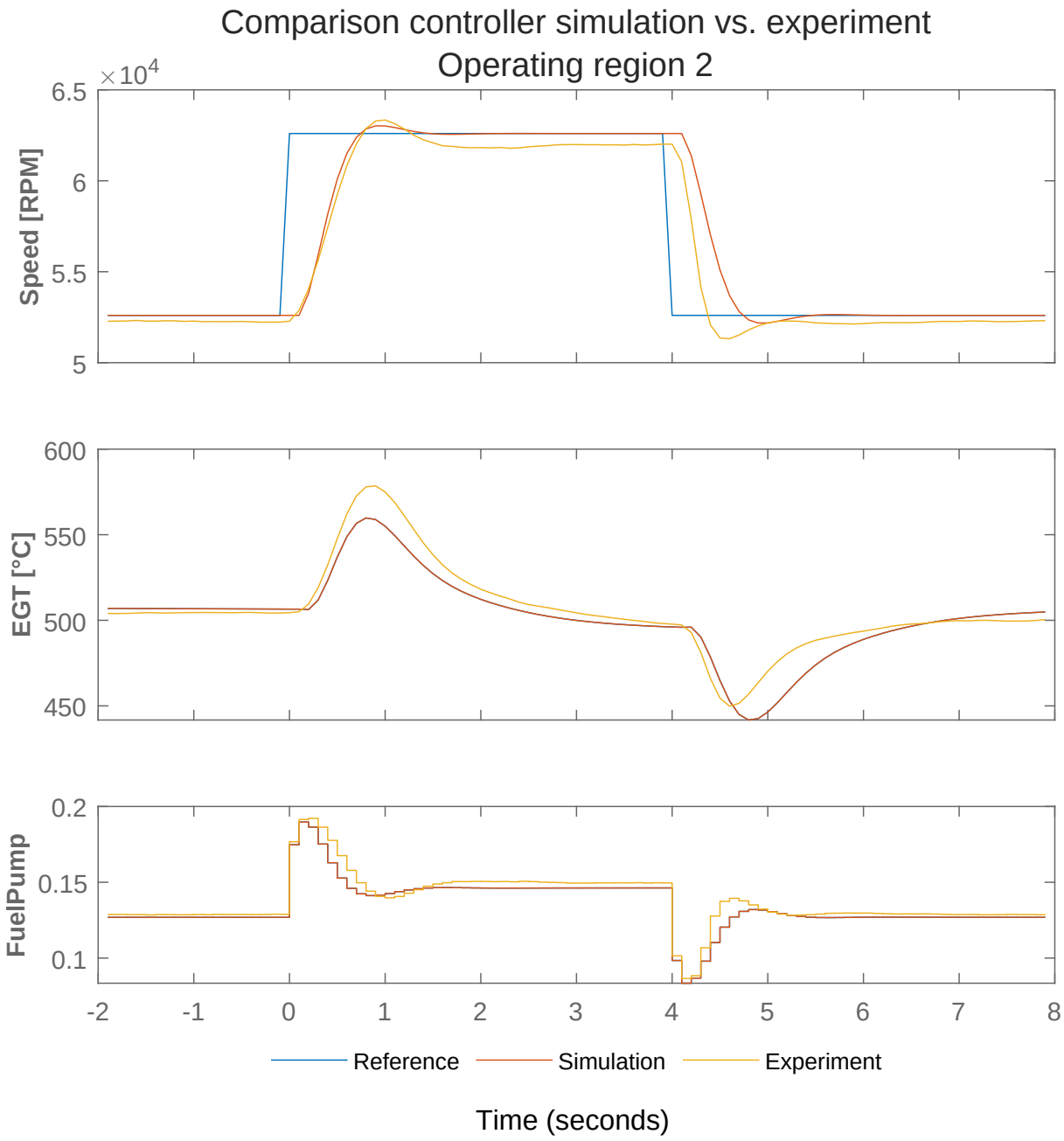



Figure 4.8: Comparison of the simulation and the HIL-experiment in operating region 2 for a step of 10 000RPM.

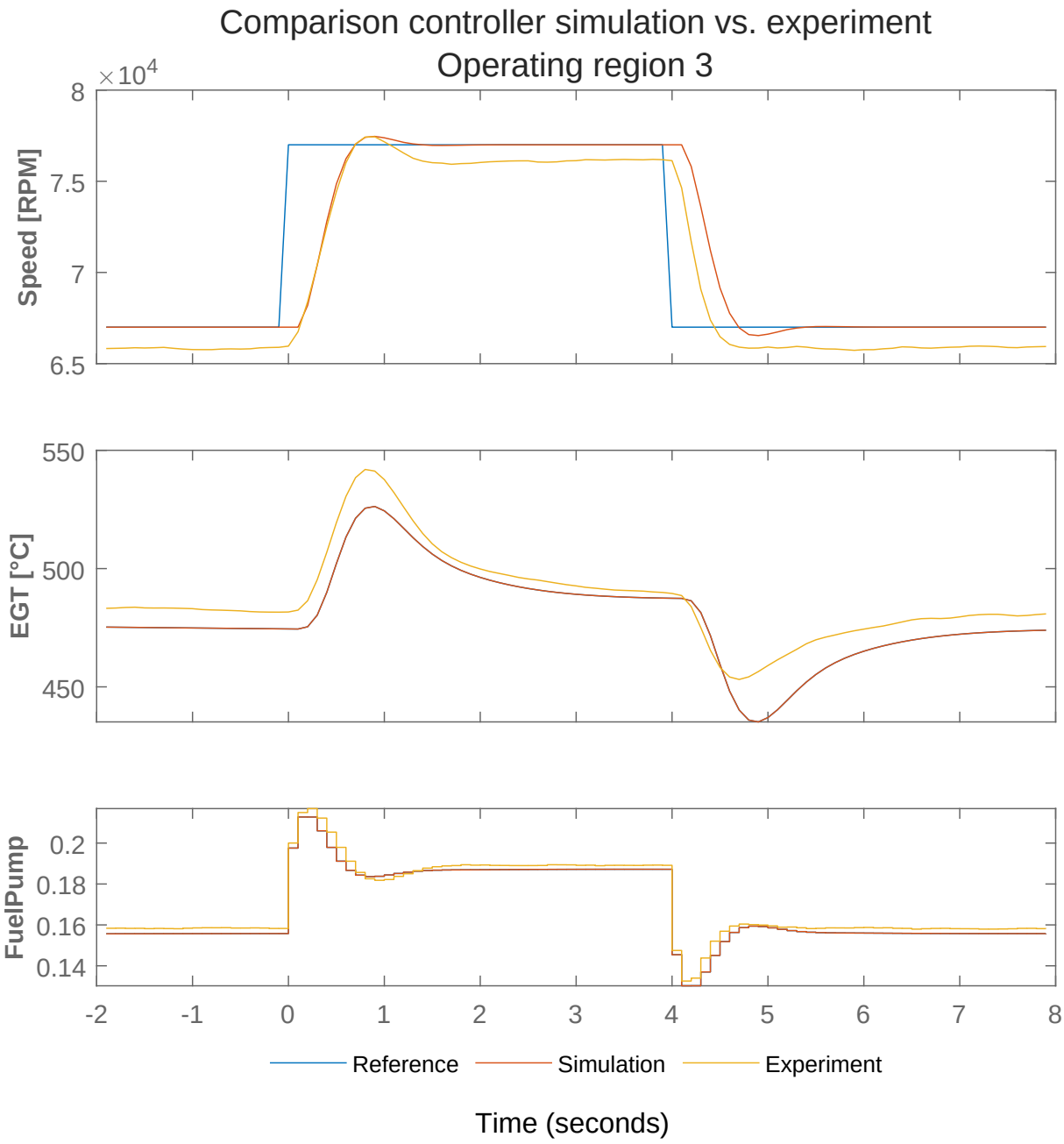



Figure 4.9: Comparison of the simulation and the HIL-experiment in operating region 3 for a step of 10 000RPM.

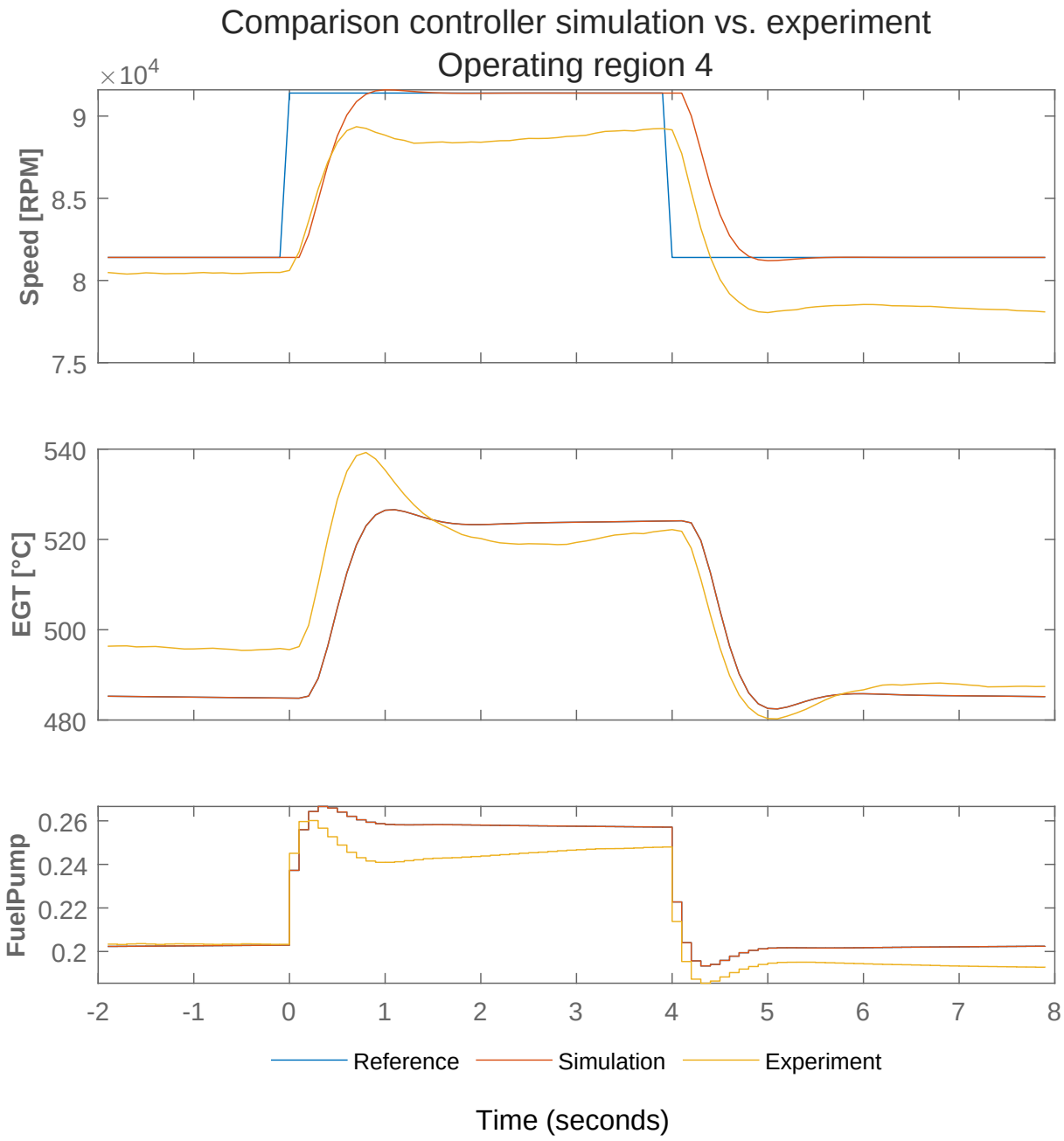



Figure 4.10: Comparison of the simulation and the HIL-experiment in operating region 4 for a step of 10 000RPM.

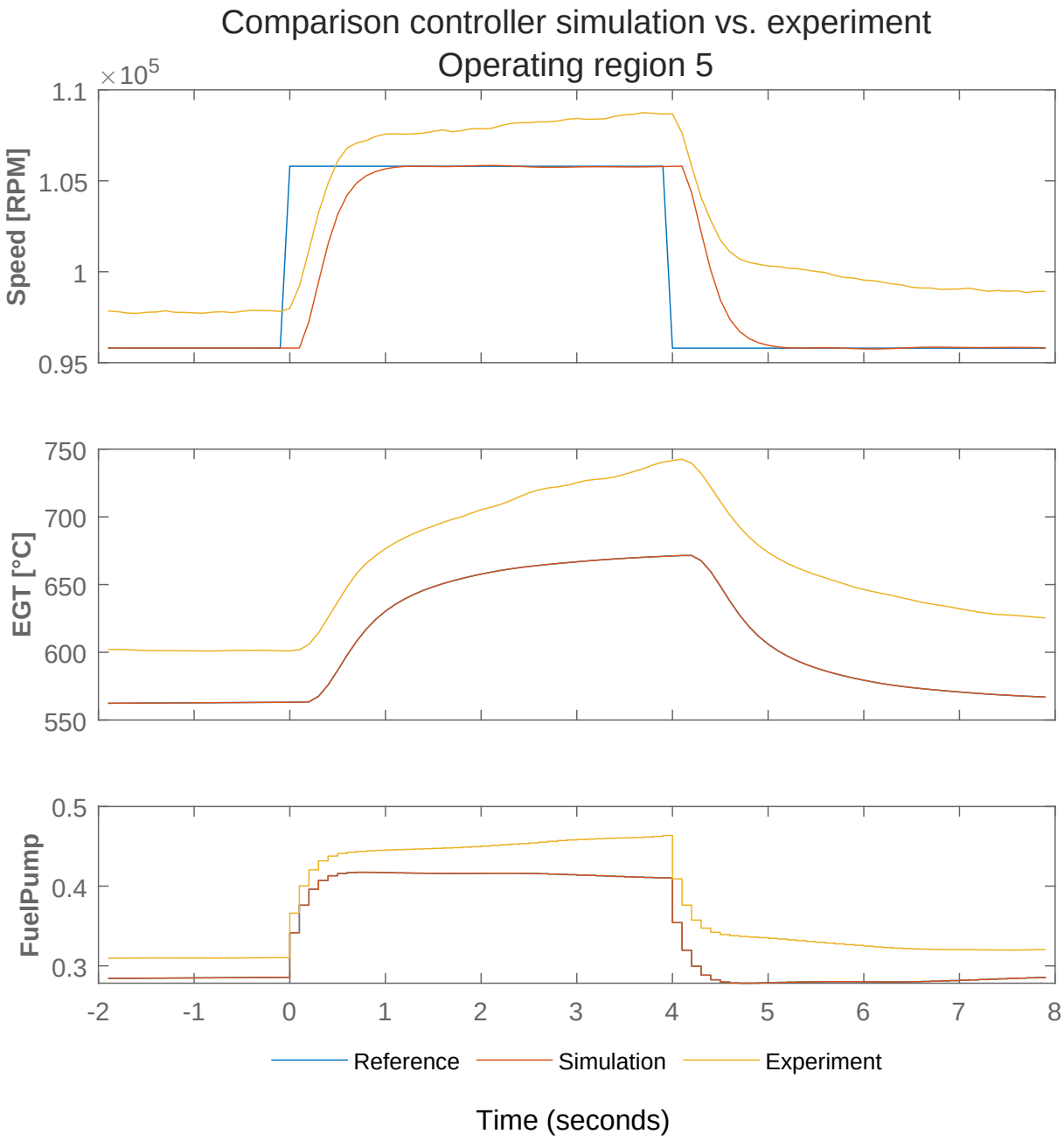



Figure 4.11: Comparison of the simulation and the HIL-experiment in operating region 5 for a step of 10 000RPM.

| Operating region | Rise/Fall time [s] | | Settling time [s] | | Over/Undershoot [%] | |
|---|---|---|---|---|---|---|
| | Simulation | Experiment | Simulation | Experiment | Simulation | Experiment |
| 1 | 0.70 | 1.07 | 1.26 | 4.18 | 1.16 | 2.50 |
| 2 | 0.53 | 0.45 | 0.86 | 2.19 | 2.31 | 6.41 |
| 3 | 0.49 | 0.5 | 0.75 | 1.64 | 2.50 | 4.33 |
| 4 | 0.58 | 0.4 | 1 | N/A | 1.31 | 9.56 |
| 5 | 0.67 | N/A | 1.33 | N/A | 0.97 | 3.78 |

| Operating region | Steady state error [%] | | $NRMSE_{ref}$ [%] | | Fuel consumption [mL] | |
|---|---|---|---|---|---|---|
| | Simulation | Experiment | Simulation | Experiment | Simulation | Experiment |
| 1 | 0.05 | 2.22 | 39.0 | 37.1 | 24 | 26 |
| 2 | 0 | 0.67 | 45.8 | 50.7 | 35 | 36 |
| 3 | 0 | 1.72 | 46.2 | 48.5 | 46 | 47 |
| 4 | 0 | 1.15 | 46.2 | 34.3 | 64 | 62 |
| 5 | 0 | 2.12 | 46.3 | 22.3 | 96 | 107 |

Table 4.3: Comparison of performance metrics of the controller simulation and the experiments in every operating region.

The calculation of the performance metrics is explained in Section 3.2.2. For $NRMSE_{ref}$, a higher value is better, for all other metrics lower is better.

**Simulation and experiments spanning the whole operating range**

To evaluate the composite gain-scheduled controller, several reference trajectories spanning multiple operating regions are given to the controller and the in- and outputs from the real-time simulation and the HIL experiment is compared. The data in presented in this section and a short discussion of the results is given.
In Figure 4.12, the real-time simulation to the reference trajectory used to generate the data for Figures 4.7 to 4.11 is compared to the experimental data. The real-time simulation as seen in Figure 4.12 and the simulations as seen in Figures 4.7 to 4.11 differ in the initial conditions for each operating region and in the environment where the simulation is performed.
At the transition from operating region 3 to operating region 4 around 90 s, an oscillation between the two operating regions can be observed in the experimental data, the same phenomenon can also be seen in Figure 4.13 and is discussed in the next paragraph.
Figure 4.13 shows the real-time simulation and the experimental data for a ramp reference from idle speed of 36 000 RPM to full speed of 108 000 RPM over a duration of 40 s. The same oscillations between operating regions 3 and 4 can be observed in the experimental data as in Figure 4.12, however in this plot the cause can be identified. In operating region 4, the steady state error is much increased compared to the other operating regions. This is visible both in the experimental data and in the real-time simulation, as opposed to the simulations in Figures 4.7 to 4.11, the simulation doesn't start with the nominal operating conditions as initial conditions. As discussed in Section 4.2.3, this problem is expected to improve with a revised state estimator design.
During the negative slope of the reference signal at the transition from operating region 3 to 2 and 2 to 1 respectively, the simulator largely overestimates the EGT. This is assumed to be an effect of the remaining non-linearity in the plant. Because the simulated EGT exceeds the safety limit for the engine, the safety module iniates a forced shutdown around 36 s.
In Figure 4.14, the real-time simulation and the experimental data for a reference step from idle speed to full speed can be seen. While both in the simulation and the experimental data the EGT gets close to the maximum, the underestimated overshoot of the model causes a slight exceedance of the EGT safety limit, which causes the safety module of the controller to initiate a forced shutdown. Hence, it was not possible to record the response to a proper reference step over the whole operating region.

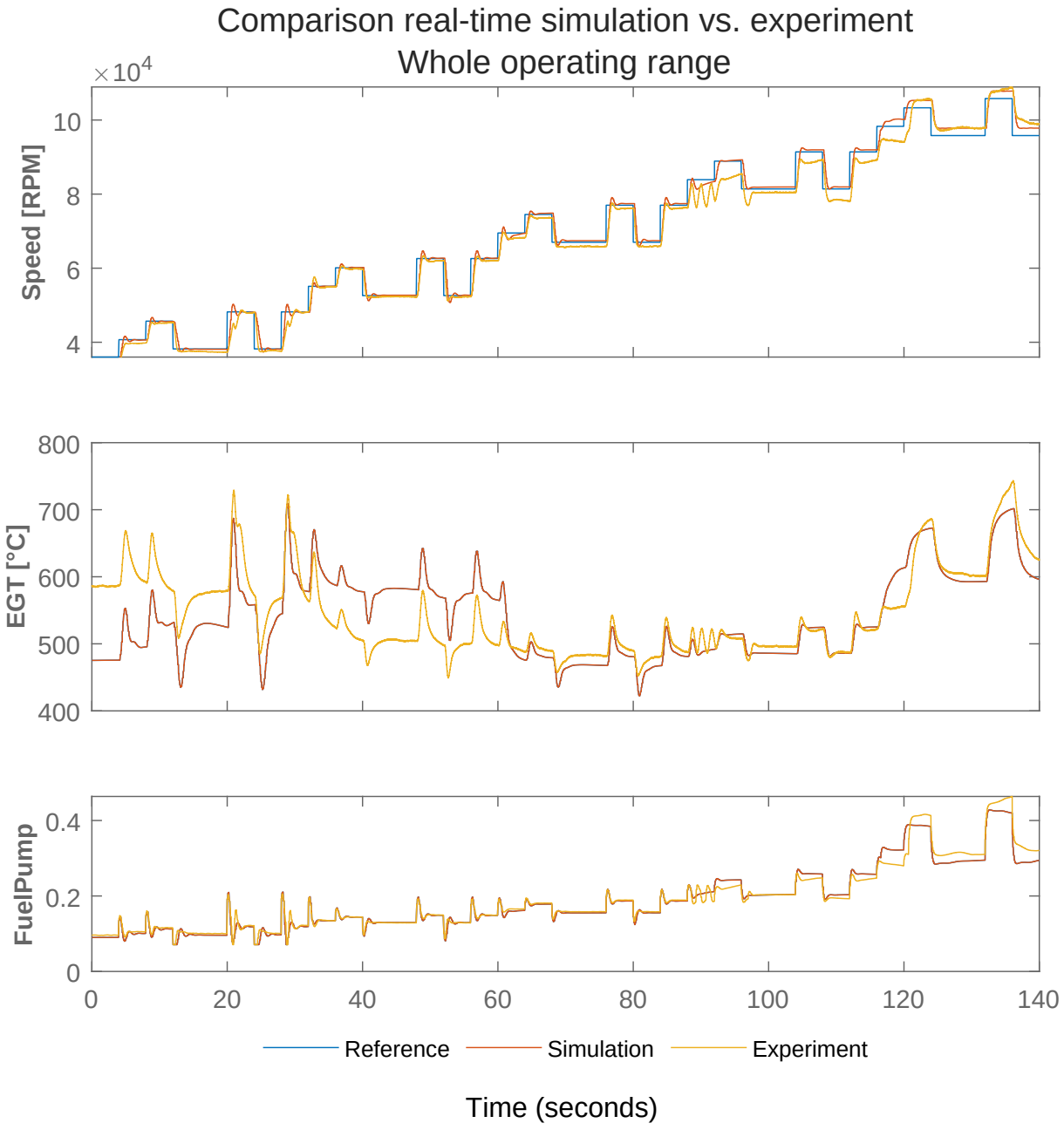



Figure 4.12: Comparison of the real-time simulation and the HIL-experiment for the same reference trajectory as used to generate the experimental data for Figures 4.7 to 4.11.

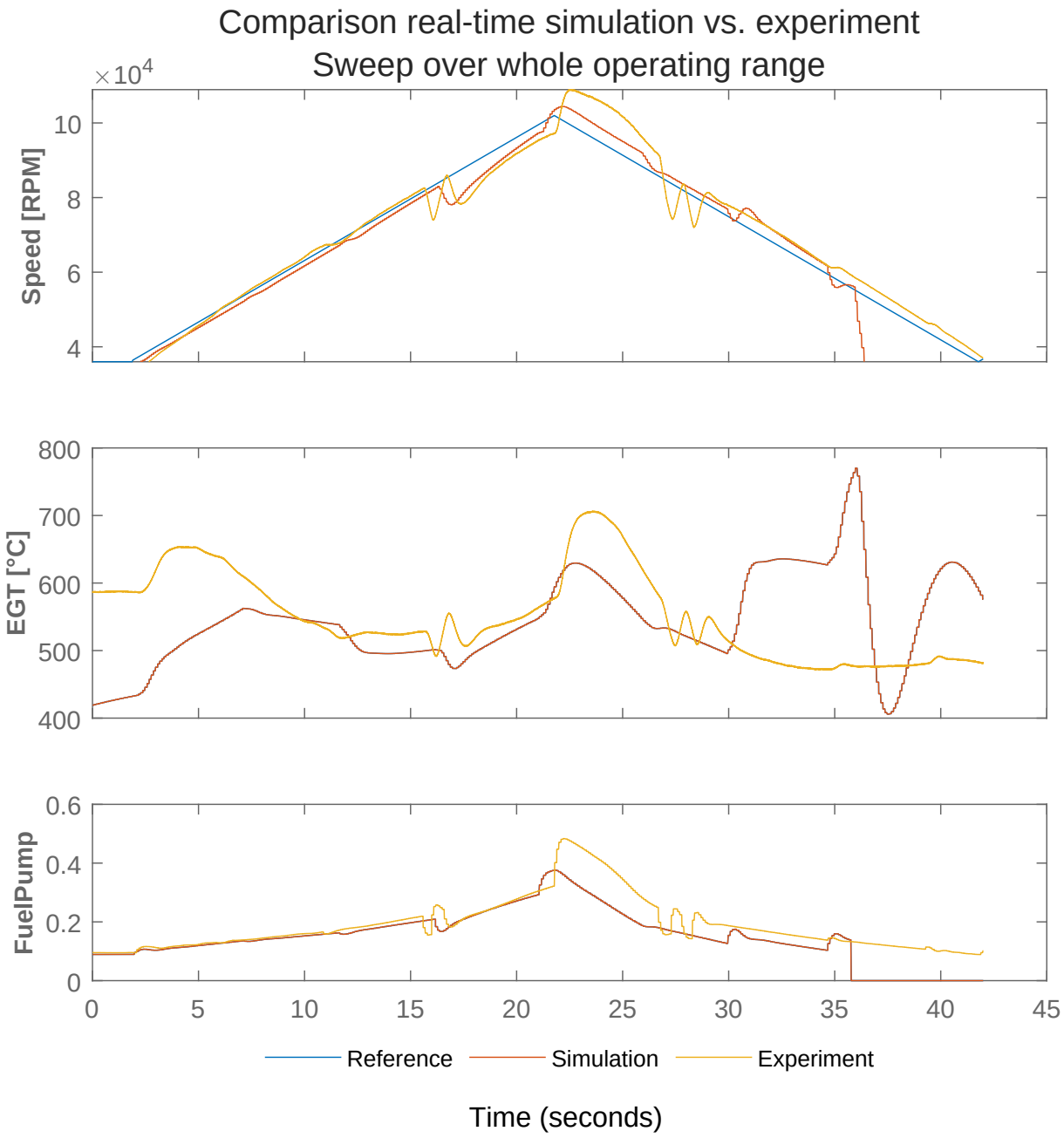



Figure 4.13: Comparison of the simulated response and the experiment for a triangle reference sweeping over the whole operating range.

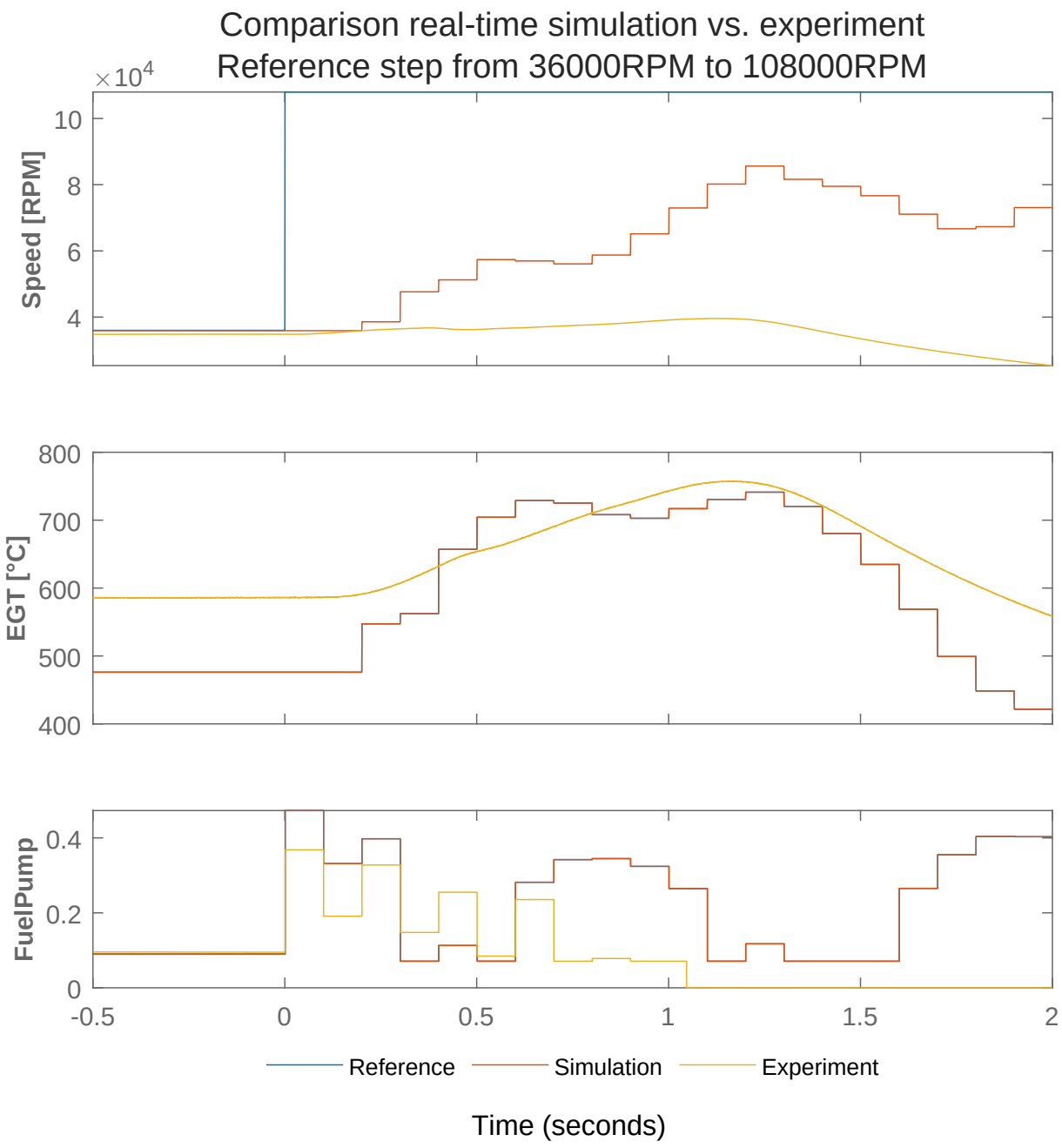



Figure 4.14: Comparison of the simulated response and the experiment for a reference step from 36 000 RPM to 108 000 RPM.

# Chapter 5

# Conclusion

During this thesis, the following things were achieved:

- A test stand for the AMT Olympus HP micro-gas turbine (MGT) is designed. It is equipped with sensors for the engine shaft speed, four channels for thermocouples, two flow rate sensors and ten pressure transducers. It comes with a dSPACE MicroLabBox real-time controller, which runs the control algorithm, controls the inputs of the MGT and reads the sensor values. While the test bed is designed for the AMT Olympus HP, the setup is extensible and can be adapted for different engines.
- The corresponding software framework is developed. It incorporates code to start and shutdown the MGT and ensures the operation of the engine within safe limits. New control algorithms can be implemented using MathWorks Simulink, embedded into this framework and tested in simulation and on the physical engine. This enables to quickly implement and test a variety of new control algorithms for MGTs.
- Discrete-time state-space models are identified for five different operating regions of the engine using Numerical algorithms for Subspace State Space System IDentification (N4SID).
- For each operating region, a model predictive control (MPC) controller is designed using the previously identified models.
- The controllers are implemented and tested both in simulation and on the real-time hardware. The results are presented and evaluated.

While the identified models can simulate the MGT behavior with good accuracy and the designed MPC controllers have acceptable performance within the operating region they are designed for, the control of the engine across several operating regions still shows significant problems. The following things need further investigation to obtain a robust and user-friendly controller:

- The state estimator design has to be improved.
- To prevent exceedances of the exhaust gas temperature (EGT) limit when a reference step of high amplitude is prescribed, a reliable method has to be found to predict EGT overshoots also in off-design conditions.

### Outlook

The work in the present thesis could be extended, ideas for future investigations include:

**Improve state estimation with additional measurement signals** The multitude of available signals in the test bed could be leveraged by incorporating them in the engine model and exploiting them for improved state estimation.

**Online state estimation** As done in [31], the MGT model could be estimated online. Doing so would ease the controller design process, as part of the model identification process is done in an online fashion. Also, model drift and changes in the environmental conditions might be mitigated automatically.

**Control of different engine models** While the presented method to design an MPC controller for MGTs is expected to be applicable to other MGT models with little modifications, this should be tested by implementing an MPC controller for other MGTs.

**Non-linear models and control** The literature review in [13] finds that for the control of gas turbines increasingly non-linear models and controllers are employed. Such an approach could be tested on the AMT Olympus HP MGT, to compare the performance characteristics with the array of linear controllers developed in this thesis.

# Bibliography


[1] AMT Netherlands. *Olympus HP gasturbine Manual & engine log (V30 C)*. Ed. by B.J.J. van de Goor. 2013.

[2] AMT Netherlands. *Olympus HP in the University configuration*. Feb. 2020. URL: www.amtjets.com/pdf/Olympus-HP-in-University-config-feb-2020.pdf.

[3] AMT Netherlands. *Olympus HP specification*. URL: https://www.amtjets.com/pdf/Olympus_HP_specification.pdf.

[4] Lukas Badum, Boris Leizeronok, and Beni Cukurel. "New Insights From Conceptual Design of an Additive Manufactured 300 W Microgas Turbine Toward Unmanned Aerial Vehicle Applications". In: *Journal of Engineering for Gas Turbines and Power* 143.2 (Jan. 18, 2021). DOI: 10.1115/1.4048695. URL: https://doi.org/10.1115/1.4048695 (visited on 04/14/2023).

[5] Stefano Bracco and Federico Delfino. "A mathematical model for the dynamic simulation of low size cogeneration gas turbines within smart microgrids". In: *Energy* 119 (Jan. 15, 2017), pp. 710–723. DOI: 10.1016/j.energy.2016.11.033. URL: https://www.sciencedirect.com/science/article/pii/S0360544216316383 (visited on 03/23/2023).

[6] Duwei Chen. *Micro Gas Turbine Project Setup*. Tech. rep. IDSC ETH Zürich, Sept. 2011.

[7] Zin Eddine Dadach. "Applied Research Activities Chemical & Petroleum Engineering Department". May 27, 2018.

[8] dSPACE. *MicroLabBox. Compact prototyping unit for the laboratory*. 2023. URL: https://www.dspace.com/shared/data/pdf/2020/dSPACE-MicroLabBox_Product-Brochure_2020-01_EN.pdf (visited on 04/12/2023).

[9] K. Goorts and S. Narasimhan. "Adaptive Model Predictive Control for Deployable Control Systems with Constraints". In: *Journal of Structural Engineering* 145.10 (Oct. 1, 2019), p. 04019110. DOI: 10.1061/(ASCE)ST.1943-541X.0002392. URL: https://ascelibrary.org/doi/10.1061/%28ASCE%29ST.1943-541X.0002392 (visited on 04/30/2023).

[10] Zafer Leylek et al. "An Investigation Into Performance Modeling of a Small Gas Turbine Engine". In: *Proceedings of ASME Turbo Expo 2013*. ASME Turbo Expo 2013: Turbine Technical Conference and Exposition (June 3, 2013). 2013.

[11] Ping Lin et al. "Modeling and controller design of a micro gas turbine for power generation". In: *ISA Transactions* 124 (May 1, 2022), pp. 411–426. DOI: 10.1016/j.isatra.2020.05.050. URL: https://www.sciencedirect.com/science/article/pii/S001905782030238X (visited on 04/16/2023).

[12] Lennart Ljung. *System Identification. Theory for the User (2nd Edition)*. Prentice Hall PTR, 1998, p. 609.

[13] Chengkun Lv et al. “Recent research progress on airbreathing aero-engine control algorithm”. In: *Propulsion and Power Research* 11.1 (Mar. 1, 2022), pp. 1–57. DOI: 10.1016/j.jppr.2022.02.003. URL: https://www.sciencedirect.com/science/article/pii/S2212540X22000177 (visited on 04/15/2023).

[14] Peter Van Overschee and Bart De Moor. *Subspace Identification for Linear Systems*. Springer US, 1996. DOI: 10.1007/978-1-4613-0465-4.

[15] S. Joe Qin. “An overview of subspace identification”. In: *Computers & Chemical Engineering*. Papers form Chemical Process Control VII 30.10 (Sept. 12, 2006), pp. 1502–1513. DOI: 10.1016/j.compchemeng.2006.05.045. URL: https://www.sciencedirect.com/science/article/pii/S009813540600158X (visited on 03/05/2023).

[16] Johannes F. Rist et al. “Economic dispatch of a single micro-gas turbine under CHP operation”. In: *Applied Energy* 200.Germany (Aug. 2017), pp. 1–18. DOI: 10.1016/j.apenergy.2017.05.064.

[17] Alessandro Rosini et al. “A Model Predictive Control Design for Power Generation Heavy-Duty Gas Turbines”. In: *Energies* 12.11 (Jan. 2019), p. 2182. DOI: 10.3390/en12112182. URL: https://www.mdpi.com/1996-1073/12/11/2182 (visited on 02/19/2023).

[18] C. Schmid and L.T. Biegler. “Quadratic programming methods for reduced hessian SQP”. In: *Computers & Chemical Engineering* 18.9 (Sept. 1994), pp. 817–832. DOI: 10.1016/0098-1354(94)E0001-4. URL: https://linkinghub.elsevier.com/retrieve/pii/0098135494E00014 (visited on 05/01/2023).

[19] Miel Sharf et al. “Economic dispatch of a single micro gas turbine under CHP operation with uncertain demands”. In: *Applied Energy* 309 (Mar. 2022), p. 118391. DOI: 10.1016/j.apenergy.2021.118391.

[20] terraholdingco. *Jet Powered Blower*. Mar. 4, 2019. URL: https://www.youtube.com/watch?v=BcEkt5vQTVQ (visited on 04/12/2023).

[21] Inc. The MathWorks. *Implement Custom State Estimator Equivalent to Built-In Kalman Filter*. 2023. URL: https://ch.mathworks.com/help/releases/R2022a/mpc/ug/design-estimator-equivalent-to-mpc-built-in-kf.html (visited on 05/01/2023).

[22] Inc. The MathWorks. *Optimization Problem*. 2023. URL: https://ch.mathworks.com/help/releases/R2022a/mpc/ug/optimization-problem.html (visited on 05/01/2023).

[23] Inc. The MathWorks. *QP Solver*. 2023. URL: https://ch.mathworks.com/help/releases/R2022a/mpc/ug/qp-solver.html (visited on 05/01/2023).

[24] Inc. The MathWorks. *Quadratic Programming Algorithms*. 2023. URL: https://ch.mathworks.com/help/releases/R2022a/optim/ug/quadratic-programming-algorithms.html (visited on 05/01/2023).

[25] Peter Van Overschee and Bart De Moor. “N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems”. In: *Automatica*. Special issue on statistical signal processing and control 30.1 (Jan. 1, 1994), pp. 75–93. DOI: 10.1016/0005-1098(94)90230-5. URL: https://www.sciencedirect.com/science/article/pii/0005109894902305 (visited on 04/03/2023).

[26] Rejin John Varghese. *Model based Control of a Micro-turbine*. Tech. rep. ETHZ - DMAVT - IDSC, Dec. 2010.

[27] Bo Wahlberg et al. “Experiences from Subspace System Identification - Comments from Process Industry Users and Researchers”. In: *Lecture Notes in Control and Information Sciences*. Vol. 364. Oct. 24, 2007, pp. 315–327. DOI: 10.1007/978-3-540-73570-0_24.

[28] Wikipedia contributors. *Flameout*. Ed. by The Free Encyclopedia. Wikipedia. May 2, 2023. URL: https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Flameout&id=1137349162&wpFormIdentifier=titleform.

[29] Wikipedia contributors. *Gas turbine*. Ed. by The Free Encyclopedia. Wikipedia. Mar. 18, 2023. URL: https://en.wikipedia.org/w/index.php?title=Gas_turbine&oldid=1145271000#Theory_of_operation (visited on 04/30/2023).

[30] Wikipedia contributors. *Subspace identification method*. Ed. by The Free Encyclopedia. Wikipedia. Oct. 18, 2021. URL: https://en.wikipedia.org/w/index.php?title=Subspace_identification_method&oldid=1050483636 (visited on 04/13/2023).

[31] Xiao Wu et al. "Data-driven Predictive Control of Micro Gas Turbine Combined Cooling Heating and Power system". In: *IFAC-PapersOnLine*. IFAC Workshop on Control of Transmission and Distribution Smart Grids CTDSG 2016 49.27 (Jan. 1, 2016), pp. 419–424. DOI: 10.1016/j.ifacol.2016.10.769. URL: https://www.sciencedirect.com/science/article/pii/S2405896316323989 (visited on 03/23/2023).

[32] Mingjuan Zhu et al. "Modeling and model predictive control of micro gas turbine-based combined cooling, heating and power system". In: *2016 Chinese Control and Decision Conference (CCDC)*. 2016 Chinese Control and Decision Conference (CCDC). May 2016, pp. 65–70. DOI: 10.1109/CCDC.2016.7530956.

# Appendix A

# Plots for system identification parameter selection

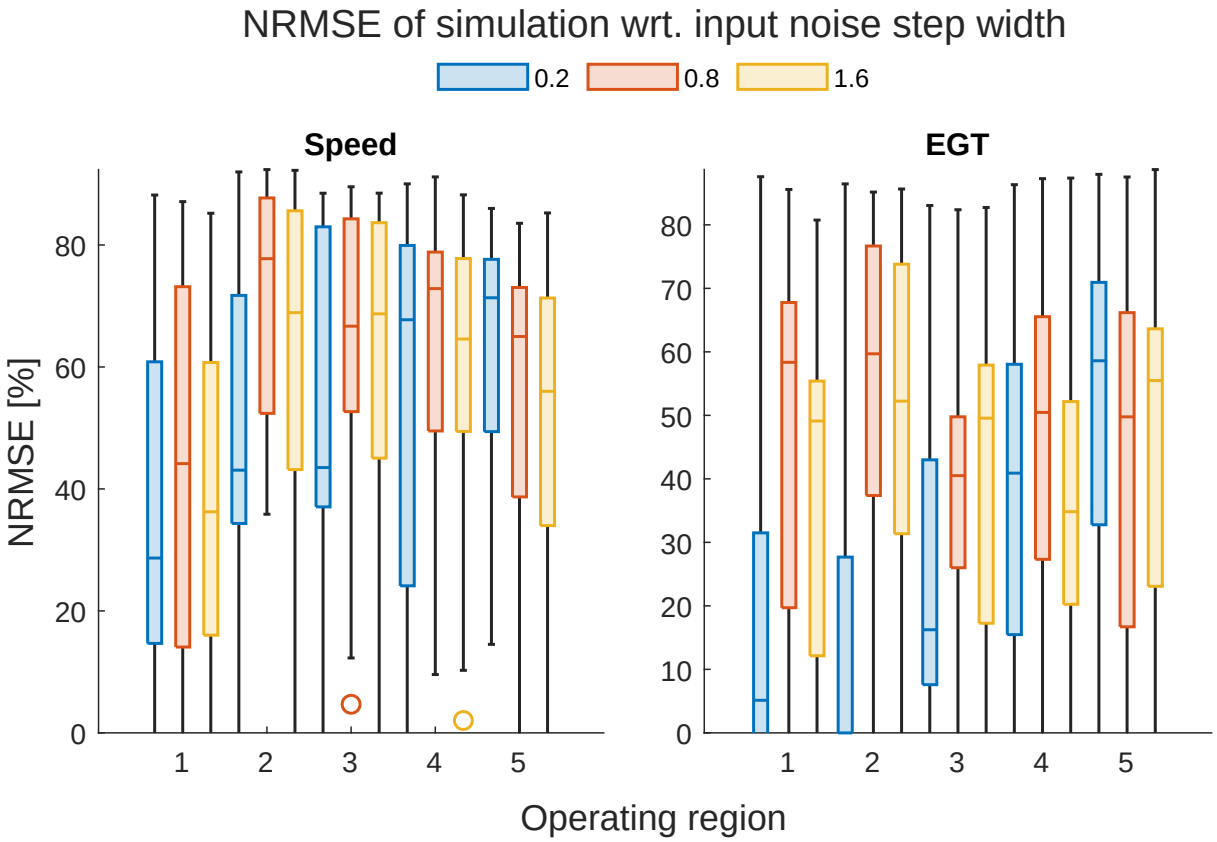



Figure A.1: Comparison of model accuracy with respect to different input noise step widths

Each box shows the statistical distribution of the NRMSE metric as described in Section 3.1.5 for the models identified with all combinations of the parameters model step size 0.1 s and 0.2 s, exponentiation of the speed output to a power of 0.5, 1, 2 and 4, temperature unit in °C or K, exponentiation of the temperature outputs to a power of 1, 2 and 3, normalizing the outputs with the maximum value in the dataset or no normalization and a model order of 7, 12 and 20. All other parameters are as specified in Table 4.1.
The input noise step width 0.8 s has been selected for all operating regions.

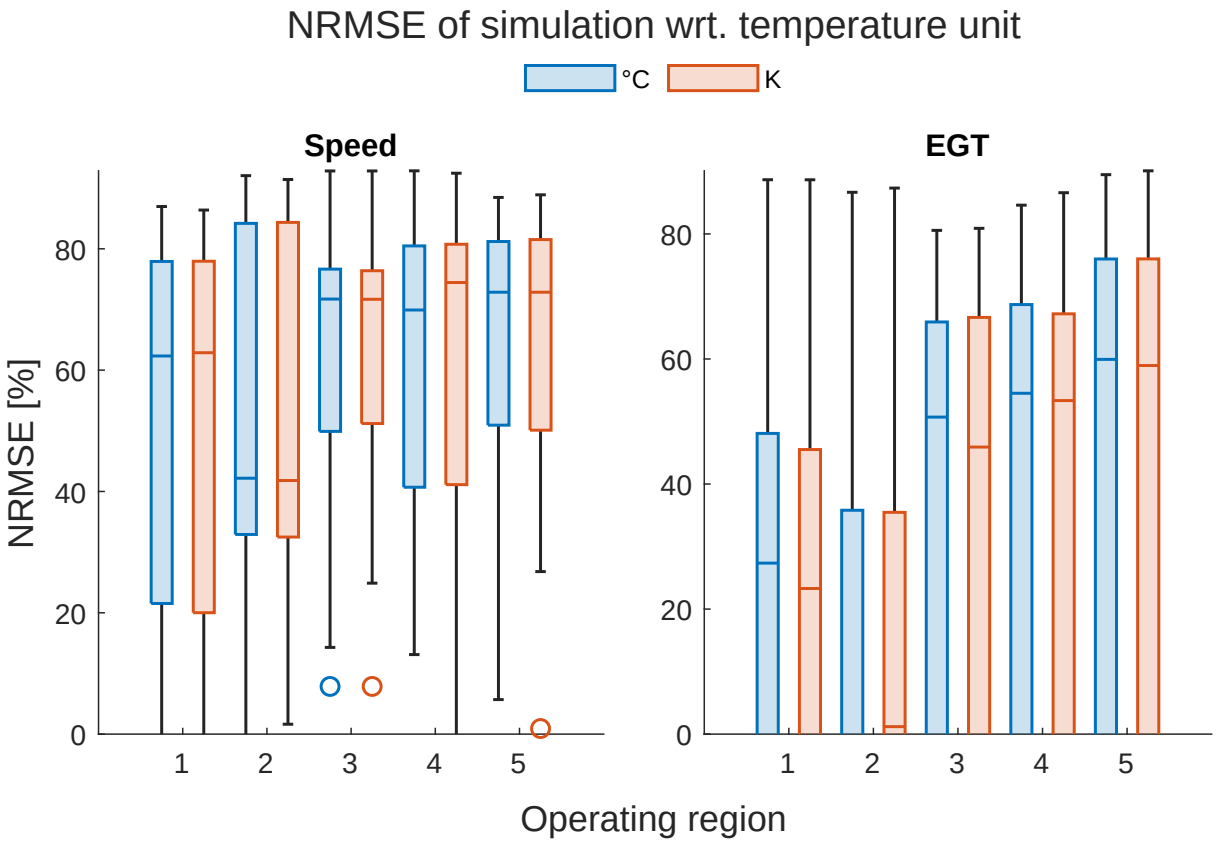



Figure A.2: Comparison of model accuracy with respect to used unit for temperature outputs

Each box shows the statistical distribution of the NRMSE metric as described in Section 3.1.5 for the models identified with all combinations of the parameters exponentiation of the speed output to a power of 0.5, 1, 2, 3 and 4, exponentiation of the temperature outputs to a power of 0.5, 1, 2 and 3, normalizing the outputs with the maximum value in the dataset or no normalization and a model order of 7, 12 and 20. All other parameters are as specified in Table 4.1.
While the choice of temperature does not seem to have a big impact on the model accuracy, °C seems to have a slight advantage over K, thus the temperature outputs were used in this unit.

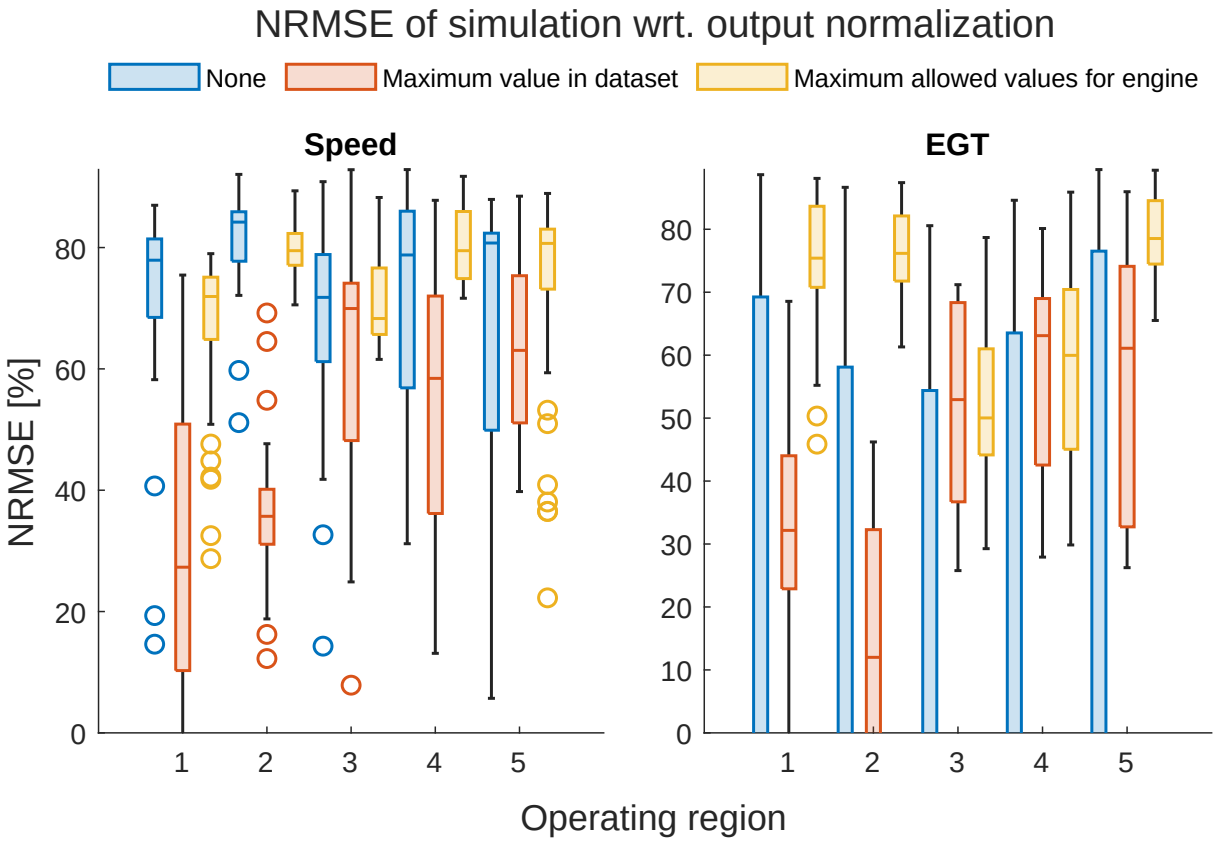



Figure A.3: Comparison of model accuracy with respect to no output normalization, normalization with the maximum value in the dataset and normalization with the maximum allowed values for the engine

Each box shows the statistical distribution of the NRMSE metric as described in Section 3.1.5 for the models identified with all combinations of the parameters exponentiation of the speed output to a power of 0.5, 1, 2, 3 and 4, exponentiation of the temperature outputs to a power of 0.5, 1, 2 and 4 and a model order of 7, 12 and 20. All other parameters are as specified in Table 4.1.
Normalization with the maximum allowed values was selected.

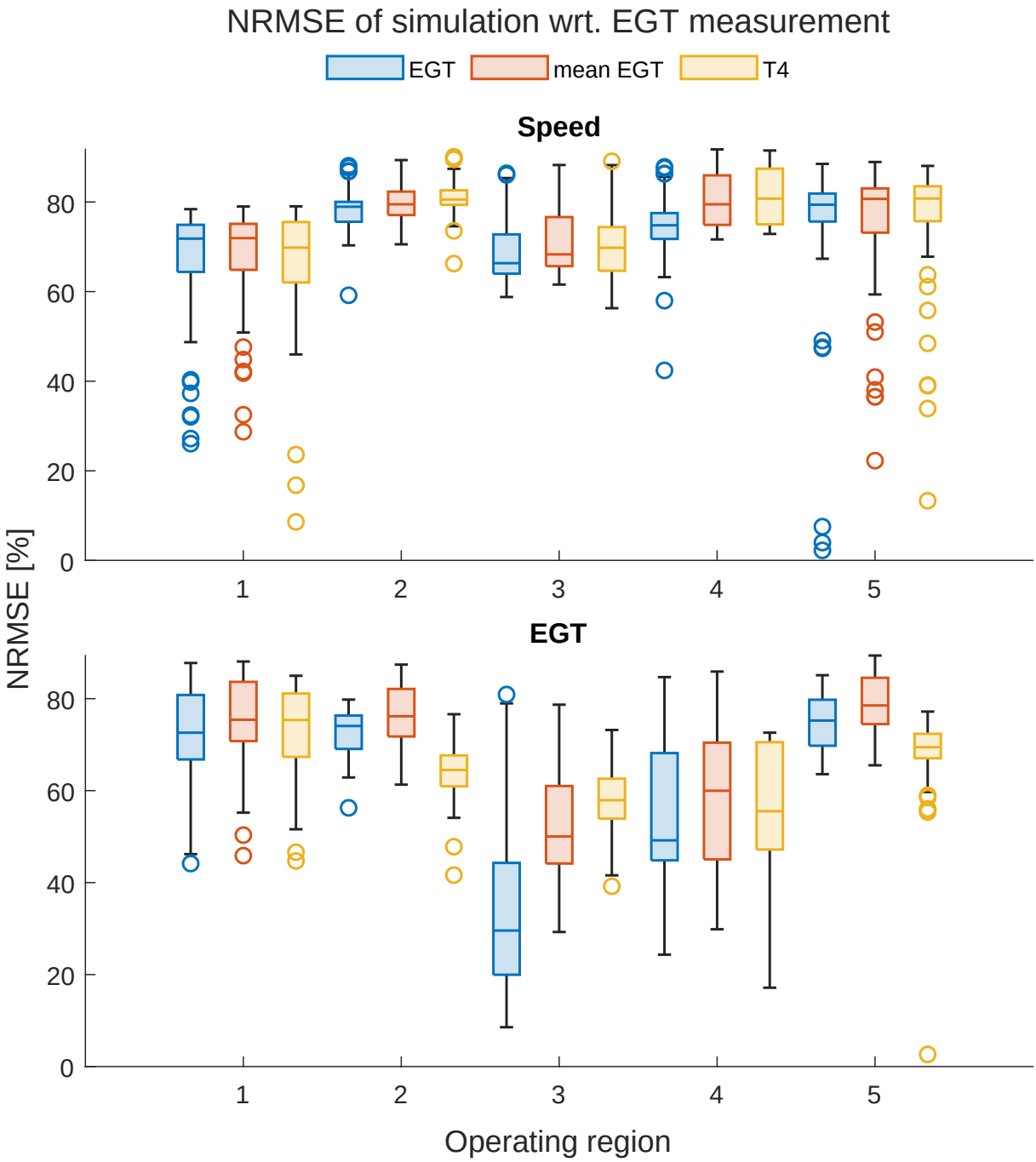



Figure A.4: Comparison of model accuracy with respect to EGT measurement option

Each box shows the statistical distribution of the NRMSE metric as described in Section 3.1.5 for the models identified with all combinations of the parameters exponentiation of the speed output to a power of 0.5, 1, 2, 3 and 4, exponentiation of the temperature outputs to a power of 0.5, 1, 2 and 4 and a model order of 7, 12 and 20. All other parameters are as specified in Table 4.1.
Using the mean of both EGT channels showed to yield the best model accuracy. This is expected, as taking the mean of the two channels reduces measurement noise.

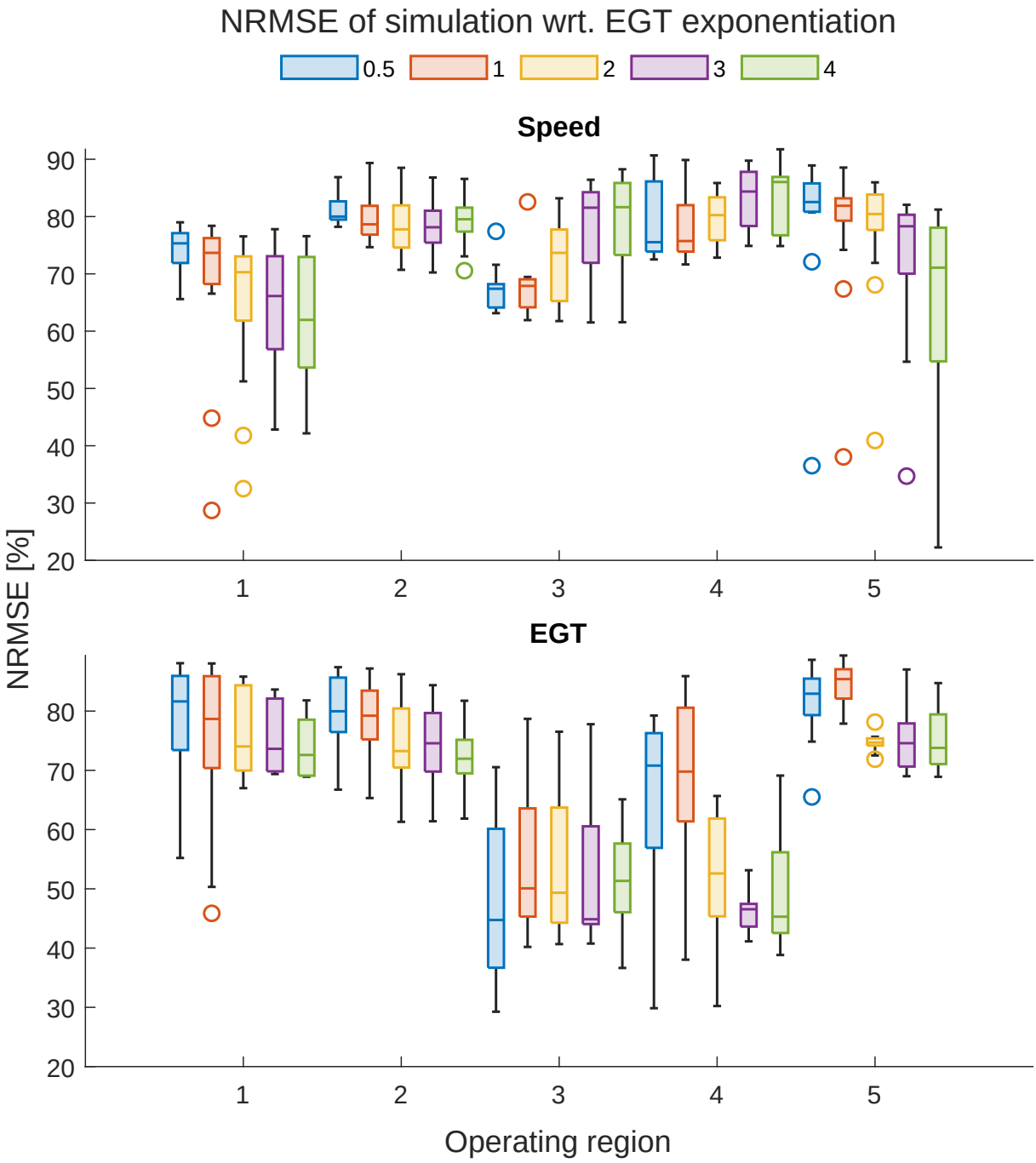



Figure A.5: Comparison of model accuracy with respect to raising the EGT measurement to different powers

Each box shows the statistical distribution of the NRMSE metric as described in Section 3.1.5 for the models identified with all combinations of the parameters exponentiation of the speed output to a power of 0.5, 1, 2, 3 and 4 and a model order of 7, 12 and 20. All other parameters are as specified in Table 4.1. While this is not a priori clear, not exponentiating the exhaust gas temperature (EGT) measurement shows to produce the most accurate models.

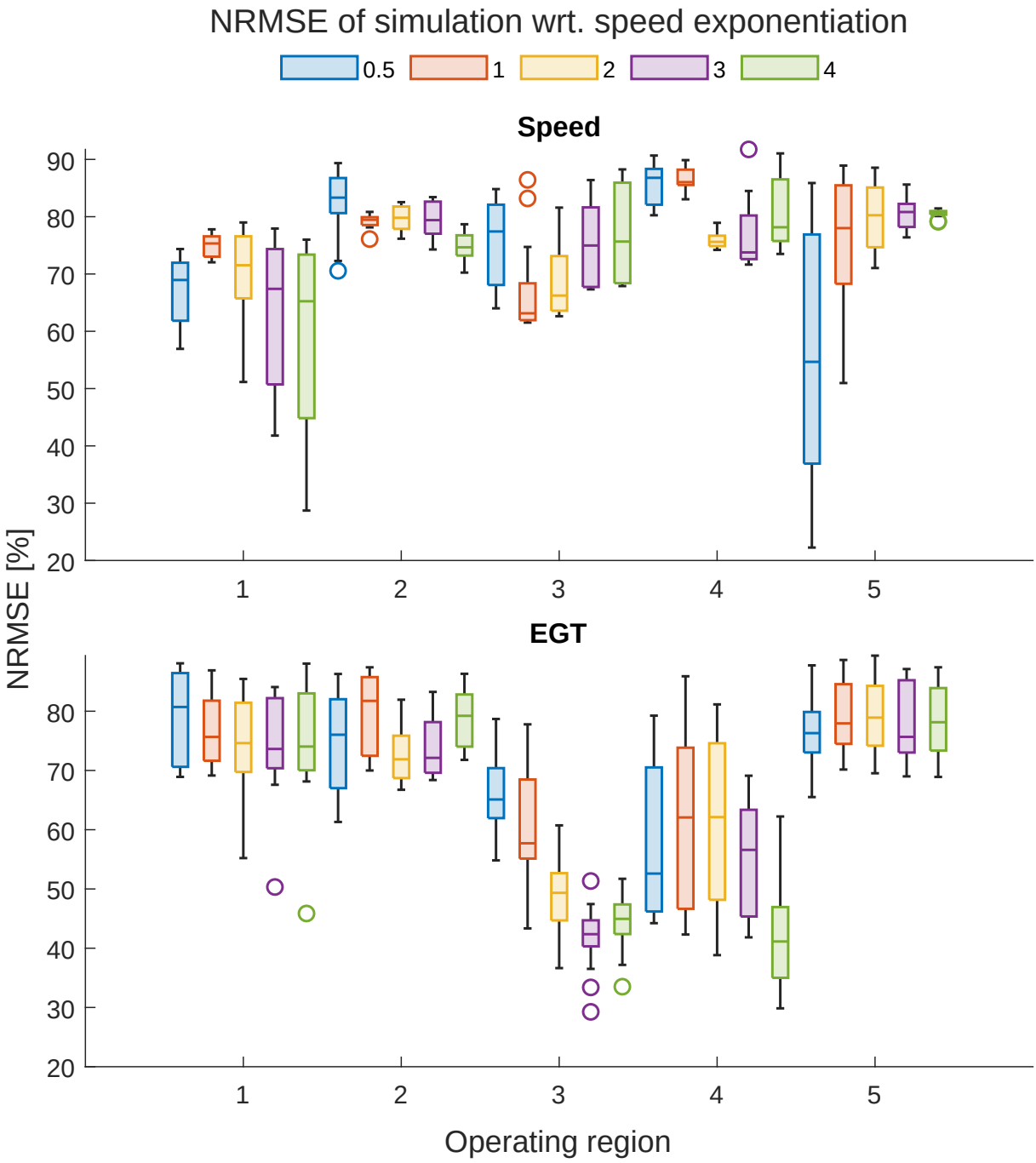



Figure A.6: Comparison of model accuracy with respect to raising the speed measurement to different powers

Each box shows the statistical distribution of the NRMSE metric as described in Section 3.1.5 for the models identified with all combinations of the parameters exponentiation of the temperature outputs to a power of 0.5, 1, 2, 3 and 4 and a model order of 7, 12 and 20. All other parameters are as specified in Table 4.1.
Due to technical limitations in the implementation of the model predictive control (MPC) controller in the Simulink environment, the same power for the speed output had to be selected for all operating regions. No exponentiation (power of 1) showed to have the most consistent results over all operating regions.

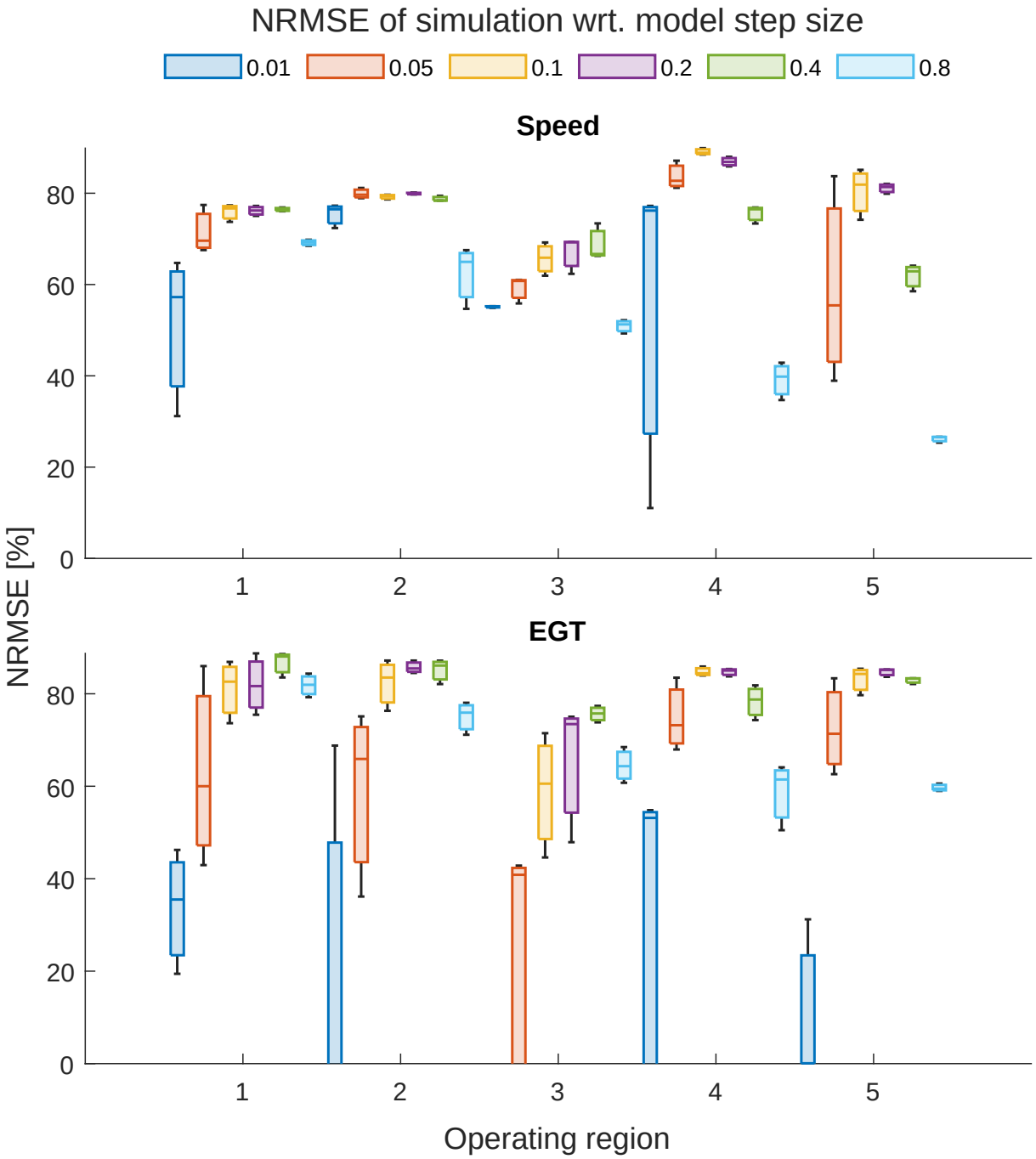



Figure A.7: Comparison of model accuracy with respect to the model step size

Each box shows the statistical distribution of the NRMSE metric as described in Section 3.1.5 for the models identified with a model order of 7, 12 and 20. All other parameters are as specified in Table 4.1. In most of the operating regions a step size of 0.1 s is the optimal step size. Again, due to technical limitations in the implementation, the same step size had to be picked for all operating regions.

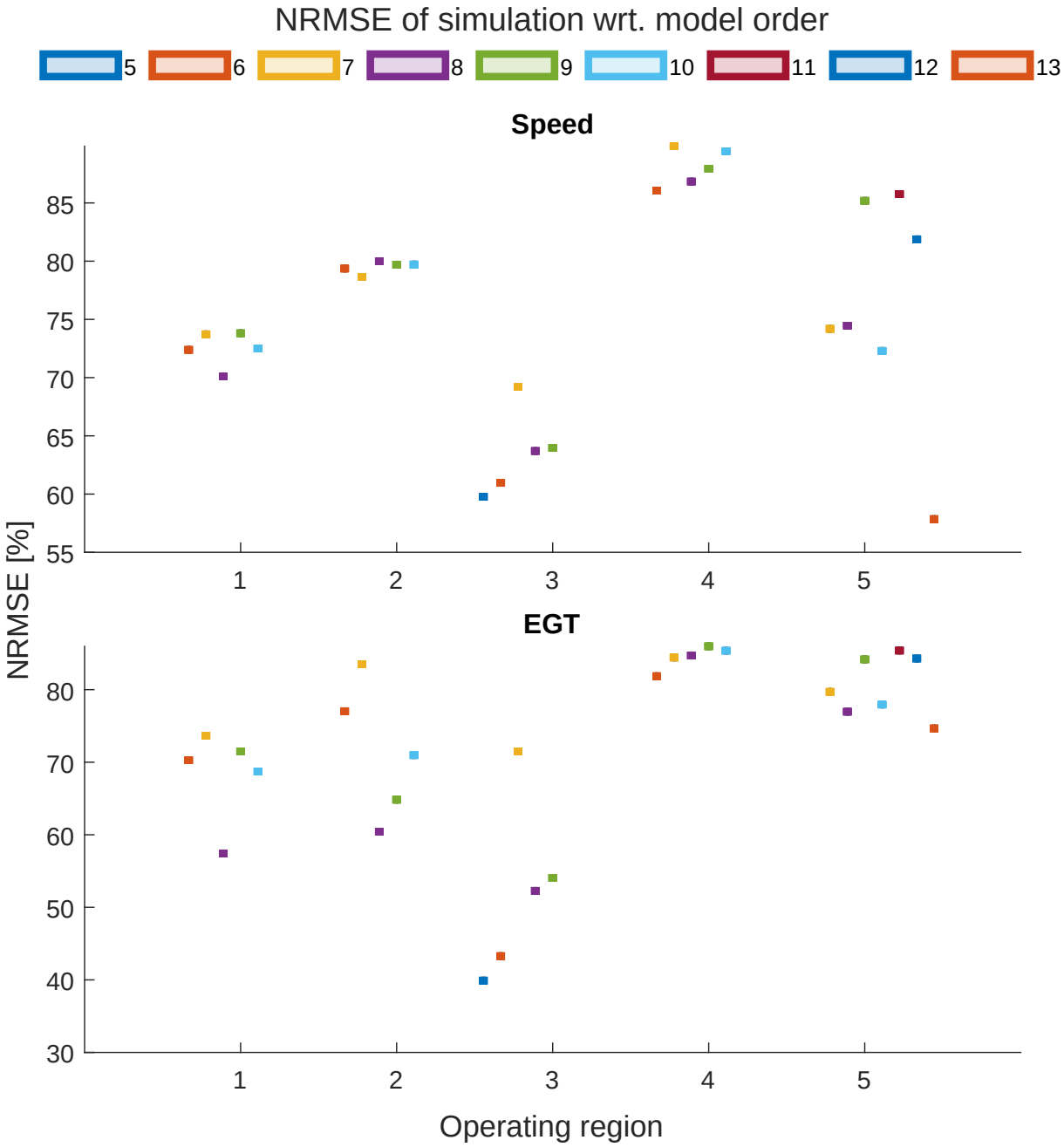



Figure A.8: Comparison of model accuracy with respect to the order of the identified model

Each box shows the NRMSE metric as described in Section 3.1.5 for the model identified with a model order as indicated. All other parameters are as specified in Table 4.1.
Trading off accuracy of the speed output and the EGT output, a model order of 7 has been selected in the operating regions 1 to 4, a model order of 11 for the operating region 5.

# Appendix B

# Plots for MPC parameter selection

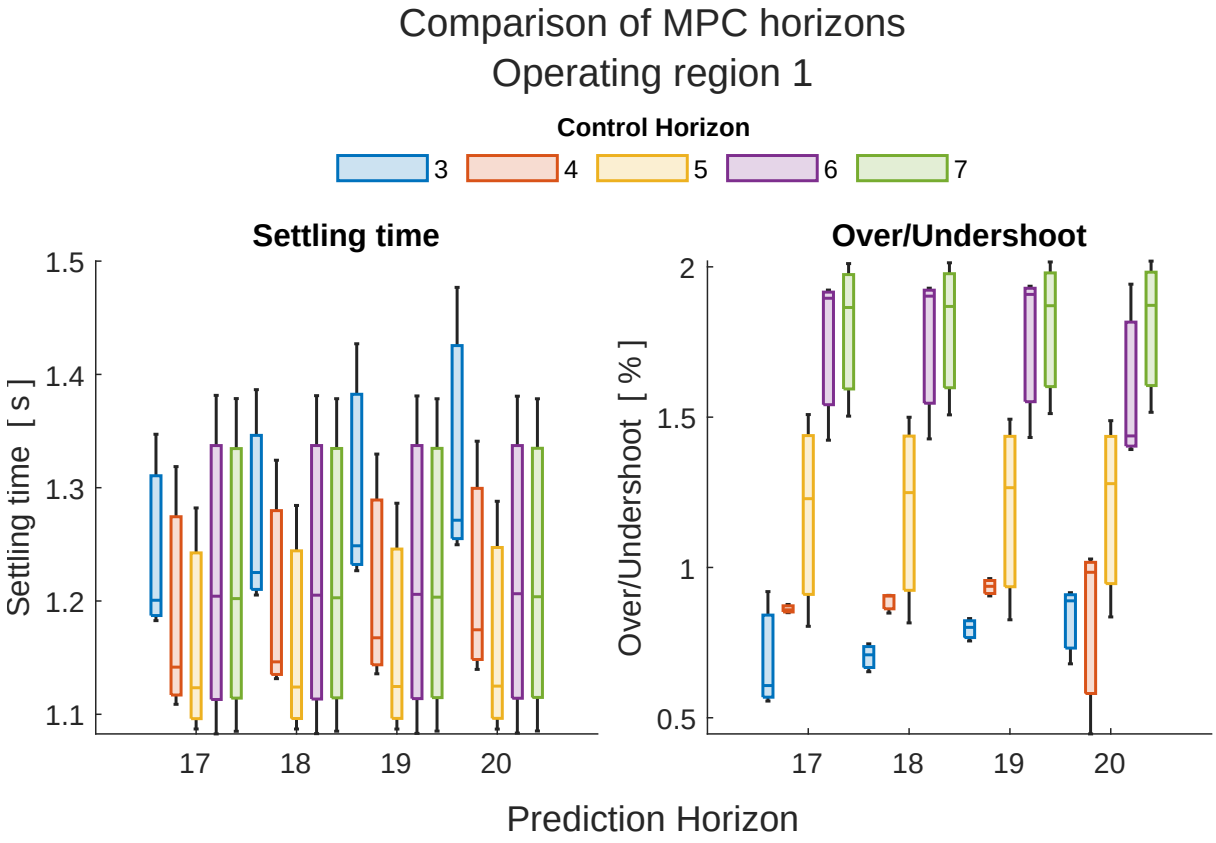



Figure B.1: Comparison of performance metrics for model predictive control (MPC) controllers with different prediction horizons $p$ and control horizons $c$ in operating region 1.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a tuning weight for the control variable penalization $w^u$ of 0, 0.1 and 0.2 and a tuning weight for the control variable move penalization $w^{\Delta u}$ of 0.2, 1 and 2.
The selected parameters are $p = 19$ and $c = 5$ as listed in Table 4.2.

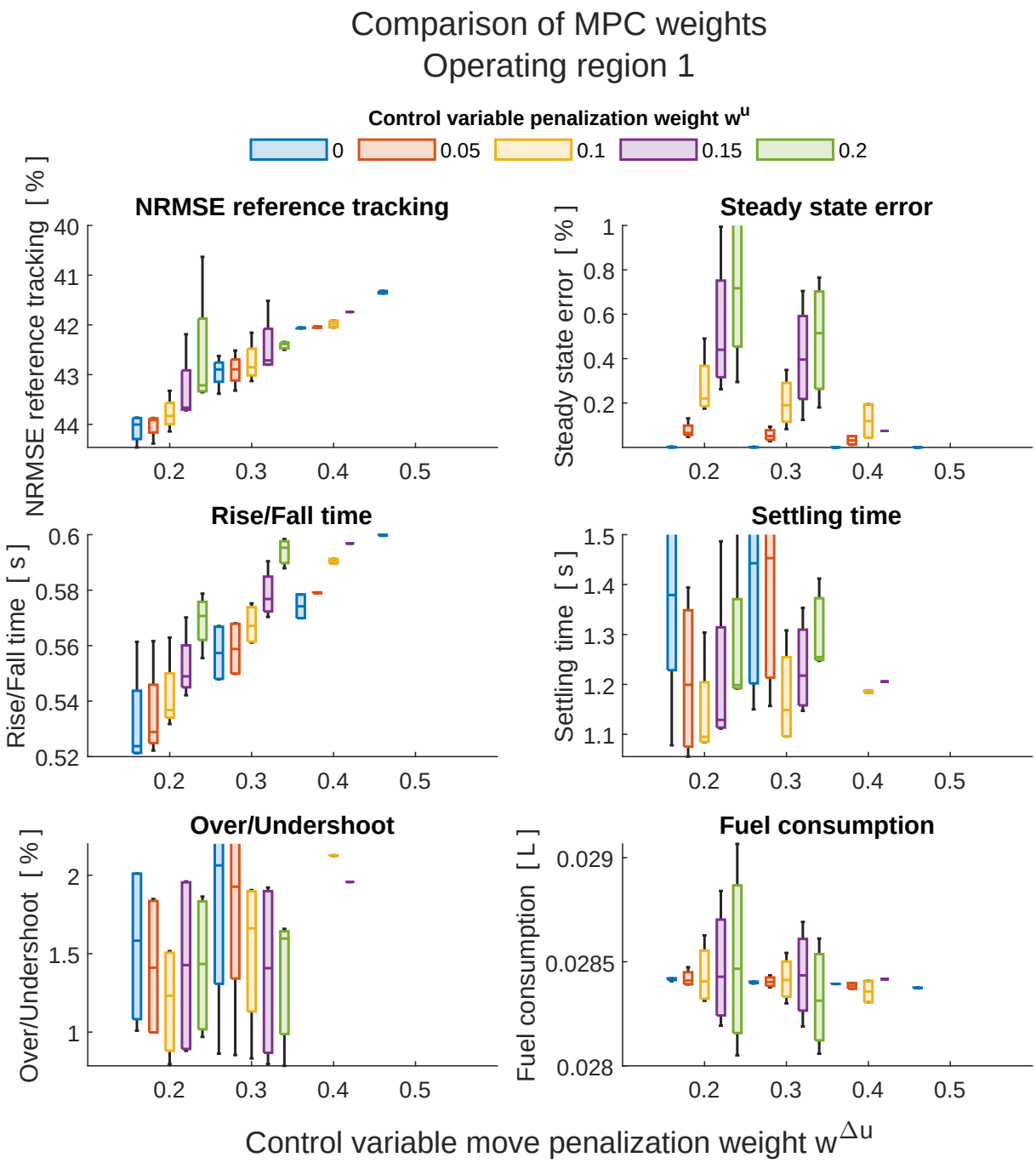



Figure B.2: Comparison of performance metrics for MPC controllers with different tuning weights for the control variable penalization $w^u$ and tuning weights for the control variable move penalization $w^{\Delta u}$ in operating region 1.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a prediction horizon $p$ of 4, 12 and 20 and a control horizon $c$ of 2, 8 and 14.
The selected parameters are $w^u = 0.1$ and $w^{\Delta u} = 0.4$ as listed in Table 4.2.

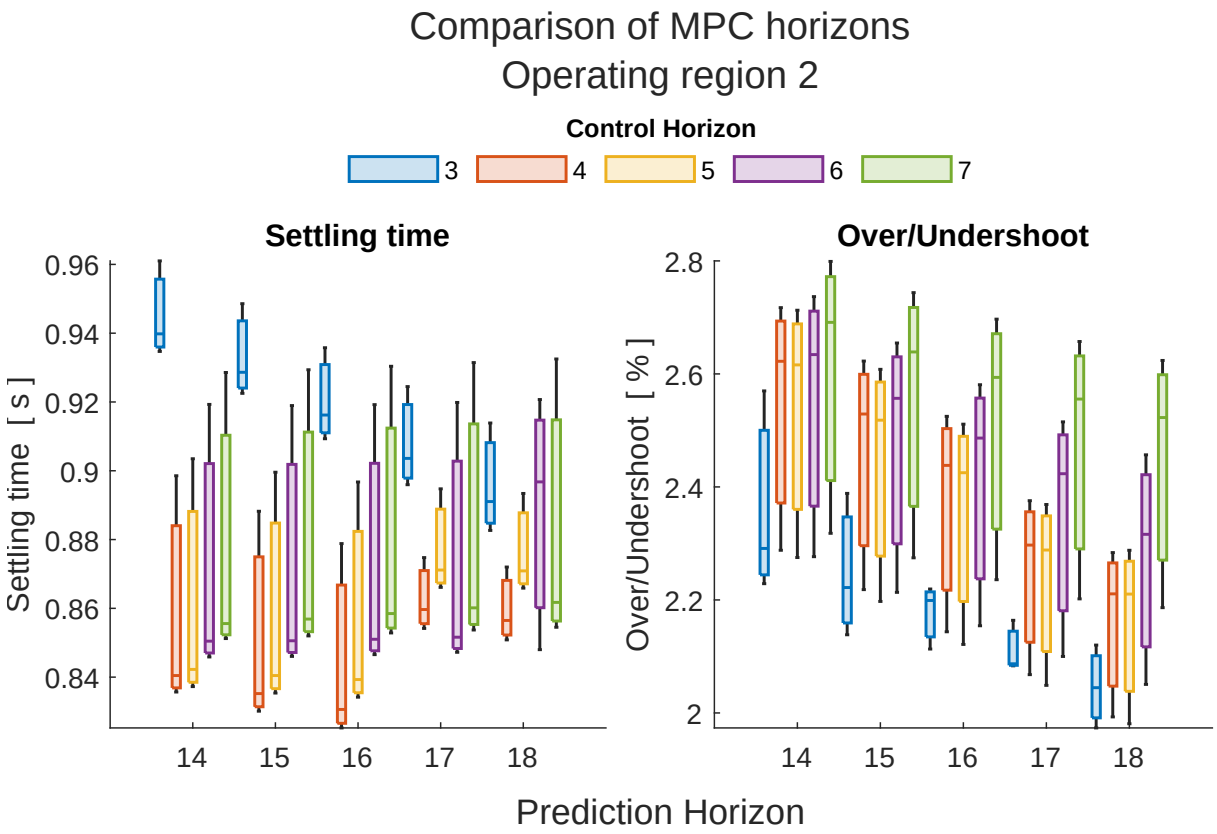



Figure B.3: Comparison of performance metrics for MPC controllers with different prediction horizons $p$ and control horizons $c$ in operating region 2.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a tuning weight for the control variable penalization $w^u$ of 0, 0.1 and 0.2 and a tuning weight for the control variable move penalization $w^{\Delta u}$ of 0.2, 1 and 2.
The selected parameters are $p = 16$ and $c = 5$ as listed in Table 4.2.

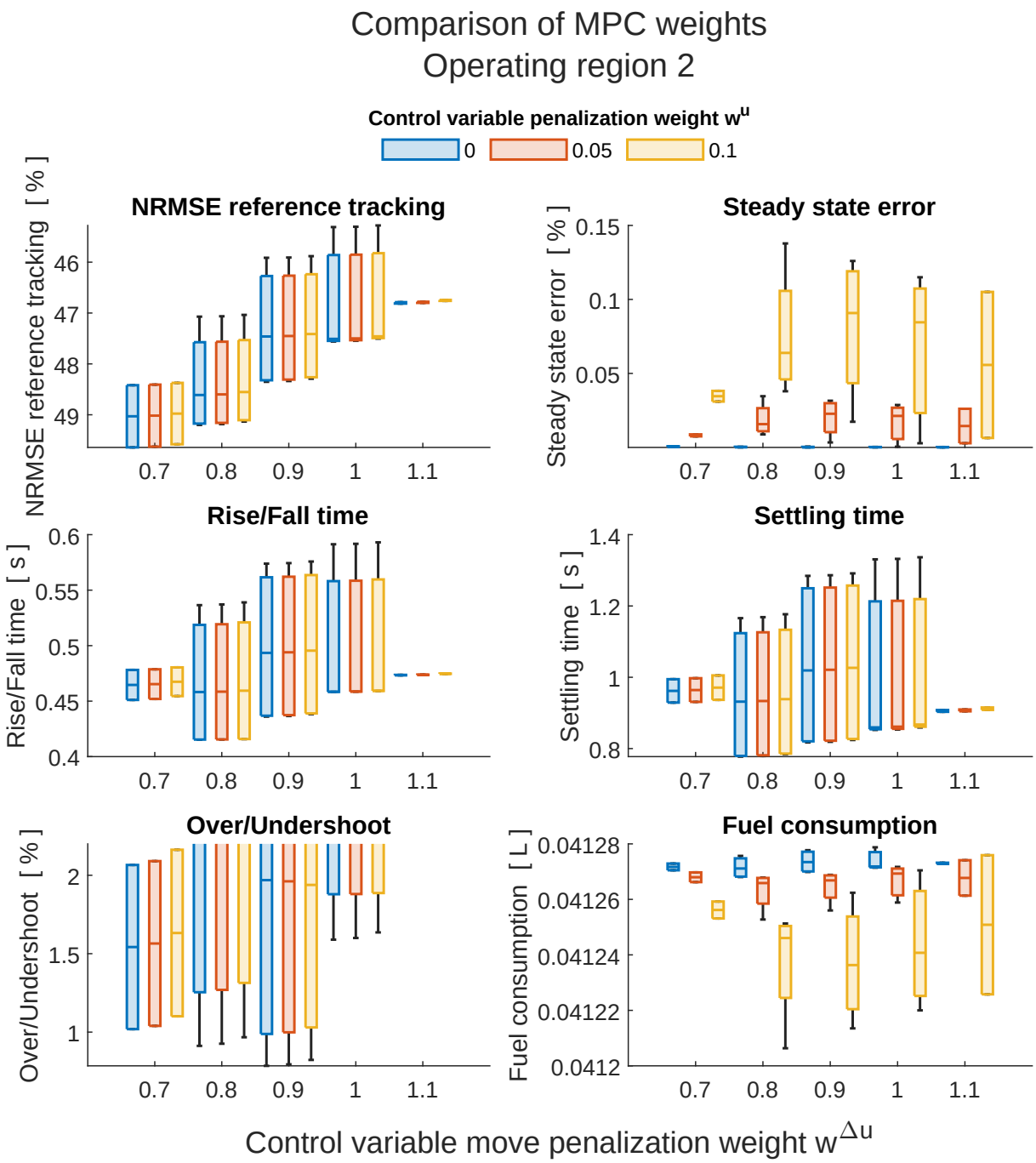



Figure B.4: Comparison of performance metrics for MPC controllers with different tuning weights for the control variable penalization $w^u$ and tuning weights for the control variable move penalization $w^{\Delta u}$ in operating region 2.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a prediction horizon $p$ of 4, 12 and 20 and a control horizon $c$ of 2, 8 and 14.
The selected parameters are $w^u = 0$ and $w^{\Delta u} = 0.9$ as listed in Table 4.2.

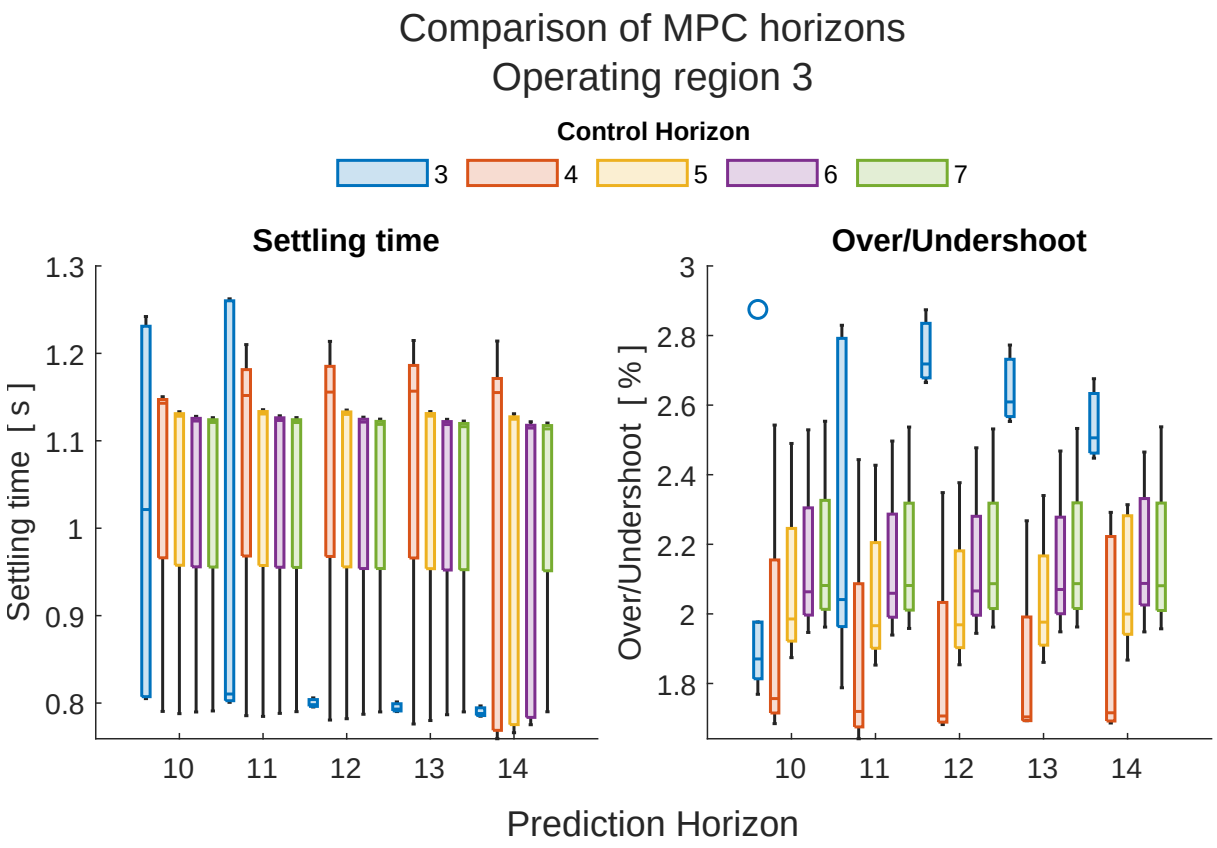



Figure B.5: Comparison of performance metrics for MPC controllers with different prediction horizons $p$ and control horizons $c$ in operating region 3.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a tuning weight for the control variable penalization $w^u$ of 0, 0.1 and 0.2 and a tuning weight for the control variable move penalization $w^{\Delta u}$ of 0.2, 1 and 2.
The selected parameters are $p = 12$ and $c = 5$ as listed in Table 4.2.

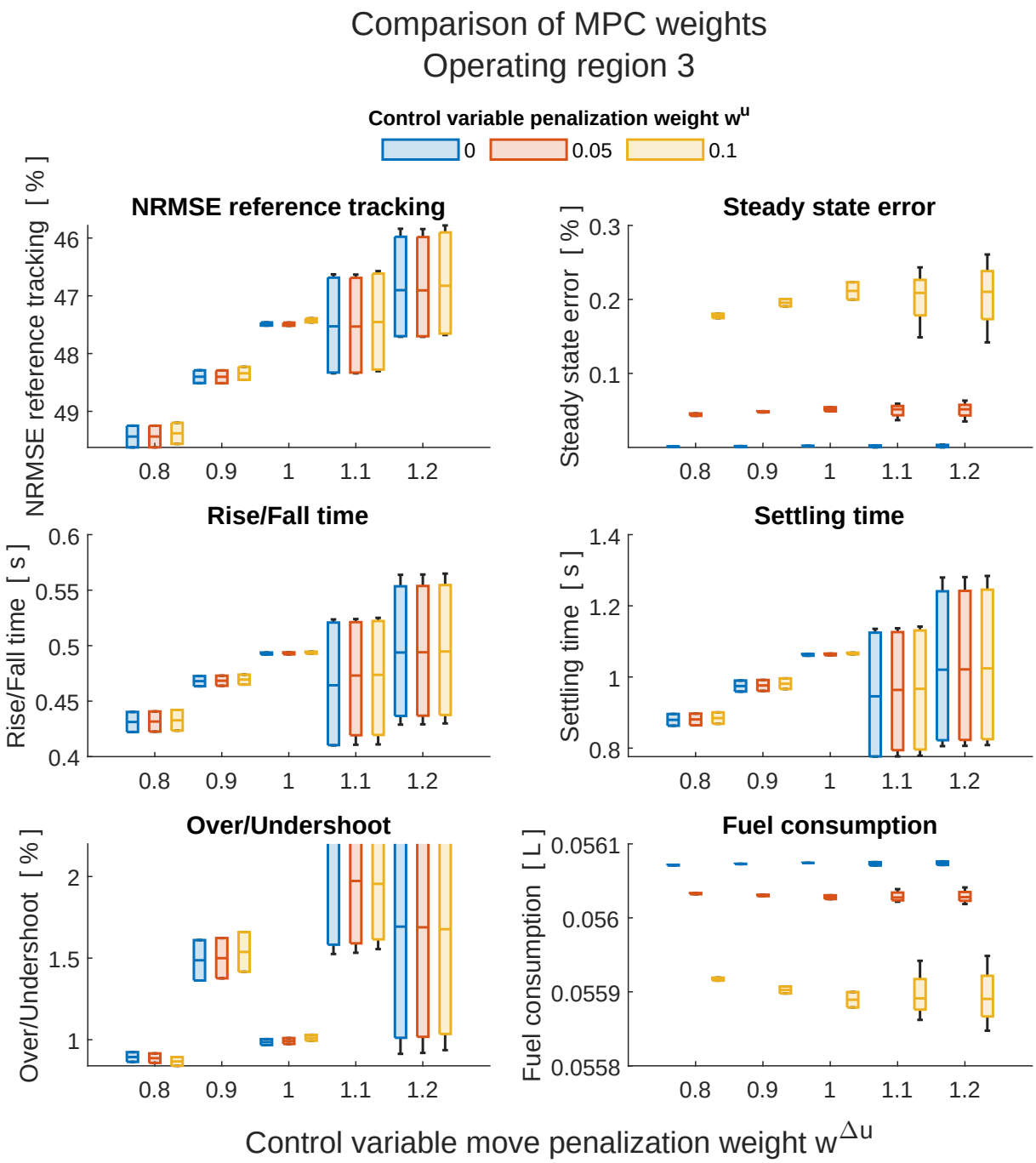



Figure B.6: Comparison of performance metrics for MPC controllers with different tuning weights for the control variable penalization $w^u$ and tuning weights for the control variable move penalization $w^{\Delta u}$ in operating region 3.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a prediction horizon $p$ of 4, 12 and 20 and a control horizon $c$ of 2, 8 and 14.
The selected parameters are $w^u = 0$ and $w^{\Delta u} = 1$ as listed in Table 4.2.

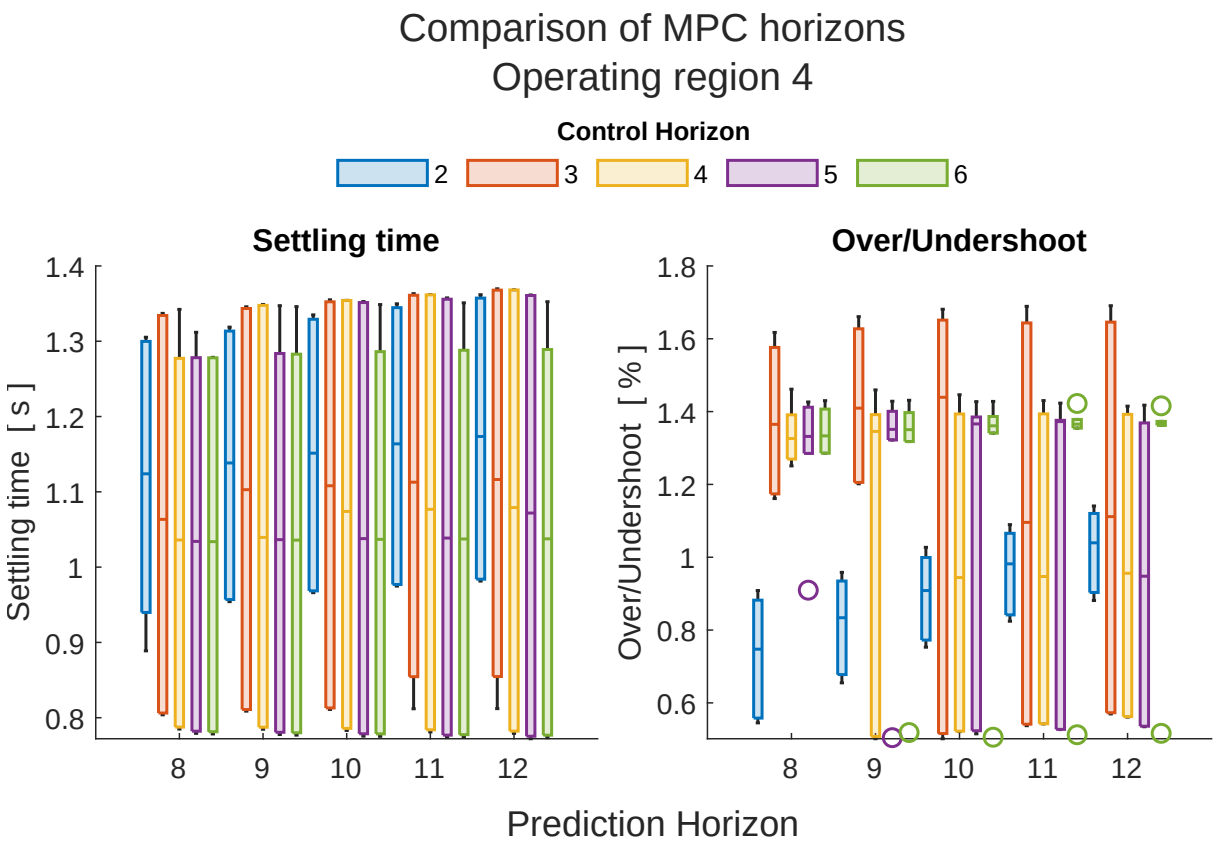



Figure B.7: Comparison of performance metrics for MPC controllers with different prediction horizons $p$ and control horizons $c$ in operating region 4.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a tuning weight for the control variable penalization $w^u$ of 0, 0.1 and 0.2 and a tuning weight for the control variable move penalization $w^{\Delta u}$ of 0.2, 1 and 2.
The selected parameters are $p = 10$ and $c = 4$ as listed in Table 4.2.

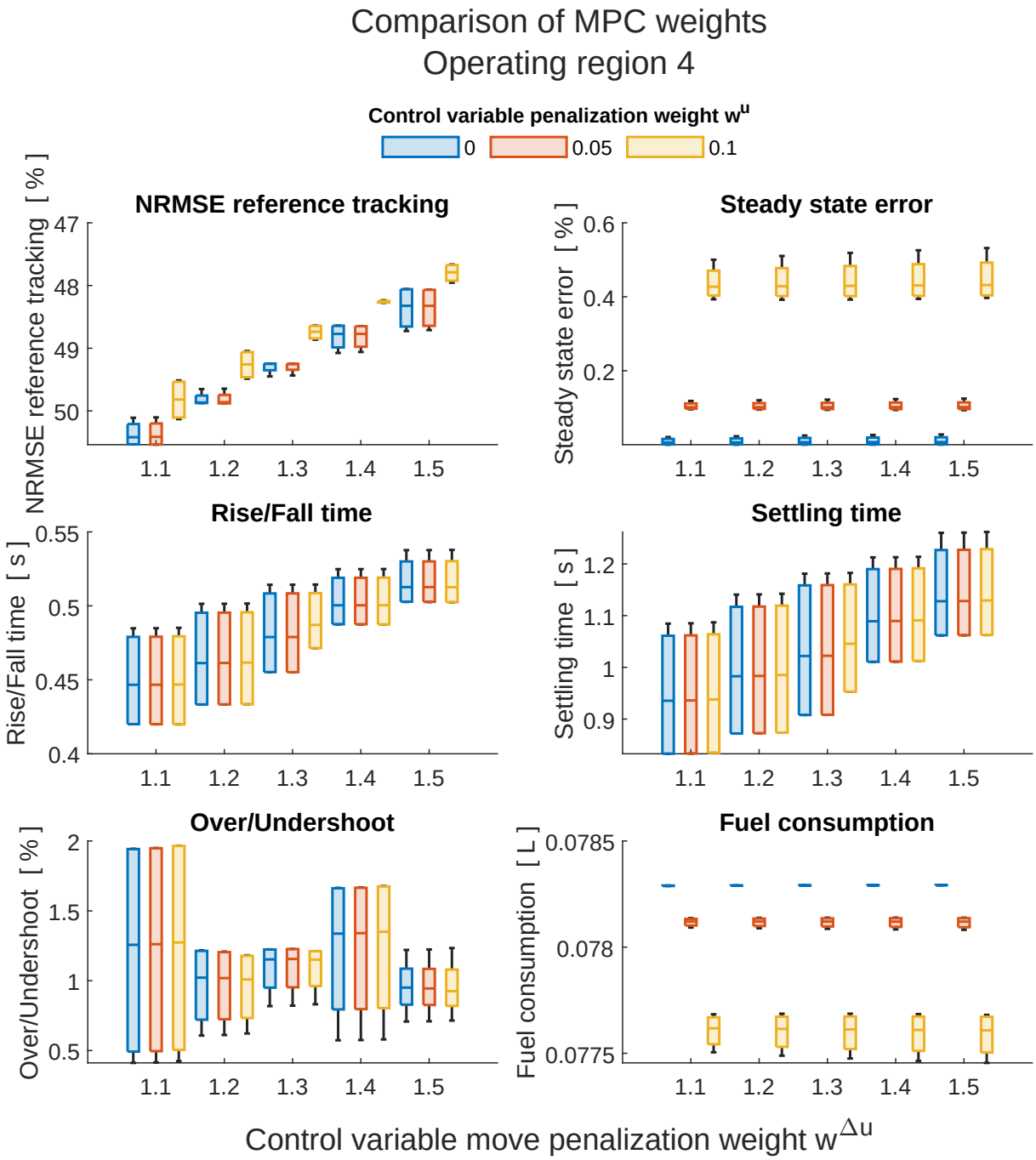



Figure B.8: Comparison of performance metrics for MPC controllers with different tuning weights for the control variable penalization $w^u$ and tuning weights for the control variable move penalization $w^{\Delta u}$ in operating region 4.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a prediction horizon $p$ of 4, 12 and 20 and a control horizon $c$ of 2, 8 and 14.
The selected parameters are $w^u = 0$ and $w^{\Delta u} = 1.3$ as listed in Table 4.2.

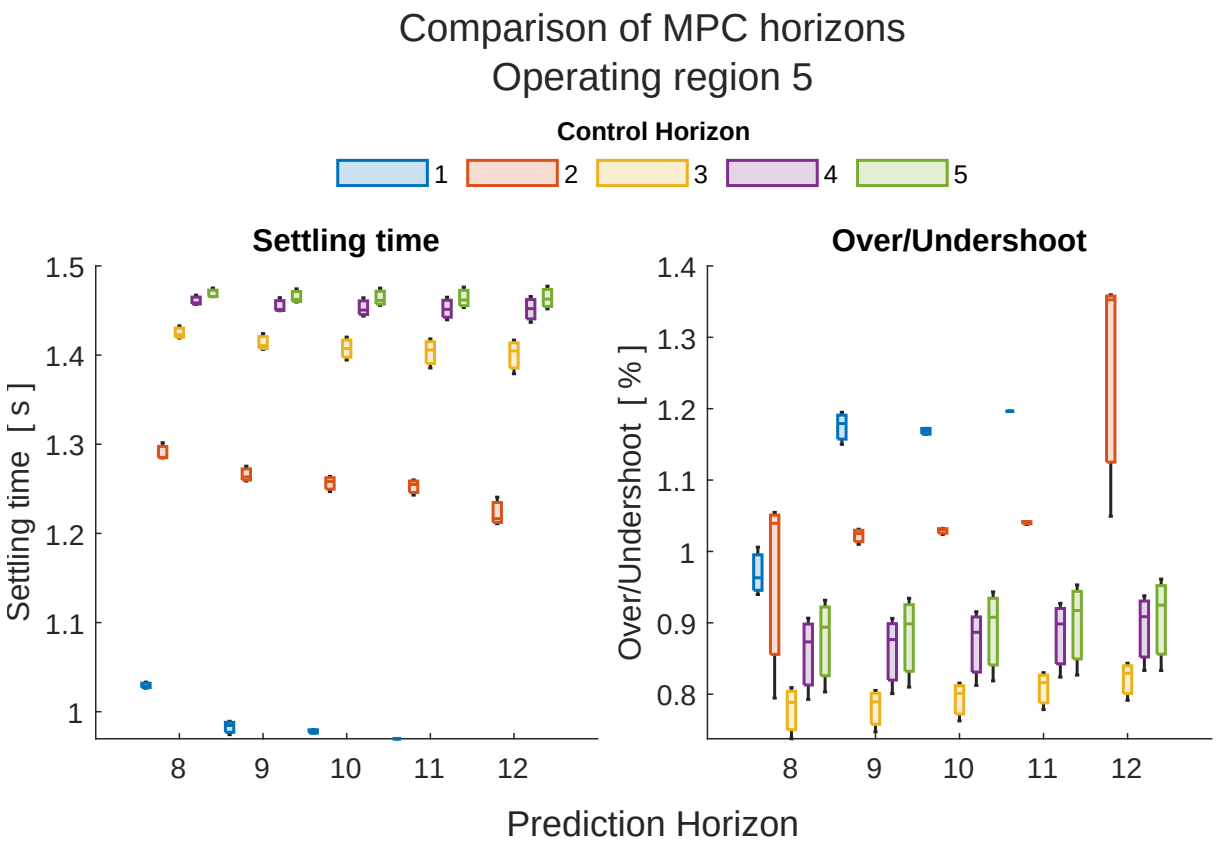



Figure B.9: Comparison of performance metrics for MPC controllers with different prediction horizons $p$ and control horizons $c$ in operating region 5.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a tuning weight for the control variable penalization $w^u$ of 0, 0.1 and 0.2 and a tuning weight for the control variable move penalization $w^{\Delta u}$ of 0.2, 1 and 2.
The selected parameters are $p = 10$ and $c = 3$ as listed in Table 4.2.

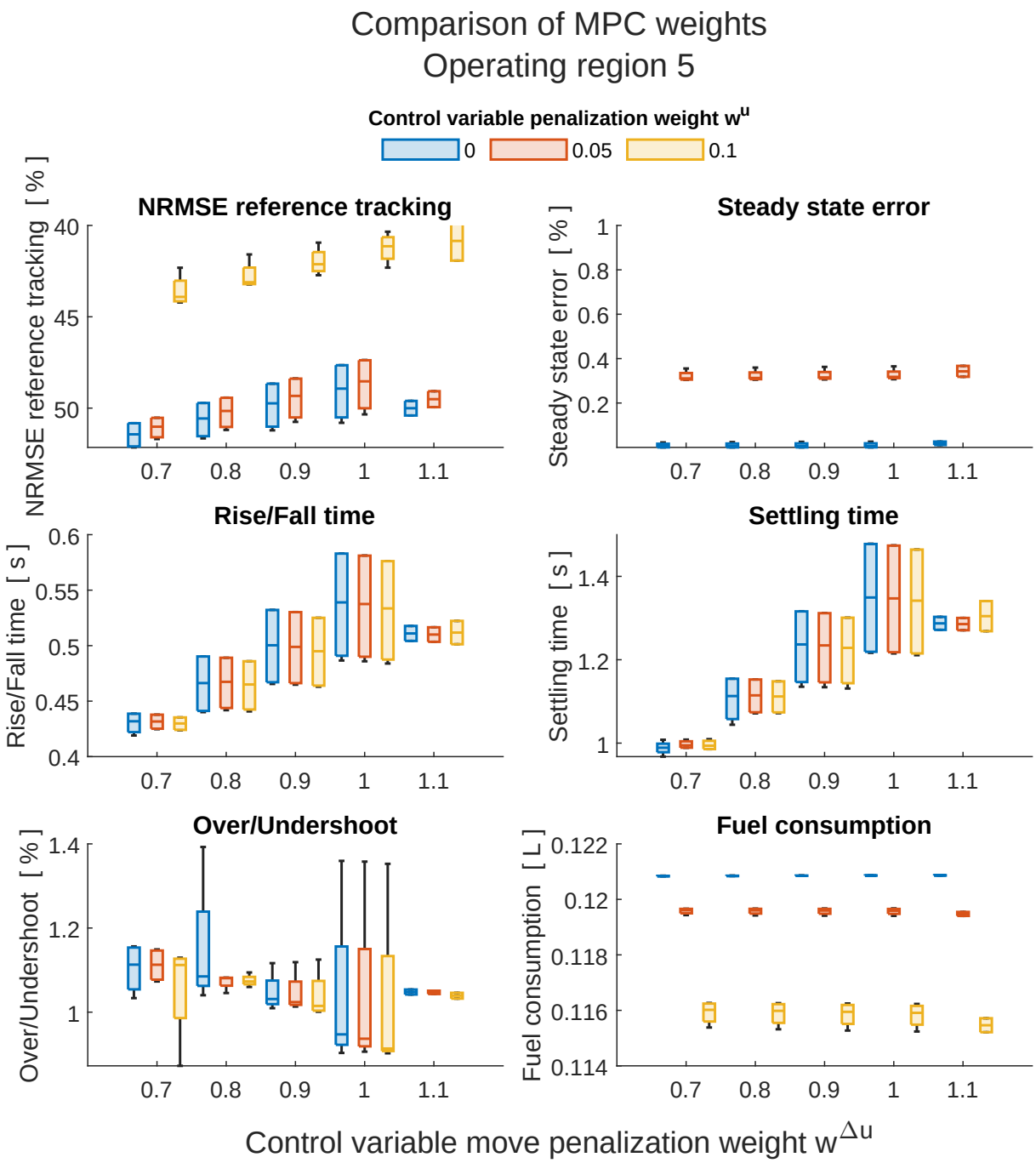



Figure B.10: Comparison of performance metrics for MPC controllers with different tuning weights for the control variable penalization $w^u$ and tuning weights for the control variable move penalization $w^{\Delta u}$ in operating region 5.

In all plots, closer to the x-axis is better. The metrics are explained in Section 3.2.2.
Each box displays the minimum, lower quartile, median, upper quartile, maximum (displaying outliers separately) statistical values for the metrics of a set of 9 MPC controllers with each combination of a prediction horizon $p$ of 4, 12 and 20 and a control horizon $c$ of 2, 8 and 14.
The selected parameters are $w^u = 0$ and $w^{\Delta u} = 0.9$ as listed in Table 4.2.

# Appendix C

# Used software versions

Throughout the project, the following software versions were used:

| Software | Release | Version |
|---|---|---|
| MATLAB | R2022a | 9.12.0.2170939 |
| dSPACE ControlDesk | 2022-A | 7.6.0.0 |
| dSPACE RCP and HIL Software | 2022-A | 22.1.0.0 |

Output of the Matlab `ver` command:

```
-----------------------------------------------------------------------------------------------------
MATLAB Version: 9.12.0.2170939 (R2022a) Update 6
MATLAB License Number: xxxxxx
Operating System: Microsoft Windows 11 Enterprise Version 10.0 (Build 22621)
Java Version: Java 1.8.0_202-b08 with Oracle Corporation Java HotSpot(TM) 64-Bit Server VM mixed mode
-----------------------------------------------------------------------------------------------------
MATLAB                                                Version 9.12        (R2022a)
Simulink                                              Version 10.5        (R2022a)
5G Toolbox                                            Version 2.4         (R2022a)
AUTOSAR Blockset                                      Version 2.6         (R2022a)
Aerospace Blockset                                    Version 5.2         (R2022a)
Aerospace Toolbox                                     Version 4.2         (R2022a)
Antenna Toolbox                                       Version 5.2         (R2022a)
Audio Toolbox                                         Version 3.2         (R2022a)
Automated Driving Toolbox                             Version 3.5         (R2022a)
Bioinformatics Toolbox                                Version 4.16        (R2022a)
Bluetooth Toolbox                                     Version 1.0         (R2022a)
Communications Toolbox                                Version 7.7         (R2022a)
Computer Vision Toolbox                               Version 10.2        (R2022a)
Control System Toolbox                                Version 10.11.1     (R2022a)
Curve Fitting Toolbox                                 Version 3.7         (R2022a)
DDS Blockset                                          Version 1.2         (R2022a)
DSP HDL Toolbox                                       Version 1.0         (R2022a)
DSP System Toolbox                                    Version 9.14        (R2022a)
Data Acquisition Toolbox                              Version 4.5         (R2022a)
Database Toolbox                                      Version 10.3        (R2022a)
Datafeed Toolbox                                      Version 6.2         (R2022a)
Deep Learning HDL Toolbox                             Version 1.3         (R2022a)
Deep Learning Toolbox                                 Version 14.4        (R2022a)
Econometrics Toolbox                                  Version 6.0         (R2022a)
Embedded Coder                                        Version 7.8         (R2022a)
Filter Design HDL Coder                               Version 3.1.11      (R2022a)
Financial Instruments Toolbox                         Version 3.4         (R2022a)
Financial Toolbox                                     Version 6.3         (R2022a)
Fixed-Point Designer                                  Version 7.4         (R2022a)
Fuzzy Logic Toolbox                                   Version 2.9         (R2022a)
GPU Coder                                             Version 2.3         (R2022a)
Global Optimization Toolbox                           Version 4.7         (R2022a)
HDL Coder                                             Version 3.20        (R2022a)
HDL Verifier                                          Version 6.5         (R2022a)
Image Acquisition Toolbox                             Version 6.6         (R2022a)
Image Processing Toolbox                              Version 11.5        (R2022a)
Industrial Communication Toolbox                      Version 6.0         (R2022a)
Instrument Control Toolbox                            Version 4.6         (R2022a)
LTE Toolbox                                           Version 3.7         (R2022a)
Lidar Toolbox                                         Version 2.1         (R2022a)
MATLAB Coder                                          Version 5.4         (R2022a)
MATLAB Compiler                                       Version 8.4         (R2022a)
MATLAB Compiler SDK                                   Version 7.0         (R2022a)
MATLAB Report Generator                               Version 5.12        (R2022a)
Mapping Toolbox                                       Version 5.3         (R2022a)
Mixed-Signal Blockset                                 Version 2.2         (R2022a)
Model Predictive Control Toolbox                      Version 7.3         (R2022a)
Model-Based Calibration Toolbox                       Version 5.12        (R2022a)
```

```
Motor Control Blockset                                  Version 1.4         (R2022a)
Navigation Toolbox                                      Version 2.2         (R2022a)
Optimization Toolbox                                    Version 9.3         (R2022a)
Parallel Computing Toolbox                              Version 7.6         (R2022a)
Partial Differential Equation Toolbox                   Version 3.8         (R2022a)
Phased Array System Toolbox                             Version 4.7         (R2022a)
Powertrain Blockset                                     Version 1.11        (R2022a)
Predictive Maintenance Toolbox                          Version 2.5         (R2022a)
RF Blockset                                             Version 8.3         (R2022a)
RF PCB Toolbox                                          Version 1.1         (R2022a)
RF Toolbox                                              Version 4.3         (R2022a)
ROS Toolbox                                             Version 1.5         (R2022a)
Radar Toolbox                                           Version 1.2         (R2022a)
Reinforcement Learning Toolbox                          Version 2.2         (R2022a)
Requirements Toolbox                                    Version 2.0         (R2022a)
Risk Management Toolbox                                 Version 2.0         (R2022a)
Robotics System Toolbox                                 Version 4.0         (R2022a)
Robust Control Toolbox                                  Version 6.11.1      (R2022a)
Satellite Communications Toolbox                        Version 1.2         (R2022a)
Sensor Fusion and Tracking Toolbox                      Version 2.3         (R2022a)
SerDes Toolbox                                          Version 2.3         (R2022a)
Signal Integrity Toolbox                                Version 1.1         (R2022a)
Signal Processing Toolbox                               Version 9.0         (R2022a)
SimBiology                                              Version 6.3         (R2022a)
SimEvents                                               Version 5.11        (R2022a)
Simscape                                                Version 5.3         (R2022a)
Simscape Driveline                                      Version 3.5         (R2022a)
Simscape Electrical                                     Version 7.7         (R2022a)
Simscape Fluids                                         Version 3.4         (R2022a)
Simscape Multibody                                      Version 7.5         (R2022a)
Simulink 3D Animation                                   Version 9.4         (R2022a)
Simulink Check                                          Version 6.0         (R2022a)
Simulink Code Inspector                                 Version 4.1         (R2022a)
Simulink Coder                                          Version 9.7         (R2022a)
Simulink Compiler                                       Version 1.4         (R2022a)
Simulink Control Design                                 Version 6.1         (R2022a)
Simulink Coverage                                       Version 5.4         (R2022a)
Simulink Design Optimization                            Version 3.11        (R2022a)
Simulink Design Verifier                                Version 4.7         (R2022a)
Simulink Desktop Real-Time                              Version 5.14        (R2022a)
Simulink PLC Coder                                      Version 3.6         (R2022a)
Simulink Real-Time                                      Version 8.0         (R2022a)
Simulink Report Generator                               Version 5.12        (R2022a)
Simulink Test                                           Version 3.6         (R2022a)
SoC Blockset                                            Version 1.6         (R2022a)
Spreadsheet Link                                        Version 3.4.7       (R2022a)
Stateflow                                               Version 10.6        (R2022a)
Statistics and Machine Learning Toolbox                 Version 12.3        (R2022a)
Symbolic Math Toolbox                                   Version 9.1         (R2022a)
System Composer                                         Version 2.2         (R2022a)
System Identification Toolbox                           Version 9.16        (R2022a)
Text Analytics Toolbox                                  Version 1.8.1       (R2022a)
UAV Toolbox                                             Version 1.3         (R2022a)
Vehicle Dynamics Blockset                               Version 1.8         (R2022a)
Vehicle Network Toolbox                                 Version 5.2         (R2022a)
Vision HDL Toolbox                                      Version 2.5         (R2022a)
WLAN Toolbox                                            Version 3.4         (R2022a)
Wavelet Toolbox                                         Version 6.1         (R2022a)
Wireless HDL Toolbox                                    Version 2.4         (R2022a)
Wireless Testbench                                      Version 1.0         (R2022a)
dSPACE MATLAB Integration                               Version 22.1        (Release 2022-B)
dSPACE MotionDesk Blockset                              Version 2.6.5
dSPACE RTI Bypass Blockset                              Version 3.18
dSPACE RTI CAN Blockset                                 Version 3.4.14
dSPACE RTI CAN MultiMessage Blockset                    Version 5.8p1
dSPACE RTI EMC Blockset                                 Version 1.4.5
dSPACE RTI ETHERNET Blockset                            Version 1.2.7
dSPACE RTI ETHERNETUDP Blockset                         Version 1.4.7
dSPACE RTI FPGA Programming Blockset                    Version 3.13
dSPACE RTI FlexRay Configuration Blockset               Version 4.9
dSPACE RTI Gigalink Blockset                            Version 2.4.4
dSPACE RTI LIN MultiMessage Blockset                    Version 3.8
dSPACE RTI MultiMessage Shared Utilities                Version 2.8
dSPACE RTI RPCU Blockset                                Version 2.2.7
dSPACE RTI Synchronized Time Base Manager Blockset      Version 1.4.4p1
dSPACE RTI USBFLIGHTREC Blockset                        Version 1.2.6
dSPACE RTI WATCHDOG Blockset                            Version 2.1.5
dSPACE RTI XCP on Ethernet Blockset                     Version 1.2.14
dSPACE RTI for Multiprocessor Systems RTI-MP            Version 7.18
dSPACE Real-Time Interface (RTI1202)                    Version 7.18
dSPACE Real-Time Interface Shared Utilities             Version 7.18
dSPACE Run-Time Target Shared Utilities                 Version 22.1
```

# Appendix D

# Declaration of Originality

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

## Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

---

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

**Title of work** (in block letters):

Model-based control of Micro-Gas Turbines

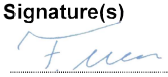
**Authored by** (in block letters):
*For papers written by groups the names of all authors are required.*

| **Name(s):** | **First name(s):** |
|---|---|
| Ulmer | Fabian |
| | |
| | |
| | |
| | |

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

| **Place, date** | **Signature(s)** |
|---|---|
| Haifa, 2. May 2023 |  |
| | |
| | |
| | |
| | |

*For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.*