Higher Order Sliding Mode Control

A literature study and application to underwater snake robots

Ida-Louise G. Borlaug

December 2016

PROJECT REPORT

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Supervisor: Professor Jan Tommy Gravdahl

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

PROSJEKTOPPGAVE

Kandidatens navn: Ida-Louise G. Borlaug

Fag: Teknisk Kybernetikk

Oppgavens tittel: **Higher order sliding mode control: a literature study 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, sliding mode control is to be studied.

Assignment:

1. Perform a detailed literature study covering sliding mode control. First, second and higher order SM is to be researched.
2. Chose the appropriate methods found in 1. and apply them to a second order test-system. Perform simulations and compare the results.
3. Test the SM algorithms selected in 2. for control of an underwater swimming manipulator with thrusters. Use the model presented in [1].

Besvarelsen leveres innen: 21. Desember 2016

[1] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, Sliding mode control of Underwater Swimming Manipulators with thrusters, submitted to the 2017 IEEE International conference on Robotics and Automation.

Trondheim, Autumn, 2016

Tommy Gravdahl
Faglærer

Abstract

In this 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. The first-order relay controller, the first-order saturation controller, the super-twisting algorithm, with and without adaptive gains, are tested on two different systems. A test system and an underwater swimming manipulator, a snake-like, multi-articulated, underwater robot equipped with thrusters. The control objective is to make the state trajectory follow a pre-defined path. The test system is a mass-spring-damper system, and it 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 best results was achieved by using the second-order algorithms. To the author's knowledge, second-order algorithms have not been previously tested on an underwater swimming manipulator. Therefore, it was interesting to see if the second-order algorithms performed better than the first-order algorithms and a regular PD controller. As expected the second-order algorithms gave the best results. Of the two second-order algorithms, the super-twisting algorithm with adaptive gains gave the best result for practical use, as there is no problem with finding the optimal gain.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

SMC Sliding mode control

STA Super-twisting algorithm

HOSM Higher-order sliding mode

VSS Variable structure systems

SISO Single-input-single-output

MIMO Multi-input-multi-output

BIBO Bounded-input, bounded-output

USM Underwater swimming manipulator

LOS Line-of-sight

CM Centre of mass

List of Symbols

Symbol	Description	Vector
x	Position of the mass	
m	Mass [kg]	
c	Damping coefficient [N s/m]	
k	Spring constant [N/m]	
u	Control input	
$d(t)$	Time-varying disturbance	
y	Output variable mass-spring-damper system	
y_{des}	Desired position mass-spring-damper system	
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 $i-1$	$h_x, h_y \in R^{n-1}$
e	Error variable	
σ	Sliding surface	
K	Control gain	
α, β	Adaptive control gains	
$\epsilon, \lambda, \gamma_1, \omega_1$	Arbitrary positive constants	

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 report is an attempt to gather information about the research done in the field related to SMC. Based on the literature study some higher-order SMC algorithm will be chosen for further research. The algorithms will be tested on one test system and an underwater swimming manipulator (USM), a snake-like, multi-articulated, underwater robot equipped with thrusters. The test system is a mass-spring-damper system. It will be used to easily see the advantages and disadvantages with the different SMC algorithms. The USM has a complex control design problem. That 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. There have been cases where SMC has been used to control a snake-like manipulator. In Rezapour et al. (2014), a relay inspired controller was used for a snake robot, and in Sverdrup-Thygeson et al. (2016b), the saturation controller was used for an USM. To the authors knowledge no second or higher-order SMC algorithm has been tested on an USM.

This report is divided into 6 chapters. Chapter 2 presents a detailed literature study covering sliding mode control. SMC is presented in general, and first, second and higher-order SMC is researched. Chapter 3 presents why the different SMC algorithms were chosen for further research, the test system, i.e. the mass-spring-damper system, its control input design and the results from each SMC algorithm. Chapter 4 presents the USM, its control input design and the results from each SMC algorithm. In chapter 5 the results for each algorithm is discussed and compared. A conclusion and future work is found in chapter 6.

Chapter 2

Literature Study: Sliding Mode Control

SMC systems are designed to drive the system state trajectories onto a particular surface in the state space, named sliding surface, in finite time. Once the sliding surface is reached, sliding mode control keeps the states on the close neighbourhood of the sliding surface for all future time. Hence the sliding mode control 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 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 method.

In the following chapters there will be given an introduction to SMC and an explanation of different types of SMC algorithms that have been proposed over the years. There will be given detailed explanations of first, second and higher-order sliding modes. Remark that the symbols used to represent the different variables can be somewhat different from one algorithm to another. This is because the variables used will be the ones that were used in the article where the algorithm first was presented. The symbols used in this chapter will therefore not be found in the list of abbreviations.

Since SMC is such a big field, it was impossible to read every relevant article. The selected articles that have been used as a basis for the literature study are therefore survey papers, articles that emphasize the different algorithms with good mathematical explanation where the stabilization of the algorithm was shown or where the different algorithms were compared.

2.1 General sliding mode control

2.1.1 Basics

In Variable Structure Control: A Survey, Hung et al. (1993), the basic idea behind SMC is illustrated by using a second-order system

$$\dot{x} = y \tag{2.1}$$

$$\dot{y} = 2y - x + u \tag{2.2}$$

$$u = -\psi x \tag{2.3}$$

where

$$\begin{aligned} \psi &= 4 \quad \text{when} \quad s(x,y) > 0 \\ &= -4 \quad \text{when} \quad s(x,y) < 0 \end{aligned} \tag{2.4}$$

and

$$s(x,y) = x\sigma, \qquad \sigma = 0.5x + y \tag{2.5}$$

$s(x,y)$ is called a switching function, this function describes lines that divide the phase plane into regions where $s(x,y)$ has different signs. These lines are called switching lines. The lines define the sliding surface. The sliding surface is the set of points in the phase plane where the switching function $s(x,y) = 0$.

The system is analytically defined in two regions of the phase plane, since the feedback gain ψ is switched according to the sign of $s(x,y)$. When $s(x,y) = x\sigma > 0$, the model is

$$\dot{x} = y \tag{2.6}$$

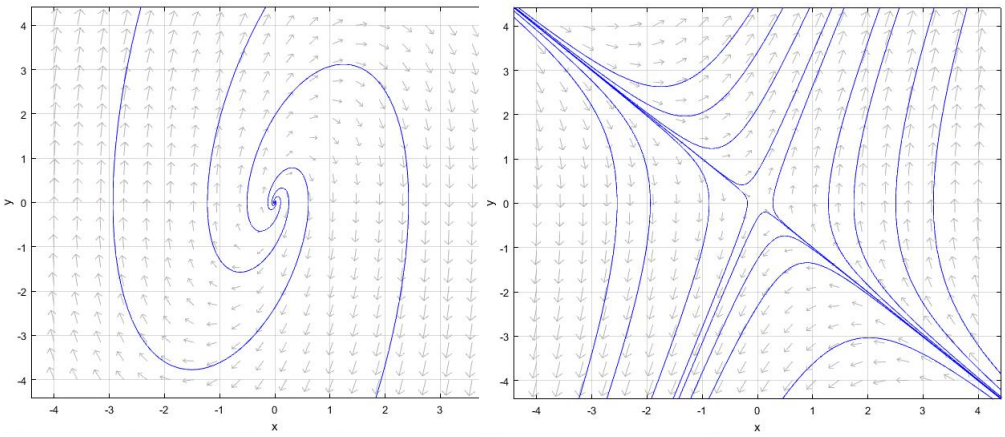
$$\dot{y} = 2y - x - 4x = 2y - 5x \tag{2.7}$$

When $s(x,y) = x\sigma < 0$, the model is

$$\dot{x} = y \tag{2.8}$$

$$\dot{y} = 2y - x + 4x = 2y + 3x \tag{2.9}$$

The phase portraits from both models can be seen in figure 2.1.



(a) System 2.7 **(b)** System 2.9

Figure 2.1: Phase portraits system 2.7 and 2.9

From the figure, it is easy to see that both equilibrium points are unstable.

To get the phase portrait from the complete system, the trajectory of the system on the set $s(x, y) = 0$ must be described. When $s(x, y) = 0$ then $x = 0$, on this line, the trajectories of both models are joined together without any ambiguity. This means that the phase portrait is a trajectory along the switching line $\sigma = 0$.

The motion that makes the trajectories of the system move toward the line $\sigma = 0$ is called reaching mode, and the motion along the line $\sigma = 0$ is called a sliding mode. The reaching mode makes the state trajectory, starting from anywhere on the phase plane, move toward a switching line and it makes the state trajectory reach the line in finite time. The sliding mode makes the trajectory asymptotically tend to the origin of the phase plane, Hung et al. (1993).

2.1.2 Sliding mode control for linear systems

Problem

Hung et al. (1993) formulates the SMC problem as:

For a given control system represented by the state equation

$$\dot{x} = A(x, t) + B(x, t)u \tag{2.10}$$

where dim-$x = n$ and dim-$u = m$, find:

1. m switching functions, represented in vector form as $s(x)$, and
2. a SMC

$$\begin{aligned} u(x,t) &= u^{+}(x,t) \quad \text{when} \quad s(x) > 0 \\ &= u^{-}(x,t) \quad \text{when} \quad s(x,y) < 0 \end{aligned} \tag{2.11}$$

such that the reaching modes satisfy the reaching condition, namely, reach the set $s(x) = 0$ (sliding surface) in finite time.

The physical meaning of above statement is as follows:

1. Design a sliding surface $s(x) = 0$ to represent a desired system dynamics that is of lower order than the given plant.
2. Design a SMC $u(x, t)$ such that any state x outside the sliding surface is driven to reach the surface in finite time. On the sliding surface, the sliding mode takes place, following the desired system dynamics. In this way, the overall SMC system is globally asymptotically stable.

Switching schemes, reaching conditions and control laws

In this section different switching schemes, their reaching laws and control laws that have been presented in Hung et al. (1993) will be explained. In these explanations, S_E will be the eventual sliding mode, that is the sliding surface that has dimension $n - m$, where n is the dimension of u, and m is the dimension of x.

A switching scheme is a method to make the sliding motion begin in a SMC algorithm. The condition under which the state will move towards and reach a sliding surface is called a reaching condition. The system trajectory under the reaching condition is called the reaching mode or reaching phase. The control law is what makes the state trajectories stay on the close neighbourhood of the sliding surface for all future time. In SMC the design of the control law is affected by two factors: the choice of a sliding mode entering scheme (switching scheme) or whether or not the structure of the control law has been pre-specified.

Effect of switching schemes:

Fixed-order switching scheme: In this scheme, sliding mode takes place in a fixed order. It moves from the initial state x_0 into the switching surface S_1, with dimensions $n-1$. It then moves to the switching surface $S_{12} = (S_1 \cap S_2)$, with dimension $n-2$. The sliding moves progressively to sliding surfaces with lower dimensions until it reaches the surface S_E, which has dimension $n - m$:

$$x_0 \to S_1 \to (S_1 \cap S_2) \to (S_1 \cap S_2 \cap S_3) \to \ldots \to S_E \tag{2.12}$$

This scheme is called the hierarchical SMC scheme. It is slow and has a poor transient response in general, resulting in large magnitude for control effort, and great difficulty in its solution.

To determine the controller U for this scheme m pairs of inequalities has to be solved:

$$\dot{s_i} = \frac{\partial s_i}{\partial x}(Ax + Bu) = \begin{cases} > 0 & \text{when } s_i < 0 \\ < 0 & \text{when } s_i > 0 \end{cases} \text{ for } i = 1, \cdots, m \tag{2.13}$$

Let b_i be the ith column vector of the B matrix. By differentiating 2.13

$$\begin{aligned} c_1^T Ax + c_1^T b_1 u_1 + \cdots + c_1^T b_m u_m &\begin{cases} < 0 & \text{when } s_1 > 0 \\ > 0 & \text{when } s_1 < 0 \end{cases} \\ &\vdots \\ c_m^T Ax + c_m^T b_1 u_1 + \cdots + c_m^T b_m u_m &\begin{cases} < 0 & \text{when } s_m > 0 \\ > 0 & \text{when } s_m < 0 \end{cases} \end{aligned} \tag{2.14}$$

this gives $2m$ conditional inequalities for $2m$ unknowns. The unknowns are the control signals $u_i(x)$, $i = 1$ to m. Every $u_i(x)$ has two values for two different conditions. This means that solving 2.13 for $u_i(x)$ is very difficult. The solutions are also very often conservative, meaning that the magnitudes are large. Therefore, this scheme is not often used.

Free-order switching scheme: In this scheme the order is not fixed but it follows the natural trajectory on a first-reach-first-switch scheme. The switching occurs on the location of the initial state in the state space. The scheme is much easier to use then the fixed-order-scheme. This is because the solution of SMC is easily determined, the dynamic characteristics of the reaching mode is better and the magnitude of the resulting control effort is smaller.

The reaching law that results in the free-order-switching scheme is the reaching law method. In this method the dynamic of the switching functions is directly specified as:

$$\dot{s} = -Qsgn(s) - Kf(s) \tag{2.15}$$

where the gains Q and K are diagonal matrices with positive elements, and

$$sgn(s) = [sgn(s_1) \cdots sgn(s_m)]^T \tag{2.16}$$

$$f(s) = [f_1(s_1) \cdots f_m(s_m)]^T \tag{2.17}$$

The scalar functions f, satisfy the condition

$$s_i f_i(s_i) > 0 \quad \text{when} \quad s_i \neq 0,\ i = 1 \text{ to } m \tag{2.18}$$

Equation 2.15 is called the reaching law. By varying Q and K different rates for s can be specified, this results in different structures in the reaching law. Three examples are

1. The constant rate reaching law

$$\dot{s} = -Qsgn(s) \tag{2.19}$$

2. The constant plus proportional rate reaching law

$$\dot{s} = -Qsgn(s) - Ks \tag{2.20}$$

3. The power rate reaching law

$$\dot{s_i} = -k_i|s_i|^{\alpha} sgn(s_i) \quad 0 < \alpha < 1, \quad i = 1 \text{ to } m \tag{2.21}$$

In the reaching law approach the reaching conditions are established and the dynamic characteristics of the system are specified during the reaching phase. The approach provides a simple SMC solution and a measure for the reduction of chattering.

The controller for this scheme is solved directly from the reaching law specification described by 2.15

$$\dot{s} = \frac{\partial s}{\partial x}(Ax + Bu) = -Qsgn(s) - Kf(s) \tag{2.22}$$

The controller is easily solved from 2.22 as

$$u(x) = -\left(\frac{\partial s}{\partial x}B\right)\left[\frac{\partial s}{\partial x}Ax + Qsgn(s) + kf(s)\right] \tag{2.23}$$

Eventual sliding mode switching scheme: In this scheme the SMC takes place on the eventual sliding surface S_E. The state is driven from any initial state to S_E. There can also be other sliding surfaces. The scheme is easy to implement and the control is easy to make smooth, but it does not guarantee good transient characteristics.

The reaching condition for the eventual sliding mode switching scheme can be found by choosing the Lyapunov function candidate:

$$V(x,t) = s^T s \tag{2.24}$$

A global reaching condition is given by

$$\dot{V}(x,t) < 0 \quad \text{when} \quad s \neq 0 \tag{2.25}$$

Finite reaching time is guaranteed by modifying 2.25 to

$$\dot{V}(x,t) < -\epsilon \quad \text{when} \quad s \neq 0, \tag{2.26}$$

where ϵ is positive. The control law for the eventual sliding mode switching scheme is designed by satisfying a Lyapunov stability criteria

$$\dot{V} = \frac{d}{dt}(s^T s) = 2s^T \dot{s} = 2s^T \frac{\partial s}{\partial x}(Ax + Bu) < 0 \tag{2.27}$$

Finding the solution for the control law here is harder than for the free-order switching scheme but is easier than that for the fixed-order switching.

Decentralized switching scheme: In this scheme the system is treated as m single-input subsystems, where each subsystem has a scalar switching function and an associated sliding mode. The subsystems are coupled in general. The combined vector switching function has the form

$$\begin{aligned} s(x) &= [s_1(x_1), \cdots, s_m(x_m)] \\ s_i(x_i) &= c_i^T x_i, \qquad i = 1 \text{ to } m \end{aligned} \tag{2.28}$$

where x_i and c_i are n_i-dimensional vectors with

$$\sum_{i=1}^{m} n_i = n \tag{2.29}$$

In the decentralized switching scheme the controller for each subsystem are designed separately. The controller for each subsystems can be obtained using any of the schemes above. Consider an n-th-order large-scale system with m single-input subsystems

$$\dot{x_i} = A_i x_i + B_i u_i + \sum_{j=1, j\neq i}^{m} A_{ij} x_j \quad i = 1, \cdots, m \tag{2.30}$$

here dim-$x_i = n_i$, and $n_1 + n_2 + \cdots + n_m = n$. For the ith subsystem, A_i represents dynamics internal to the subsystem, B_i is the input vector, and A_{ij} represents coupling to other subsystems. Let there be m switching functions, one for each subsystem

$$s_i(x_i) = C_i^T x_i \tag{2.31}$$

The overall system sliding mode then consists of m independent sliding modes, each moving on its own sliding surface $s_i(x_i) = 0$ such that $x_i \to 0$. This scheme is meant for large-scale systems, and works better than the free-order switching scheme for these types of systems.

Effect of structure of the control-law

Free structure control: In the free structure approach, the control $u(x)$ can be solved by constraining the sliding function to any one of the following conditions:

1. Direct switching approach: $s_i \dot{s}_i < 0$.
   This reaching condition is global but does not guarantee a finite reaching time.
2. Lyapunov function approach: $\dot{V} = \frac{d}{dt}(s^T s) < 0$
3. Reaching law approach: $\dot{s_i} = -q_i sgn(s_i) - k_i f_i(s_i)$
4. $\dot{V} = -q - kV \qquad V = s^T s$
5. $\frac{d}{dt} s_i^3 \leq -\eta |s_i|$
6. $\dot{s} = -F(s)$

the latter three forms are seldom used in practice.

Relay control: The SMC for each element of the control vector U takes the form of a relay. The relay gain may be either fixed or state dependent

$$\begin{aligned} u_i(x) &= k_i^+, \quad \text{when } s_i(x) > 0 \\ &= k_i^-, \quad \text{when } s_i(x) < 0 \quad i = 1, \cdots, m \end{aligned} \tag{2.32}$$

the values of k_i^+ and k_i^- are chosen to satisfy the desired reaching condition.

Linear feedback switched gains: The preassigned structure is

$$u(x) = \psi(x)x \tag{2.33}$$

where $\psi = [\psi_{ij}(x)]$ is an $m \times n$ matrix of state-dependent gains. A popular structure for the gain is

$$\begin{aligned} \psi_{ij}(x) &= \alpha_{ij}, \quad \text{when } s_i(x)x_j > 0 \\ &= \beta_{ij}, \quad \text{when } s_i(x)x_j < 0 \quad \text{for} \begin{cases} i = 1, \cdots, m \\ j = 1, \cdots, n \end{cases} \end{aligned} \tag{2.34}$$

parameters α_{ij} and β_{ij} are chosen to satisfy the desired reaching condition. The details of 2.34 can be varied to suit the problem at hand.

Augmenting the equivalent control: The SMC control takes the form

$$u = u_e + \Delta u \tag{2.35}$$

where u_e is the equivalent sliding mode control, whereas Δu is added to satisfy the reaching condition. A commonly used Δu is a relay control.

2.1.3 Sliding mode control for non-linear systems

The most general state equation for a non-linear system is

$$\dot{x} = f(x, u, t) \tag{2.36}$$

where dim-$x = n$ and dim-$u = m$. The basic concepts and the fundamental theory for SMC for non-linear systems are similar to those for linear system. It is simple and straightforward to find the controller $u(x)$, but it is difficult to find the sliding function and to analyse the sliding mode. To analyse the stability of sliding modes in non-linear systems the states of the system has to be transformed to reduced form, controllability form, the input-output decoupled form or normal form. The characteristics of the canonical form used, can then be used to design a reaching law, Hung et al. (1993).

Sliding mode control for canonical forms

In Hung et al. (1993) optimal sliding mode approaches are given based on the canonical form of the system, they are described as follows:

Reduced form: Let the state vector be partitioned according to

$$\dot{x}_1 = A_1(x) \quad \text{dim-}x_1 = n - m \tag{2.37}$$

$$\dot{x}_2 = A_2(x) + B^*(x)u \quad \text{dim-}x_2 = \text{dim-}u = m \tag{2.38}$$

Consider a general m-dimensional sliding surface equation

$$s(x) = s(x_1, x_2) = 0 \tag{2.39}$$

Theoretically, this equation can be solved for x_2 in terms of x_1 in the form of $x_2 = w(x_1)$. This is equivalent to expressing the sliding function as

$$s(x) = x_2 - w(x_1) \tag{2.40}$$

The problem of determining the sliding function $s(x)$ then reduces to find $w(x_1)$ such that the sliding mode is asymptotically stable. This means that the system 2.37, with $x_2 = w(x_1)$ has to be asymptotically stable. It can be difficult to find a desired $w(x_1)$, but if A_1 in 2.37 is in a special structure it gets easier. When $w(x_1)$ is found so is $s(x)$. To design the controller a reaching law can be used in a similar manner as for the linear systems. Using the reaching law 2.15 in system 2.37-2.38 yields the SMC control law

$$u(x) = -\left(\frac{\partial s}{\partial x_2}B^*(x)\right)^{-1}\left(Qsgn(s) + kf(s) + \frac{\partial s}{\partial x_1}A_1(x) + \frac{\partial s}{\partial x_2}A_2(x)\right) \tag{2.41}$$

Controllability form: In this form, the entire system is decomposed into m subsystems, so it is useful to use a decentralized sliding mode scheme with decoupled sliding functions, such as

$$s_i = c_i^T x_i \quad i = 1, \cdots, m \tag{2.42}$$

The stability of the sliding mode of each subsystem is guaranteed if c_i^T is chosen correctly. If the system is on controllable canonic form, then x is a sub-state vector in the phase variable form and the equations of the subsystem are

$$\begin{aligned}
\dot{x}_{i1} &= x_{i2} \\
\dot{x}_{i2} &= x_{i3} \\
&\vdots \\
\dot{x}_i(n_i + 1) &= x_{in_i} \\
\dot{x}_{in_i} &= \alpha_i(x) + \beta_i u_i \quad i = 1, \cdots, m
\end{aligned} \tag{2.43}$$

Therefore the switch plane $c_i^T x_i = 0$ becomes

$$x_{il}^{(n_i)} + c_{il}x_{il}^{(n_i-1)} + \cdots + c_{i(n_i-1)}\dot{x}_i + c_{in_i}x_i = 0 \tag{2.44}$$

The SMC can then be obtained from

$$x = \begin{bmatrix} x_1^T \\ \vdots \\ x_m^T \end{bmatrix}, \quad \text{dim-}x_i = n_i \quad \text{and} \quad \sum_{i=1}^{m} n_i = n \tag{2.45}$$

$$\dot{x}_i = A_i x_i + \alpha_i(x) + \beta_i(x)u \qquad i = 1, \cdots, m \tag{2.46}$$

where

$$A_i = \begin{bmatrix} 0 & I_{n_i-1} \\ 0 & 0 \end{bmatrix} \quad \text{dim-}A_i = n_i \times n_i$$

$$\alpha_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \alpha_{i0}(x) \end{bmatrix} \quad \text{dim-}\alpha_i = n_i$$

$$\beta_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_{i0}(x) \end{bmatrix} \quad \text{dim-}\beta_i = n_i \times m$$

and 2.42 by using the reaching law method. Since each subsystem is in controllable canonic form, the dynamics of the sliding modes are easily determined. The model is also in controllable canonic form, and the characteristic polynomial is specified by the coefficients of the vector c_i. The stability of each sliding mode can therefore be determined by choosing the elements of c_i, to match a desired characteristic equation.

The input-output decoupled form: This case is just like the one for the controllability form case, the only difference is the meaning of the transformed state variables. The SMC control issues are the same. In this case the sliding functions takes a specific form:

$$s_i(x_i) = (p + \lambda)^{n_i - 1} x_i \quad p \equiv \frac{d}{dt}(\cdot), \quad \lambda > 0 \tag{2.47}$$

Here it is advisable to apply the centralized SMC where a set of subsystem switching function is given by

$$s_i(x_i) = c_i^T x_i \quad i = 1 \text{ to } m \tag{2.48}$$

Normal form: The system dynamics for normal form are given as

$$\dot{z}_{i,j} = z_{i,j+1}, \qquad j = 1, \cdots, r_i - 1$$

$$\vdots$$

$$\dot{z}_{i,r_i} = \alpha_i(z, \eta) + \sum_{k=1}^{m} \beta_{i,k}(z, \eta) u_k \qquad i = 1, \cdots, m$$

$$\dot{\eta} = \gamma(z, \eta) \qquad \dim \eta = n - r \tag{2.49}$$

When the system is in normal form the stability of the zero dynamics is assumed stable. Zero dynamics is dynamics of the system under the condition that outputs and their derivatives are equal to zero. Zero dynamics is therefore described by the internal dynamic variables with $z = 0$:

$$\dot{\eta} = \gamma(0, \eta) \tag{2.50}$$

then, the switching function for a system in the normal form can be chosen as

$$s_i = c_i^T z_i \qquad i = 1, \cdots, m \tag{2.51}$$

How to guarantee the stability of the zero dynamics and how to relate to the formulation of the output vector functions $y = c(x)$ are the problems with this approach.

2.1.4 Characteristics of sliding mode control

Robustness

Robustness is one of the most important properties of SMC systems. If a system is represented by either a linear or non-linear high-order differential equation, the differential equation of the sliding mode can be entirely independent of effects due to modelling error and external disturbances. It can therefore be said that the sliding mode is invariant (better than just robust) to modelling error and disturbances. The invariance property requires that certain matching conditions are satisfied; Hung et al. (1993) examines these and states that for linear system on the form

$$\dot{x} = (A + \Delta A)x + Bu + f(t) \tag{2.52}$$

where ΔA and $f(t)$ represent the modelling error and external disturbance. If there exist $\Delta\tilde{A}$ and $\tilde{f}(t)$ such that matching conditions

$$\Delta A = B\Delta\tilde{A} \quad \text{and} \quad f(t) = B\Delta\tilde{f} \tag{2.53}$$

are satisfied, then the sliding mode is invariant. The physical meaning of 2.53 is that all modelling uncertainties and disturbances enter the system through the control channel.

The same result has been extended to non-linear systems

$$\dot{x} = A(x, t) + \Delta A(x, p, t) + B(x, t)u + \Delta B(x, p, t)u + f(x, p, t) \tag{2.54}$$

where p is an uncertain parameter vector, it has been shown in Hung et al. (1993) that invariance hold true if the following matching conditions are satisfied

$$\begin{aligned} \Delta A(x, p, t) &= B(x, t)\Delta\tilde{A}(x, p, t) \\ \Delta B(x, p, t) &= B(x, t)\Delta\tilde{B}(x, p, t) \\ f(x, p, t) &= B(x, t)\Delta\tilde{f}(x, p, t) \end{aligned} \tag{2.55}$$

for certain $\Delta\tilde{A}$, $\Delta\tilde{B}$ and $\tilde{f}$.

Chattering

In ideal SMC the state trajectory should reach the sliding surface $s = 0$ 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 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).

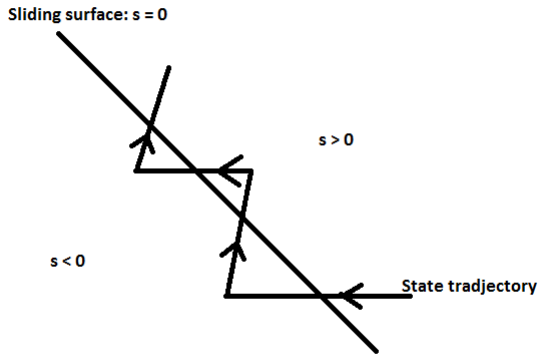


Figure 2.2: Chattering due to delay in control switching

The reason chattering appears as a result of delays in control computations is because when the state trajectory first hits the sliding surface it is supposed to stay on the sliding surface. But instead there will be a delay between the time the sign of s changes and the time the control switches. During this delay the state trajectory will cross the sliding surface, and reach the other region. When the control switches, the trajectory changes its direction and again heads toward the sliding surface. It then crosses the sliding surface again, and repetition of this process creates the oscillation called chattering Khalil (2002). This process is shown in figure 2.2.

2.1.5 Observer-based sliding mode control

Observer-based sliding mode control is proposed as a solution to chattering. Observers can either be used to estimate states that are not measurable, or to lock the chattering inside a high-frequency loop that bypasses the plant. This is done by localization of the high-frequency phenomenon in the feedback loop by introducing a discontinuous feedback control loop which is closed through an asymptotic observer of the plant. This will cause the chattering to be localized inside the loop because the observer has less modelling error than the plant and the control is discontinuous only with respect to the observer variable. It is assumed that the observer can be designed in a way that makes the observation error converge to zero asymptotically. Observer-based sliding mode control can also be used to create a disturbance compensator that will estimated the disturbance. If the disturbance is sufficiently compensated, there will be no need for a discontinuous feedback control to achieve sliding mode. This will also remove the chattering problem, Young et al. (1999). The research done in this field show that it is possible, but there are still a lot of engineering issues do be dealt with. This is an interesting topic that should be given some thought, but as observers are beyond the scope of this report the topic will not be discussed further.

2.2 First-order sliding mode control

2.2.1 Ideal relay control

The ideal relay controller takes the form:

$$u(s) = sgn(s) = \begin{cases} 1 & \text{when } s > 0 \\ -1 & \text{when } s < 0 \end{cases} \tag{2.56}$$

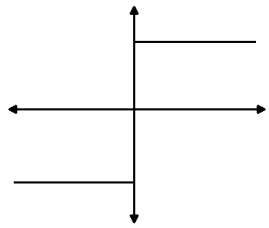


Figure 2.3: Phase portrait: ideal relay control

The controller is ideal in the sense that it switches instantly at the value $s = 0$. This means that ideal sliding exist on the line $s = 0$, 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

$$u(s) = sat(s) = \begin{cases} 1 & \text{when } s > L \\ \frac{s}{L} & \text{when } |s| \leq L \\ -1 & \text{when } s < -L \end{cases} \tag{2.57}$$

where $L > 0$ and $\pm L$ defines the threshold for entering the boundary layer.

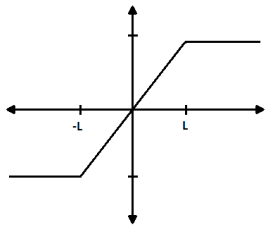


Figure 2.4: Phase portrait: ideal saturation control

The ideal saturation controller is given as a solution to 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 $s = 0$, 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 challenges 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.

2.2.3 Practical relay control and practical saturation control

The practical relay controller takes the form

$$u(s) = hys(s) = \begin{cases} 1 & \text{when } s > \Delta, \quad \text{or when } \dot{s} < 0 \text{ and } |s| < \Delta \\ -1 & \text{when } s < \Delta, \quad \text{or when } \dot{s} > 0 \text{ and } |s| < \Delta \end{cases} \tag{2.58}$$

where $2\Delta > 0$ is the amount of hysteresis in s.

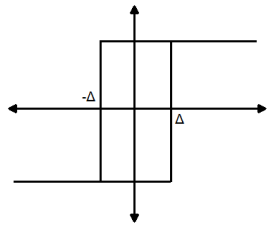


Figure 2.5: Phase portrait: practical relay control

In the practical relay controller the switching does not occur on the surface $s = 0$, it takes place on the lines $s = \pm\Delta$, this is because of the hysteresis characteristic. This leads to non-ideal sliding, which means that there will always be chattering present, and it will be impossible to eliminate. The system also has limit cycle behaviour in the steady-state mode, and the origin will no longer be an equilibrium point, Hung et al. (1993).

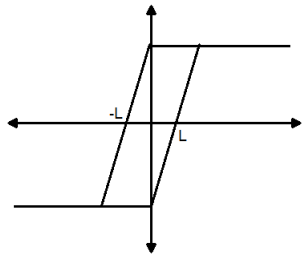


Figure 2.6: Phase portrait: practical saturation control

As the practical relay controller the practical saturation controller exhibits hysteresis. An analysis of the practical saturation control is very difficult, especially in the case of non-linear systems. An analysis has been done in Hung et al. (1993), where it is stated that in the practical relay controller the sliding mode does not exist at all. This means that the invariance characteristics do not exist. It is stable in the large sense, but it has two equilibrium points, which results in steady state error. In conclusion it eliminates the high-frequency chattering at the price of losing invariance. With a small boundary width some robustness can be maintained, but because of delays in the control actuator it may be necessary with a wider boundary layer. As a result the system may gain large amplitude low-frequency oscillation, and the system may lose all variable structure behaviour.

2.3 Second-order sliding algorithms

The second-order sliding algorithms are given by a continuous function U and a bounded discontinuous function ψ. This gives the system a continuous control that reduces chattering considerably. Consider the closed loop control system

$$\dot{x} = f(t, x, u) \tag{2.59}$$

$$u = U(t, x, \xi) \tag{2.60}$$

$$\xi = \psi(t, x, \xi) \tag{2.61}$$

where U is a feedback operator, ξ is a special operator variable. Equation 2.60 and 2.61 are called the second-order sliding algorithm on the constraint $\sigma = 0$ if a stable sliding mode of the second-order on the surface $\sigma = 0$ is achieved, and with every initial condition (t_0, x_0) the state x is transformed to the sliding mode in a finite time, Levant (1993).

In second-order sliding with $\sigma = 0$, the system is described by the equation

$$\dot{x} = f(t, x, u_{eq}(t, x)) \tag{2.62}$$

where u_{eq} is the equivalent control Utkin (1977) that is evaluated from the equation

$$\dot{\sigma} = \sigma'_t(t, x) + \sigma'_x(t, x) f(t, x, u_{eq}) = 0 \tag{2.63}$$

which is assumed to have a unique solution.

Problem

In Levant (1993) the second-order sliding mode problem is formulated as:

Assume $\sigma \in R$, $u \in R$ and t, $\sigma(t, x(t))$, $u(t)$ are available. The goal is to force the constraint σ to vanish. Assume positive constant σ_0, K_m, K_M and C_0 are given. The following conditions are imposed:

1. Concerning the constraint function σ and equation 2.59. Assume the following: $|u| \leq k, k = constant > 1$, f is a C^1 function, $\sigma(t, x)$ is a C^2 function. Here, $x \in X$, where X is a smooth finite-dimensional manifold. Any solution of 2.59 is well defined for all t provided $u(t)$ is continuous and satisfies $|u(t)| \leq k$ for each t.

2. Assume there exists $u_1 \in (0, 1)$ such that for any continuous function u with $|u(t)| > u_1$ for all t, $\sigma(t)u(t) > 0$ for some finite time t. This condition implies that there is at least one t such that $\sigma(t) = 0$ provided u has a certain structure.

3. There are positive constants σ_0, K_m, K_M, u_0, $u_0 < 1$, such that if $|\sigma(t, x)| < \sigma_0$ then $0 < K_m \leq \frac{\partial \dot{\sigma}}{\partial u} \leq K_M$ for all u, and the inequality $|u| > u_0$ implies $\dot{\sigma} u > 0$. The set $\{t, x, u | \sigma(t, x)| < \sigma_0\}$ is called the linearity region.

4. Consider the boundedness of the second derivative of the constraint function σ with every fixed value of control. Within the linearity region $|\sigma < \sigma_0|$ for all t, x, u the inequality $|L_u L_u \sigma(t,x)| < C_0$ holds. Where L_u is the total derivative defined as: $L_u(\cdot) = \frac{\partial}{\partial t}(\cdot) + \frac{\partial}{\partial x}(\cdot) f(t,x,u)$.

The SMC theory deals with the following class of systems

$$\dot{x} = a(t,x) + b(t,x)u \tag{2.64}$$

where $x \in R^n$. The task of keeping the constraint $\sigma = 0$ is reduced to the problem stated above. A new control μ and a constraint function ϕ are to be defined in this case by the transformation $u = \mu k \Phi(x)$, $\phi = \sigma(t,x)/\Phi(x)$, where $\Phi(x) = [x^t D x + h]^{1/2}$, k, $h > 0$ are constants, D is a non-negative definite symmetric matrix.

From section 2.3.1 to section 2.3.7 $u = -sgn(\sigma)$.

2.3.1 A_μ-algorithm

In Levant (1993) the A_μ-algorithm is stated as follows:

$$\dot{u} = \begin{cases} -u & \text{with } |u| > 1 \\ -\alpha sgn(\sigma) & \text{with } |u| \leq 1 \end{cases} \tag{2.65}$$

Assume that conditions (1), (3) and (4) is satisfies and let the control algorithm be

$$\dot{u} = \psi(t,x,u) \tag{2.66}$$

where ψ is a bounded measurable (Lebesgue) function. Assume also that for every second-order sliding point M and for every neighbourhood $V(M)$

$$\mu(V(M) \cap \psi^{-1}((-\infty, -C_0/K_m])) > 0$$

$$\mu(V(M) \cap \psi^{-1}([C_0/K_m, \infty))) > 0$$

where μ is a Lebesgue measure. Then there is a second-order siding mode on the constraint $\sigma = 0$ and the motion in this mode is described by equation 2.62. This implies that there exist a second-order sliding mode for system 2.59 with control input as in equation 2.65 for any sufficiently large α. But it is only stable in some cases. When the sliding mode is stable $|\sigma|$ and $|\dot{\sigma}|$ will exponentially decay to zero. The A_μ-algorithm algorithm approximates the properties of the regular first-order-sliding algorithm when $\alpha \to \infty$.

2.3.2 Twisting algorithm

In Levant (1993) the twisting algorithm is stated as follows:

$$\dot{u} = \begin{cases} -u & \text{with } |u| > 1 \\ -\alpha_m sgn(\sigma) & \text{with } \sigma\dot{\sigma} \leq 0, |u| \leq 1 \\ -\alpha_M sgn(\sigma) & \text{with } \sigma\dot{\sigma} > 0, |u| \leq 1 \end{cases} \tag{2.67}$$

where $\alpha_M > \alpha_m > 0$. Assume

$$\begin{aligned} &\alpha_m > 4K_M/\sigma_0,\ \alpha_m > C_0/K_M \\ &K_m\alpha_M - C_0 > K_M\alpha_m + C_0 \end{aligned} \tag{2.68}$$

Under the assumptions (1)-(4) in the problem statement and conditions 2.68 the twisting algorithm is a second-order sliding algorithm.

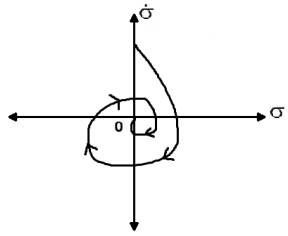


Figure 2.7: Phase portrait: twisting algorithm

As it can be seen in figure 2.7 the state paths for system 2.59 with control input as in equation 2.67 are encircling the sliding surface $\sigma = \dot{\sigma} = 0$, and it will converge to the surface in finite time. This is achieved by switching α in 2.67. As the derivative $\dot{\sigma}$ is difficult to calculate, the difference $\Delta\sigma$ is often used instead. To be able to use the difference the algorithm has to be made in discrete time, and the $\dot{\sigma}$ in 2.67 has to be replaced with:

$$\Delta\sigma(t_i) = \begin{cases} 0, & i = 0 \\ \sigma(t_i, x(t_i)) - \sigma(t_{i-1}, x(t_{i-1})), & i \geq 1 \end{cases} \tag{2.69}$$

where the measurements of σ are being made at times $t_0, t_1, t_2, \ldots, t_i - t_{i-1} = \tau > 0$.

2.3.3 Drift algorithm

In Levant (1993) the drift algorithm is stated as follows:

$$\dot{u} = \begin{cases} -u(t_i), & |u(t_i)| > 1 \\ -\alpha_M sgn(\Delta\sigma(t_i)),\ \sigma(t_i)\Delta\sigma(t_i) > 0, & |u(t_i)| \leq 1 \\ -\alpha_m sgn(\Delta\sigma(t_i)),\ \sigma(t_i)\Delta\sigma(t_i) \leq 0, & |u(t_i)| \leq 1 \end{cases} \tag{2.70}$$

Here, $\alpha_M > \alpha_m > 0$. This is similar to the twisting algorithm in discrete time, but $\sigma(t_i)$ is replaced with $\Delta\sigma(t_i)$. This will ensure a real sliding on $\dot{\sigma}$. Let

$$t_{i+1} - t_i = \tau_{i+1} = \tau(\sigma(t_i)),\ i = 0, 1, 2, \ldots \tag{2.71}$$

and let

$$\tau(S) = \begin{cases} \tau_M, & v|S|^p \geq \tau_M \\ v|S|^p, & \tau_m < v|S|^p < \tau_M \\ \tau_m & v|S|^p \leq \tau_m \end{cases} \tag{2.72}$$

where $0.5 \leq p \leq 1$, $\tau_M > \tau_m > 0$, $v > 0$. The region $|\sigma| \leq \sigma_0 - \delta$ where δ is any real number, such that $0 < \delta < \sigma_0$, is called the reduced linearity region. With initial conditions within the reduced linearity region, v and α_m/α_M sufficiently small and α_m sufficiently large, the algorithm 2.70, 2.71, 2.72 constitutes a second-order real sliding algorithm on the constraint $\sigma = 0$ with respect to $\tau_m \to 0$.

2.3.4 Algorithm with a prescribed law

In Levant (1993) an algorithm with a prescribed law of variation of σ is stated as follows:

$$\dot{u} = \begin{cases} -u, & |u| > 1 \\ -\alpha sgn(\dot{\sigma} - g(\sigma)), & |u| \leq 1 \end{cases} \tag{2.73}$$

where $\alpha > 0$, and the continuous function $g(\sigma)$ is smooth everywhere except on $\sigma = 0$. Assume that all the solutions of the equation $\dot{\sigma} = g(\sigma)$ vanish in a finite time, and the function $g'(\sigma)g(\sigma)$ is bounded. With initial conditions within the reduced linearity region and for α sufficiently large, the algorithm 2.73 constitutes a second-order sliding algorithm on the constrain $\sigma = 0$. By making algorithm 2.73 in discrete time and replacing $\dot{\sigma} - g(\sigma)$ with $\Delta\sigma(t_i) - \tau g(\sigma(t_i))$, it becomes a real sliding algorithm.

2.3.5 Algorithm without derivative of σ

In Levant (1993) an algorithm that does not utilize the derivative of σ is stated as follows:

$$u = u_1 + u_2 \tag{2.74}$$

$$\dot{u}_1 = \begin{cases} -u, & |u| > 1 \\ -\alpha sgn(\sigma), & |u| \leq 1 \end{cases} \tag{2.75}$$

$$u_2 = \begin{cases} -\lambda|\sigma_0|^\rho sgn(\sigma), & |\sigma| > \sigma_0 \\ -\lambda|\sigma|^\rho sgn(\sigma), & |\sigma| \leq \sigma_0 \end{cases} \tag{2.76}$$

where α, $\lambda > 0$, $\rho \in (0.1)$, and the initial values $u_1(t_0)$ are to be chosen from the condition $|u| = |u_1(t_0) + u_2(t_0, x_0)| \leq k$. The following inequalities are to be satisfied

$$\alpha > C_0/k_m, \quad \alpha > 4K_M/\sigma_0 \tag{2.77}$$

$$\rho(\lambda K_m)^{\frac{1}{\rho}} > (K_M\alpha + C_0)(2K_M)^{\frac{1}{\rho}-2} \tag{2.78}$$

Assume that conditions 2.77, 2.78 are satisfies and $0 < \rho \leq 1/2$. Then the algorithm 2.74, 2.75, 2.76 is a second-order sliding algorithm.

It can be remarked that by selecting $\rho = 0.5$ this algorithm is the super-twisting algorithm that will be explained in section 2.3.7.

2.3.6 General second-order sliding mode

Problem:

In Bartolini et al. (2003) the second-order sliding mode control problem is stated as follows:

Consider an uncertain system, possibly non-linear, characterized by the dynamics

$$\dot{x}(t) = F(x(t), t) + G(x(t), t)u(t) \tag{2.79}$$

where $x \in X \subset R^n$ is the state vector, $u \in U \subset R$ is the scalar control variable, $F : R^{n+1} \to R^n$ and $G : R^{n+1} \to R^n$ are uncertain, sufficiently smooth, vector fields. Assume that the control specifications are fulfilled by constraining to zero a suitable output variable i.e. the sliding variable

$$\sigma(t) = \sigma(x(t), t) \tag{2.80}$$

having well-defined relative degree $r = 2$ with respect to the control variable u, and that a diffeomorphism $\Psi : R^{n-2} \times R^2 \to R^n$ exists such that the dynamics of the internal state $w(t) \in R^{n-2}$ are bounded-input, bounded-output (BIBO) stable. Therefore, system 2.79 can be reduced to the normal form

$$\begin{aligned} \ddot{\sigma}(t) &= \varphi(w(t), \sigma(t), \dot{\sigma}(t), t) + \gamma(w(t), \sigma(t), \dot{\sigma}(t), t)u(t) \\ \dot{w}(t) &= \psi(w(t), \sigma(t), \dot{\sigma}(t), t) \end{aligned} \tag{2.81}$$

with

$$x(t) = \Psi(w(t), \sigma(t), \dot{\sigma}(t)) \tag{2.82}$$

Assume that the second-order input-output dynamics

$$\ddot{\sigma}(t) = \varphi(w(t), \sigma(t), \dot{\sigma}(t), t) + \gamma(w(t), \sigma(t), \dot{\sigma}(t), t)u(t) \tag{2.83}$$

are globally bounded, and let also the sign of the control gain $\gamma(\cdot)$ be constant and known. Then, the second-order sliding mode control problem for system 2.79 entails the finite-time stabilization of system 2.83, that satisfies the global boundedness condition

$$\begin{aligned} &|\varphi(w, \sigma, \dot{\sigma}, t)| \leq \Phi \\ &0 < \Gamma_m \leq \gamma(w, \sigma, \dot{\sigma}, t) \leq \Gamma_M \end{aligned} \tag{2.84}$$

Let the sliding variable σ and the sign of its total time derivative $\dot{\sigma}$ be available for feedback. Bartolini et al. (2003) have assumed that the conditions in equation 2.84 hold globally.

Control law:

In Bartolini et al. (2003) a general control law has been proposed, where setting the parameters of the control properly can practically implement a large number of second-order sliding mode control algorithms. The control law is as follows:

$$\begin{aligned} &u(t) = -\alpha(t) U sgn(\sigma - \beta\sigma_M) \\ &\alpha = \begin{cases} 1, & \text{if } (\sigma - \beta\sigma_M)\sigma_M \geq 0 \\ \alpha^*, & \text{if } (\sigma - \beta\sigma_M)\sigma_M < 0 \end{cases} \\ &\beta \in [0; 1) \end{aligned} \tag{2.85}$$

where $U > 0$ is the minimum control magnitude, $\alpha^* > 1$ is called the modulation factor, β is the anticipation factor, and σ_M is the last extremal value of the sliding variable σ (i.e. the value of σ at the last time instant at which a local maximum, minimum or horizontal flex point of $\sigma(t)$ has occurred). U, α^* and β are control parameters that need to fulfil the inequalities

$$\begin{aligned} &U > \frac{\Phi}{\Gamma_M} \\ &\alpha^* \in [1; +\infty) \cap \left(\frac{2\Phi + (1-\beta)\Gamma_M U}{(1+\beta)\Gamma_m U}; +\infty \right) \end{aligned} \tag{2.86}$$

The first part of 2.86, the dominance condition, ensures that the control can sufficiently effect the sign of $\ddot{\sigma}$. The second part of 2.86, the convergence condition, ensures two things. First of all it ensures that the controller can sufficiently guarantee the stability of the sliding mode and secondly it determines the rate of convergence. This leads to transient trajectories twisting around the origin of the $\dot{\sigma} - \sigma$ plane. By imposing an even stricter inequality than 2.86, the monotonic convergence to zero of the sliding variable is also fulfilled. The stricter inequality can be:

$$\begin{aligned} &U > \frac{\Phi}{\Gamma_M} \\ &\alpha^* \in [1; +\infty) \cap \left(\frac{\Phi + (1-\beta)\Gamma_M U}{\beta\Gamma_m U}; +\infty \right) \end{aligned} \tag{2.87}$$

Controller 2.85 satisfying either conditions 2.86 or 2.87 assured the establishment of the second-sliding mode behaviour in a finite time.

Some of the algorithms that can be made by changing the variables in this general formulation are:

Twisting algorithm: Bartolini et al. (2003) states that by choosing $\beta = 0$ the control law 2.85 causes the system to have the same trajectories of the twisting algorithm, that was mentioned earlier in section 2.3.2, with $\alpha_2 = U$ and $\alpha_1 = \alpha^* U$. The algorithm is then:

$$u = -\alpha_1 sgn(\sigma) + \alpha_2 sgn(\dot{\sigma}) \tag{2.88}$$

The convergence conditions of for the twisting algorithm are easily obtained by setting $\beta = 0$.

$$\begin{aligned} U &> \frac{\Phi}{\Gamma_M} \\ \alpha^* &\geq \frac{2\Phi + \Gamma_M U}{\Gamma_m U} \end{aligned} \tag{2.89}$$

Sub-optimal algorithm: Bartolini et al. (2003) states that by choosing $\beta = 0.5$ the control law 2.85 causes the system to have the same trajectories of the sub-optimal algorithm. The convergence conditions of for the sub-optimal algorithm are easily obtained by setting $\beta = 0.5$.

$$\begin{aligned} U &> \frac{\Phi}{\Gamma_M} \\ \alpha^* &\in [1; +\infty) \cap \left(\frac{4\Phi + \Gamma_M U}{3\Gamma_m U}; +\infty \right) \end{aligned} \tag{2.90}$$

2.3.7 Super-twisting algorithm

One of the most powerful second-order continuous sliding mode control algorithms is the super-twisting algorithm (STA) that handles a relative degree equal to one. It generates the continuous control function that drives the sliding variable and its derivative to zero in finite time in the presence of the 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 variable, Chalanga et al. (2016).

In Levant (1993) the STA was introduced as:

$$\begin{aligned} u &= -k_1 |\sigma|^{1/2} sgn(\sigma) + v \\ \dot{v} &= -k_2 sgn(\sigma) \end{aligned} \tag{2.91}$$

where k_i are gains to be designed.

Let the input signal $f(t)$ be a measurable locally bounded function, and let 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:

$$k_2 > C, \qquad k_1^2 \geq 4C \frac{k_2 + C}{k_2 - C} \tag{2.92}$$

Condition 2.92 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. An STA with adaptive gains is therefore proposed.

STA with adaptive gains

Problem

In Shtessel et al. (2010) the problem has been formulated as:

Consider a single-input uncertain non-linear system,

$$\dot{x} = f(x,t) + h(x,t)u \tag{2.93}$$

where $x \in R^n$ is a state vector, $u \in R$ is a control function, $f(x,t) \in R^n$ is a differentiable, partially known vector-field. Assume that

1. A sliding variable $\sigma = \sigma(x,t) \in R$ is designed so that the system's (2.93) desirable compensated dynamics are achieved in the sliding mode $\sigma = \sigma(x,t) = 0$.

2. The system's (2.93) input-output $(u \to \sigma)$ dynamics are of a relative degree one, and the internal dynamics are stable. Therefore, the input-output dynamics can be presented as

$$\begin{gathered} \dot{\sigma} = \frac{\partial \sigma}{\partial t} + \frac{\partial \sigma}{\partial x} f(x,t) + \frac{\partial \sigma}{\partial x} h(x) u \quad \to \\ \dot{\sigma} = \varphi(x,t) + b(x,t)u \quad \to \\ \dot{\sigma} = \varphi(x,t) + \omega, \quad \omega = b(x,t)u \quad \leftrightarrow \quad u = b^{-1}(x,t)\omega \end{gathered} \tag{2.94}$$

   The solution of system 2.94 is understood in the sense of Filippov, (Filippov and Arscott (1988)).

3. The function $b(x,t) \in R$ is known and not equal to zero $\forall\, x$ and $t \in [0, \infty)$.

4. The function $\varphi(x,t) \in R$ is bounded

$$|\varphi(x,t)| \leq \delta |\sigma|^{1/2} \tag{2.95}$$

where the finite boundary $\delta > 0$ exists but is not known. The problem is to drive the sliding variable σ and its derivative $\dot{\sigma}$ to zero in finite time in the presence of the bounded perturbation with the unknown boundary by means of continuous control.

Control structure

In Shtessel et al. (2010) the following STA was proposed:

$$\begin{aligned} \omega &= -\alpha |\sigma|^{1/2} sgn(\sigma) + v \\ \dot{v} &= -\beta sgn(\sigma) \end{aligned} \tag{2.96}$$

where the adaptive gains

$$\begin{aligned} \alpha &= \alpha(\sigma, \dot{\sigma}, t) \\ \beta &= \beta(\sigma, \dot{\sigma}, t) \end{aligned} \tag{2.97}$$

are to be defined. The control system given by eq. 2.94 and 2.96 is presented in a form

$$\begin{aligned} \dot{\sigma} &= -\alpha|\sigma|^{1/2} sgn(\sigma) + v + \varphi(x,t) \\ \dot{v} &= -\beta sgn(\sigma) \end{aligned} \tag{2.98}$$

The control design problem is reduced to designing an adaptive STA 2.96, 2.97 that drives σ, $\dot{\sigma} \rightarrow 0$ in finite time in the presence of the bounded perturbation with the unknown boundary. Consider system 2.98 and assume that the perturbation $\varphi(x,t)$ satisfies assumption (4), for some unknown constant $\delta > 0$. Then for any initial conditions $x(0)$, $\sigma(0)$ the sliding surface $\sigma = 0$ will be reached in finite time with the STA as 2.96 and adaptive gains as

$$\begin{aligned} \dot{\alpha} &= \begin{cases} \omega_1 \sqrt{\frac{\gamma_1}{2}}, & \text{if } \sigma \neq 0 \\ 0, & \text{if } \sigma = 0 \end{cases} \\ \beta &= 2\varepsilon\alpha + \lambda + 4\varepsilon^2 \end{aligned} \tag{2.99}$$

where $\varepsilon, \lambda, \gamma_1, \omega_1$ are arbitrary positive constants. The proof can be found in Shtessel et al. (2010).

In Shtessel et al. (2012) another super-twisting algorithm with adaptive gains was proposed. The problem was formulated a little bit different, but the control structure used where nearly the same, the difference was that the adaptive gain β was multiplied by $\frac{1}{2}$, so that the control structure became:

$$\begin{aligned} \omega &= -\alpha|\sigma|^{1/2} sgn(\sigma) + v \\ \dot{v} &= -\frac{\beta}{2} sgn(\sigma) \end{aligned} \tag{2.100}$$

where α and β are the same adaptive gains as in 2.97. For this formulation the adaptive gains become:

$$\begin{aligned} \dot{\alpha} &= \begin{cases} \omega_1 \sqrt{\frac{\gamma_1}{2}} sgn(|\sigma| - \mu), & \text{if } \alpha > \alpha_m \\ \eta, & \text{if } \alpha \leq \alpha_m \end{cases} \\ \beta &= 2\varepsilon\alpha \end{aligned} \tag{2.101}$$

where $\varepsilon, \lambda, \gamma_1, \omega_1, \eta$ are arbitrarily positive constants and the parameter α_m is an arbitrarily small positive constant. This controller with adaptive gains as in 2.101 ensures real second-order sliding mode, and that the sliding surface will be reached in finite time.

2.3.8 Twisting-like controllers

There exist some different types of twisting, super-twisting and integral-twisting-like controllers. Mostly they are quite similar to the original algorithms, but a linear or non-linear link is added. As there are a lot of them, and as they are built on other algorithms that are mentioned here, they will not be stated here. Some examples of these algorithms can be found in: Basin et al. (2016), Edwards and Shtessel (2016a), Plestan et al. (2010) and Yan et al. (2016).

2.4 Higher-order sliding mode

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).

2.4.1 Higher-order sliding mode for a universal single-input-single-output uncertainty system

In Levant (2001) and Levant (2003) a HOSM controller for a single-input-single output (SISO) system is presented. For this controller only a qualitative model for the system is needed, because the controller only needs the relative degree of the uncertain SISO system and the bounds of two input-output differential expressions. For uncertain SISO minimum-phase dynamic systems with known relative degree, the controller provides full real-time output control. This controller provides rth-order precision with respect to the sampling time step and it increases the relative degree of the system artificially. This can completely remove the chattering effect, Levant (2001).

Problem

In Levant (2001) and in Levant (2003) the problem is formulated as:
Consider a dynamic system of the form

$$\dot{x} = a(t,x) + b(t,x)u \qquad \sigma = \sigma(t,x) \tag{2.102}$$

where $x \in R^n$, $u \in R$ is control, the smooth functions a, b, σ and the dimension n are unknown. The relative degree r of the system is assumed to be constant and known. The task is to make the measured output σ vanish in finite time and to keep $\sigma \equiv 0$ by discontinuous feedback control. As the relative degree is equal to r, the control input u will appear explicitly in the rth total derivative of σ. The regularity condition is then

satisfied and $(\partial/\partial u)\sigma^{(r)} \neq 0$ at that given point. The output σ then satisfies an equation of the form

$$\sigma^{(r)} = h(t,x) + g(t,x)u \tag{2.103}$$

It is easy to see that $g = L_{b_e} L_{a_e}^{r-1}\sigma = (\partial/\partial u)\sigma^{(r)}$, $h = L_{a_e}^{r}\sigma$. This means that h is the rth total time derivative of σ calculated with $u = 0$. The unknown functions h and g can therefore be defined using only input-output relations. Because of the heavy uncertainties in the problem, 2.102 cannot be reduced to standard forms based on the knowledge of a, b and σ. The problem is then to find a discontinuous feedback $u = U(t,x)$, that will cause the appearance of an attracting r-sliding mode in finite time. The new controller has to generalize the standard first-order relay controller $u = -Ksgn\sigma$. Thus, we require that for some K_m, K_M, $C > 0$

$$0 < K_m \leq \frac{\partial}{\partial u}\sigma^{(r)} \leq K_M, \qquad |L_a^r \sigma| \leq C \tag{2.104}$$

where $L_a^r\sigma$ is the rth total time derivative of σ calculated with $u = 0$. Hence, conditions 2.104 can be defined in input-output terms only.

An arbitrary-order sliding controller

In Levant (2001) and Levant (2003) the building of an arbitrary-order sliding controller is formulated as:

Let p be the least common multiple of $1, 2, \ldots, r$. Denote

$$\begin{aligned}
N_{1,r} &= |\sigma|^{(r-1)/r}, \\
N_{i,r} &= \left(|\sigma|^{p/r} + |\dot{\sigma}|^{p/(r-1)} + \cdots + |\sigma^{(i-1)}|^{p/(r-i+1)}\right)^{(r-i)/p}, \qquad i = 1, \ldots, r-1 \\
N_{r-1,r} &= \left(|\sigma|^{p/r} + |\dot{\sigma}|^{p/(r-1)} + \cdots + |\sigma^{(r-2)}|^{p/2}\right)^{1/p}, \\
\phi_{0,r} &= \sigma, \\
\phi_{1,r} &= \dot{\sigma} + \beta_1 N_{1,r} sgn(\sigma), \\
\phi_{i,r} &= \sigma^{(i)} + \beta_i N_{i,r} sgn(\phi_{i-1,r}), \qquad i = 1, \ldots, r-1
\end{aligned} \tag{2.105}$$

where $\beta_1, \ldots, \beta_{r-1}$ are positive numbers. Let system 2.102 have relative degree r with respect to the output function σ and 2.104 be fulfilled. Suppose also that trajectories of system 2.102 are infinitely extendible in time for any Lebesgue-measurable bounded control function. Then with properly chosen positive parameters $\beta_1, \ldots, \beta_{r-1}$, α the controller

$$u = -\alpha sgn\Big(\phi_{r-1,r}\Big(\sigma, \dot{\sigma}, \cdots, \sigma^{(r-1)}\Big)\Big) \tag{2.106}$$

leads to the establishment of an r-sliding mode $\sigma \equiv 0$ attracting each trajectory in finite time. 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. Parameter $\alpha > 0$ is to be chosen specifically for any fixed C, K_m, K_M. 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. Controller 2.106 requires the availability of $\sigma, \dot{\sigma}, \cdots, \sigma^{(r-1)}$. The information demand may be lowered if the measurements are carried out at times t_i with constant step $\tau > 0$. Let

$$\begin{aligned} u = &- \alpha sgn\Big(\Delta\sigma_i^{(r-2)} + \beta_{r-1}\tau N_{r-1,r}\Big(\sigma_i, \dot{\sigma}_i, \cdots, \sigma_i^{(r-2)}\Big)\times \\ &sgn\Big(\phi_{r-2,r}\Big(\sigma_i, \dot{\sigma}_i, \cdots, \sigma_i^{(r-2)}\Big)\Big)\Big) \end{aligned} \tag{2.107}$$

where $\sigma_i^{(j)} = \sigma^{(j)}(t_i, x(t_i))$, $\Delta\sigma_i^{(r-2)} = \sigma_i^{(r-2)} - \sigma_{i-1}^{(r-2)}$, $t \in [t_i, t_{i+1})$. Under the same conditions as algorithm 2.106, with discrete measurements both algorithms 2.106 and 2.107 provide in finite time for fulfilment of the inequalities $|\sigma| < a_0\tau^r$, $|\dot{\sigma}| < a_1\tau^{r-1}, \cdots, |\sigma^{r-1}| > a_{r-1}\tau$ for some positive constants $a_0, a_1, \cdots, a_{r-1}$. This accuracy is the best possible with discontinuous $\sigma^{(r)}$ separated from zero.

To enable implementing the controller 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 (2003) 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. In Levant (2005) a robust homogeneous differentiator is included in the control structure thus yielding robust output-feedback controllers with finite time convergence. It is also demonstrated that the homogeneity features significantly simplifies the design and investigation of the HOSM. These differentiators will not be presented further, as they are beyond the scope of this report.

2.4.2 Higher-order sliding mode control scheme for a multi-input-multi-output non-linear system

In Defoort et al. (2009) a higher-order sliding mode control scheme for a class of uncertain multi-input-multi-output (MIMO) non-linear systems is proposed.

Problem

In Defoort et al. (2009) the problem is formulated as:

Consider the following general multi-input-multi-output non-linear uncertain system

$$\begin{aligned} \dot{x} &= f(x) + \sum_{i=1}^{m} g_i(x)u_i \\ y_1 &= \sigma_1(x) \\ &\vdots \\ y_m &= \sigma_m(x) \end{aligned} \tag{2.108}$$

where $x \in R^n$ and $u = [u_1, \ldots, u_m]^T \in R^m$ are the state variable and the control input, respectively. $f(x)$ and $g(x) = [g_1(x), \ldots, g_m(x)]^T$ are uncertain smooth functions, $\sigma(x) = [\sigma_1(x), \ldots, \sigma_m(x)]^T \in R^m$ is a smooth measurable output vector. The uncertainties on $f(x)$ and $g(x)$ are due to parameter variations, un-modelled dynamics or external disturbances and do not satisfy the well-known matching condition. The control objective consists in the vanishing of output $\sigma(x)$ in finite time.

Assume:

1. The relative degree vector $r = [r_1, \ldots, r_m]^T$ of system 2.108 with respect to $\sigma(x)$ is assumed to be constant and known. It means that the $m \times m$ matrix:

$$B(x) = \begin{bmatrix} L_{g_1} L_f^{r_1 - 1}\sigma_1(x) & \cdots & L_{g_m} L_f^{r_1 - 1}\sigma_1(x) \\ \vdots & & \vdots \\ L_{g_1} L_f^{r_m - 1}\sigma_m(x) & \cdots & L_{g_m} L_f^{r_m - 1}\sigma_m(x) \end{bmatrix}$$

   is non-singular and $L_{g_j} L_f^k \sigma_i(x) = 0$, for $1 \leq i \leq m$, $1 \leq j \leq m$ and $0 \leq k \leq r_1 - 1$. Moreover, it is supposed that the associated zero dynamics are asymptotically stable.

2. Consider the non-linear system 2.108 and sliding variable $\sigma(x)$. Assume that the time derivatives $\sigma_i, \dot{\sigma}_i, \ldots, \sigma_i^{r_i - 1}$ for all $i = 1, \ldots, m$ are continuous functions. The manifold defined as

$$s^T = \left\{ x : \left| \begin{array}{c} \sigma_1(x) = \dot{\sigma}_1(x) = \cdots = \sigma_1^{(r_1 - 1)}(x) = 0 \\ \vdots \\ \sigma_m(x) = \dot{\sigma}_m(x) = \cdots = \sigma_m^{(r_m - 1)}(x) = 0 \end{array} \right. \right\}$$

   is called the rth-order sliding set. If it is non-empty and locally an integral set in the Filippov sense (Filippov and Arscott (1988)), the motion on s^T is called rth-order sliding mode with respect to the sliding variable σ. The rth-order SMC approach allows the finite time stabilization of each variable σ_i and its $r_1 - 1$ first time derivatives by defining a suitable discontinuous control law. The r_ith time derivative of each function σ_i yields:

$$\left[\sigma_1^{(r_1)}(x), \ldots, \sigma_m^{(r_m)}(x)\right]^T = A(x) + B(x)u \tag{2.109}$$

with

$$A(x) = [L_f^{r_1}\sigma_1(x), \ldots, L_f^{r_m}\sigma_m(x)]^T$$

Assume that the solutions of the state differential equation 2.109 with discontinuous right-hand side are defined in the sense of Filippov (Filippov and Arscott (1988)).

3. Vector $A(x)$ and matrix $B(x)$:

$$\begin{cases} A(x) & = \overline{A}(x) + \Delta_A(X) \\ B(x) & = \overline{B}(x) + \Delta_B(X) \end{cases} \tag{2.110}$$

are partitioned into nominal part (i.e. $\overline{A}$ and $\overline{B}$), known a priori, and uncertain bounded functions Δ_A and Δ_B. Matrix $\overline{B}$ is non-singular. Furthermore, there are an a priori known non-linear function $\rho(x)$ and an a priori known constant $0 < \alpha \leq 1$ such that the uncertain functions satisfy the following inequalities:

$$\begin{cases} ||\Delta_A(x) - \Delta_B(x)\overline{B}^{-1}(x)\overline{A}(x)|| \leq \rho(x) \\ ||\Delta_B(x)\overline{B}^{-1}(x)|| \leq 1 - \alpha \end{cases} \tag{2.111}$$

for $x \in \chi \subset R^n$, χ being an open subset of R^n within which the boundedness of the system trajectories is ensured.

Then by applying to system 2.109 the following preliminary feedback:

$$u = \overline{B}^{-1}\{-\overline{A} + w\} \tag{2.112}$$

where $w = [w_1, \ldots, w_m]^T \in R^m$ is the auxiliary control input. This feedback partially decouples the nominal system (i.e without uncertainty). Thus system 2.109 can be expressed as follows:

$$[\sigma_1^{(r_1)}, \ldots, \sigma_m^{(r_m)}]^T = [I_m + \Delta_B\overline{B}^{-1}]w - \Delta_B\overline{B}^{-1}\overline{A} + \Delta_A \tag{2.113}$$

where I_m denotes the $m \times m$ identity matrix. The rth-order SMC of system 2.108 with respect to the sliding variable σ is equivalent to the finite time stabilization of the multi-variable uncertain system:

$$\begin{cases} \begin{cases} \dot{z}_{1,i} = z_{2,i} \\ \quad \vdots \qquad\qquad \forall i = \{1, \ldots, m\} \\ \dot{z}_{r_i-1,i} = z_{r_i,i} \end{cases} \\ [\dot{z}_{r_1,1}, \dot{z}_{r_2,2}, \ldots, \dot{z}_{r_m,m}]^T = [I_m + \Delta_B\overline{B}^{-1}]w - \Delta_B\overline{B}^{-1}\overline{A} + \Delta_A \end{cases} \tag{2.114}$$

with $1 \leq i \leq m$, $1 \leq j \leq r_i$, $z_{j,i} = \sigma_i^{(j-1)}$, $z_i = [z_{1,i}, z_{2,i}, \ldots, z_{r_1,i}]^T$ and $z = [z_1^T, \ldots, z_m^T]^T$.

Design of HOSM controller

In Defoort et al. (2009) a robust finite time controller design is proposed. The higher order SMC algorithm is designed in two steps:

1. The design of a finite time controller $w_{nom}(z)$ which guarantees the finite time stabilization of nominal system

$$\forall i \in \{1, \cdots, m\}, \qquad \begin{cases} \dot{z}_{1,i} = z_{2,i} \\ \quad \vdots \\ \dot{z}_{r_i-1,i} = z_{r_i,i} \\ \dot{z}_{r_i,i} = w_{nom,i} \end{cases}$$

   at the origin

2. The design of discontinuous control $w_{disc}(z)$ which enables to reject the uncertainties of system 2.114 and ensures that control objectives are fulfilled.

In order to stabilize in finite time uncertain system 2.114, define the following control law:

$$\begin{cases} w(z) = w_{nom}(z) + w_{disc}(z, z_{aux}) \\ \dot{z}_{aux} = -w_{nom}(z) \end{cases} \tag{2.115}$$

Auxiliary function $z_{aux} \in R^m$ will be used in the design of the sliding variable associated with the discontinuous control law $w_{disc}(z, z_{aux}) \in R^m$. The control law for the nominal system is

$$w_{nom}(z) = [w_{nom,1}(z_1), \ldots, w_{nom,m}(z_m)]^T \in R^m \tag{2.116}$$

where

$$w_{nom,i}(z_i) = -k_{1,i} sgn(z_{1,i}) |z_{1,i}|^{\nu_{1,i}} - \cdots - k_{r_i,i} sgn(z_{r_i,i}) |z_{r_i,i}|^{\nu_{r_i,i}}$$

where $\nu_{1,i}, \cdots, \nu_{r_i,i}$ satisfies:

$$\nu_{j-1,i} = \frac{\nu_{j,i}\nu_{j+1,i}}{2\nu_{j+1,i} - \nu_{j,i}}, \quad j \in \{2, \ldots, r_i\}$$

with $\nu_{r_i+1,i} = 1$ and $\nu_{r_i,i} = \nu_i$. Define the sliding variable $s(z) \in R^m$ associated with w_{disc} as follows:

$$s(z) = [z_{r_1,1}, z_{r_2,2}, \ldots, z_{r_m,m}]^T + z_{aux} \tag{2.117}$$

The time derivative of s along the system trajectories, can be expressed as:

$$\begin{aligned} \dot{s} &= [\dot{z}_{r_1,1}, \dot{z}_{r_2,2}, \ldots, \dot{z}_{r_m,m}]^T + \dot{z}_{aux} \\ &= [I_m + \Delta_B \overline{B}^{-1}]w - \Delta_B \overline{B}^{-1} \overline{A} + \Delta_A - w_{nom} \\ &= [I_m + \Delta_B \overline{B}^{-1}]w_{disc} - \Delta_B \overline{B}^{-1} \overline{A} + \Delta_A + \Delta_B \overline{B}^{-1} w_{nom} \end{aligned} \tag{2.118}$$

The control law w_{disc} is defined to ensure the sliding mode on $\{x \in \chi : s = 0\}$ is guaranteed despite uncertainties, and given as:

$$w_{disc} = -G(z)sgn(s) \tag{2.119}$$

where the gain satisfies:

$$G(z) \geq \frac{(1-\alpha)||w_{nom}(z)|| + \rho + \eta}{\alpha} \tag{2.120}$$

with $\eta > 0$. The notation $sgn([s_1, \ldots, s_m]^T)$ denotes $[sgn(s_1), \ldots, sgn(s_m)]^T$. Consider the non-linear system 2.108 and assume that assumptions 1-3 are fulfilled. Then, the control law

$$u = \overline{B}^{-1}\{-\overline{A} + w_{nom}(z) + w_{disc}(z, z_{aux})\} \tag{2.121}$$

where $\dot{z}_{aux} = -w_{nom}(z)$, $w_{nom}(z)$ and $w_{disc}(z, z_{aux})$ are given by equation 2.116 and 2.119, respectively. This ensures the establishment of a higher order sliding mode with respect to σ in finite time. The proof is shown in Defoort et al. (2009), with the use of Lyapunov functions.

2.4.3 Adaptive continuous higher-order sliding mode control

In most controllers a bound on the disturbance has to be know, therefore when a controller is designed, a conservative upper bound is used to ensure that sliding takes place. This conservative choice worsens the chattering associated with the implementation. This has led to adaptive SMC, where the adaptive gains can adapt to a level where they are as small as possible but still guarantee that sliding is maintained. In Edwards and Shtessel (2016b) an adaptive continuous HOSM control has been proposed as:

Consider the generic sliding mode equation

$$\sigma^{(n)}(t) = u(t) + a(t) \tag{2.122}$$

where the integer $n > 2$, and $u(t)$ represent the control variable while $a(t)$ is an unknown disturbance. The objective is to force $\sigma, \dot{\sigma}, \cdots, \sigma^{(n)} = 0$ in finite time. Consider the control law comprising two parts:

$$u(t) = -u_b(t) - u_s(t) \tag{2.123}$$

where

$$u_b(\cdot) = \gamma_1|\sigma|^{\alpha_1}sgn(\sigma) \ldots + \gamma_n|\sigma^{(n-1)}|^{\alpha_n}sgn(\sigma^{(n-1)}) \tag{2.124}$$

and

$$u_s(t) = \lambda|s|^{1/2}sgn(s(t)) + \int_0^t k(t)sgn(s(\tau))d\tau \tag{2.125}$$

where the auxiliary sliding variable

$$s(t) = \sigma^{(n-1)}(t) + \int_0^t u_b(\tau)d\tau \tag{2.126}$$

In 2.124 the scalars $\gamma_1, \gamma_2, \ldots, \gamma_n$ must be chosen such that the polynomial $p^n + \gamma_n p^{n-1} + \cdots + \gamma_2 p + \gamma_1$ is Hurwitz and the scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$ are chosen recursively as

$$\alpha_{i-1} = \frac{\alpha_i \alpha_{i+1}}{2\alpha_{i+1} - \alpha_i}, \quad i = 2, \ldots, n \tag{2.127}$$

with $\alpha_{n+1} = 1$ and $\alpha_n = \bar{\alpha}$. In 2.125 the positive scalar λ must be chosen sufficiently large and the time varying gain $k(t)$ adapts according to the dual layer structure:

$$\begin{aligned} \dot{k}(t) &= -\rho(t) sgn(\delta(t)) \\ \rho(t) &= r_0 + r(t) \\ \dot{r}(t) &= \begin{cases} \gamma|\delta(t)| & \text{if } |\delta(t)| > \delta_0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.128}$$

where $\delta(t)$ is an error variable, r_0 is a fixed positive scalar and $r(t)$ is an adaptive law where γ and δ_0 are positive scalars. Consider the system in 2.122 with uncertainty $a(t)$ that is twice differentiable and subject to $|\dot{a}(t)| < a_1$ and $|\ddot{a}(t)| < a_2$. Using the control law from 2.123 in 2.122 where $k(t)$ adapts according to the dual layer equations 2.128, and assume that

$$\frac{1}{4}\epsilon^2 > \delta_0^2 + \frac{1}{\gamma}\left(\frac{\bar{q}a_2}{\alpha}\right)^2$$

is satisfied, where $\bar{q} > 1$ represents a user defined gain. Then for sufficiently large λ, there exists an $\epsilon_b \in (0, 1)$ such that for every $\bar{\alpha} \in (1 - \epsilon_b, 1)$ the origin $\sigma, \dot{\sigma}, \ldots, \sigma^{(n)}$ is a finite time stable equilibrium point. Proof is given in Edwards and Shtessel (2016b).

2.4.4 Higher-order sliding mode control with optimal reaching

In Dinuzzo and Ferrara (2009) a higher-order sliding mode control is proposed by solving the Robust Fuller's problem.

The Robust Fuller's Problem

In Dinuzzo and Ferrara (2009), the Robust Fuller's Problem is formulated as:

$$\min_{||u||_\infty \leq \alpha} \quad \max_{\substack{||f_1||_\infty \leq C \\ K_m \leq f_2(t) \leq K_M \\ a.e}} \int_0^{+\infty} |\sigma^{(0)}(t)|^\nu dt \tag{2.129}$$

subject to

$$\dot{\sigma} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{pmatrix} \sigma + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ f_1 + f_2 u \end{pmatrix} \qquad \text{and} \qquad \sigma(0) = \sigma_0 \tag{2.130}$$

where $f_1(t) \in [-C, C]$, $f_2(t) \in [K_m, K_M]$. Under the assumptions of matched bounded disturbance, seek the control law that guarantees optimality for the worst-case trajectory, that is the trajectory of the differential inclusion 2.130 such that the maximum with respect to f_1 and f_2 is attained. It turns out, that independently of ν, the worst-case trajectory is such that $f_1 = -Csgn(u)$ and $f_2 = K_m$ almost everywhere. This means that in the worst-case trajectory, the additive uncertainty f_1 always opposes the control u with, maximum amplitude, while the multiplicative uncertainty reduces the control amplitude αf_2 to the smallest possible value αK_m.

Introduce the reduced control amplitude $\alpha_r := \alpha K_m - C$, suppose that $\alpha_r > 0$. Then, if v^* is an optimal control for

$$\min_{||\nu||_\infty \leq 1} \int_0^{+\infty} |z_1(t)|^\nu dt \tag{2.131}$$

subject to

$$\dot{z} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{pmatrix} z + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ v \end{pmatrix}, \quad z \in R^r \quad \text{and} \quad z(0) = z_0 = \alpha_r^{-1}\sigma_0, \tag{2.132}$$

then an optimal control u^* for 2.129 is given by

$$u^*(t) = \alpha v^*(t) \tag{2.133}$$

Relation 2.133 still hold when u^* and v^* are the limiting optimal controls for $\nu \to 0^+$.

Generic order sliding mode algorithm

In Dinuzzo and Ferrara (2009) it is shown that the solution to the Robust Fuller's problem with $\nu \to 0^+$ is given by the following rth order HOSM algorithm:

$$U(\sigma) = \alpha(-1)^{r+1}\tilde{U}(\sigma) \tag{2.134}$$

where $\tilde{U} \in \{-1, +1\}$ is such that the system of equations and inequalities

$$\begin{aligned} & z_1^{r-i+1} + 2\sum_{n=2}^{r}(-1)^{n+1} z_n^{r-i+1} = \tilde{U}\frac{(r-i+1)!}{\alpha_r}\sigma^{(i-1)}, \\ & i = 1, \ldots, r \quad z_1 \leq \ldots \leq z_r \leq 0 \end{aligned} \tag{2.135}$$

admits solution. The complexity of the switching surface grows very fast with r.

Chapter 3

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 investigate the main advantages and disadvantages of different SMC algorithms. The SMC algorithms that are to be investigated with this system is:

- the relay controller
- the saturation controller
- the super-twisting algorithm
- the super-twisting algorithm with adaptive gains

The reason for choosing the relay controller is that it was the first SMC algorithm proposed. The saturation controller was chosen because it is the most used SMC algorithm. These two algorithms is also important for comparison purposes. The super-twisting algorithm was chosen because it is the best known and most widely used second-order algorithm. The super-twisting algorithm with adaptive gains was chosen because it gives all the same properties as the super-twisting algorithm, but in addition the bound on the disturbance does not need to be known. The reason for not choosing any algorithms that have higher order than two, is that for a second order system it is not possible to implement a higher-order SMC algorithm without observers (differentiators), which is out of the scope of this report.

3.1 System

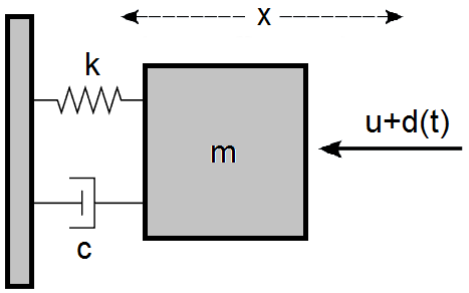


Figure 3.1: Schematic of mass-spring-damper system

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

$$m\ddot{x} + c\dot{x} + kx = u + d(t) \tag{3.1}$$

where x is the position of the mass, $m\,[kg]$ is the weight of the mass, $c\,[Ns/m]$ is the damping coefficient, $k\,[N/m]$ is the spring constant, $u\,[N]$ is the force input and $d(t)$ is a time-varying disturbance.

3.2 Control design

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

3.2.1 Sliding surface design

To design a sliding surface an error variable has to be introduced. Define the output as

$$y = x \tag{3.2}$$

The error variable can then be defined as:

$$e = y - y_{des} \tag{3.3}$$

where y_{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:

$$\sigma = \dot{e} + e \tag{3.4}$$

By inserting the states into the equation for the sliding surface, the sliding surface becomes:

$$\sigma = \dot{e} + e = \dot{y} - \dot{y}_{des} + y - y_{des} \tag{3.5}$$

The derivative of the sliding surface $\dot{\sigma}$ is then:

$$\dot{\sigma} = \ddot{y} - \ddot{y}_{des} + \dot{y} - \dot{y}_{des} = \frac{1}{m}(-c\dot{y} - ky + u + d(t)) - \ddot{y}_{des} + \dot{y} - \dot{y_{des}} \tag{3.6}$$

The objective is to get $\sigma = \dot{\sigma} = 0$. As the model for the mass-spring-damper is known, a new controller could have been introduced as $u = u_{eq} + u_0$, where u_0 would be the control input, and u_{eq} would cancel out the known terms in equation 3.6. The u_{eq} controller is described in section 2.3, and was first introduces in Utkin (1977). But since one of the properties that need to be tested is how well the different controllers handle unknown models, the u_{eq} controller will not be used in these simulations. This implies that when simulating, the controller's ability to handle disturbances and modelling errors are tested. The states y and $\dot{y}$ are available for measurement.

3.2.2 Control input design

The sufficiently large constant K that will be mentioned in the following sections can be found by finding an upper bound on the disturbance as mentioned in previous sections. As this is difficult, the trial and error method has been used instead.

The relay controller

When using the relay controller, the control input can be written as:

$$u = -Ksgn(\sigma) \tag{3.7}$$

where K is a sufficiently large positive constant and $sgn(\sigma)$ is defined as in equation 2.56.

The saturation controller

When using the saturation controller, the control input can be written as:

$$u = -Ksat(\sigma) \tag{3.8}$$

where K is a sufficiently large positive constant, and $sat(\sigma)$ is defined as in equation 2.57 with $L = 0.01$.

The super-twisting algorithm

By choosing the gains in equation 2.91 to be $k_1 = 1.5\sqrt{K}$ and $k_2 = 1.1K$, where K is a sufficiently large positive constant. The control input can be written as:

$$\begin{aligned} u &= -1.5\sqrt{K}|\sigma|^{1/2} sgn(\sigma) + v \\ \dot{v} &= -1.1K sgn(\sigma) \end{aligned} \tag{3.9}$$

The super-twisting algorithm with adaptive gains

By using the STA with adaptive gains proposed in Shtessel et al. (2010) the control input can be written as:

$$\begin{aligned} u &= -\alpha|\sigma|^{1/2} sgn(\sigma) + v \\ \dot{v} &= -\beta sgn(\sigma) \end{aligned} \tag{3.10}$$

where the adaptive gains are defined as:

$$\begin{aligned} \dot{\alpha} &= \begin{cases} \omega_1\sqrt{\frac{\gamma_1}{2}}, & \text{if } \sigma \neq 0 \\ 0, & \text{if } \sigma = 0 \end{cases} \\ \beta &= 2\varepsilon\alpha + \lambda + 4\varepsilon^2 \end{aligned} \tag{3.11}$$

where $\varepsilon, \lambda, \gamma_1, \omega_1$ are arbitrary positive constants. Since the sliding surface is nearly never equal to zero, a small bound is put on the sliding surface for implementation purposes, and the adaptive gains can be expressed as:

$$\begin{aligned} \dot{\alpha} &= \begin{cases} \omega_1\sqrt{\frac{\gamma_1}{2}}, & \text{if } |\sigma| > \alpha_m \\ 0, & \text{if } |\sigma| \leq \alpha_m \end{cases} \\ \beta &= 2\varepsilon\alpha + \lambda + 4\varepsilon^2 \end{aligned} \tag{3.12}$$

where the parameter α_m is an arbitrarily small positive constant, in this case chosen to be 0.1.

3.3 Implementation

For the mass-spring damper system MATLAB Simulink was used to implement the different controllers and the system model.

3.3.1 System model

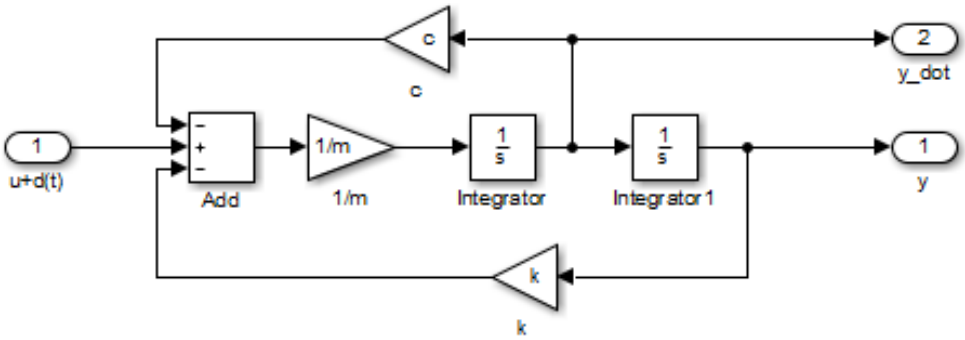


Figure 3.2: Implementation of mass-spring-damper system in Simulink

3.3.2 Control input

The relay controller

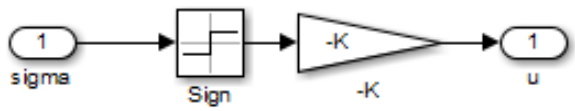


Figure 3.3: Implementation of relay control in Simulink

The saturation controller

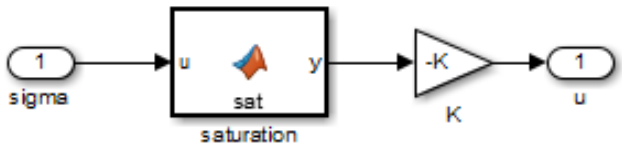


Figure 3.4: Implementation of saturation control in Simulink

The super-twisting algorithm

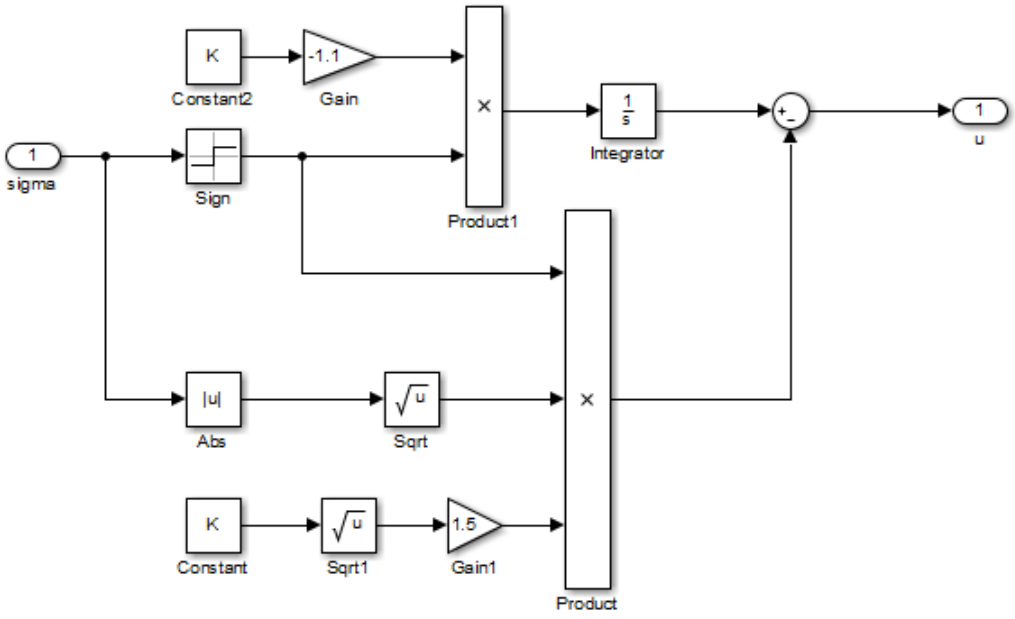


Figure 3.5: Implementation of super-twisting algorithm in Simulink

The super-twisting algorithm with adaptive gains

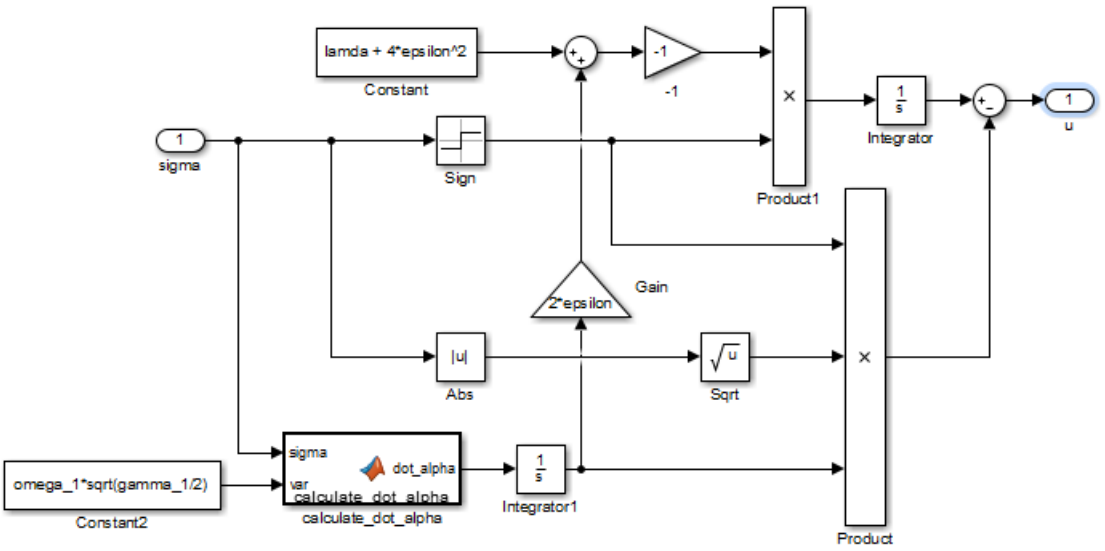


Figure 3.6: Implementation of super-twisting algorithm with adaptive gains in Simulink

3.4 Results

For the simulation 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 $y_{des} = 5\sin(2t)$.

3.4.1 The relay controller

The gain K for the relay control was set to 60.

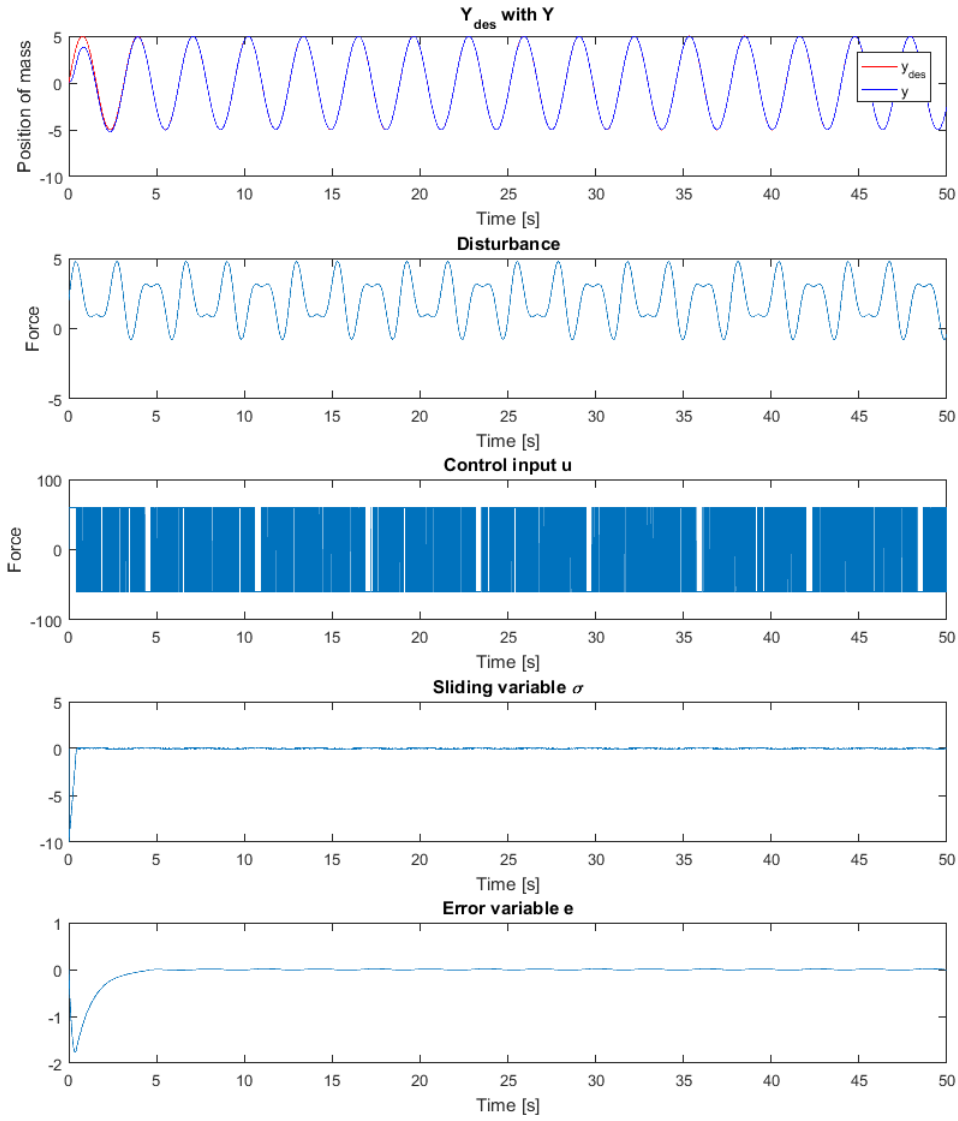


Figure 3.7: Results mass-spring-damper system: relay controller

3.4.2 The saturation controller

The gain K for the saturation control was set to 65.

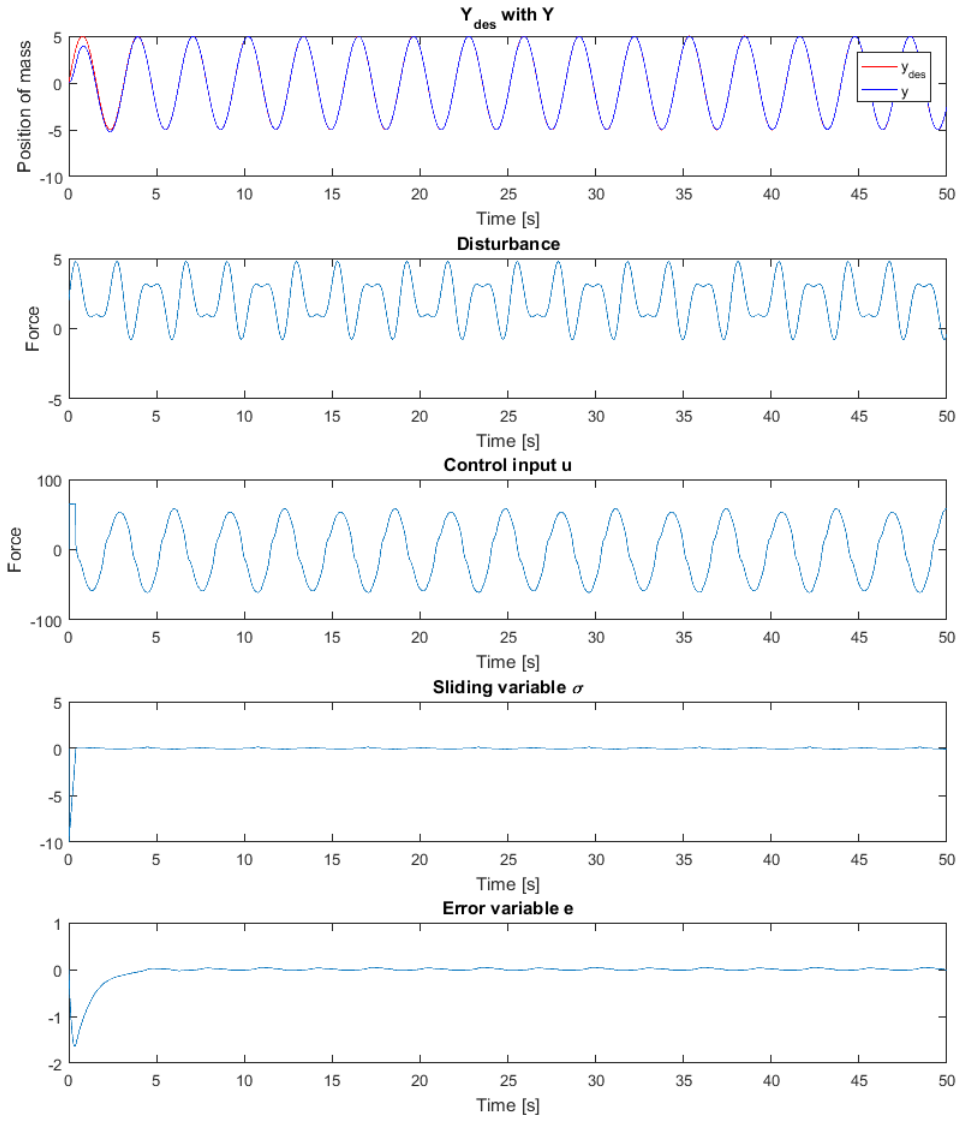


Figure 3.8: Results mass-spring-damper system: saturation controller

3.4.3 The super-twisting algorithm

The gain K for the super-twisting algorithm was set to 100.

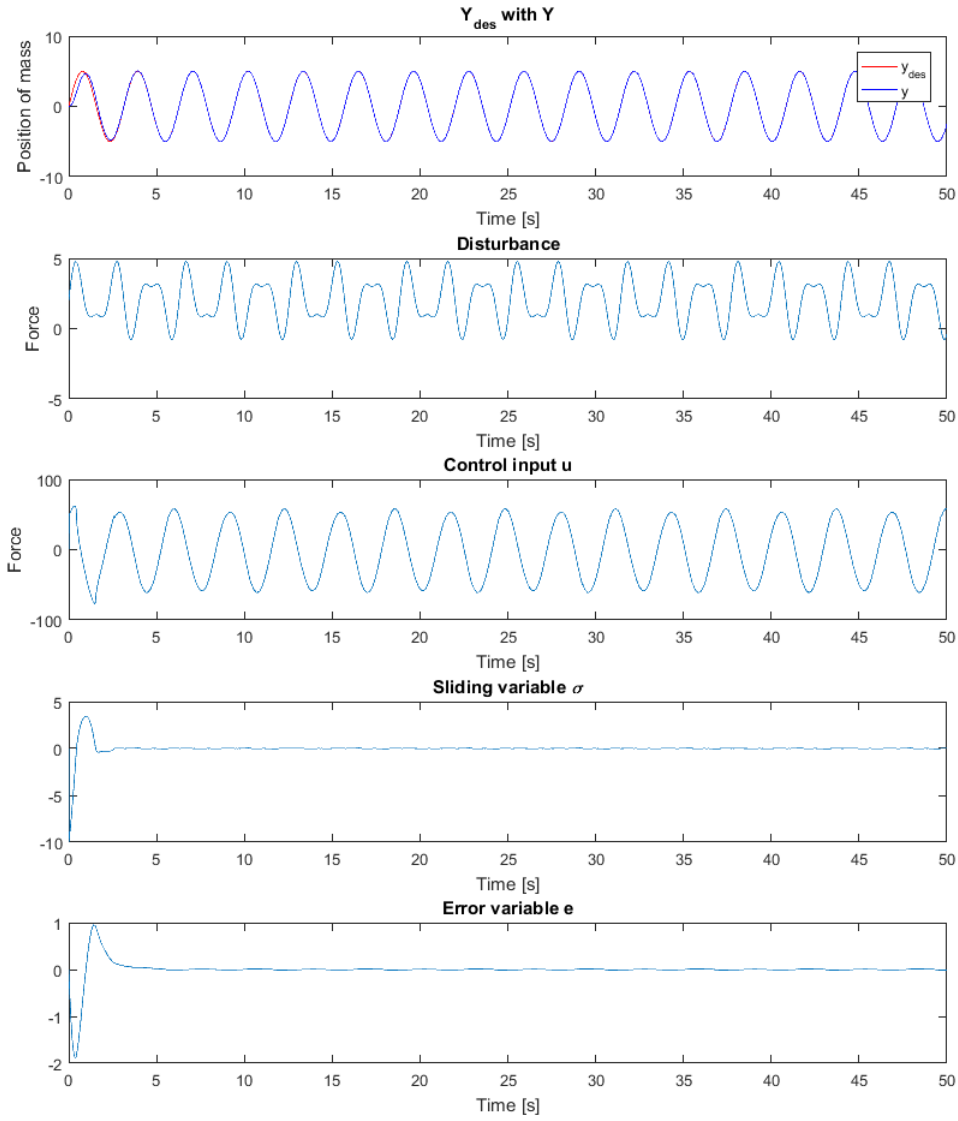


Figure 3.9: Results mass-spring-damper system: super-twisting algorithm

3.4.4 The super-twisting algorithm with adaptive gains

The gains for the super-twisting algorithm with adaptive gains were set to: $\epsilon = 1$, $\lambda = 1$, $\gamma_1 = 1$ and $\omega_1 = 20$.

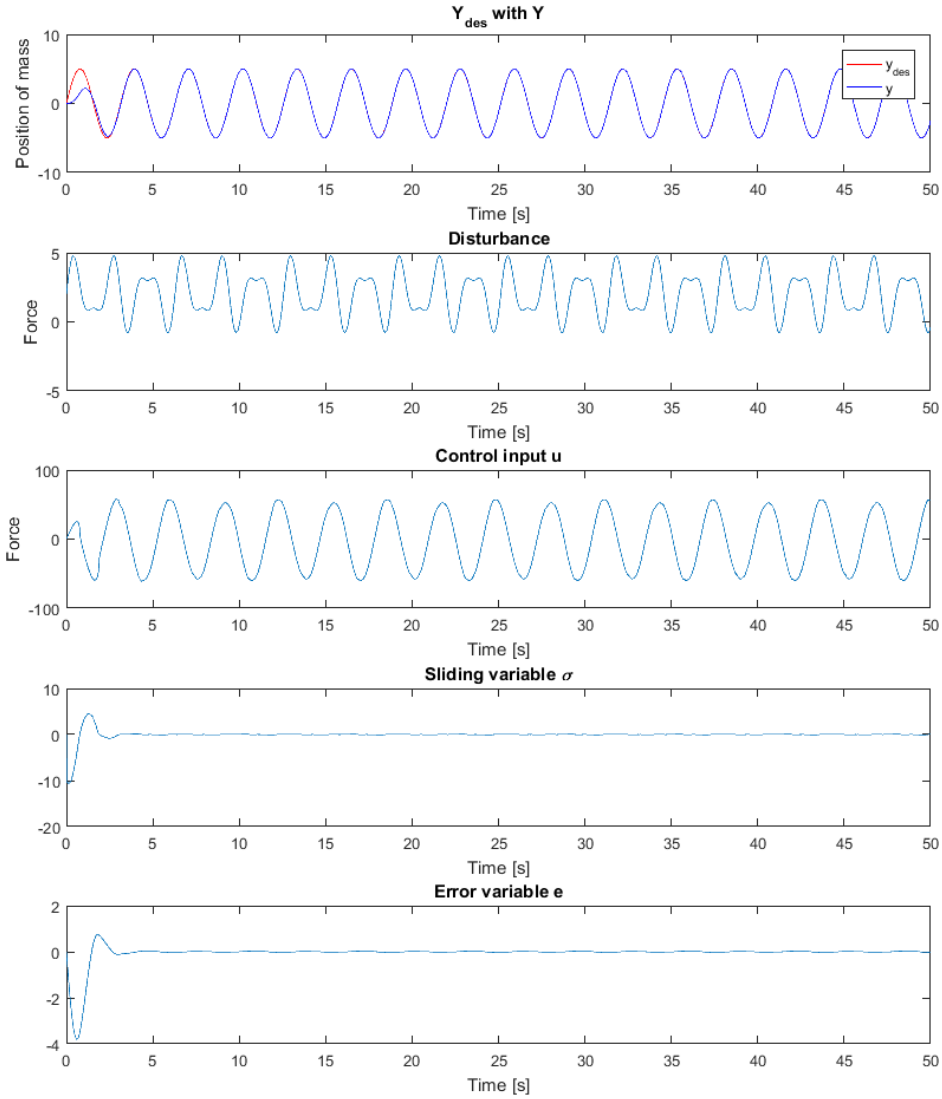


Figure 3.10: Results mass-spring-damper system: super-twisting algorithm with adaptive gains

In order to better compare the different algorithms the absolute maximum value for the error variable for each algorithm have been gather in table 3.1. Here the first 5 seconds were not considered so that the absolute maximum error was found when the control input had stabilized.

Table 3.1: Mass-spring-damper system: absolute maximum value for error variable

Algorithm	**Error**
Relay controller	0.0229
Saturation controller	0.0441
Super-twisting algorithm	0.0195
Super-twisting algorithm with adaptive gains	0.0206

Chapter 4

Underwater Swimming Manipulator

The underwater swimming manipulator, is a snake-like, multi-articulated, underwater robot equipped with thrusters. The USM has a complex control design problem. That 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. There have been cases where SMC has been used to control a snake like manipulator. In Rezapour et al. (2014), a relay inspired controller was used for a snake robot, and in Sverdrup-Thygeson et al. (2016b), the saturation controller was used for an USM. To the author's knowledge no second-order SMC algorithm has been tested on an USM. The algorithms that will be tested on this system are the same as for the test system.

4.1 System

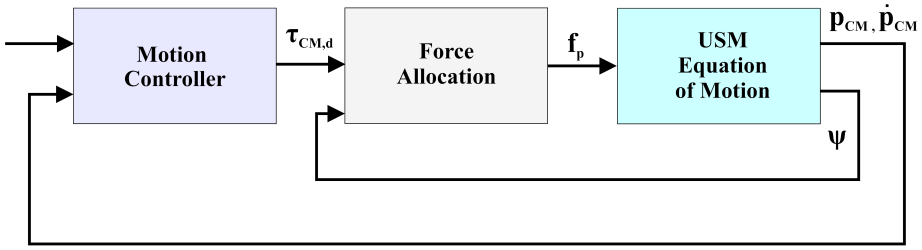


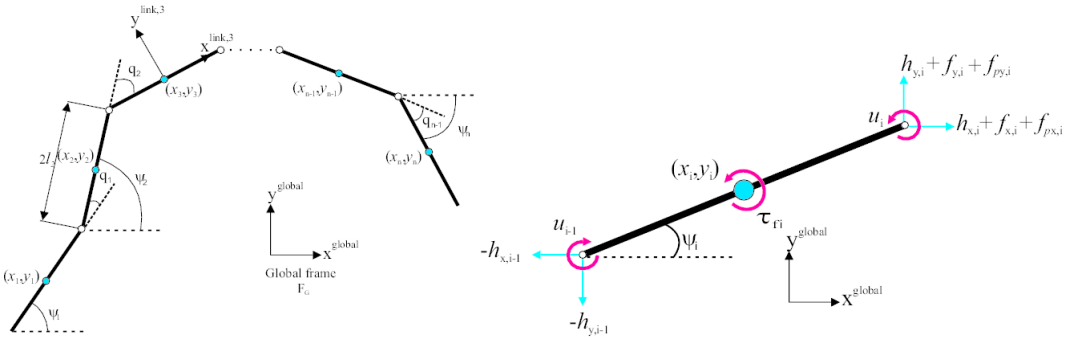
Figure 4.1: System overview USM, Sverdrup-Thygeson et al. (2016a)

In Sverdrup-Thygeson et al. (2016a) a model of an USM with additional effectors is presented together with a force allocation matrix for a 2D applications. 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. (2016a). The kinematic equations in Sverdrup-Thygeson et al. (2016a) are developed for 2D based on the method outlined in Liljeback et al. (2012).

4.1.1 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 4.2.



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

Figure 4.2: Underwater swimming manipulator, Sverdrup-Thygeson et al. (2016a)

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 $p_{CM} \in R^2$ of the CM of the USM is defined as

$$p_{CM} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{1}{m_t} \sum_{i=1}^{n} m_i x_i \\ \frac{1}{m_t} \sum_{i=1}^{n} m_i y_i \end{bmatrix} = \frac{1}{m_t} \begin{bmatrix} e^T M X \\ e^T M Y \end{bmatrix} \tag{4.1}$$

where (x_i, y_i), $i = 1, \cdots, n$ 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. This is valid because it is assumed that the mass of each link is uniformly disturbed. The matrix representation of the force balance for all links with different link mass is expressed by

$$M\ddot{X} = D^T h_x + f_x + f_{px}, \qquad M\ddot{Y} = D^T h_y + f_y + f_{py} \tag{4.2}$$

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 4.1 and inserting 4.2, the joint constraint forces cancel out, and the translational motion of the CM of the USM can be written as

$$m_t \ddot{p}_x = e^T (f_x + f_{px}), \qquad m_t \ddot{p}_y = e^T (f_y + f_{py}). \tag{4.3}$$

The equation of motion can be expressed as

$$\begin{aligned} M_\psi \ddot{\psi} + W_\psi \dot{\psi}^2 + V_\psi \dot{\psi} + \Lambda_3 |\dot{\psi}| \dot{\psi} - K_1 \mu (S_\psi e \ddot{p}_x - C_\psi e \ddot{p}_y) \\ + S_\psi K (f_{Dx} + f_{px}) - C\psi K (f_{Dy} + f_{py}) = D^T u \end{aligned} \tag{4.4}$$

where

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

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 4.3. 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.

4.1.2 Force allocation

The force allocation distribution is given by:

$$\tau_{CM} = \begin{bmatrix} F_{CM,x} \\ F_{CM,y} \\ M_{CM,z} \end{bmatrix} = T(\psi) f_p, \tag{4.5}$$

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. 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. (2016a) the force allocation matrix for 2D application can be expressed as:

$$T(\psi) = \begin{bmatrix} b_x^T \\ b_y^T \\ e^T S_\psi K B_X^T - e^T C_\psi K B_Y^T \end{bmatrix} \tag{4.6}$$

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.

4.2 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, by using a SMC algorithm, that calculates the desired forces for the translational motion F_{CM}. To calculate the desired moments for rotation motion of the USM M_{CM}, the P controller defined in Sverdrup-Thygeson et al. (2016a) will be used. The desired forces and moments is defines as:*

$$\tau_{CM,d} = \begin{bmatrix} F_{CM,d} \\ M_{CM,d} \end{bmatrix} \tag{4.7}$$

.

4.2.1 Sliding surface design

First of all an error variable has to be introduced. As the output variable for the translational motion of the USM is p_{CM}, the error variable can be defined as:

$$e = \begin{bmatrix} e_x \\ e_y \end{bmatrix} = p_{CM} - p_{CM,d} = \begin{bmatrix} p_x - p_{x,ref} \\ p_y - p_{y,ref} \end{bmatrix} \tag{4.8}$$

where $p_{CM,d}$ 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:

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_y \end{bmatrix} = \dot{e} + e = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \end{bmatrix} + \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} \dot{p}_x - \dot{p}_{x,ref} \\ \dot{p}_y - \dot{p}_{y,ref} \end{bmatrix} + \begin{bmatrix} p_x - p_{x,ref} \\ p_y - p_{y,ref} \end{bmatrix} \tag{4.9}$$

For the same reasons as for the mass-spring-damper system the u_{eq} controller will not be used her either. The states p_{CM} and $\dot{p}_{CM}$ is available for measurement.

4.2.2 Control input design

The control input $u = F_{CM,d}$. To find the sufficiently large constant K that will be mentioned in the following sections, the trial and error method was used for the same reasons as for the mass-spring-damper system.

The relay controller

When using the relay controller, the control input can be written as:

$$u = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = -K \begin{bmatrix} sgn(\sigma_x) \\ sgn(\sigma_y) \end{bmatrix} \tag{4.10}$$

where K is a sufficiently large positive constant and $sgn(\sigma)$ is defined as in equation 2.56.

The saturation controller

When using the saturation controller, the control input can be written as:

$$u = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = -K \begin{bmatrix} sat(\sigma_x) \\ sat(\sigma_y) \end{bmatrix} \tag{4.11}$$

where K is a sufficiently large positive constant, and $sat(\sigma)$ is defined as in equation 2.57 with $L = 0.3$.

The super-twisting algorithm

By choosing the gains in equation 2.91 to be $k_1 = 1.5\sqrt{K}$ and $k_2 = 1.1K$, where K is a sufficiently large positive constant. The control input can be written as:

$$\begin{aligned} u &= \begin{bmatrix} u_x \\ u_y \end{bmatrix} = -1.5\sqrt{K} \begin{bmatrix} |\sigma_x|^{1/2} sgn(\sigma_x) + v_x \\ |\sigma_y|^{1/2} sgn(\sigma_y) + v_y \end{bmatrix} \\ \dot{v} &= \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = -1.1K \begin{bmatrix} sgn(\sigma_x) \\ sgn(\sigma_y) \end{bmatrix} \end{aligned} \tag{4.12}$$

The super-twisting algorithm with adaptive gains

By using the STA with adaptive gains proposed in Shtessel et al. (2010) the control input can be written as:

$$\begin{aligned} u &= \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} -\alpha_x |\sigma_x|^{1/2} sgn(\sigma_x) + v_x \\ -\alpha_y |\sigma_y|^{1/2} sgn(\sigma_y) + v_y \end{bmatrix} \\ \dot{v} &= \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} -\beta_x sgn(\sigma_x) \\ -\beta_y sgn(\sigma_y) \end{bmatrix} \end{aligned} \tag{4.13}$$

where the adaptive gains are defined as:

$$\begin{aligned} \dot{\alpha} &= \begin{bmatrix} \dot{\alpha}_x \\ \dot{\alpha}_y \end{bmatrix} = \begin{bmatrix} \begin{cases} \omega_1 \sqrt{\frac{\gamma_1}{2}}, & \text{if } \sigma_x \neq 0 \\ 0, & \text{if } \sigma_x = 0 \end{cases} \\ \begin{cases} \omega_1 \sqrt{\frac{\gamma_1}{2}}, & \text{if } \sigma_y \neq 0 \\ 0, & \text{if } \sigma_y = 0 \end{cases} \end{bmatrix} \\ \beta &= \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix} = \begin{bmatrix} 2\varepsilon\alpha_x + \lambda + 4\varepsilon^2 \\ 2\varepsilon\alpha_y + \lambda + 4\varepsilon^2 \end{bmatrix} \end{aligned} \tag{4.14}$$

here $\varepsilon, \lambda, \gamma_1, \omega_1$ are arbitrary positive constants. As the sliding surface is nearly never equal to zero, a small boundary is put on the sliding surface for implementation purposes, and the adaptive gains can be expressed as:

$$\begin{aligned} \dot{\alpha} &= \begin{bmatrix} \dot{\alpha}_x \\ \dot{\alpha}_y \end{bmatrix} = \begin{bmatrix} \begin{cases} \omega_1 \sqrt{\frac{\gamma_1}{2}}, & \text{if } |\sigma_x| > \alpha_m \\ 0, & \text{if } |\sigma_x| \leq \alpha_m \end{cases} \\ \begin{cases} \omega_1 \sqrt{\frac{\gamma_1}{2}}, & \text{if } |\sigma_y| > \alpha_m \\ 0, & \text{if } |\sigma_y| \leq \alpha_m \end{cases} \end{bmatrix} \\ \beta &= \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix} = \begin{bmatrix} 2\varepsilon\alpha_x + \lambda + 4\varepsilon^2 \\ 2\varepsilon\alpha_y + \lambda + 4\varepsilon^2 \end{bmatrix} \end{aligned} \tag{4.15}$$

where the parameter α_m is an arbitrarily small positive constant, in this case chosen to be 0.05.

PD-controller

In Sverdrup-Thygeson et al. (2016a) a PD-controller is proposed to find $F_{CM,d}$:

$$u = F_{CM,d} = k_d^{CM} \begin{bmatrix} \dot{p}_{x,ref} - \dot{p}_x \\ \dot{p}_{y,ref} - \dot{p}_y \end{bmatrix} + k_p^{CM} \begin{bmatrix} p_{x,ref} - p_x \\ p_{y,ref} - p_y \end{bmatrix} \tag{4.16}$$

where k_d^{CM} and k_p^{CM} are controller gains. The PD-controller will be used to compare how well the SMC algorithms work compared to a linear controller with regards to disturbances and modelling errors.

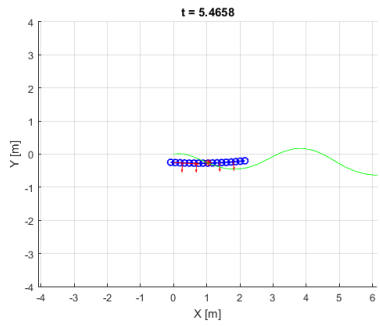
4.3 Implementation

4.3.1 System

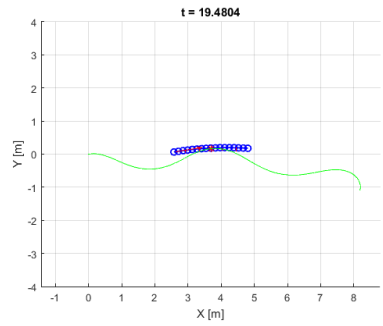
The complete model with force allocation matrix was implemented in MATLAB, this was done by Sverdrup-Thygeson et al. (2016a). The USM implemented has $n = 16$ links, each one having length $2l_i = 0.14\,m$ and mass $m_i = 0.6597\,kg$. The thruster configuration used is configuration 2 in Sverdrup-Thygeson et al. (2016a), this has one tail effector 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 see in Sverdrup-Thygeson et al. (2016a). See appendix A.2 for the code.

Torpedo like USM

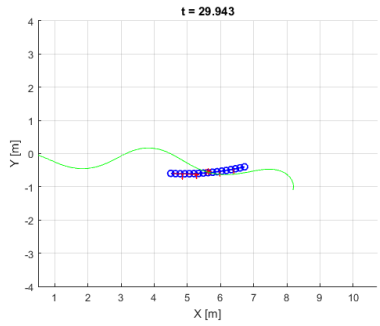
The USM will be used as a torpedo when it is moving from one place to another. To simulate this type of behaviour the link angels where set to zero, and a LOS guidance law was used for the heading. This was done by choosing no lateral undulation in file $calculate\_u\_lateral\_undulation.m$, i.e. $alpha = 0$. This type of simulation is shown in figure 4.3.



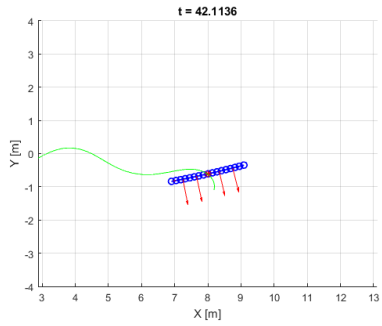
(a) Torpedo: t = 5



(b) Torpedo: t = 19



(c) Torpedo: t = 29

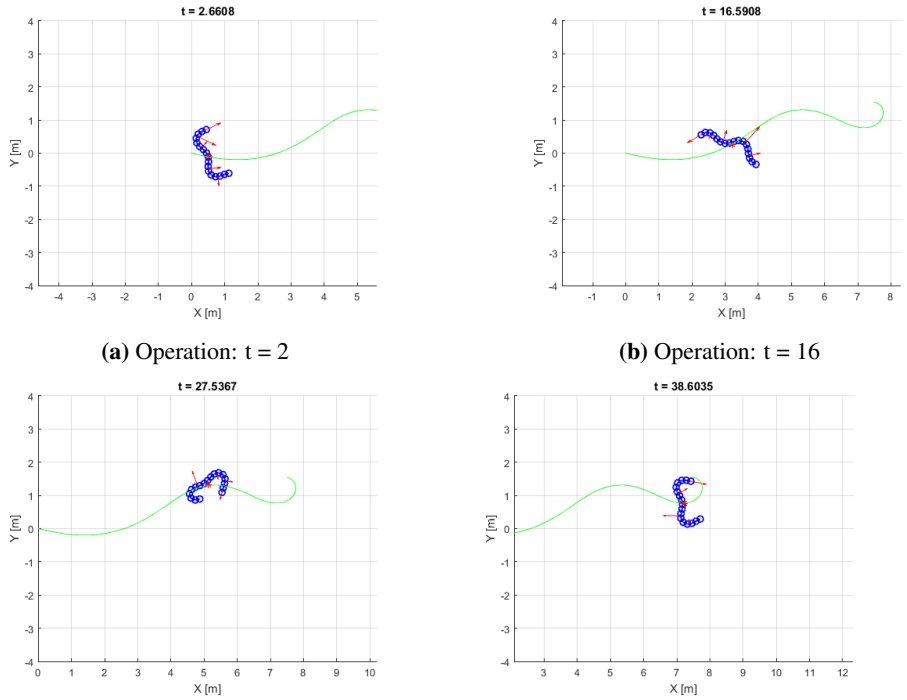


(d) Torpedo: t = 42

Figure 4.3: Torpedo like USM simulation

Operation like USM

When the USM is doing an operation, it will use the thrusters to stay in one place or move around and the end-effectors to do some type of operation. The undulation effects from the operation can be seen as a disturbance, as it will to some degree be random, depending on what type of operation the USM is performing. This means that the undulation effect will not help the USM to keep its place, it will make it harder. To simulate that the USM is doing an operation while the thrusters are keeping the USM on the reference path, the LOS guidance law was changed. This was done in $calculate\_u\_lateral\_undulation.m$ by changing $heading\_ref$ to $\sin(t)$ (random movement), and removing the no undulation restriction. This type of simulation is shown in figure 4.4.



(a) Operation: t = 2

(b) Operation: t = 16

(c) Operation: t = 27

(d) Operation: t = 38

Figure 4.4: Operation like USM simulation

4.3.2 Sliding surface and control input

The sliding surface was made by using the error variables available from the PD-controller that was previously implemented in $calculate\_desired\_forces\_moments.m$. The controllers were implemented in the same file. The code for the implementation is given in appendix A.2. The PD-controller that was previously implemented, was kept for comparing purposes.

4.4 Results

The relative and absolute error tolerance for the ode23tb solver that were used for the different simulations, can be found in table 4.1. This can be changed in $startSimulation\_ver3.m$.

Table 4.1: ode23tb solver: relative and absolute error tolerance

Algorithm	**Error**	
	Torpedo	Operation
Relay controller	10^{-1}	1
Saturation controller	10^{-6}	10^{-8}
Super-twisting algorithm	10^{-2}	10^{-2}
Super-twisting algorithm with adaptive gains	10^{-2}	10^{-1}
PD-controller	10^{-8}	10^{-4}

4.4.1 Torpedo like USM

The relay controller

The gain K for the relay control was set to 20.

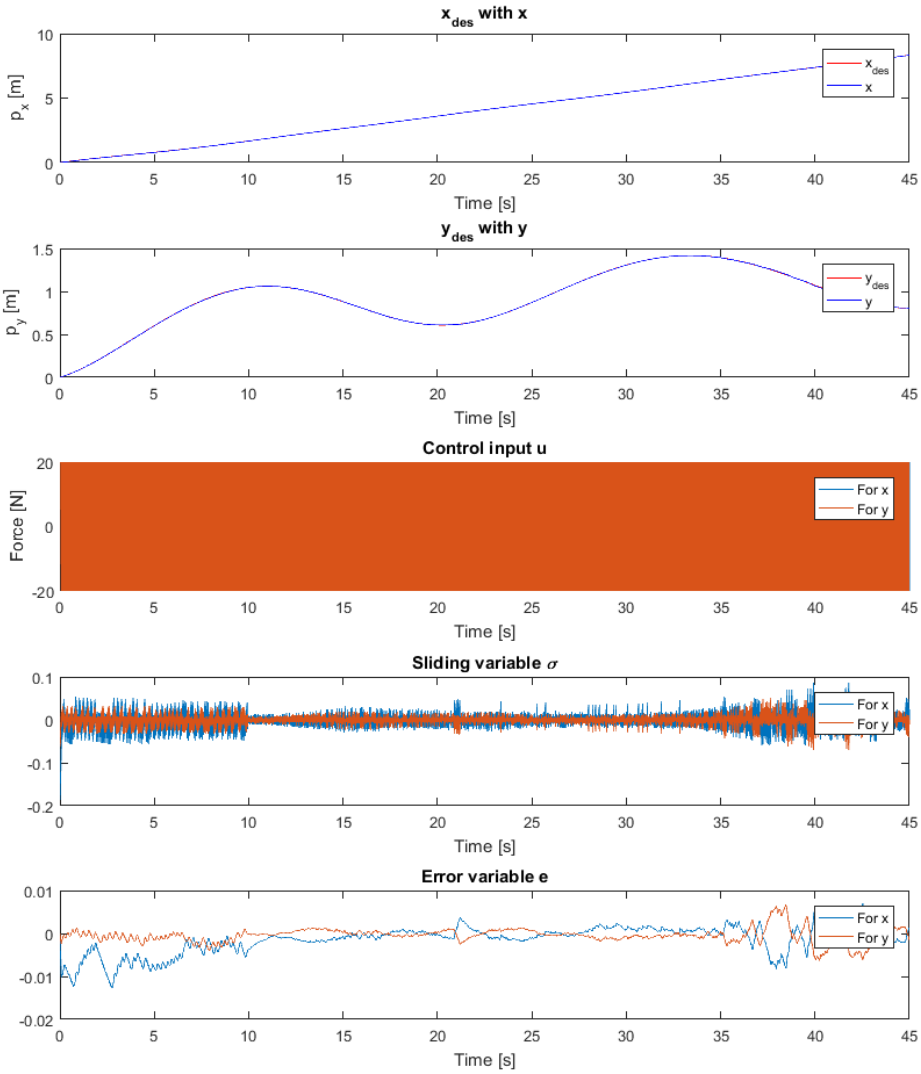


Figure 4.5: Results torpedo like USM: relay controller

The saturation controller

The gain K for the saturation control was set to 30.

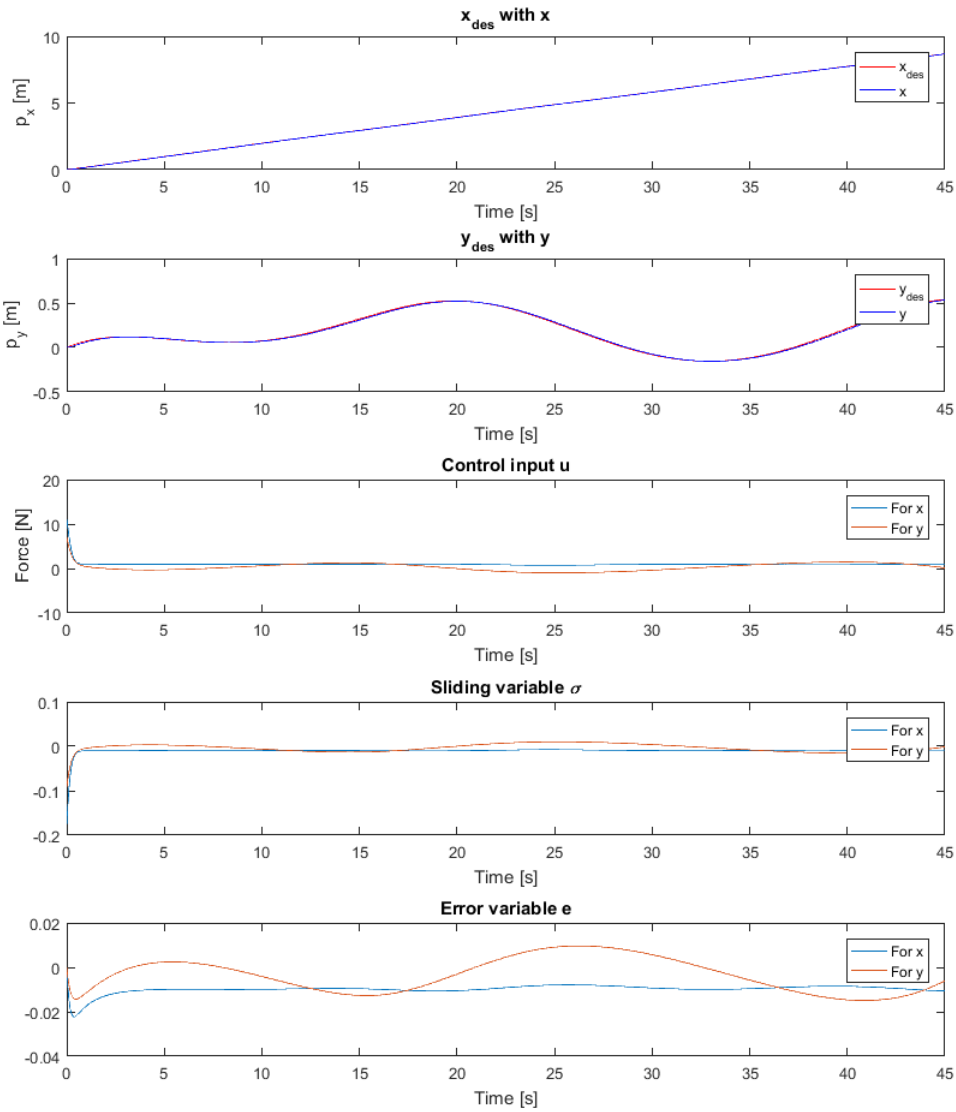


Figure 4.6: Results torpedo like USM: saturation controller

The super-twisting algorithm

The gain K for the super-twisting algorithm was set to 10.

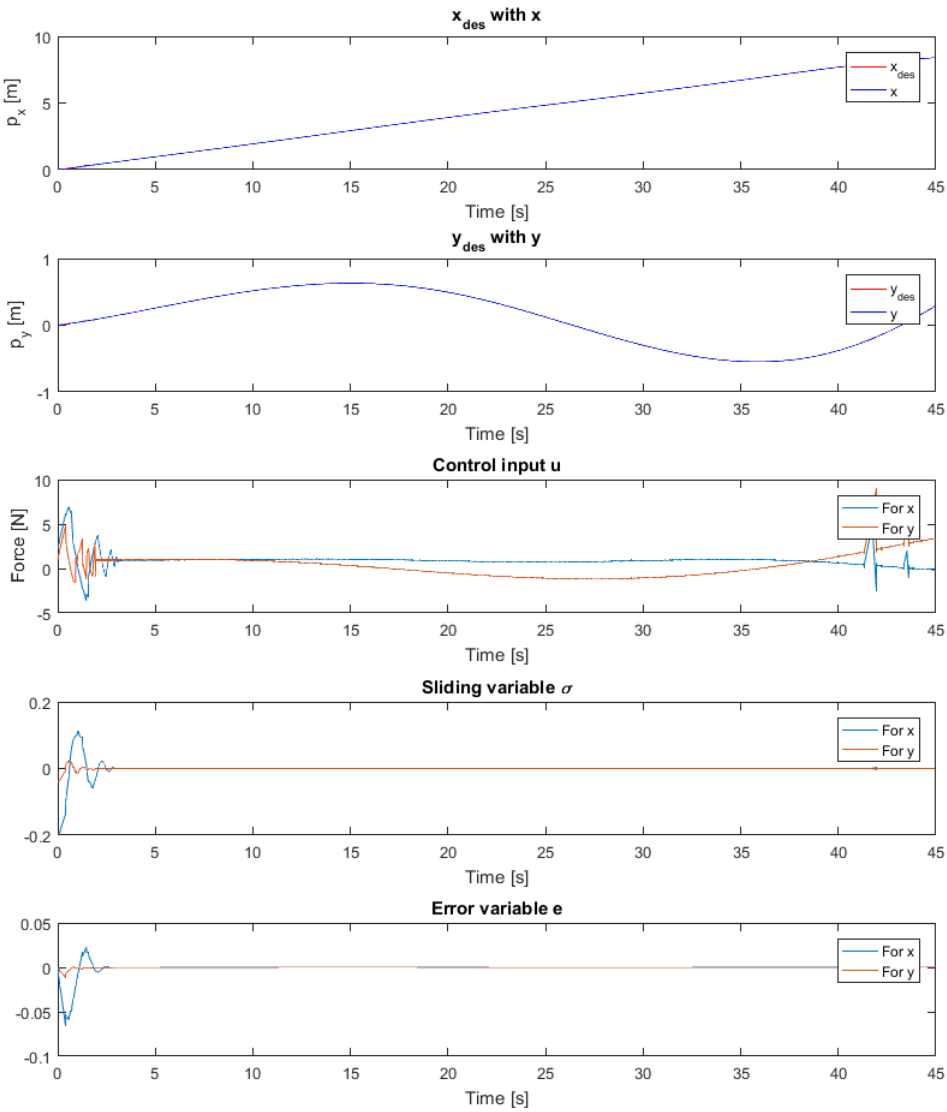


Figure 4.7: Results torpedo like USM: super-twisting algorithm

The super-twisting algorithm with adaptive gains

The gains for the super-twisting algorithm with adaptive gains were set to: $\epsilon = 1$, $\lambda = 1$, $\gamma_1 = 1$ and $\omega_1 = 5$.

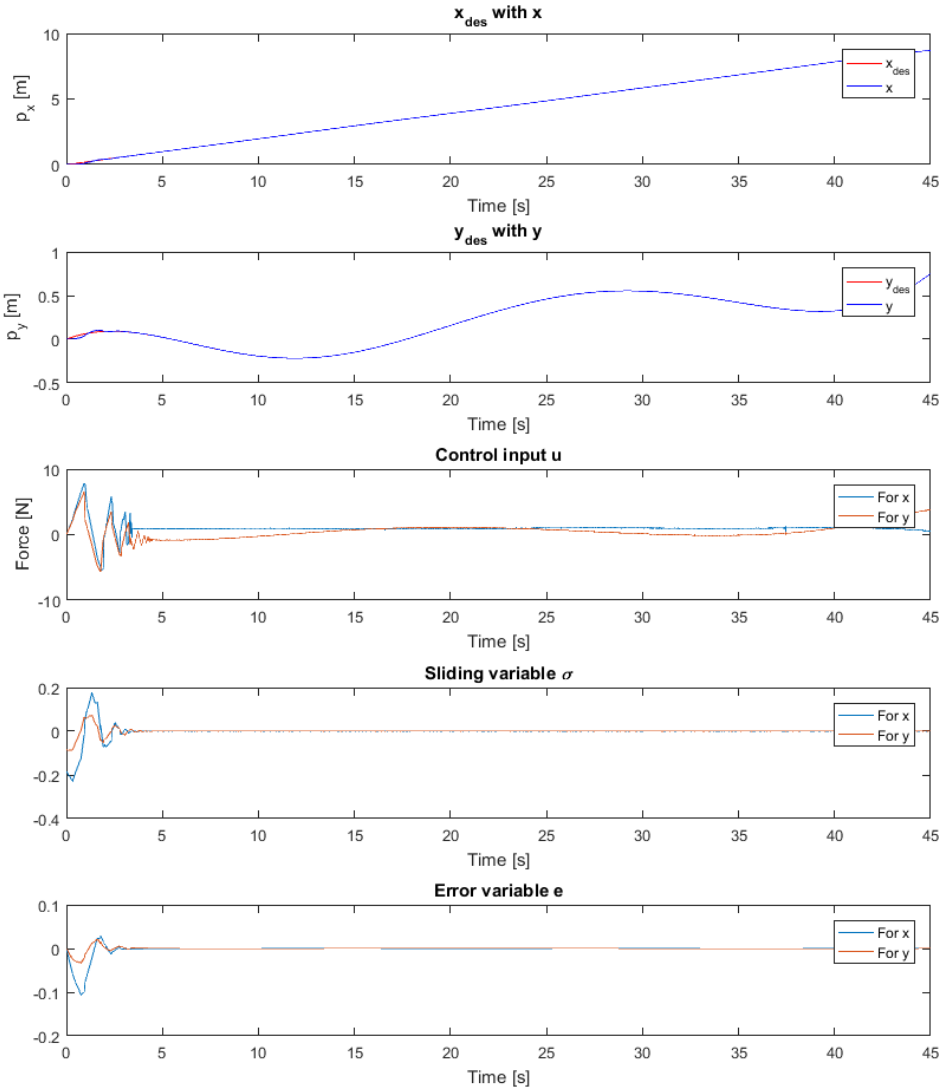


Figure 4.8: Results torpedo like USM: super-twisting algorithm with adaptive gains

The PD-controller

The gains for the PD-controller were set to the same as in Sverdrup-Thygeson et al. (2016a): $k_d^{CM} = 0.6$ and $k_p^{CM} = 0.06$.

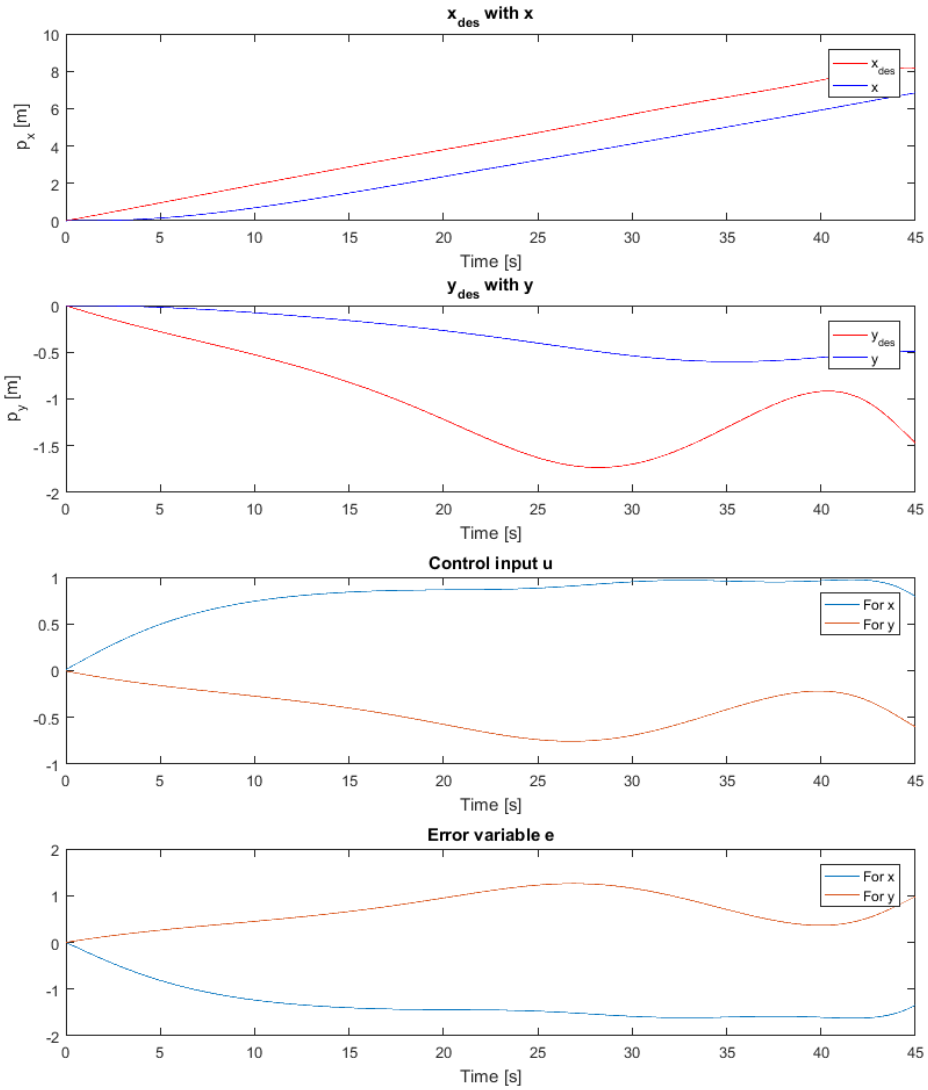


Figure 4.9: Results torpedo like USM: PD-controller

4.4.2 Operation like USM

The relay controller

The gain K for the relay control was set to 20.

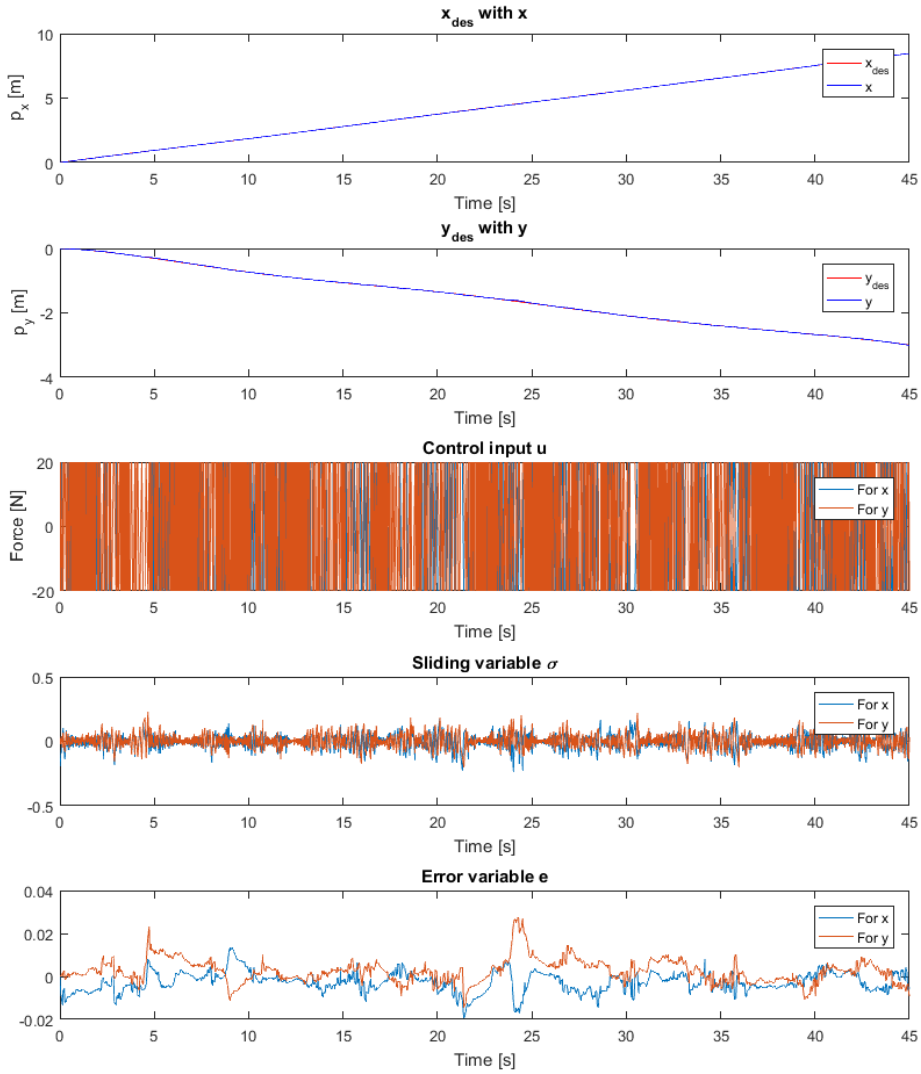


Figure 4.10: Results operation like USM: relay controller

The saturation controller

The gain K for the saturation control was set to 30.

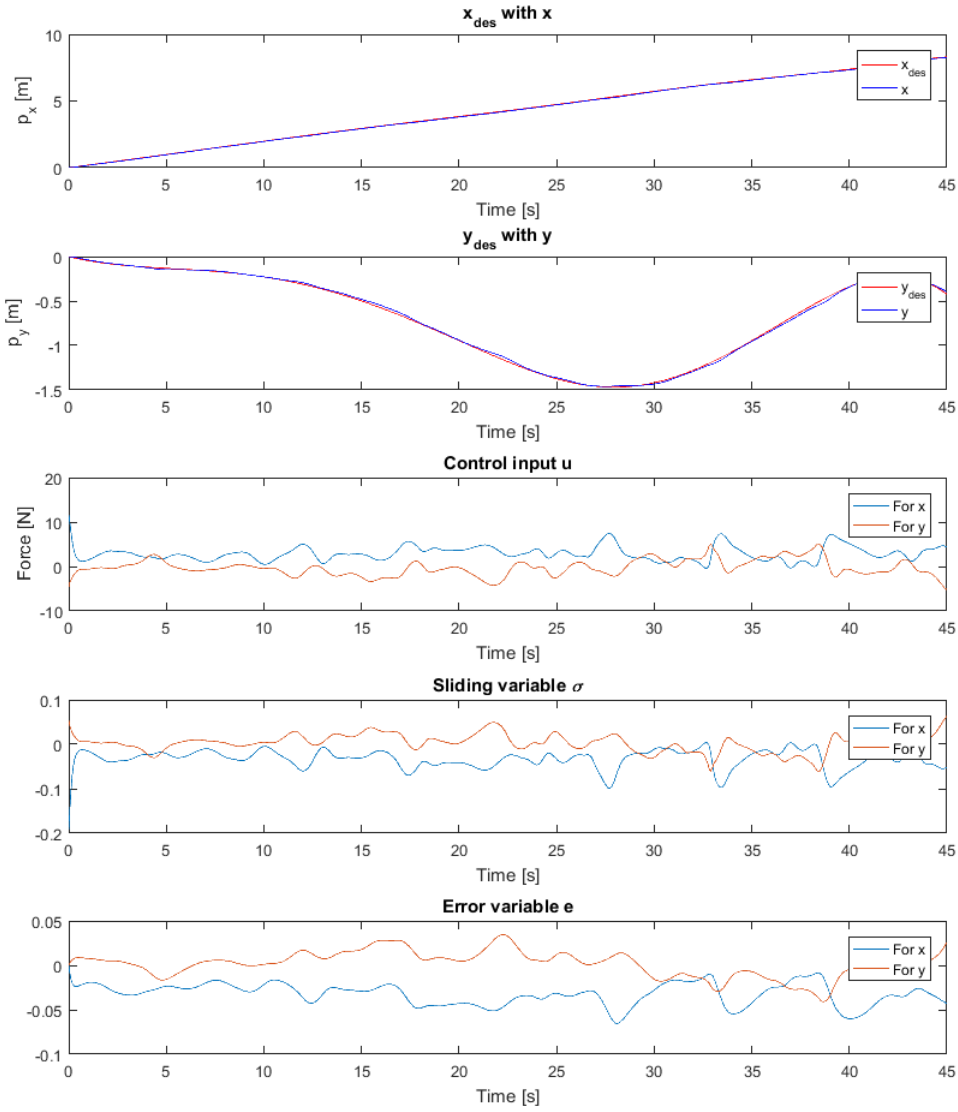


Figure 4.11: Results operation like USM: saturation controller

The super-twisting algorithm

The gain K for the super-twisting algorithm was set to 10.

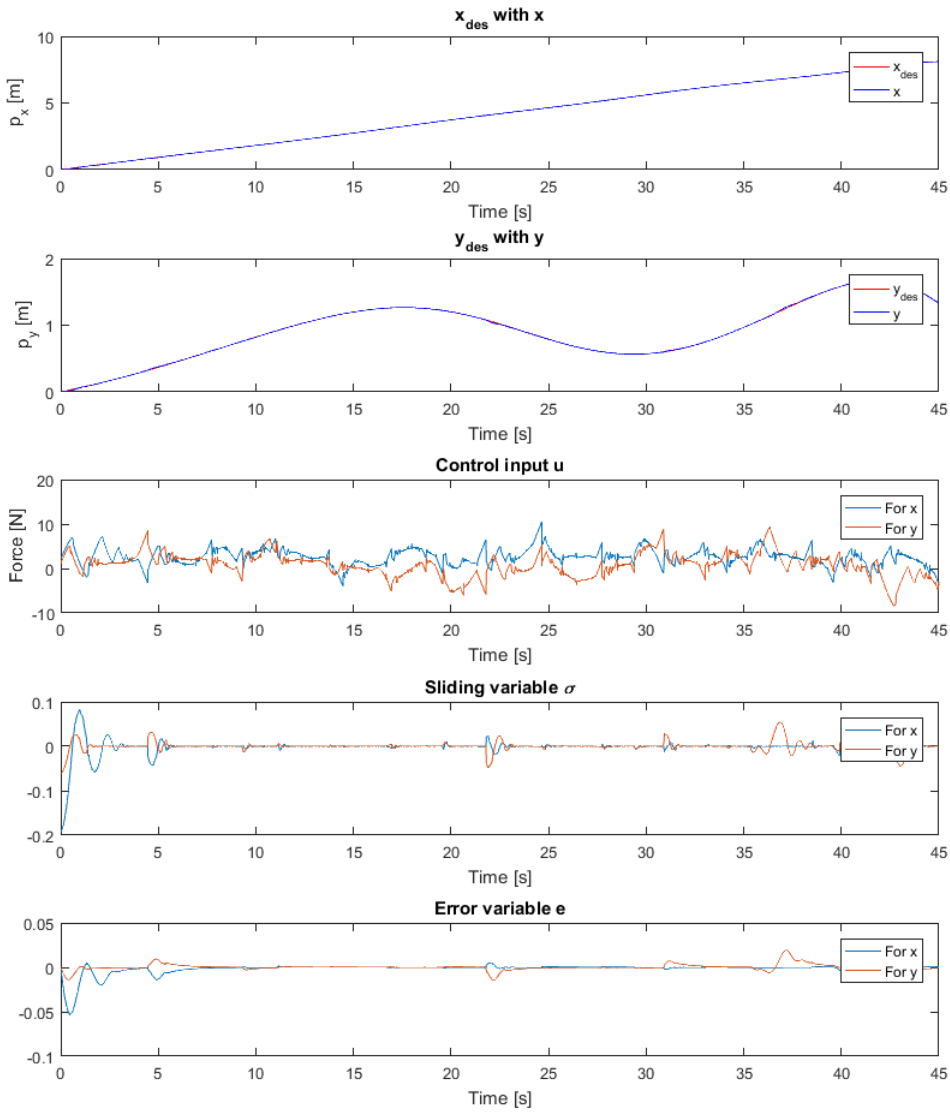


Figure 4.12: Results operation like USM: super-twisting algorithm

The super-twisting algorithm with adaptive gains

The gains for the super-twisting algorithm with adaptive gains were set to: $\epsilon = 1$, $\lambda = 1$, $\gamma_1 = 1$ and $\omega_1 = 5$.

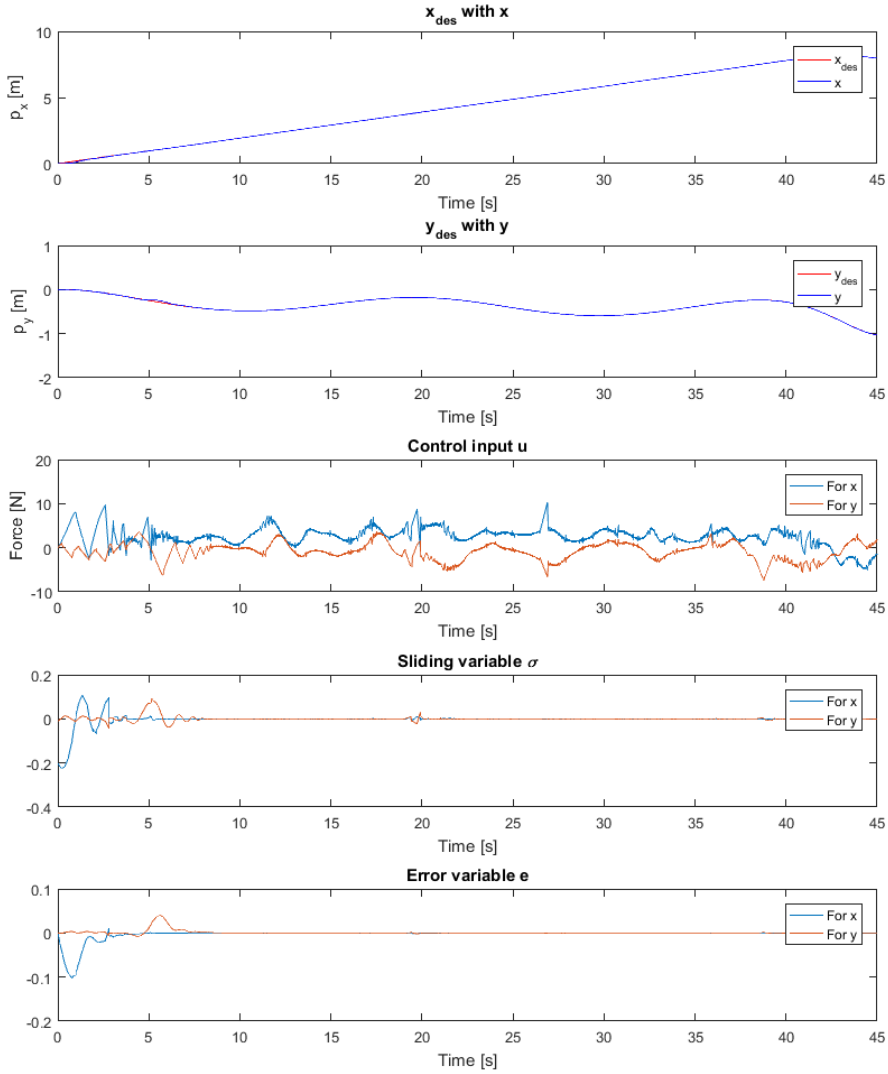


Figure 4.13: Results operation like USM: super-twisting algorithm with adaptive gains

The PD-controller

The gains for the PD-controller were set to the same as in Sverdrup-Thygeson et al. (2016a): $k_d^{CM} = 0.6$ and $k_p^{CM} = 0.06$.

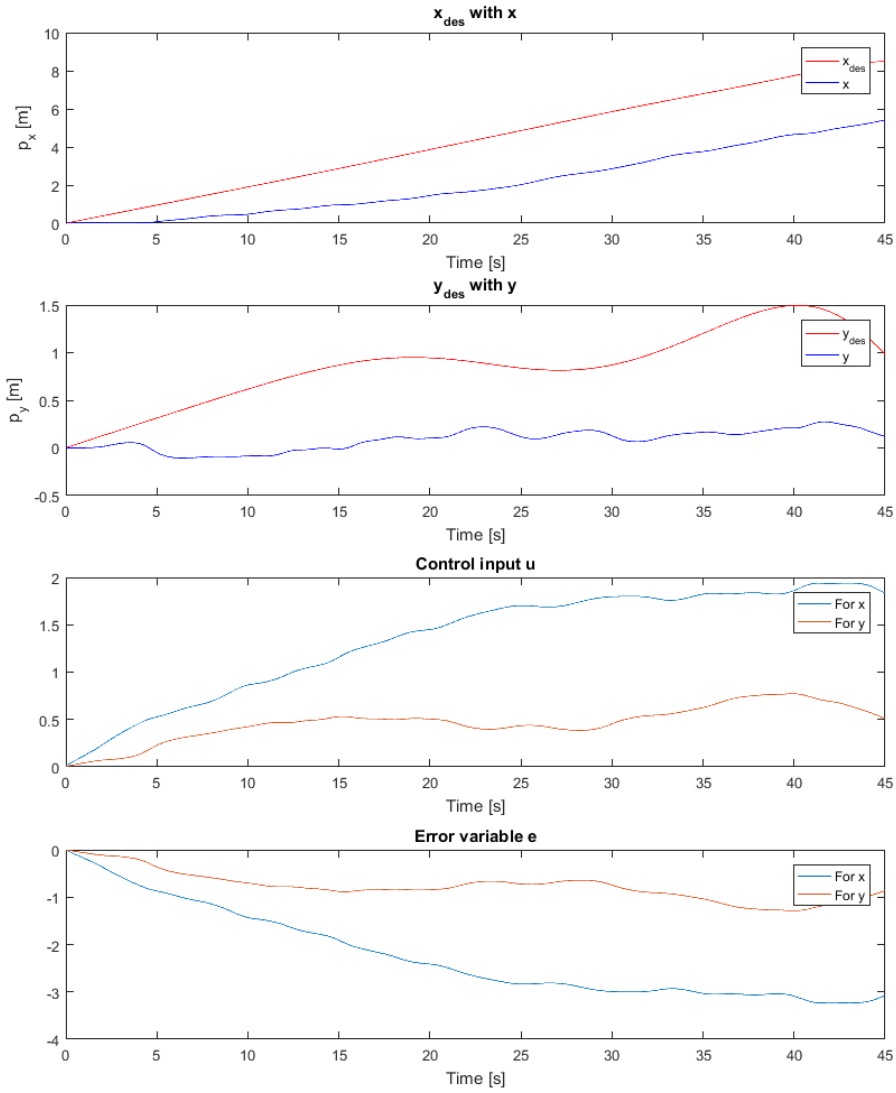


Figure 4.14: Results operation like USM: PD-controller

In order to better compare the different algorithms the absolute maximum value for the error variable for each algorithm have been gather in table 4.2. Here the first 5 seconds were not considered so that the absolute maximum error was found when the control input had stabilized.

Table 4.2: USM: absolute maximum value for error variable

Algorithm	**Error**			
	Torpedo		Operation	
	x	y	x	y
The relay controller	0.0082	0.0065	0.0195	0.0273
The saturation controller	0.0222	0.0149	0.0654	0.0407
The super-twisting algorithm	$6.5885 \cdot 10^{-4}$	$9.1247 \cdot 10^{-4}$	0.0053	0.0246
The super-twisting algorithm with adaptive gains	$1.5632 \cdot 10^{-4}$	$6.5611 \cdot 10^{-5}$	0.0019	0.0020
PD-controller	1.6230	1.2579	3.2339	1.2868

Chapter 5

Discussion

This chapter will present a discussion regarding the four different algorithms that have been tested on the two different systems, and a comparison between the different SMC algorithms. The SMC algorithms will also be compared to the PD controller that was implemented in Sverdrup-Thygeson et al. (2016a).

5.1 The relay controller

The relay controller gave great results both for the mass-spring-damper system and for the USM. It makes the state trajectories follow the desired path very nicely, and it has a small error variable, this means it handles the disturbance and modelling errors very well. But the controller has some issues. First of all the chattering in the control input is a big problem. If this control input was given to a real system, the mechanics would most likely be destroyed. Second, it needs a very high control gain to be able to handle the disturbance. The optimal gain is also very difficult to find, this is because it needs to be sufficiently large and it needs to satisfy the reaching law. This can be difficult when the disturbance is unknown.

5.2 The saturation controller

The saturation controller also gave great results for both systems. The state trajectories follow the desired path, and the error variable is small, but it is a little bit bigger than for the relay controller. This means it handles disturbance and modelling errors very well, but not as well as the relay controller. The control input is however much smoother for the saturation controller than it is for the relay controller. This is because of the linear control that takes place within the boundary layer of the saturation controller. This linear control is also the reason why the controller has a larger error variable. The saturation controller also need high gains, and it is also difficult to find the optimal gain, this is for the same reasons as the relay controller.

5.3 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. The error variable is even smaller than for the relay controller. This means it handles disturbances and modelling errors very well. The control input for the mass-spring-damper system and for the torpedo like USM is very smooth and there is no sign of chattering. The control input for the operation like USM is however intense and varying, even more so than for the saturation controller. This is most likely because the error variable is noticeably smaller than for the saturation controller. This means that the super-twisting controller can handle a great deal of disturbance, but it will affect the smoothness of the control input. With this controller it is also a problem that the control gain needs to be very high and that it is difficult finding the optimal gain. This is because a bound on the disturbance needs to be known, and that can be hard to find when the disturbance is unknown. This often gives a very crude estimation of what the gain needs to be which leads to a much higher gain than the optimal one.

5.4 The super-twisting algorithm with adaptive gains

The super-twisting algorithm with adaptive gains has the same control abilities as the super-twisting algorithm with constant gain. The only difference lies in the how the control gain is found. For the super-twisting algorithm with adaptive gains a bound on the disturbance does not need to be known, that is because the gains adapts, i.e. finds the optimal gain. All the user has to do is select the gains depending on how fast the rate of convergence need to be. In the end the optimal gain will always be found, but the convergence time may vary depending on the control gains.

5.5 Comparison between the sliding mode control algorithms

All four algorithms manage to follow the desired path without any large errors, but the second-order algorithms were the ones with the smallest error. The biggest difference in the algorithms is regarding the control input and control gains. The relay controller suffers from severe chattering. The saturation controller gives smooth control input, but the error variable is considerably larger. The super-twisting algorithm also gives a smooth control input, but with a smaller error variable. The control gains for these three algorithms are hard to find, and it is even more difficult to find an optimal control gain. The super-twisting algorithm with adaptive gains gives a smooth control input as the super-twisting algorithm with constant gains, but it also solves the problem regarding the control gain.

The simulation results from the operation like USM gives some strange control inputs for all of the controllers. That is because of the amount of disturbance that is inflicted on the system by letting the USM move in a sine-wave. This is more visible in both the super-twisting algorithm. This is because the error variable is noticeably smaller for these two controllers. The movement of the USM will never be that severe and the control

inputs would therefore probably never be that bad. But it was a great way to show how much these types of controllers actually can handle regarding disturbances even though the control input did not look so great.

5.6 Comparison between the PD-controller and sliding mode control

The PD-controller could not handle a torpedo like USM or an operational like USM. There were large errors and offsets on both simulation modes. This means that the SMC algorithms are much more suited for these type of systems. The PD-controller gains were chosen by the previous users, and in Sverdrup-Thygeson et al. (2016a) there is only simulations results for a straight line. Therefore the gains might not be optimal for the type of simulations that were performed here. It is therefore possible that by changing the gains or replacing the PD with a PID controller that the linear controller would have given the SMC more competition regarding results.

Chapter 6

Conclusion and Further Work

6.1 Conclusion

In this report an in-depth study of sliding mode control was presented. It explains sliding mode control in general and several first, second and higher-order sliding mode control algorithms in detail. Four different SMC algorithms were tested on two different systems to see how well they performed. They have also been compared to a PD-controller.

All the SMC algorithms gave great results regarding their ability to follow a desired path when disturbances were present. However, the relay controller cannot be used in practice because of chattering in the control input. The saturation controller and the second-order algorithms can be used, as they all have smooth control inputs. Out of these, the second-order algorithms are the best as they have the smallest error variable. As the saturation controller and the super-twisting algorithm have problems with finding the optimal gain, the super-twisting algorithm with adaptive gains is the best choice for practical use. The PD-controller gave no competition to the SMC algorithms.

6.2 Further work

The higher-order sliding mode control algorithms mentioned in section 2.4 should also be tested with the help of observers (differentiators). The algorithms tested here, especially the super-twisting algorithm with adaptive gains, should also be tested on a 3D-model of the USM to see if it gives as good results as it does in 2D. The algorithms should also be tested in a situation where the USM is moving in a way that is more similar to the way the USM actually will move under an operation, to get a clearer picture of how the control input will look like under an operation.

Bibliography

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.

Basin, M., Panathula, C. B., Shtessel, Y., 2016. Adaptive uniform finite-/fixed-time convergent second order sliding mode control. International Journal of Control, 1–17.

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.

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., 2016a. Adaptive dual-layer super-twisting control and observation. International Journal of Control, 1–8.

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

Filippov, A. F., Arscott, F. M., 1988. Differential equations with discontinuous righthand sides. Kluwer Academic Publishers, Dordrecht.

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

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

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

Levant, A., 2005. Homogeneity approach to high-order sliding mode design. Automatica 41 (5), 823–830.

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

Plestan, F., Moulay, E., Glumineau, A., Cheviron, T., 2010. Robust output feedback sampling control based on second-order sliding mode. Automatica 46 (6), 1096–1100.

Rezapour, E., Pettersen, K. Y., Liljeback, P., Gravdahl, J. T., 2014. Differential geometric modelling and robust path following control of snake robots using sliding mode techniques.

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., 2010. Super-twisting adaptive sliding mode control: A lyapunov design. In: 2010 49th IEEE Conference on Decision and Control, CDC 2010. pp. 5109–5113.

Sverdrup-Thygeson, J., Kelasidi, E., Pettersen, K. Y., Gravdahl, J. T., 2016a. Modeling of underwater swimming manipulators. IFAC PapersOnLine 49 (23), 81–88.

Sverdrup-Thygeson, J., Kelasidi, E., Pettersen, K. Y., Gravdahl, J. T., 2016b. Sliding mode control of underwater swimming manipulators with thrusters. Submitted to: IEEE International Conference on Robotics and Automation (Unpublished).

Utkin, V. I., 1977. Survey paper: Variable structure systems with sliding modes. IEEE Transactions on Automatic Control 22 (2), 212–222.

Yan, X., Primot, M., Plestan, F., 2016. An unified formalism based on gain switching for second order sliding mode control. In: 2016 14th International Workshop on Variable Structure Systems (VSS). pp. 71–76.

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

Appendix A

Attachment Description and MATLAB code

Attached to this report 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 Mass-spring-damper system, and one named Underwater swimming manipulator in each folder the code for each system can be found.

A.1 Mass-spring-damper system

A.1.1 Attachment description

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

.m-files

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

The relay controller

- $relay\_control\_MSDS.slx$ - Contains the implementation of the relay controller and the mass-spring-damper system.
- $run\_relay\_control\_MSDS.m$ - Initializes the parameters, runs $relay\_control\_MSDS.slx$ and plots the results.

The saturation controller

- $saturation\_control\_MSDS.slx$ - Contains the implementation of the saturation controller and the mass-spring-damper system.
- $run\_saturation\_control\_MSDS.m$ - Initializes the parameters, runs $saturation\_control\_MSDS.slx$ and plots the results.

Super-twisting algorithm

- $super\_twisting\_algorithm\_MSDS.slx$ - Contains the implementation of the super-twisting algorithm and the mass-spring-damper system.
- $run\_super\_twisting\_algorithm\_MSDS.m$ - Initializes the parameters, runs $super\_twisting\_algorithm\_MSDS.slx$ and plots the results.

Super-twisting algorithm with adaptive gains

- $super\_twisting\_adaptive\_gains\_MSDS.slx$ - Contains the implementation of the super-twisting algorithm with adaptive gains and the mass-spring-damper system.
- $run\_super\_twisting\_adaptive\_gains\_MSDS.m$ - Initializes the parameters, runs $super\_twisting\_adaptive\_gains\_MSDS.slx$ and plots the results.

A.1.2 MATLAB code

The relay controller: $run\_relay\_control\_MSDS.m$

```
%% Run relay controller

% Parameter values

p = 1;
m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Input
K = 60;

sim('relay_control_MSDS');

%% Plot

figure(1)
subplot(5,1,1);
plot(y_des.time,y_des.signals.values,'r')
hold on
plot(y.time,y.signals.values,'b')
title('Y_{des} with Y'); xlabel('Time [s]');ylabel('Position of mass');
legend('y_{des}','y');
```

```
subplot(5,1,2);
plot(disturbance.time,disturbance.signals.values)
title('Disturbance'); xlabel('Time [s]');ylabel('Force');

subplot(5,1,3);
plot(control_input.time,control_input.signals.values)
title('Control input u'); xlabel('Time [s]');ylabel('Force');

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

subplot(5,1,5);
plot(e.time,e.signals.values)
title('Error variable e'); xlabel('Time [s]');
```

The saturation controller: $run\_saturation\_control\_MSDS.m$

```
%% Run saturation control

% Parameter values

p = 1;
m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m


% Input
K = 65;

sim('saturation_control_MSDS');

%% Plot

figure(1)
subplot(5,1,1);
plot(y_des.time,y_des.signals.values,'r')
hold on
plot(y.time,y.signals.values,'b')
title('Y_{des} with Y'); xlabel('Time [s]');ylabel('Position of mass');
legend('y_{des}','y');

subplot(5,1,2);
plot(disturbance.time,disturbance.signals.values)
title('Disturbance'); xlabel('Time [s]');ylabel('Force');

subplot(5,1,3);
plot(control_input.time,control_input.signals.values)
title('Control input u'); xlabel('Time [s]');ylabel('Force');

subplot(5,1,4);
plot(sliding_var.time,sliding_var.signals.values)
title('Sliding variable \sigma'); xlabel('Time [s]');;
```

```
subplot(5,1,5);
plot(e.time,e.signals.values)
title('Error variable e'); xlabel('Time [s]');
```

Super-twisting algorithm: *run_super_twisting_algorithm_MSDS.m*

```
%% Run super-twisting algorithm

% Parameter values

p = 1;
m = 2; % Kg
c = 5; % N/ms*2
k = 2; % N/m

% Input
K = 100;

sim('super_twisting_algorithm_MSDS');

%% Plot

figure(1)
subplot(5,1,1);
plot(y_des.time,y_des.signals.values,'r')
hold on
plot(y.time,y.signals.values,'b')
title('Y_{des} with Y'); xlabel('Time [s]');ylabel('Position of mass');
legend('y_{des}','y');

subplot(5,1,2);
plot(disturbance.time,disturbance.signals.values)
title('Disturbance'); xlabel('Time [s]');ylabel('Force');

subplot(5,1,3);
plot(control_input.time,control_input.signals.values)
title('Control input u'); xlabel('Time [s]');ylabel('Force');

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

subplot(5,1,5);
plot(e.time,e.signals.values)
title('Error variable e'); xlabel('Time [s]');
```

Super-twisting algorithm with adaptive gains:
run_super_twisting_adaptive_gains_MSDS.m

```
%% Run super-twisting algorithm with adaptive gains
% Parameter values

p = 1;
m = 2; % Kg
```

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

% Control variables
epsilon = 1;
lamda =  1;
gamma_1 = 1;
omega_1 = 20;



sim('super_twisting_algorithm_with_adaptive_gains_MSDS');

%% Plot

figure(1)
subplot(5,1,1);
plot(y_des.time,y_des.signals.values,'r')
hold on
plot(y.time,y.signals.values,'b')
title('Y_{des} with Y'); xlabel('Time [s]');ylabel('Position of mass');
legend('y_{des}','y');

subplot(5,1,2);
plot(disturbance.time,disturbance.signals.values)
title('Disturbance'); xlabel('Time [s]');ylabel('Force');

subplot(5,1,3);
plot(control_input.time,control_input.signals.values)
title('Control input u'); xlabel('Time [s]');ylabel('Force');

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

subplot(5,1,5);
plot(e.time,e.signals.values)
title('Error variable e'); xlabel('Time [s]');
```

Absolute maximum error: $find\_absolute\_max\_error.m$

```
%% Find absoulte maximum error

max_error = max(abs(e.signals.values(5000:50001)))
```

A.2 Underwater swimming manipulator

A.2.1 Attachment description

The files that are described and attached here, are the files where changes were made. The code for the USM is organized as follows:

.m-files

- $startSimulation\_ver3.m$ - Starts the simulation of the USM. Here the relative and absolute error for the ode23tb can be changed.
- $calculate\_desired\_forces\_moments.m$ - Calculates and returns the desired forces and moments, by calculating the sliding surface and the control input. To switch between the different algorithms, change the variable con.
- $calculate\_u\_lateral\_undulation.m$ - Calculates and returns the actuator forces. Change the variable $heading\_ref$ and uncomment $alpha = 0$ to switch between torpedo mode and operation mode.
- $sat.m$ - Saturation function.
- $plot\_simulations.m$ - Plots the simulation results. Run it after $startSimulation\_ver3.m$.
- $find\_absolute\_max\_error.m$ - Finds the absolute maximum error. Run it after $startSimulation\_ver3.m$.

A.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


% The stop time for the simulation.
stop_time = 45; % 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 ; alpha_CM0_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

options = odeset('Events', 'off', 'RelTol', 1e-2, 'AbsTol', 1e-2);
[t, v] = ode23tb('calculate_v_dot', [0 stop_time], v0, options);

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 sliding surface and control input: *calculate_desired_forces_moments.m*

```
function [tau_d, u_CM_2_dot, alpha_CM_dot] = calculate_desired_forces_moments(t, theta,
theta_dot, p_CM, p_CM_dot,u_CM_2, alpha_CM)
% 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


if t <= 0
    tau_d = [0;0;0];
    u_CM_2_dot = zeros(2,1);
    alpha_CM_dot = zeros(2,1);
end

%Choose controller: con = 1 (relay), con = 2 (saturation), con = 3
%(super-twisting), con = 4 (super-twisting with adaptive gains),
% con = 5 (PD-controller)
con = 3;


% 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);

    %Only for super-twisting with adaptive gains
    alpha_CM_dot = zeros(2,1);

    % Line-of-Sight calculation for heading reference
    Delta=2.24;
    heading_d = -atan2(p_CM(2,1),Delta);

    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);

    error_pos_CM = p_CM - p_CM_d;
    error_vel_CM = p_CM_dot - p_CM_dot_d;
    error_heading = heading_d - heading;

    % Create sliding surfaces
```

```
    sigma_f = error_vel_CM + error_pos_CM;

    if con == 1 %Relay controller
        K_f = 20;

        for i = 1:2
            F_CM(i) = -K_f*sign(sigma_f(i));
        end

    elseif con == 2 %Saturation controller
        K_f = 30;
        L = 0.3;
        F_CM = -K_f*sat(sigma_f,L);

    elseif con == 3 %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));
            F_CM(i) = u_CM_1(i) + u_CM_2(i);
        end

    elseif con == 4 %Super-Twisting controller with adaptive gains
        %Control paramters
        beta_CM = zeros(2,1);

        epsilon = 1;
        lamda =  1;
        gamma_1 = 1;
        omega_1 = 5;

        for i = 1:2
            if abs(sigma_f(i)) <= 0.05
                alpha_CM_dot(i) = 0;
            else
                alpha_CM_dot(i) = omega_1*sqrt(gamma_1/2);
            end

            beta_CM(i) = 2*epsilon*alpha_CM(i) + lamda + 4*epsilon^2;
            u_CM_1(i) = -alpha_CM(i)*sqrt(abs(sigma_f(i)))*sign(sigma_f(i));
            u_CM_2_dot(i) = -beta_CM(i)*sign(sigma_f(i));
            F_CM(i) = u_CM_1(i) + u_CM_2(i);
        end
    else %PD-controller
        Kp_pos = 0.6;
        Kd_pos = 0.06;

        F_CM = Kd_pos*(-1)*error_vel_CM + Kp_pos*(-1)*error_pos_CM;
    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;

% Line-of-Sight heading reference
Delta=2.24;
heading_ref = sin(t); % To get line-of-sight/torpedo mode:
%-atan2(p_CM(2,1),Delta). To get opeation mode: sin(t).

% 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);
%alpha = 0;  % No lateral undulation

% 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
```

Saturation function: $sat.m$

```
function y = sat(sigma,L)

var = zeros(2,1);

for i = 1:2
    var(i) = (sigma(i)/(abs(sigma(i))+L));
end

y = var;
```

Plot simulations: $plot\_simulations.m$

```
%% Plot for SMC algorithms

figure(1)
subplot(5,1,1);
plot(t_sim,p_CM_d_sim(:,1),'r')
hold on
plot(t_sim,px_sim,'b')
title('x_{des} with x'); xlabel('Time [s]');ylabel('p_x [m]');
legend('x_{des}','x');

subplot(5,1,2);
plot(t_sim,p_CM_d_sim(:,2),'r')
hold on
plot(t_sim,py_sim,'b')
title('y_{des} with y'); xlabel('Time [s]');ylabel('p_y [m]');
legend('y_{des}','y');

subplot(5,1,3);
plot(t_sim,F_CM_sim)
title('Control input u'); xlabel('Time [s]');ylabel('Force [N]');
legend('For x','For y');


subplot(5,1,4);
plot(t_sim,sigma_f_sim)
title('Sliding variable \sigma'); xlabel('Time [s]');
legend('For x','For y');

subplot(5,1,5);
plot(t_sim,error_pos_CM_sim)
title('Error variable e'); xlabel('Time [s]');
legend('For x','For y');

%% Plot for PD-controller

figure(2)
subplot(4,1,1);
plot(t_sim,p_CM_d_sim(:,1),'r')
hold on
plot(t_sim,px_sim,'b')
```

```
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')
hold on
plot(t_sim,py_sim,'b')
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)
title('Control input u'); xlabel('Time [s]');
legend('For x','For y');

subplot(4,1,4);
plot(t_sim,error_pos_CM_sim)
title('Error variable e'); xlabel('Time [s]');
legend('For x','For y');
```

Absolute maximum error: $find\_absolute\_max\_error.m$

```
%% Find absolute maximum error

max_error_x = max(abs(error_pos_CM_sim(2000:11784,1)))
max_error_y = max(abs(error_pos_CM_sim(2000:11784,2)))
```