NTNU
Norwegian University of Science and Technology

Higher-Order Sliding Mode Control

Stability analysis and application to underwater snake robots

Ida-Louise Garmann Borlaug

Master of Science in Cybernetics and Robotics
Submission date: June 2017
Supervisor: Jan Tommy Gravdahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

MASTEROPPGAVE

Kandidatens navn:	Ida-Louise G. Borlaug
Fag:	Teknisk Kybernetikk
Oppgavens tittel:	**Higher-order sliding mode control: stability analysis and application to underwater snake robots**

Background

Sliding mode (SM) control is a well-known nonlinear control design method. First-order SM is widely applied to control of mechanical systems, but second and higher-order algorithms can also be used. In this project, second and higher-order sliding mode control is to be studied together with a sliding mode observer (SMO).

Assignment:

1. Analyse the stability properties for the closed-loop dynamics for a general second-order system in cascade with second and higher-order SMC algorithms with a state SMO.
2. Apply the second and higher-order algorithms with a state SMO to a second-order test-system. Perform analysis on the tracking control problem, simulations and compare the results.
3. Test the SM algorithms with a state SMO, for control of an underwater swimming manipulator with thrusters and analyse the tracking control problem. Use the model presented in [1].
4. Based on the above work, write and submit an article for the SWARM conference in Kyoto, Japan 2017.

Besvarelsen leveres innen: 19. Juni 2017

[1] Sverdrup-Thygeson, J., Kelasidi, E., Pettersen, K. Y., Gravdahl, J. T., Sept. 13-16, 2016. Modeling of underwater swimming manipulators. In: Proc. 10th IFAC Conference on Control Applications in Marine Systems. Vol. 49. Trondheim, Norway, pp. 81–88.

Trondheim, Spring, 2017

Jan Tommy Gravdahl

Tommy Gravdahl
Faglærer

Abstract

In this thesis sliding mode control, sliding mode observers and stability theory regarding cascaded systems and finite-time stable properties are presented. The stability properties for the closed-loop dynamics for a general second-order system in cascade with a second or higher-order sliding mode control algorithm with a state sliding mode observer, are analysed. The second-order algorithm is the super-twisting algorithm, and the higher-order sliding mode control algorithms are the nested and the quasi-continuous third-order sliding mode control algorithm. The state observer is included because the variables needed for the control algorithms are not always available for measurement. To see how the state observer affects the performance and control abilities of the algorithms, a tracking control law where the state observer was not used, is also tested. It is shown that the error dynamics for the test system, i.e. the mass-spring-damper system, and that the error dynamics for an underwater swimming manipulator, a snake-like, multi-articulated, underwater robot equipped with thrusters, fits the assumptions made for the error dynamics for the general system, which means that the stability analysis conducted for the general system also holds for both those systems. The closed-loop dynamics for the general system is proven uniformly globally asymptotically stable in all cases except one.

A simulation study is performed on both the systems to verify the applicability of the proposed control laws. The control objective is to make the state trajectories follow a pre-defined path. The mass-spring-damper system is used to get a better understanding of the algorithms and to easily observe what the advantages and disadvantages are with the different algorithms. The algorithm that gave the best result was the second-order algorithm, i.e. the super-twisting algorithm, as it gave the smallest errors in all cases and the smoothest control input. The higher-order algorithms, i.e. the nested third-order and quasi-continuous SMC algorithms, also gave small errors, but they had chattering in the control input. It was possible to get a smooth control input with the quasi-continuous SMC algorithm, but the position error was then much larger.

To the author's knowledge, sliding mode control algorithms have only been tested previously on an underwater swimming manipulator by the author in the project assignment conducted in the fall 2016, Borlaug (2016). In that project only first-order and second-order algorithms were tested. Therefore, it was interesting to see if the higher-order algorithms performed better than a second-order algorithm, as the second-order algorithms gave the best result in the project assignment, and a regular PD controller. The second-order algorithm also gave the best result here, and the PD controller did not compare to the SMC algorithms. The quasi-continuous SMC algorithm was the better of the two HOSM controllers.

Sammendrag

Denne avhandlingen presenterer ”sliding mode” regulatorer, ”sliding mode” observere og stabilitetsteori knyttet til systemer i kaskade og “finite-time” stabilitetsegenskaper. Stabilitetsegenskapene for lukket sløyfe dynamikken for et generelt andre-ordens system i kaskade med andre- eller høyere-ordens ”sliding mode” regulatorer og en tilstands-”sliding mode” observer er analysert. Den undersøkte andre-ordens algoritmen er “super-twisting” algoritmen, og algoritmene av høyere-orden er “nested” og ”quasi-continuous” tredje-ordens ”sliding mode” regulatorene. Tilstandsobserveren ble inkludert fordi variablene som behøves ikke alltid er tilgjengelige eller målbare. For å undersøke hvordan tilstandsobserveren påvirker ytelsen og reguleringsevnen til algoritmene ble en reguleringslov hvor tilstandsobserverenen ikke brukes også testet. Det blir vist at feildynamikken for testsystemet, et masse-fjær-demper system, og at feildynamikken til en undervanns svømmemanipulator passer til antagelsene som er gjort for feildynamikken til det generelle systemet, noe som betyr at stabilitetsanalysen som er gjort for det generelle systemet også holder for disse to systemene. Lukket sløyfe dynamikken for det generelle systemet blir bevist uniformt, globalt, asymptotisk stabilt i alle tilfeller, med unntak av ett.

Det blir gjort en simuleringsundersøkelse på begge testsystemene for å verifisere anvendbarheten til de foreslåtte reguleringslovene. Reguleringsmålet er å få tilstandsbanen til å følge en forhåndsdefinert bane. Masse-fjær-demper systemet blir brukt til å få en bedre forståelse for de respektive algoritmene og enkelt kunne observere fordelene og ulempene med de. Algoritmen som ga best resultat var ”super-twisting” algoritmen, ettersom den førte til de minste feilvariablene i alle tilfeller, samt glattest pådrag. Algoritmene av høyere orden, “nested” og ”quasi-continuous” tredje-ordens ”sliding mode” regulatorene, ga også små feilvariabler, men hadde ”chattering” i pådraget. Det var mulig å få et glatt pådrag med ”quasi-continuous sliding mode” algoritmen, men posisjonsfeilen ble da mye større.

Såvidt forfatteren vet, har ”sliding mode” regulatorer bare blitt testet på en undervanns smømmemanipulator av forfatteren selv, under fordypningsprosjektet utført høsten 2016, Borlaug (2016). I dette prosjektet ble bare første- og andre-ordens algoritmer testet. Det var derfor interessant å se om algoritmene av høyere-orden ga bedre ytelse enn andre-ordens algoritmen som ga best ytelse under fordypningsprosjektet, samt en PD-regulator. Andre-ordens algoritmen ga best resultat også her, og PD-regulatoren kunne ikke sammenlignes med ”sliding mode” regulatorene. Av de to høyere-ordens ”sliding mode” regulatorene var ”quasi continuous” regulatoren best.

Preface

This master thesis is submitted as a part of the requirements for the M.Sc. degree within the field of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU). The thesis has been written for the department of Engineering Cybernetics at NTNU, and was developed together with Professor Jan Tommy Gravdahl. The contributions of this thesis are within the area of non-linear control, and can be seen as a study to investigate sliding mode control in general and the possibilities of using sliding mode control to control an underwater swimming manipulator.

I would like to thank my supervisor, Professor Jan Tommy Gravdahl at the Department of Engineering Cybernetics for his guidance and feedback during the work. I would also like to thank Senior researcher Antonio Loria at CNRS (The National Center for Scientific Research) for feedback and discussion regarding the stability analysis, and Jørgen Sverdrup-Thygeson, PhD candidate at the Department of Engineering Cybernetics, for discussion regarding the model of the underwater swimming manipulator. Finally I would like to thank my family for their support and encouragement, especially Andreas Bognøy for corrective reading and the invaluable support during the most difficult stages of working on this thesis.

Trondheim, June 19, 2017

Ida-Louise G. Borlaug

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ALE	Algebraic Lyapunov Equation
AS	Asymptotically stable
BIBO	Bounded-input, bounded-output
CM	Centre of mass
FTS	Finite-time stable
GAS	Globally asymptotically stable
HOSM	Higher-order sliding mode
HOSMO	Higher-order sliding mode observer
LOS	Line-of-sight
MSDS	Mass-spring-damper system
SISO	Single-input-single-output
SMC	Sliding mode control
SMO	Sliding mode observer
STA	Super-twisting algorithm
UAS	Uniformly asymptotically stable
UGAS	Uniformly globally asymptotically stable
UGB	Uniformly globally bounded
USM	Underwater swimming manipulator
USR	Underwater snake robot
VSS	Variable structure systems

List of Symbols

Symbol	Description	Vector
σ	Sliding surface	
u	Control input	
x	State variable	
$\hat{x}$	Estimated state variable	
e	State observer error	
m	Mass [kg]	
c	Damping coefficient [N s/m]	
k	Spring constant [N/m]	
$d(t)$	Time-varying disturbance	
x_{des}	Desired position mass-spring-damper system	
$\tilde{x}$	State error MSDS	
z	Differentiator variables	
n	Number of links	
l_i	The half length of a link	$L \in R^{n\times n}$
m_i	Mass of each link	$M \in R^{n\times n}$
j_i	Moment of inertia of each link	$J \in R^{n\times n}$
ψ_i	Angle between link i and the global x axis	$\psi \in R^n$
q_i	Angle of joint i	$q \in R^{n-1}$
(x_i, y_i)	Global coordinates of the CM of link i	$X, Y \in R^n$
(p_x, p_y)	Global coordinates of the CM of the robot	$p_{CM} \in R^2$
$(f_{x,i}, y_{y,i})$	Fluid force on link i	$f_x, f_y \in R^n$
$(f_{px,i}, y_{py,i})$	Added force on link i	$f_{px}, f_{py} \in R^n$
$(h_{x,i}, h_{y,i})$	Joint constraint force on link i from link $i+1$	$h_x, h_y \in R^{n-1}$
$-(h_{x,i-1}, h_{y,i-1})$	Joint constraint force on link i from link $i-1$	$h_x, h_y \in R^{n-1}$
$(\tilde{p}_x, \tilde{p}_y)$	State error USM	$\tilde{p} \in R^2$
$(\hat{p}_x, \hat{p}_y)$	Estimated state variable USM	$\hat{p} \in R^2$

Chapter 1

Introduction

The Sliding Mode Control (SMC) approach is a well-known non-linear control design method that is recognized as a powerful tool to robustly control systems with uncertainties. The advantage of SMC is that it eliminates the need for exact modelling, because it is robust against parametric uncertainty, external disturbances and modelling error, Hung et al. (1993). The research in this area was initiated in the former Soviet Union in the 1950's, and since then there has been done a great deal of research in the field. Fist-order SMC is now widely applied to control of mechanical systems, but second and higher-order algorithms can also be used.

This thesis is an attempt to further test higher-order sliding mode (HOSM) control algorithms studied in my project assignment conducted in the fall in 2016. The project report can be found in the . zip folder attached to this thesis, and will hereafter be cited as, Borlaug (2016). In the project report an in-depth study of sliding mode control is presented. It explains sliding mode control in general and several first, second and higher-order sliding mode control algorithms in detail. Based on the literature study some SMC algorithms were chosen for further research. The algorithms that were chosen were the first-order relay controller, Hung et al. (1993); the first-order saturation controller, Hung et al. (1993); the super twisting algorithm (STA), Levant (1993); and the STA with adaptive gains, Shtessel et al. (2010). The HOSM controllers were not investigated, because to be able to implement them a differentiator or observer need to be used to estimate the derivatives of the sliding surface. The first and second-order algorithms were tested on two different systems: a test system and an underwater swimming manipulator (USM), a snake-like, multi-articulated, underwater robot equipped with thrusters. The first-order relay controller had large problems with chattering. Chattering is the high-frequency switching of the control signals commonly associated with SMC. Chattering was not an issue for the saturation control, but since sliding mode does not exist inside the boundary layer the effectiveness of the controller is challenged when parasitic dynamics are considered, Young et al. (1999). The STA was therefore the one that gave the best result, as it had a smooth control input and was most robust against modelling errors and disturbances. The largest problem with the STA is that it only works with bounded perturbations, and therefore a conservative upper bound had to be used when designing the controller to ensure that sliding is maintained. To eliminate this problem adaptive STA can be used, since the gains can then adapt to a level where they are as small as possible but still guarantee that sliding is maintained. The STA with adaptive gains was therefore the controller most suited for practical use, Borlaug (2016).

In this thesis some off the HOSM algorithms that were not tested in the project assignment will be tested. They need differentiators or observers to estimate the derivatives of the sliding surface, therefore relevant sliding mode observers (SMO) are presented. A higher-order sliding mode observer (HOSMO) will also be used to estimate the states of the systems, because both the position and the velocity might not always be available for measurement. To be able to see how the state observer affects the performance and control abilities of the algorithms, a tracking control law where the state observer is not used will also be tested. In Borlaug (2016) such a control law was proposed for the relay controller, the saturation controller, the STA and the STA with adaptive gains. Since the STA gave the best result in the project assignment it will be used to compare the performance of the HOSM controllers. To make the comparison as fair as possible, the STA with adaptive gains will not be used, as adaptive gains will not be used for the HOSM controllers. The stability of the overall-closed-loop dynamics for a general system will be analysed, when the STA and when the HOSM controllers are used for tracking control in cascade with a state observer. The algorithms will be tested on the same system as in the project assignment, i.e. on one test system and an underwater swimming manipulator. The test system is a mass-spring-damper system (MSDS), it will be used to illustrate the advantages and disadvantages with the different SMC algorithms.

An USM is an underwater snake robot (USR) equipped with thrusters, Sverdrup-Thygeson et al. (2016). The USM has a complex control design problem. This is because the USM is subject to hydrodynamic and hydrostatic parameter uncertainties, uncertain thruster characteristics, unknown disturbances, and un-modelled dynamic effects, e.g. thruster dynamics and coupling forces caused by joint motion. SMC is a robust and versatile non-linear control approach, and it will therefore be shown in this thesis that it is well suited for control of USMs. For underwater vehicles, in general, some important contributions are given in Antonelli and Chiaverini (1998), Fossen (1991), Fossen and Sagatun (1991), Cristi et al. (1990), Dannigan and Russell (1998) and Soylu et al. (2008). In Antonelli and Chiaverini (1998), a singularity-free SMC approach, inspired by Fjellstad and Fossen (1994), is used for set-point regulation of an underwater robot with uncertainties in the hydrodynamic parameters. In Fossen (1991) and Fossen and Sagatun (1991), SMC is employed to cope with multiplicative uncertainty in the thruster configuration matrix. The combination of sliding mode and adaptive control is studied in Fossen (1991), Fossen and Sagatun (1991) and Soylu et al. (2008). In particular, in Soylu et al. (2008), sliding mode control is combined with adaptive PID controller gains and an adaptive update of the upper bound on the disturbances and the parameter uncertainties. SMC is also applicable to deal with linearisation errors, Cristi et al. (1990), and the coupling effects between an underwater vehicle and an attached manipulator arm, Dannigan and Russell (1998). Sliding mode techniques have been applied to land-based snake robots in Rezapour et al. (2014) to achieve robust tracking of a desired gait pattern and under-actuated straight line path following, Borlaug et al.

(2017). To the author's best knowledge, sliding mode control algorithms have only been tested previously on an USM by the author in the project assignment conducted during the fall 2016, Borlaug (2016).

This thesis is divided into 6 chapters. Chapter 2 presents theory related to sliding mode controllers, sliding mode observers, stability theory to investigate cascaded systems and finite-time stability (FTS) theory. Chapter 3 presents a stability analysis for the closed-loop dynamics for a general system in cascade with the sliding mode control algorithms and observers that have been presented in Chapter 2. The test system, i.e. the MSDS, its control input design, analysis of its error dynamics, implementation and the results from each SMC algorithm are presented in Chapter 4. Chapter 5 presents the USM, its control input design, analysis of its error dynamics, implementation and the results from each SMC algorithm. In Chapter 6 the results for each algorithm are discussed and compared. Conclusion and further work is found in Chapter 7.

Chapter 2

Theory

In this chapter, sliding mode controllers (SMC), sliding mode observers (SMO) and stability theory to investigate cascaded systems will be presented. Finite-time stability, which is an important property of SMC algorithms, will also be presented in the stability section. The theory presented in Section 2.1 is mainly taken from the literature study presented in Borlaug (2016).

2.1 Sliding mode control

Sliding Mode Control (SMC) systems are designed to drive the system state trajectories onto a particular surface in the state space, named sliding surface σ, in finite time. Once the sliding surface is reached, SMC keeps the states on the close neighbourhood of the sliding surface for all future time. Hence SMC is a two part controller design. The first part involves the design of a sliding surface so that the sliding motion satisfies design specifications. The second is concerned with the selection of a control law that will make the sliding surface attractive to the system state, Utkin (1977). The first part is called the reaching phase, and the second part is called the sliding phase. The state-feedback control law is not a continuous function of time. Instead, it can switch from one continuous structure to another based on the current position in the state space. Hence, sliding mode control is a variable structure control (VSS) method.

The largest problem with SMC is chattering. Chattering is the high-frequency switching of the control signals. In ideal SMC the state trajectory should reach the sliding surface $\sigma = 0$ in finite time and stay on it forever. But as the controller cannot be switched infinitely fast from one value to another chattering appears as a high-frequency oscillation around the desired equilibrium point. The reason high switching control is impossible to achieve in practical systems is because of finite time delays for control computation and limitations of physical actuators. Chattering results in low control accuracy, high heat losses in electrical power circuits, and high wear of moving mechanical parts. It may also excite unmodelled high-frequency dynamics, which degrades the performance of the system and may even lead to instability Hung et al. (1993).

In the following sections there will be given an introduction to first-order SMC and an explanation of the SMC algorithms that are investigated in this thesis. For more information regarding general SMC the project report can be consulted, Borlaug (2016).

2.2 First-order sliding mode control

The first generation of sliding modes (1960-1990) are called first-order sliding modes. They use only the sliding surface to create a control input. First-order SMC algorithms reduces the systems order on the sliding surface, they have finite time convergence and they are robust with respect to matching disturbance. The challenges with these SMC algorithms are chattering and noise sensitivity. The most common first-order SMC algorithms are the relay controller and the saturation controller, these will therefore be described in detail, even though they will not be investigated further as they were tested in the project assignment. Information about the practical relay controller and the practical saturation controller can be found in Borlaug (2016).

2.2.1 Ideal relay control

The ideal relay controller takes the form

$$u(\sigma) = -K\,\mathrm{sgn}(\sigma) \quad \text{where } \mathrm{sgn}(\sigma) = \begin{cases} 1 & \text{when } \sigma > 0 \\ -1 & \text{when } \sigma < 0 \end{cases} \tag{2.1}$$

where K is a control gain and σ is the sliding surface. The controller is ideal in the sense

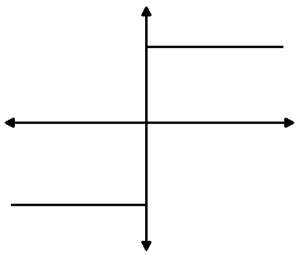


Figure 2.1: Phase portrait: ideal relay control

that it switches instantly at the value $\sigma = 0$. This means that ideal sliding exist on the line $\sigma = 0$, meaning there are no chattering, no steady-state error and that the invariance property holds Hung et al. (1993).

The ideal SMC looks really good on paper, but it is impossible to implement because it is impossible to achieve the high switching control that is necessary for ideal SMC to exist. Chattering is therefore a large problem with this controller.

2.2.2 Ideal saturation control

The ideal saturation controller takes the form

$$u(\sigma) = -Ksat(\sigma) \quad \text{where } sat(\sigma) = \begin{cases} 1 & \text{when } \sigma > L \\ \frac{s}{L} & \text{when } |\sigma| \leq L \\ -1 & \text{when } \sigma < -L \end{cases} \tag{2.2}$$

where $L > 0$ and $\pm L$ defines the threshold for entering the boundary layer, K is a control gain and σ is the sliding surface. The ideal saturation controller is given as a solution to

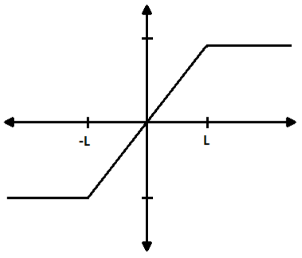


Figure 2.2: Phase portrait: ideal saturation control

the chattering problem. It is a combination of ideal relay control and a high-gain linear control that takes place within the boundary layer. This means that the state trajectories will be driven towards the boundary layer, but once it is inside the boundary layer the trajectories will not be forced to follow the line $\sigma = 0$, it will only be forced stay inside the boundary layer, Hung et al. (1993). As a result sliding mode does not exist inside the boundary layer. The effectiveness of the ideal saturation controller is therefore immediately challenged when parasitic dynamics are considered. In order for the ideal saturation controller to handle these type of disturbances they have to be carefully modelled and considered in the feedback design in order to avoid instability inside the boundary layer. If that is not possible a worst case boundary layer has to be used, which compromises the disturbance rejection properties of the SMC, Young et al. (1999). In conclusion, this approach eliminates the high-frequency chattering at the price of losing invariance.

2.3 Second-order sliding mode control

Many different second-order SMC algorithms exists, some of them are gathered in Bartolini et al. (2003) and Levant (1993). In this section only the super-twisting algorithm (STA) will be given in detail. It was the algorithm that gave the best result in the project assignment, and will therefore be investigated further in this thesis. Information about the A_μ-algorithm, the twisting algorithm, the drift algorithm, an algorithm with a prescribed law, an algorithm without the derivative of σ and a general second-order algorithm can be found in Borlaug (2016).

2.3.1 Super-twisting algorithm

The STA is the most powerful second-order continuous sliding mode control algorithm. It generates the continuous control function that drives the sliding variable and its derivative to zero in finite time in the presence of smooth matched disturbances with bounded gradient, when this boundary is known. As the integrand of the STA contains a discontinuous function, chattering is not eliminated but attenuated. The main drawback of the STA is the requirements to know the boundaries of the disturbance gradient. In many practical cases this boundary cannot be easily estimated, Shtessel et al. (2010). Unlike other second-order sliding mode controllers, STA is applicable to a system (in general, any order) where control appears in the first derivative of the sliding surface σ, Chalanga et al. (2016).

In Levant (1993) the STA was introduced as

$$\begin{aligned} u &= -k_1|\sigma|^{1/2}\operatorname{sgn}(\sigma) + v \\ \dot{v} &= -k_2\operatorname{sgn}(\sigma) \end{aligned} \tag{2.3}$$

where k_i are gains to be designed. If the input signal $f(t)$ is a measurable locally bounded function, and it consist of a base signal having a derivative with Lipschitz's constant $C > 0$. Then sufficient conditions for the converges of $\sigma = \dot{\sigma} = 0$ is

$$k_2 > C, \qquad k_1^2 \geq 4C\frac{k_2 + C}{k_2 - C} \tag{2.4}$$

The conditions in Equation (2.4) results from a very crude estimation Levant (1998). Calculations show that many other values, e.g. $k_1 = 1.5\sqrt{C}$ and $k_2 = 1.1C$ may also be taken. Since this algorithm only works with bounded perturbations a conservative upper bound has to be used when designing the controller to ensure that sliding is maintained. This can worsen the chattering effects. If an adaptive STA is used, the gains can adapt to a level where they are as small as possible but still guarantee that sliding is maintained. Adaptive STA is proposed in Shtessel et al. (2010) and Shtessel et al. (2012). Information about the STA with adaptive gains can also be found in Borlaug (2016).

2.4 Higher-order sliding mode control

Higher-order sliding mode (HOSM) generalizes the sliding mode motion and removes the restriction that for the mode to have full output control the controller u has to appear in the first total derivative of σ. This is the case for the standard sliding modes. HOSM is sliding mode with sliding order higher than 2. The rth-order sliding mode is determined by the equalities $\sigma = \dot{\sigma} = \ddot{\sigma} = \cdots = \sigma^{(r-1)} = 0$ which impose an r-dimensional condition on the state of the dynamic system. This realization can provide up to rth-order of sliding precision, with respect to the measurement interval. If HOSM is properly designed the convergence of HOSM can be asymptotic and it can totally remove the chattering effect, Levant (2001). If the relative degree of the system is less than r, the HOSM algorithms can still be used by artificially increasing the relative degree. This can also help remove the chattering effect, Levant (2003a).

There exist many HOSM control algorithms, but the ones that will be presented and investigated in this thesis are the nested and the quasi-continuous HOSM controller. In Defoort et al. (2009) a higher-order sliding mode control scheme for a multi-input-output non-linear system can be found. An adaptive continuous higher-order sliding mode control algorithm can be found in Edwards and Shtessel (2016). In Dinuzzo and Ferrara (2009) a higher-order sliding mode control with optimal reaching is proposed. They are also gathered in my project report, Borlaug (2016). Recently a new HOSM controller with a Lyapunov function has been proposed in Cruz-Zavala and Moreno (2017), this is a very interesting result, but because of too little time the idea of using the new HOSM controller was not investigated further, and will therefore be a part of further work.

2.4.1 Arbitrary-order sliding mode controllers

The relative degree r of the system that needs to be controlled is assumed to be known and constant, so that

$$\sigma^{(r)} = h(t,x) + g(t,x)u. \tag{2.5}$$

Assume that $g(t,x) > 0$ and that for some K_m, K_M, $C < 0$ the inequalities

$$0 < K_m \leq g(t,x) \leq K_M, \qquad |h(t,x)| \leq C \tag{2.6}$$

hold. Then the controller, both nested and quasi-continuous, takes the form

$$u = -\alpha \Psi_{r-1,r}(\sigma, \dot{\sigma}, \dots, \sigma^{(r-1)}) \tag{2.7}$$

where $\alpha > 0$ and $\Psi_{r-1,r}(\sigma, \dot{\sigma}, \dots, \sigma^{(r-1)})$ is defined by a recursive procedure. With properly chosen positive parameters the controller in Equation (2.7) leads to the establishment of an r-sliding mode $\sigma \equiv 0$ attracting each trajectory in finite time. Parameter $\alpha > 0$ is to be chosen specifically for any fixed C, K_m, K_M. Note that in practice the exact value of the parameters K_m, K_M, C do not have to be known, Shtessel et al. (2014).

Nested arbitrary-order sliding mode controllers

In Levant (2001) and Levant (2003a) the building of an nested arbitrary-order sliding controller is described. Let q be the least common multiple of $1, 2, \dots, r$. Then the nested arbitrary-order sliding controller is built by the following recursive procedure

$$\begin{aligned} N_{i,r} &= \left(|\sigma|^{q/r} + |\dot{\sigma}|^{q/(r-1)} + \cdots + |\sigma^{(i-1)}|^{q/(r-i+1)}\right)^{1/q}, \\ \Psi_{0,r} &= \operatorname{sgn}(\sigma), \\ \Psi_{i,r} &= \operatorname{sgn}(\sigma^{(i)} + \beta_i N_{i,r} \Psi_{i-1,r}), \qquad i = 1, \dots, r-1 \end{aligned} \tag{2.8}$$

where $\beta_1, \dots, \beta_{r-1}$ are positive numbers. The positive parameters $\beta_1, \cdots, \beta_{r-1}$ are to be chosen sufficiently large in the index order and may be fixed in advance for each relative degree r. These control parameters can be chosen in advance, so that there is only one parameter that needs to be adjusted to control a system with a given relative degree.

Quasi-continuous arbitrary-order sliding mode controllers

In Levant (2003b) and Levant (2005) a quasi-continuous arbitrary-order sliding mode controller is described. Let $i = 0, \dots, r-1$, then the controller can be constructed by the following recursive procedure

$$\begin{aligned}
\varphi_{0,r} &= \sigma \quad N_{0,r} = |\sigma| \quad \Psi_{0,r} = \varphi_{0,r}/N_{0,r} = \operatorname{sgn}(\sigma) \\
\varphi_{i,r} &= \sigma^{(i)} + \beta_i N_{i-1,r}^{(r-i)/(r-i+1)} \Psi_{i-1,r} \\
N_{i,r} &= |\sigma^{(i)}| + \beta_i N_{i-1,r}^{(r-i)/(r-i+1)} \quad \Psi_{i,r} = \varphi_{i,r}/N_{i,r}
\end{aligned} \tag{2.9}$$

where $\beta_1, \dots, \beta_{r-1}$ are positive numbers and are chosen sufficiently large in the list order.

To enable implementing these controllers in real-time, robust estimation of the higher-order total output derivatives is required. For this different types of observers (differentiators) can be used. In Levant (2003a) there has been proposed an arbitrary order robust exact finite-time-convergent differentiator that will solve the problem. It allows real-time robust exact differentiation, and its performance is proven to be asymptotically optimal in the presence of small Lebesgue measurable input noises.

2.5 Sliding mode observer

To be able to make the HOSM controllers with the algorithms described in Section 2.4, a differentiator (observer) need to be used to calculate $\sigma, \dot{\sigma}, \dots, \sigma^{r-1}$. In Levant (2003a) an arbitrary-order robust exact differentiator that is finite-time stable was proposed for this use, Section 2.5.1. Since both the position and velocity need to be available for measurement, a state observer is used for the case when only the position measurements are available. In Kumari et al. (2016) a HOSMO is proposed for this use, Section 2.5.2. It insures that the sliding surface dynamics does not contain discontinuous or non-differentiable terms.

2.5.1 Arbitrary-order robust exact differentiator

Recursive form

The algorithm for the recursive form of this differentiator is

$$\begin{aligned}
\dot{z}_0 &= v_0, \quad v_0 = -\lambda_0 |z_0 - f(t)|^{(n/(n+1))} \operatorname{sgn}(z_0 - f(t)) + z_1 \\
\dot{z}_1 &= v_1, \quad v_1 = -\lambda_1 |z_1 - v_0|^{((n-1)/n)} \operatorname{sgn}(z_1 - v_0) + z_2 \\
&\vdots \\
\dot{z}_{n-1} &= v_{n-1}, \quad v_{n-1} = -\lambda_{n-1} |z_{n-1} - v_{n-2}|^{(1/2)} \operatorname{sgn}(z_{n-1} - v_{n-2}) + z_n \\
\dot{z}_n &= -\lambda_n \operatorname{sgn}(z_n - v_{n-1})
\end{aligned} \tag{2.10}$$

where the parameters $\lambda_0, \lambda_1, \dots, \lambda_n$ have to be chosen according to Levant (1998) and Levant (2003a), and $n = r - 1$.

Non-recursive form

The algorithm for the non-recursive form of this differentiator is

$$
\begin{aligned}
\dot{z}_0 &= v_0, \quad v_0 = -\lambda_0 |z_0 - f(t)|^{(n/(n+1))} \operatorname{sgn}(z_0 - f(t)) + z_1 \\
\dot{z}_1 &= v_1, \quad v_1 = -\lambda_1 |z_0 - f(t)|^{((n-1)/n)} \operatorname{sgn}(z_0 - f(t)) + z_2 \\
&\vdots \\
\dot{z}_{n-1} &= v_{n-1}, \quad v_{n-1} = -\lambda_{n-1} |z_0 - f(t)|^{(1/2)} \operatorname{sgn}(z_0 - f(t)) + z_n \\
\dot{z}_n &= -\lambda_n \operatorname{sgn}(z_0 - f(t))
\end{aligned}
\tag{2.11}
$$

where $n = r - 1$ and $\lambda_0, \lambda_1, \ldots, \lambda_n$ are calculated on the basis of the parameters from the recursive algorithm, i.e. the λ in the recursive algorithm and the non-recursive algorithm are not the same.

The differentiators can also be used as an observer for system states by replacing $f(t)$ with the system output.

Universal output-feedback SISO controller

By using the recursive robust exact differentiation to calculate $\sigma, \dot{\sigma}, \ldots, \sigma^{r-1}$ the HOSM algorithm becomes:

$$
\begin{aligned}
u &= -\alpha \Psi_{r-1,r}(z_0, z_1, \ldots, z_{r-1}) \\
\dot{z}_0 &= v_0, \quad v_0 = -\lambda_0 |z_0 - \sigma|^{(r-1/r)} \operatorname{sgn}(z_0 - \sigma) + z_1 \\
\dot{z}_1 &= v_1, \quad v_1 = -\lambda_1 |z_1 - v_0|^{((r-2)/(r-1))} \operatorname{sgn}(z_1 - v_0) + z_2 \\
&\vdots \\
\dot{z}_{r-2} &= v_{r-2}, \quad v_{r-2} = -\lambda_{r-2} |z_{r-2} - v_{r-3}|^{(1/2)} \operatorname{sgn}(z_{r-2} - v_{r-3}) + z_{r-1} \\
\dot{z}_{r-1} &= -\lambda_{r-1} \operatorname{sgn}(z_{r-1} - v_{r-2})
\end{aligned}
\tag{2.12}
$$

where $\lambda_i = \lambda_{0,i} L^{1/r-i}$ are chosen according to the condition $|\sigma^{(r)}| \leq L$, $L \geq C + \alpha K_M$. Some possible choices for the parameters are given in Levant (2003a). By changing the way $z_0, \ldots, z_{r-1}$ is calculated the recursive form can easily be changed to the non-recursive form.

2.5.2 $n+1$th-order sliding mode observer

Given the nth-order perturbed multiple integrator system

$$
\begin{aligned}
\dot{x}_i &= x_{i+1} \quad i = 1, \ldots, n-1 \\
\dot{x}_n &= u + \rho \\
y &= x_1
\end{aligned}
\tag{2.13}
$$

an $n+1$th order SMO is proposed in Kumari et al. (2016) as

$$\begin{aligned}\dot{\hat{x}}_i &= \hat{x}_{i+1} + z_i \quad i = 1, \dots, n-1 \\ \dot{\hat{x}}_n &= \hat{x}_{n+1} + u + z_n \\ \dot{\hat{x}}_{n+1} &= z_{n+1}\end{aligned} \tag{2.14}$$

where

$$\begin{aligned}z_i &= k_i |e_1|^{(n-i+1)/(n+1)} \operatorname{sgn}(e_1) \quad i = 1, \dots, n \\ z_{n+1} &= k_{n+1} \operatorname{sgn}(e_1)\end{aligned} \tag{2.15}$$

and $e_1 = x_1 - \hat{x}_1$. By defining $e_{n+1} = -\hat{x}_{n+1} + \rho$ the error dynamics can be defined as

$$\begin{aligned}\dot{e}_1 &= -k_1 |e_1|^{n/(n+1)} \operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 &= -k_2 |e_1|^{(n-1)/(n+1)} \operatorname{sgn}(e_1) + e_3 \\ &\vdots \\ \dot{e}_n &= -k_n |e_1|^{1/(n+1)} \operatorname{sgn}(e_1) + e_{n+1} \\ \dot{e}_{n+1} &= -k_{n+1} \operatorname{sgn}(e_1) + \dot{\rho}\end{aligned} \tag{2.16}$$

The error dynamics is the same as the non-recursive form of the arbitrary order robust exact differentiator, which means that the error dynamics have the same stability properties of the differentiator with properly chosen gains. When $e \to 0$, $x = \hat{x}$.

Third-order state observer

In this thesis only second-order systems will be considered. To make sure that the sliding surface dynamics does not contain discontinuous or non-differentiable terms a third-order sliding mode observer, is used to observe the states, Kumari et al. (2016). By using the algorithm for the $n+1$th order SMO, a third order sliding mode observer can be designed as

$$\begin{aligned}\dot{\hat{x}}_1 &= \hat{x}_2 + z_1 = \hat{x}_2 + k_1 |e_1|^{2/3} \operatorname{sgn}(e_1) \\ \dot{\hat{x}}_2 &= \hat{x}_3 + z_2 + g(t,x)u = \hat{x}_3 + k_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + g(t,x)u \\ \dot{\hat{x}}_3 &= z_3 = k_3 \operatorname{sgn}(e_1)\end{aligned} \tag{2.17}$$

where k_1, k_2 and k_3 are gains to be chosen according to Levant (1998) and Levant (2003a) and $e_1 = x_1 - \hat{x}_1$. By defining $e_2 = x_2 - \hat{x}_2$ and $e_3 = -\hat{x}_3 + f(t,x)$, the error dynamics can be written as

$$\begin{aligned}\dot{e}_1 &= -k_1 |e_1|^{2/3} \operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 &= -k_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 &= -k_3 \operatorname{sgn}(e_1) + \dot{f}(t,x)\end{aligned} \tag{2.18}$$

2.6 Stability theory

2.6.1 Cascaded systems

The systems that will be evaluated in this thesis are defined as illustrated in Figure 2.3.

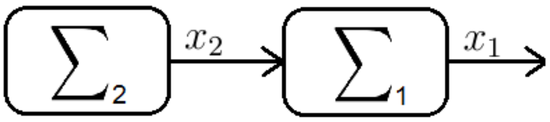


Figure 2.3: Cascade interconnection

It can be represented with equations as

$$\begin{aligned} &\textstyle\sum_1 \left\{ \dot{x}_1 = f_1(t, x_1) + g(t, x)x_2, \right. \\ &\textstyle\sum_2 \left\{ \dot{x}_2 = f_2(t, x_2). \right. \end{aligned} \tag{2.19}$$

From lemma 2.1 in Loría and Panteley (2005), A.2.2 in the appendix, it is stated that if the origin of each nominal system is uniformly globally asymptotically stable (UGAS) and the solutions of Equation (2.19) are uniformly globally bounded (UGB) then the complete system is UGAS.

2.6.2 Finite-time stability

Finite-time stability (FTS) is the property, where the trajectories of a non-Lipschitz system reach a Lyapunov stable equilibrium point in finite time. This finite time point can be calculated and is called the settling time, noted by $T(x)$. $T(x)$ can be a function depending on initial states. If the equilibrium point is zero, which is the case for SMC algorithms and SMO, the state variables can be viewed as zero after the settling time has been reached, Polyakov and Fridman (2014).

Definition 2.6.1 (Finite-time stability, Definition 12 Polyakov and Fridman (2014))**.** The origin of the system $\dot{x} \in F(t, x)$, $t \in R$, is said to be finite-time stable if it is Lyapunov stable and finite-time attractive.

Definition 2.6.2 (Finite-time attractivity, Definition 11 Polyakov and Fridman (2014))**.** The origin of the system $\dot{x} \in F(t, x)$, $t \in R$, is said to be finite-time attractive if for $\forall t_0 \in R$ there exists a set $\mathcal{V}(t_0) \subseteq R^n : 0 \in int(\mathcal{V}(t_0))$ such that $\forall x_0 \in \mathcal{V}(t_0)$

- any solution $x(t, t_0, x_0)$ of Cauchy problem $\dot{x} \in F(t, x)$, $t \in R$, $x(t_0) = x_0$ exists for $t > t_0$;
- $T(t_0, x_0) < +\infty$ for $x_0 \in \mathcal{V}(t_0)$ and for $t_0 \in R$.

The set $\mathcal{V}(t_0)$ is called finite-time attraction domain.

Definition 2.6.3 (Lyapunov stability, Definition 5 Polyakov and Fridman (2014))**.** The origin of the system $\dot{x} \in F(t,x)$, $t \in R$, is said to be Lyapunov stable if for $\forall \varepsilon \in R_+$ and $\forall t_0 \in R$ there exists $\delta = \delta(\varepsilon, t_0) \in R_+$ such that for $\forall x_0 \in \mathcal{B}(\delta)$

1. any solution $x(t, t_0, x_0)$ of Cauchy problem, $\dot{x} \in F(t,x)$, $t \in R$, $x(t_0) = x_0$ exists for $t > t_0$;

2. $x(t, t_0, x_0) \in \mathcal{B}(\varepsilon)$ for $t > t_0$.

If the function δ does not depend on t_0 then the origin is called uniformly Lyapunov stable.

Proposition 2.6.1 (Proposition 3 Polyakov and Fridman (2014))**.** *If the origin of the system $\dot{x} \in F(t,x)$, $t \in R$, is finite-time stable then it is asymptotically stable and $x(t, t_0, x_0) = 0$ for $t > t_0 + T_0(t_0, x_0)$.*

This means that if a system is FTS, it is also AS (asymptotically stable). For autonomous system, uniformity in t_0 is given, which means that if an autonomous system is FTS it is also UAS (uniformly asymptomatically stable), (The National Center for Scientific Research), personal communication, June 16-19, 2017). For globality, all properties must hold for all $x_0 \in R^n$.

Chapter 3

Stability Analysis General System

In this section, tracking control laws based on the super-twisting algorithm (Section 2.3.1), nested third-order SMC (Section 2.4.1) and quasi-continuous third-order SMC (Section 2.4.1) are proposed for a general system. The tracking control objective is to make the error dynamics origin asymptotically stable, so that when $t \to \infty$ the error dynamics approach zero. The different cascaded systems will be analysed by using cascaded theory (Section 2.6.1), Lyapunov theory (Appendix A) and finite-time stable properties (Section 2.6.2). The type of cascade systems that will be analysed are:

- Super-twisting algorithm with a state observer, when the estimated value of x_2 is used in the sliding surface.
- Super-twisting algorithm with a state observer, when the estimated value of x_1 and x_2 are used in the sliding surface.
- Nested third-order sliding mode control with differentiator and a state observer, when the estimated value of x_2 is used in the sliding surface.
- Nested third order-sliding mode control with differentiator and a state observer, when the estimated value of x_1 and x_2 are used in the sliding surface.
- Quasi-continuous third-order sliding mode control with differentiator and a state observer, when the estimated value of x_2 is used in the sliding surface.
- Quasi-continuous third-order sliding mode control with differentiator and a state observer, when the estimated value of x_1 and x_2 are used in the sliding surface.

The state observer that will be used is described in Section 2.5.2 and the differentiator is described in Section 2.5.1.

3.1 Error dynamics general system

Assume that the error dynamics for a second-order general system can be represented by

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(t,x) + g(t,x)u\end{aligned} \tag{3.1}$$

where $|\dot{f}(t,x)| \leq \Delta$ and $g(t,x)$ is known and the sliding surface is

$$\sigma = c_1(x_1 - x_{1,des}) + x_2 - x_{2,des} \tag{3.2}$$

where $c_1 > 0$. The desired value for the error dynamics is zero. By replacing $x_{1,des}$ and $x_{2,des}$ with zero the sliding surface becomes

$$\sigma = c_1 x_1 + x_2. \tag{3.3}$$

3.2 Analysis of the super-twisting algorithm with state observer

3.2.1 The estimated value for x_2 is used in the sliding surface

Overall closed-loop dynamics:

When the estimated value $\hat{x}_2$ is used in the sliding surface, it is

$$\hat{\sigma} = c_1 x_1 + \hat{x}_2. \tag{3.4}$$

By using the fact that $\hat{x}_2 = x_2 - e_2$, from Section 2.5.2, and that $\dot{x}_1 = x_2$, from Equation (3.1), $\hat{\sigma}$ can be written as

$$\hat{\sigma} = c_1 x_1 + \dot{x}_1 - e_2 \tag{3.5}$$

and

$$\dot{x}_1 = \hat{\sigma} - c_1 x_1 + e_2. \tag{3.6}$$

By differentiating Equation (3.4) and using the fact that $\dot{\hat{x}}_2 = \hat{x}_3 + z_2 + g(t,x)u$, from Section 2.5.2, $\dot{\hat{\sigma}}$ becomes

$$\begin{aligned}\dot{\hat{\sigma}} &= c_1\dot{x}_1 + \dot{\hat{x}}_2 = c_1 x_2 + \hat{x}_3 + z_2 + g(t,x)u \\ &= c_1(e_2 + \hat{x}_2) + \hat{x}_3 + z_2 + g(t,x)u.\end{aligned} \tag{3.7}$$

By choosing

$$u = \frac{1}{g(t,x)}(-c_1\hat{x}_2 - \hat{x}_3 - z_2 - k_1|\hat{\sigma}|^{1/2}\operatorname{sgn}(\hat{\sigma}) - \int_0^t k_2 \operatorname{sgn}(\hat{\sigma})d\tau) \tag{3.8}$$

and inserting u in Equation (3.7)

$$\dot{\hat{\sigma}} = c_1 e_2 - k_1|\hat{\sigma}|^{1/2}\operatorname{sgn}(\hat{\sigma}) - \int_0^t k_2 \operatorname{sgn}(\hat{\sigma})d\tau. \tag{3.9}$$

The overall closed-loop dynamics can then be written as

$$\begin{aligned} &\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 + e_2 \\ \dot{\hat{\sigma}} = c_1 e_2 - k_1 |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2 \operatorname{sgn}(\hat{\sigma}) \end{cases} \\ &\sum\nolimits_2 \begin{cases} \dot{e}_1 = -\lambda_1 |e_1|^{2/3} \operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 = -\lambda_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 = -\lambda_3 \operatorname{sgn}(e_1) + \dot{f}(t,x) \end{cases} \end{aligned} \tag{3.10}$$

Note that $\sum_2$ is the error dynamics for the state observer, from Equation (2.18).

Theorem 3.2.1. *Assume that the error dynamics for a second-order system has the form as in Equation* (3.1)*, where $|\dot{f}(t,x)| \leq \Delta$ and $g(t,x)$ is known and the sliding surface is as in Equation* (3.4) *with $c_1 > 0$. Assume that a state observer with the form as in Equation* (2.17) *is used to estimate x_2. Let the control input be given by Equation* (3.8)*. Then the origin of the cascaded system in Equation* (3.10) *is uniformly globally asymptotically stable (UGAS), which ensures asymptotic convergence of the tracking error.*

Proof. **Lyapunov analysis:**

Analysis of subsystem 1, with $e_2 = 0$: With $e_2 = 0$, subsystem 1 can be written as

$$\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 \\ \dot{\hat{\sigma}} = -k_1 |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2 \operatorname{sgn}(\hat{\sigma}) \end{cases} \tag{3.11}$$

This can then be divided in two subsystems:

$$\begin{aligned} &\sum\nolimits_{11} \left\{ \dot{x}_1 = \hat{\sigma} - c_1 x_1 \right. \\ &\sum\nolimits_{12} \begin{cases} \dot{\hat{\sigma}} = -k_1 |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2 \operatorname{sgn}(\hat{\sigma}) \end{cases} \end{aligned} \tag{3.12}$$

where Lemma A.2.2 can be used. Subsystem $\sum_{11}$ with $\hat{\sigma} = 0$ is analysed first. For analysis the Lyapunov function $V_{11}(x) = \frac{1}{2}x_1^2$ is used. The derivative of the Lyapunov function is

$$\begin{aligned} \dot{V}_{11}(x) &= x_1 \dot{x}_1 = x_1(-c_1 x_1) \\ &= -c_1 x_1^2 \leq -c_1 ||x||^2. \end{aligned} \tag{3.13}$$

This means that the Lyapunov function satisfies:

$$\begin{aligned} &k_1 ||x||^a \leq V(t,x) \leq k_2 ||x||^a \\ &\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t,x) \leq -k_3 ||x||^a \end{aligned} \tag{3.14}$$

with $k_1 = k_2 = \frac{1}{2}$, $k_3 = c_1$ and $a = 2$. By using Theorem A.1.4 subsystem $\sum_{11}$ is proven globally exponentially stable with $\hat{\sigma} = 0$, when $c_1 > 0$.

Subsystem $\sum_{12}$ has the structure of the STA algorithm, in Moreno and Osorio (2012) a strict Lyapunov function is proposed. To be able to use that Lyapunov function a change of variable is needed:

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) \\ v \end{bmatrix} \tag{3.15}$$

$$\dot{z} = \frac{1}{|z_1|} Az \quad \text{where } A = \begin{bmatrix} -\frac{1}{2}k_1 & \frac{1}{2} \\ -k_2 & 0 \end{bmatrix} \tag{3.16}$$

A is Hurwitz if $k_1, k_2 > 0$. The Lyapunov function proposed is then $V_{12}(z) = z^T P z$, where it is proven that $\dot{V}_{12}(z) = -\frac{1}{|z_1|} z^T Q z$ almost everywhere. The relationship between P and Q is related by the Algebraic Lyapunov Equation (ALE) $A^T P + PA = -Q$. It is proven that the origin of subsystem $\sum_{12}$ is FTS, as long as the matrix A is Hurwitz, and that the Lyapunov function proposed is a global, strict Lyapunov function for every $P = P^T > 0$ that is a solution of the ALE with an arbitrary matrix $Q = Q^T > 0$. As subsystem $\sum_{12}$ is FTS and autonomous it will be UGAS, this also implies $||\hat{\sigma}(t)|| < \beta_1$.

To check if the solutions of $\sum_1$ are UGB, subsystem $\sum_{11}$ has to be analysed with $\hat{\sigma} \neq 0$. The derivative of the Lyapunov function V_{11} is then

$$\begin{aligned} \dot{V}_{11}(x) &= -c_1||x||^2 + \hat{\sigma} x_1 \\ &\leq -c_1||x||^2 + \theta||x||^2 - \theta||x||^2 + \beta_1||x|| \\ &\leq -(c_1 - \theta)||x||^2 \quad \forall \quad ||x|| \geq \frac{\beta_1}{\theta} \end{aligned} \tag{3.17}$$

where $0 < \theta < c_1$. The solutions are then UGB, since the inequality in Equation (A.5) in Theorem A.1.5 is satisfied. This means that subsystem $\sum_1$ is UGAS since Lemma A.2.2 is satisfied.

Analysis of subsystem 2:

$$\sum\nolimits_2 \begin{cases} \dot{e}_1 = -\lambda_1 |e_1|^{2/3} \operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 = -\lambda_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 = -\lambda_3 \operatorname{sgn}(e_1) + \dot{f}(t, x) \end{cases} \tag{3.18}$$

In Moreno (2012) a Lyapunov function is proposed for a third order observer. To be able to use that Lyapunov function a change of variable is needed:

$$\xi = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} = \begin{bmatrix} |e_1|^{2/3} \operatorname{sgn}(e_1) \\ e_2 \\ |e_3|^2 \operatorname{sgn}(e_3) \end{bmatrix} \tag{3.19}$$

If $|\dot{f}(t, x)| \leq \Delta$, it is proven in Moreno (2012) that the Lyapunov function

$$V_2(\xi) = \xi^T \Gamma \xi, \quad \text{where } \Gamma = \begin{bmatrix} \gamma_1 & -\frac{1}{2}\gamma_{12} & 0 \\ -\frac{1}{2}\gamma_{12} & \gamma_2 & -\frac{1}{2}\gamma_{23} \\ 0 & -\frac{1}{2}\gamma_{23} & \gamma_3 \end{bmatrix} \tag{3.20}$$

satisfies

$$\dot{V}_2(\xi) = -\kappa V^{\frac{3}{4}} \tag{3.21}$$

for $\kappa > 0$, is radially unbounded, positive definite and that it is a Lyapunov function for subsystem $\sum_2$, whose trajectories converge in finite time to the origin $e = 0$ for every value of $|\dot{f}(t,x)|$. This means that the origin is FTS for every value of $|\dot{f}(t,x)|$, which means that the origin is also UGAS, this implies $|e(t)|| \leq \beta_2$.

Analysis of the complete system: To analyse the complete system Lemma A.2.2 is used. To check if the solutions of the complete system are UGB, the boundedness of $\hat{\sigma}$ and x_1 have to be evaluated when $e_2 \neq 0$. First the boundedness of $\hat{\sigma}$ is checked by using the Lyapunov function V_{12}. From earlier calculations $||e(t)|| < \beta_2$. The differentiation of the Lyapunov function is

$$\begin{aligned}
\dot{V}_{12}(z) &= -\frac{1}{|z_1|}z^TQz + \frac{2}{|z_1|}\begin{bmatrix}\frac{1}{2}c_1e_2 & 0\end{bmatrix}Pz \\
&= -\frac{1}{|z_1|}z^TQz + \frac{c_1e_2}{|z_1|}(p_1z_1 + p_{12}z_2) \\
&\leq -\frac{1}{|z_1|}\lambda_{min}(Q)||z||_2^2 + \frac{c_1\beta_2}{|z_1|}\sqrt{p_1^2 + p_{12}^2}||z||_2 \\
&\leq -\lambda_{min}(Q)||z||_2^2 + \theta||z||_2^2 - \theta||z||_2^2 + c_1\beta_2\sqrt{p_1^2 + p_{12}^2}||z||_2 \\
&\leq -(\lambda_{min}(Q) - \theta)||z||_2^2 \quad \forall \quad ||z||_2 \geq \frac{c_1\beta_2\sqrt{p_1^2 + p_{12}^2}}{\theta}
\end{aligned} \tag{3.22}$$

where $0 < \theta < \lambda_{min}(Q)$. The solutions are then UGB, since the inequality in Equation (A.5) in Theorem A.1.5 is satisfied. This means that $\hat{\sigma}(t)$ is still bounded when $e_2 \neq 0$. To check the boundedness of x_1 when $e_2 \neq 0$ the Lyapunov function V_{11} is used.

$$\begin{aligned}
\dot{V}_{11}(x) &= -c_1||x||^2 + (\hat{\sigma} + e_2)x_1 \\
&\leq -c_1||x||^2 + \theta||x||^2 - \theta||x||^2 + (\beta_1 + \beta_2)||x|| \\
&\leq -(c_1 - \theta)||x||^2 \quad \forall \quad ||x|| \geq \frac{\beta_1 + \beta_2}{\theta}
\end{aligned} \tag{3.23}$$

where $0 < \theta < c_1$. The solutions are then UGB, since the inequality in Equation (A.5) in Theorem A.1.5 is satisfied. This means that the complete system is UGAS as Lemma A.2.2 is satisfied. $\square$

The closed-loop dynamics can also be proven stable by using FTS properties, this has been done in Chalanga et al. (2016), and will be given in the appendix, Section B.1.1.

3.2.2 The estimated value for x_1 and x_2 are used in the sliding surface

Overall closed-loop dynamics:

When $\hat{x}_1$ and $\hat{x}_2$ is used in the sliding surface, it is

$$\hat{\sigma} = c_1\hat{x}_1 + \hat{x}_2. \tag{3.24}$$

By using the fact that $\hat{x}_1 = x_1 - e_1$, $\hat{x}_2 = x_2 - e_2$, from Section 2.5.2, and that $\dot{x}_1 = x_2$, from Equation (3.1), $\hat{\sigma}$ can be written as

$$\hat{\sigma} = c_1(x_1 - e_1) + \dot{x}_1 - e_2 \tag{3.25}$$

and

$$\dot{x}_1 = \hat{\sigma} - c_1x_1 + c_1e_1 + e_2. \tag{3.26}$$

By differentiating Equation (3.24) and using the fact that $\dot{\hat{x}}_1 = \hat{x}_2 + z_1$ and $\dot{\hat{x}}_2 = \hat{x}_3 + z_2 + g(t,x)u$, from Section 2.5.2, $\dot{\hat{\sigma}}$ becomes

$$\dot{\hat{\sigma}} = c_1\dot{\hat{x}}_1 + \dot{\hat{x}}_2 = c_1(\hat{x}_2 + z_1) + \hat{x}_3 + z_2 + g(t,x)u. \tag{3.27}$$

By choosing

$$u = \frac{1}{g(t,x)}(-c_1\hat{x}_2 - c_1z_1 - \hat{x}_3 - z_2 - k_1|\hat{\sigma}|^{1/2}\operatorname{sgn}(\hat{\sigma}) - \int_0^t k_2\operatorname{sgn}(\hat{\sigma})d\tau) \tag{3.28}$$

and inserting u in Equation (3.27)

$$\dot{\hat{\sigma}} = -k_1|\hat{\sigma}|^{1/2}\operatorname{sgn}(\hat{\sigma}) - \int_0^t k_2\operatorname{sgn}(\hat{\sigma})d\tau. \tag{3.29}$$

The overall closed-loop dynamics can then be written as

$$\begin{aligned}
&\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1x_1 + c_1e_1 + e_2 \\ \dot{\hat{\sigma}} = -k_1|\hat{\sigma}|^{1/2}\operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2\operatorname{sgn}(\hat{\sigma}) \end{cases} \\
&\sum\nolimits_2 \begin{cases} \dot{e}_1 = -\lambda_1|e_1|^{2/3}\operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 = -\lambda_2|e_1|^{1/3}\operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 = -\lambda_3\operatorname{sgn}(e_1) + \dot{f}(t,x) \end{cases}
\end{aligned} \tag{3.30}$$

Note that $\sum_2$ is the error dynamics for the state observer, from Equation 2.18.

Theorem 3.2.2. *Assume that the error dynamics for a second-order system has the form as in Equation* (3.1)*, where $|\dot{f}(t,x)| \le \Delta$ and $g(t,x)$ is known and the sliding surface is as in Equation* (3.24) *with $c_1 > 0$. Assume that a state observer with the form as in Equation* (2.17) *is used to estimate x_1 and x_2. Let the control input be given by Equation* (3.28)*. Then the origin of the cascaded system in Equation* (3.30) *is uniformly globally asymptotically stable (UGAS), which ensures asymptotic convergence of the tracking error.*

Proof. **Lyapunov analysis:**

Analysis of subsystem 1, with $e_1 = 0$ and $e_2 = 0$: With $e_1 = 0$ and $e_2 = 0$, subsystem 1 can be written as

$$\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 \\ \dot{\hat{\sigma}} = -k_1 |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2 \operatorname{sgn}(\hat{\sigma}) \end{cases} \tag{3.31}$$

which is proven UGAS in Section 3.2.1.

Analysis of subsystem 2: Subsystem 2 has also been analysed in Section 3.2.1 to be UGAS.

Analysis of the complete system: To analyse the complete system Lemma A.2.2 is used. To check if the solutions of the complete system are UGB, the boundedness of x_1 has to be evaluated when $e_1 \neq 0$ and $e_2 \neq 0$, for this the Lyapunov function V_{11} is used. The boundedness of $\hat{\sigma}$ has already been checked in Section 3.2.1, since it is not effected by e.

$$\begin{aligned} \dot{V}_{11}(x) &= -c_1||x||^2 + (\hat{\sigma} + e_1 + e_2)x_1 \\ &\leq -c_1||x||^2 + \theta||x||^2 - \theta||x||^2 + (\beta_1 + 2\beta_2)||x|| \\ &\leq -(c_1 - \theta)||x||^2 \quad \forall \quad ||x|| \geq \frac{\beta_1 + 2\beta_2}{\theta} \end{aligned} \tag{3.32}$$

where $0 < \theta < c_1$. The solutions are then UGB, because the inequality in Equation (A.5) in Theorem A.1.5 is satisfied. This means that the complete system is UGAS because Lemma A.2.2 is satisfied. □

The closed-loop dynamics can also be proven stable by using FTS properties. This will be shown in the appendix, Section B.1.2.

3.3 Analysis of the HOSM algorithms with differentiator and state observer

As both the HOSM algorithms (nested and quasi-continuous) with differentiators are FTS and autonomous, they will be UGAS and be bounded no matter what order or type that is used. So the control structure for the HOSM algorithms with the differentiator will in this section be called u_c, and the analysis will be made with u_c so that it holds for both the HOSM controller types and all orders. The nested HOSM algorithm with differentiator is proven FTS in Levant (2003a) and the quasi-continuous HOSM algorithm with differentiator is proven FTS in Levant (2003b) and Levant (2005).

3.3.1 The estimated value for x_2 is used in the sliding surface

Overall closed-loop dynamics:

From Section 3.2.1:

$$\dot{x}_1 = \hat{\sigma} - c_1 x_1 + e_2 \tag{3.33}$$

$$\dot{\hat{\sigma}} = c_1(e_2 + \hat{x}_2) + \hat{x}_3 + z_2 + g(t,x)u. \tag{3.34}$$

By choosing

$$u = \frac{1}{g(t,x)}(-c_1\hat{x}_2 - \hat{x}_3 - z_2 + u_c) \tag{3.35}$$

and inserting u in Equation (3.34)

$$\dot{\hat{\sigma}} = c_1 e_2 + u_c. \tag{3.36}$$

The overall closed-loop dynamics can then be written as

$$\begin{gathered} \sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 + e_2 \\ \dot{\hat{\sigma}} = c_1 e_2 + u_c \end{cases} \\ \sum\nolimits_2 \begin{cases} \dot{e}_1 = -\lambda_1 |e_1|^{2/3}\operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 = -\lambda_2 |e_1|^{1/3}\operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 = -\lambda_3 \operatorname{sgn}(e_1) + \dot{f}(t,x) \end{cases} \end{gathered} \tag{3.37}$$

Note that $\sum_2$ is the error dynamics for the state observer, from Equation (2.18).

Theorem 3.3.1. *Assume that the error dynamics for a second-order system has the form as in Equation* (3.1)*, where* $|\dot{f}(t,x)| \leq \Delta$ *and* $g(t,x)$ *is known and the sliding surface is as in Equation* (3.4) *with* $c_1 > 0$*. Assume that a state observer with the form as in Equation* (2.17) *is used to estimate* x_2*. Let the control input be given by Equation* (3.35)*. Then the origin of each subsystem in the cascaded system in Equation* (3.37) *is uniformly globally asymptotically stable (UGAS), but the solutions of the complete system can not be proven bounded because there does not exist, to the authors knowledge, a Lyapunov function, at this time, that can be used to prove it. Therefore, the complete cascaded system can not be proven UGAS.*

Proof. **Lyapunov analysis:**

Analysis of subsystem 1, with $e_2 = 0$: With $e_2 = 0$, subsystem 1 can be written as

$$\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 \\ \dot{\hat{\sigma}} = u_c \end{cases} \tag{3.38}$$

this can then be divided in two subsystems:

$$\begin{gathered} \sum\nolimits_{11} \left\{ \dot{x}_1 = \hat{\sigma} - c_1 x_1 \right. \\ \sum\nolimits_{12} \left\{ \dot{\hat{\sigma}} = u_c \right. \end{gathered} \tag{3.39}$$

where Lemma A.2.2 can be used. Subsystem $\sum_{11}$ with $\hat{\sigma} = 0$ has been analysed in Section 3.2.1 to be GES. Since subsystem $\sum_{12}$ is the control input directly from the HOSM algorithm with a differentiator it is FTS, and will therefore be UGAS and $|\hat{\sigma}(t)| < \beta_1$. To check if the solutions of $\sum_1$ are UGB, subsystem $\sum_{11}$ has to be analysed when $\hat{\sigma} \neq 0$. The derivative of the Lyapunov function V_{11} is then

$$\begin{aligned}\dot{V}_{11}(x) &= -c_1||x||^2 + \hat{\sigma}x_1 \\ &\leq -c_1||x||^2 + \theta||x||^2 - \theta||x||^2 + \beta_1||x|| \\ &\leq -(c_1 - \theta)||x||^2 \quad \forall \quad ||x|| \geq \frac{\beta_1}{\theta}\end{aligned} \tag{3.40}$$

where $0 < \theta < c_1$. The solutions are then UGB, as the inequality in Equation (A.5) in Theorem A.1.5 is satisfied. This means that subsystem $\sum_1$ is UGAS because Lemma A.2.2 is satisfied.

Analysis of subsystem 2: Subsystem 2 has been analysed in Section 3.2.1 to be UGAS.

Analysis of the complete system: To check if the solutions of the complete system are UGB, the boundedness of $\hat{\sigma}$ and x_1 have to be evaluated when $e_2 \neq 0$. First the boundedness of $\hat{\sigma}$ needs to be checked. As there does not yet exist a Lyapunov function for this controller in the literature and it would be to time consuming to try to find such a Lyapunov function the boundedness can not be checked. Therefore, the stability proof can not be taken any further before such a Lyapunov function is found. □

The closed-loop dynamics can not be analysed by FTS properties either because there is no way of knowing that $\hat{\sigma}$ does not escape to infinity in finite time when $e_2 \neq 0$.

3.3.2 The estimated value for x_1 and x_2 are used in the sliding surface

Overall closed-loop dynamics:

From Section 3.2.2:

$$\dot{x}_1 = \hat{\sigma} - c_1x_1 + c_1e_1 + e_2 \tag{3.41}$$

$$\dot{\hat{\sigma}} = c_1(\hat{x}_2 + z_1) + \hat{x}_3 + z_2 + g(t,x)u. \tag{3.42}$$

By choosing

$$u = \frac{1}{g(t,x)}(-c_1\hat{x}_2 - c_1z_1 - \hat{x}_3 - z_2 - k_1 + u_c) \tag{3.43}$$

and inserting u in Equation (3.42), $\dot{\hat{\sigma}}$ becomes

$$\dot{\hat{\sigma}} = u_c. \tag{3.44}$$

The overall closed-loop dynamics can then be written as

$$\begin{aligned}
&\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 + c_1 e_1 + e_2 \\ \dot{\hat{\sigma}} = u_c \end{cases} \\
&\sum\nolimits_2 \begin{cases} \dot{e}_1 = -\lambda_1 |e_1|^{2/3} \operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 = -\lambda_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 = -\lambda_3 \operatorname{sgn}(e_1) + \dot{f}(t,x) \end{cases}
\end{aligned} \tag{3.45}$$

Note that $\sum_2$ is the error dynamics for the state observer, from Equation (2.18).

Theorem 3.3.2. *Assume that the error dynamics for a second-order system has the form as in Equation* (3.1)*, where* $|\dot{f}(t,x)| \leq \Delta$ *and* $g(t,x)$ *is known and the sliding surface is as in Equation* (3.24) *with* $c_1 > 0$*. Assume that a state observer with the form as in Equation* (2.17) *is used to estimate* x_1 *and* x_2*. Let the control input be given by Equation* (3.43)*. Then the origin of the cascaded system in Equation* (3.45) *is uniformly globally asymptotically stable (UGAS), which ensures asymptotic convergence of the tracking error.*

Proof. **Lyapunov analysis:**

Analysis of subsystem 1, with $e_1 = 0$ *and* $e_2 = 0$*:* With $e_1 = 0$ and $e_2 = 0$, subsystem 1 can be written as

$$\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 \\ \dot{\hat{\sigma}} = u_c \end{cases} \tag{3.46}$$

which is proven UGAS in Section 3.3.1.

Analysis of subsystem 2: Subsystem 2 has been analysed in Section 3.2.1 to be UGAS.

Analysis of the complete system: To analyse the complete system, Lemma A.2.2 is used. To check if the solutions of the complete system are UGB, the boundedness of x_1 has to be evaluated when $e_1 \neq 0$ and $e_2 \neq 0$, for this the Lyapunov function V_{11} is used. The boundedness of $\hat{\sigma}$ has already been checked in Section 3.3.1, as it is not effected by e.

$$\begin{aligned}
\dot{V}_{11}(x) &= -c_1||x||^2 + (\hat{\sigma} + e_1 + e_2)x_1 \\
&\leq -c_1||x||^2 + \theta||x||^2 - \theta||x||^2 + (\beta_1 + 2\beta_2)||x|| \\
&\leq -(c_1 - \theta)||x||^2 \quad \forall \quad ||x|| \geq \frac{\beta_1 + 2\beta_2}{\theta}
\end{aligned} \tag{3.47}$$

where $0 < \theta < c_1$. The solutions are then UGB, since the inequality in Equation (A.5) in Theorem A.1.5 is satisfied. This means that the complete system is UGAS since Lemma A.2.2 is satisfied. □

The closed-loop dynamics can also be proven stable by using FTS properties. This will be shown in the appendix, Section B.2.1.

Chapter 4

Mass-Spring-Damper System

The mass-spring damper system is a simple yet challenging motion control problem. It is therefore used as a test system to compare different SMC algorithms. The SMC algorithms that are to be investigated with this system are:

- Super-twisting algorithm with a state observer, when the estimated value of x_2 is used in the sliding surface.
- Super-twisting algorithm with a state observer, when the estimated value of x_1 and x_2 are used in the sliding surface.
- Nested third-order sliding mode control with differentiator and a state observer, when the estimated value of x_2 is used in the sliding surface.
- Nested third-order-sliding mode control with differentiator and a state observer, when the estimated value of x_1 and x_2 are used in the sliding surface.
- Quasi-continuous third-order sliding mode control with differentiator and a state observer, when the estimated value of x_2 is used in the sliding surface.
- Quasi-continuous third-order sliding mode control with differentiator and a state observer, when the estimated value of x_1 and x_2 are used in the sliding surface.

The STA is tested here for comparison purposes as it gave the smoothest control input and the smallest error in Borlaug (2016). The HOSM algorithms are tested to see if they give a smoother control input or smaller errors than the lower-order SMC algorithm. To be able to implement the HOSM algorithms a differentiator was also included. The recursive form of the arbitrary-order robust exact differentiator, Section 2.5.1, was chosen for the job. The reason is that it is the most known and common differentiator designed for HOSM algorithms. Both the position and velocity need to be available for measurement. For the case when only the position measurements are available a higher-order sliding mode observer, Section 2.5.2, will be used to estimate the states. The reason for using this state observer is to make sure that the sliding surface dynamics does not contain discontinuous or non-differentiable terms, Kumari et al. (2016). To be able to see how the state observer affects the performance and control abilities of the algorithms, a tracking control law where the state observer is not used, i.e. the actual value for the states are used in the sliding surface, will also be tested. In Borlaug (2016) such a control law was proposed for the relay controller, the saturation controller, the STA and the STA with adaptive gains.

Stability for the error dynamics for a second-order general system was proven in Chapter 3, for all the algorithms given above, except for the HOSM algorithms when the estimated value of x_2 was used in the sliding surface. In Section 4.3 it will be shown that the error dynamics for the mass-spring-damper system fits the assumptions made for the error dynamics for the general system, which means that the theorems in Chapter 3 also hold for the error dynamics for the mass-spring-damper system.

4.1 System

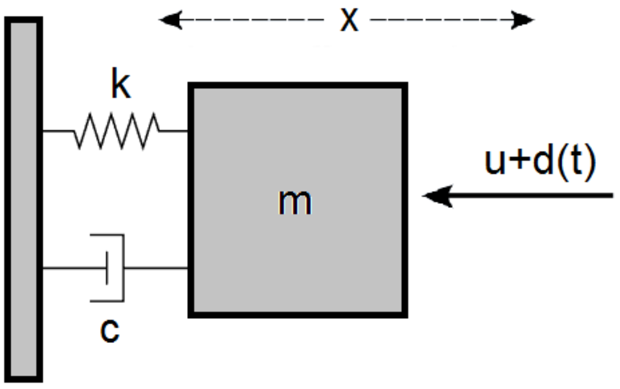


Figure 4.1: Schematic of mass-spring-damper system

The mass-spring-damper system is a second order system, where the differential equation that describes the system can be written as

$$m\ddot{x} + c\dot{x} + kx = u + d(t) \tag{4.1}$$

where x [m] is the position of the mass, m [kg] is the weight of the mass, c [N s/m] is the damping coefficient, k [N/m] is the spring constant, u [N] is the force input and $d(t)$ is a bounded time-varying disturbance. By setting $x = x_1$ and $\dot{x} = x_2$, Equation (4.1) can be written as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}(-cx_2 - kx_1 + u + d(t)) = f(t,x) + g(t,x)u \end{aligned} \tag{4.2}$$

4.2 Control design

Control problem: *The control objective is to make the position of the mass x, follow a pre-defined trajectory.*

4.2.1 Sliding surface design

To design a sliding surface an error variable has to be introduced. Define the output as x_1, the error variable can then be defined as

$$\tilde{x}_1 = x_1 - x_{\text{des}} \tag{4.3}$$

where x_{des} is the desired position of the mass. This can be both time-varying and time-invariant. The sliding surface should then be selected such that the state trajectories of the controlled system is forced onto the sliding surface $\sigma = \dot{\sigma} = 0$, where the system behaviour meets the design specifications. The controller u should also appear in the first derivative of σ, so that the relative degree is equal to 1. The sliding surface σ can then be chosen as

$$\sigma = \tilde{x}_1 + \dot{\tilde{x}}_1 = \tilde{x}_1 + \tilde{x}_2 \tag{4.4}$$

where $\tilde{x}_2 = x_2 - \dot{x}_{\text{des}}$. It is assumed that only the position, x_1, is available for measurement, an observer for the states are therefore designed, Section 4.2.2. The observer states will be used in the sliding surface, and following the structure of Equation 4.4, the revised sliding surface is then

$$\hat{\sigma} = \tilde{x}_1 + \hat{\tilde{x}}_2 \tag{4.5}$$

when the estimated value of x_2 is used in the sliding surface, and

$$\hat{\sigma} = \hat{\tilde{x}}_1 + \hat{\tilde{x}}_2 \tag{4.6}$$

when both the estimated value of x_1 and x_2 are used in the sliding surface. $\hat{\tilde{x}}_1 = \hat{x}_1 - x_{\text{des}}$ and $\hat{\tilde{x}}_2 = \hat{x}_2 - \dot{x}_{\text{des}}$, where $\hat{x}_1$ and $\hat{x}_2$ are the estimated states. For the remaining sections of this chapter the sliding surface in Equation (4.4) will be called sliding surface 1, the sliding surface in Equation (4.5) will be called sliding surface 2 and the sliding surface in Equation (4.6) will be called sliding surface 3.

4.2.2 Observer and differentiator design

State observer

In Section 2.5.2 a third-order sliding mode observer was designed as

$$\begin{aligned}
\dot{\hat{x}}_1 &= \hat{x}_2 + z_1 = \hat{x}_2 + k_1 |e_1|^{2/3} \operatorname{sgn}(e_1) \\
\dot{\hat{x}}_2 &= \hat{x}_3 + z_2 + g(t,x)u = \hat{x}_3 + k_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + g(t,x)u \\
\dot{\hat{x}}_3 &= z_3 = k_3 \operatorname{sgn}(e_1)
\end{aligned} \tag{4.7}$$

where k_1, k_2 and k_3 are gains to be chosen according to Levant (1998) and Levant (2003a) and $e_1 = x_1 - \hat{x}_1$. One choice of parameters that meets the requirements are $k_1 = 6L_{\text{SO}}^{1/3}$, $k_2 = 11L_{\text{SO}}^{1/2}$ and $k_3 = 6L_{\text{SO}}$, where L_{SO} is a sufficiently large constant, Chalanga et al. (2016).

Differentiator

To be able to use the HOSM algorithms proposed in Section 2.4 a differentiator has to be design. By using the recursive form of the arbitrary order robust exact differentiator with $r = 3$, described in Equation (2.10), a third-order differentiator for σ can be designed as

$$\begin{aligned} \dot{z}_0 &= v_0 \quad v_0 = -\lambda_2 L_D^{1/3} |z_0 - \sigma|^{2/3} \operatorname{sgn}(z_0 - \sigma) + z_1 \\ \dot{z}_1 &= v_1 \quad v_1 = -\lambda_1 L_D^{1/2} |z_1 - v_0|^{1/2} \operatorname{sgn}(z_1 - v_0) + z_2 \\ \dot{z}_2 &= -\lambda_0 L_D \operatorname{sgn}(z_2 - v_1) \end{aligned} \tag{4.8}$$

where the gains λ_0, λ_1 and λ_2 are to be chosen according to Levant (1998) and Levant (2003a). One choice of parameters that meets the requirements are $\lambda_0 = 1.1$, $\lambda_1 = 1.5$ and $\lambda_2 = 3$. Here $z_0 = \sigma$, $z_1 = \dot{\sigma}$ and $z_2 = \ddot{\sigma}$ in the HOSM control algorithms. L_D needs to be sufficiently large.

4.2.3 Control algorithms

The super-twisting algorithm

By choosing the gains in Equation (2.3) to be $k_1 = 1.5\sqrt{K}$ and $k_2 = 1.1K$, where K is a sufficiently large positive constant. The STA can be written as

$$\begin{aligned} u_{\text{STA}} &= -1.5\sqrt{K}|\sigma|^{1/2} \operatorname{sgn}(\sigma) + v \\ \dot{v} &= -1.1K \operatorname{sgn}(\sigma) \end{aligned} \tag{4.9}$$

Higher-order sliding mode controller (third-order)

Nested third-order sliding mode: By choosing $r = 3$ in Equation (2.8), the nested third-order sliding mode controller can be described as

$$u_N = -\alpha \operatorname{sgn}(\ddot{\sigma} + \beta_2(|\dot{\sigma}|^3 + |\sigma|^2)^{1/6} \times \operatorname{sgn}(\dot{\sigma} + \beta_1 |\sigma|^{2/3} \operatorname{sgn}(\sigma))) \tag{4.10}$$

where $\beta_2 = 2$, $\beta_1 = 1$ and α is a sufficiently large constant.

Quasi-continuous third-order sliding mode: By choosing $r = 3$ in Equation (2.9), the quasi-continuous third-order sliding mode controller can be described as

$$u_Q = -\alpha \frac{\ddot{\sigma} + \beta_2(|\dot{\sigma}| + |\sigma|^{2/3})^{-1/2}(\dot{\sigma} + \beta_1|\sigma|^{2/3}\operatorname{sgn}(\sigma))}{|\ddot{\sigma}| + \beta_2(|\dot{\sigma}| + |\sigma|^{2/3})^{1/2}} \tag{4.11}$$

where $\beta_2 = 2$, $\beta_1 = 1$ and α is a sufficiently large constant.

4.2.4 Control input

In Chapter 3 the control input was found for the different sliding surfaces and control algorithms, they can be generalized so that by changing u_c in the following sections to u_{STA}, u_N or u_Q it is valid for all the control algorithms. In the case where the state observer is not used, the sliding surface will be sliding surface 1, and the control input is $u = \frac{1}{g(t,x)}u_c$, Borlaug (2016).

Sliding surface 1: the estimated value for x_2 is used in the sliding surface

$$u = \frac{1}{g(t,x)}(-\hat{x}_2 - \hat{x}_3 - z_2 + u_c) \tag{4.12}$$

Sliding surface 2: the estimated value for x_1 and x_2 are used in the sliding surface

$$u = \frac{1}{g(t,x)}(-\hat{x}_2 - z_1 - \hat{x}_3 - z_2 + u_c) \tag{4.13}$$

4.3 Stability analysis

Stability for the error dynamics for a second-order general system was proven in Chapter 3. In this section it will be shown that the error dynamics for the mass-spring-damper system fits the assumptions made for the error dynamics for the general system, which means that the theorems in Chapter 3 also hold for the error dynamics for the mass-spring-damper system. The assumptions made for the error dynamics for the second-order general system were

Assumption 1 $|\dot{f}(t,x)| < \Delta$

Assumption 2 $g(t,x)$ known

Assumption 3 $c_1 > 0$

The controller gains also have to be properly chosen according to Section 2.1, 2.5 and 4.2. The sliding surface for this system is $\sigma = \tilde{x}_1 + \tilde{x}_2$, which means that $c_1 = 1$. This is also the case for the sliding surfaces where the estimated states are used. Therefore, assumption 3 hold. To check if assumption 1 and 2, hold the error dynamics has to be found and analysed.

4.3.1 Error dynamics

The MSDS is described by

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}(-cx_2 - kx_1 + u + d(t))\end{aligned} \tag{4.14}$$

and the error variables are: $\tilde{x}_1 = x_1 - x_{\text{des}}(t)$ and $\tilde{x}_2 = x_2 - \dot{x}_{\text{des}}(t)$. By differentiating the error variables, the error dynamics can be described as

$$\begin{aligned}\dot{\tilde{x}}_1 &= \dot{x}_1 - \dot{x}_{\text{des}}(t) = x_2 - \dot{x}_{\text{des}}(t) = \tilde{x}_2 \\ \dot{\tilde{x}}_2 &= \dot{x}_2 - \ddot{x}_{\text{des}}(t) = \frac{1}{m}(-cx_2 - kx_1 + u + d(t)) - \ddot{x}_{\text{des}}(t)\end{aligned} \tag{4.15}$$

by using the fact that $x_1 = \tilde{x}_1 + x_{\text{des}}(t)$ and $x_2 = \tilde{x}_2 + \dot{x}_{\text{des}}(t)$ it can be rewritten as

$$\begin{aligned}\dot{\tilde{x}}_1 &= \tilde{x}_2 \\ \dot{\tilde{x}}_2 &= \frac{1}{m}(-c(\tilde{x}_2 + \dot{x}_{\text{des}}(t)) - k(\tilde{x}_1 + x_{\text{des}}(t)) + u + d(t)) - \ddot{x}_{\text{des}}(t) \\ &= \frac{1}{m}(-c\tilde{x}_2 - k\tilde{x}_1 + u) + D(t) = f(t,\tilde{x}) + g(t,\tilde{x})u\end{aligned} \tag{4.16}$$

where $D(t) = (1/m)(-c\dot{x}_{\text{des}}(t) - kx_{\text{des}}(t) + d(t)) - \ddot{x}_{\text{des}}(t)$, $f(t,\tilde{x}) = (1/m)(-c\tilde{x}_2 - k\tilde{x}_1) + D(t)$ and $g(t,\tilde{x}) = 1/m$. Since m is known, $g(t,x)$ is also known, and assumption 2 is satisfied.

4.3.2 Lyapunov analysis

The error dynamic can be written with $u = 0$ as

$$\dot{\tilde{x}} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ D(t) \end{bmatrix} \tag{4.17}$$

The nominal system can then be written as

$$\dot{\tilde{x}} = A\tilde{x} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} \tag{4.18}$$

Let $m = 2$, $k = 2$ $c = 5$, this is the values used in the result section, Section 4.5. The eigenvalues of A is then $\lambda_1 = -0.5$ and $\lambda_2 = -2.5$, this means that $Re\{\lambda_i\} < 0$, and A is Hurwitz, Theorem A.1.1. Let $Q = I$ to maximize the ratio $\lambda_{\min}(Q)/\lambda_{\max}(P)$, then a solution to $PA + A^T P = -Q$ in Theorem A.1.1 is given by

$$P = \begin{bmatrix} 1.65 & 0.5 \\ 0.5 & 0.4 \end{bmatrix} \tag{4.19}$$

The Lyapunov function $V = \tilde{x}^T P \tilde{x}$ then satisfies

$$\begin{aligned} \lambda_{\min}(P)||\tilde{x}||_2^2 \leq V(t,\tilde{x}) \leq \lambda_{\max}(P)||\tilde{x}||_2^2 \\ \frac{\partial V}{\partial \tilde{x}} A\tilde{x} = -\tilde{x}^T Q \tilde{x} \leq -\lambda_{\min}(Q)||\tilde{x}||_2^2 \\ \left|\left|\frac{\partial V}{\partial \tilde{x}}\right|\right|_2 \leq 2\lambda_{max}(P)||\tilde{x}||_2 \end{aligned} \tag{4.20}$$

which means that the nominal system has a globally exponentially stable origin, since Theorem A.1.4 is satisfied. Lemma A.2.1 can then be used to analyse the perturbed system. Assume that $|D(t)| < \delta$.

$$\begin{aligned} \dot{V}(t,\tilde{x}) = 2\dot{\tilde{x}}^T P \tilde{x} &= 2\begin{bmatrix} \tilde{x}_2 \\ -2.5\tilde{x}_2 - \tilde{x}_1 + D(t) \end{bmatrix}^T \begin{bmatrix} 1.65 & 0.5 \\ 0.5 & 0.4 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} \\ &= 2((1.65\tilde{x}_1 + 0.5\tilde{x}_2)\tilde{x}_2 + ((-2.5\tilde{x}_2 - \tilde{x}_1 + D(t))(0.5\tilde{x}_1 + 0.4\tilde{x}_2))) \\ &= 2((-0.5\tilde{x}_1^2 - 0.5\tilde{x}_2^2) + (0.5D(t)\tilde{x}_1 + 0.4D(t)\tilde{x}_2)) \\ &\leq -||\tilde{x}||_2^2 + \delta\tilde{x}_1 + 0.8\delta\tilde{x}_2 \\ &\leq -||\tilde{x}||_2^2 + \frac{\sqrt{41}}{5}\delta||\tilde{x}||_2 \\ &= -||\tilde{x}||_2^2 + \theta||\tilde{x}||_2^2 - \theta||\tilde{x}||_2^2 + \frac{\sqrt{41}}{5}\delta||\tilde{x}||_2 \\ &\leq -(1-\theta)||\tilde{x}||_2^2 \quad \forall \quad ||\tilde{x}||_2 \geq \frac{\frac{\sqrt{41}}{5}\delta}{\theta} \end{aligned} \tag{4.21}$$

where $0 < \theta < 1$. Then, for all $||\tilde{x}(t_0)|| < \sqrt{c_1/c_2}r$, the solution $\tilde{x}(t)$ of the perturbed system

$$\dot{\tilde{x}} = f(t,\tilde{x}) + g(t,\tilde{x})u \tag{4.22}$$

satisfies

$$||\tilde{x}(t)|| \leq kexp[-\gamma(t-t_0)]||\tilde{x}(t_0)||, \quad \forall t_0 \leq t < t_0 + T \tag{4.23}$$

and

$$||\tilde{x}(t)|| \leq b, \quad \forall t \geq t_0 + T \tag{4.24}$$

for some finite T, where

$$k = \sqrt{\frac{c_2}{c_1}}, \quad \gamma = \frac{(1-\theta)c_3}{2c_2}, \quad b = \frac{c_4}{c_3}\sqrt{\frac{c_2}{c_1}}\frac{\delta}{\theta} \tag{4.25}$$

with $c_1 = \lambda_{\min}(P)$, $c_2 = \lambda_{\max}(P)$, $c_3 = \lambda_{\min}(Q)$ and $c_4 = 2\lambda_{\max}(P)$. This means that $||\tilde{x}(t)||$ is ultimately bounded by b. Note that this holds for all m, k and c that makes A Hurwitz. The assumption that $D(t)$ is bounded holds, if the disturbance $d(t)$ and the desired trajectory $x_{\text{des}}(t)$ are bounded. The latter is almost always bounded by design, and $d(t)$ is in this case assumed bounded. Since $f(t,\tilde{x})$ is a function of a bounded function $D(t)$ and bounded signals $\tilde{x}(t)$, the function $f(t,\tilde{x})$ will be bounded, which means that

$\dot{f}(t, \tilde{x})$ also will be bounded, and assumption 1 holds. The error dynamics for the mass-spring-damper system therefore satisfy all of the assumptions made by the error dynamics for the general system, and the theorems in Chapter 3 also holds for the error dynamics for the mass-spring-damper system.

4.4 Implementation

For the mass-spring damper system MATLAB Simulink was used to implement the different controllers, observers and the system model. Code for running the systems can be found in the Appendix D.1.

4.4.1 System model

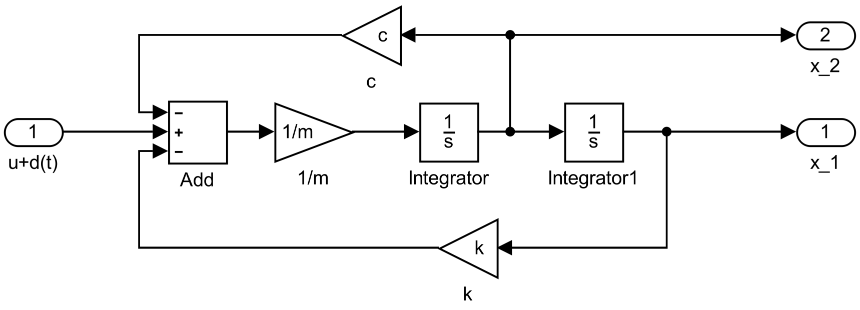


Figure 4.2: Implementation of the mass-spring-damper system

4.4.2 Sliding surface

Sliding surface 1

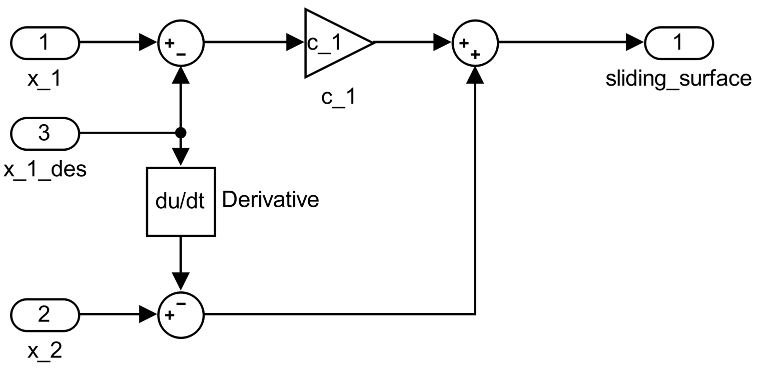


Figure 4.3: Implementation of sliding surface 1

Sliding surface 2

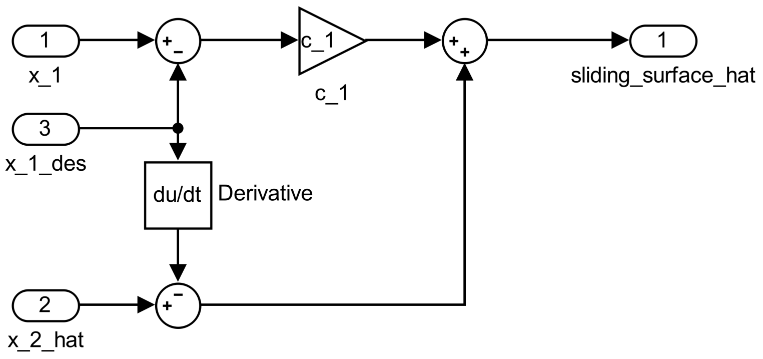


Figure 4.4: Implementation of sliding surface 2

Sliding surface 3

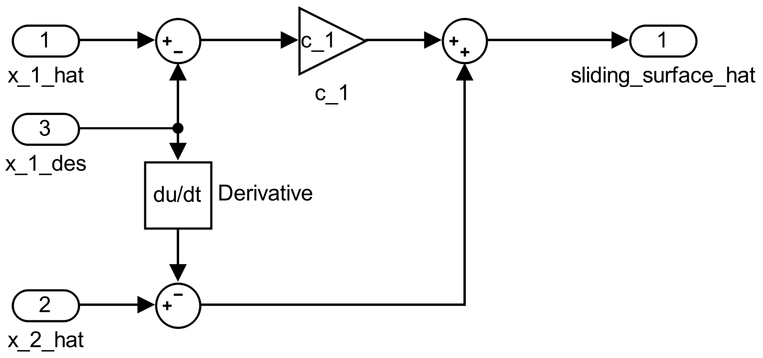


Figure 4.5: Implementation of sliding surface 3

4.4.3 Observer and differentiator

State observer

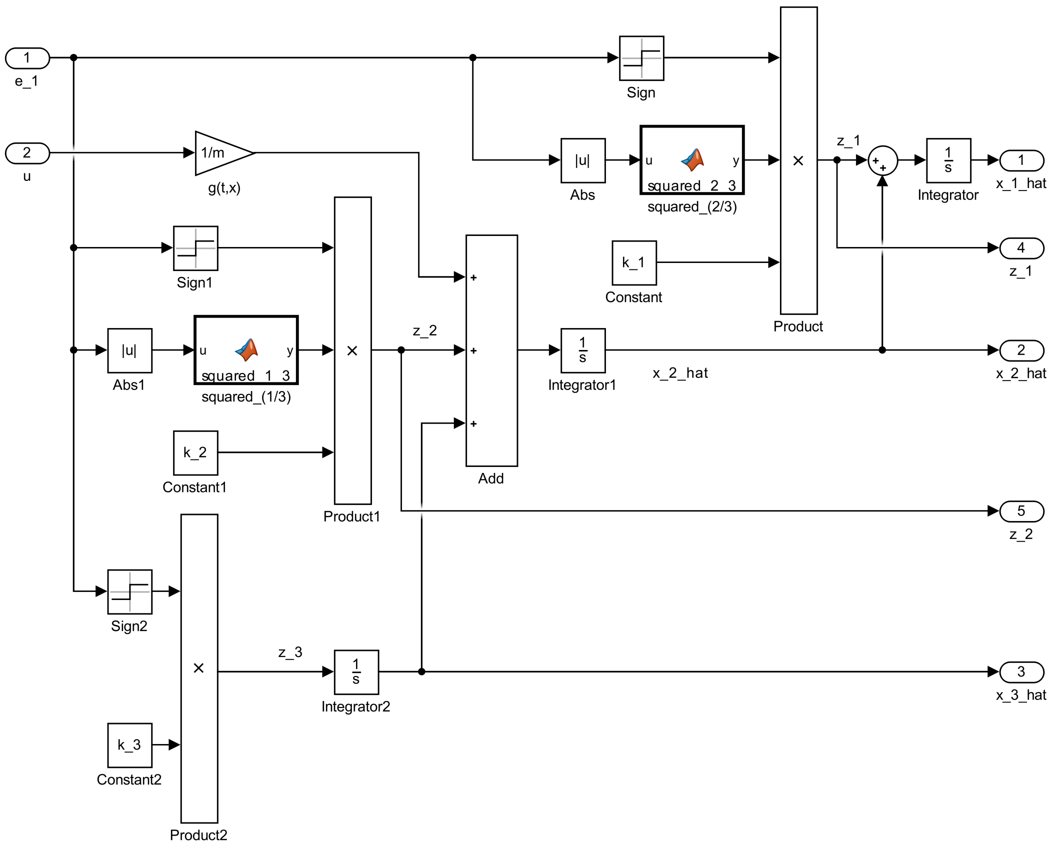


Figure 4.6: Implementation of the state observer

Differentiator

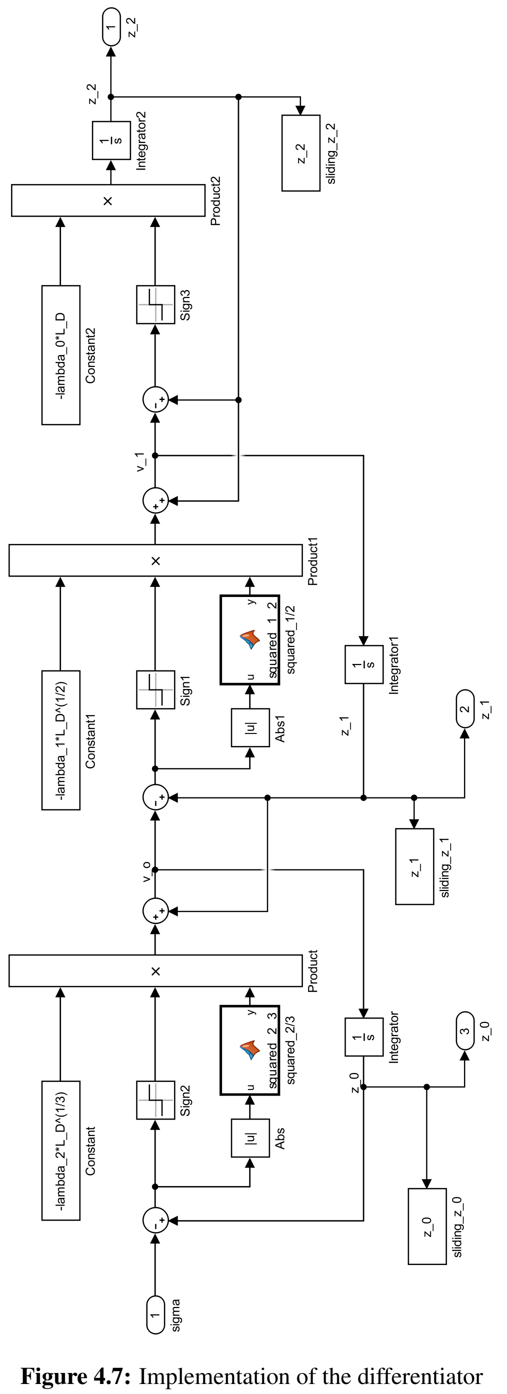


Figure 4.7: Implementation of the differentiator

4.4.4 Control input

Sliding surface 1

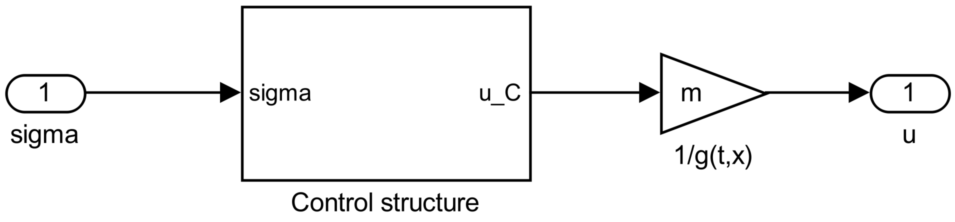


Figure 4.8: Implementation of the control input, sliding surface 1

Sliding surface 2

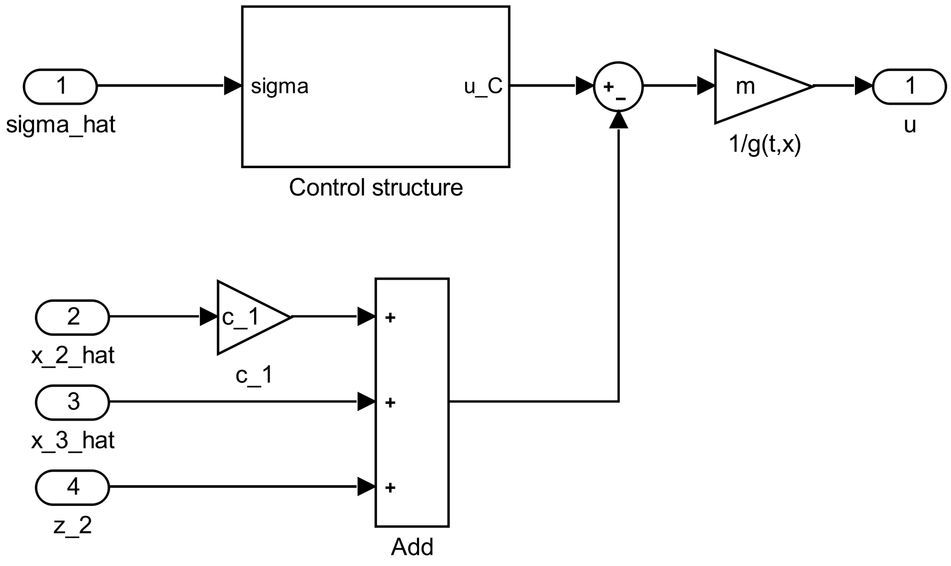


Figure 4.9: Implementation of the control input, sliding surface 2

Sliding surface 3

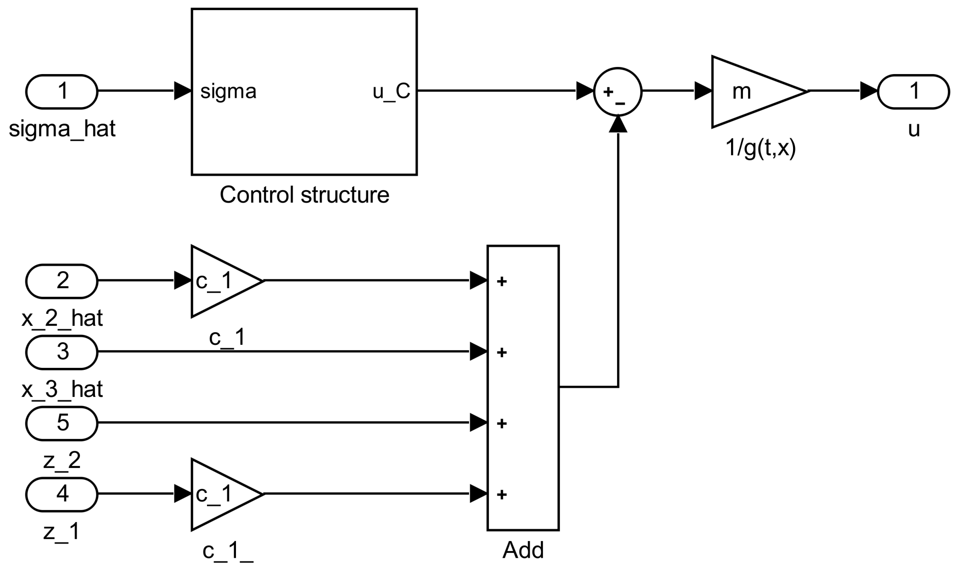


Figure 4.10: Implementation of the control input, sliding surface 3

Super-twisting algorithm

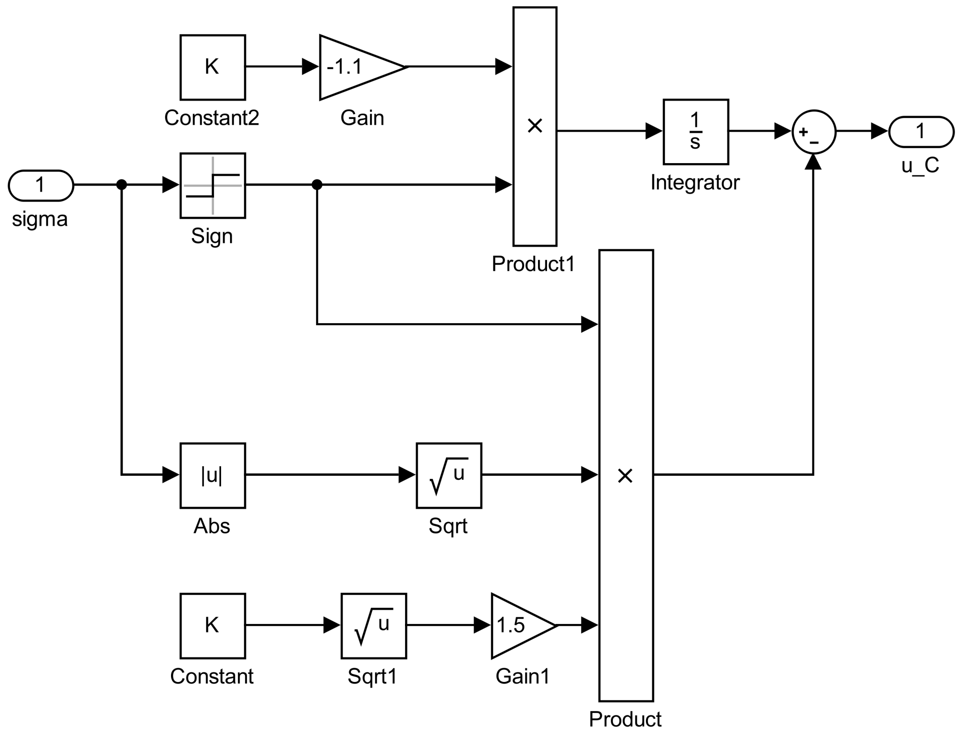


Figure 4.11: Implementation of the STA

Third-order sliding mode control

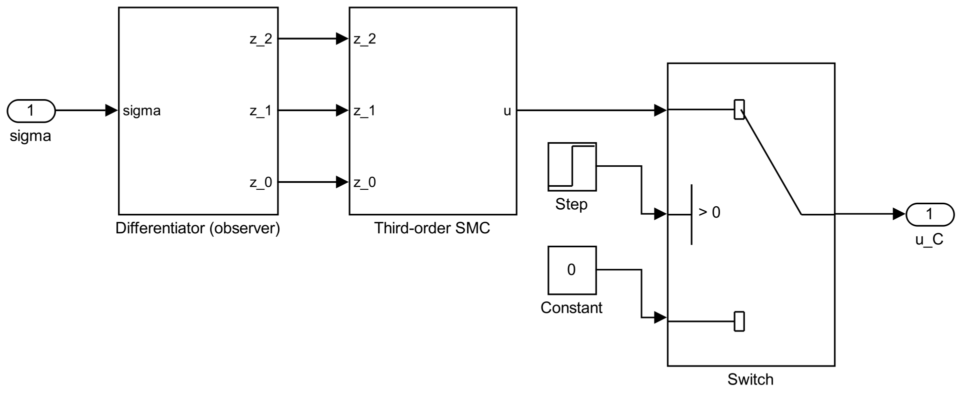


Figure 4.12: Implementation of the third-order SMC with differentiator

Nested third-order sliding mode controller

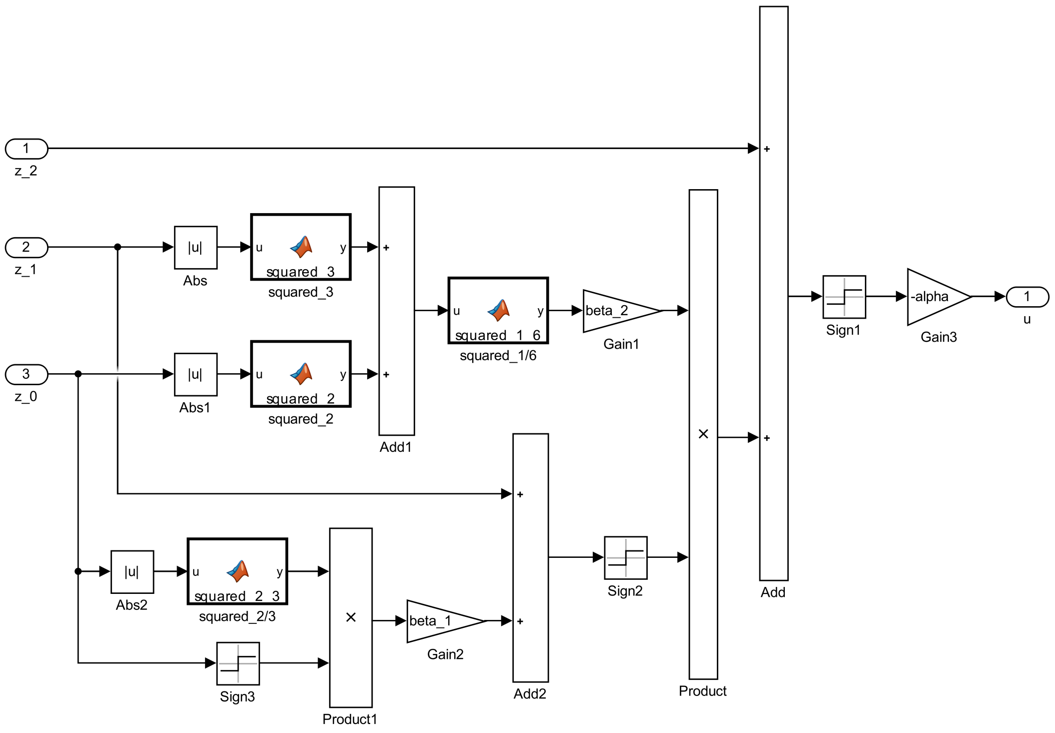


Figure 4.13: Implementation of the nested third-order SMC

Quasi-continuous third-order sliding mode controller

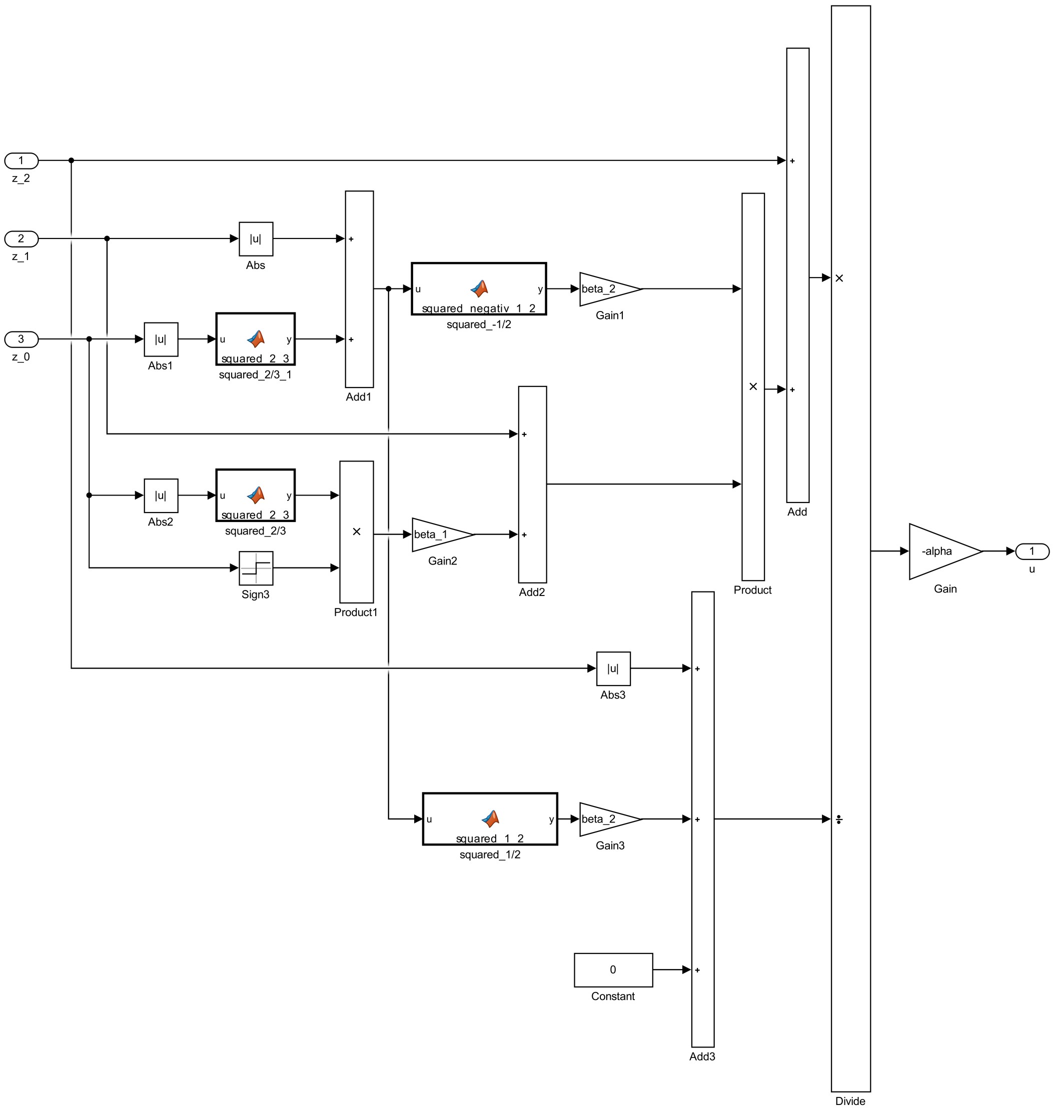


Figure 4.14: Implementation of the quasi-continuous third-order SMC

4.4.5 Complete system:

Sliding surface 1

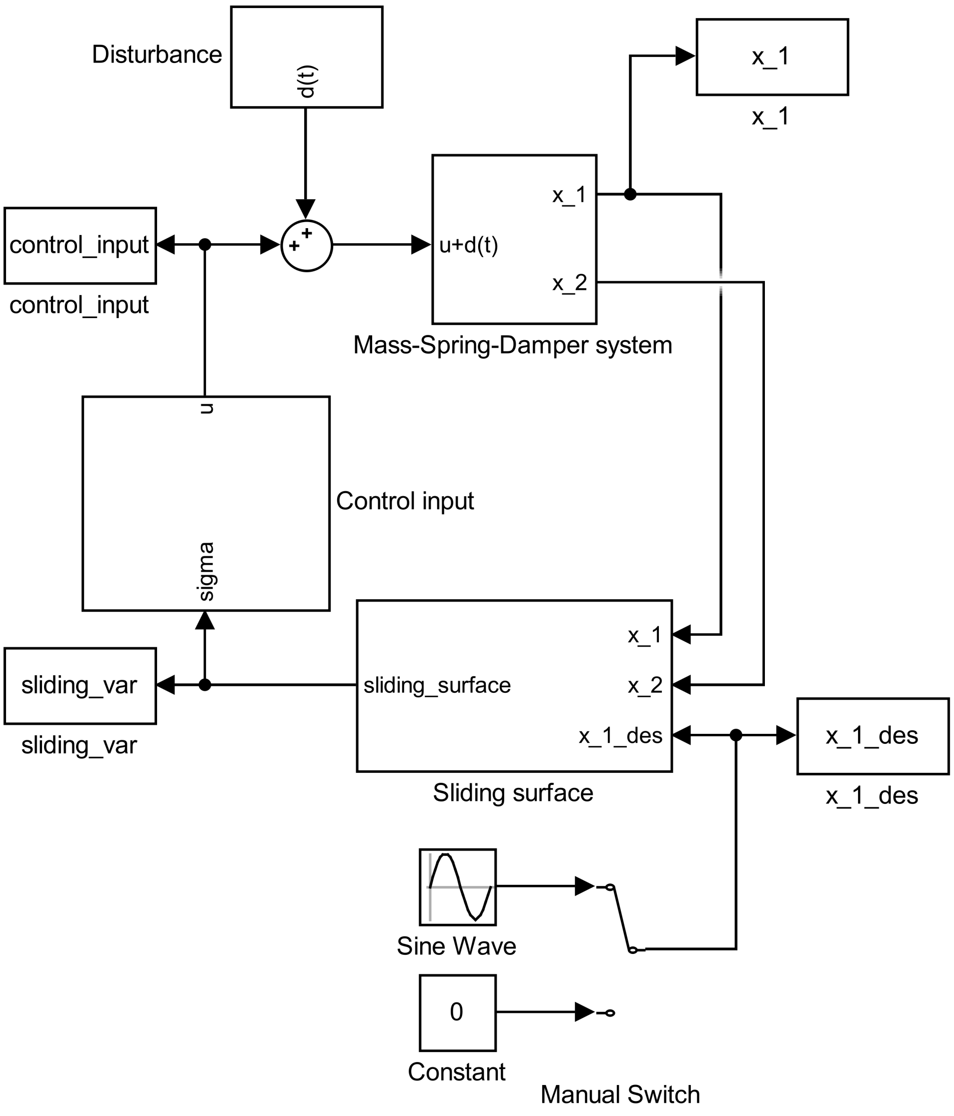


Figure 4.15: Implementation of the complete system, sliding surface 1

Sliding surface 2

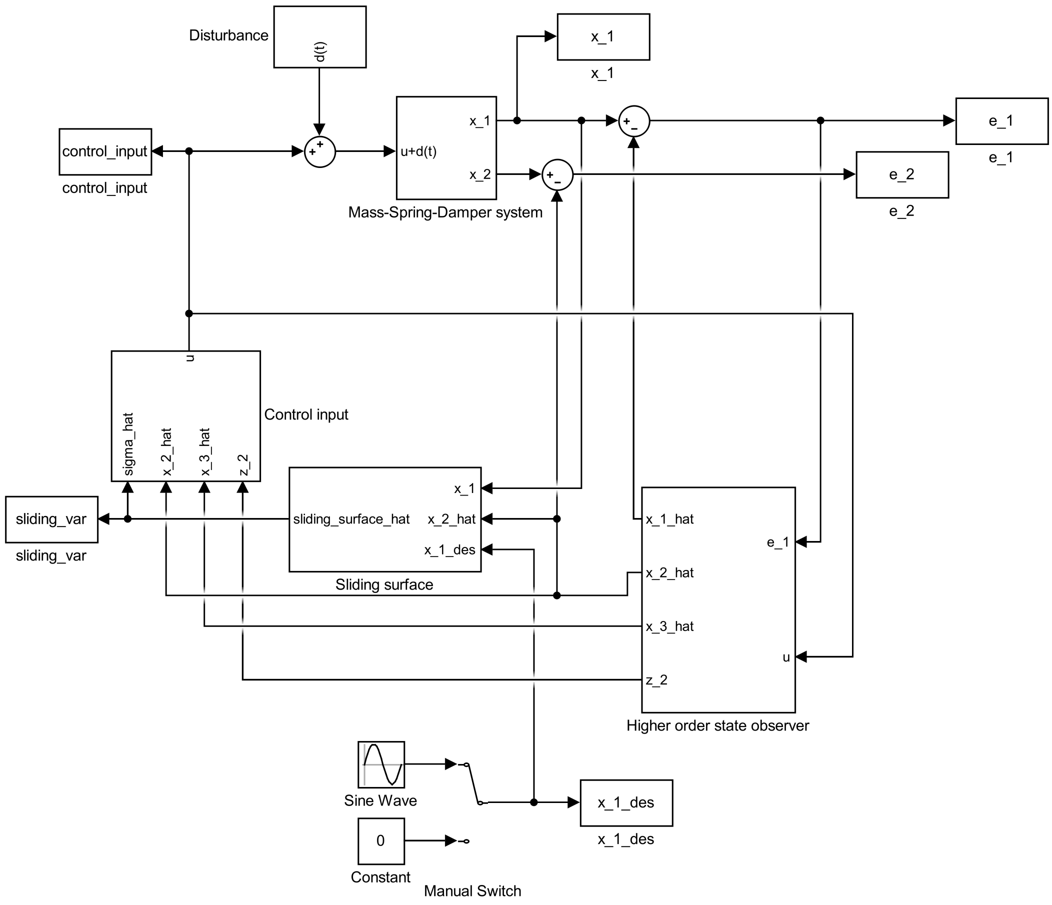


Figure 4.16: Implementation of the complete system, sliding surface 2

Sliding surface 3

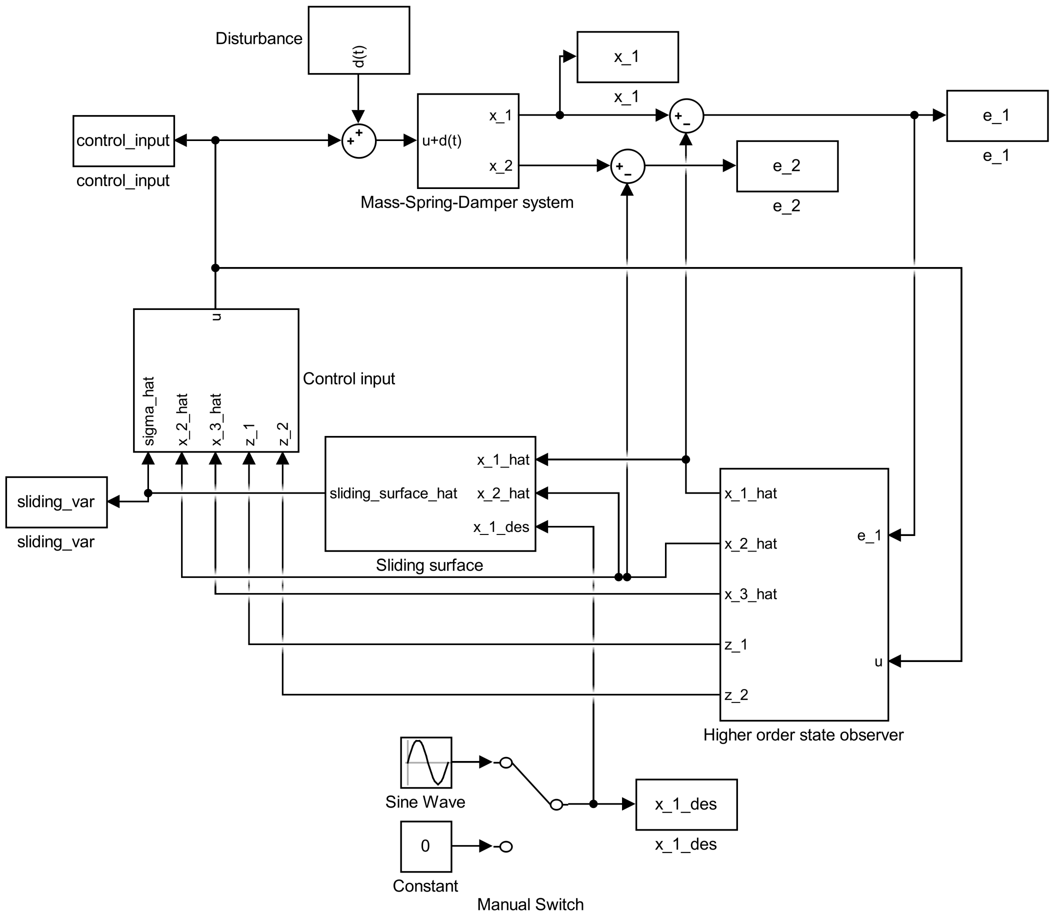


Figure 4.17: Implementation of the complete system, sliding surface 3

4.5 Results

For the simulations the constants in the mass-spring-damper system are set to: $m = 2$, $c = 5$ and $k = 2$. The disturbance is set to: $d(t) = 2\sin(3t) + \sin(5t) + 2$, and the reference input is set to $x_{des} = 5\sin(2t)$. The sufficiently large K, L_{SO}, L_D and α that were mentioned in the previous section were chosen with the trial and error method. This is because it is difficult finding a bound on the disturbance. The gains were found by starting with a very high or low gain, and then increasing or decreasing it until the error started to increase. The gain with the lowest error was then chosen.

4.5.1 The super-twisting algorithm

Sliding surface 1: the actual value of x_1 and x_2 are used
The gain was set to: $K = 100$.

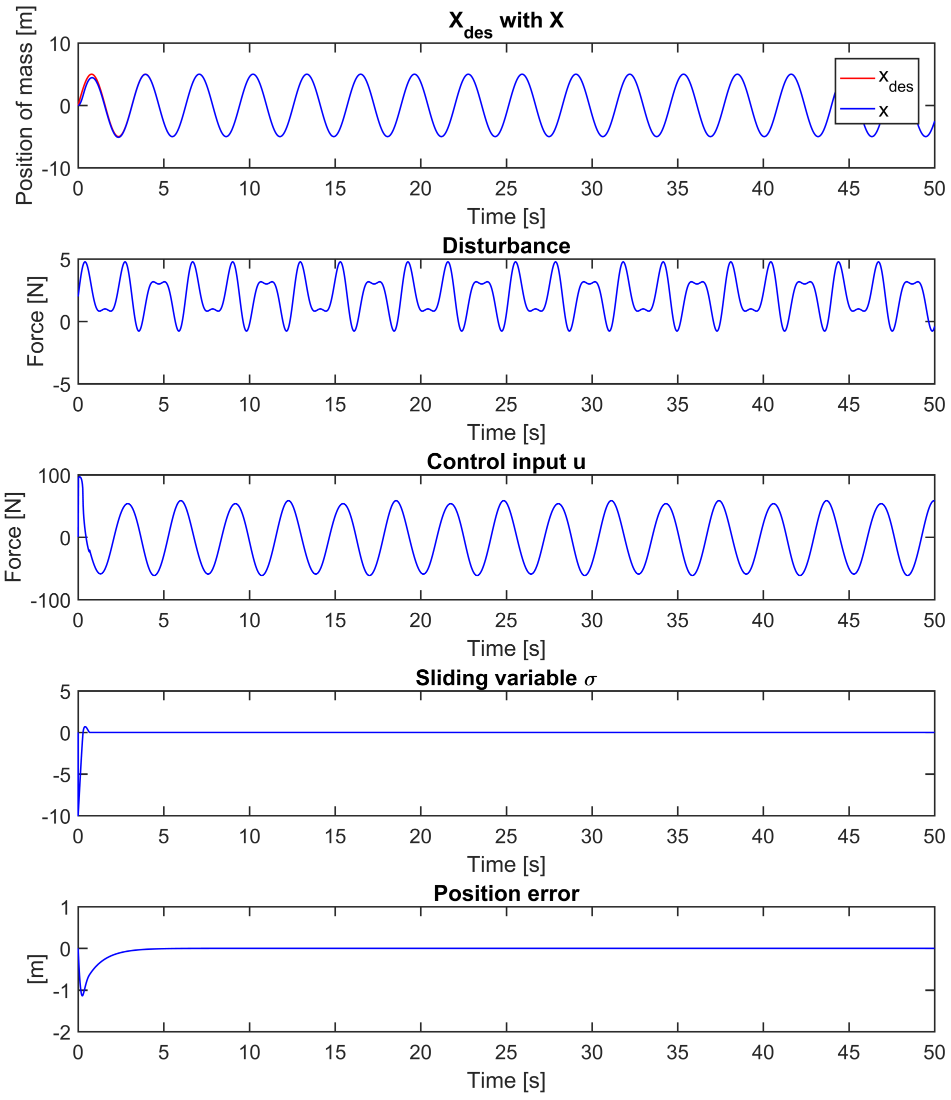


Figure 4.18: MSDS: Simulation of STA, sliding surface 1

The gains were set to: $K = 75$ and $L_{\mathrm{SO}} = 16$.
Sliding surface 2: the estimated value for x_2 is used

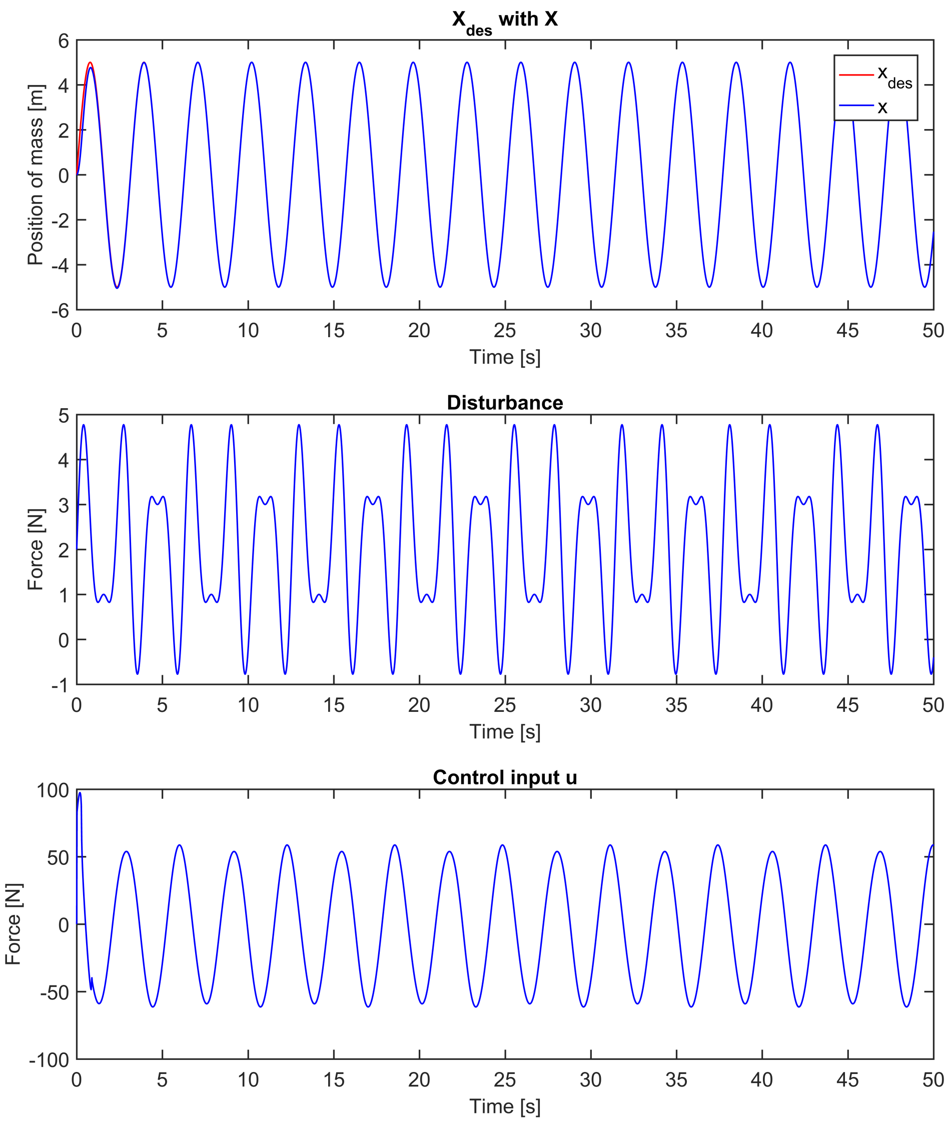


Figure 4.19: MSDS: Simulation of STA with state observer, sliding surface 2 (a)

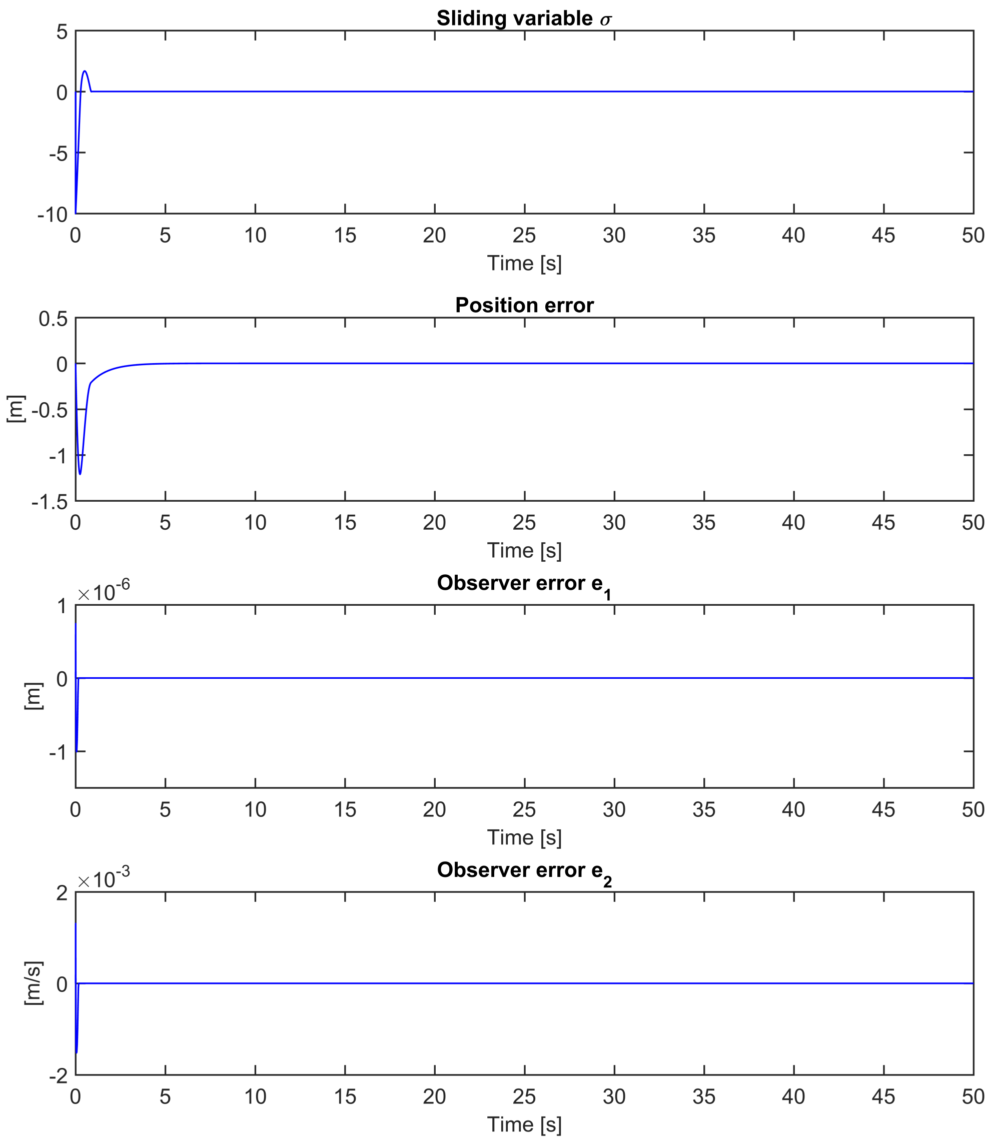


Figure 4.20: MSDS: Simulation of STA with state observer, sliding surface 2 (b)

Sliding surface 3: the estimated value for x_1 and x_2 are used

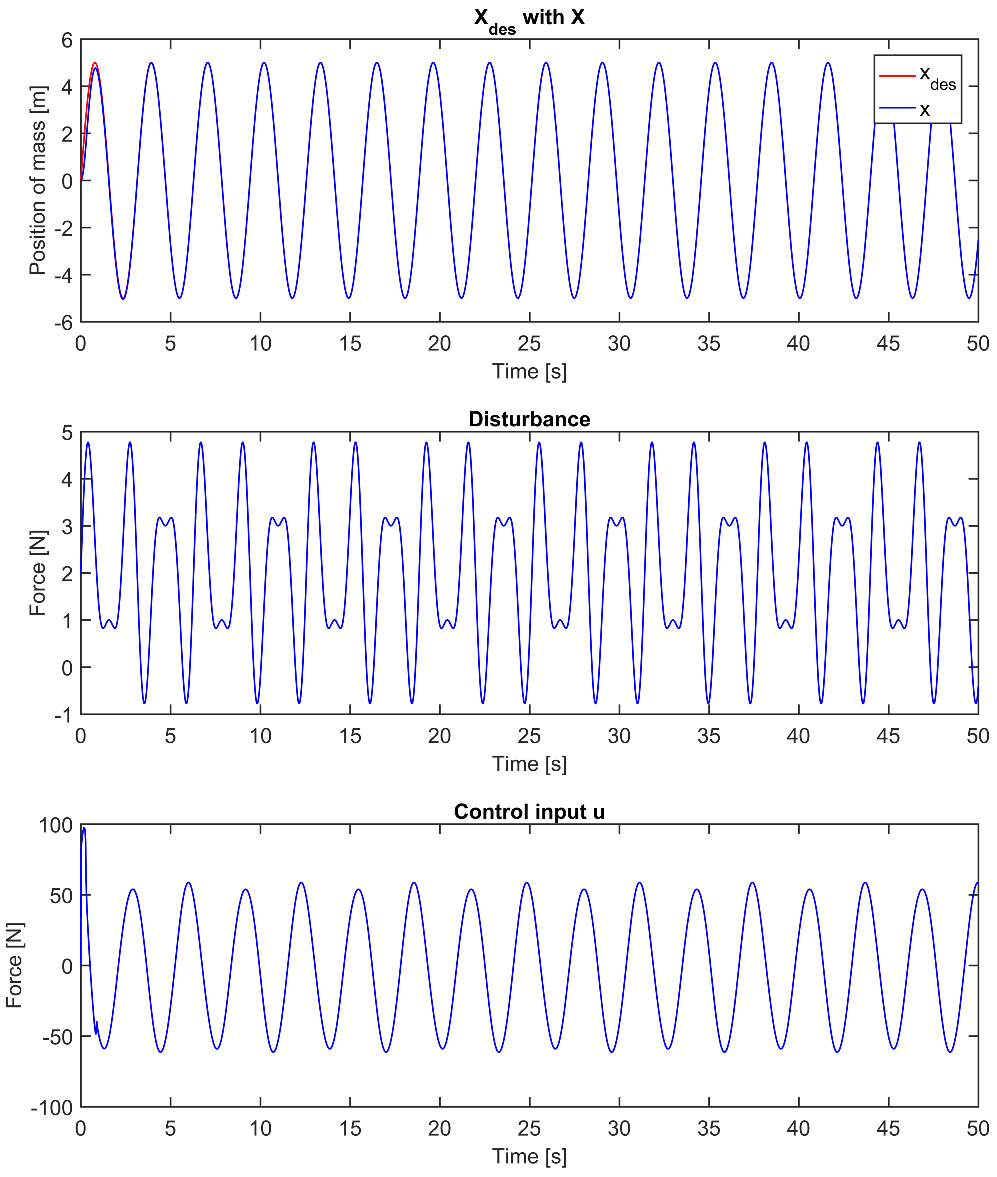


Figure 4.21: MSDS: Simulation of STA with state observer, sliding surface 3 (a)

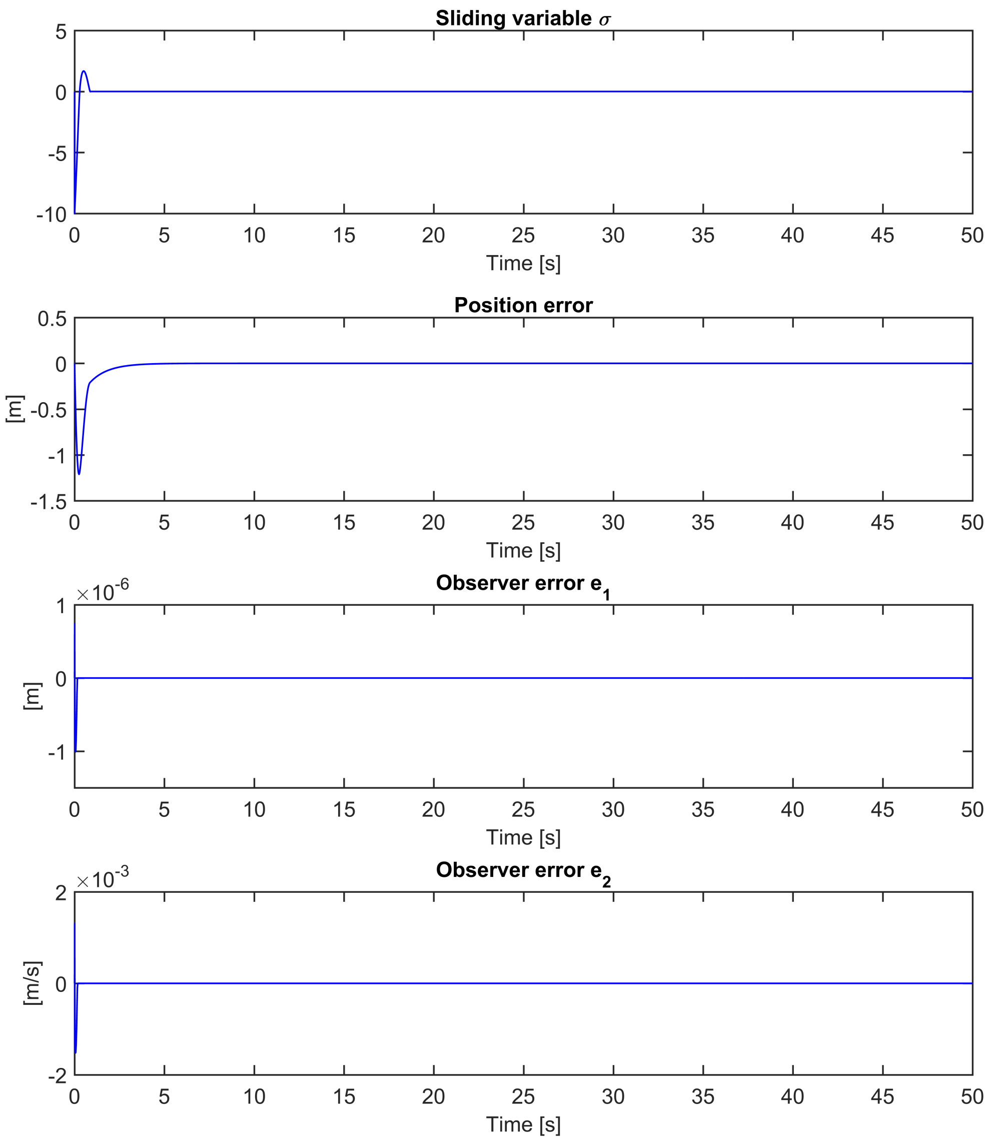


Figure 4.22: MSDS: Simulation of STA with state observer, sliding surface 3 (b)

4.5.2 Nested third-order sliding mode control

Sliding surface 1: the actual value of x_1 and x_2 are used
The gains were set to: $\alpha = 40$ and $L_D = 20$.

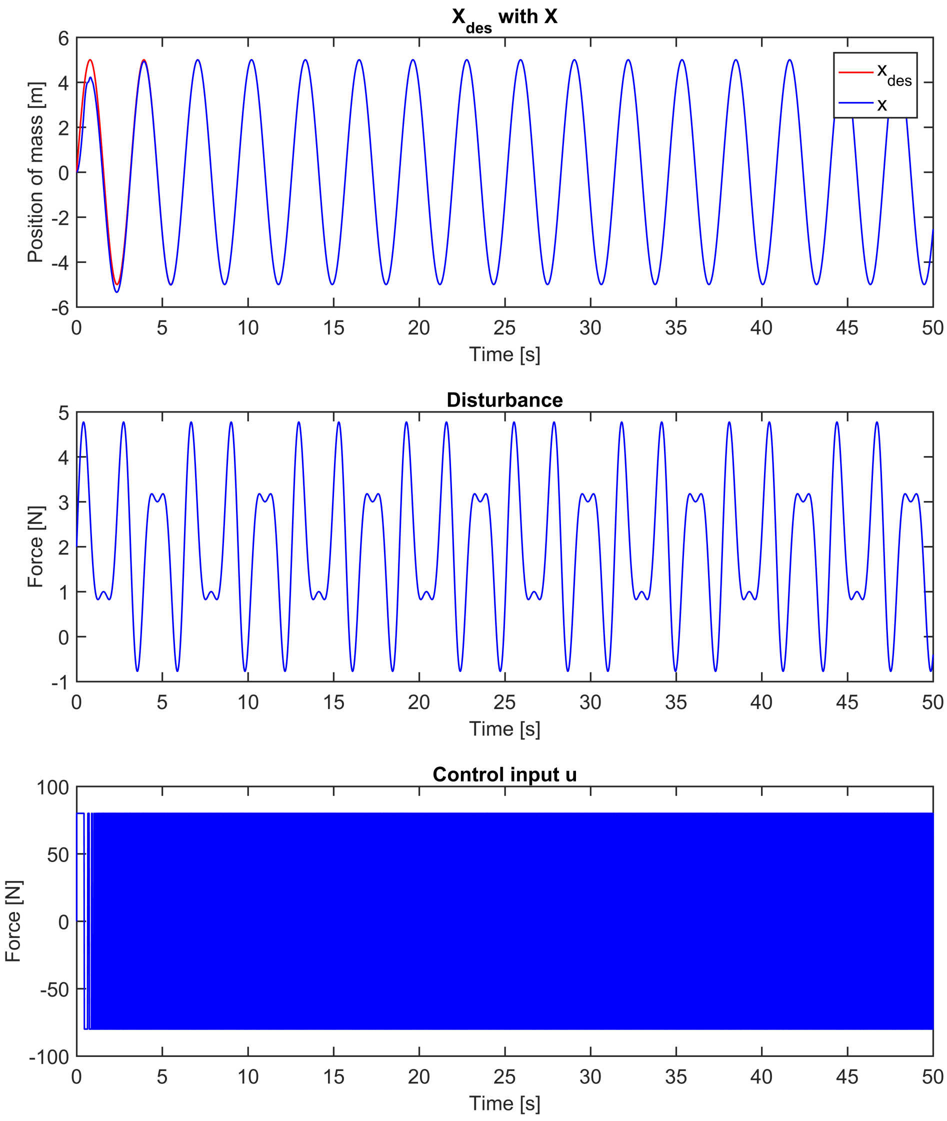


Figure 4.23: MSDS: Simulation of nested third-order SMC with differentiator, sliding surface 1 (a)

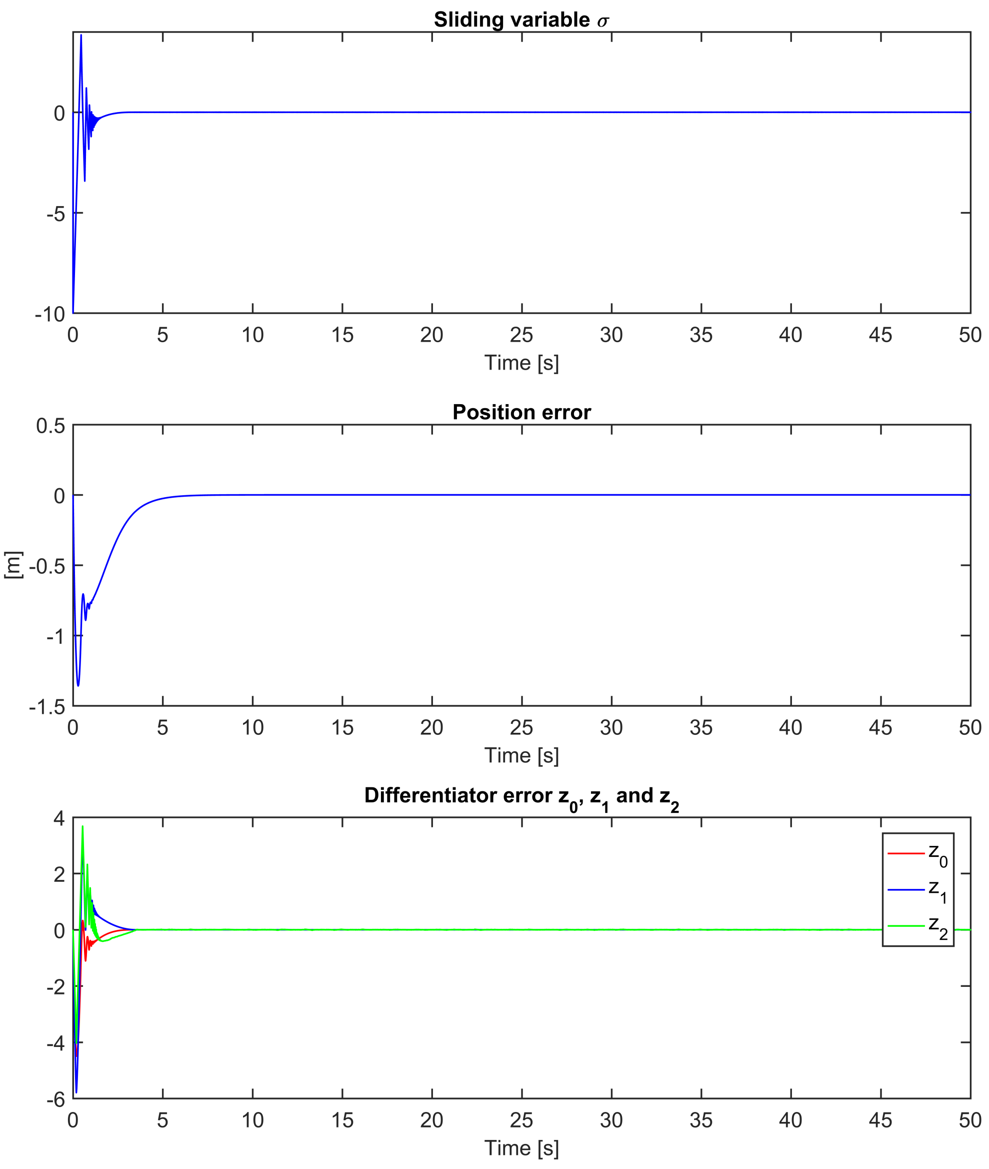


Figure 4.24: MSDS: Simulation of nested third-order SMC with differentiator, sliding surface 1 (b)

Sliding surface 2: the estimated value for x_2 is used
The gains were set to: $\alpha = 28$, $L_{\text{SO}} = 16$ and $L_D = 20$.

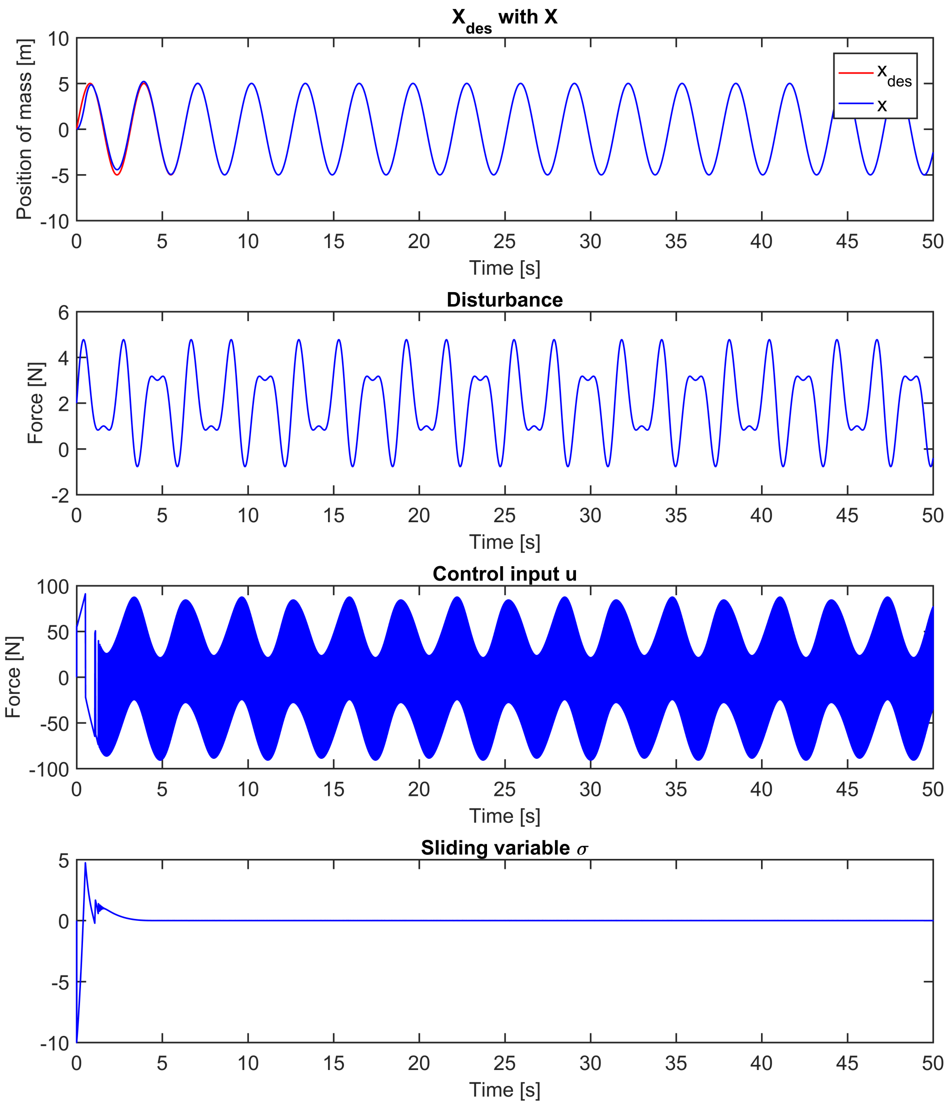


Figure 4.25: MSDS: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 2 (a)

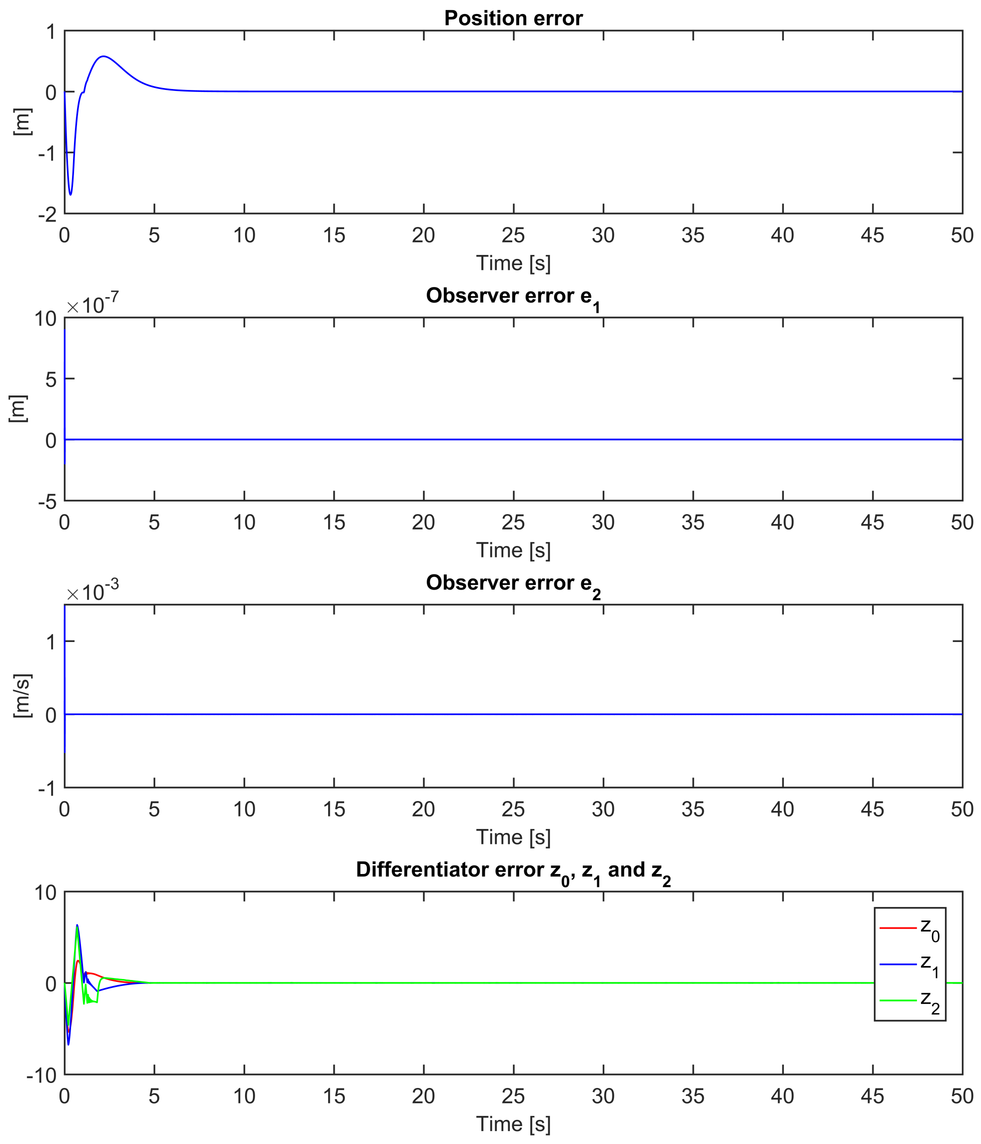


Figure 4.26: MSDS: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 2 (b)

Sliding surface 3: the estimated value for x_1 and x_2 are used
The gains were set to: $\alpha = 28$, $L_{\text{SO}} = 16$ and $L_D = 5$.

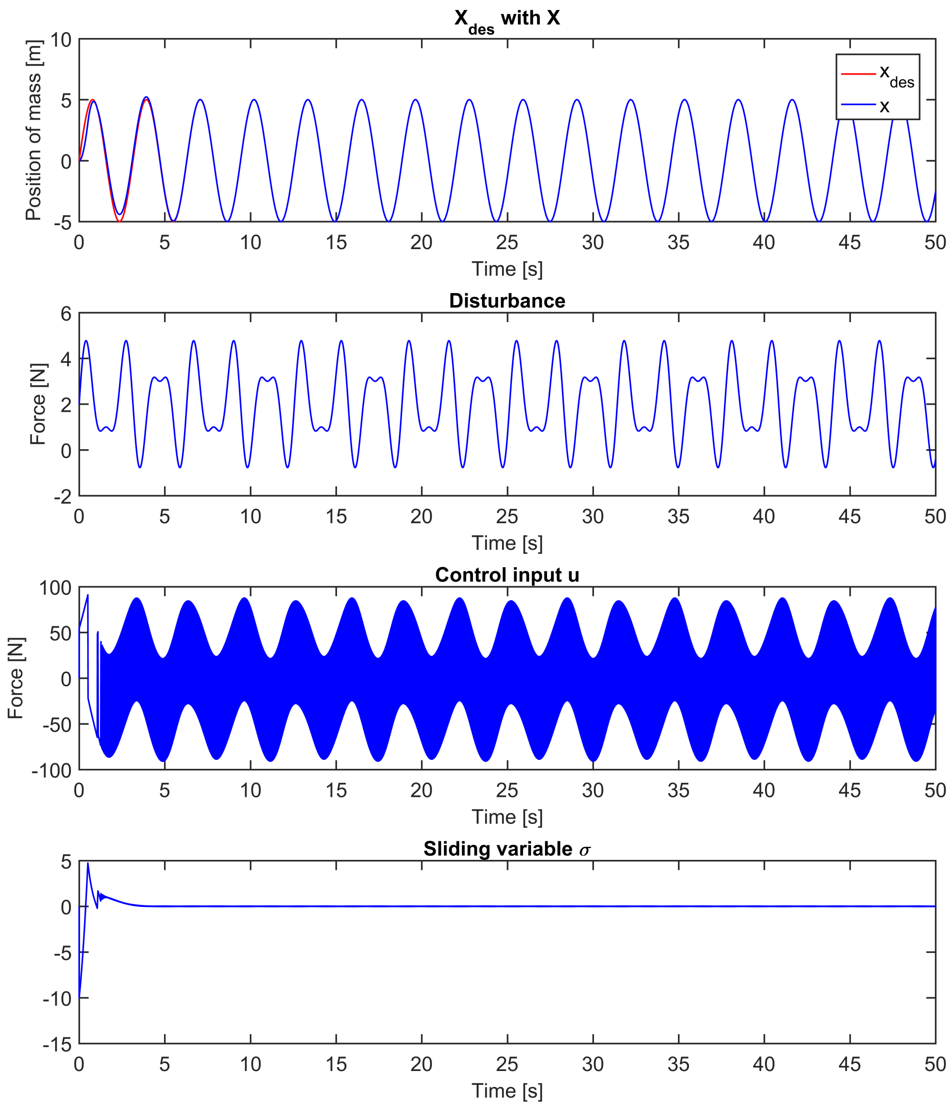


Figure 4.27: MSDS: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 3 (a)

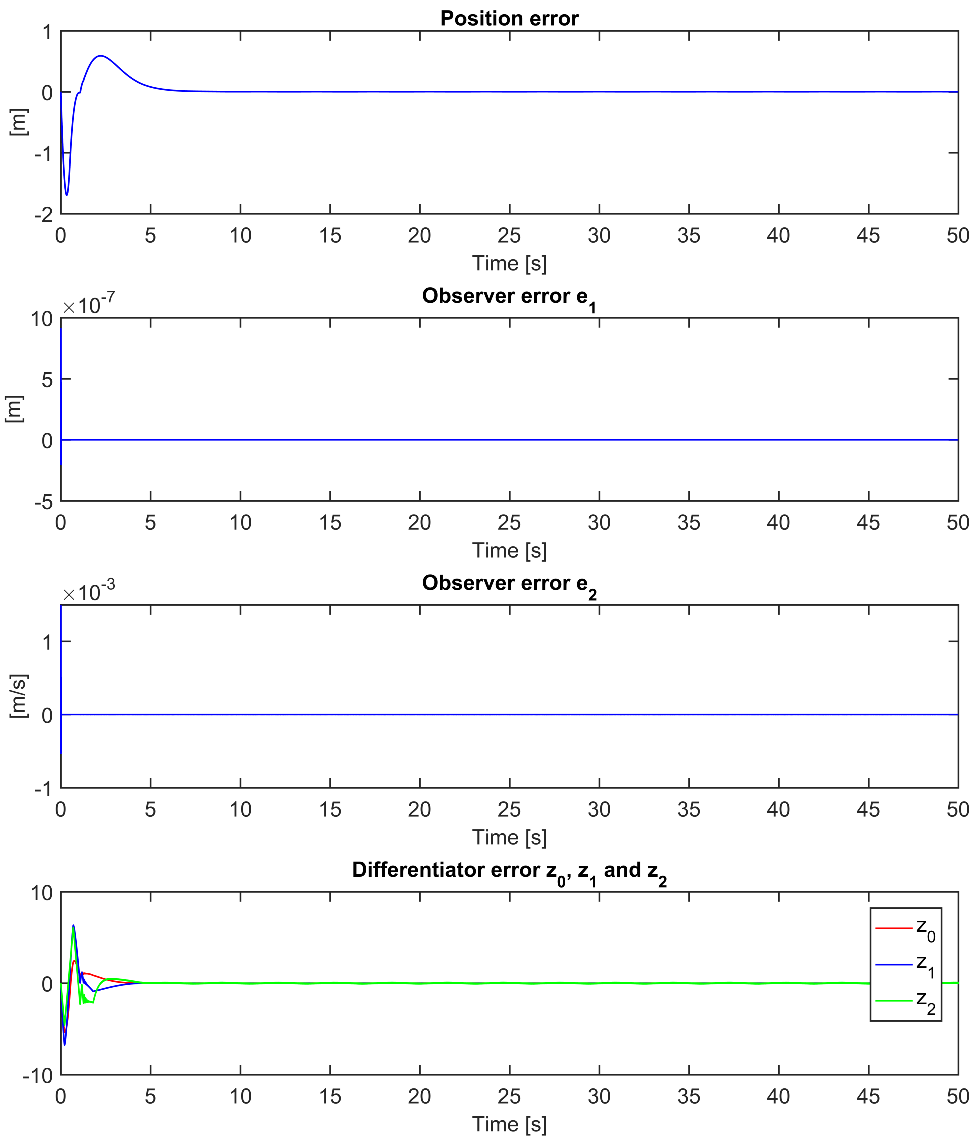


Figure 4.28: MSDS: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 3 (b)

4.5.3 Quasi-continuous third-order sliding mode control

Sliding surface 1: the actual value of x_1 and x_2 are used
The gains were set to: $\alpha = 40$ and $L_D = 1$.

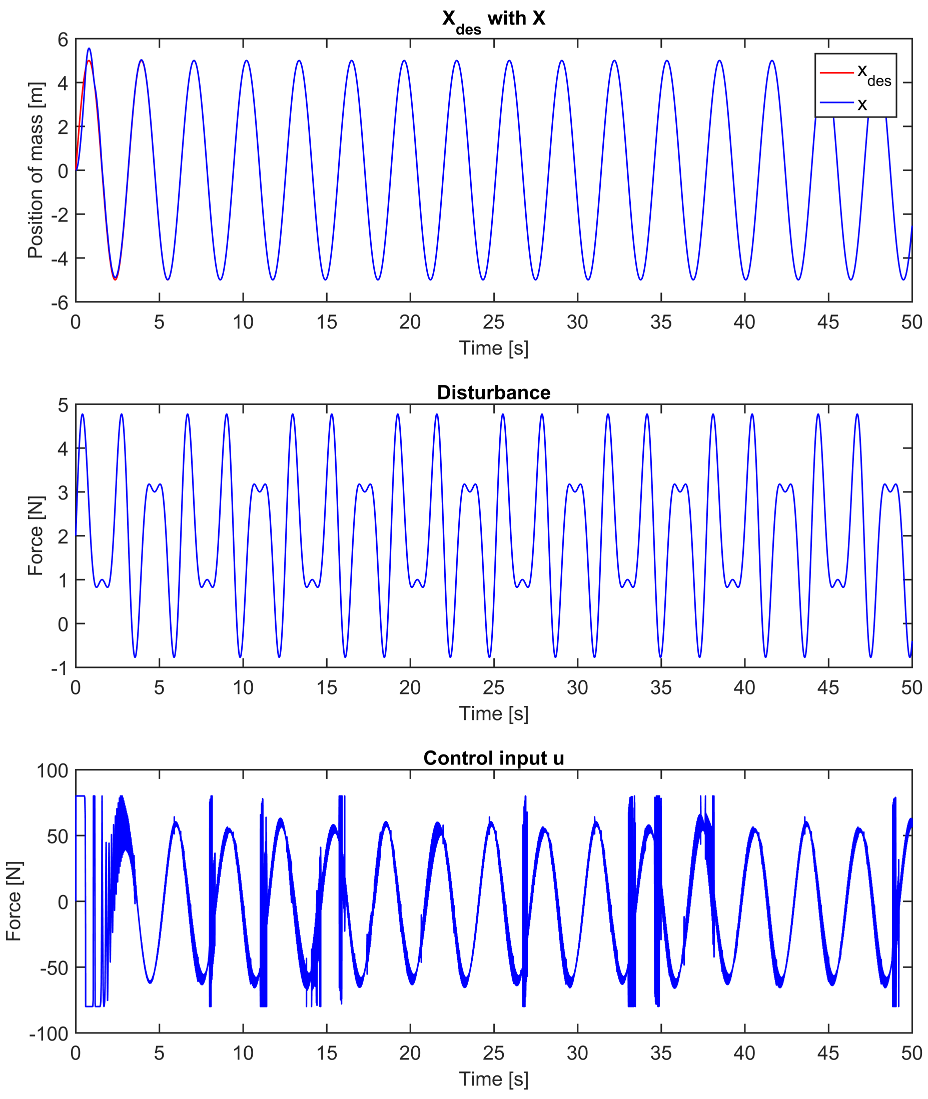


Figure 4.29: MSDS: Simulation of quasi-continuous third-order SMC with differentiator, sliding surface 1 and $\alpha = 40$ (a)

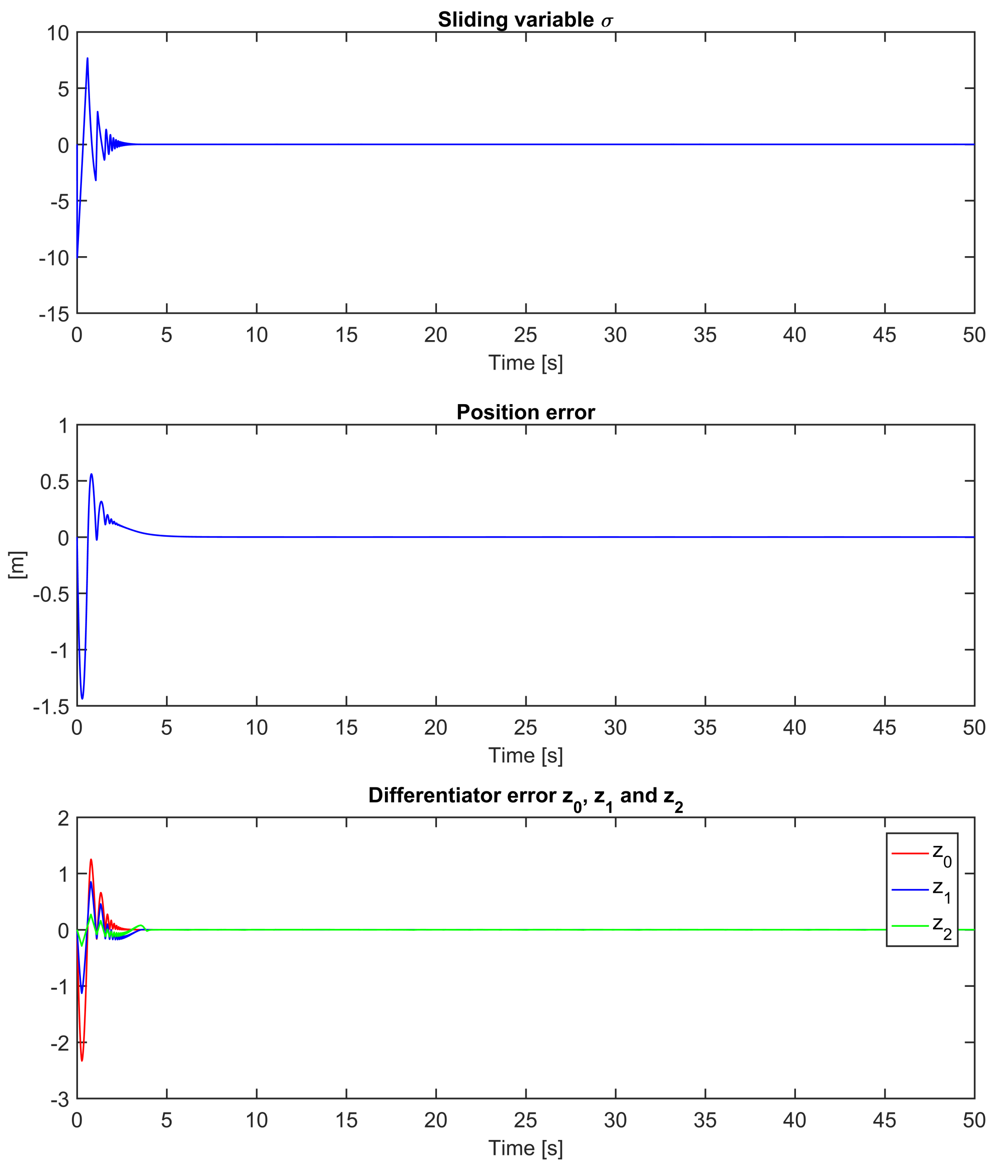


Figure 4.30: MSDS: Simulation of quasi-continuous third-order SMC with differentiator, sliding surface 1 and $\alpha = 40$ (b)

The gains where set to: $\alpha = 30$, $L_{\text{SO}} = 16$ and $L_D = 1$.
Sliding surface 2: the estimated value for x_2 is used

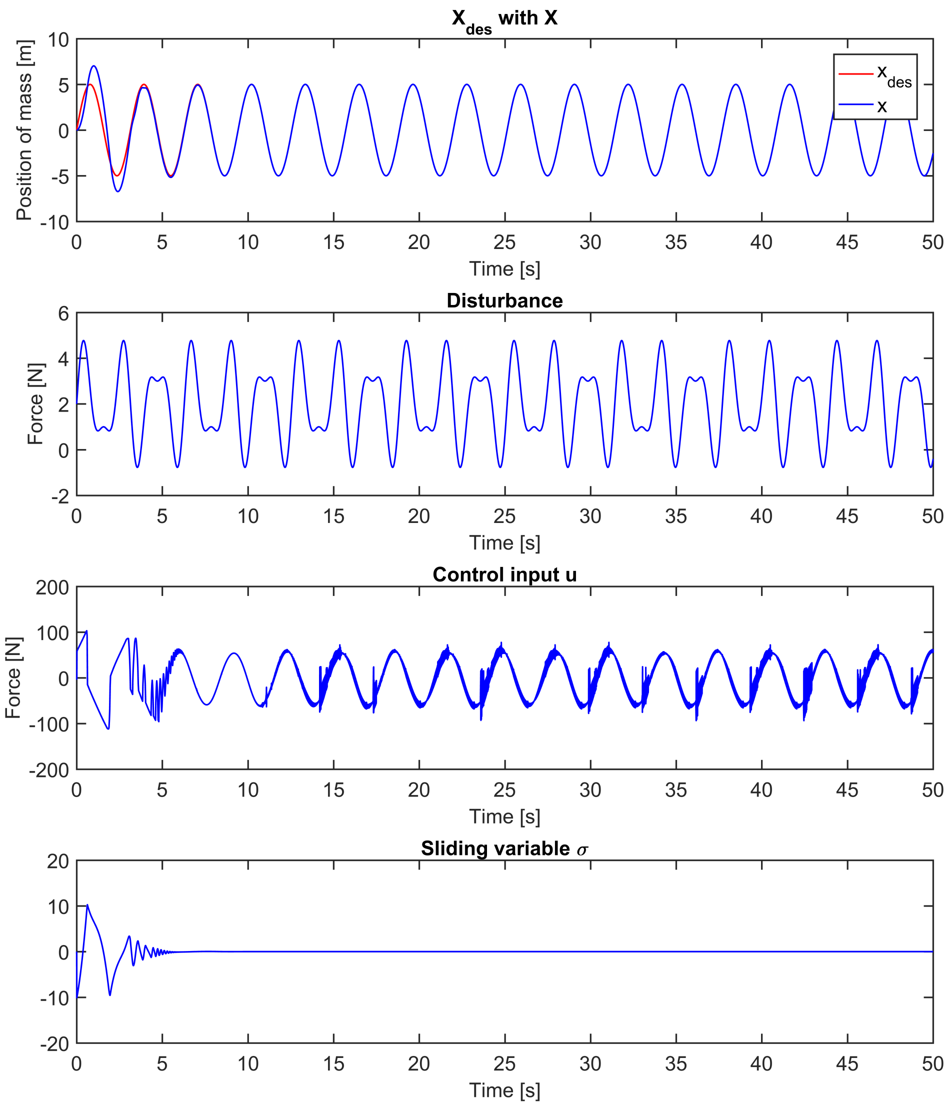


Figure 4.31: MSDS: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 2 and $\alpha = 30$ (a)

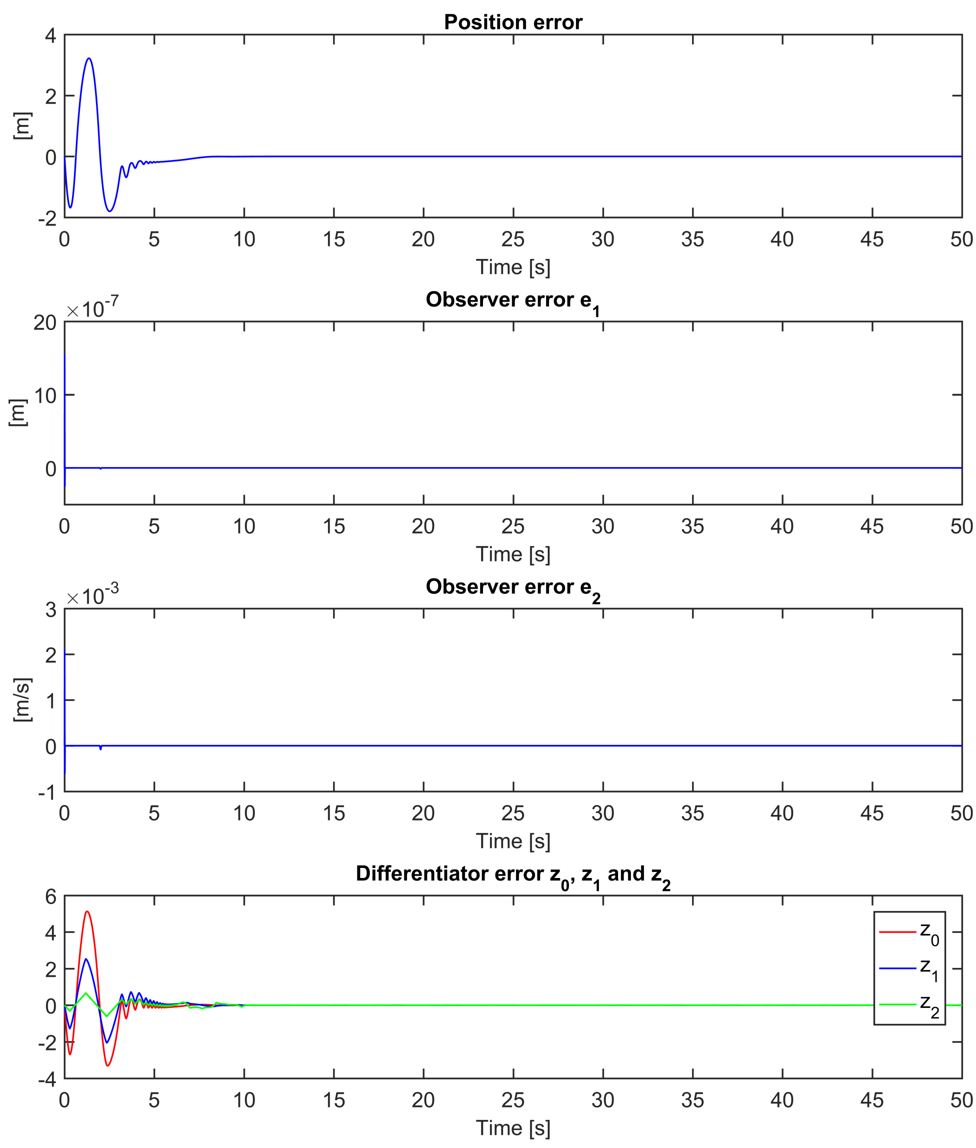


Figure 4.32: MSDS: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 2 and $\alpha = 30$ (b)

Sliding surface 3: the estimated value for x_1 and x_2 are used

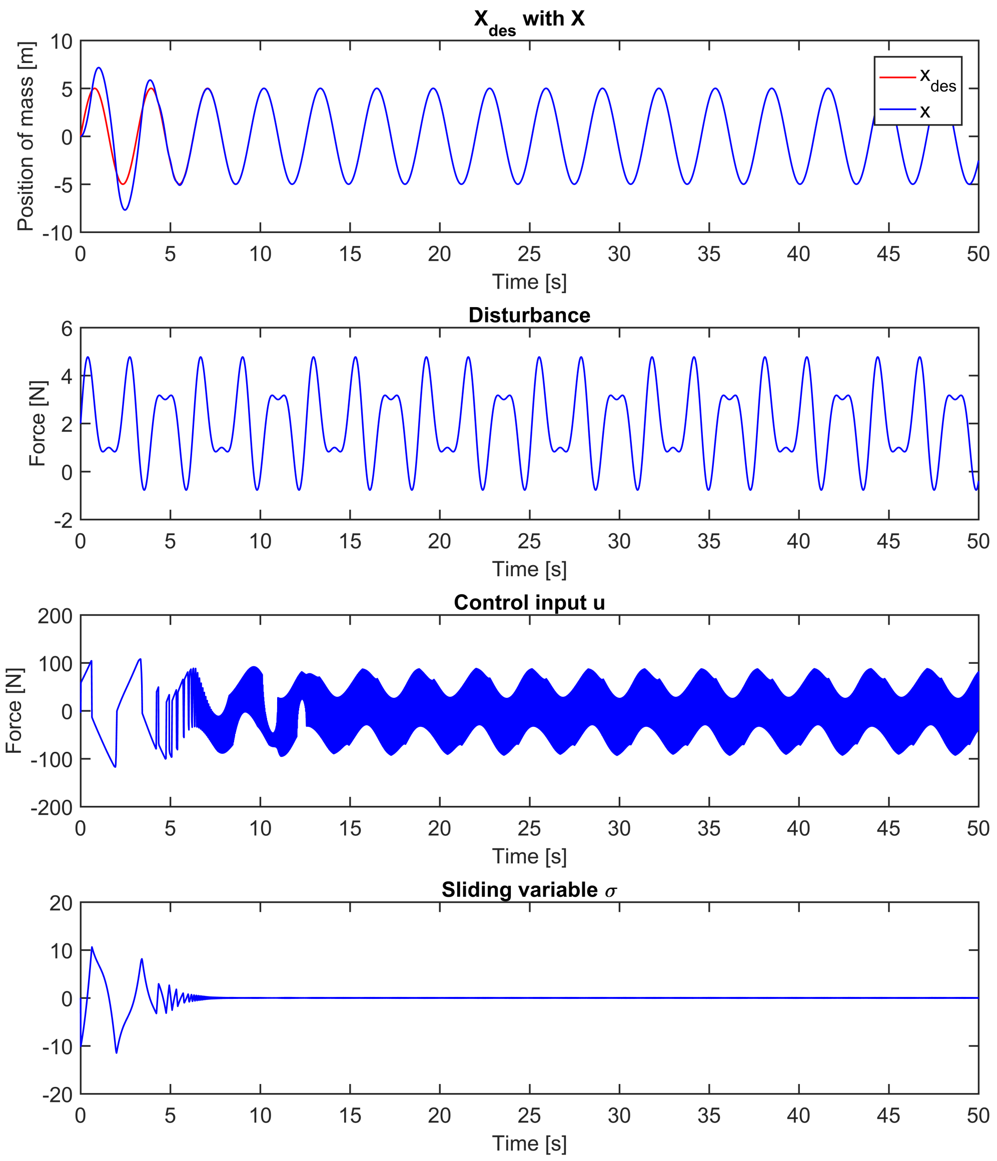


Figure 4.33: MSDS: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 3 and $\alpha = 30$ (a)

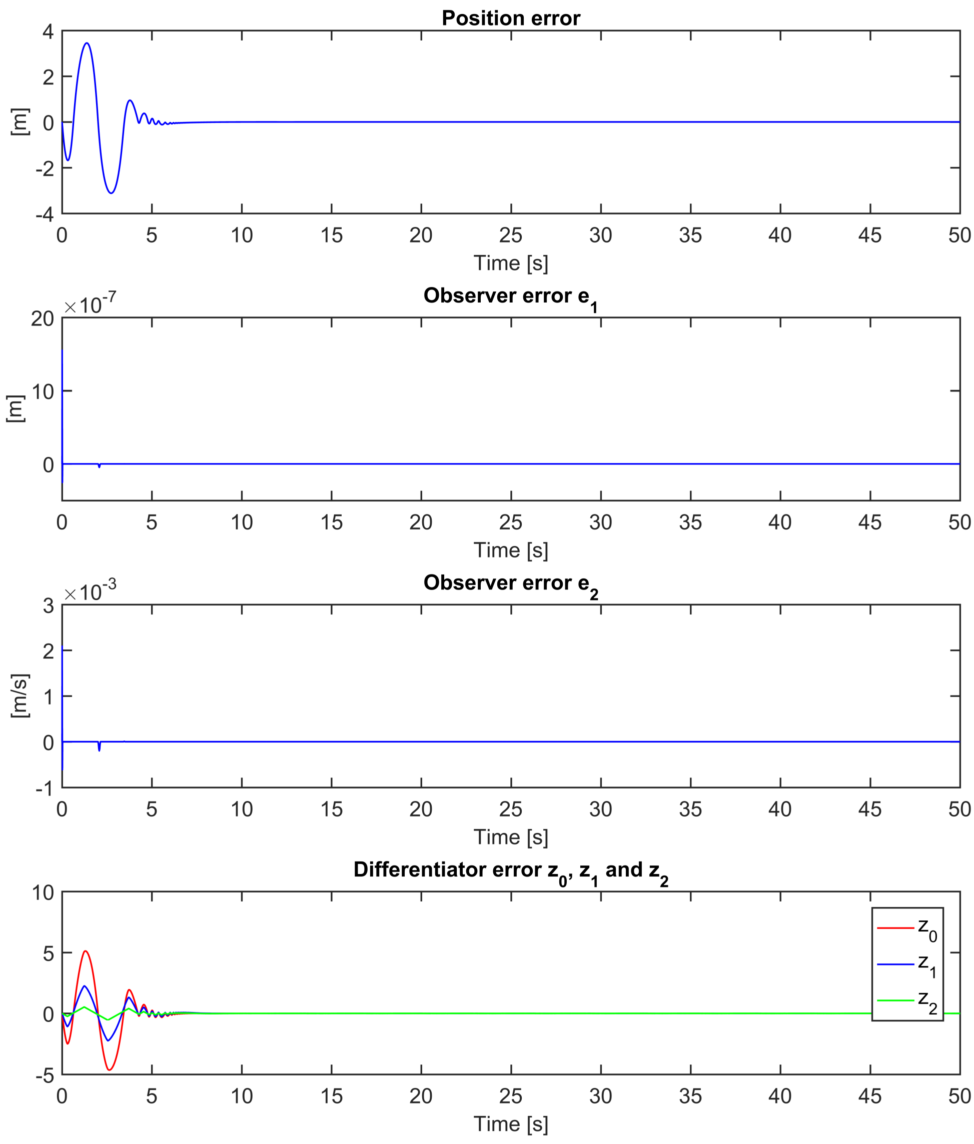


Figure 4.34: MSDS: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 3 and $\alpha = 30$ (b)

Sliding surface 1: the actual value of x_1 and x_2 are used
The gains were set to: $\alpha = 46$ and $L_D = 20$.

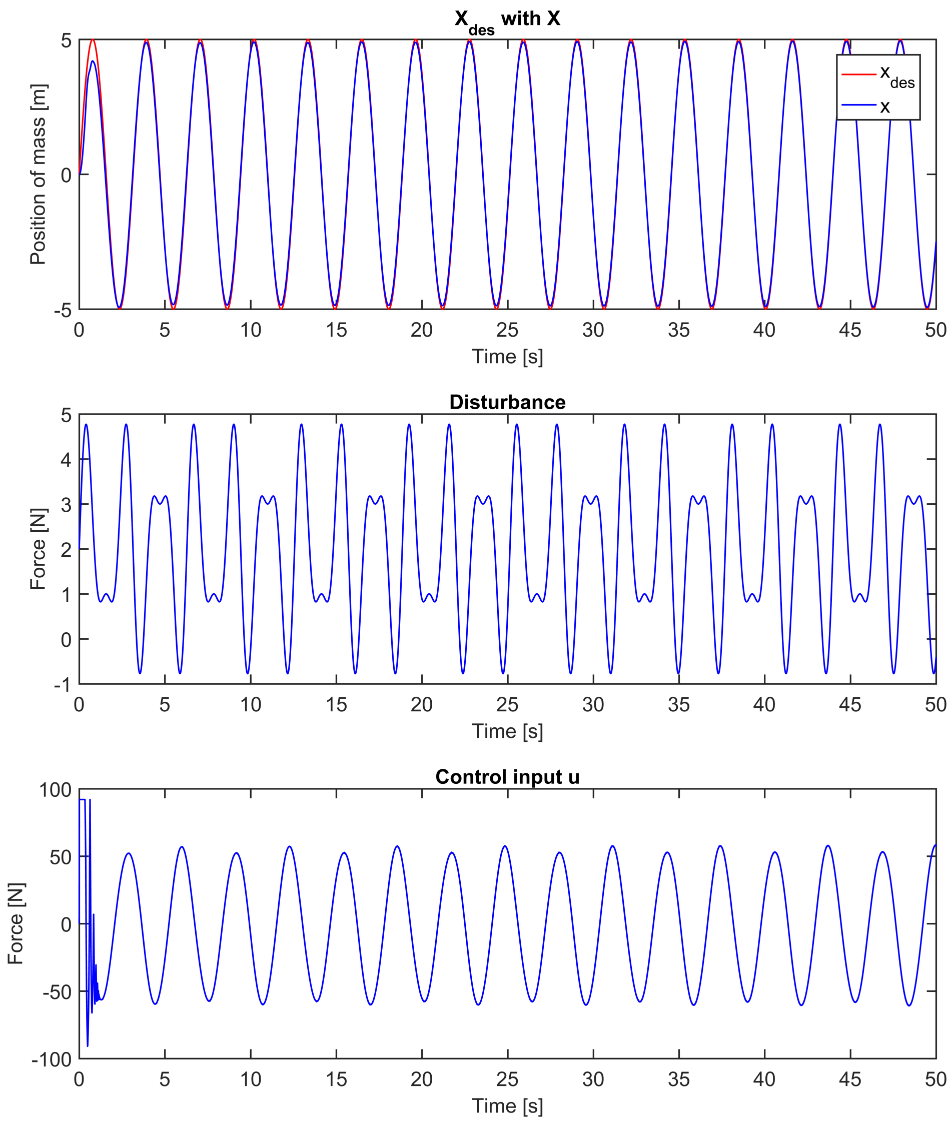


Figure 4.35: MSDS: Simulation of quasi-continuous third-order SMC with differentiator, sliding surface 1 and $\alpha = 46$ (a)

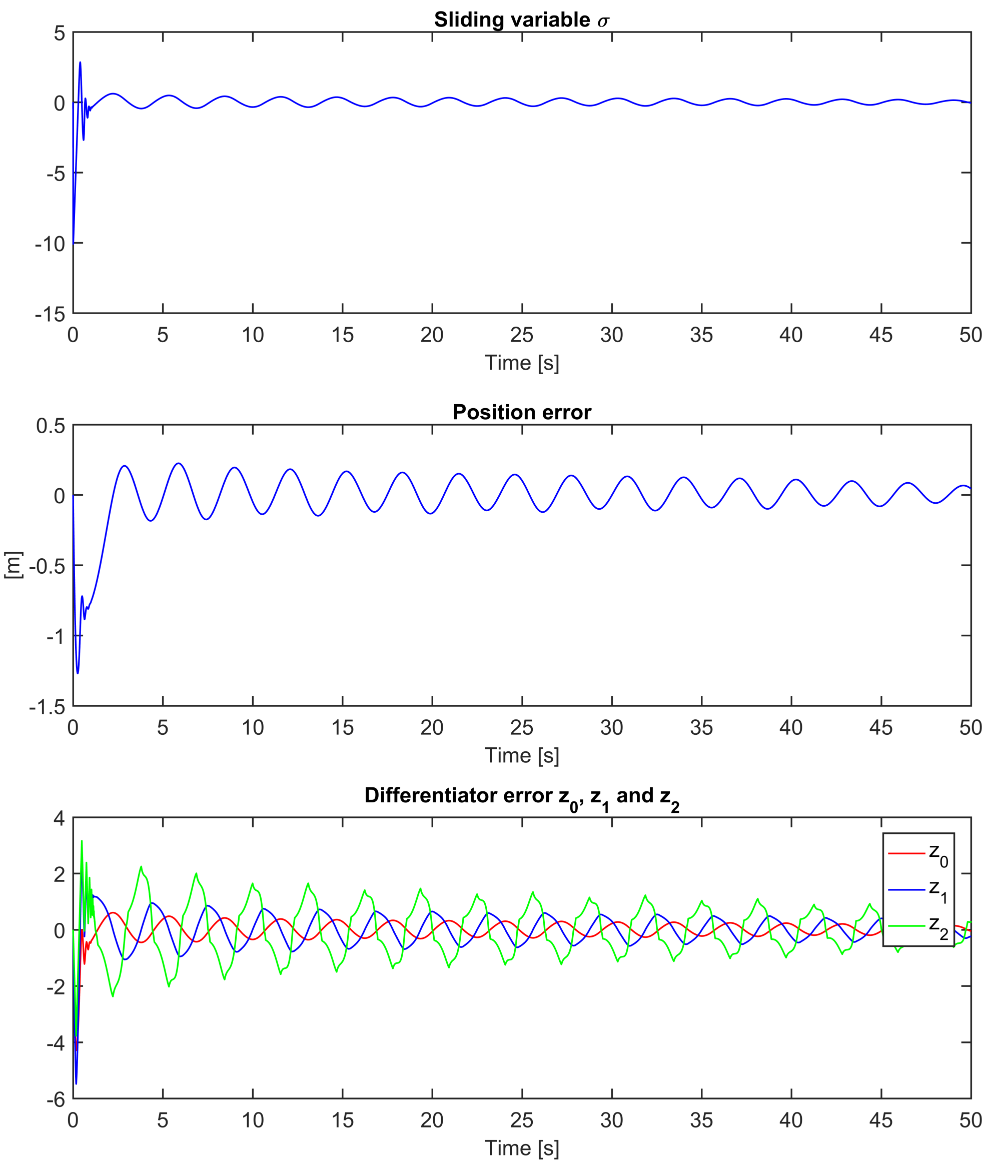


Figure 4.36: MSDS: Simulation of quasi-continuous third-order SMC with differentiator, sliding surface 1 and $\alpha = 46$ (b)

Sliding surface 2: the estimated value for x_2 is used
The gains were set to: $\alpha = 35.25$, $L_{\text{SO}} = 16$ and $L_D = 20$.

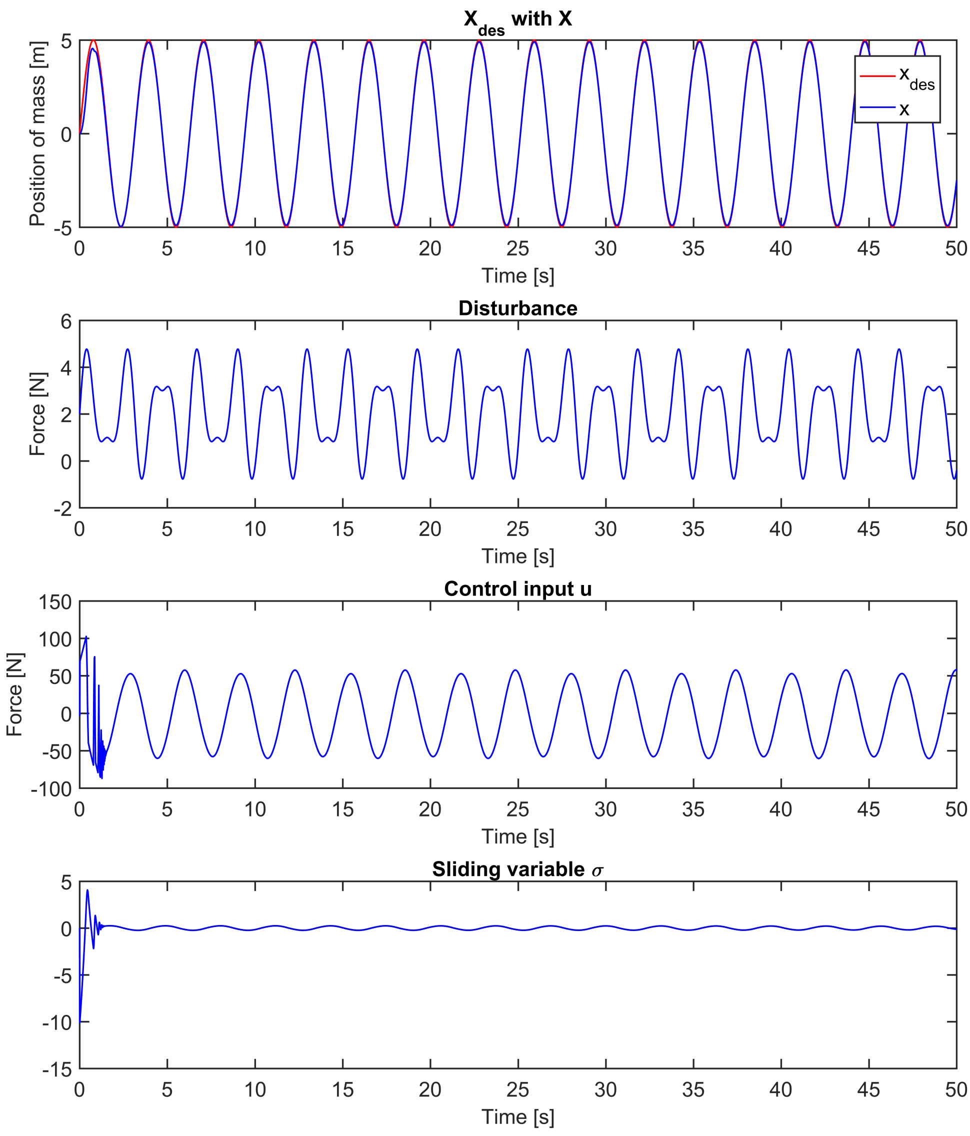


Figure 4.37: MSDS: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 2 and $\alpha = 35.25$ (a)

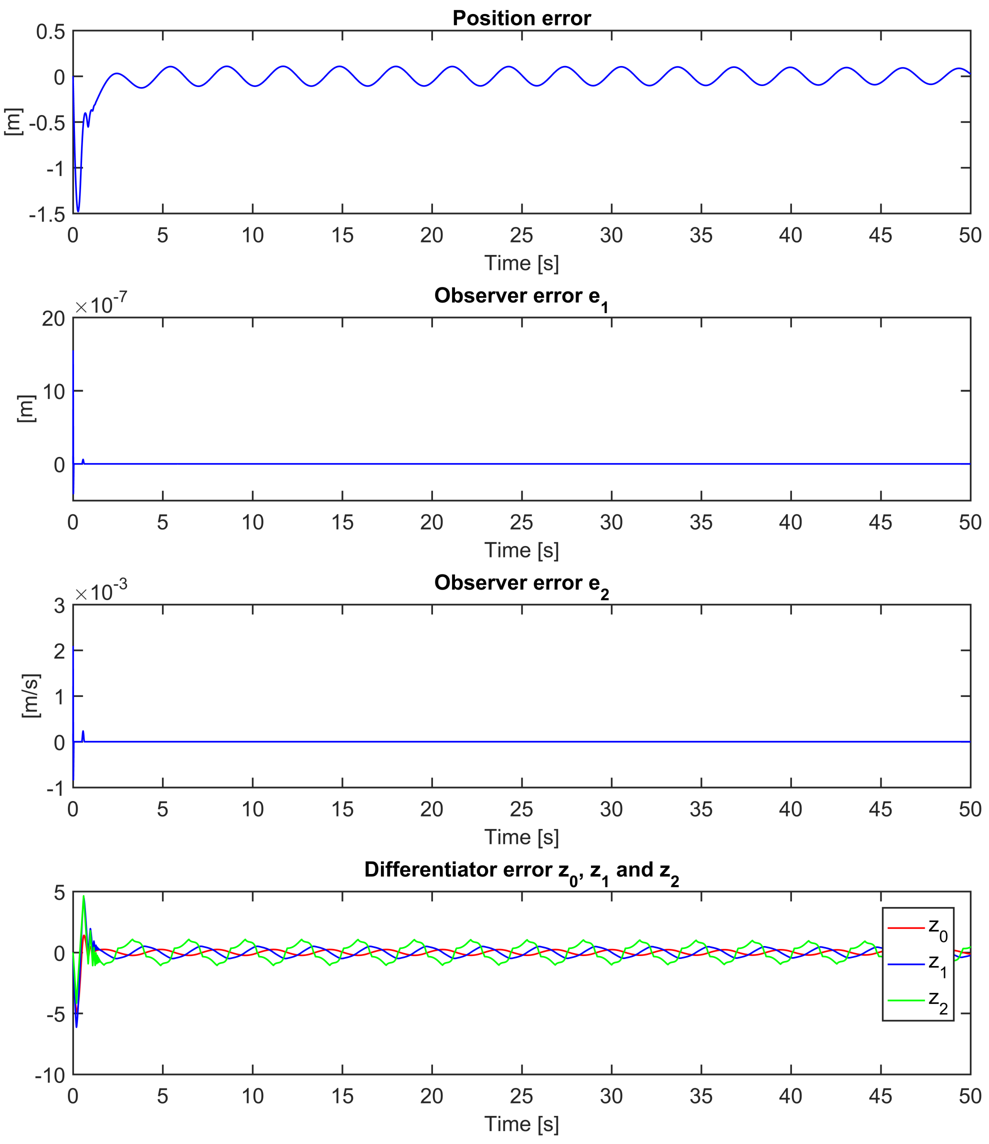


Figure 4.38: MSDS: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 2 and $\alpha = 35.25$ (b)

Sliding surface 3: the estimated value for x_1 and x_2 are used
The gains were set to: $\alpha = 34.2$, $L_{\mathrm{SO}} = 16$ and $L_D = 20$.

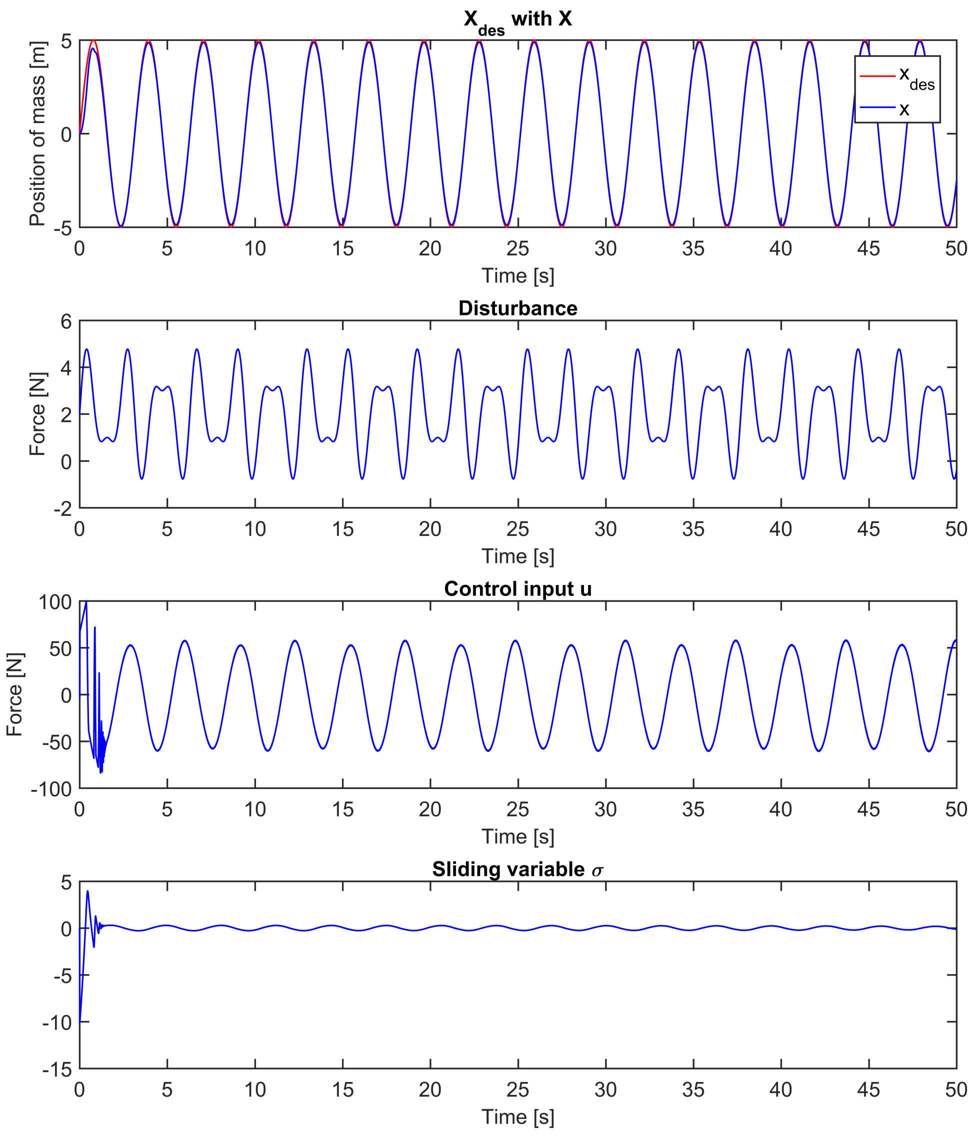


Figure 4.39: MSDS: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 3 and $\alpha = 34.2$ (a)

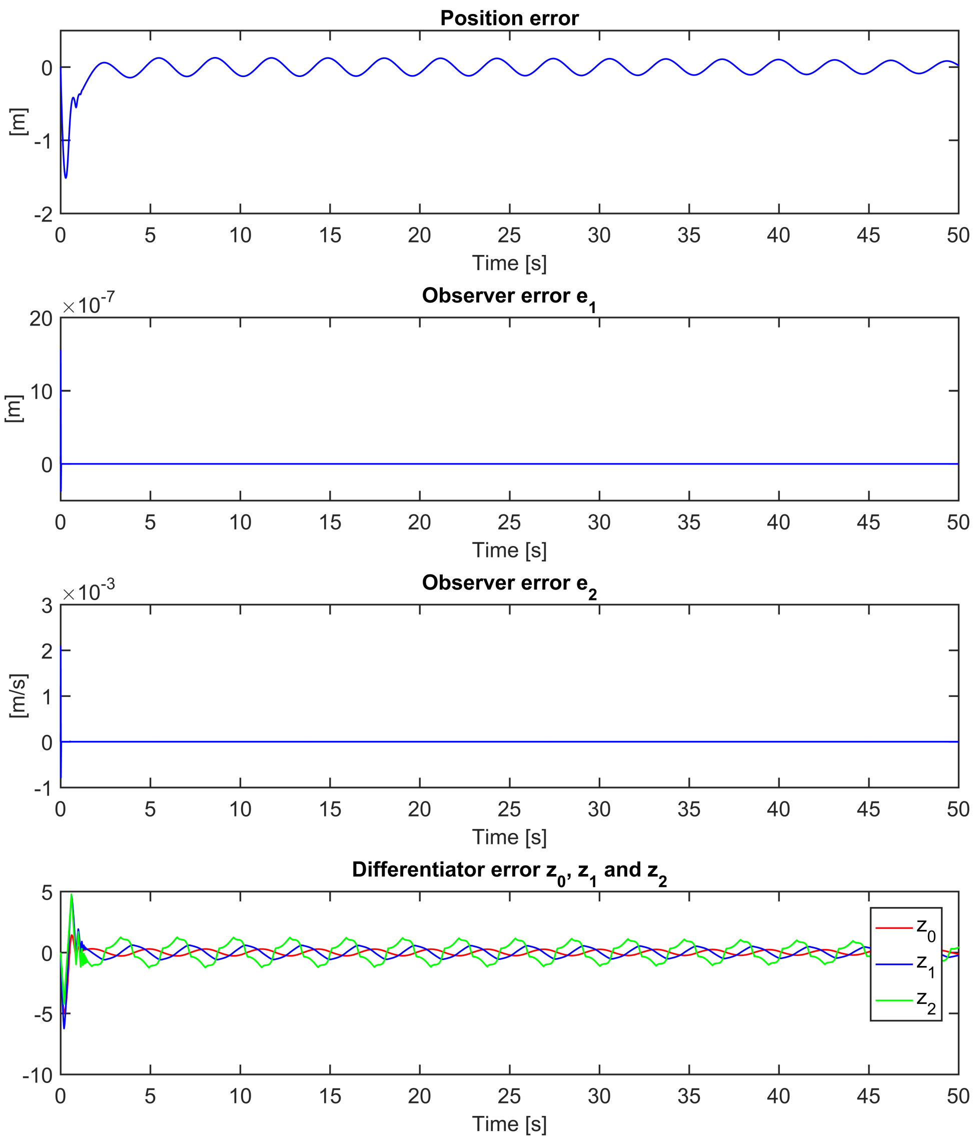


Figure 4.40: MSDS: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 3 and $\alpha = 34.2$ (b)

In order to better compare the different algorithms the absolute maximum value for the error variables for each algorithm have been gather in Table 4.1, 4.2, 4.3, 4.4 and 4.5. Here the first 15 seconds were not considered so that the absolute maximum value for the error variables were found when the control input had stabilized.

Table 4.1: MSDS: absolute maximum value for position error

Algorithm	**Position error**		
	Sliding 1	Sliding 2	Sliding 3
Super-twisting algorithm	$3.3984 \cdot 10^{-5}$	$3.4021 \cdot 10^{-5}$	$3.4021 \cdot 10^{-5}$
Nested 3rd-order SMC	$3.6205 \cdot 10^{-5}$	$3.5976 \cdot 10^{-5}$	$6.2447 \cdot 10^{-4}$
Quasi-continuous 3rd-order SMC, $\alpha = 40/30/30$	$3.2918 \cdot 10^{-5}$	$5.0395 \cdot 10^{-5}$	$3.5110 \cdot 10^{-4}$
Quasi-continuous 3rd-order SMC, $\alpha = 46/35.25/34.2$	0.1673	0.1073	0.1232

Table 4.2: MSDS: absolute maximum value for state observer error

Algorithm	e_1		e_2	
	Sliding 2	Sliding 3	Sliding 2	Sliding 3
Super-twisting algorithm	$1.9451 \cdot 10^{-13}$	$1.9451 \cdot 10^{-13}$	$3.2623 \cdot 10^{-8}$	$3.3151 \cdot 10^{-8}$
Nested 3rd-order SMC	$3.9202 \cdot 10^{-13}$	$1.2077 \cdot 10^{-12}$	$8.1141 \cdot 10^{-8}$	$1.7150 \cdot 10^{-7}$
Quasi-continuous 3rd-order SMC, $\alpha = 30$	$1.9718 \cdot 10^{-13}$	$1.5124 \cdot 10^{-12}$	$3.4181 \cdot 10^{-8}$	$1.9921 \cdot 10^{-7}$
Quasi-continuous 3rd-order SMC, $\alpha = 35.25/34.2$	$1.9362 \cdot 10^{-13}$	$4.2633 \cdot 10^{-13}$	$3.2161 \cdot 10^{-8}$	$7.4957 \cdot 10^{-8}$

Table 4.3: MSDS: absolute maximum value for differentiator error, sliding surface 1

Algorithm	z_0	z_1	z_2
Nested 3r-order SMC	$4.1483 \cdot 10^{-5}$	$7.4184 \cdot 10^{-4}$	0.0144
Quasi-continuous 3rd-order SMC, $\alpha = 40$	$4.0002 \cdot 10^{-6}$	$1.4264 \cdot 10^{-4}$	0.0039
Quasi-continuous 3rd-order SMC, $\alpha = 46$	0.3333	0.6897	1.4675

Table 4.4: MSDS: absolute maximum value for differentiator error, sliding surface 2

Algorithm	z_0	z_1	z_2
Nested 3rd-order SMC	$3.5271 \cdot 10^{-5}$	$6.1956 \cdot 10^{-4}$	0.0141
Quasi-continuous 3rd-order SMC, $\alpha = 30$	$8.1039 \cdot 10^{-7}$	$9.8327 \cdot 10^{-5}$	0.0030
Quasi-continuous 3rd-order SMC, $\alpha = 35.25$	0.2389	0.4962	1.0570

Table 4.5: MSDS: absolute maximum value for differentiator error, sliding surface 3

Algorithm	z_0	z_1	z_2
Nested 3rd-order SMC	0.0020	0.0128	0.0783
Quasi-continuous 3rd-order SMC, $\alpha = 30$	0.0010	0.0030	0.0045
Quasi-continuous 3rd-order SMC, $\alpha = 34.2$	0.2746	0.5746	1.2140

Chapter 5

Underwater Swimming Manipulator

The underwater swimming manipulator (USM), is a snake-like, multi-articulated, underwater robot equipped with thrusters. The USM has a complex control design problem. This is because the USM is subject to hydrodynamic and hydrostatic parameter uncertainties, uncertain thruster characteristics, unknown disturbances, and un-modelled dynamic effects, e.g. thruster dynamics and coupling forces caused by joint motion. Sliding mode techniques have been applied to land-based snake robots in Rezapour et al. (2014) to achieve robust tracking of a desired gait pattern and under-actuated straight line path following. To the author's best knowledge, sliding mode control algorithms have only been tested previously on an USM by the author in Borlaug (2016). The cases that will be tested on this system are the same as for the test system, i.e. MSDS.

In Section 5.3 it will be shown that the error dynamics for the USM system fits the assumptions made for the error dynamics for the general system, which means that the theorems in Chapter 3 also hold for the error dynamics for the USM.

5.1 System

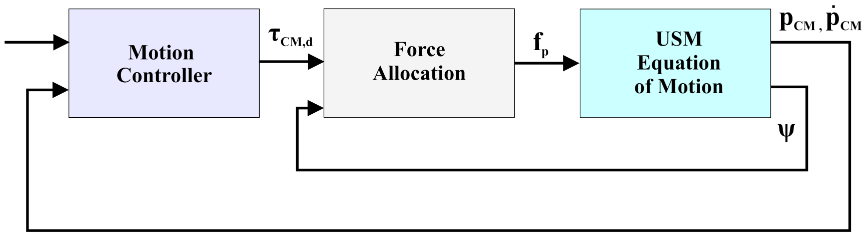
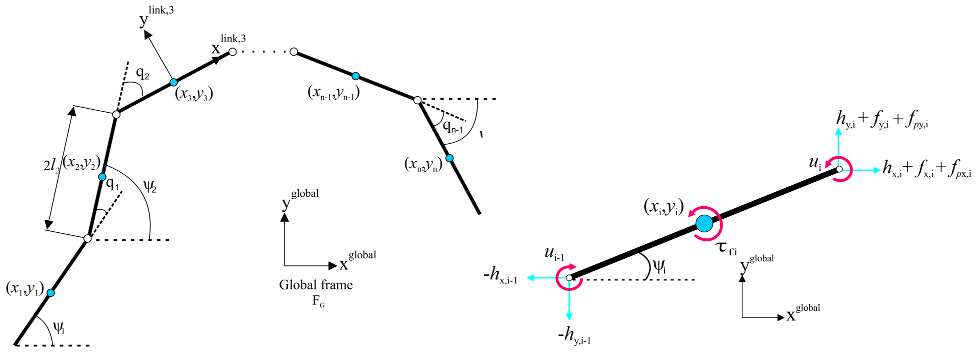


Figure 5.1: System overview USM, Sverdrup-Thygeson et al. (2016)

In this chapter the robot model presented in Sverdrup-Thygeson et al. (2016) will be used. The model in Sverdrup-Thygeson et al. (2016) extends the 2D model proposed in Kelasidi et al. (2014), by modelling also additional effectors and considering the force allocation among these effectors. In this section the equation of motion for the USM and the force allocation matrix will be explained, detailed calculations can be found in Sverdrup-Thygeson et al. (2016). The kinematic equations are developed for 2D based on the method outlined in Liljebäck et al. (2012). The following two Sections 5.1.1 and 5.1.2 are mainly taken from Borlaug (2016).

5.1.1 Equation of motion USM

The USM consists of n rigid links, connected by n-1 motorized joints, equipped with r additional effectors producing forces and moments on the centre of mass (CM) of the USM, that is moving fully submerged in a 2D virtual horizontal plane. The length of each link is defined as $2l_i$, were $i = 1, \ldots, n$ is the link number. The links can have different mass and length depending on the module configuration of the USM. The joint angels are $q = [q_1, \ldots, q_{n-1}]^T \in R^{n-1}$ and the global link angels are $\psi = [\psi_1, \ldots, \psi_n]^T \in R^n$. The kinematics and the forces and torques acting on each link is illustrated in Figure 5.2.



(a) Kinematic parameters **(b)** Forces and torques acting on each link

Figure 5.2: Underwater swimming manipulator, Sverdrup-Thygeson et al. (2016)

Definitions that will be used:

$$A = \begin{bmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix} \in R^{(n-1)\times n},\ D = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in R^{(n-1)\times n},$$

$$e = [1 \cdots 1]^T \ \in R^n,\ E = \begin{bmatrix} e & 0_{n\times 1} \\ 0_{n\times 1} & e \end{bmatrix} \in R^{2n\times 2}$$

$$\sin\psi = [\sin\psi_1 \ \ldots \ \sin\psi_n]^T \ \in R^n,\ S_\psi = diag(\sin\psi) \in R^{n\times n},$$

$$\cos\psi = [\cos\psi_1 \ \ldots \ \cos\psi_n]^T \ \in R^n,\ C_\psi = diag(\cos\psi) \in R^{n\times n},$$

$$\dot{\psi}^2 = [\dot{\psi}_1^2 \ \ldots \ \dot{\psi}_n^2]^T \ \in R^n.$$

$$M = diag([m_1 \ \ldots \ m_n]) \ \in R^{n\times n},\ L = diag([l_1 \ \ldots \ l_n]) \ \in R^{n\times n},$$

$$J = diag([j_i \ \ldots \ j_n]) \ \in R^{n\times n}$$

M is the mass matrix, L is the length matrix and J is the inertia matrix.

The global frame position $p_{\text{CM}} \in R^2$ of the CM of the USM is defined as

$$p_{CM} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{1}{m_t}\sum_{i=1}^n m_i x_i \\ \frac{1}{m_t}\sum_{i=1}^n m_i y_i \end{bmatrix} = \frac{1}{m_t}\begin{bmatrix} e^T M X \\ e^T M Y \end{bmatrix} \tag{5.1}$$

where (x_i, y_i), $i = 1, \ldots, n$ are the coordinates of the CM of link i in global frame, m_i is the mass of link i and $m_t = \sum_{i=1}^n m_i$ is the total mass of the USM. Equation (5.1) is valid because it is assumed that the mass of each link is uniformly distributed. The matrix representation of the force balance for all links with different link mass is expressed by

$$M\ddot{X} = D^T h_x + f_x + f_{px}, \qquad M\ddot{Y} = D^T h_y + f_y + f_{py} \tag{5.2}$$

where f_{px} and f_{py} are the forces from the additional effectors, h_x and h_y are the joint constraint forces and f_x and f_y are the fluid forces on all links. By differentiating Equation (5.1) and inserting Equation (5.2), the joint constraint forces cancel out, and the translational motion of the CM of the USM can be written as

$$m_t\ddot{p}_x = e^T(f_x + f_{px}), \qquad m_t\ddot{p}_y = e^T(f_y + f_{py}). \tag{5.3}$$

The equation of motion can be expressed as

$$\begin{aligned} M_\psi\ddot{\psi} + W_\psi\dot{\psi}^2 + V_\psi\dot{\psi} + \Lambda_3|\dot{\psi}|\dot{\psi} - K_1\mu(S_\psi e\ddot{p}_x - C_\psi e\ddot{p}_y) \\ + S_\psi K(f_{Dx} + f_{px}) - C\psi K(f_{Dy} + f_{py}) = D^T u \end{aligned} \tag{5.4}$$

where

$$
\begin{aligned}
&M_\psi = J + V_1 + K_1 \mu K_1^T + \Lambda_1,\\
&W_\psi = V_2 - K_1 \mu K_2^T\\
&V_\psi = \Lambda_2 - K_1 \mu (C_\psi V_x^a + S_\psi V_y^a)\\
&K_1 = S_\psi K S_\psi + C_\psi K C_\psi,\ K_2 = S_\psi K C_\psi - C_\psi K S_\psi\\
&V_1 = S_\psi V S_\psi + C_\psi V C_\psi,\ V_2 = S_\psi V C_\psi - C_\psi V S_\psi\\
&V = L A^T (D M^{-1} D^T)^{-1} A L\\
&K = L A^T (D M^{-1} D^T)^{-1} D M^{-1} \ \in R^{n\times n}\\
&\Lambda_1 = diag(\lambda_{1,1}, \dots, \lambda_{1,n}) \ \in R^{n\times n}\\
&\Lambda_2 = diag(\lambda_{2,1}, \dots, \lambda_{2,n}) \ \in R^{n\times n}\\
&\Lambda_3 = diag(\lambda_{3,1}, \dots, \lambda_{3,n}) \ \in R^{n\times n}\\
&\mu = diag(\mu_1, \dots, \mu_n) \ \in R^{n\times n}
\end{aligned}
$$

here V_x^a and V_y^a are the ocean current velocity expressed in inertial frame coordinates, f_{Dx} and f_{Dy} are the drag forces (linear and non-linear) on the USM, and $\ddot{p}_x$ and $\ddot{p}_y$ can be found by rearranging Equation (5.3). The coefficients $\lambda_{2,i}$, $\lambda_{3,i}$ represent the drag parameters due to the pressure difference between the two sides of the body, and the parameters μ_i and $\lambda_{1,i}$ represent the added mass of the fluid carried by the moving body.

5.1.2 Force allocation

The force allocation distribution is given by

$$
\tau_{\mathrm{CM}} = \begin{bmatrix} F_{\mathrm{CM},x} \\ F_{\mathrm{CM},y} \\ M_{\mathrm{CM},z} \end{bmatrix} = \begin{bmatrix} e^T & 0^{1\times n} \\ 0^{1\times n} & e^T \\ e^T S_\psi K & -e^T C_\psi K \end{bmatrix} \begin{bmatrix} f_{px} \\ f_{py} \end{bmatrix} = T(\psi) f_p, \tag{5.5}
$$

where $T(\psi)$ is the allocation matrix and $f_p = [f_{p,k_1}, \dots, f_{p,k_r}]$ is the vector of scalar effector forces. It is the mapping between the effector forces and the forces and moments acting on the CM of the USM. According to Sverdrup-Thygeson et al. (2016) the force allocation matrix for 2D application can be expressed as

$$
T(\psi) = \begin{bmatrix} b_x^T \\ b_y^T \\ e^T S_\psi K B_X^T - e^T C_\psi K B_Y^T \end{bmatrix} \tag{5.6}
$$

where b_x, b_y, B_X and B_Y are configuration vectors. It is assumed that the additional effector forces are acting through the CM of each link. The primary objective for the force allocation method is to distribute the efforts among the additional effectors to obtain the forces and moments required to maintain the desired heading and follow the path with non-zero forward velocity.

5.2 Control design

Control problem: *Assume that there exist a guidance system which determines a suitable path for the USM to follow. The task at hand is to design a motion controller that calculates the desired forces for the translational motion F_{CM}, and the desired moments for the rotational motion M_{CM}, of the USM.*

A sliding mode control algorithm will be used to calculate the desired forces, F_{CM}. To calculate the desired moments, M_{CM}, a proportional controller will be used. The desired forces and moments are represented by

$$\tau_{\text{CM},d} = \begin{bmatrix} F_{\text{CM},d} \\ M_{\text{CM},d} \end{bmatrix} = \begin{bmatrix} F_{\text{CM},d_x} \\ F_{\text{CM},d_y} \\ M_{\text{CM},d} \end{bmatrix} \tag{5.7}$$

Since it is the tracking problem for the position of the centre of mass of the USM that will be considered the equations of motion that are relevant for the design are the translational motion of the CM, Equation (5.3). By dividing the equations by m_t (the total mass of the USM), and using the fact that $F_{\text{CM},x} = e^T f_{px}$ and $F_{\text{CM},y} = e^T f_{py}$ from Equation (5.5), the equations of motion can be written as

$$\ddot{p}_x = \frac{1}{m_t}(e^T f_x + F_{\text{CM},x}) = \frac{1}{m_t} e^T f_x + \frac{1}{m_t} F_{\text{CM},x} \tag{5.8}$$

$$\ddot{p}_y = \frac{1}{m_t}(e^T f_y + F_{\text{CM},y}) = \frac{1}{m_t} e^T f_y + \frac{1}{m_t} F_{\text{CM},y} \tag{5.9}$$

where $e^T f_x$ is the sum of all forces acting on the CM in x-direction and $e^T f_y$ is the sum of all forces acting on the CM in y-direction. These forces are hard to model exactly and will therefore be interpreted as a time-varying bounded disturbance called $f(t)$. The equations can then be written as

$$\ddot{p}_x = \frac{1}{m_t} f_x(t) + \frac{1}{m_t} F_{\text{CM},x} \tag{5.10}$$

$$\ddot{p}_y = \frac{1}{m_t} f_y(t) + \frac{1}{m_t} F_{\text{CM},y} \tag{5.11}$$

5.2.1 Sliding surface design

First the error variable is defined. As the output variable for the translational motion of the USM is p_{CM}, the error variable can be defined as

$$\tilde{p} = \begin{bmatrix} \tilde{p}_x \\ \tilde{p}_y \end{bmatrix} = p_{\text{CM}} - p_{\text{CM,des}} = \begin{bmatrix} p_x - p_{x,\text{des}} \\ p_y - p_{y,\text{des}} \end{bmatrix} \tag{5.12}$$

where $p_{\text{CM,des}}$ is the desired position of the global frame position of the CM of the USM. To recap the sliding surface should be selected such that the state trajectories of the controlled system is forced onto the sliding surface $\sigma = \dot{\sigma} = 0$, where the system behaviour meets the design specifications. The controller u should also appear in the first derivative of σ, so that the relative degree is equal to 1. The sliding surface σ can then be chosen as

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_y \end{bmatrix} = \tilde{p} + \dot{\tilde{p}} = \begin{bmatrix} \tilde{p}_x \\ \tilde{p}_y \end{bmatrix} + \begin{bmatrix} \dot{\tilde{p}}_x \\ \dot{\tilde{p}}_y \end{bmatrix} = \begin{bmatrix} p_x - p_{x,\mathrm{des}} \\ p_y - p_{y,\mathrm{des}} \end{bmatrix} + \begin{bmatrix} \dot{p}_x - \dot{p}_{x,\mathrm{des}} \\ \dot{p}_y - \dot{p}_{y,\mathrm{des}} \end{bmatrix} \tag{5.13}$$

It is assumed that only the position, p_{CM}, of the centre of mass is available for measurement, an observer for the states are therefore designed, Section 5.2.2. The observer states will be used in the sliding surface, and following the structure of Equation (5.13), the revised sliding surface is then

$$\hat{\sigma} = \begin{bmatrix} \hat{\sigma}_x \\ \hat{\sigma}_y \end{bmatrix} = \tilde{p} + \dot{\hat{\tilde{p}}} = \begin{bmatrix} \tilde{p}_x \\ \tilde{p}_y \end{bmatrix} + \begin{bmatrix} \dot{\hat{\tilde{p}}}_x \\ \dot{\hat{\tilde{p}}}_y \end{bmatrix} = \begin{bmatrix} p_x - p_{x,\mathrm{des}} \\ p_y - p_{y,\mathrm{des}} \end{bmatrix} + \begin{bmatrix} \dot{\hat{p}}_x - \dot{p}_{x,\mathrm{des}} \\ \dot{\hat{p}}_y - \dot{p}_{y,\mathrm{des}} \end{bmatrix}. \tag{5.14}$$

when the estimated value of $\dot{p} = \begin{bmatrix} \dot{p}_x & \dot{p}_y \end{bmatrix}^T$ is used in the sliding surface, and

$$\hat{\sigma} = \begin{bmatrix} \hat{\sigma}_x \\ \hat{\sigma}_y \end{bmatrix} = \hat{\tilde{p}} + \dot{\hat{\tilde{p}}} = \begin{bmatrix} \hat{\tilde{p}}_x \\ \hat{\tilde{p}}_y \end{bmatrix} + \begin{bmatrix} \dot{\hat{\tilde{p}}}_x \\ \dot{\hat{\tilde{p}}}_y \end{bmatrix} = \begin{bmatrix} \hat{p}_x - p_{x,\mathrm{des}} \\ \hat{p}_y - p_{y,\mathrm{des}} \end{bmatrix} + \begin{bmatrix} \dot{\hat{p}}_x - \dot{p}_{x,\mathrm{des}} \\ \dot{\hat{p}}_y - \dot{p}_{y,\mathrm{des}} \end{bmatrix}. \tag{5.15}$$

when both the estimated value of p_{CM} and $\dot{p}$ are used in the sliding surface. For the remaining sections of this chapter the sliding surface in Equation (5.13) will be called sliding surface 1, the sliding surface in Equation (5.14) will be called sliding surface 2 and the sliding surface in Equation (5.15) will be called sliding surface 3.

5.2.2 Observer and differentiator design

State observer

By designing the observer structure as in Section 2.5.2, the state observer is

$$\begin{aligned}
\dot{\hat{p}}_1 &= \begin{bmatrix} \dot{\hat{p}}_{1,x} \\ \dot{\hat{p}}_{1,y} \end{bmatrix} = \begin{bmatrix} \hat{p}_{2,x} + z_{1,x} \\ \hat{p}_{2,y} + z_{1,y} \end{bmatrix} = \begin{bmatrix} \hat{p}_{2,x} + k_1 \|e_{1,x}\|^{2/3} \operatorname{sgn}(e_{1,x}) \\ \hat{p}_{2,y} + k_1 \|e_{1,y}\|^{2/3} \operatorname{sgn}(e_{1,y}) \end{bmatrix} \\
\dot{\hat{p}}_2 &= \begin{bmatrix} \dot{\hat{p}}_{2,x} \\ \dot{\hat{p}}_{2,y} \end{bmatrix} = \begin{bmatrix} \hat{p}_{3,x} + z_{2,x} + \frac{1}{m_t} u_x \\ \hat{p}_{3,y} + z_{2,y} + \frac{1}{m_t} u_y \end{bmatrix} \\
&= \begin{bmatrix} \hat{p}_{3,x} + k_2 \|e_{1,x}\|^{1/3} \operatorname{sgn}(e_{1,x}) + \frac{1}{m_t} u_x \\ \hat{p}_{3,y} + k_2 \|e_{1,y}\|^{1/3} \operatorname{sgn}(e_{1,y}) + \frac{1}{m_t} u_y \end{bmatrix} \\
\dot{\hat{p}}_3 &= \begin{bmatrix} \dot{\hat{p}}_{3,x} \\ \dot{\hat{p}}_{3,y} \end{bmatrix} = \begin{bmatrix} z_{3,x} \\ z_{3,y} \end{bmatrix} = \begin{bmatrix} k_3 \operatorname{sgn}(e_{1,x}) \\ k_3 \operatorname{sgn}(e_{1,y}) \end{bmatrix}
\end{aligned} \tag{5.16}$$

where k_1, k_2 and k_3 are gains to be chosen according to Levant (1998) and Levant (2003a), $e_{1,x} = p_x - \hat{p}_{1,x}$ and $e_{1,y} = p_y - \hat{p}_{1,y}$. $\hat{p}_1$ is the estimated value for p_{CM}, and $\hat{p}_2$ is the estimated value for $\dot{p}$. One choice of parameters that meets the requirements in Levant (1998) and Levant (2003a), is according to Chalanga et al. (2016), $k_1 = 6L_{\mathrm{SO}}^{1/3}$, $k_2 = 11L_{\mathrm{SO}}^{1/2}$ and $k_3 = 6L_{\mathrm{SO}}$, where L_{SO} is a sufficiently large constant.

Differentiator

To be able to use the HOSM algorithms proposed in Section 2.4 a differentiator has to be design. By using the recursive form of the arbitrary order robust exact differentiator with $r = 3$, described in Equation (2.10), a third order differentiator for σ can be designed as

$$
\begin{aligned}
\dot{z}_0 &= \begin{bmatrix} z_{0,x} \\ z_{0,y} \end{bmatrix} = \begin{bmatrix} v_{0,x} \\ v_{0,y} \end{bmatrix}, \quad \begin{bmatrix} v_{0,x} \\ v_{0,y} \end{bmatrix} = \begin{bmatrix} -\lambda_2 L_D^{1/3} |z_{0,x} - \sigma_x|^{2/3} \operatorname{sgn}(z_{0,x} - \sigma_x) + z_{1,x} \\ -\lambda_2 L_D^{1/3} |z_{0,y} - \sigma_y|^{2/3} \operatorname{sgn}(z_{0,y} - \sigma_y) + z_{1,y} \end{bmatrix} \\
\dot{z}_1 &= \begin{bmatrix} z_{1,x} \\ z_{1,y} \end{bmatrix} = \begin{bmatrix} v_{1,x} \\ v_{1,y} \end{bmatrix}, \quad \begin{bmatrix} v_{1,x} \\ v_{1,y} \end{bmatrix} = \begin{bmatrix} -\lambda_1 L_D^{1/2} |z_{1,x} - v_{0,x}|^{1/2} \operatorname{sgn}(z_{1,x} - v_{0,x}) + z_{2,x} \\ -\lambda_1 L_D^{1/2} |z_{1,y} - v_{0,y}|^{1/2} \operatorname{sgn}(z_{1,y} - v_{0,y}) + z_{2,y} \end{bmatrix} \\
\dot{z}_2 &= \begin{bmatrix} z_{2,x} \\ z_{2,y} \end{bmatrix} = \begin{bmatrix} v_{2,x} \\ v_{2,y} \end{bmatrix}, \quad \begin{bmatrix} v_{2,x} \\ v_{2,y} \end{bmatrix} = \begin{bmatrix} -\lambda_0 L_D \operatorname{sgn}(z_{2,x} - v_{1,x}) \\ -\lambda_0 L_D \operatorname{sgn}(z_{2,y} - v_{1,y}) \end{bmatrix}
\end{aligned}
\tag{5.17}
$$

where the gains λ_0, λ_1 and λ_2 are to be chosen according to Levant (1998) and Levant (2003a). One choice of parameters that meets the requirements are $\lambda_0 = 1.1$, $\lambda_1 = 1.5$ and $\lambda_2 = 3$. Here $z_0 = \sigma$, $z_1 = \dot{\sigma}$ and $z_2 = \ddot{\sigma}$ in the HOSM algorithms. L_D needs to be sufficiently large.

5.2.3 Control algorithms

The super-twisting algorithm

By choosing the gains in Equation (2.3) to be $k_1 = 1.5\sqrt{K}$ and $k_2 = 1.1K$, where K is a sufficiently large positive constant. The control structure can be written as

$$
\begin{aligned}
u_{\text{STA}} &= \begin{bmatrix} u_x \\ u_y \end{bmatrix} = -1.5\sqrt{K} \begin{bmatrix} |\sigma_x|^{1/2} \operatorname{sgn}(\sigma_x) + v_x \\ |\sigma_y|^{1/2} \operatorname{sgn}(\sigma_y) + v_y \end{bmatrix} \\
\dot{v} &= \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = -1.1K \begin{bmatrix} \operatorname{sgn}(\sigma_x) \\ \operatorname{sgn}(\sigma_y) \end{bmatrix}
\end{aligned}
\tag{5.18}
$$

Higher-order sliding mode controller (third-order):

Nested third-order sliding mode: By choosing $r = 3$ in Equation (2.8), the nested third-order sliding mode controller can be written as

$$
u_N = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = -\alpha \begin{bmatrix} \operatorname{sgn}(\ddot{\sigma}_x + \beta_2(|\dot{\sigma}_x|^3 + |\sigma_x|^2)^{1/6} \times \operatorname{sgn}(\dot{\sigma}_x + \beta_1 |\sigma_x|^{2/3} \operatorname{sgn}(\sigma_x))) \\ \operatorname{sgn}(\ddot{\sigma}_y + \beta_2(|\dot{\sigma}_y|^3 + |\sigma_y|^2)^{1/6} \times \operatorname{sgn}(\dot{\sigma}_y + \beta_1 |\sigma_y|^{2/3} \operatorname{sgn}(\sigma_y))) \end{bmatrix}
\tag{5.19}
$$

where $\beta_2 = 2$, $\beta_1 = 1$ and α is a sufficiently large constant.

Quasi-continuous third-order sliding mode: By choosing $r = 3$ in Equation (2.9), the quasi-continuous third-order sliding mode controller is

$$u_Q = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = -\alpha \begin{bmatrix} \dfrac{\ddot{\sigma}_x + \beta_2(|\dot{\sigma}_x| + |\sigma_x|^{2/3})^{-1/2}(\dot{\sigma}_x + \beta_1|\sigma_x|^{2/3}\operatorname{sgn}(\sigma_x))}{|\ddot{\sigma}_x| + \beta_2(|\dot{\sigma}_x| + |\sigma_x|^{2/3})^{1/2}} \\ \dfrac{\ddot{\sigma}_y + \beta_2(|\dot{\sigma}_y| + |\sigma_y|^{2/3})^{-1/2}(\dot{\sigma}_y + \beta_1|\sigma_y|^{2/3}\operatorname{sgn}(\sigma_y))}{|\ddot{\sigma}_y| + \beta_2(|\dot{\sigma}_y| + |\sigma_y|^{2/3})^{1/2}} \end{bmatrix} \tag{5.20}$$

where $\beta_2 = 2$, $\beta_1 = 1$ and α is a sufficiently large constant.

PD-controller

In order to compare the performance of the SMC algorithms to a linear controller with respect to disturbances and modelling errors, the standard PD-controller that was implemented Sverdrup-Thygeson et al. (2016) was used.

$$u_{\text{PD}} = k_d^{\text{CM}} \begin{bmatrix} \dot{p}_{x,\text{des}} - \dot{p}_x \\ \dot{p}_{y,\text{des}} - \dot{p}_y \end{bmatrix} + k_p^{\text{CM}} \begin{bmatrix} p_{x,\text{des}} - p_x \\ p_{y,\text{des}} - p_y \end{bmatrix} \tag{5.21}$$

where k_d^{CM} and k_p^{CM} are controller gains. The estimated value for p_{CM} and $\dot{p}$ will also be used here, so that the simulated cases are as similar as possible.

5.2.4 Control input

The control input can be written, by using Equation (5.5) and Equation (5.7), as

$$u = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = F_{CM} = \begin{bmatrix} F_{\text{CM},x} \\ F_{\text{CM},y} \end{bmatrix} \tag{5.22}$$

In Chapter 3 the control input was found for the different sliding surfaces and control algorithms, they can be generalized so that by changing u_c in the following equations to u_{STA}, u_N, u_Q or u_{PD} it is valid for all the control algorithms. In the case where the state observer is not used, the sliding surface will be sliding surface 1, and the control input is $u = \frac{1}{g(t,x)} u_c$, Borlaug (2016).

Sliding surface 2: the estimated value for $\dot{p}$ is used in the sliding surface

$$u = \frac{1}{g(t,x)}(-\hat{p}_2 - \hat{p}_3 - z_2 + u_c) \tag{5.23}$$

Sliding surface 3: the estimated value for p_{CM} and $\dot{p}$ are used in the sliding surface

$$u = \frac{1}{g(t,x)}(-\hat{p}_2 - z_1 - \hat{p}_3 - z_2 + u_c) \tag{5.24}$$

5.3 Stability analysis

Stability for the error dynamics for a second-order general system was proven in Chapter 3. In this section it will be shown that the error dynamics for the USM fits the assumptions made for the error dynamics for the general system, which means that the theorems in Chapter 3 also hold for the error dynamics for the USM. The assumptions made for the error dynamics for the second-order general system were

Assumption 1 $|\dot{f}(t,x)| < \Delta$

Assumption 2 $g(t,x)$ known

Assumption 3 $c_1 > 0$

The controller gains also have to be properly chosen according to Section 2.1, 2.5 and 5.2. The sliding surface for this system is $\sigma = \tilde{p} + \dot{\tilde{p}}$, which means that $c_1 = 1$. This is also the case for the sliding surfaces where the estimated states are used. Therefore, assumption 3 hold. To check if assumption 1 and 2 hold the error dynamics in both x and y direction have to be found and analysed.

5.3.1 Stability analysis in x-direction

Error dynamics

The translational motion in x-direction is described by

$$\ddot{p}_x = \frac{1}{m_t} f_x(t) + \frac{1}{m_t} F_{CM,x} \tag{5.25}$$

By setting $p_{x_1} = p_x$ and $p_{x_2} = \dot{p}_x$, it can be written as:

$$\begin{aligned} \dot{p}_{x_1} &= p_{x_2} \\ \dot{p}_{x_2} &= \frac{1}{m_t} f_x(t) + \frac{1}{m_t} F_{CM,x} \end{aligned} \tag{5.26}$$

The error variables are: $\tilde{p}_{x_1} = \tilde{p}_x = p_x - p_{x,\text{des}}$ and $\tilde{p}_{x_2} = \dot{\tilde{p}}_x = \dot{p}_x - \dot{p}_{x,\text{des}}$. By differentiating the error variables, the error dynamics can be described as

$$\begin{aligned} \dot{\tilde{p}}_{x_1} &= \dot{\tilde{p}}_x = \dot{p}_x - \dot{p}_{x,\text{des}}(t) = \tilde{p}_{x_2} \\ \dot{\tilde{p}}_{x_2} &= \ddot{p}_x - \ddot{p}_{x,\text{des}}(t) = \frac{1}{m_t} f_x(t) + \frac{1}{m_t} F_{CM,x} - \ddot{p}_{x,\text{des}}(t) \end{aligned} \tag{5.27}$$

Since the reference trajectory will be bounded by design and since $f_x(t)$ is assumed bounded, a new bounded function called $F_x(t)$ can be introduced, $F_x(t) = (1/m_t) f_x(t) - \ddot{p}_{x,\text{des}}(t)$. $F_x(t)$ is a function of two bounded signals, and therefore will be bounded, which means that the derivative will be bounded and assumption 1 is satisfied. Since $g(t,x)$ is $1/m_t$ and m_t is known assumption 2 is also satisfied. The error dynamics in x-direction for the USM therefore satisfy all of the assumptions made by the error dynamics for the general system, and the theorems in Chapter 3 also holds for the error dynamics in x-direction for the USM.

5.3.2 Stability analysis in y-direction

Error dynamics

The translational motion in y-direction is described by

$$\ddot{p}_y = \frac{1}{m_t} f_y(t) + \frac{1}{m_t} F_{CM,y} \tag{5.28}$$

By setting $p_{y_1} = p_y$ and $p_{y_2} = \dot{p}_y$, it can be written as:

$$\begin{aligned} \dot{p}_{y_1} &= p_{y_2} \\ \dot{p}_{y_2} &= \frac{1}{m_t} f_y(t) + \frac{1}{m_t} F_{CM,y} \end{aligned} \tag{5.29}$$

The error variables are: $\tilde{p}_{y_1} = \tilde{p}_y = p_y - p_{y,\text{des}}$ and $\tilde{p}_{y_2} = \dot{\tilde{p}}_y = \dot{p}_y - \dot{p}_{y,\text{des}}$. By differentiating the error variables, the error dynamics can be described as

$$\begin{aligned} \dot{\tilde{p}}_{y_1} &= \dot{\tilde{p}}_y = \dot{p}_y - \dot{p}_{y,\text{des}}(t) = \tilde{p}_{y_2} \\ \dot{\tilde{p}}_{y_2} &= \ddot{p}_y - \ddot{p}_{y,\text{des}}(t) = \frac{1}{m_t} f_y(t) + \frac{1}{m_t} F_{CM,y} - \ddot{p}_{y,\text{des}}(t) \end{aligned} \tag{5.30}$$

Since the reference trajectory will be bounded by design and since $f_y(t)$ is assumed bounded, a new bounded function called $F_y(t)$ can be introduced, $F_y(t) = (1/m_t) f_y(t) - \ddot{p}_{y,\text{des}}(t)$. $F_y(t)$ is a function of two bounded signals, and therefore will be bounded, which means that the derivative will be bounded and assumption 1 is satisfied. Since $g(t,x)$ is $1/m_t$ and m_t is known assumption 2 is also satisfied. The error dynamics in y-direction for the USM therefore satisfy all of the assumptions made by the error dynamics for the general system, and the theorems in Chapter 3 also holds for the error dynamics in y-direction for the USM.

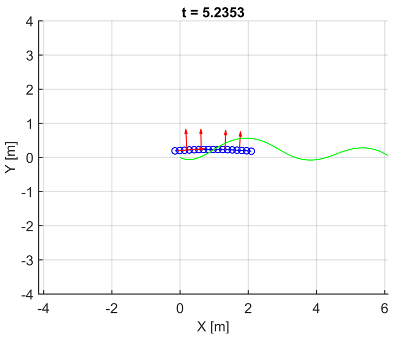
5.4 Implementation

5.4.1 System

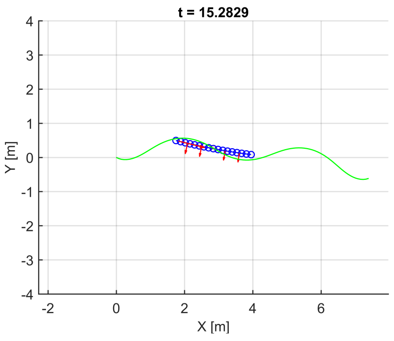
The complete model with force allocation matrix was implemented in MATLAB. The USM implemented has $n = 16$ links, each one having length $2l_i = 0.14$ m and mass $m_i = 0.6597$ kg. The thruster configuration used corresponds to configuration 2 in Sverdrup-Thygeson et al. (2016), this has one tail effector attached at link 1 exerting force along the link x-axis and four additional effectors located at link number 3, 6, 11 and 14 exerting forces normal to the links. For more details regarding the parameters used in the model, please see Sverdrup-Thygeson et al. (2016). There has been implemented two different case studies, one called torpedo mode, which is described in Section 5.4.1, and one called operation mode, described in Section 5.4.1. See Appendix D.2 for the code.

Torpedo mode

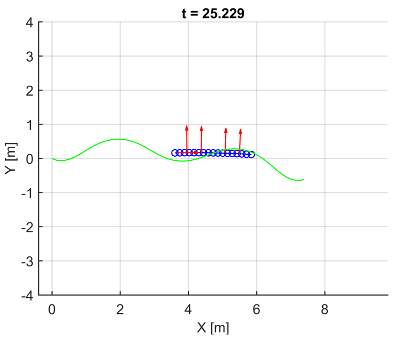
The USM will be moving as a torpedo-shaped AUV when it is moving from one place to another. To simulate this type of behaviour, the link angles were set to zero, i.e. there was no lateral undulation, and a line-of-sight (LOS) guidance law was used for heading control. For information on how the LOS guidance law was incorporated into the system and the motivation behind this choice, see Sverdrup-Thygeson et al. (2016). This was implemented by choosing no lateral undulation in file $calculate\_u\_lateral\_undulation.m$, i.e. $alpha = 0$. This simulation case is shown in Figure 5.3.



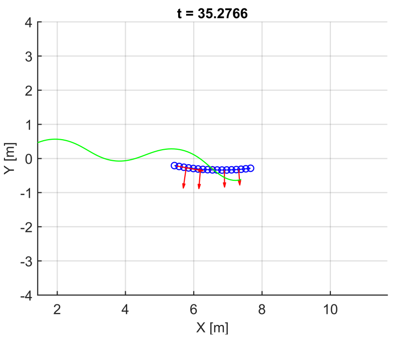
(a) Torpedo: t = 5



(b) Torpedo: t = 15



(c) Torpedo: t = 25

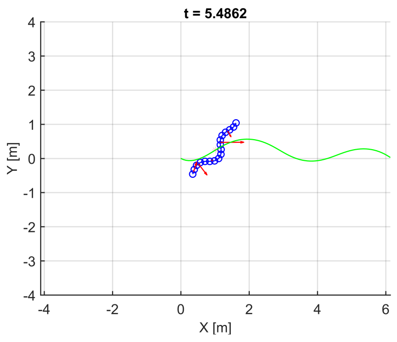


(d) Torpedo: t = 35

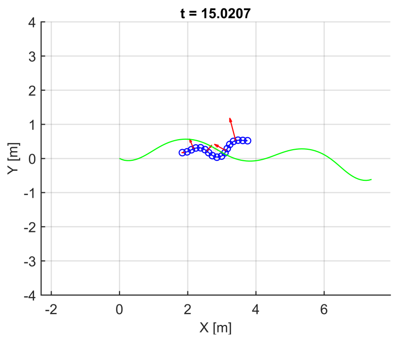
Figure 5.3: Torpedo mode simulation

Operation mode

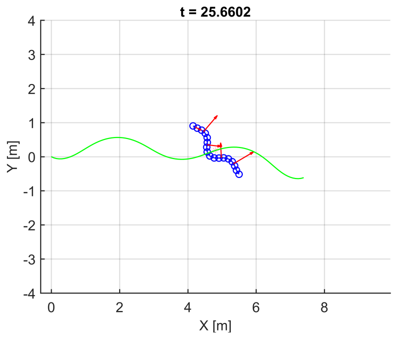
When the USM is in operation mode, it will use the thrusters to stay in one place or move around, and use the end-effectors at the head of the USM to do the operation. The motion of the joints can be seen as a disturbance to the CM position control system, as it will inflict unwanted motion on the CM of the USM. This simulation case investigates how well the proposed control laws attenuates the unwanted effects of the joint motion. The simulated operation is an inspection, which entails that the head of the USM first moves in one direction and then the other, while the thrusters should keep the USM on the reference path. The USM head changes direction at 10, 20 and 30 seconds. This was implemented in $calculate\_u\_lateral\_undulation.m$ by setting $heading\_ref = \sin(\pi/2)$ for the first 10 seconds, $heading\_ref = \sin(\pi)$ from 10-20 seconds, $heading\_ref = \sin(3\pi/2)$ from 20-30 seconds, $heading\_ref = \sin(\pi)$ from 30-40 seconds and removing the no undulation restriction. The same changes were also made in $calculate\_desired\_forces\_moments.m$ for the desired heading variable $heading\_d$. This type of simulation is shown in Figure 5.4.



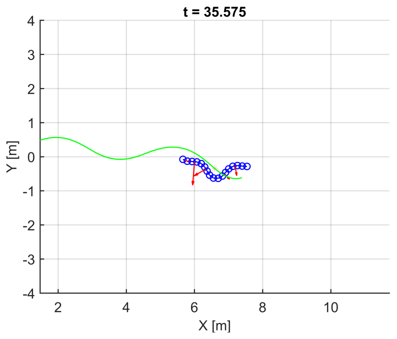
(a) Operation: t = 5



(b) Operation: t = 15



(c) Operation: t = 25



(d) Operation: t = 35

Figure 5.4: Operation mode simulation

5.4.2 Sliding surface and control input

The sliding surfaces were made both by using the error variables available from the PD-controller that was previously implemented in $calculate\_desired\_forces\_moments.m$ and the state observer variables. The controllers, observers and differentiators were implemented in the same file. The code for the implementation is given in Appendix D.2. The PD-controller that was previously implemented, was kept for comparing purposes.

5.5 Results

The sufficiently large K, L_D, L_{SO} and α that were mentioned in the previous section were chosen with the trial and error method. That is because it is difficult finding a bound on the disturbance. The gains were found by starting with a very high or low gain, and then increasing or decreasing it until the error started to increase. The gain with the lowest error was then chosen. For the simulations an ODE5 solver, with fixed step size 10^{-5} was used, because when using a variable step solver the step size got to small in the beginning of the simulation, which then never really progressed.

5.5.1 Torpedo mode

The super-twisting algorithm

The gains were set to: $K = 10$ and $L_{\mathrm{SO}} = 50$.
Sliding surface 1: the actual value of p_{CM} and $\dot{p}$ are used

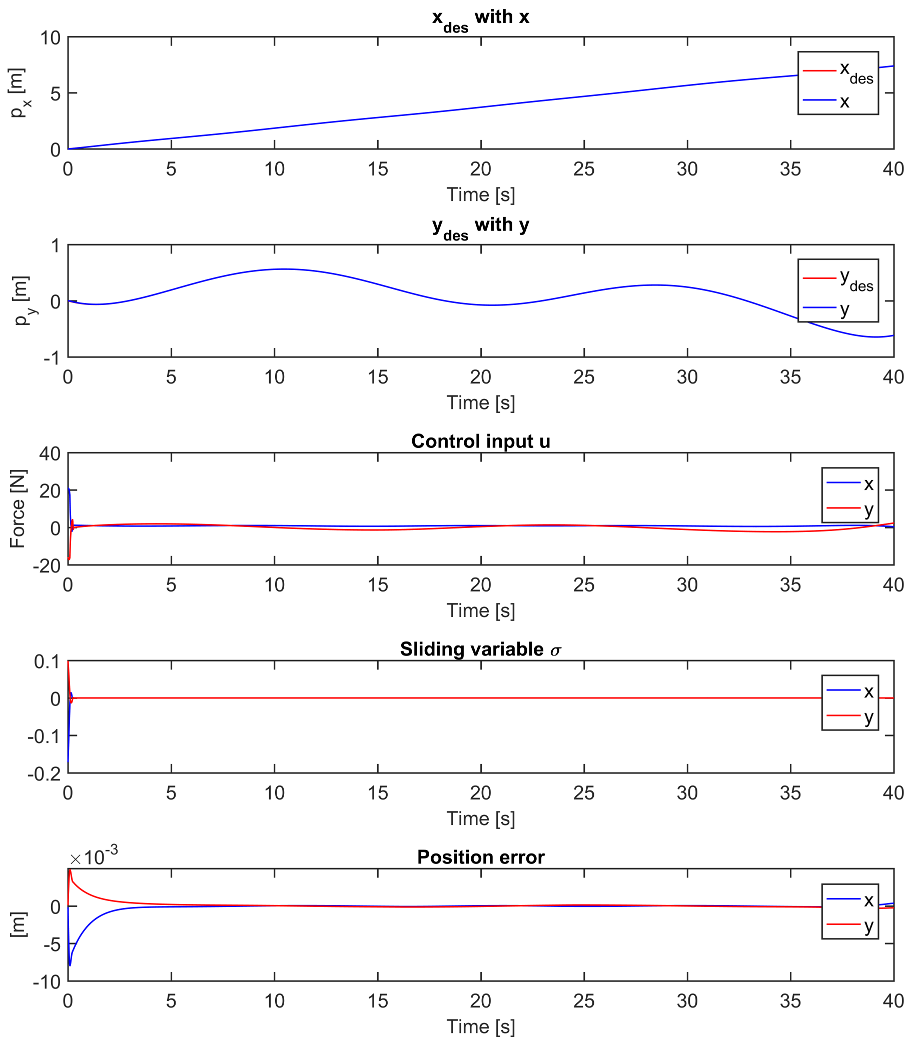


Figure 5.5: USM torpedo mode: Simulation of STA, sliding surface 1

Sliding surface 2: the estimated value for $\dot{p}$ is used

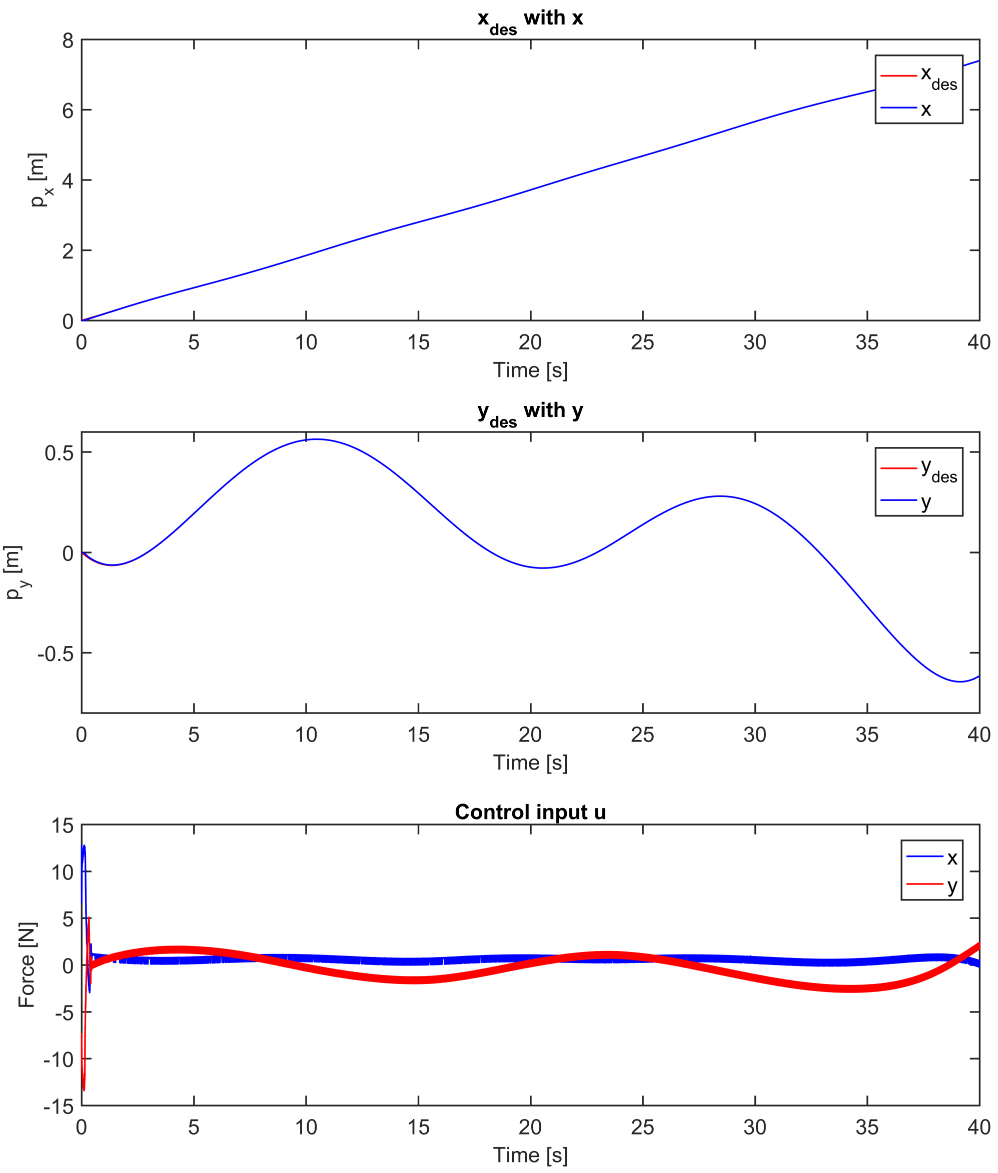


Figure 5.6: USM torpedo mode: Simulation of STA with state observer, sliding surface 2 (a)

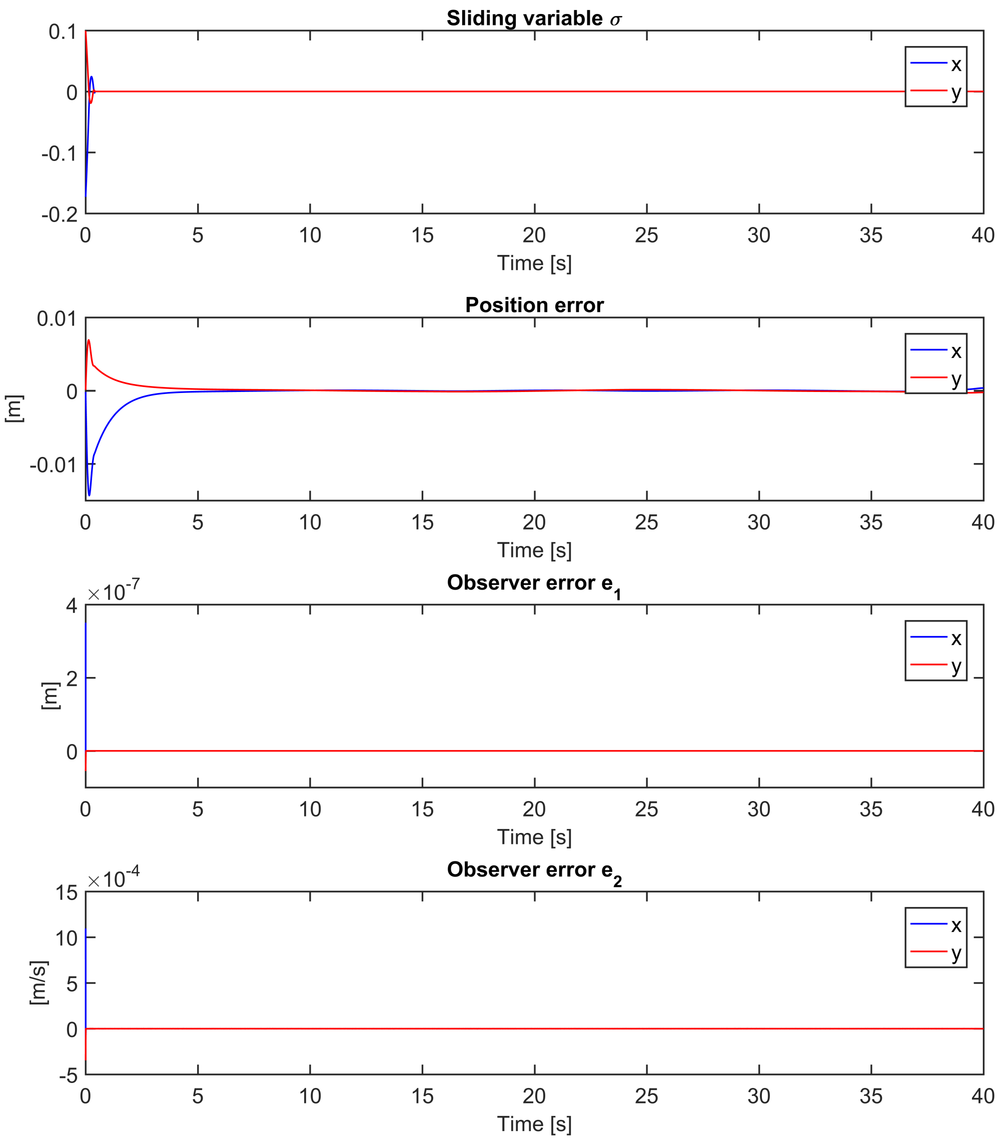


Figure 5.7: USM torpedo mode: Simulation of STA with state observer, sliding surface 2 (b)

Sliding surface 3: the estimated value for p_{CM} and $\dot{p}$ are used

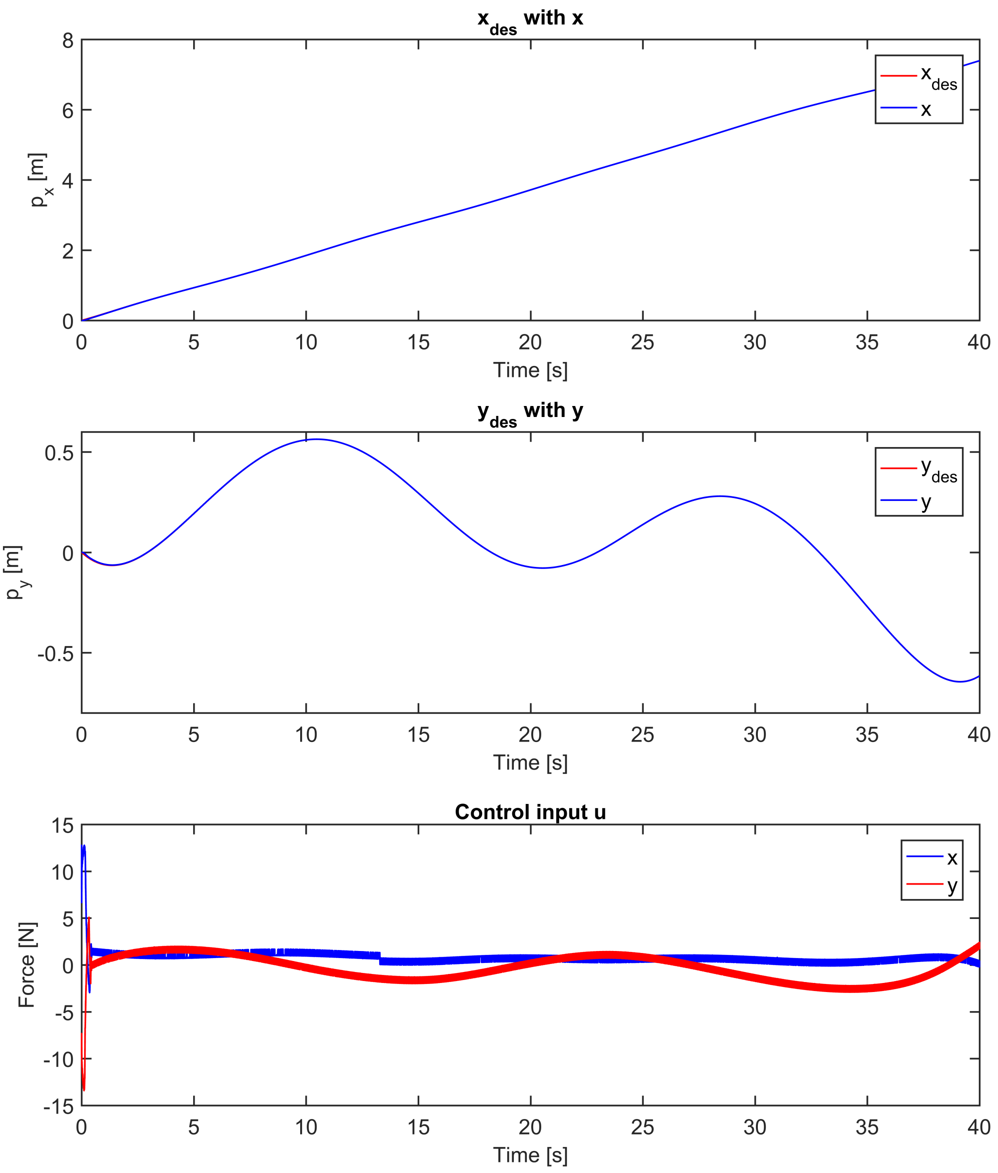


Figure 5.8: USM torpedo mode: Simulation of STA with state observer, sliding surface 3 (a)

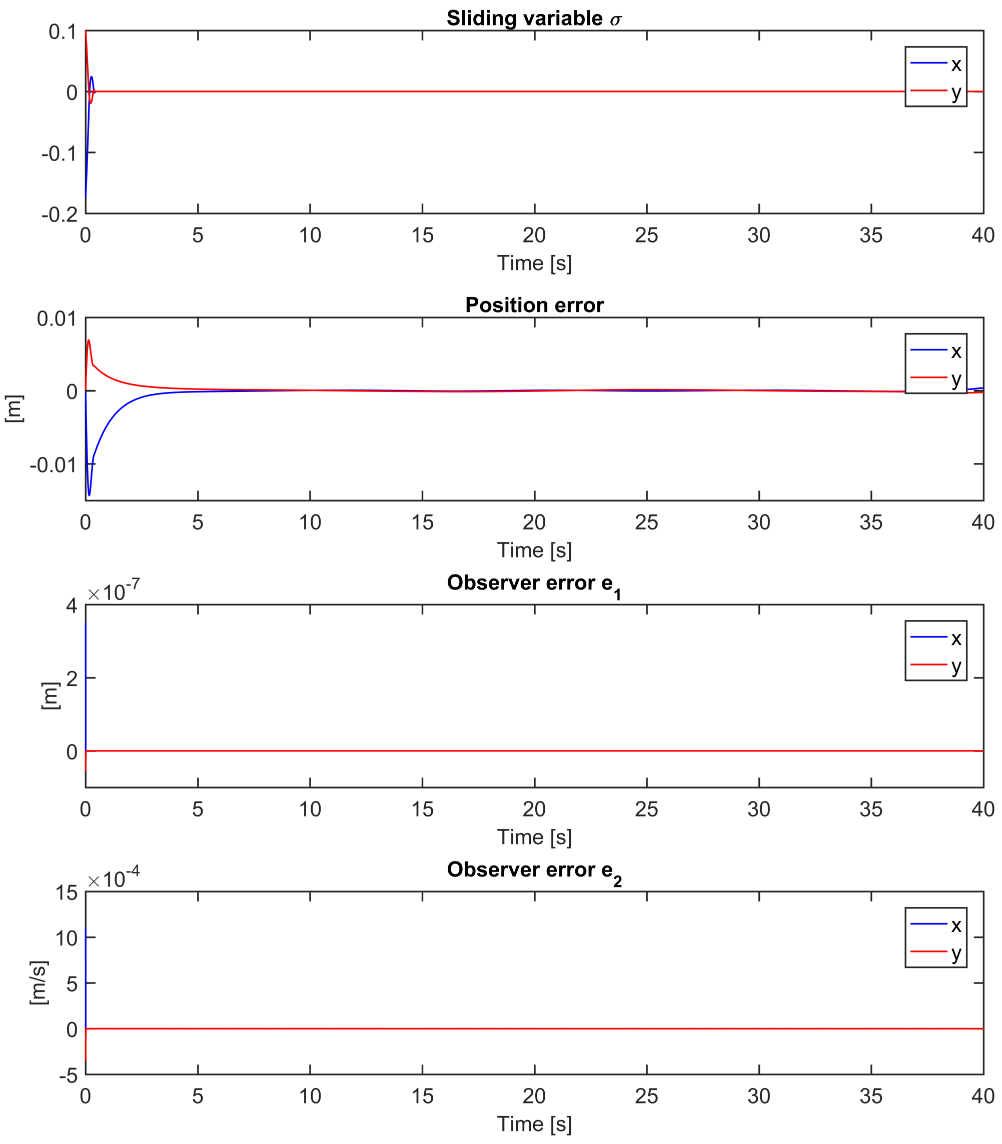


Figure 5.9: USM torpedo mode: Simulation of STA with state observer, sliding surface 3 (b)

Nested third-order sliding mode control

The gains were set to: $\alpha = 10$, $L_D = 0.1$ and $L_{\text{SO}} = 50$.
Sliding surface 1: the actual value of p_{CM} and $\dot{p}$ are used

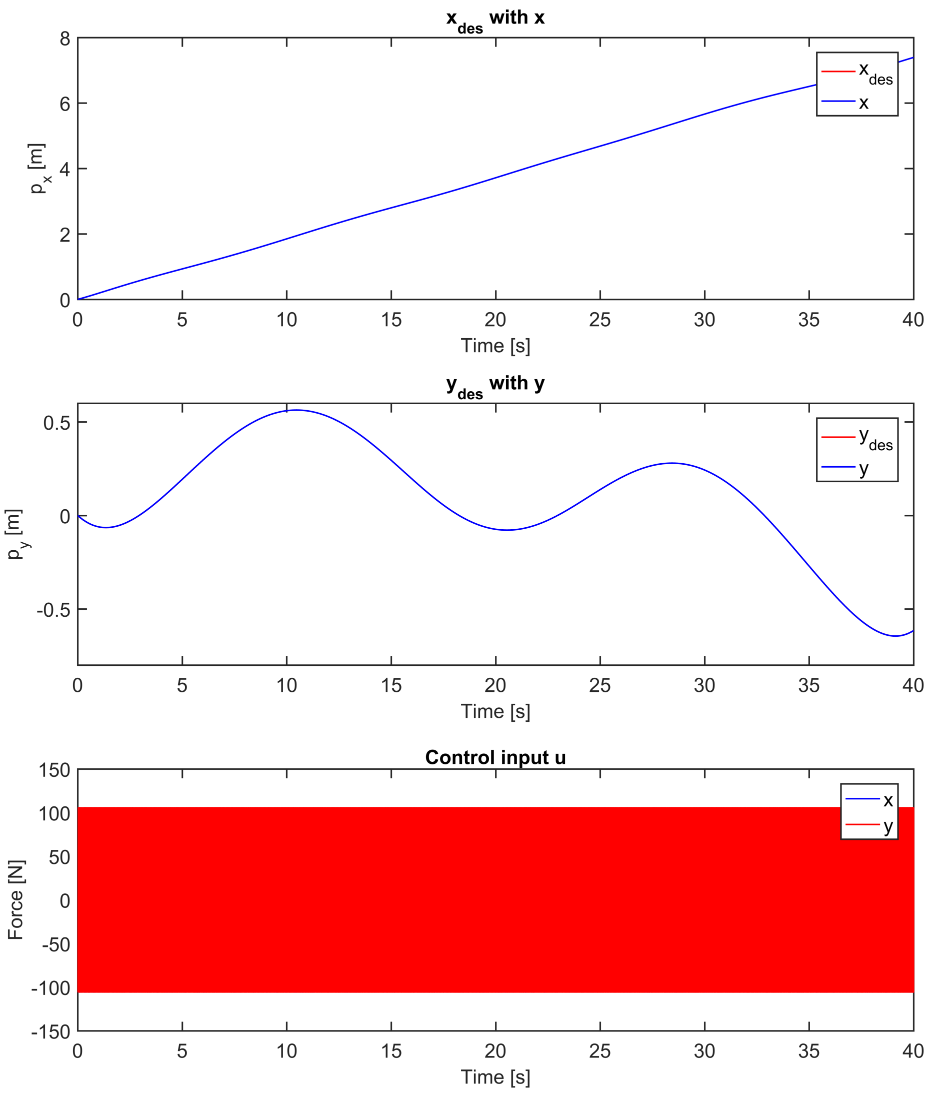


Figure 5.10: USM torpedo mode: Simulation of nested third-order SMC with differentiator, sliding surface 1 (a)

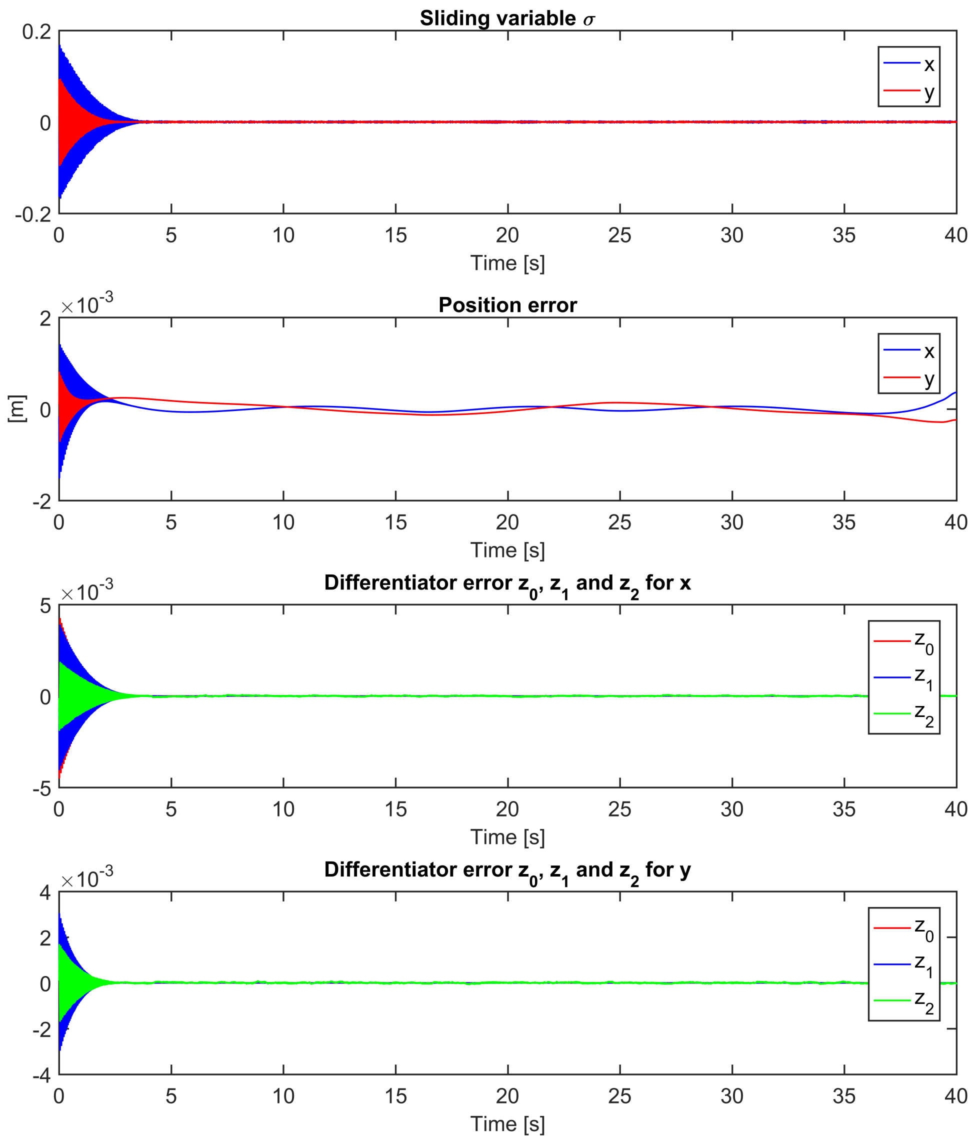


Figure 5.11: USM torpedo mode: Simulation of nested third-order SMC with differentiator, sliding surface 1 (b)

Sliding surface 2: the estimated value for $\dot{p}$ is used

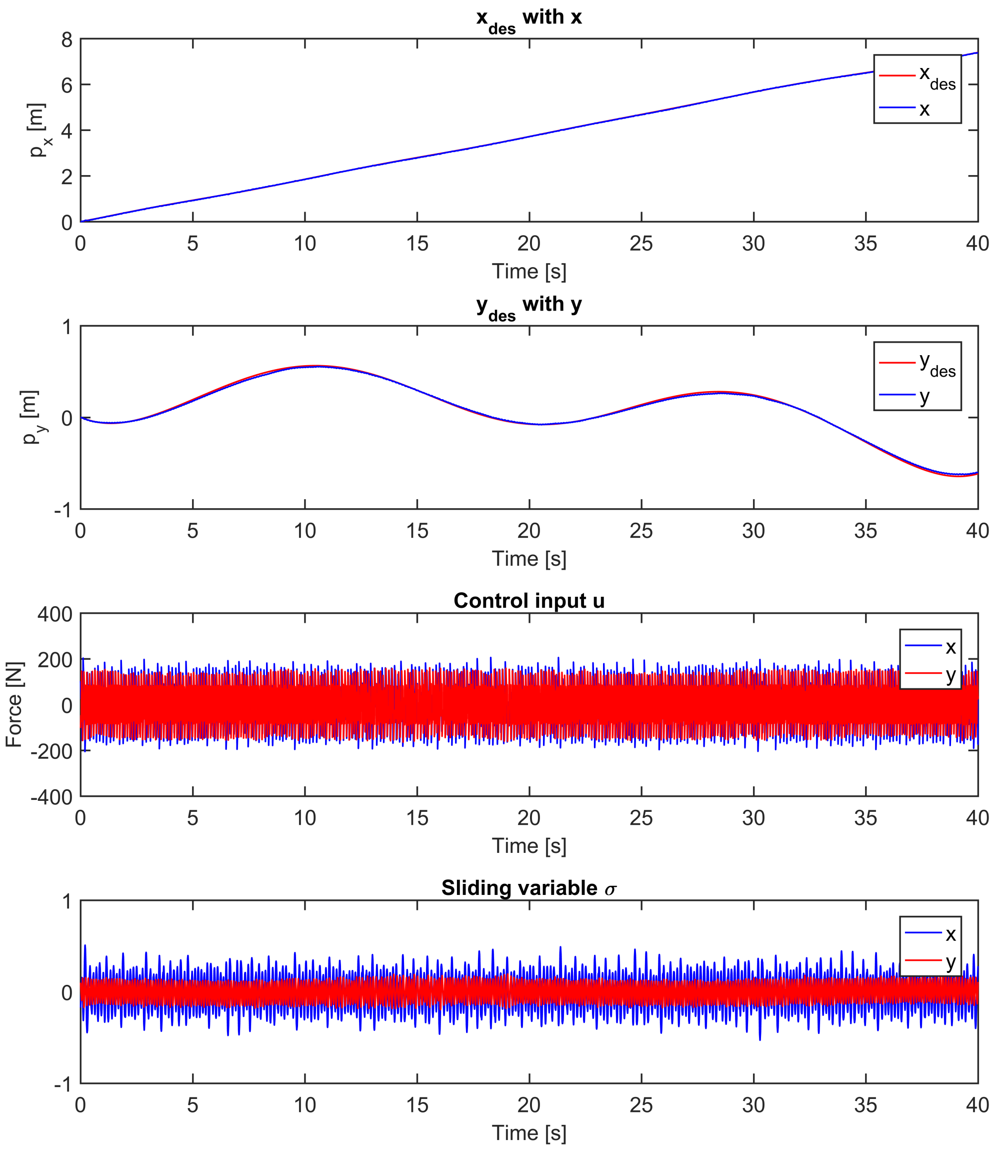


Figure 5.12: USM torpedo mode: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 2 (a)

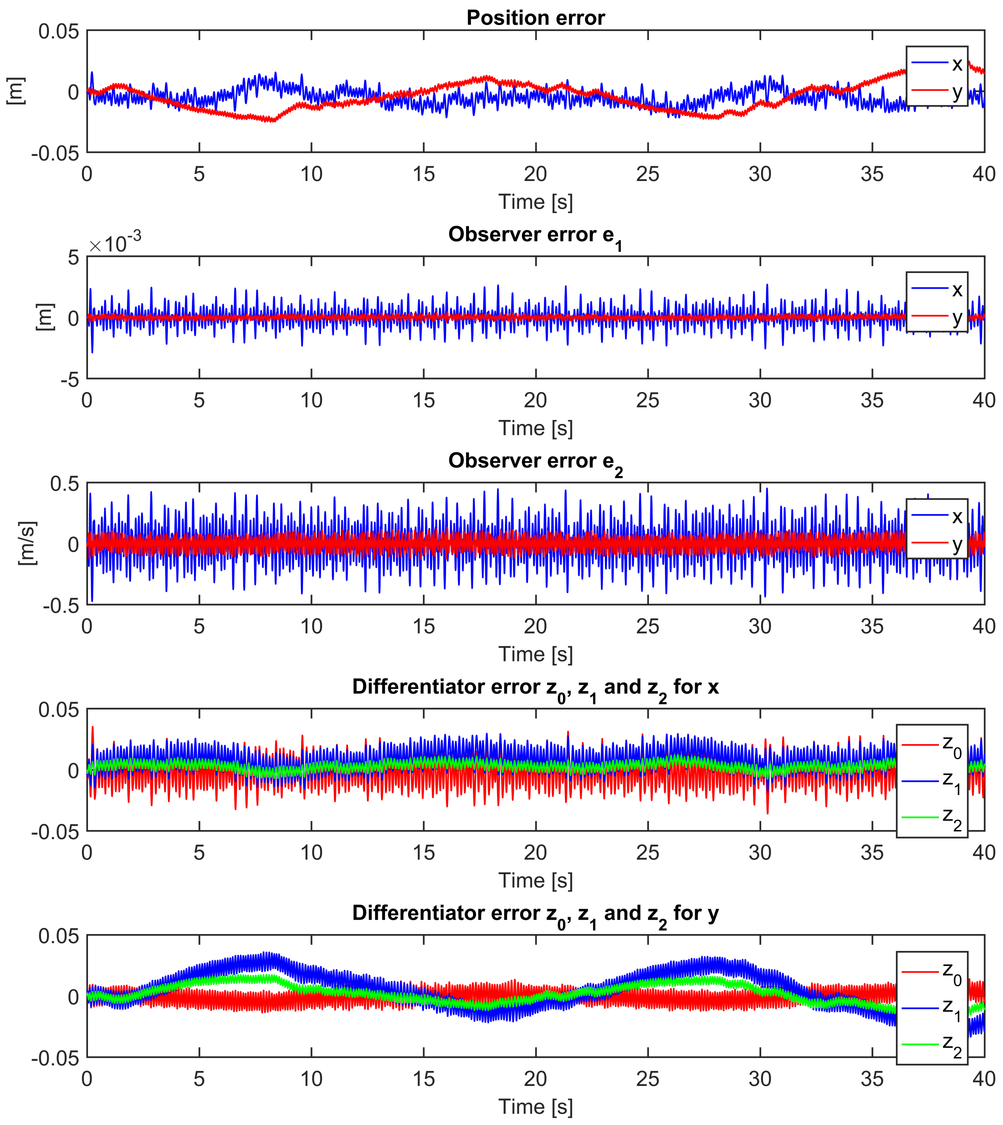


Figure 5.13: USM torpedo mode: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 2 (b)

Sliding surface 3: the estimated value for p_{CM} and $\dot{p}$ are used

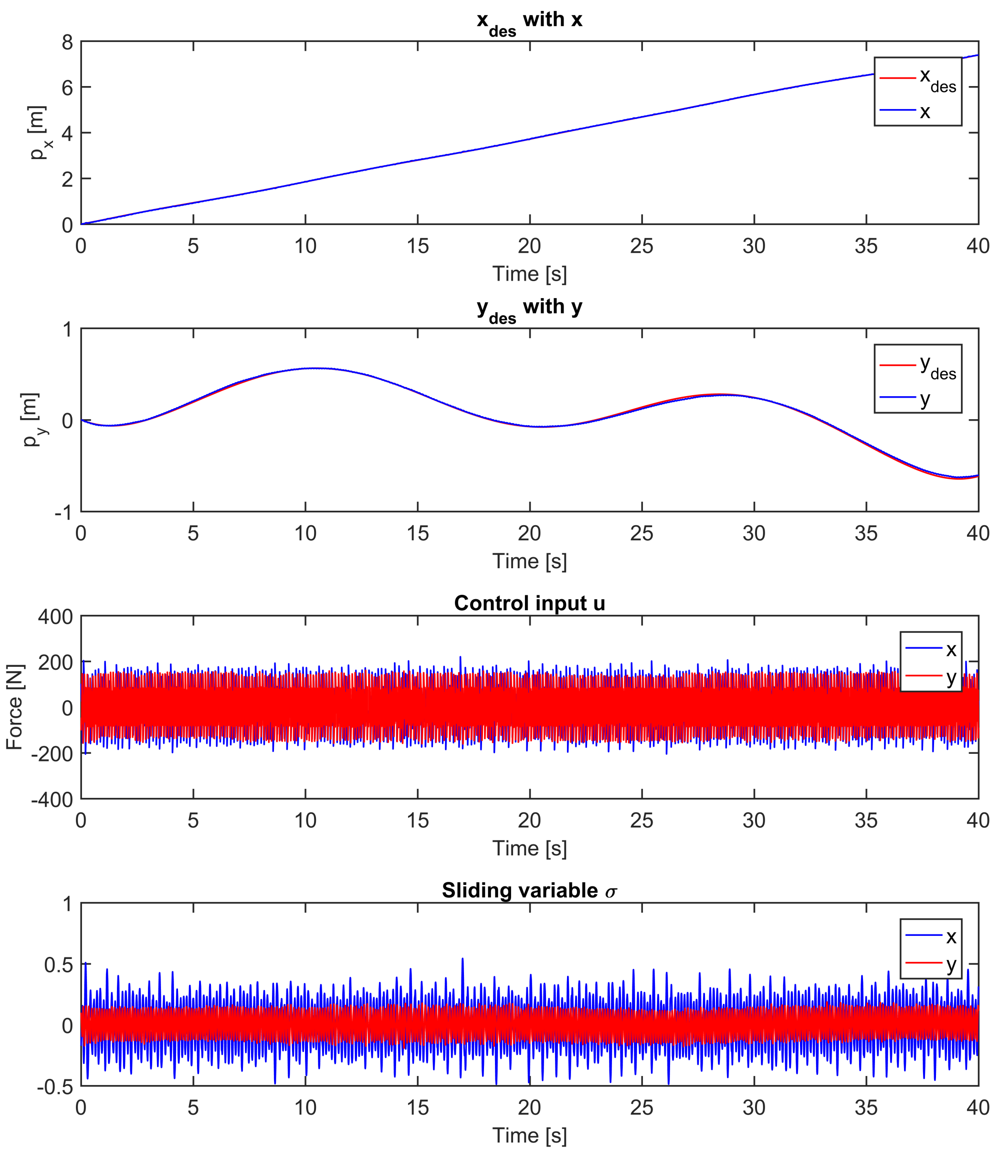


Figure 5.14: USM torpedo mode: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 3 (a)

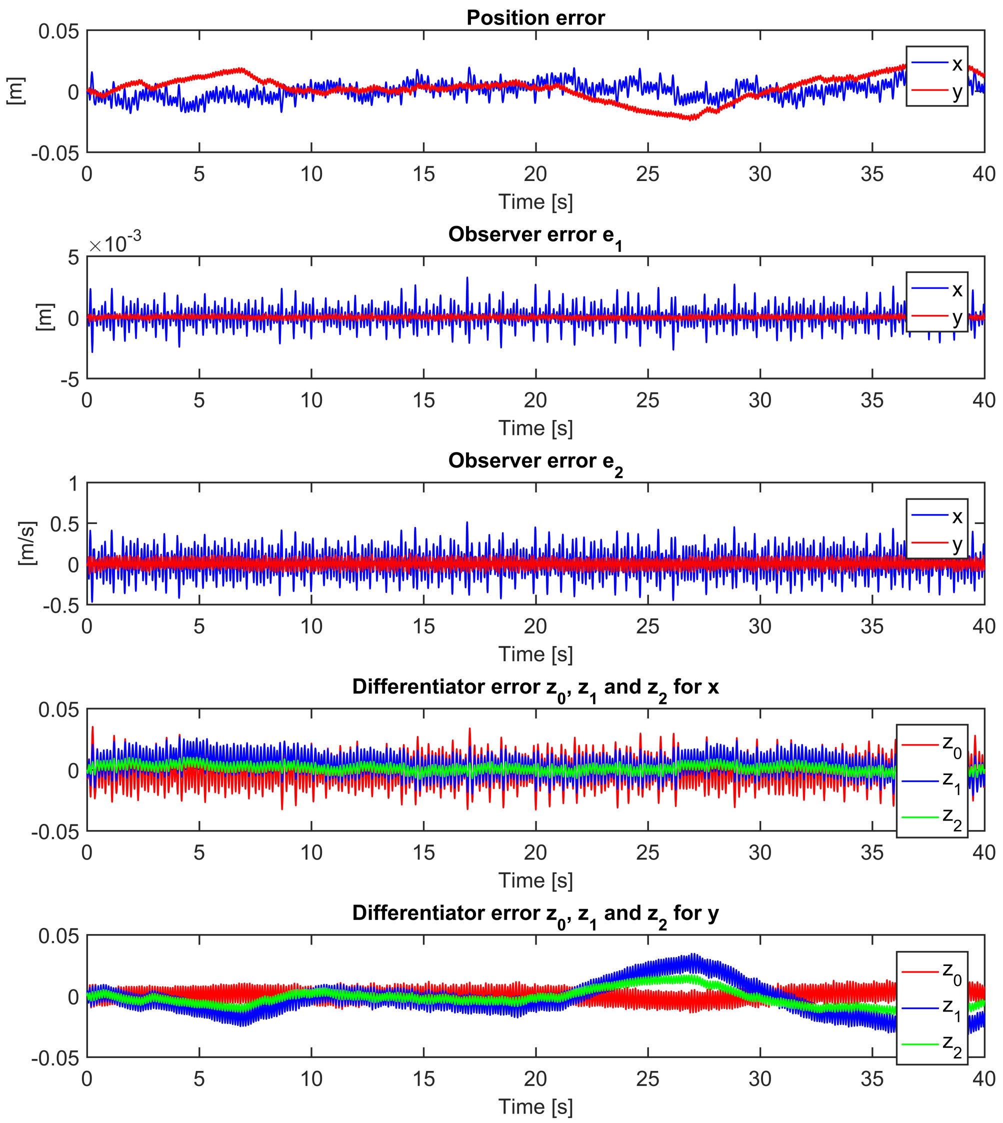


Figure 5.15: USM torpedo mode: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 3 (b)

Quasi-continuous third-order sliding mode control

The gains were set to: $\alpha = 10$, $L_D = 1$ and $L_{\mathrm{SO}} = 50$.
Sliding surface 1: the actual value of p_{CM} and $\dot{p}$ are used

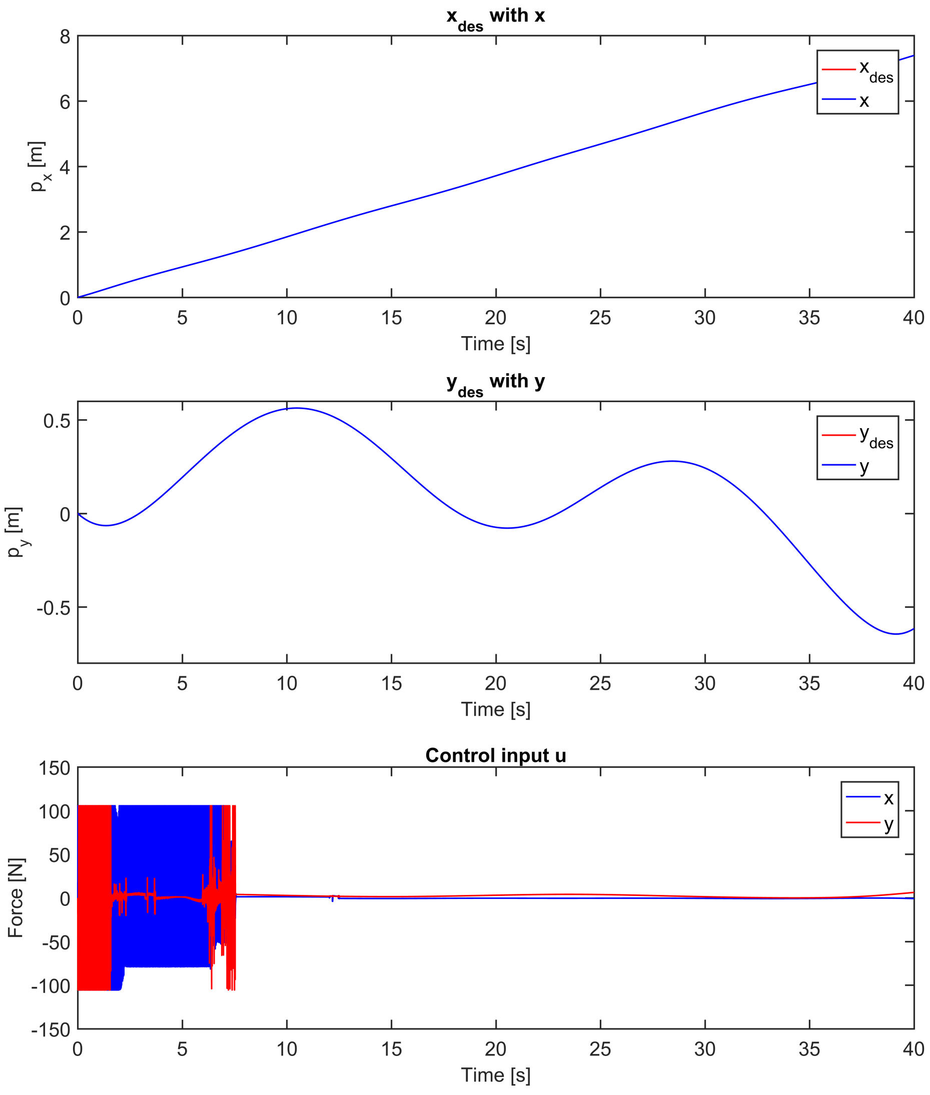


Figure 5.16: USM torpedo mode: Simulation of quasi-continuous third-order SMC with differentiator, sliding surface 1 (a)

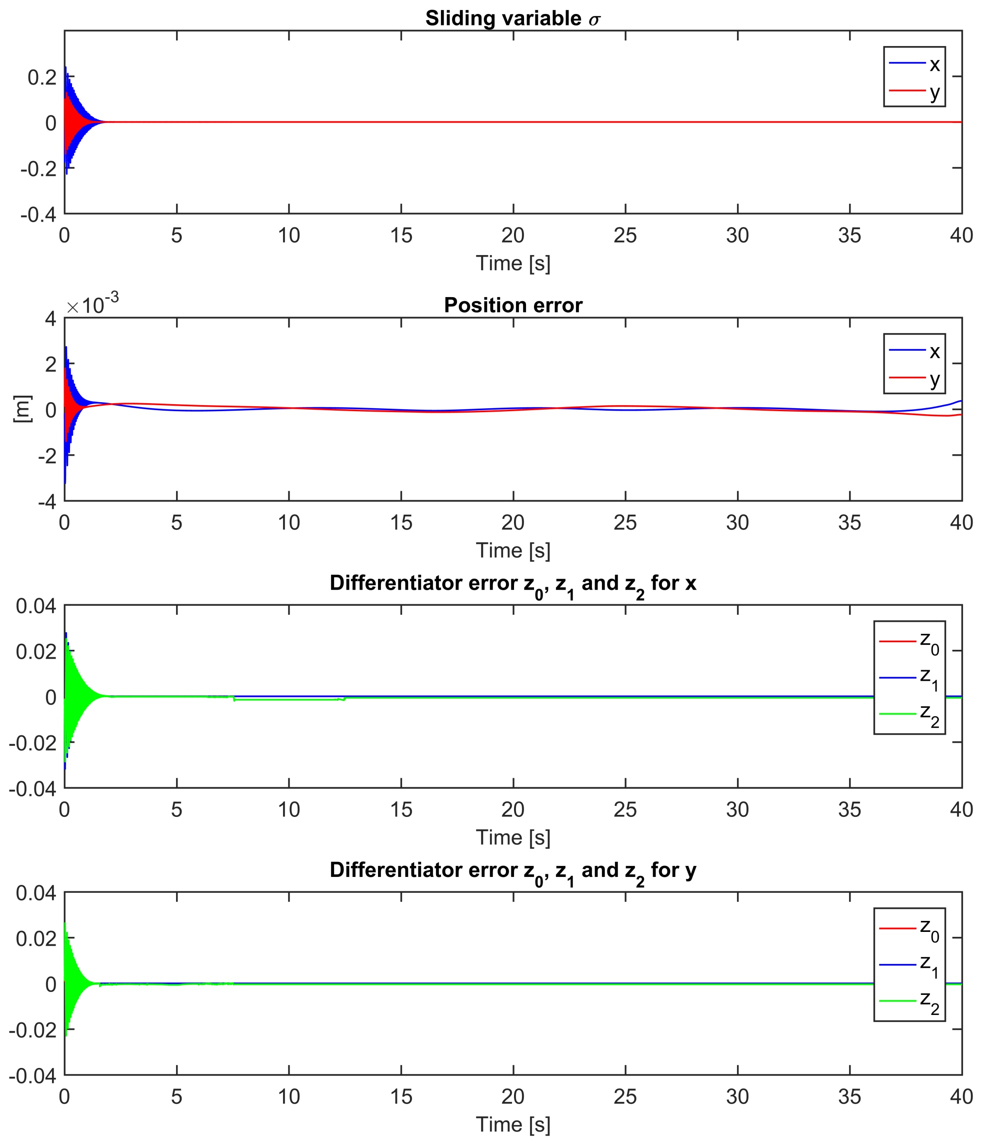


Figure 5.17: USM torpedo mode: Simulation of quasi-continuous third-order SMC with differentiator, sliding surface 1 (b)

Sliding surface 2: the estimated value for $\dot{p}$ is used

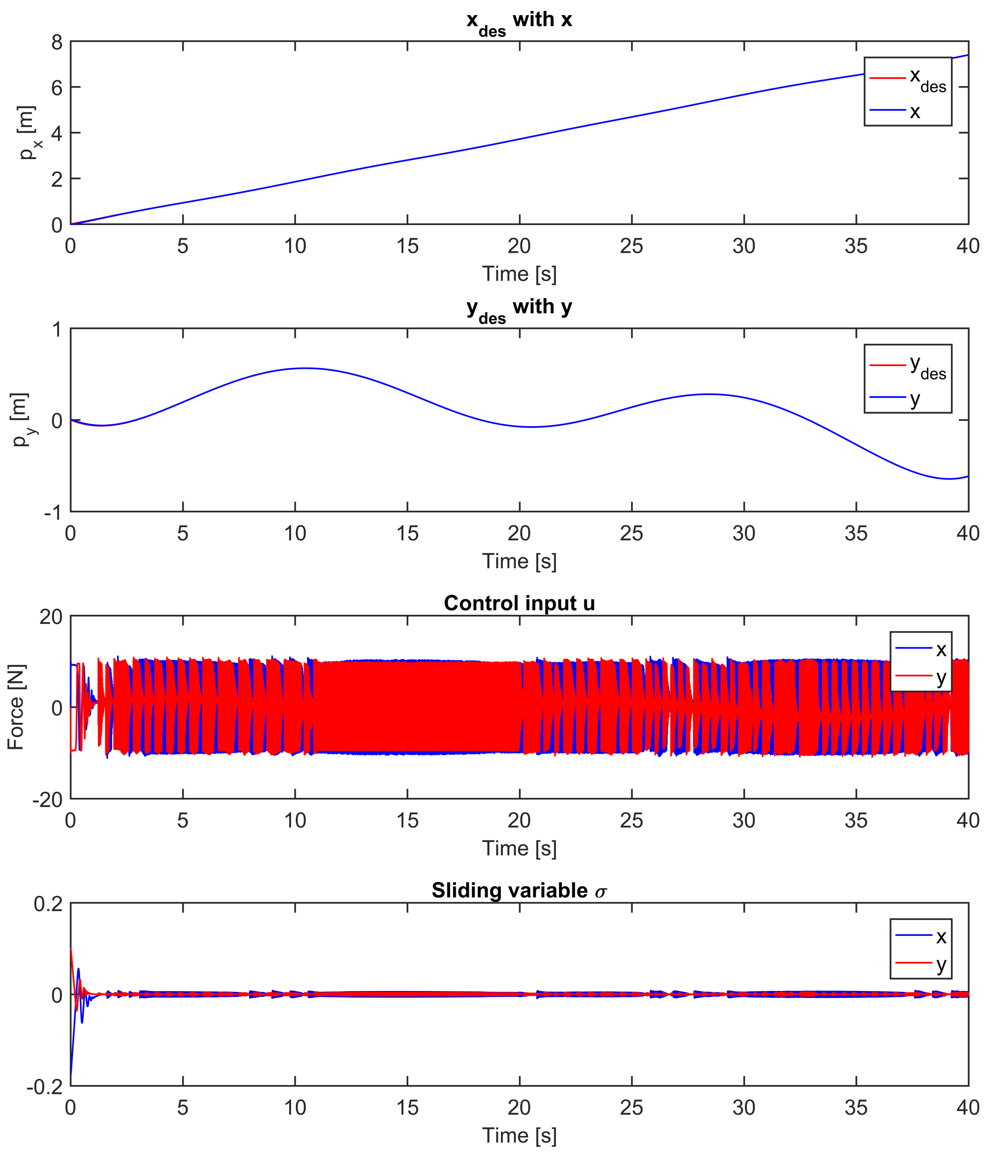


Figure 5.18: USM torpedo mode: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 2 (a)

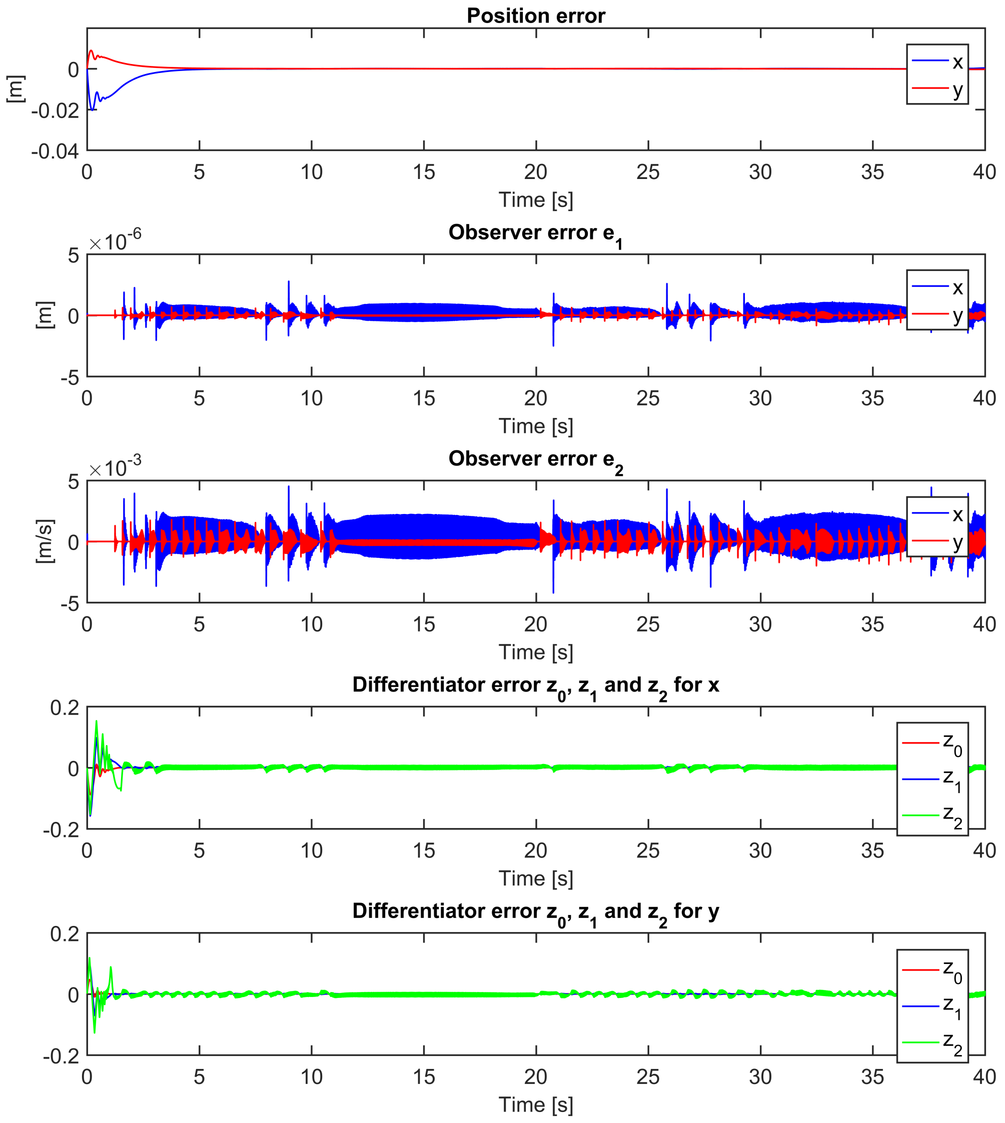


Figure 5.19: USM torpedo mode: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 2 (b)

Sliding surface 3: Estimated value for p_{CM} and $\dot{p}$ are used

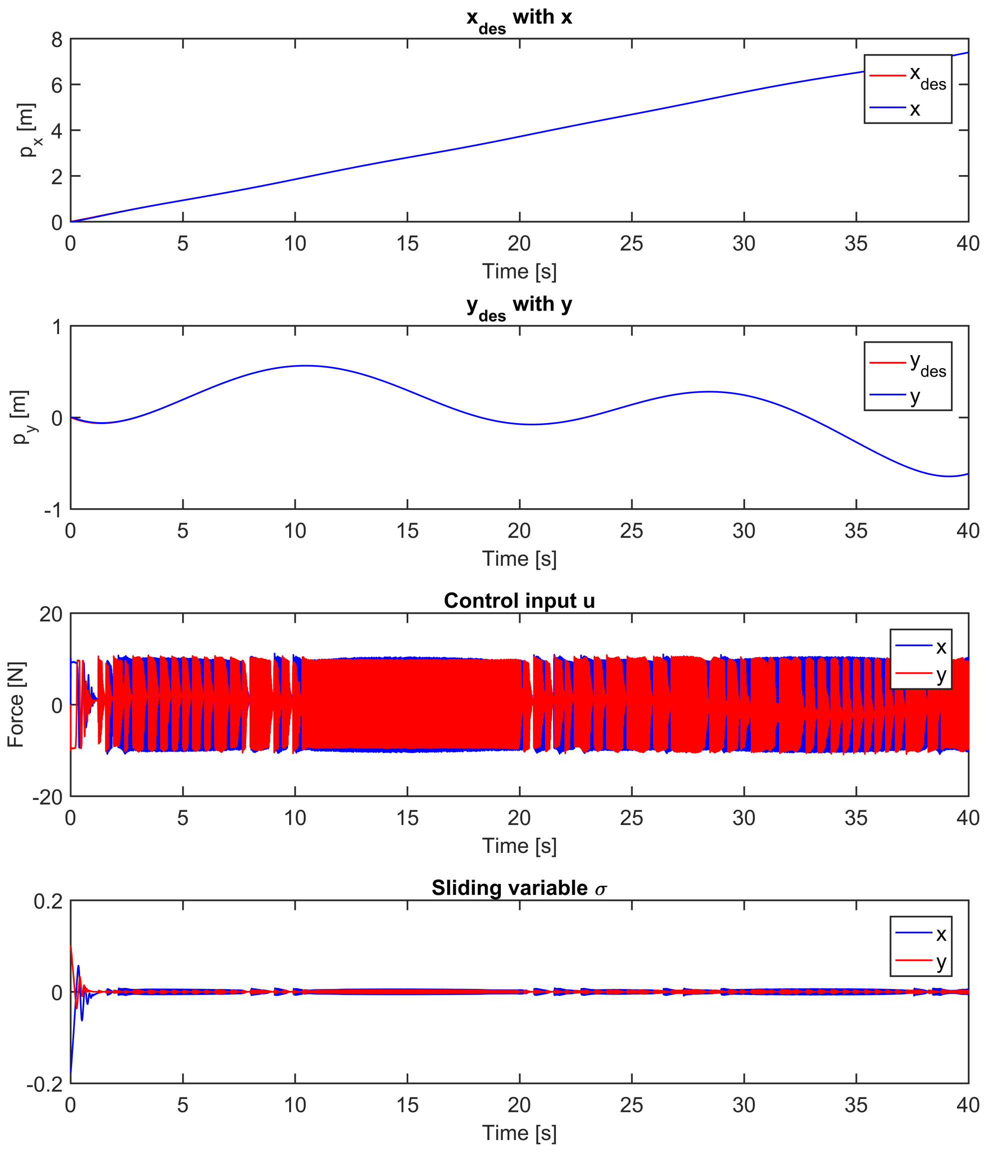


Figure 5.20: USM torpedo mode: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 3 (a)

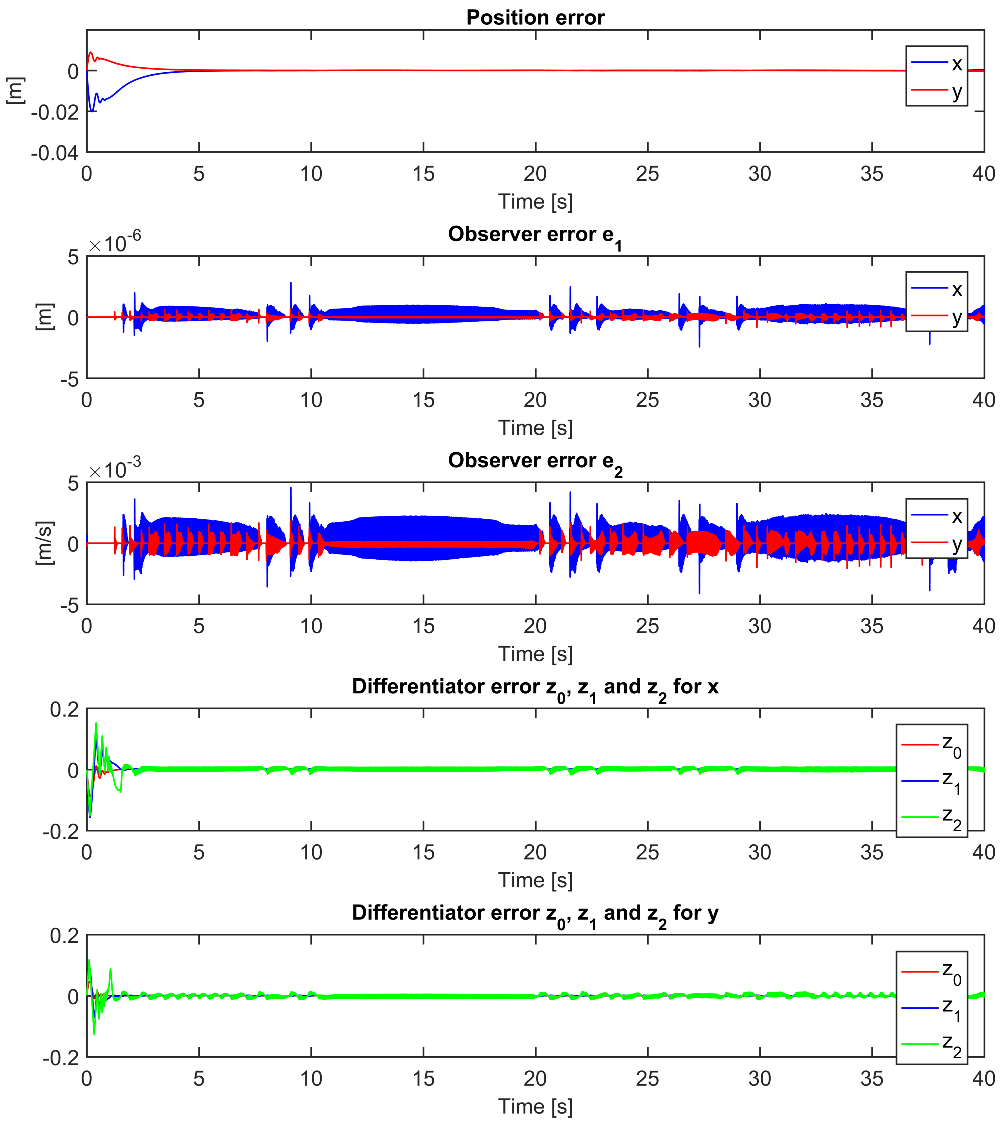


Figure 5.21: USM torpedo mode: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 3 (b)

The PD-controller

The gains were set to: $k_d^{CM} = 10$, $k_p^{CM} = 0.1$ and $L_{\text{SO}} = 50$.
Actual value of p_{CM} and $\dot{p}$ are used

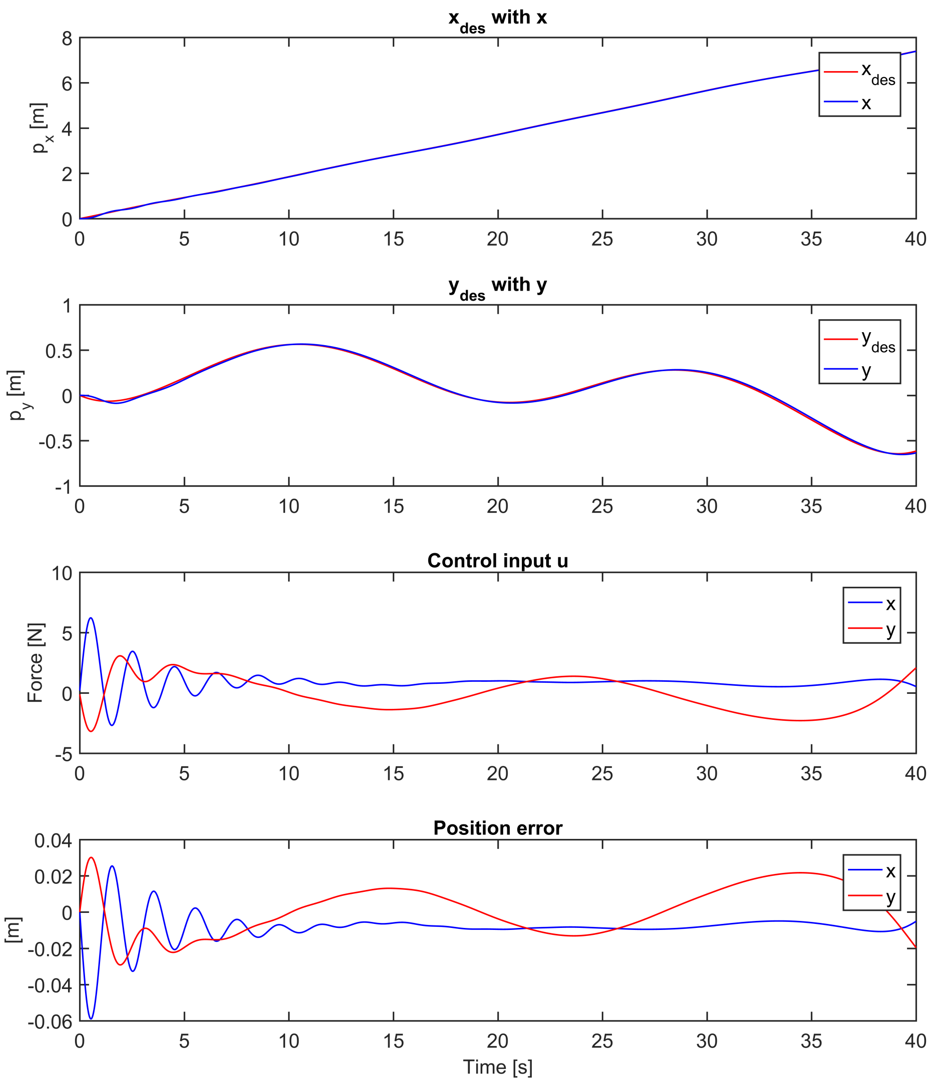


Figure 5.22: USM torpedo mode: Simulations of PD-controller when the actual value of p_{CM} and $\dot{p}$ are used

Estimated value for $\dot{p}$ is used

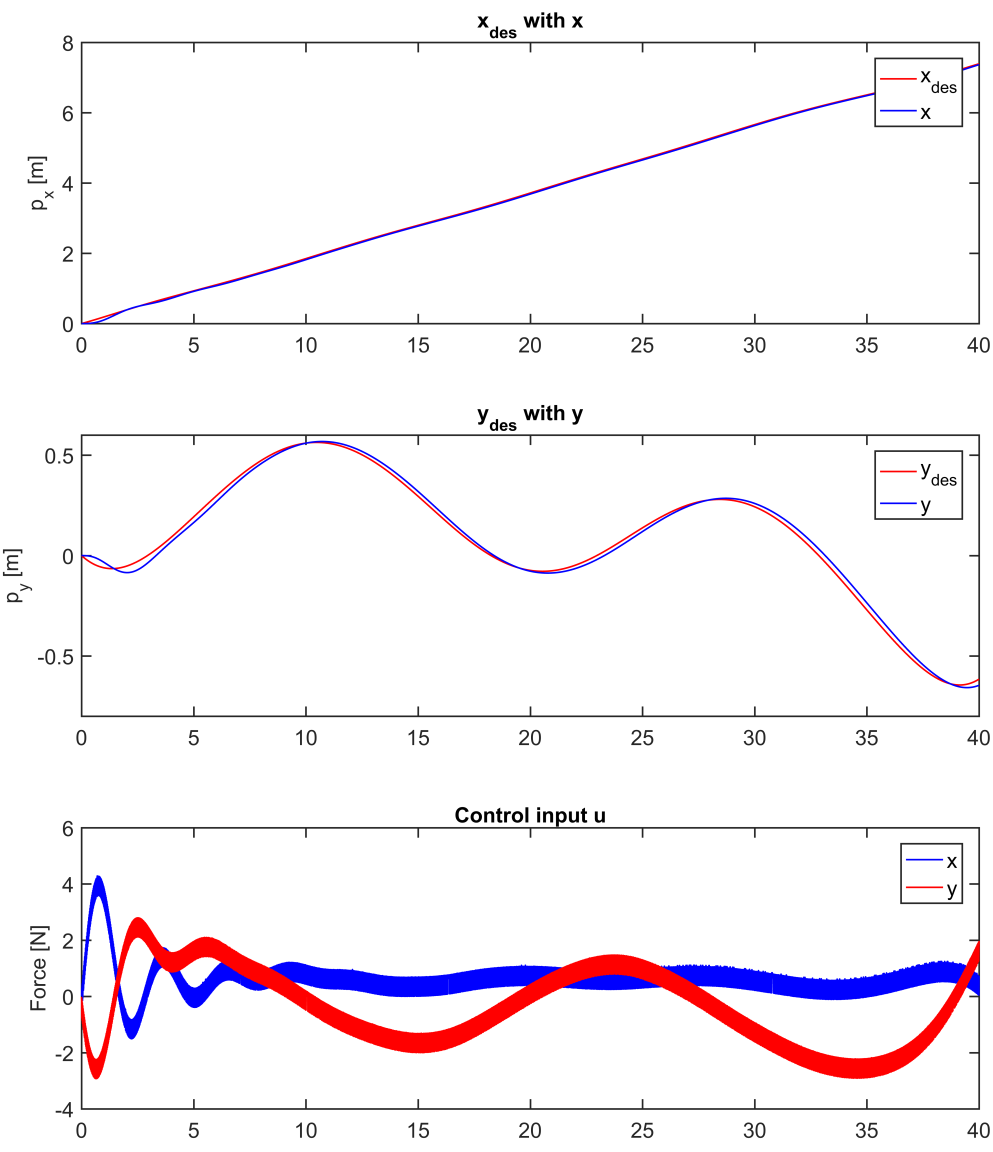


Figure 5.23: USM torpedo mode: Simulations of PD-controller when the estimated value for $\dot{p}$ is used (a)

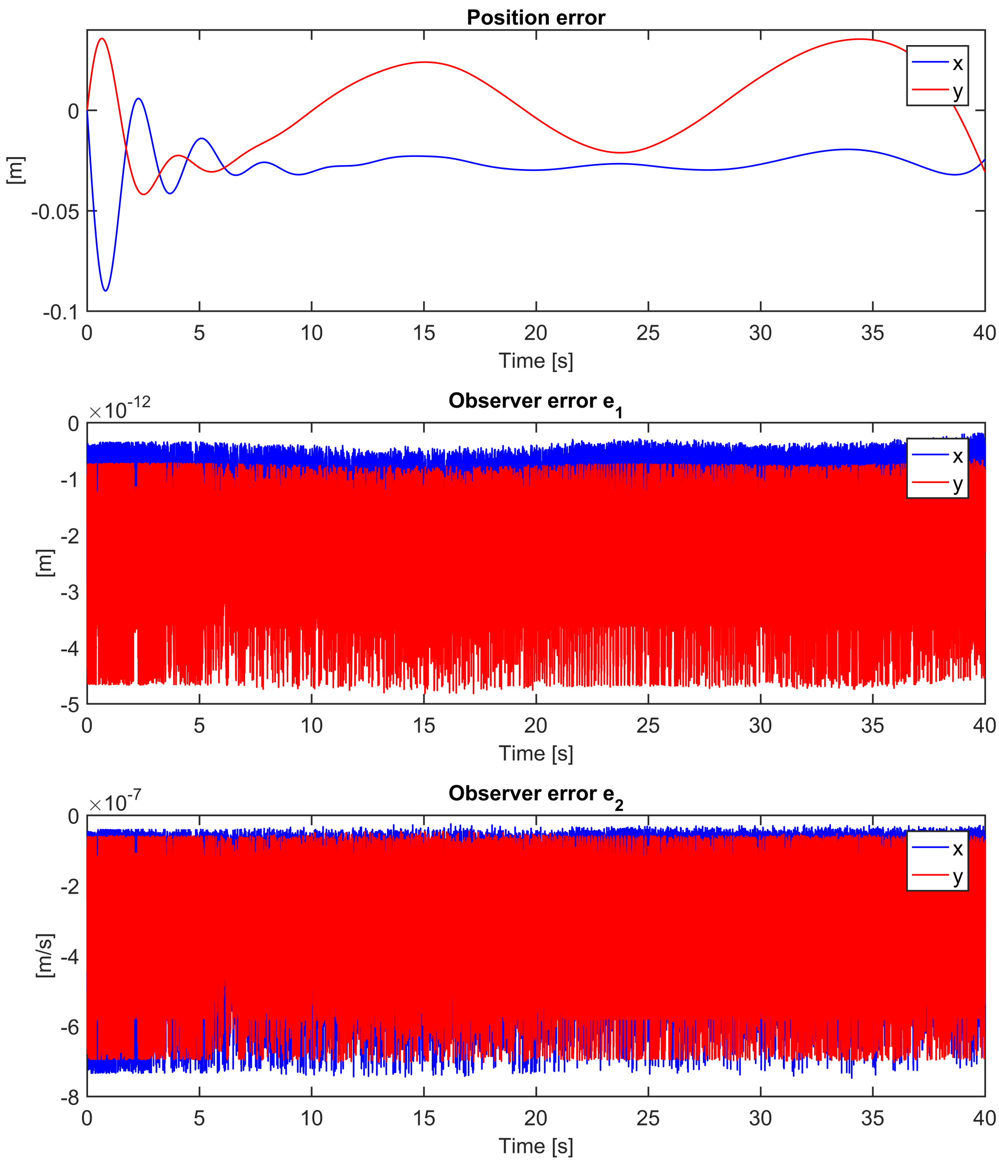


Figure 5.24: USM torpedo mode: Simulations of PD-controller when the estimated value for $\dot{p}$ is used (b)

Estimated value for p_{CM} and $\dot{p}$ are used

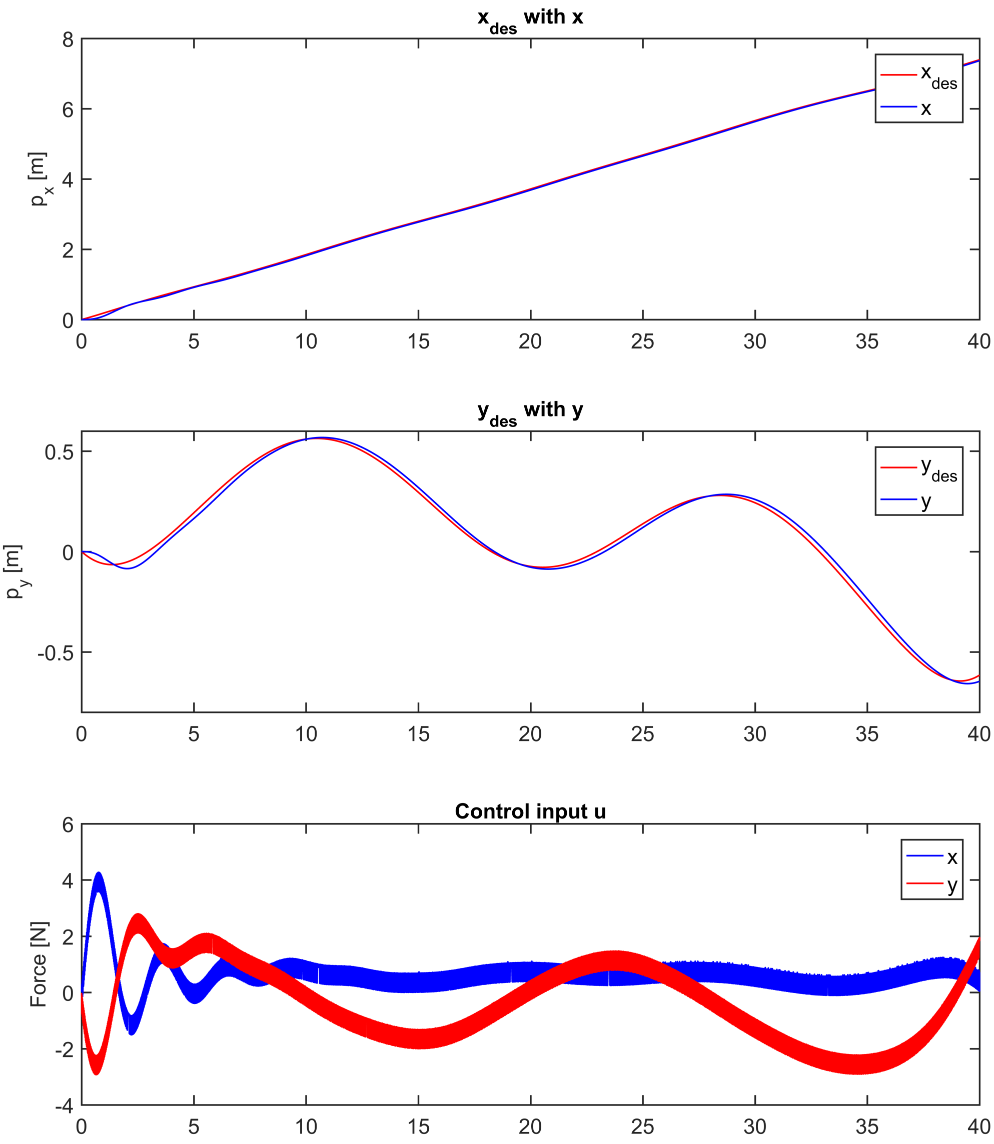


Figure 5.25: USM torpedo mode: Simulations of PD-controller when the estimated value for p_{CM} and $\dot{p}$ are used (a)

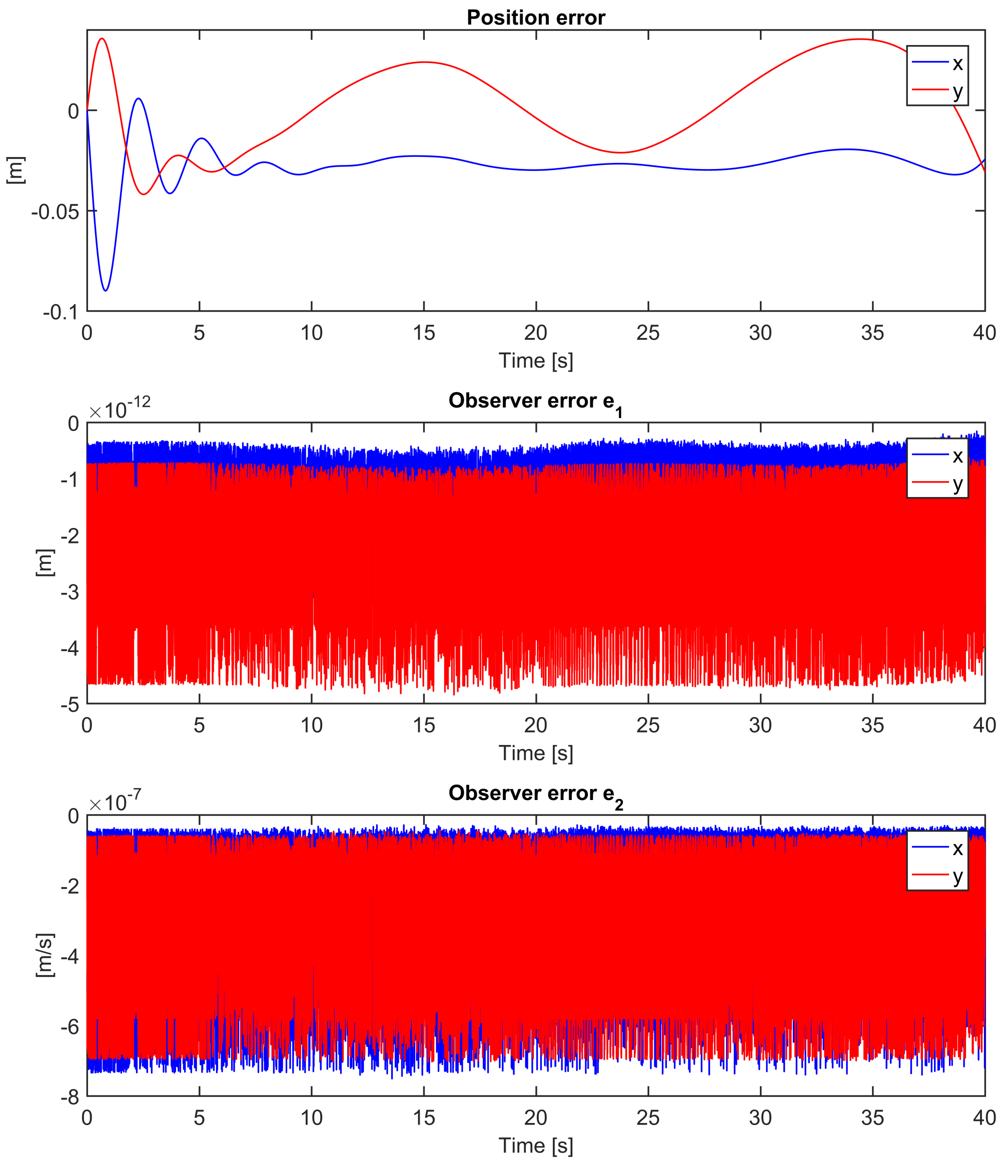


Figure 5.26: USM torpedo mode: Simulations of PD-controller when the estimated value for p_{CM} and $\dot{p}$ are used (b)

5.5.2 Operation mode

The super-twisting algorithm

The gains were set to: $K = 10$ and $L_{\mathrm{SO}} = 50$.
Sliding surface 1: the actual value of p_{CM} and $\dot{p}$ are used

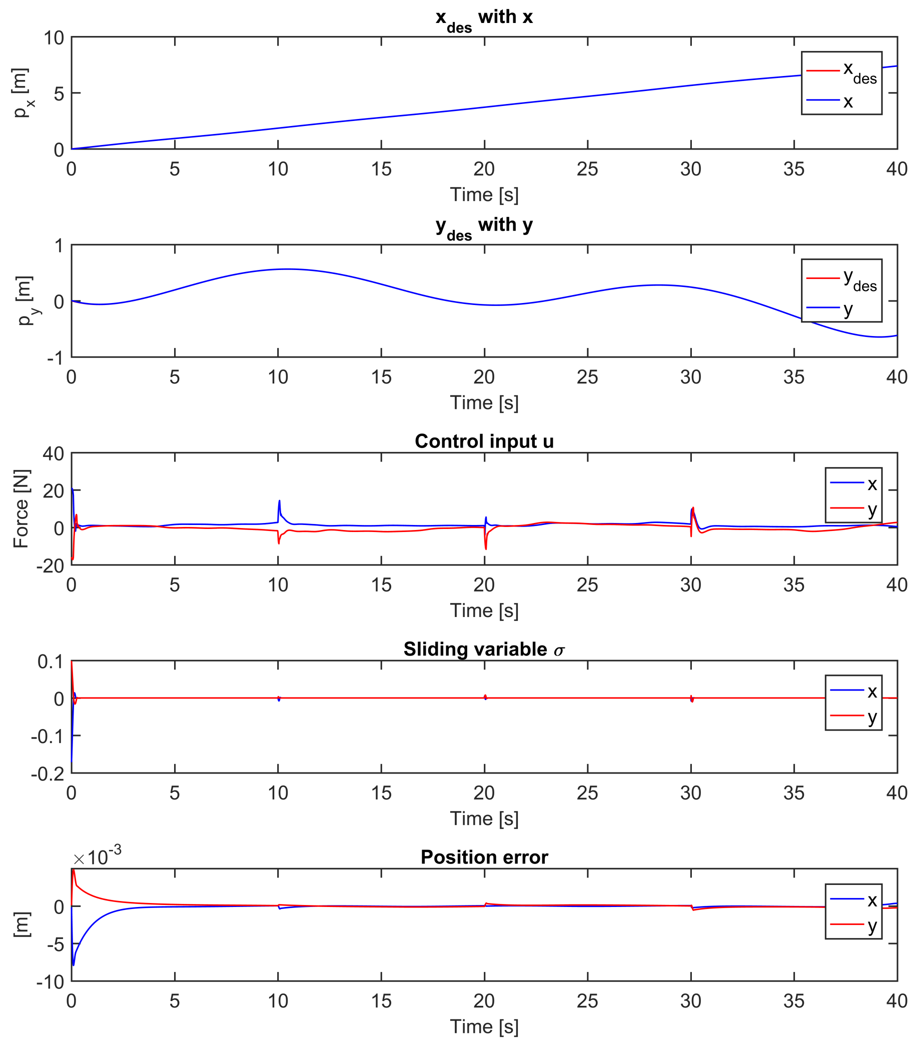


Figure 5.27: USM operation mode: Simulation of STA, sliding surface 1

Sliding surface 2: the estimated value for $\dot{p}$ is used

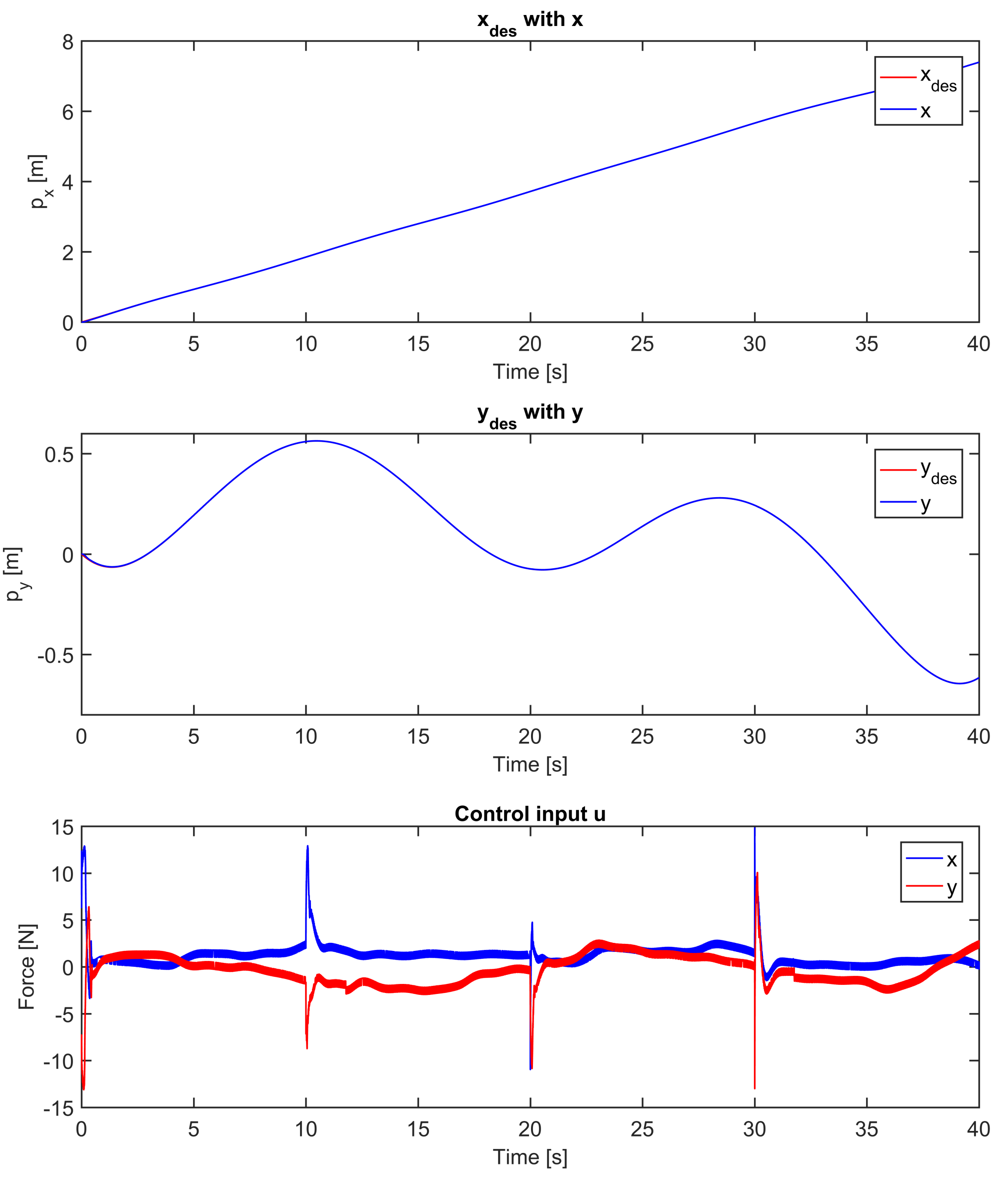


Figure 5.28: USM operation mode: Simulation of STA with state observer, sliding surface 2 (a)

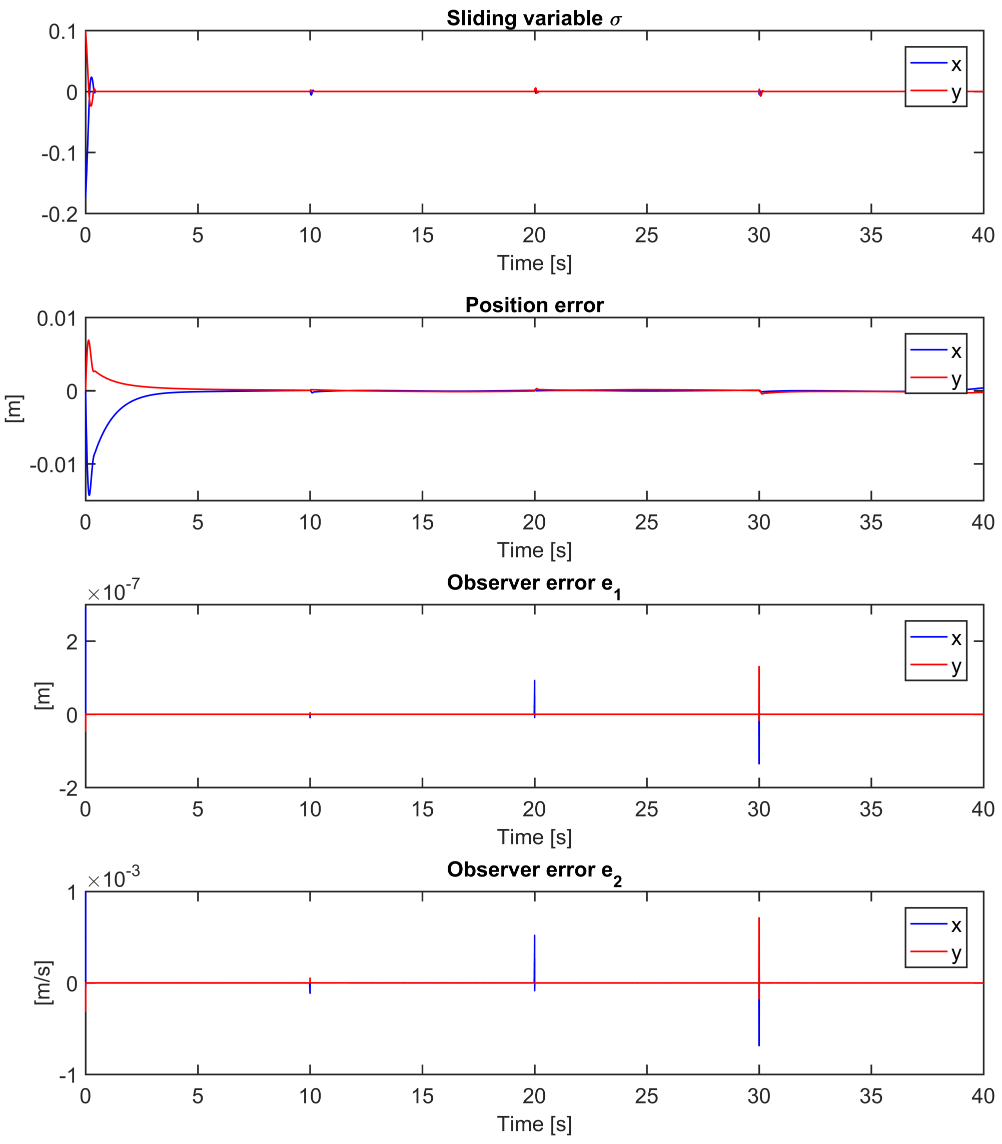


Figure 5.29: USM operation mode: Simulation of STA with state observer, sliding surface 2 (b)

Sliding surface 3: the estimated value for p_{CM} and $\dot{p}$ are used

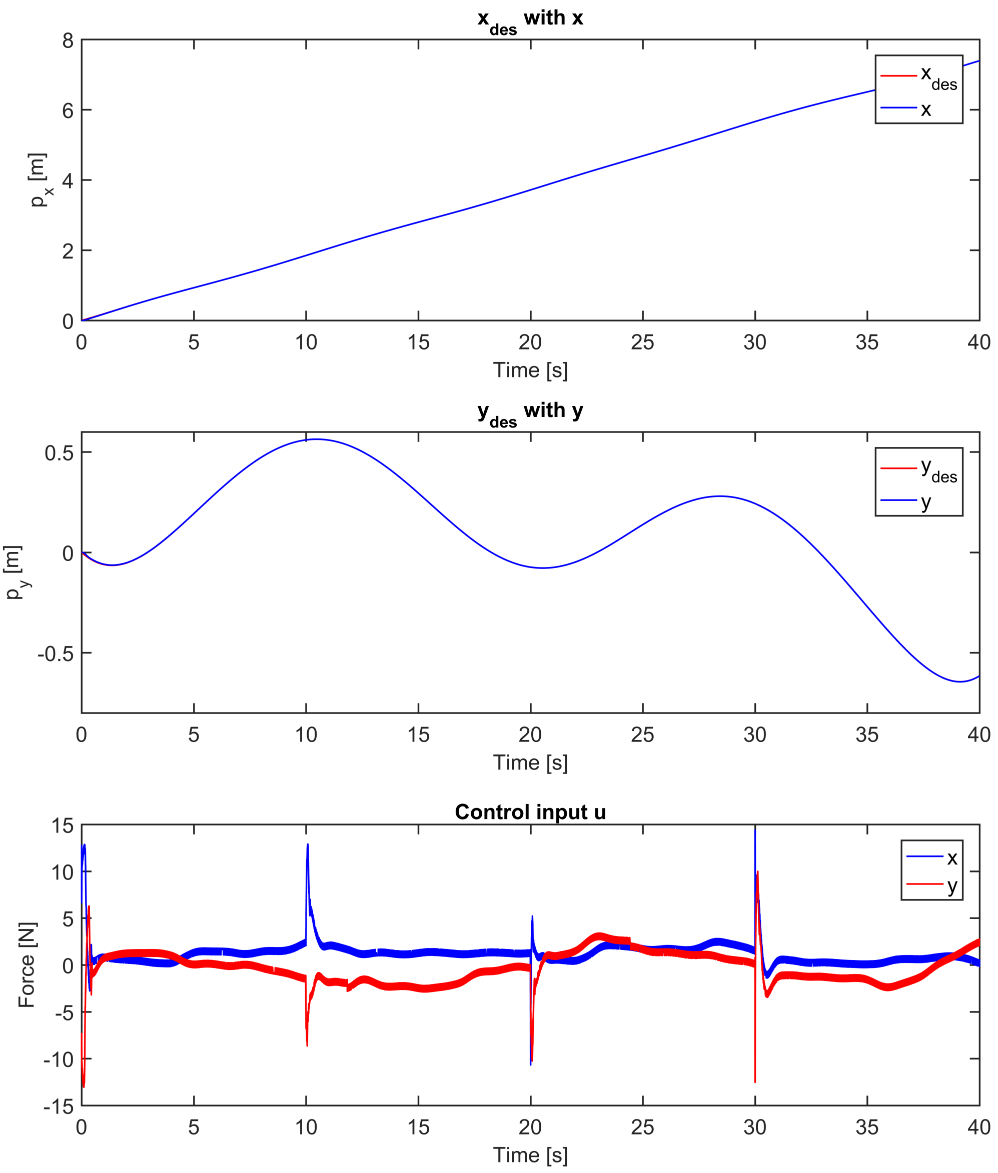


Figure 5.30: USM operation mode: Simulation of STA with state observer, sliding surface 3 (a)

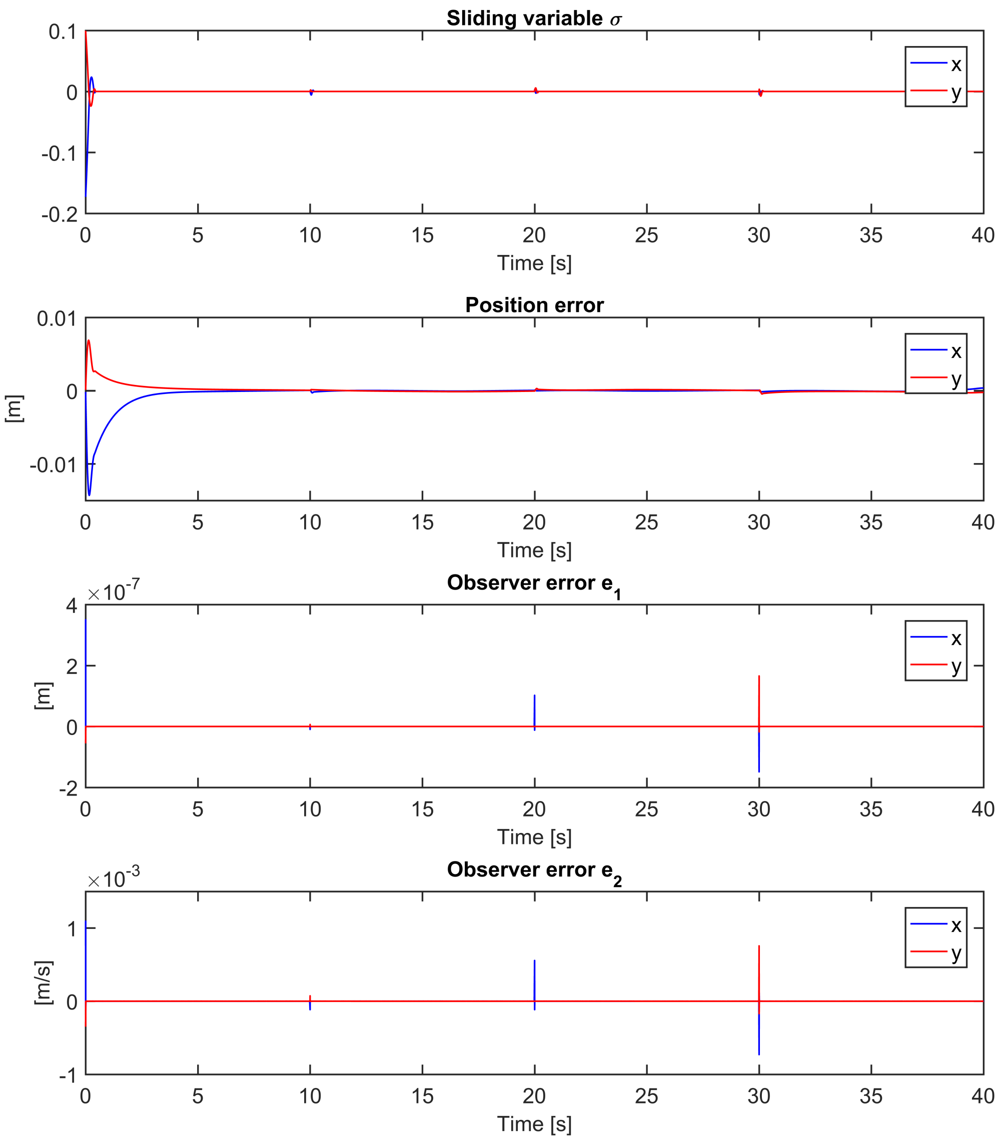


Figure 5.31: USM operation mode: Simulation of STA with state observer, sliding surface 3 (b)

Nested third-order sliding mode control

The gains were set to: $\alpha = 10$, $L_D = 0.1$ and $L_{\mathrm{SO}} = 50$.
Sliding surface 1: the actual value of p_{CM} and $\dot{p}$ are used

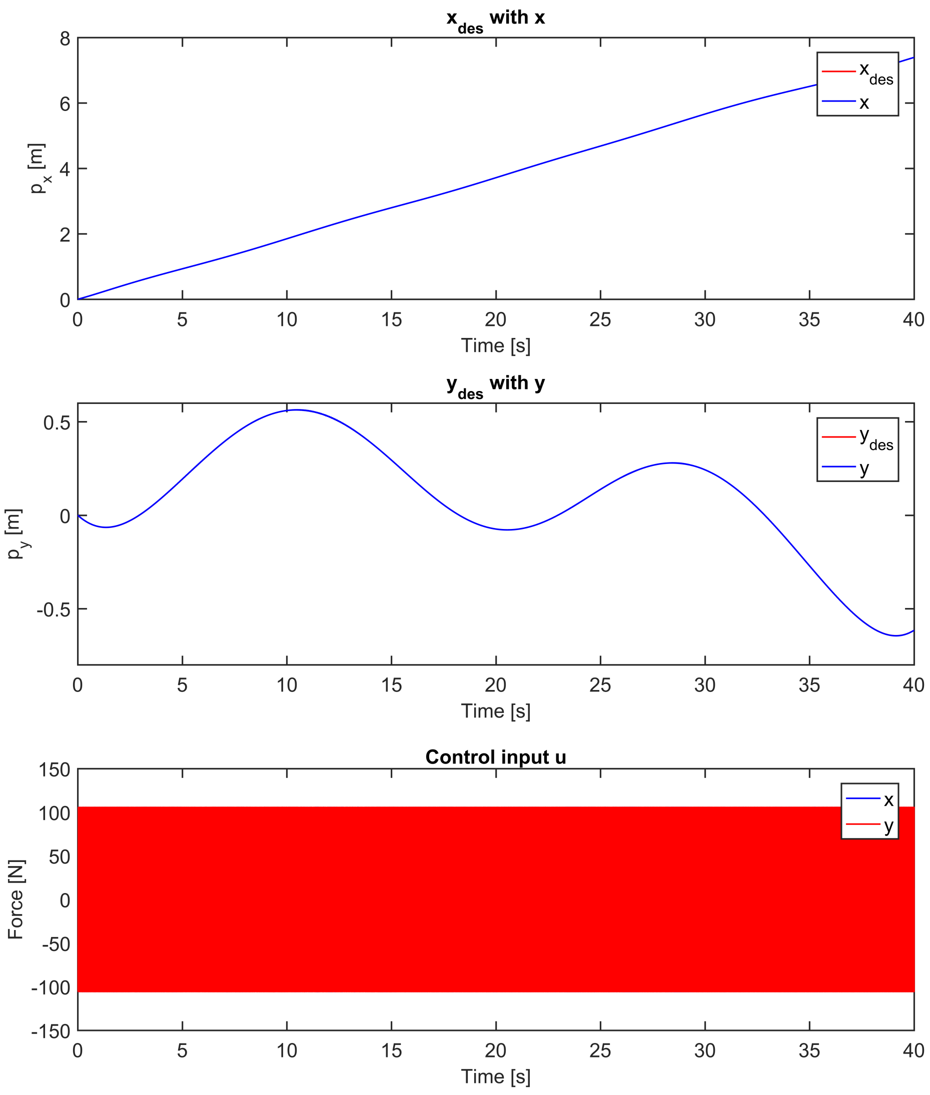


Figure 5.32: USM operation mode: Simulation of nested third-order SMC with differentiator, sliding surface 1 (a)

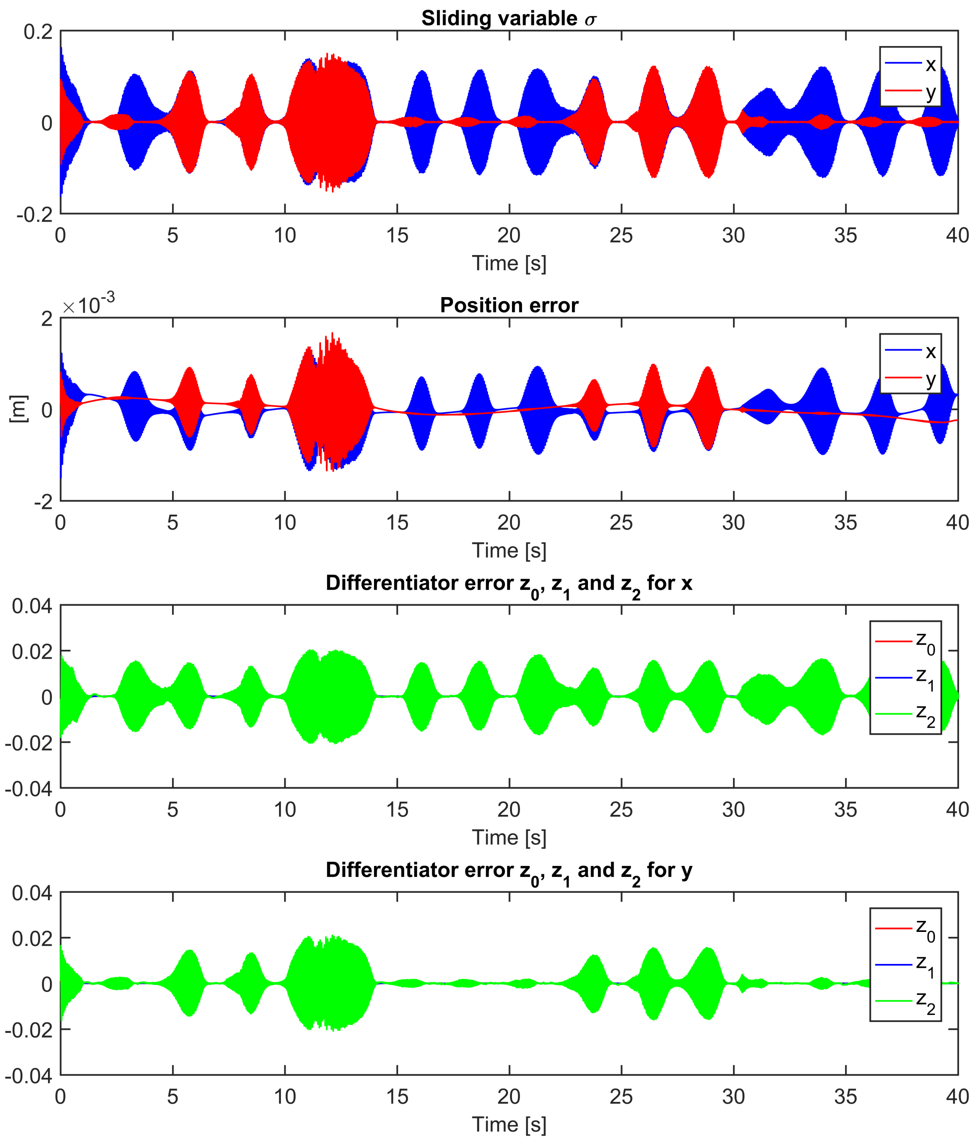


Figure 5.33: USM operation mode: Simulation of nested third-order SMC with differentiator, sliding surface 1 (b)

Sliding surface 2: the estimated value for $\dot{p}$ is used

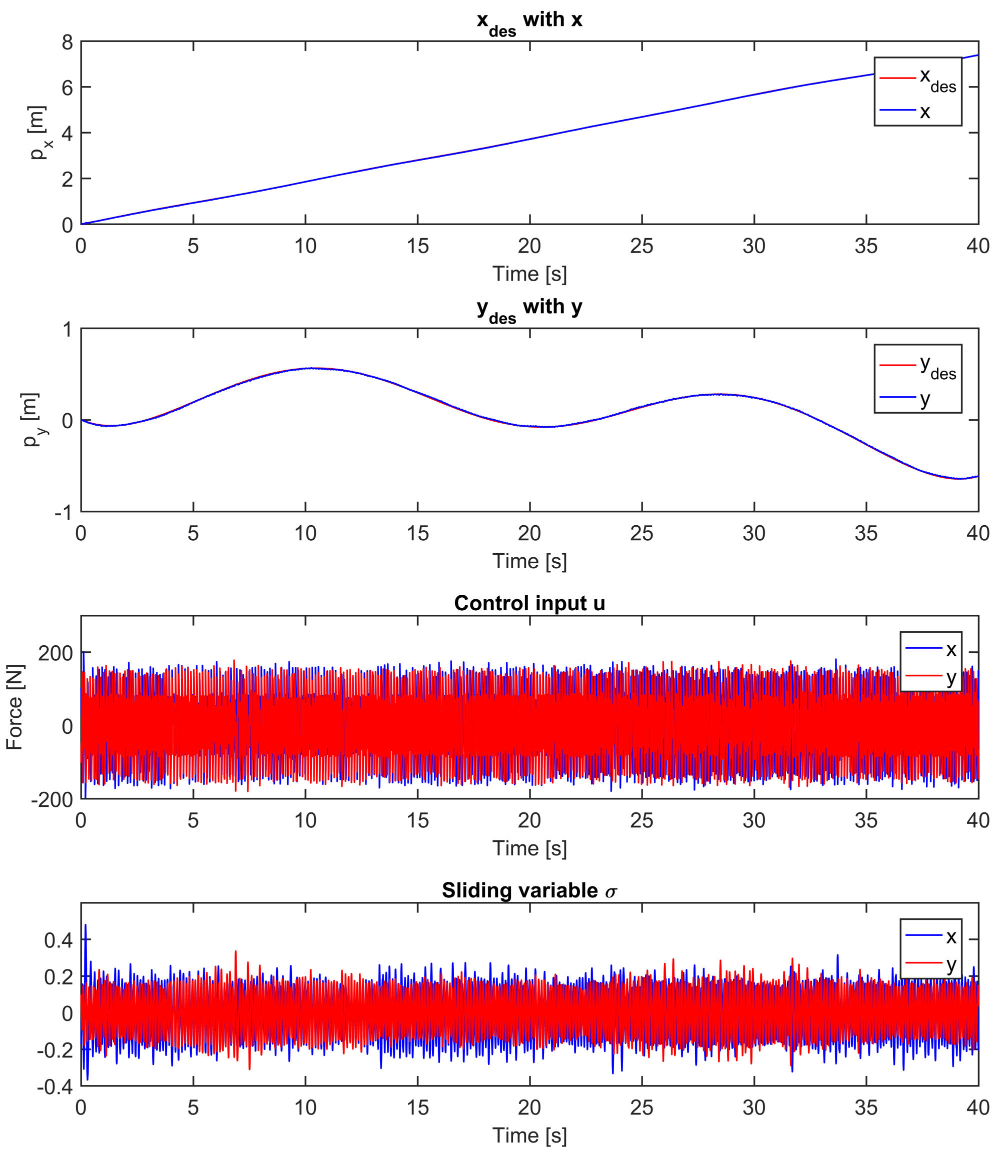


Figure 5.34: USM operation mode: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 2 (a)

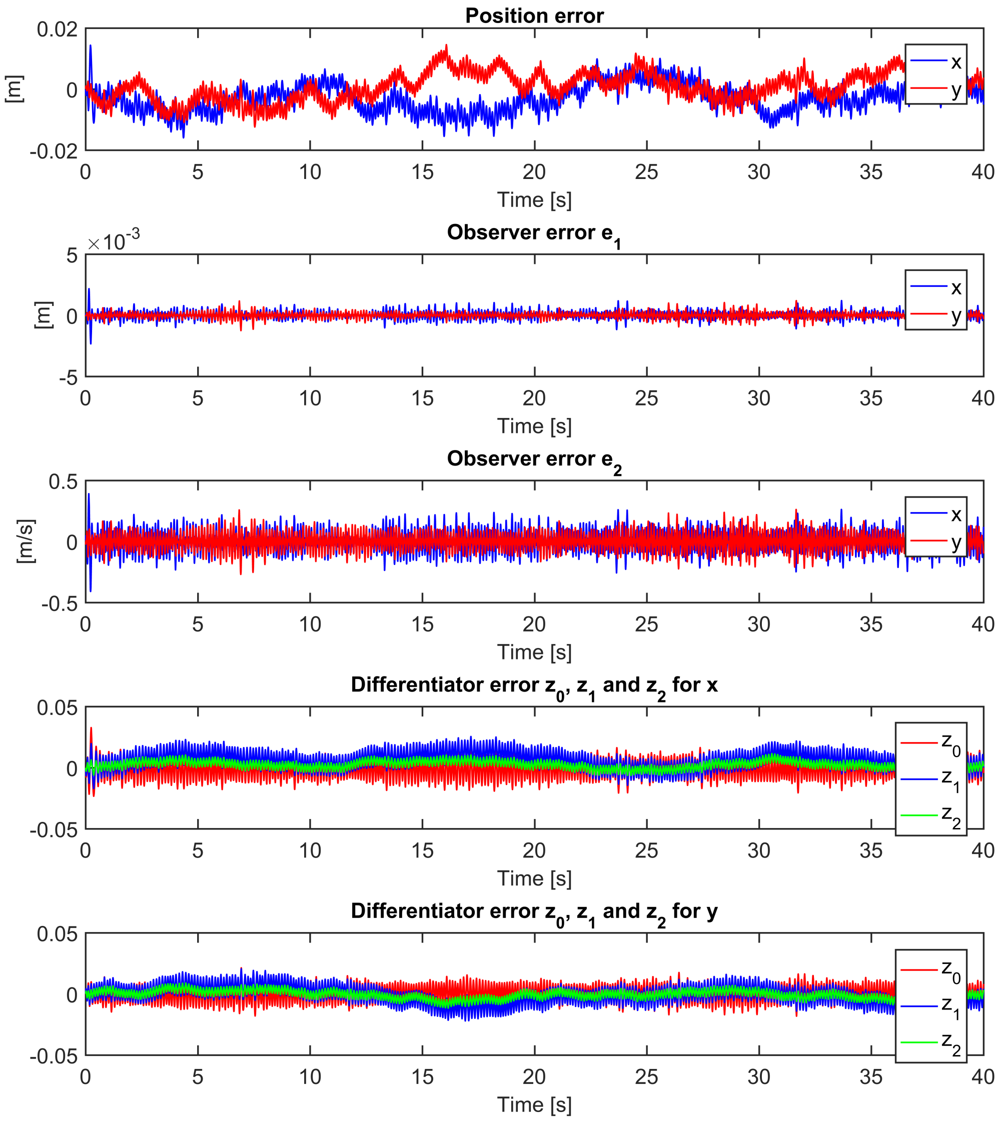


Figure 5.35: USM operation mode: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 2 (b)

Sliding surface 3: the estimated value for p_{CM} and $\dot{p}$ are used

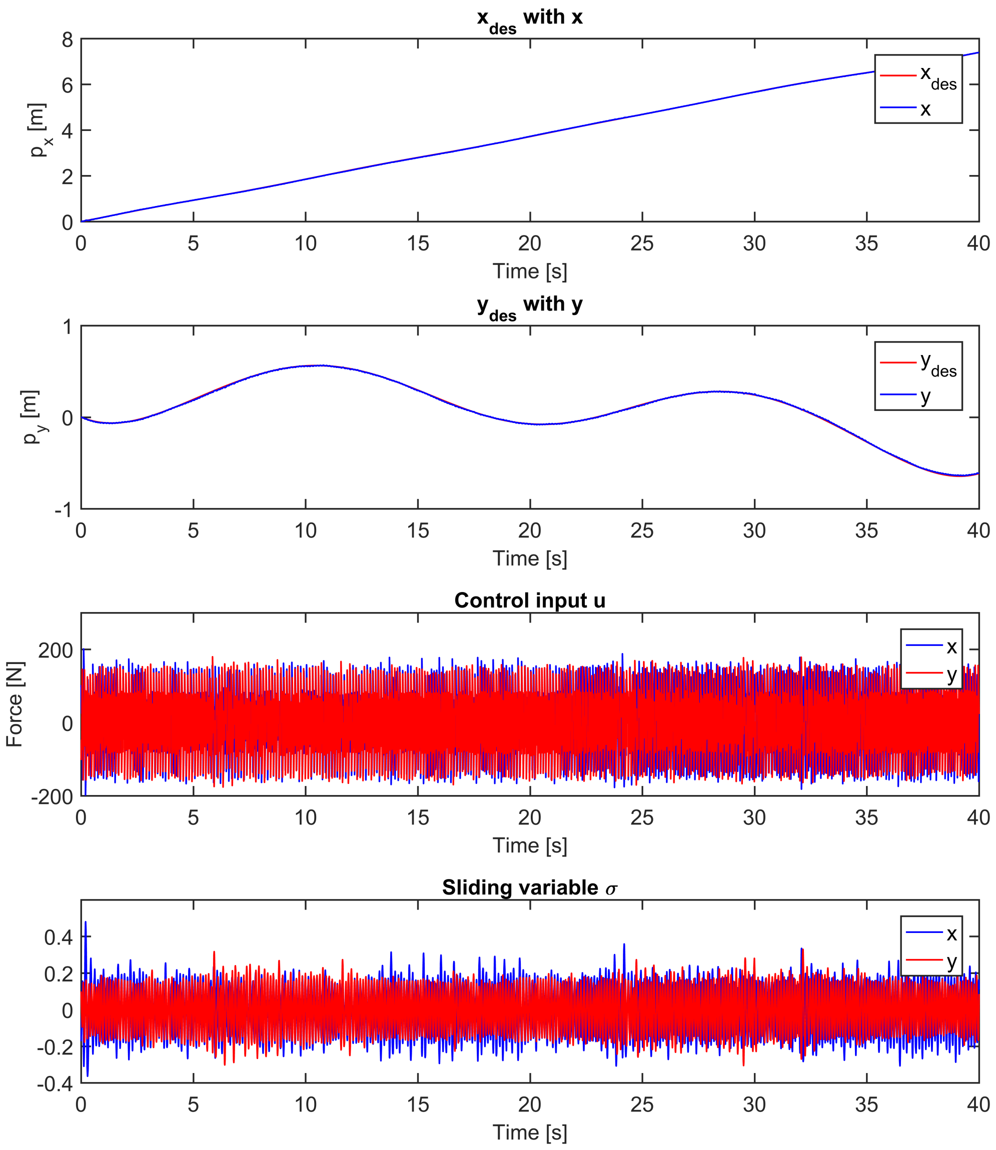


Figure 5.36: USM operation mode: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 3 (a)

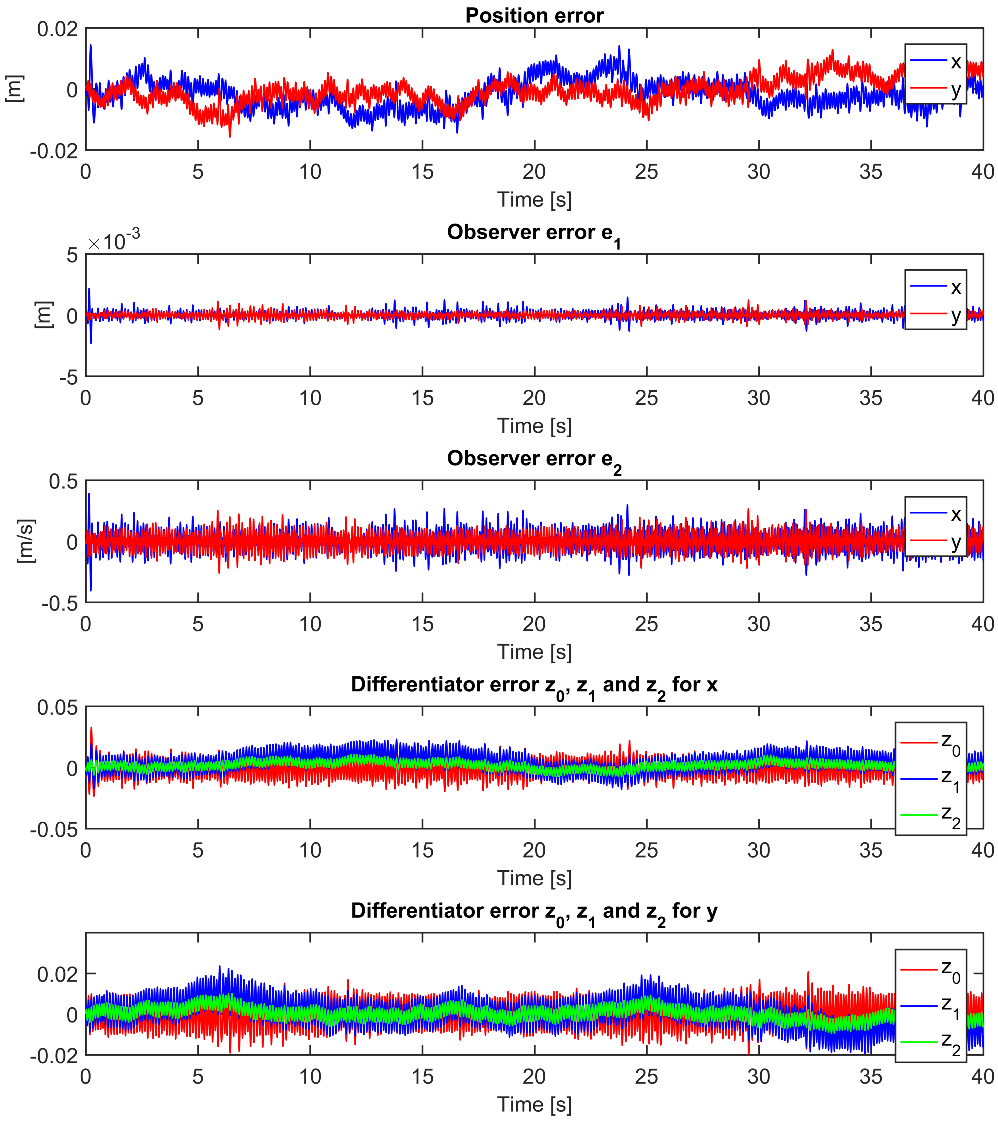


Figure 5.37: USM operation mode: Simulation of nested third-order SMC with differentiator and state observer, sliding surface 3 (b)

Quasi-continuous third-order sliding mode control

The gains were set to: $\alpha = 10$, $L_D = 1$ and $L_{\mathrm{SO}} = 50$.
Sliding surface 1: the actual value of p_{CM} and $\dot{p}$ are used

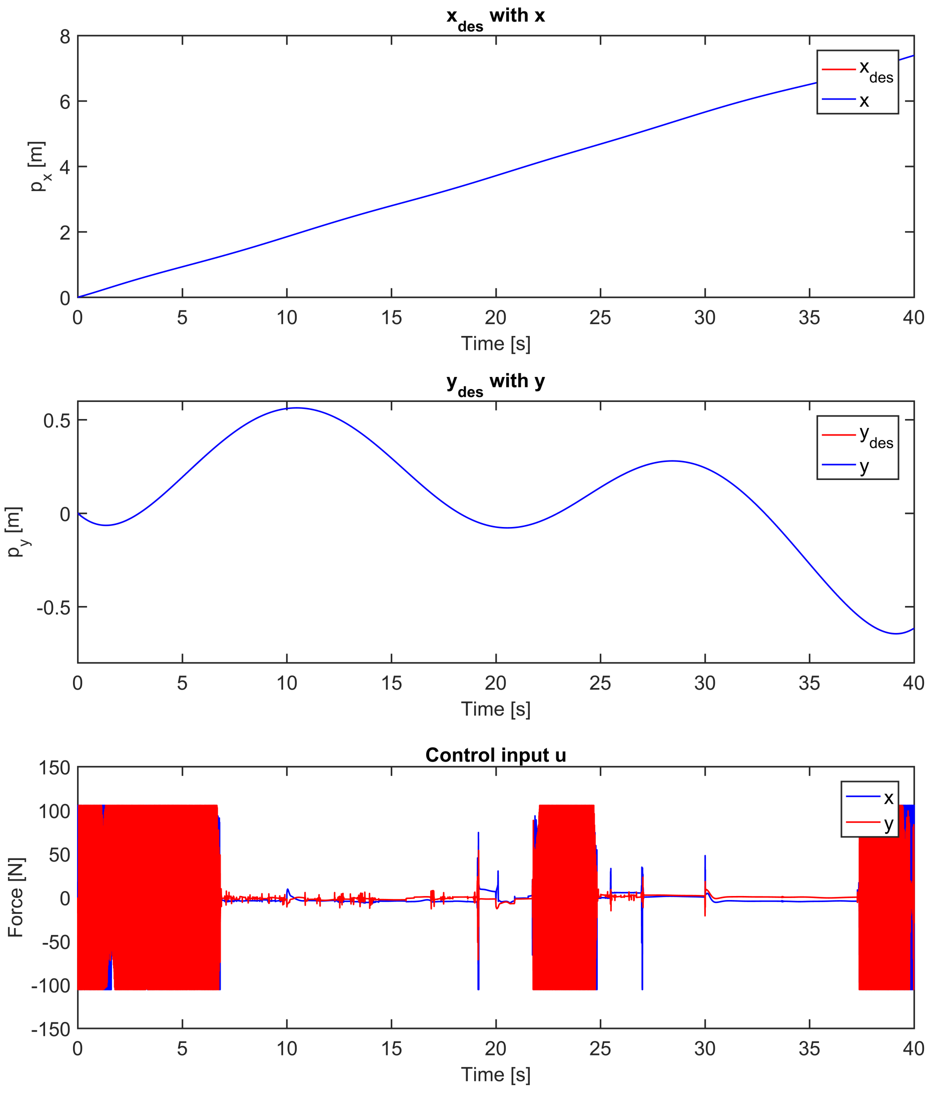


Figure 5.38: USM operation mode: Simulation of quasi-continuous third-order SMC with differentiator, sliding surface 1 (a)

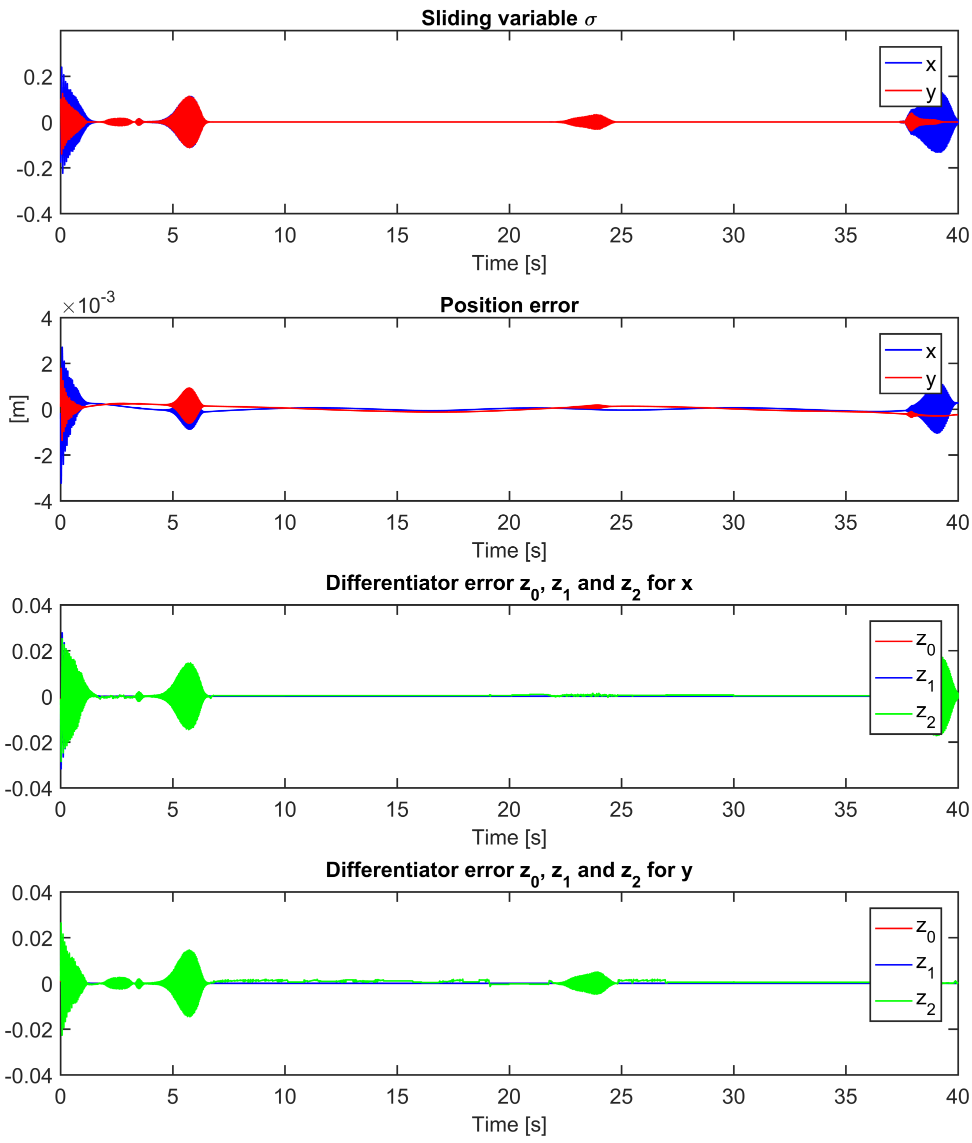


Figure 5.39: USM operation mode: Simulation of quasi-continuous third-order SMC with differentiator, sliding surface 1 (b)

Sliding surface 2: the estimated value for $\dot{p}$ is used

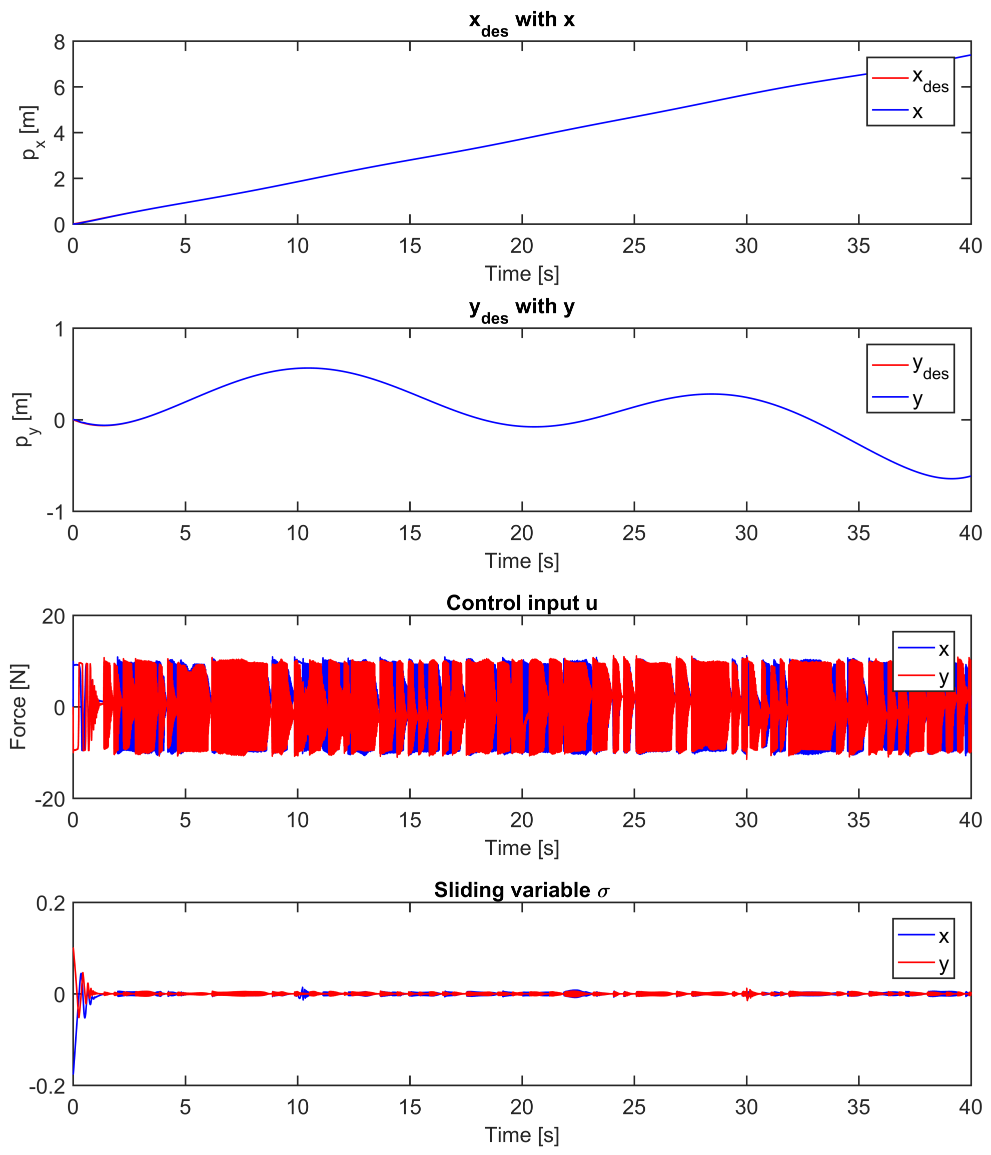


Figure 5.40: USM operation mode: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 2 (a)

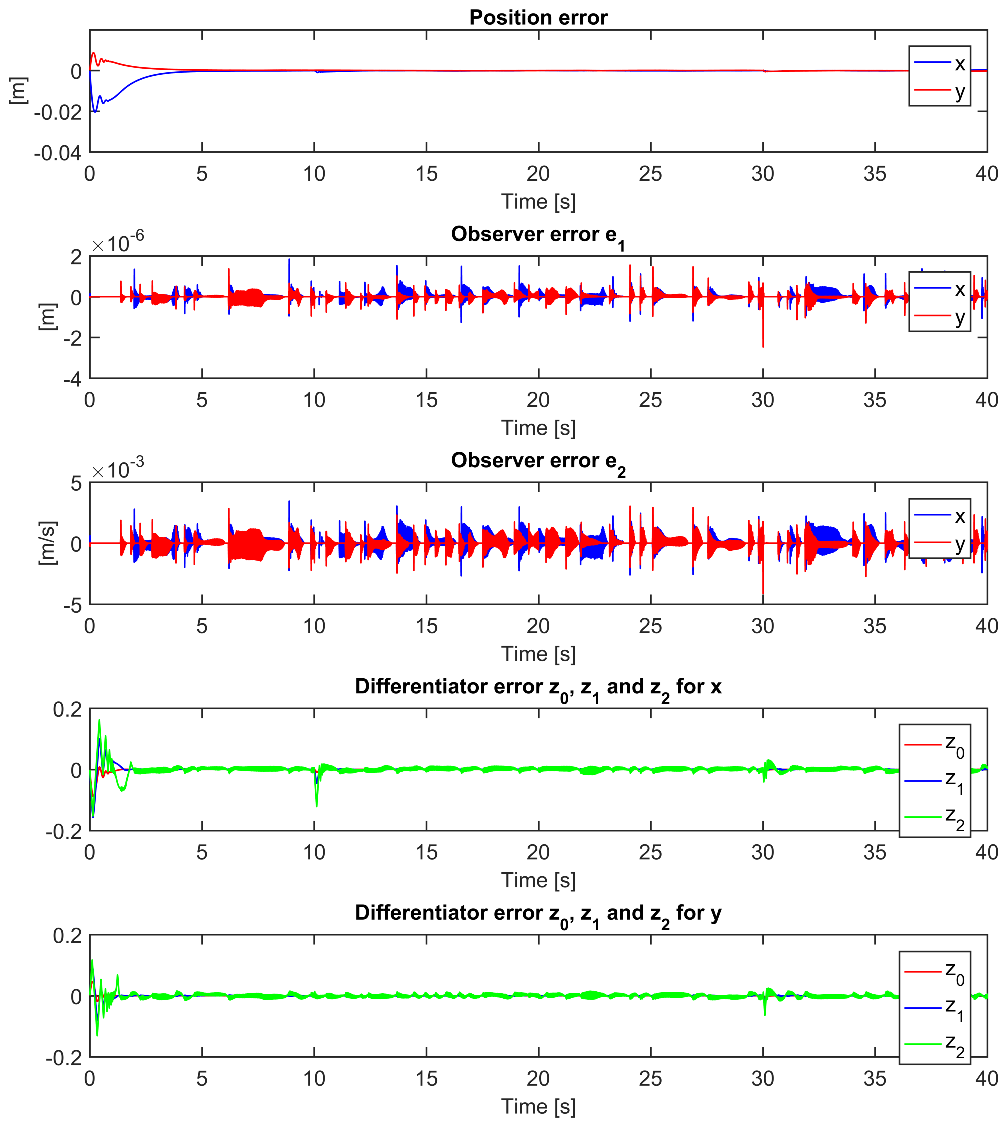


Figure 5.41: USM operation mode: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 2 (b)

Sliding surface 3: the estimated value for p_{CM} and $\dot{p}$ are used

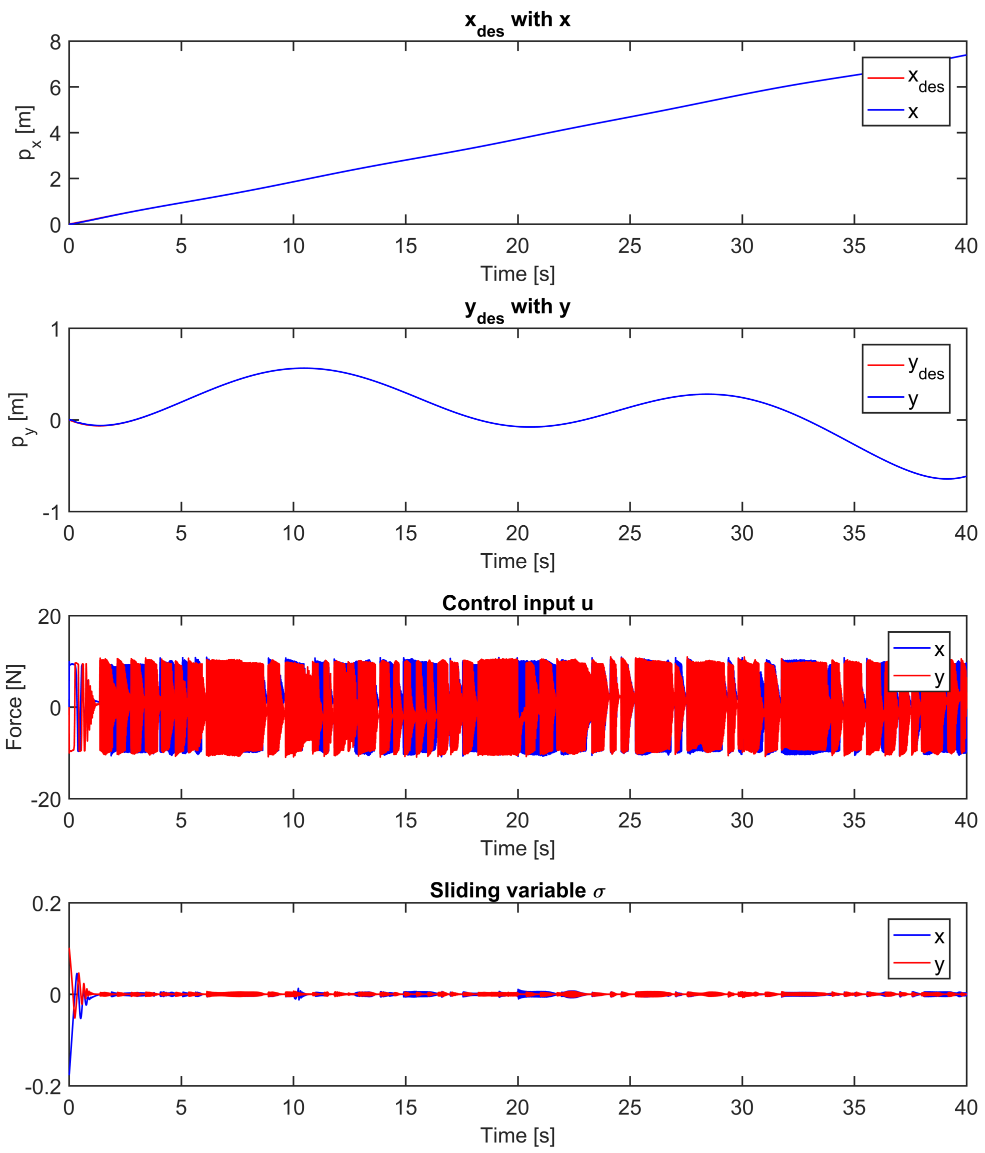


Figure 5.42: USM operation mode: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 3 (a)

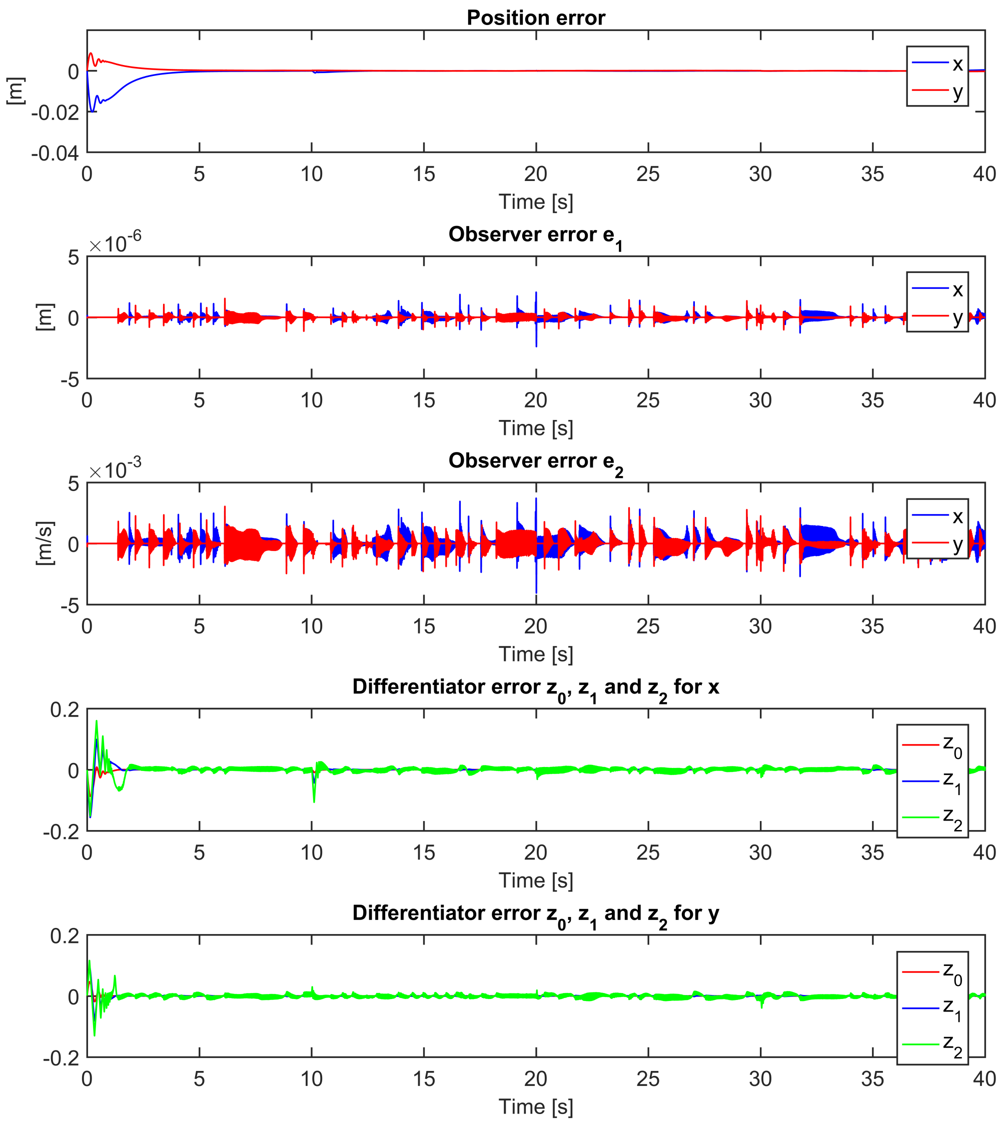


Figure 5.43: USM operation mode: Simulation of quasi-continuous third-order SMC with differentiator and state observer, sliding surface 3 (b)

The PD-controller

The gains were set to: $k_d^{CM} = 10$, $k_p^{CM} = 0.1$ and $L_{SO} = 50$.
Actual value of p_{CM} and $\dot{p}$ are used

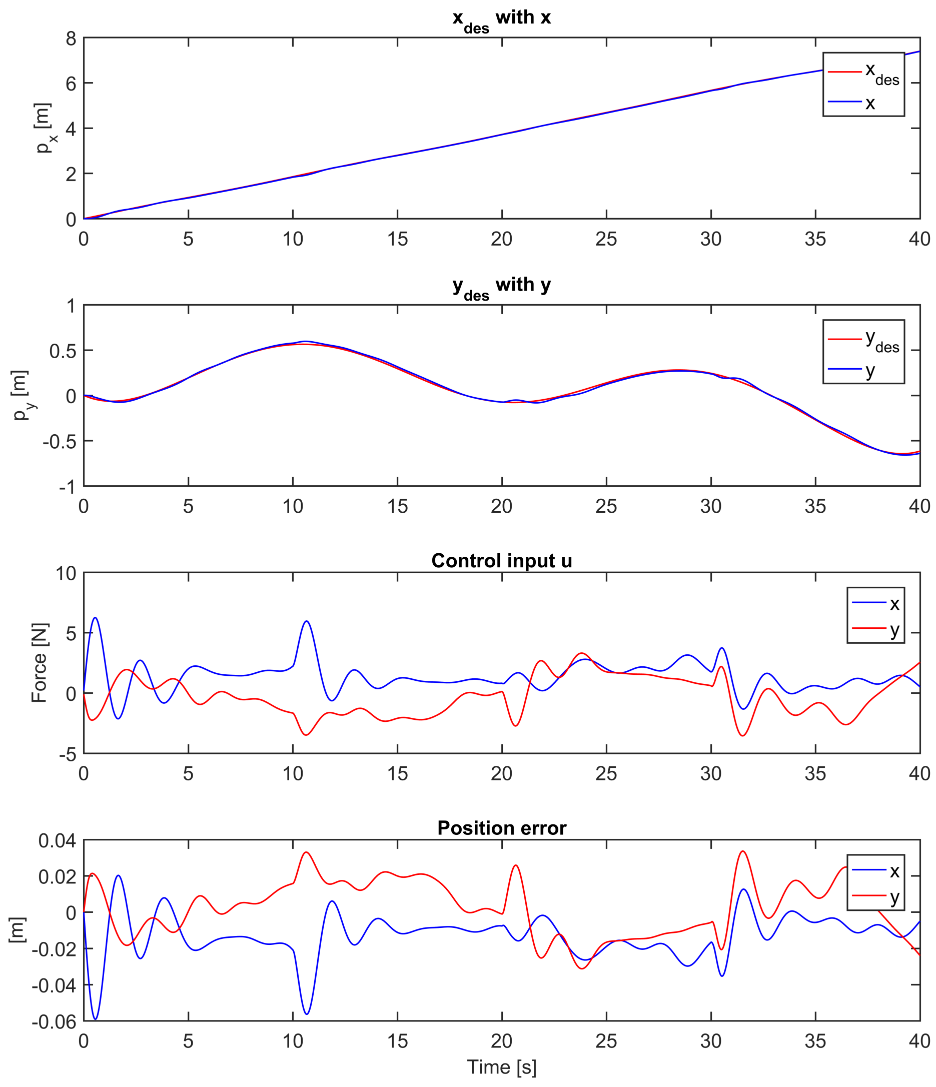


Figure 5.44: USM operation mode: Simulations of PD-controller when the actual value of p_{CM} and $\dot{p}$ are used

Estimated value for $\dot{p}$ is used

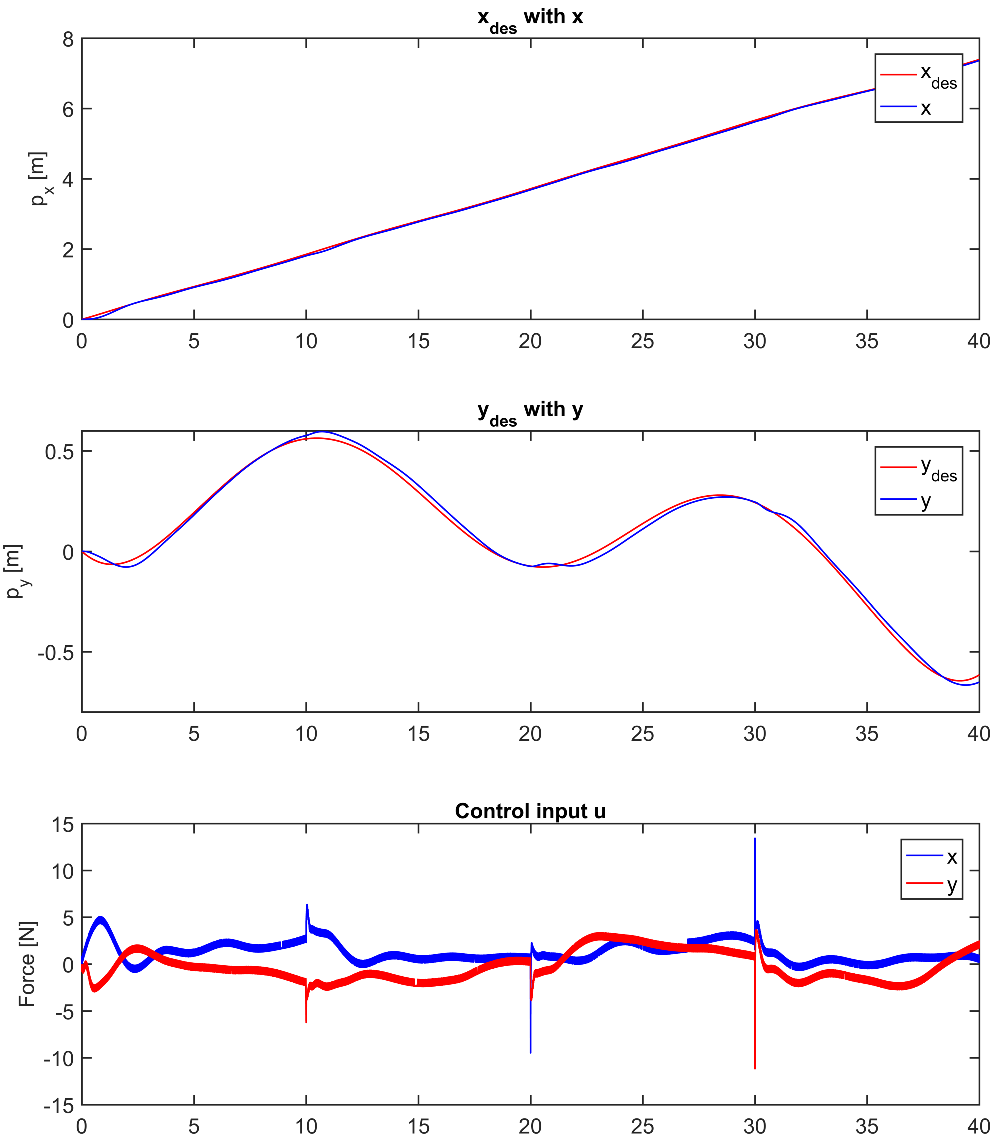


Figure 5.45: USM operation mode: Simulations of PD-controller when the estimated value of $\dot{p}$ is used (a)

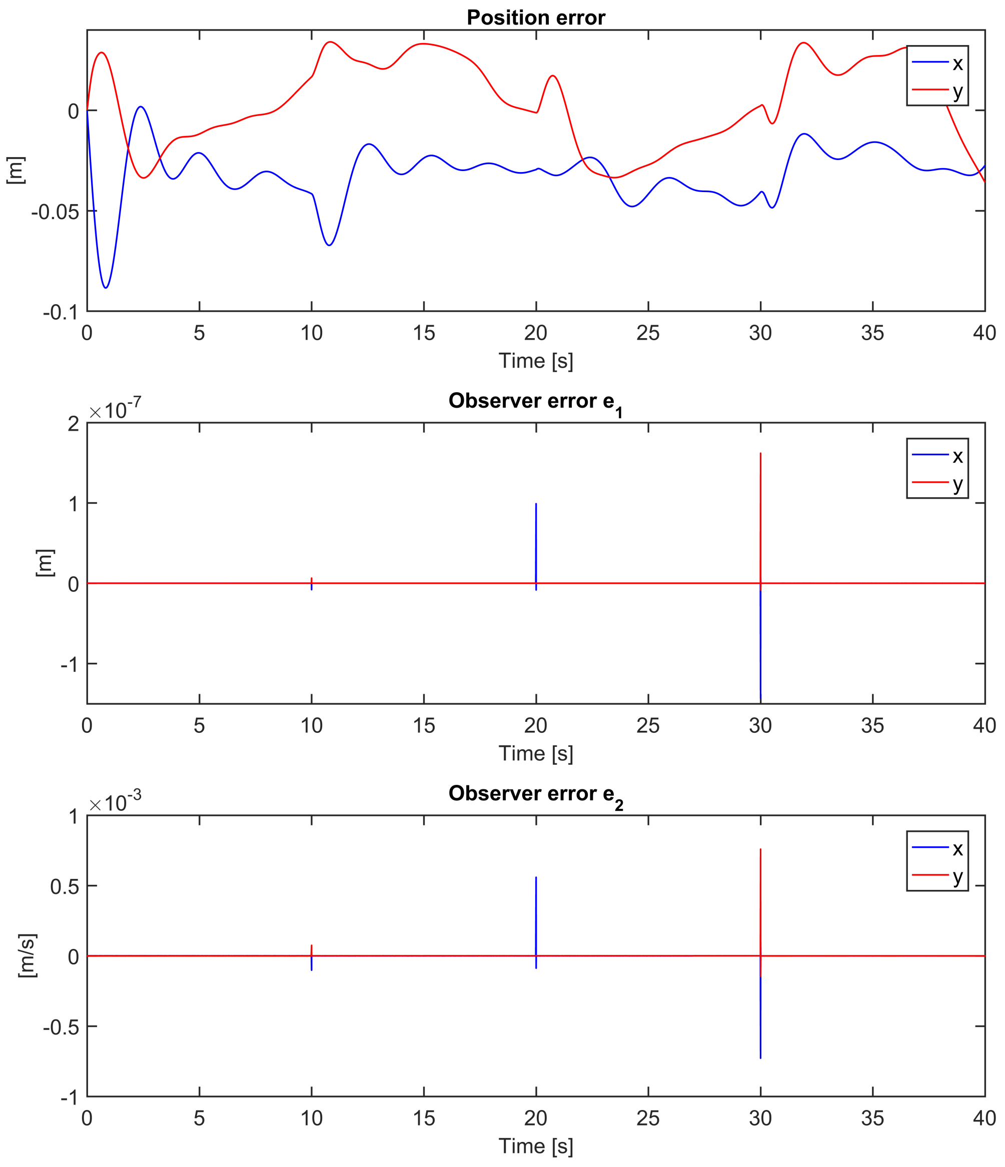


Figure 5.46: USM operation mode: Simulations of PD-controller when the estimated value of $\dot{p}$ is used (b)

Estimated value for p_{CM} and $\dot{p}$ are used

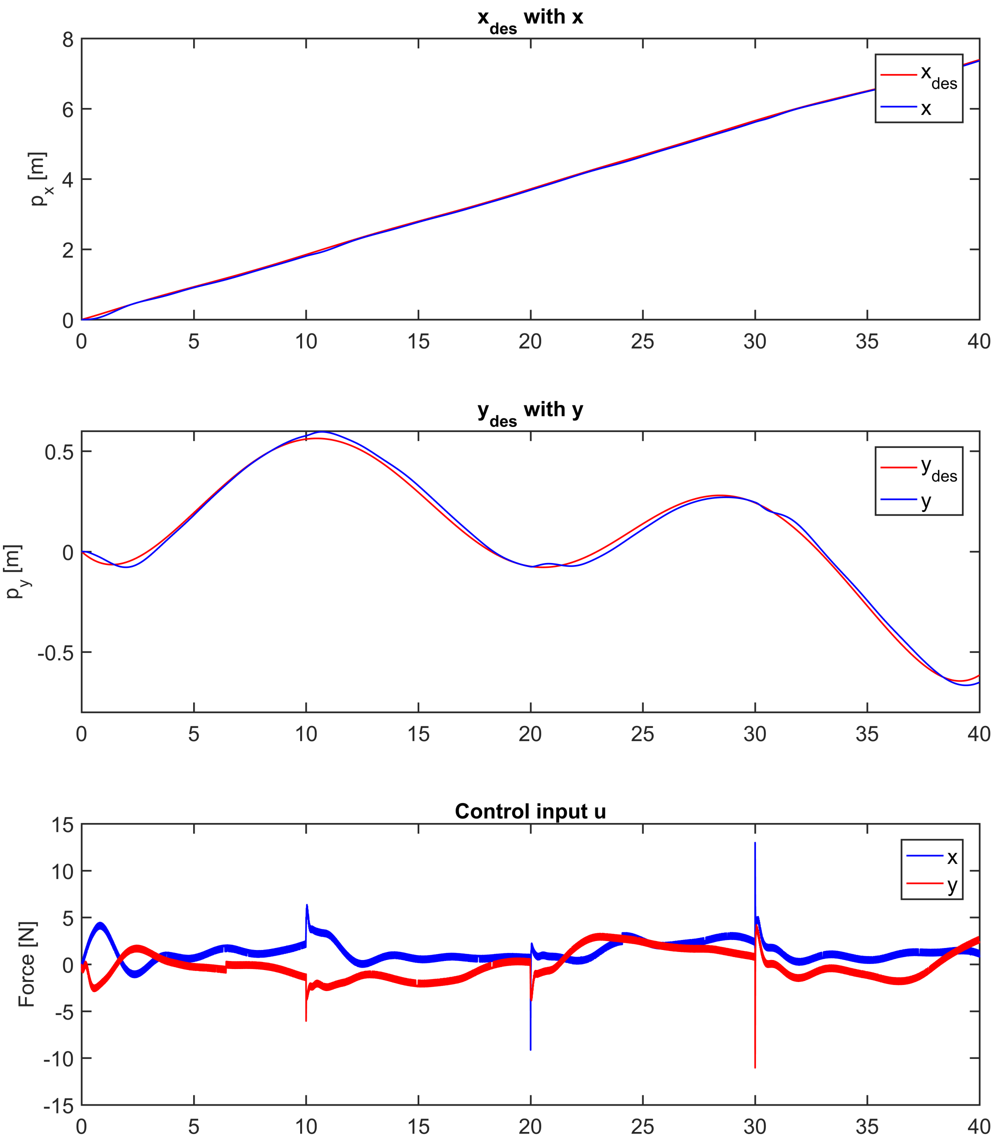


Figure 5.47: USM operation mode: Simulations of PD-controller when the estimated value of p_{CM} and $\dot{p}$ are used (a)

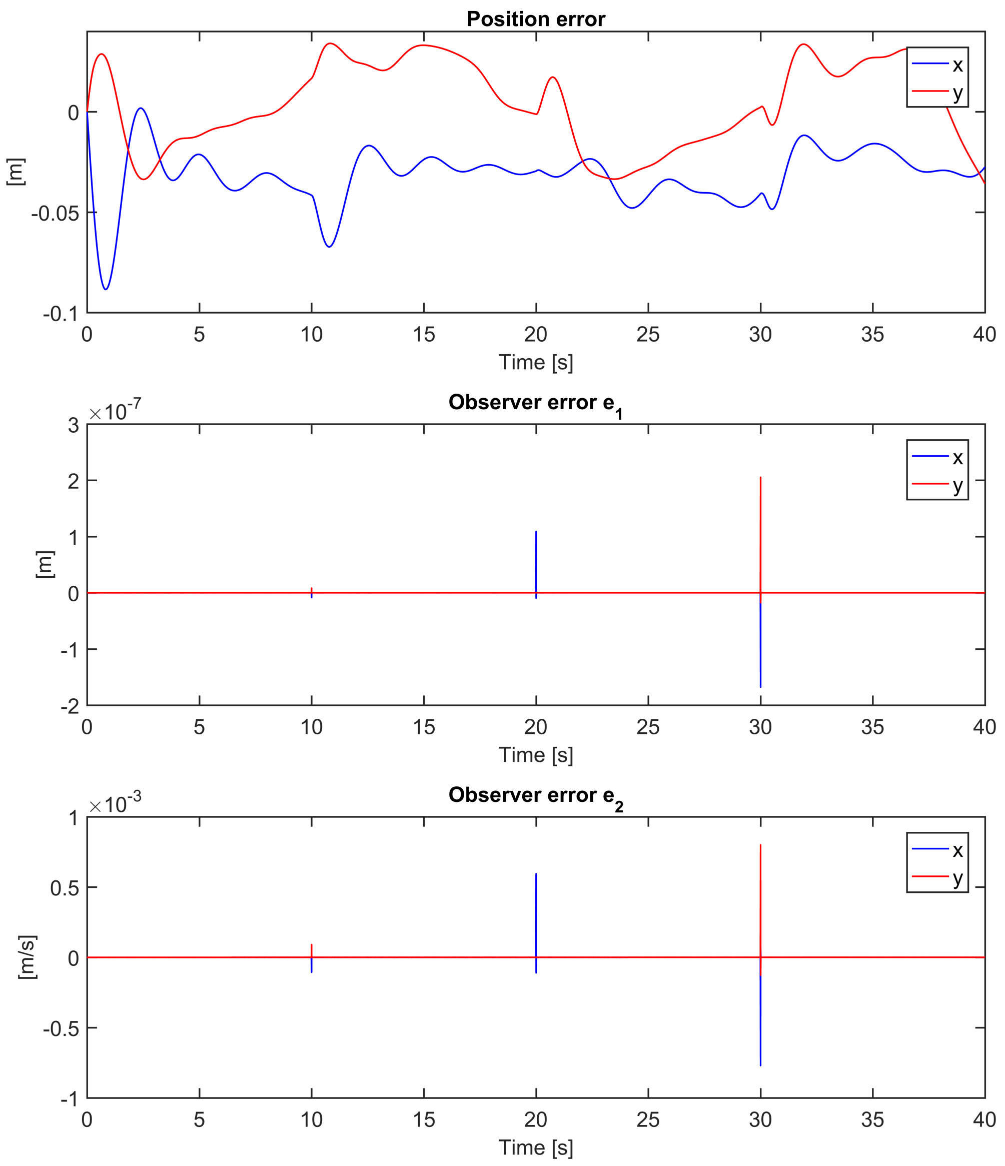


Figure 5.48: USM operation mode: Simulations of PD-controller when the estimated value of p_{CM} and $\dot{p}$ are used (b)

In order to better compare the different algorithms the absolute maximum value for the error variables for each algorithm have been gather. The position error can be found in Table 5.1, 5.2 and 5.3, the state observer error can be found in Table 5.4, 5.5, 5.6 and 5.7 and the differentiator error can be found in Table 5.8, 5.9, 5.10, 5.11, 5.12 and 5.13. Here the first 15 seconds were not considered so that the absolute maximum value for the error variables were found when the control input had stabilized.

Table 5.1: USM: absolute maximum value for position error, sliding surface 1

Algorithm	**Position error**			
	Torpedo mode		Operation mode	
	x	y	x	y
The super-twisting algorithm	$3.6131 \cdot 10^{-4}$	$2.8766 \cdot 10^{-4}$	$3.6130 \cdot 10^{-4}$	$5.5242 \cdot 10^{-4}$
Nested 3rd-order SMC	$3.6126 \cdot 10^{-4}$	$2.8769 \cdot 10^{-4}$	$9.8822 \cdot 10^{-4}$	$9.8006 \cdot 10^{-4}$
Quasi-continuous 3rd-order SMC	$3.6144 \cdot 10^{-4}$	$2.8763 \cdot 10^{-4}$	0.0012	$3.5804 \cdot 10^{-4}$
PD-controller	0.0107	0.0217	0.0354	0.0336

Table 5.2: USM: absolute maximum value for position error, sliding surface 2

Algorithm	**Position error**			
	Torpedo mode		Operation mode	
	x	y	x	y
The super-twisting algorithm	$3.6129 \cdot 10^{-4}$	$2.8770 \cdot 10^{-4}$	$3.6127 \cdot 10^{-4}$	$4.5509 \cdot 10^{-4}$
Nested 3rd-order SMC	0.0217	0.0254	0.0155	0.0146
Quasi-continuous 3rd-order SMC	$3.6398 \cdot 10^{-4}$	$3.0753 \cdot 10^{-4}$	$4.2522 \cdot 10^{-4}$	$5.8820 \cdot 10^{-4}$
PD-controller	0.0321	0.0354	0.0486	0.0359

Table 5.3: USM: absolute maximum value for position error, sliding surface 3

Algorithm	**Position error**			
	Torpedo mode		Operation mode	
	x	y	x	y
The super-twisting algorithm	$3.6129 \cdot 10^{-4}$	$2.8770 \cdot 10^{-4}$	$3.6128 \cdot 10^{-4}$	$4.5499 \cdot 10^{-4}$
Nested 3rd-order SMC	0.0245	0.0248	0.0144	0.0128
Quasi-continuous 3rd-order SMC	$3.8723 \cdot 10^{-4}$	$2.7268 \cdot 10^{-4}$	$3.8441 \cdot 10^{-4}$	$3.3079 \cdot 10^{-4}$
PD-controller	0.0321	0.0354	0.0486	0.0359

Table 5.4: USM torpedo mode: absolute maximum value for state observer error, sliding surface 2

Algorithm	**State observer error, torpedo mode**			
	e_1		e_2	
	x	y	x	y
The super-twisting algorithm	$3.8267 \cdot 10^{-12}$	$5.0679 \cdot 10^{-12}$	$7.5145 \cdot 10^{-7}$	$6.9141 \cdot 10^{-7}$
Nested 3rd-order SMC	0.0027	$4.3933 \cdot 10^{-4}$	0.4514	0.1335
Quasi-continuous 3rd-order SMC	$2.7061 \cdot 10^{-6}$	$8.3924 \cdot 10^{-7}$	0.0044	0.0020
PD-controller	$3.3542 \cdot 10^{-12}$	$4.1889 \cdot 10^{-12}$	$7.4797 \cdot 10^{-7}$	$7.0394 \cdot 10^{-7}$

Table 5.5: USM torpedo mode: absolute maximum value for state observer error, sliding surface 3

Algorithm	**State observer error, torpedo mode**			
	e_1		e_2	
	x	y	x	y
The super-twisting algorithm	$3.8183 \cdot 10^{-12}$	$5.0305 \cdot 10^{-12}$	$7.4071 \cdot 10^{-7}$	$6.9252 \cdot 10^{-7}$
Nested 3rd-order SMC	0.0033	$3.6455 \cdot 10^{-4}$	0.5128	0.1179
Quasi-continuous 3rd-order SMC	$2.4727 \cdot 10^{-6}$	$8.5851 \cdot 10^{-7}$	0.0042	0.0021
PD-controller	$3.3773 \cdot 10^{-12}$	$4.8455 \cdot 10^{-12}$	$7.3987 \cdot 10^{-7}$	$7.0306 \cdot 10^{-7}$

Table 5.6: USM operation mode: absolute maximum value for state observer error, sliding surface 2

Algorithm	**State observer error, operation mode**			
	e_1		e_2	
	x	y	x	y
The super-twisting algorithm	$1.3525 \cdot 10^{-7}$	$1.3044 \cdot 10^{-7}$	$6.8605 \cdot 10^{-4}$	$7.1110 \cdot 10^{-4}$
Nested 3rd-order SMC	0.0013	0.0012	0.2778	0.2618
Quasi-continuous 3rd-order SMC	$1.5046 \cdot 10^{-6}$	$2.4677 \cdot 10^{-6}$	0.0030	0.0041
PD-controller	$1.4334 \cdot 10^{-7}$	$1.6189 \cdot 10^{-7}$	$7.2829 \cdot 10^{-4}$	$7.5891 \cdot 10^{-4}$

Table 5.7: USM operation mode: absolute maximum value for state observer error, sliding surface 3

Algorithm	**State observer error, operation mode**			
	e_1		e_2	
	x	y	x	y
The super-twisting algorithm	$1.4877 \cdot 10^{-7}$	$1.6572 \cdot 10^{-7}$	$7.3004 \cdot 10^{-4}$	$7.5646 \cdot 10^{-4}$
Nested 3rd-order SMC	0.0015	0.0012	0.2986	0.2648
Quasi-continuous 3rd-order SMC	$2.3978 \cdot 10^{-6}$	$1.4249 \cdot 10^{-6}$	0.0041	0.0029
PD-controller	$1.6783 \cdot 10^{-7}$	$2.0535 \cdot 10^{-7}$	$7.6944 \cdot 10^{-4}$	$7.9966 \cdot 10^{-4}$

Table 5.8: USM torpedo mode: absolute maximum value for differentiator error, sliding surface 1

Differentiator error, torpedo mode	**Algorithm**			
	Nested 3rd-order SMC		Quasi-continuous 3rd-order SMC	
	x	y	x	y
z_0	$3.6045 \cdot 10^{-6}$	$2.9539 \cdot 10^{-6}$	$6.2474 \cdot 10^{-8}$	$1.6923 \cdot 10^{-8}$
z_1	$2.3498 \cdot 10^{-5}$	$1.7452 \cdot 10^{-5}$	$1.3899 \cdot 10^{-5}$	$6.0308 \cdot 10^{-6}$
z_2	$7.4344 \cdot 10^{-5}$	$1.0770 \cdot 10^{-4}$	$6.5584 \cdot 10^{-4}$	$4.0530 \cdot 10^{-4}$

Table 5.9: USM torpedo mode: absolute maximum value for differentiator error, sliding surface 2

Algorithm	**Differentiator error, torpedo mode**					
	z_0		z_1		z_2	
	x	y	x	y	x	y
Nested 3rd-order SMC	0.0360	0.0138	0.0295	0.0353	0.0122	0.0167
Quasi-continuous 3rd-order SMC	$8.3460 \cdot 10^{-4}$	$3.8488 \cdot 10^{-4}$	0.0039	0.0028	0.0177	0.0147

Table 5.10: USM torpedo mode: absolute maximum value for differentiator error, sliding surface 3

Algorithm	Differentiator error, torpedo mode					
	z_0		z_1		z_2	
	x	y	x	y	x	y
Nested 3rd-order SMC	0.0338	0.0137	0.0304	0.0346	0.0121	0.0172
Quasi-continuous 3rd-order SMC	$7.9601 \cdot 10^{-4}$	$3.8537 \cdot 10^{-4}$	0.0038	0.0028	0.0176	0.0128

Table 5.11: USM operation mode: absolute maximum value for differentiator error, sliding surface 1

Algorithm	Differentiator error, operation mode					
	z_0		z_1		z_2	
	x	y	x	y	x	y
Nested 3rd-order SMC	0.0070	0.0066	0.0157	0.0142	0.0181	0.0159
Quasi-continuous 3rd-order SMC	0.0079	0.0013	0.0164	0.0042	0.0181	0.0068

Table 5.12: USM operation mode: absolute maximum value for differentiator error, sliding surface 2

Algorithm	Differentiator error, operation mode					
	z_0		z_1		z_2	
	x	y	x	y	x	y
Nested 3rd-order SMC	0.0207	0.0181	0.0254	0.0220	0.0115	0.0115
Quasi-continuous 3rd-order SMC	0.0035	0.0062	0.0162	0.0272	0.0408	0.0641

Table 5.13: USM operation mode: absolute maximum value for differentiator error, sliding surface 3

Algorithm	Differentiator error, operation mode					
	z_0		z_1		z_2	
	x	y	x	y	x	y
Nested 3rd-order SMC	0.0220	0.0207	0.0218	0.0192	0.0100	0.0095
Quasi-continuous 3rd-order SMC	0.0018	0.0018	0.0100	0.0117	0.0315	0.0394

Chapter 6

Discussion

This chapter will present a discussion regarding the stability analysis and the three algorithms that have been tested on two different systems. The algorithms will be compared against each other and to a regular PD-controller. The affects of using a state observer will be discussed by comparing the results achieved with the different sliding surfaces. A simulation issue that caused some problems will also be discussed.

6.1 Stability

Stability for the error dynamics for a second-order general system was proven in Chapter 3. In Section 4.3 it was shown that the error dynamics for the mass-spring-damper system fits the assumptions made for the error dynamics for the general system, which means that the theorems in Chapter 3 also hold for the error dynamics for the mass-spring-damper system. It was shown in Section 5.3, that the error dynamics for the USM also fits the assumptions made for the error dynamics for the general system, which means that the theorems in Chapter 3 also hold for the error dynamics for the USM.

The stability analysis done for the STA, can be extended to the STA with adaptive gains, by using the Lyapunov function proposed in Shtessel et al. (2010) or Shtessel et al. (2012). This has not been done due to lack of time, and will therefore be suggested as further work.

The stability proof of the HOSM controller when the estimated value of x_2 was used in the sliding surface, Section 3.3.1, was not completed. This is because no Lyapunov function exist for the HOSM algorithms. In Cruz-Zavala and Moreno (2017) a very interesting new HOSM controller with explanations on how to create a Lyapunov function for the controller is proposed. This can be a HOSM controller that can be proven stable in both cases, when the estimated value of x_2 is used in the sliding surface and when the estimated value of both x_1 and x_2 is used in the sliding surface. Due to lack of time it was not possible to try to prove stability with this controller or to test it. This will therefore be a topic for further investigation.

6.2 The super-twisting algorithm

The super-twisting algorithm gave very good results, both for the mass-spring damper system and for the USM. The state trajectories follows the desired path almost perfectly, and it has a very small error in all cases. This means it handles disturbances and modelling errors very well. The control input for the mass-spring-damper system in all cases is smooth and there is no sign of chattering. From Figures 4.18, 4.19, 4.20, 4.21 and 4.22, one can see that there is nearly no difference between the three sliding surfaces. Tables 4.1 and 4.2 confirm this, as the position error and the observer errors are nearly the same for all the sliding surfaces. From the figures and tables mentioned, it can be seen that the state observer does not affect the control abilities or performance of the algorithm noticeably, as the position errors and the control inputs are very similar.

The control input for the USM when it is in torpedo mode has some differences from one sliding surface to another. When the state observer is not included it has a completely smooth control input, which can be seen from Figure 5.5. In the cases where the state observer is used, the control input does have some chattering. This can be seen from the very thick line in Figures 5.6 and 5.8. This is most likely coming from the chattering in the observer errors, as this is not a problem for the case without the state observer. It was also not a problem for the mass-spring-damper system when the state observer was used, because there the observer errors were smoother. This can be seen from Figures 6.1 and 6.2, where the observer errors from the simulation of the mass-spring-damper system and the USM torpedo mode have been zoomed in on. In the cases where the state observer is used, there is nearly no difference between the two sliding surfaces. Tables 5.2, 5.3, 5.4 and 5.5 confirm this.

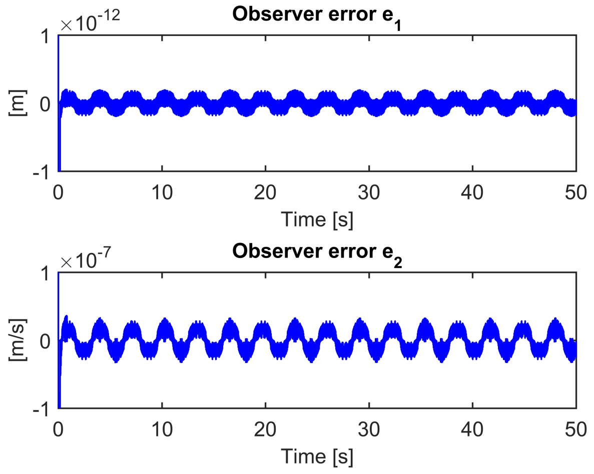


Figure 6.1: MSDS: Zoom in on observer errors from Figure 4.20

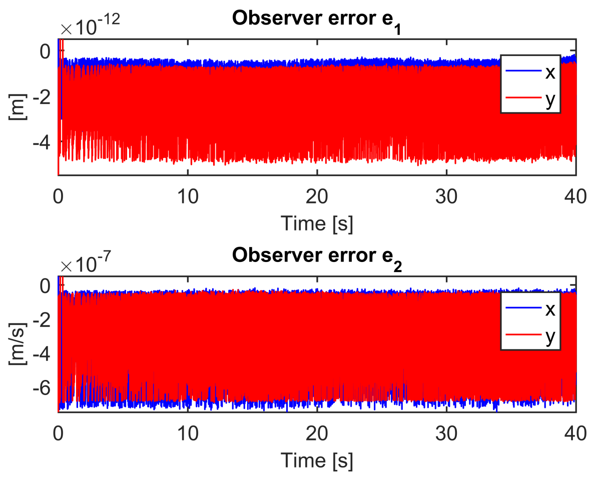


Figure 6.2: USM torpedo mode: Zoom in on observer errors from Figure 5.7

In the case where the USM is in operation mode, the control input is more intense and varying. The reason being, when the USM shifts position the controller reacts fast, causing peaks in the control input at 10, 20 and 30 seconds (Figures 5.27, 5.28 and 5.30). This means that the super-twisting controller can handle a great deal of disturbance, but it will affect the smoothness of the control input. These peaks can also be found in the position errors and the observer errors. For the position errors this can be seen from Figure 6.3, and for the observer errors this can be seen from Figures 5.29 and 5.31. This means that the errors are not as large as in Tables 5.1, 5.2, 5.3, 5.6 and 5.7 all the time, it is only that large when the USM shifts position. The error the rest of the time, is more like the errors for the torpedo mode case. Here there are also some issues with chattering in the cases when the state observer is used, that is for the same reasons as for the torpedo mode case. In operation mode, there is also nearly no difference in the two sliding surfaces when the state observer was used, the tables mentioned confirm that. The chattering in the control input for the USM when the state observer is used can probably be reduced by using a filter on the estimated states.

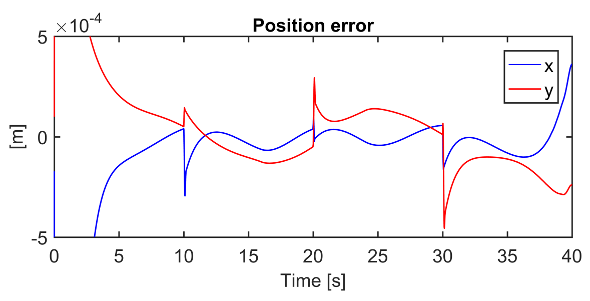


Figure 6.3: USM operation mode: Zoom in on position error from Figure 5.27

The problem with this controller is that a bound on the disturbance needs to be known to be able to find the optimal gain, which can be difficult when the disturbance is unknown. This often leads to a very crude estimation of what the gain needs to be which leads to a much higher gain than the optimal one. This can however be avoided by using the STA with adaptive gains, which was tested in my project. It has the same control abilities as the super-twisting algorithm with constant gain, the only difference is that the gains adapt, i.e. find the optimal gain, without knowing the bound on the disturbance. The gains in the STA with adaptive gains decides how fast the rate of convergence is, Borlaug (2016).

6.3 Nested third-order sliding mode control

The nested third-order sliding mode control algorithm gave very good tracking control for the mass-spring-damper system. The state trajectory follows the desired path almost perfectly, and it has a very small error. This means that in this case it handles the disturbance very well. The control input is however filled with chattering. In Levant (2001) it is said that if the parameters are chosen correctly, chattering will not appear. Since the only difference between the nested third-order SMC controller and the relay controller is that which is inside the sgn, chattering is unlikely to be removed completely, as the relay controller has large problems with chattering. Now, if there does exist some combination of variables that removes the chattering effect, they are certainly difficult to find. Also for this controller there is nearly no difference between the sliding surfaces. This can be seen from Figures 4.23, 4.24, 4.25, 4.26, 4.27 and 4.28. In Tables 4.1 and 4.2 one can however see that there is a small difference for the position error and the observer errors. The errors are largest when both the estimated value of x_1 and x_2 are used in the sliding surface. This is probably due to that when the observer errors are combined from both estimated states it gives a larger error in the sliding surface, causing larger errors in the differentiator, which leads to a larger position error. This is also confirmed by Tables 4.3, 4.4 and 4.5 as the differentiator errors are largest when the position error and observer errors are largest.

For the USM the tracking control was only good when the state observer was not used, this can be seen from Tables 5.1, 5.2 and 5.3. This is probably because the observer errors were quite large, as can be seen from Tables 5.4, 5.5, 5.6 and 5.7. The observer errors are also filled with severe chattering, shown in Figures 5.13, 5.15, 5.35 and 5.37. This is probably why the observer errors are so large. It might also be the reason why there is so much chattering in the differentiator error, as for the case when the state observer was not used the differentiator errors were smooth in comparison. This can be seen from Figures 5.11, 5.13, 5.15, 5.33, 5.35 and 5.37. It can also be seen from Tables 5.8, 5.9, 5.10, 5.11, 5.12 and 5.13 that the differentiator errors are much larger when the state observer is used. This supports the theory that the reason for the large position errors in the cases when the state observer is used, are the large errors in the observer errors. The reason for the severe chattering in the observer errors are difficult to find. Choice of parameters can be one reason, as it was not possible to test that many different parameters in the given time frame, because of a simulation issue, Section 6.8. This means that the algorithm with a state observer might be able to perform better with other gains than what were used here, but finding those gains can be difficult and time-consuming. The control input in both the torpedo mode case and in the operation mode case, were filled with chattering as it was for the mass-spring-damper system, this can be seen from Figures 5.10, 5.12, 5.14, 5.32, 5.34 and 5.36. This supports the theory made in the previous paragraph that it is not possible to find gains which gives a smooth control input.

6.4 Quasi-continuous third-order sliding mode control

The quasi-continuous third-order sliding mode control algorithm gave very good results, both for the mass-spring damper system and for the USM. The state trajectories follows the desired path almost perfectly, and it has a very small error. This means it handles disturbances and modelling errors very well. The control input for the mass-spring-damper system has some chattering, but there does exist gains that make the control input completely smooth, Figures 4.35, 4.37 and 4.39. There is however a trade-off between how smooth the control input is and how large the errors get. From Tables 4.1, 4.3, 4.4 and 4.5 it is apparent that the position error and the differentiator errors when the control input is smooth is much larger than for the case when there is some chattering in the control input. Also here there is nearly no difference between the sliding surfaces. This can be seen from Figures 4.29, 4.30, 4.31, 4.32, 4.33 and 4.34. In Tables 4.1 and 4.2 one can see that there is a small difference for the position error and the observer errors. The errors are largest when both the estimated value of x_1 and x_2 is used in the sliding surface. This is probably because the error combined from both estimated values gives a larger error in the sliding surface, that gives larger errors in the differentiator, which leads to a larger position error. This is also confirmed by Tables 4.3, 4.4 and 4.5 as the differentiator errors are largest when the position error and observer errors are largest.

From Tables 5.1, 5.2, 5.3, it is apparent that the algorithm gave very good tracking control in all cases for the USM, as the position error is small for all of them. The control input in both the torpedo mode case and in the operation mode case, were filled with chattering when the state observer was used, which can be seen from Figures 5.18, 5.20, 5.40 and 5.42. In the case when the state observer was not used, there was only chattering in the beginning for the torpedo mode case, which can be seen from Figure 5.16. In the operation mode case there was only chattering in the beginning and when the USM shifted position. This can be seen from Figure 5.38. This means that the chattering in the control input in the cases when the state observer is used, is most likely because of the state observer. It can also be seen from Figures 5.17, 5.19, 5.21, 5.39, 5.41 and 5.43 that there is more chattering in the differentiator errors when the state observer is used. This leads to larger errors in the differentiator in the torpedo mode case, which can be seen from Tables 5.8, 5.9, 5.10. In the operation mode case the differentiator errors are however nearly the same, which can be seen from Tables 5.11, 5.12 and 5.13. The reason for this might be because the error is nearly the same for all the sliding surfaces when the USM shifts position. This algorithm does however, handle these larger errors because they do not affect the position errors.

6.5 Comparison between the sliding mode control algorithms

All three algorithms managed to follow the desired path with small errors in all cases when the state observer was not used. In the cases where the state observer was used, the algorithms did very good for the mass-spring-damper system, but for the USM the nested third-order SMC algorithm had large errors compared to the other two algorithms. This was because the observer errors got very large and manifested through the system. The biggest difference between the algorithms is regarding the smoothness of the control input. Both the third-order SMC algorithms had chattering in the control input. The nested third-order SMC algorithm had chattering in the control input in all cases simulated, and if it is possible to remove the chattering phenomena it would be very difficult finding the gains to do so. The quasi-continuous third-order SMC algorithm was not as bad as the nested third-order SMC algorithm, as it did not have chattering in all cases. It was possible to find a choice of gains that did remove the chattering, but it will be a trade-off between how much error there will be in the system, and the smoothness of the control input. When the state observer was not used it also had a smooth control input, after some time, for the USM torpedo mode. In operation mode it had a smooth control input, except when the USM switched positions. The STA does however give a smooth control input in all cases, except for when the state observer is used for the USM. It then has some chattering, but it was much smaller than the chattering in the third-order algorithms and it was because of chattering in the estimated states. It is also easier to find controller gains that work, close to optimal for the STA. The problem with finding the gains for these three algorithms can be removed by using adaptive gains. There has been suggested some adaptive HOSM algorithms, as in Edwards and Shtessel (2016), but because of a limited amount of time they were not tested in this thesis. As mentioned in Section 6.2, the STA with adaptive gains has already been tested and has the same control abilities as the STA, but it also finds the optimal gains.

The simulation results from the USM in operation mode give stranger control inputs for the STA than the other controllers. This is because the third-order SMC algorithms have so much chattering that the infliction of the movement of the USM is not noticeable. But since the STA has almost a smooth control input from the beginning, the affects from the movement of the USM is very noticeable. One way of making the control input smoother in operation mode for the STA algorithm might be to either set a bound on how fast the controller is allowed to react, or to make the USM move slower as the changes from one direction to another are quite fast in these simulations.

In the simulations for the mass-spring-damper system, there is nearly no difference in the state observer errors between the different algorithms. The only thing worth noticing is that the error for the third-order algorithm is a bit larger, than for the STA when the estimated value for x_1 and x_2 are used in the sliding surface. This is probably also the reason the position errors for the third-order SMC algorithm are a little bit bigger when the estimated value for x_1 and x_2 are used in the sliding surface. The differentiator errors are nearly the same for the two third-order algorithms. In the simulations for the USM there are much larger differences between the algorithms. In both torpedo mode and operation mode the STA gives the smallest observer errors. There is quite a difference from torpedo mode to operation mode, but the STA still has the smallest observer errors. The quasi-continuous third-order SMC algorithm is the one that gives the second best result, and the nested third-order SMC algorithm gives the poorest result. This is reflected in the position error for the cases when the state observer is used. This is also reflected in the differentiator error for the third-order algorithms. The nested third-order SMC algorithm has the largest observer errors and the largest differentiator errors. This is also the case for when the state observer is not used, but for that there is no obvious reason. It might be the choice of gains.

One other thing that is worth noticing is that the quasi-continuous third-order SMC algorithm has a bit larger settling time than the other algorithms. This can be because of the chosen controller gains. To summarise the STA algorithm performs best in all cases, and does have the smallest errors. The quasi-continuous third-order SMC algorithm gives quite good results, but has some issues with chattering. The nested third-order SMC algorithm had chattering in all cases, even in the cases where the state observer was not used, and should therefore not be used in practice. Therefore, in the cases where it is not possible to use the STA, i.e. in the cases where the control input does not appear in the first derivative of the sliding surface, the quasi-continuous third-order SMC algorithm should be used.

6.6 Comparison between the PD-controller and sliding mode controllers

The PD-controller has much larger errors than the SMC algorithms in all simulation cases, when the state observer is not used. This can be seen from Table 5.1 and Figure 5.22. This means that the PD-controller does handle disturbances and modelling errors much more poorly than the SMC algorithms. When the state observer was used, the PD-controller did get a small increase in error, which can be seen from Figures 5.23, 5.25, 5.45, 5.47 and Tables 5.2 and 5.2. It then gave nearly the same results for the position error as the nested third-order SMC algorithm with a state observer. This does however say more about the poor behaviour of the nested third-order SMC algorithm with a state observer, than about the good behaviour of the PD-controller in this case. The STA and the quasi-continuous third-order SMC algorithm did however work much better than the PD-controller in all cases, i.e. with and without state observer. This means that these two SMC algorithms are much more suited for these types of systems and that they are more robust against disturbances. Most likely the nested third-order SMC algorithm would have also worked better if the state observer errors were smaller. The PD-controller gains were chosen by trial and error, same as the gains for the SMC algorithms, and should therefore give a fair comparison between the algorithms. Since the position error for the PD-controller is quite varying, there is no guarantee that a PID-controller would have given the SMC more competition regarding results.

6.7 Comparison of the sliding surfaces

The difference between the different sliding surfaces is not very large, especially for the mass-spring-damper system. Here the errors are a bit larger for the sliding surface where two estimated states are used. However, this is expected as the estimated states will never be exactly the same at the actual states and it will therefore inflict more error on the system than when one or no estimated states are used. In the USM simulations there is however more noticeable differences. The biggest difference is between no state observer and when a state observer is used. When the state observer is not used, the errors are noticeably smaller for the differentiator error in the USM torpedo mode case and for the position error in both cases, i.e. USM torpedo and operation mode, when the nested third-order SMC algorithm is used. The control input of the quasi-continuous third-order SMC algorithm is also much smoother. The reason for this might be that for the USM simulations there is much more chattering in the observer errors than for the mass-spring-damper system, Figures 6.1 and 6.2. This leads to larger errors when the estimated states are used, which then manifest through the system. This then gives a significant difference between when the state observer is used, and when it is not used. This is very obvious when the nested third-order SMC algorithm is used, as it has very large observer errors, which leads to large position errors for the cases when the state observer is used. To summarise, there is no larger difference than expected between using one or two estimated states in the sliding surface, but the state observer does in general have a large impact as the errors from the state observer does have an impact on the control abilities and performance of the algorithms. Some of that impact might be removed by using a filter on the estimated states before using them in the sliding surface, as the impact was largest when there was severe chattering in them.

6.8 Simulation issues

There was some trouble with the simulation of the USM. In Sverdrup-Thygeson et al. (2016) an ODE23tb solver was used, but since that is a variable step solver the step size got very small in the beginning of the simulation, and the simulation therefore never progressed. The reason for this might be that the variables that are to be integrand have a very steep curve in the beginning of the simulation. This is a known simulation problem. The solver was therefore changed to a fixed step solver (ODE5). The ODE5 solver also had quite a long simulation time, but at least it finished. Because of the long simulation time, it was not possible to try as many different controller gains as wanted. The controller gains might therefore not be optimal for the SMC algorithms, which means that the SMC algorithms might be able to give even better results than what were presented here.

Chapter 7

Conclusion and Further Work

7.1 Conclusion

In this thesis sliding mode control, sliding mode observers and stability theory regarding cascaded systems and finite-time stable properties were presented. The stability properties for a second-order general system, in cascade with the proposed SMC controllers and the state observer, were analysed. The closed-loop dynamics for the general system is proven uniformly globally asymptotically stable in all cases except one. The proposed SMC algorithms were tested on two different systems to see how well they performed. The SMC algorithms have also been compared to a PD-controller.

Two of the SMC algorithms, the super twisting algorithm and the quasi-continuous third-order SMC algorithm, gave great results regarding their ability to follow a desired path when disturbances were present. However, the nested third-order SMC algorithm only gave good results for the mass-spring-damper system and for the USM when the state observer was not used, i.e. the state observer had an effect on the performance of the algorithm. The two other algorithms were not affected, but they did not have such large observer errors. Both the nested and the quasi-continuous third-order SMC algorithms had issues with chattering, but with properly chosen gains it was possible to make the control input for the quasi-continuous third-order SMC algorithm smooth. This will, however, be a trade-off between the position error and the smoothness of the control input. It also got smoother by not using the state observer, i.e. the chattering from the estimated states can be a reason for the chattering in the control input. The super-twisting algorithm gave the overall best results, i.e. smallest error in all cases and smooth control input. The HOSM controllers are therefore not better than the lower-order algorithm tested here. In the cases when the control input does not appear in the first derivative of the sliding surface, and a HOSM algorithm has to be used, the quasi-continuous third-order SMC algorithm should be used as this was the one that gave the better results out of the HOSM algorithms. The PD-controller gave some competition to the nested third-order SMC algorithm with a state observer, but it was no competition to the other two SMC algorithms.

7.2 Further work

Some time should be spent on trying to find more optimal gains, to see if it is possible to improve the performance of the algorithms further. The new higher-order sliding mode control algorithm presented in Cruz-Zavala and Moreno (2017) and the adaptive higher-order sliding mode control presented in Edwards and Shtessel (2016), that were mentioned in Section 2.4 should also be tested and analysed. The new higher-order sliding mode controller should especially be analysed as there is also proposed a way to make Lyapunov functions for that controller. The stability analysis conducted when the super-twisting algorithm was used, should be extended to the super-twisting algorithm with adaptive gains. The algorithms tested here, especially the super-twisting algorithm, should also be tested on a 3D-model of the USM to see if it gives as good results as it does in 2D.

Bibliography

Antonelli, G., Chiaverini, S., June 24-26 1998. Singularity-free regulation of underwater vehicle-manipulator systems. In: Proc. American Control Conference. Philadelphia, Pennsylvania, pp. 399–403.

Bartolini, G., Pisano, A., Punta, E., Usai, E., 2003. A survey of applications of second-order sliding mode control to mechanical systems. International Journal of Control 76 (9-10), 875–892.

Borlaug, I.-L. G., 2016. Higher Order Sliding Mode Control: A literature study and application to underwater snake robots. Project report, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Borlaug, I.-L. G., Gravdahl, J. T., Sverdrup-Thygeson, J., Pettersen, K. Y., Okt. 29 - Nov. 1 2017. Trajectory tracking for underwater swimming manipulators using a super twisting algorithm. In: Submitted to: SWARM 2017: The 2nd International Symposium on Swarm Behavior and Bio-Inspired Robotics (Submitted). Kyoto, Japan.

Chalanga, A., Kamal, S., Fridman, L. M., Bandyopadhyay, B., Moreno, J. A., 2016. Implementation of Super-Twisting Control: Super-Twisting and Higher Order Sliding-Mode Observer-Based Approaches. IEEE Transactions on Industrial Electronics 63 (6), 3677–3685.

Cristi, R., Papoulias, F. A., Healey, A. J., 1990. Adaptive Sliding Mode Control of Autonomous Underwater Vehicles in the Dive Plane. IEEE Journal of Oceanic Engineering 15 (3), 152–160.

Cruz-Zavala, E., Moreno, J. A., 2017. Homogeneous High Order Sliding Mode design: A Lyapunov approach. Automatica 80, 232–238.

Dannigan, M. W., Russell, G. T., 1998. Evaluation and reduction of the dynamic coupling between a manipulator and an underwater vehicle. IEEE Journal of Oceanic Engineering. 23 (3), 260–273.

Defoort, M., Floquet, T., Kokosy, A., Perruquetti, W., 2009. A novel higher order sliding mode control scheme. Systems and Control Letters 58 (2), 102–108.

Dinuzzo, F., Ferrara, A., 2009. Higher order sliding mode controllers with optimal reaching. IEEE Transactions on Automatic Control 54 (9), 2126–2136.

Edwards, C., Shtessel, Y. B., 2016. Adaptive continuous higher order sliding mode control. Automatica 65, 183–190.

Fjellstad, O. E., Fossen, T. I., Dec. 14-16 1994. Singularity-free tracking of unmanned underwater vehicles in 6 DOF. In: Proc. 33rd IEEE Conference on Decision and Control. Lake Buena Vista, Florida, pp. 1128–1133.

Fossen, T., Sagatun, S., 1991. Adaptive control of nonlinear underwater robotic systems. Modeling, Identification and Control 12 (2), 95–105.

Fossen, T. I., June 19-22 1991. Adaptive macro-micro control of nonlinear underwater robotic systems. In: Proc. 5th International Conference on Advanced Robotics. Pisa, Italy, pp. 1569–1572.

Hung, J. Y., Gao, W., Hung, J. C., 1993. Variable Structure Control: A Survey. IEEE Transactions on Industrial Electronics 40 (1), 2–22.

Kelasidi, E., Pettersen, K. Y., Gravdahl, J. T., Liljebäck, P., May 31 - June 7 2014. Modeling of underwater snake robots. In: Proc. 2014 IEEE International Conference on Robotics and Automation. Hong Kong, China, pp. 4540–4547.

Khalil, H. K., 2002. Nonlinear systems, 3rd Edition. Prentice Hall, Upper Saddle River, N.J.

Kumari, K., Chalanga, A., Bandyopadhyay, B., Aug. 23-25 2016. Implementation of Super-Twisting Control on Higher Order Perturbed Integrator System using Higher Order Sliding Mode Observer. In: Proc. 10th IFAC Symposium on Nonlinear Control Systems. Vol. 49. California, USA, pp. 873–878.

Levant, A., 1993. Sliding order and sliding accuracy in sliding mode control. International Journal of Control 58 (6), 1247–1263.

Levant, A., 1998. Robust exact differentiation via sliding mode technique. Automatica 34 (3), 379–384.

Levant, A., 2001. Universal single-input-single-output (SISO) sliding-mode controllers with finite-time convergence. Automatic Control, IEEE Transactions on 46 (9), 1447–1451.

Levant, A., 2003a. Higher-order sliding modes, differentiation and output-feedback control. International Journal of Control 76 (9-10), 924–941.

Levant, A., Dec. 9-12 2003b. Quasi-continuous high-order sliding-mode controllers. In: 42nd IEEE International Conference on Decision and Control. Vol. 5. Maui, Hawaii USA, pp. 4605–4610.

Levant, A., 2005. Quasi-continuous high-order sliding-mode controllers. Automatic Control, IEEE Transactions on 50 (11), 1812–1816.

Liljebäck, P., Pettersen, K. Y., Stavdahl, Ø., Gravdahl, J. T., 2012. Snake Robots : Modelling, Mechatronics, and Control. Snake Robots : Modelling, Mechatronics, and Control. Springer, Dordrecht.

Loría, A., Panteley, E., 2005. Cascaded nonlinear time-varying systems: Analysis and design. Lecture Notes in Control and Information Sciences 311, 23–64.

Moreno, J. A., Dec 10-13 2012. Lyapunov function for Levant's Second Order Differentiator. In: Proc. IEEE 51st IEEE Conference on Decision and Control. Maui, Hawaii, USA, pp. 6448–6453.

Moreno, J. A., Osorio, M., 2012. Strict Lyapunov Functions for the Super-Twisting Algorithm. Automatic Control, IEEE Transactions on 57 (4), 1035–1040.

Polyakov, A., Fridman, L., 2014. Stability notions and lyapunov functions for sliding mode control systems. Journal of the Franklin Institute 351 (4), 1831–1865.

Rezapour, E., Pettersen, K. Y., Liljebäck, P., Gravdahl, J. T., May 31 - June 7 2014. Differential geometric modelling and robust path following control of snake robots using sliding mode techniques. In: Proc. 2014 IEEE International Conference on Robotics and Automation. Hong Kong, China, pp. 4532–4539.

Shtessel, Y., Edwards, C., Fridman, L., Levant, A., 2014. Sliding Mode Control and Observation. Control Engineering. Springer New York, NY, New York, NY.

Shtessel, Y., Taleb, M., Plestan, F., 2012. A novel adaptive-gain super-twisting sliding mode controller: Methodology and application. Automatica 48 (5), 759–769.

Shtessel, Y. B., Moreno, J. A., Plestan, F., Fridman, L. M., Poznyak, A. S., Dec. 15-17 2010. Super-twisting adaptive sliding mode control: A Lyapunov design. In: Proc. 49th IEEE Conference on Decision and Control. Atlanta, GA, USA, pp. 5109–5113.

Soylu, S., Buckham, B. J., Podhorodeski, R. P., 2008. A chattering-free sliding-mode controller for underwater vehicles with fault-tolerant infinity-norm thrust allocation. Ocean Engineering 35 (16), 1647–1659.

Sverdrup-Thygeson, J., Kelasidi, E., Pettersen, K. Y., Gravdahl, J. T., Sept. 13-16, 2016. Modeling of underwater swimming manipulators. In: Proc. 10th IFAC Conference on Control Applications in Marine Systems. Vol. 49. Trondheim, Norway, pp. 81–88.

Utkin, V. I., 1977. Survey Paper: Variable Structure Systems with Sliding Modes. IEEE Transactions on Automatic Control 22 (2), 212–222.

Young, K. D., Utkin, V. I., Özgüner, Ü., 1999. A control engineer's guide to sliding mode control. IEEE Transactions on Control Systems Technology 7 (3), 328–342.

Appendix A

Theorems and Lemmas

A.1 Theorems

Theorem A.1.1 (Hurwitz and Lyapunov function, theorem 4.6 Khalil (2002))**.** *A matrix A is Hurwitz; that is, $Re\{\lambda_i\} < 0$ for all eigenvalues of A, if and only if for any given positive definite symmetric matrix Q there exists a positive definite symmetric matrix P that satisfies the Lyapunov equation $PA + A^T P = -Q$. Moreover, if A is Hurwitz, then P is the unique solution of $PA + A^T P = -Q$.*

Theorem A.1.2 (Uniformly stable, theorem 4.8 Khalil (2002))**.** *Let $x = 0$ be an equilibrium point for $\dot{x} = f(t, x)$ and $D \subset R^n$ be a domain containing $x = 0$. Let $V : [0, \infty) \times D \to R$ be a continuously differentiable function such that*

$$\begin{aligned} &W_1(x) \leq V(t, x) \leq W_2(x) \\ &\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t, x) \leq 0 \end{aligned} \tag{A.1}$$

$\forall t \geq 0$ and $\forall x \in D$, where $W_1(x)$ and $W_2(x)$ are continuous positive definite functions on D. Then, $x = 0$ is uniformly stable.

Theorem A.1.3 (Uniformly asymptotically stable, theorem 4.9 Khalil (2002))**.** *Suppose the assumptions of Theorem A.1.2 are satisfied with the last inequality strengthened to*

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t, x) \leq -W_3(x) \tag{A.2}$$

$\forall t \geq 0$ and $\forall x \in D$, where $W_3(x)$ are continuous positive definite functions on D. Then, $x = 0$ is uniformly asymptotically stable. Moreover, if r and c are chosen such that $B_r = \{||x|| \leq r\} \subset D$ and $c < min_{||x||=r} W_1(x)$, then every trajectory starting in $\{x \in B_r \quad | \quad W_2(x) \leq c\}$ satisfies

$$||x(t)|| \leq \beta(||x_0(t)||, t - t_0), \quad \forall t \geq t_0 > 0 \tag{A.3}$$

for some class $\mathcal{KL}$ function β. Finally, if $D = R^n$ and $W_1(x)$ is radially unbounded, then $x = 0$ is globally uniformly asymptotically stable.

Theorem A.1.4 (Exponential stability, theorem 4.10 Khalil (2002))**.** *Let x = 0 be an equilibrium point for $\dot{x} = f(t, x)$ and $D \subset R^n$ be a domain containing $x = 0$. Let $V : [0, \infty) \times D \to R$ be a continuously differentiable function such that*

$$\begin{aligned} k_1||x||^a \leq V(t,x) \leq k_2||x||^a \\ \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t,x) \leq -k_3||x||^a \end{aligned} \tag{A.4}$$

$\forall t \geq 0$ and $\forall x \in D$, where k_1, k_2, k_3, and a are positive constants. Then, $x = 0$ is exponentially stable. If the assumptions hold globally, then $x = 0$ is globally exponentially stable.

Theorem A.1.5 (Bounded, theorem 4.18 Khalil (2002))**.** *Let $D \subset R^n$ be a domain that contains the origin and $V : [0, \infty) \times D \to R$ be a continuously differentiable function such that*

$$\begin{aligned} &\alpha_1(||x||) \leq V(t,x) \leq \alpha_2(||x||) \\ &\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t,x) \leq -W_3(x), \quad \forall ||x|| \geq \mu > 0 \end{aligned} \tag{A.5}$$

$\forall t \geq 0$ and $\forall x \in D$, where α_1 and α_2 are class $\mathcal{K}$ functions and $W_3(x)$ is continuous positive definite function. Take $r > 0$ such that $B_r \subset D$ and suppose that

$$\mu < \alpha_2^{-1}(\alpha_1(r)) \tag{A.6}$$

Then there exists a class $\mathcal{KL}$ function β and for every initial state $x(t_0)$, satisfying $||x(t_0)|| \leq \alpha_2^{-1}(\alpha_1(r))$, there is $T \geq 0$ (dependent on $x(t_0)$ and μ) such that the solution of $\dot{x} = f(t, x)$ satisfies

$$\begin{aligned} ||x(t)|| \leq \beta(||x(t_0)||, t - t_0), \quad \forall \quad t_0 \leq t \leq t_0 + T \\ ||x(t)|| \leq \alpha_1^{-1}(\alpha_2(\mu)), \quad \forall \quad t \geq t_0 + T \end{aligned} \tag{A.7}$$

Moreover, if $D = R^n$ and α_1 belong to class $\mathcal{K}_\infty$, then (A.7) *hold for any initial state $x(t_0)$, with no restrictions on how large μ is.*

A.2 Lemmas

Lemma A.2.1 (Non-vanishing perturbation, lemma 9.2 Khalil (2002))**.** *Let $x = 0$ be an exponentially stable equilibrium point of the nominal system*

$$\dot{x} = f(t, x). \tag{A.8}$$

Let $V(t, x)$ be a Lyapunov function of the nominal system that satisfies

$$\begin{aligned} c_1||x||^2 \leq V(t,x) \leq c_2||x||^2 \\ \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t,x) \leq -c_3||x||^2 \\ \left|\left|\frac{\partial V}{\partial x}\right|\right| \leq c_4||x|| \end{aligned} \tag{A.9}$$

in $[0, \infty) \times D$*, where* $D = \{x \in R^n \mid ||x|| < r\}$*. Suppose the perturbation term* $g(t, x)$ *satisfies*

$$||g(t, x)|| \leq \delta < \frac{c_3}{c_4}\sqrt{\frac{c_1}{c_2}}\theta_r \tag{A.10}$$

for all $t \geq 0$*, all* $x \in D$*, and some positive constant* $\theta < 1$*. Then, for all* $||x(t_0)|| < \sqrt{c_1/c_2}r$*, the solution* $x(t)$ *of the perturbed system*

$$\dot{x} = f(t, x) + g(t, x) \tag{A.11}$$

satisfies

$$||x(t)|| \leq kexp[-\gamma(t - t_0)]||x(t_0)||, \quad \forall t_0 \leq t < t_0 + T \tag{A.12}$$

and

$$||x(t)|| \leq b, \quad \forall t \geq t_0 + T \tag{A.13}$$

for some finite T, where

$$k = \sqrt{\frac{c_2}{c_1}}, \quad \gamma = \frac{(1 - \theta)c_3}{2c_2}, \quad b = \frac{c_4}{c_3}\sqrt{\frac{c_2}{c_1}}\frac{\delta}{\theta} \tag{A.14}$$

Lemma A.2.2 (Uniformly globally asymptotically stable of cascades, lemma 2.1 Loría and Panteley (2005))**.** *Consider the cascaded system:*

$$\begin{aligned} &\sum\nolimits_1 \left\{ \dot{x}_1 = f_1(t, x_1) + g(t, x)x_2 \right. \\ &\sum\nolimits_2 \left\{ \dot{x}_2 = f_2(t, x_2) \right. \end{aligned} \tag{A.15}$$

where $x_1 \in R^n$*,* $x_2 \in R^m$ *and the function* $f_1(x_1, x_2)$ *is continuously differentiable in* (x_1, x_2)*. The cascade* (A.15) *is UGAS if and only if the systems* $\dot{x}_1 = f_1(t, x_1)$ *and* $\dot{x}_2 = f_2(t, x_2)$ *are UGAS and the solutions of* (A.15) *are uniformly globally bounded (UGB).*

Appendix B

Finite-Time Stable Analysis

B.1 Analysis of the super-twisting algorithm with state observer

B.1.1 The estimated value for x_2 is used in the sliding surface

Overall closed-loop dynamics:

The overall closed-loop dynamics can be written as

$$\begin{aligned}
&\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 + e_2 \\ \dot{\hat{\sigma}} = c_1 e_2 - k_1 |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2 \operatorname{sgn}(\hat{\sigma}) \end{cases} \\
&\sum\nolimits_2 \begin{cases} \dot{e}_1 = -\lambda_1 |e_1|^{2/3} \operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 = -\lambda_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 = -\lambda_3 \operatorname{sgn}(e_1) + \dot{f}(t,x) \end{cases}
\end{aligned} \tag{B.1}$$

Proof. **Finite-time stable:**

In this section stability will be proven by using the fact that both the STA and the state observer are FTS, and the fact that the trajectories of $\sum_1$ cannot escape to infinity in finite time, as done in Chalanga et al. (2016). The closed-loop dynamics is described as in Equation (B.1). It is proven in Levant (2003a) and Moreno (2012), that the error dynamics represented in subsystem $\sum_2$ is FTS. This means that after a finite time T, $e = 0$. So after this time T, the closed-loop dynamic is

$$\begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 \\ \dot{\hat{\sigma}} = -k_1 |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2 \operatorname{sgn}(\hat{\sigma}) \end{cases} \tag{B.2}$$

As the STA is also FTS, which is proven in Moreno and Osorio (2012), $\sigma = v = 0$ after a finite time T. This leads to a closed-loop dynamic represented by

$$\begin{aligned} \dot{x}_1 &= -c_1 x_1 \\ x_2 &= \dot{x}_1 = -c_1 x_1 \end{aligned} \tag{B.3}$$

which is AS, when $c_1 > 0$. □

B.1.2 The estimated value for x_1 and x_2 are used in the sliding surface

Overall closed-loop dynamics:

The overall closed-loop dynamics can be written as

$$\begin{aligned} &\textstyle\sum_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 + c_1 e_1 + e_2 \\ \dot{\hat{\sigma}} = -k_1 |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2 \operatorname{sgn}(\hat{\sigma}) \end{cases} \\ &\textstyle\sum_2 \begin{cases} \dot{e}_1 = -\lambda_1 |e_1|^{2/3} \operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 = -\lambda_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 = -\lambda_3 \operatorname{sgn}(e_1) + \dot{f}(t, x) \end{cases} \end{aligned} \tag{B.4}$$

Proof. **Finite-time stable:**

The FTS analysis is exactly the same as for the system in the Section B.1.1. After a finite time T, $e = 0$. So after this time T, the closed-loop dynamic is

$$\begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 \\ \dot{\hat{\sigma}} = -k_1 |\hat{\sigma}|^{1/2} \operatorname{sgn}(\hat{\sigma}) + v \\ \dot{v} = -k_2 \operatorname{sgn}(\hat{\sigma}) \end{cases} \tag{B.5}$$

As the STA is also FTS, which is proven in Moreno and Osorio (2012), $\sigma = v = 0$ after a finite time T. This leads to a closed-loop dynamic represented by

$$\begin{aligned} \dot{x}_1 &= -c_1 x_1 \\ x_2 &= \dot{x}_1 = -c_1 x_1 \end{aligned} \tag{B.6}$$

which is AS, when $c_1 > 0$. □

B.2 Analysis of the HOSM algorithms with differentiator and state observer

B.2.1 The estimated value for x_1 and x_2 are used in the sliding surface:

Overall closed-loop dynamics:

The overall closed-loop dynamics can be written as

$$\begin{aligned}
&\sum\nolimits_1 \begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 + c_1 e_1 + e_2 \\ \dot{\hat{\sigma}} = u_c \end{cases} \\
&\sum\nolimits_2 \begin{cases} \dot{e}_1 = -\lambda_1 |e_1|^{2/3} \operatorname{sgn}(e_1) + e_2 \\ \dot{e}_2 = -\lambda_2 |e_1|^{1/3} \operatorname{sgn}(e_1) + e_3 \\ \dot{e}_3 = -\lambda_3 \operatorname{sgn}(e_1) + \dot{f}(t,x) \end{cases}
\end{aligned} \tag{B.7}$$

Proof. **Finite-time stable:**

The FTS analysis is exactly the same as for the system in Section B.1.1 and B.1.2. After a finite time T, $e = 0$. So after this time T, the closed-loop dynamic is

$$\begin{cases} \dot{x}_1 = \hat{\sigma} - c_1 x_1 \\ \dot{\hat{\sigma}} = u_c \end{cases} \tag{B.8}$$

As the HOSM algorithm with a differentiator is also FTS, $\hat{\sigma} = 0$ after a finite time T. This leads to a closed-loop dynamic represented by

$$\begin{aligned} \dot{x}_1 &= -c_1 x_1 \\ x_2 = \dot{x}_1 &= -c_1 x_1 \end{aligned} \tag{B.9}$$

which is AS, when $c_1 > 0$. □

Appendix C

SWARM 2017 article

Trajectory tracking for underwater swimming manipulators using a super twisting algorithm

I.-L. G. Borlaug[1†] J. T. Gravdahl[1] J. Sverdrup-Thygeson[2] and K.Y. Pettersen[2]

[1]Department of Engineering Cybernetics, NTNU, Norwegian University of Science and Technology, Trondheim, Norway
(E-mail:ilborlaug@stud.ntnu.no, Tommy.Gravdahl@ntnu.no)
[2]Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, NTNU, Norwegian University of Science and Technology, Trondheim, Norway
(E-mail: {Jorgen.Sverdrup-Thygeson, Kristin.Y.Pettersen}@ntnu.no

Abstract: The Underwater Swimming Manipulator (USM) is a snake-like, multi-articulated, underwater robot equipped with thrusters. One of the main purposes of the USM is to act like an underwater floating base manipulator. As such, it is essential to achieve good station-keeping and trajectory tracking performance for the USM using the thrusters, while using the joints to attain a desired position and orientation of the head and tail of the USM. In this paper, we propose a sliding mode control (SMC) law, in particular the super-twisting algorithm with adaptive gains, for trajectory tracking of the USMs center of mass. A higher-order sliding mode observer is proposed for state estimation. Furthermore, we perform a simulation study to verify the applicability of the proposed control law and show that it has better tracking performance than a linear PD-controller.

Keywords: Underwater Swimming Manipulator, Super-Twisting, Siding Mode Control, Sliding Mode Observer.

1. INTRODUCTION

An underwater swimming manipulator (USM) is an underwater snake robot (USR) equipped with thrusters [1]. The main purposes of the thrusters are to provide forward thrust without requiring the snake robot to follow an undulating gait pattern, which is of particular importance in narrow, confined environments, and to provide sideways thrust for station-keeping and trajectory tracking. The station-keeping and trajectory tracking capabilities enable the USM to act like an underwater floating base manipulator. The slender, multi-articulated body provides the USM with outstanding accessibility and flexibility. As such, the USM is a crossover between a small autonomous underwater vehicle (AUV) and an underwater snake robot. The USM possesses the high kinematic redundancy of the USR, and at the same time it has the advantages of the AUV in terms of full energy-efficient hydrodynamic properties and tether-less operation. Moreover, the USM has the advantages of remotely operated vehicles (ROVs) regarding full actuation and the capability of doing intervention operations. Since the USM can use the thrusters instead of the joints to create forward propulsion, the joints can be used to perform manipulation tasks and, thus, exploit the full potential of the inherent kinematic redundancy. This has been addressed in details in [2], [3].

As a floating base manipulator, the USM can move itself to an area of interest, position its tail at the initial base location, and then start to operate as a robotic manipulator. When the USM carries out a manipulation task, the overall motion of the USM and the joint angle velocities can be determined by the desired velocities of the end-effector, i.e. the desired motion of the head of the USM. One approach for this is described in [4], where the base motion and the joint angle motion of the USM are assigned using a redundancy resolution technique based on inverse kinematics. The outputs of this procedure are time-varying velocity references for the base and the joints. This inverse kinematics method is only one of many ways to calculate the velocity references.

Controller design for underwater robots (URs) such as the USM and ROVs, is a complex problem [5]. URs are often subject to hydrodynamic and hydrostatic parameter uncertainties, uncertain thruster characteristics, unknown disturbances, and unmodelled dynamic effects, e.g. thruster dynamics and coupling forces caused by joint motion. As the USM has no separate vehicle base and a low mass compared to an ROV, the motion of the joints become more significant for the overall motion of the USM as a rigid body than for the ROV. The coupling forces are therefore more prominent for the USM, and this increases the complexity of motion control of the USM, compared to an ROV.

Sliding mode control (SMC) is a robust and versatile non-linear control approach, and we will in this paper show that it is well suited for control of USMs. For underwater vehicles, in general, some important contributions are given in [6], [7], [8], [9], [10] and [11]. In [6], a singularity-free SMC approach, inspired by [12], is used for set-point regulation of a UR with uncertainties in the hydrodynamic parameters. In [7], [8], SMC is employed to cope with multiplicative uncertainty in the thruster configuration matrix. The combination of sliding mode and adaptive control is studied in [7], [8], [11]. In particular, in [11], sliding mode control is combined with adaptive PID controller gains and an adaptive update of the upper bound on the disturbances and the parameter uncertainties. SMC is also applicable to deal

† I-L. G. Borlaug is the presenter of this paper.

with linearisation errors [9] and the coupling effects between an underwater vehicle and an attached manipulator arm [10]. Sliding mode techniques have been applied to land-based snake robots in [13] to achieve robust tracking of a desired gait pattern and under-actuated straight line path following. However, SMC have to the authors' best knowledge, never been applied to underwater snake robots, and in particular, to USRs with thrusters.

In this paper SMC is applied to the robot model proposed in [1]. The model in [1] extends the 2D model proposed in [14], by modelling also additional effectors and considering the force allocation among these effectors. In this paper, the linear PD-controller used in [1] is replaced by a super-twisting algorithm (STA) for sliding mode control accompanied by a higher-order sliding mode observer. We consider the tracking problem for the position of the centre of mass of the USM.

The first-order relay controller [15], has large problems with chattering. Chattering is the high-frequency switching of the control signals commonly associated with SMC. To eliminate chattering, we could have used saturation control, but since the sliding mode does not exist inside the boundary layer, the effectiveness of the controller is challenged when parasitic dynamics are considered, [16]. Therefore the super-twisting algorithm will be used. The STA is one of the most powerful second-order continuous sliding mode control algorithms. It was first introduced in [17] and has thereafter been used for multiple systems. The STA attenuates chattering and will, therefore, give a smoother control signal. A challenge with the STA is that it only works with bounded perturbations, and therefore a conservative upper bound has to be used when designing the controller to ensure that sliding is maintained. To eliminate this problem, we will use adaptive STA [18]. The gains can then adapt to a level where they are as small as possible but still guarantee that sliding is maintained. Since the STA is only applicable to systems where the control input appears in the equation for the first derivative of the sliding variable, both the position and velocity of the USM need to be available for measurement. For the case when only the position measurements are available, we will use a higher-order sliding mode observer, proposed in [19], to estimate the states. As such, we combine the results from [18] and [19], as done in [20], but we will replace the regular STA with a STA with adaptive gains. We will then apply this control structure to the USM. We also present simulations that verify that the proposed approach applies well to USMs, and also compare the results to a standard PD-controller.

The remainder of this paper is organized as follows. In Section 2 the robot model used will be explained in more detail. The control and observer design is presented in Section 3, and in Section 4 the simulation results will be presented. Conclusions and suggestions for future work are given in Section 5.

2. UNDERWATER SWIMMING MANIPULATOR (USM) MODEL

In this section, the equations of motion for the USM and the force allocation matrix will be explained. We refer to [1] and [14] for further details.

2.1. Kinematics

The position of the center of mass (CM) of the USM, $p_{CM} \in R^2$, expressed in the global frame is

$$p_{CM} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{1}{m_t}\sum_{i=1}^{n} m_i x_i \\ \frac{1}{m_t}\sum_{i=1}^{n} m_i y_i \end{bmatrix} = \frac{1}{m_t}\begin{bmatrix} e^T M X \\ e^T M Y \end{bmatrix} \quad (1)$$

where (x_i, y_i), $i = 1, \cdots, n$ are the coordinates of the CM of link i in the global frame, m_i is the mass of link i and $m_t = \sum_{i=1}^{n} m_i$ is the total mass of the USM. Eq. (1) is valid because it is assumed that the mass of each link is uniformly distributed. The matrix representation of the force balance for all the links is

$$M\ddot{X} = D^T h_x + f_x + f_{px}, \quad M\ddot{Y} = D^T h_y + f_y + f_{py} \quad (2)$$

where f_{px} and f_{py} are the forces from the additional effectors, h_x and h_y are the joint constraint forces and f_x and f_y are the fluid forces acting on the links. By differentiating Eq. (1) and inserting Eq. (2), the joint constraint forces cancel out, and the translational motion of the CM of the USM can be written as

$$m_t \ddot{p}_x = e^T(f_x + f_{px}), \qquad m_t \ddot{p}_y = e^T(f_y + f_{py}). \quad (3)$$

2.2. Force Allocation

The force allocation distribution is given by

$$\begin{aligned} \tau_{CM} &= \begin{bmatrix} F_{CM,x} \\ F_{CM,y} \\ M_{CM,z} \end{bmatrix} \\ &= \begin{bmatrix} e^T & 0^{1\times n} \\ 0^{1\times n} & e^T \\ e^T S_\psi K & -e^T C_\psi K \end{bmatrix} \begin{bmatrix} f_{px} \\ f_{py} \end{bmatrix} = T(\psi) f_p, \end{aligned} \quad (4)$$

where $T(\psi)$ is the allocation matrix and $f_p = [f_{p,k_1}, \ldots, f_{p,k_r}]$ is the vector of scalar effector forces. The allocation matrix represents the mapping between the effector forces and the forces and moments acting on the CM of the USM. It is assumed that the additional effector forces are acting through the CM of each link. The primary objective for the force allocation method is to distribute the efforts among the additional effectors to obtain the desired forces and moments. In the next section, we propose a novel method for calculation of the desired forces and moments, together with a non-linear observer for position and velocity.

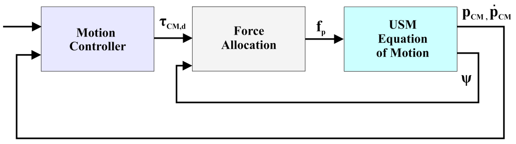


Fig. 1 System overview USM, [1]

3. CONTROL AND OBSERVER DESIGN

Control problem: *Assume that there exist a guidance system which determines a suitable path for the USM to follow. The task at hand is to design a motion controller that calculates the desired forces for the translational motion F_{CM}, and the desired moments for the rotational motion M_{CM}, of the USM.*

We will in the following use a super-twisting algorithm with adaptive gains to calculate the desired forces, F_{CM}. To calculate the desired moments, M_{CM}, we will use a proportional controller. The desired forces and moments are represented by

$$\tau_{\mathrm{CM,d}} = \begin{bmatrix} F_{\mathrm{CM,d}} \\ M_{\mathrm{CM,d}} \end{bmatrix} = \begin{bmatrix} F_{\mathrm{CM,d_x}} \\ F_{\mathrm{CM,d_y}} \\ M_{\mathrm{CM,d}} \end{bmatrix} \tag{5}$$

3.1. Sliding surface design

First we define the error variable. As the output variable for the translational motion of the USM is p_{CM}, the error variable can be defined as

$$\tilde{p} = \begin{bmatrix} \tilde{p}_x \\ \tilde{p}_y \end{bmatrix} = p_{\mathrm{CM}} - p_{\mathrm{CM,ref}} = \begin{bmatrix} p_x - p_{x,\mathrm{ref}} \\ p_y - p_{y,\mathrm{ref}} \end{bmatrix} \tag{6}$$

where $p_{\mathrm{CM,ref}}$ is the desired position of the CM of the USM in the global frame. The sliding surface should be selected such that the state trajectories of the controlled system are forced onto the sliding surface $\sigma = \dot{\sigma} = 0$, where the system behaviour meets the design specifications. The controller u should also appear in the first derivative of σ, so that the relative degree is equal to 1. The sliding surface σ can then be chosen as

$$\begin{aligned} \sigma = \begin{bmatrix} \sigma_x \\ \sigma_y \end{bmatrix} &= \tilde{p} + \lambda\dot{\tilde{p}} = \begin{bmatrix} \tilde{p}_x \\ \tilde{p}_y \end{bmatrix} + \begin{bmatrix} \lambda\dot{\tilde{p}}_x \\ \lambda\dot{\tilde{p}}_y \end{bmatrix} \\ &= \begin{bmatrix} p_x - p_{x,\mathrm{ref}} \\ p_y - p_{y,\mathrm{ref}} \end{bmatrix} + \begin{bmatrix} \lambda(\dot{p}_x - \dot{p}_{x,\mathrm{ref}}) \\ \lambda(\dot{p}_y - \dot{p}_{y,\mathrm{ref}}) \end{bmatrix} \end{aligned} \tag{7}$$

Since only the position, p_{CM}, of the centre of mass is available for measurement, an observer for the states is designed. The observer states will be used in the sliding surface, and following the structure of Eq. (7), the revised sliding surface is then

$$\hat{\sigma} = \begin{bmatrix} \hat{\sigma}_x \\ \hat{\sigma}_y \end{bmatrix} = \begin{bmatrix} \hat{p}_x - p_{x,\mathrm{ref}} \\ \hat{p}_y - p_{y,\mathrm{ref}} \end{bmatrix} + \begin{bmatrix} \lambda(\dot{\hat{p}}_x - \dot{p}_{x,\mathrm{ref}}) \\ \lambda(\dot{\hat{p}}_y - \dot{p}_{y,\mathrm{ref}}) \end{bmatrix} \quad . \tag{8}$$

3.2. Control input design

The control input can be written, by using Eq. (4) and Eq. (5), as

$$u = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = F_{CM} = \begin{bmatrix} F_{\mathrm{CM,x}} \\ F_{\mathrm{CM,y}} \end{bmatrix} = \begin{bmatrix} e^T f_{px} \\ e^T f_{py} \end{bmatrix}. \tag{9}$$

By replacing $e^T f_{px}$ and $e^T f_{py}$ in Eq. (3), with u_x and u_y, the translational motion of the CM of the USM can be rewritten as

$$m_t\ddot{p}_x = e^T f_x + u_x, \qquad m_t\ddot{p}_y = e^T f_y + u_y. \tag{10}$$

3.2.1. The super-twisting algorithm with adaptive gains

The STA with adaptive gains proposed in [18] can be written as

$$\begin{aligned} u_{\mathrm{STA}} = \begin{bmatrix} u_{\mathrm{STA,x}} \\ u_{\mathrm{STA,y}} \end{bmatrix} &= \begin{bmatrix} -\alpha_x \|\sigma_x\|^{1/2}\operatorname{sgn}(\sigma_x) + v_x \\ -\alpha_y \|\sigma_y\|^{1/2}\operatorname{sgn}(\sigma_y) + v_y \end{bmatrix} \\ \dot{v} = \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} &= \begin{bmatrix} -\beta_x \operatorname{sgn}(\sigma_x) \\ -\beta_y \operatorname{sgn}(\sigma_y) \end{bmatrix} \end{aligned} \tag{11}$$

where the adaptive gains are defined as

$$\dot{\alpha} = \begin{bmatrix} \dot{\alpha}_x \\ \dot{\alpha}_y \end{bmatrix} = \begin{bmatrix} \begin{cases} \omega_1\sqrt{\frac{\gamma_1}{2}}, & \text{if } \sigma_x \neq 0 \\ 0, & \text{if } \sigma_x = 0 \end{cases} \\ \begin{cases} \omega_1\sqrt{\frac{\gamma_1}{2}}, & \text{if } \sigma_y \neq 0 \\ 0, & \text{if } \sigma_y = 0 \end{cases} \end{bmatrix} \tag{12}$$

and

$$\beta = \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix} = \begin{bmatrix} 2\varepsilon\alpha_x + \lambda + 4\varepsilon^2 \\ 2\varepsilon\alpha_y + \lambda + 4\varepsilon^2 \end{bmatrix}, \tag{13}$$

where ε, λ, γ_1 and ω_1 are positive constants. For implementation purposes, a small boundary is put on the sliding surface and the adaptive gains can be expressed as

$$\begin{aligned} \dot{\alpha} = \begin{bmatrix} \dot{\alpha}_x \\ \dot{\alpha}_y \end{bmatrix} &= \begin{bmatrix} \begin{cases} \omega_1\sqrt{\frac{\gamma_1}{2}}, & \text{if } \|\sigma_x\| > \alpha_m \\ 0, & \text{if } \|\sigma_x\| \leq \alpha_m \end{cases} \\ \begin{cases} \omega_1\sqrt{\frac{\gamma_1}{2}}, & \text{if } \|\sigma_y\| > \alpha_m \\ 0, & \text{if } \|\sigma_y\| \leq \alpha_m \end{cases} \end{bmatrix} \\ \beta = \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix} &= \begin{bmatrix} 2\varepsilon\alpha_x + \lambda + 4\varepsilon^2 \\ 2\varepsilon\alpha_y + \lambda + 4\varepsilon^2 \end{bmatrix} \end{aligned} \tag{14}$$

where the design parameter α_m is a small positive constant.

3.2.2. State observer

By designing the observer structure as in [19], the state observer is chosen as

$$\begin{aligned} \dot{\hat{p}}_1 = \begin{bmatrix} \dot{\hat{p}}_{1,x} \\ \dot{\hat{p}}_{1,y} \end{bmatrix} &= \begin{bmatrix} \hat{p}_{2,x} + z_{1,x} \\ \hat{p}_{2,y} + z_{1,y} \end{bmatrix} = \begin{bmatrix} \hat{p}_{2,x} + k_1\|e_{1,x}\|^{2/3}\operatorname{sgn}(e_{1,x}) \\ \hat{p}_{2,y} + k_1\|e_{1,y}\|^{2/3}\operatorname{sgn}(e_{1,y}) \end{bmatrix} \\ \dot{\hat{p}}_2 = \begin{bmatrix} \dot{\hat{p}}_{2,x} \\ \dot{\hat{p}}_{2,y} \end{bmatrix} &= \begin{bmatrix} \hat{p}_{3,x} + z_{2,x} + \frac{1}{m_t}u_x \\ \hat{p}_{3,y} + z_{2,y} + \frac{1}{m_t}u_y \end{bmatrix} \\ &= \begin{bmatrix} \hat{p}_{3,x} + k_2\|e_{1,x}\|^{1/3}\operatorname{sgn}(e_{1,x}) + \frac{1}{m_t}u_x \\ \hat{p}_{3,y} + k_2\|e_{1,y}\|^{1/3}\operatorname{sgn}(e_{1,y}) + \frac{1}{m_t}u_y \end{bmatrix} \\ \dot{\hat{p}}_3 = \begin{bmatrix} \dot{\hat{p}}_{3,x} \\ \dot{\hat{p}}_{3,y} \end{bmatrix} &= \begin{bmatrix} z_{3,x} \\ z_{3,y} \end{bmatrix} = \begin{bmatrix} k_3\operatorname{sgn}(e_{1,x}) \\ k_3\operatorname{sgn}(e_{1,y}) \end{bmatrix} \end{aligned} \tag{15}$$

where k_1, k_2 and k_3 are gains to be chosen according to [21] and [22], $e_{1,x} = p_x - \hat{p}_{1,x}$ and $e_{1,y} = p_y - \hat{p}_{1,y}$. One choice of parameters that meets the requirements in [21] and [22], is according to [20], $k_1 = 6L^{1/3}$, $k_2 = 11L^{1/2}$ and $k_3 = 6L$, where L is a sufficiently large constant.

3.2.3. Control input

In order for the STA to be applicable, the control input needs to be chosen such that the control appears in the equation of the first derivative of the sliding variable. In particular, we want to have $\dot{\hat{\sigma}} = u_{\text{STA}}$. Taking the time derivative of Eq. (8) and substituting $\dot{\hat{p}}_1$ and $\dot{\hat{p}}_2$, defined in Eq. (15), we find that

$$\begin{aligned}\dot{\hat{\sigma}} &= (\dot{\hat{p}}_1 - \dot{p}_{\text{ref}}) + (\dot{\hat{p}}_2 - \ddot{p}_{\text{ref}}) \\ &= (\hat{p}_2 + z_1 - \dot{p}_{\text{ref}}) + (\hat{p}_3 + z_2 + \frac{1}{m_t}u - \ddot{p}_{\text{ref}})\end{aligned} \tag{16}$$

where u is defined as in Eq. (9). By choosing u to be

$$u = m_t(-\hat{p}_2 - z_1 + \dot{p}_{\text{ref}} - \hat{p}_3 - z_2 + \ddot{p}_{\text{ref}} + u_{\text{STA}}) \tag{17}$$

we obtain

$$\dot{\hat{\sigma}} = u_{\text{STA}}. \tag{18}$$

3.2.4. PD-controller

We want to compare the performance of the SMC algorithms to an existing controller for USMs with respect to disturbances and modelling errors. We will to the standard PD-controller that was proposed in [1]. This is implemented by replacing u_{STA} in Eq. (17) with

$$u_{\text{PD}} = k_d^{CCM}\begin{bmatrix}\dot{p}_{x,\text{ref}} - \dot{\hat{p}}_x \\ \dot{p}_{y,\text{ref}} - \dot{\hat{p}}_y\end{bmatrix} + k_p^{CM}\begin{bmatrix}p_{x,\text{ref}} - \hat{p}_x \\ p_{y,\text{ref}} - \hat{p}_y\end{bmatrix} \tag{19}$$

where k_d^{CM} and k_p^{CM} are controller gains.

4. SIMULATION RESULTS

4.1. Implementation

The complete model with the force allocation matrix is implemented in MATLAB. The USM implemented has $n = 16$ links, each one having length $2l_i = 0.14$ m and mass $m_i = 0.6597$ kg. The thruster configuration used corresponds to configuration 2 in [1]. This has one tail thruster attached to link 1 exerting a force along the x-axis of the link and four additional thrusters located at link number 3, 6, 11 and 14, exerting forces normal to the links. For more details regarding the parameters used in the model, please see [1]. We have implemented two different case studies, one called torpedo mode, which is described in Section 4.1.1, and one called operation mode, described in Section 4.1.2.

4.1.1. Case 1 - Torpedo mode

We want the USM to move as a torpedo-shaped AUV when it is moving from one place to another. To simulate this type of behaviour, the link angles were set to zero, i.e. there was no lateral undulation, and a line-of-sight (LOS) guidance law was used for heading control. For information on how the LOS guidance law was incorporated into the system and the motivation behind this choice, see [1]. This simulation case is shown in Fig. 2.

4.1.2. Case 2- Operation mode

When the USM is in operation mode, it will use the thrusters to stay in one place or move around, and use the end-effector at the head of the USM to do the operation. The motion of the joints can be seen as a disturbance to the CM position control system, as it will inflict unwanted motion on the CM of the USM. This simulation case investigates how well the proposed STA attenuates the unwanted effects of the joint motion. The simulated operation is an inspection, which entails that the head of the USM first moves in one direction and then the other, while the thrusters should keep the USM on the reference path. This type of simulation is shown in Fig. 3, where the USM head changes direction at 10, 20 and 30 seconds.

4.2. Simulations

As described in Section 3.2.2 the gain parameter L needs to be chosen sufficiently large, and for the simulations L was chosen through trial and error. The PD-controller gains were also chosen by this method. λ in Eq. (8) was set to 1. For the simulations an ode5 solver, with fixed step size 10^{-5} was used. In Table 1 the maximum position error after settling is presented for both the

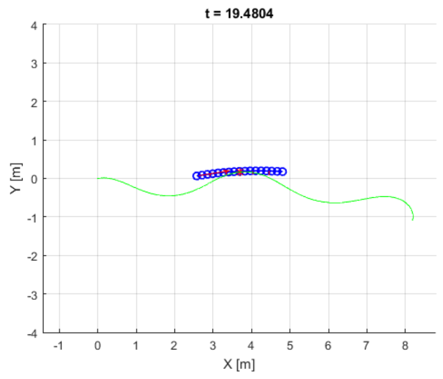


Fig. 2 Torpedo mode USM simulation

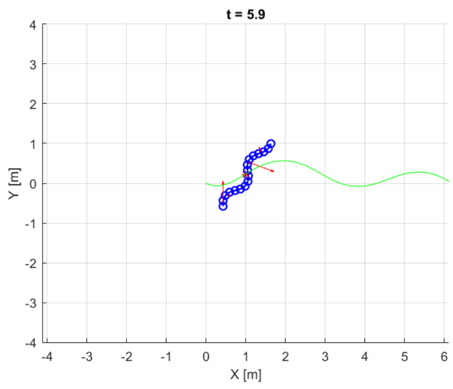


Fig. 3 Operation mode USM simulation

PD and SMC controllers.

4.2.1. The super-twisting algorithm with adaptive gains: The gains in the super-twisting algorithm with adaptive gains were set to: $\varepsilon = 1, \lambda = 1, \gamma_1 = 1, \omega_1 = 8, \alpha_m = 0.05$, and the observer gain was set to: $L = 55$. The simulations for torpedo mode can be seen in Fig. 4, and the simulations for operation mode can be seen in Fig. 5.

4.2.2. The PD-controller
The gains for the PD-controller were set to: $k_d^{CM} = 10$ and $k_p^{CM} = 0.1$. The torpedo mode simulation can be seen in Fig. 7 and the operation mode simulation can be seen in Fig. 8.

Table 1 Absolute maximum value for position error

Algorithm	**Error**			
	Torpedo		Operation	
	x	y	x	y
The STA with adaptive gains	$3.6134 \cdot 10^{-4}$	$2.8763 \cdot 10^{-4}$	$3.6127 \cdot 10^{-4}$	0.0014
PD-controller	0.0321	0.0354	0.0486	0.0359

4.3. Discussion

From Figs. 4 and 5 we can see that the proposed control law is indeed applicable. From Figs. 4 to 8 and Table 1, we can also see that the STA algorithm with adaptive gains is superior to the PD-controller because it has smaller position errors in both cases.

It is worth noticing that for case 2, operation mode, the difference in error is not very large in y-direction. From Fig. 9 and Fig. 6, it can be seen that for the PD-controller the absolute position error is more constant around 0.04 than for the STA with adaptive gains. The reason for the larger absolute position error in the y-direction for the STA is the peaks that can be seen in Fig. 6. These peaks are from when the USM shifts position, and the error is therefore only larger in some small time period when the USM shifts position.

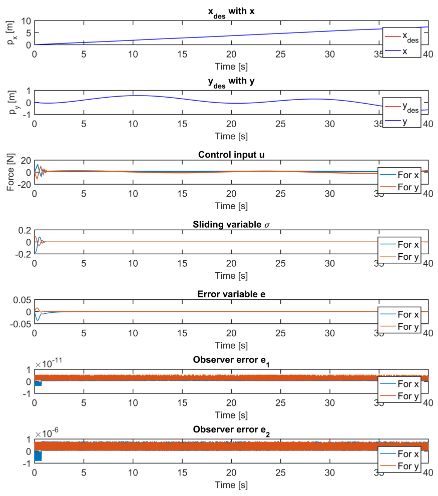


Fig. 4 Torpedo mode:Simulation of STA with state observer

5. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have discussed the use of the USM as a floating base manipulator, for which the trajectory tracking performance is so important, and how the complexity of motion control is larger for USMs than for ROVs. We have proposed a second-order sliding mode control law for trajectory tracking, with the use of sliding mode observer for the case when velocity measurements are not available. Furthermore, we have performed a simulation study to verify the applicability of the proposed control law and shown that it gives better tracking

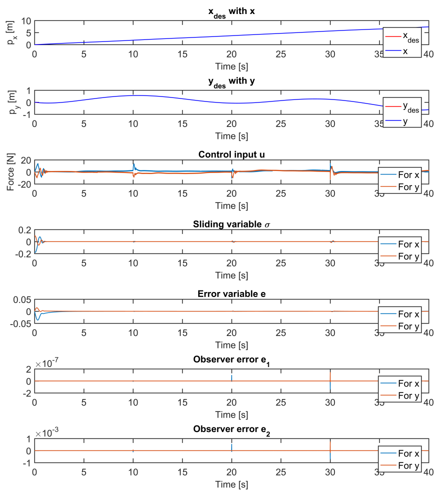


Fig. 5 Operation mode: Simulation of STA with state observer

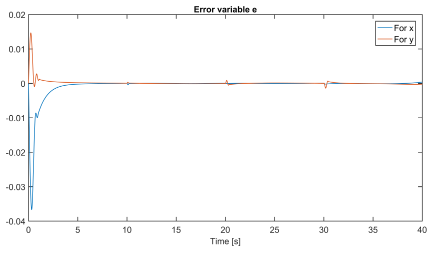


Fig. 6 Operation mode: Position error for the STA

performance than a linear PD-controller.

Future work includes investigating the best choice of control parameters, stability analysis of the closed-loop-dynamics, and extending the results to 3D.

6. ACKNOWLEDGMENTS

This research was funded by the Research Council of Norway through the Centres of Excellence funding scheme, project No. 223254 NTNU AMOS, and by VISTA, a basic research program in collaboration between The Norwegian Academy of Science and Letters, and Statoil.

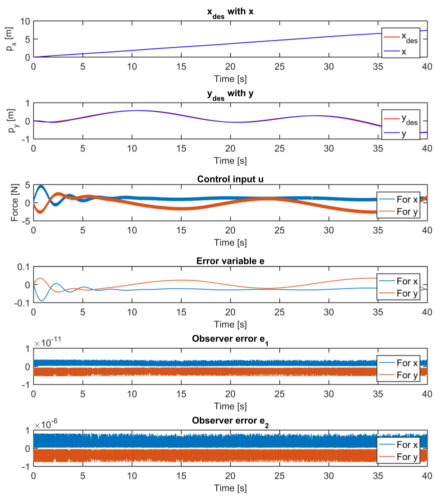


Fig. 7 Torpedo mode: Simulation of PD-controller

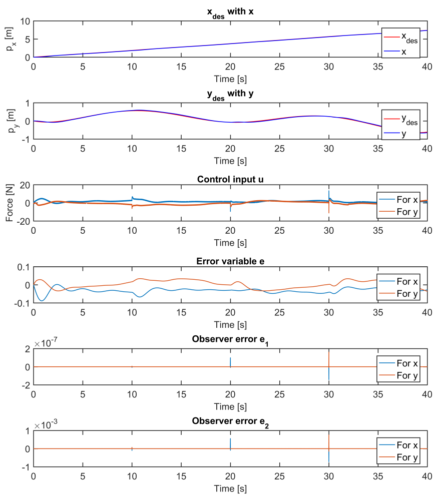


Fig. 8 Operation mode: Simulation of PD-controller

REFERENCES

[1] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, "Modeling of underwater swimming manipulators," in *Proc. 10th IFAC Conference on Control Applications in Marine Systems*, vol. 49, no. 23, Trondheim, Norway, Sept. 13-16, 2016, pp. 81–88.

[2] ——, "A control framework for biologically inspired underwater swimming manipulators equipped with thrusters," in *Proc. 10th IFAC Conference on Control Applications in Marine Systems*, vol. 49, no. 23, Trondheim, Norway, Sep. 13-16 2016, pp. 89–96.

[3] ——, "The Underwater Swimming Manipulator - A Bio-Inspired AUV," in *Proc. 2016 IEEE OES Autonomous Underwater Vehicles.*, Tokyo, Japan, Nov. 6-8 2016.

[4] J. Sverdrup-Thygeson, S. Moe, K. Y. Pettersen, and J. T. Gravdahl, "Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks," in *Proc. 1st IEEE Conference on Control Technology and Applications.*, Kohala Coast, Hawaii, Aug. 27-30 2017, (accepted).

[5] G. Antonelli, *Underwater Robots*, ser. Springer Tracts in Advanced Robotics. Springer International Publishing, 2014, vol. 96.

[6] G. Antonelli and S. Chiaverini, "Singularity-free regulation of underwater vehicle-manipulator systems," in *Proc. American Control Conference.*, Philadelphia, Pennsylvania, June 24-26 1998, pp. 399–403.

[7] T. I. Fossen, "Adaptive macro-micro control of nonlinear underwater robotic systems," in *Proc. 5th International Conference on Advanced Robotics.*, Pisa, Italy, June 19-22 1991, pp. 1569–1572.

[8] T. Fossen and S. Sagatun, "Adaptive control of nonlinear underwater robotic systems," *Modeling, Identification and Control*, vol. 12, no. 2, pp. 95–105, 1991.

[9] R. Cristi, F. A. Papoulias, and A. J. Healey, "Adap-

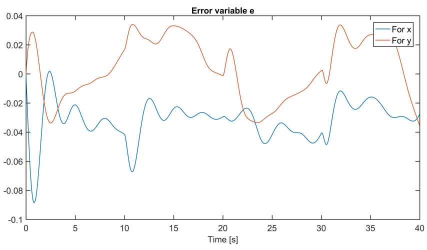


Fig. 9 Operation mode: Position error for the PD-controller

tive Sliding Mode Control of Autonomous Underwater Vehicles in the Dive Plane," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 152–160, 1990.

[10] M. W. Dannigan and G. T. Russell, "Evaluation and reduction of the dynamic coupling between a manipulator and an underwater vehicle," *IEEE Journal of Oceanic Engineering.*, vol. 23, no. 3, pp. 260–273, 1998.

[11] S. Soylu, B. J. Buckham, and R. P. Podhorodeski, "A chattering-free sliding-mode controller for underwater vehicles with fault-tolerant infinity-norm thrust allocation," *Ocean Engineering*, vol. 35, no. 16, pp. 1647–1659, 2008.

[12] O. E. Fjellstad and T. I. Fossen, "Singularity-free tracking of unmanned underwater vehicles in 6 DOF," in *Proc. 33rd IEEE Conference on Decision and Control.*, Lake Buena Vista, Florida, Dec. 14-16 1994, pp. 1128–1133.

[13] E. Rezapour, K. Y. Pettersen, P. Liljebäck, and J. T. Gravdahl, "Differential geometric modelling and robust path following control of snake robots using sliding mode techniques," in *Proc. 2014 IEEE International Conference on Robotics and Automation.*, Hong Kong, China, May 31 - June 7 2014, pp. 4532–4539.

[14] E. Kelasidi, K. Y. Pettersen, J. T. Gravdahl, and P. Liljebäck, "Modeling of underwater snake robots," in *Proc. 2014 IEEE International Conference on Robotics and Automation.*, Hong Kong, China, May 31 - June 7 2014, pp. 4540–4547.

[15] 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, 1993.

[16] K. D. Young, V. I. Utkin, and Ü. Özgüner, "A control engineer's guide to sliding mode control," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 3, pp. 328–342, 1999.

[17] A. Levant, "Sliding order and sliding accuracy in sliding mode control," *International Journal of Control*, vol. 58, no. 6, pp. 1247–1263, 1993.

[18] Y. B. Shtessel, J. A. Moreno, F. Plestan, L. M. Fridman, and A. S. Poznyak, "Super-twisting adaptive sliding mode control: A Lyapunov design," in *Proc. 49th IEEE Conference on Decision and Control.*, Atlanta, GA, USA, Dec. 15-17 2010, pp. 5109–5113.

[19] K. Kumari, A. Chalanga, and B. Bandyopadhyay, "Implementation of Super-Twisting Control on Higher Order Perturbed Integrator System using Higher Order Sliding Mode Observer," in *Proc. 10th IFAC Symposium on Nonlinear Control Systems.*, vol. 49, no. 18, California, USA, Aug. 23-25 2016, pp. 873–878.

[20] A. Chalanga, S. Kamal, L. M. Fridman, B. Bandyopadhyay, and J. A. Moreno, "Implementation of Super-Twisting Control: Super-Twisting and Higher Order Sliding-Mode Observer-Based Approaches," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 6, pp. 3677–3685, 2016.

[21] A. Levant, "Robust exact differentiation via sliding mode technique," *Automatica*, vol. 34, no. 3, pp. 379–384, 1998.

[22] ——, "Higher-order sliding modes, differentiation and output-feedback control," *International Journal of Control*, vol. 76, no. 9-10, pp. 924–941, 2003.

Appendix D

Attachment Description and MATLAB code

In the ”Attachments. zip” file attached to this thesis is a folder named ”Code”, where all the MATLAB code and Simulink models can be found. Inside the code folder, there is two folders one named ”MSDS” and one named ”USM”, in each folder the code for each system can be found. There is also a folder named ”Simulation videos”. Inside that folder is two videos, one named $torpedo\_mode$ and one named $operation\_mode$. The one named $torpedo\_mode$ shows how the USM torpedo mode was simulated, and the one named $operation\_mode$ shows how the USM operation mode was simulated. There is also a file named ”Project report, Ida-Louise G. Borlaug. pdf”, which is my project report.

D.1 Mass-spring-damper system

D.1.1 Attachment description

The mass-spring-damper system folder is divided up in three folders, one for each algorithm. The code is organized as follows:

.m-files

- $find\_absolute\_max\_error.m$ - Finds the absolute maximum error for the error variables. Run one of the following models before running this file.

STA (Super-twisting algorithm)

- STA:
  - $MSDS\_STA.slx$ - Contains the implementation of the super-twisting algorithm and the mass-spring-damper system, where the actual values of x_1 and x_2 is used in the sliding surface.

- $run\_MSDS\_STA.m$ - Initializes the parameters, runs $MSDS\_STA.slx$ and plots the results.

- STA with HOSMO estimating x_2:
  - $MSDS\_STA\_HOSMO\_est\_x\_2.slx$ - Contains the implementation of the super-twisting algorithm with a state observer and the mass-spring-damper system, where the estimated value of x_2 is used in the sliding surface.
  - $run\_MSDS\_STA\_HOSMO\_est\_x\_2.m$ - Initializes the parameters, runs $MSDS\_STA\_HOSMO\_est\_x\_2.slx$ and plots the results.

- STA with HOSMO estimating x_1 and x_2:
  - $MSDS\_STA\_HOSMO\_est\_x\_1\_2.slx$ - Contains the implementation of the super-twisting algorithm with a state observer and the mass-spring-damper system, where the estimated value of x_1 and x_2 are used in the sliding surface.
  - $run\_MSDS\_STA\_HOSMO\_est\_x\_1\_2.m$ - Initializes the parameters, runs $MSDS\_STA\_HOSMO\_est\_x\_1\_2.slx$ and plots the results.

Nested 3rd-order SMC + differentiator (Nested third-order SMC + differentiator)

- Nested 3rd SMC with differentiator:
  - $MSDS\_3\_nested\_SMC.slx$ - Contains the implementation of the nested third-order sliding mode control algorithm with differentiator and the mass-spring-damper system, where the actual values of x_1 and x_2 is used in the sliding surface.
  - $run\_MSDS\_3\_nested\_SMC.m$ - Initializes the parameters, runs $MSDS\_3\_nested\_SMC.slx$ and plots the results.

- Nested 3rd SMC with diff and HOSMO estimating x_2:
  - $MSDS\_3\_nested\_SMC\_HOSMO\_est\_x\_2.slx$ - Contains the implementation of the nested third-order sliding mode control algorithm with differentiator, a state observer and the mass-spring-damper system, where the estimation of x_2 is used in the sliding surface.
  - $run\_MSDS\_3\_nested\_SMC\_HOSMO\_est\_x\_2.m$ - Initializes the parameters, runs $MSDS\_3\_nested\_SMC\_HOSMO\_est\_x\_2.slx$ and plots the results.

- Nested 3rd SMC with diff and HOSMO estimating x_1 and x_2:
  - $MSDS\_3\_nested\_SMC\_HOSMO\_est\_x\_1\_2.slx$ - Contains the implementation of the nested third-order sliding mode control algorithm with differentiator, a state observer and the mass-spring-damper system, where the estimation of x_1 and x_2 is used in the sliding surface.

- $run\_MSDS\_3\_nested\_SMC\_HOSMO\_est\_x\_1\_2.m$ - Initializes the parameters, runs $MSDS\_3\_nested\_SMC\_HOSMO\_est\_x\_1\_2.slx$ and plots the results.

Quasi 3rd-order SMC + differentiator (Quasi-continuous third-order SMC + differentiator)

- Quasi 3rd SMC with differentiator:
  - $MSDS\_3\_quasi\_SMC.slx$ - Contains the implementation of the quasi-continuous third-order sliding mode control algorithm with differentiator and the mass-spring-damper system, where the actual values of x_1 and x_2 is used in the sliding surface.
  - $run\_MSDS\_3\_quasi\_SMC.m$ - Initializes the parameters, runs $MSDS\_3\_quasi\_SMC.slx$ and plots the results.
- Quasi 3rd SMC with diff and HOSMO estimating x_2:
  - $MSDS\_3\_quasi\_SMC\_HOSMO\_est\_x\_2.slx$ - Contains the implementation of the quasi-continuous third-order sliding mode control algorithm with differentiator, a state observer and the mass-spring-damper system, where the estimation of x_2 is used in the sliding surface.
  - $run\_MSDS\_3\_quasi\_SMC\_HOSMO\_est\_x\_2.m$ - Initializes the parameters, runs $MSDS\_3\_quasi\_SMC\_HOSMO\_est\_x\_2.slx$ and plots the results.
- Quasi 3rd SMC with diff and HOSMO estimating x_1 and x_2:
  - $MSDS\_3\_quasi\_SMC\_HOSMO\_est\_x\_1\_2.slx$ - Contains the implementation of the quasi-continuous third-order sliding mode control algorithm with differentiator, a state observer and the mass-spring-damper system, where the estimation of x_1 and x_2 is used in the sliding surface.
  - $run\_MSDS\_3\_quasi\_SMC\_HOSMO\_est\_x\_1\_2.m$ - Initializes the parameters, runs $MSDS\_3\_quasi\_SMC\_HOSMO\_est\_x\_1\_2.slx$ and plots the results.

D.1.2 MATLAB code

STA:

run_MSDS_STA.m

```
%% Run super-twisting algorithm

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Sliding surface
c_1 = 1;

% Control gains
K = 100;

sim('MSDS_STA');

%% Plot

figure(1)
subplot(5,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(5,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(5,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(5,1,4);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])

subplot(5,1,5);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])
```

STA with HOSMO estimating x_2:

$run\_MSDS\_STA\_HOSMO\_est\_x\_2.m$

```
%% Run super-twisting algorithm with HOSMO to estimate x_2

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Observer gains
L = 16;
k_1 = 6*L^(1/3);
k_2 = 11*L^(1/2);
k_3 = 6*L;

% Sliding surface
c_1 = 1;

% Control gains
K = 75;

sim('MSDS_STA_HOSMO_est_x_2');

%% Plot

figure(1)
subplot(3,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(3,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(3,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

figure(2)
subplot(4,1,1);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])

subplot(4,1,2);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
```

```
xlim([0 50])

subplot(4,1,3);
plot(e_1.time,e_1.signals.values,'b','LineSmoothing','on')
title('Observer error e_1'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,4);
plot(e_2.time,e_2.signals.values,'b','LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
xlim([0 50])
```

STA with HOSMO estimating x_1 and x_2:

$run\_MSDS\_STA\_HOSMO\_est\_x\_1\_2.m$

```
%% Run super-twisting algorithm with HOSMO to estimate x_1 and x_2

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Observer gains
L = 16;
k_1 = 6*L^(1/3);
k_2 = 11*L^(1/2);
k_3 = 6*L;

% Sliding surface
c_1 = 1;

% Control gains
K = 75;

sim('MSDS_STA_HOSMO_est_x_1_2');

%% Plot
figure(1)
subplot(3,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(3,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(3,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
```

```
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

figure(2)
subplot(4,1,1);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])

subplot(4,1,2);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,3);
plot(e_1.time,e_1.signals.values,'b','LineSmoothing','on')
title('Observer error e_1'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,4);
plot(e_2.time,e_2.signals.values,'b','LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
xlim([0 50])
```

Nested 3rd SMC with differentiator:

run_MSDS_3_nested_SMC.m

```
%% Run nested third-order SMC with HOSMO to estimate x_2

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Sliding surface
c_1 = 1;

% Differentiator parameters
L_D = 20;
lambda_2 = 3;
lambda_1 = 1.5;
lambda_0 = 1.1;

% Control paramters

beta_1 = 1;
beta_2 = 2;

alpha = 40;

sim('MSDS_3_nested_SMC');

%% Plot
```

```
figure(1)
subplot(3,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(3,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(3,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

figure(2)
subplot(3,1,1);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])

subplot(3,1,2);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(3,1,3);
plot(z_0.time,z_0.signals.values,'r','LineSmoothing','on')
hold on
plot(z_1.time,z_1.signals.values,'b','LineSmoothing','on')
hold on
plot(z_2.time,z_2.signals.values,'g','LineSmoothing','on')
title('Differentiator error z_0, z_1 and z_2'); xlabel('Time [s]');
legend('z_0','z_1','z_2');
xlim([0 50])
```

Nested 3rd SMC with diff and HOSMO estimating x_2:

$run\_MSDS\_3\_nested\_SMC\_HOSMO\_est\_x\_2.m$

```
%% Run nested third-order SMC with HOSMO to estimate x_2

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Observer gains
L_SO = 16;
k_1 = 6*L_SO^(1/3);
```

```
k_2 = 11*L_SO^(1/2);
k_3 = 6*L_SO;

% Sliding surface
c_1 = 1;

% Differentiator parameters
L_D = 20;
lambda_2 = 3;
lambda_1 = 1.5;
lambda_0 = 1.1;

% Control paramters

beta_1 = 1;
beta_2 = 2;

alpha = 28;

sim('MSDS_3_nested_SMC_HOSMO_est_x_2');

%% Plot
figure(1)
subplot(4,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(4,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(4,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(4,1,4);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])

figure(2)
subplot(4,1,1);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,2);
plot(e_1.time,e_1.signals.values,'b','LineSmoothing','on')
title('Observer error e_1'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])
```

```
subplot(4,1,3);
plot(e_2.time,e_2.signals.values,'b','LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
xlim([0 50])

subplot(4,1,4);
plot(z_0.time,z_0.signals.values,'r','LineSmoothing','on')
hold on
plot(z_1.time,z_1.signals.values,'b','LineSmoothing','on')
hold on
plot(z_2.time,z_2.signals.values,'g','LineSmoothing','on')
title('Differentiator error z_0, z_1 and z_2'); xlabel('Time [s]');
legend('z_0','z_1','z_2');
xlim([0 50])
```

Nested 3rd SMC with diff and HOSMO estimating x_1 and x_2:

$run\_MSDS\_3\_nested\_SMC\_HOSMO\_est\_x\_1\_2.m$

```
%% Run nested third-order SMC with HOSMO to estimate x_2

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Observer gains
L_SO = 16;
k_1 = 6*L_SO^(1/3);
k_2 = 11*L_SO^(1/2);
k_3 = 6*L_SO;

% Sliding surface
c_1 = 1;

% Differentiator parameters
L_D = 20;
lambda_2 = 3;
lambda_1 = 1.5;
lambda_0 = 1.1;

% Control paramters

beta_1 = 1;
beta_2 = 2;

alpha = 28;

sim('MSDS_3_nested_SMC_HOSMO_est_x_2');

%% Plot
figure(1)
subplot(4,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
```

```
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(4,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(4,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(4,1,4);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])

figure(2)
subplot(4,1,1);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,2);
plot(e_1.time,e_1.signals.values,'b','LineSmoothing','on')
title('Observer error e_1'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,3);
plot(e_2.time,e_2.signals.values,'b','LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
xlim([0 50])

subplot(4,1,4);
plot(z_0.time,z_0.signals.values,'r','LineSmoothing','on')
hold on
plot(z_1.time,z_1.signals.values,'b','LineSmoothing','on')
hold on
plot(z_2.time,z_2.signals.values,'g','LineSmoothing','on')
title('Differentiator error z_0, z_1 and z_2'); xlabel('Time [s]');
legend('z_0','z_1','z_2');
xlim([0 50])
```

Quasi 3rd SMC with differentiator:

$run\_MSDS\_3\_quasi\_SMC.m$

```
%% Run quasi-continuous third-order SMC with HOSMO to estimate x_2

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Sliding surface
c_1 = 1;

% Differentiator parameters
L_D = 1;
lambda_2 = 3;
lambda_1 = 1.5;
lambda_0 = 1.1;

% Control paramters

beta_1 = 1;
beta_2 = 2;

alpha = 40;

sim('MSDS_3_quasi_SMC');

%% Plot

figure(1)
subplot(3,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(3,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(3,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

figure(2)
subplot(3,1,1);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])
```

```
subplot(3,1,2);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(3,1,3);
plot(z_0.time,z_0.signals.values,'r','LineSmoothing','on')
hold on
plot(z_1.time,z_1.signals.values,'b','LineSmoothing','on')
hold on
plot(z_2.time,z_2.signals.values,'g','LineSmoothing','on')
title('Differentiator error z_0, z_1 and z_2'); xlabel('Time [s]');
legend('z_0','z_1','z_2');
xlim([0 50])
```

Quasi 3rd SMC with diff and HOSMO estimating x_2:

$run\_MSDS\_3\_quasi\_SMC\_HOSMO\_est\_x\_2.m$

```
%% Run quasi-continuous third-order SMC with HOSMO to estimate x_2

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Observer gains
L_SO = 16;
k_1 = 6*L_SO^(1/3);
k_2 = 11*L_SO^(1/2);
k_3 = 6*L_SO;

% Sliding surface
c_1 = 1;

% Differentiator parameters
L_D = 1;
lambda_2 = 3;
lambda_1 = 1.5;
lambda_0 = 1.1;

% Control paramters

beta_1 = 1;
beta_2 = 2;

alpha = 30;

sim('MSDS_3_quasi_SMC_HOSMO_est_x_2');

%% Plot

figure(1)
```

```
subplot(4,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(4,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(4,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(4,1,4);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])

figure(2)
subplot(4,1,1);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,2);
plot(e_1.time,e_1.signals.values,'b','LineSmoothing','on')
title('Observer error e_1'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,3);
plot(e_2.time,e_2.signals.values,'b','LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
xlim([0 50])

subplot(4,1,4);
plot(z_0.time,z_0.signals.values,'r','LineSmoothing','on')
hold on
plot(z_1.time,z_1.signals.values,'b','LineSmoothing','on')
hold on
plot(z_2.time,z_2.signals.values,'g','LineSmoothing','on')
title('Differentiator error z_0, z_1 and z_2'); xlabel('Time [s]');
legend('z_0','z_1','z_2');
xlim([0 50])
```

Quasi 3rd SMC with diff and HOSMO estimating x_1 and x_2:

$run\_MSDS\_3\_quasi\_SMC\_HOSMO\_est\_x\_1\_2.m$

```
%% Run quasi-continuous third-order SMC with HOSMO to estimate x_2

% Model parameter values

m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Observer gains
L_SO = 16;
k_1 = 6*L_SO^(1/3);
k_2 = 11*L_SO^(1/2);
k_3 = 6*L_SO;

% Sliding surface
c_1 = 1;

% Differentiator parameters
L_D = 1;
lambda_2 = 3;
lambda_1 = 1.5;
lambda_0 = 1.1;

% Control paramters

beta_1 = 1;
beta_2 = 2;

alpha = 30;

sim('MSDS_3_quasi_SMC_HOSMO_est_x_2');

%% Plot

figure(1)
subplot(4,1,1);
plot(x_1_des.time,x_1_des.signals.values,'r','LineSmoothing','on')
hold on
plot(x_1.time,x_1.signals.values,'b','LineSmoothing','on')
title('X_{des} with X'); xlabel('Time [s]');ylabel('Position of mass [m]');
legend('x_{des}','x');
xlim([0 50])

subplot(4,1,2);
plot(disturbance.time,disturbance.signals.values,'b','LineSmoothing','on')
title('Disturbance'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])

subplot(4,1,3);
plot(control_input.time,control_input.signals.values,'b','LineSmoothing','on')
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
xlim([0 50])
```

```
subplot(4,1,4);
plot(sliding_var.time,sliding_var.signals.values,'b','LineSmoothing','on')
title('Sliding variable \sigma'); xlabel('Time [s]');
xlim([0 50])

figure(2)
subplot(4,1,1);
plot(x_1.time,x_1.signals.values-x_1_des.signals.values,'b','LineSmoothing','on')
title('Position error'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,2);
plot(e_1.time,e_1.signals.values,'b','LineSmoothing','on')
title('Observer error e_1'); xlabel('Time [s]');ylabel('[m]');
xlim([0 50])

subplot(4,1,3);
plot(e_2.time,e_2.signals.values,'b','LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
xlim([0 50])

subplot(4,1,4);
plot(z_0.time,z_0.signals.values,'r','LineSmoothing','on')
hold on
plot(z_1.time,z_1.signals.values,'b','LineSmoothing','on')
hold on
plot(z_2.time,z_2.signals.values,'g','LineSmoothing','on')
title('Differentiator error z_0, z_1 and z_2'); xlabel('Time [s]');
legend('z_0','z_1','z_2');
xlim([0 50])
```

Absolute maximum error:

find_absolute_max_error.m

```
%% Find absoulte maximum error
%Position error

max_error_position = max(abs(x_1.signals.values(1500000:5000001)-
x_1_des.signals.values(1500000:5000001)))

%% Observer error

max_error_e_1 = max(abs(e_1.signals.values(1500000:5000001)))
max_error_e_2 = max(abs(e_2.signals.values(1500000:5000001)))

%% Only for HOSM
%Differentiator error

max_error_z_0 = max(abs(z_0.signals.values(1500000:5000001)))
max_error_z_1 = max(abs(z_1.signals.values(1500000:5000001)))
max_error_z_2 = max(abs(z_2.signals.values(1500000:5000001)))
```

D.2 Underwater swimming manipulator

D.2.1 Attachment description

The files that are described and attached here, are the files where changes can be made to able to run the different algorithms that are implemented. The code for the USM is organized as follows:

.m-files

- $startSimulation\_ver3.m$ - Starts the simulation of the USM. Change the variable ode, to change between ODE5 and ODE23tb solver. The fixed step length for the ODE5 solver and the relative and absolute error for the ODE23tb can also be changed her.
- $calculate\_desired\_forces\_moments.m$ - Calculates and returns the desired forces and moments, by calculating the sliding surface, state observer, differentiator and the control input. To switch between the different algorithms, change the variable con. To switch between torpedo mode and operation mode change the variable $mode$ (remember to change $mode$ in $calculate\_u\_lateral\_undulation.m$ as well). The variable $estimat$ can also be changed to choose witch sliding surface that is to be used in the simulation.
- $calculate\_u\_lateral\_undulation.m$ - Calculates and returns the actuator forces. Change the variable $mode$ to switch between torpedo mode and operation mode.
- $plot\_simulations.m$ - Plots the simulation results. Run it after $startSimulation\_ver3.m$.
- $find\_absolute\_max\_error.m$ - Finds the absolute maximum error. Run it after $startSimulation\_ver3.m$.

D.2.2 MATLAB code

Start simulation:

$startSimulation\_ver3.m$

```
%
% Starts the snake robot simulation.
%

global visualize_motion previous_draw_time t_previous waypoints ...
    WP_switch_times t_sim theta_pathframe_sim p_pathframe_sim ...
    vt_sim psi_path_sim z_sim z_dot_sim f_thr_sim tau_d_sim  ...
    T_allocation_sim phi_sim phi_ref_sim...
    fTx_sim fTy_sim fx_sim fy_sim orientation_ref_sim...
    orientation_sim px_sim py_sim phi_offset_sim ...
    sigma_f_sim F_CM_sim error_pos_CM_sim p_CM_d_sim ...
    observer_error_e_1_sim observer_error_e_2_sim differentiator_error
```

```
% The stop time for the simulation.
stop_time = 40; % Straight motion
%stop_time = 200; % Turning motion

% Initializes controller parameters.
initControllerParameters;

% Initializes model parameters.
initModelParameters;


disp('Starting simulation...')

% Matlab function used to measure how long the simulation takes
tic

% The simulation time of the previous time step
t_previous = 0;

% Constructs the initial values for the state vector.
v0 = [theta0 ; p_CM0 ; theta0_dot ; p_CM0_dot ; u_CM0_2 ; x_estimat_dot;
    z_dot;];

% Starts the snake robot visualization
if visualize_motion
    % The time of the previously drawn sample
    previous_draw_time = 0;

    % Draws the initial position of the snake robot
    drawSnakeRobot(0, theta0, p_CM0, 0);

    % Create a slider with callback function to change \omega
%     f33 = figure(33);
%     set(f33,'Position', [800 600 130 30]);
%     sld = uicontrol('Style', 'slider',...
%         'Min',20,'Max',100,'Value',70,...
%         'Position', [0 0 120 20],...
%         'Callback', @changeOmega);

    % Waits for the figure window to display
    pause(0.1);
end
%return

if 1

ode = 5;
% ode23tb solver
if ode == 23
    options = odeset('Events', 'off', 'RelTol', 6e-2, 'AbsTol', 6e-2);
    [t, v] = ode23tb('calculate_v_dot', [0 stop_time], v0, options);
end

% ode5 solver
if ode == 5
```

```
    nstep = 4000000;
    tspan = linspace(0,stop_time,nstep);
    v = ode5('calculate_v_dot',tspan,v0);
end

disp('Simulation complete...')

% Displays how long the simulation took to complete
toc

disp(['Simulated time: ' num2str(t(end, 1)) 's'])

% Extracts the states from the state vector.
theta        = v(:, 1:n);
p_CM         = v(:, n+1:n+2);
theta_dot    = v(:, n+3:2*n+2);
p_CM_dot     = v(:, 2*n+3:2*n+4);
%phi          = v(:, 2*n+5:3*n+4);
%phi_dot      = v(:, 3*n+5:4*n+4);
%phi_o        = v(:, 4*n+5);
%psi_ref      = v(:, 4*n+8);

end
%return
```

Calculate desired forces and moments:

calculate_desired_forces_moments.m

```
function [tau_d, u_CM_2_dot,x_estimat_dot,differentiator_dot] = calculate_
desired_forces_moments(t, theta,theta_dot, p_CM, p_CM_dot,u_CM_2,x_estimat,
differentiator)
% Simple PD controller that returns the desired generalized forces and moments


global n p_CM_desired p_CM_dot_desired heading_desired ...
    heading_dot_desired t_vector sigma_f F_CM error_pos_CM p_CM_d e_1 e_2

%Paramters:
m_t = 16*0.6597; %total mass

if t <= 0
    tau_d = [0;0;0];
    u_CM_2_dot = zeros(2,1);
    x_estimat_dot = zeros(6,1); % x_1_dot = (1:2,1), x_2_dot = (3:4,1),
    %x_3_dot = (5:6,1);
    differentiator_dot = zeros(6,1); % z_1_dot = (1:2,1), z_2_dot = (3:4,1),
    %z_3_dot = (5:6,1); Differentiator
end

%Choose controller: con = 1 (super-twisting), con = 2 (Nested third-order
%SMC), con = 3 (Quasi-continuous third-order SMC), con = 4 (PD-controller)
con = 1;

%Choose state observer: estimat = 0 (No state observer)
```

```
%estimat = 1 (State observer, estimate velocity),
%estimat = 2 (Full state observer, estimate velocity and position).
estimat = 1;

%Choose mode: torpedo mode = 1, operation mode = 2.
mode = 1;

% Calculates the desired thrust vector in 3 DOF (transport mode)
if t > 0
    %Control paramters
    F_CM = zeros(2,1);

    Kp_theta = 6;

    %Only for super-twisting
    u_CM_1 = zeros(2,1);
    u_CM_2_dot = zeros(2,1);
    STA_CM = zeros(2,1);

    %Only for third-order SMC
    third_C = zeros(2,1);
    u_1 = zeros(2,1);
    u_2 = zeros(2,1);

    %State observer
    z_1 = zeros(2,1);
    z_2 = zeros(2,1);
    z_3 = zeros(2,1);
    e_1 = zeros(2,1);
    e_2 = zeros(2,1);
    x_estimat_dot = zeros(6,1);

    % Differentiator
    v_0 = zeros(2,1);
    v_1 = zeros(2,1);
    v_2 = zeros(2,1);
    differentiator_dot = zeros(6,1);

    if mode == 1
        % Tordpedo mode
        Delta=2.24;
        heading_d = -atan2(p_CM(2,1),Delta);
    else
        % Operation mode
        if (t <= 10)
            heading_d = sin(pi/2);
        end
        if (t > 10 && t <= 20) || (t > 30 && t <= 40)
            heading_d = sin(pi);
        end
        if (t > 20 && t <= 30)
            heading_d = sin(3*pi/2);
        end
    end

    p_CM_dot_d = interp1(t_vector,p_CM_dot_desired',t,'spline')';
    p_CM_d = interp1(t_vector,p_CM_desired',t,'spline')';
```

```
heading = theta(n);

%Observer dynamics
if estimat ~= 0
    L_SO = 50;
    k_1 = 6*L_SO^(1/3);
    k_2 = 11*L_SO^(1/2);
    k_3 = 6*L_SO;

    e_1 = p_CM - x_estimat(1:2,1);
    e_2 = p_CM_dot - x_estimat(3:4,1);

    for i = 1:2
        z_1(i) = k_1*abs(e_1(i))^(2/3)*sign(e_1(i));
        z_2(i) = k_2*abs(e_1(i))^(1/3)*sign(e_1(i));
        z_3(i) = k_3*sign(e_1(i));
    end

    x_estimat_dot(5:6,1) = z_3;
    x_estimat_dot(3:4,1) = x_estimat(5:6,1) + z_2 + (1/m_t)*(F_CM);
    x_estimat_dot(1:2,1) = x_estimat(3:4,1) + z_1;
end

% Error variables
error_pos_CM = p_CM - p_CM_d;
error_vel_CM = p_CM_dot - p_CM_dot_d;
error_heading = heading_d - heading;

error_pos_CM_estimat = x_estimat(1:2,1) - p_CM_d;
error_vel_CM_estimat = x_estimat(3:4,1) - p_CM_dot_d;


% Create sliding surfaces
c_1 = 1;
if estimat == 0
    sigma_f = error_vel_CM + error_pos_CM;
elseif estimat == 1
    sigma_f = error_vel_CM_estimat + error_pos_CM;
else
    sigma_f = error_vel_CM_estimat + c_1*error_pos_CM_estimat;
end

%Differentiator dynamics
if (con == 2) || (con == 3)
    if con == 2
        L_D = 0.1;
    else
        L_D = 1;
    end
    lambda_0 = 1.1;
    lambda_1 = 1.5;
    lambda_2 = 3;

    z_0 = differentiator(1:2,1);
    z_1 = differentiator(3:4,1);
    z_2 = differentiator(5:6,1);
```

```
        for i = 1:2
            v_0(i) = -lambda_2*(L_D)^(1/3)*(abs(z_0(i)-sigma_f(i)))^(2/3)*
            sign(z_0(i)-sigma_f(i))+z_1(i);
            v_1(i) = -lambda_1*(L_D)^(1/2)*(abs(z_1(i)-v_0(i)))^(1/2)*
            sign(z_1(i)-v_0(i))+z_2(i);
            v_2(i) = -lambda_0*L_D*sign(z_2(i)-v_1(i));
        end

        differentiator_dot(5:6,1) = v_2;
        differentiator_dot(3:4,1) = v_1;
        differentiator_dot(1:2,1) = v_0;
     end

    if con == 1 %Super-Twisting controller
        K_f = 10;

        for i = 1:2
            u_CM_1(i) = -1.5*sqrt(K_f)*sqrt(abs(sigma_f(i)))*sign(sigma_f(i));
            u_CM_2_dot(i) = -1.1*K_f*sign(sigma_f(i));
            STA_CM(i) = u_CM_1(i) + u_CM_2(i);
        end
        u_C = STA_CM;

    elseif  con == 2 %Nested third-order SMC
        alpha = 10;
        beta_2 = 1;
        beta_1 = 1;

        z_0 = differentiator(1:2,1);
        z_1 = differentiator(3:4,1);
        z_2 = differentiator(5:6,1);

        for i = 1:2
            third_C(i) = -alpha*(sign(z_2(i)+beta_2*((abs(z_1(i)))^3+
            (abs(z_0(i)))^2)^(1/6)*sign(z_1(i)+beta_1*((abs(z_0(i)))^(2/3)*
            sign(z_0(i))))));
        end

        u_C = third_C;

    elseif  con == 3 %Quasi-continuous third-order SMC
        alpha = 10;
        beta_2 = 2;
        beta_1 = 1;

        z_0 = differentiator(1:2,1);
        z_1 = differentiator(3:4,1);
        z_2 = differentiator(5:6,1);

        for i = 1:2
            u_1(i) = z_2(i) + beta_2*(abs(z_1(i))+abs(z_0(i))^(2/3))^(-1/2)*
            (z_1(i)+beta_1*abs(z_0(i))^(2/3)*sign(z_0(i)));
            u_2(i) = abs(z_2(i))+beta_2*(abs(z_1(i))+abs(z_0(i))^(2/3))^(1/2);

            third_C(i)= -alpha*(u_1(i)/u_2(i));
        end
        if t < 0.01
            u_C = zeros(2,1);
        else
            u_C = third_C;
        end

    else %PD-controller
        Kp_pos = 10; %0.6
        Kd_pos = 0.1; %0.06;

        if estimat == 0
            u_C = Kd_pos*(-1)*error_vel_CM + Kp_pos*(-1)*error_pos_CM;
        elseif estimat == 1
            u_C = Kd_pos*(-1)*error_vel_CM_estimat + Kp_pos*(-1)*
            error_pos_CM;
        else
            u_C = Kd_pos*(-1)*error_vel_CM_estimat + Kp_pos*(-1)*
            error_pos_CM_estimat;
        end
    end
    if estimat == 0
        F_CM = m_t*(u_C);
    elseif estimat == 1
        if con == 3
            if t < 0.01
                F_CM = zeros(2,1);
            else
                F_CM = (u_C - c_1*x_estimat(3:4,1) - x_estimat(5:6,1) - z_2);
            end
        else
            F_CM = m_t*(u_C - c_1*x_estimat(3:4,1) - x_estimat(5:6,1) - z_2);
        end
    else
        if con == 3
            if t < 0.01
                F_CM = zeros(2,1);
            else
                F_CM = (u_C - c_1*x_estimat(3:4,1) - x_estimat(5:6,1) - z_2
                - c_1*z_1);
            end
        else
            F_CM = m_t*(u_C - c_1*x_estimat(3:4,1) - x_estimat(5:6,1) - z_2
            - c_1*z_1);
        end
    end
    T_CM = min(Kp_theta*error_heading,100);
    tau_d = [F_CM; T_CM];
end

end
```

Calculate actuator forces:

calculate_u_lateral_undulation.m

```
function [u, phi_ref] = calculate_u_lateral_undulation(t, phi, phi_dot,
theta,p_CM, p_CM_dot)
%
% Calculates and returns the actuator forces. This is the joint controller
% of the snake robot.
%

global n Kp_joint Kd_joint alpha omega delta heading phi_offset heading_ref

k_psi = 1*0.8;

%Choose mode: torpedo mode = 1, operation mode = 2
mode = 1;

if mode == 1
    % Tordpedo mode
    Delta=2.24;
    heading_ref = -atan2(p_CM(2,1),Delta);
else
    % Operation mode
    if (t <= 10)
        heading_ref = sin(pi/2);
    end
    if (t > 10 && t <= 20) || (t > 30 && t <= 40)
        heading_ref = sin(pi);
    end
    if (t > 20 && t <= 30)
        heading_ref = sin(3*pi/2);
    end
end

% Here you choose whether you want the undulation pattern to follow a
% line-of-sight reference heading, and if you want undulation or not
heading = theta(n);
phi_offset = k_psi * (heading_ref - heading);

if mode == 1
    alpha = 0;  % No lateral undulation
end

% Calculates references for joint angles.
phi_ref = zeros(n-1, 1);
for i = 1:n-1
    phi_ref(i, 1) = alpha*sin(omega*t + (i-1)*delta) - phi_offset;
end

% Calculates references for joint velocities.
phi_ref_d = zeros(n-1, 1);
for i = 1:n-1
    phi_ref_d(i, 1) = alpha*omega*cos(omega*t + (i-1)*delta);
end
% Calculates references for joint accelerations.
```

```
phi_ref_dd = zeros(n-1, 1);
for i = 1:n-1
    phi_ref_dd(i, 1) = -alpha*omega^2*sin(omega*t + (i-1)*delta);
end

% Calculates the actuator forces.
u = zeros(n-1, 1);
for i = 1:n-1
    error = phi_ref(i, 1) - phi(i, 1);
    error_d = phi_ref_d(i, 1) - phi_dot(i, 1);
    %u(i, 1) = Kp_joint*error - Kd_joint*phi_dot(i, 1);
    u(i, 1) = phi_ref_dd(i, 1) + Kd_joint*error_d + Kp_joint*error;
end
```

Plot simulations

plot_simulations.m

```
%% Plot for STA
figure(1)
subplot(3,1,1);
plot(t_sim,p_CM_d_sim(:,1),'r', 'LineSmoothing','on')
hold on
plot(t_sim,px_sim,'b', 'LineSmoothing','on')
title('x_{des} with x');xlabel('Time [s]');ylabel('p_x [m]');
legend('x_{des}','x');

subplot(3,1,2);
plot(t_sim,p_CM_d_sim(:,2),'r', 'LineSmoothing','on')
hold on
plot(t_sim,py_sim,'b', 'LineSmoothing','on')
title('y_{des} with y');xlabel('Time [s]');ylabel('p_y [m]');
legend('y_{des}','y');

subplot(3,1,3);
plot(t_sim,F_CM_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,F_CM_sim(:,2),'r', 'LineSmoothing','on')
title('Control input u');xlabel('Time [s]');ylabel('Force [N]');
legend('x','y');

figure(2)
subplot(4,1,1);
plot(t_sim,sigma_f_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,sigma_f_sim(:,2),'r', 'LineSmoothing','on')
title('Sliding variable \sigma');xlabel('Time [s]');
legend('x','y');

subplot(4,1,2);
plot(t_sim,error_pos_CM_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,error_pos_CM_sim(:,2),'r', 'LineSmoothing','on')
title('Position error');xlabel('Time [s]');ylabel('[m]');
legend('x','y');
```

```
subplot(4,1,3);
plot(t_sim,observer_error_e_1_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,observer_error_e_1_sim(:,2),'r', 'LineSmoothing','on')
title('Observer error e_1');xlabel('Time [s]');ylabel('[m]');
legend('x','y');

subplot(4,1,4);
plot(t_sim,observer_error_e_2_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,observer_error_e_2_sim(:,2),'r', 'LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
legend('x','y');


%% Plot for the third-order algorithms
figure(1)
subplot(4,1,1);
plot(t_sim,p_CM_d_sim(:,1),'r', 'LineSmoothing','on')
hold on
plot(t_sim,px_sim,'b', 'LineSmoothing','on')
title('x_{des} with x');xlabel('Time [s]');ylabel('p_x [m]');
legend('x_{des}','x');

subplot(4,1,2);
plot(t_sim,p_CM_d_sim(:,2),'r', 'LineSmoothing','on')
hold on
plot(t_sim,py_sim,'b', 'LineSmoothing','on')
title('y_{des} with y');xlabel('Time [s]');ylabel('p_y [m]');
legend('y_{des}','y');

subplot(4,1,3);
plot(t_sim,F_CM_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,F_CM_sim(:,2),'r', 'LineSmoothing','on')
title('Control input u');xlabel('Time [s]');ylabel('Force [N]');
legend('x','y');

subplot(4,1,4);
plot(t_sim,sigma_f_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,sigma_f_sim(:,2),'r', 'LineSmoothing','on')
title('Sliding variable \sigma');xlabel('Time [s]');
legend('x','y');

figure(2)
subplot(5,1,1);
plot(t_sim,error_pos_CM_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,error_pos_CM_sim(:,2),'r', 'LineSmoothing','on')
title('Position error');xlabel('Time [s]');ylabel('[m]');
legend('x','y');

subplot(5,1,2);
plot(t_sim,observer_error_e_1_sim(:,1),'b', 'LineSmoothing','on')
hold on
```

```
plot(t_sim,observer_error_e_1_sim(:,2),'r', 'LineSmoothing','on')
title('Observer error e_1');xlabel('Time [s]');ylabel('[m]');
legend('x','y');

subplot(5,1,3);
plot(t_sim,observer_error_e_2_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,observer_error_e_2_sim(:,2),'r', 'LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
legend('x','y');

subplot(5,1,4);
plot(t_sim,differentiator_error(:,1),'r', 'LineSmoothing','on')
hold on
plot(t_sim,differentiator_error(:,3),'b', 'LineSmoothing','on')
hold on
plot(t_sim,differentiator_error(:,5),'g', 'LineSmoothing','on')
title('Differentiator error z_0, z_1 and z_2 for x'); xlabel('Time [s]');
legend('z_0','z_1','z_2');

subplot(5,1,5);
plot(t_sim,differentiator_error(:,2),'r', 'LineSmoothing','on')
hold on
plot(t_sim,differentiator_error(:,4),'b', 'LineSmoothing','on')
hold on
plot(t_sim,differentiator_error(:,6),'g', 'LineSmoothing','on')
title('Differentiator error z_0, z_1 and z_2 for y'); xlabel('Time [s]');
legend('z_0','z_1','z_2');


%% Plot for PD-controller
figure(1)
subplot(3,1,1);
plot(t_sim,p_CM_d_sim(:,1),'r', 'LineSmoothing','on')
hold on
plot(t_sim,px_sim,'b', 'LineSmoothing','on')
title('x_{des} with x');ylabel('p_x [m]');
legend('x_{des}','x');

subplot(3,1,2);
plot(t_sim,p_CM_d_sim(:,2),'r', 'LineSmoothing','on')
hold on
plot(t_sim,py_sim,'b', 'LineSmoothing','on')
title('y_{des} with y');ylabel('p_y [m]');
legend('y_{des}','y');

subplot(3,1,3);
plot(t_sim,F_CM_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,F_CM_sim(:,2),'r', 'LineSmoothing','on')
title('Control input u');ylabel('Force [N]');
legend('x','y');

figure(2)
subplot(3,1,1);
plot(t_sim,error_pos_CM_sim(:,1),'b', 'LineSmoothing','on')
hold on
```

```
plot(t_sim,error_pos_CM_sim(:,2),'r', 'LineSmoothing','on')
title('Position error');xlabel('Time [s]');ylabel('[m]');
legend('x','y');

subplot(3,1,2);
plot(t_sim,observer_error_e_1_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,observer_error_e_1_sim(:,2),'r', 'LineSmoothing','on')
title('Observer error e_1');xlabel('Time [s]');ylabel('[m]');
legend('x','y');

subplot(3,1,3);
plot(t_sim,observer_error_e_2_sim(:,1),'b', 'LineSmoothing','on')
hold on
plot(t_sim,observer_error_e_2_sim(:,2),'r', 'LineSmoothing','on')
title('Observer error e_2'); xlabel('Time [s]');ylabel('[m/s]');
legend('x','y');
```

Absolute maximum error:

$find\_absolute\_max\_error.m$

```
%% Find absolute maximum error
% Position error

interval = 15000:39999;
max_error_x = max(abs(error_pos_CM_sim(interval,1)))
max_error_y = max(abs(error_pos_CM_sim(interval,2)))

%% Observer error
max_e_1_x = max(abs(observer_error_e_1_sim(interval,1)))
max_e_1_y = max(abs(observer_error_e_1_sim(interval,2)))
max_e_2_x = max(abs(observer_error_e_2_sim(interval,1)))
max_e_2_y = max(abs(observer_error_e_2_sim(interval,2)))

%% Only for the third-order algorithms
% Differentiator error

interval = 15000:39999;
max_z_0_x = max(abs(differentiator_error(interval,1)))
max_z_0_y = max(abs(differentiator_error(interval,2)))
max_z_1_x = max(abs(differentiator_error(interval,3)))
max_z_1_y = max(abs(differentiator_error(interval,4)))
max_z_2_x = max(abs(differentiator_error(interval,5)))
max_z_2_y = max(abs(differentiator_error(interval,6)))
```