Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for
Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme
und Regelungstechnik

151-0567-00
Engine Systems

Exercise Instructions

Idle Speed Control of a 2.5-liter 5-Cylinder SI Engine

Institute for Dynamic Systems and Control
Swiss Federal Institute of Technology (ETH) Zurich

Fall Semester 2017

Lecturer:
Prof. Christopher Onder
onder@idsc.mavt.ethz.ch

Exercise Supervision:
Johannes Ritzmann, ML K 41
Raffi Hedinger, ML K 41

Contents

Introduction

Idle Speed Control System Design Exercise

The aim of this tutorial is to overview the design process of a model-based idle-speed control system. It is not inteded to show all steps of the design and realization of such a control system: The control problem here will be formulated as a MISO problem and only feedback action will be investigated. Instead, the focus will be on showing how some of the basic building blocks introduced in the class text book *Introduction to Modeling and Control of Internal Combustion Engine Systems* (which will from now on be referred to as 'the text book') can be combined to form a mathematical model and on how the parameters of this model can be identified using measurements. Using this mathematical model, a control system will be designed.

Students attending the lecture *Engine Systems* should gain experience on their own treating this simplified task. Since the exercise is rather extensive, the students will have to work in a team of three to five persons and an important step will be the scheduling and structuring of the exercise into subproblems that can be solved by individual subteams.

To emphasize the importance of splitting the job into subtasks, the following milestones will have to be handed in on a defined date:

1. Synthesis of a mathematical model describing the dynamic behavior of the relevant engine parts.
2. Parameter identification and model validation using data obtained on a dynamic test bench with a 2.5-liter SI engine.
3. Model linearization to obtain a finite-dimensional linear system description.
4. Synthesis of an idle speed control system.

At the end of the course, a direct comparison between the performances of the controllers of the various teams will be organized. A predefined cost function will have to be minimized and the competition winning group will not only get fame, honor and recognition but also a voucher for a meal at Jimmy's Pizza.

Preliminary Manifold Modeling Exercise

A relatively small preliminary exercise will be the modeling of the intake manifold of an engine. Most of the tools required to solve the idle speed control system (ISCS) design exercise are introduced here in an easier-to-grasp environment and understanding the general concepts will be a great benefit for the ISCS exercise. A solution, including programming code, will be handed out when the exercise is finished and most Matlab-related problems arising in the ISCS exercise can be solved by the students autonomously by studying the provided code.

Chapter 1

Preliminary Exercise - Modeling of an Intake Manifold

1.1 Task Description

The aim of this exercise is to model an intake manifold of a 2.5-liter SI engine, which is operated at constant speed (note that this engine differs from the one used in the ISCS exercise). Certain unknown parameters have to be estimated from measurements. In the end, the system is linearized and normalized for the controller design. Figure 1.1 shows a sketch of the system. The control input of the system is the mass flow $\dot{m}_{in}(t)$ into the intake manifold. (Note that in general, the input to the engine is not the mass flow. This is only a simplification for the current problem setting.)

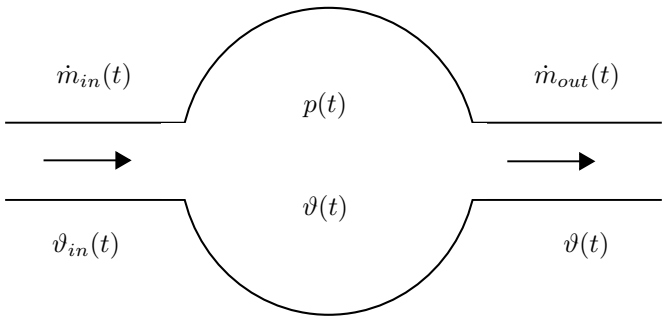

Figure 1.1: Sketch of the Intake Manifold.

The differential equations describing the pressure $p(t)$ and temperature $\vartheta(t)$ in the intake manifold can be found in the text book in the chapter *Mean Value Models* but they are repeated here for convenience:

Assuming that no heat losses from the intake manifold to the ambient take place (the adiabatic case) and considering that the changes of the inlet (ambient) temperature $\vartheta_{in}(t)$ are small, the following relations can be stated:

$$\frac{\partial p(t)}{\partial t} = \frac{\kappa \cdot R}{V_m} \cdot (\dot{m}_{in}(t) \cdot \vartheta_{in}(t) - \dot{m}_{out}(t) \cdot \vartheta(t)) \tag{1.1}$$

$$\begin{aligned}\frac{\partial \vartheta(t)}{\partial t} = \frac{\vartheta(t) \cdot R}{p(t) \cdot V_m \cdot c_v} \cdot \Big[& c_p \cdot \dot{m}_{in}(t) \cdot \vartheta_{in}(t) - c_p \cdot \dot{m}_{out}(t) \cdot \vartheta(t) \\ & - c_v \cdot (\dot{m}_{in}(t) - \dot{m}_{out}(t)) \cdot \vartheta(t) \Big]\end{aligned} \tag{1.2}$$

If the gas remaining in the cylinder at TDC is assumed to expand isentropically into the exhaust manifold once the exhaust valve is opened, the massflow out of the manifold can be modeled using the following equation:

$$\dot{m}_{out}(t) = \frac{\dot{V}_{\text{ideal}}}{R \cdot \vartheta(t)} \cdot \left(p(t) \cdot \frac{\epsilon + 1}{\epsilon} - \frac{p_e \cdot \vartheta(t)}{\epsilon \cdot \vartheta_e} \right). \tag{1.3}$$

The engine is running at a constant speed and therefore the volume flow $\dot{V}_{\text{ideal}}$ remains constant. In the table below a short explanation of all the parameters is given:

c_p	=	1003 $\frac{\text{J}}{\text{kg} \cdot \text{K}}$	Specific heat capacity at constant pressure.
c_v	=	717 $\frac{\text{J}}{\text{kg} \cdot \text{K}}$	Specific heat capacity at constant volume.
κ	=	$\frac{c_p}{c_v}$	
ϵ	=	10	Compression Ratio: Ratio between the cylinder volume at bottom dead center (maximal volume) to the volume at top dead center (minimal volume).
p_e	=	1×10^5 Pa	Pressure of the exhaust gas remaining in the cylinder.
R	=	287 $\frac{\text{J}}{\text{kg} \cdot \text{K}}$	Specific gas constant of air.
ϑ_e	=	400 K	Temperature of the exhaust gas remaining in the cylinder.
ϑ_{in}	=	295 K	Temperature of the air entering the intake manifold.
V_m	=	*to be identified* m^3	Volume of the intake manifold.
$\dot{V}_{\text{ideal}}$	=	*to be identified* $\frac{\text{m}^3}{\text{s}}$	Volume flow of the engine at constant speed.

The following points need to be treated to accomplish this exercise.

1. **Week 1:** *Build a SIMULINK model describing the system.*
   This is: implement the equations given above and make sure your system is running. Make reasonable assumptions for the unknown parameter values V_m and $\dot{V}_{ideal}$.

2. **Week 2:** *Estimate the unknown parameters V_m and $\dot{V}_{ideal}$ using the given measurement data. The data can be downloaded from the class web page.*
   Note that you have dynamic measurement and therefore an analytic solution to the parameter identification problem is extremely hard to find (if not impossible). Check the additional material from the web page (especially the example of how to use the command `fminsearch`) if you are not sure how to proceed.

3. **Week 3:** *Normalize and linearize the model using the Matlab command* `linmod`.
   Perform a steady-state simulation with the nominal input $\dot{m}_{in} = 0.01\ \frac{kg}{s}$ in

order to get the equilibrium values of the states and the output of the system. See Appendix A.2.2 and Appendix A.2.3 for clarification of the normalization and linearization procedure. Figure 1.2 shows the Simulink structure of the system that is used for the linearization.

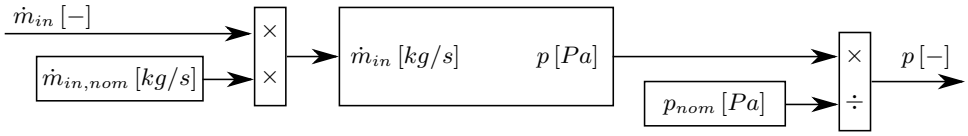

Figure 1.2: Structure of the normalized nonlinear system which is used for the linearization. The dimensionless quantities are connected to the physical values according to Eq. (A.4).

4. *Compare the system responses of the normalized and linearized model to the one of the nonlinear model.*
   Take into account that the input and the output of the linearized and normalized system are around the linearized values and that they are scaled according to the normalization. Figure 1.3 shows the model structure that is used for the simulation of the normalized and linearized system.

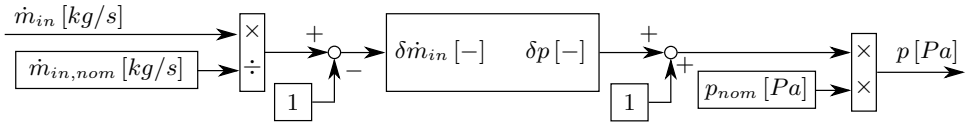

Figure 1.3: Structure used for the simulation of the normalized and linearized system.

5. *Optional: Compare the behavior to an isothermal manifold. You can find the corresponding equations in the text book.*

1.2 Hand In

Hand in a written report that summarizes your solution of the intake manifold exercise until the Monday of the 4th lecture week (see the factsheet on the class web page for the specific date).

1.3 File Structure

Figure 1.4 shows a general structure of a parameter identification, normalization and linearization process:

- **Main.m**
  First, the identification data and the parameters are loaded. Then the other scripts are called from this main script. In addition, the identified parameters as well as the normalized and linearized model are validated. If you run the main file, the entire process should be executed.

- **Parameters.m**
  The parameter file contains the known parameters, the initial conditions of the integrators in the system and the simulation options. Once these values are specified, they can easily be loaded by running the parameter script.

- **ParID.m**
  The unknown parameters of all submodels are identified. If the submodel to be identified is linear in the parameters, the parameters can be identified directly via the least squares algorithm. For the general nonlinear identification, a separate error function *Modelerror.m* and model *IdentificationModel.slx* have to be built for *each* identification process.

- **Linearization.m**
  First, a steady-state simulation is performed in order to obtain the normalization and linearization point. Second, the normalized nonlinear model *NonlinearModelNorm.slx* is built identical to the structure shown in Fig. 1.2. The normalized and linearized model *LinearizedModel.slx* is then obtained by linearizing the normalized nonlinear model around the linearization point.

- **Modelerror.m**
  This is the error function which is called via fminsearch. It simulates the model *IdentificationModel.slx* with the current parameters and calculates the error between the model and the measurement.

- **NonlinearModel.slx**
  This the entire nonlinear model of the system which is used for validating the identified parameters.

- **NonlinearModelNorm.slx**
  This model is built by normalizing the inputs and outputs of the model *NonlinearModel.m*.

- **LinearizedModel.slx**
  The linear model is obtained when the normalized nonlinear model is linearized.

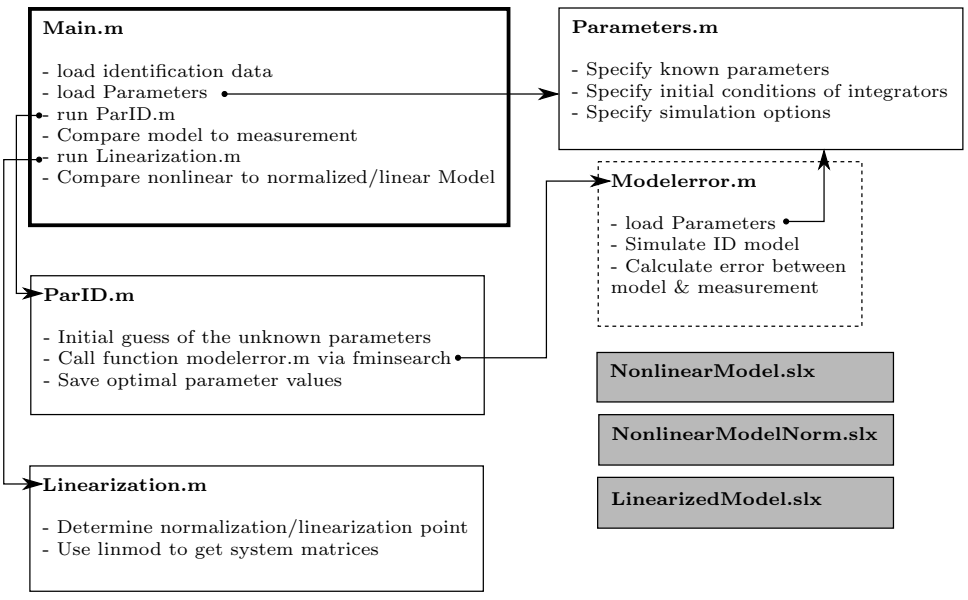

Figure 1.4: General file structure of a parameter identification and linearization process.

1.4 Hints

- All of the measured signals in this preliminary exercise are recorded at 100 Hz. The *solver options* in the Simulink *configuration parameters* should therefore be set to *Type: Fixed-step* and *Fixed-step size (fundamental sample time):* $0.01\,s$
- A Simulink model can be simulated directly from a *m-file* by using the `sim` command (type `doc sim` for further information). The simulation options can be specified as well via the `simset` command. This should be done in the *parameter* file. Use the following specification:
  simopt = simset('SrcWorkspace','current','FixedStep',1e-2)

Chapter 2

Idle Speed Control Exercise

2.1 Problem Setting

Modern spark-ignited (SI) engines have rather low idle speed limits (around 700 rpm) in order to minimize fuel consumption and pollutant emissions. The drawback of these low limits is that sudden changes in engine torque (electric load on the alternator, AC compressor, etc.) might stall the engine[1]. A fast idle speed control system (ISCS) is therefore mandatory.

The basic structure of a modern ISCS for SI engines is shown in Fig. 2.1. The engine speed is the main input signal while other engine variables (oil temperature, intake air pressure and temperature, etc.) are used to compute the reference speed at which the engine is supposed to run.

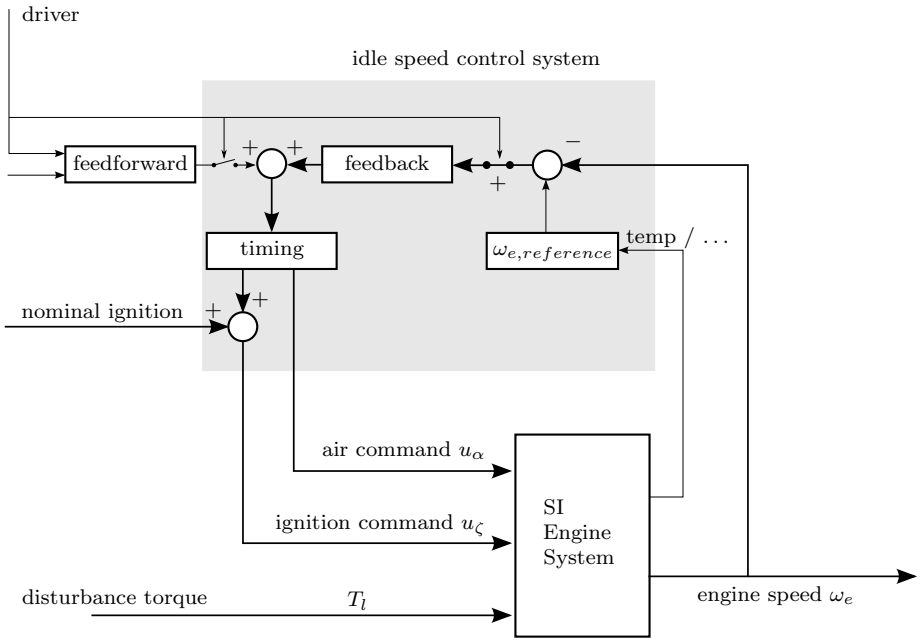

Figure 2.1: Basic Structure of an Idle Speed Controller

[1]The higher the idle speed, the more kinetic energy is saved in the moving parts (this is a reservoir), which leads to a slower decrease of the engine speed when negative torques are applied.

As shown in Fig. 2.1, the disturbance torque is compensated in a coordinated way using both the air command and the spark command. This approach allows for a fast reaction to unmeasurable load disturbances. In fact, as shown in Sect. 3.2.1 of the text book, the ignition input is able to change the engine torque almost instantaneously albeit at the price of a reduced engine efficiency and higher pollutant emissions. Therefore the 'timing' block in the ISCS utilizes the spark channel in a first phase in which the air command has not yet produced the desired change in torque. As soon as that command starts to produce the desired action the ignition command is phased back to its nominal value.

In conventional engine systems with mechanical throttle valves the air command actuates a parallel idle-speed control bypass valve. In engine systems with electronically controlled throttle valves, the idle-speed controller usually directly commands that element. Figure 2.2 shows the layout of such a modern engine control system.

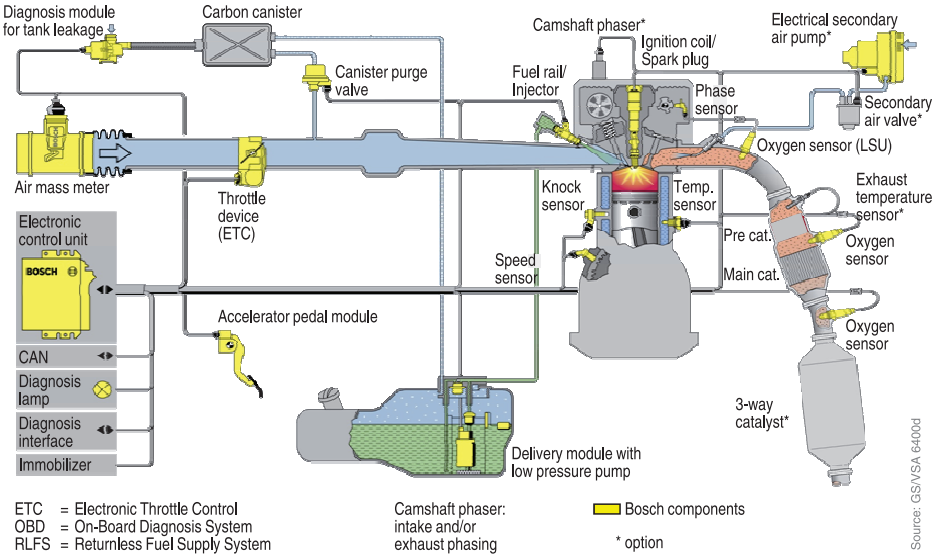

Figure 2.2: Layout of a modern engine management system, reprinted with the permission of Robert Bosch GmbH.

The design of a complete ISCS is well beyond the scope of this tutorial. The following simplification are therefore made:

1. The injection control system is assumed to be able to keep the air/fuel ratio at its stoichiometric value of $\lambda = 1$.

2. The disturbance torque is assumed not to be measurable, no feedforward control is therefore possible and only the feedback part will be considered.

The engine used in this case study is a 2.5-liter 5-cylinder engine with conventional port fuel injection. The air path control and the control of the ignition angle will be designed and implemented in a digital signal processor real-time system. The output of that control system are the throttle valve position command signal u_α and the retardation of the ignition angle from its maximum brake torque (MBT) position du_{ign}. The air/fuel ratio is kept at the desired value by a separate controller.

The controller can realize an engine torque increase and decrease by opening and closing the throttle respectively. In order to realize this ability with the ignition command, the engine has to be operated at an inefficient ignition angle ζ_0 which is the nominal ignition angle setpoint for idling conditions. The ignition angle ζ_0 corresponds to a retardation by $du_{ign,nom}$ from the MBT ignition angle ζ_{MBT}. As a consequence, the control system is able to react to an engine speed increase as well as a decrease by adapting the value of du_{ign} accordingly, see Fig. 2.3.

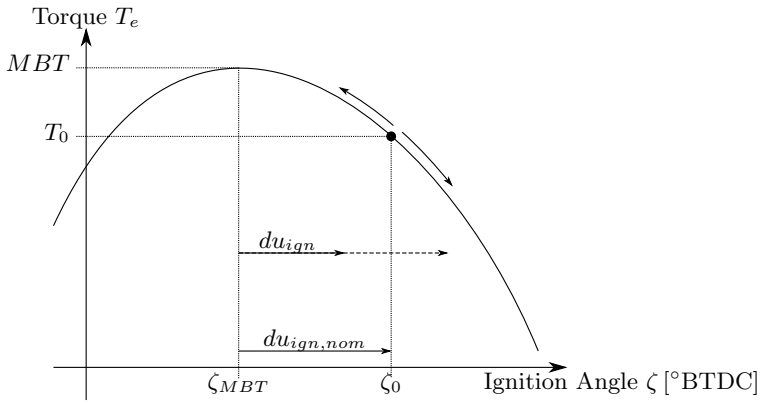

Figure 2.3: The engine is operated at an operating point with ignition angle ζ_0 in order to enable a torque increase and decrease by advancing and retarding the ignition angle respectively. Be aware that the ignition angle has the unit *degree before top dead center* (BTDC). Therefore, du_{ign} is negative for a later ignition.

The engine speed responds faster to a change in the ignition angle than to a change in the throttle plate angle. The SIMO controller can use this property to allow a fast tracking of the reference engine speed $\omega_{e,ref}$. At first, the retardation of the ignition angle du_{ign} is adjusted to get a fast response. Afterwards, as soon as the engine speed reacts to the change of the throttle plate angle u_α, the retardation of the ignition angle is controlled back to its nominal value $du_{ign,nom} = \zeta_0 - \zeta_{MBT}$ in order to enable a increase and decrease of the torque again. The desired step response of the output signals of the controller is shown in figure Fig. 2.4.

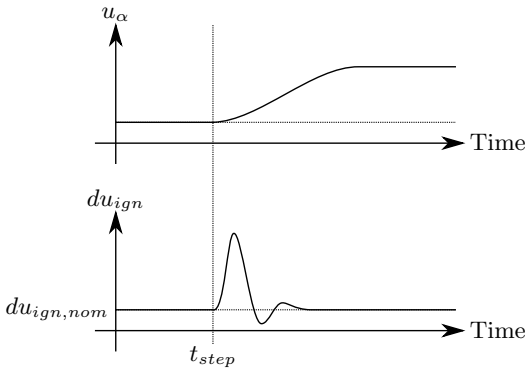

Figure 2.4: Desired step response of the individual channels of the SIMO controller

2.2 Milestone 1
Mathematical Model of the Plant

2.2.1 Task Description

For milestone 1, each group has to create a simplified mathematical model of an engine. It has to be built in Simulink and reasonable values for the model parameters have to be estimated. In milestone 2, this model will be used to identify the model parameters from measurements. Carefully read the entire Section 2.2 before you start to implement your model.

2.2.2 The Plant

The system to be modeled is shown in Fig. 2.5. It is a simplified version of the general SI engine system as introduced in Section 2.2.1 in the text book. According to the simplifications listed in the previous section, the fueling part may be omitted. The three input signals (bold) and some of the measured variables are shown in Fig. 2.5 (see following paragraph for a complete list of all measured variables).

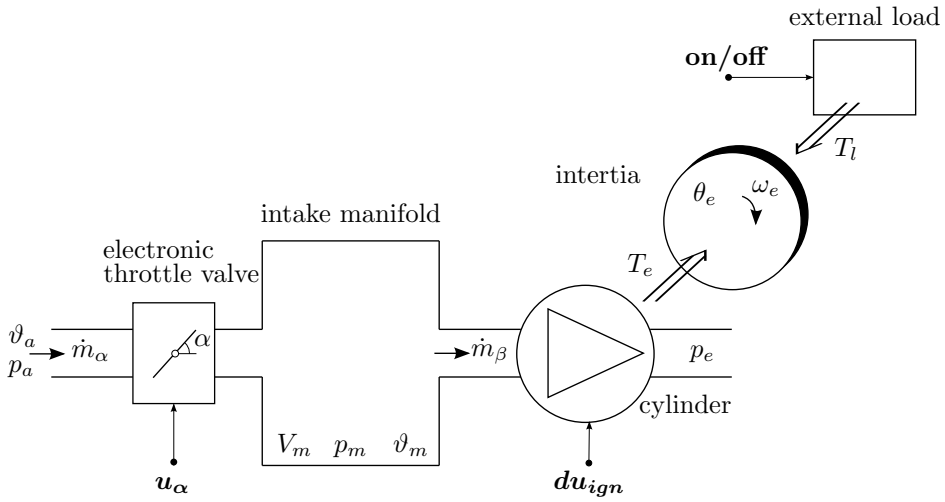

Figure 2.5: Simplified representation of the problem setting.

The engine is modeled as a mean-value volumetric pump, according to the theoretical background in Chapter 2 of the text book. The torque produced by the engine is linked to the amount of air aspirated by the engine and acts positively on the engine inertia. The engine is assumed to be decoupled from the rest of the power train (this means that the clutch is opened, which is the worst-case situation because the inertia is minimal) and the load torque is presumed to act on the engine flywheel directly.

A tricky task is the generation of a known load disturbance for test purposes. The dynamometer cannot be used since its inertia would change the dynamic behavior of the system considerably. A viable alternative is to utilize the engine's alternator as a variable load source by connecting it to an electronically controllable load. In this case a 1 kW load is used, which yields a load disturbance of up to 15 Nm (see Section 2.2.3 for further information).

The corresponding cause and effect diagram of the combined ISCS is shown in Fig. 2.6. While the shaded boxes indicate dynamic modules, the non-shaded boxes represent algebraic modules. The control inputs are u_α and du_{ign}, i.e. the command signal for the electronic throttle valve and the retardation of the ignition angle. The disturbance input u_l, which in the series application is assumed not to be measurable is the only exogenous disturbance signal. However, in order to test the performance of the ISCS, this disturbance has to be modeled in this exercise study.

The measured system variables are:

- $\dot{m}_\alpha$: the air mass flowing through the throttle into the intake manifold,
- ω_e: the engine speed,
- p_m: the intake manifold pressure,
- ϑ_m: the intake manifold temperature,
- p_a: the ambient pressure,
- ϑ_a: the ambient temperature,
- p_e: the exhaust manifold pressure,
- U_{bat}: the voltage at the alternator, and
- I_{gen}: the current flowing through the alternator.

The inputs are recorded as well:

- u_l: load on/off signal,
- u_α: throttle plate angle command signal,
- du_{ign}: retardation of the ignition angle from ζ_{MBT}

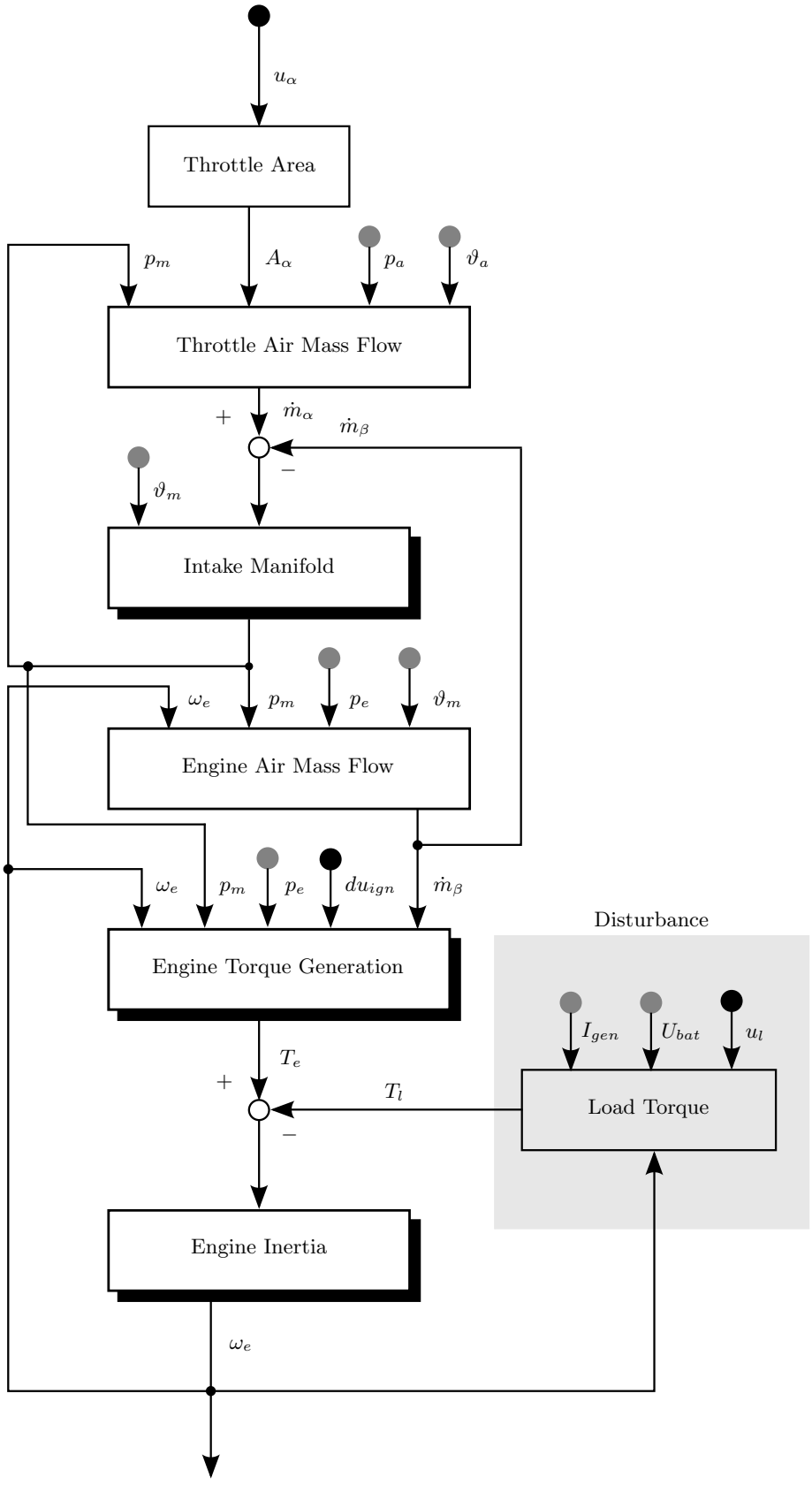

Figure 2.6: Cause and Effect Diagram of the Engine Model.

2.2.3 The Subsystems of the Plant

Most of the subsystems indicated by boxes in Fig. 2.6 are discussed in chapter 2 of the text book. Here, those results are adapted to the ISCS problem and a formal description of each block is given to facilitate the subsequent parameter identification and control system synthesis steps. The underlying physical principles of each subsystem are described in this section. The structure of the Simulink model is exactly as illustrated in Fig. 2.6. The parameter identification is presented in Section 2.3.

Throttle Area

Object:	Models the open area A_α as a function of the input signal $u_\alpha(t)$ (defines how wide the throttle is opened).	
Signals:	Input:	Control signal $u_\alpha(t) \in [0, \ldots, 100]$
	Output:	Open area A_α, $[m^2]$
Assumptions:	The dynamics of the throttle actuator may be neglected because the throttle used for this engine is much faster than the dynamics of the engine speed we are interested in.	
Equations:	*Affine Model*	

$$A_\alpha\left(u_\alpha(t)\right) = \alpha_0 + \alpha_1 \cdot u_\alpha$$

Parameters:	$\alpha_\mathbf{0}$	**To be identified**	$\left[\mathrm{m}^2\right]$	initial value $\sim 3 \cdot 10^{-6}$
	$\alpha_\mathbf{1}$	**To be identified**	$\left[\mathrm{m}^2\right]$	initial value $\sim 6 \cdot 10^{-6}$

Throttle Air Mass Flow

Object:	Models the air mass flow through the throttle which is caused by the pressure difference.	
Signals:	Input:	Open area A_α, $[m^2]$ Ambient air temperature upstream $\vartheta_a(t)$, $[K]$ Pressure upstream $p_a(t)$, $[Pa]$ Pressure downstream $p_m(t)$, $[Pa]$
	Output:	Air mass flow $\dot{m}_\alpha(t)$, $\left[\frac{kg}{s}\right]$
Assumptions:	No friction in the flow; no inertial effects in the flow, i.e. the piping around the valves is small compared to the receivers to which they are attached; completely insulated conditions (no additional energy, mass, etc. enters the system; all flow phenomena are zero dimensional, i.e. no spatial effects need to be considered.	

Equations: *Approximation*

$$\dot{m}_\alpha(t) \approx \begin{cases} A_\alpha(t) \cdot \frac{p_a(t)}{\sqrt{R \cdot \vartheta_a(t)}} \frac{1}{\sqrt{2}}, & \text{if } \frac{p_m(t)}{p_a(t)} < \frac{1}{2}, \\ A_\alpha(t) \cdot \frac{p_a(t)}{\sqrt{R \cdot \vartheta_a(t)}} \sqrt{\frac{2p_m(t)}{p_a(t)} \left[1 - \frac{p_m(t)}{p_a(t)}\right]}, & \text{if } \frac{p_m(t)}{p_a(t)} \geq \frac{1}{2}, \end{cases}$$

Parameters:	R	287	$\left[\frac{\mathrm{J}}{\mathrm{kg} \cdot \mathrm{K}}\right]$

Intake Manifold

Object: Describes the emptying and filling dynamics of the intake manifold.

Signals:
- Input: Difference of air mass flows $\dot{m}_\alpha(t) - \dot{m}_\beta(t)$, $[\frac{kg}{s}]$
- Input: Air temperature in the intake manifold $\vartheta_m(t)$, $[K]$
- Output: Intake manifold air pressure $p_m(t)$, $[Pa]$

Assumptions: Pressure and temperature in the intake manifold are constant (zero dimensional model). The fluids in the intake manifold are ideal gases. Slowly varying intake manifold temperature.

Equations: *Total derivative of the ideal gas law for a slowly varying intake manifold temperature ($\frac{d}{dt}\vartheta_m \approx 0$)*

$$\frac{\mathrm{d}}{\mathrm{d}t} p_m(t) = \frac{R \cdot \vartheta_m(t)}{V_m} \left[\dot{m}_\alpha(t) - \dot{m}_\beta(t)\right]$$

Parameters:	R	287	$\left[\frac{\mathrm{J}}{\mathrm{kg} \cdot \mathrm{K}}\right]$	
	$\mathbf{V_m}$	**To be identified**	$[\mathrm{m}^3]$	initial value $\sim 7 \cdot 10^{-3}$

Engine Air Mass Flow

Object: Models the mass flow through and induced by the engine which is equal to the air mass flow that exits the intake manifold.

Signals:
- Input: Intake manifold gas pressure $p_m(t)$, $[Pa]$
- Input: Exhaust manifold gas pressure $p_e(t)$, $[Pa]$
- Input: Intake manifold gas temperature $\vartheta_m(t)$, $[K]$
- Input: Engine speed $\omega_e(t)$, $[\frac{rad}{s}]$
- Output: Air mass flow out of the intake manifold $\dot{m}_\beta(t)$, $[\frac{kg}{s}]$

Assumptions: The pressure at the intake valves is equal to the pressure in the intake manifold. The temperature drop due to the evaporation of the injected fuel is neglected. The air/fuel ratio is assumed to be constant and stoichiometric ($\lambda = 1$). The engine is a four-stroke engine, which is the reason for the factor $\frac{1}{2}$ in front of the equation for the fresh mixture mass flow (one engine cycle corresponds to two revolutions of the crank shaft).

Equations: *Fresh mixture that enters the cylinder*

$$\dot{m}_e(t) = \frac{1}{2} \cdot \frac{p_m(t)}{R \cdot \vartheta_m(t)} \cdot \lambda_l(p_m, \omega_e) \cdot V_d \cdot \frac{\omega_e(t)}{2\pi}$$

Air mass flow out of the intake manifold

$$\dot{m}_\beta(t) = \frac{\dot{m}_e(t)}{1 + \frac{1}{\lambda(t) \cdot \sigma_0}}$$

where $\sigma_0 = 14.67$ is the stoichiometric air/fuel ratio for gasoline.
Note: $\lambda \neq \lambda_l$! λ is associated with the air/fuel ratio, whereas λ_l represents the volumetric efficiency.

Volumetric efficiency

$$\lambda_l(p_m, \omega_e) = \lambda_{lp}(p_m) \cdot \lambda_{l\omega}(\omega_e)$$

and

$$\begin{aligned} \lambda_{lp}(p_m) &= \frac{V_c + V_d}{V_d} - \frac{V_c}{V_d} \cdot \left(\frac{p_e(t)}{p_m(t)} \right)^{\frac{1}{\kappa}} \\ \lambda_{l\omega}(\omega_e) &= \gamma_0 + \gamma_1 \cdot \omega_e(t) \end{aligned}$$

where V_d is the engine's displacement volume and V_c is its compression volume, i.e. the volume in the cylinder at TDC.

Parameters:				
	R	287	$\left[\frac{\mathrm{J}}{\mathrm{kg} \cdot \mathrm{K}}\right]$	
	V_d	$2.48 \cdot 10^{-3}$	$[\mathrm{m}^3]$	
	V_c	$2.48 \cdot 10^{-4}$	$[\mathrm{m}^3]$	
	σ_0	14.7	$[-]$	
	κ	1.35	$[-]$	
	$\boldsymbol{\gamma_0}$	**To be identified**	[-]	initial value ~ 0.6
	$\boldsymbol{\gamma_1}$	**To be identified**	$\left[\frac{\mathrm{s}}{\mathrm{rad}}\right]$	initial value ~ 0.002

Engine Torque Generation

Object:	Estimation of the torque produced by the engine.	
Signals:	Input:	Air mass flow into the cylinder $\dot{m}_\beta(t)$, $[\frac{kg}{s}]$
		Engine speed $\omega_e(t)$, $[\frac{rad}{s}]$
		Intake manifold gas pressure $p_m(t)$, $[Pa]$
		Exhaust gas pressure $p_e(t)$, $[Pa]$
		Retardation of the ignition angle $du_{ign}(t)$, $[^\circ CA]$
	Output:	Engine torque $T_e(t)$, $[Nm]$
Assumptions:	The engine can be simplified as a Willans machine. The air/fuel ratio is constant ($\lambda = 1$).	
Equations:	*Torque produced by the engine*	

$$\begin{aligned} T_e(t) &= p_{me} \cdot \frac{V_d}{4\pi} \\ &= (e \cdot p_{m\varphi} - p_{me0f} - p_{me0g}) \cdot \frac{V_d}{4\pi} \end{aligned}$$

where

$$\begin{aligned} e \cdot p_{m\varphi} &= (\eta_0 + \eta_1 \cdot \omega_e(t)) \cdot e_\zeta \cdot \frac{H_l}{V_d} \cdot m_\varphi \\ &= (\eta_0 + \eta_1 \cdot \omega_e(t)) \cdot e_\zeta \cdot \frac{H_l}{V_d} \cdot \frac{4\pi \cdot \dot{m}_\varphi(t)}{\omega_e(t)} \\ &= (\eta_0 + \eta_1 \cdot \omega_e(t)) \cdot e_\zeta \cdot \frac{H_l}{V_d} \cdot \frac{4\pi}{\lambda \cdot \sigma_0} \cdot \frac{\dot{m}_\beta(t - \tau_{IPS})}{\omega_e(t - \tau_{IPS})} \end{aligned}$$

and

$$e_\zeta = 1 - k_\zeta \cdot \underbrace{(\zeta(t - \tau_{seg}/2) - \zeta_{MBT}(t - \tau_{seg}/2))^2}_{du_{ign}(t - \tau_{seg}/2)^2}$$

the delays are defined as follows

$$\begin{aligned} \tau_{IPS} &= \frac{2\pi}{\omega_e(t)} \\ \tau_{seg} &= \frac{4\pi}{5 \cdot \omega_e(t)} \end{aligned}$$

and the friction and gas exchange losses are given by

$$\begin{aligned} p_{me0f} &= \beta_0 \cdot \frac{4\pi}{V_d} \\ p_{me0g} &= p_e(t) - p_m(t) \end{aligned}$$

Remarks: As delays occur within the feedback loop, they have an influence on the dynamic behavior of the idle speed control system.
Usually, the engine friction is assumed to be of the form $p_{me0f} = \beta_0 + \beta_2 \cdot \omega_e^2$. However, in this specific case where only low engine speeds are considered, a constant friction gives good results.
ζ_{MBT} is the ignition angle that leads to the highest possible efficiency (highest torque, MBT). All the measurement data you are provided with was taken at this ignition angle. This means that e_ζ is equal to 1. The value of ζ_{MBT} itself is not required, since only the retardation of the ignition angle from the ignition angle at maximum brake torque du_{ign} is used. The parameter k_ζ was identified separately by the exercise supervisors and its value is given to you.
The torque produced by the engine does not respond immediately to an increase in the manifold pressure. The new engine torque will be achieved only after the induction-to-power-stroke (IPS) delay τ_{IPS} which is approximately the time for one revolution.
The torque production acts as a discrete system with a sampling period of τ_{seg}. The delay is on average one-half of this sampling time.

Parameters:				
	V_d	$2.48 \cdot 10^{-3}$	$\left[\mathrm{m}^3\right]$	
	H_l	$42.5 \cdot 10^{6}$	$\left[\frac{\mathrm{J}}{\mathrm{kg}}\right]$	
	k_ζ	$2.3345 \cdot 10^{-4}$	$\left[\frac{1}{(^\circ \mathrm{CA})^2}\right]$	
	$\boldsymbol{\eta_0}$	**To be identified**	[-]	initial value ~ 0.3
	$\boldsymbol{\eta_1}$	**To be identified**	$\left[\frac{\mathrm{s}}{\mathrm{rad}}\right]$	initial value $\sim -3 \cdot 10^{-4}$
	$\boldsymbol{\beta_0}$	**To be identified**	[Nm]	initial value ~ 7

Engine Inertia

Object: Describes the dynamic behavior of the engine speed.

Signals: Input: Engine torque $T_e(t)$, $[Nm]$
Load torque $T_l(t)$, $[Nm]$
Output: Engine speed $\omega_e(t)$, $[\frac{rad}{s}]$

Assumptions: The inertia of the engine is constant, all internal friction phenomena are included in T_e.

Equations: *Engine speed behavior*

$$\frac{\mathrm{d}}{\mathrm{d}t}\omega_e(t) = \frac{1}{\Theta_e}\left[T_e(t) - T_l(t)\right]$$

Parameters:				
	$\boldsymbol{\Theta_e}$	**To be identified**	$\left[\mathrm{kg} \cdot \mathrm{m}^2\right]$	initial value ~ 0.2

Load Torque

Object:	Models the behavior of the load torque unit.	
Signals:	Input:	On/off signal $u_l(t)$, [1/0]
		Battery voltage $U_{bat}(t)$, $[V]$
		Generator current $I_{gen}(t)$, $[A]$
		Engine speed $\omega_e(t)$, $[\frac{rad}{s}]$
	Output:	Load torque $T_l(t)$, $[Nm]$
Assumptions:	The generator has a constant efficiency η_{gen}.	
Equations:	*Load torque*	

$$T_l = \frac{U_{bat}(t) \cdot I_{gen}(t)}{\omega_e(t)} \cdot \frac{1}{\eta_{gen}}$$

Remarks: This block is only used to run simulations when designing the controller. It can obviously not be part of the controller as it describes the unpredictable disturbance and no feed forward control action is allowed.
A maximum load torque of $\approx 15\ Nm$ can be generated using our $1\ kW$ load, however, from the equation above one can see that the generated load torque highly depends on the engine speed. Furthermore, this very simplified model of the generator neglects the fact that the generator has a maximum torque characteristics that depends on the generator speed: the required $1\ kW$ can not be generated at every generator speed. In those cases, the battery delivers the additional power and has to be recharged by the generator when the load is shut off again. It is thus worth analyzing the relation between engine speed and generated electrical power before using this block to design any controller!

Parameters:	η_{gen}	0.7	$[-]$

2.2.4 Remarks

All of these blocks form a mathematical description that approximates the dynamic behavior of the idle speed system. The engine speed is limited to the range of interest (70 - 200 rad/s).

2.2.5 Hints

- All of the measured signals are recorded at 1 kHz. The *solver options* in the Simulink *configuration parameters* should therefore be set to *Type: Fixed-step* and *Fixed-step size (fundamental sample time):* $0.001\,s$ (this is much faster than the modeled dynamics of the system and ensures consistent results).
- If an algebraic equation has several inputs and only one output, it is generally easier to use the Simulink block *Fcn* (search for 'Fcn' in the Simulink Library Browser) than establishing a complex arrangement of basic Simulink blocks. An implemented example can be found in the solution of the preliminary intake manifold exercise.
- Each additional delay increases the complexity of any mathematical model. Therefore, add as few delays as possible to the system. In this case, two delays are required.
- Use the block *variable Transport Delay* and set *Select delay type* to *Variable time delay* for the description of the time delay.
- Specify the following simulation options in the parameter file or the parameter structure:
  simopt = simset('Solver','ode1','FixedStep',1e-3,'SrcWorkspace','current')

2.3 Milestone 2 Parameter Identification and Validation

2.3.1 Task Description

In milestone 2, the parameters of the engine model created in milestone 1 have to be identified using measurements taken on the real engine. This model will be used to design a model based idle speed controller in future milestones. Carefully read the entire Section 2.3 before you start with the identification process.

2.3.2 Introduction

In principle, all of the parameters introduced in the last chapter could be estimated at once using nonlinear least-squares (LS) methods (for example the function *fminsearch* in Matlab), with the error to be minimized defined by

$$E = \sum_{i=1}^{N} \left[k_1 (p_{m,i} - \bar{p}_{m,i})^2 + k_2 (\dot{m}_{\alpha,i} - \bar{\dot{m}}_{\alpha,i})^2 + k_3 (\omega_{e,i} - \bar{\omega}_{e,i})^2 \right]. \tag{2.1}$$

Here the variables $p_{m,i}$, $\dot{m}_{\alpha,i}$, $\omega_{e,i}$ are the measured and $\bar{p}_{m,i}$, $\bar{\dot{m}}_{\alpha,i}$, $\bar{\omega}_{e,i}$ are the predicted system outputs, whereas $i = 1, \ldots, N$ are the measured data points. The constants k_i are weighting factors that are chosen to normalize the three different parts in the sum.

However, such a direct approach is not very likely to succeed, especially in the noisy environment of SI engines. It is difficult to find reasonable initial parameters and therefore the solution will rarely converge towards the *real* parameters. Only parameter values that locally minimize the cost function will be identified. Usually, a modular approach is more promising, i.e., the first subsystems in the causality chain are identified first using intermediate system outputs. Moreover, whenever possible, the problems should be reformulated such that equations result which are linear in the parameters (lip), in which case a linear LS estimation becomes feasible[2]. This approach is used below starting with the throttle valve as a first element.

2.3.3 The Throttle Parameters

Since the throttle parameters can be identified via the least square procedure, no Simulink model is required yet.

Combining the subsystems defined in Section 2.2.3 the air mass flow can be expressed as a function that linearly dependends on the throttle model parameters:

$$A_\alpha \left(u_\alpha(t) \right) = \alpha_0 + \alpha_1 \cdot u_\alpha(t) = \dot{m}_\alpha(t) \cdot \frac{\sqrt{2 \cdot R \cdot \vartheta_a(t)}}{p_a(t)} \tag{2.2}$$

The key assumption is that $\frac{p_m(t)}{p_a(t)} \leq \frac{1}{2}$, which can be guaranteed for this engine when the measurements are taken with the engine running in idling conditions.

Applying the nomenclature defined in Appendix B, the following LS problem can

[2]For further information on the solution of linear LS problems see Appendix B.

be formulated:

$$M \cdot \vec{x} = \alpha_0 + \alpha_1 \cdot u_\alpha(t) = \begin{bmatrix} 1 & u_{\alpha,1} \\ 1 & u_{\alpha,2} \\ \dots & \dots \\ 1 & u_{\alpha,N} \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix}; \quad \vec{y} = \begin{bmatrix} \dot{m}_{\alpha,1} \cdot \frac{\sqrt{2 \cdot R \cdot \vartheta_{a,1}}}{p_{a,1}} \\ \dot{m}_{\alpha,2} \cdot \frac{\sqrt{2 \cdot R \cdot \vartheta_{a,2}}}{p_{a,2}} \\ \dots \\ \dot{m}_{\alpha,N} \cdot \frac{\sqrt{2 \cdot R \cdot \vartheta_{a,N}}}{p_{a,N}} \end{bmatrix}, \quad (2.3)$$

where the subscripts $1, \dots, N$ represent the discrete measurement samples. The measurements used for this identification should be of the quasi-static type in order not to have any dynamics distort the assumptions.

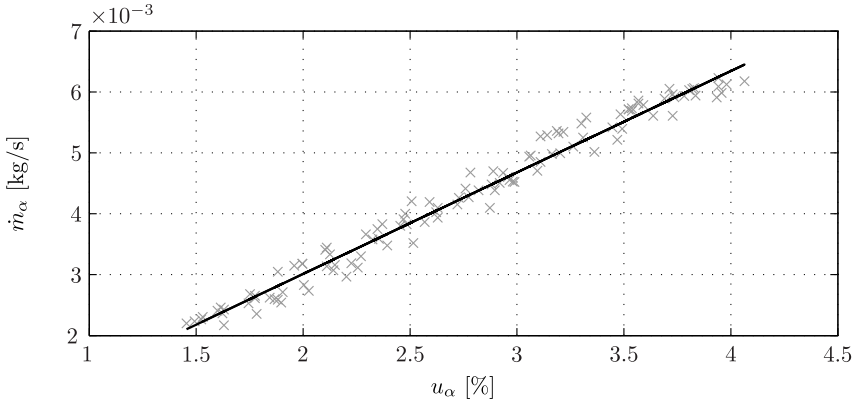

Figure 2.7: Comparison between measurements (x-entries) and parametrized throttle model (solid line)

2.3.4 The Engine Mass Flow Parameters

Since the engine mass flow parameters can be identified via the least square procedure, still no Simulink model is required.

In Section 2.2.3, the volumetric efficiency $\lambda_l(p_m, \omega_e)$ is introduced which is assumed to be a multi-linear function of the intake manifold pressure p_m and the engine speed ω_e, i.e., $\lambda_l(p_m, \omega_e) = \lambda_{lp}(p_m) \cdot \lambda_{l\omega}(\omega_e)$. The unknown function is the engine speed dependent part $\lambda_{l\omega}(\omega_e)$, which is modeled as an affine function. All of the equations describing the engine mass flow behavior can be reformulated to form the following equation:

$$\lambda_{l\omega}(\omega_e) = \frac{4\pi \cdot R}{V_d} \cdot \frac{\vartheta_m(t) \cdot \dot{m}_\alpha(t)}{p_m(t) \cdot \omega_e(t) \cdot \lambda_{lp}(p_m)} \cdot \left(1 + \frac{1}{\lambda(t) \cdot \sigma_0}\right). \quad (2.4)$$

Equation (2.4) is obtained assuming that the air mass entering the intake manifold $\dot{m}_\alpha$ equals the air mass entering the cylinder $\dot{m}_\beta$ which corresponds to static conditions. Therefore, it is mandatory to use quasistatic measurements here.

The following LS problem can be formulated:

$$M \cdot \vec{x} = \gamma_0 + \gamma_1 \cdot \omega_e(t) = \begin{bmatrix} 1 & \omega_{e,1} \\ 1 & \omega_{e,2} \\ \dots & \dots \\ 1 & \omega_{e,N} \end{bmatrix} \cdot \begin{bmatrix} \gamma_0 \\ \gamma_1 \end{bmatrix}; \qquad \vec{y} = \begin{bmatrix} \lambda_{l\omega,1} \\ \lambda_{l\omega,2} \\ \dots \\ \lambda_{l\omega,N} \end{bmatrix}, \tag{2.5}$$

where the subscripts $1, \dots, N$ represent the discrete measurement samples and $\lambda_{l\omega}$ is defined in Eq. (2.4).

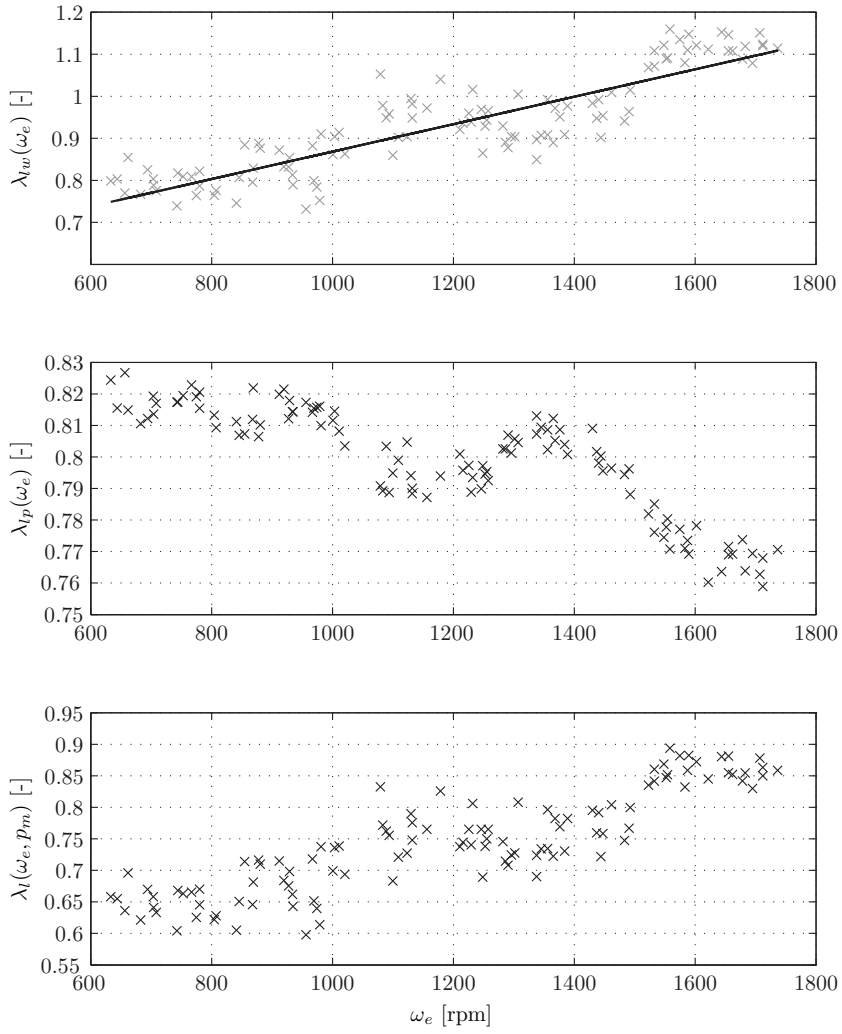

Figure 2.8: Comparison between measurements (x-entries) and parametrized volumetric efficiency model (solid line)

The volumetric efficiency does usually not take on values larger than 1[3] which is true for the engine used here, as shown in the bottom plot in Fig. 2.8. However, since it is modeled as a multi-linear function the individual factors are allowed to be outside the interval [0,...,1] as they do not represent real physical efficiencies.

[3]It is possible to get values > 1 if the intake manifold is aerodynamically optimized so that pressure waves supercharge the engine artificially.

2.3.5 The Intake Manifold Volume

Since the parameter V_m can not be identified with the least square procedure, a Simulink model with the previously identified blocks *Throttle Area*, *Throttle Air Mass Flow* and *Engine Air Mass Flow* and the block of interest, *Intake Manifold*, is required. In order to optimized the performance of the complete model we recommend not using the measured $\dot{m}_\alpha$ signal but to use the throttle air mass flow model instead. The only unknown parameter left in these first four blocks in the causality chain is the intake manifold volume V_m. As this is a parameter that has an influence on the model dynamics, it is necessary to use dynamic measurements in order to identify it properly. Therefore, throttle steps were applied to the engine in idling conditions (the clutch is disengaged) and the Matlab routine *fminsearch* was used to identify the value of V_m that minimizes the error $e = \sum_{i=1}^{N}(p_{m,measurement}(i) - p_{m,model}(i))^2$. The values $p_{m,measurement}(i)$ are obtained from the measurement and the values $p_{m,model}(i)$ are the predicted system outputs. Finally, $i = 1, \ldots, N$ is the index of the individual data points. The engine speed ω_e that is fed back in the model can be taken directly from the measurements for identification purposes.

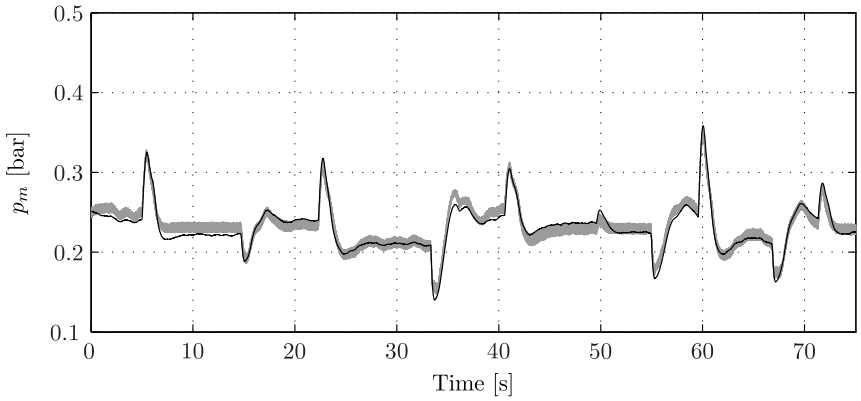

Figure 2.9: Comparison between the measurement (gray) and the model (black) intake manifold volume pressure p_m.

Figure 2.9 displays the agreement between model and measurements. Clearly, there is a reasonable range for possible values of V_m, which is several times larger than the displacement volume of the engine.

2.3.6 The Generator Efficiency

Since the load torque submodel represents the disturbance in the given problem setting, it is not part of the model used for the model-based controller design. Yet it is useful when it comes to simulating a load torque with a reasonable amplitude in order to test the performance of the controller. It is rather easy to measure the battery voltage U_{bat} and the generator current I_{gen}, which makes the identification of the quasistatic generator efficiency straight forward as well.

The engine is connected to the test bench which runs at a constant speed $\omega_{e,0}$. The torque $T_{e,0}$ produced by the engine is measured. When the electric load (a water boiler) is turned on, a part of the torque $T_{e,0}$ is used to run the alternator ('alternator' and 'generator' are synonyms in the given context). This leads, on the

one hand, to a generator current $I_{gen,1}$ (which is zero if the load is shut off) at a battery voltage $U_{bat,1}$ and on the other hand to a smaller measured torque $T_{e,1}$. The generator efficiency can finally be calculated as the ratio of the produced electric power and the drop in mechanical power:

$$\eta_{gen} = \frac{U_{bat,1} \cdot I_{gen,1}}{\omega_{e,0} \cdot (T_{e,0} - T_{e,1})} \tag{2.6}$$

Measurements of this sort were taken at different engine speeds. In the range of $\omega_e \in [70, \ldots, 200]\ rad/s$, the results show that a value of $\eta_{gen} = 0.7$ is a good approximation for the generator's efficiency. No further parameters have to be identified for this submodel.

2.3.7 The Engine Torque Generation and the Engine Inertia

A Simulink model with all the blocks of the causality chain is required for the identification process. Physically meaningless engine speeds resulting during the identification process can be avoided by limiting the engine speed signal with a saturation block to $20 - 500$ rad/s.

Deriving an expression of the engine friction p_{me0f} is a rather difficult task. Although it is straightforward in theory, the practical realization requires a sophisticated data acquisition system as well as a good understanding of the engine geometry. The key idea is to relate the mechanical work in the cylinder during the compression stroke and expansion stroke, $\int p\,dv$, to $e \cdot p_{m\varphi}$, which is the normalized mechanical work as a function of the injected fuel energy. This approach requires expensive, high-resolution in-cylinder pressure sensors, which are not available on the test bench available for this exercise. As a result a different approach was performed.

All of the cylinders of the engine were individually disabled, i.e. fuel cut off to one individual cylinder alternatingly. It is possible to express the term p_{me0} in the Willans Approach as a function of the absolute values of the torques generated with all cylinders working and with one cylinder disabled, respectively. The results of the measurements show that the friction can be assumed to be constant in the range of interest.

The remaining parameters in the subsystems *Engine Torque Generation* and *Engine Inertia* are η_0, η_1, β_0 and θ_e. They can be identified using a nonlinear optimization routine. This can lead to parameter values that are not necessarily physically reasonable but which will compensate for the errors made with the assumed form of the identified subsystems. Especially, the constant term of the thermodynamic efficiency η_0 directly compensates for an inaccurate value of the friction parameter β_0.

2.3.8 Validation of the Model

An important step after the parameter identification is the validation of the model with the newly identified parameters. It is important to show that the parameters identified in the last chapter yield a model that performs well when compared to different sets of measurements, especially data that was not used for the parameter identification.

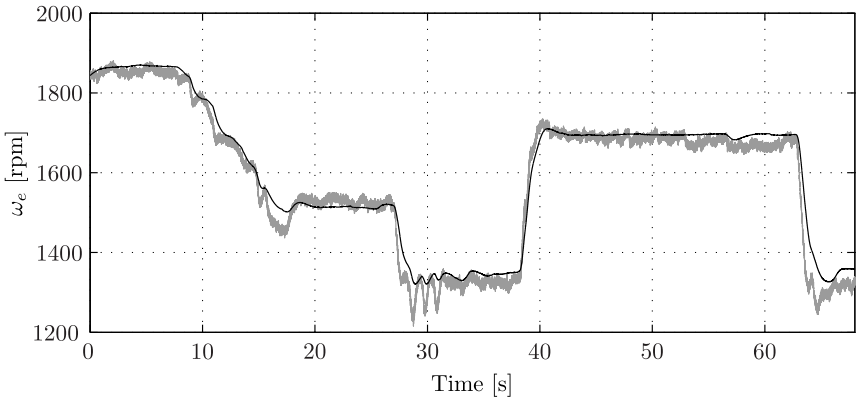

Figure 2.10: Comparison between measured (gray) and simulated (black) engine speed ω_e.

Figure 2.10 shows that the model is able to predict the engine speed for an arbitrary throttle input u_α (containing stepwise as well as slow changes) reasonably well.

2.3.9 Hints

- The MATLAB funtions *fminsearch* as well as *fmincon* can be used for the nonlinear optimization. The function *fmincon* provides an additional feature to define lower and upper bounds on the parameters. This can be useful to ensure that the identified parameters are not completely physically meaningless (e.g. a negative inertia). These optimization functions are very sensitive to the initial conditions of the parameters. Make sure that you choose them wisely, otherwise your solution will be a local optimum instead of the global optimum.

- If the load is switched off, the power of the alternator ($P_{alt} = I_{gen} \cdot U_{bat}$) is not zero, as there is always a power consumption e.g. by the spark plug. As a consequence, the alternator has a constant power which has to be considered when modeling the load. This constant offset of the power has to be subtracted from the power signal in order to apply only the disturbance torque to the model. The resulting signal is P_l which you can find in the measurement structure and use directly for your model instead of $I_{gen} \cdot U_{bat}$.

2.4 Milestone 3 Controller Synthesis

2.4.1 Task Description

In milestone 3, the engine model created in the previous milestones has to be normalized and linearized around reasonable nominal input values. Based on this linear model, an idle speed controller is created in Simulink. In order to pass milestone 3, the controller has to work in the Simulink environment. The nonlinear engine model has to be used to represent the real engine when tuning the controller. Only in milestone 4 the controller will be used on the real engine.

2.4.2 Normalization and Linearization

In order to apply the control theory presented in the previous control classes, the model needs to be normalized and linearized around reasonable nominal values of the inputs, the states and the output. Note that the model which is to be linearized does not contain all of the subsystems necessary to identify the model parameters. You will therefore have to delete certain subsystems shown in Fig. 2.6 and define your in- and output(s).
Hint: If you take a look at the controller structure, this should be a rather easy task.

See Appendix A.2.2 and Appendix A.2.3 for theoretical background on the normalization and linearization procedure. During the normalization and linearization, the delays will be treated externally. This means that the system without time delays is linearized with the `linmod` command, whereas the pure time delays are approximated by padé elements. In the end, the three systems are interconnected to get a normalized and linearized system with two inputs and one output as shown in Fig. 2.11. Take the following steps:

1. Perform a steady-state simulation with meaningful nominal inputs in order to obtain the equilibrium values of the states, the outputs, the delay times and the delayed values.

2. Split your model into three systems: Define two new inputs and two new outputs representing the delayed values in and out, see Fig. 2.11. The System P1 does not include any delays, only the nonlinear system. Systems P2 and P3, on the other hand, represent only the two delays.

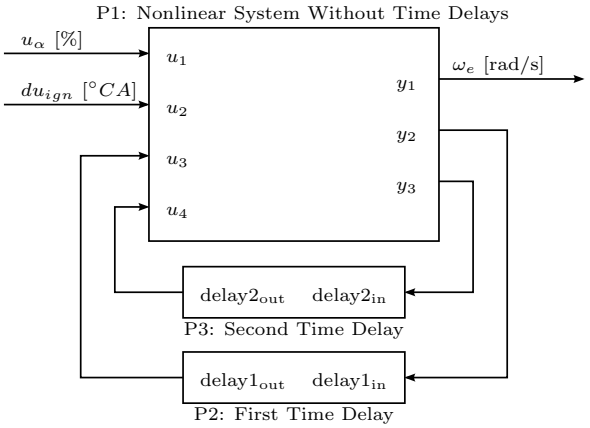

Figure 2.11: How to split up the system to get the nonlinear system without time delays P1 which is used for the normalization and linearization.

3. Build a normalized model of the nonlinear system without time delays P1. The structure of the normalized nonlinear model is shown in Fig. 2.12. Use the values obtained in step 1 to normalize the inputs ($u_{\alpha,nom}$ and $du_{ign,nom}$), the output ($\omega_{e,nom}$) and the delayed values ($delay1_{nom}$ and $delay2_{nom}$).

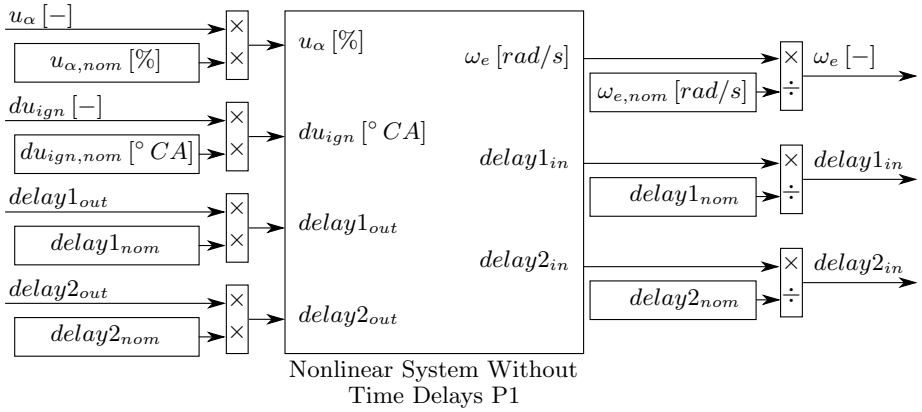

Figure 2.12: Structure of the normalized nonlinear system which is used for the linearization.

4. Linearize this normalized model using the Matlab command `linmod` and use the command `tf(ss(`$A_{lin}, B_{lin}, C_{lin}, D_{lin}$`))` with the resulting matrices to get the transfer functions of P1.

5. Approximate[4] the pure delays P2 and P3 by a 4^{th}-order padé element using the Matlab command `pade` (type `doc pade` for further information) with the equilibrium values of the delay times determined in step 1.

6. Connect the three systems using the Matlab command `iconnect` (type `doc iconnect` for further information and examples) according to Fig. 2.11. To do this, you need to define transfer functions. Proceed as shown in the examples in the Matlab documentation.
   Hint: One transfer function is: $y_1 = \mathrm{P1}(1,1) \cdot u_1 + \mathrm{P1}(1,2) \cdot u_2 + \mathrm{P1}(1,3) \cdot u_3 + \mathrm{P1}(1,4) \cdot u_4$. The third and fourth inputs are defined as: $u_3 = \mathrm{P2} \cdot y_2$ and $u_4 = \mathrm{P3} \cdot y_3$. In the end, you will have a system with 2 inputs and 1 output.

7. You can extract the system matrices A, B, C and D from the system defined above using the Matlab command `ssdata(tf(`yoursystem`))`.

8. Proceeding as stated in the previous steps, you will now have a set of system matrices A, B, C and D that represent the nonlinear system including the delays. The order of the system can be extracted from the size of the system matrix A, which is 20×20. Additionally to the two physical states of the real system (the intake manifold pressure and the engine speed), you will get four more states per delay (a n^{th}-order padé approximation introduces n additional states). The ten remaining states were introduced using the `iconnect` command. They basically represent the delay signals shown in Fig. 2.11. Their influence is negligible and you can get rid of them using the Matlab command `minreal(ss(`plant`))` that represents the system in the minimal possible order (size of A should be 10×10).

[4]Note that a time delay is linear by nature (The condition for a system to be linear is: applying the input $\alpha_1 \cdot u_1(t) + \alpha_2 \cdot u_2(t)$ will lead to the output $\alpha_1 \cdot y_1(t) + \alpha_2 \cdot y_2(t)$, where $\alpha_{1,2} \in \mathbb{R}$. This is also known as the superposition property.) It does, however, have infinite order!

2.4.3 Controller Synthesis

The aim of this milestone is to design a model-based single input multiple output (SIMO) idle speed controller for a 2.5-liter 5-cylinder engine. The input is the difference between a predefined reference engine speed and the current engine speed. The outputs are the command of the throttle plate angle u_α and the ignition angle du_{ign}. We assume a small sampling time T_s and therefore design a continuous-time controller that will be discretized using an emulation method.

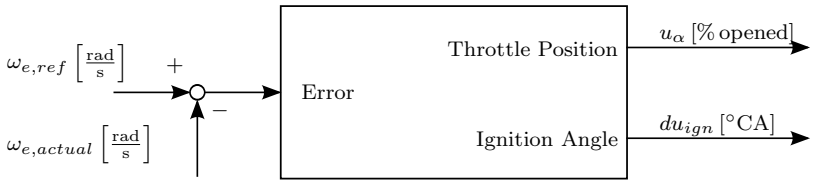

Figure 2.13: Basic Structure of the Idle Speed Controller.

The specifications for this control system are listed below:

- The reference speed varies between 70 and 200 rad/s.
- Load disturbances occur at random instances.
- The steady-state error in the engine speed caused by a constant load disturbance and a change in the reference speed has to be zero.
- The engine speed may not drop below 50 rad/s in order to prevent the engine from stalling.
- The overshoot of the engine speed after a load disturbance is acceptable.
- A minimum return difference $\mu = min\,|1 + L(j\omega)| = 0.5$ has to be achieved in order to cope with model uncertainties and neglected nonlinearities.

Hint: Try to avoid a controller cross-over frequency higher than 4 rad/s.

In this text, an LQGI approach is presented. At the end of the semester, you need to have at least one working controller. See Section 2.5 for more information about the competition.

Design of an LQG Controller with Integral Action - LQGI

This is an instruction for designing a LQG controller with integral action. We assume a basic understanding of LQR and LQG, but there is a short theory part in Appendix A. Additional information can be found in the script of the lecture *control systems II* (Regelungstechnik II) by Prof. Lino Guzzella.

1. **Plant extension**

The standard LQR controller only tries to bring the system back to the equilibrium point if a disturbance is present. In order to avoid having a steady state error, the controller has to be extended with an integrating part. As stated in Section 2.1 and shown in Fig. 2.4, the goal is to get back to the nominal ignition angle once the throttle plate angle has been adjusted to achieve the desired engine speed. Therefore, in a first step, the linearized system with the matrices A_s, B_s, C_s, D_s (which respresents the engine on the test bench) is extended by an integrator acting on the throttle plate angle signal, see Fig. 2.14. As a consequence, the command signal of the throttle plate angle is not regulated back to zero since the integrator is summing the signal up and is charged during the adaption of the throttle plate angle.

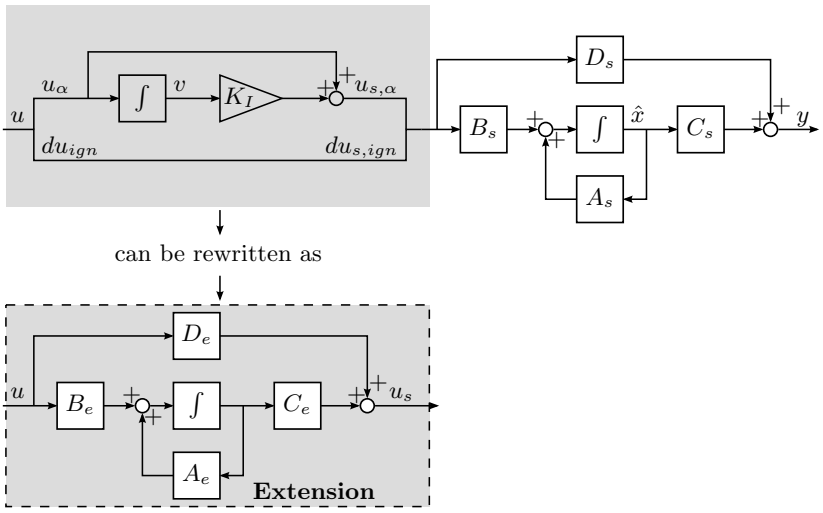

Figure 2.14: Extension of the plant.

The matrices of the extension are defined as follows:

$$A_e = 0, \qquad B_e = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad C_e = \begin{bmatrix} K_I \\ 0 \end{bmatrix}, \qquad D_e = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

As shown in Fig. 2.15, the series connection of the plant extension (index "e") and the linearized system (index "s") yields the extended plant with the new state vector

$$x = \begin{bmatrix} \hat{x} \\ v \end{bmatrix} \tag{2.8}$$

and the new system matrices

$$\frac{d}{dt}x = \underbrace{\begin{bmatrix} A_s & B_s C_e \\ 0 & A_e \end{bmatrix}}_{=:A} \cdot \begin{bmatrix} \hat{x} \\ v \end{bmatrix} + \underbrace{\begin{bmatrix} B_s D_e \\ B_e \end{bmatrix}}_{=:B} \cdot u \tag{2.9}$$

$$y = \underbrace{\begin{bmatrix} C_s & D_s C_e \end{bmatrix}}_{=:C} \cdot \begin{bmatrix} \hat{x} \\ v \end{bmatrix} + \underbrace{\begin{bmatrix} D_s D_e \end{bmatrix}}_{=:D} \cdot u \tag{2.10}$$

where the inputs vector is defined as:

$$u = \begin{bmatrix} u_\alpha \\ du_{ign} \end{bmatrix} \tag{2.11}$$

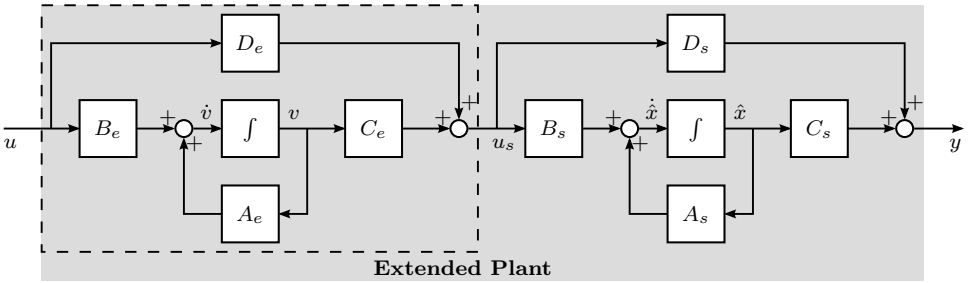

Figure 2.15: The series connection of the extension system and the linearized system yields the extended plant.

In the following steps, the LQG controller is designed for this extended plant.

2. **Feedback Gain**

   In a second step, the state feedback gain K is designed. See Appendix A.3.2 for more information. The two tuning parameters are r_1, the penalization of u_α, and r_2, the penalization of du_{ign}. Define the matrix R as:

$$R = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix} \qquad \text{with} \qquad R = R^T > 0 \tag{2.12}$$

$$r_i \to 0\text{: cheap control} \tag{2.13}$$
$$r_i \to \infty\text{: expensive control} \tag{2.14}$$

   The penalization of the states is given according to the standard definition of *Control Systems II*:

$$Q = C^T \cdot C \qquad \text{with} \qquad Q = Q^T \geq 0 \tag{2.15}$$

   In Matlab the gain K can be calculated with the `lqr` command (type `doc lqr` for further information):

$$K = lqr(A, B, Q, R) \tag{2.16}$$
$$\tag{2.17}$$

3. **Observer**

In a third step, a state observer is built because the LQG approach requires the knowledge of all states within the system. In reality, this is generally not possible without an observer. Figure 2.16 shows the structure of an observer. Only the gain L (observer gain) is unknown since the system matrices A, B and C are the matrices derived in step 1.

An observer is designed by solving a Matrix-Riccati equation. In Matlab, use again the command `lqr` to derive the observer gain L:

$$L^T = lqr(A^T, C^T, B \cdot B^T, q) \tag{2.18}$$

The tuning parameter q can be used the following way:

$q \to 0$: fast observer
$q \to \infty$: slow observer

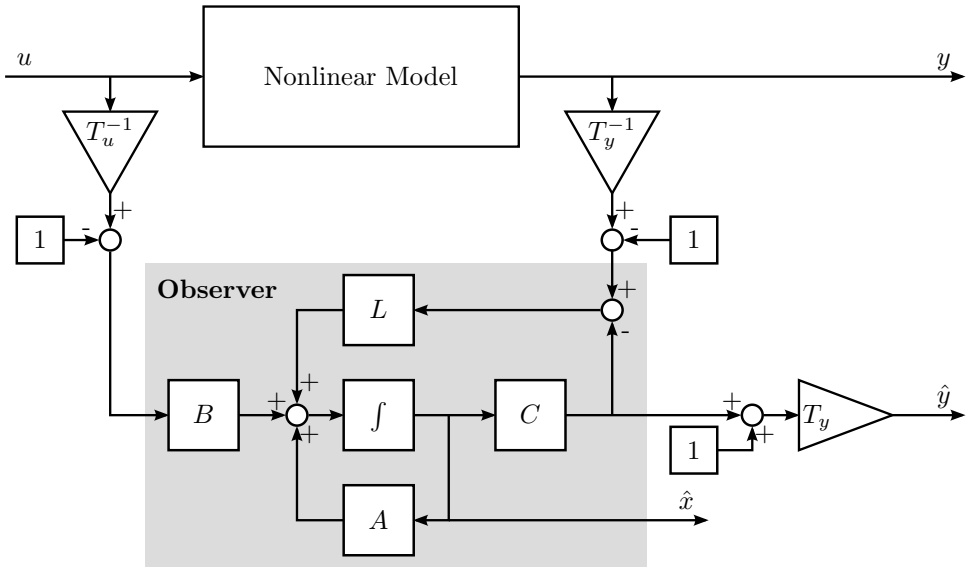

Figure 2.16: Configuration to test the performance of the observer

Tune the parameter q by comparing the observer output $\hat{y}$ to the output of the nonlinear model y with the configuration presented in Fig. 2.16. In addition, the parameters r_1, r_2 and q should be tuned such that the poles of the observer $(A - LC)$ are approximately three times bigger than the poles of the system $(A - BK)$.

4. **Controller matrices**

After obtaining all necessary parameters, the controller matrices $LLDR_{Ac}$, $LLDR_{Bc}$, $LLDR_{Cc}$ and $LLDR_{Dc}$ are derived in a fourth step. Since the system on the test bench is represented by the system using only the matrices A_s, B_s, C_s, D_s (not by the extended system derived in step 1), the LQG controller matrices have to be connected in series with the extension matrices

A_e, B_e, C_e, D_e. See Fig. 2.17. This reults in the the following controller matrices:

$$
\begin{aligned}
LLDR_{Ac} &= \begin{bmatrix} A_e & -B_eK \\ 0 & A - BK - LC \end{bmatrix} \\
LLDR_{Bc} &= \begin{bmatrix} B_eD_c \\ -L \end{bmatrix} \\
LLDR_{Cc} &= \begin{bmatrix} C_e & -D_eK \end{bmatrix} \\
LLDR_{Dc} &= \begin{bmatrix} D_eD_c \end{bmatrix},
\end{aligned}
$$

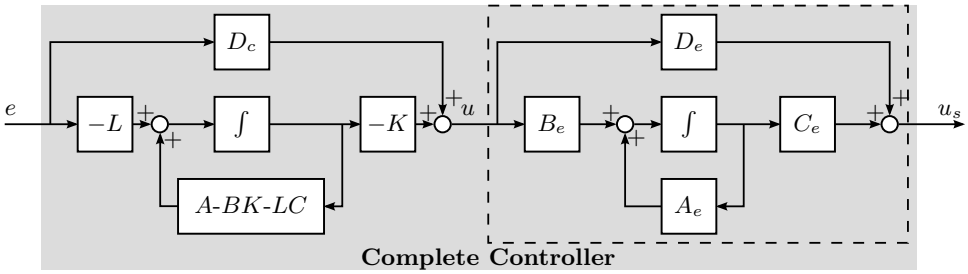

Figure 2.17: Series connection of the LQG controller matrices and the extension matrices in order to get the matrices of the complete controller. The matrix D_c of the LQG controller is by definition a zero matrix with appropriate dimensions.

5. **Discretization**

   In a fifth and final step the continuous controller is discretized. We recommend that the controller is discretized using a Tustin emulation (Matlab command: `c2d`) with a sampling time of $T_s = 0.001s$. The sampling time needs to be fast enough compared to the system dynamics ($T_s < \frac{\pi}{5 \cdot \omega_c}$).

   Resulting matrices: $LLDR_{Ad}$, $LLDR_{Bd}$, $LLDR_{Cd}$ and $LLDR_{Dd}$.

Simulations

The behavior of the discretized controller can be simulated using the derived engine model. Apply steps on the engine speed and on the load to check the stability and robustness of your controller. If you want us to check your controller, send it to one of the teaching assistants.

Hint: Try to implement noise in your simulation, use for example, the signal generator *Band-Limited White Noise* in Simulink.

Validation on the Test bench

Once your controllers work on Matlab/Simulink you have two opportunities to validate the behavior of two controllers on the IDSC test bench in room ML E44. A doodle link will be sent to you in order to organize the test bench schedule.

See the next chapter to get an overview of the required matrices to run your controller on the test bench

Useful Matlab Commands

Some useful commands are: `ssdata, ss, tf, margin, bode, nyquist, lqr, c2d`, (Use the Matlab command `'doc` *`matlab command`*`'` to get detailed information about any command.).

2.5 Milestone 4 Competition

2.5.1 Rules

- The engine speed varies between 70 and 200 rad/s.
- The load is switched on and off randomly.
- The sampling time T_s is 1 ms.
- Two different controllers are allowed.
- The controller with the lowest cost according to the provided cost function wins.

2.5.2 Expected Files for the Competition

To implement your controller on the test bench we need the following discretized matrices of your controller with appropriate dimensions:

- $LLDR\_Ad$ (Dimension: 12×12)
- $LLDR\_Bd$ (Dimension: 12×1)
- $LLDR\_Cd$ (Dimension: 2×12)
- $LLDR\_Dd$ (Dimension: 2×1)

as well as the normalization matrices defined as:

- $LLDR\_Tu = \begin{bmatrix} u_{\alpha,nom} & 0 \\ 0 & du_{ign,nom} \end{bmatrix}$
- $LLDR\_Ty = \omega_{e,nom}$

Save all these matrices in a file named $controller1.mat$ ($controller2.mat$ for an additional controller etc.) and send them to a teaching assistant.

2.5.3 Cost Function

Your controller should minimize a cost function that penalizes both deviations from the reference speed and fast throttle behavior. The specific cost function will be published at some point during this milestone.

Appendix A

Introduction to Modeling and Control

A.1 Introduction

During the last few years we have had to learn that some students lack the knowledge of a few basic concepts of modeling and control theory. This chapter should therefore be understood as a very brief summary of the important concepts that should be known when solving the idle speed control exercise.

In the first part, an introduction to modeling and control will be given while in the second part the concept of the LQR/LQG control approach will be discussed. Obviously, this is not even slightly a comprehensive course on control and for interested students who did not get their bachelors degree at ETH Zürich we recommend the two courses Control Systems I and II to get a better overview.

A.2 Modeling

A.2.1 General Form of First Principles Models

In this text, we will restrict ourselves to nonlinear, continuous-time and time-invariant, systems which describe the vast majority of engineering problems.
First principles models are mathematical models that describe physical principles which are generally stated using differential equations [1](difference equations in the discrete-time case). Any n-th order differential equation can be rewritten as a system of n 1^{st}-order differential equations by introduction of additional physical states. This means that we can write any such systems in the following form:

$$\begin{aligned} \frac{d}{dt}z(t) &= f(z(t), v(t)), \\ w(t) &= g(z(t), v(t)). \end{aligned} \tag{A.1}$$

[1]Finding these equations is the most difficult part of control tasks and the main scope of the engine system lecture. Some remarks on this topics are stated in the appendix of the text book.

The following definition apply:

- z: The *state vector*, a $n \times 1$-vector that contains all of the relevant system states.
- v: The *input vector*, a $m \times 1$-vector that contains all of the possible system inputs.
- w: The *output vector*, a $p \times 1$-vector that contains all of the system outputs.
- f: The *system dynamics*, a set of differential equations that describe the system dynamics according to physical principles.
- g: The *output dynamics*, a system of differential equations that describe the system output behavior according to physical principles.

A.2.2 Normalization

The normalization step makes the model numerically more stable, i.e. it can also be processed by low resolution computer processors. Model normalization can be done in several ways. Here, a setpoint-based approach is followed, i.e., it is assumed that the system usually operates around a constant nominal point such that for all level variables $z_i(t)$ with $i = 1, ..., n$ there is a known and physically meaningful nominal value $z_{i,0} \neq 0$. Reference inputs $v_{j,0}$ with $j = 1, ..., m$ and nominal outputs $w_{k,0}$ with $k = 1, .., p$ are associated to this nominal point z_0. The normalization is usually done around a corresponding equilibrium point for given nominal constant inputs.

The normalization then consists of replacing $z_i(t)$, $v_j(t)$ and $w_k(t)$ by their normalized counterparts $x_i(t)$, $u_j(t)$ and $y_k(t)$ through the transformation

$$x_i(t) = \frac{z_i(t)}{z_{i,0}} \qquad u_j(t) = \frac{v_j(t)}{v_{j,0}} \qquad y_k(t) = \frac{w_k(t)}{w_{k,0}}$$

The new variables $x_i(t)$, $u_j(t)$ and $w_k(t)$ will have no physical units and have magnitude of approximately 1, if the system is close to its expected operating point. The normalization does not change the fundamental systems characteristics (stability, input-output behavior, etc.).

The system after the normalization has the form

$$\begin{aligned} \frac{d}{dt}x(t) &= f_0(x(t), u(t)), \\ y(t) &= g_0(x(t), u(t)). \end{aligned} \tag{A.2}$$

A.2.3 Linearization

A beautiful theory for the control of linear systems has been derived over the years which, for example, makes it possible to analyze the stability properties of a closed-loop control system solely based on its open-loop properties. This is not the case for nonlinear control problems and therefore, in most cases, nonlinear control problems are approximated by a linear system description using a first-order Taylor-series approximation of the nonlinear system around a nominal operating point. This does of course limit the accuracy of the system description to regions around the nominal operating point. However, this is not an issue in most cases since we apply a controller which makes sure that we stay in that region.

The linearization of the nonlinear system using a Taylor-series approximation around the nominal operating point $\bar{x}$ looks as follows:

$$f_0(x) \approx f_0(\bar{x}) + \frac{\partial f_0}{\partial x}|_{x=\bar{x}} \cdot (x - \bar{x}), \text{ where}$$
$$\frac{\partial f_0}{\partial x} = \begin{pmatrix} \frac{\partial f_{0,1}}{\partial x_1} & \frac{\partial f_{0,1}}{\partial x_2} & \cdots & \frac{\partial f_{0,1}}{\partial x_n} \\ \cdots & & & \cdots \\ \frac{\partial f_{0,n}}{\partial x_1} & \frac{\partial f_{0,n}}{\partial x_2} & \cdots & \frac{\partial f_{0,n}}{\partial x_n} \end{pmatrix}$$

Usually, $\bar{x}$ is a stationary operating point x_0 (the same setpoint used for the normalization) that is chosen according to some specifications (e.g. a given engine speed or a given relative engine load). The following equation holds for an equilibrium point.

$$\frac{d}{dt}x_0 = f(x_0, u_0) = 0$$

Only small deviations from the normalized setpoints are analyzed in order to linearize the system, and the following new variables are introduced:

$$\begin{aligned} x(t) &= x_0 + \delta x(t) && \text{with} && |\delta x(t)| \ll 1, \\ u(t) &= u_0 + \delta u(t) && \text{with} && |\delta u(t)| \ll 1, \\ y(t) &= y_0 + \delta y(t) && \text{with} && |\delta y(t)| \ll 1. \end{aligned}$$

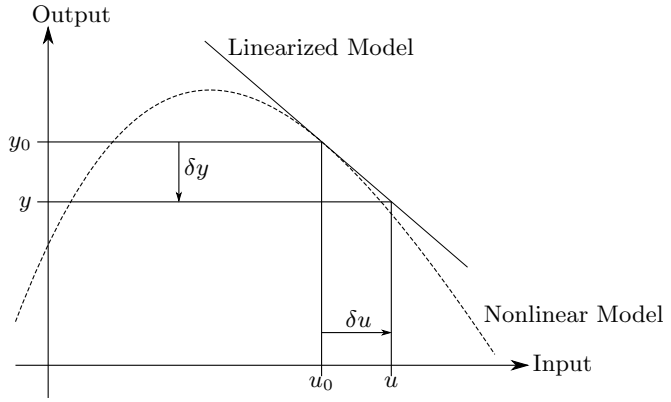

Figure A.1: Linearization of the nonlinear model.

The introduced variables are illustrated in Fig. A.1. It follows that:

$$\frac{d}{dt}x = \underbrace{f(x_0, u_0)}_{=0} + \underbrace{\frac{\partial f_0}{\partial x}|_{x=x_0,u=u_0}}_{=:A} \cdot \underbrace{(x - x_0)}_{=:\delta x} + \underbrace{\frac{\partial f_0}{\partial u}|_{x=x_0,u=u_0}}_{=:B} \cdot \underbrace{(u - u_0)}_{=:\delta u}$$
$$\frac{d}{dt}\delta\dot{x} = A \cdot \delta x + B \cdot \delta u$$
$$y = \underbrace{g_0(x_0, u_0)}_{=y_0} + \underbrace{\frac{\partial g_0}{\partial x}|_{x=x_0,u=u_0}}_{=:C} \cdot \underbrace{(x - x_0)}_{=:\delta x} + \underbrace{\frac{\partial g_0}{\partial u}|_{x=x_0,u=u_0}}_{=:D} \cdot \underbrace{(u - u_0)}_{=:\delta u}$$
$$\delta y = C \cdot \delta x + D \cdot \delta u$$

If we only consider deviations from the nominal (equilibrium) values and redefine the following variables[2]:

$$\begin{aligned} x &:= \delta x = x - x_0, \\ u &:= \delta u = u - u_0, \\ y &:= \delta y = y - y_0. \end{aligned}$$

we get the LTI (linear, time-invariante) state-space description:

$$\begin{aligned} \frac{d}{dt}x &= Ax + Bu, \\ y &= Cx + Du. \end{aligned} \tag{A.3}$$

A is the *system matrix*, B the *input matrix*, C the *output matrix* and D the *throughput matrix* (which is often equal to zero in physical systems).

A.2.4 Consequences for inputs/outputs of the System

If the normalization and linearization are performed with the same nominal values u_0 for the input and y_0 for the output, the dimensionless quantities δu and δy are connected to the physical values u and y in the following way:

$$\delta u = \frac{u - u_0}{u_0}, \qquad \delta y = \frac{y - y_0}{y_0} \tag{A.4}$$

A.2.5 A Simple Example: The Pendulum

Here, a small example will be given for a better understanding of the previous paragraphs. The normalization is not performed since it is not required in this case.

The simple system as illustrated in Fig. A.2 consists of a pendulum that is hung from a ceiling. It has a mass m that is concentrated at its lower end and is connected to the ceiling with a weightless rod of length l. A torque T_c can be applied around the point where it is attached to the ceiling which will deflect the pendulum. The angle from a virtual vertical line is defined as θ.

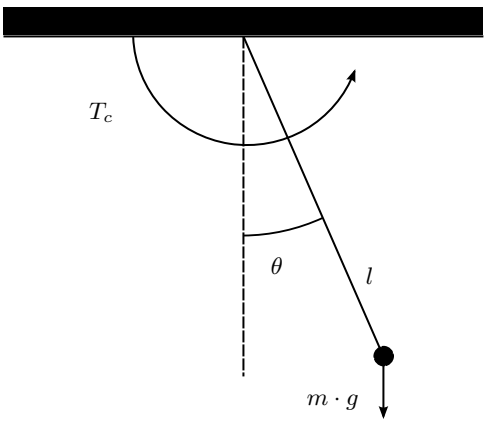

Figure A.2: pendulum

[2]Note that the new variables are the differences between the calculated values and the nominal values. This will be of importance for the model and controller implementation.

Using Newton's second law: $\ddot{\phi} \cdot J = T^+ - T^-$ and extracting the following quantities:

$\ddot{\phi} = \ddot{\theta}$	The second derivative with respect to time of the deflection angle.
$J = ml^2$	The moment of inertia of the pendulum with respect to the rotational axis.
$T^+ = T_c$	The torque applied to the pendulum in positive θ-direction.
$T^- = mgl\sin(\theta)$	The torque applied to the pendulum in negative θ-direction.

the dynamics of this system can be stated as:

$$ml^2\ddot{\theta} = T_c - mgl\sin(\theta). \tag{A.5}$$

This is a 2^{nd}-order differential equation which can be formulated as a system of 1^{st}-order differential equations as follows:
First, we define the following system states and inputs, respectively:

$x_1 := \theta$	The first state of the system.
$x_2 := \dot{\theta}$	The second state of the system.
$u := \frac{T_c}{ml^2}$	The input of the system.
$y := \theta$	The output of the system (this means we measure the first state only).

The following system description, according to Eq. (A.1) results:

$$\dot{x} = \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) + u \end{pmatrix} = f(x,u) \tag{A.6}$$

$$y = x_1 = g(x,u) \tag{A.7}$$

For the linearization, we choose the nominal point to be $x_s = \left(\frac{\pi}{4}, 0\right)$ and therefore $u_s = \frac{g}{l}\sin(\frac{\pi}{4})$. The matrices of the systems result as:

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} \cdot cos(\frac{\pi}{4}) & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = 0$$

A.2.6 Frequency Domain

In order to analyze a system, the frequency domain is a helpful tool. If a linear time-invariant system is fed with a sinusoidal input signal $u(t) = \cos(\omega t)$, the system will respond (after a transient part) with a steady-state part that has the same frequency ω but another amplitude and phase lag. The response is of the form $y(t) = A(\omega) \cdot \cos\left(\omega \cdot t + \varphi(\omega)\right)$. The so-called **Bode plot** can then be used to plot all $A(\omega)$ and $\varphi(\omega)$ over the frequency ω. Bode diagrams are usually plotted with a logarithmic scale in the amplitude and a linear scale in the phase, such that a series connection can be visualized by simply adding the Bode diagrams of the two systems. Often the phase is given in degrees and the amplitude in dB ($A_{dB} = 20 \cdot log_{10}$). A related tool is the **Nyquist plot** where the phase and the amplitude is plotted within one figure.

Another way to get a frequency domain representation is the **Laplace Transformation**, where $\Sigma(s) = \frac{Y(s)}{U(s)} = c(sI - A)^{-1}b$ is the transfer function from the input u to the output y.
Check the Matlab commands `bode` and `nyquist` for frequency domain analysis as well as the command `step` for time domain analysis. The laplace transformation can be performed with the command `tf(ss(A,b,c,d))`

A.3 Control

This section will be about designing an LQR/LQG controller for a linear system of the form shown in Eq. (A.3), preferably normalized as shown in Appendix A.2.2.

First, some general terms that are used in control theory will be described using the structure of a control system shown in Fig. A.3.

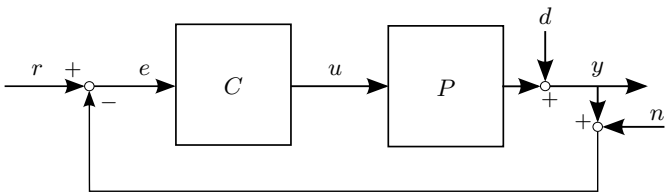

Figure A.3: Structure of a simple feedback control system.

- P is the system we want to control (in our case, it is the linearized and normalized version).
- C is the controller we want to design.
- r is the reference signal which we want our system to follow.
- u is the output of the controller.
- d is one or many disturbance(s).
- y is the real output of the system.
- n is noise.
- e is the difference between the measured system output and the reference signal and, therefore, the error the controller should minimize.

A.3.1 Important Definitions

- The **loop gain (or open loop transfer function)** $L(s) = P(s) \cdot C(s)$ is the series interconnection of the controller and the controlled system (open loop) in the frequency domain. The Bode diagram allows us to analyze the closed loop behavior (nonlinear) by plotting the open loop transfer function (linear) as explained in some of the next definitions.

- The **sensitivity** $S(s) = \frac{1}{1+L(s)}$ is the closed loop transfer function from the disturbance d to the system output y and, therefore, used to analyze the influence of disturbances on the system's output. Generally, $S(s)$ should have a low amplitude ($-20dB$ or less) at low frequencies as disturbances are low frequency signals.

- The **complementary sensitivity** $T = \frac{L(s)}{1+L(s)}$ is the closed loop transfer function from the reference value r to the system output y. And can be used to analyze the step response behavior. Generally, $T(s)$ should have a low pass behavior ($-20dB/dec$) for frequencies higher than the crossover frequency (see next definition) in order to guarantee noise suppression.

- The **crossover frequency** ω_c can be seen in a Bode plot as the frequency at which the open loop system $L(s)$ crosses the $0\,dB$ line (or in the Nyquist diagram: the frequency at which the system crosses the unit circle for the first

time).
A rule of thumb is $t_{90} \approx \frac{1.7}{\omega_c}$, where t_{90} is defined as the time when the controlled system has reached 90% of the value of a reference step input in the time domain and hence ω_c is a helpful tool to analyze the speed of the closed loop system.

- The **phase margin** (PM or φ) is defined as the angle at which the open-loop transfer function $L(s)$ crosses the unit circle (in the Nyquist diagram). Hence, it is also the difference of the phase at the crossover frequency to $-180°$ in the Bode diagram. The most important rule is that if the PM of the open loop system is negative, the closed loop system will be unstable.
A rule of thumb is $\hat{\epsilon} \approx \frac{71° - \varphi}{117°}$, where $\hat{\epsilon}$ is the maximal overshoot to a step response in the time domain.

- The **steady state error** e_∞ is defined as the difference between the desired and the obtained value when the system is at steady state (most often, this is tested by applying a step to the input). The steady state error is zero if the closed loop system has an integrating behavior, i.e. it integrates the error e and applies an appropriate control action until e is zero. The complementary sensitivity $T(s)$ of such a control system has an infinite magnitude at $omega = 0$. The magnitude in the Bode plot of $T(s)$ therefore, tends to infinity at low frequencies.

- The **minimum return difference** $\mu = \min_\omega |1 + L(j\omega)| = \frac{1}{\max_\omega |S(j\omega)|}$ is the minimal distance from the point $(-1, 0j)$ in the complex plane, which is also called the 'horror' point. The closer the system gets to this horror point the higher is the possibility that the system gets unstable in case of modeling errors. (When we have $L(s) = -1$, the denominator of the closed loop transfer function $T(s)$ is zero and, therefore, the system output will be undefined (or ∞)).

- The **Nyquist theorem** says that the closed loop system $T(s)$ is stable if and only if $\gamma = \rho + \sigma/2$ holds, where γ is the number of encirclements of $L(s)$ around the -1 point (counter clockwise if $\omega = [-\infty, \infty]$). Further, ρ is the number of poles with positive real part and σ the number of poles with real part equal to zero (poles of the open loop transfer function $L(s)$).

- The system is **controllable** if

  $\text{rank}(R_n) = n$, with $R_n = \begin{bmatrix} B & AB & A^2B & .. & A^{n-1}B \end{bmatrix}$.

- The system is **observable** if

  $$\text{rank}(O_n) = n, \text{ with } O_n = \begin{bmatrix} C \\ CA \\ CA^2 \\ ... \end{bmatrix}.$$

A.3.2 Standard LQR Problem

The LQG method is based on a Matrix-Riccati equation (Eq. (A.8a)) and is optimal in terms of the cost function defined in Eq. (A.8b). The feedback part is assumed to depend on the states only: $u(t) = -K \cdot x(t)$. For further details, see the control systems II script.

$$0 = \Phi \cdot \widehat{B} \cdot R^{-1} \cdot \widehat{B}^T \cdot \Phi - \Phi \cdot \widehat{A} - \widehat{A}^T \cdot \Phi - Q \tag{A.8a}$$

$$J(t) = \int_0^\infty (x^T \cdot Q \cdot x + u^T \cdot R \cdot u)\, dt \tag{A.8b}$$

$$K = R^{-1} \cdot \widehat{B}^T \cdot \Phi = lqr(\widehat{A}, \widehat{B}, Q, R) \tag{A.8c}$$

$R = R^T > 0$ can be defined as $R = r \cdot I_{mxm}$ and $Q = Q^T = \widehat{C}^T \cdot \widehat{C} \geq 0$ is often used. x and u are the states and inputs known from the equation $\dot{x} = \widehat{A} \cdot x + \widehat{B} \cdot u$. However, one could choose totally different R and Q matrices (as long as $R = R^T > 0$ and $Q = Q^T \geq 0$ hold). For example, different r values can be chosen for every control input. Similarly, Q can be chosen to penalize other states than $Q = Q^T = \widehat{C}^T \cdot \widehat{C}$ would. However, it makes sense to penalize the state of interest and, therefore, use this relationship.

Some aspects of the LQR controller are worth mentioning:

- The LQR controller is linear in the state variable x.
- K is time-invariant (only true for infinite-horizon LQR problems).
- The resulting closed-loop system matrix $A - B \cdot K$ is a Hurwitz matrix, i.e., all eigenvalues of $A - B \cdot K$ have negative real parts.
- The return difference μ is always greater or equal to 1 and, therefore, the phase margin is guaranteed to be greater than 60°.
- The controller K is the best resulting out of the optimization problem, but not the 'best' possible one.
- In the idle speed exercise, the controller has to be extended with an integrating part in order to have no steady state error. More on that is in Section 2.4.3 as it is of importance for the exercise. The integral action also helps us to implement the reference signal as the standard LQR system only tries to bring the system back to the equilibrium point in case of any disturbance.

A.3.3 Observer or LQG Approach

If the states x are unknown (e.g. not measured), an observer has to be used to estimate them. This is done by an other Riccati Equation (called the dual approach):

$$0 = \psi \cdot \widehat{C}^T \cdot q^{-1} \cdot \widehat{C} \cdot \psi - \psi \cdot \widehat{A}^T - \widehat{A} \cdot \psi - \widehat{B} \cdot \widehat{B}^T \tag{A.9a}$$

$$L^T = q^{-1} \cdot \widehat{C} \cdot \psi = lqr(\widehat{A}^T, \widehat{C}^T, \widehat{B} \cdot \widehat{B}^T, q) \tag{A.9b}$$

Comparing this Riccati Equation with the one before, it is easy to see the analogy between those two:

$$\begin{aligned} \widehat{A} &\rightarrow \widehat{A}^T \\ \widehat{B} &\rightarrow \widehat{C}^T \\ \widehat{Q} = \widehat{C}^T \cdot \widehat{C} &\rightarrow \widehat{B} \cdot \widehat{B}^T \end{aligned}$$

The system A,B,C has to be controllable (A, B) and observable (A, C) in order to guarantee a solution. The LQR approach (only state feedback) with the dynamic matrix $A - B \cdot K$ is asymptotically stable and robust (phase margin $\geq 60°$). Introducing an observer still results in a stable system, but the minimal distance to the point -1 in the Nyquist diagram and the phase margin are not as good as with an LQR approach. However, the observer must be used to have access to the states. Therefore, a detailed analysis is necessary in order to make sure that your observer is robust enough. There are two options to design your observer (Both of them need a robustness analysis!):

- Design the poles of your observer $(A - L \cdot C)$ to be three times faster than the one of the system $(A - B \cdot K)$
- Check whether your system reacts fast enough to a disturbance step input.

Appendix B

Least Squares Problem

Whenever possible, a parameter identification problem should be formulated such that it is linear in the parameters (lip). If this is the case, it can be solved as a system of linear equations as presented in Eq. (B.1).

B.1 Solution of a linear Least Squares Problem

The following algorithm can be used to provide the solution that minimizes the 2-norm of the residual $\vec{r}$.

$$M\vec{x} = \vec{y} + \vec{r} \tag{B.1}$$

The variables in the equation above have the following dimensions: $M \in \mathbb{R}_{m\times n}$, $\vec{x} \in \mathbb{R}_{1\times n}$, $\vec{y} \in \mathbb{R}_{m\times 1}$ and $\vec{r} \in \mathbb{R}_{m\times 1}$, where $m >> n$, as m is the number of samples in one measurement (in our case in the order of 10^5) and n the number of parameters to be identified. M is the so-called *regression matrix*, and $\vec{r}$ the so-called *residual* that comes into play because the equation system is over-determined. With the following steps, the 2-norm of the residual $\vec{r}$ can be minimized:

$$||\vec{r}||^2 = ||M\vec{x} - \vec{y}|| = (M\vec{x} - \vec{y})^T \cdot (M\vec{x} - \vec{y}) \tag{B.2}$$

The 2-norm of the residual $\vec{r}$ is minimized if the derivative with respect to $\vec{x}$ is zero. Thus, using the rules presented below($A, B \in \mathbb{R}_{m\times n}$, $\vec{z} \in \mathbb{R}_{m\times 1}$), Eq. (B.3) is obtained.

$$(A+B)^T = A^T + B^T \qquad (A\cdot B)^T = B^T \cdot A^T$$

$$\frac{d}{dz}(\vec{z}^T \cdot A) = A \qquad \frac{d}{dz}(A \cdot \vec{z}) = A^T$$

$$\vec{x} = \left(M^T M\right)^{-1} \cdot M^T \vec{y} \tag{B.3}$$

When static parameters are identified using quasistatic measurements, this approach is very promising since it is accurate and does not require a high computational effort. However, reasonable results can only be obtained if the condition of the regression matrix is low and therefore in most cases, a normalization step should be done preliminarily (see Appendix B.2 for a further discussion). In Matlab, the command *lsqr* solves this type of problem.

Remark: Several numerically more accurate approaches to solve an over-determined system of linear equations exist (for example the QR method) but they do generally require a better knowledge of algebra and more computational effort.

B.2 The Condition Number of a Matrix

Below, an expression for the condition of a square matrix is derived[1]. It will be shown how unavoidable numerical errors can lead to substantial errors when solving a system of linear equations as presented in Eq. (B.4) on a computer.

$$A\vec{z} = \vec{b}, \qquad \text{where } A \in n \times n, \text{ regular; } \vec{b} \neq 0 \tag{B.4}$$

The linear equation system has the solution

$$\vec{z} = A^{-1}b \neq 0. \tag{B.5}$$

We assume A to be exact and $\vec{b}$ to be inaccurate (this is always the case because a computer system can only be accurate up to the machine epsilon *eps* which in common 32 BIT systems is approximately $6 \cdot 10^{-8}$; additionally, the measurement data is inherently approximative), i.e. instead of $\vec{b}$, the calculation is done with $\hat{\vec{b}}$, leading to an absolute error of $\Delta\vec{b} := \hat{\vec{b}} - \vec{b}$.
More important than the absolute error is of course the relative error

$$\delta\vec{b} := \frac{1}{||\vec{b}||}\Delta\vec{b}. \tag{B.6}$$

Thus, the equation system $A\hat{\vec{z}} = \hat{\vec{b}}$ has the (exact) solution $\hat{\vec{z}} = A^{-1}\hat{\vec{b}}$. The interesting question is: How does the error in $\vec{b}$ influence the relative accuracy error of the result $\delta\vec{z} := \frac{1}{||\vec{z}||}\Delta\vec{z}$? The following answer can be found:

$$\begin{aligned}
\Delta\vec{z} &= \hat{\vec{z}} - \vec{z} = A^{-1}\hat{\vec{b}} - A^{-1}\vec{b} = A^{-1}(\hat{\vec{b}} - \vec{b}) = A^{-1}\Delta\vec{b} \\
||\delta\vec{z}|| &= \frac{||\vec{b}|| \cdot ||A^{-1}\Delta\vec{b}|| \cdot ||\Delta\vec{b}||}{||\vec{z}|| \cdot ||\Delta\vec{b}|| \cdot ||\vec{b}||} \overset{\vec{b}=A\vec{z}}{=} \underbrace{\frac{||A\vec{z}||}{||\vec{z}||}}_{\leq ||A||} \cdot \underbrace{\frac{||A^{-1}\Delta\vec{b}||}{||\Delta\vec{b}||}}_{\leq ||A^{-1}||} \cdot ||\delta\vec{b}|| \\
||\delta\vec{z}|| &\leq ||A|| \cdot ||A^{-1}|| \cdot ||\delta\vec{b}||
\end{aligned}$$

Definition:
According to the inequality above, the term $\kappa(A) := ||A|| \cdot ||A^{-1}||$ is defined as the *condition of the matrix A*.
The following inequality holds: $1 = ||I_n|| = ||A \cdot A^{-1}|| \leq ||A|| \cdot ||A^{-1}|| = \kappa(A)$
Conclusion:
The error in $\vec{b}$ can be amplified by the factor $\kappa(A)$ in the solution.

If also A is not exact ($\hat{A}\hat{\vec{z}} = \hat{\vec{b}}$), one can show that

$$||\delta\vec{z}|| \leq \frac{\kappa(A)}{1 - \kappa(A)||\delta A||} \cdot (||\delta A|| + ||\delta\vec{b}||).$$

This theory can be applied to a least squares problem as discussed in Appendix B.1: Eq. (B.3) can be rewritten in the form of Eq. (B.4), with the following relations

$$A = M^T M,$$
$$\vec{b} = M^T\vec{y}.$$

Keeping in mind that $\kappa(M^T M) = \kappa(M)^2$, it becomes clear why least squares problems should be performed with normalized data values in any case.

[1]Source: Numerische Mathematik, Kaspar Nipp, Seminar für angewandte Mathematik ETH Zürich, Skript zur Vorlesung am Departement Maschinenbau und Verfahrenstechnik