University of Stavanger

Faculty of Science and Technology

BACHELOR`S THESIS

Study program/Specialization: Control Engineering and Circuit Design - Bachelor's Degree Programme	Spring semester, 2021 Open
Writers: Gent Luta John Håvard Aarvåg	Gent Luta…………………… John Håvard Aarvåg……… (Writer's signature)
Faculty supervisor: Damiano Rotondo	
Thesis title: *Study on the applicability of the Carleman embedding to the control of the water level in a tank.*	
Credits (ECTS): 20	
Key words: Control theory Modelling Simulation Programming	Pages: 80…………… + enclosure: 84…… Stavanger, 15/05-2021 Date/year

Acknowledgements

We would like to express our utmost gratitude to our faculty supervisor Assoc.Prof. Damiano Rotondo for his guidance throughout this project. He has been incredibly helpful, always available and very patient with us. We are very lucky to have had such a good supervisor.

Unë do të doja të shprehja mirënjohjen time të sinqertë për familjen time për dashurinë dhe mbështetjen e tyre të pakushtëzuar. Jam i mirënjohës motrës time qe ishte gjithmonë aty për mua dhe më brohoriti. Jam i mirënjohës vëllait tim qe më dha mësime të vlefshme për jetë. Jam i mirënjohës prindërve të mi qe më dhanë mua dhe vëllaut dhe motres time një jetë të mirë, edhe pse kjo nënkuptonte sakrifica të panumërta. Unë do të jem përgjithmonë mirënjohës.

- Gent Luta

Jeg ønsker å dedikere denne oppgaven til mine nieser og nevøer. Takk til alle mine nærmeste for enorm støtte. En spesiell takk til Ida Marita og Mons, for gode arbeidsforhold under arbeidet med oppgaven.

- John Håvard Aarvåg

Abstract

This thesis presents control design of an approximated dynamical model, which is derived by using the Carleman embedding technique. To perform the Carleman embedding, the nonlinear dynamical model should be expressed in a polynomial form. By using the higher order Taylor approximation, a nonlinear system, if analytical, can be expressed in such a form. Carleman embedding of the *cubic* two-tank model proves to give a compromise between accuracy and computational labour. However, Carleman linearization of this model results in an uncontrollable system. Therefore, controller design for the quadratic Carleman approximation is considered. This controller is a state feedback controller. The process of finding the controller gain $\mathcal{K}$, can be expressed as an optimization problem in terms of *linear matrix inequalities.* A quadratic controller has to fulfil some necessary constraints to be operational. The main benefit of this type of controller design, is that it ensures stability in the given region that the controller was designed for. However, this controller design leads to poor feasibility, which limits its usefulness in a practical setting.

Sammendrag

Denne avhandlingen presenterer kontroller design av en estimert dynamisk modell, hvor modellen er utledet ved bruk av teknikken *Carleman embedding.* For å utføre denne teknikken, burde den ikkelineære dynamiske modellen være uttrykt på polynom form. Hvis et ikkelineært system er analytisk, kan det uttrykkes som et polynom ved hjelp av Taylor approksimasjon av høyere orden. Taylor approksimasjonen av orden tre, også kalt *cubic* modell, viser seg å være den best egnede modellen for utføringen av Carleman embedding teknikken. *Carleman linearisering* av denne cubic modellen resulterer i et ukontrollerbart system. Derfor vil kontroller design for *quadratic Carleman approksimasjonen* bli vurdert. Kontrolleren for dette systemet er basert på tilbakekobling. Prosessen i å finne kontroller forsterkningen $\mathcal{K}$ kan utrykkes som et optimaliseringsproblem i form av *linear matrix inequalities.* En slik kontroller er nødt til å tilfredsstille visse kriterier for å være operasjonell. Hovedfordelen i en slik kontroller er at den garanterer stabilitet i den regionen som kontrolleren ble designet for. Ulempen med denne typen kontroller design er at numeriske problemer oppstår, som begrenser dens praktiske nytte.

Table of Contents

List of Figures . . . VI

List of Tables . . . VII

I Introduction **1**

1 Motivation **2**

1.1 Introduction . . . 2

1.2 The history of Carleman embedding . . . 3

1.3 Objective . . . 3

2 The two-tank system **5**

2.1 Description . . . 5

2.2 Nonlinear model of tank 1 . . . 9

2.3 Nonlinear model of tank 2 . . . 10

II Modelling using Taylor and Carleman approximations **14**

3 Taylor approximations **15**

3.1 Taylor series . . . 15

3.1.1 Taylor series in several variables . . . 16

3.2 Taylor models . . . 17

3.2.1 Linear model . . . 17

3.2.2 Quadratic model . . . 19

3.2.3 Partially quadratic models . . . 20

3.2.4 Cubic model . . . 21

3.2.5 Higher order models . . . 22

3.3 Taylor model comparison . . . 23

4 Modelling using the Carleman embedding 33
4.1 Carleman embedding . . . 33
4.1.1 Carleman embedding technique . . . 34
4.1.2 Truncation . . . 34
4.2 Carleman approximation of the cubic tank model . . . 36
4.3 Carleman approximation comparison . . . 38

III Control 43

5 State feedback control using the linear Carleman approximation 44
5.1 Control . . . 44
5.2 Controllability of the linear Carleman approximation . . . 46

6 State feedback control using the quadratic Carleman approximation 48
6.1 Supplemental theory . . . 48
6.1.1 Polytopes . . . 48
6.1.2 Lyapunov stability criterion . . . 50
6.1.3 State feedback control of nonlinear quadratic systems . . . 52
6.2 Controller for the quadratic Carleman approximation . . . 54
6.3 Controller for the quadratic Carleman approximation using two input variables . . . 58

7 Experimental results 62
7.1 Application of the quadratic controller on to the two-tank system . . . 62
7.2 Analysis of the experimental results . . . 63

IV Conclusions and future work 65

8 Conclusions 66

9 Future work 67

Bibliography 69

V Appendices 71

A Taylor model comparison for Scenario 2 72

B Carleman approximation comparison for Scenario 2 77

C Controller of the quadratic Carleman approximation for Scenario 2 80

D Experimental results for Scenario 2 85

E Simulation of the controller designed with a reduced polytope 87

F MATLAB code and Simulink schemes 89

F.1 *totank_main.m* . . . 89
F.2 *Carleman_lin_ss.m* . . . 112
F.3 *Carleman_Linearized_func.m* . . . 116
F.4 *Carleman_NonLinearized_func_2Var.m* . . . 117
F.5 *Carleman_NonLinearized_func.m* . . . 118
F.6 *carleman_statespace_2Var.m* . . . 118
F.7 *carleman_statespace.m* . . . 119
F.8 *example_file_solver.m* . . . 119
F.9 *Quadratic_controller.m* . . . 119
F.10 *Quadratic_system.m* . . . 120
F.11 *Polytope_figures.m* . . . 121
F.12 *vert2con.m* [11] . . . 121
F.13 *EKSEMPEL_CARLEMAN.m* . . . 122
F.14 Simulink schemes . . . 124

List of Figures

2.1 This is a schematic sketch of the two-tank processing plant [7]. 6

2.2 Simplified version of the schematic sketch with limited functionality [7]. 7

2.3 The relation between the control signal and the relative flow through the valve with respect to the maximum capacity [7]. 8

2.4 The relation between the control signal and the flow through the pump [7]. 9

2.5 Schematic sketch of tank 2 for area calculation [7]. 11

2.6 Triangle similarity. 12

3.1 Calculating $\frac{\delta f_1}{\delta u_{LV001}}$ in Simulink. 18

3.2 3-D plot of the nonlinear model. 25

3.3 Level curve representing the operating points of the nonlinear model. 25

3.4 Comparisons between the nonlinear and the Taylor approximation models. 26

3.5 Contour plots of the Taylor models and the nonlinear model, where ζ is given by (3.32). 28

3.6 Color maps comparing the different Taylor models. 29

3.7 Model comparison in Simulink. 31

4.1 Carleman approximations of (4.4) compared to the nonlinear system. 35

4.2 Utilising the State-Space block in Simulink. 38

4.3 Step response with an increment in $u_{LV001}(t)$ of 0.02 at 10 seconds. 39

4.4 $\delta = 0.05m$ gives a starting point of 0.3m. 40

4.5 $\delta = 0.2m$ gives a initial condition of 0.45m. 41

4.6 Simulation of the quadratic Carleman approximations. 42

4.7 Simulation of the quadratic Carleman approximations with a larger step. 42

5.1 Feedback control system [1]. 44

6.1 Polytope $\mathcal{P}$ is contained by the blue half-spaces, and represented by the green region. 50

6.2 Polytope for the $2nd$ order quadratic Carleman approximation. 55
6.3 Simulation of Eq. (6.29), with initial condition equal to $0.05m$. 57
6.4 Signal $\delta u_{LV001}(t)$, calculated by $\mathcal{K}z(t)$. 57
6.5 $4th$ order quadratic Carleman approximation with two actuator. 59
6.6 Signal $u(t)$, calculated by $\mathcal{K}z(t)$. 59
6.7 $4th$ order quadratic Carleman approximation with two actuator and implemented convergence rate. 60
6.8 Signal $u(t)$, calculated by $\mathcal{K}z(t)$ with implemented convergence rate. 61

7.1 Experimental results from the quadratic controller. 63
7.2 Experimental results from the quadratic controller. 64

9.1 Modified polytope. 67

A.1 3-D plots of the Taylor models and the nonlinear model as reference. 73
A.2 Contour plots of the Taylor models and the nonlinear model, where ζ is given by (3.32). 74
A.3 Color maps comparing the different Taylor models. 75
A.4 Model comparison in Simulink. 76

B.1 Step response with an increment in $u_{LV001}(t)$ of 0.02 at 10 seconds. 77
B.2 $\delta = 0.05m$ gives a starting point of 0.8m. 78
B.3 $\delta = 0.2m$ gives a starting point of 0.95m. 78
B.4 Simulation of the quadratic Carleman approximations. 79
B.5 Simulation of the quadratic Carleman approximations with a larger step. 79

C.1 Simulation of Eq. (6.29), with initial condition equal to $0.05m$. 80
C.2 Signal $\delta u_{LV001}(t)$, calculated by $\mathcal{K}z(t)$. 81
C.3 $4th$ order quadratic Carleman approximation with two actuator. 82
C.4 Signal $u(t)$, calculated by $\mathcal{K}z(t)$. 82
C.5 $4th$ order quadratic Carleman approximation with two actuator and implemented convergence rate. 83
C.6 Signal $u(t)$, calculated by $\mathcal{K}z(t)$ with implemented convergence rate. 84

D.1 Experimental results from the quadratic controller. 85
D.2 Experimental results from the quadratic controller. 86

E.1 Simulation of the state $z_1(t)$ for the reduced polytope. 87

E.2 Simulation of the input variables $\delta u_{LV001}(t)$ and $\delta u_{PA001}(t)$ for the reduced polytope. 88

List of Tables

2.1 System instrumentation. 5

2.2 Variables regarding tank 1. 10

2.3 Variables regarding tank 2. 10

2.4 The dimensions of tank 2. 12

3.1 Table presenting the $2nd$ order partial derivatives of the nonlinear function. 19

3.2 Table presenting the $3rd$ order partial derivatives of the nonlinear function. 21

3.3 Operating values for *Scenario 1* and *2*. 23

3.4 Labels for the different values of ψ. 27

3.5 Definition of not applicable points. 27

3.6 Numerical interpretation of the area within the hollow rectangles in Figure 3.6. . . . 30

3.7 Integral performance criteria. 32

4.1 Integral performance criteria of the linearized Carleman approximations, with $\delta = 0.05m$. 40

4.2 Integral performance criteria of the linearized Carleman approximations, with $\delta = 0.2m$. 41

7.1 Experiment procedure. 63

A.1 Numerical interpretation of the area within the hollow rectangles in Figure A.3 . . . 75

A.2 Integral performance criteria. 76

Part I

Introduction

Chapter 1

Motivation

1.1 Introduction

A mathematical model for a system is the equation or the equations that describes the system's behaviour. Based on a mathematical model for a given system, it becomes possible to predict how the system will behave at a given state. A mathematical model that describes the system's *dynamics* is the basis for development of simulations, analysis of the system's stability properties, and the design of processes such as signal filtering and regulatory systems.

Unfortunately, a mathematical model, though very useful, can never describe a physical system with absolute certainty. There will always be aspects of a physical system which cannot be modelled. However, a model that only describes parts of the physical system is still useful, if those parts are the dominant parts of the system's dynamical properties [10].

If a physical system is to be described as accurately as possible by mathematical equations, it is highly probable that one or more of these equations will be nonlinear (differential) equations. Linearizing these nonlinear (differential) equations is a key concept in engineering. This is often done by performing the Taylor linearization on the nonlinear equations.

Transforming a nonlinear system into a linear system enables the applications of simple yet effective design techniques for fulfilling a wide number of tasks. One of these tasks includes controlling the system about an equilibrium point. It is also known, and often stressed, that the quality of the approximation fades the more the nonlinear system steers away from the equilibrium point. The reason for this is that the higher order terms in the Taylor series stop being negligible.

Including higher order terms in the approximated model would provide more precision, in the sense of achieving a better approximation of the nonlinear dynamics. However, this increases the complexity of the control design approach, i.e. Laplace transform and frequency response stop being useful tools. Hence, whether there is any practical advantage in using a nonlinear model, although less nonlinear than the original model, is dubious. The advantages, if any, would probably depend on the system under consideration and the nonlinearities therein included.

A way of mimicking the higher order terms, while still keeping linearity, is to utilize the *Carleman linearization*. This may allow for linear design approaches for the controller design of the system including higher order terms. Whether this method of controller design proves advantageous or not, will be explored in this thesis.

1.2 The history of Carleman embedding

In 1932 Torsten Carleman showed that a finite dimensional system of nonlinear differential equations can be embedded into an infinite system of linear differential equations, an embedding technique known as Carleman embedding. Since then it has been applied to several fields in the last 89 years. The most notable works are mentioned:

- In **1963 Bellman and Richardson** [6] discussed the use of Carleman embedding for approximate solutions of nonlinear differential equations.
- In **1980 Steeb and Wilhelm** [18] explored approximate solutions of Lotka-Volterra models using Carleman embedding.
- In **1989 Steeb** [17] discussed the correspondence between solutions of nonlinear systems and the infinite linear system resulting from the Carleman embedding.
- In **1991 Steeb and Kowalski** [12] wrote the book *Nonlinear Dynamical Systems And Carleman Linearization*, discussing several methods of using the Carleman embedding on nonlinear systems.
- In **2008 Mozyrska and Bartosiewicz** [14] talked about the Carleman embedding of linearly observable polynomial systems.
- In **2009 Rauh and Minisini and Aschemann** [16] discussed the use of Carleman embedding in *Carleman Linearization for Control and for State and Disturbance Estimation of Nonlinear Dynamical Processes.*
- In **2019 Amini, Sun and Motee** [5] discussed another approach for optimizing a controller for a nonlinear system approximated with use of the truncated Carleman embedding.

1.3 Objective

The main objectives in this thesis can be segmented into the following parts:

- Develop a mathematical model that describes the two-tank system's dynamics.
- Develop multiple Taylor approximations of the mathematical model. The Taylor approximations differ in the sense of including a different amount of higher order terms.
- Compare the different Taylor approximations, and select the model that is best suited for the Carleman embedding.
- Perform the Carleman embedding on the best suited Taylor approximation, which results in a *infinite* dimensional nonlinear system. Approximating this system as a *finite* dimensional nonlinear system is deemed as the quadratic Carleman approximation.
- Linearize the quadratic Carleman approximation, resulting in the linearized Carleman approximation.
- Design a controller for the linearized Carleman approximation.
- Design a controller for the quadratic Carleman approximation.

The above-mentioned objectives will be reached by using the model of the two-tank process available in the room KE E-459, and some results will be obtained using extensive simulations in MATLAB and, possibly, experimental validation.

The *learning objectives* related to this thesis are:

- Learn how to perform the Carleman linearization of a nonlinear system.
- Get a first contact with design approaches for nonlinear systems.
- Learn how available toolboxes/solvers for linear matrix inequalities can be applied to control problems.

Chapter 2

The two-tank system

The theoretical results described in this report are applied to the two-tank process plant located at the University of Stavanger in the room KE E-459. In Section 2.1, the two-tank processing plant is described in detail. Section 2.2 and 2.3 derive the mathematical expressions of the dynamical models for tank 1 and tank 2.

2.1 Description

The process plant consists of two containers, tank 1 and tank 2. Tank 1 has a rectangular shape, whereas tank 2 has a conical shape. Tank 1 has two inlets, one from the pump PA001, and one from the mixer tap LV003, and two outlets, one which goes to the valve FV001 via a hose coil, and one which goes to tank 2 via the valve LV001. There is also a 2kW heating flask HE001 mounted in tank 1, in order to heat up the liquid.

Tank 2 has only one inlet, which is the one that comes from tank 1 via valve LV001. Tank 2 has two outlets, one via the valve FV002, and one via the valve LV002. Both of these outlets lead to the same collection vessel. The liquid in this vessel is the same liquid that gets pumped back into tank 1 by the pump PA001.

The plant is equipped with a variety of instrumentation. Table 2.1 describes the type of instrument, its name code, and purpose. Figure 2.1 shows a schematic sketch of the processing plant.

Type	**Code**	**Purpose**
Pressure gauge	PT001	Measures the pressure in the tap water
Temperature meter	TT003	Measures the temperature of the tap water
Temperature meter	TT001	Measures the temperature of tank 1
Level meter	LT001	Measures the level of tank 1
Temperature meter	TT002	Measures the temperature of the water from tank 1 which is delayed through the hose coil
Level meter	LT002	Measures the level of tank 2
Flow meter	FT001	Measures the water flow from the pump PA001

Table 2.1: System instrumentation.

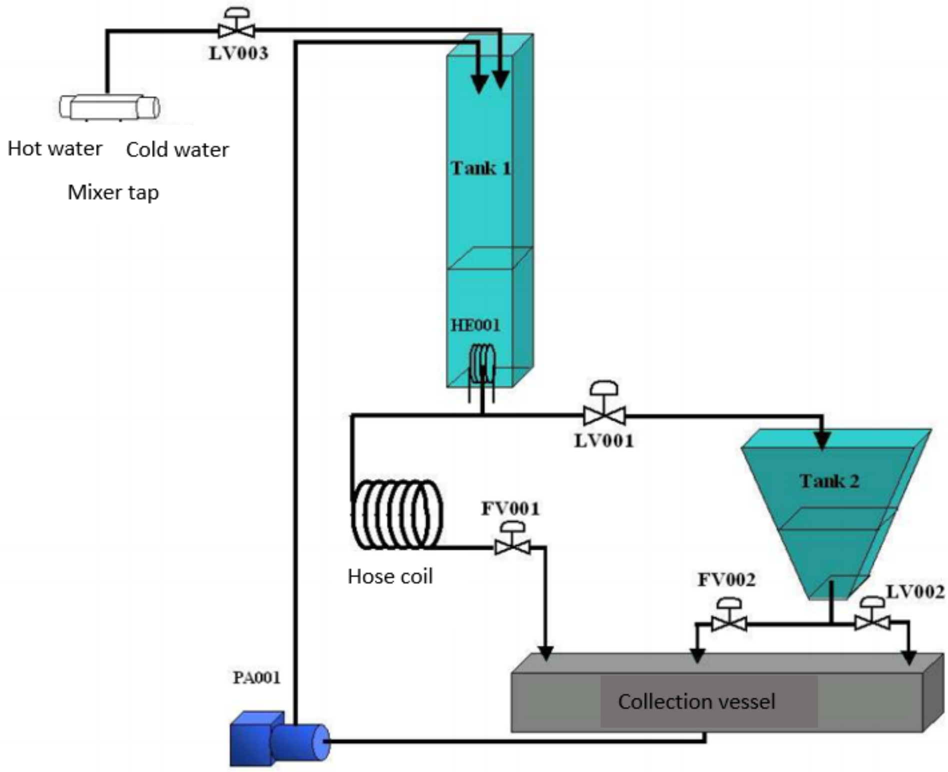


Figure 2.1: This is a schematic sketch of the two-tank processing plant [7].

The processing plant can be used to simulate a number of different industrial scenarios, but this report will use only a setting with limited functions. This means that some of the available functionalities will not be taken into account. This is done such that the complexity of the system is reduced, while the integrity of the verifiability and testing performed on the system is maintained.

The system that is taken into account when calculating a dynamical model, contains tank 1 and 2, the pump PA001, valve LV001, valve LV002 and the collection vessel. The instruments that will be used are FT001, LT001 and LT002. Figure 2.2 shows a simplified version of the schematic sketch that only includes the parts and instruments that are relevant.

With this configuration, one is able to control the water level of tank 1 via the LV001-valve, and the water level of tank 2 via the LV002-valve. The water pumped into tank 1 via the pump PA001 can also be regulated.

With a defined system and its boundaries, it becomes possible to create a mathematical model of the system. In order to make this mathematical model as accurate as possible, the characteristics of the valves and the pump must be considered. (2.1) is the mathematical model of the valves, it describes the volume flow q $[m^3/h]$ through the valves as a function of the valve constant $K_v[\frac{m^3/h}{\sqrt{bar}}]$, valve opening x and the pressure drop across the valve $\Delta p[bar]$.

$$q(t) = K_v\, x(t)\sqrt{\Delta p(t)} \tag{2.1}$$

As evident from (2.1), the results have the unit $[m^3/h]$ instead of $[m^3/s]$ for the volume flow. The results from (2.1) will be used later in other calculations, therefore, it is more convenient to have the units in SI format. This means dividing equation (2.1) by 3600.

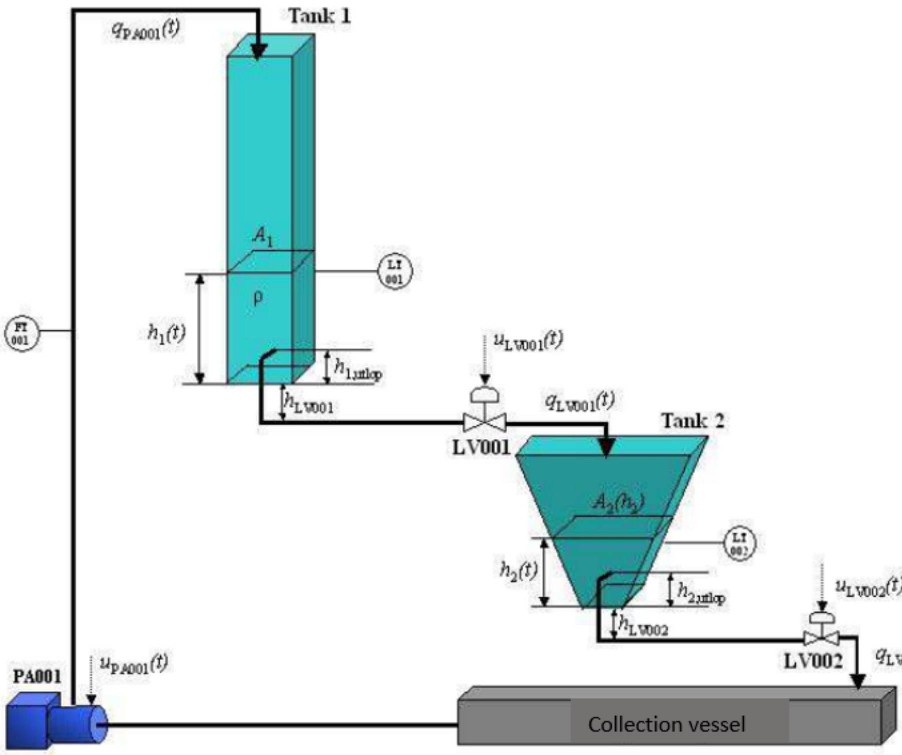


Figure 2.2: Simplified version of the schematic sketch with limited functionality [7].

In the processing plant, the pressure that will occur upstream of a valve will consist of the atmospheric pressure p_0 plus the water pressure p_w. The pressure downstream of the valve is only the atmospheric pressure, since the water flows out to an open tank. Thus, the differential pressure Δp is given by:

$$\Delta p(t) = p_{upstream} - p_{downstream} = (p_w(t) + p_0) - p_0 = p_w(t) \tag{2.2}$$

The pressure caused by the water is given by:

$$p_w(t) = \rho\, g\, h(t) \tag{2.3}$$

where ρ is the density of the liquid, g is the gravitational constant and $h(t)$ is the height that the liquid holds above the valve.

(2.3) is known as *Pascal's law*, and it could be inserted into (2.1) after converting from Pa to bar through a division by 100000.

$$q(t) = \frac{K_v\, x(t)}{3600} \sqrt{\frac{\rho\, g\, h(t)}{100000}} \tag{2.4}$$

Due to the flow characteristics of valve LV001 and LV002, it becomes necessary to take into account the nonlinearities between the actual flow through the valve, and the valve opening. This relation is nonlinear, which means that a certain percentage of the maximum opening of the valves does not necessarily correspond to the same percentage of the maximum flow through the valves. The relation between the valve opening $z(t)$ and the flow $f(\cdot)$ can be approximated by:

$$f(z) = \frac{e^{z(t)^{1.2}} - 1}{e - 1} \tag{2.5}$$

The system uses compressed air in order to make the valves move. The compressed air enters the lower side of the diaphragm, which opens the valve as the control signal $u(t)$ is increased. This applies for valve LV001 and LV002, and they are therefore regarded as normally closed valves. However,

In other words, these valves will close if the compressed air disappears. This type of actuator usually has $1st$ order dynamics. However, this can be neglected due to the slower dynamic of the processing plant. The control signal will be the same as the actual valve opening, hence (2.5) can be rewritten as (2.6). The relation between $f(u(t))$ and $u(t)$ is shown in Figure 2.3.

$$f(u(t)) = \frac{e^{u(t)^{1.2}} - 1}{e - 1} \tag{2.6}$$

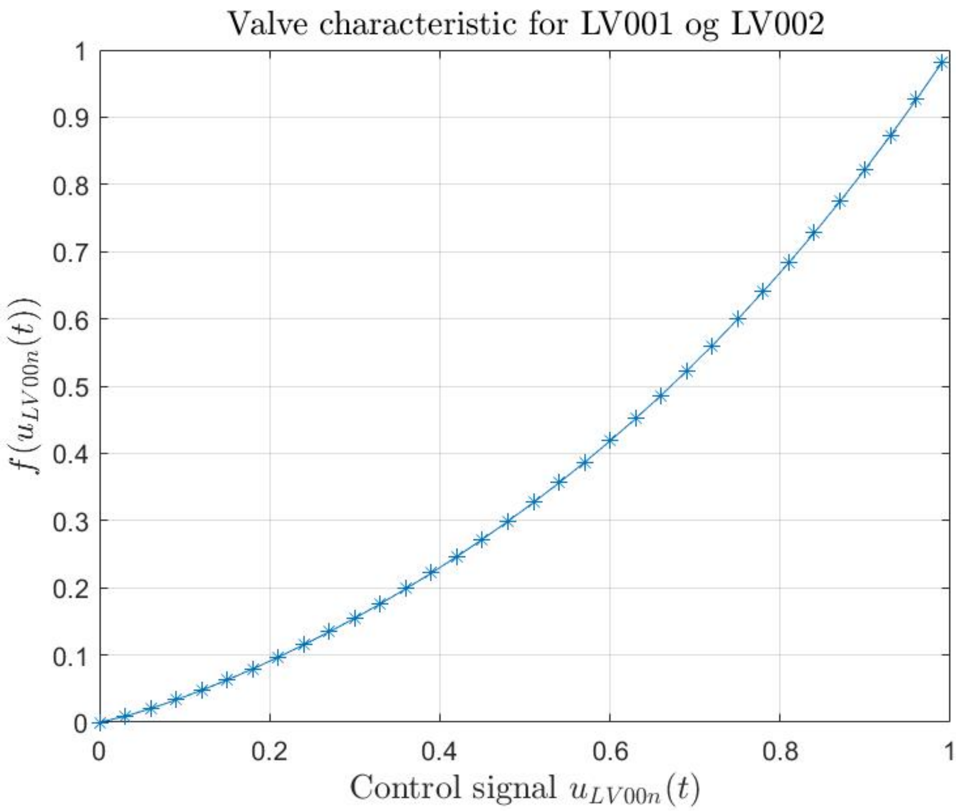


Figure 2.3: The relation between the control signal and the relative flow through the valve with respect to the maximum capacity [7].

With this relation defined, (2.4) can be rewritten as follows:

$$q = \frac{K_v \, f_n\big(u_{LV00n}(t)\big)}{3600} \sqrt{\frac{\rho \, g \, h(t)}{100000}} \tag{2.7}$$

where $n \in \{1, 2\}$, depending on which valve is considered.

As with the valves, the pump also has nonlinearities between the control signal $u_{PA001}(t)$ and the volume flow through the pump $q_{PA001}(t)$. The characteristics are usually provided by the supplier, but may vary depending on where and how the pump functions under operation. Variables such as pipe resistance and lifting height may affect the characteristics of the pump. Therefore, after the pump is installed, it is required to find the characteristics that apply for the pump in the processing plant. This is done by increasing the control signal incrementally, while also measuring the flow through the pump. Figure 2.4 presents the results of such a work process. The flow through the pump, with respect to the control signal, will be denoted as $f_3(u_{PA001}(t))$.

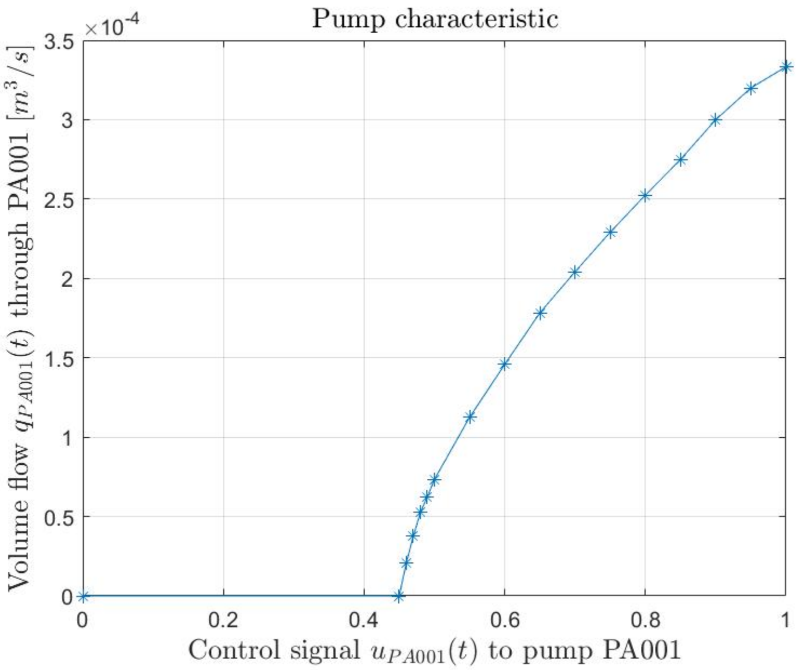


Figure 2.4: The relation between the control signal and the flow through the pump [7].

2.2 Nonlinear model of tank 1

With a restricted system, and the assumptions concerning the dynamics for the valves and the pump, it becomes fairly straightforward to create a dynamical model for the processing plant. The first step in achieving this is to use the balance law, which states the following:

A change in the amount per time in any system is equal to the net amount flow of the system.

The *amount* can be regarded as energy, mass, momentum, charge and also population. Net flow means the sum of all the inflows minus the sum of all the outflows plus the generated amount within the system:

$$\frac{d(amount)}{dt} = \sum In - \sum Out + \sum Generated \tag{2.8}$$

(2.8) results in one or more differential equations. In this case, the *amount* in the balance law is regarded as mass. The following part will derive the dynamical model for tank 1. Table 2.2 provides the relevant information regarding tank 1.

Since the *amount* in this case is mass, (2.8) can be written as (2.9), where m [kg] is the mass, w_i $[\frac{kg}{s}]$ is the mass flow, i denotes the different mass flows and t [s] is the time.

$$\frac{dm(t)}{dt} = \sum w_i(t) \tag{2.9}$$

where $m(t) = \rho\, V(t) = \rho\, A\, h(t)$. The mass flows are expressed as:

$$w_{IN}(t) = \rho\, q_{PA001}(t) = \rho\, f_3(u_{PA001}(t)) \tag{2.10}$$

$$w_{OUT}(t) = \rho\, q_{LV001}(t) \tag{2.11}$$

By substituting (2.7) into (2.11), we obtain the following equation:

$$w_{OUT}(t) = \rho \, \frac{K_v \, f_1\big(u_{LV001}(t)\big)}{3600} \sqrt{\frac{\rho \, g \, (h_1(t) + h_{LV001})}{100000}} \tag{2.12}$$

where the actual height $h(t)$ in (2.7) is the sum of h_{LV001} and $h_1(t)$.

Given this information the differential equation for the height of tank 1 can be expressed as:

$$\frac{dh_1(t)}{dt} = \frac{1}{A_1} \left(f_3(u_{PA001}(t)) - \frac{K_v \, f_1\big(u_{LV001}(t)\big)}{3600} \sqrt{\frac{\rho \, g \, (h_1(t) + h_{LV001})}{100000}} \right) \tag{2.13}$$

The reason why the term w_{OUT} is subtracted, is because it represents the mass flow *out* of the system. With (2.13), it becomes possible to describe the behaviour of the system. A mathematical model of any dynamical system is the basis for developing simulations and for analysing the characteristics of the system.

Name	**Description**	**Unit**	**Value**
$h_1(t)$	Water level in tank 1 measured with LT001	m	0-1
A_1	Area for tank 1	m^2	0.0096
ρ	Density of water	$\frac{kg}{m^3}$	1000
$q_{PA001}(t)$	Volume flow from pump PA001	$\frac{l}{min}$	0 -12
$u_{PA001}(t)$	Control signal to pump PA001	-	0 - 1
$q_{LV001}(t)$	Volume flow through LV001	$\frac{l}{min}$	0 - 17
$u_{LV001}(t)$	Control signal to valve LV001	-	0 - 1
$K_{v,LV001}$	Valve constant for valve LV001	$\frac{m^3}{time \sqrt{bar}}$	11.25
h_{LV001}	Height from the bottom of tank 1 down to LV001	m	0.05
$h_{1,utlop}$	Height from the bottom of tank 1 up to the tank outlet	m	0.14
g	Acceleration of gravity	$\frac{m}{s^2}$	9.81

Table 2.2: Variables regarding tank 1.

2.3 Nonlinear model of tank 2

This section will show how to obtain the mathematical model for tank 2. As for tank 1, Table 2.3 is a helpful tool which explains the variables seen in Figure 2.2. Since the procedure is more or less the same as for tank 1, this section will mostly present the mathematical expressions which leads to the differential equation.

Name	**Description**	**Unit**	**Value**
$h_2(t)$	Water level in tank 2 measured with LT002	m	0-0.4
$A_2(h_2(t))$	Area for tank 2 (function of $h_2(t)$)	m^2	0.025 - 0.07
$q_{LV002}(t)$	Volume flow through LV002	$\frac{l}{min}$	0 - 17
$u_{LV002}(t)$	Control signal to valve LV002	-	0 - 1
$K_{v,LV002}$	Valve constant for valve LV002	$\frac{m^3}{time \sqrt{bar}}$	11.25
h_{LV002}	Height from the bottom of tank 2 down to LV002	m	0.25
$h_{2,utlop}$	Height from the bottom of tank 2 up to the tank outlet	m	0.03

Table 2.3: Variables regarding tank 2.

As before, the first step is to set up the balance law, with mass as the amount. The structure is still identical to equation (2.9). As seen in Figure 2.2, the inflow comes from the valve LV001, and the outflow goes to the valve LV002. Recall that these in and outflows need to be regarded as *mass flows* in order to fit in (2.9). Equation (2.14) and (2.15) shows the corresponding mass flows to and from the system.

$$w_{IN}(t) = \rho\, q_{LV001}(t) = \rho\, \frac{K_v\, f_1\big(u_{LV001}(t)\big)}{3600} \sqrt{\frac{\rho\, g\, (h_1(t) + h_{LV001})}{100000}} \tag{2.14}$$

$$w_{OUT}(t) = \rho\, q_{LV002}(t) = \rho\, \frac{K_v\, f_2\big(u_{LV002}(t)\big)}{3600} \sqrt{\frac{\rho\, g\, (h_2(t) + h_{LV002})}{100000}} \tag{2.15}$$

Before inserting (2.14) and (2.15) into (2.9), recall also that the mass can be rewritten as a product between the density of the liquid, the area of the tank at a certain height, and the height of the liquid. Doing this results in the following equation:

$$\begin{aligned}\frac{dh_2(t)}{dt} = \frac{1}{A_2(h_2(t))} \Bigg(& \frac{K_v\, f_1\big(u_{LV001}(t)\big)}{3600} \sqrt{\frac{\rho\, g\, (h_1(t) + h_{LV001})}{100000}} \\ & - \frac{K_v\, f_2\big(u_{LV002}(t)\big)}{3600} \sqrt{\frac{\rho\, g\, (h_2(t) + h_{LV002})}{100000}} \Bigg)\end{aligned} \tag{2.16}$$

Note that the area, $A_2(h_2(t))$, is a function of the height $h_2(t)$ due to the conical shape of tank 2. In order to find this function, it is helpful to use the information presented in Figure 2.5 and Table 2.4.

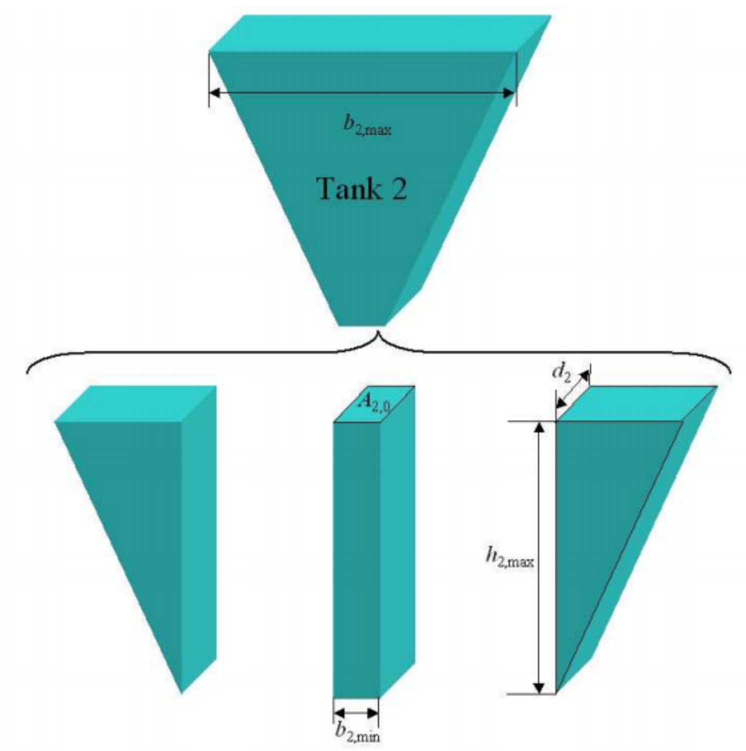


Figure 2.5: Schematic sketch of tank 2 for area calculation [7].

$A_{2,0}$	The area at the bottom of tank 2	0.004 m^2
d_2	Depth of tank 2	0.08 m
$b_{2,max}$	Upper width of tank 2	0.4 m
$b_{2,min}$	Lower width of tank 2	0.05 m
$h_{2,max}$	The height of tank 2	0.4 m

Table 2.4: The dimensions of tank 2.

As shown in Figure 2.5, the conical tank can be segmented into three parts, which consists of two identical right angled prisms and one rectangle shaped prism. The area of the conical tank is given by summing the area of these three segmented parts.

The area of the rectangle shaped prism is already stated in Table 2.3, and it is calculated by multiplying the lower width $b_{2,min}$ and the depth d_2. This area is always constant and independent of the water level.

The area of the two right angled prisms can be calculated by using the triangle similarity theorem. This theorem entails that if two triangles of different sizes have the same shape, then the ratio between two sides of triangle A is equal to the ratio between the same two sides of triangle B. This is illustrated in Figure 2.6.

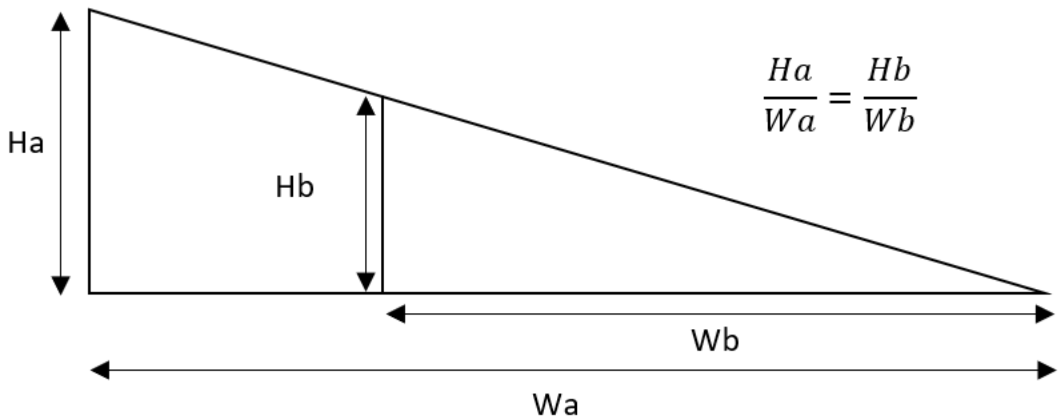


Figure 2.6: Triangle similarity.

In the case of the conical tank, W_a is $h_{2,max}$, W_b is $h_2(t)$ and H_a is given by:

$$H_a = \frac{b_{2,max} - b_{2,min}}{2} \tag{2.17}$$

In order to calculate the area of the right angled prisms, it is desired to multiply the current width H_b with the tanks depth $d2$. However, it is evident from Figure 2.5 that H_b is a function of the height $h_2(t)$. By using the triangular similarities, H_b can be expressed as:

$$H_b(t) = \frac{H_a W_b}{W_a} = \frac{\left(\frac{b_{2,max} - b_{2,min}}{2}\right) h_2(t)}{h_{2,max}} \tag{2.18}$$

which leads to:

$$A_p(t) = \frac{\left(\frac{b_{2,max} - b_{2,min}}{2}\right) h_2(t)}{h_{2,max}} d_2 \tag{2.19}$$

The total area for the conical tank is finally given by:

$$
\begin{aligned}
A_2(t) &= 2\,A_p + A_{2,0} \\
&= 2\,\frac{\left(\frac{b_{2,max}-b_{2,min}}{2}\right)h_2(t)}{h_{2,max}}\,d_2 + A_{2,0} \\
&= \frac{\left(b_{2,max}-b_{2,min}\right)h_2(t)}{h_{2,max}}\,d_2 + A_{2,0} \\
&= 0.07\,h_2(t) + 0.004
\end{aligned}
\tag{2.20}
$$

Inserting (2.20) into (2.16) gives the final differential equation for tank 2:

$$
\begin{aligned}
\frac{dh_2(t)}{dt} = \frac{1}{0.07\,h_2(t)+0.004}\Bigg(&\frac{K_v\,f_1\big(u_{LV001}(t)\big)}{3600}\sqrt{\frac{\rho\,g\,(h_1(t)+h_{LV001})}{100000}} \\
&-\frac{K_v\,f_2\big(u_{LV002}(t)\big)}{3600}\sqrt{\frac{\rho\,g\,(h_2(t)+h_{LV002})}{100000}}\Bigg)
\end{aligned}
\tag{2.21}
$$

This report will focus on the dynamical model for tank 1, see (2.13). Tank 2 will not be considered, however, the dynamical model (2.21) is included as an example and for possible use in future work.

Part II

Modelling using Taylor and Carleman approximations

Chapter 3

Taylor approximations

In this chapter, the nonlinear model:

$$\begin{aligned} \dot{h}_1(t) &= f\Big(h_1(t), u_{LV001}(t), u_{PA001}(t)\Big) \\ &= \frac{1}{A_1}\left(f_3(u_{PA001}(t)) - \frac{K_v\, f_1\big(u_{LV001}(t)\big)}{3600}\sqrt{\frac{\rho\, g\,(h_1(t) + h_{LV001})}{100000}}\right) \end{aligned} \tag{3.1}$$

is used as a starting point to obtain approximated models using the truncated Taylor series. This chapter is structured as follows. Section 3.1 summarises the key theoretical concepts related to the Taylor series and its use in approximating nonlinear differential equations. Section 3.2 derives and presents the Taylor approximations of the nonlinear tank model. Section 3.3 compares the obtained models so that a model can be chosen as a starting point for Chapter 4.

3.1 Taylor series

A mathematical pattern of summations can be expressed as a series. For instance $1 + 2 + ... + n$ can be expressed in the form of the summation $\sum_{i=1}^{n} i$. The *Taylor series* is indeed based upon the same principles, however, it is an infinite sum of the derivatives of a function at a single point. The definition of a single variable Taylor series is as follows:

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + \frac{f'(a)}{1!}(x-a) + ... \tag{3.2}$$

given that $f(x)$ is infinitely differentiable at a [24].

The Taylor series of an analytic function[1] will be equal to the function itself in the defined area around the point a.

As calculating the infinite sum of a series is not always feasible, an approximation using the finite partial sum is commonly utilised instead. The partial sum of the $n + 1$ first terms of the Taylor series forms the *nth Taylor polynomial* of the function, hereafter referred to as the *nth order Taylor approximation*. It is an approximation which will generally become better as n increases.

[1] A function where its Taylor series converges to the function itself at every point in a defined area around the working point.

The Taylor series has several uses in mathematics, with the most relevant to this report being the approximation of the nonlinear differential equation (3.1). Using an approximation can make an otherwise unsolvable, or hard to solve problem, solvable "near" a working point[2]. The *nth* order Taylor approximation of a n-times differentiable function allows for an approximation, which can be used to compute the functions value numerically. For the Taylor series, the function has to be infinitely differentiable at a, while the truncated version at n only requires that the function is n-times differentiable.

3.1.1 Taylor series in several variables

As we step into multivariable calculus, the Taylor series still applies and is described by:

$$\begin{aligned} f(x_1, x_2, ..., x_d) = & \\ & f(a_1, ..., a_d) + \sum_{j=1}^{d} \frac{\partial f(a_1, ..., a_d)}{\partial x_j}(x_j - a_j) \\ & + \frac{1}{2!} \sum_{j=1}^{d} \sum_{k=1}^{d} \frac{\partial^2 f(a_1, ..., a_d)}{\partial x_j \partial x_k}(x_j - a_j)(x_k - a_k) \\ & + \frac{1}{3!} \sum_{j=1}^{d} \sum_{k=1}^{d} \sum_{l=1}^{d} \frac{\partial^3 f(a_1, ..., a_d)}{\partial x_j \partial x_k \partial x_l}(x_j - a_j)(x_k - a_k)(x_l - a_l) + ... \end{aligned} \tag{3.3}$$

Consider the following nonlinear differential equation:

$$\frac{dx}{dt} = e^x \sqrt{y}$$

Calculate the 2nd order Taylor approximation around $(0, 1)$.

Compute all the necessary partial derivatives:

$$\begin{aligned} f_x &= e^x \sqrt{y} \\ f_y &= \tfrac{e^x}{2\sqrt{y}} \\ f_{xx} &= e^x \sqrt{y} \\ f_{yy} &= -\tfrac{e^x}{4y^{3/2}} \\ f_{xy} &= \tfrac{e^x}{2\sqrt{y}} \end{aligned}$$

Evaluate the partial derivatives at (0,1):

$$\begin{aligned} f_x(0, 1) &= 1 \\ f_y(0, 1) &= \tfrac{1}{2} \\ f_{xx}(0, 1) &= 1 \\ f_{yy}(0, 1) &= -\tfrac{1}{4} \\ f_{xy}(0, 1) &= \tfrac{1}{2} \end{aligned}$$

[2] In this context "near a working point" means the operating range around the approximated function where the approximation is adequate.

Inserting the partial derivatives into (3.3) truncated at the 2nd order gives:

$$T(0,1) = \frac{1}{2}x^2 - \frac{1}{8}y^2 + \frac{1}{2}xy + \frac{1}{2}x + \frac{3}{4}y + \frac{3}{8}$$

3.2 Taylor models

The dynamical model (3.1) will be the starting point for developing the following models. It is easier to perform the *Carleman linearization*, discussed in the next chapter, if the function under consideration is a polynomial. It is evident that (3.1) is not a polynomial. However, by means of the Taylor series, (3.1) can be rewritten as an infinite amount of terms which are polynomials as explained in Section 3.1.

This leads to an intermediate step where (3.1) must be transformed, via the Taylor series, into an equation involving polynomials.

Since the Taylor series describes a function with an infinite amount of terms, it becomes impossible to compute this in any numerical way. A practical solution is to truncate terms at a certain point. This truncation will lead to a numerically computable approximation at the expense of precision.

This section will focus on using the *Taylor approximation* on (3.1), and develop a variety of models where the amount of terms that get truncated will differ. In Section 3.3, these models will be compared to each other, and the most successful [3] approximation will become the starting point for the Carleman linearization.

3.2.1 Linear model

As described in Chapter 1, the 1st order Taylor approximation is the most common method used to linearize a system. By applying the concepts presented in Section 3.1, the linearized model can be written as:

$$\delta\dot{h}_1(t) \approx \left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t) \tag{3.4}$$

where O indicates the partial derivatives in the operating point.

(3.4) is a general equation, where the values of the partial derivatives are yet to be determined. Their values will differ depending on which operating point is selected. Therefore, they will be presented as symbolic functions.

Note that u_{LV001} and u_{PA001} are both independent variables in the functions f_1 and f_3. Due to this fact, in order to find the partial derivatives $\frac{\partial f}{\partial u_{LV001}}$ and $\frac{\partial f}{\partial u_{PA001}}$, one must take into account the chain rule for derivatives, as follows:

$$\frac{\partial f}{\partial u_{LV001}} = \frac{\partial f}{\partial f_1}\frac{\partial f_1}{\partial u_{LV001}} \tag{3.5}$$

$$\frac{\partial f}{\partial u_{PA001}} = \frac{\partial f}{\partial f_3}\frac{\partial f_3}{\partial u_{PA001}} \tag{3.6}$$

[3] This report will define a 'successful approximation' as a trade-off between precision and computational labour. If a model is slightly more accurate, but it requires substantially more calculations, then it will be discarded.

The partial derivatives $\frac{\partial f}{\partial f_1}$ and $\frac{\partial f}{\partial f_3}$ are found by straight forward differentiation of the non-linear function (3.1) at the operating point:

$$\left.\frac{\partial f}{\partial f_1}\right|_O = \frac{-K_{v,LV001}\sqrt{\frac{g\,\rho\,(h_{1,O}+h_{LV001})}{10^5}}}{3600\,A_1} \tag{3.7}$$

$$\frac{\partial f}{\partial f_3} = \frac{1}{A_1} \tag{3.8}$$

The next step is finding the partial derivatives $\frac{\partial f_1}{\partial u_{LV001}}$ and $\frac{\partial f_3}{\partial u_{PA001}}$. These can be found by using the definition of the derivative on the valve and pump characteristics. Note that these will also vary depending on the selected operating point:

$$\left.\frac{\partial f_1}{\partial u_{LV001}}\right|_O = \lim_{\Delta\to 0}\frac{f_1(u_{LV001,O}+\Delta)-f_1(u_{LV001,O})}{\Delta} \approx \frac{\Delta f_1}{\Delta u_{LV001}} \tag{3.9}$$

$$\left.\frac{\partial f_3}{\partial u_{PA001}}\right|_O = \lim_{\Delta\to 0}\frac{f_3(u_{PA001,O}+\Delta)-f_3(u_{PA001,O})}{\Delta} \approx \frac{\Delta f_3}{\Delta u_{PA001}} \tag{3.10}$$

Figure 3.1 shows a schematic sketch on how (3.9) is calculated manually, using the valve characteristics in Simulink. The exact same concept applies for (3.10), but the pump characteristics and the input variable u_{PA001} are used instead. In a practical setting, $\Delta \to 0.01$, instead of $\Delta \to 0$.

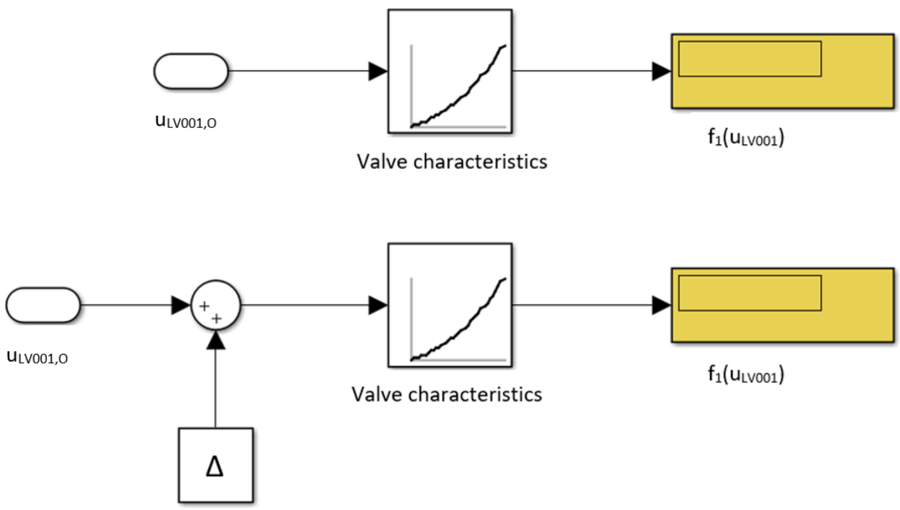


Figure 3.1: Calculating $\frac{\delta f_1}{\delta u_{LV001}}$ in Simulink.

Inserting (3.7)-(3.10) into (3.5)-(3.6) results in the final expressions for the partial derivatives regarding the inputs $u_{LV001}(t)$ and $u_{PA001}(t)$:

$$\left.\frac{\partial f}{\partial u_{LV001}}\right|_O = \frac{-K_{v,LV001}\sqrt{\frac{g\,\rho\,(h_{1,O}+h_{LV001})}{10^5}}}{3600\,A_1}\,\frac{\delta f_1}{\delta u_{LV001}} \tag{3.11}$$

$$\left.\frac{\partial f}{\partial u_{PA001}}\right|_O = \frac{1}{A_1}\,\frac{\delta f_3}{\delta u_{PA001}} \tag{3.12}$$

while the partial derivative $\frac{\partial f}{\partial h_1}$ is given by:

$$\left.\frac{\partial f}{\partial h_1}\right|_O = -\frac{K_{v,LV001}\, g\, \rho\, f_1(u_{LV001,O})}{7.2 \cdot 10^8\, A_1 \sqrt{\frac{g\, \rho\, (h_{1,O}+h_{LV001})}{10^5}}} \tag{3.13}$$

3.2.2 Quadratic model

Performing the $2nd$ order Taylor approximation on the nonlinear function (3.1), results in another approximated model which will be referred to as the *quadratic model*:

$$\begin{aligned}
\delta\dot{h}_1(t) =& \left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t) \\
&+ \frac{1}{2}\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t) + \frac{1}{2}\left.\frac{\partial^2 f}{\partial u_{LV001}^2}\right|_O \delta u_{LV001}^2(t) + \frac{1}{2}\left.\frac{\partial^2 f}{\partial u_{PA001}^2}\right|_O \delta u_{PA001}^2(t) \\
&+ \left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\, \delta u_{LV001}(t) + \left.\frac{\partial^2 f}{\partial h_1 \partial u_{PA001}}\right|_O \delta h_1(t)\, \delta u_{PA001}(t) \\
&+ \left.\frac{\partial^2 f}{\partial u_{LV001} \partial u_{PA001}}\right|_O \delta u_{LV001}(t)\, \delta u_{PA001}(t)
\end{aligned} \tag{3.14}$$

As evident by (3.14), the number of partial derivatives in the nth order Taylor approximation increases rapidly. Expanding beyond the quadratic model with a multivariable function results in lengthy equations. Therefore, one can opt to present the same partial derivatives using tables. This is simply done in order to have a more compact notation.

Table 3.1 illustrates this form of notation for the quadratic model. The f in each cell is the function that is being derived. In this case, f represents the nonlinear function. The subscripts represents what the nonlinear function f is being derived with respect to. In this case, $x = h_1$, $y = u_{LV001}$ and $z = u_{PA001}$. Note that Table 3.1 only shows the $2nd$ order terms. For the complete quadratic model, these terms comes as an *addition to* the terms found in the linear model.

f_{xx}	f_{xy}	f_{xz}
f_{yx}	f_{yy}	f_{yz}
f_{zx}	f_{zy}	f_{zz}

Table 3.1: Table presenting the $2nd$ order partial derivatives of the nonlinear function.

The red cells in Table 3.1 represents the non-zero partial derivatives. The non-zero $2nd$ order partial derivatives are as follows:

$$\begin{aligned}
&\frac{1}{2!}\left(\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t) + 2\left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\, \delta u_{LV001}(t) \right) = \\
&\frac{1}{2}\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t) + \left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\, \delta u_{LV001}(t)
\end{aligned} \tag{3.15}$$

Note that $f_{xy} = f_{yx}$.

Since the partial derivatives in (3.15) are only the $2nd$ order partials, these need to be added to the linear model in order to complete the quadratic model:

$$\begin{aligned}\delta\dot{h}_1(t) =& \left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t) \\ &+ \frac{1}{2}\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t) + \left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\,\delta u_{LV001}(t)\end{aligned} \tag{3.16}$$

As with the linear model, the following equations only shows the symbolic functions of the $2nd$ order partial derivatives:

$$\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O = \frac{\mathrm{Kv}_{\mathrm{LV001}}\, f_1(u_{LV001,\,O})\, g^2\, \rho^2}{1.44\cdot 10^{14}\, A_1 \left(\frac{g\,\rho\left(h_{1,\,O}+h_{\mathrm{LV001}}\right)}{10^5}\right)^{3/2}}$$

$$\left.\frac{\partial^2 f}{\partial h_1 \partial f_1}\right|_O = -\frac{\mathrm{Kv}_{\mathrm{LV001}}\, g\, \rho}{7.2\cdot 10^8\, A_1 \sqrt{\frac{g\,\rho\left(h_{1,O}+h_{\mathrm{LV001}}\right)}{10^5}}}$$

$$\left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O = \frac{\partial^2 f}{\partial h_1 \partial f_1}\,\frac{\delta f1}{\delta u_{LV001}} = -\frac{\mathrm{Kv}_{\mathrm{LV001}}\, g\, \rho}{7.2\cdot 10^8\, A_1 \sqrt{\frac{g\,\rho\left(h_{1,O}+h_{\mathrm{LV001}}\right)}{10^5}}}\,\frac{\delta f1}{\delta u_{LV001}}$$

$$\frac{\partial^2 f}{\partial f_1^2} = \frac{\partial^2 f}{\partial f_3^2} = \frac{\partial^2 f}{\partial h_1 \partial f_3} = \frac{\partial^2 f}{\partial f_1 \partial f_3} = 0$$

3.2.3 Partially quadratic models

Before performing the $3rd$ order Taylor approximation on the nonlinear function, it may be of interest to inspect further the quadratic model. A question that arises is what are the consequences of excluding certain $2nd$ order terms from the quadratic model. Doing this, will result in a so-called *Partially Quadratic* (PQ) model. As evident by (3.16), there are only two $2nd$ order terms. This gives the opportunity to create two different PQ models.

These two PQ models will be referenced using subscripts throughout this report. The subscript $_A$ means that the $2nd$ order term that was *included* is $\frac{\partial^2 f}{\partial h_1^2}$. The subscript $_B$ implies that the $2nd$ order term that was *included* is $\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}$.

PQ_A:

$$\begin{aligned}\delta\dot{h}_1(t) =& \left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t) \\ &+ \frac{1}{2}\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t)\end{aligned} \tag{3.17}$$

PQ_B:

$$\delta \dot{h}_1(t) = \left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t) + \left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\, \delta u_{LV001}(t) \tag{3.18}$$

3.2.4 Cubic model

Performing the $3rd$ order Taylor approximation on the nonlinear function (3.1), results in the following model referred to as the *cubic model*:

$$\begin{aligned}\delta \dot{h}_1(t) = &\left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t) \\ &+ \left.\frac{1}{2}\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t) + \left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\, \delta u_{LV001}(t) \\ &+ \left.\frac{1}{6}\frac{\partial^3 f}{\partial h_1^3}\right|_O \delta h_1^3(t) + \left.\frac{1}{2}\frac{\partial^3 f}{\partial h_1^2\, \partial u_{LV001}}\right|_O \delta h_1^2(t)\, \delta u_{LV001}(t)\end{aligned} \tag{3.19}$$

As stated earlier, the amount of partial derivatives in a multivariable nth order Taylor approximation increases rapidly as n increases. The amount of partial derivatives that are *added* to a *(n-1)th* order Taylor approximation as the order increases to n, is given by d^n, where d is the amount of variables and n is the order of the Taylor approximation of $f(x_1, x_2..., x_d)$.

This implies that increasing the order of the Taylor approximation from 2 to 3 would require $d^n = 3^3 = 27$ partial derivatives, in addition to those calculated for the $2nd$ order Taylor approximation. This is the case for the nonlinear system, since the amount of variables are equal to three. These *additional* partial derivatives are presented in Table 3.2.

f_{xxx}	f_{xxy}	f_{xxz}	f_{xyx}	f_{xyy}	f_{xyz}	f_{xzx}	f_{xzy}	f_{xzz}
f_{yxx}	f_{yxy}	f_{yxz}	f_{yyx}	f_{yyy}	f_{yyz}	f_{yzx}	f_{yzy}	f_{yzz}
f_{zxx}	f_{zxy}	f_{zxz}	f_{zyx}	f_{zyy}	f_{zyz}	f_{zzx}	f_{zzy}	f_{zzz}

Table 3.2: Table presenting the $3rd$ order partial derivatives of the nonlinear function.

Note that Table 3.2 only shows the $3rd$ order partials. As with the quadratic model, the red cells in Table 3.2 represent the non-zero partial derivatives, given by:

$$\begin{aligned}&\frac{1}{3!}\Big(\frac{\partial^3 f}{\partial h_1^3}\,\delta h_1^3(t) + 3\,\frac{\partial^3 f}{\partial h_1^2 \partial u_{LV001}}\,\delta h_1^2(t)\,\delta u_{LV001}(t)\Big) = \\ &\frac{1}{6}\frac{\partial^3 f}{\partial h_1^3}\,\delta h_1^3(t) + \frac{1}{2}\frac{\partial^3 f}{\partial h_1^2 \partial u_{LV001}}\,\delta h_1^2(t)\,\delta u_{LV001}(t)\end{aligned} \tag{3.20}$$

Note that $f_{xxy} = f_{xyx} = f_{yxx}$.

Adding (3.20) to the quadratic model (3.16), results in the cubic model presented in (3.19).

The following equations show the symbolic functions of the non-zero $3rd$ order partial derivatives:

$$\left.\frac{\partial^3 f}{\partial h_1^3}\right|_O = -\frac{\mathrm{Kv}_{\mathrm{LV001}}\, f_1(u_{LV001,O})\, g^3\, \rho^3}{9.6\cdot 10^{18}\, A_1 \left(\frac{g\,\rho\left(h_{1,O}+h_{\mathrm{LV001}}\right)}{100000}\right)^{5/2}} \tag{3.21}$$

$$\left.\frac{\partial^3 f}{\partial h_1^2\,\partial u_{LV001}}\right|_O = \left.\frac{\partial^3 f}{\partial h_1^2 \partial f_1}\right|_O \frac{\delta f_1}{\delta u_{LV001}} = \frac{\mathrm{Kv}_{\mathrm{LV001}}\, g^2\, \rho^2}{1.44\cdot 10^{14}\, A_1 \left(\frac{g\,\rho\left(h_{1,O}+h_{\mathrm{LV001}}\right)}{100000}\right)^{3/2}} \frac{\delta f_1}{\delta u_{LV001}} \tag{3.22}$$

3.2.5 Higher order models

The $4th$ and $5th$ order Taylor approximations of the nonlinear function proves rather tedious to calculate. For the $5th$ order approximation, it includes a total of 363 partial derivatives. Presenting these in detail will add little to no value to this report. Therefore, only the final equation for the $4th$ and $5th$ order models are presented. The $4th$ order model is given by:

$$\begin{aligned}\delta\dot{h}_1(t) =& \left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t)\\ &+ \frac{1}{2}\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t) + \left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\,\delta u_{LV001}(t) + \frac{1}{6}\left.\frac{\partial^3 f}{\partial h_1^3}\right|_O \delta h_1^3(t)\\ &+ \frac{1}{2}\left.\frac{\partial^3 f}{\partial h_1^2 \partial u_{LV001}}\right|_O \delta h_1^2(t)\,\delta u_{LV001}(t) + \frac{1}{24}\left.\frac{\partial^4 f}{\partial h_1^4}\right|_O \delta h_1^4(t) + \frac{1}{6}\left.\frac{\partial^4 f}{\partial h_1^3 \partial u_{LV001}}\right|_O \delta h_1^3(t)\,\delta u_{LV001}(t)\end{aligned} \tag{3.23}$$

The $5th$ order model is given by:

$$\begin{aligned}\delta\dot{h}_1(t) =& \left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t)\\ &+ \frac{1}{2}\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t) + \left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\,\delta u_{LV001}(t) + \frac{1}{6}\left.\frac{\partial^3 f}{\partial h_1^3}\right|_O \delta h_1^3(t)\\ &+ \frac{1}{2}\left.\frac{\partial^3 f}{\partial h_1^2 \partial u_{LV001}}\right|_O \delta h_1^2(t)\,\delta u_{LV001}(t) + \frac{1}{24}\left.\frac{\partial^4 f}{\partial h_1^4}\right|_O \delta h_1^4(t) + \frac{1}{6}\left.\frac{\partial^4 f}{\partial h_1^3 \partial u_{LV001}}\right|_O \delta h_1^3(t)\,\delta u_{LV001}(t)\\ &+ \frac{1}{120}\left.\frac{\partial^5 f}{\partial h_1^5}\right|_O \delta h_1^5(t) + \frac{1}{24}\left.\frac{\partial^5 f}{\partial h_1^4 \partial u_{LV001}}\right|_O \delta h_1^4(t)\,\delta u_{LV001}(t)\end{aligned} \tag{3.24}$$

The reason why the $5th$ order model has a noticeably lower amount of terms than 363, is because the majority of these partial derivatives result in zero. Whether it is appropriate to consider this model in further research, will be discussed in the next section. The $4th$ order model is presented for completeness, but will not be further considered.

3.3 Taylor model comparison

This section will analyse the different models obtained in Section 3.2. The models will be compared to each other, and the results of these comparisons will help determine which model is to be used as a starting point for the *Carleman linearization*.

Before we can start the different comparisons, we have to make some general assumptions regarding the different models. As stated in Section 3.2, the different models will vary depending on the chosen operating point. The degree of nonlinearity of tank 1 will differ with the water level, therefore, it makes sense to choose two operating points that are at the lower- and higher end of the tank. By doing this, we can also see if the degree of nonlinearity will have an impact on the results. The operating points for the height are set at $0.25m$ and $0.75\ m$. $h_{1,O} = 0.25m$ will be referred to as *Scenario 1*, and $h_{1,O} = 0.75m$ as *Scenario 2*. In order to keep this section tidy, we have decided to present only the results regarding *Scenario 1*. The corresponding results obtained from *Scenario 2* will be presented in Appendix A.

The next assumption is that the pump's behaviour is constant. The control signal $u_{PA001,O} = 0.65$ is therefore set for both *Scenario 1* and *2*. Reading Figure 2.4 at $u_{PA001,O} = 0.65$ gives $f_{3,O} = 0.0001783$. It will be stated explicitly when this assumption is no longer true. Otherwise, the reader can safely assume that the pump is constant at $u_{PA001,O} = 0.65$.

To find the operating point for the valve LV001, we follow this four step process:

1. Set the nonlinear equation (3.1) equal to zero.
2. Insert the chosen operating points $h_{1,O}$ and $f_{3,O}$.
3. Solve the equation with respect to $f_{1,O}$.
4. Insert $f_{1,O}$ into Figure 2.3 in order to find $u_{LV001,O}$.

Obviously, since we are considering two different operating points for the height, this process is done once for each scenario. These results are presented in Table 3.3.

	Scenario 1	***Scenario 2***
$\mathbf{f}_{1,O}$	0.3326	0.2037
$\mathbf{u}_{LV001,O}$	0.5159	0.3666
$\mathbf{f}_{3,O}$	0.0001783	0.0001783
$\mathbf{u}_{PA001,O}$	0.65	0.65
$\mathbf{h}_{1,O}$**[m]**	0.25	0.75

Table 3.3: Operating values for *Scenario 1* and *2*.

The equations and the methods for calculating the different partials are explained in Section 3.2. With these operating points, it becomes possible to write down the corresponding models. The only modification to the nonlinear model (3.1) is that $f_3(u_{PA001}(t))$ is replaced by $f_{3,O} = 0.0001783$:

$$\dot{h}_1(t) = \frac{1}{A_1}\left(0.0001783 - \frac{K_v\, f_1(u_{LV001}(t))}{3600}\sqrt{\frac{\rho\, g\,(h_1(t) + h_{LV001})}{100000}}\right) \tag{3.25}$$

The Taylor models for *Scenario 1* are as follows:

Linear model:

$$\delta\dot{h}_1(t) = -0.0297\,(h_1(t) - 0.25) - 0.0525\,(u_{LV001}(t) - 0.5159) \tag{3.26}$$

Quadratic model:

$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0297\,(h_1(t)-0.25) - 0.0525\,(u_{LV001}(t)-0.5159)\\ &+0.0248\,(h_1(t)-0.25)^2 - 0.0876\,(h_1(t)-0.25)\,(u_{LV001}(t)-0.5159)\end{aligned} \tag{3.27}$$

PQ_A model:

$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0297\,(h_1(t)-0.25) - 0.0525\,(u_{LV001}(t)-0.5159)\\ &+0.0248\,(h_1(t)-0.25)^2\end{aligned} \tag{3.28}$$

PQ_B model:

$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0297\,(h_1(t)-0.25) - 0.0525\,(u_{LV001}(t)-0.5159)\\ &-0.0876\,(h_1(t)-0.25)\,(u_{LV001}(t)-0.5159)\end{aligned} \tag{3.29}$$

Cubic model:

$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0297\,(h_1(t)-0.25) - 0.0525\,(u_{LV001}(t)-0.5159)\\ &+0.0248\,(h_1(t)-0.25)^2 - 0.0876\,(h_1(t)-0.25)\,(u_{LV001}(t)-0.5159)\\ &-0.0413\,(h_1(t)-0.25)^3 + 0.0730\,(h_1(t)-0.25)^2\,(u_{LV001}(t)-0.5159)\end{aligned} \tag{3.30}$$

$5th$ order model:

$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0297\,(h_1(t)-0.25) - 0.0525\,(u_{LV001}(t)-0.5159)\\ &+0.0248\,(h_1(t)-0.25)^2 - 0.0876\,(h_1(t)-0.25)\,(u_{LV001}(t)-0.5159)\\ &-0.0413\,(h_1(t)-0.25)^3 + 0.0730\,(h_1(t)-0.25)^2\,(u_{LV001}(t)-0.5159)\\ &+0.0860\,(h_1(t)-0.25)^4 - 0.1216\,(h_1(t)-0.25)^3\,(u_{LV001}(t)-0.5159)\\ &-0.2006\,(h_1(t)-0.25)^5 + 0.2534\,(h_1(t)-0.25)^4\,(u_{LV001}(t)-0.5159)\end{aligned} \tag{3.31}$$

Note that every term regarding the pump will result in zero, as $u_{PA001}(t)$ is constant at the operating point. This is given by:

$$\delta u_{PA001}(t) = u_{PA001}(t) - u_{PA001,O} = u_{PA001,O} - u_{PA001,O} = 0$$

Given that Eqs. (3.25)-(3.31) only depend on two variables, it becomes possible to create a 3-D plot that shows combinations of $h_1(t)$ and $u_{LV001}(t)$, and the resulting $\delta\dot{h}_1(t)$. In order to create a 2-D grid with uniformly spaced u_{LV001}-coordinates and h_1-coordinates in the interval [0,1], we use the `meshgrid` [4] MATLAB function. Using this 2-D grid as an input in the Eqs. (3.25)-(3.31) results in the matrix ξ for each model. The elements in ξ represent $\delta\dot{h}_1(t)$, while the indices of the columns and rows represent u_{LV001} and h_1, respectively. Figure 3.2 shows such a 3-D plot for the nonlinear model.

The intersection between the nonlinear model and the u_{LV001}-h_1 plane forms a level curve at $\delta\dot{h}_1(t) = 0$. It represents the operating points of the system. This level curve shows every combination of $h_1(t)$ and $u_{LV001}(t)$, where the system is at an equilibrium point (see Figure 3.3).

[4] Meshgrid

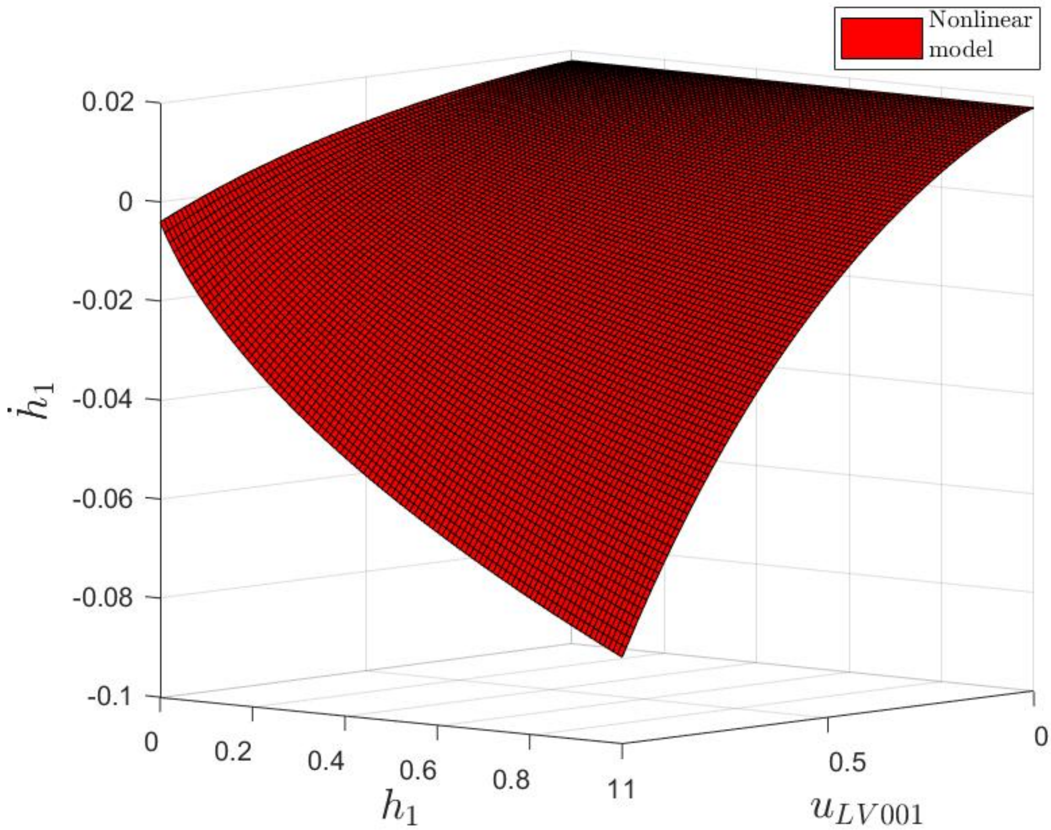


Figure 3.2: 3-D plot of the nonlinear model.

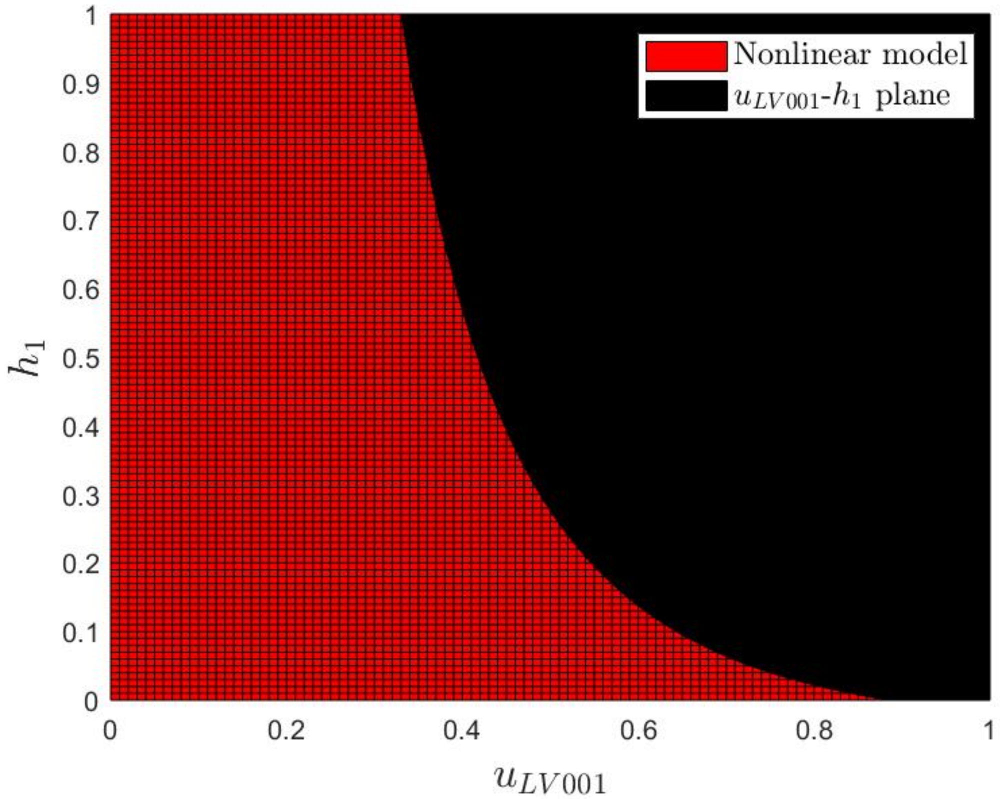
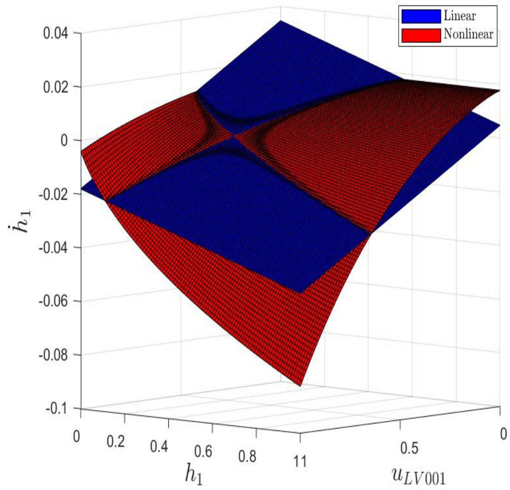
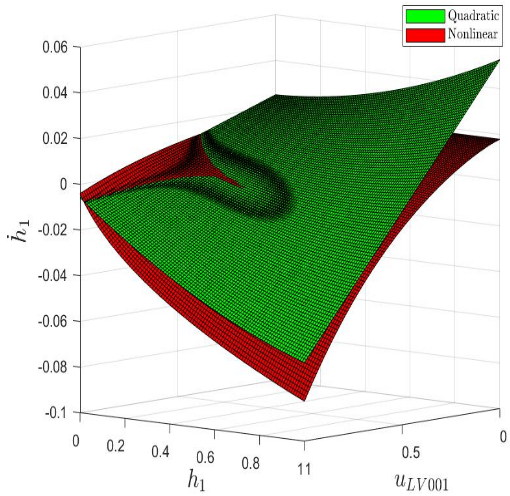


Figure 3.3: Level curve representing the operating points of the nonlinear model.

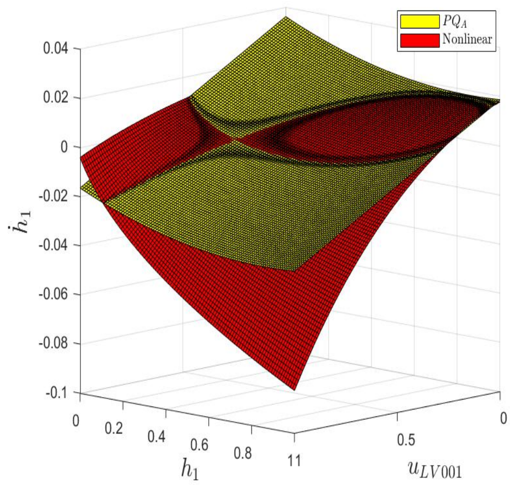
Figure 3.4 shows 3-D plots of the obtained Taylor models, with the nonlinear model as reference.



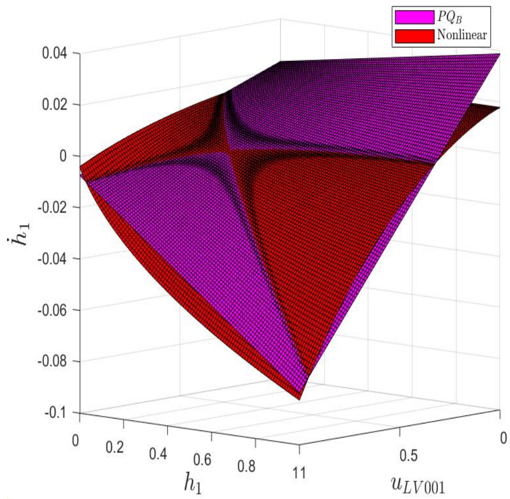
(a) Linear and nonlinear model.



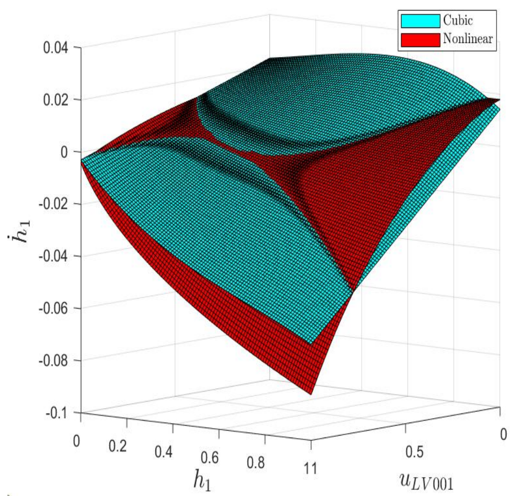
(b) Quadratic and nonlinear model.



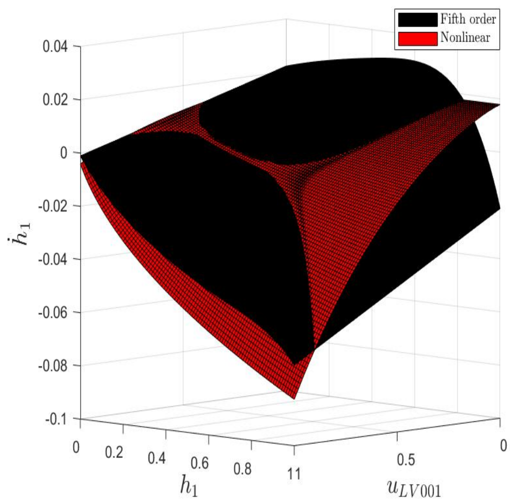
(c) PQ_A and nonlinear model.



(d) PQ_B and nonlinear model.



(e) Cubic and nonlinear model.



(f) $5th$ order and nonlinear model.

Figure 3.4: Comparisons between the nonlinear and the Taylor approximation models.

The series of Figures 3.4a through 3.4f shows that the Taylor models become more similar to the nonlinear model as the order n of the Taylor approximation increases.

The 3-D plots gives a general idea of how well the models are performing. To be able to quantify the performance of the Taylor models, we use contour plots [5], where the height values on the $u_{LV001} - h_1$ plane are given by:

$$\zeta = \left|\xi_{Nonlinear\_model} - \xi_{Taylor\_model}\right| \tag{3.32}$$

Contour plotting ζ provides an intuitive comparison between the nonlinear model and the Taylor model in question. Whether the Taylor models approximates a higher or lower value for $\delta h_1(t)$ than the nonlinear model may be useful information in some cases. However, in this case, we are only interested in the magnitude of the error between our approximations and the nonlinear model. Figure 3.5 presents the contour plots for the different Taylor models.

Figures 3.5a- 3.5f show that the *region* where ζ is close to zero, which means that the Taylor approximation and the nonlinear model are similar, increases as we consider higher order Taylor approximations.

The presented 3-D and contour plot gives the general notion on how a Taylor series approximation represents its original function more accurately as the order of the approximation increases. However, the definition of a successful model in this report was not based on accuracy alone. It is also of interest to know the *degree of improvement* as the Taylor model goes from an nth order to an $(n+1)th$ order approximation. If this *degree of improvement* is not considered significant enough, then it leads to selecting the final truncation to happen at n.

In order to see how well the models compare, we calculate the difference in the obtained ζ-value for each model, given by:

$$\psi = \zeta_A - \zeta_B \tag{3.33}$$

where the subscripts $_A$ and $_B$ are used to separate the ζ-value for the models in question. The sign and magnitude of ψ will indicate which model provides a better approximation. Table 3.4 shows how the values of ψ have been considered.

ψ **value**	**Result**	**Color code**
ψ >0.005	B >>	Green
0.005 >ψ >0	B >	Light green
0 >ψ >-0.005	A >	Light Blue
-0.005 >ψ	A>>	Blue

Table 3.4: Labels for the different values of ψ.

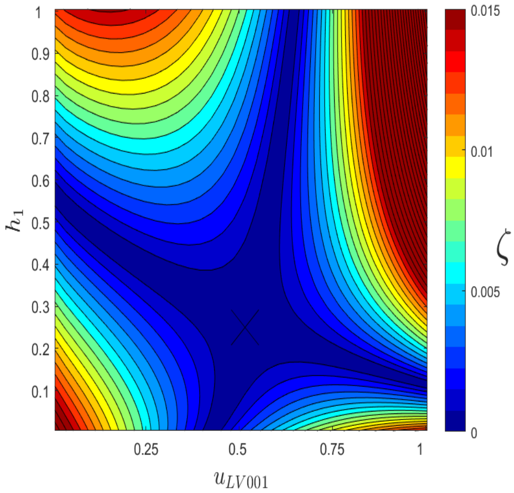
Certain points where both models approximate the nonlinear model poorly are considered non-relevant for the comparison. These occurrences are identified as shown in Table 3.5.

Criteria	**Result**	**Color code**
$\lvert\zeta_A\rvert > 0.005$ & $\lvert\zeta_B\rvert > 0.005$	Not applicable (N/A)	Red

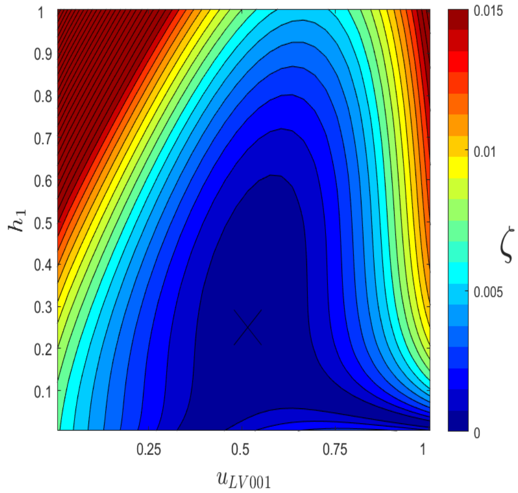
Table 3.5: Definition of not applicable points.

Figure 3.6 shows color maps which uses the color codes given in Table 3.4 and 3.5.

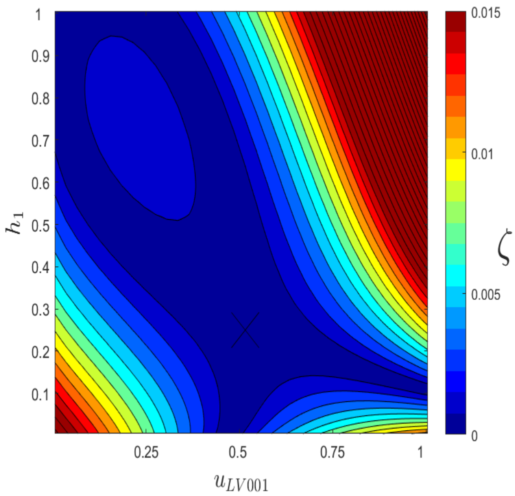
[5] In order to make these plots, the MATLAB function *contourf* is utilised: Contourf



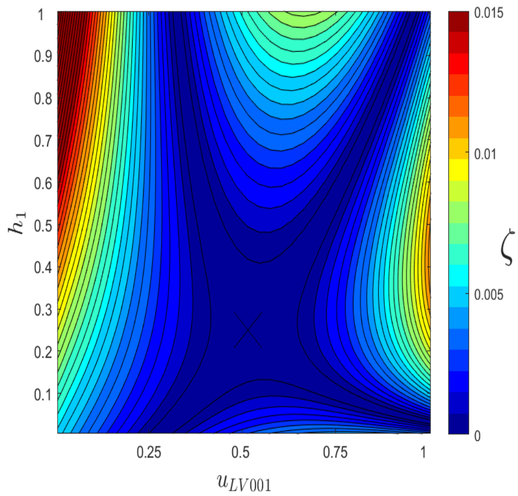
(a) Linear and nonlinear model.



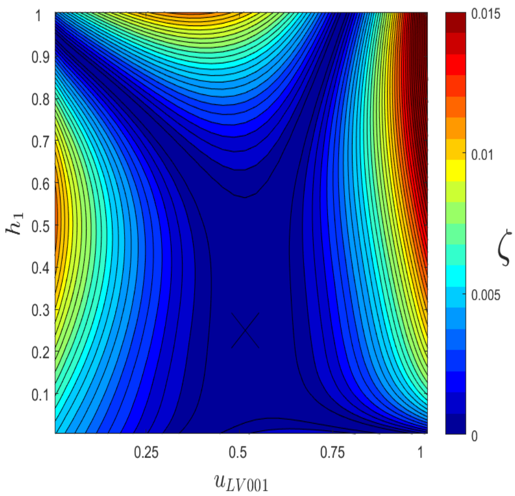
(b) Quadratic and nonlinear model.



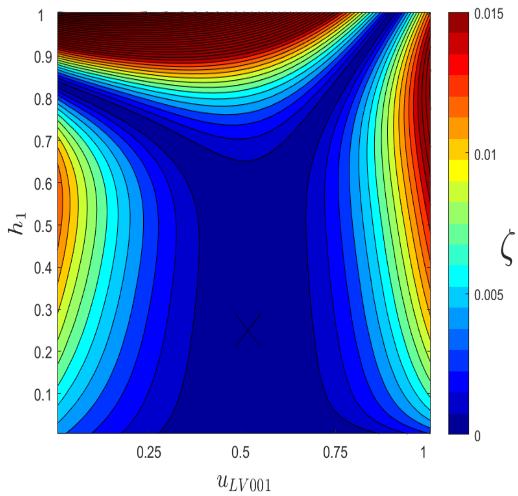
(c) PQ_A and nonlinear model.



(d) PQ_B and nonlinear model.

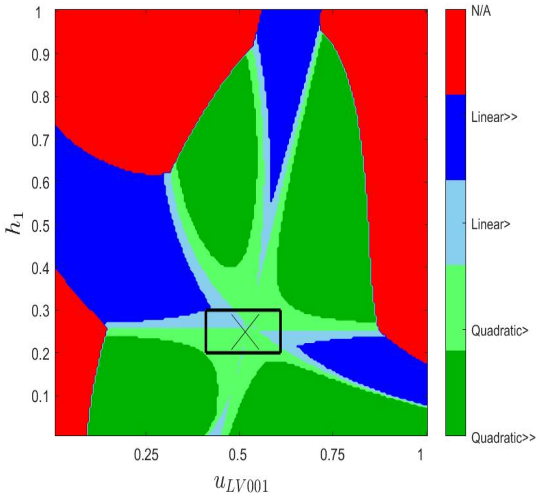


(e) Cubic and nonlinear model.

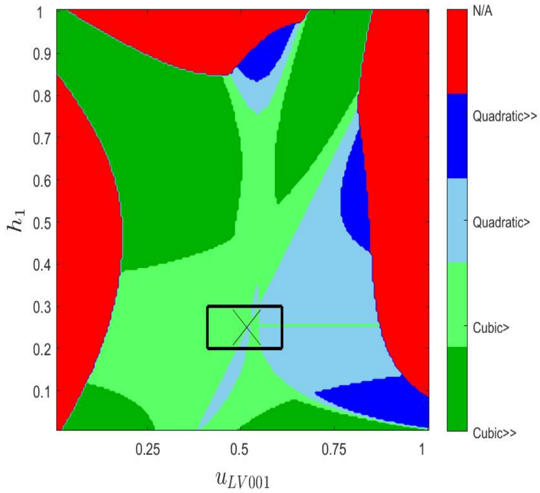


(f) $5th$ order and nonlinear model.

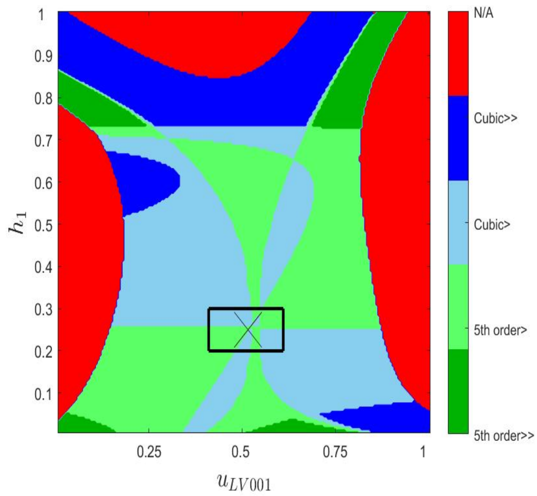
Figure 3.5: Contour plots of the Taylor models and the nonlinear model, where ζ is given by (3.32).



(a) $\psi = \zeta_{Linear} - \zeta_{Quadratic}$.



(b) $\psi = \zeta_{Quadratic} - \zeta_{Cubic}$.



(c) $\psi = \zeta_{Cubic} - \zeta_{5th\_Order}$.

Figure 3.6: Color maps comparing the different Taylor models.

It is important to note that the Taylor models only approximates the nonlinear model *around the operating point*. Therefore, to test the *degree of improvement* from an nth order to an $(n+1)th$ order approximation, only the intervals $[0.2m, 0.3m]$ for h_1 and $[0.41, 0.61]$ for u_{LV001} will be considered. These intervals are represented by the hollow rectangle. The large cross indicates the operating point, where $h_1 = 0.25m$ and $u_{LV001} = 0.5159$.

By extracting a sub matrix from the matrix ψ, which corresponds to the data points within the hollow rectangle, we can find the amount of data points representing the different color codes in that interval. This gives a numerical interpretation of how the area within the hollow rectangle for the figures in Figure 3.6 is divided. These results are presented in Table 3.6.

Case	**Data points**
Quadratic >>	0
Quadratic >	579
Linear >	241
Linear >>	0
N/A	0

(a) Data points in sub matrix of ψ from Figure 3.6a.

Case	**Data points**
Cubic >>	0
Cubic>	509
Quadratic>	311
Quadratic>>	0
N/A	0

(b) Data points in sub matrix of ψ from Figure 3.6b.

Case	**Data points**
$5th$ order >>	0
$5th$ order>	388
Cubic>	432
Cubic>>	0
N/A	0

(c) Data points in sub matrix of ψ from Figure 3.6c.

Table 3.6: Numerical interpretation of the area within the hollow rectangles in Figure 3.6.

Table 3.6a shows that the quadratic model approximates the nonlinear model slightly better than the linear model at 579 different data points, which is 338 points more than the other way around. The entire hollow rectangle under consideration consists of 820 individual data points. This means that the quadratic model covers a total of $\frac{579 \cdot 100\%}{820} = 70.61\%$ of the area around the operating point. Given the fact that it requires an additional $3^2 = 9$ partial derivatives to obtain the $2nd$ order approximation from the $1st$ order approximation, it makes sense to discard the linear model for the more improved quadratic model.

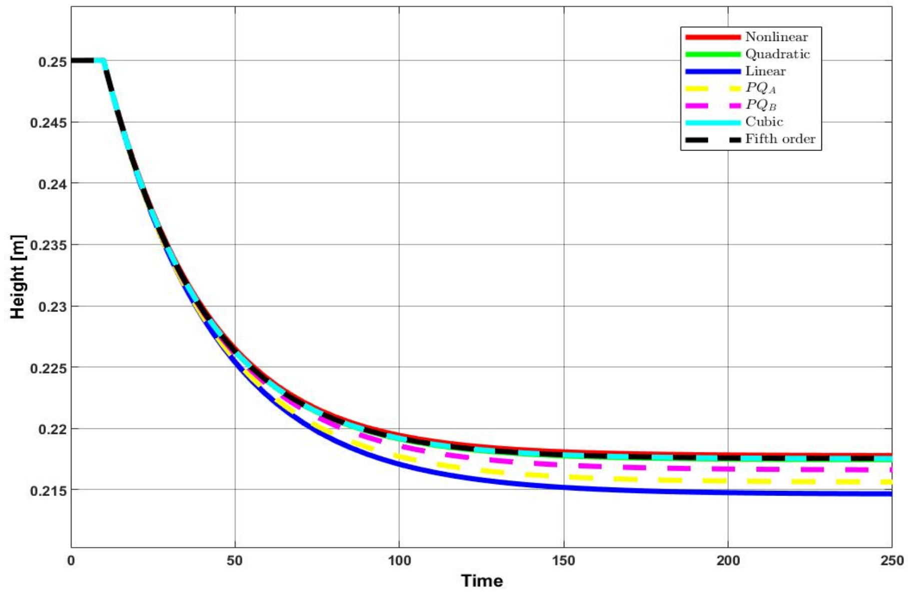
Following the same logic, the cubic model is slightly better than the quadratic model at 198 different data points more that the other way around. This can be calculated by the information in Table 3.6b. The cubic model covers a total of $\frac{509 \cdot 100\%}{820} = 62.07\%$ of the area within the hollow rectangle. This improvement is significant enough to discard the quadratic model for the cubic model, even tough it requires an additional $3^3 = 27$ partial derivatives to obtain.

From Table 3.6c we see that, in contrast to the previous comparisons, the accuracy of the higher order Taylor approximation was not improved. The cubic model is actually slightly better than the $5th$ order approximation at 44 different data points more than the other way around. In this case, the cubic model covers $\frac{432 \cdot 100\%}{820} = 52.68\%$ of the area within the hollow rectangle. Given this information, it does *not* make sense to discard the cubic model for the higher order approximation.

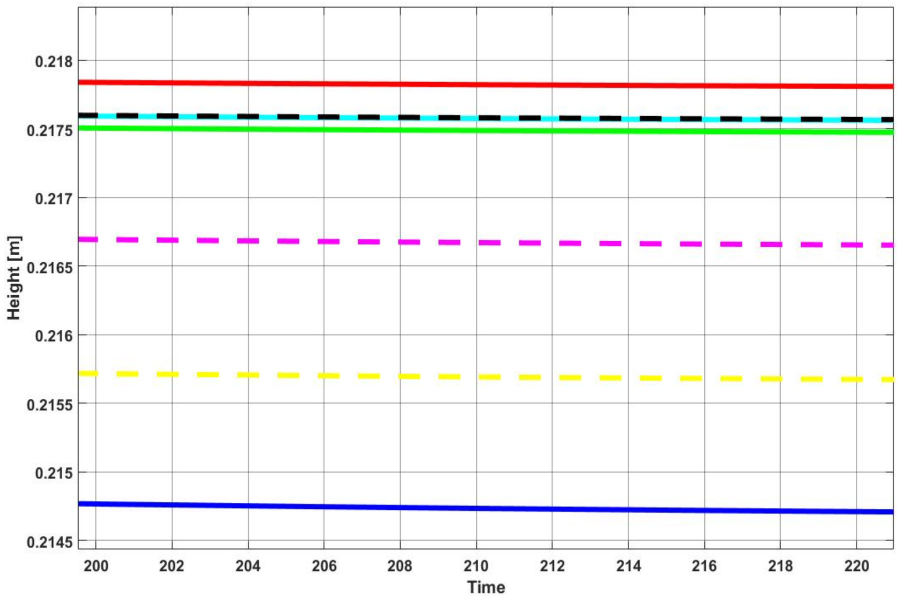
Note that none of the models were significantly better or worse in the immediate surroundings of the operating point. Every single data point within the hollow rectangle was either slightly in favour or slightly in disfavour of the models. There were also no cases where both models approximated the nonlinear model poorly in the set area around the operating point. Given these results, we can temporarily conclude that the cubic model is the best suited approximation of the nonlinear model.

It is possible to simulate the different Taylor models, and see how they respond to different input signals. In order to perform the simulations, the software Simulink and MATLAB are used. These simulations will be based on a *step response*, where the control signal $u_{LV001}(t)$ changes stepwisely. The control signal $u_{PA001}(t)$ will be constant at the operating point 0.65.

Figure 3.7 shows the step response of the different Taylor models together with the nonlinear model. The simulation lasted for a total of 250 seconds, and the step in $u_{LV001}(t)$ happened at 10 seconds. The control signal $u_{LV001}(t)$ went from its operating point of 0.5159 to 0.5359, which is an increment of 0.02.



(a) Step response with an increment in u_{LV001} of 0.02 at 10 seconds.



(b) Close up of 3.7a.

Figure 3.7: Model comparison in Simulink.

A standardised method to compare the performance of models and controllers is integrating the error over a time period. The *Integral performance criteria* is a set of commonly used integrals which quantify the error in a model or controller. The different integrals emphasise on different aspects of the error, for instance **ITAE** penalises errors more as time passes.

IAE - Integral Absolute Error: IAE is a commonly used error integral to compare errors over an interval.

$$IAE = \int_0^t \left|e(t)\right| dt$$

It integrates the absolute value of the error to make sure that positive and negative error do not cancel out. **IAE** does not add weight to any of the errors, it penalises small and large errors equally.

ISE - Integral Square Error:

$$ISE = \int_0^t e(t)^2 dt$$

The **ISE** penalises larger errors more than small errors.

ITAE- Integral Time Absolute Error:

$$ITAE = \int_0^t t\left|e(t)\right| dt$$

The **ITAE** penalises errors occurring at a later time more than at the start.

ITSE - Integral Time Square Error:

$$ITSE = \int_0^t te(t)^2 dt$$

The **ITSE** penalises large errors more than small, it also penalises errors at a later time more than at the start.

ISTE- Integral Square Time Error:

$$ISTE = \int_0^t t^2 e(t)^2 dt$$

The **ISTE** penalises large errors more than small, it also penalises errors at a later time more than at the start and it penalises more as time passes.

Table 3.7 quantifies the performance of the different models in the step response.

	IAE	**ISE**	**ITAE**	**ITSE**	**ISTE**
Linear	0.54638	0.001469	85.6911	0.24681	45.4302
PQ$_A$	0.39043	0.00073252	60.1709	0.12058	21.9437
PQ$_B$	0.20247	0.00020149	31.7901	0.034027	6.2837
Quadratic	0.066824	2.001e-05	9.7707	0.0031159	0.5532
Cubic	0.052571	1.2066e-05	7.4364	0.0017891	0.30999
5th order	0.051707	1.1652e-05	7.2912	0.0017189	0.29703

Table 3.7: Integral performance criteria.

It is evident that the cubic model, alongside with the $5th$ order model, are the approximations describing the nonlinear function the best. This agrees with previous results. Applying the same comparisons to *Scenario 2* leads to the same conclusion, for which reason the cubic model is chosen as the starting point for the *Carleman linearization*, which is described in the following chapter.

An important remark regarding the two scenarios, is that the approximations behave more similarly among them for *Scenario 2* than for *Scenario 1*. From Figure 3.3, we can see that the degree of nonlinearity for the nonlinear model is bigger at 0.25m than at 0.75m. This means that the nonlinear properties arising as the order of the approximation increases have a smaller impact on the results.

Chapter 4

Modelling using the Carleman embedding

In the previous chapter, it was shown that the following model:

$$
\begin{aligned}
\delta\dot{h}_1(t) =& \left.\frac{\partial f}{\partial h_1}\right|_O \delta h_1(t) + \left.\frac{\partial f}{\partial u_{LV001}}\right|_O \delta u_{LV001}(t) + \left.\frac{\partial f}{\partial u_{PA001}}\right|_O \delta u_{PA001}(t) \\
&+ \frac{1}{2}\left.\frac{\partial^2 f}{\partial h_1^2}\right|_O \delta h_1^2(t) + \left.\frac{\partial^2 f}{\partial h_1 \partial u_{LV001}}\right|_O \delta h_1(t)\,\delta u_{LV001}(t) \\
&+ \frac{1}{6}\left.\frac{\partial^3 f}{\partial h_1^3}\right|_O \delta h_1^3(t) + \frac{1}{2}\left.\frac{\partial^3 f}{\partial h_1^2\,\partial u_{LV001}}\right|_O \delta h_1^2(t)\,\delta u_{LV001}(t)
\end{aligned}
\tag{4.1}
$$

was the best approximation of the nonlinear tank model. This approximation will be the starting point of the *Carleman embedding* presented in this chapter. Section 4.1 introduces the key theoretical concepts related to the Carleman embedding and its practical use. Section 4.2 derives a general expression for the *Carleman approximation* of the cubic model. Section 4.3 presents the results of the Carleman approximation of the cubic model truncated at different n.

4.1 Carleman embedding

The general idea of Carleman embedding is that a finite dimensional set of nonlinear differential equations, can be embedded into an infinite dimensional set of linear differential equations. More specifically, a polynomial or analytical [1] model defined on a finite dimensional space, can be transformed into a linear or bilinear model on an infinite dimensional space [12] [16].

The following finite dimensional nonlinear differential equation (4.2), with V consisting of polynomials in x, can be embedded into the infinite dimensional linear differential equation (4.3):

$$\dot{x}(t) = V(x(t)) \tag{4.2}$$

[1] By analytical model we mean a model where its Taylor series converges to the model itself at every point in a defined area around the working point. This means that we can make a non-polynomial model into a polynomial by means of *Taylor approximation* as seen in Section 3.2.

$$\dot{z}(t) = A\, z(t) \tag{4.3}$$

by using an infinite vector of state variables: $z(t) = (z_1(t), z_2(t), \cdots)^T$.

(4.3) is *the infinite linear model associated to the finite nonlinear model* given in (4.2). This is further explained in [14].

4.1.1 Carleman embedding technique

The *Carleman embedding technique* is introduced with an example.

Consider the following system:

$$\dot{x}(t) = x(t) + x^2(t) + x^3(t) \tag{4.4}$$

The goal is to describe this system with a linear state-space representation, where the state variables are defined as the infinite sequence of functions $z = (z_1, z_2, \cdots)^T$. This can be achieved by expanding the state vector with the monomials corresponding to the state variable z:

$$\begin{aligned} z_1(t) &= x(t) \\ z_2(t) &= x^2(t) \\ z_3(t) &= x^3(t) \\ &\vdots \\ z_n(t) &= x^n(t) \\ &\vdots \end{aligned}$$

By differentiating the expanded state variables, we obtain:

$$\begin{aligned} \dot{z}_1(t) &= \dot{x}(t) = x(t) + x^2(t) + x^3(t) = z_1(t) + z_2(t) + z_3(t) \\ \dot{z}_2(t) &= 2\,x(t)\,\dot{x}(t) = 2\,x^2(t) + 2\,x^3(t) + 2\,x^4(t) = 2\,z_2(t) + 2\,z_3(t) + 2\,z_4(t) \\ \dot{z}_3(t) &= 3\,x^2(t)\,\dot{x}(t) = 3\,x^3(t) + 3\,x^4(t) + 3\,x^5(t) = 3\,z_3(t) + 3\,z_4(t) + 3\,z_5(t) \\ &\vdots \\ \dot{z}_n(t) &= n\,x^{n-1}(t)\,\dot{x}(t) = n\,x^n(t) + n\,x^{n+1}(t) + n\,x^{n+2}(t) = n\,z_n(t) + n\,z_{n+1}(t) + n\,z_{n+2}(t) \end{aligned}$$

which can be put into the compact form (4.3) with state matrix A given by:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & \cdots & & & \\ 0 & 2 & 2 & 2 & 0 & \cdots & & \\ \vdots & 0 & 3 & 3 & 3 & 0 & \cdots & \\ \vdots & \vdots & 0 & 4 & 4 & 4 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{4.5}$$

4.1.2 Truncation

As (4.3) is an infinite set of linear differential equations, numerical calculations using the result become nontrivial. A common take on this problem is truncating the Carleman embedding at the *nth* state variable. This results in the *Carleman approximation*, which will become more accurate as n increases, disregarding computational errors.

Truncating (4.5) at $n = 3$ results in:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 3 \end{bmatrix}$$

Truncating (4.5) at $n = 5$ results in:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

while truncating at a generic n results in the $n \times n$-matrix:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 2 & 2 & 2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & n-1 & n-1 \\ \vdots & \vdots & \ddots & \ddots & 0 & n \end{bmatrix}$$

The truncation leads to approximations that become more accurate as n increases. A comparison between the nonlinear system (4.4) and some Carleman approximations are presented in Figure 4.1. The comparison shows that in this case, the Carleman approximation leads to an adequate approximation at $n = 3$.

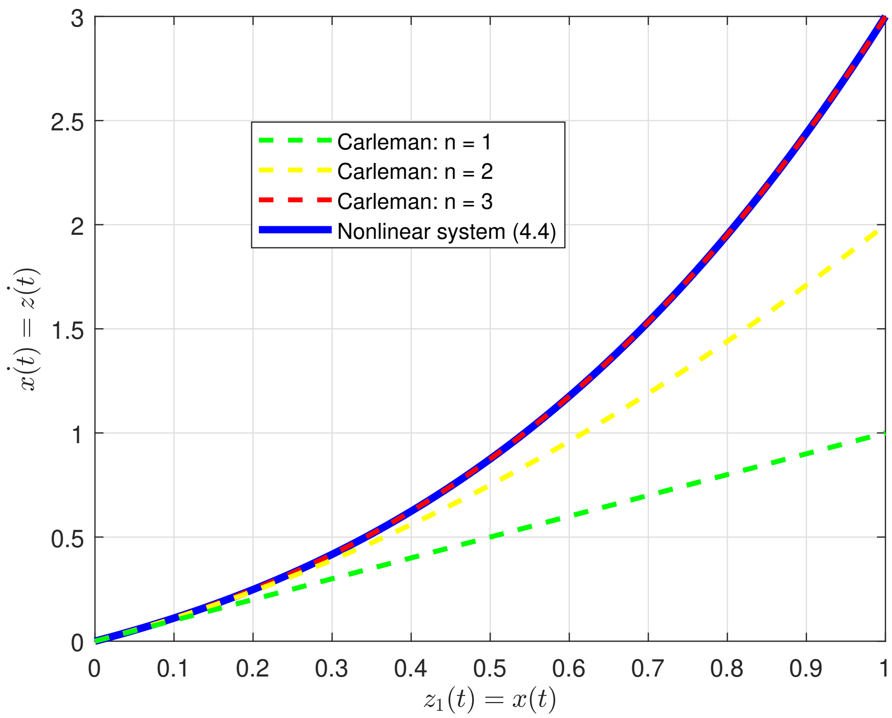


Figure 4.1: Carleman approximations of (4.4) compared to the nonlinear system.

4.2 Carleman approximation of the cubic tank model

The cubic tank model (4.1) is the starting point of the Carleman approximation. Applying the Carleman embedding technique shown in the previous section to (4.1), results in a nonlinear quadratic system. By means of the $1st$ order Taylor approximation, the nonlinear quadratic system is linearized. The *linearized Carleman approximation* is the state-space representation truncated at the nth state variable.

Let us introduce the following compact notation for the partial derivatives appearing in (4.1):

$$\begin{aligned}
a &= \frac{\partial f}{\partial h_1}\bigg|_O \\
b &= \frac{\partial f}{\partial u_{LV001}}\bigg|_O \\
c &= \frac{1}{2}\frac{\partial^2 f}{\partial h_1^2}\bigg|_O \\
d &= \frac{\partial^2 f}{\partial h_1\,\partial u_{LV001}}\bigg|_O \\
e &= \frac{1}{6}\frac{\partial^3 f}{\partial h_1^3}\bigg|_O \\
f &= \frac{1}{2}\frac{\partial^3 f}{\partial h_1^2\,\partial u_{LV001}}\bigg|_O
\end{aligned} \tag{4.6}$$

The Carleman embedding of (4.1) is derived from the following steps:

- Expand the state vector with the monomials corresponding to the state variable

  $z(t) = (z_1(t), z_2(t), \cdots)^T$:

$$\begin{aligned}
z_1(t) &= \delta h_1(t) \\
z_2(t) &= \delta h_1^2(t) \\
z_3(t) &= \delta h_1^3(t) \\
&\vdots \\
z_n(t) &= \delta h_1^n(t) \\
&\vdots
\end{aligned}$$

- Differentiate the expanded state variables:

$$\begin{aligned}
\dot{z}_1(t) = \delta\dot{h}_1(t) &= a\,\delta h_1(t) + b\,\delta u_{LV001}(t) + c\,\delta h_1^2(t) + d\,\delta h_1(t)\,\delta u_{LV001}(t) + e\,\delta h_1^3(t) \\
&\quad + f\,\delta h_1^2(t)\,\delta u_{LV001}(t) = \\
&= a\,z_1(t) + b\,\delta u_{LV001}(t) + c\,z_2(t) + d\,z_1(t)\,\delta u_{LV001}(t) + e\,z_3(t) + f\,z_2(t)\,\delta u_{LV001}(t)
\end{aligned}$$

$$\begin{aligned}
\dot{z}_2(t) = 2\,\delta h_1(t)\,\delta\dot{h}_1(t) &= 2\,a\,\delta h_1^2(t) + 2\,b\,\delta h_1(t)\,\delta u_{LV001}(t) + 2\,c\,\delta h_1^3(t) \\
&\quad + 2\,d\,\delta h_1^2(t)\,\delta u_{LV001}(t) + 2\,e\,\delta h_1^4(t) + 2\,f\,\delta h_1^3(t)\,\delta u_{LV001}(t) = \\
&= 2\,a\,z_2(t) + 2\,b\,z_1(t)\,\delta u_{LV001}(t) + 2\,c\,z_3(t) + 2\,d\,z_2(t)\,\delta u_{LV001}(t) + 2\,e\,z_4(t) \\
&\quad + 2\,f\,z_3(t)\,\delta u_{LV001}(t)
\end{aligned}$$

$$
\begin{aligned}
\dot{z}_3(t) &= 3\,\delta h_1^2(t)\,\delta \dot{h}_1(t) = \\
&= 3\,a\,z_3(t) + 3\,b\,z_2(t)\,\delta u_{LV001}(t) + 3\,c\,z_4(t) + 3\,d\,z_3(t)\delta u_{LV001}(t) + 3\,e\,z_5(t) \\
&\quad + 3\,f\,z_4(t)\,\delta u_{LV001}(t) \\
&\;\vdots \\
\dot{z}_n(t) &= n\,\delta h_1^{n-1}(t)\,\delta \dot{h}_1(t) = \\
&= n\,a\,z_n(t) + n\,b\,z_{n-1}(t)\,\delta u_{LV001}(t) + n\,c\,z_{n+1}(t) + n\,d\,z_n(t)\delta u_{LV001}(t) + n\,e\,z_{n+2}(t) \\
&\quad + n\,f\,z_{n+1}(t)\,\delta u_{LV001}(t)
\end{aligned}
$$

The Carleman embedding results in a nonlinear quadratic system in the form:

$$
\dot{z}(t) = A\,z(t) + B\,\delta u_{LV001}(t) + \begin{bmatrix} z(t)^T\,E_1 \\ z(t)^T\,E_2 \\ \vdots \\ z(t)^T\,E_n \end{bmatrix} \delta u_{LV001}(t) \tag{4.7}
$$

where $z(t)$ is the system state, $\delta u_{LV001}(t)$ is the control input and:

$$
A = \begin{bmatrix}
a & c & e & 0 & \cdots & & \\
0 & 2a & 2c & 2e & 0 & \ddots & \\
\vdots & 0 & 3a & 3c & 3e & 0 & \ddots \\
\vdots & \vdots & 0 & 4a & 4c & 4e & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix}
$$

$$
B = \begin{bmatrix}
b & 0 & \cdots \\
0 & 0 & \cdots \\
\vdots & \vdots & \cdots
\end{bmatrix}
$$

$$
E_1 = \begin{bmatrix} d \\ f \\ 0 \\ \vdots \end{bmatrix}
E_2 = \begin{bmatrix} 2b \\ 2d \\ 2f \\ 0 \\ \vdots \end{bmatrix}
E_3 = \begin{bmatrix} 0 \\ 3b \\ 3d \\ 3f \\ 0 \\ \vdots \end{bmatrix} \cdots
$$

The linearized version of the embedded cubic model can be found by neglecting the nonlinear term $[z(t)^T E_1, \cdots, z(t)^T E_n]^T \delta u_{LV001}(t)$. If the approximation is truncated at $n = 1$, then this corresponds to the *Taylor linearization* about the equilibrium point. This leads to the *Linear Time Invariant*(LTI) state-space representation:

$$
\begin{aligned}
\dot{z}(t) &= A\,z(t) + B\,\delta u_{LV001}(t) \\
y(t) &= C\,z(t)
\end{aligned}
$$

where the *output matrix* is given by:

$$C = \begin{bmatrix} 1 & 0 & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots \\ \vdots & \vdots & \cdots & \ddots \end{bmatrix}$$

This representation is the *infinite linear model associated to the finite nonlinear cubic model* (4.1).

The linearized Carleman approximation is the state-space representation truncated at the nth state variable. Comparison between models truncated at different values of n will be discussed in the next section.

4.3 Carleman approximation comparison

In Section 4.2, we derived a general expression for the nth order linearized Carleman approximation of the cubic model. In this section, the goal is to determine the accuracy of this approximation and to find a value of n at which the truncation can happen without losing relevant information about the process' dynamics. In order to do this, we will use Simulink as a tool to simulate different linearized Carleman approximations of the cubic model, where n will differ. As with the previous chapter, we consider the two operating points named *Scenario 1* and *Scenario 2*. This section will present and discuss the results related to *Scenario 1*. The reader is referred to Appendix B for the results related to *Scenario 2*.

In the Simulink library browser, there is a block named *State-Space*, which will be used to simulate the different linearized Carleman approximations. This block uses the state matrix A, the input matrix B, the output matrix C and the feedthrough matrix D as parameters. How to obtain these matrices was explained in Section 4.2. The input is $\delta u_{LV001}(t)$ and the output is $\delta h_1(t)$. Figure 4.2 shows how this block is used in Simulink.

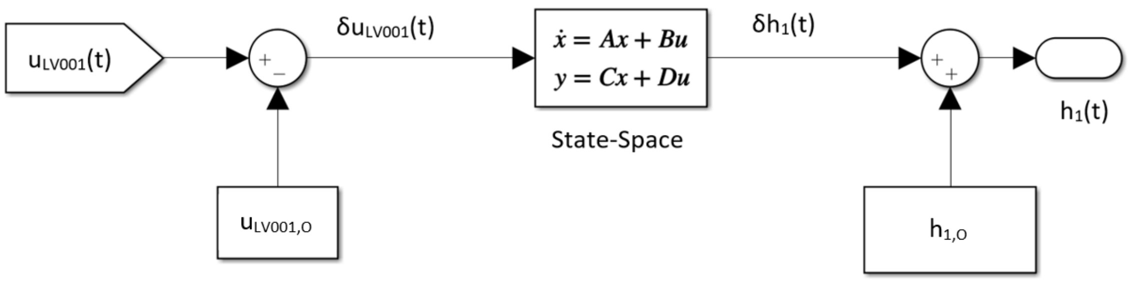


Figure 4.2: Utilising the State-Space block in Simulink.

This first simulation will be based on a *forced response*, where the control signal $u_{LV001}(t)$ changes stepwisely, and the initial condition is set to zero. The signal $u_{PA001}(t)$ will be kept constant at the value 0.65, which corresponds to the equilibrium point, as explained in Section 3.3. The total length of each simulation is 250 seconds. There will be a total of six different linearized Carleman approximations, where $n \in \{1, 2, 3, 4, 5, 10\}$. Figure 4.3 presents the results of this simulation.

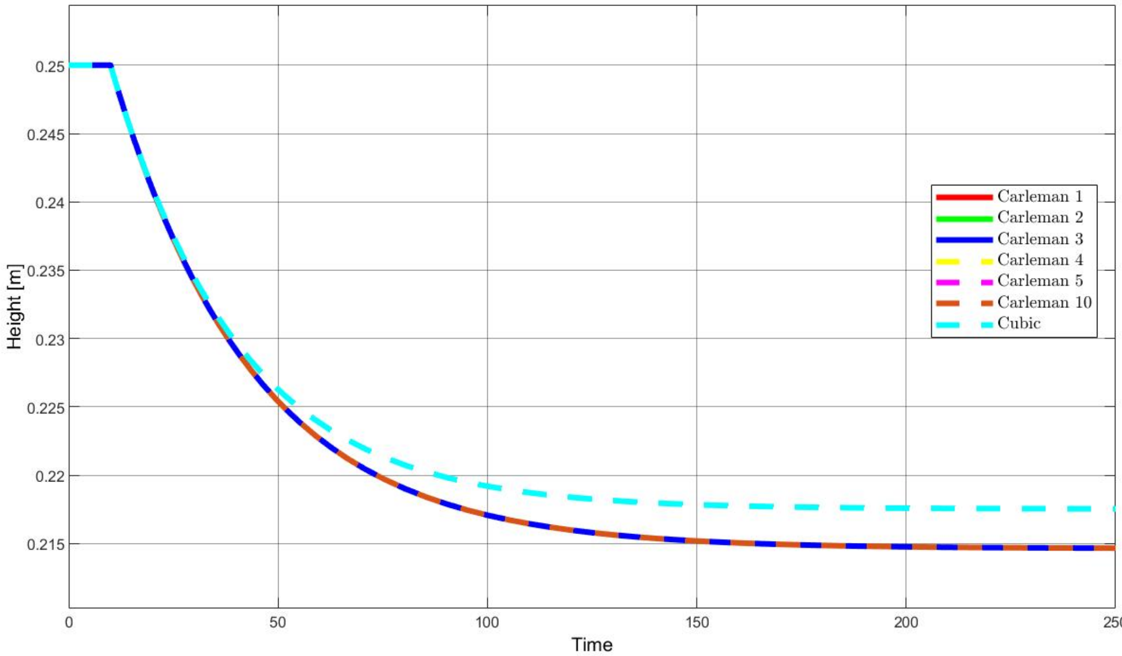


Figure 4.3: Step response with an increment in $u_{LV001}(t)$ of 0.02 at 10 seconds.

Figure 4.3 shows that all of the linearized Carleman approximations are exactly the same, and that they do not follow the cubic model. Upon further inspection, the linearized Carleman approximations actually follow the $1st$ order Taylor approximation. This can be seen by comparing the results from Figure 3.7a with the results from Figure 4.3. This comparison tells us that, when considering the contribution of the input signal to the overall response, the linearized Carleman approximations behave exactly as the linearized model obtained using a $1st$ order Taylor approximation, and that the order n is irrelevant. The cause for this behaviour will be further discussed in the next chapter.

Another way to observe the linearized Carleman approximations, is by simulating the *free response* of the models, where the input signal $u_{LV001(t)}$ is constant and the initial conditions are altered. If the initial condition is different from the operating point, the model will eventually converge to the operating point because the system is open-loop stable, otherwise it would not. By studying this convergence, it becomes possible to compare the cubic model with the linearized Carleman approximations. Note that the operating point for the cubic model is $h_{1,O} = 0.25m$ for *Scenario 1*. However, since the elements in the state vector $z(t)$ for the Carleman approximations is defined as $\delta h_1(t)^n$, the operating point is $0m$ when defined with respect to $\delta h_1(t)$ and its powers. This means that if the cubic model and the linearized Carleman approximations are meant to start from the same initial water level, they need to have an initial condition of $0.25m + \delta m$ and δm, respectively, where δm is the offset from the equilibrium points. Since the Carleman approximations consists of n states, it needs a $n \times 1$ matrix which includes the initial condition for each state:

$$\delta\_matrix = \begin{bmatrix} \delta^1 \\ \delta^2 \\ \vdots \\ \delta^n \end{bmatrix} \tag{4.8}$$

Defining the δ variable as 0.05m, means that the models will start at $0.25m + 0.05m = 0.3m$.

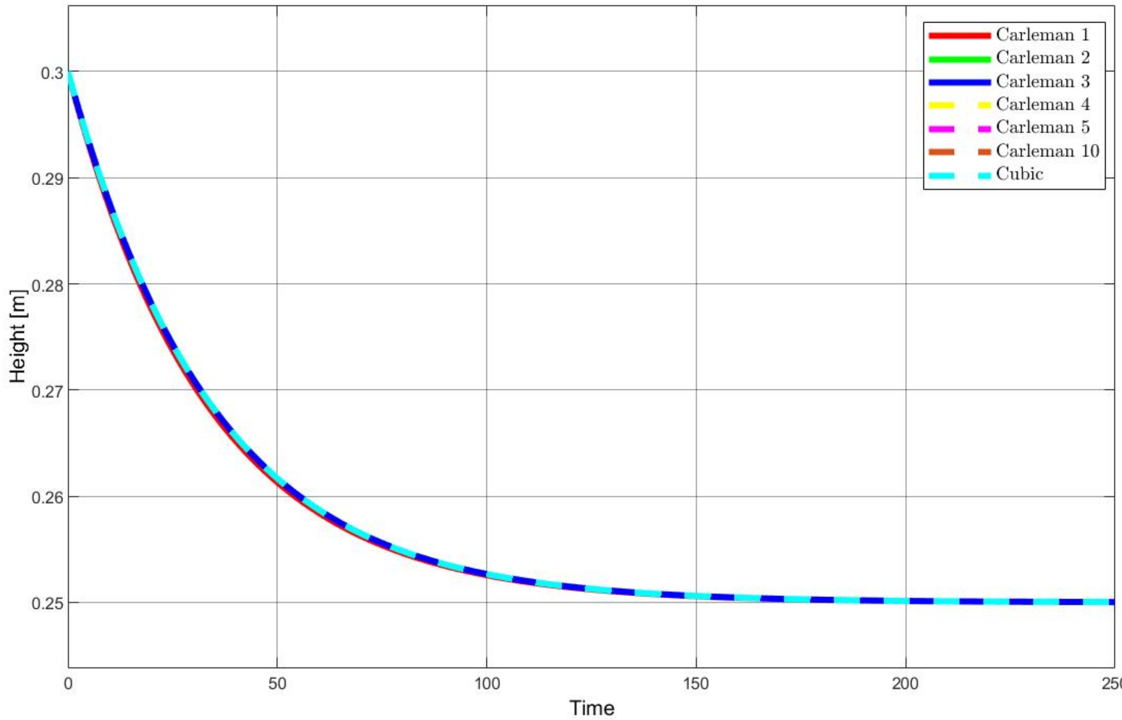


Figure 4.4: $\delta = 0.05m$ gives a starting point of 0.3m.

From Figure 4.4, it is evident that the linearized Carleman approximations do indeed follow the cubic model, and that they converge to the operating point at 0.25m. Section 3.3 introduced *integral performance criteria* as a method to compare the different Taylor approximations. Applying that same concept here, results in Table 4.1[2].

	IAE	**ISE**	**ITAE**	**ITSE**	**ISTE**
$Carleman\, 1$	0.033946	1.1314e-05	1.7255	0.00042025	0.021141
$Carleman\, 2$	0.0010657	1.9028e-08	0.032067	4.1273e-07	1.2094e-05
$Carleman\, 3$	9.2014e-05	8.8891e-11	0.0047623	3.4878e-09	1.7297e-07
$Carleman\, 4$	1.6684e-06	2.1323e-14	0.00010913	1.143e-12	8.1992e-11
$Carleman\, 5$	1.8461e-07	-1.0709e-15	8.7203e-06	-1.0774e-14	-1.2033e-13
$Carleman\, 10$	3.0644e-12	-1.4964e-15	1.5481e-10	-2.6411e-14	-8.1151e-13

Table 4.1: Integral performance criteria of the linearized Carleman approximations, with $\delta = 0.05m$.

Table 4.1 shows that around $n = 2$, the linearized Carleman approximation approximates the cubic model with small discrepancies. However, since the starting point only has a 0.05m deviation from the operating point, the behaviour required in order to follow the cubic model is less nonlinear. This allows a low order linearized Carleman approximation to have a high accuracy. Increasing the δ variable to 0.2m, gives a starting point of 0.45m.

[2]Due to the precision of numerical calculations in MATLAB, some values that are small appear as negative values. These values can be regarded as zero.

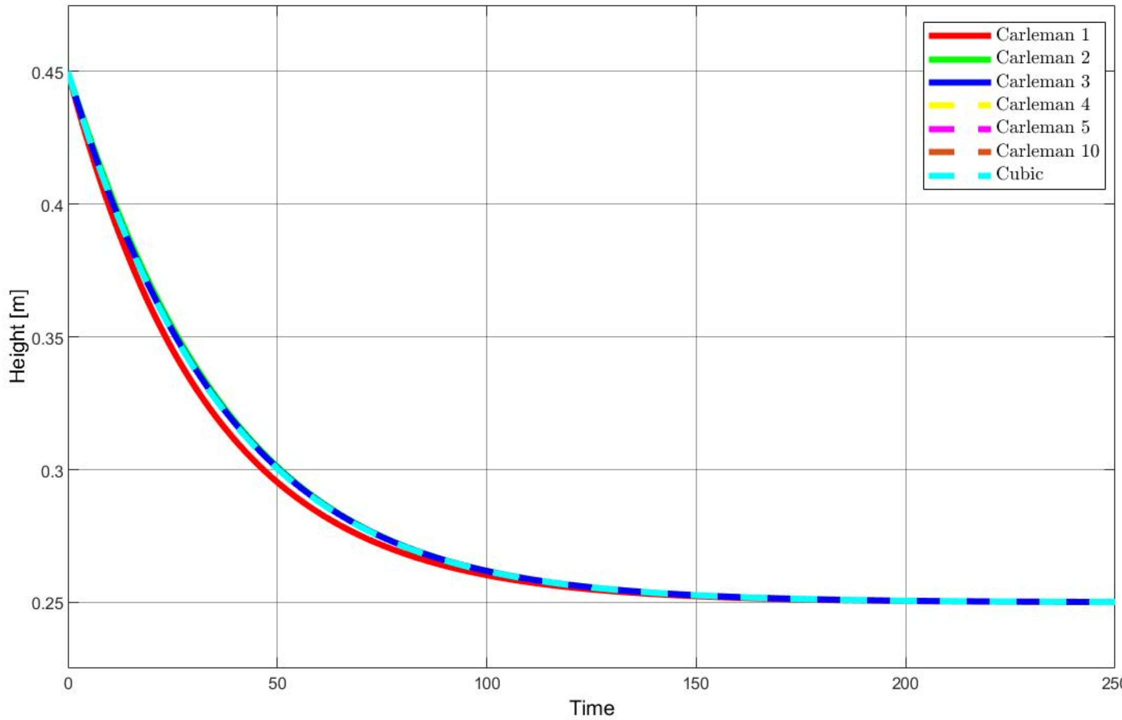


Figure 4.5: $\delta = 0.2m$ gives a initial condition of 0.45m.

From Figure 4.5 we see that the $1st$ order linearized Carleman approximation is not able to follow the cubic model as well as the other models. Table 4.2 also expresses this notion.

	IAE	**ISE**	**ITAE**	**ITSE**	**ISTE**
$Carleman\,1$	0.47434	0.0021312	25.1139	0.084099	4.4267
$Carleman\,2$	0.085848	0.00010673	3.0072	0.0026045	0.087056
$Carleman\,3$	0.023531	5.6366e-06	1.2597	0.00023117	0.011911
$Carleman\,4$	0.0015259	2.259e-08	0.089447	8.2185e-07	5.1763e-05
$Carleman\,5$	0.00084781	8.4265e-09	0.0425	3.2894e-07	1.5478e-05
$Carleman\,10$	2.9821e-07	-8.067e-15	1.5674e-05	-1.8252e-13	-5.7072e-12

Table 4.2: Integral performance criteria of the linearized Carleman approximations, with $\delta = 0.2m$.

Table 4.2 shows that the linearized Carleman approximations of lower order, are significantly less accurate. In this case, the preferred models are Carleman 4, 5 and 10. Obviously, increasing the order above 10 will make the approximation more accurate. However, since the level of improvement from $n = 5$ to $n = 10$ is so small, considering $n > 10$ is redundant with respect to this project.

It was shown that the linearized Carleman approximation only follows the $1st$ order Taylor approximation when there is a change in the control signal $u_{LV001}(t)$. This behaviour makes the linearized Carleman approximation inadequate for the cubic model. Because of this, the quadratic Carleman approximation (4.7), obtained in Section 4.2 will be considered.

To simulate the quadratic Carleman approximations, we use the same settings as in the previous step response shown in Figure 4.3. $u_{LV001}(t)$ changes stepwisely with an increment of 0.02 at 10 seconds, $u_{PA001}(t)$ is kept constant at 0.65 and the total simulation length is 250 seconds. There will be a total of six different quadratic Carleman approximations, where $n \in \{1, 2, 3, 4, 5, 10\}$.

Figure 4.6 shows that the quadratic Carleman approximations follow the cubic model to a great extent, with an exception of the $1st$ order quadratic Carleman approximation. Given that the change in $u_{LV001}(t)$ is only 0.02 away from the operating point, it is expected that the order n required to follow the cubic model is small. However, if the step in $u_{LV001}(t)$ is bigger in magnitude, it will be harder for the quadratic Carleman approximations of lower order to follow the cubic model. Figure 4.7 shows an equivalent simulation, but the step in $u_{LV001}(t)$ is now -0.1.

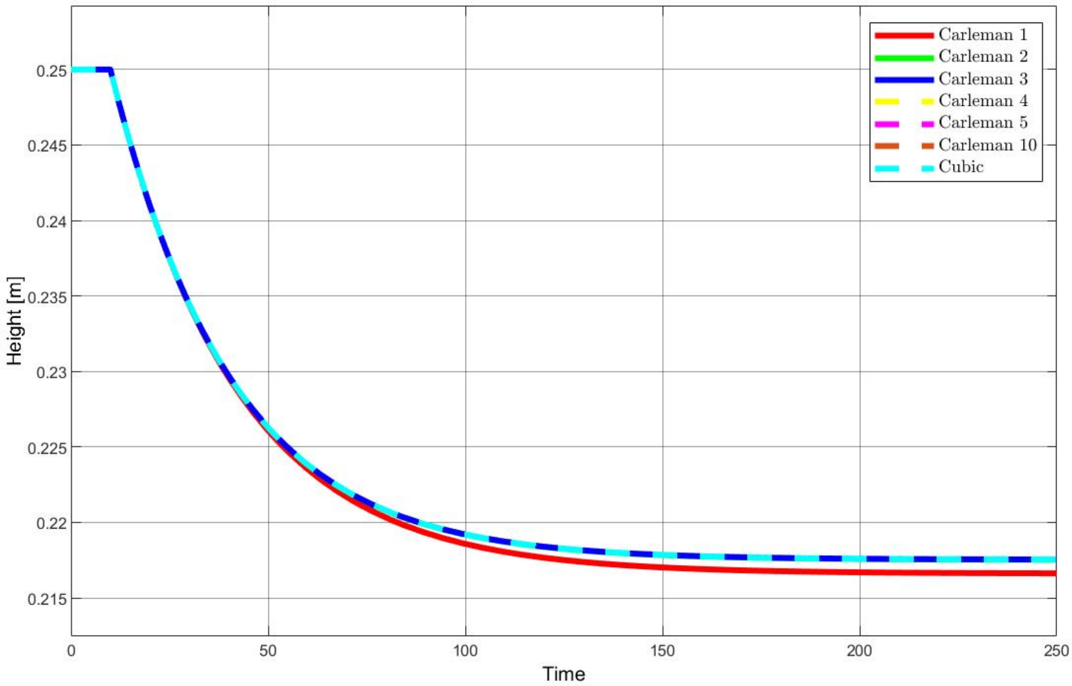


Figure 4.6: Simulation of the quadratic Carleman approximations.

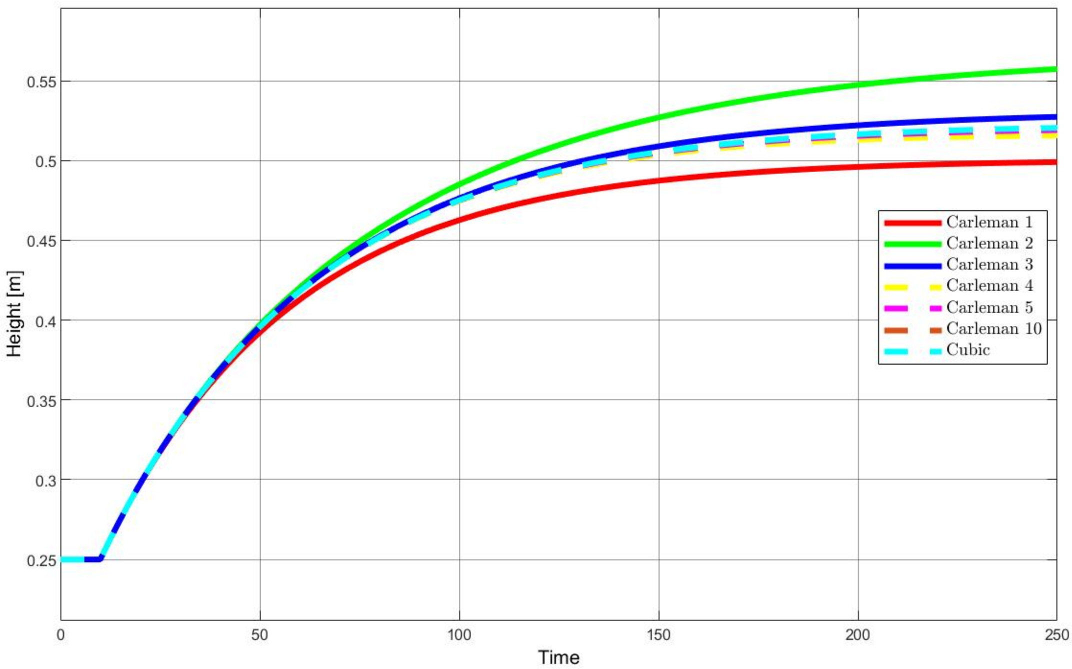


Figure 4.7: Simulation of the quadratic Carleman approximations with a larger step.

From Figure 4.7, we see again that the preferred order for the quadratic Carleman approximations are $n \in \{4, 5, 10\}$. This shows that the Carleman embedding technique from Section 4.2 works as intended, and that the nonlinear quadratic system, truncated at $n \in \{4, 5, 10\}$, gives an accurate representation of the cubic model.

Part III

Control

Chapter 5

State feedback control using the linear Carleman approximation

In the previous chapter, we saw that the linearized Carleman approximation did not follow the cubic model when a forced response was applied. This behaviour will be further investigated in this chapter. Section 5.1 gives a brief introduction to some relevant elements from control theory, while Section 5.2 further investigates the *controllability* of the linearized Carleman approximation.

5.1 Control

Control of a dynamical system involves using the system inputs to drive it to a desired state with some prescribed performance [20]. The controller for the Carleman approximations will be a state feedback controller. This means that the control input is dependent on the current value of the state, which is "fed back" to the controller. The current state is compared to the reference as the goal is to move the system back to the equilibrium point. This gives the measured error that the controller has to correct.

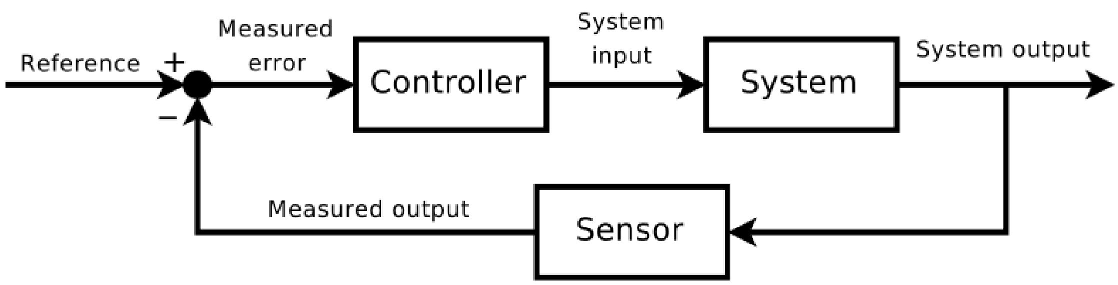


Figure 5.1: Feedback control system [1].

Controllability is a key property of a system which describes its ability to reach any point in the state-space by applying the correct sequence of control inputs.

Theorem 1 *The LTI system:*

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{5.1}$$

is controllable if and only if the controllability matrix:

$$\mathcal{C} = [B, AB, A^2B, \cdots, A^{n-1}B] \tag{5.2}$$

has full row rank.

The column rank of a matrix is determined by the number of linearly independent columns of the matrix. Equivalently the row rank is the number of dimensions of the vector space spanned by its rows. A fundamental result in linear algebra shows that the column rank and row rank are always equal [23]. Therefore, either can be calculated to check the rank of the controllability matrix.

Theorem 2 *The $n \times nm$ matrix $\mathcal{C}$ has full row rank if and only if:*

$$rank(\mathcal{C}) = min(n, nm) = n \tag{5.3}$$

Consider the following system:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 1 & 0 & 3 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

then the controllability matrix is given by:

$$\mathcal{C} = \begin{bmatrix} 1 & 2 & 5 \\ 0 & 2 & 6 \\ 0 & 1 & 5 \end{bmatrix}$$

To examine the rank of the controllability matrix, we first have to make sure that the columns are *linearly independent.* Therefore, a common approach to finding the rank is to perform *Gaussian elimination* on the matrix to get it in a simple *row echelon form.* This way, all the dependent rows become 0 and the *rank* is equal to the number of non-zero rows remaining.
The $\mathcal{C}$ matrix in *row echelon* form:

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

clearly shows that there are 3 non-zero rows giving it $rank = 3$, revealing that the system is controllable.

5.2 Controllability of the linear Carleman approximation

In order to find the controllability property of the linearized Carleman approximation, the state matrix A, and the input matrix B is required. The general expression for these matrices was derived in Section 4.2, and are presented below:

$$A = \begin{bmatrix} a & c & e & 0 & \cdots & & \\ 0 & 2a & 2c & 2e & 0 & \ddots & \\ \vdots & 0 & 3a & 3c & 3e & 0 & \ddots \\ \vdots & \vdots & 0 & 4a & 4c & 4e & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, B = \begin{bmatrix} b & 0 & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \cdots \end{bmatrix} \tag{5.4}$$

With the equations (5.2) and (5.4), the controllability matrix for the linearized Carleman approximation can be calculated. As an example, we can truncate the linearized Carleman approximation at $n = 3$, which gives the following A and B matrices:

$$A = \begin{bmatrix} a & c & e \\ 0 & 2a & 2c \\ 0 & 0 & 3a \end{bmatrix}, B = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \tag{5.5}$$

Inserting the A and B matrices from (5.5) into (5.2) yields:

$$\mathcal{C} = \begin{bmatrix} \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} a & c & e \\ 0 & 2a & 2c \\ 0 & 0 & 3a \end{bmatrix} \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} a & c & e \\ 0 & 2a & 2c \\ 0 & 0 & 3a \end{bmatrix}^2 \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} b & ab & a^2b \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.6}$$

If the calculations are repeated for a higher order system, e.g., for $n = 6$, one gets:

$$A = \begin{bmatrix} a & c & e & 0 & 0 & 0 \\ 0 & 2a & 2c & 2e & 0 & 0 \\ 0 & 0 & 3a & 3c & 3e & 0 \\ 0 & 0 & 0 & 4a & 4c & 4e \\ 0 & 0 & 0 & 0 & 5a & 5c \\ 0 & 0 & 0 & 0 & 0 & 6a \end{bmatrix}, B = \begin{bmatrix} b \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.7}$$

As in the previous example, the matrices in (5.7) are inserted into Eq. (5.2):

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & A^3B & A^4B & A^5B \end{bmatrix} = \begin{bmatrix} b & ab & a^2b & a^3b & a^4b & a^5b \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.8}$$

It is clear that for a generic nth order linearized Carleman approximation, the controllability matrix is given by:

$$\mathcal{C} = \begin{bmatrix} b & ab & a^2b & \dots & a^{n-1}b \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{5.9}$$

Note that all of the elements $[A_{2,1}, A_{3,1}, \dots, A_{n,1}]$ in the A^{n-1} matrix are all equal to zero, with the exception of $A_{1,1}$ which is equal a^{n-1}. Note also that the only non-zero element in the B matrix is $B_{1,1}$, which is equal to b. This explains why the multiplication $A^{n-1}B$ always results in a column vector, where the only non-zero element is the first entry. This is the reason why the first row in (5.9) is the only non-zero row of the controllability matrix.

Given (5.9), it is evident that the order of the linearized Carleman approximation will not affect the fact that the first row is the only non-zero row of the controllability matrix. In Section 5.1, it was stated that the rank of the controllability matrix needed to be equal to n, in order for the system to be controllable. It is evident from (5.9) that the rank of the controllability matrix will always be equal to 1, regardless of the order n of the linearized Carleman approximation. This means that the linearized Carleman approximation will not be controllable unless the order $n = 1$, which is equal to the $1st$ order Taylor approximation of the nonlinear system. The lack of controllability means that the system's eigenvalues can not be arbitrarily located in the complex plane, which means that some eigenvalues can not be reached [26]. This explains the behaviour observed in Figure 4.3, and why all of the linearized Carleman approximations follow the $1st$ order Taylor approximation.

Chapter 6

State feedback control using the quadratic Carleman approximation

In the previous chapter we saw that the linearized Carleman approximation was not controllable and that the forced response followed the $1st$ order Taylor linearization, regardless of the order n. In this chapter, we will consider the quadratic system arising from the Carleman approximation of the cubic model. In Section 6.1, some introductory theory is presented to better grasp the process of making a state feedback controller for a quadratic system. Section 6.2 describes the design of the controller gain $\mathcal{K}$ for the *quadratic Carleman approximation*. In Section 6.3, necessary modifications of LMIs are implemented to include the pump $PA001$ as an actuator, in addition to the valve $LV001$, resulting in a new controller gain $\mathcal{K}$. The results presented in this chapter are related to *Scenario 1*. For the results related to *Scenario 2*, the reader is referred to Appendix C.

6.1 Supplemental theory

6.1.1 Polytopes

A *polytope* is a geometric object, with flat surfaces and straight edges. Convex polytopes are the simplest kind of polytopes, and they are defined as the intersection of a set of half-spaces [22]. The actual points at which these half-spaces intersect and form the corners of the convex polytopes are defined as the *vertices* of the polytope [25].

It is possible to create a convex polytope by applying the *convex hull* on a finite set of points. A convex hull is defined as the minimal convex set that contains all of the points [21]. This hull creates a convex polytope, where each extreme point of the hull is a vertex. It is also possible to express the half-spaces, containing the polytope, if the vertices are known. In this project, this was done by using the `vert2con` MATLAB function, which requires the vertices as an input argument, and returns a set of constraints such that $Ax \leq b$ defines the region of space enclosing the convex hull of the given points [11].

A convex polytope, in $\mathbb{R}^2$, in the shape of a box is given by: $\mathcal{P} = [-1, 2] \times [-1, 3]$. The points at which these four half-spaces intersect are:

$$x_{(1)} = (2, -1)^T \,,\, x_{(2)} = (2, 3)^T \,,\, x_{(3)} = (-1, 3)^T \,,\, x_{(4)} = (-1, -1)^T \tag{6.1}$$

which are the vertices of the convex polytope.

Using the vertices from (6.1) as an input argument in the `vert2con` function, gives the following inequalities:

$$Ax \leq b,\; A = \begin{bmatrix} 0 & 0.5000 \\ 0 & -0.5000 \\ 0.6667 & 0 \\ -0.6667 & 0 \end{bmatrix},\; b = \begin{bmatrix} 1.5000 \\ 0.5000 \\ 1.3333 \\ 0.6667 \end{bmatrix} \tag{6.2}$$

Usually, the constraints that expresses the region of the polytope are described as:

$$Ax \leq 1 \tag{6.3}$$

Therefore, we divide both sides of (6.2) by b. This results in a matrix denoted as A_k:

$$A_k = A \oslash b^{=} \begin{bmatrix} 0 & 0.3333 \\ 0 & -1.0000 \\ 0.5000 & 0 \\ -1.0000 & 0 \end{bmatrix} \rightarrow A_k x \leq 1 \tag{6.4}$$

where each *row vector* in A_k represents the transposed of the half-space vector that contains the polytope. These vectors can be written as:

$$a_1^T = (0, \frac{1}{3}) \,,\, a_2^T = (0, -1) \,,\, a_3^T = (\frac{1}{2}, 0) \,,\, a_4^T = (-1, 0) \tag{6.5}$$

Plotting these half-spaces shows the points at which they intersect, and the region in which they cut off. This contained region is known as the polytope $\mathcal{P}$. Figure 6.1 illustrates this notion.
In fact, the inequality in (6.4) can be segmented into the following inequalities:

$$a_1^T x \leq 1 \rightarrow \begin{bmatrix} 0 & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq 1 \tag{6.6}$$

$$a_2^T x \leq 1 \rightarrow \begin{bmatrix} 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq 1 \tag{6.7}$$

$$a_3^T x \leq 1 \rightarrow \begin{bmatrix} \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq 1 \tag{6.8}$$

$$a_4^T x \leq 1 \rightarrow \begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq 1 \tag{6.9}$$

In order for the inequalities (6.6) and (6.7) to be true, the variable x_2 must be confined within the range $-1 \leq x_2 \leq 3$, regardless of what x_1 is.
For the inequalities (6.8) and (6.9) to be true, the variable x_1 must be confined within the range $-1 \leq x_2 \leq 2$, regardless of what x_2 is.
This confirms the confined region that is contained by the half-spaces, defined as the polytope $\mathcal{P}$, is as shown in Figure 6.1.

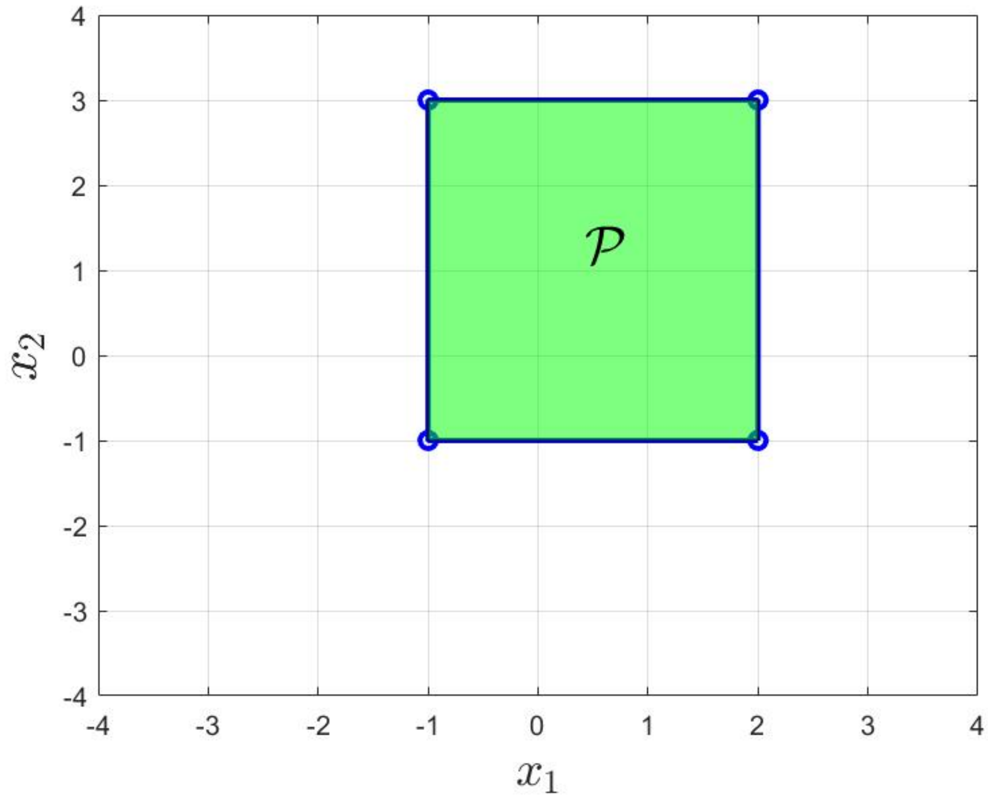


Figure 6.1: Polytope $\mathcal{P}$ is contained by the blue half-spaces, and represented by the green region.

6.1.2 Lyapunov stability criterion

To ensure stability of a dynamical system, stability theory is explored. We saw that the linear Carleman approximation was uncontrollable, therefore, some of its eigenvalues could not be moved to make it follow the cubic model. If an LTI-system is controllable, then the closed-loop eigenvalues can be assigned at arbitrary desired locations of the complex plane. The stability of a nonlinear dynamical system *around* an equilibrium point of the solution can be proven using the *Lyapunov stability criterion.*

Theorem 3 *An equilibrium is said to be locally asymptotically stable if the Lyapunov function is locally positive definite for* $x \neq x_{equilibrium}$*:*

$$V(x) > 0 \tag{6.10}$$

and its derivative is locally negative definite for $x \neq x_{equilibrium}$*:*

$$\dot{V}(x) < 0 \tag{6.11}$$

These criterions are often explained using $V(x)$ as a general energy function. If a system $V(x)$ has positive definite energy and a negative definite derivative in some neighbourhood around the equilibrium point, then this point is said to be *locally asymptotically stable*(L.A.S).

Finding the *Lyapunov function* $V(x)$ can be hard in some cases, but for a linear system $\dot{x} = Ax$ it can be done as follows [2]:

$$V(x) = x^T P x \tag{6.12}$$

where $P = P^T > 0$ ensures that (6.12) is positive definite as required in (6.10).

From this, the derivative can also be found:

$$\dot{V}(x) = \frac{\partial V}{x}\frac{dx}{dt} = x^T(A^T P + PA)x \tag{6.13}$$

where $A^T P + PA < 0$ ensures that (6.13) is negative definite as required in (6.11).

The linear system is *Lyapunov stable* if there exists some positive definite symmetrical matrix P with a negative definite derivative in some neighbourhood around the equilibrium point.

Region of attraction

For any Lyapunov function satisfying the stability criterion, there exists a *region of attraction*(RA)[1]. This is a firmly stated region which describes the renowned "around the equilibrium", however, it is hard or even impossible to find the RA except for in some simple cases. An estimate of the RA can be found by using *linear matrix inequalities*(LMI's) as described in [4]. The idea is that given the nonlinear system corresponding to the quadratic Carleman model (4.7):

$$\dot{x}(t) = Ax(t) + \begin{bmatrix} x^T(t)B_1x \\ x^T(t)B_2x \\ \vdots \\ x^T(t)B_nx \end{bmatrix}$$

a quadratic Lyapunov function satisfying the stability criterion over an *invariant set*[2] is built using an LMI-based optimization problem[27]. If solveable, these inequalities ensure that the polytope $\mathcal{P}$ belongs to the RA.

[1] "...given an asymptotically stable system, the RA is defined as the largest connected set Ω containing the origin and such that every solution starting in Ω converges to 0" [4].

[2] If, for all solutions of a system starting in a set remains in the set as $t \to \infty$, then the set is said to be invariant.

6.1.3 State feedback control of nonlinear quadratic systems

The goal of this section is to explore a method of finding a L.A.S state feedback controller in the form:

$$u(t) = \mathcal{K}x(t) \tag{6.14}$$

for the nonlinear system [3]:

$$\dot{x} = Ax(t) + Bu(t) + \begin{bmatrix} x^T(t)E_1 \\ x^T(t)E_2 \\ \vdots \\ x^T(t)E_n \end{bmatrix} u(t) \tag{6.15}$$

which is L.A.S on the polytope $\mathcal{P}$ enclosed by the RA.

Theorem 4 *Given system (6.15), with $A \in n \times n$ and $B \in m \times n$ and the polytope $\mathcal{P}$, a controller gain $\mathcal{K}$ can be found by solving:*

$$0 < \gamma < 1 \tag{6.16a}$$

$$P > 0 \tag{6.16b}$$

$$\begin{bmatrix} 1 & \gamma a_k^T P \\ P a_k \gamma & P \end{bmatrix} \geq 0, \quad k = 1, 2, \cdots, q \tag{6.16c}$$

$$\begin{bmatrix} 1 & x_{(i)}^T \\ x_{(i)} & P \end{bmatrix} \geq 0, \quad i = 1, 2, \cdots, p \tag{6.16d}$$

$$\gamma(AP + PA^T) + \gamma(BL + L^T B^T) + \begin{bmatrix} x_{(i)}^T E_1 L \\ x_{(i)}^T E_2 L \\ \vdots \\ x_{(i)}^T E_n L \end{bmatrix} + \Big((L^T E_1^T) x_{(i)} \cdots (L^T E_n^T) x_{(i)} \Big) \tag{6.16e}$$

$$< 0 \quad i = 1, 2, \cdots, p$$

with a symmetric matrix $P \in \mathbb{R}^{n \times n}$ and a matrix $L \in \mathbb{R}^{m \times n}$, where p is the amount of vertices and q is the amount of constraints a_k. Then, the controller gain in (6.14) can then be calculated as $\mathcal{K} = LP^{-1}$.

In (6.16) we can see that (6.16b) is the positive definite Lyapunov function, $V(x)$ shown in (6.10) and (6.16e) is the negative definite $\dot{V}(x)$ shown in (6.11) with a given polytope $\mathcal{P}$. (6.16a) is a scalar that can be set to a fixed value, an optimal value can be found through parameter searches if necessary [3].

Condition (6.16c) ensures that the upscaled polytope $\frac{1}{\gamma}\mathcal{P}$ contains a level curve of the Lyapunov function, $\frac{1}{\gamma}\mathcal{P} \supset \mathcal{E}$, which is a subset of the RA. The level curve is expressed by the ellipsoid:

[3]Corresponding to the quadratic Carleman model (4.7).

$$\mathcal{E} := \{x \in \mathbb{R}^n : x^T P^{-1} x \leq 1\}$$

(6.16c) ensures therefore that the ellipsoid $\mathcal{E}$ is invariant.

Condition (6.16d) ensures that the polytope $\mathcal{P}$ is a subset of the ellipsoid $\mathcal{E}$, $\mathcal{E} \supset \mathcal{P}$. This, alongside with (6.16c), ensures that $\mathcal{P}$ is in the RA.

Given the Lyapunov function:

$$V(x) = x^T P^{-1} x$$

with $P > 0$, find $\dot{V}(x) < 0$.

$$\begin{aligned}\dot{V}(x) &= \dot{x}^T P^{-1} x + x^T P^{-1} \dot{x} \\ &= x^T \Big\{ \Big[(A + B\mathcal{K})^T + \big((E_1\mathcal{K})^T x_{(i)} \cdots (E_n\mathcal{K})^T x_{(i)} \big) \Big] P^{-1} \\ &\quad + P^{-1} \left[A + B\mathcal{K} + \begin{bmatrix} x_{(i)}^T E_1 \mathcal{K} \\ \vdots \\ x_{(i)}^T E_n \mathcal{K} \end{bmatrix} \right] \Big\} x\end{aligned}$$

Pre- and post-multiply by P.

$$\begin{aligned} x^T \Big\{ &\Big[PA^T + P\mathcal{K}^T B^T + \big(P\mathcal{K}^T E_1^T x_{(i)} \cdots P\mathcal{K}^T E_n^T x_{(i)} \big) \Big] \\ &+ \left[AP + B\mathcal{K}P + \begin{bmatrix} x_{(i)}^T E_1 \mathcal{K}P \\ \vdots \\ x_{(i)}^T E_n \mathcal{K}P \end{bmatrix} \right] \Big\} x \end{aligned}$$

Introduce $L = \mathcal{K}P$.

$$\begin{aligned} x^T \Big\{ &\Big[PA^T + L^T B^T + \big(L^T E_1^T x_{(i)} \cdots L^T E_n^T x_{(i)} \big) \Big] \\ &+ \left[AP + BL + \begin{bmatrix} x_{(i)}^T E_1 L \\ \vdots \\ x_{(i)}^T E_n L \end{bmatrix} \right] \Big\} x \end{aligned}$$

This results in the inequality:

$$PA^T + L^T B^T + \big(L^T E_1^T x_{(i)} \cdots L^T E_n^T x_{(i)} \big) + AP + BL + \begin{bmatrix} x_{(i)}^T E_1 L \\ \vdots \\ x_{(i)}^T E_n L \end{bmatrix} < 0 \tag{6.17}$$

By introducing γ we get (6.16e).

Solvers

The LMIs presented are solved using the MATLAB LMI toolboxes `YALMIP` [13] and `SEDUMI` [19]. The steps for solving LMIs are:

- The A and B matrices must be available for the solvers.

- Create symbolic decision variables[4] for the symmetric matrix P and the *full* matrix L.
- Create a list of LMIs/constraints.
- Optimize the LMIs.
- If the problem has a feasible solution, then values for the matrices P and L are found.
- The controller gain is $\mathcal{K} = LP^{-1}$

6.2 Controller for the quadratic Carleman approximation

In Section 6.1, we introduced several concepts that are related to finding the controller gain $\mathcal{K}$. In this section, those concepts will be applied to the quadratic Carleman approximation. The goal is to create a state feedback controller for this system.

Before applying the constraints introduced in 6.1.3, it is important to identify certain limitations of the system. These limitations are included as additional constraints to those shown in 6.1.3, they ensure that the controller operates within its capabilities. The main limitation of our system is the range at which the valve $LV001$ can operate. It is not possible for the valve to open more than its max capacity, and it is not possible to close the valve more than when it is shut tight. For the signal $u_{LV001}(t)$, these limitations are defined as:

$$0 \leq u_{LV001}(t) \leq 1 \tag{6.18}$$

Recall that for the quadratic Carleman approximation, the state is defined as:

$$z_n(t) = (h_1(t) - h_{1,O})^n \tag{6.19}$$

which means that the input is calculated as:

$$\delta u_{LV001}(t) = u_{LV001}(t) - u_{LV001,O} \tag{6.20}$$

For *Scenario 1*, this means that the signal $\delta u_{LV001}(t)$ cannot exceed the following boundaries:

$$-0.5159 \leq \delta u_{LV001}(t) \leq 0.4841 \tag{6.21}$$

Due to this, we define the upper and lower bound of $\delta u_{LV001}(t)$ to ± 0.45.

The constraints that ensures that the controller does not exceed these boundaries are defined as:

$$\begin{bmatrix} P & L^T \\ L & \gamma_*^2 \end{bmatrix} > 0 \tag{6.22}$$

where $\gamma_*^2 = 0.45$ is the boundary for the signal $\delta u_{LV001}(t)$ [15].

In Chapter 4, Section 4.3, we concluded that the quadratic Carleman approximation of order $n \geq 4$ was an accurate approximation of the cubic model. Therefore, for the remaining part of this project, the $4th$ order quadratic Carleman approximation will be considered. The state vector

[4] *sdpvar* in MATLAB.

$z(t)$, state matrix A, input matrix B and the matrices regarding the nonlinear parts $E_1, \cdots, E_4$ for the $4th$ order quadratic Carleman approximation are reiterated below:

$$z(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \\ z_4(t) \end{bmatrix}, A = \begin{bmatrix} a & c & e & 0 \\ 0 & 2a & 2c & 2e \\ 0 & 0 & 3a & 3c \\ 0 & 0 & 0 & 4a \end{bmatrix}, B = \begin{bmatrix} b \\ 0 \\ 0 \\ 0 \end{bmatrix}, E_1 = \begin{bmatrix} d \\ f \\ 0 \\ 0 \end{bmatrix}, E_2 = \begin{bmatrix} 2b \\ 2d \\ 2f \\ 0 \end{bmatrix}, E_3 = \begin{bmatrix} 0 \\ 3b \\ 3d \\ 3f \end{bmatrix}, E_4 = \begin{bmatrix} 0 \\ 0 \\ 4b \\ 4d \end{bmatrix} \tag{6.23}$$

where $a, b, \cdots, f$ are defined as in Section 4.2, Eq.(4.6).

The vertices of the polytope $\mathcal{P}$ are yet to be defined. Since the order of the quadratic Carleman approximation is $n = 4$, this means that the vertices will exist in the $4th$ dimensional space, where the axes are $[z_1, z_2, z_3, z_4]$. Given equation (6.19), the *value* on the z_1-axis for each vertex will determine how far the polytope $\mathcal{P}$ reaches from the operating point. A desired range for the controller is $\pm 0.05m$ from the operating point. By defining the *values* for each state at which a vertex lies, we can use the MATLAB function `combvec`[5] to express the actual coordinates of the vertices.

Consider the following example for a $2nd$ order quadratic Carleman approximation:

The desired range of our controller is $\pm 0.05m$. Recall that:

$$z_1(t) = h_1(t) - h_{1,O} \tag{6.24}$$

Thus, the two *values* for this state are $[-0.05, 0.05]$. Given equation (6.19), we can see that the *values* for z_2 are $[(-0.05)^2, (0.05)^2]$. However, this action looses the lower bound for z_2, due to the exponent being an even number. This is why the lower bound is manually set to 0. If the exponent is odd, say for z_3, then the lower bound is just -0.05^3. This gives the actual values for z_2 to be $[0, 0.05^2]$. Using the `combvec` function, with these values as inputs, gives the following coordinates:

$$\texttt{combvec}([-0.05, 0.05], [0, 0.05^2]) = \begin{bmatrix} -0.0500 & 0.0500 & -0.0500 & 0.0500 \\ 0 & 0 & 0.0025 & 0.0025 \end{bmatrix} \tag{6.25}$$

where each column is the coordinates of a vertex. Inserting these vertices in the `vert2con` function gives the half-spaces that contain the polytope. Figure 6.2 shows the plot of this polytope.

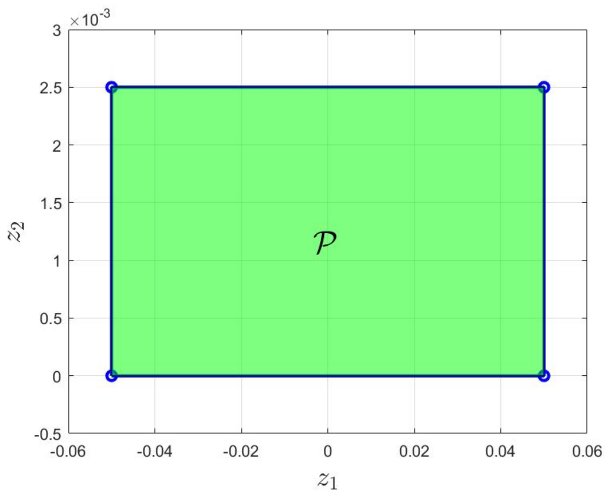


Figure 6.2: Polytope for the $2nd$ order quadratic Carleman approximation.

[5] combvec.

Following the same logic as in the example above, the *values* of the states for the $4th$ order quadratic Carleman approximation are[6]:

$$\begin{aligned} &z_{1,V} = [-0.05, 0.05],\ z_{2,V} = [\approx 0, 0.0025], \\ &z_{3,V} = [-0.000125, 0.000125],\ z_{4,V} = [\approx 0, 0.00000625] \end{aligned} \tag{6.26}$$

Inserting these values in the `combvec` function yields the following vertices:

$$\begin{aligned}
x_1 &= \begin{bmatrix} -0.0500 \\ -0.0003 \\ -0.0001 \\ -0.0000 \end{bmatrix}, x_2 = \begin{bmatrix} 0.0500 \\ -0.0003 \\ -0.0001 \\ -0.0000 \end{bmatrix}, x_3 = \begin{bmatrix} -0.0500 \\ 0.0025 \\ -0.0001 \\ -0.0000 \end{bmatrix}, x_4 = \begin{bmatrix} 0.0500 \\ 0.0025 \\ -0.0001 \\ -0.0000 \end{bmatrix} \\
x_5 &= \begin{bmatrix} -0.0500 \\ -0.0003 \\ 0.0001 \\ -0.0000 \end{bmatrix}, x_6 = \begin{bmatrix} 0.0500 \\ -0.0003 \\ 0.0001 \\ -0.0000 \end{bmatrix}, x_7 = \begin{bmatrix} -0.0500 \\ 0.0025 \\ 0.0001 \\ -0.0000 \end{bmatrix}, x_8 = \begin{bmatrix} 0.0500 \\ 0.0025 \\ 0.0001 \\ -0.0000 \end{bmatrix} \\
x_9 &= \begin{bmatrix} -0.0500 \\ -0.0003 \\ -0.0001 \\ 0.0000 \end{bmatrix}, x_{10} = \begin{bmatrix} 0.0500 \\ -0.0003 \\ -0.0001 \\ 0.0000 \end{bmatrix}, x_{11} = \begin{bmatrix} -0.0500 \\ 0.0025 \\ -0.0001 \\ 0.0000 \end{bmatrix}, x_{12} = \begin{bmatrix} 0.0500 \\ 0.0025 \\ -0.0001 \\ 0.0000 \end{bmatrix} \\
x_{13} &= \begin{bmatrix} -0.0500 \\ -0.0003 \\ 0.0001 \\ 0.0000 \end{bmatrix}, x_{14} = \begin{bmatrix} 0.0500 \\ -0.0003 \\ 0.0001 \\ 0.0000 \end{bmatrix}, x_{15} = \begin{bmatrix} -0.0500 \\ 0.0025 \\ 0.0001 \\ 0.0000 \end{bmatrix}, x_{16} = \begin{bmatrix} 0.0500 \\ 0.0025 \\ 0.0001 \\ 0.0000 \end{bmatrix}
\end{aligned} \tag{6.27}$$

The scalar γ is set to:

$$\gamma = 0.1$$

When solving the constraints shown in 6.1.3, Eq. (6.16c) will be excluded from these calculations. The reason for this will be further discussed in Chapter 9. This implies that the solved controller can not guarantee that $\mathcal{E}$ is invariant.[7] Since (6.16c) will not be included in this part, this means that there is no need to present the half-spaces a_k. With that, all of the variables that are required in order to solve the constrains for $\mathcal{K}$ are now defined.

Solving the constrains (6.16a)-(6.16e), with the exclusion of (6.16c), together with the additional constraint (6.22), results in the following controller gain[8]:

$$\mathcal{K} = \begin{bmatrix} 0.2345 & 0.0129 & -0.0464 & -0.0325 \end{bmatrix} \tag{6.28}$$

With the $\mathcal{K}$ matrix defined, it is now possible to rewrite the quadratic Carleman approximation, by replacing $\delta u_{LV001}(t)$ with $\mathcal{K}z(t)$:

$$\dot{z}(t) = Az(t) + B\mathcal{K}z(t) + \begin{bmatrix} z^T(t)E_1 \\ z^T(t)E_2 \\ z^T(t)E_3 \\ z^T(t)E_4 \end{bmatrix} \mathcal{K}z(t) \tag{6.29}$$

[6] Due to numerical issues, the lower bound of the states $z_n(t)$ where n is an even number, are set to small negative numbers that are approximately zero.

[7] The consequences of $\mathcal{E}$ not being invariant is that it cannot be guaranteed that a solution of the inequalities starting inside $\mathcal{E}$ stays inside $\mathcal{E}$ as $t \to \infty$, which can lead to instability/divergence.

[8] The procedure of how to solve the different constraints are explained in 6.1.3.

To verify that the state feedback controller works, Equation (6.29) is simulated. The system will start with an initial condition of $0.05m$, which is the same as:

$$h_1(t) = z_1(t) + h_{1,O} = 0.05m + 0.25m = 0.3m \tag{6.30}$$

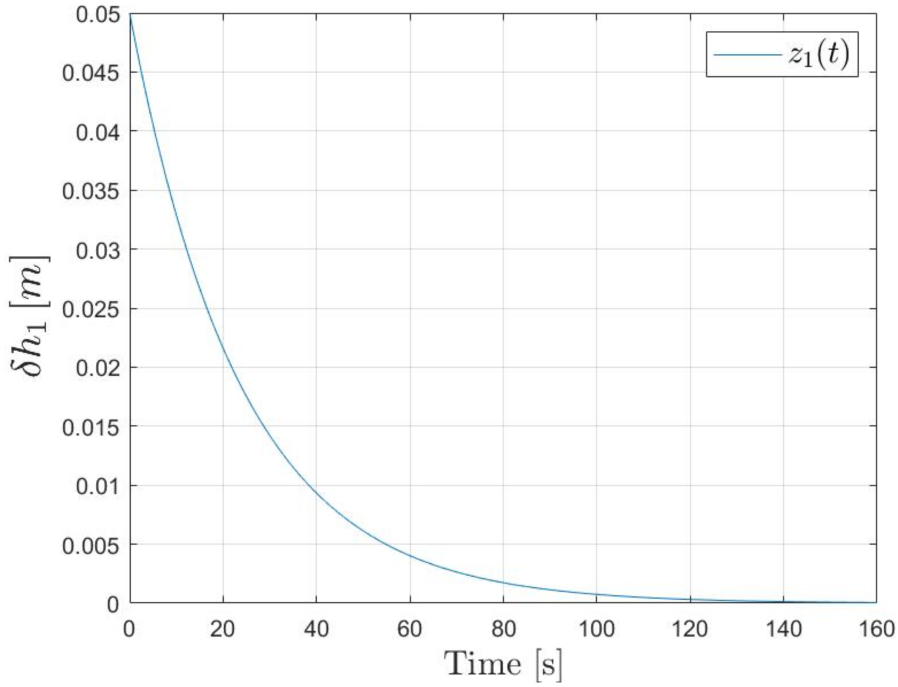


Figure 6.3: Simulation of Eq. (6.29), with initial condition equal to $0.05m$.

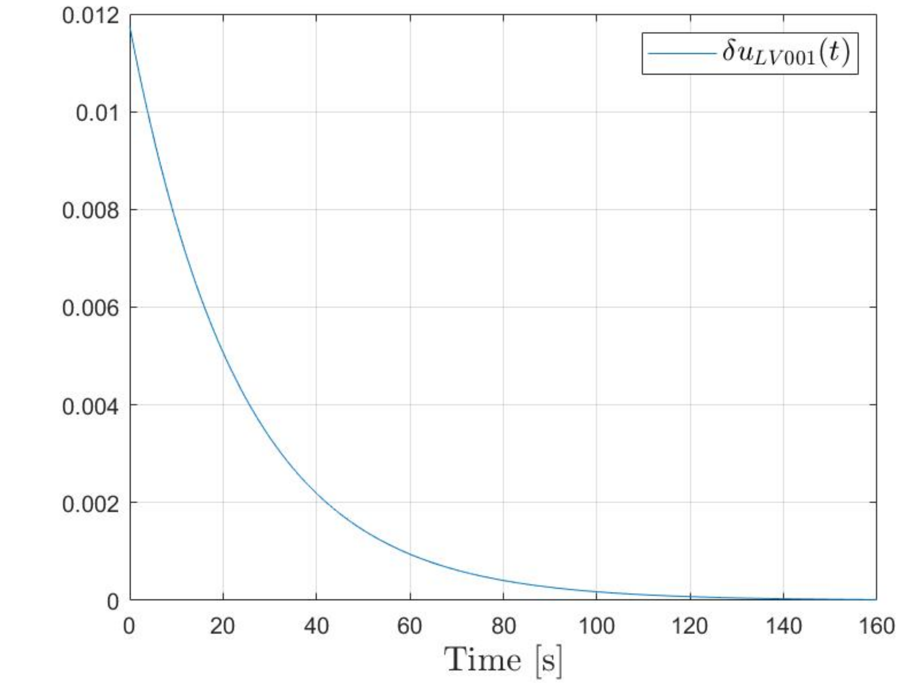


Figure 6.4: Signal $\delta u_{LV001}(t)$, calculated by $\mathcal{K}z(t)$.

Figure 6.3 shows that the state $z_1(t)$ converges to zero, which is equivalent to the water level converging to its operating point of $0.25m$. Figure 6.4 shows that the signal $\delta u_{LV001}(t)$ also converges to zero, which means that the signal $u_{LV001}(t)$ converges to *its* operating point of 0.5159.

6.3 Controller for the quadratic Carleman approximation using two input variables

Up until now, this project has only considered the valve $LV001$ as an actuator. However, the system does have the option of regulating the contribution of the pump $PA001$ as well. Introducing the pump into the quadratic Carleman approximation leads to small adjustments in the input vector $u(t)$, input matrix B and the matrices regarding the nonlinear parts $E_1, E_2, \cdots, E_n$. These adjusted variables are now defined as[9]:

$$u(t) = \begin{bmatrix} u_{LV001}(t) \\ u_{PA001}(t) \end{bmatrix},\ B = \begin{bmatrix} b & g \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},\ E_1 = \begin{bmatrix} d & 0 \\ f & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} E_2 = \begin{bmatrix} 2b & 2g \\ 2d & 0 \\ 2f & 0 \\ 0 & 0 \end{bmatrix} E_3 = \begin{bmatrix} 0 & 0 \\ 3b & 3g \\ 3d & 0 \\ 3f & 0 \end{bmatrix} E_4 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 4b & 4g \\ 4d & 0 \end{bmatrix} \tag{6.31}$$

where:

$$g = \left. \frac{\partial f}{\partial u_{PA001}} \right|_O \tag{6.32}$$

All of the other variables that were defined in Section 6.2 remain unchanged.

With this new actuator, it is again necessary to identify the limitation of the system. After introducing the pump to the controller, it is now important to define certain constraints such that the controller operates within its capability. It is not possible for the pump to supply the tank with more than a certain amount of water. The least amount of water that the pump can supply the tank with is zero. The pump cannot extract water from the tank. Given the pump characteristics, shown in Figure 2.4, one can see that the pump stops adding water to the tank at $u_{PA001}(t) = 0.45$[10]. The maximum water that the pump can supply occurs at $u_{PA001}(t) = 1$. This gives the following constraint:

$$0.45 \leq u_{PA001}(t) \leq 1 \tag{6.33}$$

The quadratic Carleman approximation works with the signal:

$$\delta u_{PA001}(t) = u_{PA001}(t) - u_{PA001,O} \tag{6.34}$$

where $u_{PA001,O} = 0.65$. For this reason, we define the upper and lower bound of $\delta u_{PA001}(t)$ to ± 0.2 [11]. This gives another additional constraint, (6.22), where $\gamma_*^2 = 0.2$.

To find the controller gain $\mathcal{K}$, we solve the same constraints as in Section 6.2, with the addition of the saturation constraint for the pump. This results in the following:

$$\mathcal{K} = \begin{bmatrix} 0.2345 & -0.0076 & -0.0446 & -0.0227 \\ -0.1029 & -0.0223 & 0.0178 & 0.0177 \end{bmatrix} \tag{6.35}$$

Replacing the input vector $u(t)$ with $\mathcal{K}z(t)$, results in a quadratic system, as shown by (6.29). Verification of the controller is done by simulating this system. The initial level is set to $0.05m$.

[9] Note that these matrices only apply for the $4th$ order quadratic Carleman approximation, which is still being regarded from the previous section.

[10] The pump needs a value of at least 0.45 to pump water up to the tank, any lower values and the pump will not be able to pump water up to the intake of the tank.

[11] As (6.22) only works in symmetrical ranges.

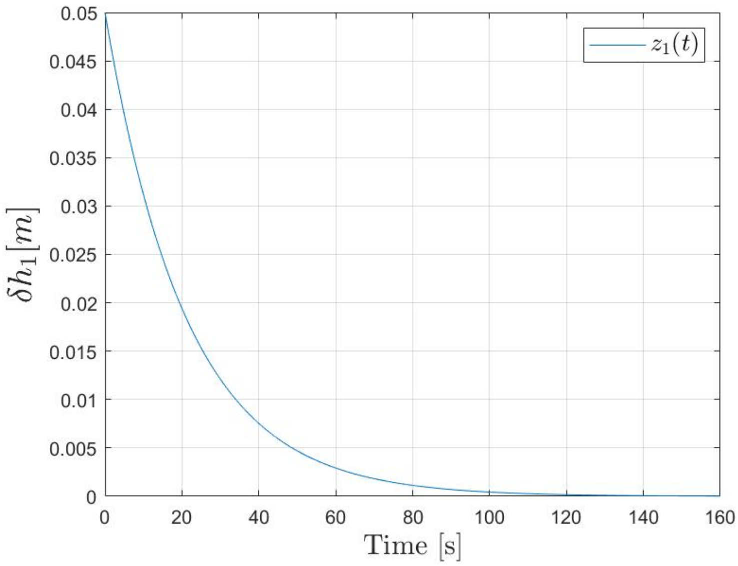


Figure 6.5: $4th$ order quadratic Carleman approximation with two actuator.

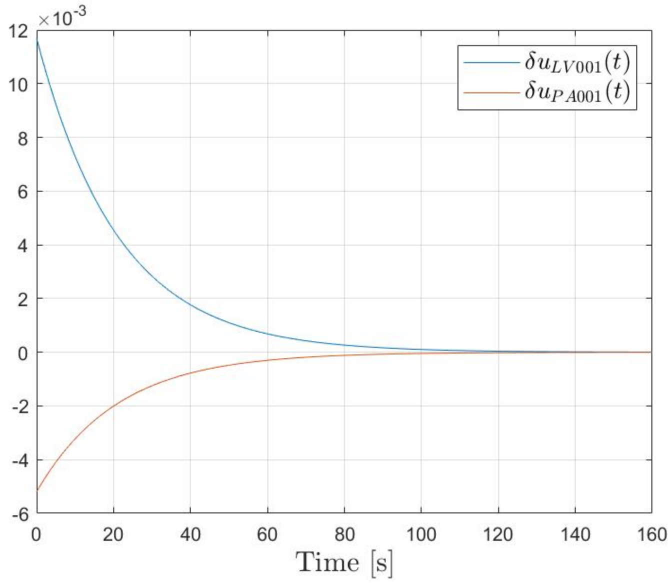


Figure 6.6: Signal $u(t)$, calculated by $\mathcal{K}z(t)$.

Note that the response in Figure 6.5 is slightly faster than the response in Figure 6.3. However, the main benefit of creating the controller gain $\mathcal{K}$ with two actuators, is that the degree of freedom is increased. This means that the probability of finding a solution to the constraints in (6.16) is higher.

The rate of convergence can be added as an additional constraint. By introducing an α, we can make the controller less conservative, leading in faster responses. In [3], it is introduced with:

$$2\alpha P + AP + PA^T + BL + L^T B^T < 0$$

This implementation of α amplifies the linear part of the quadratic Carleman approximation, which makes the quadratic parts of the system negligible. This is the equivalent to the linearized Carleman approximation, which was proven to be uncontrollable in the previous chapter. Therefore, we include α to the vertex that corresponds to the maximum positive deviation from the equilibrium point. This makes the linear part impact the system the least, allowing for a bigger α and more feasible solutions.

$$\begin{aligned}\gamma(AP + PA^T + BL + L^T B^T) &+ \left(L^T G_1^T x_{(i)} \cdots L^T G_n^T x_{(i)}\right) \\ &+ \begin{bmatrix} x_{(i)}^T G_1 L \\ \vdots \\ x_{(i)}^T G_n L \end{bmatrix} + 2\,\gamma\, P\alpha < 0 \\ &\qquad\qquad i = 1, 2, \cdots, p\end{aligned} \tag{6.36}$$

Equation (6.36) shows the additional constraint that includes α for the maximum positive vertex. By further inspecting the vertices in (6.27), it is possible to see that x_{16} is the vertex that corresponds to the maximum positive deviation from the equilibrium point. The biggest α that allows for a feasible solution is:

$$\alpha = 0.096 \tag{6.37}$$

The constraints are solved as usual, with the addition of (6.36), which results in the following controller gain:

$$\mathcal{K} = \begin{bmatrix} 1.3312 & -0.3827 & -1.4645 & 0.2560 \\ -0.0861 & -0.2364 & -0.0611 & 0.5122 \end{bmatrix} \tag{6.38}$$

The input vector $u(t)$ is replaced by $Kz(t)$, which gives the quadratic Carleman approximation the same form as (6.29). To verify the quadratic controller, the quadratic Carleman approximation is simulated with an initial condition of $0.05m$.

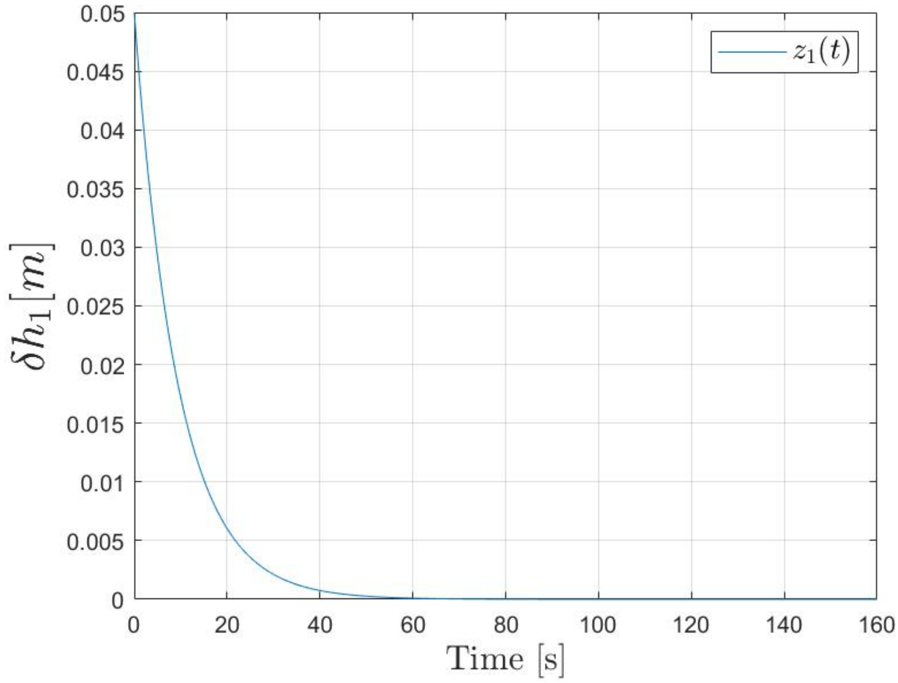


Figure 6.7: $4th$ order quadratic Carleman approximation with two actuator and implemented convergence rate.

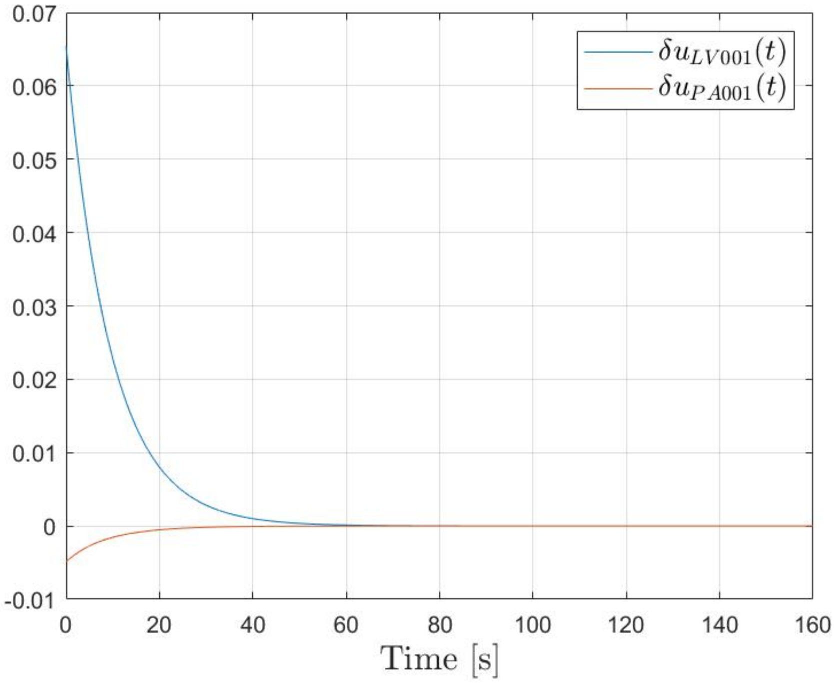


Figure 6.8: Signal $u(t)$, calculated by $\mathcal{K}z(t)$ with implemented convergence rate.

Comparing the response of Figure 6.7 with the response of Figure 6.5, we see that the time it takes for the system to converge to its operating point is greatly reduced. The signal $\delta u_{LV001}(t)$ is also less conservative with its contribution. These improvements makes for this final iteration of the quadratic controller to be considered further in the next chapter.

Chapter 7

Experimental results

In Chapter 6, a quadratic controller was calculated for the $4th$ order quadratic Carleman approximation. There were presented several results, regarding different iterations of the controller. The conclusion was that the controller that took into account two input variables and the convergence rate α was the superior controller. This chapter will look to apply this controller to the two-tank system, and observe how well the controller works in a practical setting. Section 7.1 will describe the approach to implementing the controller to the system, and describe the procedure of how the data was collected. Section 7.2 will analyze and discuss the obtained results from the experiments. This chapter will only include results from *Scenario 1*. For the results regarding *Scenario 2*, see Appendix D.

7.1 Application of the quadratic controller on to the two-tank system

In this project, the Simulink file that connects to the system was made accessible by [8]. This file is slightly modified, such that the input vector $u(t)$ follows the quadratic controller. In Chapter 6, the controller gain was calculated to be:

$$\mathcal{K} = \begin{bmatrix} 1.3312 & -0.3827 & -1.4645 & 0.2560 \\ -0.0861 & -0.2364 & -0.0611 & 0.5122 \end{bmatrix} \tag{7.1}$$

This gives the following δ input vector:

$$\delta u(t) = \begin{bmatrix} \delta u_{LV001}(t) \\ \delta u_{PA001}(t) \end{bmatrix} = \mathcal{K} z(t) = \mathcal{K} \begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \\ z_4(t) \end{bmatrix} = \mathcal{K} \begin{bmatrix} h_1(t) - h_{1,O} \\ (h_1(t) - h_{1,O})^2 \\ (h_1(t) - h_{1,O})^3 \\ (h_1(t) - h_{1,O})^4 \end{bmatrix} \tag{7.2}$$

Note that the quadratic controller was made with respect to the quadratic Carleman approximation, which works with δ-variables, i.e. $\delta u_{LV001}(t) = u_{LV001}(t) - u_{LV001,O}$ and $\delta u_{PA001}(t) = u_{PA001}(t) - u_{PA001,O}$. Therefore, the input vector $u(t)$ of the *two-tank system*, is calculated as:

$$u(t) = u_O + \delta u(t) = \begin{bmatrix} u_{LV001,O} \\ u_{PA001,O} \end{bmatrix} + \begin{bmatrix} -\delta u_{LV001}(t) \\ \delta u_{PA001}(t) \end{bmatrix} \tag{7.3}$$

Note that the variable $\delta u_{LV001}(t)$ is inverted when calculating the input vector $u(t)$. The reason for this is that the signal $u_{LV001}(t)$ has an inverse correlation with the water level $h_1(t)$. If the signal $u_{LV001}(t)$ is increased, then the valve $LV001$ will open more, which leads to a reduction in the water level $h_1(t)$.

The procedure of the experiments is as shown in Table 7.1[1]. The two first periods of the procedure are used to drive the system to some desired initial condition before the quadratic controller is implemented.

Period [s]	$\mathbf{u}_{PA001}(t)$	$\mathbf{u}_{LV001}(t)$	**Description**
$0 \leq t \leq 10$	0	1	Empty the tank by turning off $PA001$ and opening $LV001$ to the max.
$10 \leq t \leq 30$	$0 \leq \lambda \leq 1$	0	Fill up the tank manually by closing $LV001$ and turning $PA001$ on.
$30 \leq t$	$u_{PA001,O} + \delta u_{PA001}(t)$	$u_{LV001,O} - \delta u_{LV001}(t)$	Introduce the quadratic controller to the system.

Table 7.1: Experiment procedure.

7.2 Analysis of the experimental results

Following the procedure from Table 7.1, and introducing the controller when $h_1(t) \approx 0.3m$ resulted in the data presented in Figure 7.1.

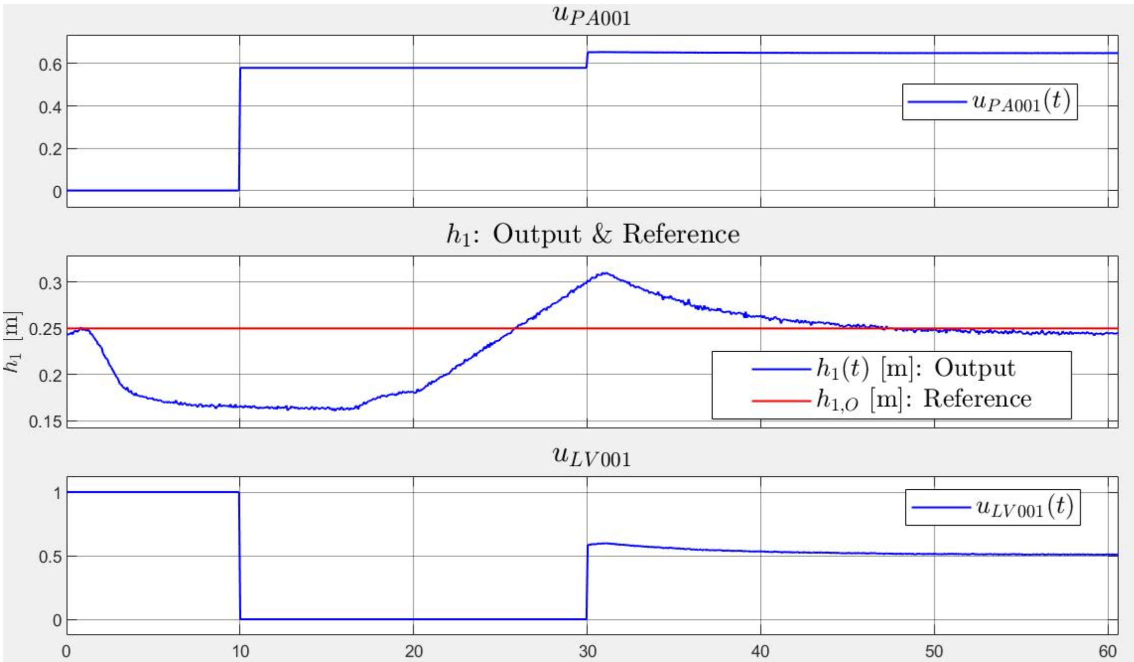


Figure 7.1: Experimental results from the quadratic controller.

Figure 7.1 shows that the system, with the introduced controller, converges to the operating point of $0.25m$. The controller is introduced to the system when the water level is $h_1(t) \approx 0.3m$, which is within the polytope $\mathcal{P}$ that the controller was designed for. It looks as if the signal $u_{PA001}(t)$ is constant, however, this is not the case. The explanation to this can be found by further analysing the controller gain (7.1). The $2nd$ row vector in (7.1), which is the gain for $\delta u_{PA001}(t)$, consists mostly of values that are of the order 10^{-1} or smaller. This is why the contribution of $\delta u_{PA001}(t)$ is so small when compared to $\delta u_{LV001}(t)$, which explains why it looks as if $u_{PA001}(t)$ is constant at $u_{PA001,O}$.

[1] λ is an arbitrary value, depending on the desired water level before the quadratic controller is implemented.

An additional experiment is performed, where the goal is to test the controller when the water level $h_1(t)$ is outside of the polytope $\mathcal{P}$. In this case, the controller is introduced to the system when $h_1(t) \approx 0.5m$, which is $0.2m$ outside of the reach of the polytope $\mathcal{P}$ that the controller was designed for. This means that it is not possible to guarantee that the controller will work within its capabilities and not saturate the input variables. Figure 7.2 shows the results of this experiment.

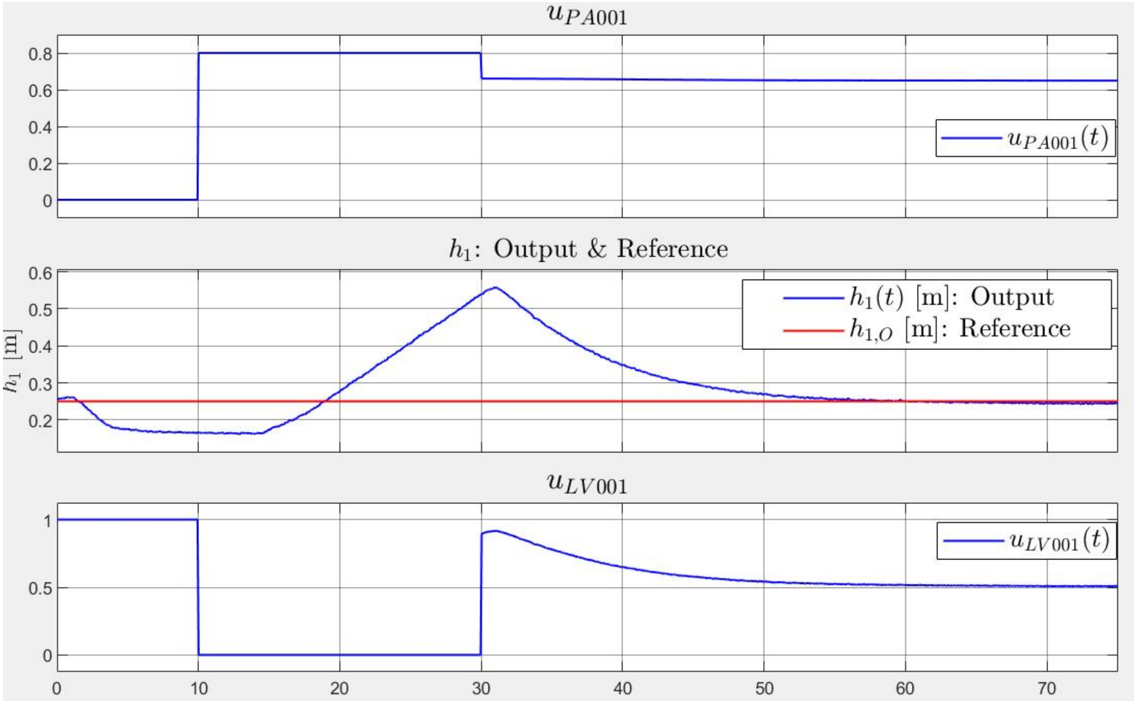


Figure 7.2: Experimental results from the quadratic controller.

Figure 7.2 shows that, even though the controller is working outside of its designed region, the system still converges to its operating point. In this case, the controller does not saturate the input variables. However, an equivalent experiment was performed for *Scenario 2*[2], where the controller demanded that $u_{LV001}(t) < 0$, which is not possible.

Figure 7.1, together with Figure D.1, shows that the controller works symmetrically, and that the desired output is achieved in both cases.

[2]See Figure D.2.

Part IV

Conclusions and future work

Chapter 8

Conclusions

The main goal of this thesis was to design controllers for the two-tank process, using models derived with the Carleman embedding technique. A hypothesis is that designing a controller based upon the linearized Carleman approximation, derived from a higher order Taylor approximation, would lead to a better controller than one based upon the Taylor linearization.
The conclusions obtained in this thesis are the following:

- Among the higher order Taylor approximations, the cubic model was concluded to be best suited for the Carleman approximation when applied to the two-tank system.

- The Carleman embedding technique was applied to the cubic model, resulting in a quadratic Carleman approximation. Linearizing this approximation resulted in an uncontrollable system. Designing a controller for this system would be the equivalent of designing a controller for the Taylor linearized model. Therefore, design of a controller for the quadratic Carleman approximation was explored instead.

- The most successful controller design included two input variables, which resulted in a wider feasibility of the linear matrix inequalities. The convergence rate α was also included in the control design, which made the response of the system faster.

- The main drawback in this controller design was numerical issues related to solving the LMIs. Therefore, an exclusion of the condition (6.16c) ensuring invariancy had to be considered. However, this could lead to divergence in some cases. It is also worth mentioning that in order to prevent saturation in the input variables, symmetrical ranges must be considered. This limits the input variables, leading to a slower response.

- There is not necessarily a clear advantage in designing a controller for the quadratic Carleman approximation, but this approach has some potential. A quadratic controller fulfilling the criterions in (6.16), will ensure stability in a given region that the controller was designed for.

Chapter 9

Future work

Further improvements and interesting aspects to be pursued:

- Improvements on the design of the polytopes could be explored further in [9]. Better constructed polytopes would lead to a more feasible controller design.

- The controller input limitations (6.22) on $u_{PA001}(t)$ and $u_{LV001}(t)$ only work in symmetrical ranges. This means that it does not use the whole range of the inputs available, leading to a potentially slower response and less feasible controller design.

- Numerical issues lead to poor feasibility especially when including condition (6.16c). Scaling the variables from meters to centimeters could have a positive effect on the feasibility of the controller design, leading to less numerical issues.

- An interesting scenario for the controller design was briefly investigated, where the quadratic Carleman approximation is truncated at $n = 2$. The polytope $\mathcal{P}$ is modified to be strictly positive.

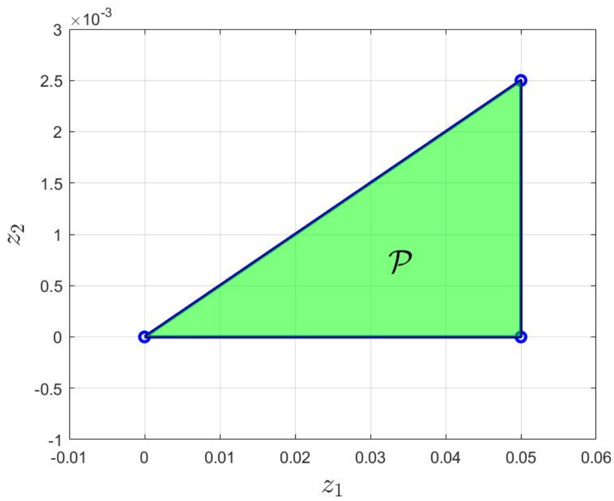


Figure 9.1: Modified polytope.

α is introduced to the vertex with the maximum positive deviation from the equilibrium point. This controller results in a faster response and a bigger controller gain, in addition to being feasible when including constraint (6.16c)[1]. The drawbacks of this is that it only works for positive δh_1. A suggested workaround would be to introduce a type of switching which has one controller for positive and one for negative δh_1. This would require some modifications of the Lyapunov theory to ensure that stability is preserved in spite of the switching.

[1]See Appendix E.

- The tank system is open-loop stable, therefore, the benefits of a quadratic controller become unclear. Testing a quadratic Carleman approximation on a system that is not open-loop stable could be explored. The benefit would be that the quadratic controller would ensure stability in a given region around the equilibrium point.

- A parameter search for optimal γ could be conducted to more easily find feasible solutions and a bigger α [3].

- The nonlinear dynamical model of tank 2 was derived in Chapter 2. However, this model was not considered. It would be interesting to see if the linearized Carleman approximation of this dynamical model is controllable.

Bibliography

[1] *Open-loop-and-closed-loop-feedback-control*. Read: 27.04.2021, Available here.

[2] *Quadratic Lyapunov function*. Read: 02.05.2021, Available here.

[3] F. Amato, R. Ambrosino, M. Ariola, C. Cosentino, and A. Merola. State feedback control of nonlinear quadratic systems. In *2007 46th IEEE Conference on Decision and Control*, pages 1699–1703, 2007.

[4] F. Amato, C. Cosentino, and A. Merola. On the region of attraction of nonlinear quadratic systems. *Automatica*, 43(12):2119–2123, 2007.

[5] Arash Amini, Qiyu Sun, and Nader Motee. Carleman state feedback control design of a class of nonlinear control systems. *IFAC-PapersOnLine*, 52(20):229–234, 2019. 8th IFAC Workshop on Distributed Estimation and Control in Networked Systems NECSYS 2019.

[6] Bellman and Richardson. *On some questions arising in the approximate solution of nonlinear differential equations*, 1963. Read: 11.05.2021, Available here.

[7] Tormod Drengstig. *ELE320 totank1 motivasjon modellering*, 2020.

[8] Tormod Drengstig. *ELE320 Totankøving 4: Bestemmelse av regulatorparametre og regulering av totankprosessen*, 2020.

[9] Christoph Fünfzig, Dominique Michelucci, and Sebti Foufou. Polytope-based computation of polynomial ranges. *Computer Aided Geometric Design*, 29:18–29, 01 2012.

[10] Finn Haugen. *Dynamiske systemer, modellering, analyse og simulering.* Vigmostad & Bjørke AS, 2016.

[11] Michael Kleder. *VERT2CON - vertices to constraints*, 2021. Read: 06.05.2021, Available here.

[12] Willi-hans Kowalski, Krzysztof Steeb. *Nonlinear Dynamical Systems And Carleman Linearization.* World Scientific, 1991.

[13] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[14] Dorota Mozyrska and Zbigniew Bartosiewicz. On carleman linearization of linearly observable polynomial systems. *Mathematical Control Theory and Finance*, 01 2008.

[15] T Nguyen and Faryar Jabbari. Output feedback controllers for disturbance attenuation with actuator amplitude and rate saturation. *Automatica*, 36(9):1339–1346, 2000.

[16] Andreas Rauh, Johanna Minisini, and Harald Aschemann. Carleman linearization for control and for state and disturbance estimation of nonlinear dynamical processes. *IFAC Proceedings Volumes*, 42(13):455–460, 2009. 14th IFAC Conference on Methods and Models in Automation and Robotics.

[17] W.-H. Steeb. A note on carleman linearization. *Physics Letters A*, 140(6):336–338, 1989.

[18] W.-H Steeb and F Wilhelm. Non-linear autonomous systems of differential equations and carleman linearization procedure. *Journal of Mathematical Analysis and Applications*, 77(2):601–611, 1980.

[19] Jos F. Sturm. *SEDUMI-Coral lab*, 2021. Read: 06.05.2021, Available here.

[20] Wikipedia. *Control Theory*. Read: 25.04.2021, Available here.

[21] Wikipedia. *Convex hull*. Read: 06.05.2021, Available here.

[22] Wikipedia. *Polytope*. Read: 06.05.2021, Available here.

[23] Wikipedia. *Rank linear algebra*. Read: 25.04.2021, Available here.

[24] Wikipedia. *Taylor series*. Read: 20.03.2021, Available here.

[25] Wikipedia. *Vertex (geometry)*. Read: 06.05.2021, Available here.

[26] Wikipedia. *Controllability and observability*, 2021. Read: 07.05.2021, Available here.

[27] Wikipedia. *Optimization problem*, 2021. Read: 11.05.2021, Available here.

Part V

Appendices

Appendix A

Taylor model comparison for Scenario 2

Intervals under consideration for *Scenario 2*:

$[0.70, 0.80]$ for $h_1(t)$ and $[0.26, 0.46]$ for $u_{LV001}(t)$

The Taylor models for *Scenario 2* are as follows:

Linear model:

$$\delta\dot{h}_1(t) = -0.0111\,(h_1(t) - 0.75) - 0.0683\,(u_{LV001}(t) - 0.3666) \tag{A.1}$$

Quadratic model:

$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0111\,(h_1(t) - 0.75) - 0.0683\,(u_{LV001}(t) - 0.3666)\\ &+ 0.0035\,(h_1(t) - 0.75)^2 - 0.0427\,(h_1(t) - 0.75)\,(u_{LV001}(t) - 0.3666)\end{aligned} \tag{A.2}$$

PQ_A model:

$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0111\,(h_1(t) - 0.75) - 0.0683\,(u_{LV001}(t) - 0.3666)\\ &+ 0.0035\,(h_1(t) - 0.75)^2\end{aligned} \tag{A.3}$$

PQ_B model:

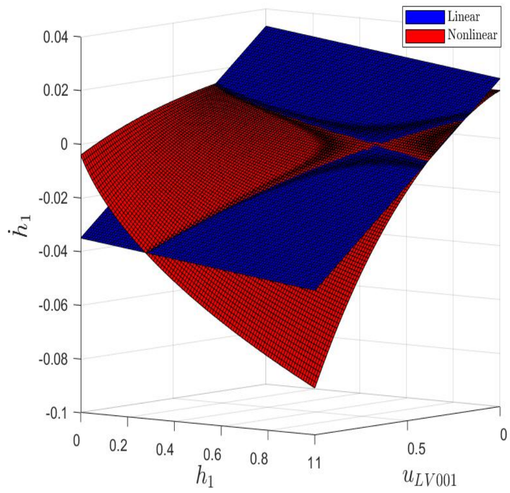
$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0111\,(h_1(t) - 0.75) - 0.0683\,(u_{LV001}(t) - 0.3666)\\ &- 0.0427\,(h_1(t) - 0.75)\,(u_{LV001}(t) - 0.3666)\end{aligned} \tag{A.4}$$

Cubic model:

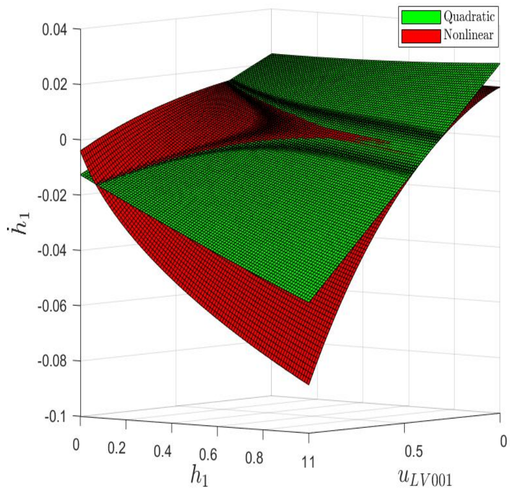
$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0111\,(h_1(t) - 0.75) - 0.0683\,(u_{LV001}(t) - 0.3666)\\ &+ 0.0035\,(h_1(t) - 0.75)^2 - 0.0427\,(h_1(t) - 0.75)\,(u_{LV001}(t) - 0.3666)\\ &- 0.0022\,(h_1(t) - 0.75)^3 + 0.0133\,(h_1(t) - 0.75)^2\,(u_{LV001}(t) - 0.3666)\end{aligned} \tag{A.5}$$

$5th$ order model:

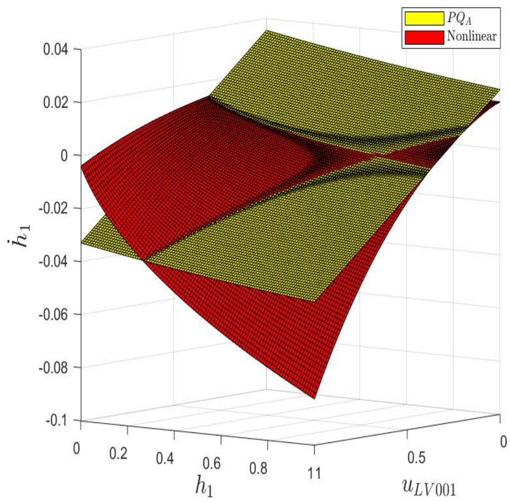
$$\begin{aligned}\delta\dot{h}_1(t) = &-0.0111\,(h_1(t) - 0.75) - 0.0525\,(u_{LV001}(t) - 0.3666)\\ &+ 0.0035\,(h_1(t) - 0.75)^2 - 0.0427\,(h_1(t) - 0.75)\,(u_{LV001}(t) - 0.3666)\\ &- 0.0022\,(h_1(t) - 0.75)^3 + 0.0133\,(h_1(t) - 0.75)^2\,(u_{LV001}(t) - 0.3666)\\ &+ 0.0017\,(h_1(t) - 0.75)^4 - 0.0083\,(h_1(t) - 0.75)^3\,(u_{LV001}(t) - 0.3666)\\ &- 0.0015\,(h_1(t) - 0.75)^5 + 0.0065\,(h_1(t) - 0.75)^4\,(u_{LV001}(t) - 0.3666)\end{aligned} \tag{A.6}$$



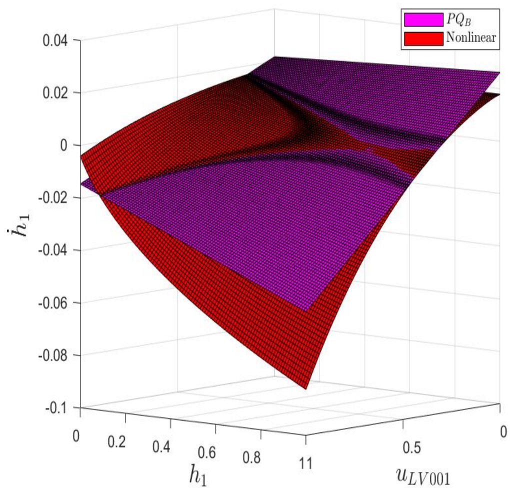
(a) Linear and nonlinear model.



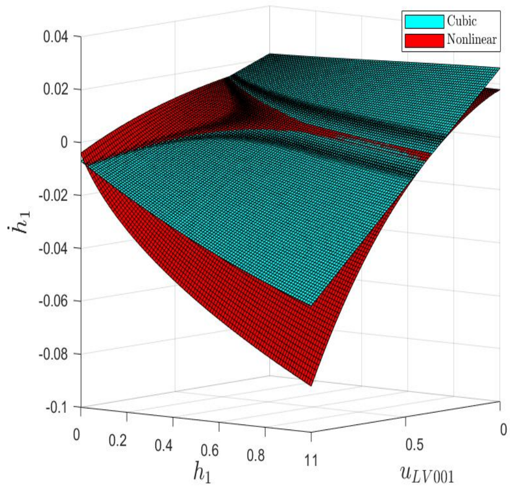
(b) Quadratic and nonlinear model.



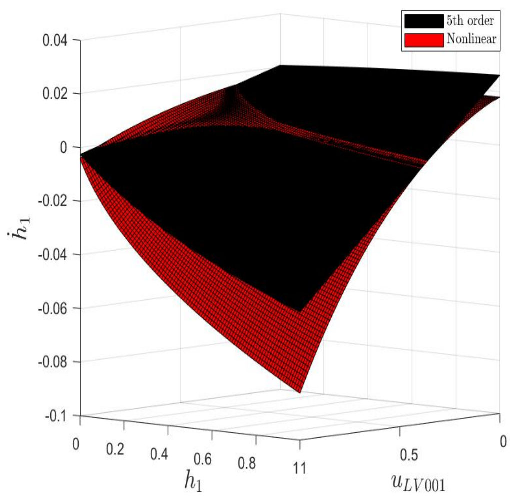
(c) PQ_A and nonlinear model.



(d) PQ_B and nonlinear model.

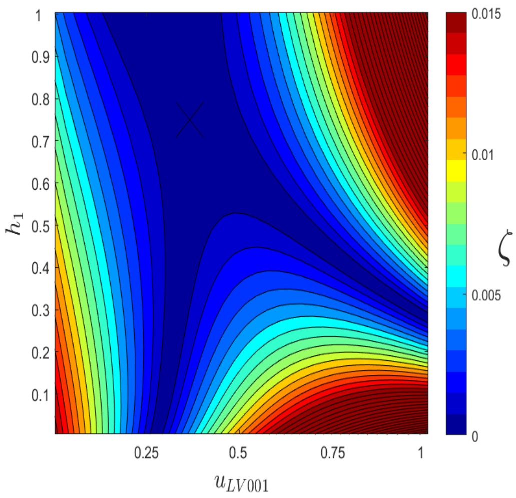


(e) Cubic and nonlinear model.

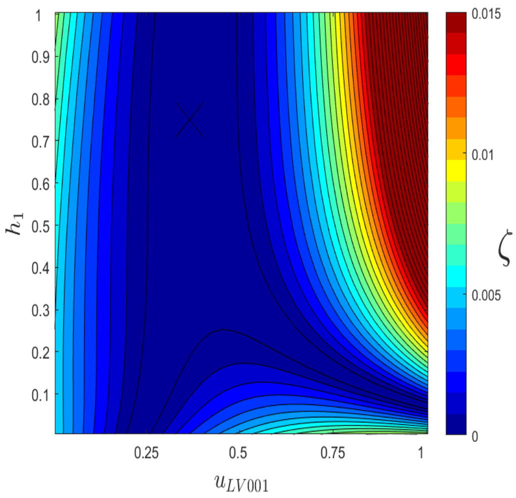


(f) $5th$ order and nonlinear model.

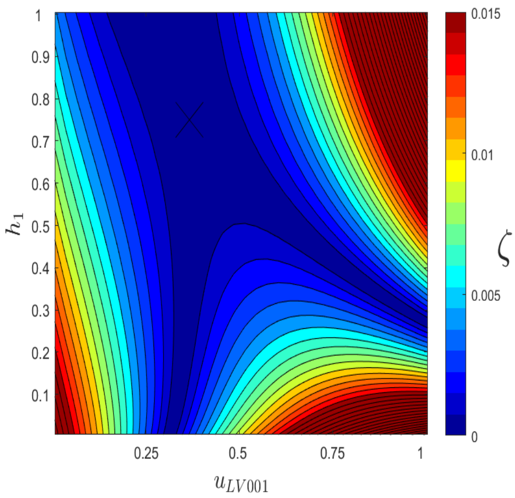
Figure A.1: 3-D plots of the Taylor models and the nonlinear model as reference.



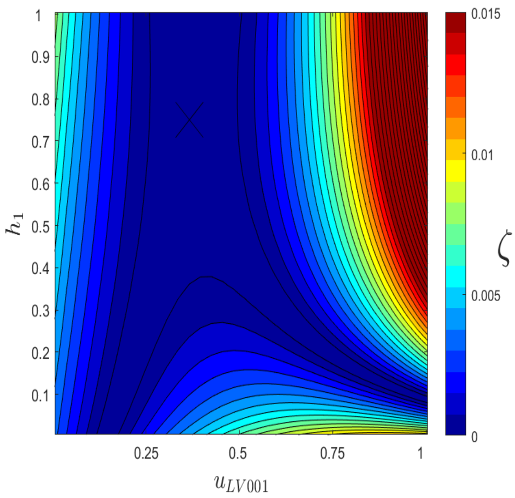
(a) Linear and nonlinear model.



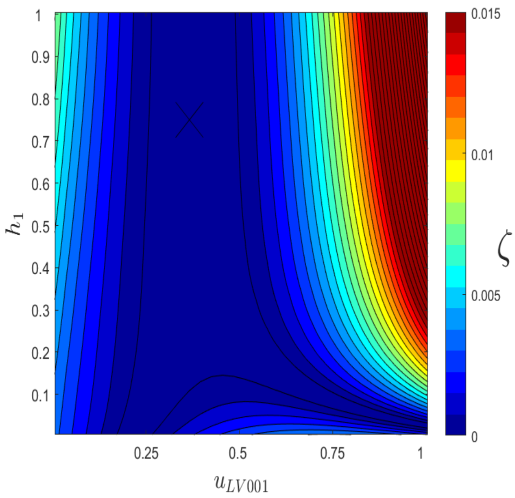
(b) Quadratic and nonlinear model.



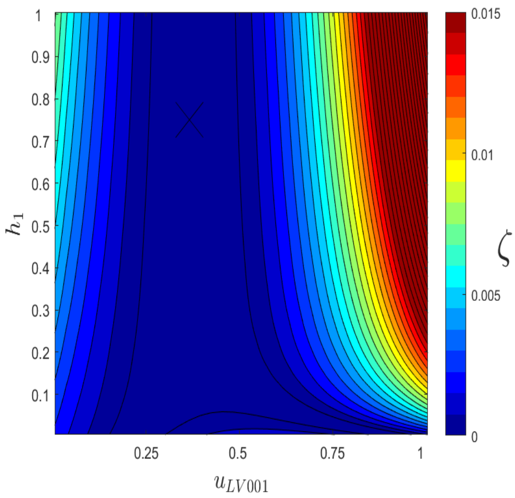
(c) PQ_A and nonlinear model.



(d) PQ_B and nonlinear model.

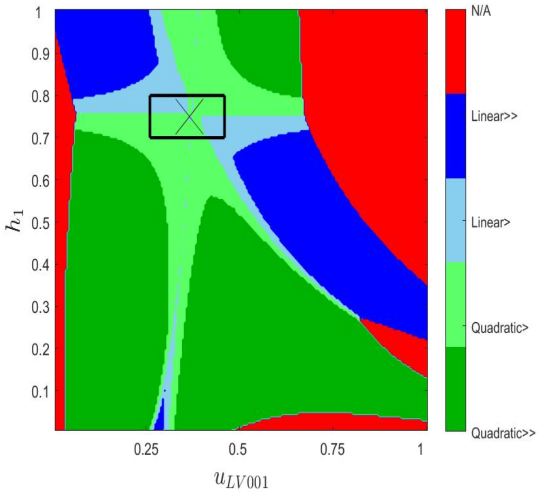


(e) Cubic and nonlinear model.

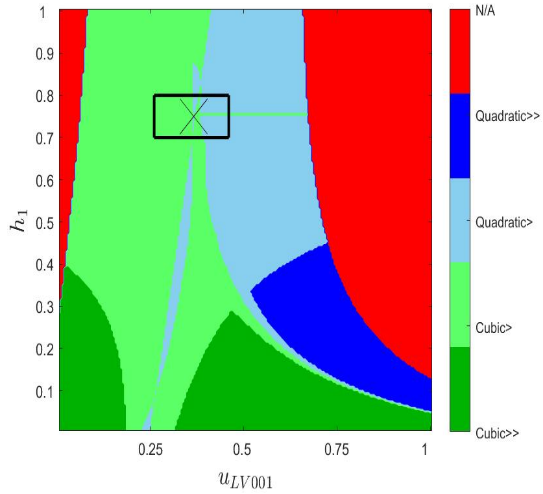


(f) $5th$ order and nonlinear model.

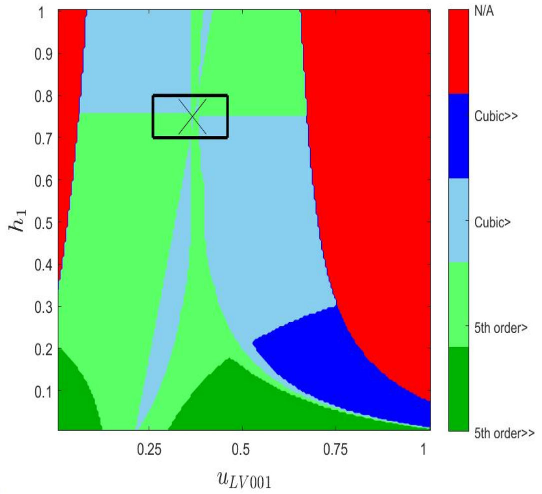
Figure A.2: Contour plots of the Taylor models and the nonlinear model, where ζ is given by (3.32).



(a) $\psi = \zeta_{Linear} - \zeta_{Quadratic}$.



(b) $\psi = \zeta_{Quadratic} - \zeta_{Cubic}$.



(c) $\psi = \zeta_{Cubic} - \zeta_{5th\_Order}$.

Figure A.3: Color maps comparing the different Taylor models.

Case	**Data points**
Quadratic $>>$	0
Quadratic $>$	458
Linear $>$	335
Linear $>>$	0
N/A	0

(a) Data points in sub matrix of ψ from Figure A.3a.

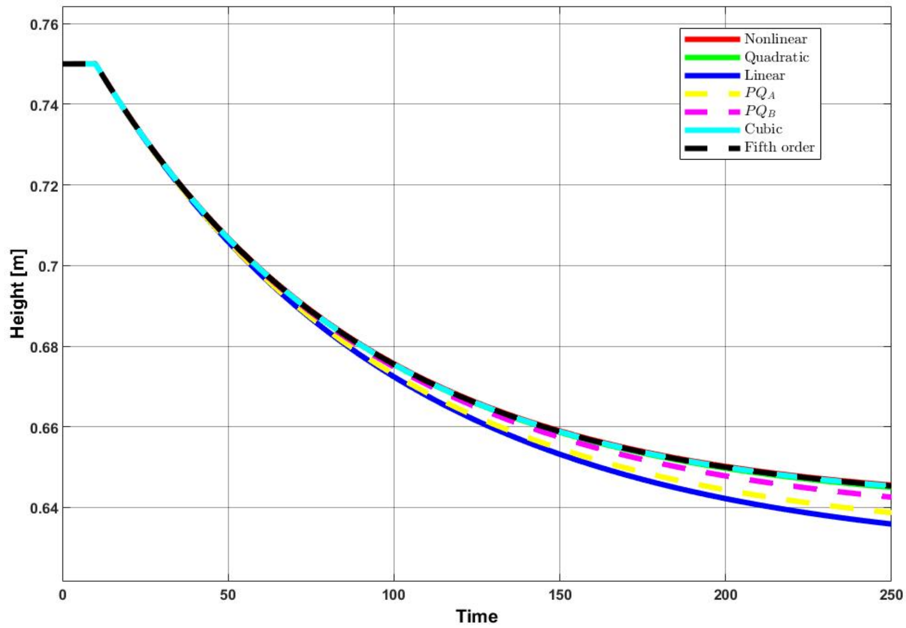
Case	**Data points**
Cubic $>>$	0
Cubic$>$	478
Quadratic$>$	342
Quadratic$>>$	0
N/A	0

(b) Data points in sub matrix of ψ from Figure A.3b.

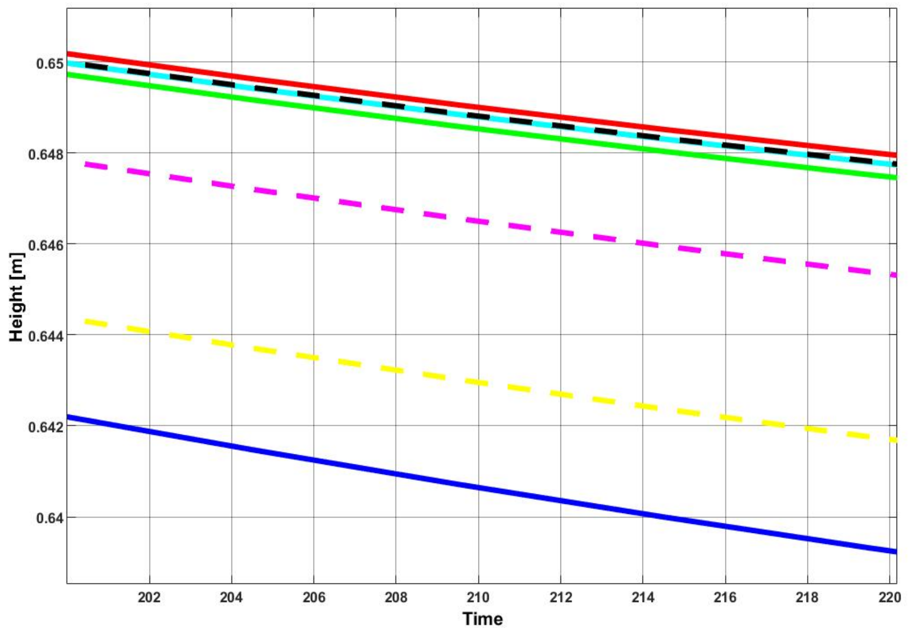
Case	**Data points**
$5th$ order $>>$	0
$5th$ order$>$	437
Cubic$>$	383
Cubic$>>$	0
N/A	0

(c) Data points in sub matrix of ψ from Figure A.3c.

Table A.1: Numerical interpretation of the area within the hollow rectangles in Figure A.3



(a) Step response with an increment of 0.02 at 10 seconds.



(b) Close up of A.4a.

Figure A.4: Model comparison in Simulink.

	IAE	ISE	ITAE	ITSE	ISTE
Linear	1.1293	0.0076354	198.3931	1.4769	299.2148
PQ$_A$	0.84907	0.0041538	146.6415	0.7893	157.9692
PQ$_B$	0.32071	0.00064944	57.5376	0.12877	26.5431
Quadratic	0.071009	2.6413e-05	11.7253	0.0049235	0.98053
Cubic	0.039517	6.9896e-06	5.8743	0.0011394	0.20972
5th order	0.03751	6.2086e-06	5.488	0.00098663	0.17856

Table A.2: Integral performance criteria.

Appendix B

Carleman approximation comparison for Scenario 2

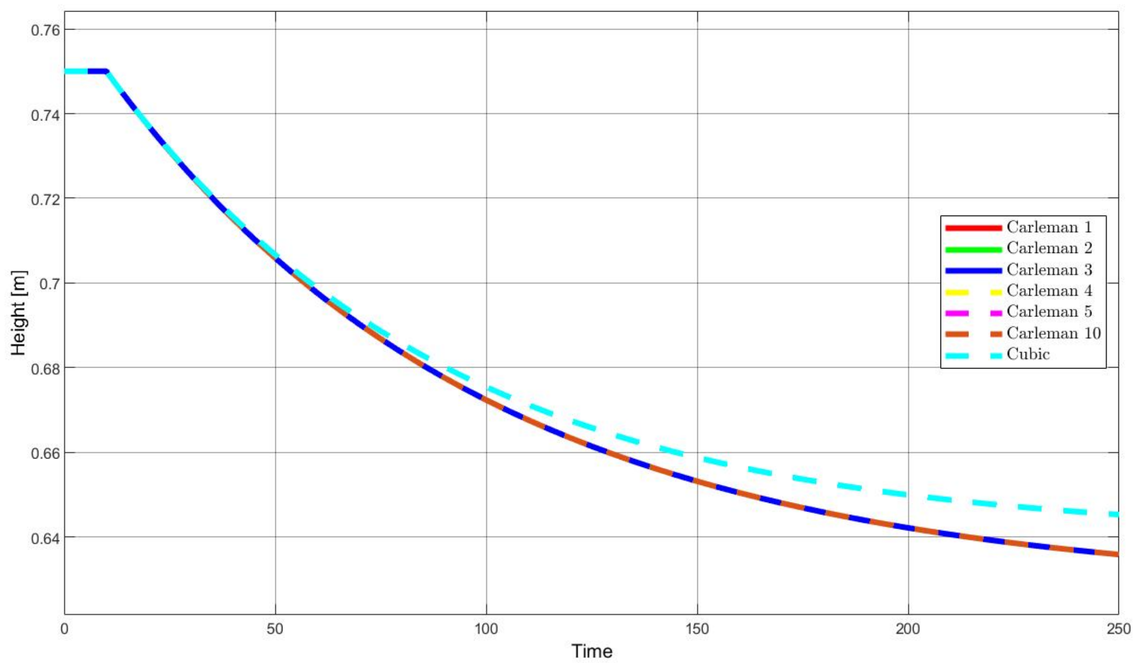


Figure B.1: Step response with an increment in $u_{LV001}(t)$ of 0.02 at 10 seconds.

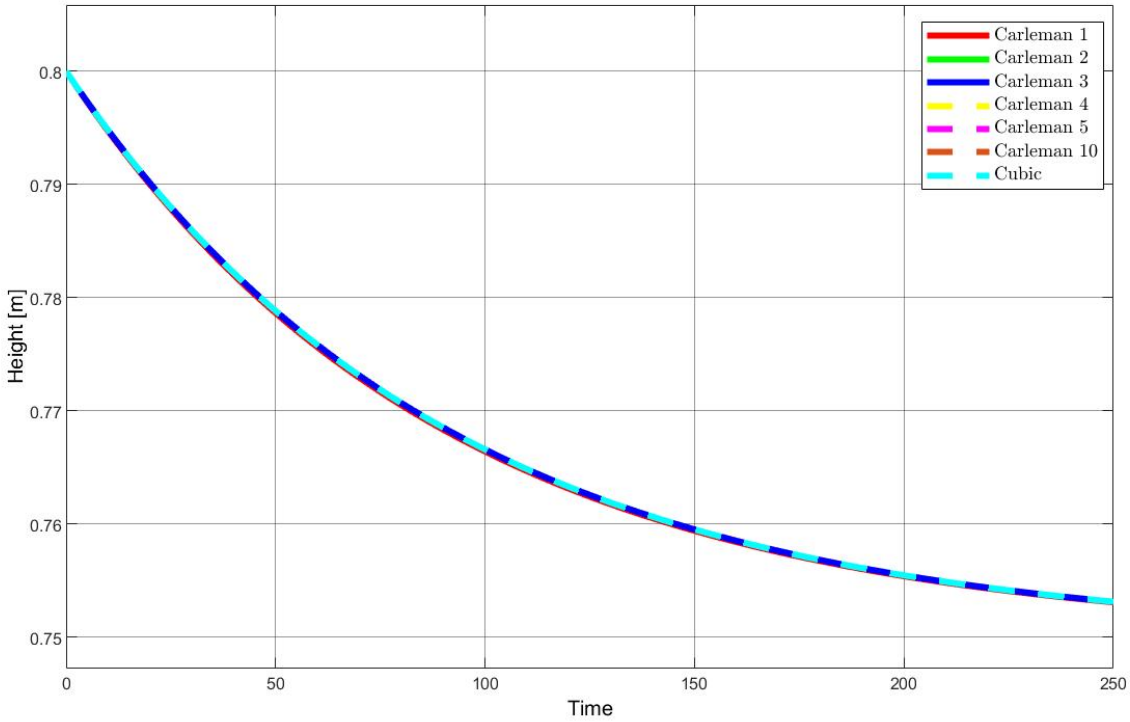


Figure B.2: $\delta = 0.05m$ gives a starting point of 0.8m.

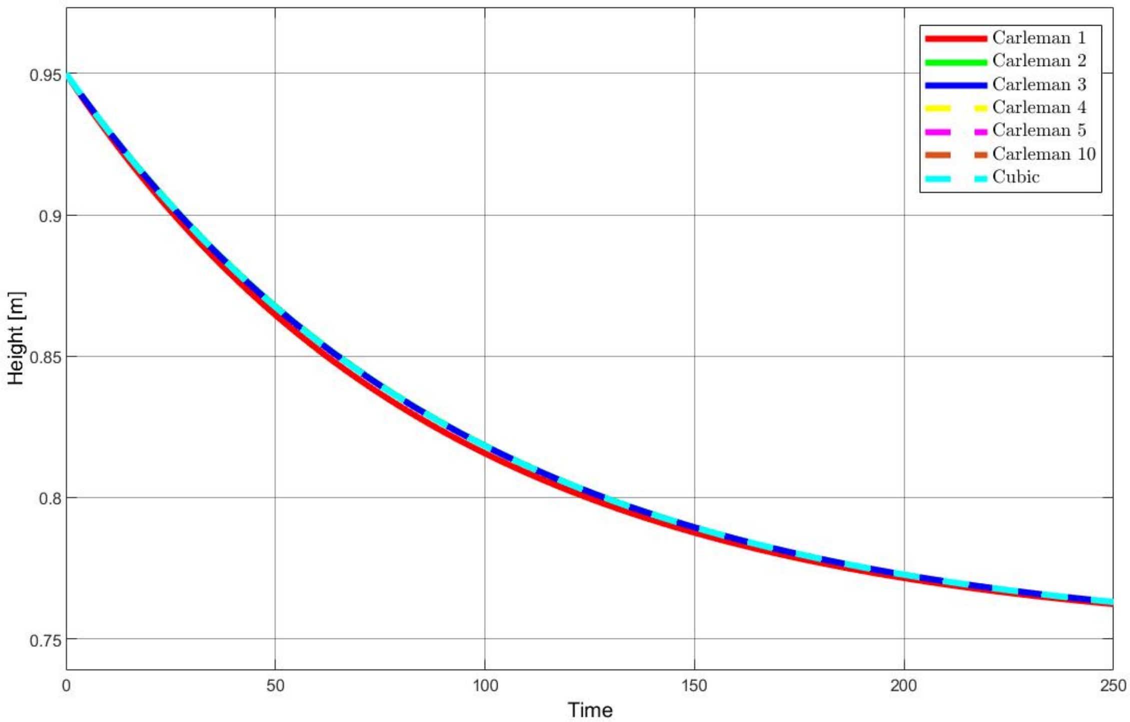


Figure B.3: $\delta = 0.2m$ gives a starting point of 0.95m.

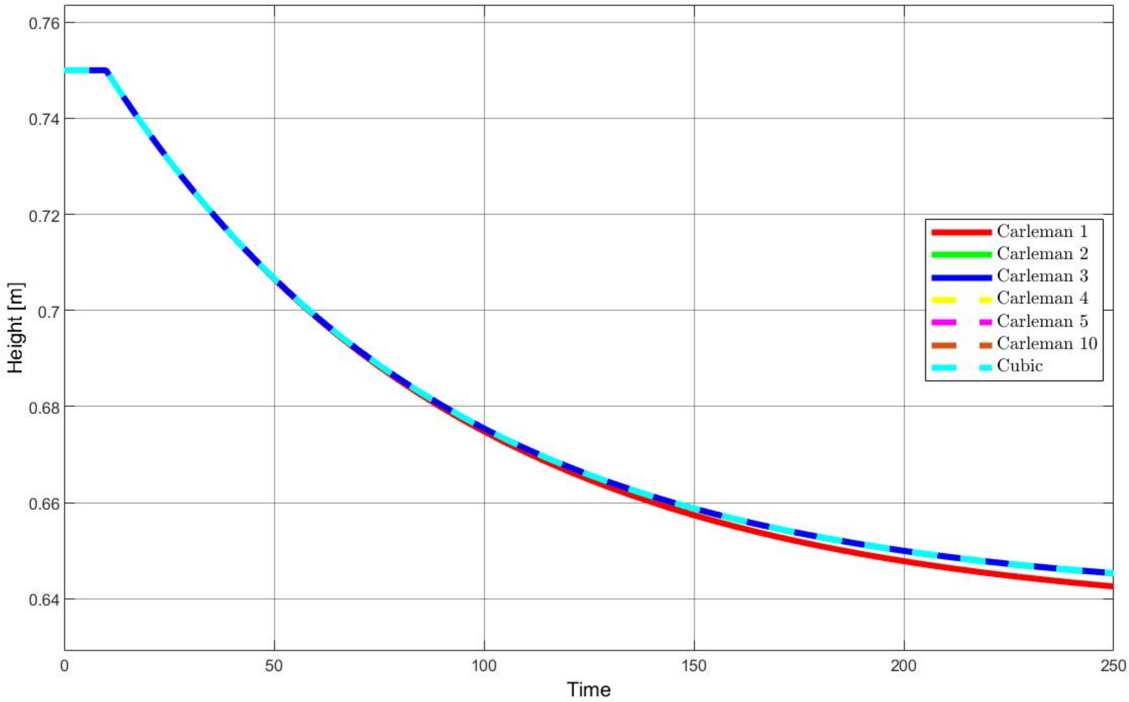
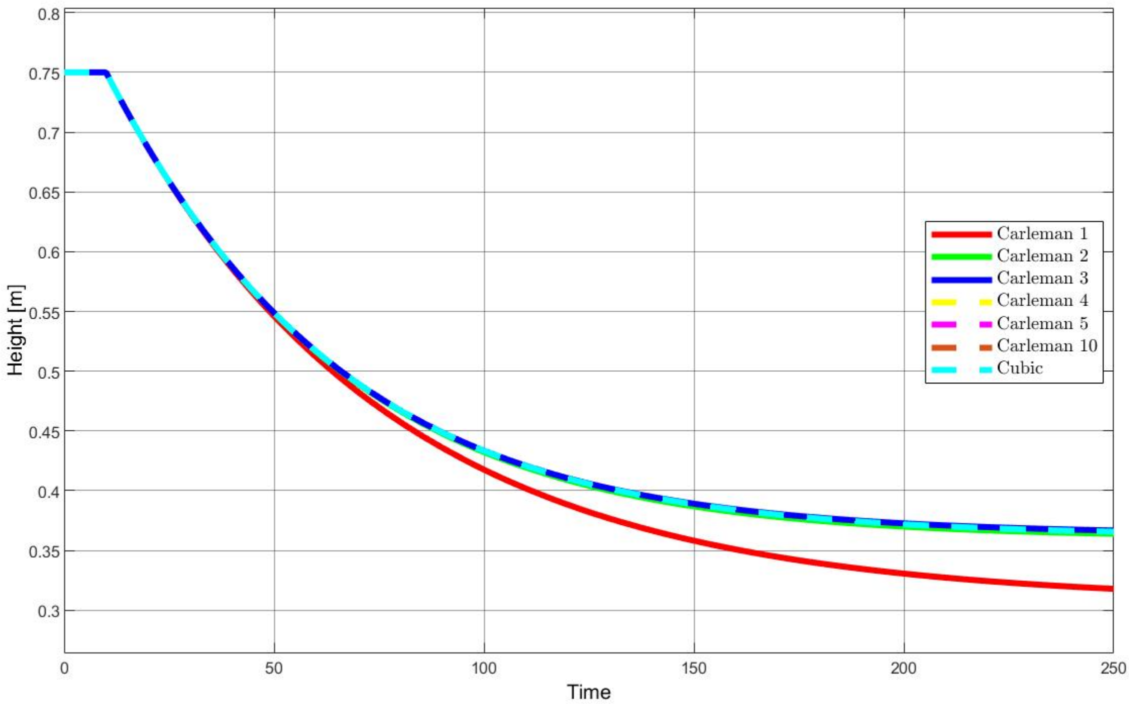


Figure B.4: Simulation of the quadratic Carleman approximations.



Note that the direction of the step is now positive, otherwise, the system would saturate.

Figure B.5: Simulation of the quadratic Carleman approximations with a larger step.

Appendix C

Controller of the quadratic Carleman approximation for Scenario 2

Controller gain for the $4th$ order quadratic Carleman approximation with one input variable ($\delta u_{LV001}(t)$):

$$\mathcal{K} = \begin{bmatrix} 0.3321 & -0.0131 & 0.0004 & -0.0004 \end{bmatrix} \tag{C.1}$$

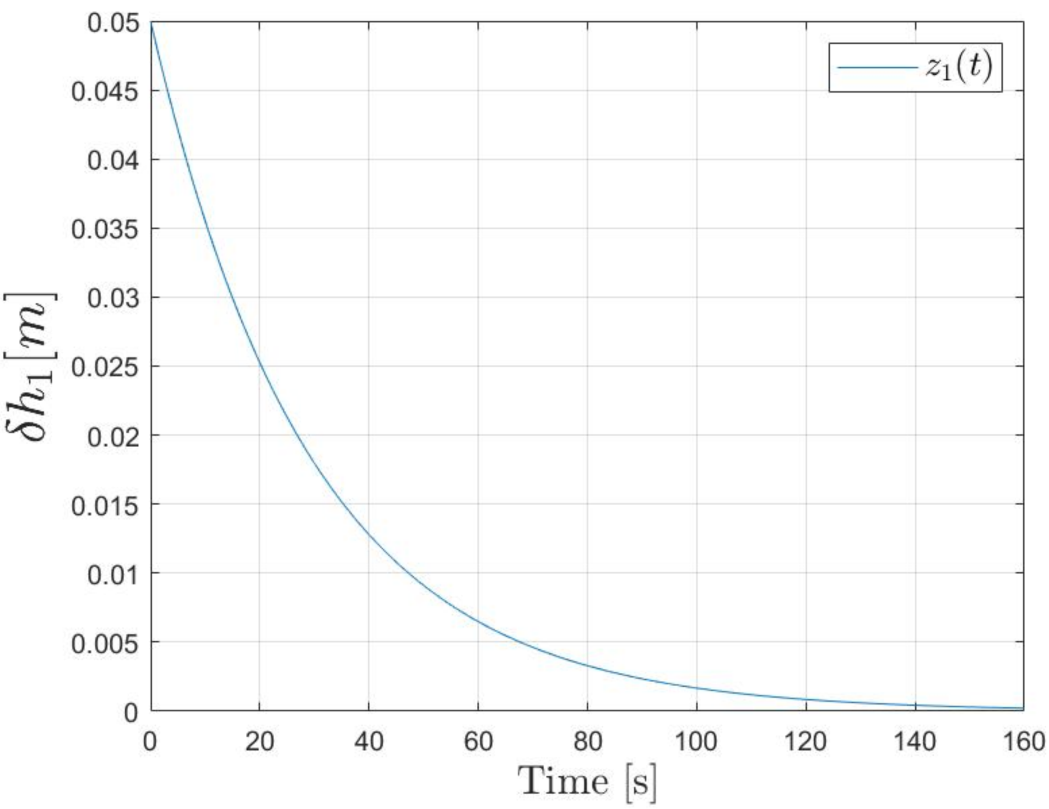


Figure C.1: Simulation of Eq. (6.29), with initial condition equal to $0.05m$.

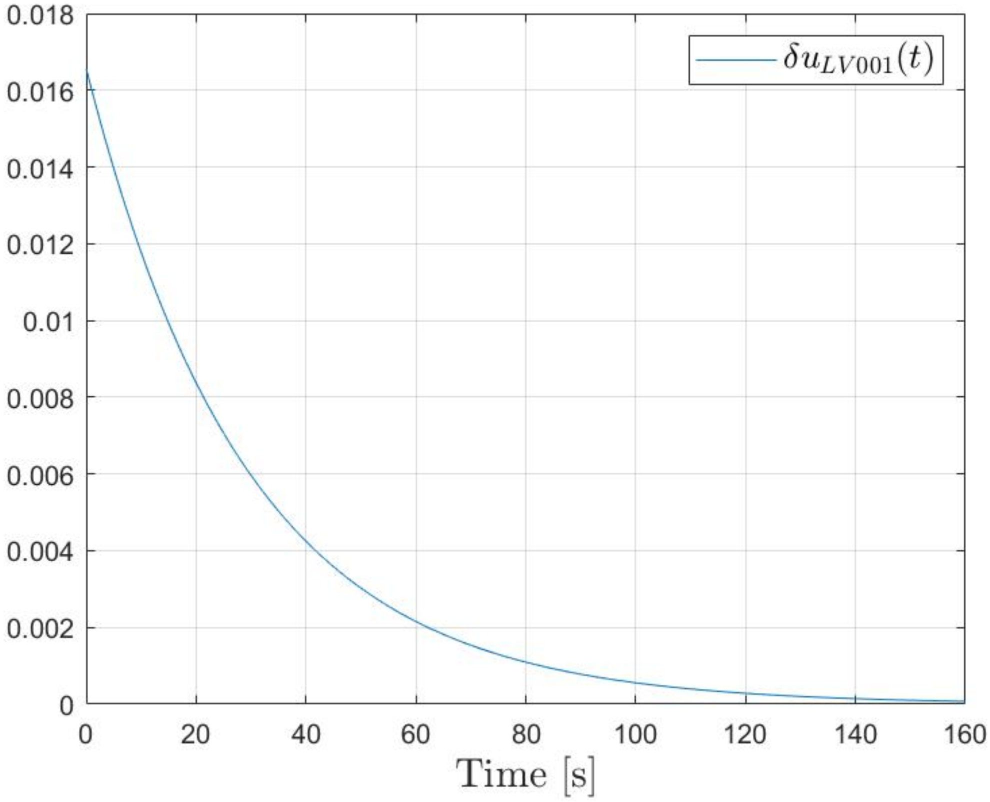


Figure C.2: Signal $\delta u_{LV001}(t)$, calculated by $\mathcal{K}z(t)$.

Controller gain for the $4th$ order quadratic Carleman approximation with two input variables ($\delta u_{LV001}(t)$, $\delta u_{PA001}(t)$):

$$\mathcal{K} = \begin{bmatrix} 0.3234 & -0.0309 & 0.0004 & 0.0004 \\ -0.1176 & -0.0194 & -0.0024 & 0.0011 \end{bmatrix} \tag{C.2}$$

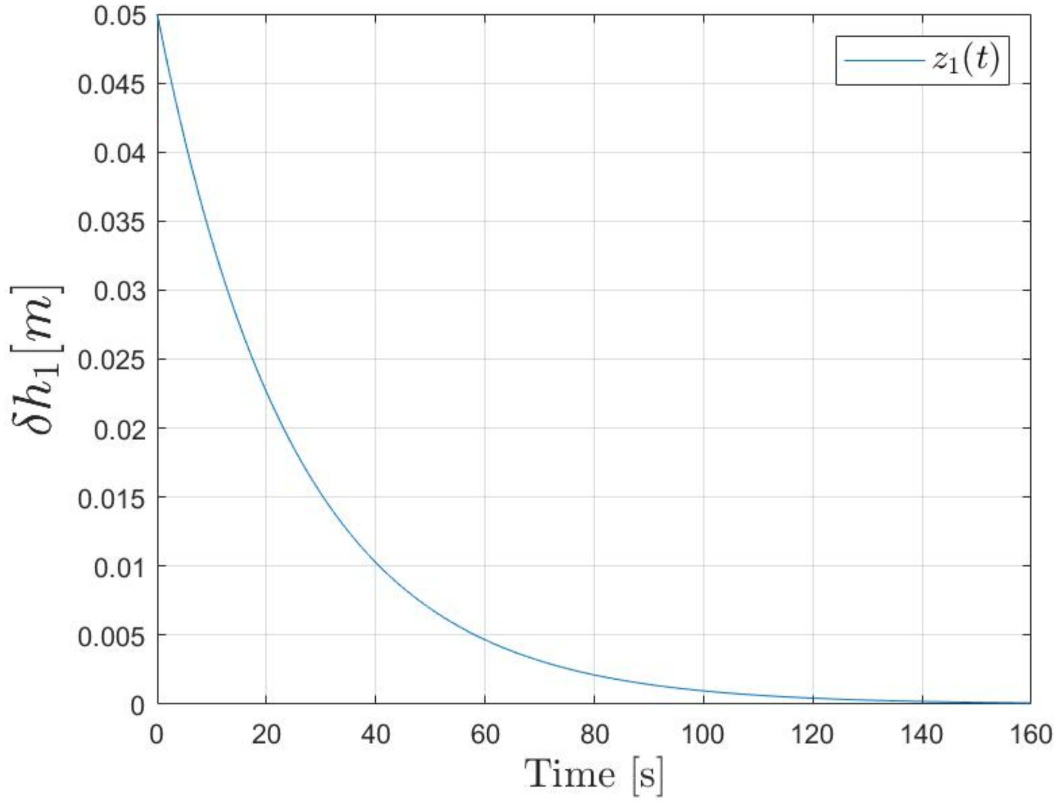


Figure C.3: $4th$ order quadratic Carleman approximation with two actuator.

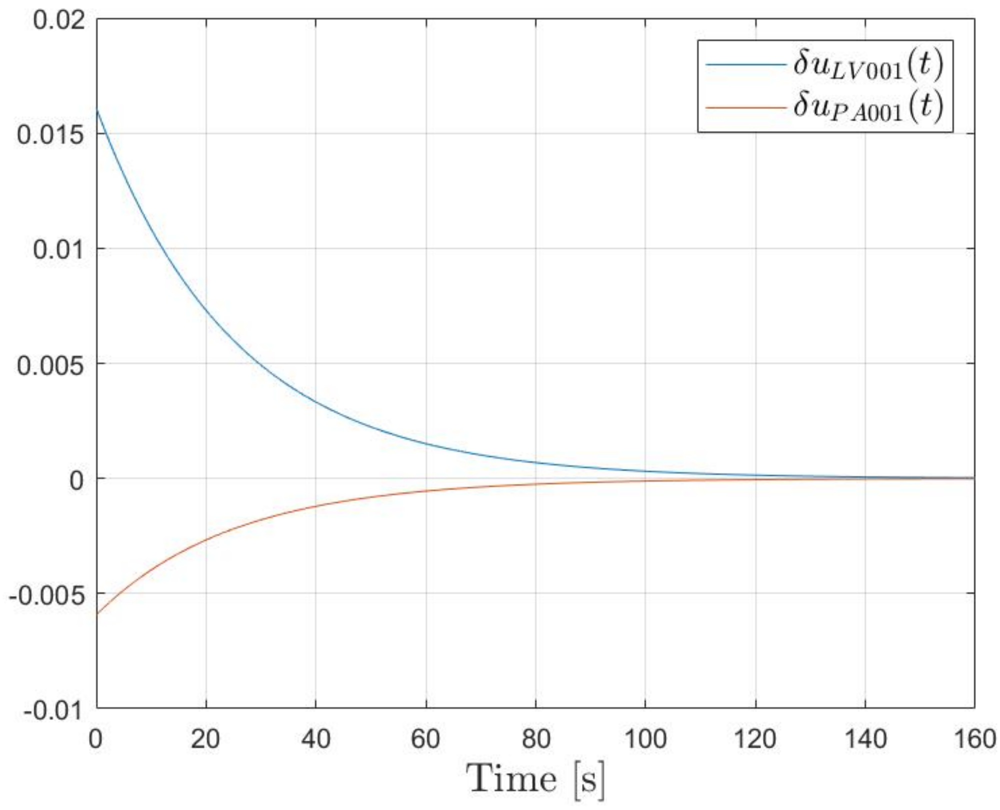


Figure C.4: Signal $u(t)$, calculated by $\mathcal{K}z(t)$.

Controller gain for the $4th$ order quadratic Carleman approximation with two input variables ($\delta u_{LV001}(t)$, $\delta u_{PA001}(t)$), and the inclusion of the convergence rate α:

$$\mathcal{K} = \begin{bmatrix} 1.9147 & -1.6602 & 0.6684 & 1.6311 \\ 0.2835 & 0.0583 & -0.4472 & -0.4170 \end{bmatrix} \tag{C.3}$$

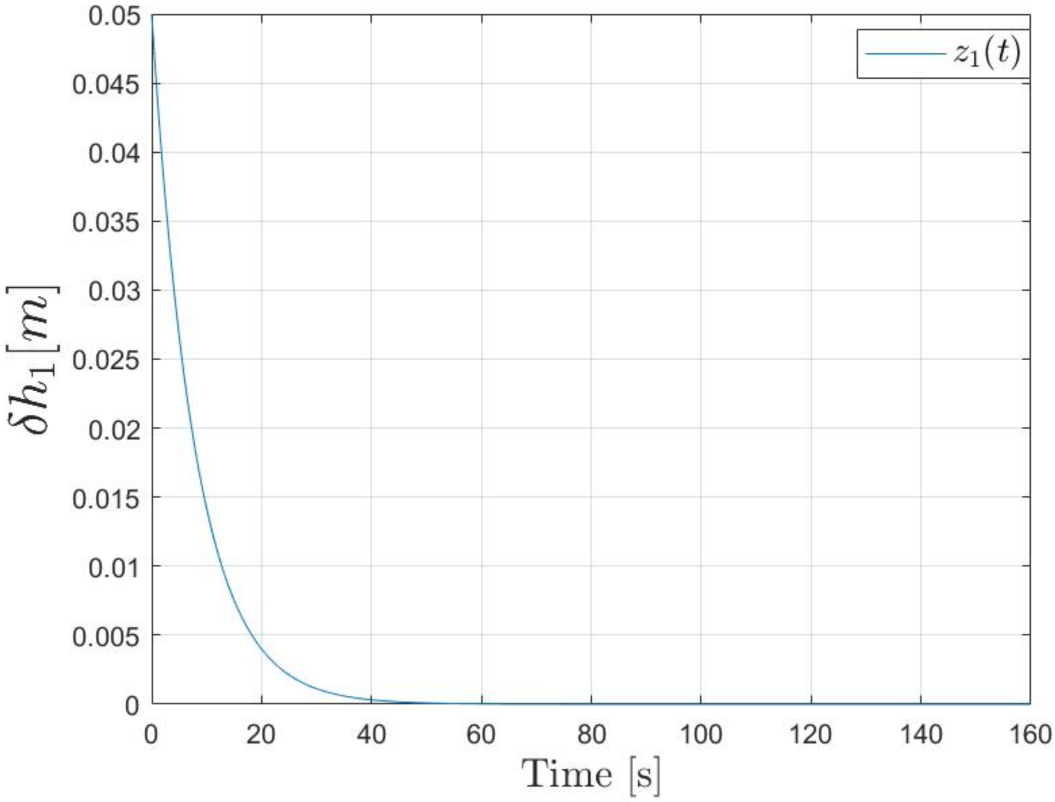


Figure C.5: $4th$ order quadratic Carleman approximation with two actuator and implemented convergence rate.

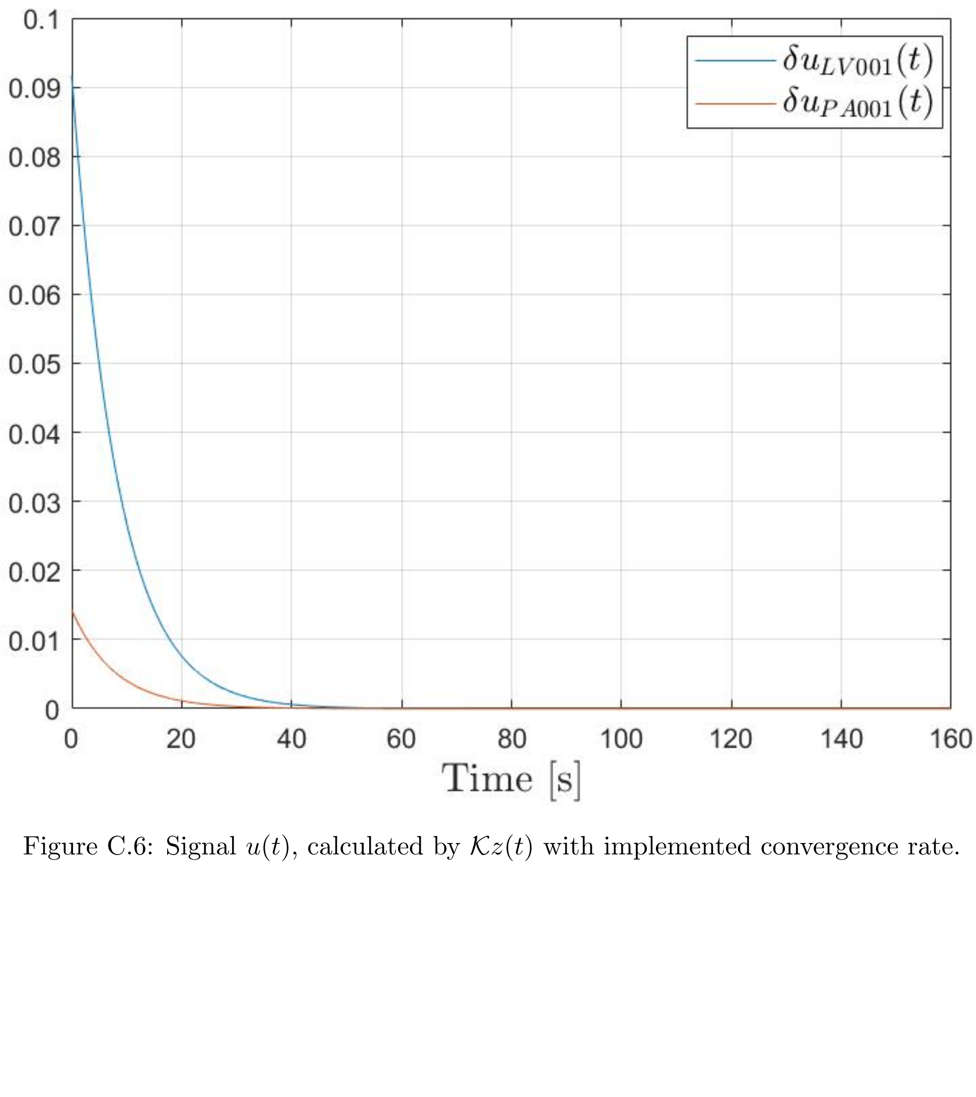


Figure C.6: Signal $u(t)$, calculated by $\mathcal{K}z(t)$ with implemented convergence rate.

Appendix D

Experimental results for Scenario 2

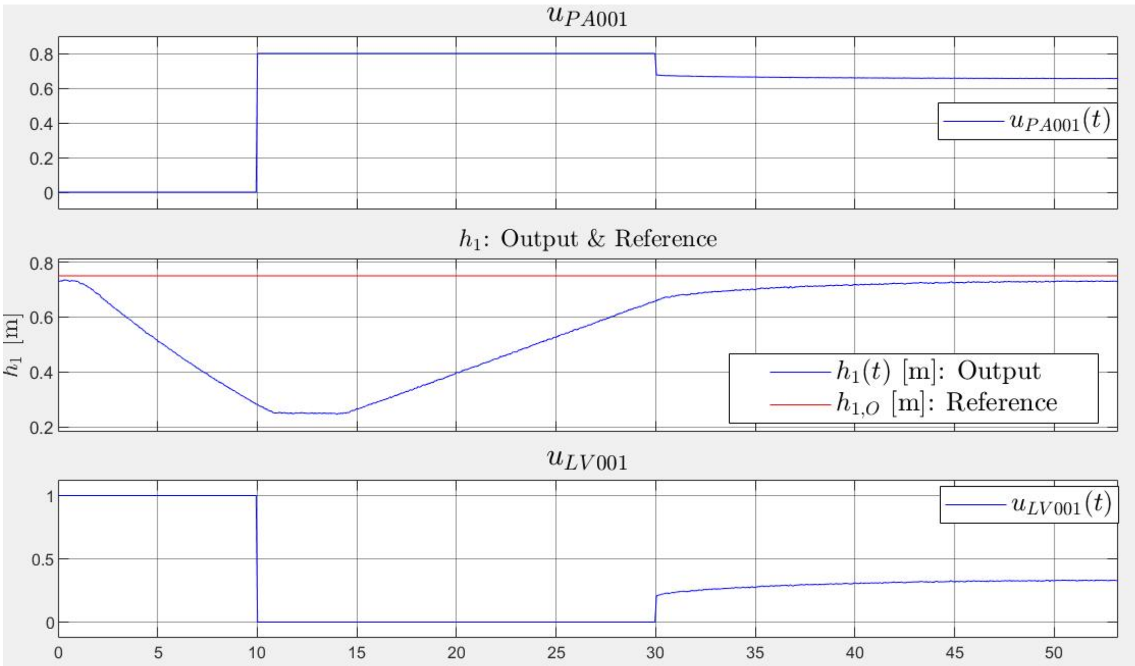


Figure D.1: Experimental results from the quadratic controller.

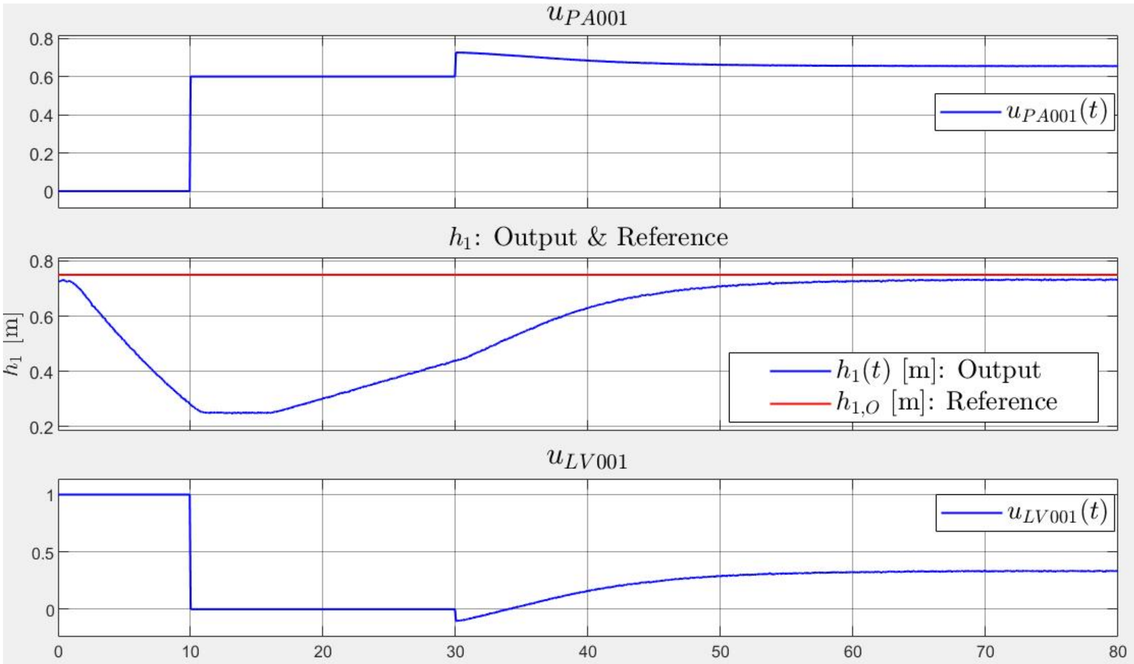


Figure D.2: Experimental results from the quadratic controller.

Appendix E

Simulation of the controller designed with a reduced polytope

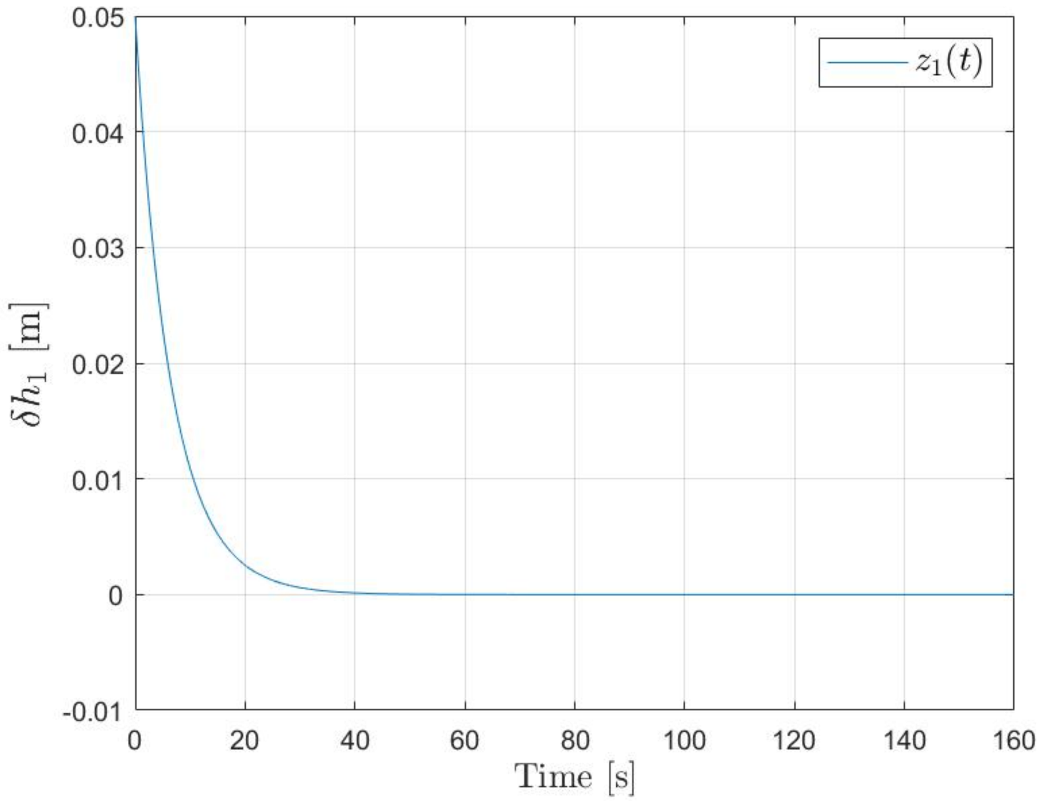


Figure E.1: Simulation of the state $z_1(t)$ for the reduced polytope.

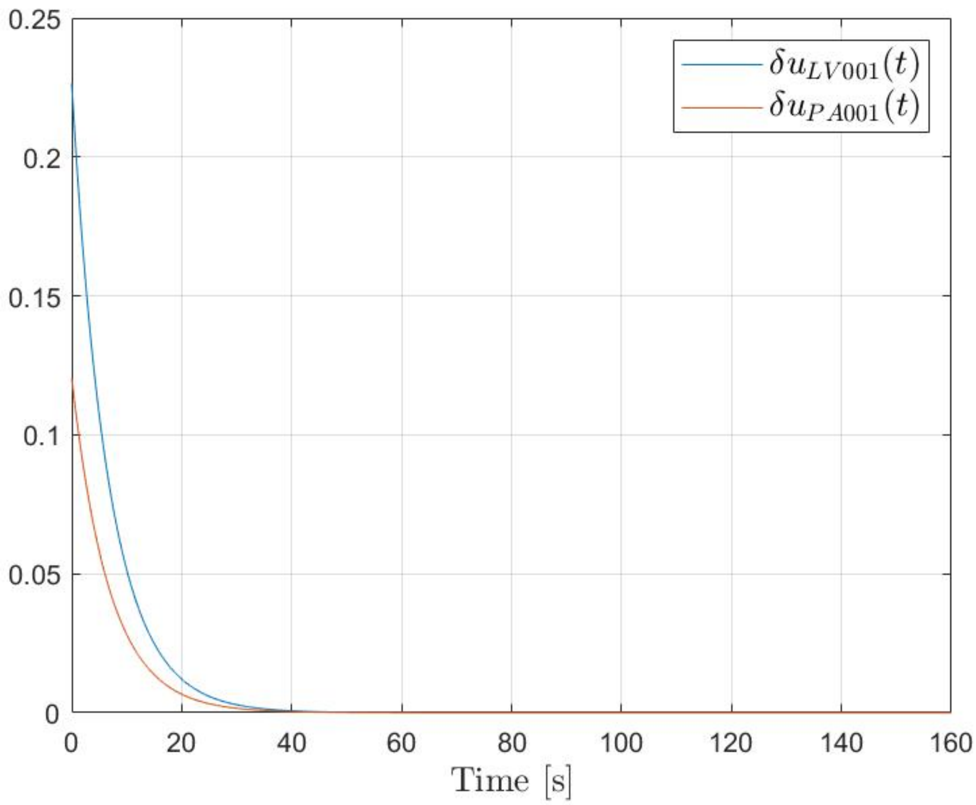


Figure E.2: Simulation of the input variables $\delta u_{LV001}(t)$ and $\delta u_{PA001}(t)$ for the reduced polytope.

Appendix F

MATLAB code and Simulink schemes

F.1 *totank_main.m*

```
%%  Variable definitions
clear all
close all

syms A1 f3 Kv_LV001 f1 rho g h1 h_LV001

%The dynamic model for tank 1
dynamic_model_tank1 = (1/A1)*(f3 - ((Kv_LV001*f1)/3600)*sqrt((rho*g*(h1+h_LV001))
    /100000));

% Calculating the partial derivatives (symbolic)
par_der_h1 = diff(dynamic_model_tank1, h1);          %   df/dh1
par_der_f1 = diff(dynamic_model_tank1, f1);          %   df/df1
par_der_f3 = diff(dynamic_model_tank1, f3);          %   df/df3
par_der_h1_h1 = diff(par_der_h1, h1);                %   d^2f/dh1^2
par_der_f1_f1 = diff(par_der_f1, f1);                %   d^2f/df1^2
par_der_f3_f3 = diff(par_der_f3, f3);                %   d^2f/df3^2
par_der_h1_f1 = diff(par_der_h1, f1);                %   d^2f/dh1df1
par_der_h1_f3 = diff(par_der_h1, f3);                %   d^2f/dh1df3
par_der_f1_f3 = diff(par_der_f1, f3);                %   d^2f/df1df3

par_der_h1_h1_h1 = diff(par_der_h1_h1, h1);          %   d^3f/dh1^3
par_der_h1_h1_f1 = diff(par_der_h1_h1, f1);          %   d^3f/dh1^2df1

par_der_h1_h1_h1_h1 = diff(par_der_h1_h1_h1,h1);     %   d^4f/dh1^4
par_der_h1_h1_h1_f1 = diff(par_der_h1_h1_h1,f1);     %    d^4f/dh1^3df1

par_der_h1_h1_h1_h1_h1 = diff(par_der_h1_h1_h1_h1,h1);  % d^5f/dh1^5
par_der_h1_h1_h1_h1_f1 = diff(par_der_h1_h1_h1_h1,f1);  % d^5f/dh1^4df1

scenario = 1;
if scenario == 1

    % Setting the working point for h1 and f3
    h1_arb = 0.25;
    f3_arb = 0.0001783;

    % Inserting every known variable in equation dynamic_model_tank1
    a = subs(dynamic_model_tank1, [A1 Kv_LV001 rho g h_LV001 h1 f3],...
        [0.01 11.25 1000 9.81 0.05 h1_arb f3_arb]);

    % Solving equation 'a' wrt f1. This gives f1_arb
    f1_arb = double(solve(a,f1));

    % Since f3_arb is chosen, we can find upa_arb from the pump-characteristics
```

```
    upa_arb = 0.65;
    % We have found f1_arb. Using this value in the valve-characteristics gives us ulv_arb
    ulv_arb = 0.5159;

    df1_duLV001    = 0.98;  %   Manually calculated in simulink. delta_f1 / delta_ulv
    df3_dPA001   = 0.00052; %   Manually calculated in simulink. delta_f3 / delta_upa

elseif scenario == 2
    % Setting the working point for h1 and f3
    h1_arb = 0.75;
    f3_arb = 0.0001783;

    % Inserting every known variable in equation dynamic_model_tank1
    a = subs(dynamic_model_tank1, [A1 Kv_LV001 rho g h_LV001 h1 f3],...
        [0.01 11.25 1000 9.81 0.05 h1_arb f3_arb]);

    % Solving equation 'a' wrt f1. This gives f1_arb
    f1_arb = double(solve(a,f1));

    % Since f3_arb is chosen, we can find upa_arb from the pump-characteristics
    upa_arb = 0.65;
    % We have found f1_arb. Using this value in the valve-characteristics gives us ulv_arb
    ulv_arb = 0.3666;

    df1_duLV001    = 0.78;  %   Manually calculated in simulink. delta_f1 / delta_ulv
    df3_dPA001   = 0.00052; %   Manually calculated in simulink. delta_f3 / delta_upa
end

    % Calculating the partial derivatives used in the Taylor series expansion
% (Value at working point!)
par_der_h1_verdi = double((subs(par_der_h1,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

par_der_uLV001_verdi = double(subs(par_der_f1 * df1_duLV001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_uPA001_verdi = double(subs(par_der_f3 * df3_dPA001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_h1_h1_verdi = double(subs(par_der_h1_h1,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_uLV001_uLV001_verdi = double(subs(par_der_f1_f1 * df1_duLV001^2,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_uPA001_uPA001_verdi = double(subs(par_der_f3_f3 * df3_dPA001^2,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_h1_uLV001_verdi = double(subs(par_der_h1_f1 * df1_duLV001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_h1_uPa001_verdi = double(subs(par_der_h1_f3 * df3_dPA001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_uLV001_uPA001_verdi = double(subs(par_der_f1_f3 * df1_duLV001 * df3_dPA001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

```

```
par_der_h1_h1_h1_verdi = double((subs(par_der_h1_h1_h1,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

par_der_h1_h1_uLV001_verdi = double((subs(par_der_h1_h1_f1 * df1_duLV001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

par_der_h1_h1_h1_h1_verdi = double((subs(par_der_h1_h1_h1_h1,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

par_der_h1_h1_h1_uLV001_verdi = double((subs(par_der_h1_h1_h1_f1 * df1_duLV001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

par_der_h1_h1_h1_h1_h1_verdi = double((subs(par_der_h1_h1_h1_h1_h1,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

par_der_h1_h1_h1_h1_uLV001_verdi = double((subs(par_der_h1_h1_h1_h1_f1 *
    df1_duLV001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));



% The following is a standard code that imports the characteristics of the
% pump and valve. This code also defines some constants. (From Reguleringsteknikk)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% data om vann og tyngdekraft
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rho = 1000;                % tetthet vann             [kg/m^3]
g   = 9.81;                % tyngdens akselerasjon     [m/s^2]
c_p = 4200;                % varmekapasitet  vann      [j/kg*K]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tank 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kv_LV001    = 11.25;         % ventilkonstant LV001  [m^3/h] ved 1 bar trykkfall
h_LV001     = 0.05;          % h yde til LV001 [m]
h1_max      = 1;             % maks h yde tank 1 [m]
h1_min      = 0.13;          % min h yde tank 1 [m]
A1          = 0.01;          % areal tank 1    [m^2]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tank 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kv_LV002    = 11.25;         % ventilkonstant LV002     [m3/h]
h_LV002     = 0.25;          % h yde fra bunn av tank 2 til LV002
h2_max      = 0.4;           % maks h yde tank 2    [m]
h2_min      = 0.02;          % min h yde tank 2    [m]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Last inn p drag og m linger
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load tankData_1    % det finnes ogs   et datasett som heter tankData_2
load tankData_2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pumpekarakteristikk
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_PA001 = [0.00  0.45  0.46  0.47  0.48  0.49  0.50  0.55...
           0.60  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00];
q_PA001 = [0.00  0.00  1.25  2.25  3.15  3.75  4.40  6.75...
           8.75 10.70 12.25 13.75 15.15 16.50 18.00 19.20 20.00];

q_PA001 = q_PA001/60000;  % liter/time -> m3/s

close all
figure
plot(u_PA001, q_PA001,'*-')
```

```matlab
title('Pump characteristic')
xlabel('Control signal u_{PA001}(t) to pump PA001')
ylabel('Volume flow q_{PA001}(t) through PA001 [m^3/s]')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ventilkarakteristikk
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_LV001 = 0:0.03:1;
f_LV001 = (exp(u_LV001.^1.2)-1)/(exp(1)-1);
u_LV002 = u_LV001;
f_LV002 = f_LV001;

figure
plot(u_LV001,f_LV001,'*-')
title('Valve characteristic for LV001 og LV002')
xlabel('Control signal u_{LV001}(t)')
ylabel('f(u_{LV001}(t))')

%End of the standard code from Reguleringsteknikk

%%  NonLinear Model 3d-Plot

close all
figure
[ulv,h1]=meshgrid(0:0.005:1,[0:0.005:1]); %0.005

%We want to mesh ulv and h1, but the nonlinear model uses f1 in it's
%calculation. The folowing finds the corresponding values for f1.
resultat = sim('f1_invers', 3000);
f1 = resultat.f1_sim.Data;

% Calculates the NonLinear model
ydot = 1/A1*(f3_arb - 1/3600 *Kv_LV001.*f1.*sqrt(rho*g.*(h1+0.05)/100000));

NonLinear = surf(ulv,h1,ydot,'FaceColor','r');      % Draws the NonLinear model
rotate3d
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
zlabel('$\dot{h}_1$', 'Interpreter','latex','fontsize',16)
title('NonLinear Model 3D-Plot')
legend(NonLinear, ['NonLinear' newline 'Model'])
hold on

Zero_Plane = surf(ulv,h1,-zeros(size(ulv)),'FaceColor','black');
legend([NonLinear Zero_Plane], {['NonLinear' newline 'Model'], ['h_{1}-u_{LV001}
    plane']})

%%  Quadratic Model 3d-Plot

close all
figure

NonLinear = surf(ulv,h1,ydot,'FaceColor','r');      % Draws the NonLinear model
rotate3d
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
zlabel('$\dot{h}_1$', 'Interpreter','latex','fontsize',16)
title('Quadratic Model @0.25m 3D-Plot')
hold on

y_quadratic = par_der_h1_verdi.*(h1-h1_arb)...
    + par_der_uLV001_verdi.*(ulv - ulv_arb) ...
    + par_der_h1_h1_verdi.*0.5.*(h1-h1_arb).^2 ...
    + par_der_h1_uLV001_verdi.*(h1-h1_arb).*(ulv-ulv_arb);  % Calculates the
    Quadratic model

Quadratic = surf(ulv,h1,y_quadratic, 'FaceColor', 'g');     % Draws the Quadratic
    model
legend([Quadratic NonLinear], {['Quadratic' newline 'Model'], ['NonLinear' newline
    'Model']})

%%  Linear 3d-Plot
```

```
close all
figure

NonLinear = surf(ulv,h1,ydot,'FaceColor','r');     % Draws the NonLinear model
rotate3d
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
zlabel('$\dot{h}_1$', 'Interpreter','latex','fontsize',16)
title('Linear Model @0.25m 3D-Plot')
hold on

y_linear = par_der_h1_verdi.*(h1-h1_arb)...
            + par_der_uLV001_verdi.*(ulv-ulv_arb); % Calculates the Linear model

Linear = surf(ulv,h1,y_linear, 'FaceColor', 'b');  % Draws the Linear model
legend([Linear NonLinear], {['Linear' newline 'Model'], ['NonLinear'  newline '
    Model']})

%%  Partially Quadratic (h1xh1) Model 3d-Plot

close all
figure

NonLinear = surf(ulv,h1,ydot,'FaceColor','r');     % Draws the NonLinear model
rotate3d
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
zlabel('$\dot{h}_1$', 'Interpreter','latex','fontsize',16)
title('Partially Quadratic(h1xh1) Model @0.25m 3D-Plot')
hold on

y_parQuad_h1xh1 = par_der_h1_verdi.*(h1-h1_arb)...
    + par_der_uLV001_verdi.*(ulv-ulv_arb)...
    + par_der_h1_h1_verdi.*0.5.*(h1-h1_arb).^2; %Calculates the Partially Quadratic
     (h1xh1) model

%Draws the Partially Quadratic (h1xh1) model
PartQuad_h1xh1 = surf(ulv,h1,y_parQuad_h1xh1, 'FaceColor', 'y');
legend([PartQuad_h1xh1 NonLinear],...
    {['Partially' newline 'Quadratic' newline '(h1xh1) Model'],...
    ['NonLinear' newline 'Model']})

%%  Partially Quadratic (h1xulv) Model 3d-Plot

close all
figure

NonLinear = surf(ulv,h1,ydot,'FaceColor','r');     % Draws the NonLinear model
rotate3d
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
zlabel('$\dot{h}_1$', 'Interpreter','latex','fontsize',16)
title('Partially Quadratic (h1xuLV) Model @0.25m 3D-Plot')
hold on

%Calculates the Partially Quadratic (h1xulv) model
y_parQuad_h1xulv = par_der_h1_verdi.*(h1-h1_arb)...
    + par_der_uLV001_verdi.*(ulv-ulv_arb)...
    +  par_der_h1_uLV001_verdi.*(h1-h1_arb).*(ulv-ulv_arb);

%Draws the Partially Quadratic (h1xulv) model
PartQuad_h1xulv = surf(ulv,h1,y_parQuad_h1xulv, 'FaceColor', 'm');
legend([PartQuad_h1xulv NonLinear],...
    {['Partially' newline 'Quadratic' newline '(h1xuLV) Model'],...
    ['NonLinear' newline 'Model']})

%%  Cubic Model 3d-Plot

close all
figure

NonLinear = surf(ulv,h1,ydot,'FaceColor','r');     % Draws the NonLinear model
```

```
rotate3d
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
zlabel('$\dot{h}_1$', 'Interpreter','latex','fontsize',16)
title('Cubic Model @0.25m 3D-Plot')
hold on

y_cubic = par_der_h1_verdi.*(h1-h1_arb)...
    + par_der_uLV001_verdi.*(ulv - ulv_arb) ...
    + par_der_h1_h1_verdi.*0.5.*(h1-h1_arb).^2 ...
    + par_der_h1_uLV001_verdi.*(h1-h1_arb).*(ulv-ulv_arb)...
    + (1/6).*par_der_h1_h1_h1_verdi.*(h1-h1_arb).^3 ...
    + (1/2).*par_der_h1_h1_uLV001_verdi.*(h1-h1_arb).^2.*(ulv-ulv_arb); %
    Calculates the Cubic model

Cubic = surf(ulv,h1,y_cubic, 'FaceColor', 'cyan');    % Draws the Cubic model
legend([Cubic NonLinear], {['Cubic' newline 'Model'], ['NonLinear' newline 'Model'
    ]})

%%  5th order T.S Model 3d-Plot

close all
figure

NonLinear = surf(ulv,h1,ydot,'FaceColor','r');     % Draws the NonLinear model
rotate3d
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
zlabel('$\dot{h}_1$', 'Interpreter','latex','fontsize',16)
title('5th Order Model @0.25m 3D-Plot')
hold on

% Calculates the 5th order model
y_fifth_order = par_der_h1_verdi.*(h1-h1_arb)...
    + par_der_uLV001_verdi.*(ulv - ulv_arb) ...
    + par_der_h1_h1_verdi.*0.5.*(h1-h1_arb).^2 ...
    + par_der_h1_uLV001_verdi.*(h1-h1_arb).*(ulv-ulv_arb)...
    + (1/6).*par_der_h1_h1_h1_verdi.*(h1-h1_arb).^3 ...
    + (1/2).*par_der_h1_h1_uLV001_verdi.*(h1-h1_arb).^2.*(ulv-ulv_arb)...
    + (1/24).*par_der_h1_h1_h1_h1_verdi.*(h1-h1_arb).^4 ...
    + (1/6).*par_der_h1_h1_h1_uLV001_verdi.*(h1-h1_arb).^3.*(ulv-ulv_arb) ...
    + (1/120).*par_der_h1_h1_h1_h1_h1_verdi.*(h1-h1_arb).^5 ...
    + (1/24).*par_der_h1_h1_h1_h1_uLV001_verdi.*(h1-h1_arb).^4.*(ulv-ulv_arb);

fifth_Order = surf(ulv,h1,y_fifth_order, 'FaceColor', 'k');    % Draws the 5th
    order model
legend([fifth_Order NonLinear], {['5th Order' newline 'Model'], ['NonLinear'
    newline 'Model']})

%%  Mix 3d-Plot

close all
figure

NonLinear = surf(ulv,h1,-ydot,'FaceColor','r');     % Draws the NonLinear model
rotate3d
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
zlabel('$\dot{h}_1$', 'Interpreter','latex','fontsize',16)
title('Mix @0.25m 3D-Plot')
hold on

% Draws the Linear model
Linear = surf(ulv,h1,-y_linear, 'FaceColor', 'b');

% Draws the Quadratic model
Quadratic = surf(ulv,h1,-y_quadratic, 'FaceColor', 'g');

% Draws the Partially Quadratic (h1xh1) model
PartQuad_h1xh1 = surf(ulv,h1,-y_parQuad_h1xh1, 'FaceColor', 'y');

% Draws the Partially Quadratic (h1xulv) model
PartQuad_h1xulv = surf(ulv,h1,-y_parQuad_h1xulv, 'FaceColor', 'm');
```

```matlab

% Draws the Cubic model
Cubic = surf(ulv,h1,-y_cubic, 'FaceColor', 'cyan');

% Draws the 5th order model
fifth_Order = surf(ulv,h1,-y_fifth_order, 'FaceColor', 'k');

legend([NonLinear Quadratic Linear PartQuad_h1xh1 PartQuad_h1xulv Cubic fifth_Order
    ],...
    {['NonLinear'  newline 'Model'],...
    ['Quadratic' newline 'Model'],...
    ['Linear' newline 'Model'],...
    ['Partially' newline 'Quadratic' newline '(h1xh1) Model'],...
    ['Partially' newline 'Quadratic' newline '(h1xuLV) Model'],...
    ['Cubic' newline 'Model'], ...
    ['5th Order' newline 'Model']})

%%  Linear VS Other Models Contour-Plot

% This section plots the Contour-Plot of the comparison between our Linear
% Model, and all of the other models

close all

%Here, we can choose which section we want to plot.
h1_low = 1;         %Minimum is 1
h1_hi = 201;          %Maximum is 201
ulv_low = 1;        %Minimum is 1
ulv_hi = 201;         %Maximum is 201
colorbar_range_min = 0;
colorbar_range_max = 0.015;
if scenario == 1

    H1_min = 20*2;
    H1_max = 30*2;
    ULV_min = 41*2;
    ULV_max = 61*2;
elseif scenario == 2
    H1_min = 70*2;
    H1_max = 80*2;
    ULV_min = 26*2;
    ULV_max = 46*2;
end
%Makes a custum colormap
map = [ 0 0.7 0;                 %Green
        0.359 1 0.402;           %Light green
        0.535 0.808 0.9375;      %Light blue
        0 0 1;                   %Blue
        1 0 0];                   %Red

% Calculates the error between the NonLinear system and our model
NL_Quad_error = abs((ydot - y_quadratic));                    %Error between
    NonLinear model and Quadratic model
NL_Lin_error = abs((ydot - y_linear));                        %Error between
    NonLinear model and Linear model
NL_ParQuad_h1xh1_error = abs((ydot - y_parQuad_h1xh1));       %Error between
    NonLinear model and Partially Quadratic (h1xh1) model
NL_ParQuad_h1xulv_error = abs((ydot - y_parQuad_h1xulv));    %Error between
    NonLinear model and Partially Quadratic (h1xulv) model
NL_Cubic_error = abs(ydot-y_cubic);                           %Error between
    NonLinear model and Cubic model
NL_fifth_error = abs(ydot-y_fifth_order);                     %Error between
    NonLinear model and 5th order model

% Calculates the error between the Linear model and our other models
Lin_Quad_diff= NL_Lin_error - NL_Quad_error;                          %Error between
    Linear model and Quadratic model
Lin_ParQuad_h1xh1_diff = NL_Lin_error - NL_ParQuad_h1xh1_error;      %Error between
    Linear model and Partially Quadratic (h1xh1) model
Lin_ParQuad_h1xulv_diff = NL_Lin_error - NL_ParQuad_h1xulv_error;   %Error between
    Linear model and Partially Quadratic (h1xulv) model
Lin_Cubic_diff = NL_Lin_error - NL_Cubic_error;                       %Error between
    Linear model and Cubic model
```

```
Lin_fifth_diff = NL_Lin_error - NL_fifth_error;                         %Error between
    Linear model and 5th order model
Cubic_fifth_diff = NL_Cubic_error - NL_fifth_error;                     %Error between
    Cubic model and 5th order model
Cubic_quadratic_diff = NL_Quad_error - NL_Cubic_error;
figure
%Plots the contour of the difference between Linear and Quadratic model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    Lin_Quad_diff(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('Linear VS Quadratic @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the difference between Linear and Partially Quadratic (h1xh1)
     model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    Lin_ParQuad_h1xh1_diff(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('Linear VS Partially Quadratic (h1xh1) @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the difference between Linear and Partially Quadratic (h1xulv
    ) model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    Lin_ParQuad_h1xulv_diff(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('Linear VS Partially Quadratic (h1xuLV) @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the difference between Linear and Cubic model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    Lin_Cubic_diff(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
```

```
caxis([colorbar_range_min, colorbar_range_max])
title('Linear VS Cubic @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the difference between Linear and 5th order model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    Lin_fifth_diff(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('Linear VS 5th Order @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the difference between Cubic and 5th order model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    Cubic_fifth_diff(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('Cubic VS 5th Order @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

%% Non-Linear VS Other Models Contour-Plot

% This section plots the Contour-Plot of the comparison between the NonLinear
% Model, and all of the other models

close all
figure
%Plots the contour of the error for the Quadratic model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    NL_Quad_error(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('NonLinear VS Quadratic @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

```

```
figure
%Plots the contour of the error for the Linear model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    NL_Lin_error(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('NonLinear VS Linear @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the error for the Partially Quadratic (h1xh1) model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    NL_ParQuad_h1xh1_error(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('NonLinear VS Partially Quadratic (h1xh1) @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the error for the Partially Quadratic (h1xulv) model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    NL_ParQuad_h1xulv_error(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('NonLinear VS Partially Quadratic (h1xuLV) @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the error for the Cubic model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    NL_Cubic_error(h1_low:h1_hi, ulv_low:ulv_hi), 40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('NonLinear VS Cubic @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
```

```
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

figure
%Plots the contour of the error for the 5th order model
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    NL_fifth_error(h1_low:h1_hi, ulv_low:ulv_hi),40)
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
c = colorbar;
c.Label.String = 'Delta';
c.Label.FontSize = 16;
caxis([colorbar_range_min, colorbar_range_max])
title('NonLinear VS 5th Order @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
colormap(jet(20))
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

%% Linear VS Quadratic Colormap

close all

%We want to make a matrix that tells us where the models differ from each
%other. Call this h
h = ones(201);

%Sorting algorithm
%Here we define a arbitrary value that tells us if the models differ. In
%this example, we chose 0.0005
for i = 1:201
    for j = 1:201
        if Lin_Quad_diff(i,j) == 0
            h(i,j) = 5;                                              %Quad
    == Lin
        elseif Lin_Quad_diff(i,j) > 0 && Lin_Quad_diff(i,j) < 0.0005         %
    Quad>
            h(i,j) = 1;
        elseif Lin_Quad_diff(i,j) > 0.0005                           %   Quad>>
                h(i,j) = 0;
        elseif Lin_Quad_diff(i,j) < 0 && Lin_Quad_diff(i,j) > -0.0005   %   Lin>
            h(i,j) = 2;
        elseif Lin_Quad_diff(i,j) < -0.0005                          %   Lin>>
                h(i,j) = 3;

        end
    end
end

%Now we have the marix h, which tells us where the models differ, but it
%does not tell us if the models are a good approximation to the NonLinear
%system.
%Sorting algorithm which gives us a 'red flag' if both models are a bad
%approximation at a certain point. Here, if both differ more than 0.0005
%from the NonLinear model, they are defined as a bad approximation.
for i = 1:201
    for j = 1:201
        if abs(NL_Lin_error(i,j)) > 0.005 && abs(NL_Quad_error(i,j)) > 0.005
            h(i,j) = 4;
        end
    end
end


C = h(H1_min:H1_max, ULV_min:ULV_max);
[ii, jj] = find(C == 5)


figure
%Plots the contour of the data-matrix h
```

```
contourf(h, 100, 'EdgeColor', 'None')
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
hold on
plot([ULV_min ULV_min], [H1_min H1_max],'k','LineWidth',2)
plot([ULV_max ULV_max], [H1_min H1_max],'k','LineWidth',2)
plot([ULV_min ULV_max], [H1_min H1_min],'k','LineWidth',2)
plot([ULV_min ULV_max], [H1_max H1_max],'k','LineWidth',2)


colorbar('Ticks', [0, 1, 2, 3, 4], 'TickLabels', {'Quad>>', 'Quad>','Lin>', 'Lin>>'
    , 'N/A'})
colormap(map)
caxis([0, 4])
title('Linear VS Quadratic @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

%% Linear VS Partially Quadratic (h1xh1) Colormap

close all

%Data-matrix
h = ones(201);

%Sorting algorithm
for i = 1:201
    for j = 1:201
        if Lin_ParQuad_h1xh1_diff(i,j) > 0 && Lin_ParQuad_h1xh1_diff(i,j) < 0.0005
     %ParQuad_h1xh1>
                h(i,j) = 1;
        elseif Lin_ParQuad_h1xh1_diff(i,j) > 0.0005 %ParQuad_h1xh1>>
                    h(i,j) = 0;
        elseif Lin_ParQuad_h1xh1_diff(i,j) < 0 && Lin_ParQuad_h1xh1_diff(i,j) >
    -0.0005 %Lin>
                h(i,j) = 2;
        elseif Lin_ParQuad_h1xh1_diff(i,j) < -0.0005    %Lin>>
                    h(i,j) = 3;

        end
    end
end


%'red flag' algorithm
for i = 1:201
    for j = 1:201
        if abs(NL_Lin_error(i,j)) > 0.005 && abs(NL_ParQuad_h1xh1_error(i,j)) >
    0.005
                h(i,j) = 4;
        end
    end
end

figure
 %Plots the contour of the data-matrix h
contourf(h, 100, 'EdgeColor', 'None')
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
colorbar('Ticks', [0, 1, 2, 3, 4], 'TickLabels', {'ParQuad>>', 'ParQuad>','Lin>', '
    Lin>>', 'N/A'})
colormap(map)
caxis([0, 4])
title('Linear VS Partially Quadratic (h1xh1) @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
```

```
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

%% Linear VS Partially Quadratic (h1xuLV) Colormap

close all

%Data-matrix
h = ones(201);

%Sorting algorithm
for i = 1:201
    for j = 1:201
        if Lin_ParQuad_h1xulv_diff(i,j) > 0 && Lin_ParQuad_h1xulv_diff(i,j) <
    0.0005   %ParQuad_h1xulv>
            h(i,j) = 1;
        elseif Lin_ParQuad_h1xulv_diff(i,j) > 0.0005    %ParQuad_h1xulv>>
                h(i,j) = 0;
        elseif Lin_ParQuad_h1xulv_diff(i,j) < 0 && Lin_ParQuad_h1xulv_diff(i,j) >
    -0.0005   %Lin>
            h(i,j) = 2;
        elseif Lin_ParQuad_h1xulv_diff(i,j) < -0.0005   %Lin>>
                h(i,j) = 3;

        end
    end
end

%'red flag' algorithm
for i = 1:201
    for j = 1:201
        if abs(NL_Lin_error(i,j)) > 0.005 && abs(NL_ParQuad_h1xulv_error(i,j)) >
    0.005
            h(i,j) = 4;
        end
    end
end

figure
contourf(h, 100, 'EdgeColor', 'None')    %Plots the contour of the data-matrix h
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
colorbar('Ticks', [0, 1, 2, 3, 4], 'TickLabels', {'ParQuad>>', 'ParQuad>','Lin>', '
    Lin>>', 'N/A'})
colormap(map)
caxis([0, 4])
title('Linear VS Partially Quadratic (h1xulv) @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

%% Linear VS Cubic Colormap

close all

%Data-matrix
h = ones(201);

%Sorting algorithm
for i = 1:201
    for j = 1:201
        if Lin_Cubic_diff(i,j) > 0 && Lin_Cubic_diff(i,j) < 0.0005        %Cubic>
            h(i,j) = 1;
        elseif Lin_Cubic_diff(i,j) > 0.0005                               %Cubic>>
                h(i,j) = 0;
        elseif Lin_Cubic_diff(i,j) < 0 && Lin_Cubic_diff(i,j) > -0.0005   %Lin>
            h(i,j) = 2;
        elseif Lin_Cubic_diff(i,j) < -0.0005                              %Lin>>
                h(i,j) = 3;

        end
```

```
    end
end

%'red flag' algorithm
for i = 1:201
    for j = 1:201
        if abs(NL_Lin_error(i,j)) > 0.005 && abs(NL_Cubic_error(i,j)) > 0.005
            h(i,j) = 4;
        end
    end
end

figure
%Plots the contour of the data-matrix h
contourf(h, 100, 'EdgeColor', 'None')
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
colorbar('Ticks', [0, 1, 2, 3, 4], 'TickLabels', {'Cubic>>', 'Cubic>','Lin>', 'Lin>>', 'N/A'})
colormap(map)
caxis([0, 4])
title('Linear VS Cubic @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

%%  Linear VS Fifth order Colormap

close all

%Data-matrix
h = ones(201);

%Sorting algorithm
for i = 1:201
    for j = 1:201
        if Lin_fifth_diff(i,j) > 0 && Lin_fifth_diff(i,j) < 0.0005        %Fifth>
            h(i,j) = 1;
        elseif Lin_fifth_diff(i,j) > 0.0005                              %Fifth>>
                h(i,j) = 0;
        elseif Lin_fifth_diff(i,j) < 0 && Lin_fifth_diff(i,j) > -0.0005   %Linear>
            h(i,j) = 2;
        elseif Lin_fifth_diff(i,j) < -0.0005                             %Linear>>
                h(i,j) = 3;

        end
    end
end

%'red flag' algorithm
for i = 1:201
    for j = 1:201
        if abs(NL_Lin_error(i,j)) > 0.005 && abs(NL_fifth_error(i,j)) > 0.005
            h(i,j) = 4;
        end
    end
end

figure
%Plots the contour of the data-matrix h
contourf(h, 100, 'EdgeColor', 'None')
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
colorbar('Ticks', [0, 1, 2, 3, 4], 'TickLabels', {'5th order>>', '5th order>','Linear>', 'Linear>>', 'N/A'})
colormap(map)
caxis([0, 4])
title('Linear VS 5th Order @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
```

```
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

%%  Cubic VS Fifth order Colormap

%close all

%Data-matrix
h = ones(201);

%Sorting algorithm
for i = 1:201
    for j = 1:201
        if Cubic_fifth_diff(i,j) == 0
            h(i,j) = 5;                                                      %Cubic == Fifth
        elseif Cubic_fifth_diff(i,j) > 0 && Cubic_fifth_diff(i,j) < 0.0005        %Fifth>
            h(i,j) = 1;
        elseif Cubic_fifth_diff(i,j) > 0.0005                                %Fifth>>
                h(i,j) = 0;
        elseif Cubic_fifth_diff(i,j) < 0 && Cubic_fifth_diff(i,j) > -0.0005   %Cubic>
            h(i,j) = 2;
        elseif Cubic_fifth_diff(i,j) < -0.0005                               %Cubic>>
                h(i,j) = 3;

        end
    end
end

%'red flag' algorithm
for i = 1:201
    for j = 1:201
        if abs(NL_Cubic_error(i,j)) > 0.005 && abs(NL_fifth_error(i,j)) > 0.005
            h(i,j) = 4;
        end
    end
end

C = h(H1_min:H1_max, ULV_min:ULV_max);
[ii, jj] = find(C == 2)

figure
%Plots the contour of the data-matrix h
contourf(h, 100, 'EdgeColor', 'None')
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
hold on
plot([ULV_min ULV_min], [H1_min H1_max],'k','LineWidth',2)
plot([ULV_max ULV_max], [H1_min H1_max],'k','LineWidth',2)
plot([ULV_min ULV_max], [H1_min H1_min],'k','LineWidth',2)
plot([ULV_min ULV_max], [H1_max H1_max],'k','LineWidth',2)
colorbar('Ticks', [0, 1, 2, 3, 4], 'TickLabels', {'5th order>>', '5th order>','Cubic>', 'Cubic>>', 'N/A'})
colormap(map)
caxis([0, 4])
title('5th VS Cubic @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)

%% Cubic VS Quadratic Colormap
close all

%Data-matrix
h = ones(201);

```

```
%Sorting algorithm
for i = 1:201
    for j = 1:201
        if Cubic_quadratic_diff(i,j) == 0
            h(i,j) = 5;                                                     %Quad == cubic
        elseif Cubic_quadratic_diff(i,j) > 0 && Cubic_quadratic_diff(i,j) < 0.0005 %Cubic>
            h(i,j) = 1;
        elseif Cubic_quadratic_diff(i,j) > 0.0005                          %Cubic>>
                h(i,j) = 0;
        elseif Cubic_quadratic_diff(i,j) < 0 && Cubic_quadratic_diff(i,j) > -0.0005 %Quad>
            h(i,j) = 2;
        elseif Cubic_quadratic_diff(i,j) < -0.0005                         %Quad>>
                h(i,j) = 3;

        end
    end
end

%'red flag' algorithm
for i = 1:201
    for j = 1:201
        if abs(NL_Cubic_error(i,j)) > 0.005 && abs(NL_Quad_error(i,j)) > 0.005
            h(i,j) = 4;
        end
    end
end

C = h(H1_min:H1_max, ULV_min:ULV_max);
[ii, jj] = find(C == 2)

figure
%Plots the contour of the data-matrix h
contourf(ulv_low:ulv_hi, h1_low:h1_hi,...
    h(h1_low:h1_hi, ulv_low:ulv_hi), 100, 'EdgeColor', 'None')
hold on
plot(200*ulv_arb,200*h1_arb,'kx','MarkerSize',30)
hold on
plot([ULV_min ULV_min], [H1_min H1_max],'k','LineWidth',2)
plot([ULV_max ULV_max], [H1_min H1_max],'k','LineWidth',2)
plot([ULV_min ULV_max], [H1_min H1_min],'k','LineWidth',2)
plot([ULV_min ULV_max], [H1_max H1_max],'k','LineWidth',2)
colorbar('Ticks', [0, 1, 2, 3, 4], 'TickLabels', {'Cubic>>', 'Cubic>','Quadratic>', 'Quadratic>>', 'N/A'})
colormap(map)
caxis([0, 4])
title('Quadratic VS Cubic @0.25m Contour-Plot')
xlabel('$u_{LV}$', 'Interpreter','latex','fontsize',16)
ylabel('$h_1$', 'Interpreter','latex','fontsize',16)
xt = get(gca, 'XTick');
set(gca, 'XTick', xt, 'XTickLabel', xt/200)
yt = get(gca, 'YTick');
set(gca, 'YTick', yt, 'YTickLabel', yt/200)
%% Carleman Linearization
close all

n = 1;
Initial_level = 0;

[A_matrix_1, B_matrix_1, C_matrix_1, D_matrix_1, Initial_vector_1] = Carleman_Linearized_func(...
                          par_der_h1_verdi,...
                          par_der_uLV001_verdi,...
                          par_der_uPA001_verdi,...
                          par_der_h1_uLV001_verdi,...
                          par_der_h1_h1_verdi,...
                          par_der_h1_h1_h1_verdi,...
                          par_der_h1_h1_uLV001_verdi,...
                          Initial_level,...
```

```
                         n);
n = 2;
[A_matrix_2, B_matrix_2, C_matrix_2, D_matrix_2, Initial_vector_2] =
    Carleman_Linearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);
n = 3;
[A_matrix_3, B_matrix_3, C_matrix_3, D_matrix_3, Initial_vector_3] =
    Carleman_Linearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);
n = 4;
[A_matrix_4, B_matrix_4, C_matrix_4, D_matrix_4, Initial_vector_4] =
    Carleman_Linearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);

n = 5;
[A_matrix_5, B_matrix_5, C_matrix_5, D_matrix_5, Initial_vector_5] =
    Carleman_Linearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);

n = 10;
[A_matrix_10, B_matrix_10, C_matrix_10, D_matrix_10, Initial_vector_10] =
    Carleman_Linearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);

n = 6;
[A_matrix_6, B_matrix_6, C_matrix_6, D_matrix_6, Initial_vector_6, E_matrix_6] =...
                        Carleman_NonLinearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
```

```
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);


                     %Non linear Carleman approximations
                     n = 1;
[A_matrix_NL_1, B_matrix_NL_1, C_matrix_NL_1, D_matrix_NL_1, Initial_vector_NL_1, E_matrix_NL_1] =...
                        Carleman_NonLinearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);

E_matrix_matrix_1 = [];

for i = 1:n
    E_matrix_matrix_1(:,i) = E_matrix_NL_1{i};
end

                                       n = 2;
[A_matrix_NL_2, B_matrix_NL_2, C_matrix_NL_2, D_matrix_NL_2, Initial_vector_NL_2, E_matrix_NL_2] =...
                        Carleman_NonLinearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);

                     E_matrix_matrix_2 = [];

for i = 1:n
    E_matrix_matrix_2(:,i) = E_matrix_NL_2{i};
end

                      n = 3;
[A_matrix_NL_3, B_matrix_NL_3, C_matrix_NL_3, D_matrix_NL_3, Initial_vector_NL_3, E_matrix_NL_3] =...
                        Carleman_NonLinearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
                         par_der_h1_uLV001_verdi,...
                         par_der_h1_h1_verdi,...
                         par_der_h1_h1_h1_verdi,...
                         par_der_h1_h1_uLV001_verdi,...
                         Initial_level,...
                         n);
 E_matrix_matrix_3 = [];

for i = 1:n
    E_matrix_matrix_3(:,i) = E_matrix_NL_3{i};
end


                      n = 4;
[A_matrix_NL_4, B_matrix_NL_4, C_matrix_NL_4, D_matrix_NL_4, Initial_vector_NL_4, E_matrix_NL_4] =...
                        Carleman_NonLinearized_func(...
                         par_der_h1_verdi,...
                         par_der_uLV001_verdi,...
                         par_der_uPA001_verdi,...
```

```
                          par_der_h1_uLV001_verdi,...
                          par_der_h1_h1_verdi,...
                          par_der_h1_h1_h1_verdi,...
                          par_der_h1_h1_uLV001_verdi,...
                          Initial_level,...
                          n);

E_matrix_matrix_4 = [];

for i = 1:n
    E_matrix_matrix_4(:,i) = E_matrix_NL_4{i};
end

                     n = 5;
[A_matrix_NL_5, B_matrix_NL_5, C_matrix_NL_5, D_matrix_NL_5, Initial_vector_NL_5,
    E_matrix_NL_5] =...
                         Carleman_NonLinearized_func(...
                          par_der_h1_verdi,...
                          par_der_uLV001_verdi,...
                          par_der_uPA001_verdi,...
                          par_der_h1_uLV001_verdi,...
                          par_der_h1_h1_verdi,...
                          par_der_h1_h1_h1_verdi,...
                          par_der_h1_h1_uLV001_verdi,...
                          Initial_level,...
                          n);
   E_matrix_matrix_5 = [];

for i = 1:n
    E_matrix_matrix_5(:,i) = E_matrix_NL_5{i};
end

                      n = 10;
[A_matrix_NL_10, B_matrix_NL_10, C_matrix_NL_10, D_matrix_NL_10,
    Initial_vector_NL_10, E_matrix_NL_10] =...
                         Carleman_NonLinearized_func(...
                          par_der_h1_verdi,...
                          par_der_uLV001_verdi,...
                          par_der_uPA001_verdi,...
                          par_der_h1_uLV001_verdi,...
                          par_der_h1_h1_verdi,...
                          par_der_h1_h1_h1_verdi,...
                          par_der_h1_h1_uLV001_verdi,...
                          Initial_level,...
                          n);
 E_matrix_matrix_10 = [];

for i = 1:n
    E_matrix_matrix_10(:,i) = E_matrix_NL_10{i};
end


[NUM,DEN] = ss2tf(A_matrix_1,B_matrix_1,C_matrix_1,D_matrix_1,1);
system = tf(NUM,DEN);
stability = isstable(system)
figure
pzmap(system)
figure
step(system)

%%  Quadratic controller NY test
n = 4;
Initial_level = 0.05;

Alpha = 0.0; %Scenario 1: 0.096   Scenario 2: Grense 0.036
%Scenario 1():0.221    Scenario 2(): Grense 0.077

%Senario 1: Ivar = 0.094

gamma = 0.1; %7e-1

%ulv_arb = 0.5159, dulv_max = 0.48
%upa_arb = 0.65, dupa_max = 0.2
```

```
dupa_max = 0.2; % var 0.2
dulv_max = 0.45; %
a = -0.05%-0.05; %delta_h1 min
b = 0.05; %delta_h1 max
x1 = [a b];
Vertex = x1;

punkt_cell = cell(n,1);

if n >= 2
    for i = 2:n
        if mod(i,2) ~= 0
            punkt_cell{i} = x1.^i;
        end
        if mod(i,2) == 0
            punkt_cell{i} = [-max(a^i, b^i)/10,max(a^i, b^i)];
        end
        Vertex = combvec(Vertex, punkt_cell{i});
    end
else
    Vertex = Vertex';
end

%%%%%%%%%%%%%%%%%
%Vertex_shifted = Vertex;
[A_test,B_test] = vert2con(Vertex');
z = B_test; %

ak_T = A_test./z;
ak = ak_T';

%%%%%%%%%%
[A_matrix_1var, B_matrix_1var, C_matrix_1var, D_matrix_1var, Initial_vector_1var,
    E_matrix_1var] = Carleman_NonLinearized_func(...
                par_der_h1_verdi,...
                par_der_uLV001_verdi,...
                par_der_uPA001_verdi,...
                par_der_h1_uLV001_verdi,...
                par_der_h1_h1_verdi,...
                par_der_h1_h1_h1_verdi,...
                par_der_h1_h1_uLV001_verdi,...
                Initial_level,...
                n);
%dupa_max = 0.2, dulv_max = 0.48
B_matrix_1var(:,2) = 0;

for i = 1:n
    E_matrix_1var{i}(:,2) = 0;
end

[K_1var, P_1var] = Quadratic_controller(A_matrix_1var, B_matrix_1var, E_matrix_1var
    , Vertex, Alpha, gamma,dupa_max,dulv_max,ak);

strucK.A = A_matrix_1var;
strucK.B = B_matrix_1var;
strucK.K = K_1var;
strucK.E = E_matrix_1var;

[T1,X] = ode45(@(t,x)Quadratic_system(t, x, strucK),0:0.001:160,Initial_vector_1var
    ');

close all
figure(1)
hold on
for i = 1:n
    plot(T1,X(:,i))
end
xlabel ('Time [s]')
ylabel ('Delta height [m]')
grid on
hold off

figure(2)
```

```
plot(T1,X(:,1))
xlabel ('Time [s]')
ylabel ('Delta height [m]')
grid on

E_matrix_matrix_1var = [];
for i = 1:n
    [r,k] = size(E_matrix_matrix_1var);
    E_matrix_matrix_1var(:,k+1:k+2) = E_matrix_1var{i};
end
[r,k] = size(X);
U = [];
for i = 1:r
    U(i,:) = (K_1var*X(i,:)')';
end

figure(3)
ulv = plot(T1,U(:,1))
hold on
%upa = plot(T1,U(:,2))
xlabel ('Time [s]')
ylabel ('Delta upa/ulv')
grid on
legend([ulv],'ulv')

%% Quadratic controller 2Var
n = 4;
Initial_level = 0.05;

Alpha = 0.096; %Scenario 1: 0.096   Scenario 2: 0.036
%Scenario 1():0.221    Scenario 2(): Grense 0.077
gamma = 0.1; %7e-1

%ulv_arb = 0.5159, dulv_max = 0.48
%upa_arb = 0.65, dupa_max = 0.2
dupa_max = 0.2; % var 0.2
dulv_max = 0.45; %
a = -0.05%-0.05; %delta_h1 min
b = 0.05; %delta_h1 max
x1 = [a b];
Vertex = x1;

punkt_cell = cell(n,1);

if n >= 2
    for i = 2:n
        if mod(i,2) ~= 0
            punkt_cell{i} = x1.^i;
        end
        if mod(i,2) == 0
            punkt_cell{i} = [-max(a^i, b^i)/10,max(a^i, b^i)];
        end
        Vertex = combvec(Vertex, punkt_cell{i});
    end
else
    Vertex = Vertex';
end

%Vertex = [-0.01 0.05  0.05;
%           -0.01 -0.01  (0.05^2)]; %Dette er en test!! Fjern etterp .

%%%%%%%%%%%%%%%%%
%Vertex_shifted = Vertex;
[A_test,B_test] = vert2con(Vertex');
z = B_test; %

ak_T = A_test./z;
ak = ak_T';
%ak = [];
%[r,k] = size(ak)

%   for i = 1:r
%       for j = 1:k
```

```
%           tall = ak(i,j)
%           if tall < 1 && tall > -1
%              ak(i,j) = 0
%           end
%     end
%   end
%
%  ak = [20,-20,0,0,0,0,0,0;
%      0,0,0,0,0,0,400,-1000;
%      0,0,0,0,1000,-1000,0,0;
%      0,0,-1000,1000,0,0,0,0];

%%%%%%%%%
[A_matrix_2var, B_matrix_2var, C_matrix_2var, D_matrix_2var, Initial_vector_2var,
    E_matrix_2var] = Carleman_NonLinearized_func_2Var(...
                par_der_h1_verdi,...
                par_der_uLV001_verdi,...
                par_der_uPA001_verdi,...
                par_der_h1_uLV001_verdi,...
                par_der_h1_h1_verdi,...
                par_der_h1_h1_h1_verdi,...
                par_der_h1_h1_uLV001_verdi,...
                Initial_level,...
                n);
%dupa_max = 0.2, dulv_max = 0.48

[K_2var, P_2var] = Quadratic_controller(A_matrix_2var, B_matrix_2var, E_matrix_2var
    , Vertex, Alpha, gamma,dupa_max,dulv_max,ak);

strucK.A = A_matrix_2var;
strucK.B = B_matrix_2var;
strucK.K = K_2var;
strucK.E = E_matrix_2var;

[T1,X] = ode45(@(t,x)Quadratic_system(t, x, strucK),0:0.001:160,Initial_vector_2var
    ');

close all
figure(1)
hold on
for i = 1:n
    plot(T1,X(:,i))
end
xlabel ('Time [s]')
ylabel ('Delta height [m]')
grid on
hold off

figure(2)
z1 = plot(T1,X(:,1))
xlabel ('Time [s]')
ylabel ('Delta height [m]')
grid on
legend([z1],'z1')

E_matrix_matrix_2var = [];
for i = 1:n
    [r,k] = size(E_matrix_matrix_2var);
    E_matrix_matrix_2var(:,k+1:k+2) = E_matrix_2var{i};
end
[r,k] = size(X);
U = [];
for i = 1:r
    U(i,:) = (K_2var*X(i,:)')';
end

figure(3)
ulv = plot(T1,U(:,1))
hold on
upa = plot(T1,U(:,2))
xlabel ('Time [s]')
%ylabel ('Delta upa/ulv')
grid on
```

```
legend([ulv, upa],'ulv','upa')
%%  Error Values for Taylor
clc
%The Following code displays the Error Value for every model, wrt the
%Nonlinear model.
sim('Taylor_Simulations', 250);

disp('IAE Error: ')
disp(['Quadratic:       ' num2str(IAE_Quadratic.Data(end))])
disp(['Linear:          ' num2str(IAE_Linear.Data(end))])
disp(['ParQuad_h1xh1:   ' num2str(IAE_ParQuad_h1xh1.Data(end))])
disp(['ParQuad_h1xulv:  ' num2str(IAE_ParQuad_h1xulv.Data(end))])
disp(['Cubic:           ' num2str(IAE_Cubic.Data(end))])
disp(['Fifth order:        ' num2str(IAE_Fifth.Data(end))])

fprintf('\n')

disp('ISE Error: ')
disp(['Quadratic:       ' num2str(ISE_Quadratic.Data(end))])
disp(['Linear:          ' num2str(ISE_Linear.Data(end))])
disp(['ParQuad_h1xh1:   ' num2str(ISE_ParQuad_h1xh1.Data(end))])
disp(['ParQuad_h1xulv:  ' num2str(ISE_ParQuad_h1xulv.Data(end))])
disp(['Cubic:           ' num2str(ISE_Cubic.Data(end))])
disp(['Fifth:        ' num2str(ISE_Fifth.Data(end))])

fprintf('\n')

disp('ITAE Error: ')
disp(['Quadratic:       ' num2str(ITAE_Quadratic.Data(end))])
disp(['Linear:          ' num2str(ITAE_Linear.Data(end))])
disp(['ParQuad_h1xh1:   ' num2str(ITAE_ParQuad_h1xh1.Data(end))])
disp(['ParQuad_h1xulv:  ' num2str(ITAE_ParQuad_h1xulv.Data(end))])
disp(['Cubic:           ' num2str(ITAE_Cubic.Data(end))])
disp(['Fifth order:        ' num2str(ITAE_Fifth.Data(end))])

fprintf('\n')

disp('ITSE Error: ')
disp(['Quadratic:       ' num2str(ITSE_Quadratic.Data(end))])
disp(['Linear:          ' num2str(ITSE_Linear.Data(end))])
disp(['ParQuad_h1xh1:   ' num2str(ITSE_ParQuad_h1xh1.Data(end))])
disp(['ParQuad_h1xulv:  ' num2str(ITSE_ParQuad_h1xulv.Data(end))])
disp(['Cubic:           ' num2str(ITSE_Cubic.Data(end))])
disp(['Fifth order:        ' num2str(ITSE_Fifth.Data(end))])

fprintf('\n')

disp('ISTE Error: ')
disp(['Quadratic:       ' num2str(ISTE_Quadratic.Data(end))])
disp(['Linear:          ' num2str(ISTE_Linear.Data(end))])
disp(['ParQuad_h1xh1:   ' num2str(ISTE_ParQuad_h1xh1.Data(end))])
disp(['ParQuad_h1xulv:  ' num2str(ISTE_ParQuad_h1xulv.Data(end))])
disp(['Cubic:           ' num2str(ISTE_Cubic.Data(end))])
disp(['Fifth order:        ' num2str(ISTE_Fifth.Data(end))])
%%  Error Values for Carleman

clc
%The Following code displays the Error Value for every model, wrt the
%Nonlinear model.
sim('Taylor_Simulations', 250);

disp('IAE Error: ')
disp(['Carleman n = 1:       ' num2str(IAE_Carleman_n_1.Data(end))])
disp(['Carleman n = 2:          ' num2str(IAE_Carleman_n_2.Data(end))])
disp(['Carleman n = 3:   ' num2str(IAE_Carleman_n_3.Data(end))])
disp(['Carleman n = 4:  ' num2str(IAE_Carleman_n_4.Data(end))])
disp(['Carleman n = 5:           ' num2str(IAE_Carleman_n_5.Data(end))])
disp(['Carleman n = 10:     ' num2str(IAE_Carleman_n_10.Data(end))])

fprintf('\n')

disp('ISE Error: ')
disp(['Carleman n = 1:       ' num2str(ISE_Carleman_n_1.Data(end))])
```

```
disp(['Carleman n = 2:          ' num2str(ISE_Carleman_n_2.Data(end))])
disp(['Carleman n = 3:   ' num2str(ISE_Carleman_n_3.Data(end))])
disp(['Carleman n = 4:  ' num2str(ISE_Carleman_n_4.Data(end))])
disp(['Carleman n = 5:           ' num2str(ISE_Carleman_n_5.Data(end))])
disp(['Carleman n = 10:          ' num2str(ISE_Carleman_n_10.Data(end))])

fprintf('\n')

disp('ITAE Error: ')
disp(['Carleman n = 1:       ' num2str(ITAE_Carleman_n_1.Data(end))])
disp(['Carleman n = 2:          ' num2str(ITAE_Carleman_n_2.Data(end))])
disp(['Carleman n = 3:   ' num2str(ITAE_Carleman_n_3.Data(end))])
disp(['Carleman n = 4:  ' num2str(ITAE_Carleman_n_4.Data(end))])
disp(['Carleman n = 5:           ' num2str(ITAE_Carleman_n_5.Data(end))])
disp(['Carleman n = 10:     ' num2str(ITAE_Carleman_n_10.Data(end))])

fprintf('\n')

disp('ITSE Error: ')
disp(['Carleman n = 1:       ' num2str(ITSE_Carleman_n_1.Data(end))])
disp(['Carleman n = 2:          ' num2str(ITSE_Carleman_n_2.Data(end))])
disp(['Carleman n = 3:   ' num2str(ITSE_Carleman_n_3.Data(end))])
disp(['Carleman n = 4:  ' num2str(ITSE_Carleman_n_4.Data(end))])
disp(['Carleman n = 5:           ' num2str(ITSE_Carleman_n_5.Data(end))])
disp(['Carleman n = 10:     ' num2str(ITSE_Carleman_n_10.Data(end))])

fprintf('\n')

disp('ISTE Error: ')
disp(['Carleman n = 1:       ' num2str(ISTE_Carleman_n_1.Data(end))])
disp(['Carleman n = 2:          ' num2str(ISTE_Carleman_n_2.Data(end))])
disp(['Carleman n = 3:   ' num2str(ISTE_Carleman_n_3.Data(end))])
disp(['Carleman n = 4:  ' num2str(ISTE_Carleman_n_4.Data(end))])
disp(['Carleman n = 5:           ' num2str(ISTE_Carleman_n_5.Data(end))])
disp(['Carleman n = 10:     ' num2str(ISTE_Carleman_n_10.Data(end))])
```

F.2 *Carleman_lin_ss.m*

```
% 19.Februar 2021
% John H vard Aarv g

%Carleman matrix

%%
close all

syms A1 f3 Kv_LV001 f1 rho g h1 h_LV001

dynamic_model_tank1 = (1/A1)*(f3 - ((Kv_LV001*f1)/3600)*sqrt((rho*g*(h1+h_LV001))/100000)); %The dynamic model for tank 1

% Calculating the partial derivatives (symbolic)
par_der_h1 = diff(dynamic_model_tank1, h1);     %   df/dh1
par_der_f1 = diff(dynamic_model_tank1, f1);     %   df/df1
par_der_f3 = diff(dynamic_model_tank1, f3);     %   df/df3
par_der_h1_h1 = diff(par_der_h1, h1);           %   d^2f/dh1^2
par_der_f1_f1 = diff(par_der_f1, f1);           %   d^2f/df1^2
par_der_f3_f3 = diff(par_der_f3, f3);           %   d^2f/df3^2
par_der_h1_f1 = diff(par_der_h1, f1);           %   d^2f/dh1df1
par_der_h1_f3 = diff(par_der_h1, f3);           %   d^2f/dh1df3
par_der_f1_f3 = diff(par_der_f1, f3);           %   d^2f/df1df3

par_der_h1_h1_h1 = diff(par_der_h1_h1, h1);     %   d^3f/dh1^3
par_der_h1_h1_f1 = diff(par_der_h1_h1, f1);     %   d^3f/dh1^2df1

par_der_h1_h1_h1_h1 = diff(par_der_h1_h1_h1,h1);%   d^4f/dh1^4
par_der_h1_h1_h1_f1 = diff(par_der_h1_h1_h1,f1);%    d^4f/dh1^3df1

par_der_h1_h1_h1_h1_h1 = diff(par_der_h1_h1_h1_h1,h1);% d^5f/dh1^5
par_der_h1_h1_h1_h1_f1 = diff(par_der_h1_h1_h1_h1,f1);% d^5f/dh1^4df1

% Setting the working point for h1 and f3
```

```
h1_arb = 0.25;
f3_arb = 0.0001783;

% Inserting every known variable in equation dynamic_model_tank1
a = subs(dynamic_model_tank1, [A1 Kv_LV001 rho g h_LV001 h1 f3], [0.01 11.25 1000 9.81 0.05 h1_arb f3_arb]);
% Solving equation 'a' wrt f1. This gives f1_arb
f1_arb = double(solve(a,f1));

% Since f3_arb is chosen, we can find upa_arb from the pump-characteristics
upa_arb = 0.65;
% We have found f1_arb. Using this value in the valve-characteristics gives us ulv_arb
ulv_arb = 0.5159;



df1_duLV001    = 0.98;  %   Manually calculated in simulink. delta_f1 / delta_ulv
df3_dPA001   = 0.00052; %   Manually calculated in simulink. delta_f3 / delta_upa

% Calculating the partial derivatives used in the Taylor series expansion
% (Value at working point!)
par_der_h1_verdi = double((subs(par_der_h1,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

par_der_uLV001_verdi = double(subs(par_der_f1 * df1_duLV001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_uPA001_verdi = double(subs(par_der_f3 * df3_dPA001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_h1_h1_verdi = double(subs(par_der_h1_h1,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_uLV001_uLV001_verdi = double(subs(par_der_f1_f1 * df1_duLV001^2,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_uPA001_uPA001_verdi = double(subs(par_der_f3_f3 * df3_dPA001^2,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_h1_uLV001_verdi = double(subs(par_der_h1_f1 * df1_duLV001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_h1_uPa001_verdi = double(subs(par_der_h1_f3 * df3_dPA001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_uLV001_uPA001_verdi = double(subs(par_der_f1_f3 * df1_duLV001 * df3_dPA001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb]));

par_der_h1_h1_h1_verdi = double((subs(par_der_h1_h1_h1,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

par_der_h1_h1_uLV001_verdi = double((subs(par_der_h1_h1_f1 * df1_duLV001,...
    [A1 Kv_LV001 rho g h_LV001 h1 f1 f3],...
    [0.01 11.25 1000 9.81 0.05 h1_arb f1_arb f3_arb])));

% The following is a standard code that imports the characteristics of the
% pump and valve. This code also defines some constants. (From Reguleringsteknikk)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% data om vann og tyngdekraft
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
rho = 1000;                 % tetthet vann              [kg/m^3]
g   = 9.81;                 % tyngdens akselerasjon     [m/s^2]
c_p = 4200;                 % varmekapasitet  vann      [j/kg*K]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tank 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kv_LV001    = 11.25;         % ventilkonstant LV001  [m^3/h] ved 1 bar trykkfall
h_LV001     = 0.05;          % h yde til LV001 [m]
h1_max      = 1;             % maks h yde tank 1 [m]
h1_min      = 0.13;          % min h yde tank 1 [m]
A1          = 0.01;          % areal tank 1    [m^2]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tank 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kv_LV002    = 11.25;         % ventilkonstant LV002     [m3/h]
h_LV002     = 0.25;          % h yde fra bunn av tank 2 til LV002
h2_max      = 0.4;           % maks h yde tank 2    [m]
h2_min      = 0.02;          % min h yde tank 2    [m]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Last inn p drag og m linger
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load tankData_1   % det finnes ogs  et datasett som heter tankData_2
load tankData_2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pumpekarakteristikk
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_PA001 = [0.00  0.45  0.46  0.47  0.48  0.49  0.50  0.55...
           0.60  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00];
q_PA001 = [0.00  0.00  1.25  2.25  3.15  3.75  4.40  6.75...
           8.75 10.70 12.25 13.75 15.15 16.50 18.00 19.20 20.00];

q_PA001 = q_PA001/60000;  % liter/time -> m3/s

% figure
% plot(u_PA001, q_PA001,'*-')
% title('Pumpekarakteristikk')
% xlabel('P drag u_{PA001}(t) til pumpe PA001')
% ylabel('Volumstr m q_{PA001}(t) gjennom PA001 [m^3/s]')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ventilkarakteristikk
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_LV001 = 0:0.03:1;
f_LV001 = (exp(u_LV001.^1.2)-1)/(exp(1)-1);
u_LV002 = u_LV001;
f_LV002 = f_LV001;

% figure
% plot(u_LV001,f_LV001,'*-')
% title('Ventilkarakteristikk for LV001 og LV002')
% xlabel('Ventilp drag u_{LV001}(t)')
% ylabel('f(u_{LV001}(t))')

%End of the standard code from Reguleringsteknikk



% %%
% %carleman matrix
% %this section creates the A matrix in the eqn xdot = Ax + Bu, y = Cx + Du
% %This "linearization" removes all the mixed, nonlinear parts of the
% %carleman embedding.
%
% %simplified variable names
% a_c = par_der_h1_verdi;
% b_c = par_der_uLV001_verdi;
% c_c = par_der_uPA001_verdi;
% d_c = par_der_h1_uLV001_verdi;
```

```
% e_c = par_der_h1_h1_verdi/2;
% f_c = par_der_h1_h1_h1_verdi/6;
% g_c = par_der_h1_h1_uLV001_verdi/2;
%
% % Truncate terms beyond n, small mismatches up till n = 18. beyond that it
% % gets worse
% n = 4;
%
% A_matrix = zeros(n+2);
% B_matrix = zeros(n,1);
% C_matrix = zeros(1,n);
% D_matrix = (0);
% C_matrix(1) = 1;
% B_matrix(1) = b_c;
%
% A_matrix(1,1) = a_c + d_c*0.01; %x1
% A_matrix(1,2)= e_c + g_c*0.01;  %x2
% A_matrix(1,3) = f_c;            %x3
% Initial_level = 0.1;
%
% Initial_vector = zeros(n,1);
% for i=1:n
%     Initial_vector(i) = Initial_level^i;
% end
% %this pattern is clear after computing the first terms by hand.
% for i = 2:n
%    A_matrix(i,i-1) = i*b_c*0.01;
%    A_matrix(i,i) = i*a_c + i*d_c*0.01;
%    A_matrix(i,i+1)= i*e_c + i*g_c*0.01;
%    A_matrix(i,i+2) = i*f_c;
% end
%
% %Removes the truncated terms, that is for x = 10, all x 11, x12 etc are
% %removed.
% A_matrix(:,n+2) = [];
% A_matrix(:,n+1) = [];
% A_matrix(n+2,:) = [];
% A_matrix(n+1,:) = [];
%
% %Makes the statespace representation into a tf
% [NUM,DEN] = ss2tf(A_matrix,B_matrix,C_matrix,D_matrix,1);
% syste = tf(NUM,DEN)
% Gent = isstable(syste)
% pzmap(syste)


%%
%carleman matrix
%this section creates the A matrix in the eqn xdot = Ax + Bu, y = Cx + Du
%This "linearization" removes all the mixed, nonlinear parts of the
%carleman embedding.

%simplified variable names
a_c = par_der_h1_verdi;
b_c = par_der_uLV001_verdi;
c_c = par_der_uPA001_verdi;
d_c = par_der_h1_uLV001_verdi;
e_c = par_der_h1_h1_verdi/2;
f_c = par_der_h1_h1_h1_verdi/6;
g_c = par_der_h1_h1_uLV001_verdi/2;

% Truncate terms beyond n, small mismatches up till n = 18. beyond that it
% gets worse
n = 4;

A_matrix = zeros(n+2);
B_matrix = zeros(n,1);
C_matrix = zeros(1,n);
D_matrix = (0);
C_matrix(1) = 1;
B_matrix(1) = b_c;
```

```
Initial_level = 0.4;
%this pattern is clear after computing the first terms by hand.
for i = 1:n
   A_matrix(i,i) = i*a_c;
   A_matrix(i,i+1)= i*e_c;
   A_matrix(i,i+2) = i*f_c;
end

Initial_vector = zeros(n,1);
for i=1:n
    Initial_vector(i) = Initial_level^i;
end

%Removes the truncated terms, that is for x = 10, all x 11, x12 etc are
%removed.
A_matrix(:,n+2) = [];
A_matrix(:,n+1) = [];
A_matrix(n+2,:) = [];
A_matrix(n+1,:) = [];



%Makes the statespace representation into a tf
[NUM,DEN] = ss2tf(A_matrix,B_matrix,C_matrix,D_matrix,1);
syste = tf(NUM,DEN)
Gent = isstable(syste)
pzmap(syste)
```

F.3 *Carleman_Linearized_func.m*

```
function [A_matrix, B_matrix, C_matrix, D_matrix, Initial_vector] =...
    Carleman_Linearized_func(h1, ulv, upa, h1_ulv, h1_h1, h1_h1_h1, h1_h1_ulv,
    initC, n)

    %Simplifies the notation of the partial derivatives
    a_c = h1;
    b_c = ulv;
    c_c = upa;
    d_c = h1_ulv;
    e_c = h1_h1/2;
    f_c = h1_h1_h1/6;
    g_c = h1_h1_ulv/2;

    %Defines the amount of terms to include in the Carleman linearization
    n = n;

    %Defines the initial condition of the water level
    Initial_level = initC;

    %Finds the initial condition for every x_dot
    Initial_vector = zeros(n,1);
    for i=1:n
        Initial_vector(i) = Initial_level^i;
    end

    %
    A_matrix = zeros(n+2);
    B_matrix = zeros(n,1);
    C_matrix = zeros(1,n);
    D_matrix = (0);
    C_matrix(1) = 1;
    B_matrix(1) = b_c;

    %Constructs the A_matrix.
    %This pattern is clear after computing the first terms by hand.
    for i = 1:n
       A_matrix(i,i) = i*a_c;
       A_matrix(i,i+1)= i*e_c;
       A_matrix(i,i+2) = i*f_c;
    end

    %Removes the truncated terms
```

```
    A_matrix(:,n+2) = [];
    A_matrix(:,n+1) = [];
    A_matrix(n+2,:) = [];
    A_matrix(n+1,:) = [];
end
```

F.4 *Carleman_NonLinearized_func_2Var.m*

```
function [A_matrix, B_matrix, C_matrix, D_matrix, Initial_vector, E_matrix] =...
    Carleman_NonLinearized_func_2Var(h1, ulv, upa, h1_ulv, h1_h1, h1_h1_h1,
    h1_h1_ulv, initC, n)

    %Simplifies the notation of the partial derivatives
    a_c = h1;
    b_c = ulv;
    c_c = upa;
    d_c = h1_ulv;
    e_c = h1_h1/2;
    f_c = h1_h1_h1/6;
    g_c = h1_h1_ulv/2;

    %Defines the amount of terms to include in the Carleman linearization
    n = n;

    %Defines the initial condition of the water level
    Initial_level = initC;

    %Finds the initial condition for every x_dot
    Initial_vector = zeros(n,1);
    for i=1:n
        Initial_vector(i) = Initial_level^i;
    end

    %
    A_matrix = zeros(n+2);
    B_matrix = zeros(n,2);
    C_matrix = zeros(1,n);
    D_matrix = (0);
    C_matrix(1) = 1;
    B_matrix(1,1) = b_c;
    B_matrix(1,2) = c_c;

    %Constructs the A_matrix.
    %This pattern is clear after computing the first terms by hand.
    for i = 1:n
       A_matrix(i,i) = i*a_c;
       A_matrix(i,i+1)= i*e_c;
       A_matrix(i,i+2) = i*f_c;
    end

    %Removes the truncated terms
    A_matrix(:,n+2) = [];
    A_matrix(:,n+1) = [];
    A_matrix(n+2,:) = [];
    A_matrix(n+1,:) = [];


    E_matrix = cell(1,n);
    E_matrix{1} = zeros(n,2);
    E_matrix{1}(1,1) = d_c;
    E_matrix{1}(2,1) = g_c;

    for i = 2:n
        E_matrix{i} = zeros(n,2);
        E_matrix{i}(i-1,1) = i*b_c;
        E_matrix{i}(i,1) = i*d_c;
        E_matrix{i}(i+1,1) = i*g_c;
        E_matrix{i}(i-1,2) = i*c_c;
    end
    E_matrix{n}(n+1,:) = [];

end
```

F.5 *Carleman_NonLinearized_func.m*

```
function [A_matrix, B_matrix, C_matrix, D_matrix, Initial_vector, E_matrix] =...
    Carleman_NonLinearized_func(h1, ulv, upa, h1_ulv, h1_h1, h1_h1_h1, h1_h1_ulv,
    initC, n)

    %Simplifies the notation of the partial derivatives
    a_c = h1;
    b_c = ulv;
    c_c = upa;
    d_c = h1_ulv;
    e_c = h1_h1/2;
    f_c = h1_h1_h1/6;
    g_c = h1_h1_ulv/2;

    %Defines the amount of terms to include in the Carleman linearization
    n = n;

    %Defines the initial condition of the water level
    Initial_level = initC;

    %Finds the initial condition for every x_dot
    Initial_vector = zeros(n,1);
    for i=1:n
        Initial_vector(i) = Initial_level^i;
    end

    %
    A_matrix = zeros(n+2);
    B_matrix = zeros(n,1);
    C_matrix = zeros(1,n);
    D_matrix = (0);
    C_matrix(1) = 1;
    B_matrix(1) = b_c;

    %Constructs the A_matrix.
    %This pattern is clear after computing the first terms by hand.
    for i = 1:n
       A_matrix(i,i) = i*a_c;
       A_matrix(i,i+1)= i*e_c;
       A_matrix(i,i+2) = i*f_c;
    end

    %Removes the truncated terms
    A_matrix(:,n+2) = [];
    A_matrix(:,n+1) = [];
    A_matrix(n+2,:) = [];
    A_matrix(n+1,:) = [];


    E_matrix = cell(1,n);
    E_matrix{1} = zeros(n,1);
    E_matrix{1}(1) = d_c;
    E_matrix{1}(2) = g_c;

    for i = 2:n
        E_matrix{i} = zeros(n,1);
        E_matrix{i}(i-1) = i*b_c;
        E_matrix{i}(i) = i*d_c;
        E_matrix{i}(i+1) = i*g_c;
    end
    E_matrix{n}(n+1) = [];

end
```

F.6 *carleman_statespace_2Var.m*

```
%Carleman lin_part
clear
clc
n = 12;
```

```
syms a b c d e f g dh dulv dupa
syms x [1 n+3]

dhdot = a.*dh^1 + b.*dulv + c.*dupa + d.*dh^1.*dulv + e.*dh^2 + f.*dh^3 + g.*dh^2.*
    dulv;
%dhdot = a.*dh + e.*dh.^2 + f.*dh.^3;

C = cell(1,n);

%x2dot ->
for i = 2:n
    C{i} = expand(i*dh^(i-1).*dhdot);
    C{i} = subs(C{i},[dh^(i+3), dh^(i+2), dh^(i+1), dh^i, dh^(i-1), dh],{x(i+3), x(
    i+2), x(i+1),x(i),x(i-1), x1})
end
```

F.7 *carleman_statespace.m*

```
%Carleman lin_part
clear
clc
n = 12;
syms a b c d e f g dh dulv
syms x [1 n+3]

dhdot = a.*dh^1 + b.*dulv + d.*dh^1.*dulv + e.*dh^2 + f.*dh^3 + g.*dh^2.*dulv;
%dhdot = a.*dh + e.*dh.^2 + f.*dh.^3;

C = cell(1,n);

%x2dot ->
for i = 2:n
    C{i} = expand(i*dh^(i-1).*dhdot);
    C{i} = subs(C{i},[dh^(i+3), dh^(i+2), dh^(i+1), dh^i, dh^(i-1), dh],{x(i+3), x(
    i+2), x(i+1),x(i),x(i-1), x1})
end
```

F.8 *example_file_solver.m*

```
A = [1 3 4 ; 0 2 1 ; 1 7 6];
B = [1 0 0]';

eig(A)

% after installing YALMIP (toolbox) and SeDuMi (solver)

Q = sdpvar(3); %variable to find is a symmetric matrix 3x3
W = sdpvar(1,3,'full'); % variable to find a full (non-symmetric) matrix 1x3

alpha = 10;

inequality = Q >= 1e-9;
% inequality = [inequality,A'*Q+W'*B'*Q+Q*A+Q*B*W <= -1e-9]; % NO!
% BILINEAR MATRIX INEQUALITY (BMI)
inequality = [inequality,Q*A'+ W'*B'+ A*Q+ B*W+ 2*alpha*Q <= -1e-9]; %YES! LINEAR
    MATRIX INEQUALITY (LMI)

optimize(inequality)

Q = double(Q) %transform symbolic into numbers
W = double(W)

P = inv(Q)
K = W*inv(Q)
```

F.9 *Quadratic_controller.m*

```
function [K,P] = Quadratic_controller(A_matrix, B_matrix, E_cell, Vertex, Alpha, gamma,dupa_max, dulv_max,ak)

[r, k] = size(B_matrix);

n = r;

Q = sdpvar(n);   %variable to find is a symmetric matrix nxn
W = sdpvar(k,n,'full'); % variable to find a full (non-symmetric) matrix
inequality = [Q >= 1e-7*eye(n)]; %P>0
inequality = [inequality, [Q W(1,:)'; W(1,:) eye(1)*dulv_max^2]>=1e-9*eye(n+1)]; % limits ux
inequality = [inequality, [Q W(2,:)'; W(2,:) eye(1)*dupa_max^2]>=1e-9*eye(n+1)]; % limits ux2

% max_vertex = 0;
% for i = 1:2^n %finds the vertex furthest away from eq.
%     agent = norm(Vertex(1:end,i));
%     if agent >= max_vertex
%         max_vertex = agent;
%     end
% end

for i = 1:2^n
    Z = [];
    Y = [];
    for j = 1:n
       Z(j,:) = Vertex(1:end,i)'*E_cell{j};    %1xn Radvektor
       Y(:,j) = E_cell{j}'*Vertex(1:end,i);  %nx1 Kolonnevektor
    end

    if i == 2^n %max_vertex == max(norm(Vertex(1:end,i)))%if vertex number x is the same size as max
        inequality = [inequality, gamma*(Q*A_matrix' + W'*B_matrix' + A_matrix*Q + B_matrix*W) + ...
        W'*Y ...
        + Z*W + gamma*Q*2*Alpha <= -1e-9*eye(n)];

    else
        inequality = [inequality, gamma*(Q*A_matrix' + W'*B_matrix' + A_matrix*Q + B_matrix*W) + ...
        W'*Y ...
        + Z*W <= -1e-9*eye(n)];
    end
    inequality = [inequality, [1 Vertex(1:end,i)'; Vertex(1:end,i) Q] >= 1e-9*eye(n+1)]; %6d
end

[ak_r,ak_k] = size(ak);
for k_ = 1:ak_k
    inequality = [inequality, [1 gamma*ak(1:end, k_)'*Q; Q*ak(1:end, k_)*gamma Q] >= 1e-9*eye(n+1)];
    disp('Test')
end

%[1 x'; x Q]% x =     6d
%[1 vertex(1:end,i)'; vertex(1:end,i) Q] >=1e-7   x(i) = vertex(1:end,i)
%1x1 1xn              nx1             nxn

optimize(inequality)

Q = double(Q); %transform symbolic into numbers
W = double(W);

P = Q;
K = W*inv(Q); %kxn
end
```

F.10 *Quadratic_system.m*

```
function dx = Quadratic_system(t, x, strucK)

```

```
A = strucK.A;
B = strucK.B;
E = strucK.E;
K = strucK.K;

SS = size(A);
n = SS(1);

resultat = [];
for i = 1:n
    resultat(i,:) = x'*E{i}*K*x;
end

dx = (A+B*K)*x + resultat;
end
```

F.11 *Polytope_figures.m*

```
P = [-0.05 0; 0.05 0; -0.05 0.025; 0.05 0.025]

%[A,b] = vert2con(P);


k = convhull(P)

%plot(P(:,1),P(:,2))
%hold on
plot(P(k,1),P(k,2), 'b', 'LineWidth', 2, 'Marker', 'o')
hold on
fill(P(k,1), P(k,2),'g','FaceAlpha',0.5)
xlabel('z1')
ylabel('z2')
grid on
xlim([-0.06 0.06])
ylim([-0.02 0.03])
```

F.12 *vert2con.m* [11]

```
function [A,b] = vert2con(V)
% VERT2CON - convert a set of points to the set of inequality constraints
%            which most tightly contain the points; i.e., create
%            constraints to bound the convex hull of the given points
%
% [A,b] = vert2con(V)
%
% V = a set of points, each ROW of which is one point
% A,b = a set of constraints such that A*x <= b defines
%       the region of space enclosing the convex hull of
%       the given points
%
% For n dimensions:
% V = p x n matrix (p vertices, n dimensions)
% A = m x n matrix (m constraints, n dimensions)
% b = m x 1 vector (m constraints)
%
% NOTES: (1) In higher dimensions, duplicate constraints can
%            appear. This program detects duplicates at up to 6
%            digits of precision, then returns the unique constraints.
%        (2) See companion function CON2VERT.
%        (3) ver 1.0: initial version, June 2005.
%        (4) ver 1.1: enhanced redundancy checks, July 2005
%        (5) Written by Michael Kleder
%
% EXAMPLE:
%
% V=rand(20,2)*6-2;
% [A,b]=vert2con(V)
% figure('renderer','zbuffer')
% hold on
```

```
% plot(V(:,1),V(:,2),'r.')
% [x,y]=ndgrid(-3:.01:5);
% p=[x(:) y(:)]';
% p=(A*p <= repmat(b,[1 length(p)]));
% p = double(all(p));
% p=reshape(p,size(x));
% h=pcolor(x,y,p);
% set(h,'edgecolor','none')
% set(h,'zdata',get(h,'zdata')-1) % keep in back
% axis equal
% set(gca,'color','none')
% title('A*x <= b  (1=True, 0=False)')
% colorbar
k = convhulln(V);
c = mean(V(unique(k),:));
V=V-repmat(c,[size(V,1) 1]);
A  = NaN*zeros(size(k,1),size(V,2));
rc=0;
for ix = 1:size(k,1)
    F = V(k(ix,:),:);
    if rank(F,1e-5) == size(F,1)
        rc=rc+1;
        A(rc,:)=F\ones(size(F,1),1);
    end
end
A=A(1:rc,:);
b=ones(size(A,1),1);
b=b+A*c';
% eliminate dumplicate constraints:
[null,I]=unique(num2str([A b],6),'rows');
A=A(I,:); % rounding is NOT done for actual returned results
b=b(I);
return
```

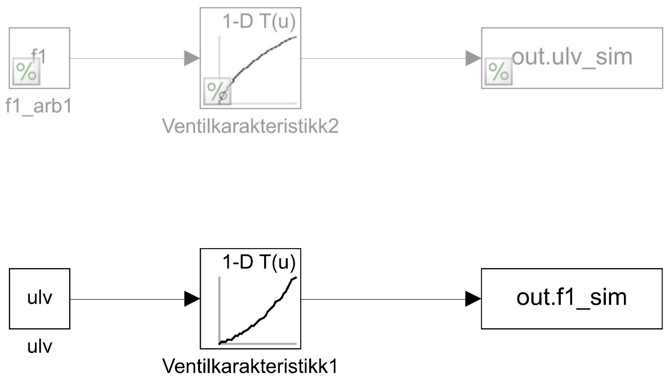
F.13 *EKSEMPEL_CARLEMAN.m*

```
x = 0:0.01:1;

dx = x + x.^2+ x.^3

y = dx;

plt_nl = plot(x,y,'b-','Linewidth',3)
%%
n = 3;

A = zeros(n+2);
B = zeros(n,1);
C = zeros(1,n);
D = (0);
C(1) = 1;

for i = 1:n
   A(i,i:i+2) = i;
end

%Removes the truncated terms
A(:,n+2) = [];
A(:,n+1) = [];
A(n+2,:) = [];
A(n+1,:) = [];

[r,k] = size(x);
z_mat = zeros(k,1);

for j = 1:k
    z_mat(1:end,j) = x.^j;
end

dz1 = zeros(1,k);
for l = 1:k
   dz1(l,1:n) = A*z_mat(l,1:n,1)';
```

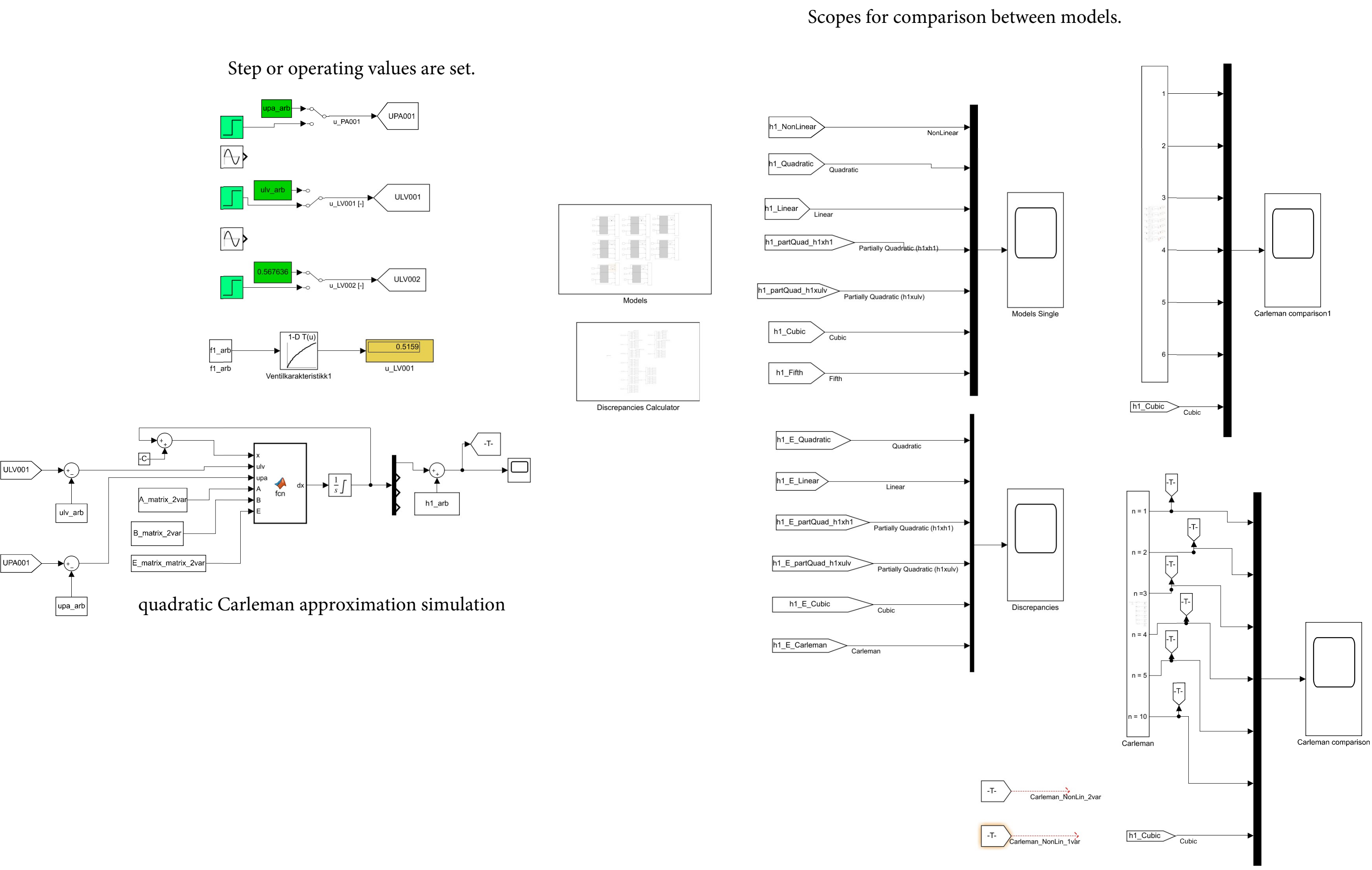
```
end

y1 = x + dz1;

hold on

plt_3 = plot(x,y1(1:end,1),'r--','Linewidth',2)

%%
n = 2;

A = zeros(n+2);
B = zeros(n,1);
C = zeros(1,n);
D = (0);
C(1) = 1;

for i = 1:n
   A(i,i:i+2) = i;
end

%Removes the truncated terms
A(:,n+2) = [];
A(:,n+1) = [];
A(n+2,:) = [];
A(n+1,:) = [];

[r,k] = size(x);
z_mat = zeros(k,1);

for j = 1:k
    z_mat(1:end,j) = x.^j;
end

dz1 = zeros(1,k);
for l = 1:k
   dz1(l,1:n) = A*z_mat(l,1:n,1)';
end

y2 = x + dz1;

hold on

plt_2 = plot(x,y2(1:end,1),'y--','Linewidth',2)


%%
n = 1;

A = zeros(n+2);
B = zeros(n,1);
C = zeros(1,n);
D = (0);
C(1) = 1;

for i = 1:n
   A(i,i:i+2) = i;
end

%Removes the truncated terms
A(:,n+2) = [];
A(:,n+1) = [];
A(n+2,:) = [];
A(n+1,:) = [];

[r,k] = size(x);
z_mat = zeros(k,1);

for j = 1:k
    z_mat(1:end,j) = x.^j;
end

dz1 = zeros(1,k);
```

```
for l = 1:k
   dz1(l,1:n) = A*z_mat(l,1:n,1)';
end

y3 = x + dz1;

hold on

plt_1 = plot(x,y3(1:end,1),'g--','Linewidth',2)

legend([plt_1, plt_2, plt_3, plt_nl],'Carleman: n = 1', 'Carleman: n = 2', '
    Carleman: n = 3', 'Nonlinear system (4.4)')
xlabel('$z_1(t) = x(t)$','Interpreter','latex')
ylabel('$\dot{x(t)} = \dot{z(t)}$','Interpreter','latex')
grid on
```

F.14 Simulink schemes

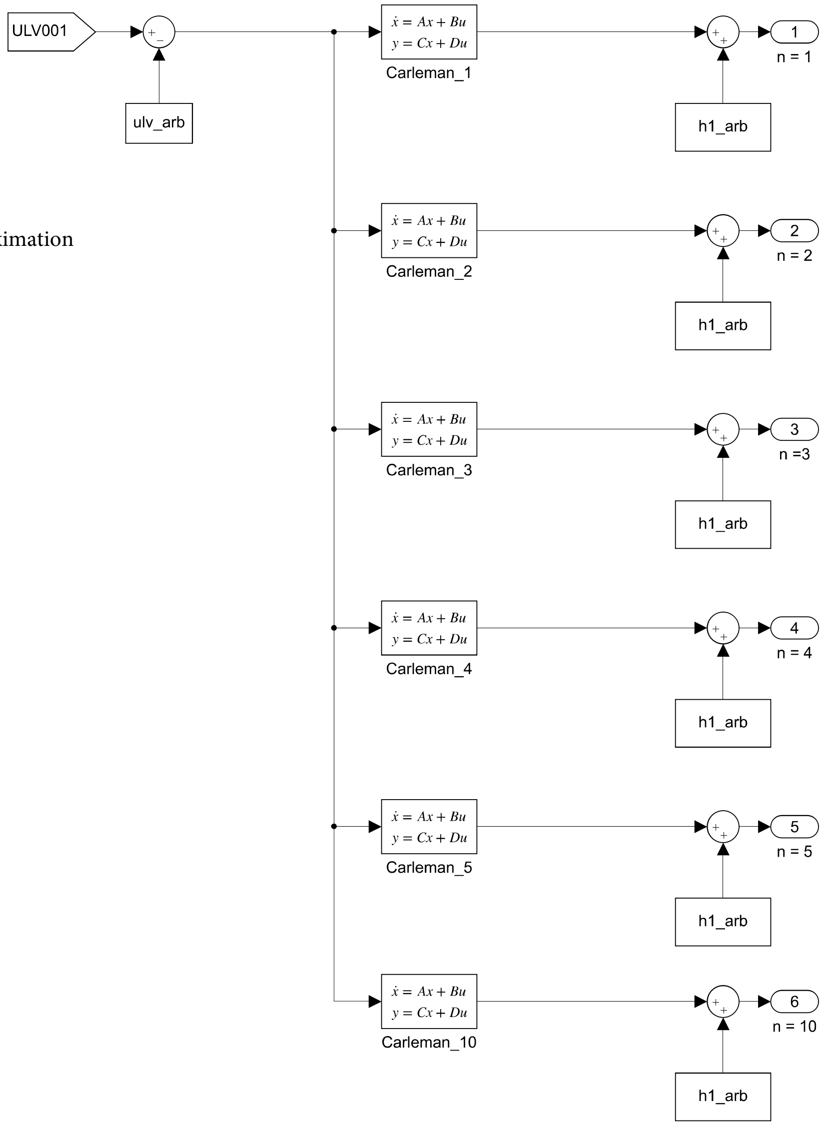


Finds the inverse of the valve characteristics.

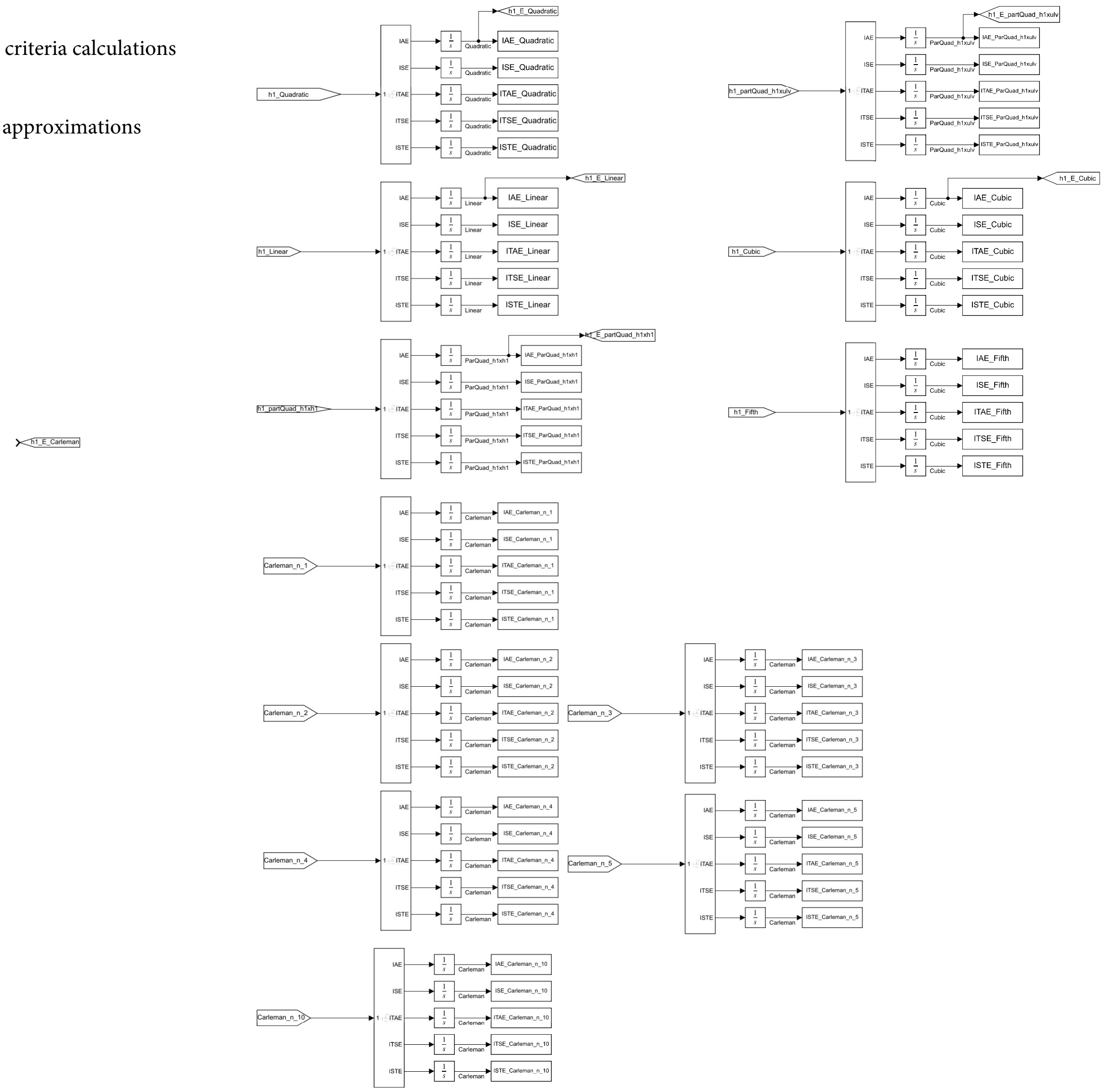


Scopes for comparison between models.
Step or operating values are set.
upa_arb
u_PA001
UPA001
ulv_arb
u_LV001 [-]
ULV001
0.567636
u_LV002 [-]
ULV002
f1_arb
1-D T(u)
Ventilkarakteristikk1
0.5159
u_LV001
Models
Discrepancies Calculator
-C-
x
ulv
upa
A
B
E
fcn
dx
A_matrix_2var
B_matrix_2var
E_matrix_matrix_2var
ulv_arb
upa_arb
h1_arb
-T-
quadratic Carleman approximation simulation
h1_NonLinear
NonLinear
h1_Quadratic
Quadratic
h1_Linear
Linear
h1_partQuad_h1xh1
Partially Quadratic (h1xh1)
h1_partQuad_h1xulv
Partially Quadratic (h1xulv)
h1_Cubic
Cubic
h1_Fifth
Fifth
Models Single
h1_E_Quadratic
h1_E_Linear
h1_E_partQuad_h1xh1
h1_E_partQuad_h1xulv
h1_E_Cubic
h1_E_Carleman
Carleman
Discrepancies
1
2
3
4
5
6
Carleman comparison1
n = 1
n = 2
n = 3
n = 4
n = 5
n = 10
Carleman comparison
Carleman_NonLin_2var
Carleman_NonLin_1var

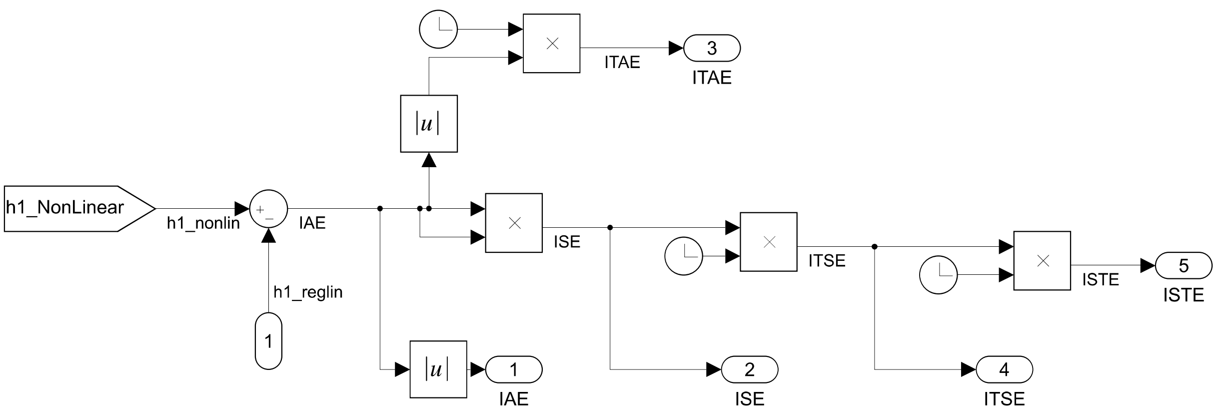
Linearized Carleman approximation for different n.



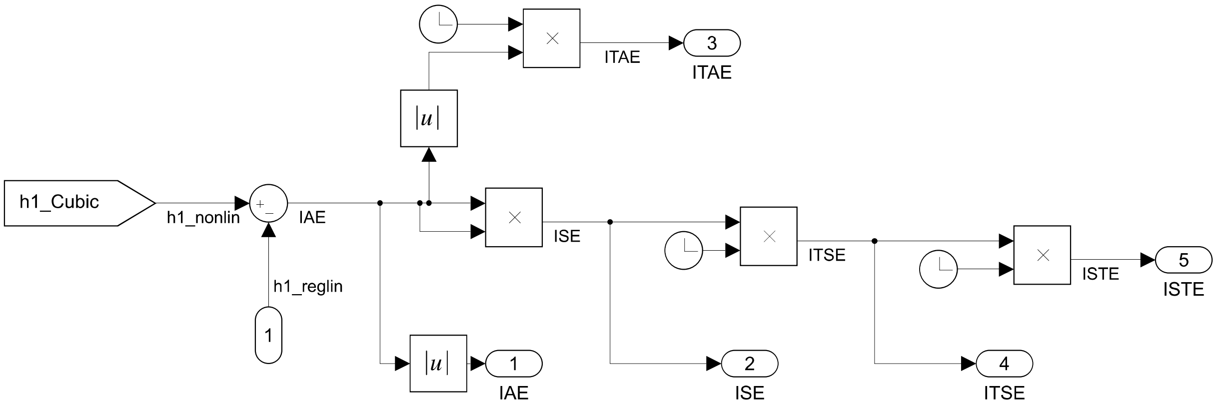
Integral performance criteria calculations
for:
Taylor models
Linearized Carleman approximations



Integral performance criteria calculations
for: The nonlinear model.

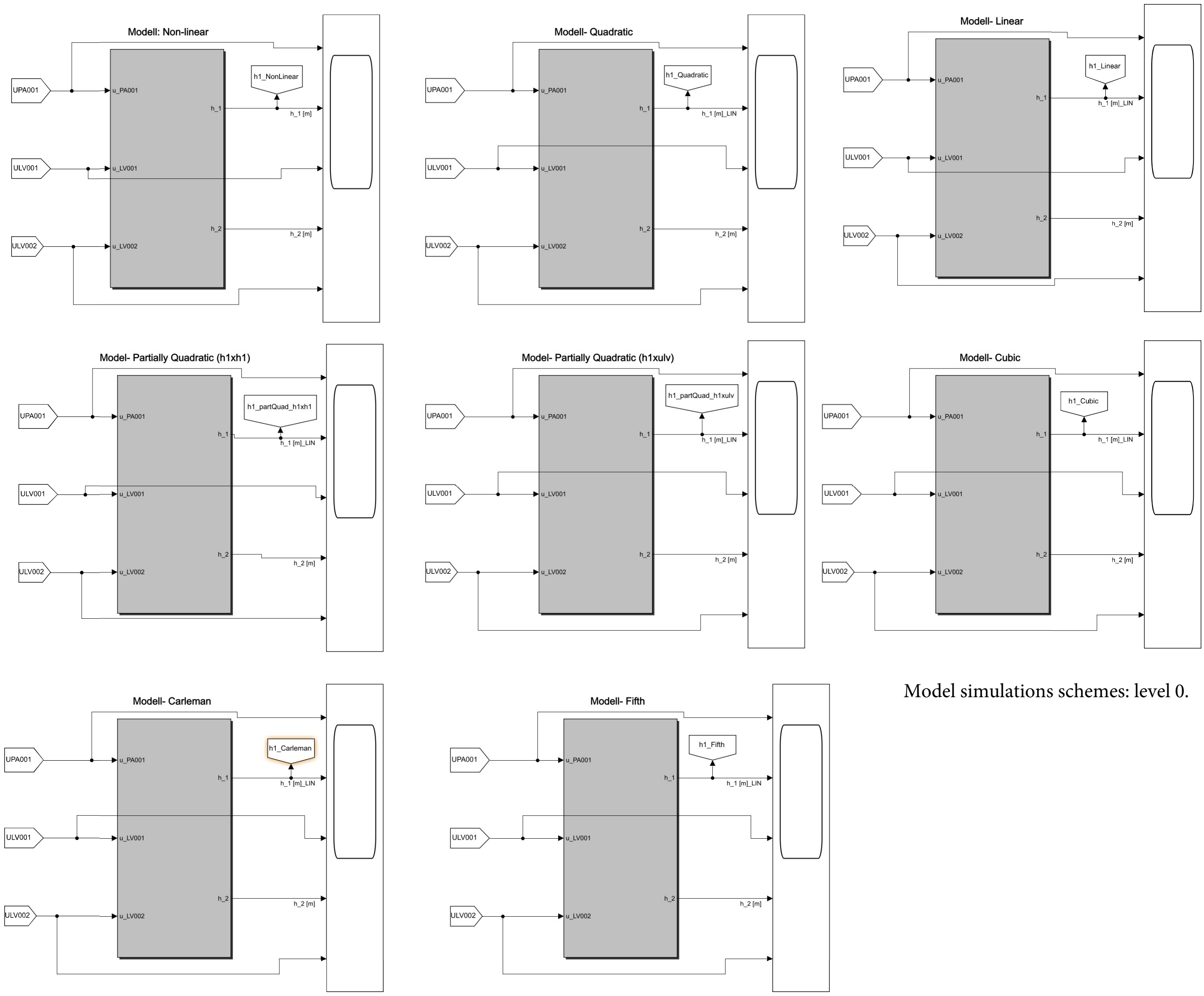


Integral performance criteria calculations
for: The cubic model.



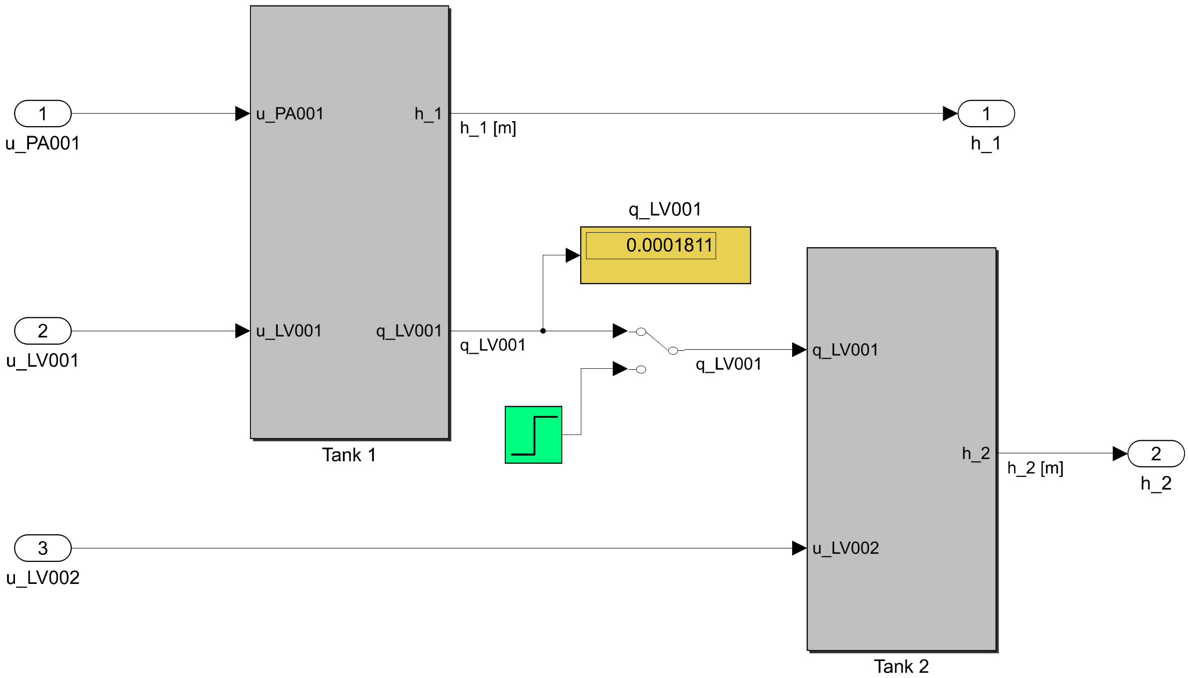
```
function dx = fcn(x,ulv,upa, A,B,E)

SS = size(A);
n = SS(1);
u = [ulv;upa];

resultat = zeros(n,1);
r = 0;
for i = 1:n
    resultat(i,:) = x'*E(:, r+1:r+2)*u;
    r = r+2;
end

dx = A*x+B*u + resultat;
end
```

Function calculating dx for the quadratic Carleman approximation using 2 variables.

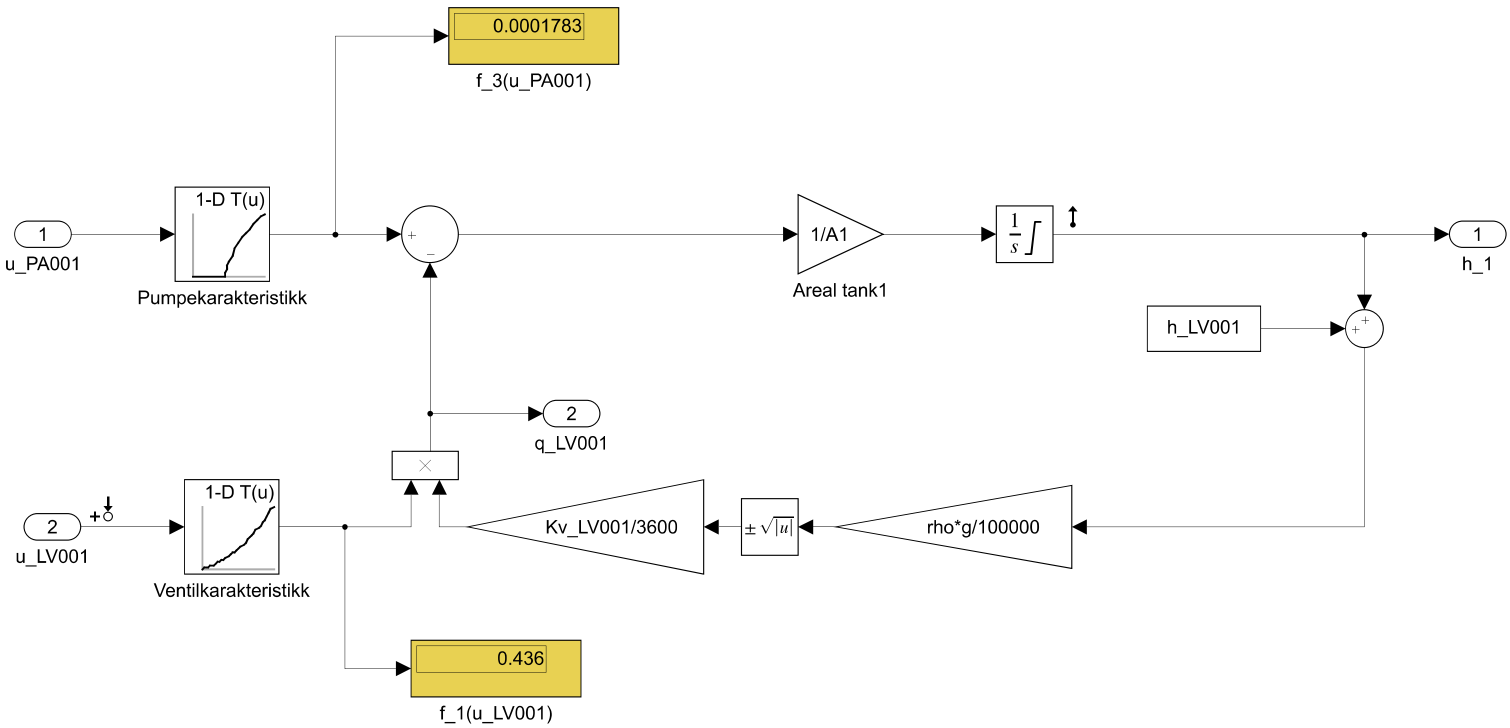


Model simulations schemes: level 0.

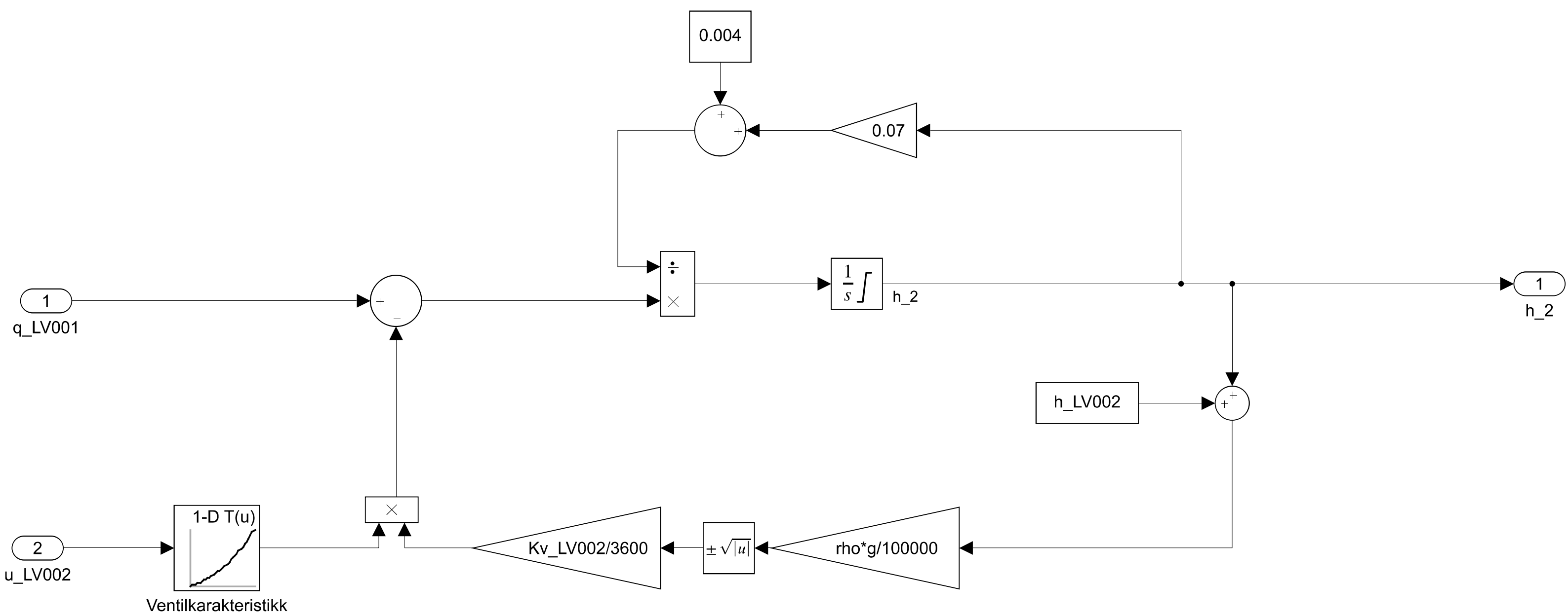
Inside a model scheme: level 1.

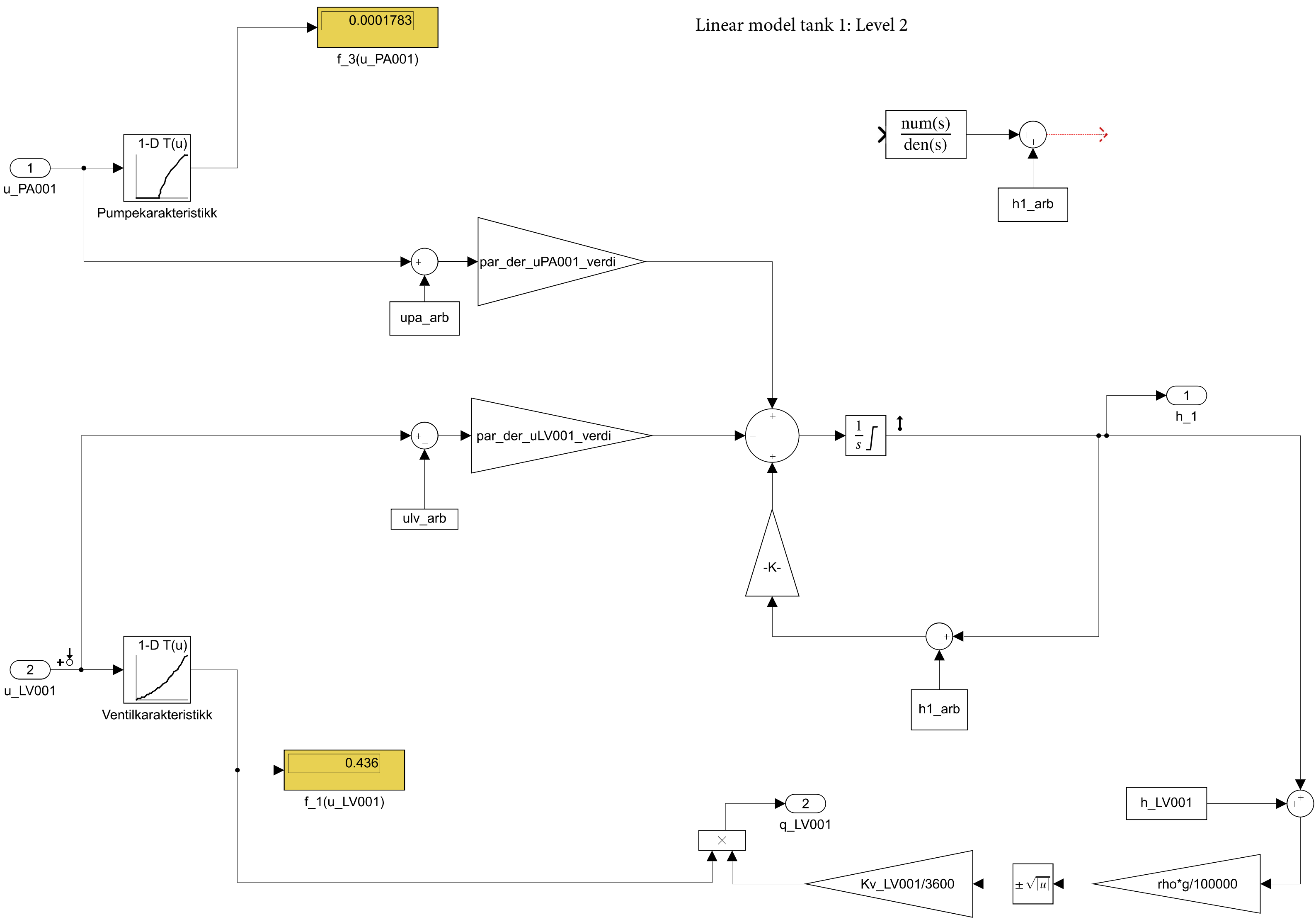


Nonlinear dynamical model model tank1: level 2.

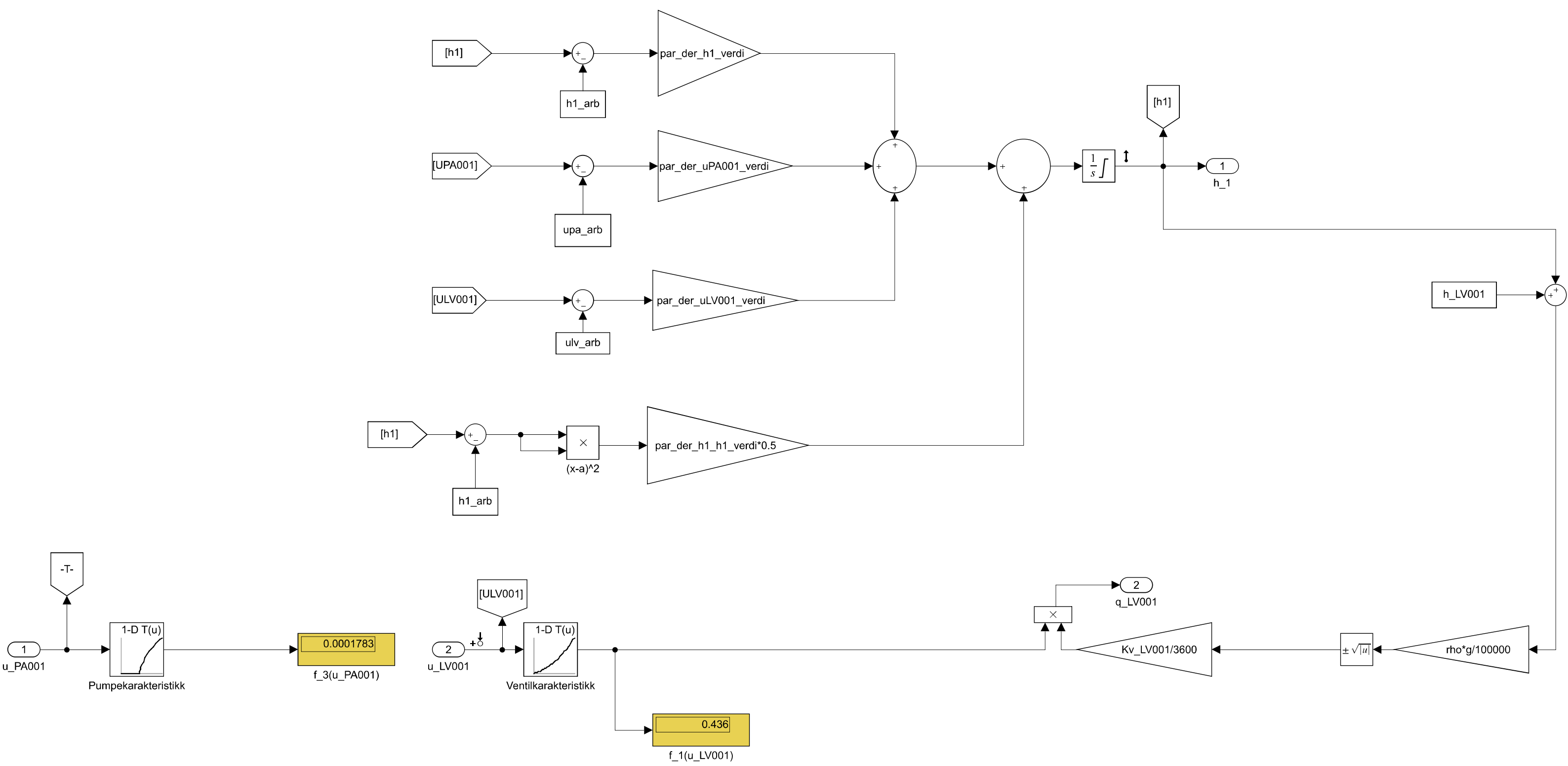


Nonlinear dynamical model tank 2: level2.



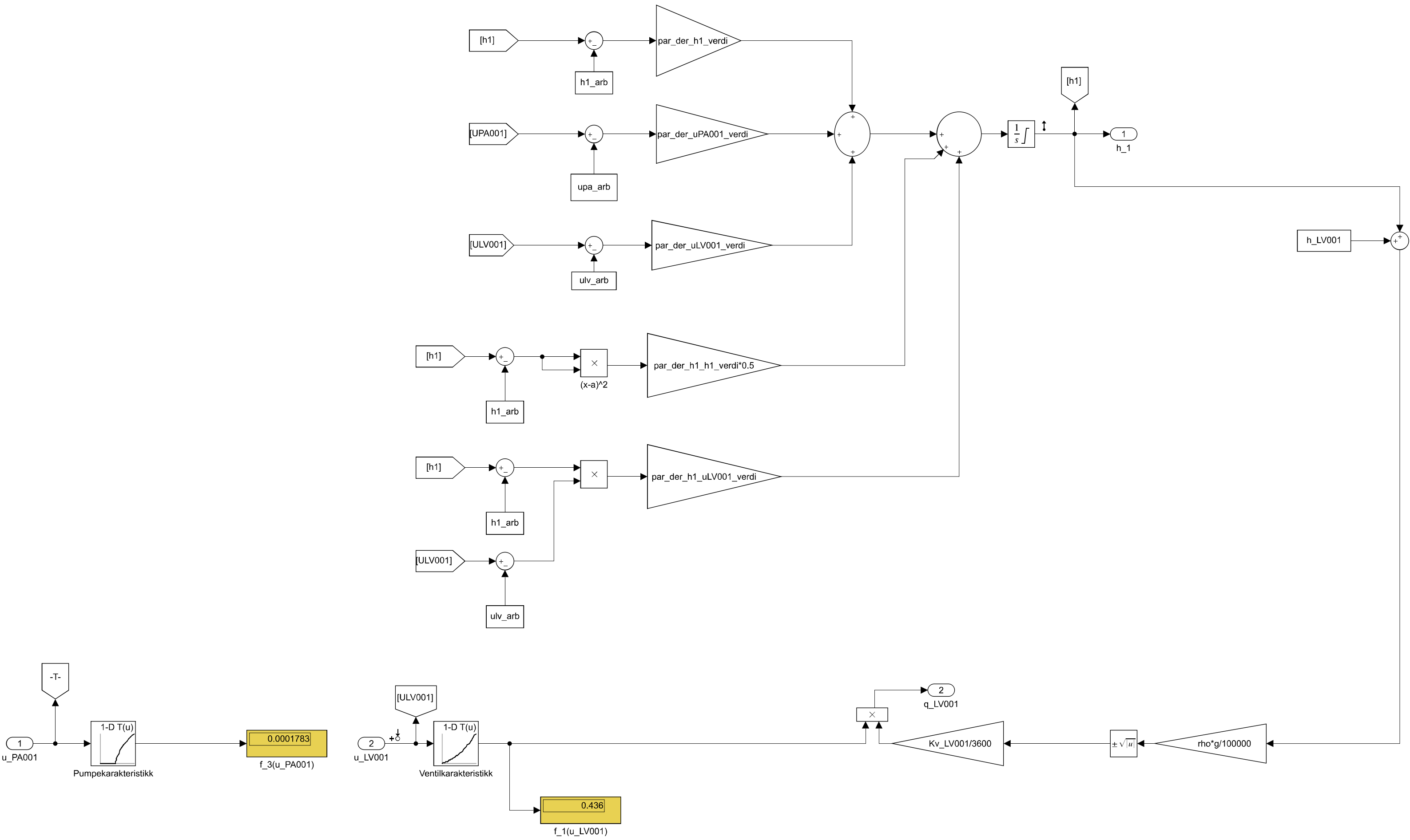

Linear model tank 1: Level 2
0.0001783
f_3(u_PA001)
num(s)
den(s)
1
u_PA001
1-D T(u)
Pumpekarakteristikk
h1_arb
par_der_uPA001_verdi
upa_arb
par_der_uLV001_verdi
ulv_arb
1
h_1
-K-
2
u_LV001
1-D T(u)
Ventilkarakteristikk
h1_arb
0.436
f_1(u_LV001)
2
q_LV001
h_LV001
Kv_LV001/3600
rho*g/100000

PQA: Level 2

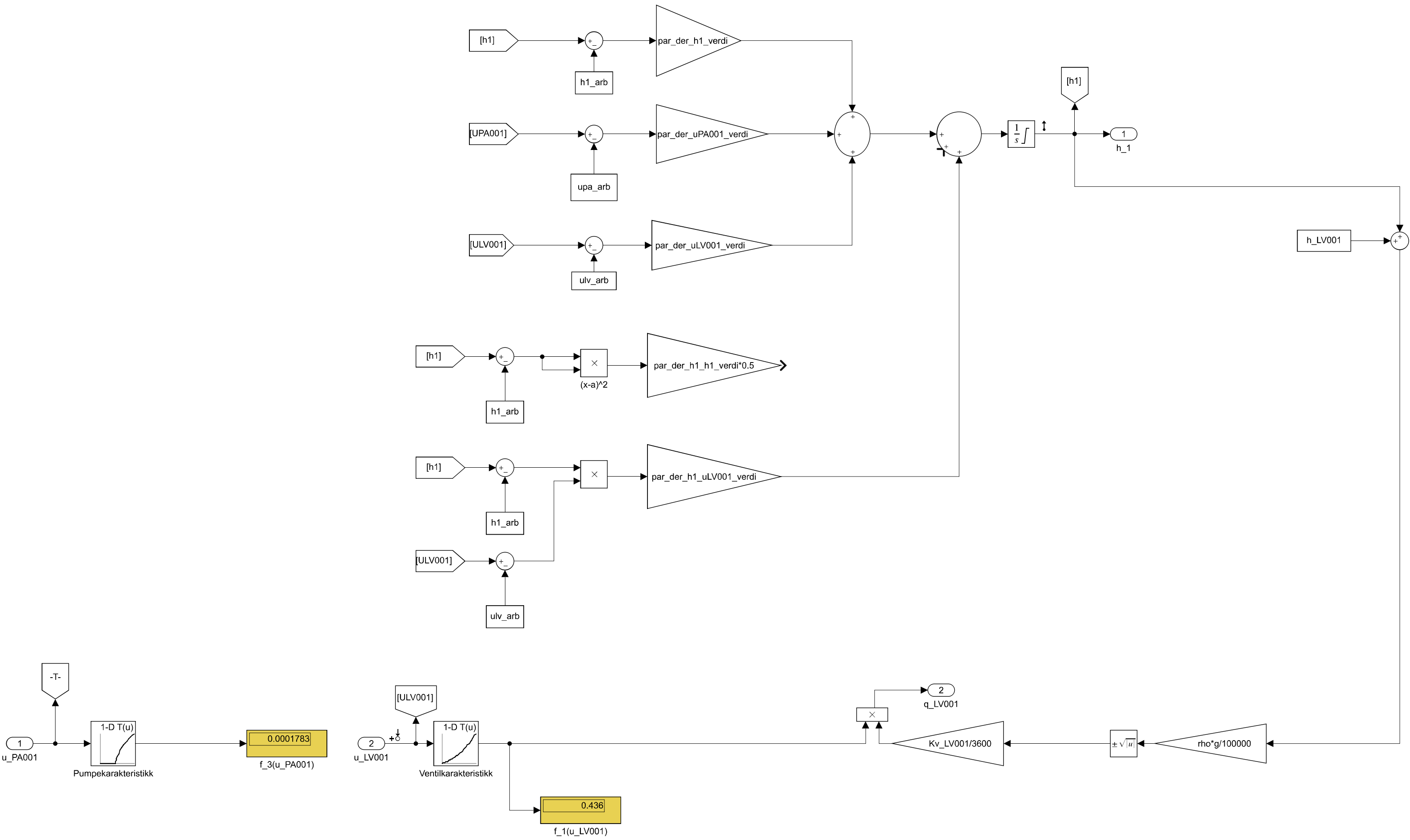


[h1]
h1_arb
par_der_h1_verdi
[UPA001]
upa_arb
par_der_uPA001_verdi
[ULV001]
ulv_arb
par_der_uLV001_verdi
[h1]
h1_arb
(x-a)^2
par_der_h1_h1_verdi*0.5
[h1]
1
h_1
h_LV001
-T-
1
u_PA001
1-D T(u)
Pumpekarakteristikk
0.0001783
f_3(u_PA001)
[ULV001]
2
u_LV001
1-D T(u)
Ventilkarakteristikk
0.436
f_1(u_LV001)
2
q_LV001
Kv_LV001/3600
rho*g/100000

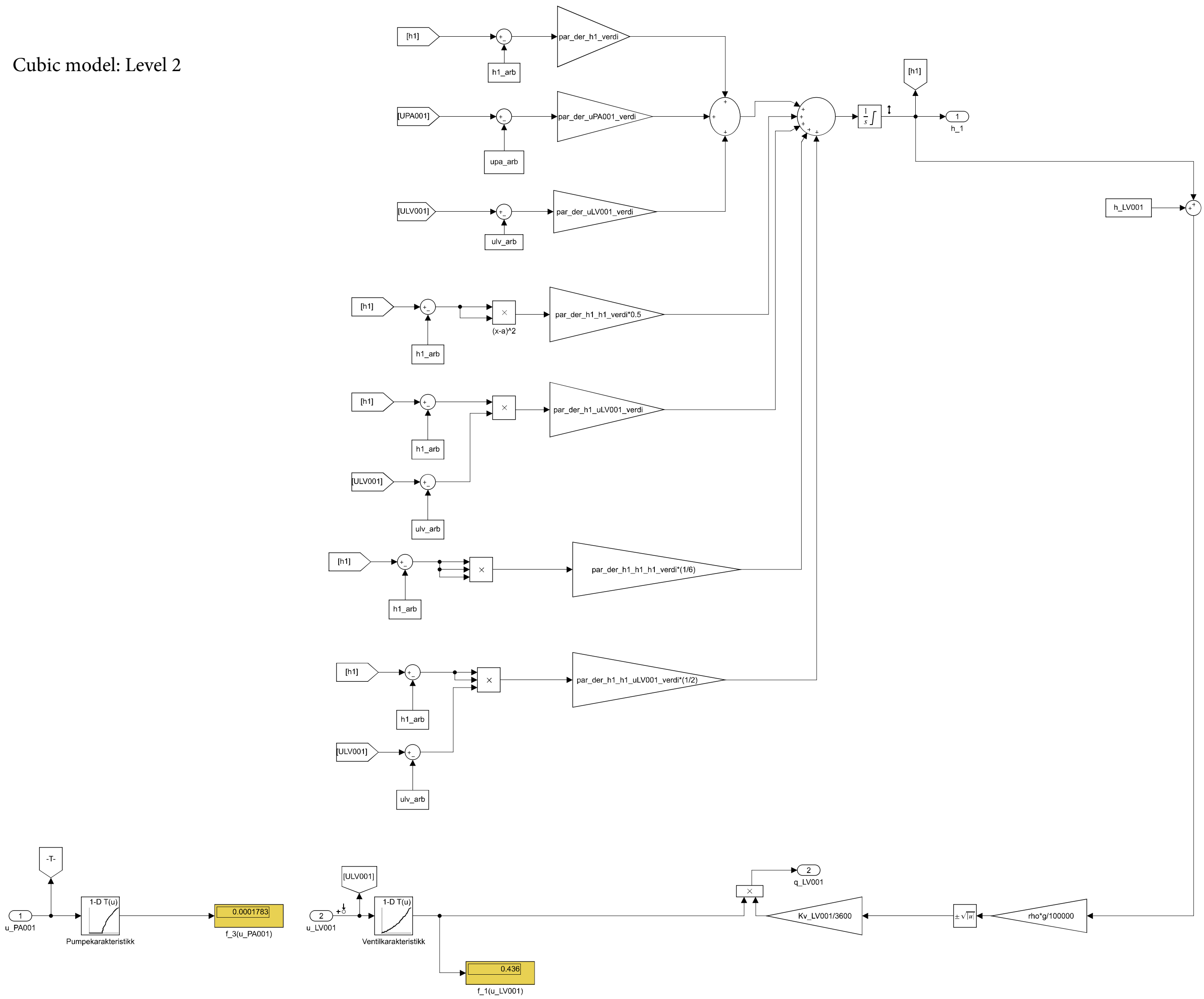
Quadratic model: Level 2



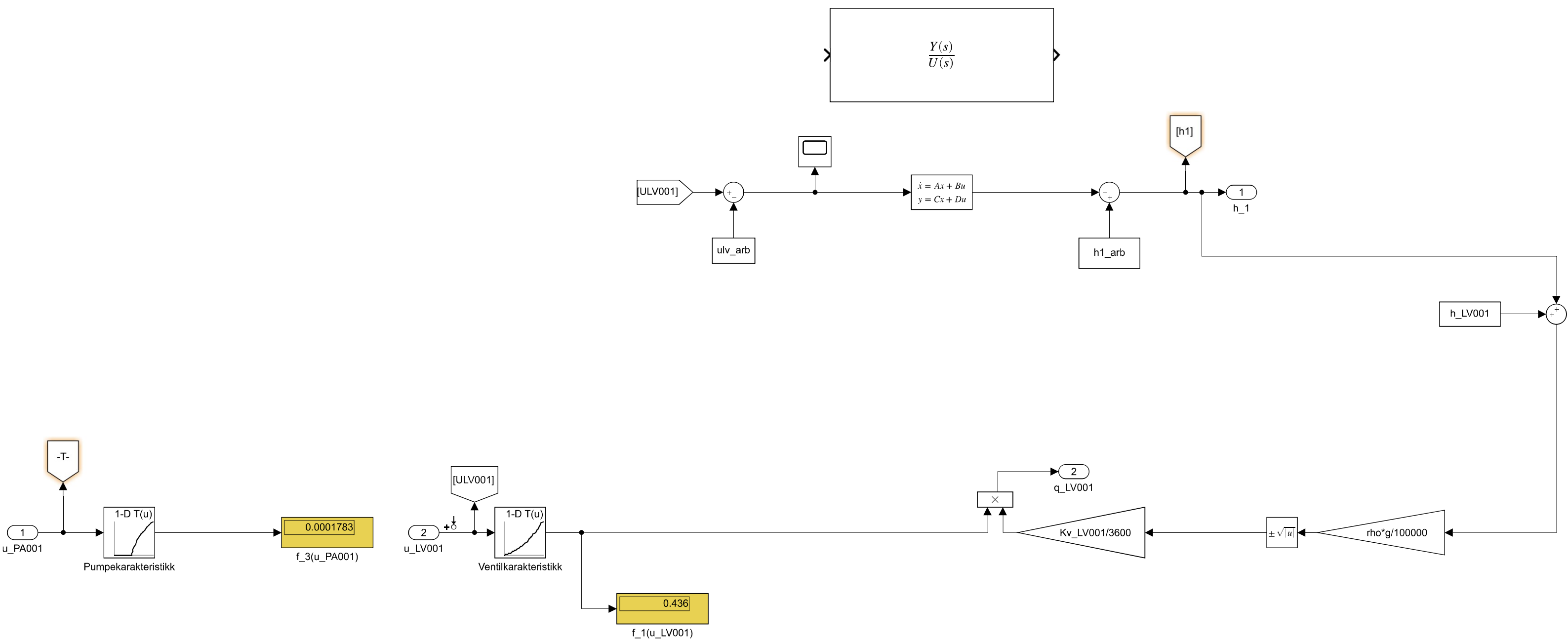
PQB: Level 2


[h1]
h1_arb
par_der_h1_verdi
[UPA001]
upa_arb
par_der_uPA001_verdi
[ULV001]
ulv_arb
par_der_uLV001_verdi
[h1]
1
h_1
h_LV001
[h1]
h1_arb
(x-a)^2
par_der_h1_h1_verdi*0.5
[h1]
h1_arb
par_der_h1_uLV001_verdi
[ULV001]
ulv_arb
-T-
1
u_PA001
1-D T(u)
Pumpekarakteristikk
0.0001783
f_3(u_PA001)
[ULV001]
2
u_LV001
1-D T(u)
Ventilkarakteristikk
0.436
f_1(u_LV001)
2
q_LV001
Kv_LV001/3600
rho*g/100000

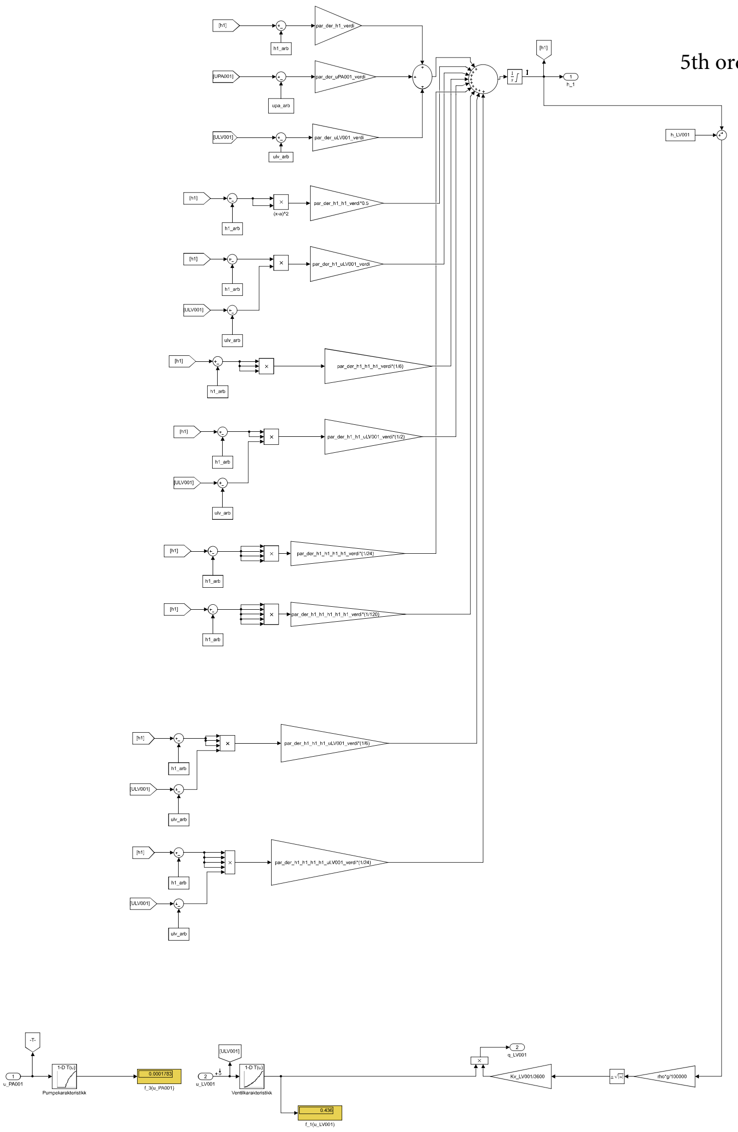
Cubic model: Level 2

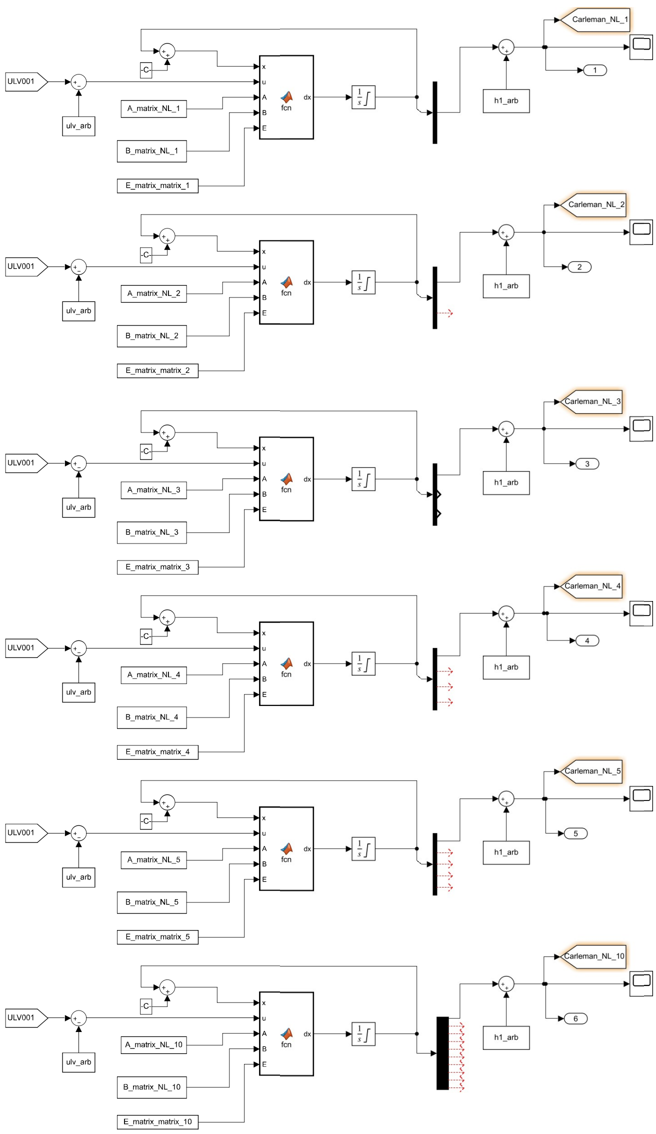


Linearized Carleman model: Level 2



5th order model: Level 2



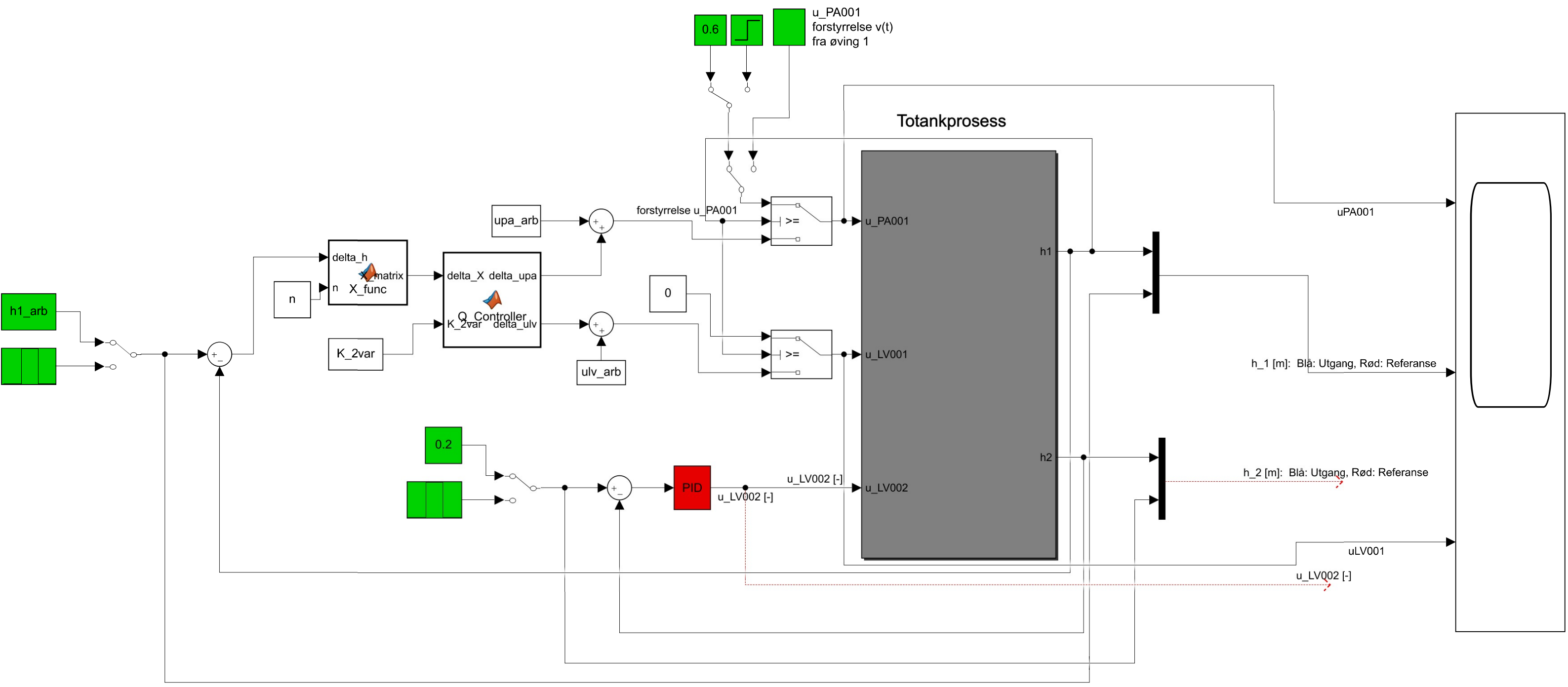


quadratic Carleman approximation models for different n.

```
function dx = fcn(x,u, A,B,E)

SS = size(A);
n = SS(1);
resultat = zeros(n,1);
for i = 1:n
         resultat(i,:) = x'*E(:,i)*u;
end
dx = A*x+B*u + resultat;
end
```

Function calculating dx for the quadratic Carleman approximation using 1 variable.

Scheme of the implemented controller for the quadratic Carleman approximation.



```
function X_matrix = X_func(delta_h, n)

X_matrix = zeros(n,1);
for i = 1:n
    x = delta_h^i;
    X_matrix(i, 1) = x;
end
```

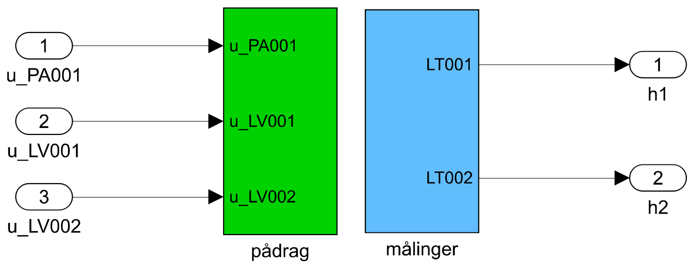
Function creating the z(t) state vector.

```
function [delta_upa,delta_ulv]= Q_Controller(delta_X,K_2var)

U = K_2var.*delta_X;

delta_ulv = U(1, 1:end);
delta_upa = U(2, 1:end);
```

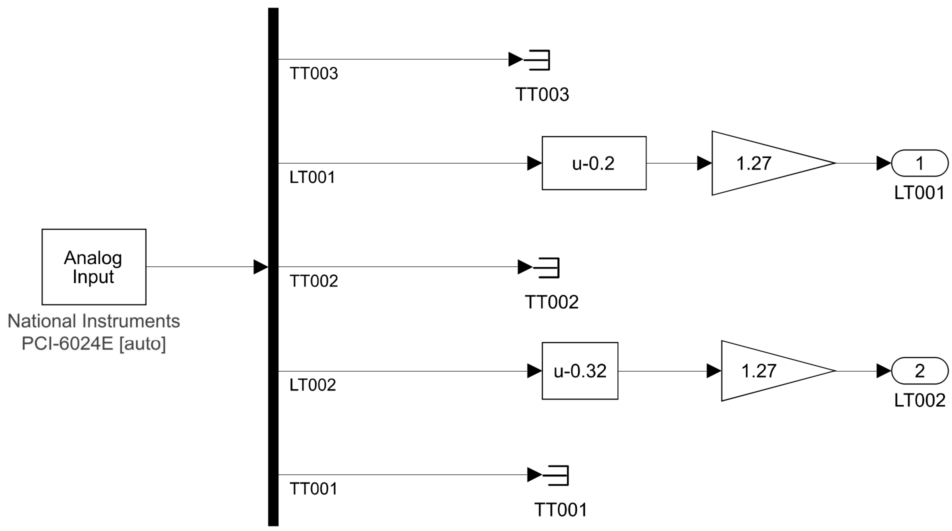
Function calculating the delta variables uLV001 and UPA001 using the controller gain.

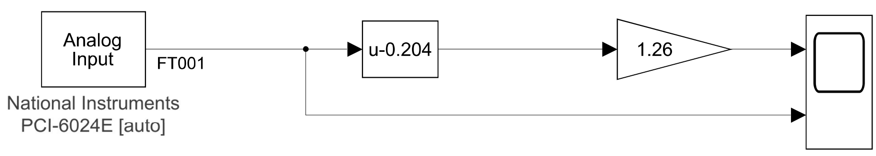
The calculated uPA001, uLV001 and uLV002 signals are sent to the actuators.

The measurements from the two-tank system feeding into the controller.

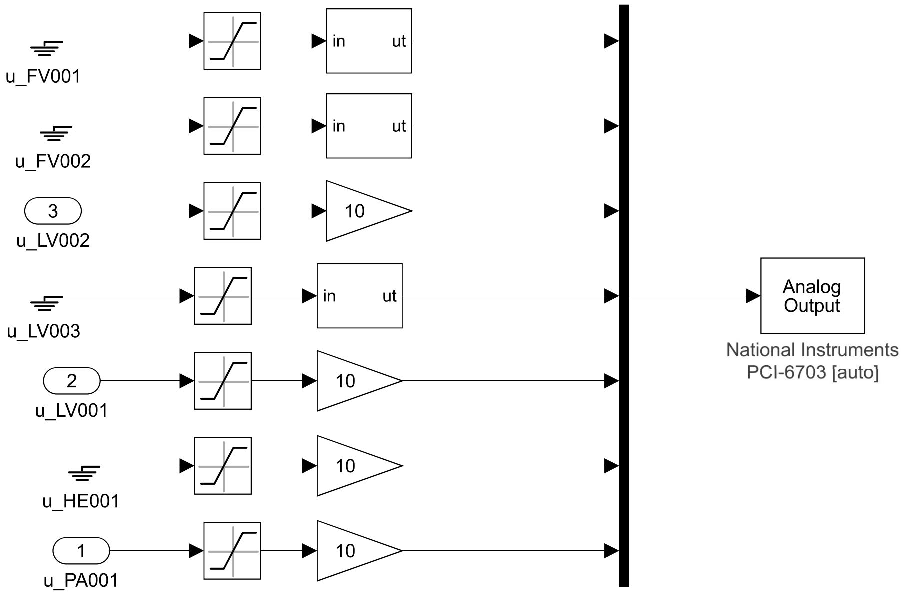


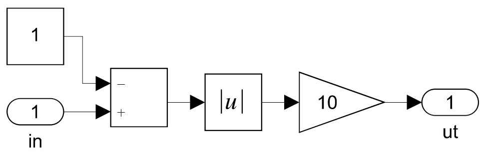
Analog inputs.

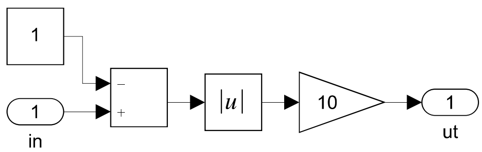


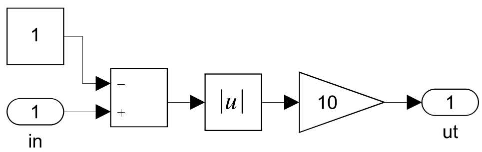


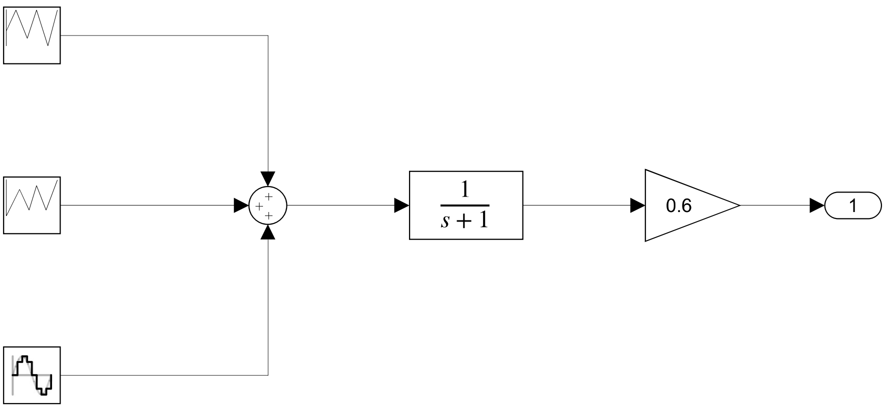
Analog outputs.




1
1
in
−
+
|u|
10
1
ut


1
−
+
|u|
10
1
in
1
ut


1
1
in
−
+
|u|
10
1
ut



+
+
+
1
s + 1
0.6
1