Jakob Ludwiger, BSc

Networked Sliding Mode Control – Explicit Analysis of Delays

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Martin Horn
Ass.Prof. Dipl.-Ing. Dr.techn. Martin Steinberger

Institute of Automation and Control

Graz, 09 2016

AFFIDAVIT

Abstract

As networked induced delays are a major problem in modern networked control systems, this article is devoted to this topic. The networked architecture was combined with a sliding mode controller to generate a very powerful system, a networked sliding mode control system. This architecture combines the advantages of the networked structure, e.g. the flexibility or the ability to overcome large physical distances with minimized wiring effort, with the advantages of the sliding mode controller, e.g. the ability to completely compensate perturbations with some defined properties. The main question is how the parameters of the sliding mode controller influences the ultimate boundedness of the closed loop solutions. In order to answer this question, conditions are derived which ensure ultimate boundedness if some preconditions are fulfilled. This step made it possible to perform several simulations with different plant configurations and parameter settings. The resulting stability regions were compared in order to analyse the influence of these configurations. This stability region plot shows the ultimately bounded regions as well as the unstable regions, depending on the sample time of the controller and the time delay.

Zeitverzögerungen, die durch Datenübertagung über Netzwerke verursacht werden, bringen bei modernen netzwerkbasierten Regelkreisen enorme Schwierigkeiten mit sich. In der folgende Arbeit wird diese Art von ungewollten Verhalten behandelt. Diese netzwerkbasierte Regelkreisarchitektur wurde mit einem Sliding Mode Regler kombiniert. Das Resultat dieser Kombination ist eine sehr mächtige Struktur und wird als „Networked Sliding Mode Control“ bezeichnet. Diese Architektur vereinigt die Vorteile der netzwerkbasierten Regelung (z.B. die Flexibilität bzw. die Möglichkeit größere physikalische Entfernungen mit sehr geringem Verdrahtungsaufwand zu überwinden) mit denen der Sliding Mode Regelung (z.B. die Möglichkeit Störungen mit bestimmten Eigenschaften komplett zu eliminieren). In dieser Arbeit wird auf die Fragestellung eingegangen, wie sich die unterschiedlichen Parameter des Sliding Mode Reglers auf die „ultimate boundedness“ auswirken. Um diese Frage beantworten zu können, werden Bedingungen benötigt, die diese ultimate boundedness garantieren. Damit wird es möglich, Simulationen mit verschiedenen Streckenkonfigurationen sowie Parametrierungen durchzuführen, die Stabilitätsregionen als Ergebnis liefern. Mittels Vergleich dieser Stabilitätsregionen wurden die Einflüsse der Parameter analysiert. Stabilitätsregionen sind Abbildungen, welche die ultimately bounded Regionen und die instabilen Regionen in Abhängigkeit der Diskretisierungszeit und der Totzeit darstellen.

Contents

Abstract **iii**

1 Introduction and Related Work **1**
- 1.1 Networked Control Systems 1
  - 1.1.1 Network Induced Delay 1
- 1.2 Sliding Mode Controller 2
- 1.3 Networked Sliding Mode Control System 4

2 Modelling a Networked Control Structure with Delays **5**
- 2.1 Control Loop Architecture 5
- 2.2 Small Delay Case 6
  - 2.2.1 Large Delay Case 8
- 2.3 Stability for NCSs using Linear State Controllers 9

3 Designing a Discrete Sliding Mode Controller **12**
- 3.1 Quasi Sliding Mode 12
  - 3.1.1 Attributes of Discrete Sliding Mode 14
- 3.2 Reaching Law Approach 14
- 3.3 Designing the Discrete Control Law 15
- 3.4 Designing the Sliding Surface 15
  - 3.4.1 Transformation Matrix 16
  - 3.4.2 Placing the Poles 16
  - 3.4.3 Example 17

4 Stability Analysis for Networked Sliding Mode Control Systems **21**
- 4.1 State Transformation 22
- 4.2 Verification of the Transformed Model 26
- 4.3 Stability Analysis for Constant Delays 28
- 4.4 Stability Regions' Dependence on Plant 29
  - 4.4.1 Configuration 29
  - 4.4.2 Results 30
  - 4.4.3 Conclusions 37
- 4.5 Stability Regions' Dependence on Linear Reaching Parameter q 37
  - 4.5.1 Configuration 37
  - 4.5.2 Results 40
  - 4.5.3 Conclusions 49
- 4.6 Comparison of Stability Regions Using Sliding Mode Controllers and Linear State Controllers 49
  - 4.6.1 Configuration 49
  - 4.6.2 Results 52
  - 4.6.3 Conclusions 63
- 4.7 Stability Regions' Dependance on Eigenvalue λ_c 63
  - 4.7.1 Configuration 63

4.7.2 Results 64
4.7.3 Conclusions 71

5 Conclusions **76**

Bibliography **79**

List of Figures

1.1 Reaching phase and sliding phase for second order system and different initial conditions . . . 3
1.2 Zigzag motion of sliding variable σ . . . 4

2.1 Architecture of the control loop . . . 5
2.2 Timing from sensor to actuator path . . . 6
2.3 Timing diagram for discrete networked control system . . . 7
2.4 Timing diagram for delays $h < \tau_k < 2h$ . . . 8
2.5 Timing diagram for delays $0 \leq \tau_k \leq 2h$ . . . 9
2.6 Stability region depending on sampling period h and the delay time τ in percentage of h . . . 10
2.7 Stability region depending on sampling period h and the absolute delay time τ . . . 11

3.1 Type-I Trajectories of discrete SMC system . . . 13
3.2 Type-P Trajectories of discrete SMC system . . . 13
3.3 Simulation result for state vector $\boldsymbol{x}_k$ . . . 19
3.4 Simulation result for state vector $\boldsymbol{\xi}$ . . . 19

4.1 Simulink model to verify the mathematical model in (4.15) . 27
4.2 Verification result for states . . . 28
4.3 Stability areas for different plant configurations of Type 1a . 32
4.4 Stability areas for different plant configurations of Type 1b . 34
4.5 Stability areas for different plant configurations of Type 2 . 36
4.6 Stability areas for different plant configurations of Type 3 . 39
4.7 Stability areas for same Type 1a plant with growing linear reaching factor q_{rel} . . . 42
4.8 Impact of relative linear reaching parameter q_{rel} on the percentage of stable areas for different Type 1a plant systems. . 42
4.9 Stability areas for same Type 1b plant with growing linear reaching factor q_{rel} . . . 44
4.10 Impact of relative linear reaching parameter q_{rel} on the percentage of stable areas for different Type 1b plant systems. . 44
4.11 Stability areas for same plant with growing linear reaching factor q_{rel} . . . 47
4.12 Impact of relative linear reaching parameter q_{rel} on the percentage of stable areas for different Type 2 plant systems. . . 47
4.13 Stability areas for same plant with growing linear reaching factor q_{rel} . . . 49
4.14 Stability areas for same plant with growing linear reaching factor q_{rel} . . . 51
4.15 Impact of relative linear reaching parameter q_{rel} on the percentage of stable areas for different Type 3 plant systems. . . 51

4.16 Comparison of stability regions by using sliding mode controller and linear state controller for Type 1a system (4.18) . 54
4.17 Comparison of stability regions by using sliding mode controller and linear state controller for Type 1a system (4.18) . 55
4.18 Comparison of stability regions by using sliding mode controller and linear state controller for Type 1b system (4.20) . 56
4.19 Comparison of stability regions by using sliding mode controller and linear state controller for Type 1b system (4.21) . 57
4.20 Comparison of stability regions by using sliding mode controller and linear state controller for Type 2 system (4.22) . . 59
4.21 Comparison of stability regions by using sliding mode controller and linear state controller for Type 2 system (4.23) . . 60
4.22 Comparison of stability regions by using sliding mode controller and linear state controller for Type 3 system (4.24) . . 61
4.23 Comparison of stability regions by using sliding mode controller and linear state controller for Type 3 system (4.25) . . 62
4.24 Change of the stability area by varying the eigenvalue λ_c by using Type 1a plant systems 66
4.25 Dependency of ultimately bounded areas on the eigenvalue λ_c when using different Type 1a plant systems 66
4.26 Change of the stability area by varying the eigenvalue λ_c by using Type 1b plant systems 68
4.27 Dependency of ultimately bounded areas on the eigenvalue λ_c when using different Type 1b plant systems 68
4.28 Change of the stability area by varying the eigenvalue λ_c by using Type 2 plant systems 70
4.29 Dependency of ultimately bounded areas on the eigenvalue λ_c when using different Type 2 plant systems 70
4.30 Change of the stability area by varying the eigenvalue λ_c by using Type 3 plant systems 73
4.31 Change of the stability area by varying the eigenvalue λ_c by using Type 3 plant systems 75
4.32 Dependency of ultimately bounded areas on the eigenvalue λ_c when using different Type 3 plant systems 75

Chapter 1

Introduction and Related Work

1.1 Networked Control Systems

As communication technology has been improved enormously in the last couple of years, entirly new control system architectures are developed. These systems are called “Networked Control Systems” (NCSs). This architecture has several advantages. One main benefit of this controller type is the flexibility [1]. It is, for instance, very easy to add new sensors to a networked control system. This flexibility is also manifested by the fact that it is very simple to share data among multiple NCSs (e.g. sensor data). Also, adding supplementary components like actuators does not result in complete structural changes. By using this networked structure, large physical distances can be overcome with minimized wiring effort. These and many more advantages triggered a great demand for these networked control systems.

Of course there are also disadvantages. Depending on the network structure there may occur different problems. One major problem is the posibility of packages getting lost or wrong information being received. There is a lot of work done dealing with this kind of perturbation [2]–[4].

Other approaches which should reduce the possibility of packages being lost are to deplete the network load. Therefore, it is assumed that the quality of the connection and the amount of transferred data are connected to each other. This means, for example, that the package loss rate increases with increasing network load. There are also several research papers dealing with this data reduction techniques (e.g. [5]–[7]).

Another challenge is the prescence of delays. The research field investigating methods to deal with this issue is vast and very active. Numerous scientific papers concerning this topic have already been published (e.g. [8]–[10]). As this work is focused on this type of undesired network behaviour, some additional insights are reasonable.

1.1.1 Network Induced Delay

When actuators, sensors and controllers exchange data over the network, network induced delayes occur in NCSs [8]. The delays cause several undesired phenomena. Not considering these delays in the controller design could not only lead to poor performance but also to unstable behaviour. These delays are of course dependent on the media access protocol (MAC). This protocol defines which node on the network is allowed to send. Also, the handling of collisions is specified in such a protocol. In general, time varying or even random delays have to be considered but there are also protocols which ensure constant delays.

So the protocols can be separated into two categories "random access" and "scheduling" [11]. Examples for random access protocols are the "Control Area Network" (CAN) and "Ethernet". Both bus standards are based on the carrier sense multiple access (CSMA) media access protocol but with different collision handling techniques. The CSMA protocol specifies that each node has to monitor the bus and ensure that no other node is currently sending before starting to send. Of course it is possible that two nodes start to send simultaneously. Then collision handling is needed. The CAN bus standard works with the collision resolution method, which means that the node sending the higher prioritised message keeps sending and the other one stops. The collision detection method is used in Ethernet bus standard. Both nodes stop to send immediately after a collision was detected. After a random time, the node tries sending again. This collision handling techniques lead to unbounded random delays, which is a main drawback.

Profinet is an example for the use of a scheduled MAC protocol. The isochronous real time (IRT) concept ensures that every node is assigned to a timeslot. This timeslot is guaranteed in every bus cycle, which leads to bounded and even constant delays.

1.2 Sliding Mode Controller

Developing a robust system is the main goal of modern control theory. This means that the system is capable of handling model uncertainties as well as perturbations. There are several approaches to achieve this goal, e.g. H_∞ control [12]. However, in this article the sliding mode control approach is investigated because this type of controller is not only capable of reaching the desired output in finite time [13] but also has some good properties.

The main idea of the sliding mode approach is illustrated in the following section by using a second order time invariant linear system in the following normal form

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= a_1x_1 + a_2x_2 + bu + b_2\xi\end{aligned}$$

where x_1 and x_2 represent the states, u is the control input and ξ denotes an unknown perturbation. The factors a_1, a_2, b and b_2 are constant scalars. The perturbation ξ is called matched, because it effects the same channel as the control input u. Assuming that the absolute value of this perturbation ξ is always smaller than a constant ξ_{max}, the perturbation ξ is called bounded matched perturbation. Probably the greatest advantage of sliding mode controllers is that this type of perturbations can entirely be compensated. The first step in order to derive a sliding mode controller is to define a sliding variable σ, which specifies a so-called switching plane. For the considered linear system it is reasonable to also use a linear combination of the states.

$$\sigma = m_1x_1 + m_2x_2$$

Then the dynamic of this sliding variable is specified in a discontinuous way. The following dynamic is often used for first order sliding mode controllers.

$$\dot{\sigma} = -\rho \,\mathrm{sign}\,(\sigma)\,, \qquad \text{with } \rho > 0 \tag{1.1}$$

Generally reaching the origin from an arbitrary initial condition can be separated into two phases. The first one is the reaching phase during which the sliding variable reaches zero in finite time, even though matched perturbations are present. This is ensured by the defined dynamic in (1.1) if ρ is appropriately chosen. Figure 1.1 shows the reaching phase for the considered system and several initial conditions in red. The sliding phase is the second one which starts after σ has reached 0 (blue lines in Figure 1.1). In this phase the dynamic order is reduced by one, which means that the considered second order system behaves in sliding phase like a first order system. This remaining dynamic is specified by the definition of the sliding variable σ (switching plane).

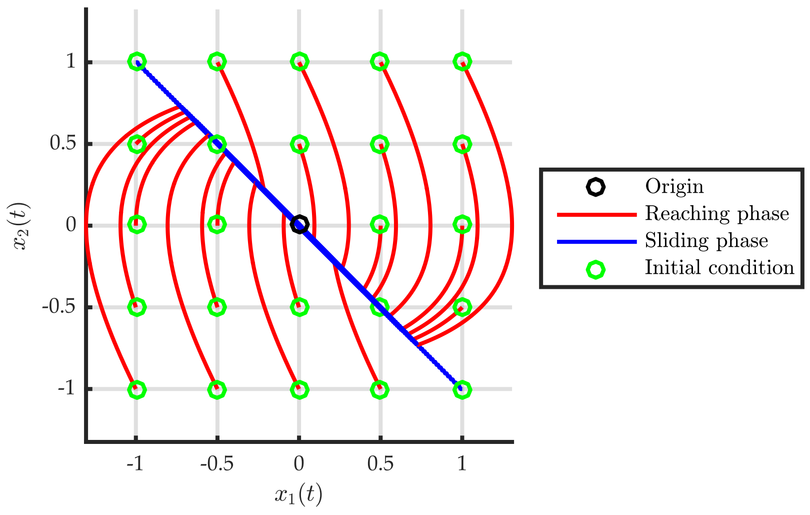


FIGURE 1.1: Reaching phase and sliding phase for second order system and different initial conditions

That the sliding variable σ reaches and is held at exactly $\sigma = 0$ is of course not possible in reality (except for special cases [14]) because it would be necessary that the controller switches infinitely often. In practical applications a finite sample time is always needed which leads to a zigzag motion of the sliding variable about the switching plane (see Figure 1.2) and also to a high frequency switching of the controller output (chattering).

This chattering is an undesired behaviour because this high frequency switching can damage actuators and could also be harmful to the system. Therefore, several chattering alleviation techniques are developed (see [13], [15]–[17])

The sliding mode control approach, which was explained in this section by using a second order system in normal form, is expandable to a variety of other system classes. There are approaches which deal with control systems with multiple inputs and multiple outputs, nonlinear systems and many more.

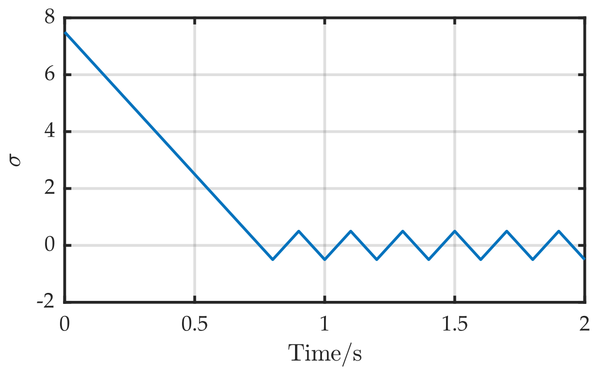


FIGURE 1.2: Zigzag motion of sliding variable σ

1.3 Networked Sliding Mode Control System

Combining the sliding mode control approach and the network structure results in a very powerful controller architecture which is called "Networked Sliding Mode Control". The networked sliding mode control architecture combines the advantages of both techniques. The ability to overcome large physical distances and the flexibility are examples of the networked control system architecture. The ability to completely compensate bounded matched perturbations and to reach the desired output in finite time are examples for the sliding mode control approach.

The following work will focus on this approach and in particular on analysing the influence of delays caused by the network structure. Most of these previously mentioned papers are devoted to methods in order to decrease the influence of this undesired phenomenon on the closed loop performance. This thesis is focused not on techniques to deal with it but rather on investigating their influence.

Chapter 2

Modelling a Networked Control Structure with Delays

2.1 Control Loop Architecture

The architecture of the considered control loop is shown in Figure 2.1. The plant consists of three blocks. The "ZOH Actuator" block represents the zero order hold in the actuator, which ensures that the output $u(t)$ is held constant until a new output command message is received over the network. The "Continuous Plant" block illustrates the continuous process of the following form.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{f}u(t)$$

with the state vector $\boldsymbol{x} \in \mathbb{R}^n$, the input $u \in \mathbb{R}$ and the matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{f} \in \mathbb{R}^n$ are of appropriate dimension. The sampling of the states $x(t)$ with the sample period h is performed in the "Sensor" block and results in the sampled states $\boldsymbol{x}_k$. It is assumed that the sensor is sampled periodically with period h and immediately sends the data. The static (time invariant) controller

$$u_k = f(\boldsymbol{x}_k), \tag{2.1}$$

in discrete representation is pictured by the "Discrete Controller" block. The controller is modelled as event driven. This means that the controller calculates the output whenever new data is available. All previous blocks are always present in a discretized control loop. The remaining two delay blocks are induced by using a network to close the control loop. The first delay τ_{sc} is caused by the network between the sensor and the controller. The second one τ_{ca} exists due to the network between the controller and the actuator.

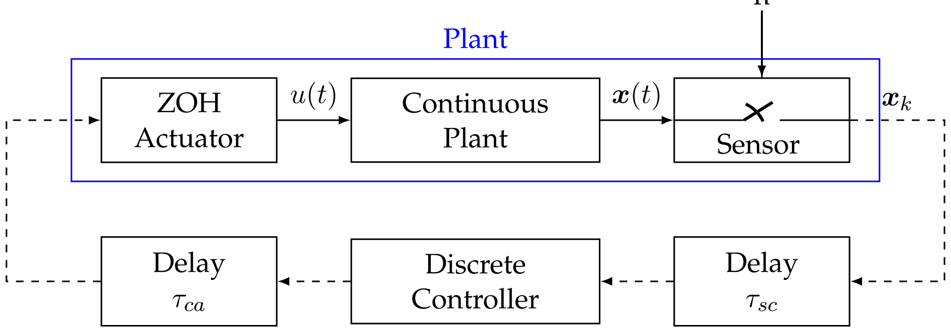


FIGURE 2.1: Architecture of the control loop

The timing for the sensor to actuator path is shown in Figure 2.2. This picture shows the periodically sampled states $\boldsymbol{x}_k$ in the timing diagram of the sensor. The sampled values are sent over the network, which induces the delay τ_{sc} and the controller calculates the output u_k based on these delayed states. The calculated control output is then sent to the actuator over the network, which induces the delay τ_{ca}. The actuator generates the output signal immediately after the controller message has been received. This picture was generated for constant delays τ_{sc} and τ_{ca}. Generally these delays are time variant ($\tau_{sc,k}$ and $\tau_{ca,k}$).

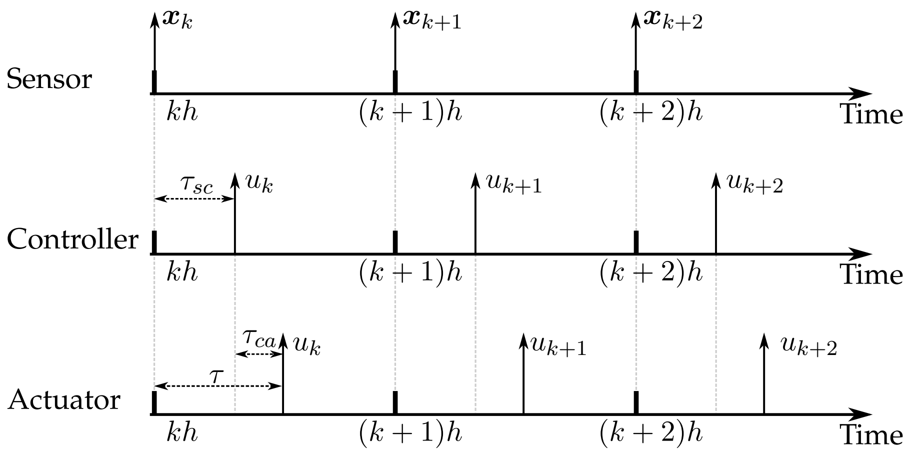


FIGURE 2.2: Timing from sensor to actuator path

As a static controller is used for the further investigations, it is possible to combine the two delays $\tau_{ca,k}$ and $\tau_{sc,k}$ to a summed delay τ_k without loss of generality.

$$\tau_k = \tau_{ca,k} + \tau_{sc,k} \tag{2.2}$$

Also, a delay caused by finite computational power of the controller could be considered in this single delay [18]. The dashed arrows represent the flow of discrete data and the solid ones the continuous data flow.

2.2 Small Delay Case

In this section it is assumed that the delay is smaller than one sampling period h.

$$\tau_k < h, \ \forall k \tag{2.3}$$

This ensures that only the two controller outputs u_{k-1} and u_k are applied to the system at the k-th sampling step. If the delay were larger, more delayed controller outputs would have to be considered. The mathematical model of the system can be written as follows

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{f}u^+ \qquad (2.4)\\ u^+ &= f(\boldsymbol{x}(t-\tau_k)), \quad t \in \{kh + \tau_k, k = 0, 1, 2, \dots\} \end{aligned}$$

where u^+ is a piecewise continuous function which changes value only at $kh + \tau_k$ [8]. The timing diagram in Figure 2.3 shows the function u^+ in blue and the sampled states $\boldsymbol{x}_k$ as vertical arrows. This figure shows that the delay τ_k causes two samples of the control output to be applied during every sampling step of the plant. By specifically having a look at the transition from the $\boldsymbol{x}_k$ to the $\boldsymbol{x}_{k+1}$ state in this figure, it is visible that the control sample u_{k-1} is applied for a duration of τ_k and the control sample u_k for the rest of the time.

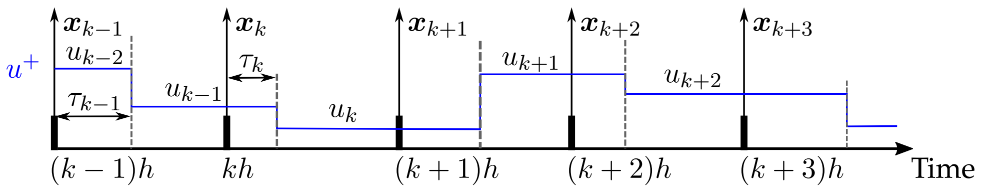


FIGURE 2.3: Timing diagram for discrete networked control system

These deliberations lead to the following sampled version of (2.4).

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}_0(\tau_k)u_k + \boldsymbol{\Gamma}_1(\tau_k)u_{k-1} \tag{2.5}$$

with

$$\begin{aligned} \boldsymbol{\Phi} &= e^{Ah} \\ \boldsymbol{\Gamma}_0(\tau_k) &= \int_0^{h-\tau_k} e^{As}\boldsymbol{f}\, ds \\ \boldsymbol{\Gamma}_1(\tau_k) &= \int_{h-\tau_k}^{h} e^{As}\boldsymbol{f}\, ds \end{aligned} \tag{2.6}$$

with $\boldsymbol{x}_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}$ and the matrices $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$, $\boldsymbol{\Gamma}_0(\tau_k),\ \boldsymbol{\Gamma}_1(\tau_k) \in \mathbb{R}^n$. The matrix $\boldsymbol{\Phi}$ equals the transition matrix $\boldsymbol{\Phi}(t)$ for $t = h$. There are several methods to calculate this matrix $\boldsymbol{\Phi}(t)$. One uses a transformation derived from the eigenvectors (if A is diagonalisable).

In (2.5) it is clearly visible that the actual (u_k) and the delayed by one sample (u_{k-1}) controller output have to be considered in the k-th sampling step. The weight vectors $\boldsymbol{\Gamma}_0(\tau_k)$ and $\boldsymbol{\Gamma}_1(\tau_k)$ are dependent on how long the corresponding controller output is applied. By setting the delay $\tau_k = 0,\ \forall k$ results in the classical sampled system representation.

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{b}u_k \tag{2.7}$$

$$\tag{2.8}$$

with

$$\begin{aligned} \boldsymbol{\Phi} &= e^{Ah} \\ \boldsymbol{b} = \boldsymbol{\Gamma}_0(0) &= \int_0^{h} e^{As}\boldsymbol{f}\, ds \\ \boldsymbol{\Gamma}_1(0) &= \boldsymbol{0} \quad . \end{aligned} \tag{2.9}$$

The system in (2.7) is further referenced as the nominal system. By combining (2.6) and (2.9) the following very important equation can be derived.

$$\boldsymbol{b} = \boldsymbol{\Gamma}_0(\tau_k) + \boldsymbol{\Gamma}_1(\tau_k) \tag{2.10}$$

2.2.1 Large Delay Case

The case, in which the constraint of limiting the delay to less than one sampling period h does not hold, can be separated into two cases. In the first case the limits of the time delay are known as

$$(m-1)h < \tau_k < mh \quad \text{with } m > 1. \tag{2.11}$$

The constant delay scenario is also covered by (2.11).

A timing diagram which shows this case for $m = 2$ is shown in Figure 2.4.

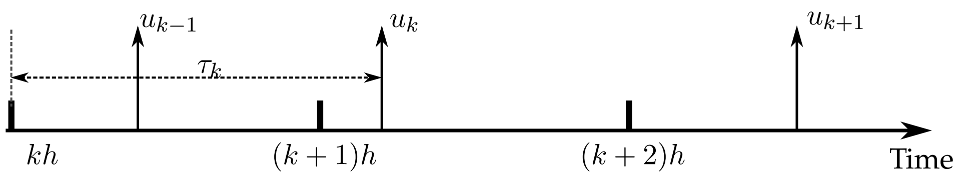


FIGURE 2.4: Timing diagram for delays $h < \tau_k < 2h$

This figure illustrates that only one control sample is received within every sampling period. This makes it possible to use the ideas from the small delay case because this case can be considered as a shifted small delay case if the boundaries of the delays are known as stated in (2.11)

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}_0(\tau_k')u_{k+1-\lceil m \rceil} + \boldsymbol{\Gamma}_1(\tau_k')u_{k-\lceil m \rceil}$$

with

$$\tau_k' = \tau_k - (m-1)h.$$

By using the augmented state vector

$$\tilde{\boldsymbol{z}}_k = \begin{bmatrix} \boldsymbol{x}_k \\ u_{k-\lceil m \rceil} \\ \vdots \\ u_{k-1} \end{bmatrix}$$

and using a linear state controller

$$u_k = -\boldsymbol{k}^T \boldsymbol{x}_k$$

stability analysis could be performed (see [8]).

Using this augmented state vector results in the following closed difference equations

$$\tilde{\boldsymbol{z}}_{k+1} = \underbrace{\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{\Gamma}_1(\tau_k') & \boldsymbol{\Gamma}_0(\tau_k') & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ -\boldsymbol{k}^T & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{\tilde{\boldsymbol{\Phi}}} \tilde{\boldsymbol{z}}_k$$

By checking the Schur-ness of the matrix $\tilde{\mathbf{\Phi}}$ (i.e. all eigenvalues have to be of magnitude less than one), the stability of the NCS can be determined for constant delays (i.e. $\tau_k = \tau \; \forall k$).

In the second and more general case the boundaries of the delay τ_k are known as

$$0 \leq \tau_k \leq mh \quad \text{with } m > 1.$$

For this setup, it is no longer sufficient to consider only two control samples. As the delay is time varying, it is possible that zero, one and up to $\lceil m \rceil$ control samples arrive in a single sampling period. A timing diagram which shows some possible scenarios for $m = 2$ is depicted in Figure 2.5.

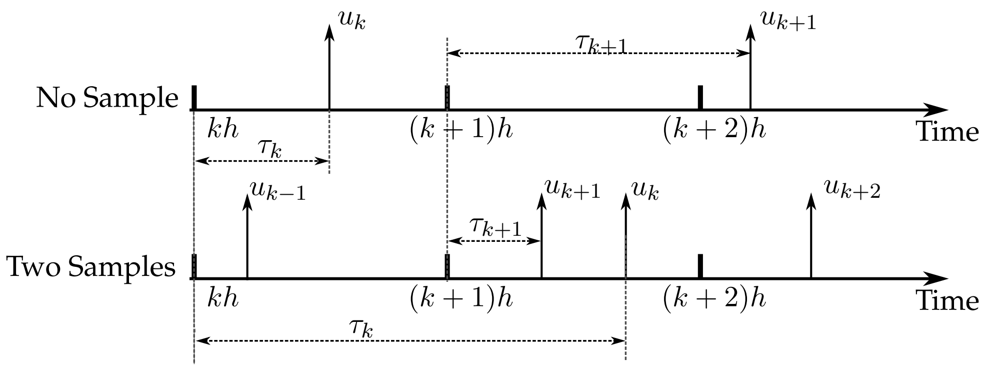


FIGURE 2.5: Timing diagram for delays $0 \leq \tau_k \leq 2h$

Two scenarios are shown in Figure 2.5. One where no control sample is received and the second one where two samples are received in one sampling period. This kind of delays are not that easy to handle, because the block matrix structure of $\tilde{\mathbf{\Phi}}$ is time varying. This phenomenon makes an analysis much more complex. So it is reasonable to only consider the small delay case for further investigations in order to gain a basic understanding.

2.3 Stability for NCSs using Linear State Controllers

In this section, an analysis is performed using a linear state controller. This analysis is reasonable in order to compare the results with other controller architectures. The state controller

$$u_k = -\boldsymbol{k}^T \boldsymbol{x}_k \tag{2.12}$$

is designed for the nominal system. Inserting this controller in the sampled NCS results in:

$$\boldsymbol{x}_{k+1} = (\mathbf{\Phi} - \mathbf{\Gamma}_0 \boldsymbol{k}^T)\boldsymbol{x}_k + \mathbf{\Gamma}_1 u_{k-1} \tag{2.13}$$

In (2.13) one can see that the linear controller vector $\boldsymbol{k}^T$ has to be designed in such a way that the matrix

$$\left(\mathbf{\Phi} - \boldsymbol{b}\boldsymbol{k}^T\right)$$

is Schur (i.e. all eigenvalues are of magnitude less than one).

For analysing purposes an augmented state vector $\boldsymbol{z}_k$ is needed which is defined as follows.

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{x}_k \\ u_{k-1} \end{bmatrix} \tag{2.14}$$

By using this state vector, the following difference equation can be constructed.

$$\begin{bmatrix} \boldsymbol{x}_{k+1} \\ u_k \end{bmatrix} = \boldsymbol{z}_{k+1} = \underbrace{\begin{bmatrix} \boldsymbol{\Phi} - \boldsymbol{\Gamma}_0(\tau_k)\boldsymbol{k}^T & \boldsymbol{\Gamma}_1(\tau_k) \\ -\boldsymbol{k}^T & 0 \end{bmatrix}}_{\tilde{\boldsymbol{\Phi}}} \boldsymbol{z}_k \tag{2.15}$$

If the delay τ_k is constant (i.e. $\tau_k = \tau,\ \forall k$), the weighting vectors $\boldsymbol{\Gamma}_0$ and $\boldsymbol{\Gamma}_1$ are also constant and the closed loop system in (2.15) is time invariant. To ensure asymptotic stability of the NCS with linear state controller and constant delays, the matrix $\tilde{\boldsymbol{\Phi}}$ has to be Schur.

With the matrix $\tilde{\boldsymbol{\Phi}}$, it is possible to check whether a stable closed loop could be expected. Alternatively, it is also possible to derive the region of stability by varying the values of the sampling period h and the time delay τ in percentage of the sampling period h.

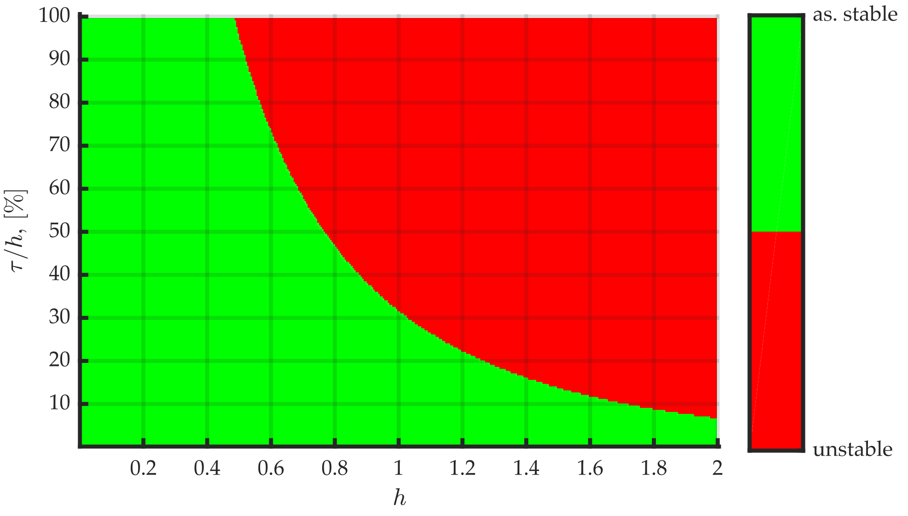


FIGURE 2.6: Stability region depending on sampling period h and the delay time τ in percentage of h

The result of such a simulation using the following first order continuous system is shown in Figure 2.6.

$$\dot{x} = ax + fu \quad \text{with } a = f = 1$$

For this basic system the transformation in discrete form depending on the sample time h can be written as follows.

$$x_{k+1} = e^{ah}x_k + \frac{f}{a}\left(e^{ah} - 1\right)u_k \tag{2.16}$$

The parameters were varied in the following intervals.

$$0.001 \leq h \leq 1.996 \quad \text{with resolution } r_h = 0.005$$
$$0.1\% \leq \frac{\tau}{h} \leq 99.6\% \quad \text{with resolution } r_\tau = 0.5\%$$

A discrete time controller of the form

$$u_k = -kx_k$$

was used. Inserting the control law into the difference equation (2.16) results in

$$x_{k+1} = \left(e^h - \left(e^h - 1\right) k\right) x_k$$

The closed loop dynamic is then specified in the continuous domain with $\lambda = -1$. With this specification the control parameter k is derived.

$$\left(e^h - \left(e^h - 1\right) k\right) \overset{!}{=} e^{\lambda h} \Rightarrow \boxed{k = \frac{e^h - e^{-h}}{e^h - 1}}$$

These settings and using the matrix $\tilde{\mathbf{\Phi}}$ to determine whether the closed loop is asymptotically stable or not results in the stability region shown in Figure 2.6. In this figure the green area specifies points where the closed loop is asymptotically stable (or $\tilde{\mathbf{\Phi}}$ is Schur) and the red region, where it is not asymptotically stable. In this plot, one can see that the control loop gets more sensitive to time delays with increasing sample period h. It is also visible that the boundary between the two regions is shaped exponentially. This means that a small increase of h can result in a large decrease of performance.

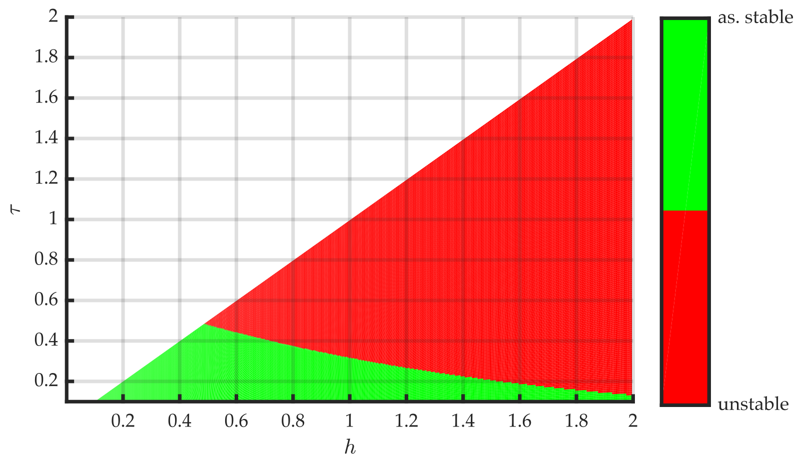


FIGURE 2.7: Stability region depending on sampling period h and the absolute delay time τ

Figure 2.7 shows the same stability region when absolute values for the delay time τ are used. The triangular shape is a result of the assumption that the time delay τ is always smaller than the sample time h. This figure depicts that the absolute value of the acceptable time delay also decreases with increasing stepsize.

Chapter 3

Designing a Discrete Sliding Mode Controller

There is a lot of literature dealing with continuous time sliding mode control [19]–[21]. It is important to consider that it is not possible to simply obtain the discrete version of the control law from the continuous one by means of simple equivalence [14].

The reaching condition for discrete time SMC, for example, cannot be derived easily from the continuous reaching condition. But the first attempt by Dote and Hoft consisted of exactly this. They used the continous reaching condition for discrete reaching condition [22].

$$[\sigma_{k+1} - \sigma_k]\,\sigma_k < 0 \tag{3.1}$$

That this condition is not sufficient for sliding mode was proved by Milosavljevic in [23]. Also, the concept of the quasi-sliding mode was introduced in this work. The next idea was elaborated by Sarpturk, Istefanopulos, and Kaynak in [24]. Where the following reaching condition was published

$$|\sigma_{k+1}| < |\sigma_k| \tag{3.2}$$

Also, deriving a reaching condition by using a Lyapunov-type of continuous reaching condition $\dot{v} < 0$ with $v = \frac{\sigma^2}{2}$ was used by Furuta in [25], which results in the following.

$$v_{k+1} - v_k < 0 \quad \text{with} \quad v_k = \frac{\sigma_k^2}{2}$$

All these reaching conditions are incomplete. So other techniques have to be investigated in order to ensure discrete time sliding mode.

3.1 Quasi Sliding Mode

The sliding mode for discrete time SMC systems is different than for continuous ones. In the continuous version the switching plane is reached exactly in finite time (reaching phase) and exactly follows the sliding surface towards the origin (sliding phase).

For discrete time SMC systems, the state response can also be separated in these two phases (reaching phase and sliding phase), but in general the behaviour is different. In [14] two types of trajectories were introduced for discrete time SMC systems. A type-I trajectory is the ideal one. An example for a second order system and a type-I trajectory is shown in figure 3.1.

This ideal trajectory appears only, if two ideal conditions are met. First, the sliding variable has to reach zero exactly at the switching time or, in other words, the states reach the switching plane exactly at the switching time. The second condition is that the natural dynamics of the plant match that of the ideal switching plane. It is easy to imagine that it is practically impossible to fulfil both conditions for a physical system. Even in simulation it is hard to achieve a type-I trajectory.

So the type-P trajectories, shown in figure 3.2, are much more interesting as this trajectory will appear in nearly all practical applications. In figure 3.2 the reaching phase looks quite the same as in figure 3.1 but the sliding phase differs. One can see that the trajectories zigzag about the ideal switching plane but never reach it exactly. As previously mentioned, type-P trajectories are much more interesting for practical use. The definition of quasi sliding mode is done for this trajectories. An interesting observation is that type-I trajectories are included in type-P ones as a special case.

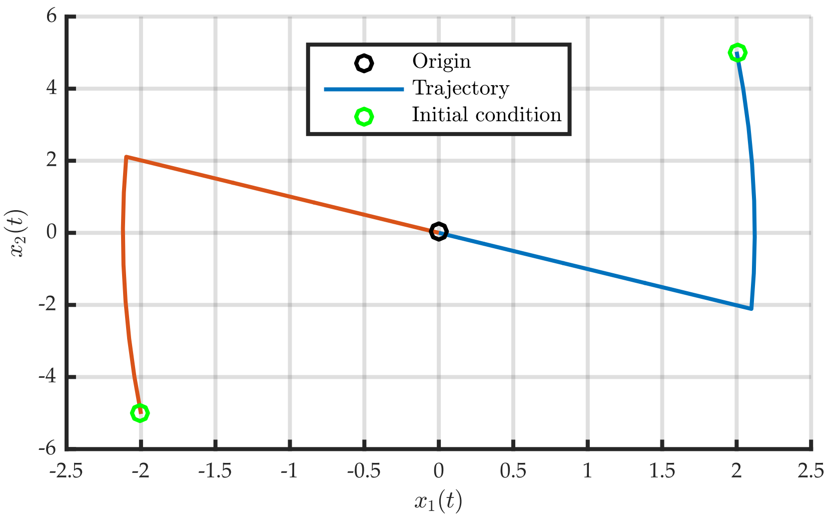


FIGURE 3.1: Type-I Trajectories of discrete SMC system

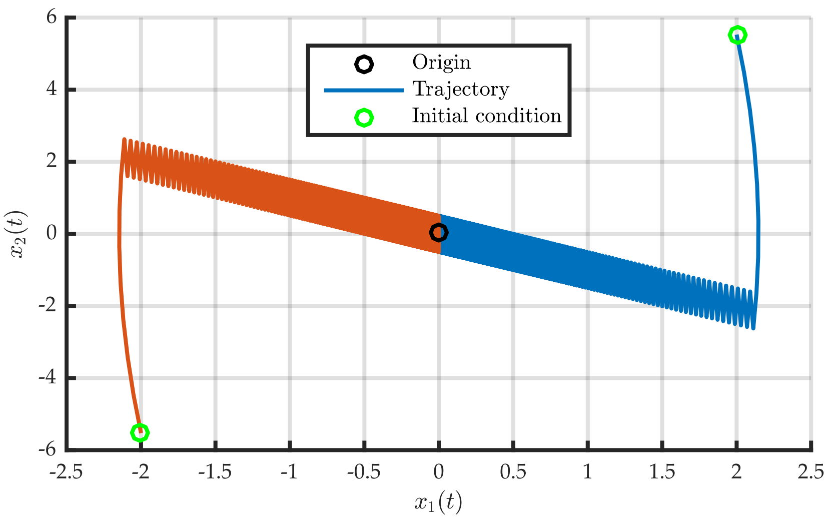


FIGURE 3.2: Type-P Trajectories of discrete SMC system

3.1.1 Attributes of Discrete Sliding Mode

The following three attributes are introduced by Gao, Wang, and Homaifa in [14]. They are all derived from the observations above.

1. Starting from any initial state, the trajectory will move monotonically towards the switching plane and cross it in finite time.
2. Once the trajectory has crossed the switching plane the first time, it will cross the plane again in every successive sampling period, resulting in a zigzag motion about the switching plane.
3. The size of each successive zigzagging step is nonincreasing and the trajectory stays within a specified band.

With this three attributes the reaching condition is formulated. A SMC system satisfies a reaching condition if all three attributes are met. A very important definition is "quasi sliding mode" (QSM). This names the motion of a discrete SMC system satisfying attributes 2 and 3. The band which contains the QSM is called the "quasi sliding mode band" (QSMB). If this band equals zero, the ideal quasi sliding mode results.

3.2 Reaching Law Approach

The classical way to construct the control law is to design it in such a way that the reaching condition is always fulfilled. An analytical expression for the reaching condition would be necessary to apply this approach. As previously seen, this analytical expression is not easy to establish. As a result an approach is needed, which could be applied without knowing a reaching condition. A renowned technique is the reaching law approach which has been proposed for continuous SMC systems [26], [27].

As the name of this method implies, the idea is to define the reaching law in such a way that all attributes are satisfied or, in other words, that the reaching condition is fulfilled. This reaching law specifies the dynamics of the sliding variable σ and the control law is then derived by using this reaching law. There are several reaching laws proposed in literature [28], [29] but the most commonly used one for continuous SMC systems is the following law.

$$\dot{\sigma} = -\rho \operatorname{sign}(\sigma) - q\sigma, \quad \text{with} \quad \rho > 0,\ q > 0$$

The equivalent form for discrete time SMC systems is

$$\begin{aligned} \sigma_{k+1} - \sigma_k &= -qh\sigma_k - \rho h \operatorname{sign}(\sigma_k) \quad \text{with} \quad \rho > 0,\ q > 0 \\ 1 - qh &> 0 \end{aligned} \tag{3.3}$$

where $h > 0$ denotes to the sampling period.

The inequality in (3.3) has to be fulfilled in order to ensure the first attribute (defined in section 3.1.1), which states that the sliding variable σ_k has to converge monotonically towards zero. The finite time convergence, which is also postulated in the first attribute, as well as the remaining two attributes are guaranteed by the presence of the $\operatorname{sign}(\cdot)$ part of the reaching law [14].

3.3 Designing the Discrete Control Law

Consider the following linear discrete time model of order n

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{b}u_k \tag{3.4}$$

$$\sigma_k = \boldsymbol{m}^T\boldsymbol{x}_k \tag{3.5}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$, σ_k and u_k are scalar and $\boldsymbol{\Phi}$, $\boldsymbol{b}$ and $\boldsymbol{m}^T$ are of appropriate dimensions. As visible in this model, the sliding variable σ_k is defined as a linear combination of the states $\boldsymbol{x}_k$. How this parameter $\boldsymbol{m}^T$ is calculated will be shown in the next section. The only important assumption at the moment is that $\boldsymbol{m}^T$ is calculated in such a way that a stable ideal quasi sliding mode results. As the reaching law should be used, the incremental change of the sliding variable is needed.

$$\begin{aligned}\sigma_{k+1} - \sigma_k &= \boldsymbol{m}^T\boldsymbol{x}_{k+1} - \boldsymbol{m}^T\boldsymbol{x}_k = \boldsymbol{m}^T(\boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{b}u_k) - \boldsymbol{m}^T\boldsymbol{x}_k \\ &= (\boldsymbol{m}^T\boldsymbol{\Phi} - \boldsymbol{m}^T)\boldsymbol{x}_k + \boldsymbol{m}^T\boldsymbol{b}u_k\end{aligned} \tag{3.6}$$

Setting (3.6) equal to the desired reaching law, i.e.,

$$(\boldsymbol{m}^T\boldsymbol{\Phi} - \boldsymbol{m}^T)\boldsymbol{x}_k + \boldsymbol{m}^T\boldsymbol{b}u_k \overset{!}{=} -qh\boldsymbol{m}^T\boldsymbol{x}_k - \rho h \operatorname{sign}\left(\boldsymbol{m}^T\boldsymbol{x}_k\right) \tag{3.7}$$

and solving for the control output u_k results in the control law.

$$\boxed{u_k = -\frac{1}{\boldsymbol{m}^T\boldsymbol{b}}\left[\rho h \operatorname{sign}\left(\boldsymbol{m}^T\boldsymbol{x}_k\right) + \boldsymbol{m}^T\Big(\boldsymbol{\Phi} - (1-qh)\boldsymbol{I}\Big)\boldsymbol{x}_k\right]} \tag{3.8}$$

with $\rho, q > 0$ and $1 - qh > 0$. An important assumption is that $\boldsymbol{m}^T\boldsymbol{b} \neq 0$. Otherwise the control input is not capable of influencing the sliding variable σ.

3.4 Designing the Sliding Surface

As previously mentioned, the sliding surface is specified by the vector $\boldsymbol{m}^T$. This vector has to be designed in such a way that the ideal quasi sliding mode is stable. This ideal quasi sliding mode is characterized by $\sigma_k = 0,\ \forall k > K$. To be able to calculate the vector $\boldsymbol{m}^T$, the system must first be transformed in such a way that the ideal quasi sliding mode is visible in the difference equations. The following normal form is often used for this purpose [20], [27].

$$\underbrace{\begin{bmatrix} \boldsymbol{z}_{1,k+1} \\ z_{2,k+1} \end{bmatrix}}_{\boldsymbol{T}\boldsymbol{x}_{k+1}} = \underbrace{\begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & A_{22} \end{bmatrix}}_{\boldsymbol{T}\boldsymbol{\Phi}\boldsymbol{T}^{-1}} \begin{bmatrix} \boldsymbol{z}_{1,k} \\ z_{2,k} \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{0} \\ b_1 \end{bmatrix}}_{\boldsymbol{T}\boldsymbol{b}} u_k \tag{3.9}$$

$$\sigma_k = \underbrace{\begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix}}_{\boldsymbol{m}^T\boldsymbol{T}^{-1}} \begin{bmatrix} \boldsymbol{z}_{1,k} \\ z_{2,k} \end{bmatrix}$$

In (3.9) it is important that $z_{2,k}$, A_{22} and b_1 are scalar. As hinted in (3.9), a linear regular transformation is performed by using the transformation matrix $\boldsymbol{T}$. The definition of the sliding variable σ_k is also very important. The sliding variable σ_k is defined as a linear combination of the $n-1$ state vector $\boldsymbol{z}_{1,k}$ with the added state $z_{2,k}$.

$$\sigma_k = \boldsymbol{m}^T \boldsymbol{z}_{1,k} + z_{2,k}$$

3.4.1 Transformation Matrix

The first big question is how to deduce the transformation matrix $\boldsymbol{T}$. The solution is found by adapting the QR-Factorization [30] algorithm. Consider the QR-Factorization of an arbitrary matrix

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R},$$

where $\boldsymbol{Q}$ is an orthogonal matrix (that means $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^T$), which simplifies some calculations later on and $\boldsymbol{R}$ is an upper right triangular matrix. This QR-Factorization is applied on the input vector $\boldsymbol{b}$. Then the matrix $\boldsymbol{R}$ changes to a vector. The same procedure could be used for multiple input and multiple output (MIMO) systems. In this case, both $\boldsymbol{b}$ and $\boldsymbol{R}$ are matrices. As $\boldsymbol{R}$ is a vector in our case, all entries are zero except of the first one.

$$\boldsymbol{b} = \boldsymbol{Q} \begin{bmatrix} b_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} b_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \boldsymbol{Q}^T \boldsymbol{b} \tag{3.10}$$

Comparing the transformed input vector $\boldsymbol{b}$ in (3.10) with the desired representation in (3.9) displays that the element b_1 should be in the last line. So a permutation matrix $\boldsymbol{E}$ is used to perform this reordering.

$$\tilde{\boldsymbol{R}} = \underbrace{\underbrace{\begin{bmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \iddots & \vdots \\ 1 & \cdots & 0 & 0 \end{bmatrix}}_{\boldsymbol{E}} \boldsymbol{Q}^T}_{\boldsymbol{T}} \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b_1 \end{bmatrix} \tag{3.11}$$

In (3.11) one can see that the permutation matrix is a flipped identity matrix. On the basis of previous deliberations the transformation matrix $\boldsymbol{T}$ has to be defined as follows.

$$\boxed{\boldsymbol{T} = \boldsymbol{E}\boldsymbol{Q}^T} \tag{3.12}$$

It is very convenient that the permutation matrix does not change the orthogonality of matrix $\boldsymbol{Q}$. As a consequence the transformation matrix $\boldsymbol{T}$ is also orthogonal, which saves computational time (especially for large systems), i.e.

$$\boxed{\boldsymbol{T}^{-1} = \boldsymbol{T}^T}$$

3.4.2 Placing the Poles

With the transformed system in (3.9) the ideal quasi sliding mode can be characterised by setting the sliding variable $\sigma_k = 0$. That this point is reached in finite time is guaranteed by the reaching law previously discussed. By setting the sliding variable to zero, the state $z_{2,k}$ can be expressed by a linear combination of the reduced state vector $\boldsymbol{z}_{1,k}$. In this step, the dynamic reduction by one (which is typically for first order SMC) is also explicitly visible.

$$0 = \sigma_k = \begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_{1,k} \\ z_{2,k} \end{bmatrix} \Rightarrow z_{2,k} = -\tilde{\boldsymbol{m}}^T \boldsymbol{z}_{1,k}$$

Applying this result to the state difference equation in (3.9) gives the following equation.

$$\boldsymbol{z}_{1,k+1} = \left[\boldsymbol{A}_{11} - \boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T\right] \boldsymbol{z}_{1,k} \tag{3.13}$$

Here the dynamic reduction is also visible because the whole dynamic in ideal quasi sliding mode phase is represented by $n-1$ states. Now the vector $\tilde{\boldsymbol{m}}^T$ can be calculated by specifying the ideal quasi sliding mode dynamic represented by (3.13). A very common approach is to specify desired poles of the matrix $\left[\boldsymbol{A}_{11} - \boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T\right]$ and then use an algorithm to calculate the vector $\tilde{\boldsymbol{m}}^T$. MATLAB offers two functions to perform this pole placement. The function `acker(·)` uses the Ackerman method and the function `place(·)` uses the algorithm described in [31].

The last step in order to calculate $\boldsymbol{m}^T$ is the following retransformation.

$$\boxed{\boldsymbol{m}^T = \begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \boldsymbol{T}}$$

3.4.3 Example

The following third order discrete time system is the starting point.

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 0 & 1.0 & 1.0 \\ -1.0 & 0 & 1.0 \\ 0 & -1.0 & 0 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 1.0 \\ -1.0 \\ -3.0 \end{bmatrix} u_k$$

Performing the QR-factorization by using the MATLAB command `qr(·)` results in

$$\underbrace{\begin{bmatrix} 1.0 \\ -1.0 \\ -3.0 \end{bmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{bmatrix} -0.302 & 0.302 & 0.905 \\ 0.302 & 0.93 & -0.21 \\ 0.905 & -0.21 & 0.371 \end{bmatrix}}_{\boldsymbol{Q}} \underbrace{\begin{bmatrix} -3.32 \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{R}}$$

With the result of the factorization, the transformation matrix $\boldsymbol{T}$ is calculated by evaluating

$$\boldsymbol{T} = \boldsymbol{E}\boldsymbol{Q}^T = \begin{bmatrix} 0.905 & -0.21 & 0.371 \\ 0.302 & 0.93 & -0.21 \\ -0.302 & 0.302 & 0.905 \end{bmatrix}$$

Applying this transformation matrix $\boldsymbol{T}$ gives the following transformed system.

$$\boldsymbol{z}_{k+1} = \underbrace{\begin{bmatrix} 0.336 & 0.413 & 0.726 \\ -0.491 & -0.0632 & 1.55 \\ -0.02 & -1.21 & -0.273 \end{bmatrix}}_{\tilde{\boldsymbol{\Phi}}} \boldsymbol{z}_k + \begin{bmatrix} 0 \\ 0 \\ -3.32 \end{bmatrix} u_k$$

The next step is to segment the matrix $\tilde{\boldsymbol{\Phi}}$ in the previously defined submatrices.

$$\boldsymbol{A}_{11} = \begin{bmatrix} 0.336 & 0.413 \\ -0.491 & -0.0632 \end{bmatrix} \qquad \boldsymbol{A}_{12} = \begin{bmatrix} 0.726 \\ 1.55 \end{bmatrix}$$

$$\boldsymbol{A}_{21} = \begin{bmatrix} -0.02 & -1.21 \end{bmatrix} \qquad \boldsymbol{A}_{22} = -0.273$$

The poles of the matrix $\left[\boldsymbol{A}_{11} - \boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T\right]$ were placed to $\lambda^T = \begin{bmatrix} 0.99 & 0.99 \end{bmatrix}$ by using the MATLAB command `acker(·)`, which returns the vector

$$\tilde{\boldsymbol{m}}^T = \begin{bmatrix} -0.153 & -1.03 \end{bmatrix}.$$

The last step to calculate the parameter vector $\boldsymbol{m}^T$ is to perform the necessary retransformation. So the calculated vector $\boldsymbol{m}^T$ can be stated as follows.

$$\boxed{\boldsymbol{m}^T = \begin{bmatrix} -0.751 & -0.625 & 1.06 \end{bmatrix}}$$

In order to verify the results, a simulation is performed using the example system and the calculated vector $\boldsymbol{m}^T$. The following parameters were chosen.

- Stepsize $h = 0.001$
- Linear reaching parameter $q = 1.0$
- Sliding mode gain $\rho = 100$
- Initial condition $\boldsymbol{x}_0 = \begin{bmatrix} -36.0 & -30.0 & 51.0 \end{bmatrix}^T$

The result of this simulation for the state vector $\boldsymbol{x}_k$ is shown in figure 3.3. As this plot does not explicitly show the reaching and the sliding phase, a transformation was performed by using the transformation matrix $\boldsymbol{T}$. This gives the state vector $\boldsymbol{z}_k$ which consists of $\boldsymbol{z}_{1,k}$ and $z_{2,k}$. By using the equation for σ_k the scalar state $z_{2,k}$ was replaced by σ_k. The new state vector was named $\boldsymbol{\xi}$.

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{z}_{1,k} \\ \sigma_k \end{bmatrix}$$

The evolution for this state vector is shown in figure 3.4. The horizontal red lines in figure 3.4 indicate the transition between the reaching phase and the sliding phase. The sliding variable σ_k is not zero during the reaching phase. This phase is on the left side of the red line. On the right side σ_k equals zero. So this is the sliding phase. One interesting observation is that σ_k does not reach zero in a linear way. As the sliding mode gain ρ equals 100 and the sliding variable at time instance zero is also 100, one second convergence time would be expected. Both observations can be explained

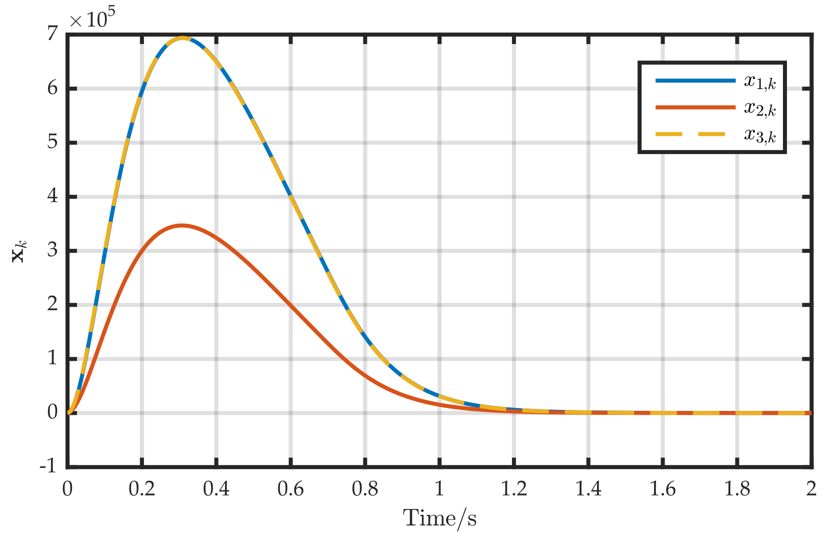


FIGURE 3.3: Simulation result for state vector $\boldsymbol{x}_k$

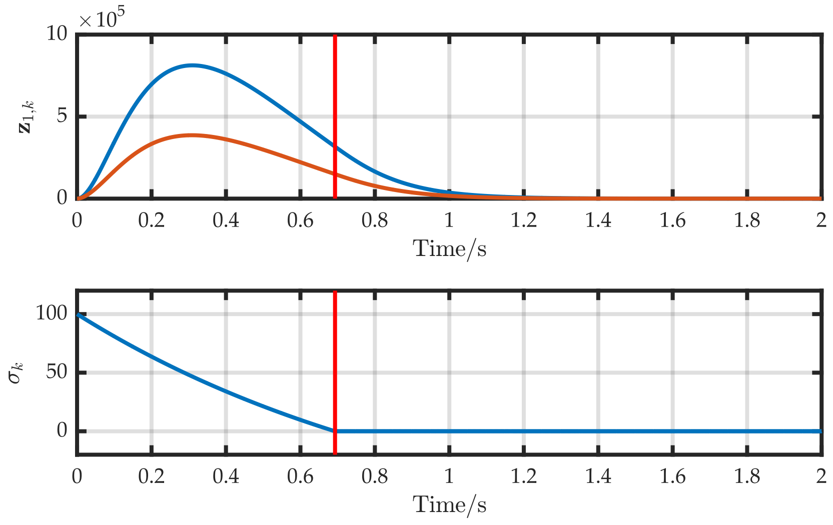


FIGURE 3.4: Simulation result for state vector $\boldsymbol{\xi}$

by the linear reaching parameter q. Increasing this parameter results in a faster convergence time for large values of σ_k and therefore also leads to a not linear convergence behaviour. It is very important that the sliding variable σ_k converges monotonically to zero, which means that no increase of the sliding variable is detectable at any time.

Inspecting the state trajectories for $z_{1,k}$ makes it visible that the monotonically converging to zero of these two states is not guaranteed during the reaching phase. One can see that these two states first grow and after approximately $0.3s$ they converge monotonically. However the specified behaviour is identifiable during sliding phase.

Chapter 4

Stability Analysis for Networked Sliding Mode Control Systems

In this chapter convergence criteria for networked sliding mode control systems are derived by using the results from chapters 2 and 3. Starting with the discrete time model of an networked control system from chapter 2

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}_0(\tau_k)u_k + \boldsymbol{\Gamma}_1(\tau_k)u_{k-1}$$

with

$$\begin{aligned} \boldsymbol{\Phi} &= e^{Ah} \\ \boldsymbol{\Gamma}_0(\tau_k) &= \int_0^{h-\tau_k} e^{As}\boldsymbol{f}\, ds \\ \boldsymbol{\Gamma}_1(\tau_k) &= \int_{h-\tau_k}^{h} e^{As}\boldsymbol{f}\, ds \end{aligned} \tag{4.1}$$

and combining it with the discrete time sliding mode controller (3.8)

$$u_k = -\frac{1}{\boldsymbol{m}^T\boldsymbol{b}}\left[\rho h \text{ sign}\left(\boldsymbol{m}^T\boldsymbol{x}_k\right) + \boldsymbol{m}^T\Big(\boldsymbol{\Phi} - (1-qh)\boldsymbol{I}\Big)\boldsymbol{x}_k\right]$$

with $\rho, q > 0$ and $1 - qh > 0$ results in the following equation

$$\begin{aligned} \boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k &- \frac{\boldsymbol{\Gamma}_0(\tau_k)}{\boldsymbol{m}^T\boldsymbol{b}}\left(\rho h \text{ sign}\left(\boldsymbol{m}^T\boldsymbol{x}_k\right) + \boldsymbol{m}^T(\boldsymbol{\Phi} - (1-ph)\boldsymbol{I})\boldsymbol{x}_k\right) && (4.2) \\ &- \frac{\boldsymbol{\Gamma}_1(\tau_k)}{\boldsymbol{m}^T\boldsymbol{b}}\left(\rho h \text{ sign}\left(\boldsymbol{m}^T\boldsymbol{x}_{k-1}\right) + \boldsymbol{m}^T(\boldsymbol{\Phi} - (1-ph)\boldsymbol{I})\boldsymbol{x}_{k-1}\right) && (4.3) \end{aligned}$$

Using the equation $\boldsymbol{b} = \boldsymbol{\Gamma}_0(\tau_k) + \boldsymbol{\Gamma}_1(\tau_k)$ resp. $\boldsymbol{\Gamma}_0(\tau_k) = \boldsymbol{b} - \boldsymbol{\Gamma}_1(\tau_k)$, (4.2) simplifies to

$$\begin{aligned} \boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k &- \frac{\boldsymbol{b}}{\boldsymbol{m}^T\boldsymbol{b}}\left(\rho h \text{ sign}\left(\boldsymbol{m}^T\boldsymbol{x}_k\right) + \boldsymbol{m}^T(\boldsymbol{\Phi} - (1-ph)\boldsymbol{I})\boldsymbol{x}_k\right) \\ &- \frac{\boldsymbol{\Gamma}_1(\tau_k)}{\boldsymbol{m}^T\boldsymbol{b}}\Big\{\rho h\left[\text{ sign}\left(\boldsymbol{m}^T\boldsymbol{x}_{k-1}\right) - \text{ sign}\left(\boldsymbol{m}^T\boldsymbol{x}_k\right)\right] \\ &+ \boldsymbol{m}^T\Big(\boldsymbol{\Phi} - (1-ph)\boldsymbol{I}\Big)(\boldsymbol{x}_{k-1} - \boldsymbol{x}_k)\Big\} \end{aligned} \tag{4.4}$$

4.1 State Transformation

As the states in (4.4) are not useful to show the remaining dynamic during sliding phase, a transformation is used to gain the following states

$$\boldsymbol{\xi}_k = \begin{bmatrix} \boldsymbol{z}_{1,k} \\ \sigma_k \end{bmatrix} = \boldsymbol{T}_2 \boldsymbol{x}_k \tag{4.5}$$

with $\boldsymbol{z}_{1,k}$ being the same $n-1$ state vector as in chapter 3 and σ_k representing the sliding variable.

$$\boldsymbol{z}_k = \boldsymbol{T}\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{z}_{1,k} \\ z_{2,k} \end{bmatrix} \qquad \text{and} \qquad \sigma_k = \begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \boldsymbol{z}_k$$

The transformation matrix $\boldsymbol{T}_2$ can be derived from these equations.

$$\boxed{\boldsymbol{T}_2 = \tilde{\boldsymbol{T}}\boldsymbol{T} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \boldsymbol{T}} \tag{4.6}$$

It is easy to show that the inverse of $\boldsymbol{T}_2$ is

$$\boxed{\boldsymbol{T}_2^{-1} = \boldsymbol{T}^T \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\tilde{\boldsymbol{m}}^T & 1 \end{bmatrix}} \quad .$$

In order to make it more readable, the transformation of (2.5) using (4.6) is performed piecewise and starting from the discrete time network model.

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{b}u_k + \boldsymbol{\Gamma}_1(\tau_k)(u_{k-1} - u_k)$$

Applying the transformation results in the following equation.

$$\boldsymbol{\xi}_{x+1} = \boldsymbol{T}_2\boldsymbol{x}_{k+1} = \underbrace{\boldsymbol{T}_2\boldsymbol{\Phi}\boldsymbol{T}_2^{-1}}_{\text{I}}\boldsymbol{\xi}_k + \underbrace{\boldsymbol{T}_2\boldsymbol{b}}_{\text{II}}\underbrace{u_k}_{\text{III}} + \underbrace{\boldsymbol{T}_2\boldsymbol{\Gamma}_1(\tau_k)}_{\text{IV}}(u_{k-1} - u_k)$$

The Roman numbers represent the individual parts for the piecewise transformation.

Transformed part I

The part I is defined as

$$\boldsymbol{T}_2\boldsymbol{\Phi}\boldsymbol{T}_2^{-1} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \boldsymbol{T}\boldsymbol{\Phi}\boldsymbol{T}^T \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\tilde{\boldsymbol{m}}^T & 1 \end{bmatrix}.$$

As the transformation matrix $\boldsymbol{T}$ equals (3.12), the same segmentation of the matrix $\boldsymbol{\Phi}$ can be performed

$$\boldsymbol{T}_2\boldsymbol{\Phi}\boldsymbol{T}_2^{-1} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\tilde{\boldsymbol{m}}^T & 1 \end{bmatrix}.$$

Performing the matrix multiplication generates the result

$$\boxed{\mathrm{I} = \begin{bmatrix} \boldsymbol{A}_{11} - \boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T & \boldsymbol{A}_{12} \\ \tilde{\boldsymbol{m}}^T\boldsymbol{A}_{11} + \boldsymbol{A}_{21} - \tilde{\boldsymbol{m}}^T\boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T - A_{22}\tilde{\boldsymbol{m}}^T & \tilde{\boldsymbol{m}}^T\boldsymbol{A}_{12} + A_{22} \end{bmatrix}} \tag{4.7}$$

for the transformed part I.

Transformed part II

Part II is defined as

$$\boldsymbol{T}_2\boldsymbol{b} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \boldsymbol{T}\boldsymbol{b}$$

Using transformation (3.9) results in

$$\boldsymbol{T}_2\boldsymbol{b} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{0} \\ b_1 \end{bmatrix}$$

The result of the matrix multiplication can be written as follows:

$$\boxed{\mathrm{II} = \begin{bmatrix} \boldsymbol{0} \\ b_1 \end{bmatrix}} \tag{4.8}$$

One can see that all elements except the last one are zero. This means that only the difference equation for σ_k is influenced by this part.

Transformed part III

The control law (3.8) was derived in chapter 3 as follows:

$$u_k = -\frac{1}{\boldsymbol{m}^T\boldsymbol{b}} \left[\rho h \operatorname{sign}\left(\boldsymbol{m}^T\boldsymbol{x}_k\right) + \boldsymbol{m}^T \Big(\boldsymbol{\Phi} - (1 - qh)\boldsymbol{I} \Big) \boldsymbol{x}_k \right] \tag{4.9}$$

It is reasonable to perform the small auxiliary calculation in order to simplify the further calculations. By using the transformation of the parameter vector $\boldsymbol{m}^T$

$$\boldsymbol{m}^T = \begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \boldsymbol{T}$$

the matrix product $\boldsymbol{m}^T\boldsymbol{b}$ simplifies to

$$\boldsymbol{m}^T\boldsymbol{b} = \begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \boldsymbol{T}\boldsymbol{b} = \begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{0} \\ b_1 \end{bmatrix} = b_1$$

Now the transformation is applied on the control law (4.9) resulting in

$$u_k = -\frac{1}{b_1} \left(\rho h \operatorname{sign}(\sigma_k) + \underbrace{\begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \boldsymbol{T}\boldsymbol{\Phi}\boldsymbol{T}_2^{-1}}_{\mathrm{III}.1} \boldsymbol{\xi}_k - \underbrace{\boldsymbol{m}^T(1 - qh)\boldsymbol{x}_k}_{(1-qh)\sigma_k} \right).$$

Transformation III.1 is accomplished separately to enhance readability.

$$
\begin{aligned}
\text{III}.1 &= \begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \underbrace{\boldsymbol{T\Phi T}^{-1}}_{\begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & A_{22} \end{bmatrix}} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \\
&= \begin{bmatrix} \tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\tilde{\boldsymbol{m}}^T & 1 \end{bmatrix} \\
&= \begin{bmatrix} \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{11} + \boldsymbol{A}_{21} - \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{12} \tilde{\boldsymbol{m}}^T - A_{22}\tilde{\boldsymbol{m}}^T & \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{12} + A_{22} \end{bmatrix} \qquad (4.10)
\end{aligned}
$$

Using the intermediate result (4.10) and including the term $(1-qh)\sigma_k$ in the vector representation leads to

$$\boxed{\text{III} = u_k = -\frac{1}{b_1}\left(\rho h \operatorname{sign}(\sigma_k) + \begin{bmatrix} \boldsymbol{d}_0^T & d_1 \end{bmatrix} \boldsymbol{\xi}_k\right)} \qquad (4.11)$$

with

$$
\begin{aligned}
\boldsymbol{d}_0^T &= \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{11} + \boldsymbol{A}_{21} - \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{12} \tilde{\boldsymbol{m}}^T - A_{22}\tilde{\boldsymbol{m}}^T \\
d_1 &= \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{12} + A_{22} - (1 - qh)
\end{aligned}
$$

Transformed part IV

This part was defined as

$$\boldsymbol{T}_2 \boldsymbol{\Gamma}_1(\tau_k).$$

The transformation is rather simple because no special attributes of this vector are known. Therefore the resulting vector

$$\boxed{\text{IV} = \begin{bmatrix} \boldsymbol{D}_0(\tau_k) \\ D_1(\tau_k) \end{bmatrix}} \qquad (4.12)$$

is split up to fit the structure defined by the state vector (4.5).

Putting the pieces together

By fitting (4.7), (4.8) and (4.12) together, the transformed system can be written as follows:

$$
\begin{aligned}
\boldsymbol{\xi}_{x+1} = &\begin{bmatrix} \boldsymbol{A}_{11} - \boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T & \boldsymbol{A}_{12} \\ \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{11} + \boldsymbol{A}_{21} - \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{12} \tilde{\boldsymbol{m}}^T - A_{22}\tilde{\boldsymbol{m}}^T & \tilde{\boldsymbol{m}}^T \boldsymbol{A}_{12} + A_{22} \end{bmatrix} \boldsymbol{\xi}_k \\
&+ \begin{bmatrix} \boldsymbol{0} \\ b_1 \end{bmatrix} u_k + \begin{bmatrix} \boldsymbol{D}_0(\tau_k) \\ D_1(\tau_k) \end{bmatrix} (u_{k-1} - u_k)
\end{aligned}
$$

Using the transformed control law (4.11) for u_k and u_{k-1} results in the following:

$$\begin{aligned}\boldsymbol{\xi}_{x+1} = &\begin{bmatrix} \boldsymbol{A}_{11} - \boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T + \frac{\boldsymbol{D}_0}{b_1}\boldsymbol{d}_0^T & \boldsymbol{A}_{12} + \frac{\boldsymbol{D}_0}{b_1}d_1 \\ \frac{D_1}{b_1}\boldsymbol{d}_0^T & 1 - qh + \frac{D_1}{b_1}d_1 \end{bmatrix}\boldsymbol{\xi}_k \\ &+ \begin{bmatrix} -\frac{\boldsymbol{D}_0}{b_1}\boldsymbol{d}_0^T & -\frac{\boldsymbol{D}_0}{b_1}d_1 \\ -\frac{D_1}{b_1}\boldsymbol{d}_0^T & -\frac{D_1}{b_1}d_1 \end{bmatrix}\boldsymbol{\xi}_{k-1} + \begin{bmatrix} -\frac{\boldsymbol{D}_0}{b_1} \\ 1 - \frac{D_1}{b_1} \end{bmatrix}(-\rho h \text{ sign}(\sigma_k)) \\ &+ \begin{bmatrix} \frac{\boldsymbol{D}_0}{b_1} \\ \frac{D_1}{b_1} \end{bmatrix}(-\rho h \text{ sign}(\sigma_{k-1}))\end{aligned} \tag{4.13}$$

An enlarged state vector

$$\tilde{\boldsymbol{\xi}}_k = \begin{bmatrix} \boldsymbol{\xi}_{k-1} \\ \boldsymbol{\xi}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}_{1,k-1} \\ \sigma_{k-1} \\ \boldsymbol{z}_{1,k} \\ \sigma_k \end{bmatrix}$$

is defined to get rid of $\boldsymbol{\xi}_{k-1}$ in (4.13).

The difference equation for $\tilde{\boldsymbol{\xi}}_k$ can be stated as follows.

$$\begin{aligned}\tilde{\boldsymbol{\xi}}_{k+1} = &\underbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & 0 & \boldsymbol{0} & 1 \\ -\frac{\boldsymbol{D}_0}{b_1}\boldsymbol{d}_0^T & -\frac{\boldsymbol{D}_0}{b_1}d_1 & \boldsymbol{A}_{11} - \boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T + \frac{\boldsymbol{D}_0}{b_1}\boldsymbol{d}_0^T & \boldsymbol{A}_{12} + \frac{\boldsymbol{D}_0}{b_1}d_1 \\ -\frac{D_1}{b_1}\boldsymbol{d}_0^T & -\frac{D_1}{b_1}d_1 & \frac{D_1}{b_1}\boldsymbol{d}_0^T & 1 - qh + \frac{D_1}{b_1}d_1 \end{bmatrix}}_{\hat{\boldsymbol{\Phi}}}\tilde{\boldsymbol{\xi}}_k \\ &+ \underbrace{\begin{bmatrix} \boldsymbol{0} \\ 0 \\ \boldsymbol{0} \\ 1 \end{bmatrix}}_{\boldsymbol{e}_{2n}}(-\rho h \text{ sign}(\sigma_k)) + \underbrace{\begin{bmatrix} \boldsymbol{0} \\ 0 \\ \frac{\boldsymbol{D}_0}{b_1} \\ \frac{D_1}{b_1} \end{bmatrix}}_{\hat{\boldsymbol{\Gamma}}}\Big(- \rho h\Big[\text{sign}(\sigma_{k-1}) - \text{ sign}(\sigma_k)\Big]\Big)\end{aligned} \tag{4.14}$$

with

$$\begin{aligned}\boldsymbol{T}\boldsymbol{\Phi}\boldsymbol{T}^T &= \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & A_{22} \end{bmatrix} \\ \boldsymbol{T}_2\boldsymbol{\Gamma}_1(\tau_k) &= \begin{bmatrix} \boldsymbol{D}_0 \\ D_1 \end{bmatrix} \\ \boldsymbol{T}\boldsymbol{b} &= \begin{bmatrix} \boldsymbol{0} \\ b_1 \end{bmatrix} \\ \boldsymbol{d}_0^T &= \tilde{\boldsymbol{m}}^T\boldsymbol{A}_{11} + \boldsymbol{A}_{21} - \tilde{\boldsymbol{m}}^T\boldsymbol{A}_{12}\tilde{\boldsymbol{m}}^T - \boldsymbol{A}_{22}\tilde{\boldsymbol{m}}^T \\ d_1 &= \tilde{\boldsymbol{m}}^T\boldsymbol{A}_{12} + A_{22} - (1 - ph)\end{aligned}$$

For sake of simplicity, the abbreviations are introduced in (4.14) to write the difference equations in a more compact way.

$$\boxed{\begin{aligned}\tilde{\boldsymbol{\xi}}_{k+1} &= \hat{\boldsymbol{\Phi}}(\tau_k)\tilde{\boldsymbol{\xi}}_k + \boldsymbol{e}_{2n}(-\rho h \operatorname{sign}(\sigma_k)) \\ &\quad + \hat{\boldsymbol{\Gamma}}(\tau_k)\Big(- \rho h\Big[\operatorname{sign}(\sigma_{k-1}) - \operatorname{sign}(\sigma_k)\Big]\Big)\end{aligned}} \tag{4.15}$$

Some observations can be made in (4.15)

- The matrix $\hat{\boldsymbol{\Phi}}(\tau_k)$ and the input vector $\hat{\boldsymbol{\Gamma}}(\tau_k)$ are time variant. A stability analysis can hardly be performed without any knowledge about the time delay τ_k .
- All inputs to the system are bounded because all inputs consist of the $\operatorname{sign}(\cdot)$ function which is bounded per definition.
- No asymptotical stability is possible because an infinitesimal small deviation from $\sigma_k = \sigma_{k-1} = 0$ causes the sliding variable at least to switch around the origin. In the worst case the solutions even diverge to infinity. Therefore, it is only possible to investigate the ultimate boundedness of the systems trajectories (see below).
- If the sign of σ_{k-1} equals the sign of σ_k, the $[\operatorname{sign}(\sigma_{k-1}) - \operatorname{sign}(\sigma_k)]$ part cancels out.

4.2 Verification of the Transformed Model

In order to accomplish a sanity check of the model in (4.15), a simulation is performed during which the trajectories of the model are compared with the ones by explicitly considering the delay. Figure 4.1 shows the used Simulink model for this verification step. This model actually consists of two separate ones. The first one, shown in front of a brown background, contains the model closest to reality. It is made up of the continuous plant followed by a sampler. The discrete time sliding mode controller receives the sampled states via the delay (orange block). The second one, shown in front of a green background, implements the model stated in (4.15). As this model returns transformed states, a retransformation is necessary in order to compare it with the other model. Consequently, the states recorded by the blue blocks should be equal.

The continuous model

$$\begin{aligned}\dot{\boldsymbol{x}} &= \begin{bmatrix} 0 & 1.0 \\ -1.0 & 1.0 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 1.0 \\ -1.0 \end{bmatrix} u \\ y &= \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u\end{aligned}$$

was used together with the following configuration to run the simulation.

- Stepsize: $h = 0.01s$
- Linear reaching parameter: $q = 0.8$
- Sliding mode gain: $\rho = 10$

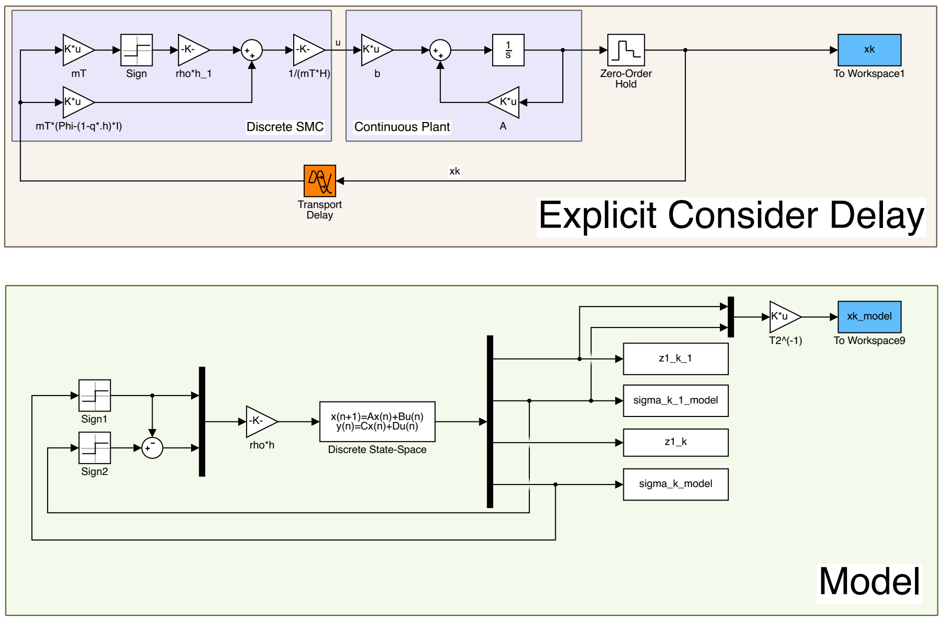


FIGURE 4.1: Simulink model to verify the mathematical model in (4.15)

- Discrete eigenvalue: $\lambda = 0.995$ which leads to $\boldsymbol{m}^T = \begin{bmatrix} -0.234 & 1.17 \end{bmatrix}$
- Time delay $\tau_k = 0.002s,\ \forall k$

The initial conditions were chosen for the brown model in such a way that a $\sigma_0 = 50.0$ results.

$$x_0 = \begin{bmatrix} -8.17 \\ 41.0 \end{bmatrix}$$

With this initial condition, one step is simulated in order to receive the two initial states needed for the green model. The initial condition of the green model is then derived by using the transformation matrix $\boldsymbol{T}_2$ as follows.

$$x_{0model} = \begin{bmatrix} 22.9 \\ 50.0 \\ 23.6 \\ 49.7 \end{bmatrix}$$

The result for both models is shown in figure 4.2. In this figure, the top plot shows the states

$$\boldsymbol{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix}$$

for the brown model as well as the sliding variable σ_k. The bottom plot shows the same quantities but received by simulating the green model. One can see that both plots are equal. As a visible comparison is not that accurate, the mean squared error of both states is computed as follows.

$$mse_x = \begin{bmatrix} 7.57 \cdot 10^{-9} & 2.47 \cdot 10^{-9} \end{bmatrix}$$

As these mean squared error values indicate, the result of both models are nearly equal.

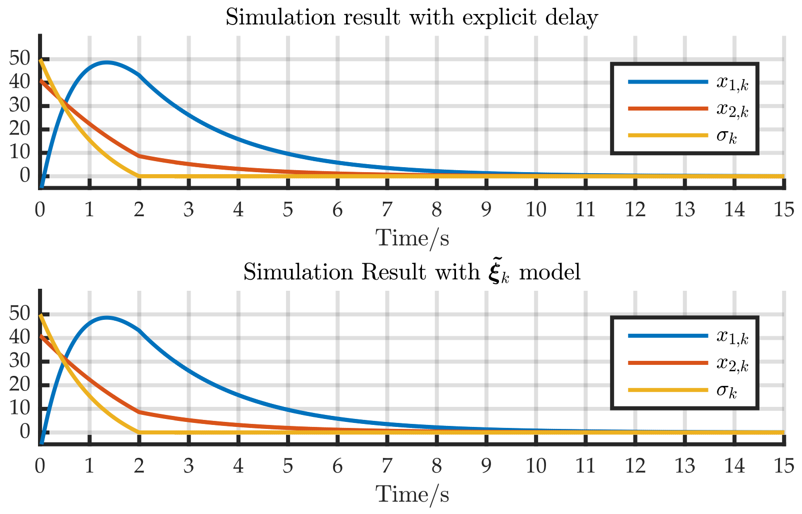


FIGURE 4.2: Verification result for states

4.3 Stability Analysis for Constant Delays

As previously mentioned, an analysis for asymptotical stability is not meaningful.

Definition: Ultimate boundedness

> The solutions of a system $\dot{\boldsymbol{x}} = f(t, \boldsymbol{x})$ are ultimately bounded if there exists a constant $B > 0$ such that every solution $\boldsymbol{x}(t)$ of the system satisfies the condition $\lim\limits_{t\to\infty} ||\boldsymbol{x}(t)|| < B$ [32]. In the further work, stability is defined as ultimate boundedness.

But checking the ultimate boundedness of the solutions of (4.15) is performed in this section. In general, an analysis of this time variant system without any knowledge of the delay time τ_k is very hard. So the assumption was made that the delay time is constant, i.e.

$$\boxed{\tau_k = \tau,\ \forall k}.$$

This assumption leads to a time invariant version of (4.15) which looks as follows:

$$\tilde{\boldsymbol{\xi}}_{k+1} = \hat{\boldsymbol{\Phi}}\tilde{\boldsymbol{\xi}}_k + \boldsymbol{e}_{2n}(-\rho h \operatorname{sign}(\sigma_k)) + \hat{\boldsymbol{\Gamma}}\left(-\rho h \left[\operatorname{sign}(\sigma_{k-1}) - \operatorname{sign}(\sigma_k)\right]\right) \quad (4.16)$$

The aim is to find conditions to ensure that the solutions of (4.16) are ultimately bounded. All inputs to system (4.16) are bounded ($\operatorname{sign}(\cdot)$). So the ultimate boundedness is ensured if the matrix $\hat{\boldsymbol{\Phi}}$ is Schur or, in other words, all eigenvalues of this matrix are of magnitude less than one. In the following sections, the stability regions for different plants are evaluated by using the conditions derived in section 4.3. The assumption that the time delay τ has to be smaller than one sampling period was made. Therefore, it was reasonable to define a relative time delay

$$\boxed{\tau_{rel} = \frac{\tau}{h}},$$

which can be varied in the interval

$$0 < \tau_{rel} < 1.$$

Also defining the linear reaching parameter q in a relative way

$$\boxed{q_{rel} = qh}$$

was performed in order to fulfil the inequality

$$1 - qh > 0$$

for all stepsizes h if the parameter q_{rel} is of the interval

$$0 < q_{rel} < 1$$

In order to make the choice of the eigenvalue λ_d, which defines the desired dynamic of the ideal sliding mode, independent of the sample time h it is reasonable to specify this eigenvalue in continuous domain. The needed discrete time eigenvalue λ_d can be derived from the specified continuous time eigenvalue λ_c as follows:

$$\lambda_d = e^{\lambda_c h}$$

4.4 Stability Regions' Dependence on Plant

Several simulations are performed for different plant configurations and the stability regions are compared.

4.4.1 Configuration

The used plant architecture was defined as follows

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{f}u \\ \boldsymbol{y} &= \boldsymbol{I}\boldsymbol{x} \end{aligned}$$

with $\boldsymbol{x} \in \mathbb{R}^n$, the identity matrix of $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ and a scalar input u. The remaining matrices are of appropriate dimensions. Furthermore, the following configuration was used.

- Order $n = 2$
- Uniformly distributed random values in the interval $\begin{bmatrix} -10 & 10 \end{bmatrix}$ for the entries in the system matrix A and the input vector b
- Stepsizes h values in the interval $\begin{bmatrix} 0.01 & 2 \end{bmatrix}$ with increment 0.01
- Relative delay $\tau_{rel} = \frac{\tau}{h}$ in the interval $\begin{bmatrix} 0.0 & 0.99 \end{bmatrix}$ with increment 0.01

- Relative linear reaching parameter $q_{rel} = qh = 0.5$
- Continuous ideal sliding mode eigenvalue $\lambda_c = -1$

After analysing the stability areas of several random systems, it turned out that the shape of the ultimately bounded area is dependant on the eigenvalue configuration of the plant. So it is reasonable to introduce three types of plants.

Type 1) The real part of all eigenvalues is negative

Type 1a) Real eigenvalues

Type 1b) Conjugate complex eigenvalues

Type 2) The real part of all eigenvalues is greater than zero

Type 3) The real part of one eigenvalue is negative, the other is greater than zero

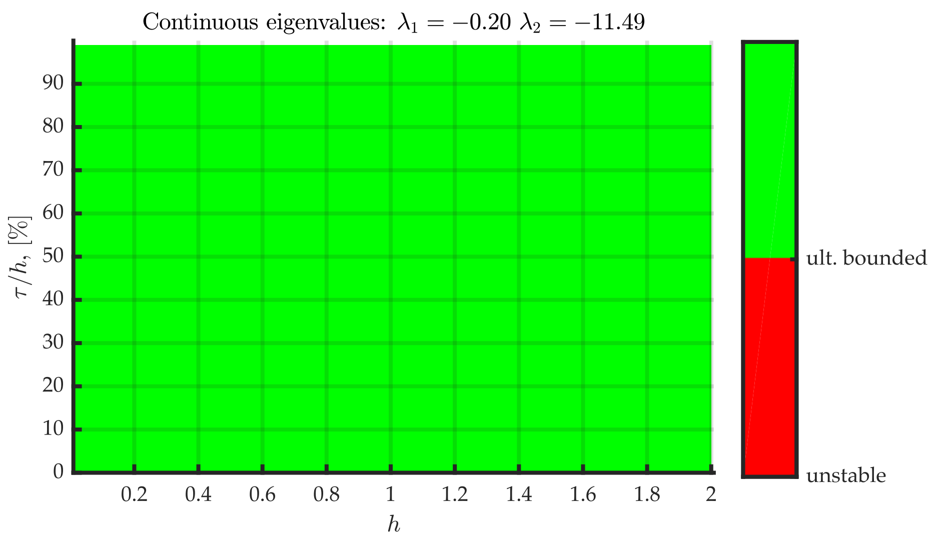
4.4.2 Results

Some simulation results for Type 1a plants are shown in figure 4.3. The plots show the area where the solution is ultimately bounded in green and the area where the solution grows to infinity in red. Additionally, the two eigenvalues of the plant are written on top of each plot. In these figures, it is visible that the control loop does not behave unstable for every configuration (at least for the considered stepsizes h). If there is an unstable area, it was always C-shaped (as illustrated in 4.3c, 4.3d, 4.3e and 4.3f). This C-shaped area shows the interesting phenomenon that with some stepsizes h, increasing the delay τ improves the performance. This is due to the observation that the closed loop system behaves unstable for smaller delays but with increasing delay ultimate boundedness is achieved.

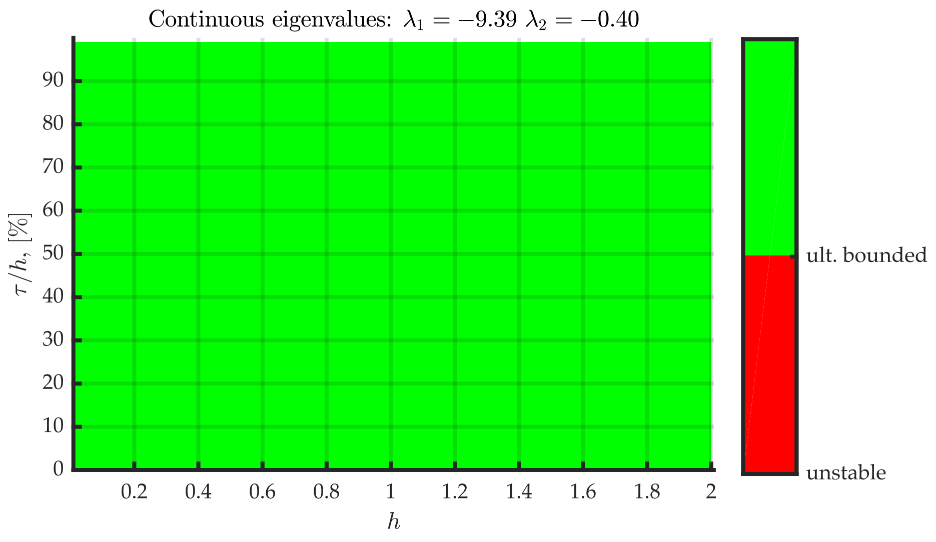
Figure 4.4 shows stability areas for Type 1b systems. The shapes of the unstable area seem to be arbitrary. Also, the phenomenon with performance enhancement with increasing delay τ for certain stepsizes h is visible in these plots. An interesting new observation can be made especially by analysing 4.4c. The performance of the closed loop system in terms of ultimate boundedness could be increased for some systems by increasing the sample time h. This was not the case for Type 1a systems.

Generally, the ultimately bounded area is much smaller when using Type 2 than when using Type 1 plants (see figure 4.5 for simulation results). The most occurring boundary between the ultimately bounded area and the unstable area was like an exponentially decreasing function as visible in figures 4.5a, 4.5b and 4.5c. Especially if the plant has conjugate complex eigenvalues, some spikes can appear (e.g. in figures 4.5d, 4.5e and 4.5f). The phenomenon of increasing performance with higher delays τ was not observed for this type of plant. However, it is also possible here that for certain stepsizes h the performance is better when choosing a higher value for h. This can be seen in figures 4.5d, 4.5e and 4.5f.

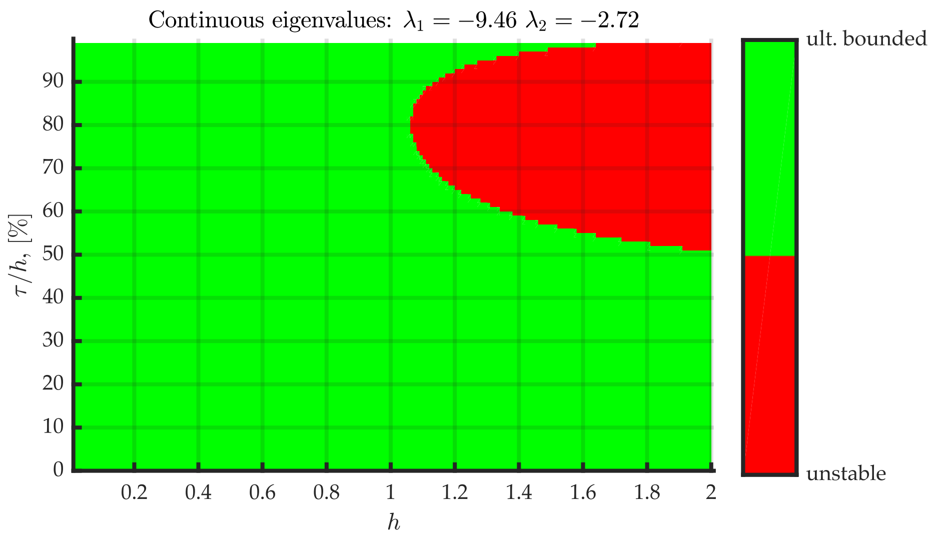
Stability areas for Type 3 plants are shown in figure 4.6. The shape of the stable areas looks the same as when using Type 2 plants except for the


Continuous eigenvalues: $\lambda_1 = -0.20$ $\lambda_2 = -11.49$
τ/h, [%]
h
ult. bounded
unstable

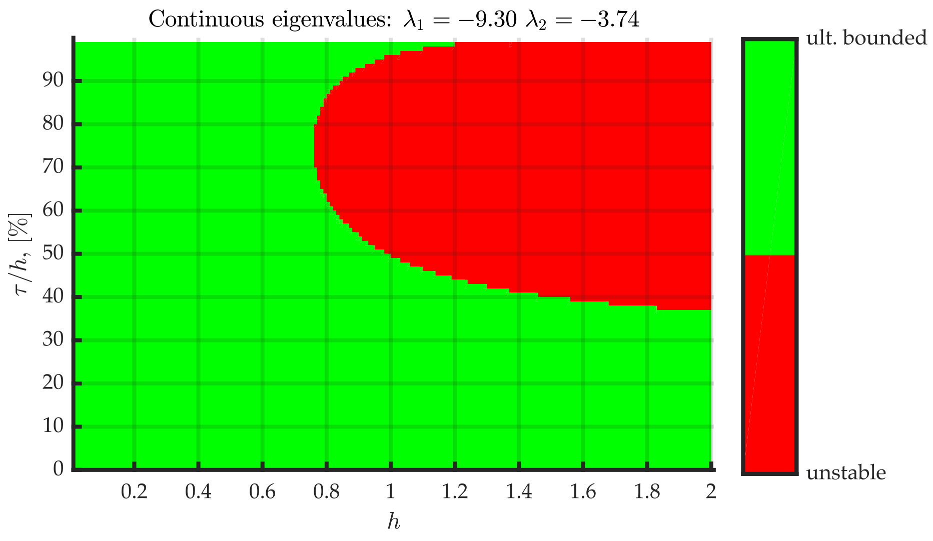
(A)


Continuous eigenvalues: $\lambda_1 = -9.39$ $\lambda_2 = -0.40$
τ/h, [%]
h
ult. bounded
unstable

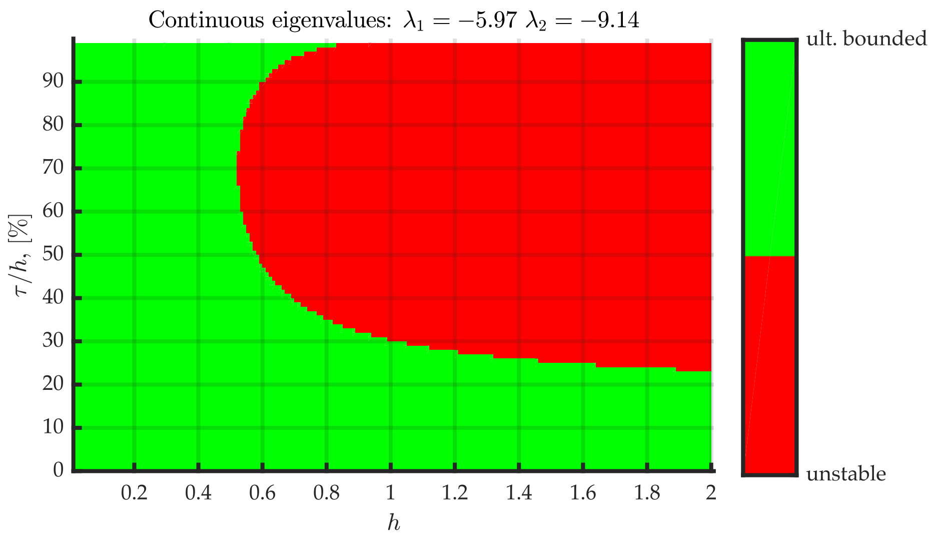
(B)


Continuous eigenvalues: $\lambda_1 = -9.46$ $\lambda_2 = -2.72$
τ/h, [%]
h
ult. bounded
unstable

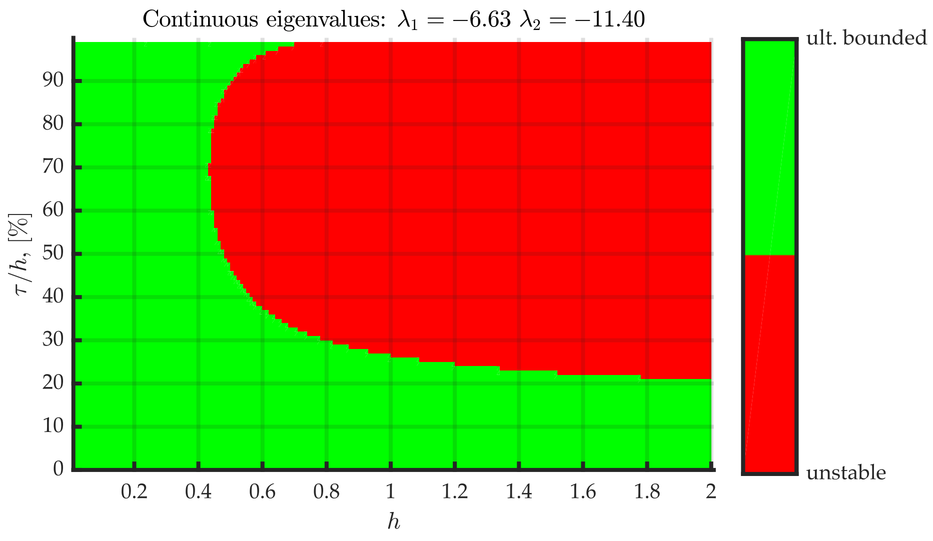
(C)



(D)

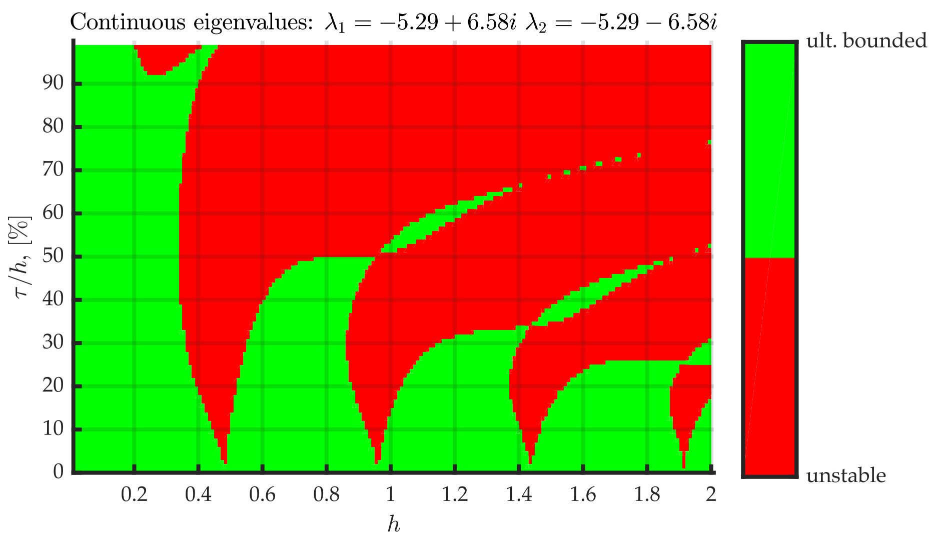


(E)

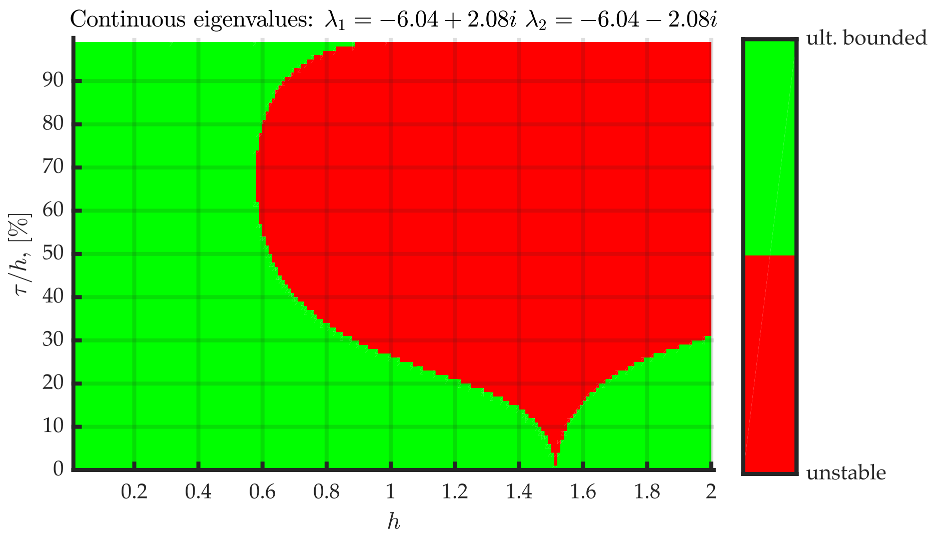


(F)

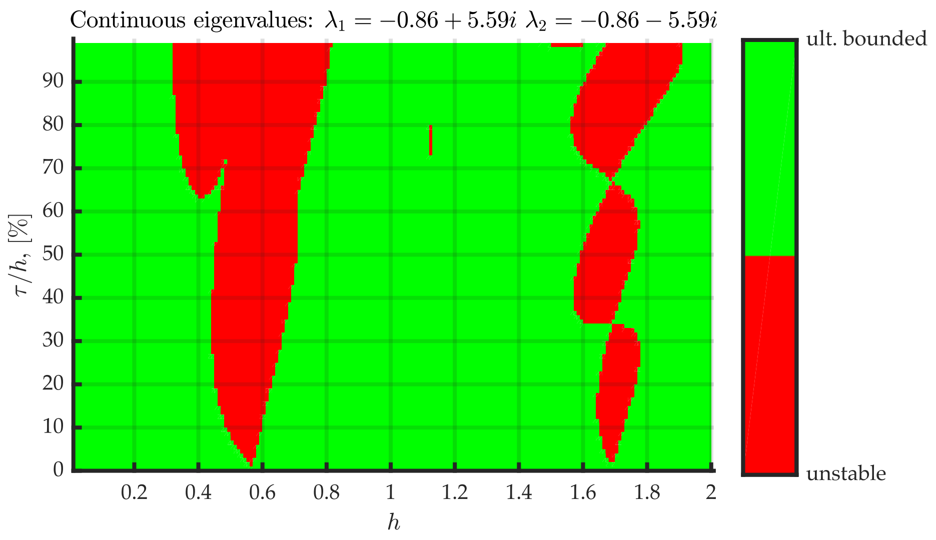
FIGURE 4.3: Stability areas for different plant configurations of Type 1a


Continuous eigenvalues: $\lambda_1 = -5.29 + 6.58i$ $\lambda_2 = -5.29 - 6.58i$
ult. bounded
unstable
τ/h, [%]
h
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

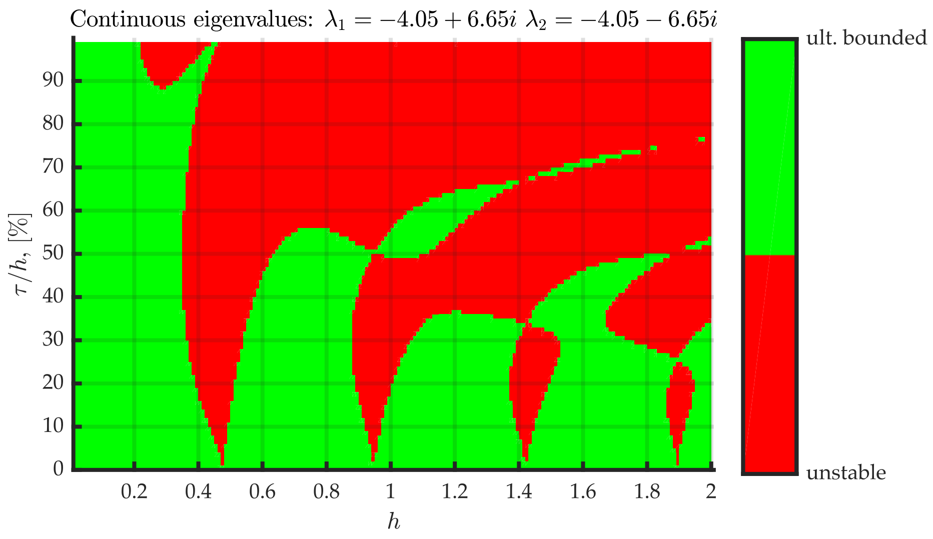
(A)


Continuous eigenvalues: $\lambda_1 = -6.04 + 2.08i$ $\lambda_2 = -6.04 - 2.08i$
ult. bounded
unstable
τ/h, [%]
h

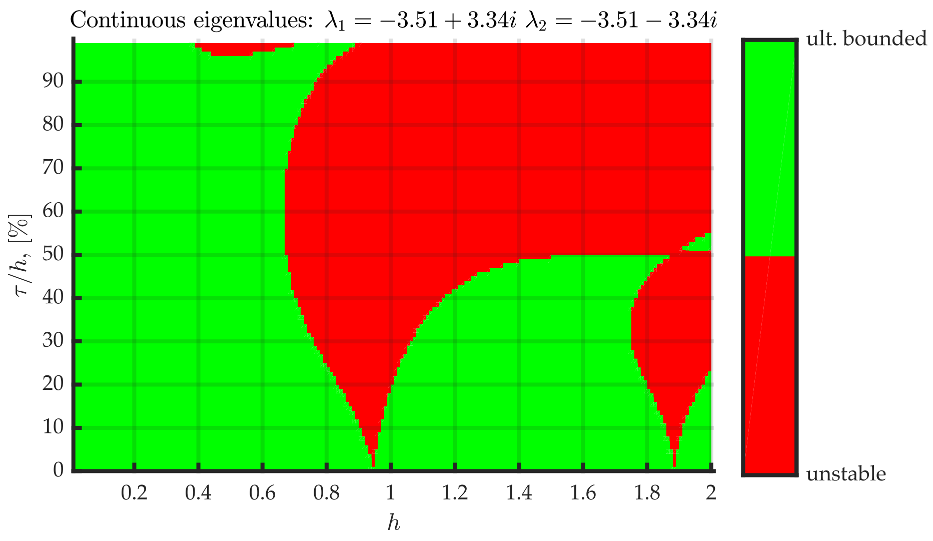
(B)


Continuous eigenvalues: $\lambda_1 = -0.86 + 5.59i$ $\lambda_2 = -0.86 - 5.59i$
ult. bounded
unstable
τ/h, [%]
h

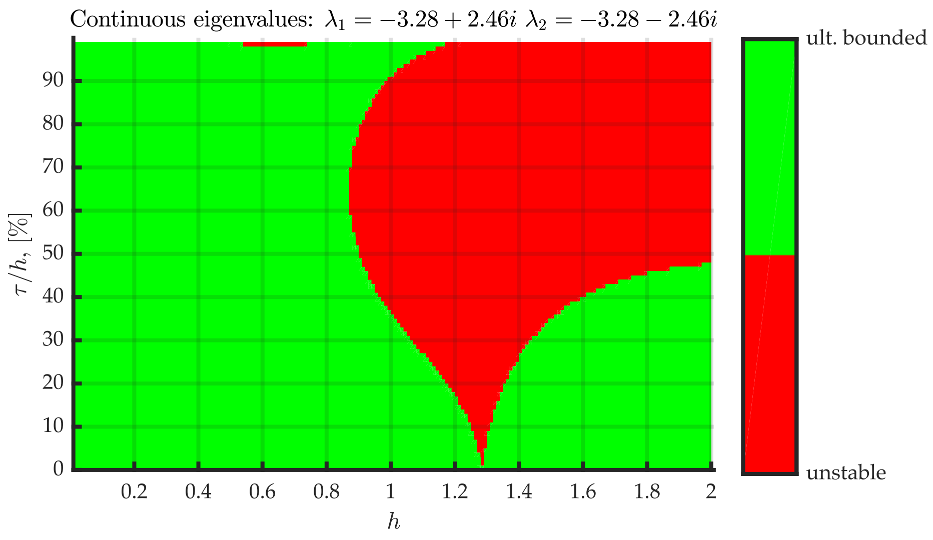
(C)



(D)

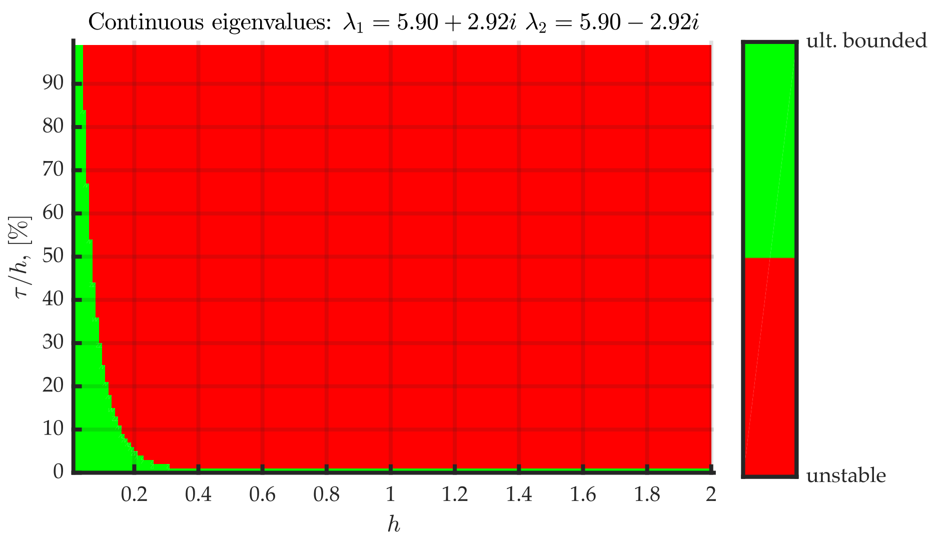


(E)

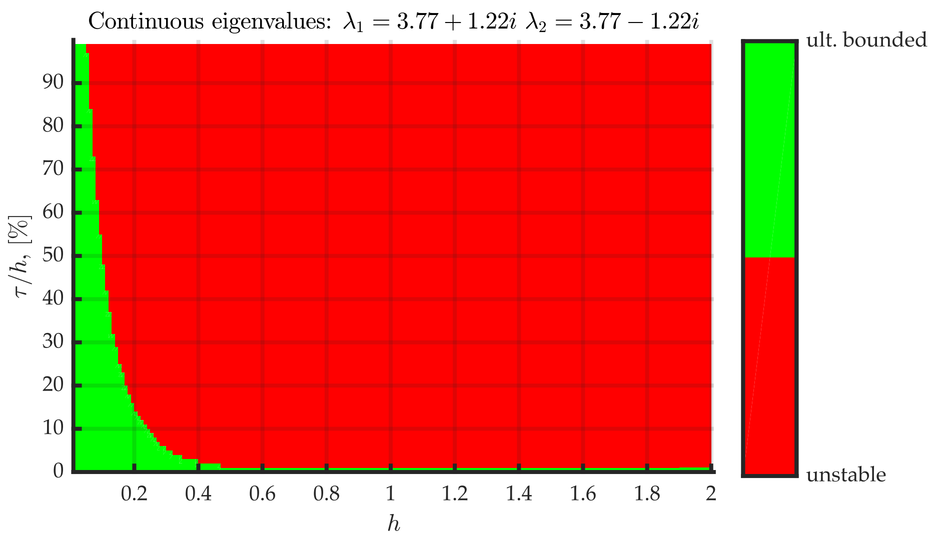


(F)

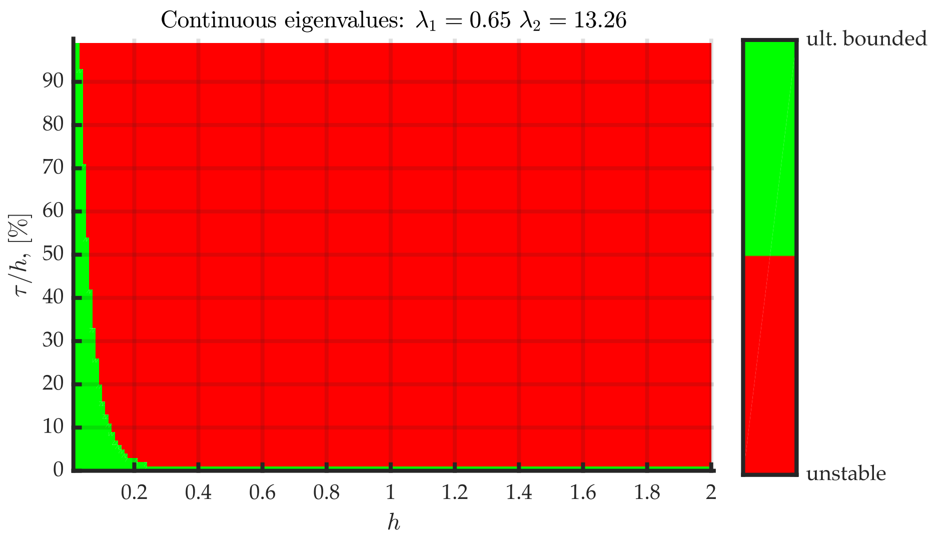
FIGURE 4.4: Stability areas for different plant configurations of Type 1b


Continuous eigenvalues: $\lambda_1 = 5.90 + 2.92i$ $\lambda_2 = 5.90 - 2.92i$
ult. bounded
unstable
τ/h, [%]
h

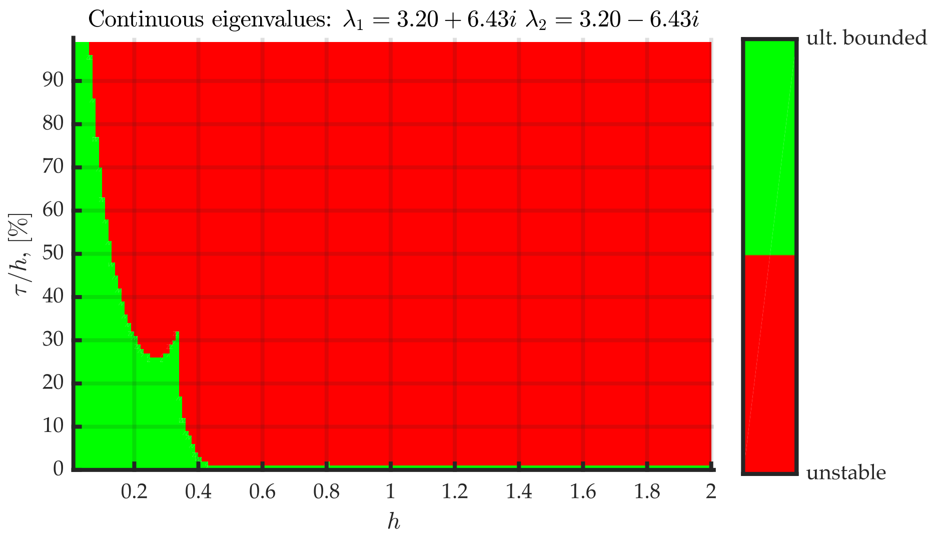
(A)


Continuous eigenvalues: $\lambda_1 = 3.77 + 1.22i$ $\lambda_2 = 3.77 - 1.22i$
ult. bounded
unstable
τ/h, [%]
h

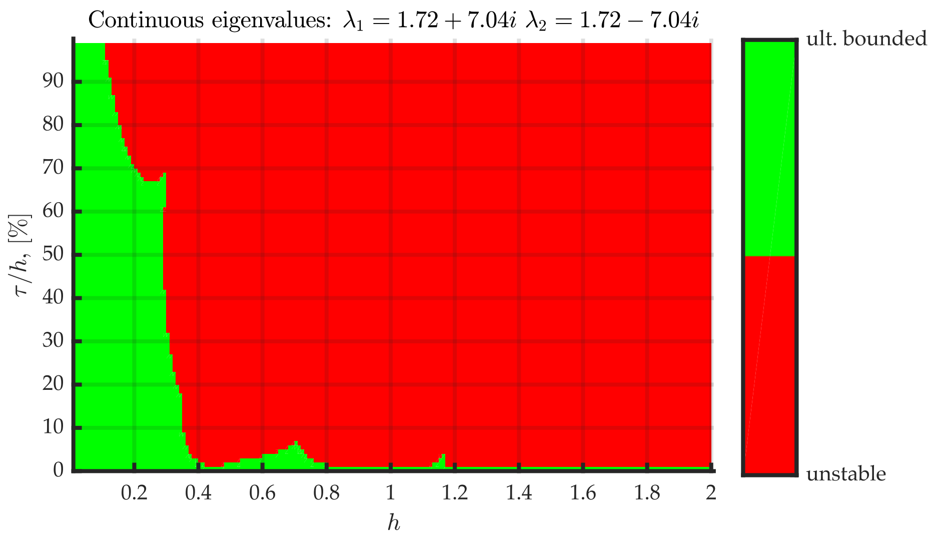
(B)


Continuous eigenvalues: $\lambda_1 = 0.65$ $\lambda_2 = 13.26$
ult. bounded
unstable
τ/h, [%]
h

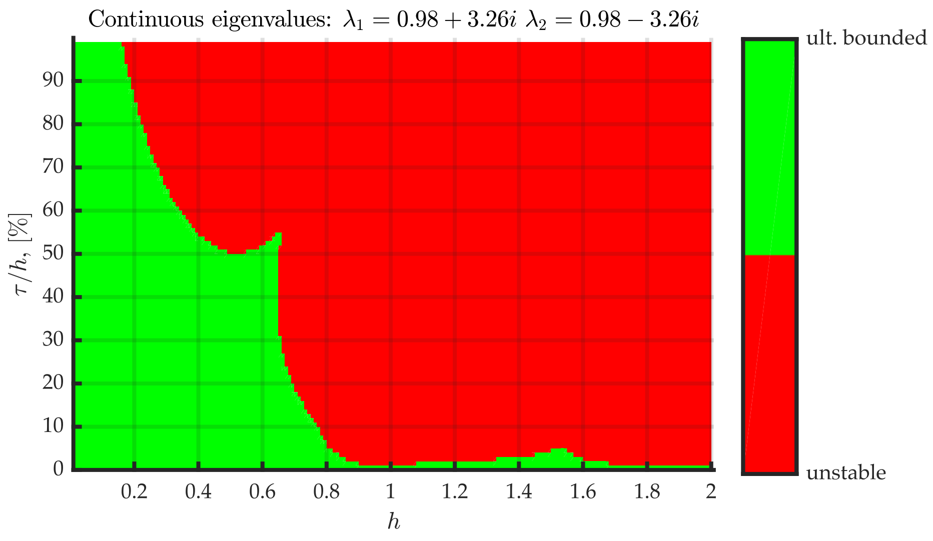
(C)



(D)



(E)



(F)

FIGURE 4.5: Stability areas for different plant configurations of Type 2

spikes. No spikes where observed when using Type 3 plants. This leads to the assumption that higher delays τ as well as higher sample times h always lead to worse performance in terms of ultimate boundedness.

4.4.3 Conclusions

By comparing the stability regions for the different types of plants, it is clearly visible that the largest areas with ultimately bounded solutions are achieved by using Type 1 plants. This was expected because controlling a stable plant is generally less problematic. Furthermore, it was investigated that for Type 1 and Type 2 systems there are cases in which a larger delay τ leads to better performance. For Type 2 systems, there are scenarios in which growing the sample time h also improves performance. When using Type 3 systems no such phenomena are investigated. Increasing the stepsize h always leads to a worse performance. Also, a higher time delay τ always leads to worse performance.

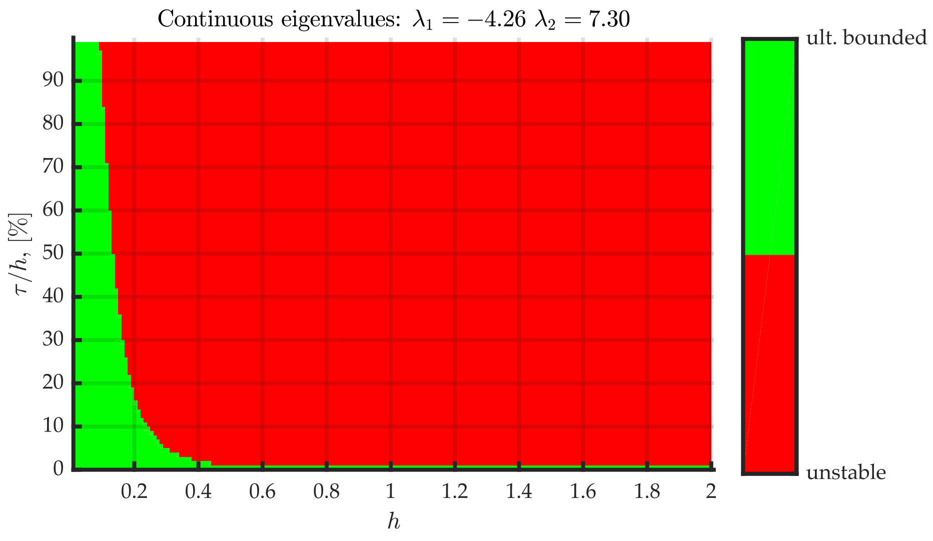
4.5 Stability Regions' Dependence on Linear Reaching Parameter q

In this section, the influence of the linear reaching parameter q is investigated. This is achieved by simulating some stability regions while increasing the parameter q_{rel} and of course keeping the plant system. In order to gain a more founded knowledge about the impact of the parameter q_{rel}, not only one simulation (with one plant) is performed but several with randomly constructed systems.

4.5.1 Configuration

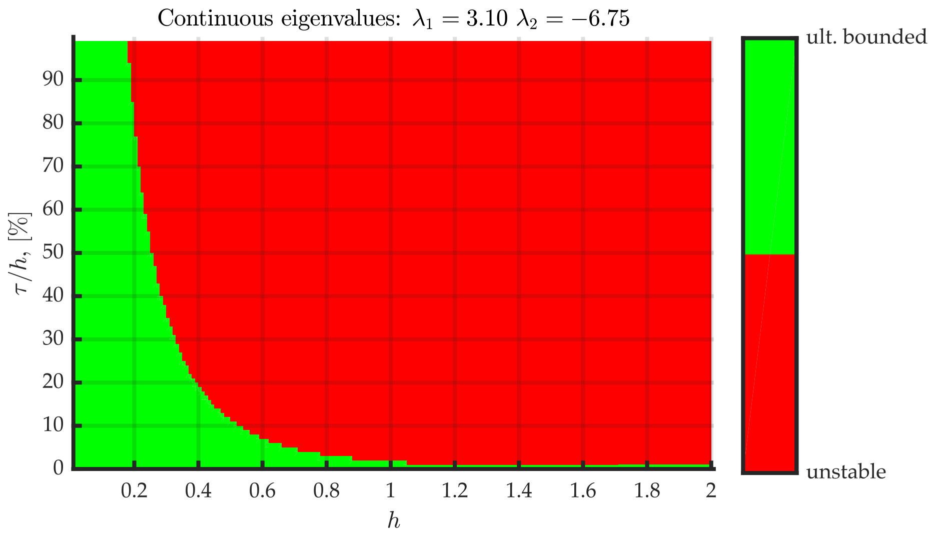
As mentioned, the systems are again designed randomly as described in section 4.4.1. It is very important to keep in mind that one simulation (with varying parameter q_{rel}) is always performed with one single plant. The following configuration was used to retrieve the results in section 4.5.2.

- Order $n = 2$
- Uniformly distributed random values in the interval $\begin{bmatrix} -10, & 10 \end{bmatrix}$ for the entries in the system matrix A and the input vector b
- Stepsizes h values in the interval $\begin{bmatrix} 0.01, & 2 \end{bmatrix}$ with increment 0.01
- Relative delay $\tau_{rel} = \frac{\tau}{h}$ in the interval $\begin{bmatrix} 0.0, & 0.99 \end{bmatrix}$ with increment 0.01
- Ten values linearly spaced in the interval $\begin{bmatrix} 0.001, & 0.99 \end{bmatrix}$ for the relative linear reaching parameter q_{rel}
- Continuous ideal sliding mode eigenvalue $\lambda_c = -1$



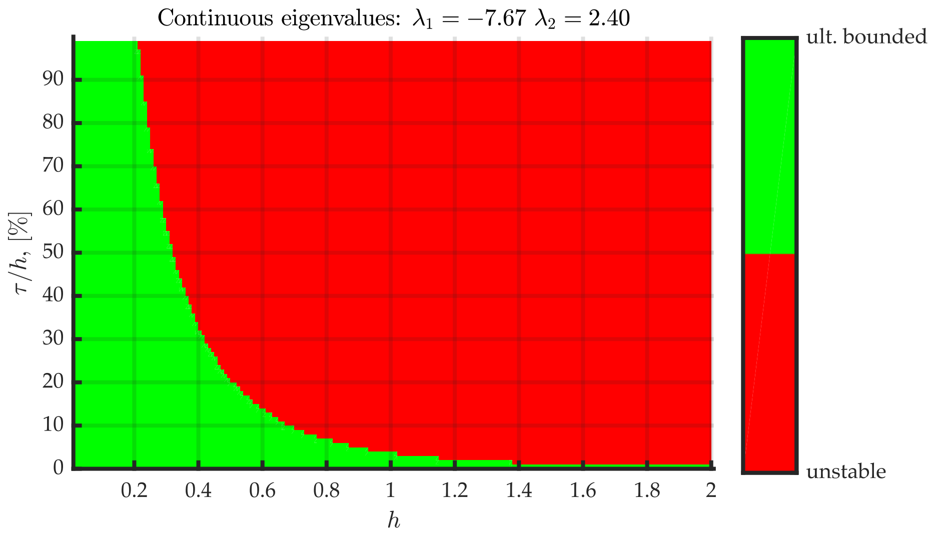
Continuous eigenvalues: $\lambda_1 = -4.26$ $\lambda_2 = 7.30$
τ/h, [%]
h
ult. bounded
unstable

(A)



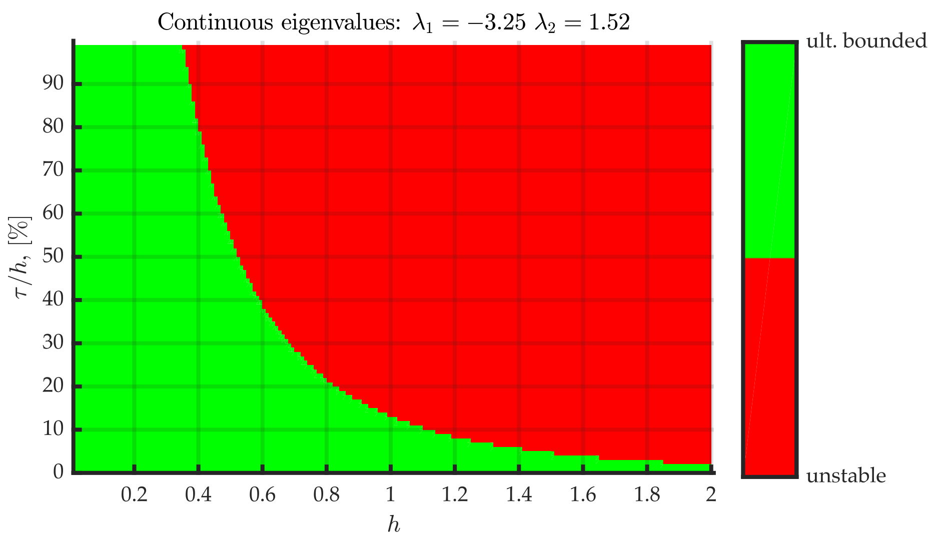
Continuous eigenvalues: $\lambda_1 = 3.10$ $\lambda_2 = -6.75$
τ/h, [%]
h
ult. bounded
unstable

(B)

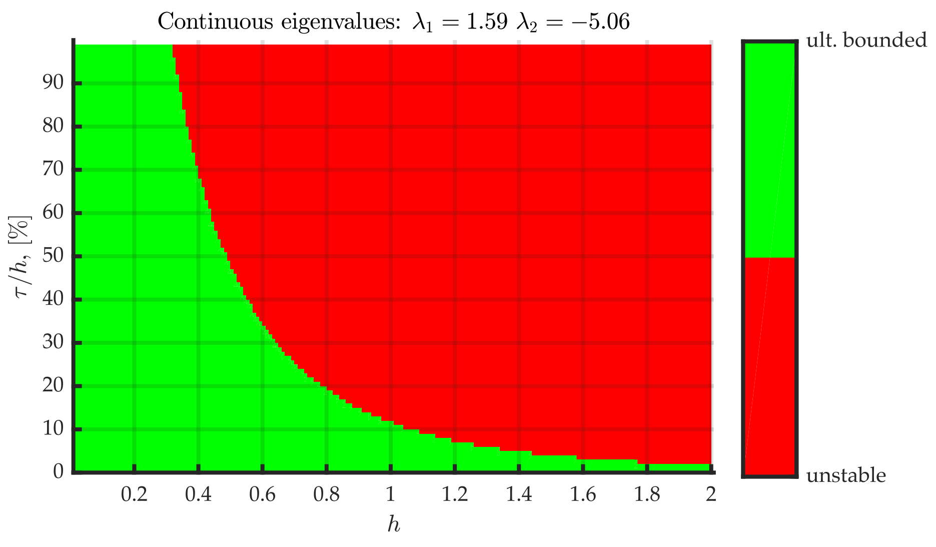


Continuous eigenvalues: $\lambda_1 = -7.67$ $\lambda_2 = 2.40$
τ/h, [%]
h
ult. bounded
unstable

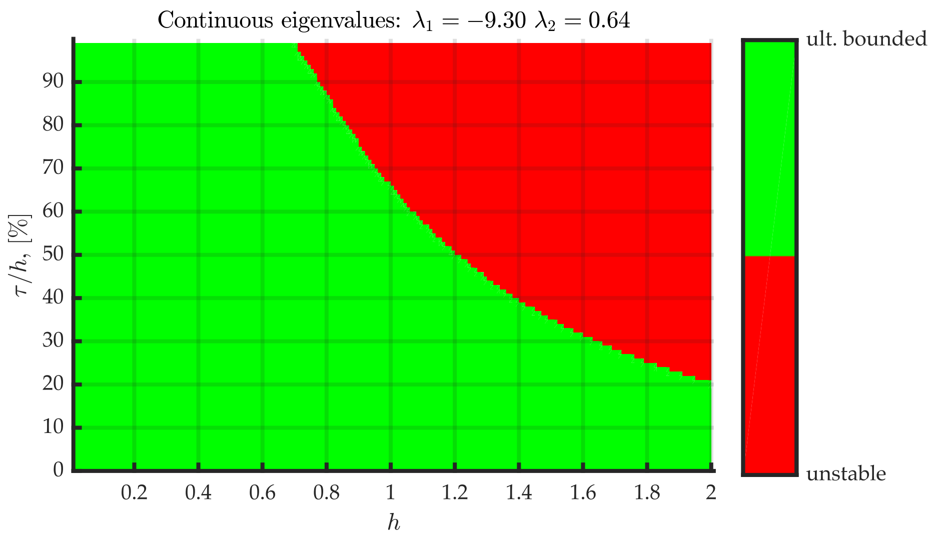
(C)



(D)



(E)



(F)

FIGURE 4.6: Stability areas for different plant configurations of Type 3

4.5.2 Results

Performing some simulations indicated that it is again reasonable to use the three Types defined in section 4.4.1 to categorize the plant systems. The stability regions with different values of the relative linear reaching parameter q_{rel} of the following Type 1a plant are shown in figure 4.7.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -9.45 & 2.2 \\ 7.52 & -5.93 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0.398 \\ -8.92 \end{bmatrix} u \tag{4.17}$$

The eigenvalues of the system matrix are

$$\lambda = \begin{bmatrix} -12.1 \\ -3.26 \end{bmatrix}$$

Figure 4.7 shows the stability region for growing values of q_{rel}. The plot with the smallest value is 4.7a and the one with the largest is 4.7d. It is easily visible that larger values of the parameter q_{rel} enhance the performance of the closed loop system because the unstable area becomes smaller with increasing q_{rel}.

As one single simulation is not very representative, several of this simulations with different plants of Type 1a are performed. In order to make a comparison of these simulations, the percentage of stable areas to the whole area is calculated and plotted for each simulation run. The result is shown in figure 4.8. The legend shows the eigenvalues of the plants system matrices. Each line shows how the percantage of stable area evolves over the parameter q_{rel}. It is interesting that all lines are monotonically growing which means that increasing the parameter q_{rel} increases the area of ultimate boundedness.

The stability regions for the Type 1b system

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -0.208 & 9.79 \\ -4.6 & -6.33 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 7.23 \\ -9.35 \end{bmatrix} u$$

with eigenvalues

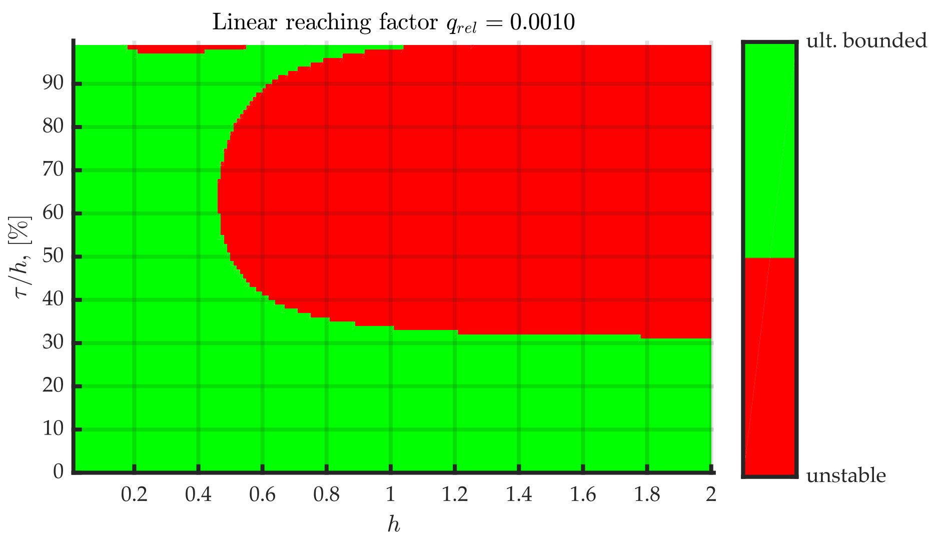
$$\lambda = \begin{bmatrix} -3.27 + 5.98\,\mathrm{i} \\ -3.27 - 5.98\,\mathrm{i} \end{bmatrix}$$

are shown in figure 4.9. This figure shows again that increasing the relative linear reaching parameter q_{rel} results in a larger stable area, which was also the case with the Type 1a plants.

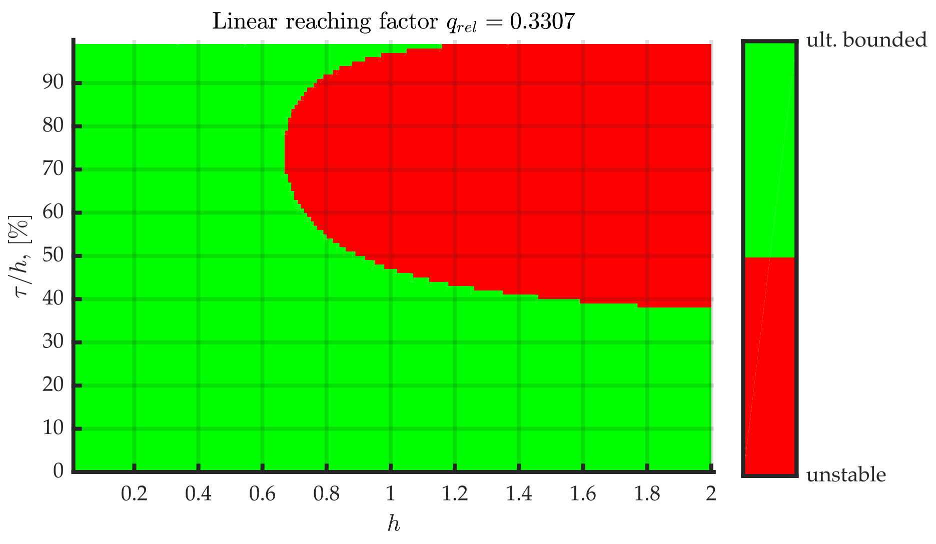
As has been done for Type 1a plants, several simulations with different plants are performed and the percentage of stable areas is evaluated and depicted in figure 4.10. This figure shows again that for every plant of Type 1b the thesis of improved performance with increasing parameter q_{rel} is satisfied. It is also interesting that the gradient of the curves in figure 4.8 significantly increases for values close to $q_{rel} = 1$.

The Type 2 plant

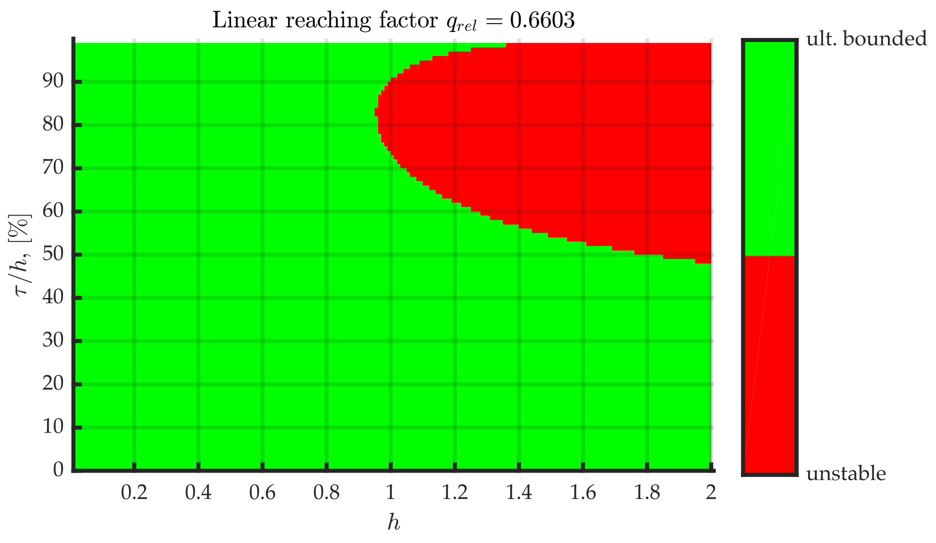
$$\dot{\boldsymbol{x}} = \begin{bmatrix} 8.0 & 5.09 \\ -6.13 & -3.07 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} -1.63 \\ -6.89 \end{bmatrix} u$$


Linear reaching factor $q_{rel} = 0.0010$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

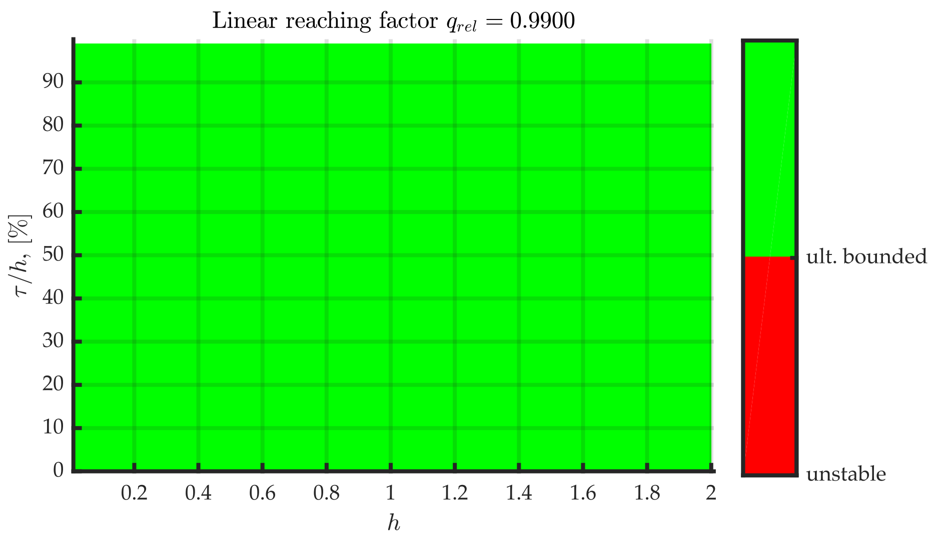
(A)


Linear reaching factor $q_{rel} = 0.3307$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(B)


Linear reaching factor $q_{rel} = 0.6603$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(C)



(D)

FIGURE 4.7: Stability areas for same Type 1a plant with growing linear reaching factor q_{rel}

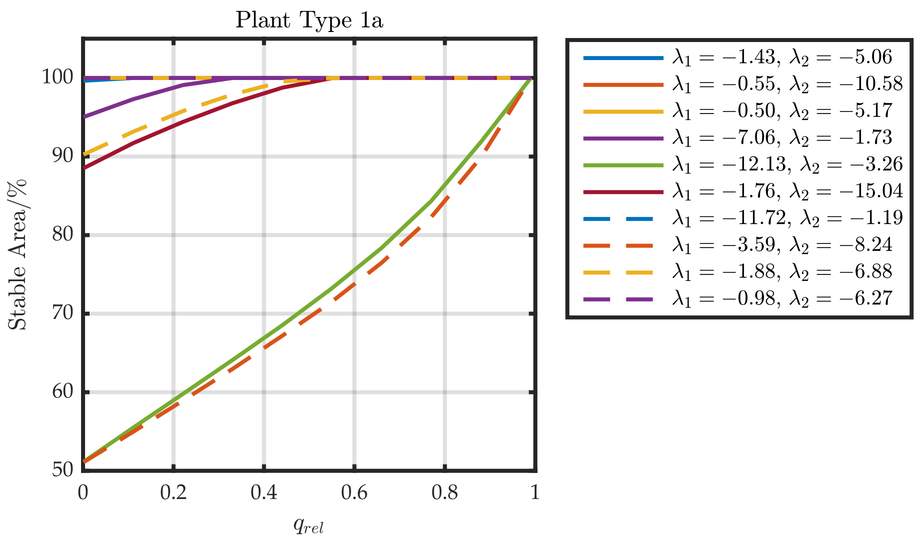
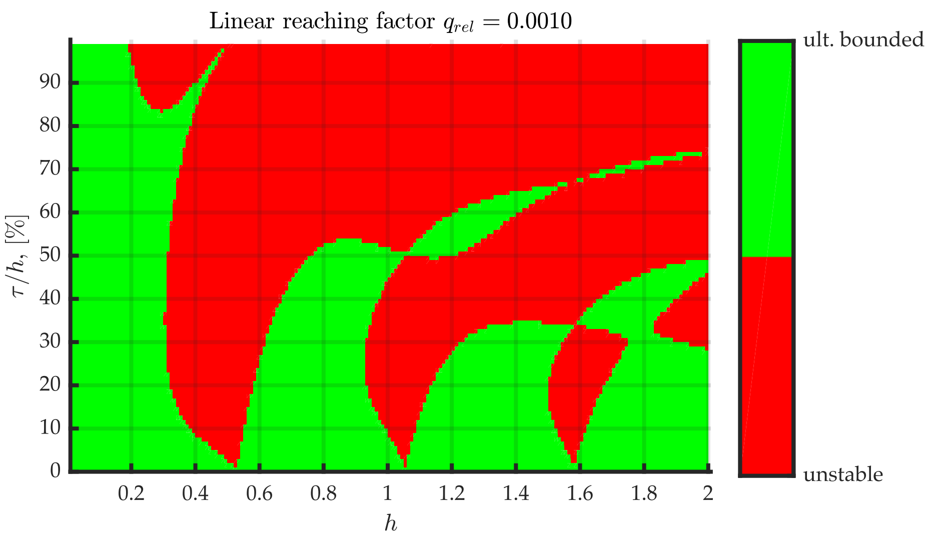
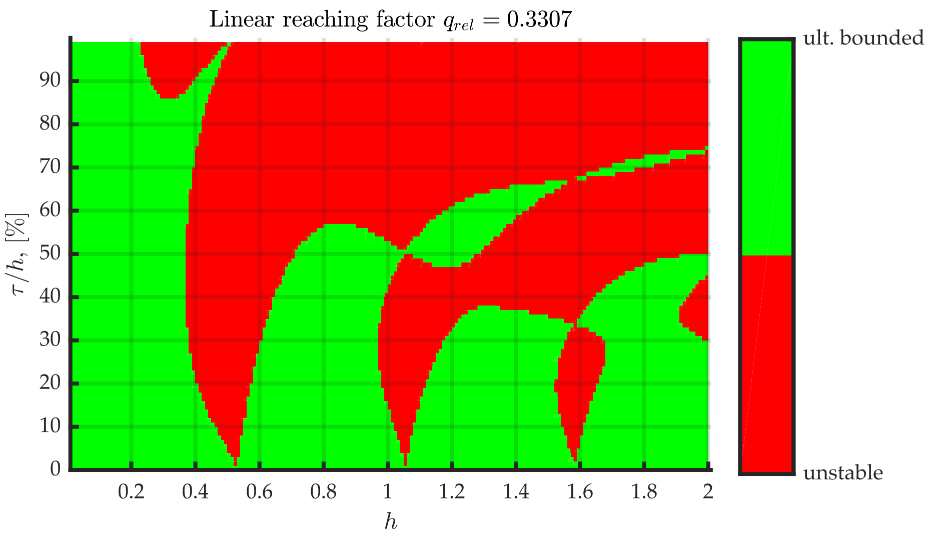


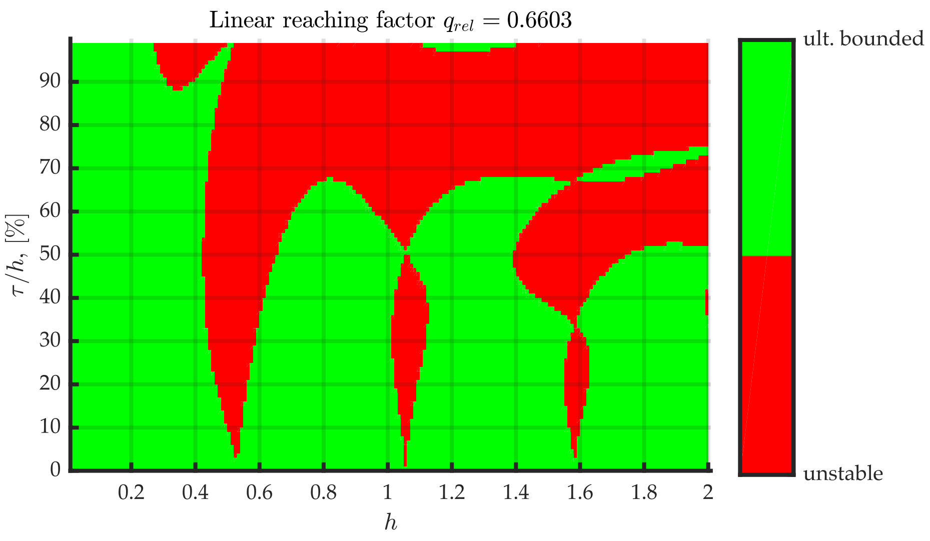
FIGURE 4.8: Impact of relative linear reaching parameter q_{rel} on the percentage of stable areas for different Type 1a plant systems.


Linear reaching factor $q_{rel} = 0.0010$
ult. bounded
unstable
τ/h, [%]
h

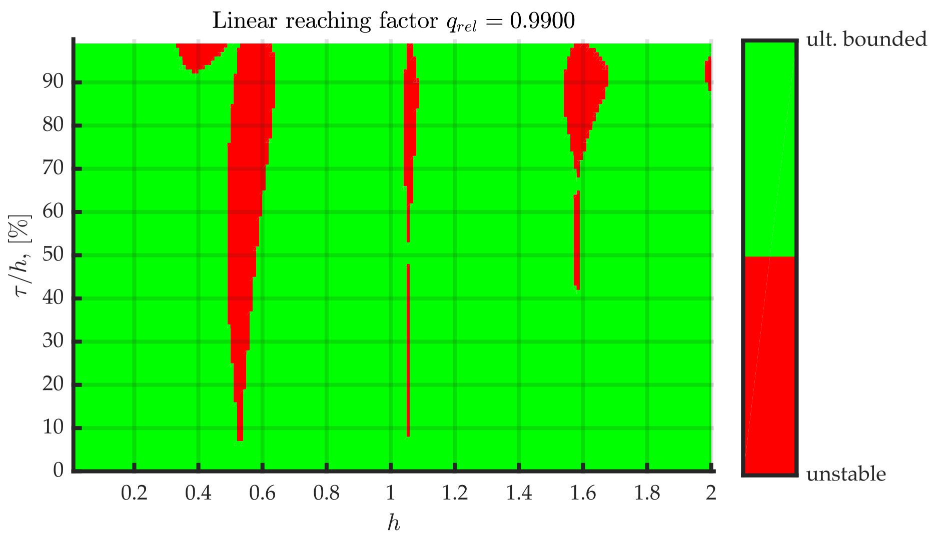
(A)


Linear reaching factor $q_{rel} = 0.3307$
ult. bounded
unstable
τ/h, [%]
h

(B)


Linear reaching factor $q_{rel} = 0.6603$
ult. bounded
unstable
τ/h, [%]
h

(C)



(D)

FIGURE 4.9: Stability areas for same Type 1b plant with growing linear reaching factor q_{rel}

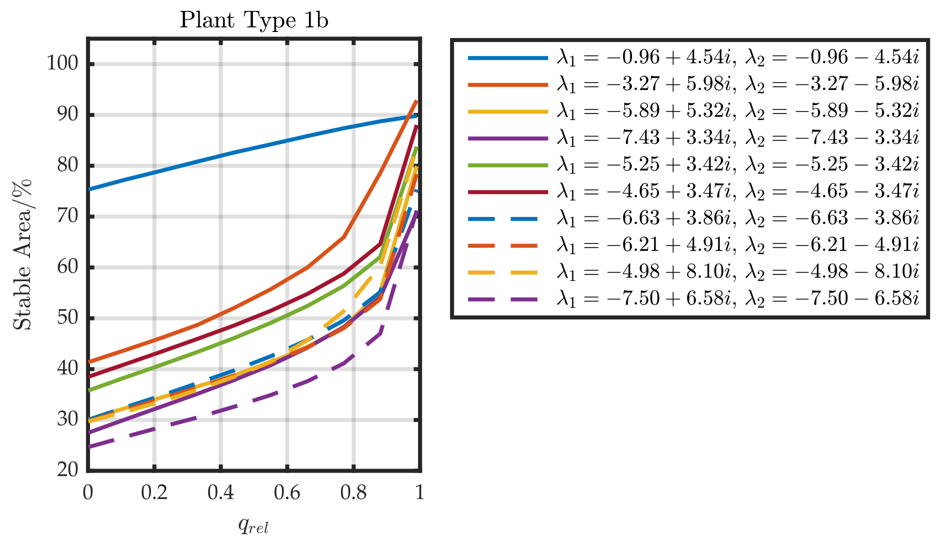


FIGURE 4.10: Impact of relative linear reaching parameter q_{rel} on the percentage of stable areas for different Type 1b plant systems.

with eigenvalues

$$\lambda = \begin{bmatrix} 2.46 + 0.734\,\mathrm{i} \\ 2.46 - 0.734\,\mathrm{i} \end{bmatrix}$$

was used in simulation and the resulting stability regions are shown in figure 4.11. In these plots, the known exponential shape is visible but it is very interesting that the area of ultimate boundedness decreases with increasing parameter q_{rel}. This is in contrast to the Type 1 plants, which causes this area increases with growing q_{rel}. The question is, if this plant produces an outlier or if this is the normal behaviour of a control loop with Type 2 plants.

In order to answer this question, several simulations with Type 2 plants are performed and the percentage area of ultimate boundedness is again drawn. The result is shown in figure 4.12. This figure shows that for all 10 different Type 2 plants the performance of the closed loop in terms of ultimate boundedness is decreasing with increasing relative linear reaching parameter q_{rel}. This is very interesting because closed loop systems with Type 1 plants perform completely controversial.

So the big question is how the Type 3 plants with the mixed (stable/unstable) poles behave. To answer this question two sample plants are needed, which was not required for the other types. Simulating the first model

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -9.91 & 7.23 \\ 0.852 & 8.18 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 6.91 \\ 7.58 \end{bmatrix} u$$

with eigenvalues of the system matrix

$$\lambda = \begin{bmatrix} -10.2 \\ 8.52 \end{bmatrix}$$

results in the stability regions shown in figure 4.13. The simulation results of the second model

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 8.89 & -5.11 \\ -6.52 & 2.82 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 6.17 \\ 7.07 \end{bmatrix} u$$

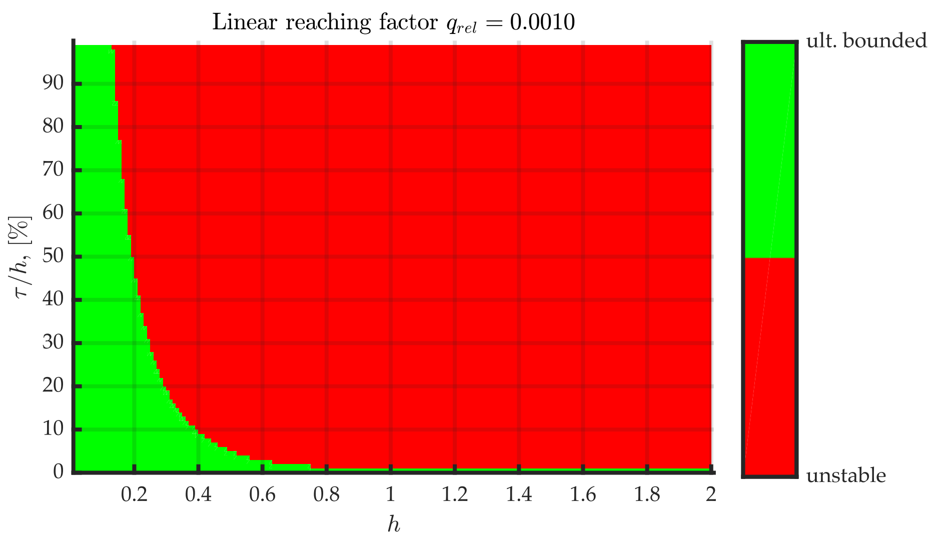
with eigenvalues of the system matrix

$$\lambda = \begin{bmatrix} 12.4 \\ -0.666 \end{bmatrix}$$

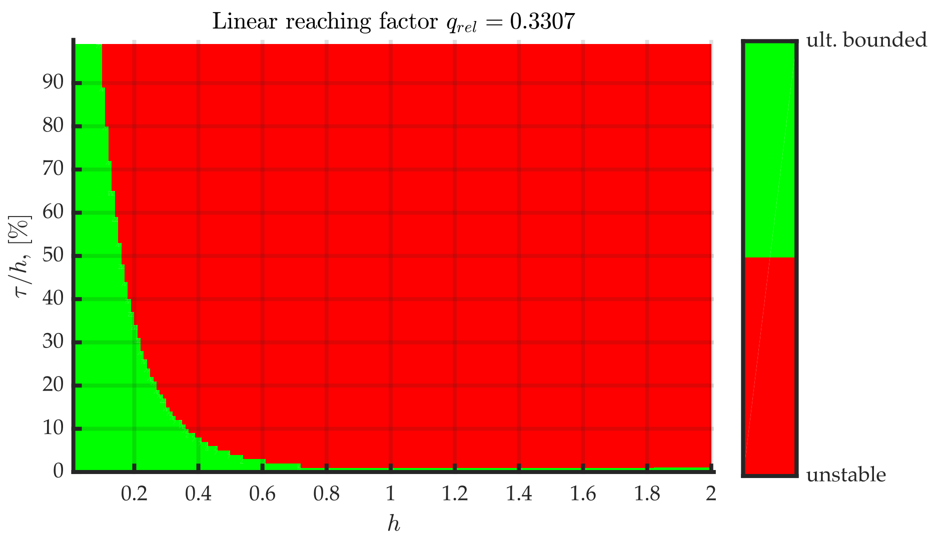
are depicted in figure 4.14.

Comparing these figures shows that the ultimatly bounded area with the first model, whose results are shown in figure 4.13, shrinks with increasing q_{rel}. The second model, whose results are depicted in figure 4.14 behaves contrariwise. Here it is visible that the green area grows for larger values of q_{rel}.

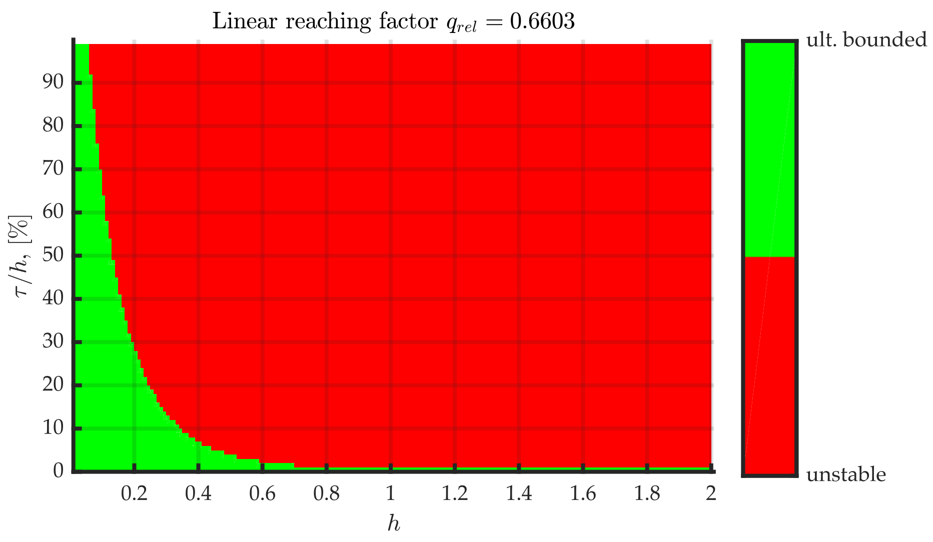
The plot with multiple simulation runs is shown in figure 4.15. As expected, there are some plants which lead to monotonically growing lines and others lead to monotonically falling lines.


Linear reaching factor $q_{rel} = 0.0010$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

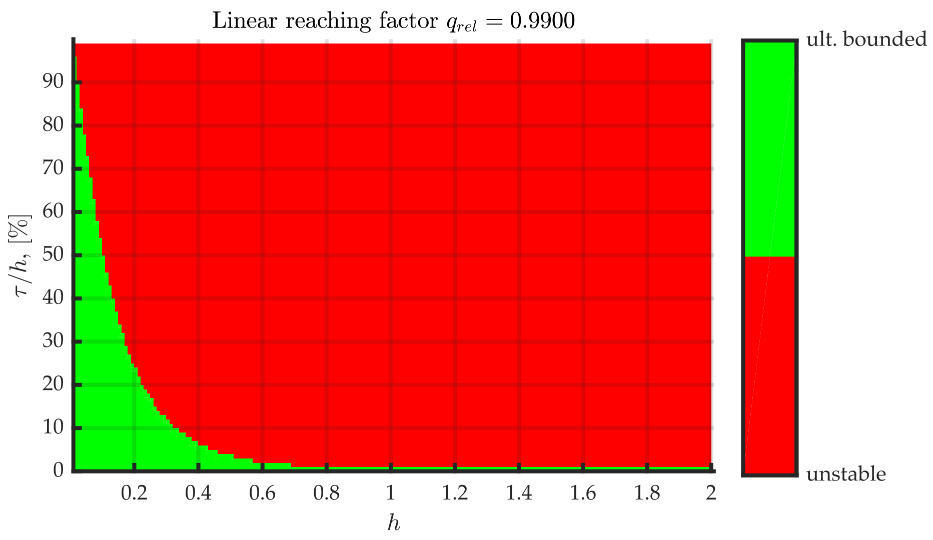
(A)


Linear reaching factor $q_{rel} = 0.3307$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(B)


Linear reaching factor $q_{rel} = 0.6603$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(C)



(D)

FIGURE 4.11: Stability areas for same plant with growing linear reaching factor q_{rel}

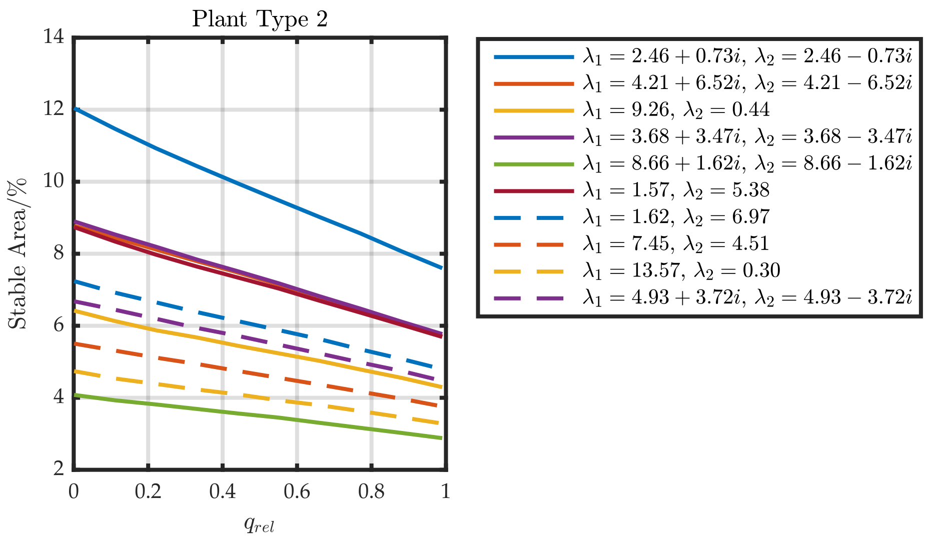
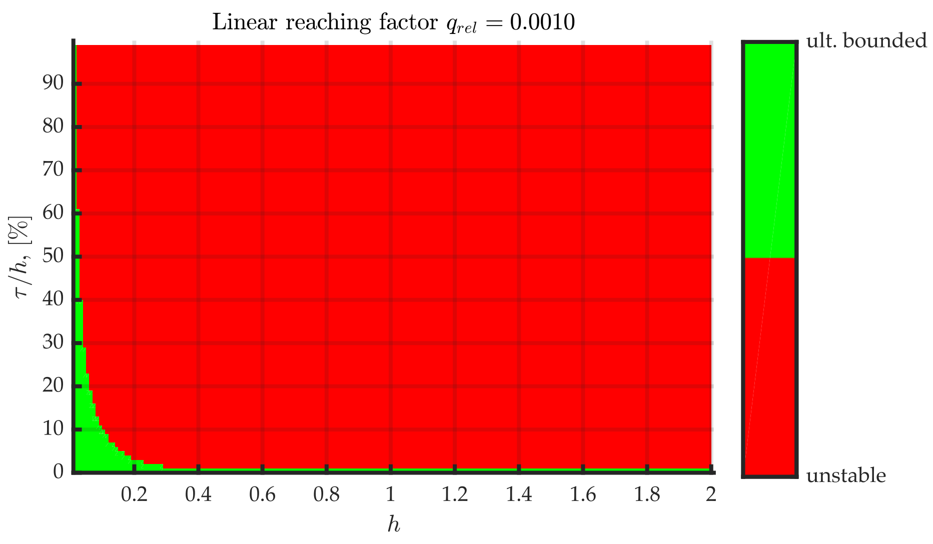
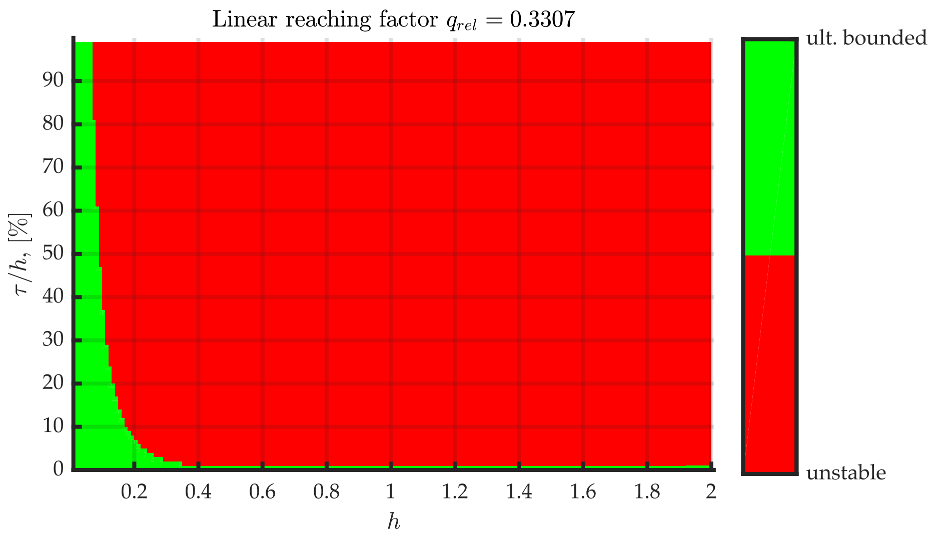


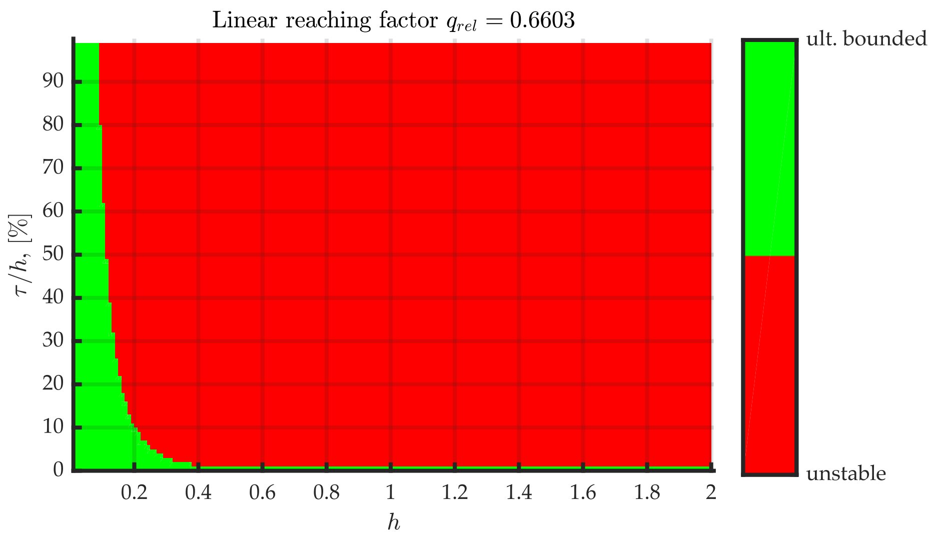
FIGURE 4.12: Impact of relative linear reaching parameter q_{rel} on the percentage of stable areas for different Type 2 plant systems.


Linear reaching factor $q_{rel} = 0.0010$
τ/h, [%]
90
80
70
60
50
40
30
20
10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

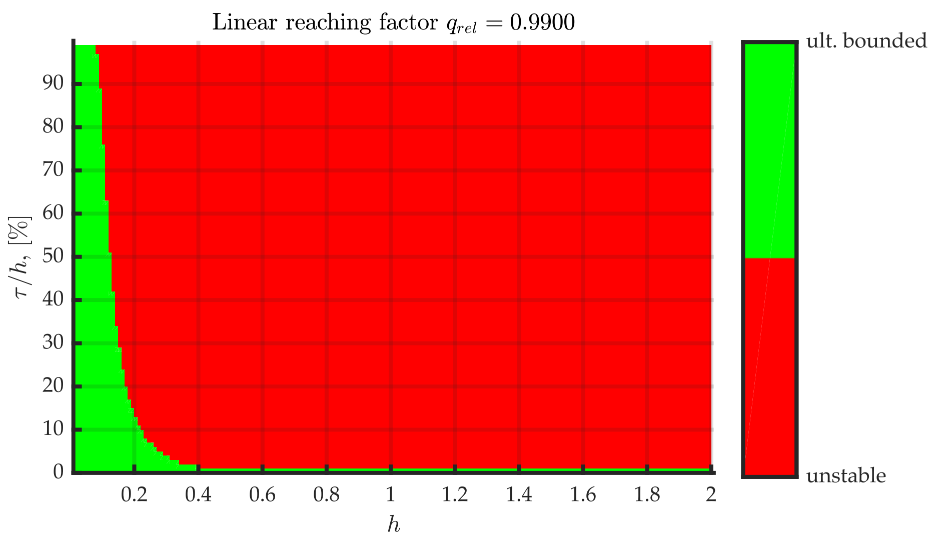
(A)


Linear reaching factor $q_{rel} = 0.3307$
τ/h, [%]
90
80
70
60
50
40
30
20
10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(B)


Linear reaching factor $q_{rel} = 0.6603$
τ/h, [%]
90
80
70
60
50
40
30
20
10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(C)



(D)

FIGURE 4.13: Stability areas for same plant with growing linear reaching factor q_{rel}

4.5.3 Conclusions

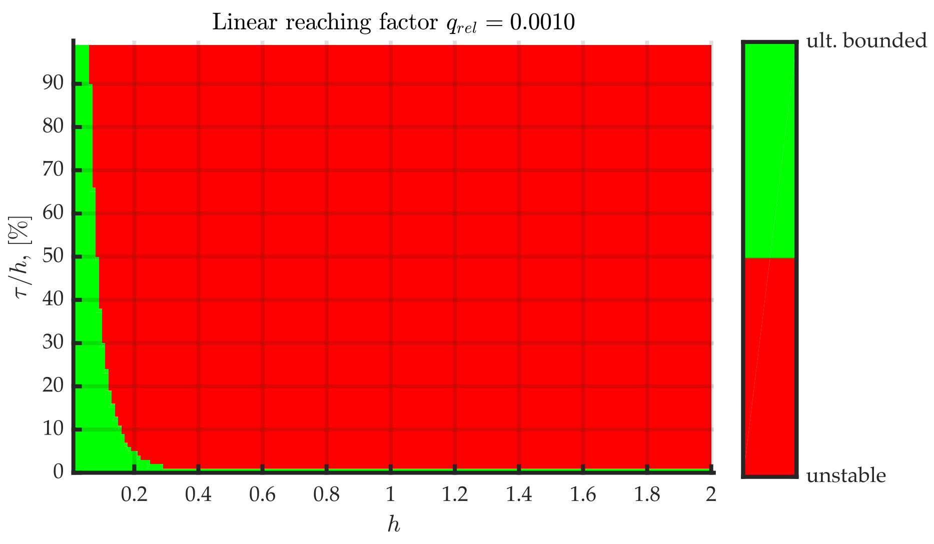
Several simulations with the different plants are performed by varying the relative linear reaching parameter q_{rel}. It was found that the differentiation of the plants in three types, as introduced in section 4.4, is also reasonable for this scenario because the effect of this parameter q_{rel} is different for these three types. For Type 1 plants, the solutions of the closed loop become ultimately bounded in a larger area by growing the parameter q_{rel}. With Type 2 plants the behaviour is completely contrariwise. The area of ultimate boundedness is shrinking for larger values of q_{rel}. The Type 3 plants are between Type 1 and Type 2 plants. There are cases in which the area grows and some in which it contracts for greater values of q_{rel}.

4.6 Comparison of Stability Regions Using Sliding Mode Controllers and Linear State Controllers

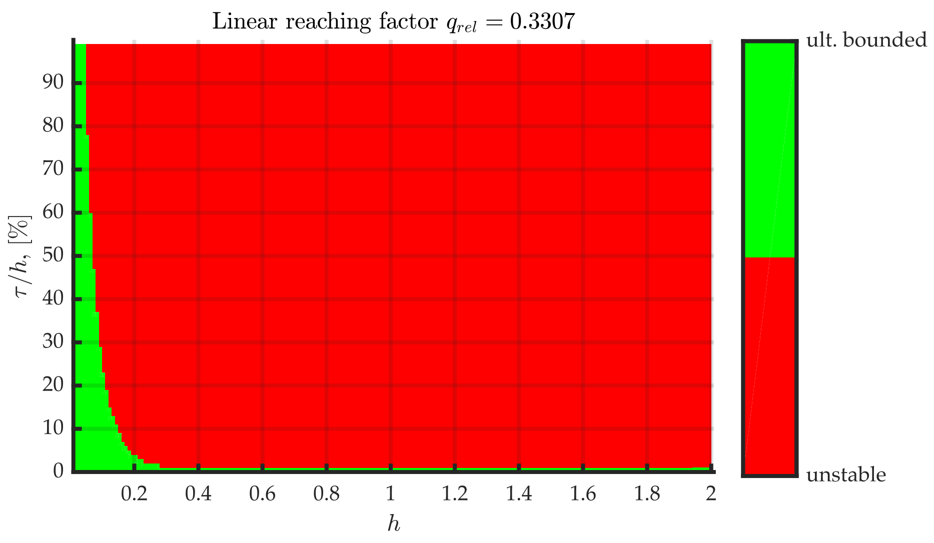
To compare the performance of the sliding mode control approach with the conventional state controller method, several simulations are performed for both controller types with the same plant system and the stability areas are plotted. As both controllers have different design parameters, it is not possible to perform an entirely realistic comparison. It is also important to keep in mind that generally two different properties are compared. By using the linear state controller it is verified that the solutions are asymptotically stable. By using the sliding mode controller it is verified that the solutions are ultimately bounded. However, a tendency of the behaviour can be derived by choosing the design parameters for the two controllers reasonably.

4.6.1 Configuration

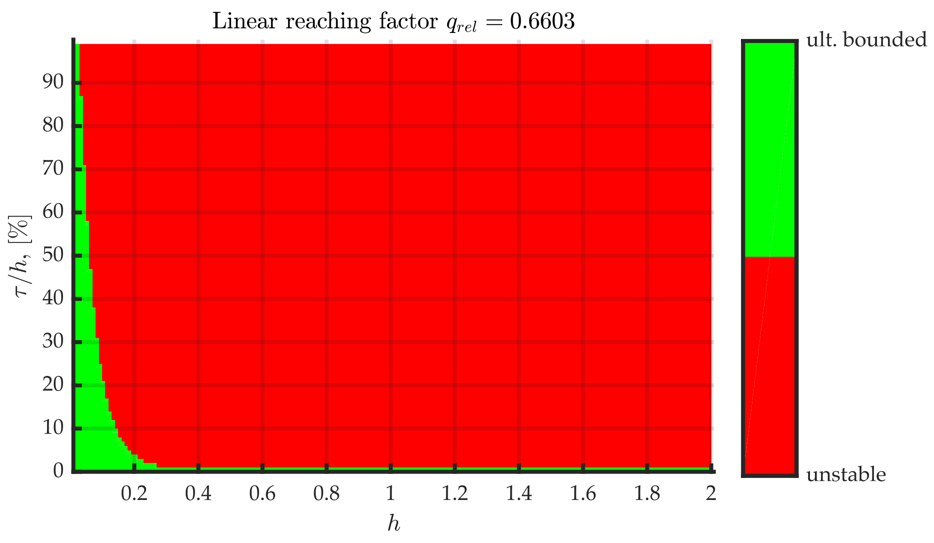
The parameters for the discrete sliding mode controller (3.8) are:


Linear reaching factor $q_{rel} = 0.0010$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

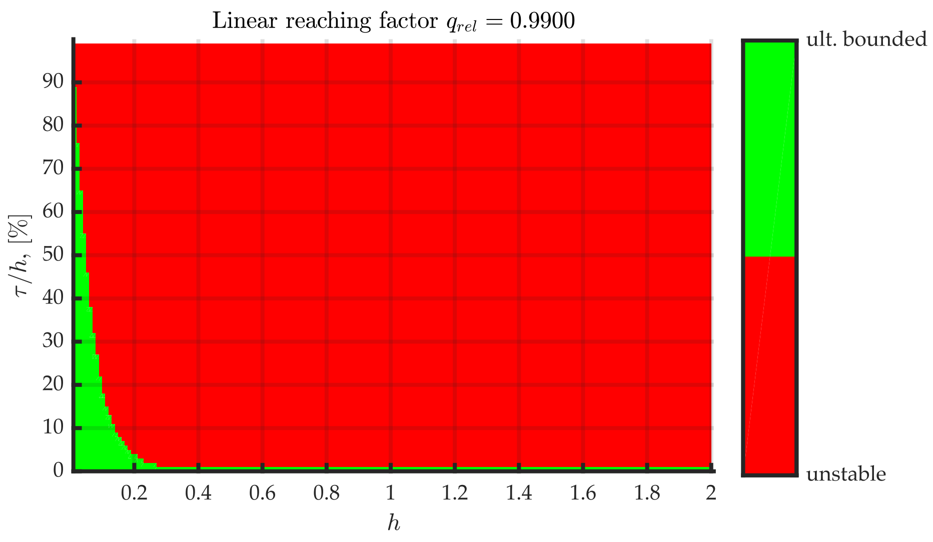
(A)


Linear reaching factor $q_{rel} = 0.3307$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(B)


Linear reaching factor $q_{rel} = 0.6603$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(C)



(D)

FIGURE 4.14: Stability areas for same plant with growing linear reaching factor q_{rel}

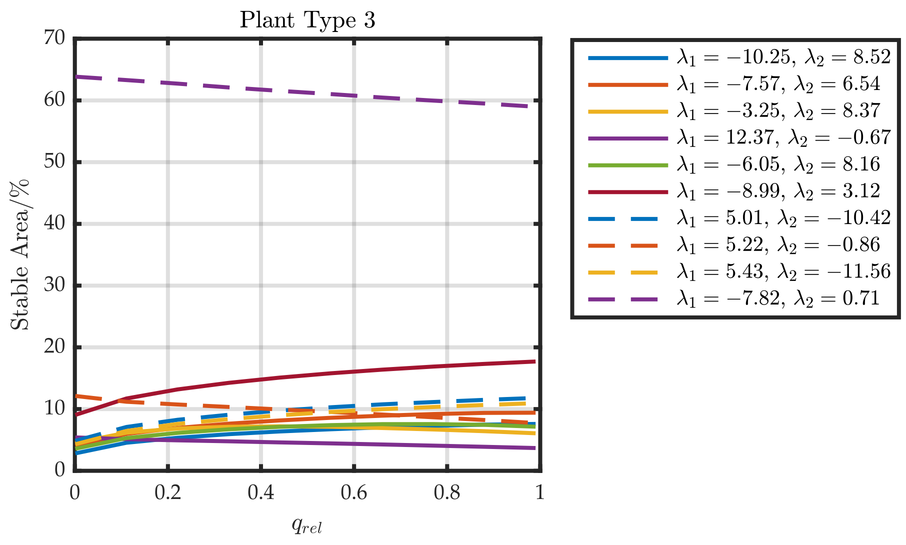


FIGURE 4.15: Impact of relative linear reaching parameter q_{rel} on the percentage of stable areas for different Type 3 plant systems.

- The $n-1$ eigenvalues, defining the quasi sliding mode
- The linear reaching parameter q
- The sliding mode gain ρ, which does not influence the ultimate boundedness analysed in this chapter. But is important if local stability or the influence of perturbations are analysed.

A linear state controller is designed by using n eigenvalues as the single design parameter.

In order to compare both controller types, it is necessary to define a reasonable choice for the n eigenvalues of the linear state controller. It was defined that using the $n-1$ eigenvalues, which define the ideal quasi sliding mode, should be used and the remaining one is received by randomly duplicating one eigenvalue. As in the further simulations only order $n=2$ plants are considered, the two eigenvalues for the linear state controller are derived by using the single eigenvalue of the sliding mode controller twice. The configuration could be summarised as follows

- Order $n=2$
- Uniformly distributed random values in the interval $\begin{bmatrix} -10, & 10 \end{bmatrix}$ for the entries in the system matrix A and the input vector b
- Stepsizes h values in the interval $\begin{bmatrix} 0.01, & 2 \end{bmatrix}$ with increment 0.01
- Relative delay $\tau_{rel} = \frac{\tau}{h}$ in the interval $\begin{bmatrix} 0.0, & 0.99 \end{bmatrix}$ with increment 0.01
- Relative linear reaching parameter $q_{rel} = 0.8801$
- Continuous ideal sliding mode eigenvalue $\lambda_c = -1 \Rightarrow \lambda_d = e^{\lambda_c h}$
- Continuous eigenvalues for the linear state controller $\lambda_{lin,c} = \begin{bmatrix} -1, & -1 \end{bmatrix}$

Several simulations are performed with this configuration and the stability regions for the different plant system types, defined in section 4.4.1, are drawn.

4.6.2 Results

Using the Type 1a plant

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -9.45 & 2.2 \\ 7.52 & -5.93 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0.398 \\ -8.92 \end{bmatrix} u \qquad (4.18)$$

with eigenvalues of the system matrix A:

$$\lambda = \begin{bmatrix} -12.1 \\ -3.26 \end{bmatrix}$$

results in the stability regions shown in figures 4.16a and 4.16b. The stability region in figure 4.16a was received by using the sliding mode controller and the one in figure 4.16b by using the linear state controller. It is clearly visible by comparing the two stability regions that the performance when using the sliding mode controller is slightly better than when using the linear state controller but the shape of the regions are quite similar. To be able to compare the performances of the two controllers more easily, the percentage of green area is written in the title of each figure.

As a single simulation is not very representative, a second one with a different Type 1a plant system was performed. The plant can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -5.72 & 5.69 \\ 0.941 & -6.11 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 4.94 \\ -0.489 \end{bmatrix} u \tag{4.19}$$

with eigenvalues of the system matrix A:

$$\lambda = \begin{bmatrix} -3.59 \\ -8.24 \end{bmatrix}.$$

The stability regions for this plant system are shown in figures 4.17a and 4.17b. These figures undergird the better performance of the sliding mode controller.

For the Type 1b plant system

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -0.208 & 9.79 \\ -4.6 & -6.33 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 7.23 \\ -9.35 \end{bmatrix} u \tag{4.20}$$

with corresponding eigenvalues

$$\lambda = \begin{bmatrix} -3.27 + 5.98\,\mathrm{i} \\ -3.27 - 5.98\,\mathrm{i} \end{bmatrix}$$

the stability regions are shown in figures 4.18a and 4.18b. The better performance of the sliding mode controller as well as the similar shape of the stability regions are cognisable by comparing the two stability regions. The plots in figures 4.19a and 4.19b are the results of simulating the plant

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -5.93 & -2.13 \\ 6.28 & -8.93 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} -2.5 \\ 5.5 \end{bmatrix} u \tag{4.21}$$

with the eigenvalues of the system matrix A

$$\lambda = \begin{bmatrix} -7.43 + 3.34\,\mathrm{i} \\ -7.43 - 3.34\,\mathrm{i} \end{bmatrix}.$$

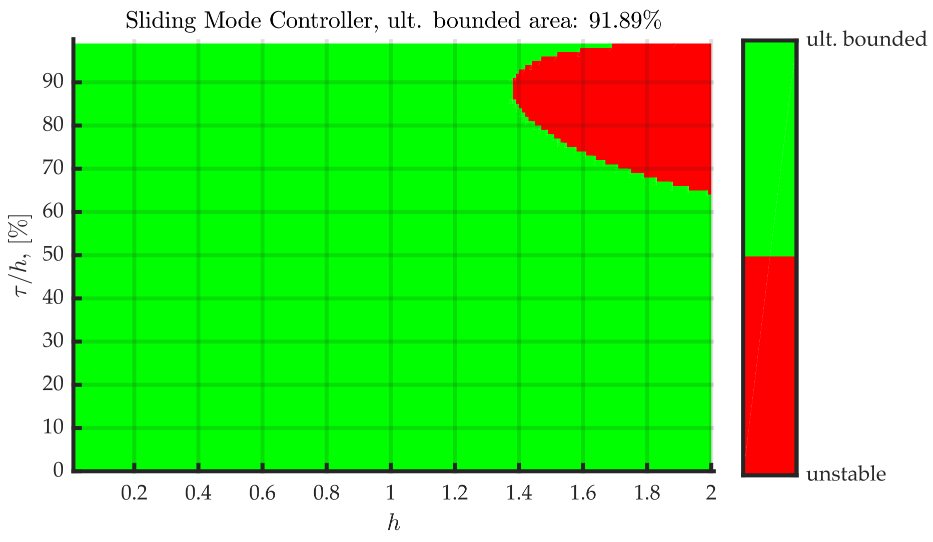
These figures again show the same behaviour of similar shape and better performance.

The same simulations are performed for Type 2 plant systems. The first results were received by using the plant system

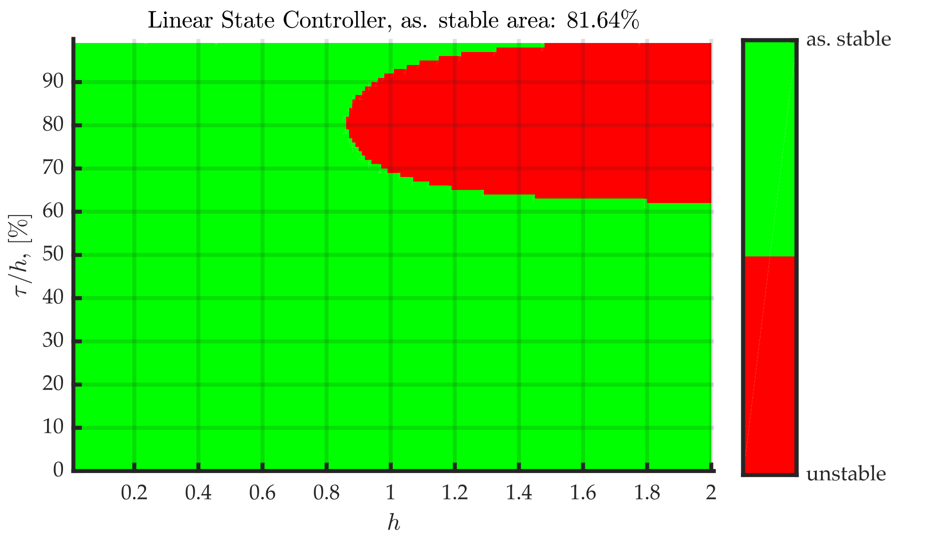
$$\dot{\boldsymbol{x}} = \begin{bmatrix} 8.0 & 5.09 \\ -6.13 & -3.07 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} -1.63 \\ -6.89 \end{bmatrix} b \tag{4.22}$$

with eigenvalues

$$\lambda = \begin{bmatrix} 2.46 + 0.734\,\mathrm{i} \\ 2.46 - 0.734\,\mathrm{i} \end{bmatrix}.$$

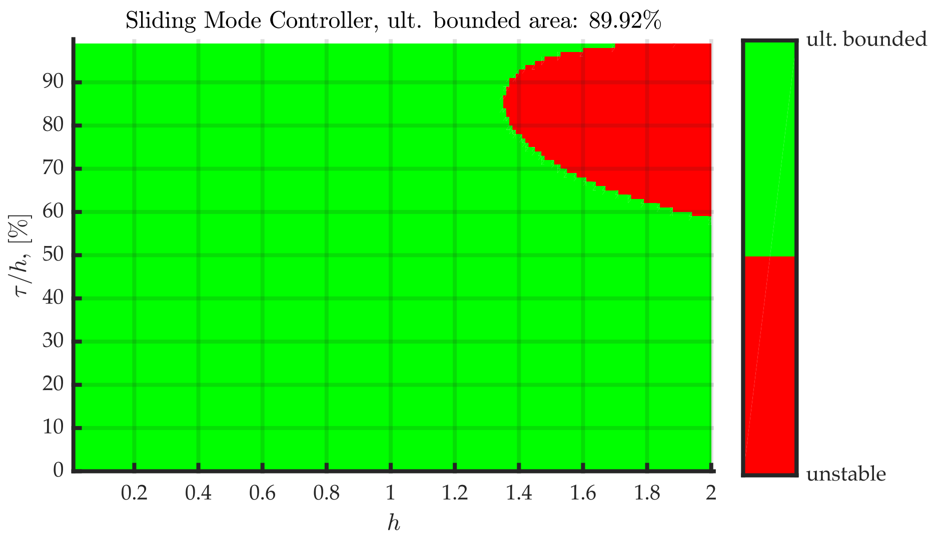


(A) Type 1a plant controlled by sliding mode controller

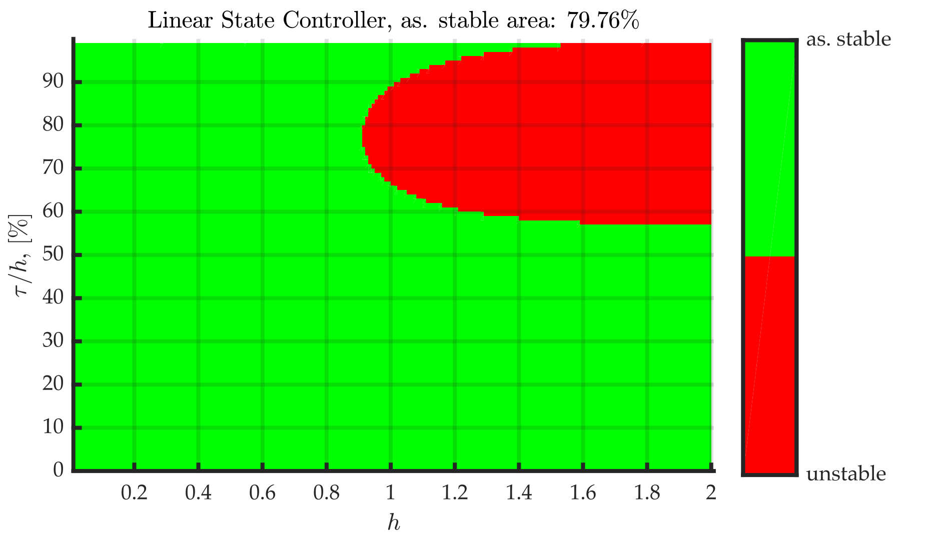


(B) Type 1a plant controlled by linear state controller

FIGURE 4.16: Comparison of stability regions by using sliding mode controller and linear state controller for Type 1a system (4.18)

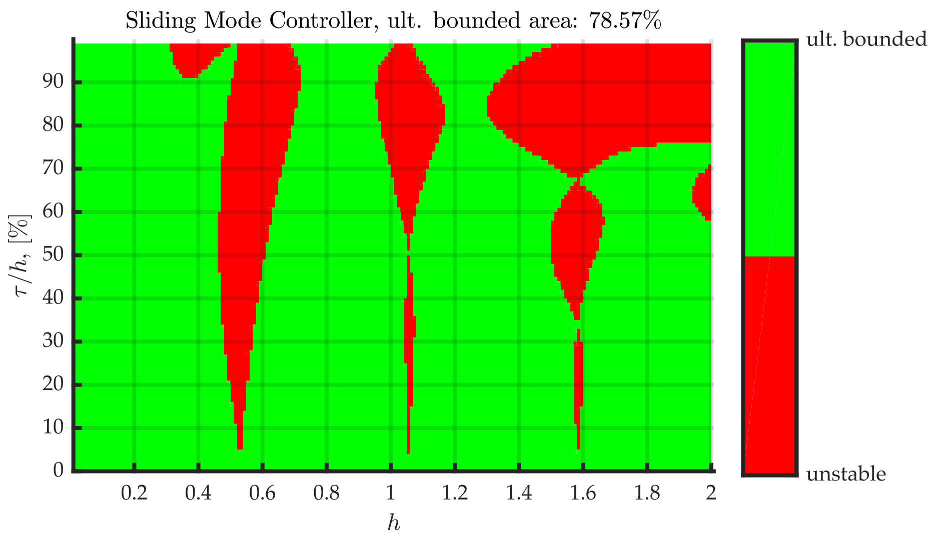


(A) Type 1a plant controlled by sliding mode controller

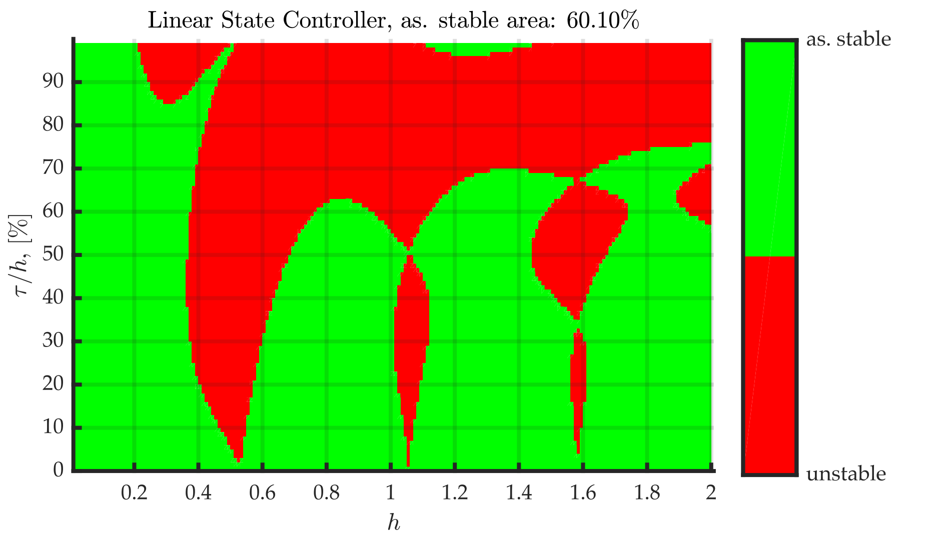


(B) Type 1a plant controlled by linear state controller

FIGURE 4.17: Comparison of stability regions by using sliding mode controller and linear state controller for Type 1a system (4.18)

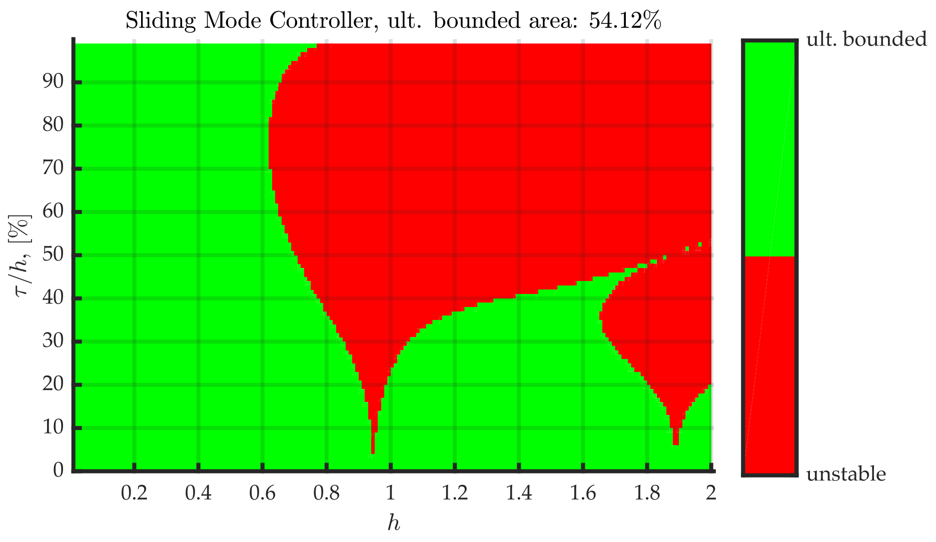


(A) Type 1b plant controlled by sliding mode controller

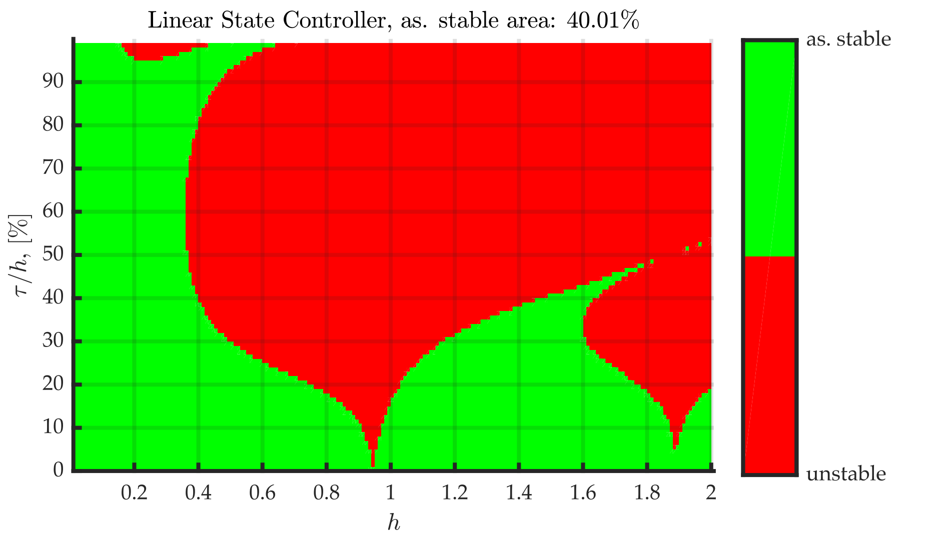


(B) Type 1b plant controlled by linear state controller

FIGURE 4.18: Comparison of stability regions by using sliding mode controller and linear state controller for Type 1b system (4.20)



(A) Type 1b plant controlled by sliding mode controller



(B) Type 1b plant controlled by linear state controller

FIGURE 4.19: Comparison of stability regions by using sliding mode controller and linear state controller for Type 1b system (4.21)

The corresponding stability areas are shown in figures 4.20a and 4.20b. One can see the similar shape of the stability areas by comparing these two figures. In contrast to the Type 1a and Type 1b case, the performance of the sliding mode controller is slightly worse than when using the linear state controller. In order to verify this hypothesis a second simulation was performed with the following plant system

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1.99 & -7.9 \\ 6.01 & 6.43 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 6.82 \\ -2.91 \end{bmatrix} u \tag{4.23}$$

with the eigenvalues of the system matrix A

$$\lambda = \begin{bmatrix} 4.21 + 6.52\,\mathrm{i} \\ 4.21 - 6.52\,\mathrm{i} \end{bmatrix}.$$

The results are shown in figures 4.21a and 4.21b. This simulation affirms the previous hypothesis because the shape is again very similar, even the small spike is visible in both plots. The slightly worse performance of the sliding mode controller is also confirmed.

The stability areas for Type 3 systems are the last missing ones. By using the Type 3 plant

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -7.75 & 2.77 \\ 4.86 & 1.88 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} -0.0275 \\ 1.36 \end{bmatrix} u \tag{4.24}$$

with corresponding eigenvalues

$$\lambda = \begin{bmatrix} -8.99 \\ 3.12 \end{bmatrix}$$

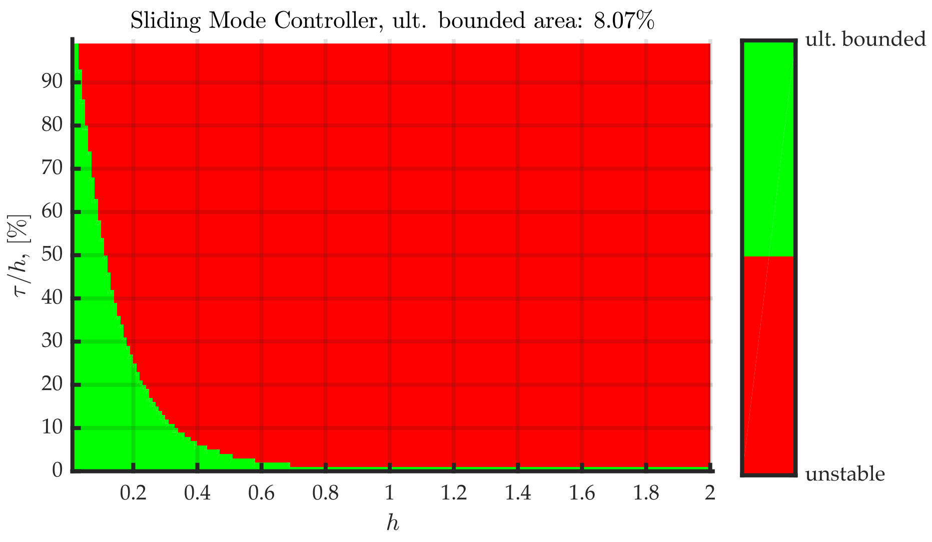
the stability areas depicted in figures 4.22a and 4.22b result. Comparing both plots makes it visible that the shape is again very similar. In this case the performance is better when using the sliding mode controller. A second simulation was performed for the Type 3 plant system

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 2.38 & -9.8 \\ -0.94 & 1.98 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 2.03 \\ 2.99 \end{bmatrix} u \tag{4.25}$$

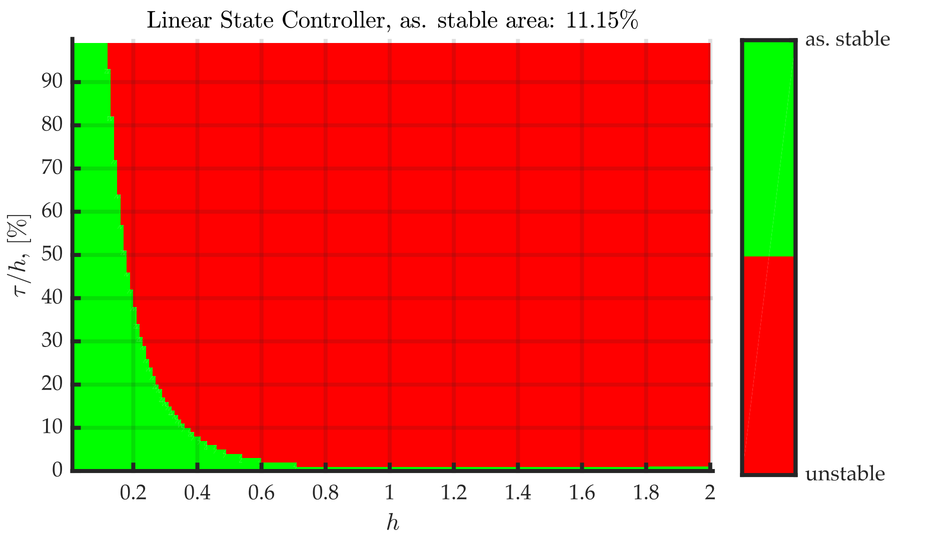
with corresponding eigenvalues

$$\lambda = \begin{bmatrix} 5.22 \\ -0.861 \end{bmatrix}.$$

The resulting stability areas are shown in figures 4.23a and 4.23b. The simulation results in a quite similar shape. For this plant system the performance is better when using the linear state controller than when using the sliding mode controller. These simulation results for Type 3 plant systems show that there are cases in which the sliding mode controller performs better and some in which the performance of the linear state controller is better.

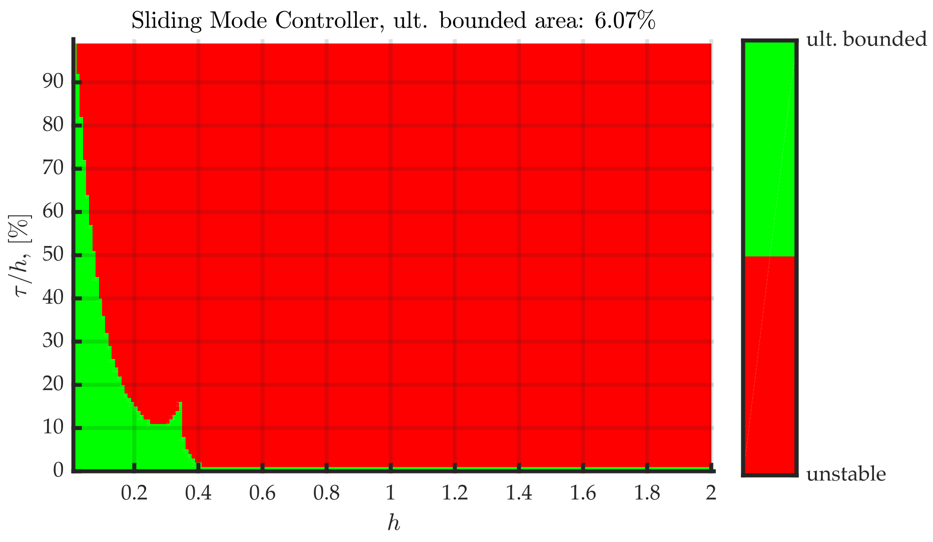


(A) Type 2 plant controlled by sliding mode controller

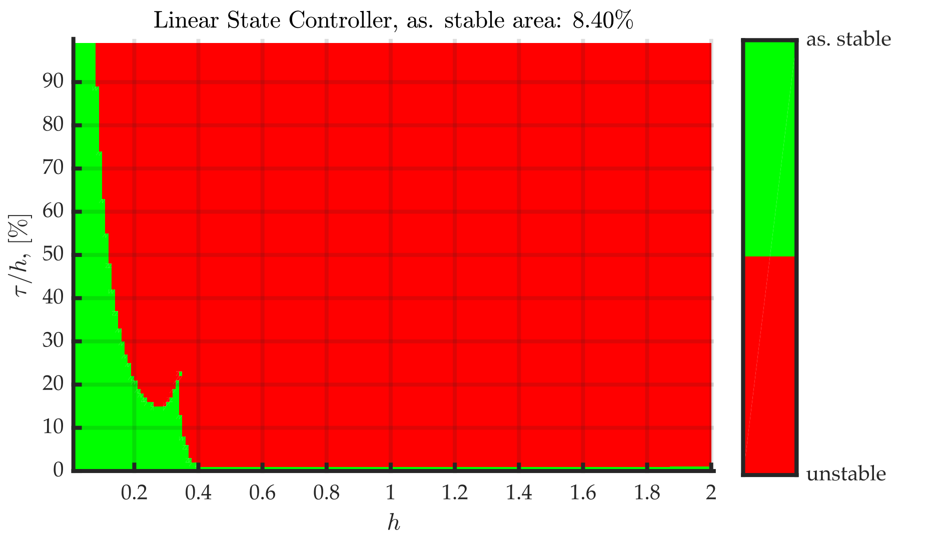


(B) Type 2 plant controlled by linear state controller

FIGURE 4.20: Comparison of stability regions by using sliding mode controller and linear state controller for Type 2 system (4.22)

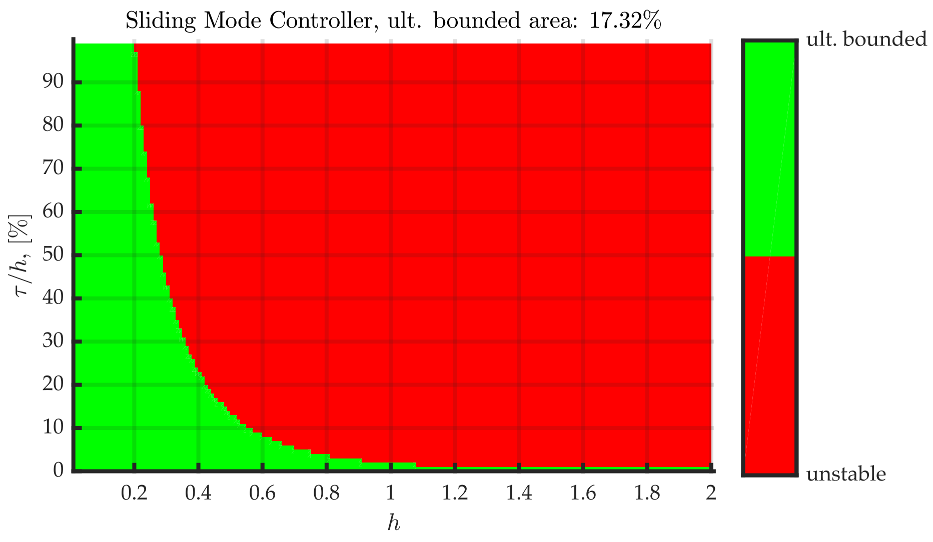


(A) Type 2 plant controlled by sliding mode controller

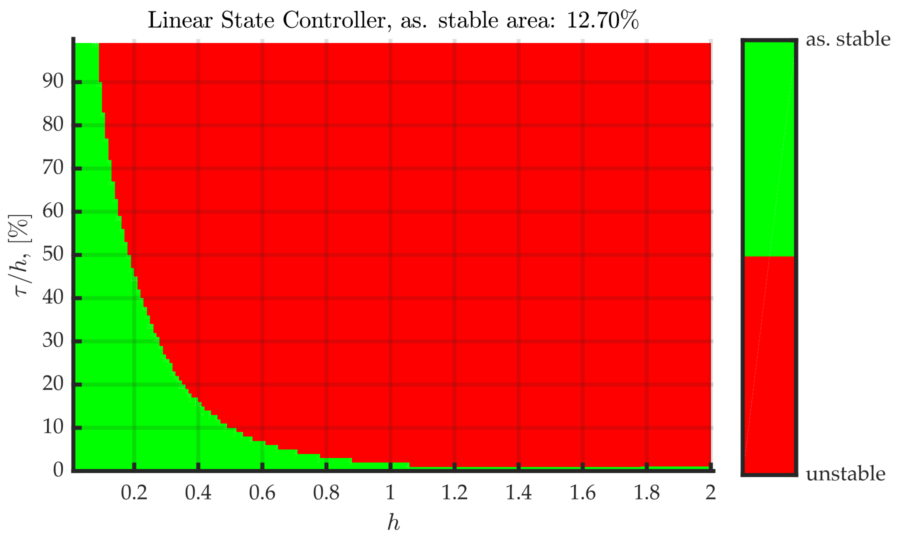


(B) Type 2 plant controlled by linear state controller

FIGURE 4.21: Comparison of stability regions by using sliding mode controller and linear state controller for Type 2 system (4.23)

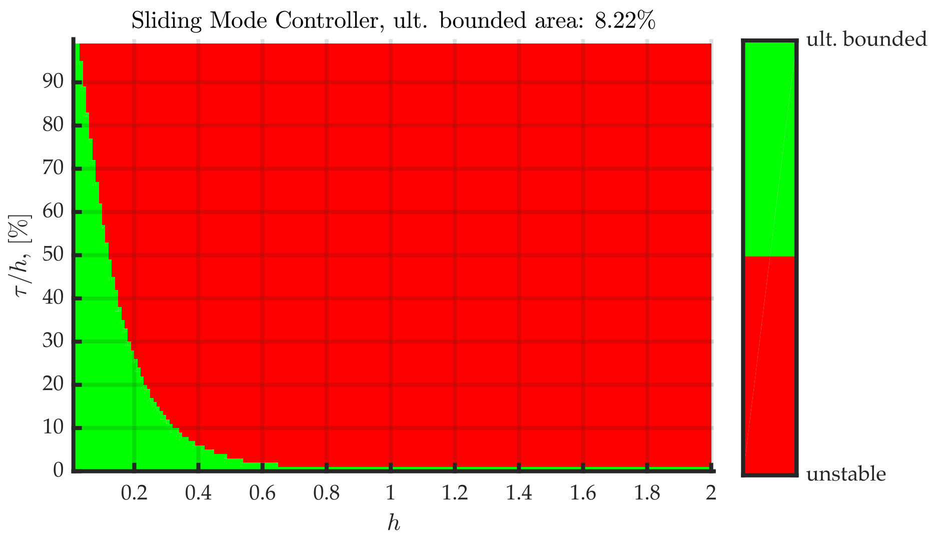


(A) Type 3 plant controlled by sliding mode controller

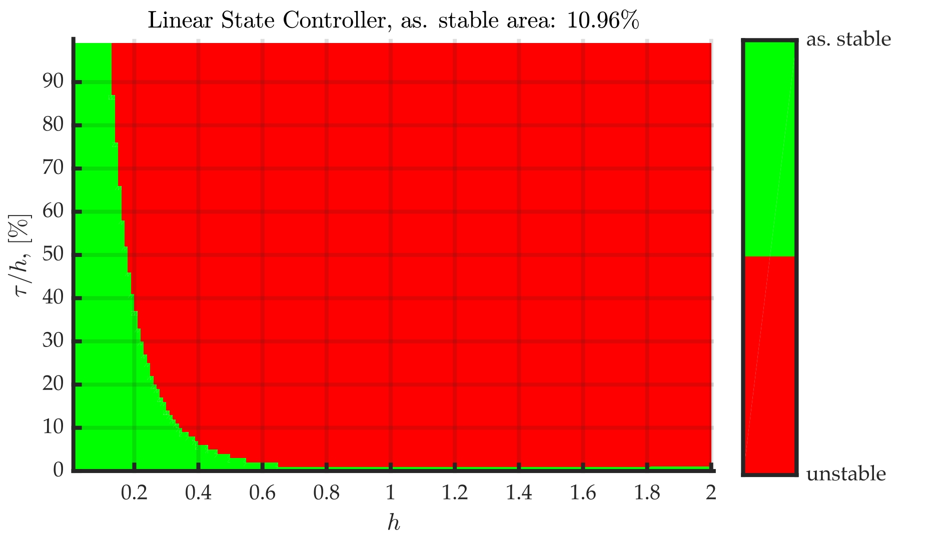


(B) Type 3 plant controlled by linear state controller

FIGURE 4.22: Comparison of stability regions by using sliding mode controller and linear state controller for Type 3 system (4.24)



(A) Type 3 plant controlled by sliding mode controller



(B) Type 3 plant controlled by linear state controller

FIGURE 4.23: Comparison of stability regions by using sliding mode controller and linear state controller for Type 3 system (4.25)

4.6.3 Conclusions

In this section, the stability areas which result when using a sliding mode controller are compared to the ones when using a linear state controller. The most interesting observation is that the shape of the stability areas for both controller types are quite similar for all plant types. By using the configuration stated in section 4.6.1 the sliding mode controller performs slightly better for Type 1a and 1b plant systems than the linear state controller. For Type 2 plant systems the linear state controller leads to a better performance. No such statement can be made for Type 3 plant systems because there are cases in which the performance when using the sliding mode controller is better and some in which it is contrariwise. It is important to keep in mind that this holds only for the defined configuration. By increasing the value q_{rel}, the ultimately bounded area for Type 1a and 1b plant systems also increases while for Type 2 plant systems this area shrinks (see section 4.5). This makes it possible to find a value for this parameter q_{rel} with which the performance of the sliding mode controller is better on Type 2 systems than the linear state controller. This parameter change does not influence the interesting fact that the stability areas when using the sliding mode controller and the linear state controller look quite similar

4.7 Stability Regions' Dependance on Eigenvalue λ_c

In this section, the influence of the eigenvalue λ_c, which defines the behaviour of the ideal sliding mode, is analysed. This is done by performing several simulations while varying this eigenvalue and keeping all other parameters constant. The simulations are again categorised by the plant's eigenvalues into the three types as described in section 4.4.1.

4.7.1 Configuration

- Order $n = 2$
- Uniformly distributed random values in the interval $\left[\ -10,\quad 10\ \right]$ for the entries in the system matrix A and the input vector b
- Stepsizes h values in the interval $\left[\ 0.01,\quad 2\ \right]$ with increment 0.01
- Relative delay $\tau_{rel} = \frac{\tau}{h}$ in the interval $\left[\ 0.0,\quad 0.99\ \right]$ with increment 0.01
- Relative linear reaching parameter $q_{rel} = 0.5$
- Ten linearly spaced values in the interval $\left[\ -3,\quad -0.01\ \right]$ as continuous ideal sliding mode eigenvalue λ_c

Several simulations are performed with this configuration and the stability regions for the different plant system types, defined in section 4.4.1, are drawn.

4.7.2 Results

The stability regions for four different values of the eigenvalue λ_c by using the Type 1a plant

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -9.45 & 2.2 \\ 7.52 & -5.93 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0.398 \\ -8.92 \end{bmatrix} u$$

are shown in figure 4.24. The used eigenvalue is attached at the top of each image and decreases from the top left to the bottom right plot. It is clearly visible that the area of ultimate boundedness increases by placing the pole at smaller values if the plant is of Type 1a. In order to confirm this hypothesis, additional simulations with different plant configurations of Type 1a are performed and the percentage of ultimately bounded areas is calculated and plotted while the eigenvalue λ_c is varied. This plot is shown in figure 4.25. This figure shows that all lines are monotonically decreasing. This means that increasing the eigenvalue λ_c decreases the ultimately bounded area.

Similar simulations are performed using Type 1b plant systems. The stability areas for the plant

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -4.0 & 8.38 \\ -3.2 & -0.875 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} -1.15 \\ -0.916 \end{bmatrix} u$$

are shown in figure 4.26, here the same phenomenon as for Type 1a plants is detectable. That the ultimately bounded area increases by decreasing the eigenvalue λ_c. This again is confirmed by performing multiple simulations with different plant configurations of Type 1b. The result is shown in figure 4.27, where also the monotonically decreasing lines are visible.

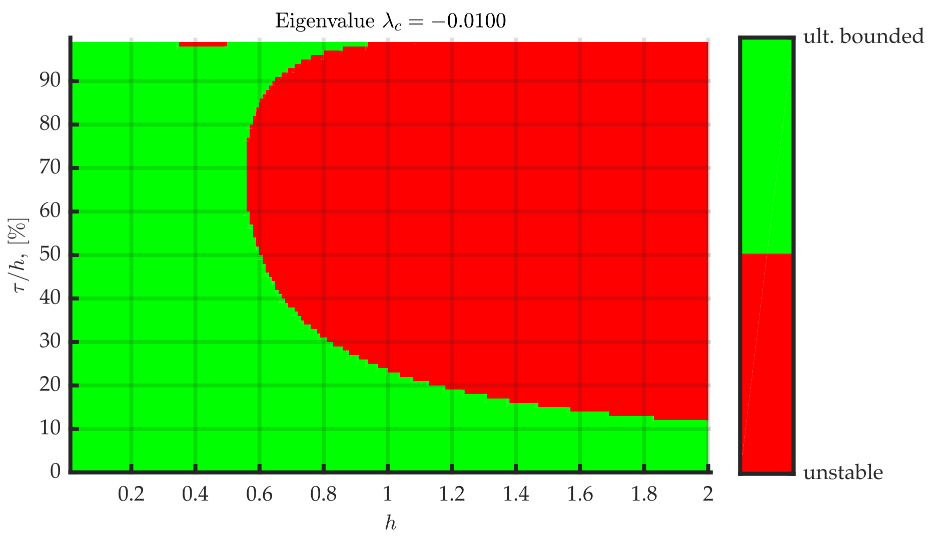
The behaviour when using Type 2 plant systems again is contrariwise to the one when using Type 1a and 1b plant systems. By carefully comparing the plots in figure 4.28, one can see that the area of ultimate boundedness slightly increases with increasing value of λ_c. These results were gained by using the following plant system.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1.45 & 0.0473 \\ -6.89 & 1.35 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} -6.23 \\ -3.52 \end{bmatrix} u$$

Multiple simulations are again performed in order to confirm that increasing the eigenvalue λ_c results generally results in a larger ultimately bounded area for Type 2 plants. The result is shown in figure 4.29. The legend shows the eigenvalues of the system matrix A of each used plant. As this picture shows, all lines are monotonically increasing, which confirms the previous assumption.

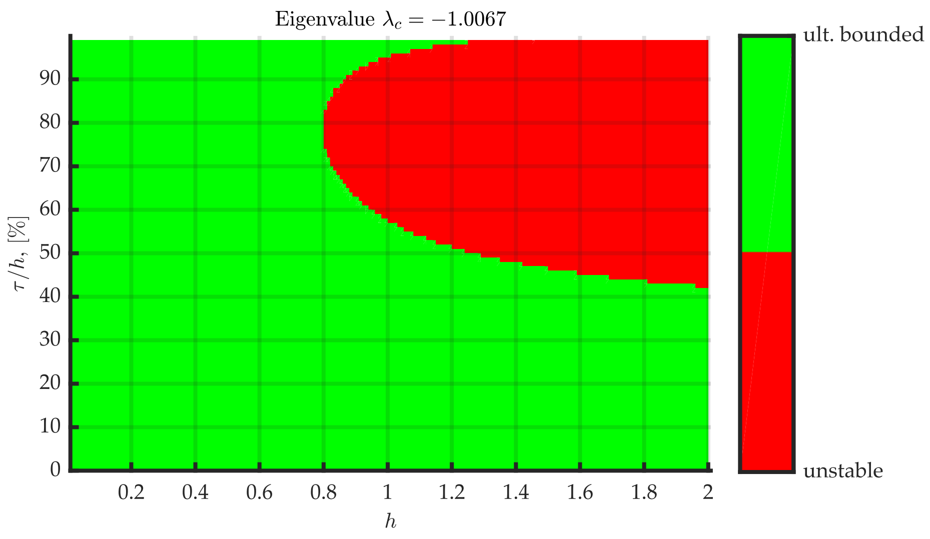
For Type 3 systems no such assumption can be made because using the plant

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1.35 & -1.73 \\ -3.34 & -1.71 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 9.68 \\ -8.85 \end{bmatrix} u$$



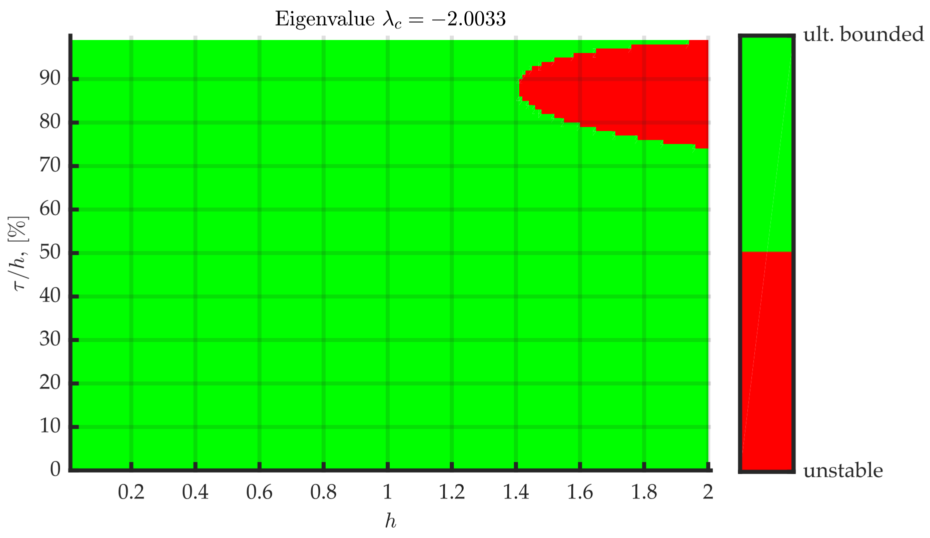
Eigenvalue $\lambda_c = -0.0100$
τ/h, [%]
h
ult. bounded
unstable

(A)



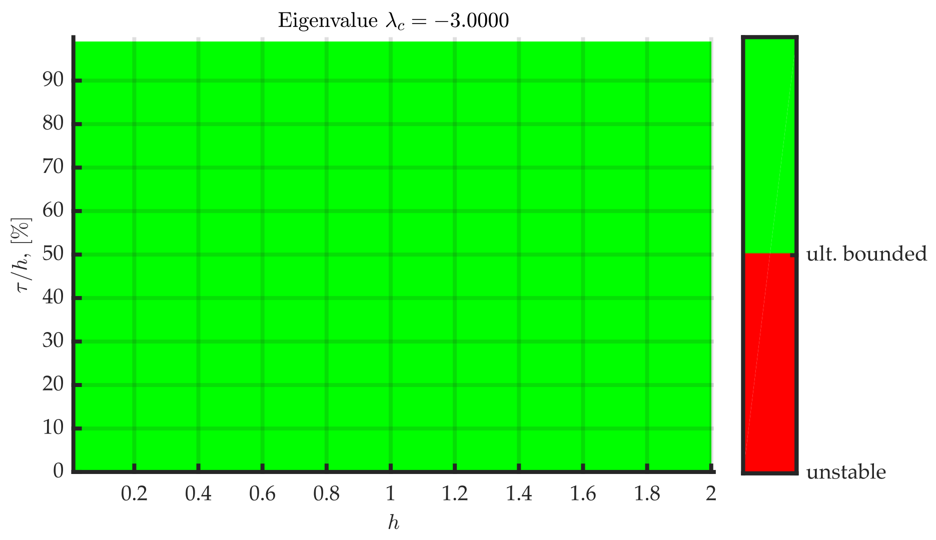
Eigenvalue $\lambda_c = -1.0067$
τ/h, [%]
h
ult. bounded
unstable

(B)



Eigenvalue $\lambda_c = -2.0033$
τ/h, [%]
h
ult. bounded
unstable

(C)



(D)

FIGURE 4.24: Change of the stability area by varying the eigenvalue λ_c by using Type 1a plant systems

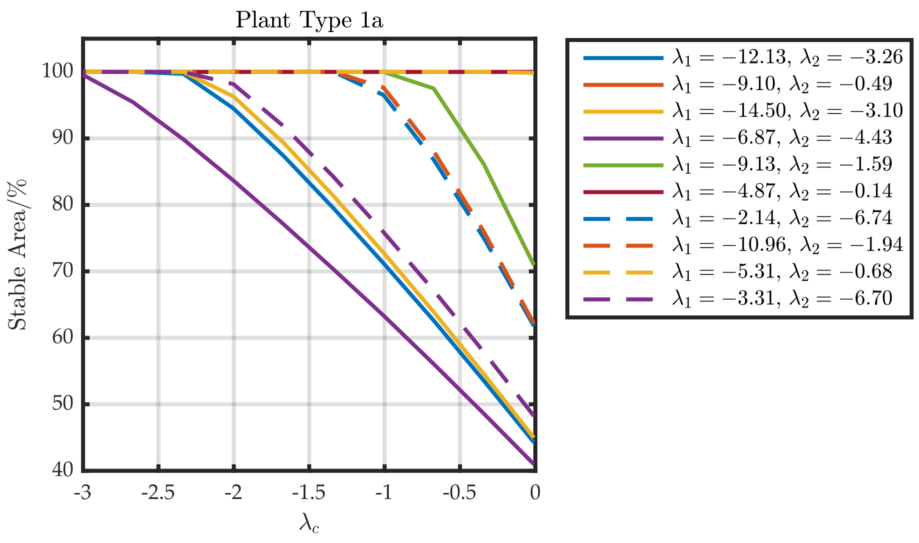
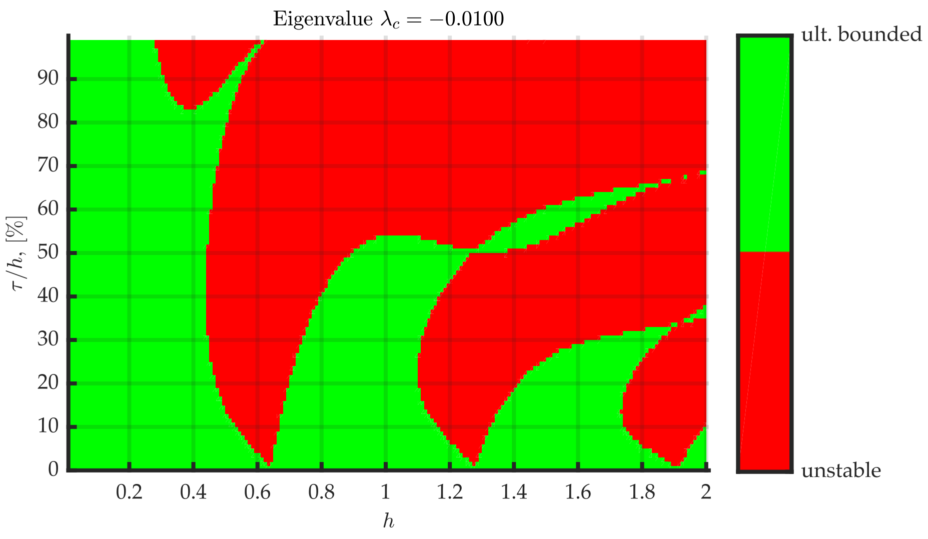
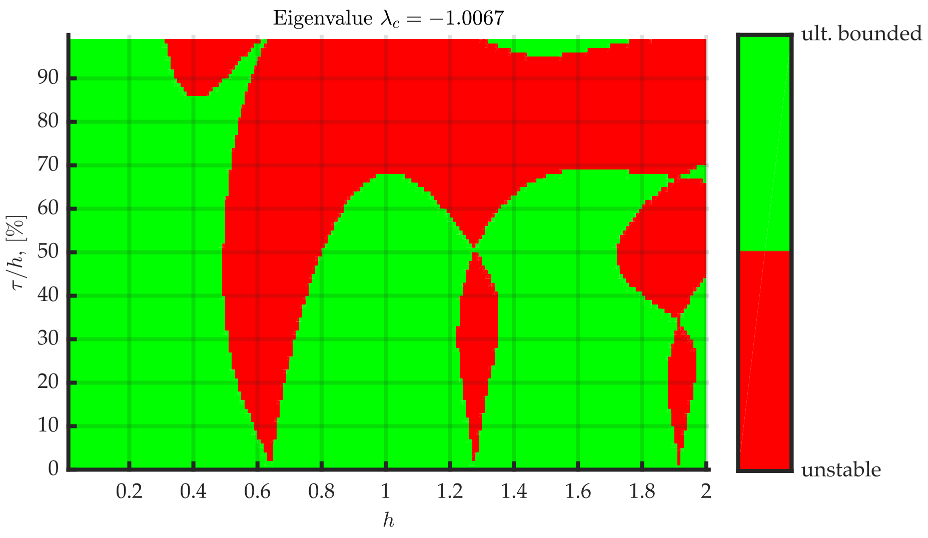


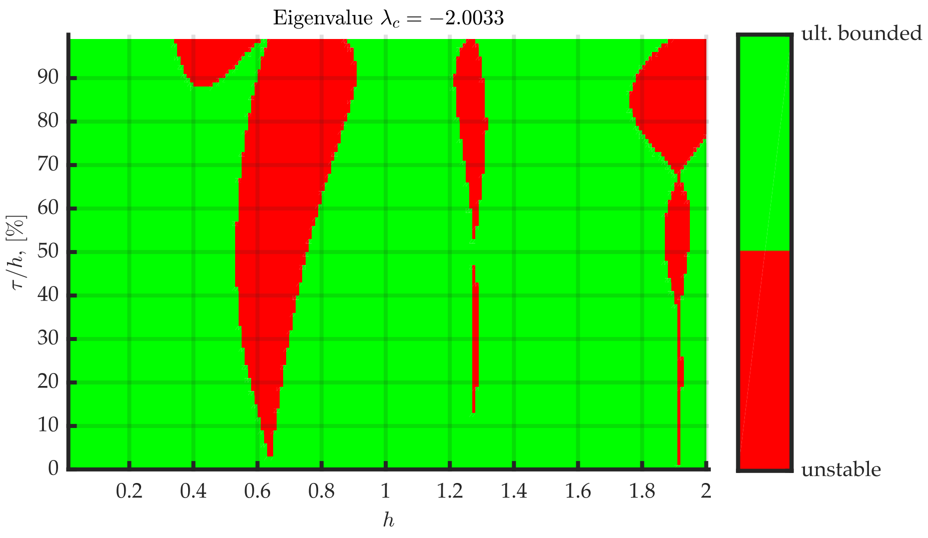
FIGURE 4.25: Dependency of ultimately bounded areas on the eigenvalue λ_c when using different Type 1a plant systems


Eigenvalue $\lambda_c = -0.0100$
ult. bounded
unstable
τ/h, [%]
h
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

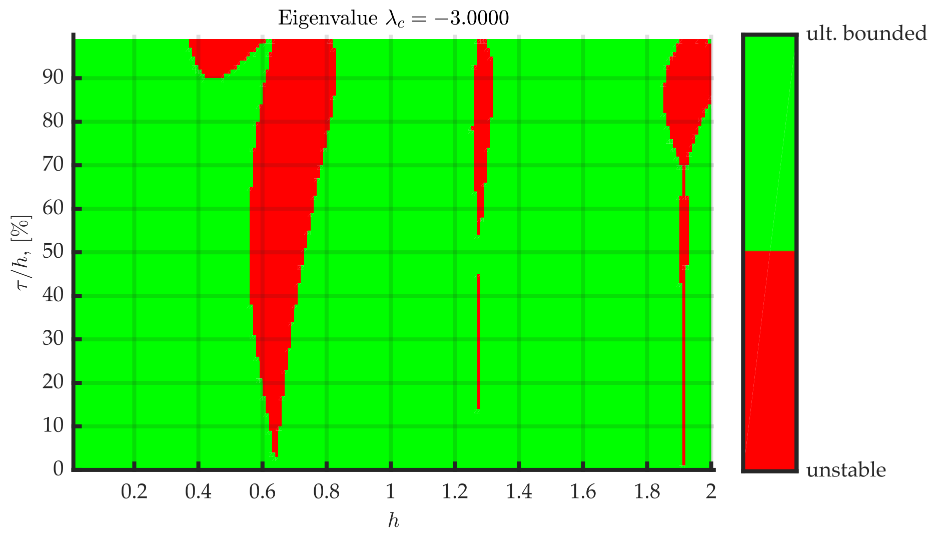
(A)


Eigenvalue $\lambda_c = -1.0067$
ult. bounded
unstable
τ/h, [%]
h

(B)


Eigenvalue $\lambda_c = -2.0033$
ult. bounded
unstable
τ/h, [%]
h

(C)



(D)

FIGURE 4.26: Change of the stability area by varying the eigenvalue λ_c by using Type 1b plant systems

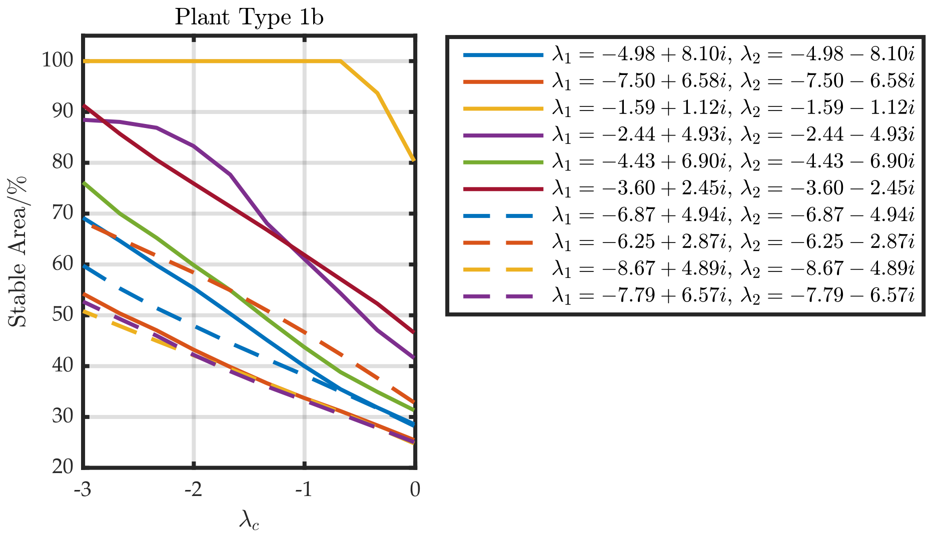
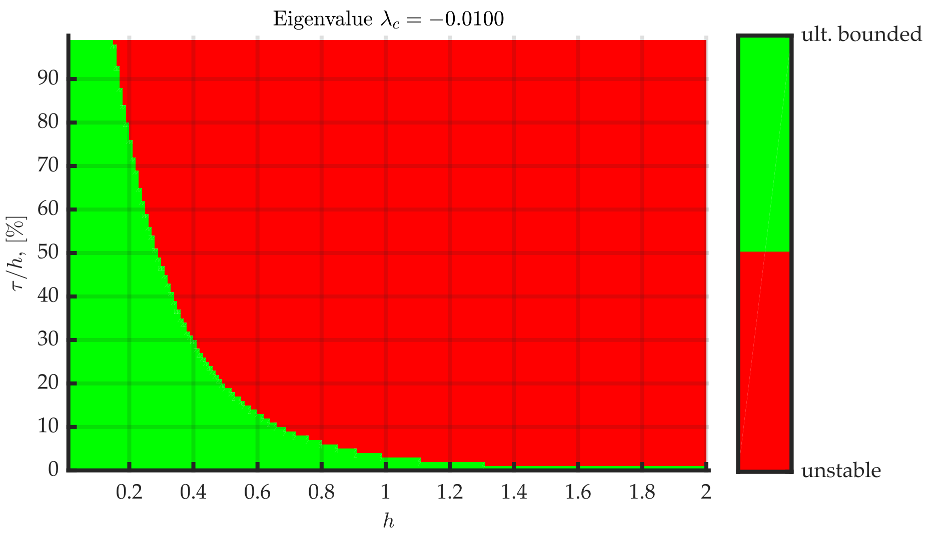
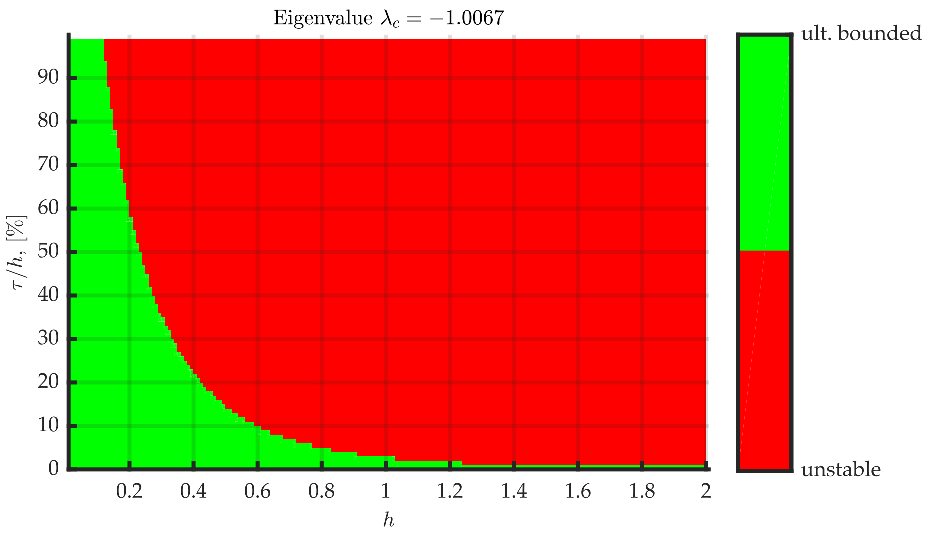


FIGURE 4.27: Dependency of ultimately bounded areas on the eigenvalue λ_c when using different Type 1b plant systems



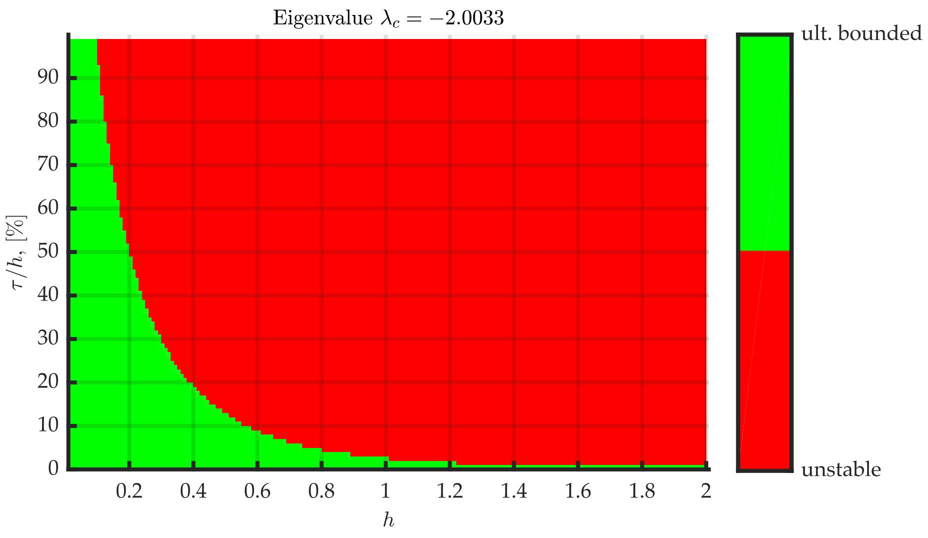
Eigenvalue $\lambda_c = -0.0100$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(A)



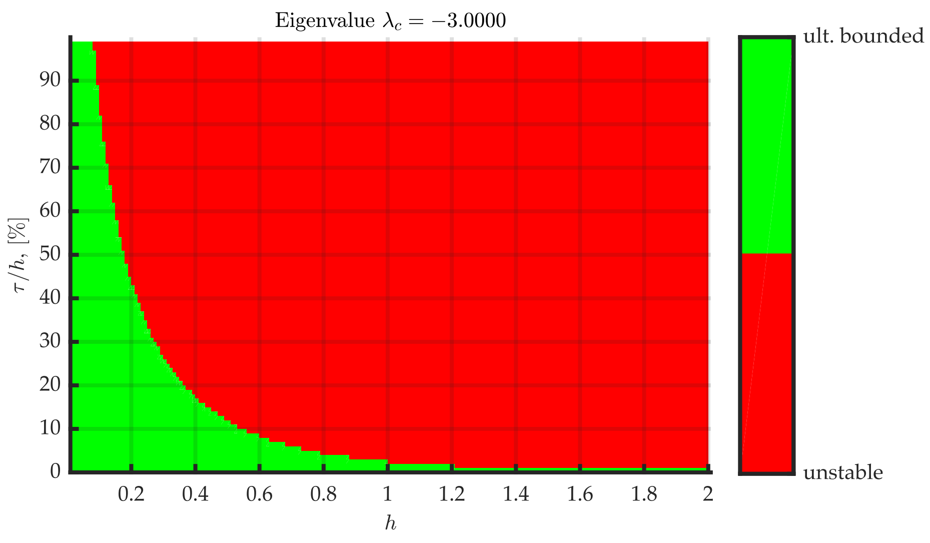
Eigenvalue $\lambda_c = -1.0067$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(B)



Eigenvalue $\lambda_c = -2.0033$
τ/h, [%]
0
10
20
30
40
50
60
70
80
90
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(C)



(D)

FIGURE 4.28: Change of the stability area by varying the eigenvalue λ_c by using Type 2 plant systems

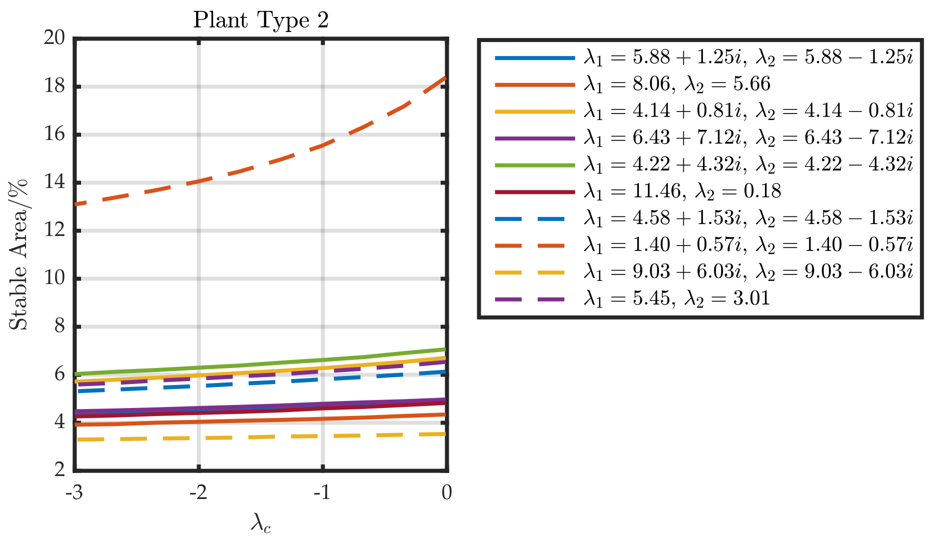


FIGURE 4.29: Dependency of ultimately bounded areas on the eigenvalue λ_c when using different Type 2 plant systems

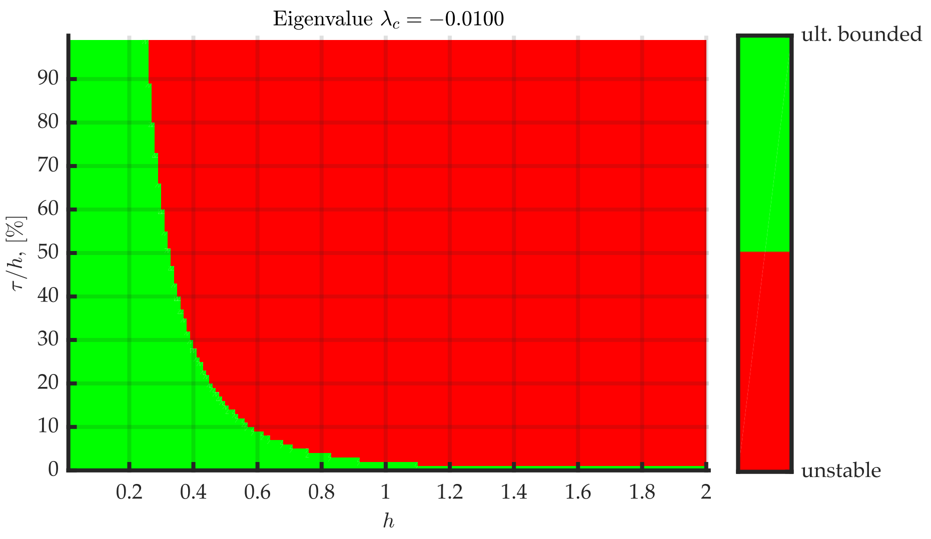
results in a decreasing ultimately bounded area with increasing value of λ_c, which is visible in figure 4.30. When using the plant

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -6.57 & 5.04 \\ 8.14 & -4.28 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 2.55 \\ -0.743 \end{bmatrix} u$$

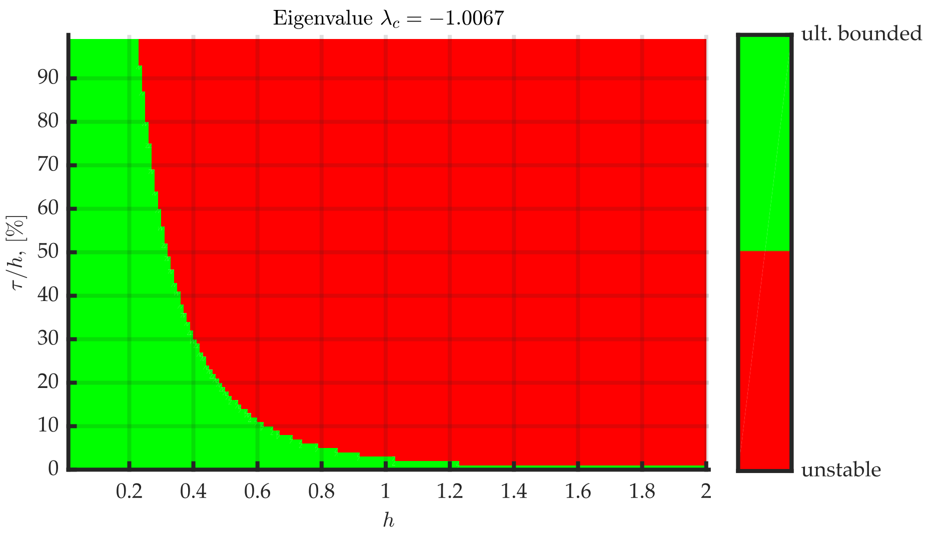
the ultimately bounded area slightly decreases with increasing eigenvalue λ_c. This is depicted in figure 4.31. This different behaviour is also visible in figure 4.32. Here one can see that some lines are monotonically increasing and some monotonically decreasing.

4.7.3 Conclusions

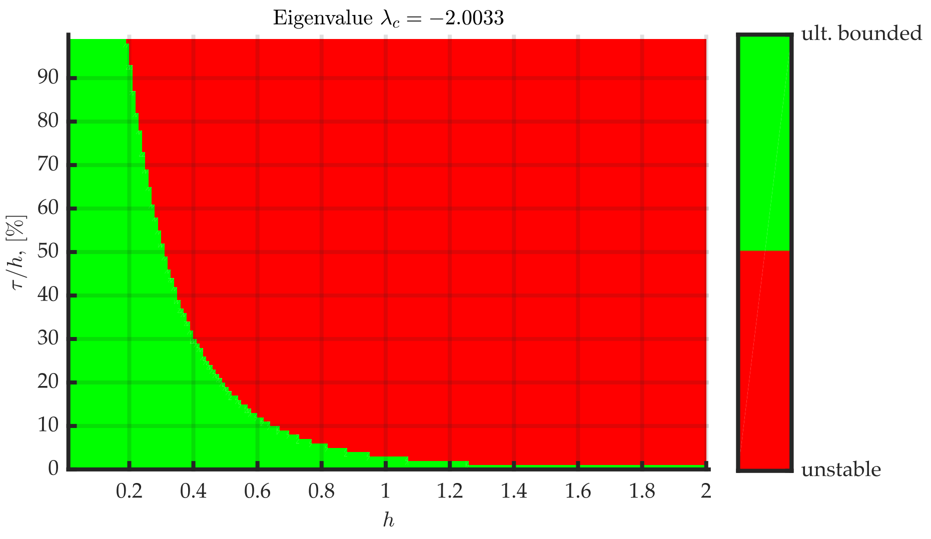
In this section, the influence of the eigenvalue λ_c, which defines the ideal quasi sliding mode dynamic on the stability region, is investigated. It turned out that the separation of the plant systems into three types, as described in section 4.4.1, is again reasonable. For both subtypes, 1a and 1b, the area of ultimate boundedness shrinks with increasing value of the parameter λ_c. Closed loop systems consisting of Type 2 plant systems react contrariwise to Type 1a and 1b plant systems because increasing the eigenvalue λ_c expands the area of ultimate boundedness but the influence is much smaller. The behaviour when using Type 3 plant systems is much less predictable. There are systems with which the ultimately bounded area grows and some with which it shrinks with increasing value of λ_c.


Eigenvalue $\lambda_c = -0.0100$
τ/h, [%]
90
80
70
60
50
40
30
20
10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

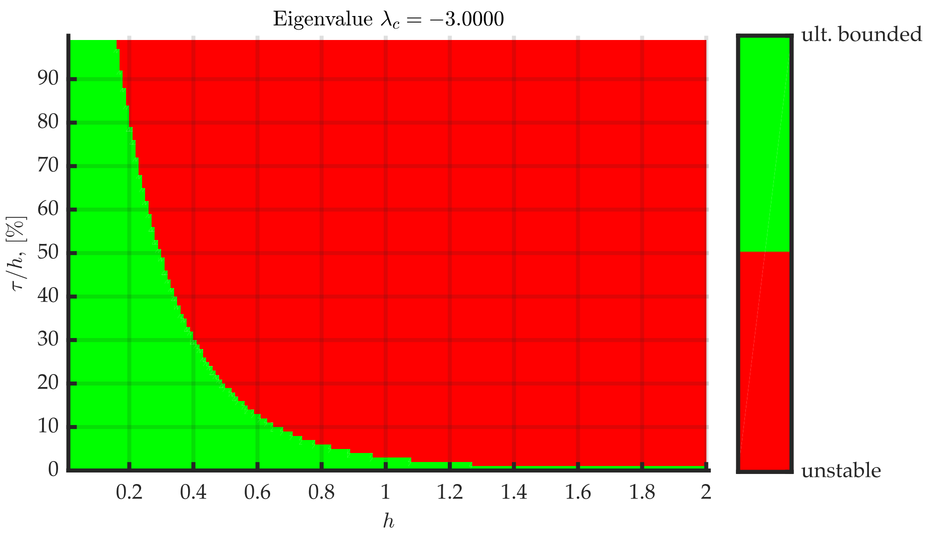
(A)


Eigenvalue $\lambda_c = -1.0067$
τ/h, [%]
90
80
70
60
50
40
30
20
10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(B)

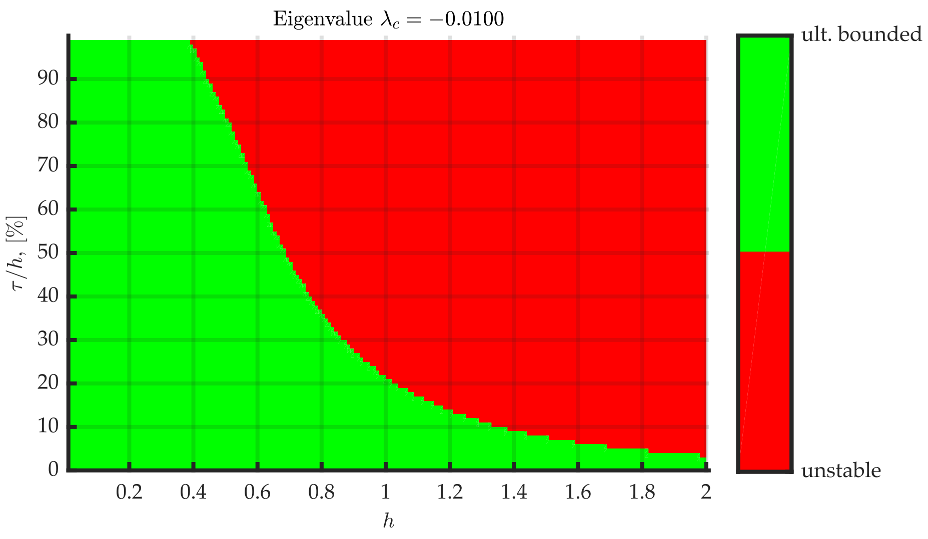

Eigenvalue $\lambda_c = -2.0033$
τ/h, [%]
90
80
70
60
50
40
30
20
10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
h
ult. bounded
unstable

(C)

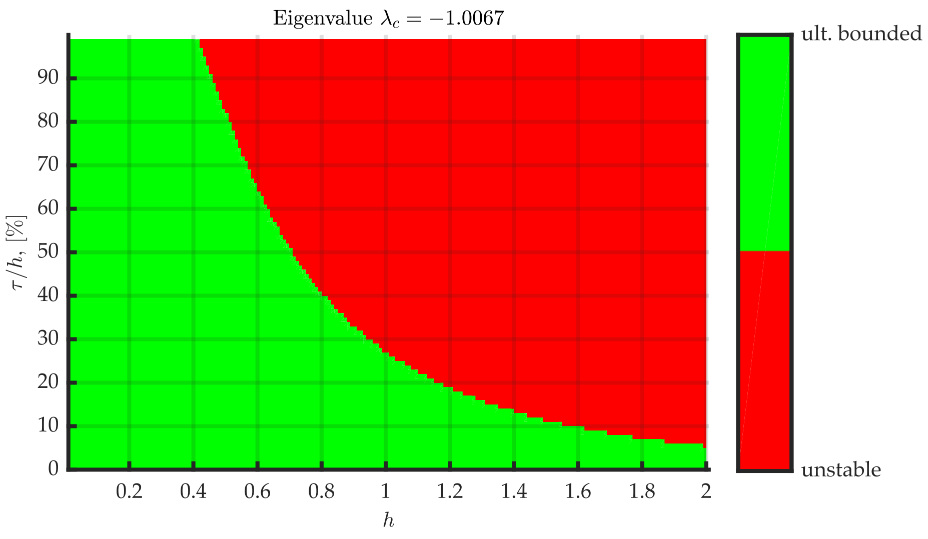


(D)

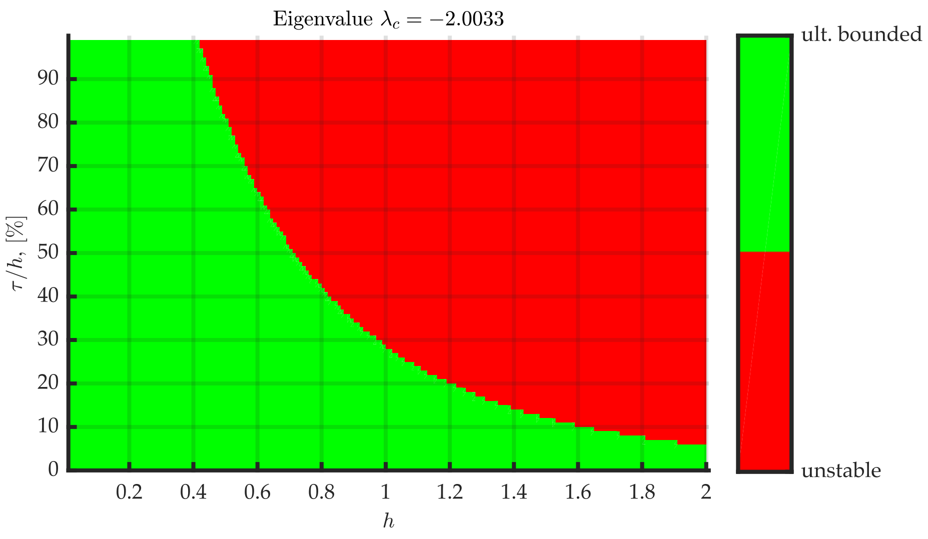
FIGURE 4.30: Change of the stability area by varying the eigenvalue λ_c by using Type 3 plant systems


Eigenvalue $\lambda_c = -0.0100$
τ/h, [%]
h
ult. bounded
unstable

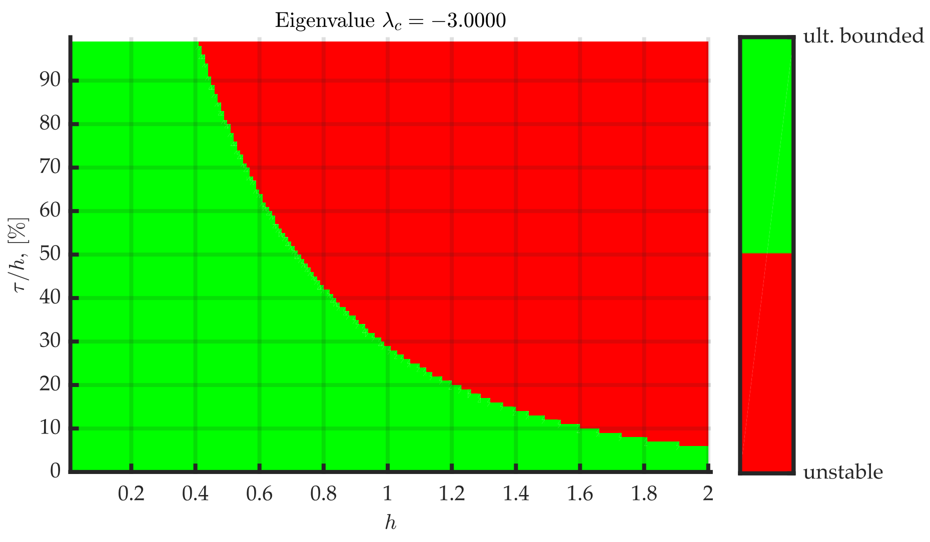
(A)


Eigenvalue $\lambda_c = -1.0067$
τ/h, [%]
h
ult. bounded
unstable

(B)


Eigenvalue $\lambda_c = -2.0033$
τ/h, [%]
h
ult. bounded
unstable

(C)



(D)

FIGURE 4.31: Change of the stability area by varying the eigenvalue λ_c by using Type 3 plant systems

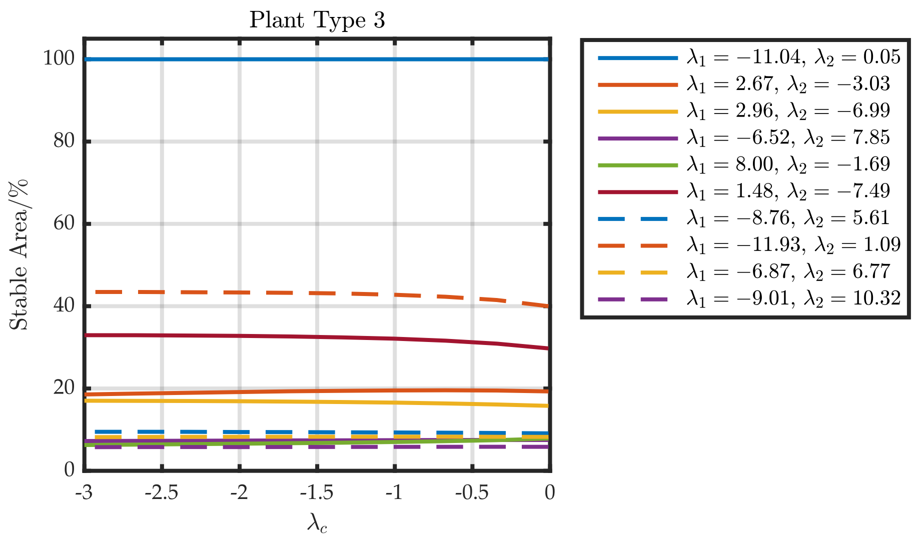


FIGURE 4.32: Dependency of ultimately bounded areas on the eigenvalue λ_c when using different Type 3 plant systems

Chapter 5

Conclusions

In this work the influence of networked induced delays on a discrete time sliding mode controller was investigated. The first step was to derive a model of the networked control structure in which the networked induced delays are explicitly considered. This model mainly consists of discrete time parts. The only continuous time block of this model is the plant. This triggered a demand of introducing a discrete time sliding mode controller, which was derived by using the reaching law approach. A transformation was introduced which makes it possible to use the pole placement technique in order to calculate the parameter vector, which defines the ideal quasi sliding mode.

The centrepiece of this work is the stability analysis of the networked sliding mode control system with explicitly considered networked induced delays. Transforming the closed loop system in order to get a reasonable representation was the first step to derive conditions for ultimate boundedness. After this transformation it was necessary to introduce a new enlarged state vector which lead to a model which consists of linear state equations with bounded nonlinear inputs. This representation made it possible to derive conditions which ensure ultimate boundedness if the assumption holds that the delays are constant.

These conditions are needed to perform simulations, which results in plots known as stability regions. Each value of the sample time h and the relative delay time τ/h is depicted as a pixel in the stability region plot. Generally, the sample time h is applied on the x-axis and the relative delay time τ/h on the y-axis. The colour of the pixel indicates whether the closed loop solutions are ultimately bounded (green colour) or unstable (red colour). This means that for every green point all solutions of the closed loop system are ultimately bounded if the corresponding sample time h and delay time τ are fulfilled.

Several simulations with plant systems of order $n = 2$ were performed in order to investigate the influence of different parameters on these stability regions. It turned out that separating the plant systems into the following three types depending on the eigenvalue configuration is reasonable.

Type 1: The real part of all eigenvalues is negative

This type is again separated into two subtypes.

Type 1a: Real eigenvalues

When using type 1a plant systems, the border between the ultimately bounded and the unstable areas is generally c-shaped. This leads to the phenomenon that there are sample times h with which it is possible to receive ultimate boundedness by increasing the delay time τ. By using this type of plant system, the ultimately bounded area grows with larger values of the relative linear reaching parameter q_{rel} or with smaller values of the eigenvalue λ_c, which specifies the dynamic for ideal quasi sliding mode.

Type 1b: Conjugate complex eigenvalues
The same reaction to the parameter is observed when using type 1b plant systems. The only difference is the shape of the border because arbitrary shapes are received for the stability regions. So the phenomenon previously discussed could also be identified by using type 1b plant systems but also a second phenomenon was discovered. The arbitrary shape makes it possible that for some plants the performance in terms of area of ultimate boundedness could be improved by increasing the sample time h of the control loop.

Type 2: The real part of all eigenvalues is greater than zero

The reaction on parameter changes is completely contrariwise when using type 2 plant systems because increasing the relative linear reaching factor q_{rel} leads to a smaller ultimately bounded area in this case. Also, the influence of the eigenvalue λ_c is different because increasing this parameter leads to a larger ultimately bounded area when using type 2 plants. The border between the ultimately bounded area and the unstable area in general is exponentially shaped when using type 2 plant systems. Especially if the eigenvalues of the system matrix of the plant are conjugate complex, peaks are observed which are attached to the exponential shape.

Type 3: The real part of one eigenvalue is negative, the other is greater than zero

The exponential shape of the border between the ultimately bounded area and the unstable area is exponentially shaped when using type 3 plant systems. The influence of the two parameters (q_{rel} and λ_c) could not be determined because the ultimately bounded area grows with increasing parameters for some type 3 plant systems and shrinks for others. No reaction to the parameters could be predicted when using this type of plant systems. Table 5.1 shows a summary of the previously discussed influence of the parameters as well as the shape of the border for the different plant system types.

TABLE 5.1: Influence of the parameters on ultimately bounded area of the stability regions when using the different plant types

Type	Border Shape	Incresing q_{rel}	Increasing λ_c
Type 1a	C-shaped	Larger	Smaller
Type 1b	arbitrary	Larger	Smaller
Type 2	exponentially + "spikes"	Smaller	Larger
Type 3	exponentially	undefined	undefined

The analysis performed in this work guarantees ultimate boundedness. It is ensured that all solutions converge to a bounded set in the state space.

The size of this bounded set was not investigated, which would be very interesting for further research in order to ensure a desired accuracy. The big challenge to find boundaries for this set is that the nonlinear part of the control law influences these boundaries. Depending on the desired precision of the boundaries the analysis becomes more complex.

As the used sliding mode controller is capable to handle bounded matched perturbations, it would be very interesting how this property is influenced by delays. This analysis is also challenging because it again requires to treat the nonlinear part of the control law.

Also considering time varying delays should be a future step as this is very often present in practice. This makes it no longer possible to check for ultimate boundedness by using the eigenvalues. It might be necessary to use some Ljapunov approaches.

The investigation of local stability and the corresponding convergence area with also considering the nonlinear part of the control law in the presence of delays should also be treated in future research. It is highly probable that Ljapunov functions are required to perform this investigation. Finding such functions is generally very difficult.

As one can see, this work offers a basis to a lot of future research.

Bibliography

[1] F.-Y. Wang and D. Liu, *Networked Control Systems*. Springer, 2008.

[2] B. Azimi-Sadjadi, "Stability of networked control systems in the presence of packet losses", in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 1, Dec. 2003, pp. 676–681.

[3] S. Hu and W.-Y. Yan, "Brief paper: Stability robustness of networked control systems with respect to packet loss", *Automatica*, vol. 43, no. 7, pp. 1243–1248, Jul. 2007.

[4] Z. Wang, F. Yang, D. W. C. Ho, and X. Liu, "Robust control for networked systems with random packet losses", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 4, pp. 916–924, Aug. 2007.

[5] A. Ferrara, G. P. Incremona, and V. Stocchetti, "Networked sliding mode control with chattering alleviation", in *53rd IEEE Conference on Decision and Control*, Dec. 2014, pp. 5542–5547.

[6] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems", *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2030–2042, Sep. 2010.

[7] G. P. Incremona and A. Ferrara, "Adaptive model-based event-triggered sliding mode control", *International Journal of Adaptive Control and Signal Processing*, n/a–n/a, 2016, acs.2665.

[8] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems", *IEEE Control Systems*, vol. 21, no. 1, pp. 84–99, Feb. 2001.

[9] L. Zhang, Y. Shi, T. Chen, and B. Huang, "A new method for stabilization of networked control systems with random delays", *IEEE Transactions on Automatic Control*, vol. 50, no. 8, pp. 1177–1181, Aug. 2005.

[10] H. Shousong and Z. Qixin, "Stochastic optimal control and analysis of stability of networked control systems with long delay", *Automatica*, vol. 39, no. 11, pp. 1877–1884, 2003.

[11] J. D. Spragins, J. L. Hammond, and K. Pawlikowski, *Telecommunications: Protocols and Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1991.

[12] B. M. Chen, *Robust and H∞ Control*. Springer London, 2000.

[13] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant, *Sliding mode control and observation*. Springer.

[14] W. Gao, Y. Wang, and A. Homaifa, "Discrete-time variable structure control systems", *IEEE Transactions on Industrial Electronics*, vol. 42, no. 2, pp. 117–122, Apr. 1995.

[15] H. Lee and V. I. Utkin, “Chattering suppression methods in sliding mode control systems”, *Annual Reviews in Control*, vol. 31, no. 2, pp. 179–188, 2007.

[16] M.-L. Tseng and M.-S. Chen, “Chattering reduction of sliding mode control by low-pass filtering the control signal”, *Asian Journal of Control*, vol. 12, no. 3, pp. 392–398, 2010.

[17] D. Efimov, A. Polyakov, L. Fridman, W. Perruquetti, and J.-P. Richard, “Delayed sliding mode control”, *Automatica*, vol. 64, pp. 37–43, 2016.

[18] J. Nilsson, “Real-time control systems with delays”, PhD thesis, 1998.

[19] V. Utkin, “Variable structure systems with sliding modes”, *IEEE Transactions on Automatic Control*, vol. 22, no. 2, pp. 212–222, Apr. 1977.

[20] V. I. Utkin, *Sliding Modes in Control and Optimization*. Springer Science, 2013.

[21] C. Edwards and S. Spurgeon, *Sliding mode control: Theory and applications*. CRC Press, 1998.

[22] Y. Dote and R. Hoft, “Microprocessor based sliding mode controller for dc motor drives”, in *IEEE/IAS Annual Meeting*, 1980, pp. 641–645.

[23] C. Milosavljevic, “General conditions for the existence of a quasi-sliding mode on the switching hyperplane in discrete variable structure systems”, *Automation and Remote Control*, vol. 46, no. 3, pp. 307–314, 1985.

[24] S. Sarpturk, Y. Istefanopulos, and O. Kaynak, “On the stability of discrete-time sliding mode control systems”, *IEEE Transactions on Automatic Control*, vol. 32, no. 10, pp. 930–932, Oct. 1987.

[25] K. Furuta, “Sliding mode control of a discrete system”, *Syst. Control Lett.*, vol. 14, no. 2, pp. 145–152, Feb. 1990.

[26] J. Y. Hung, W. Gao, and J. C. Hung, “Variable structure control: A survey”, *IEEE Transactions on Industrial Electronics*, vol. 40, no. 1, pp. 2–22, Feb. 1993.

[27] W. Gao and J. C. Hung, “Variable structure control of nonlinear systems: A new approach”, *IEEE Transactions on Industrial Electronics*, vol. 40, no. 1, pp. 45–55, Feb. 1993.

[28] C. J. Fallaha, M. Saad, H. Y. Kanaan, and K. Al-Haddad, “Sliding-mode robot control with exponential reaching law”, *IEEE Transactions on Industrial Electronics*, vol. 58, no. 2, pp. 600–610, Feb. 2011.

[29] A. Bartoszewicz, “Discrete-time quasi-sliding-mode control strategies”, *IEEE Transactions on Industrial Electronics*, vol. 45, no. 4, pp. 633–637, Aug. 1998.

[30] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[31] J. Kautsky, N. K. Nichols, and P. Van Dooren, “Robust pole assignment in linear state feedback”, *International Journal of Control*, vol. 41, no. 5, pp. 1129–1155, 1985.

[32] J. Kato, *Stability of Motion of Nonautonomous Systems (Methods of Limiting Equations)*. CRC Press, 1996, vol. 3.