NTNU
Norwegian University of Science and Technology

Biological Cell Models and Atomic Force Microscopy

Parameter Estimation with Parallel Computing

Kristin Bræck Leer

Master of Science in Cybernetics and Robotics
Submission date: December 2016
Supervisor: Jan Tommy Gravdahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

Through several decades, mathematical models have been used to describe real systems. Studying these mathematical models can give us important information about the system and its behavior. The Atomic Force Microscopy (AFM) has been described with a mathematical model and the hope is to identify and estimate unknown parameters.

This thesis presents a parameter estimation method for identifying unknown parameters in the system using parallel computing. Introducing a variety of mathematical expression that give base to a set of analyzing tools. These were used to evaluate how the estimated parameter converged to the real values. These simulations were split into experiments, that changed one or multiple parameters that affected the properties of the system.

Our experiments illustrated how we can improve the convergence of estimated parameters by tuning parameters that effect the properties of the system with basic analyzing tools (bias, relative tolerance and rate of convergence). The simulations performed, based on previous work, found a gain matrix were the estimated parameters converged exponentially fast to the real values. The results shows that our system contains two local minimizers when optimizing the gain matrix.

The cantilever dynamics were described in a linear-in-parameter form, and both the known and unknown parameters were defined along with a filter. This means that the cantilever dynamic can be simulated when finding a input signal that is PE.

Sammendrag

Gjennom flere tiår har det vært vanlig å beskrive virkelige systemer med matematiske modeller. Ved å studere disse modellen kan man finne viktig informasjon om systemet og dens oppførsel. Atomic Force Microscopy (AFM) har blitt beskrevet ved en matematisk modell og man håper å kunne identifisere og estimere ukjente variabler.

Denne oppgaven presenterer en parameter estimeringsmetode for identifiksjon av ukjent variabler i et system ved bruke av parallell databehandling. Ved å introdusere en rekke matematiske uttrykk som gir basisen for et sett med analyseringsmetoder. Disse analyseringsmetodene ble brukt til å evaluere hvordan de estimerte variablene konvergerte til de virkelige verdiene. Disse simuleringene var delt opp i eksperiment, hvor en eller flere variabler ble endret. Disse variablene påvirker egenskapene til systemet.

Våre eksperimenter illustrerer hvordan vi kan forbedre konvergeringen til de estimerte variablene ved å tune variabler som påvirker egenskapene til systemet ved hjelp av analyseringsmetoder (bias, relativ toleranse og konvergeringshastighet). Basert på tidligere arbeid, ble en forsterkningsmatrise funnet hvor de estimerte variablene konvergerte eksponentielt mot de virkelige verdiene. Resultatene viser at system vårt inneholder to lokale minimum når man optimaliserer forsterkningsmatrisen.

Cantilever dynamikken ble beskrevet i ”linear-in-parameter” form, der både kjente og ukjente variabler og et filter ble definert. Dette betyr at cantilever dynamikken kan bli simulert når man finner et inngangssignal som er PE.

Preface

This thesis is the last piece to completing my master's degree in Engineering Cybernetics at NTNU, carried out during the autumn semester of 2016.

I would like to thank my supervisor Professor Jan Tommy Gravdahl for inspiration and constructive criticism. I also want to thank my co-supervisor, PhD-student Michael R. P. Ragazzon for ideas, guidance and feedback, and also for answering questions at any time.

This thesis was supported in part with computational resources at NTNU provided by NOTUR, and I would like to thank the staff there for help and guidance when using Vilje.

I would like to thank my friends and fellow students for five and a half amazing years. I could not have done this without you. Finally, I would like to thank my family for their support, and an extra thanks to my brothers, Roald and Erik, for help and guidance whenever I needed it during the last six months.

Kristin Bræck Leer
Trondheim, December 2016

Table of Contents

List of Tables

Listings

List of Figures

Abbreviations

AFM	=	Atomic Force Microscopy
c1	=	First Real Damper Constant
c2	=	First Real Damper Constant
CPU	=	Central Processing Unit
FSAL	=	First Same As Last
GUI	=	Graphical User Interface
HPC	=	High-Performance Computing
k1	=	First Real Spring Constant
k2	=	First Real Spring Constant
ODE	=	Ordinary Differential Equation
PECE	=	Predict-Evaluate-Correct-Evaluate
RoC	=	Rate of Convergence
RT	=	Relative Tolerance
s	=	Seconds
SPM	=	Scanning Probe Microscopy
STM	=	Scanning Tunneling Microscopy

Chapter 1

Introduction

In today's society, there are many systems that are controlled and monitored autonomously. These systems can be explained with mathematical models, which means that the behavior can be tested and any discovered errors can be removed. These mathematical models can be used to find an appropriate parameter estimator that can identify unknown parameters, before the system is simulated to check its behavior. One of these systems is the Atomic Force Microscopy (AFM).

The AFM is a well known microscopy within the scientific world, and it is used in the field of biology. It is known for its ability to identify mechanical properties in biological cells, and its precision in imaging of cells at micro- and nanoscale. The ability to identify mechanical properties (e.g. stiffness in the cell) has proven to be an important in biology and the discoveries of diseases (Bao and Suresh, 2003; Suresh, 2007). The stiffness in the cell changes when it goes from being a healthy cell to a diseased cell (Haase and Pelling, 2015; Kuznetsova et al., 2007). To study cells, both healthy and diseased, with an mathematical model of the AFM would be interesting. It could give new information about the mechanical properties of the cells, which can be helpful when studying medical questions (Sokolov, 2007). However, this means that a mathematical model is needed.

In (Ragazzon et al., 2016), a mathematical model of a biological cell and an AFM is given. This model estimates the parameters over time, which allows mechanical changes in the cell. This model is a product of earlier work (Ragazzon et al., 2015; Ragazzon and Gravdahl, 2016). The work published in (Leer, 2016), presented another on-line parameter estimator. Although, the results presented could not show that the estimated parameters converged exponentially fast to the real values. Both methods (Ragazzon et al., 2016; Leer, 2016), presents a study of the whole system (biological cell and the AFM). However, there are other interesting aspect to be studied in the AFM. The cantilever dynamics (Cappella and Dietler, 1999) in the AFM have often been studied using theoretical estimations, or representations that is only usable for a specific sample (Hutter and Bechhoefer, 1993; Sader et al., 1995, 1999). However, to our knowledge, an adaptive parameter estimation for this dynamic is not known.

The system proposed here, will use parallel computing (Barney et al., 2010) on Vilje (NOTUR, 2016), due to storage problems and slow simulation time in (Leer, 2016). In addition to that, analysing tools (e.g. bias, relative tolerance and rate of convergence (Kreyszig, 2010; Walpole et al., 1993)) will be used to determine the performance on how the estimated parameters converge to the real values. Later, the cantilever dynamics will be studied to see if an adaptive parameter estimation method can be used estimate the unknown parameters.

1.1 Problem Definition

- Continuing the work from (Leer, 2016). Find a gain matrix for the gradient method which provides a good estimate.
- Learning to use Vilje and run a single simulation, before learning to run multiple simulations in parallel.
- Investigate the cantilever dynamics and discovering if it is possible to use an adaptive parameter estimator to estimate the damper and spring constant in the cantilever.

1.2 Outline

The rest of this thesis is structured as follow. Chapter 2 contains basic theory that is relevant on topics in this thesis. Chapter 3 contains the AFM and its working principles, while theory on parameter estimation and the mathematical model from (Ragazzon et al., 2016) is found in chapter 4. Chapter 5 presents parallel computing, Vilje and analysing tool used with parallel computing. In chapter 6, the cantilever dynamics are presented and derived. The simulation results are presented in chapter 7. The discussion and the conclusion are given in chapter 8 and chapter 9, respectively.

Chapter 2

Basic Theory

This chapter is a continuation of the work published in (Leer, 2016). Several sections have been modified and added.

2.1 Biological Cells

Cells are fundamental for all living systems, and all cells consist of organelles. These cells are the simplest matter that can be alive. There are two different kinds of organisms that exist; single-celled and multicellular. Plants and animals are multicellular. Multicellular organisms consist of different types of specialised cells that cannot survive long on their own while single-celled organisms only consist of one cell. Hence, the multicellular organisms are therefore more complex organelles. Even when there are higher organizations, like tissues and organs, cells still has the basic structure and function (Campbell et al., 2015; Leer, 2016).

2.1.1 Cell Types

There are two different types of cells, prokaryotic and eukaryotic cells.

prokaryotic	=	”before nucleus”
eukaryotic	=	”true nucleus”

These cell types produce different kinds of living organisms.

- Prokaryotic cells:
  - Bacteria
  - Archaea
- Eukaryotic cells:

  - Protists
  - Fungi
  - Animals
  - Plants

All cells consist of organelles (can be looked at as organs in the human body), which is small parts inside the cell that perform different tasks. Both prokaryotic and eukaryotic cells share some basic features. First, they are bounded by a selective barrier known as the *plasma membrane*. Second, inside all cells there is a semicellular, jelly-like substance, where subcellular components are suspended. This is the *cytosol*. Third, all cells carry genes in form of DNA in the *chromosomes*. Last, there are tiny complexes that produce proteins according to instructions from genes. These are called *ribosomes*. The major difference between prokaryotic and eukaryotic cells is where the DNA is located. In prokaryotic cells, the DNA is concentrated in the organelle called nucleoid. This is a region of the cell that is not membrane-enclosed. Whereas in eukaryotic cells most of the DNA is located in the organelle called *nucleus*. This organelle is bounded by a double membrane (Campbell et al., 2015; Leer, 2016).

2.1.2 Eukaryotic Cells

The main focus in this thesis will be on eukaryotic cells, since one will most likely study plant or animal (human) cells (Sokolov, 2007). Therefore it will be more important to look at how an eukaryotic cell is constructed. In a eukaryotic cell there are many organelles, which all have different jobs (Campbell et al., 2015; Leer, 2016).

Organelles

From (Campbell et al., 2015) the organelles in an animal cell are as follows:

- **Nucleus**
  - Nuclear envelope: A double membrane enclosing the nucleus which is perforated by pores, and is continuous with endoplasmic reticulum (ER)
  - Nucleolus: A non membranous structure that is involved in producing ribosomes, and a nucleus has one or more nucleoli
  - Chromatin: The material consisting of DNA and proteins, and is visible in a dividing cell as individual condensed chromosomes
- **Plasma membrane**
  The membrane enclosing the cell
- **Endoplasmic Reticulum (ER)**
  A network of sacs and tubes; active in membrane synthesis and other synthetic and metabolic processes; have both rough (ribosome-studded) and smooth regions

- **Cytoskeleton**
  Reinforces the cell's shape and the functions in cell movement. The components are made of proteins, which includes microfilaments, intermediate filaments and microtubules

- **Ribosome**
  The complexes that makes proteins; free in cytosol or bounded to rough endoplasmic reticulum or nuclear envelope

- **Lysosome**
  Digestive organelle where macromolecules are hydrolyzed

- **Mitochondrion**
  Organelle where cellular respiration occurs and most ATP is generated

- **Golgi apparatus**
  Organelle active in synthesis, modification, sorting, and secretion in cell products

- **Flagellum**
  Motility structure present in some animal cells, composed of a cluster of microtubules within an extension of the plasma membrane

- **Centrosome**
  Region where the cell's microtubules are initiated. Contains a pair of centrioles

- **Microvilli**
  Projections that increase the cell's surface area

- **Peroxisome**
  Organelle with various specialized metabolic functions. Produces hydrogen peroxide as a by-product, then converts it to water

Studying Cells

Studying cells can take place in three different states; *in vitro*, *in vivo* and *in silico*. These three ways are different and the resulting information varies. *In vitro* means that the study of microorganisms, cells or biological molecules is outside the normal habitat. The study is fast, inexpensive, but one can only look at the organism. However, it is often not translatable to real life. *In vivo* happens within the biological context, e.g. animal testing or clinical trial. After *in vitro* one often wants to observe the effects on living subjects and one will therefore preform *in vivo*. *In silico* is biological computer simulations (Iversen, 2015; Leer, 2016).

2.1.3 New Way to Study Cells

There have been two ways to study the cell's mechanical properties:

1. The mechanical properties were integrally studied when the cell was considered as a whole

2. Using isolated lipid bilayers, biomembrane and cytosolic proteins, the mechanical properties of the cell structural components were studied

(Kuznetsova et al., 2007).

AFM (atomic force microscope) makes it possible to study both dynamical and mechanical properties of cells. These including special cell events like locomotion, differentiation and aging, physiological activation and electromotility and cell pathology (Kuznetsova et al., 2007). This can be an important instrument for the future, since it is possible to recognise changes in the cell structure, which can be a sign for a disease (Sokolov, 2007; Kuznetsova et al., 2007; Leer, 2016). AFM is explained in more detail in Chapter 3.

2.2 Simulation Methods

When simulating a system, there are a diversity of simulation solvers with different properties to choose from. These properties either gives a guideline for choosing the right simulation solver for a specific problem, or they can help determining if e.g. the system is nonstiff or stiff.

The simulation methods studied in this thesis are the explicit and implicit Runge-Kutta, Runge-Kutta pair of Bogacki and Shampine, modified Rosenbrock, trapezoidal rule and Adams-Bashforth-Moulton are used as simulation solvers in Matlab. They are similar to; ode45, ode15s, ode23, ode23s, ode23t and ode113 (Grenoble, 1999; Leer, 2016).

2.2.1 Basic Terms In the Simulation Solver

Before we look into simulation solvers and their corresponding mathematical methods, some terms must be explained in advanced.

Relative tolerance in the simulation solver is described as the error measured relative to the size of each state. This is given as a percentage of the of the state value, and the default value is $1e-03$ (MatlabWorks, 2016a).

Interpolation is when there are found approximating values to points given by a function $f(x)$. The approximated values lie between the points of $f(x)$ (Kreyszig, 2010).

Local extrapolation is used when there is a desire to specify the accuracy in a local error. This is done by computing the numerical solution of two method (e.g. explicit Runge-Kutta) with different order, y_{n+1} (order p) and $\widehat{y}_{n+1}$ (order $p+1$). Starting the computation with y_n=$\widehat{y}_n$ for t_{n+1}, a local solution and the local error is found. This error is an estimate of the local error of y_{n+1}. Therefore, the next time step should be y_{n+1}. However, $\widehat{y}_{n+1}$ is more accurate, and will therefore be used as the next time step instead of y_{n+1}. When $\widehat{y}_{n+1}$ is chosen over y_{n+1} it is called local extrapolation (Egeland and Gravdahl, 2002; Shampine, 1973).

2.2.2 Nonstiff and Stiff Ordinary Differential Equation Problems

In Matlab, different methods for solving ordinary differential equations (ODE) problems is presented. Some of the most common ones are mention in the introduction to section 2.2. There are two types of problems: Nonstiff and stiff problems. The nonstiff problems are often solved with explicit methods, while stiff problems are solved with implicit methods.

Stiff ODE Problems

Stiff systems are systems with a large spread in eigenvalues of the Jacobian. When using an explicit method the time-step must be selected to ensure stability. To compute the dynamics corresponding to the small eigenvalues, a large amount of time steps are required. This will in turn have a negative impact on both simulation time and accuracy. In order to increase accuracy and decrease simulation time when dealing with stiff systems, other methods should be used, for instance the implicit Runge-Kutta (Egeland and Gravdahl, 2002; Ashino et al., 2000). Solvers that are made for stiff ODE problems are $ode15s$, $ode23s$ and $ode23t$ (Leer, 2016).

Nonstiff ODE Problems

Nonstiff ODE problems are problems when the components evolves simultaneously on the same time scale and that there is a small spread in eigenvalues of the Jacobian. Explicit methods are used for solving nonstiff ODE problems (Egeland and Gravdahl, 2002; MatlabWorks, 2016a). There are three solvers that are made for nonstiff ODE problems: $ode45$, $ode23$ and $ode113$. However, $ode15s$ and $ode23s$ can also be used for solving nonstiff ODE problems (Ashino et al., 2000).

2.2.3 Explicit Runge-Kutta

The explicit Runge-Kutta method is an extended version of the Euler's method. The Euler method is of order $p = 1$, and the modified Euler method is of order $p = 2$. These methods compute y_{n+1} as a linear combination of $f(y_n, t_n)$ and the approximation $f[y(t_n + ch), t_n + ch]$, $0 < c \leq 1$ (Egeland and Gravdahl, 2002; Leer, 2016).

The explicit Runge-Kutta method is extended to higher order ($p > 2$) from Euler and the modified Euler method. By extending to a higher order there are more approximations of f over the interval that needs to be computed. Then by using a linear combination of these approximations y_{n+1} can be computed (Egeland and Gravdahl, 2002; Leer, 2016).

Numerical Scheme

The explicit Runge-Kutta with σ stages for the system

$$\dot{y} = f(y, t) \tag{2.1}$$

is given by

$$k_i = f(y_n + h\sum_{j=1}^{i-1} a_{ij}k_j, t_n + c_i h), \quad i = 1, \dots, \sigma \tag{2.2}$$

$$y_{n+1} = y_n + h\sum_{j=1}^{\sigma} b_j k_j \tag{2.3}$$

Stability Function

The stability function is

$$R_E(h\lambda) = \det\left[\mathbf{I} - \lambda h\left(\mathbf{A} - \mathbf{1}\mathbf{b}^T\right)\right] \tag{2.4}$$

and is derived in (Egeland and Gravdahl, 2002). The stability function shows that

1. $|R_E(h\lambda)|$ will go towards infinity as $|\lambda|$ goes to infinity
2. $R_E(h\lambda)$ is a polynomial in $h\lambda$ with order $\leq \sigma$

(Leer, 2016).

Parameters

The parameters a_{ij}, b_i and c_i can be represented in a butcher array:

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_\sigma & a_{\sigma 1} & a_{\sigma 2} & \dots & a_{\sigma,\sigma-1} & \\
\hline
 & b_1 & b_2 & \dots & b_{\sigma-1} & b_\sigma
\end{array}$$

Table 2.1: Representation of the parameters a_{ij}, b_i and c_i in the explicit Runge-Kutta method in a Butcher array.

The representation in table 2.1 can also be represented as a matrix, **A**, and as vectors, **b** and **c**.

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ a_{21} & 0 & \dots & 0 & 0 \\ a_{31} & a_{32} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{\sigma 1} & a_{\sigma 2} & \dots & a_{\sigma,\sigma-1} & 0 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_\sigma \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} 0 \\ c_2 \\ c_3 \\ \vdots \\ c_\sigma \end{pmatrix} \tag{2.5}$$

A only have nonzero elements below the diagonal, and it follows that

$$\det(\mathbf{I} - \lambda h \mathbf{A}) = 1 \tag{2.6}$$

(Egeland and Gravdahl, 2002; Leer, 2016).

ode45

The $ode45$ simulation solver is based on the explicit Runge-Kutta method, Dormand-Prince (4,5) pair method. $ode45$ is used as the auto setting when simulating, because it performs well with many ODE problems (MatlabWorks, 2016b). This method solves nonstiff ODE problems.

ode23

The Bogacki-Shampoine method is also known as $ode23$, is a variable-step explicit Runge-Kutta method. In this method, y_{n+1} is computed with a third order method, and by comparing the result with an embedded second order method, the error estimate is found. Local extrapolation is used in this method (Egeland and Gravdahl, 2002). The butcher array is given in Table 2.2.

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
y	$\frac{21}{72}$	$\frac{1}{4}$	$\frac{3}{9}$	$\frac{1}{8}$
$\widehat{y}$	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
Δe	$-\frac{5}{72}$	$\frac{1}{12}$	$\frac{1}{9}$	$-\frac{1}{8}$

Table 2.2: Representation of the Bogacki-Shampine method in a Butcher array.

2.2.4 Implicit Runge-Kutta

Numerical Scheme

The implicit Runge-Kutta with σ stages for the system

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t) \tag{2.7}$$

is given by

$$\mathbf{k}_i = \mathbf{f}(\mathbf{y}_n + h(a_{11}\mathbf{k}_1 + \ldots + a_{1\sigma}\mathbf{k}_\sigma), t_n + c_1 h) \tag{2.8}$$

$$\vdots$$

$$\mathbf{k}_\sigma = \mathbf{f}(\mathbf{y}_n + h(a_{\sigma 1}\mathbf{k}_1 + \ldots + a_{\sigma\sigma}\mathbf{k}_\sigma), t_n + c_\sigma h) \tag{2.9}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h(b_1\mathbf{k}_1 + \ldots + b_\sigma\mathbf{k}_\sigma) \tag{2.10}$$

(Egeland and Gravdahl, 2002).

Stability Function

There are two alternative stability expressions for the implicit Runge-Kutta method

$$R(h\lambda) = \left[1 + \lambda h \mathbf{b}^T \left(\mathbf{I} - h\lambda\mathbf{A}\right)^{-1} \mathbf{1}\right] \tag{2.11}$$

$$R(h\lambda) = \frac{\det\left[\mathbf{I} - \lambda h\left(\mathbf{A} - \mathbf{1}\mathbf{b}^T\right)\right]}{\det\left(\mathbf{I} - \lambda h\mathbf{A}\right)} \tag{2.12}$$

(Egeland and Gravdahl, 2002; Leer, 2016).

ode15s

The $ode15s$ is based on the implicit Runge-Kutta method, and is a stiff ODE solver, which can also be used to solve nonstiff ODE problems. If nonstiff solvers like $ode45$, $ode23$ or $ode113$ are slow in terms of compute time, this is usually the first stiff solver to try (Shampine and Reichelt, 1997; MatlabWorks, 2016b).

2.2.5 Rosenbrock Methods

Numerical Scheme

A Rosenbrock method with σ stages for the system

$$\dot{y} = f(y, t) \tag{2.13}$$

is given by

$$\mathbf{k}_i = \mathbf{f}(\mathbf{y}_n + h\sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j, t_n + c_i h) + h\mathbf{J}\sum_{j=1}^{i} \rho_{ij}\mathbf{k}_j + \rho_i h\dot{\mathbf{f}}(\mathbf{y}_n, t_n) \tag{2.14}$$

$$i = 1, \ldots, \sigma$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{j=1}^{\sigma} b_j\mathbf{k}_j \tag{2.15}$$

where **J** is the Jacobian ($\mathbf{J} = \partial f(y_n, t_n)/\partial y$), the interpolation constants satisfy $c_i \sum_{j=1}^{i-1} a_{ij}$, and

$$\rho_i = \sum_{j=1}^{i} \rho_{ij} \tag{2.16}$$

(Egeland and Gravdahl, 2002). The Jacobian, **J**, and the interpolation constant are the same for the Rosenbrock method and the Runge-Kutta method (Leer, 2016).

Definition of the Rosenbrock method

First term on the right side of the stage computations (equation 2.14) has the same form as the stage in an explicit Runge-Kutta (equation 2.2). To make the method implicit a linearized term is added to the stage. In an implicit Runge-Kutta method each time step is required a Newton search to compute the stages. The stage computations in a Rosenbrock method can be performed without iterations according to the formula

$$\mathbf{V}_i\mathbf{k}_i = \mathbf{f}(\mathbf{y}_n + h\sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j, t_n + c_i h) + h\mathbf{J}\sum_{j=1}^{i-1} \rho_{ij}\mathbf{k}_j + \rho_j h\dot{\mathbf{f}}(\mathbf{y}_n, t_n) \tag{2.17}$$

where

$$\mathbf{V}_i = \mathbf{I} - h\rho_{ij}\mathbf{J} \tag{2.18}$$

is a nonsingular matrix for a sufficiently small time step h (Egeland and Gravdahl, 2002; Leer, 2016).

2.2.6 Modified Second Order Rosenbrock Method

A second order modified Rosenbrock method is given by

$$\begin{aligned} \mathbf{V}\mathbf{k}_1 &= \mathbf{f}(\mathbf{y}_n, t_n) + h\rho\dot{\mathbf{f}}(\mathbf{y}_n, t_n) \\ \mathbf{V}\mathbf{k}_2 &= \mathbf{f}(\mathbf{y}_n, t_n) + \mathbf{f}(\mathbf{y}_n + \frac{h}{2}\mathbf{k}_1, t_n + \frac{h}{2}) - \mathbf{k}_1 + h\rho\dot{\mathbf{f}}(\mathbf{y}_n, t_n) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + k\mathbf{k}_2 \\ \mathbf{V} &= \mathbf{I} - h\rho\mathbf{J}, \quad \rho = \frac{1}{2+\sqrt{2}} \end{aligned} \tag{2.19}$$

with step size control using a FSAL (First Same As Last) computation

$$\begin{aligned} \mathbf{V}\mathbf{k}_3 &= 2\mathbf{f}(\mathbf{y}_n, t_n) + (6+\sqrt{2})\mathbf{f}(\mathbf{y}_n + \frac{h}{2}\mathbf{k}_1, t_n + \frac{h}{2}) + \mathbf{f}(\mathbf{y}_{n+1}, t_{n+1}) \\ &\quad - 2\mathbf{k}_1 - (6+\sqrt{2})\mathbf{k}_2 + h\rho\dot{\mathbf{f}}(\mathbf{y}_n, t_n) \\ error &= \frac{h}{6}(\mathbf{k}_1 - 2\mathbf{k}_2 + \mathbf{k}_3). \end{aligned} \tag{2.20}$$

The modified Rosenbrock method is quite similar to the Rosenbrock method, except for the computation of the second stage has a term of type $-\mathbf{k}_1$ instead of $h\mathbf{J}\rho_{ij}\mathbf{k}_1$ (Egeland and Gravdahl, 2002; Leer, 2016).

ode23s

The $ode23s$ solver is based on the modified Rosenbrock method. Some stiff problems can be inefficient or difficult to solve using $ode15s$. The next solver one tries is often $ode23s$ (Shampine and Reichelt, 1997; MatlabWorks, 2016b).

2.2.7 Trapezoidal Rule

Numerical Scheme

Have the implicit Runge-Kutta method

$$\mathbf{k_1} = \mathbf{f}(\mathbf{y}_n, t_n) \tag{2.21}$$

$$\mathbf{k_2} = \mathbf{f}\left[\mathbf{y}_n + \frac{h}{2}(\mathbf{k_1} + \mathbf{k_2}), t_n + h\right] \tag{2.22}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}(\mathbf{k_1} + \mathbf{k_2}) \tag{2.23}$$

Equation 2.23 revealed that

$$\mathbf{k_2} = \mathbf{f}(\mathbf{y}_{n+1}, t_{n+1}) \tag{2.24}$$

This means that the last row in **A** is equal to $\mathbf{b}^T$. Then, $\mathbf{y}_n$+1 can be rewritten and becomes what is known as the *trapezoidal rule*

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}\left[\mathbf{f}(\mathbf{y}_n, t_n) + \mathbf{f}(\mathbf{y}_{n+1}, t_{n+1})\right] \tag{2.25}$$

(Egeland and Gravdahl, 2002).

Stability Function

From equation 2.25 and a test function

$$y_{n+1} = y_n + \frac{h\lambda}{2}(y_n + y_{n+1}) \tag{2.26}$$

which gives the stability function

$$R(\lambda h) = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} \tag{2.27}$$

(Egeland and Gravdahl, 2002; Leer, 2016).

ode23t

The $ode23t$ is based on the trapezoidal rule, and it is a stiff ODE problem solver (Shampine and Reichelt, 1997).

2.2.8 Adams-Bashforth-Moulton Method

The $ode113$ solver is a PECE (predict-evaluate-correct-evaluate) implementation of Adams-Bashforth-Moulton method. This method originally comes from combining two methods, Adams-Bashforth and Adams-Moulton method (Shampine and Gordon, 1975).

Adams-Bashforth Method

The method is also known as the explicit Adams method, and has the equation

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + \int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{y}(t), t)dt \tag{2.28}$$

as a starting point. The algorithm for the explicit Adams method becomes

$$\begin{aligned}
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{f}_n \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\left(\frac{3}{2}\mathbf{f}_n - \frac{1}{2}\mathbf{f}_{n-1}\right) \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\left(\frac{23}{12}\mathbf{f}_n - \frac{4}{3}\mathbf{f}_{n-1} + \frac{5}{12}\mathbf{f}_{n-2}\right) \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\left(\frac{55}{24}\mathbf{f}_n - \frac{59}{24}\mathbf{f}_{n-1} + \frac{37}{24}\mathbf{f}_{n-2} - \frac{9}{24}\mathbf{f}_{n-3}\right)
\end{aligned} \tag{2.29}$$

by using the expression for the backwards difference operator (Egeland and Gravdahl, 2002; Kreyszig, 2010).

Adams-Moulton Method

The Adams-Moulton method is known as the implicit Adams method. The numerical algorithm

$$\begin{aligned}
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{f}_{n+1} \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\left(\frac{1}{2}\mathbf{f}_{n+1} + \frac{1}{2}\mathbf{f}_n\right) \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\left(\frac{5}{12}\mathbf{f}_{n+1} + \frac{8}{12}\mathbf{f}_{n-1} - \frac{1}{12}\mathbf{f}_{n-1}\right) \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\left(\frac{9}{24}\mathbf{f}_{n+1} + \frac{19}{24}\mathbf{f}_n - \frac{5}{24}\mathbf{f}_{n-1} + \frac{1}{24}\mathbf{f}_{n-2}\right)
\end{aligned} \tag{2.30}$$

(Egeland and Gravdahl, 2002; Kreyszig, 2010).

Adams-Bashforth-Moulton Method

The Adams-Bashforth-Moulton PECE formulas of order k will refer to the Adams-Bashforth predictor of order k and the Adams-Moulton corrector of order $k + 1$ (Shampine and

Gordon, 1975). Here, the implicit Adams method is based on computing a predictor, $\widehat{\mathbf{y}}_{n+1}$, with the explicit Adams method. Using $\widehat{\mathbf{f}}_{n+1} := \mathbf{f}(t_{n+1}, \widehat{\mathbf{y}}_{n+1})$, gives the PECE method, also known as the Adams-Bashforth-Moulton method (Egeland and Gravdahl, 2002; Shampine and Gordon, 1975; Kreyszig, 2010).

ode113

The $ode113$ solver is based on the Adams-Bashforth-Moulton method, and is a nonstiff ODE problem solver (Shampine and Reichelt, 1997; Shampine and Gordon, 1975).

2.3 Mathematical Analysing Tools

When performing different experiments with simulation, it is important to determine the performance of each simulation. This can be done by using simple mathematical expressions to find bias, relative tolerance and rate of convergence for the estimated values. These expressions will be described mathematically in this section. Before this, some terms must be explained in advanced. In (Walpole et al., 1993):

> ”A **population** consists of the totality of the observations with which we are concerned.”
> ”A **sample** is a subset of a population.”

2.3.1 Mean

The numerical average, also known as mean, gives an average over the observations in a sample. The sample mean is given by

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n} = \frac{x_1 + x_2 + \cdots + x_n}{n} \tag{2.31}$$

(Walpole et al., 1993).

2.3.2 Bias

A bias is when a procedure produces inferences that overestimates or underestimates from the exact or value (Walpole et al., 1993). When estimating one value, the bias becomes

$$bias = x_0 - x \tag{2.32}$$

where x is the estimated value and x_0 is the wanted result. Having a sample of estimated values one can either compare each value in the sample with the real value

$$bias_i = \sum_{i=1}^{n} x_0 - x_i \quad i = 1, \ldots, n \tag{2.33}$$

or take the mean over the sample and compare it with the real value

$$bias = x_0 - \bar{x} = \sum_{i=1}^{n} \frac{x_i}{n} \tag{2.34}$$

where $bias_i$ is the bias for each value x_i in the sample for $i = 1, \dots, n$.
(Kreyszig, 2010).

2.3.3 Relative Tolerance

In (Kreyszig, 2010), the relative error is defined as

$$\epsilon_r = \frac{\epsilon}{a} = \frac{a - \widetilde{a}}{a} = \frac{\text{Error}}{\text{True value}} \tag{2.35}$$

where a is the true value, $\widetilde{a}$ is the estimated value, ϵ is the error and ϵ_r is the relative error. The definition of relative tolerance is given by equation 2.35 multiplied with 100%

$$\epsilon_{rt} = \epsilon_r \cdot 100\% = \frac{\epsilon}{a} \cdot 100\% \tag{2.36}$$

2.3.4 Rate of Convergence

A sample can either converge or diverge. From (Kreyszig, 2010), the terms are defined as:

Convergence: A convergent sequence z_1,z_2,$\cdots$ is one that has a limit c, written

$$\lim_{n\to\infty} z_n = c \quad \text{or simply} \quad z_n \to c \tag{2.37}$$

By definition of limit this means that for every $\epsilon \geq 0$ we can find an N such that

$$|z_n - c| \leq \epsilon \quad \text{for all } n \geq N; \tag{2.38}$$

geometrically, all terms z_n with $n \geq N$ lie in an open disk of radius ϵ and center c (complex sequence). For a real sequence, equation 2.38 gives an open interval of length 2ϵ and real midpoint c on the real line.

Divergence: A divergent sequence is one that does not converge.

The rate of convergence is described as the time it takes for a sequence to converge.

2.4 Transfer Functions

In the field of cybernetics, it is often needed to describe physical system using mathematical models. These system can be effected by the surroundings. If this effect is wanted or set, it is said to be the input, u. If not, it is described as noise, v. In (Balchen et al., 2003), a transfer function is described as:

A *transfer function* describes the context between one input and one output.

A transfer function is often given as

$$h(s) = \frac{p_p s^p + \cdots + p_1 s + p_0}{s^n + \alpha_{n-1} + \cdots + \alpha_1 s + \alpha_0} = \frac{p_p(s - v_1)(s - v_2)\cdots(s - v_p)}{(s - \lambda_1)(s - \lambda_2)\cdots(s - \lambda_n)} \tag{2.39}$$

$$h(s) = \frac{\alpha(s)}{\beta(s)} \tag{2.40}$$

where α, β are polynomials of s. $\alpha(s)$ is known as the characteristic polynomial. The roots of the characteristic polynomial is known as the poles, while the roots of β are known as zeros. In equation 2.39, λ_i are the poles. By analysing the placement of zeros and poles, the properties of the system is revealed. When the parameters in the system changes, the pole placement will change (Balchen et al., 2003). The transfer fuction of second order is given by

$$h(s) = \frac{K}{(\frac{s}{\omega_0})^2 + 2\xi\frac{s}{\omega_0} + 1} \tag{2.41}$$

where K is the steady-state gain, ξ is the damping coefficient and the ω_0 is the natural frequency (Balchen et al., 2003).

Chapter 3

Atomic Force Microscopy

This chapter is a continuation of the work published in (Leer, 2016). Several sections have been modified and subsection 3.2 was added.

The atomic force microscopy (AFM) is an imaging tool which can be used as a mechanism for studying micro- and nanostructures for both living cells and cell organelles (Sokolov, 2007). This instrument allows us to study the mechanical properties of cells (Bao and Suresh, 2003), and at the same time give the topography with high resolution and force sensitivity (Kuznetsova et al., 2007). AFM originates from scanning probe microscopy (SPM), and was invented and introduced in the 1980's (Binnig et al., 1986). The first type of SPM was the scanning tunneling microscopy (STM). The STM had some drawback, e.g some scans required ultra-high resolution, and that the samples needed to be conductive, which made the STM limited to investigate only metal and semiconductors. The AFM was made without these drawbacks (Leer, 2016).

For engineers this instrument is especially interesting, since the surface image is entirely dependent on the use of a feedback loop. Most AFMs use piezo-electric actuators, optical detection of cantilever deflection, and PI or PID control (Abramovitch et al., 2007). Another interesting aspect with AFM is that it gives a 3D image of the surface by using generated false colors for detecting height differences (Abramovitch et al., 2007; Leer, 2016).

Chapter 3 takes on the working principles of AFM and modes of operation. Section 3.2 goes more in depth of the cantilever on the AFM. Later in this thesis the dynamic in the cantilever will be looked at. Section 3.3 looks at the problems/disadvantages of AFM, and section 3.4 explains the biological advantages of using AFM (Leer, 2016).

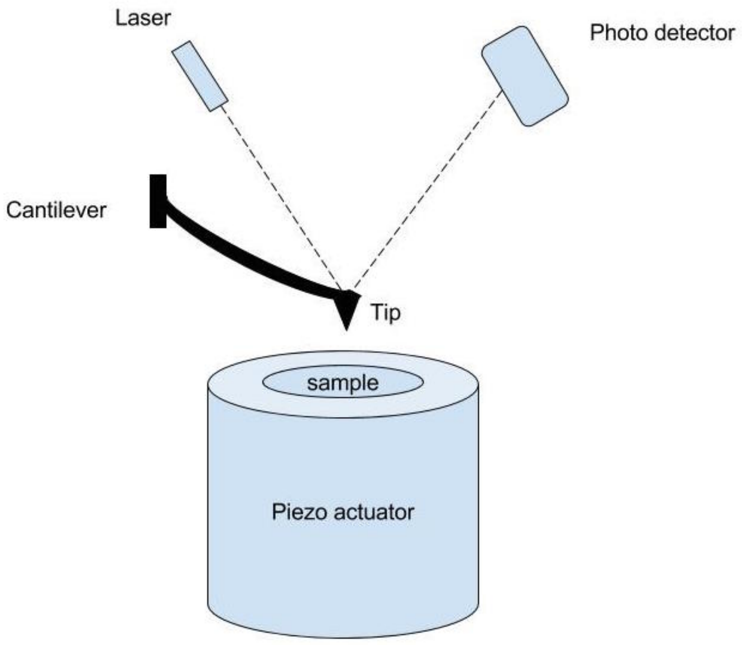

Figure 3.1: A simple AFM setup with the optical lever method.

3.1 Operations

The AFM consists of components seen in figure 3.1. The probe connected to the cantilever is scanned across the sample surface and gives an interaction force between the tip and the sample. The deflection in the cantilever can be measured and monitored by different types of methods, but the most popular and simplest to implement is the optical lever method (figure 3.1) (Cappella and Dietler, 1999). In the setup shown in figure 3.1, a laser is attached above the cantilever to make the light reflect, and is then detected by a photo detector (Sokolov, 2007).

A typical approach for moving the tip across the sample is to use a piezo actuator. This makes the sample move in x-, y- and z-direction. Alternatively, the sample is moved in the xy-plane by maneuvering an area beneath the sample. The z-axis is controlled by moving the cantilever up and down (oscillation). (Abramovitch et al., 2007; Iversen, 2015; Leer, 2016).

3.1.1 Modes of Operation

There are three main imaging modes; contact, tapping and non-contact mode. These can be divided into static and dynamic mode (Abramovitch et al., 2007). In the three main imaging modes, there is a feedback loop designed to hold a constant z-actuator. The feedback loop gives the z-axis position and a topographic map can be created (Abramovitch et al., 2007).

Contact Mode (static)

The tip is ”dragged” across the sample surface, where the cantilever deflection is kept constant by using a feedback loop. Therefore the force between the sample and probe is constant, and from this one can obtain an image of the surface. The downside of this mode is that friction induced by the feedback loop can scratch the sample, thus destroying it. The upside is that it is simple and allows fast scanning (Wilson and Bullen, n.d; Leer, 2016).

Tapping Mode (dynamic)

The cantilever oscillates above the sample and taps it in very small periods. The oscillation frequency is equal to, or somewhere near the resonance of the cantilever. When the tip touches the sample, the oscillation amplitude changes depending on the tip-sample distance (Iversen, 2015). To avoid this, a feedback loop is used to keep the amplitude constant, which keeps a constant tip-sample interaction. With that information it is possible to make a surface image. Since the tapping mode depends on slower amplitude estimates, it is slower than contact mode. On the upside, it is preferable for soft biological samples as it has a low friction rate, thereby decreasing the chances of destroying the sample due to friction (Wilson and Bullen, n.d; Leer, 2016).

Non-Contact Mode (dynamic)

The probe oscillates above the sample surface, but do not come in contact with the sample. The topography can be measured by using a feedback loop to monitor changes in the amplitude due to attractive forces (Abramovitch et al., 2007). Disadvantages of this mode is that it generally provides lower resolution than tapping mode, and it gives best results in vacuum. This mode is the best of the three when it comes to avoiding damage on the probe and sample (Wilson and Bullen, n.d; Leer, 2016).

3.2 Cantilever Dynamics

The cantilever is often made out of silicon or silicon nitride with a microfabricated tip. The shape of the tip is often rectangular or ”V”-shaped with a metallic thin layer on the back. The layer is often gold and improve the reflectivity, especially in liquid where the silicon nitride reflection is reduced (Cappella and Dietler, 1999).

As mention in section 3.1, the most common method to detect cantilever deflection is the optical lever method. Two other common deflection methods are the interferometric method and the electronic tunneling method (Cappella and Dietler, 1999).

The tip-sample force, F, is given by Hooke’s law:

$$F = -k_c \delta_c \tag{3.1}$$

where k_c is the cantilever spring constant and the δ_c is the cantilever deflection. The actual tip-sample distance, D, is related to the cantilever deflection, sample deflection and the distance between the sample surface and rest position of the cantilever, Z:

$$D = Z - (\delta_c + \delta_s) \tag{3.2}$$

The relationship in equation 3.2 is given in figure 3.2 (Cappella and Dietler, 1999). The

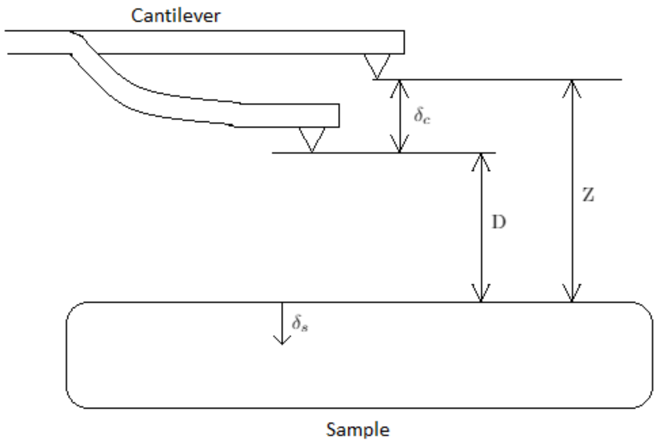

Figure 3.2: The tip sample distance given by equation 3.2.

main problem is to establish the cantilever spring constant, k_c. How the problem is solved depends on the shape and material of the cantilever (Hutter and Bechhoefer, 1993). However, for a rectangular shaped tip, k_c is equal to

$$k_c = \frac{Et_c^3 w}{4L^3} \tag{3.3}$$

and for a ”V”-shaped tip, k_c is equal to

$$k_c = \frac{Et_c^3 Wb}{2b(L_1^3 - L_2^3) + 6WL_2^3} \tag{3.4}$$

where t_c is the thickness and E is the Young modulus (Butt et al., 2005; Sader et al., 1999; Cappella and Dietler, 1999; Tortonese, 1997). The rest of the parameters in equation 3.3 and equation 3.4 are given in figure 15 in (Cappella and Dietler, 1999).

3.3 Problems with Atomic Force Microscopy Control

Some problems have been identified for the use and control AFM. Firstly, there is an issue when it comes to exchanging the cantilevers and tips, because the system then needs to be readjusted each time (Abramovitch et al., 2007). Secondly, it requires a lot of adjustment when there is a new measurement, sample or new cantilever/tip. Calibrations need to be done often, since the height measurements have to be estimated for each image. Thirdly, it can be a time consuming operation, ranging from under a minute and up to an hour (Abramovitch et al., 2007; Leer, 2016).

3.3.1 Improvement of AFM

To increase the performance, one or more of the loop components need improvement. One can either re-design one of the individual components, implement a better controller, or both (Abramovitch et al., 2007; Leer, 2016).

3.4 Biological Advantages

There is a huge advantage in using AFM when studying biological cells. The reason for this is that there is not so many restrictions when using AFM as to Scanning Probe Microscopy (SPM). When using AFM you are able to study the biological object directly in their natural conditions, *in vitro*, if not *in vivo* (see 2.1.2), in air, in water, buffers, and other ambient media. Also, there are as good as no sample preparations, except for the attachment of sample to surface, temperature of the solution/sample have literally nothing to say, as well as chemical composition or medium type (non-aqueous or aqueous liquid). The only limitation of importance is that the medium should be transparent for the laser light that is used for detection (Leer, 2016).

When scanning the sample it is possible to get resolution up to nanometer precision (molecular), and vertical resolution up to 0.01 nanometer, but this is decided by the AFMs detection sensitivity and the noise from the environment. It provides a topographic 3D image, and it is possible to measure different biophysical properties like elasticity, adhesion, hardness, friction, etc. (Sokolov, 2007; Leer, 2016).

Chapter 4

Parameter Estimation

This chapter is a continuation of the work published in (Leer, 2016). Several sections have been modified and added.

In (Leer, 2016), the system from (Ragazzon et al., 2016) was used, except that the parameter estimator was changed. (Ragazzon et al., 2016) used the least-square method with forgetting factor, while the gradient method with instantaneous cost was used in (Leer, 2016). Both are on-line parameter estimation methods, using adaptive laws with normalization. In on-line parameter estimation, the structure of the plant is known, but the parameters can be unknown and changing with time. Since the parameters are unknown or changing, it is important to use an estimation scheme that provide frequent estimates of the parameters (Ioannou and Sun, 2012).

The essential idea of on-line estimation is comparison of the observed system response $y(t)$ and the output of a parametrized model $\widehat{w}(\theta, t)$. $\widehat{w}(\theta, t)$ has the same structure as the plant model, $w(t)$. The parameter vector, $\theta(t)$, is adjusted continuously so that $\widehat{w}(\theta, t)$ approaches $w(t)$ as t increases. With some special input conditions, $\widehat{w}(\theta, t)$ will approach and be close to $w(t)$, which means that θ is close to the unknown parameter vector θ^* of the plant model (Ioannou and Sun, 2012).

Adaptive laws with normalization are developed because the plant does not have to be stable or for the plant input to be bounded a priori. The stability and boundedness of these plants are properties that have to be proven (Ioannou and Sun, 2012).

The on-line parameter estimation procedure has three steps:

1. Select an appropriate parametrization of the plant model.

2. Decide on an adaptive law for generating or updating $\theta(t)$. The adaptive law is often a differential equation whose state is $\theta(t)$ and is designed using stability considerations or simple optimization techniques to minimize the difference between $w(t)$

and $\widehat{w}(\theta, t)$ with respect to $\theta(t)$at each time t.

3. Design of the plant input so that the properties of the adaptive law imply that $\theta(t)$ approaches the unknown plant parameter vector θ^* as $t \longrightarrow \infty$.

This chapter is a continuation of the work published in (Leer, 2016). Several sections have been modified and section 4.2 I added.

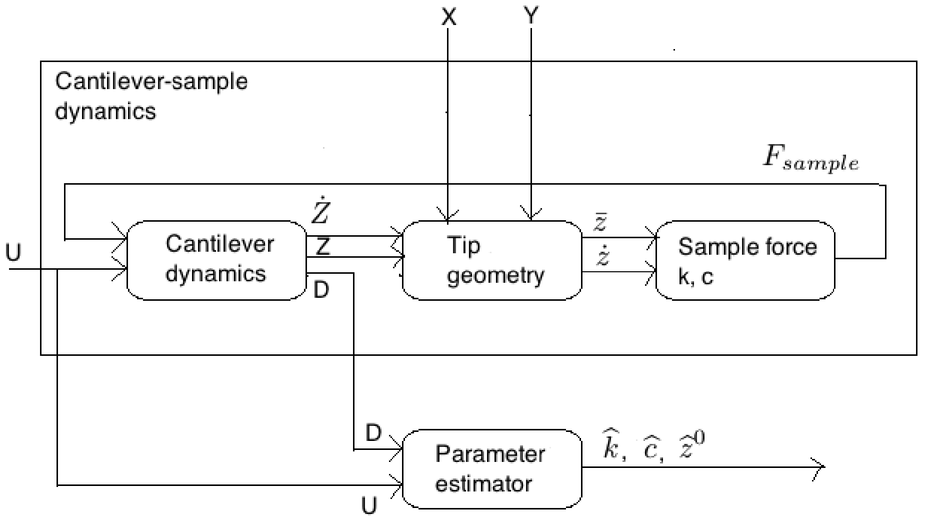

Figure 4.1: An overview of the system in a block diagram.

To be able to understand some of the properties in the methods described below, the term *persistence of excitation* (PE) will be described as in definition 4.3.1 in (Ioannou and Sun, 2012):

A piecewise continuous signal vector ϕ: $\mathcal{R}^+ \mapsto \mathcal{R}^n$ is PE in $\mathcal{R}^n$ with a level of excitation $\alpha_0 > 0$ if there exist constants α_0,T_0>0 such that

$$\alpha_1 I \geq \frac{1}{T_0} \int_t^{t+T_0} \phi(\tau)\phi^T(\tau)d\tau \geq \alpha_0 I, \quad \forall t \geq 0 \tag{4.1}$$

4.1 Gradient Method

The gradient method is an on-line parameter estimation using adaptive laws with normalization. The first adaptive laws using the gradient or steepest descent method were introduced in the early 1960s. In that period, there was a lack of stability and the scheme have been changed to increase stability properties (Ioannou and Sun, 2012; Leer, 2016).

4.1.1 Method

Within on-line parameter estimation there are adaptive laws with normalization, and for developing an adaptive law for estimating θ^* in the parametric model

$$w = W(s)\theta^{*T}\psi \tag{4.2}$$

both the gradient method and a cost function is used. The first thing one must do, is developing an algebraic estimation error equation. The error equation influence on the selection of cost function $J(\theta)$ which has to be convex over the space of $\theta(t)$, the estimate of θ^* at time t, for each time t. Then the gradient method is used to minimize $J(\theta)$ with respect to $\theta(t)$ for each time t (Ioannou and Sun, 2012; Leer, 2016).

Gradient Method

The gradient method is also known as the method of steepest descent and solves the unconstrained minimization problem

$$\begin{aligned} &\text{minimize } J(\theta) \\ &\text{subject to } \theta \in \Re^n. \end{aligned} \tag{4.3}$$

The process of this method is an initial approximation, θ_0, for the minimum θ^* to successive points θ_1, θ_2, ... in $\Re^n$ in an iterative manner until some stopping condition is satisfied (Ioannou and Sun, 2012). This is done by a linear search in direction, d_k, which is given by

$$d_k = -\nabla J(\theta_k) \tag{4.4}$$

where θ_k is the current point and d_k is the direction from θ_k in which the initial rate of decrease of $J(\theta)$ is greatest. The sequence of $\{\theta_k\}$ is defined by

$$\theta_{k+1} = \theta_k + \lambda_k d_k = \theta_k - \lambda_k \nabla J(\theta_k), \quad (k = 0, 1, 2, \dots) \tag{4.5}$$

where θ_0 is given and λ_k is known as the is step size or step length. Setting $\lambda_k = \lambda \forall k$ for a simpler expression, but still obtaining θ_{k+1} from equation 4.5

$$\theta_{k+1} = \theta_k - \lambda \nabla J(\theta_k). \tag{4.6}$$

The step length of λ is a compromise between accuracy and efficiency. Equation 4.6 can be converted into a continuous-time differential equation when assuming infinitesimally small step length,

$$\dot{\theta} = -\nabla J(\theta(t)), \quad \theta(t_0) = \theta_0. \tag{4.7}$$

The solution to equation 4.7, $\theta(t)$, is the descent path in the time domain starting at $t = t_0$.

By using a constant positive definite matrix $\Gamma = \Gamma^T$, the direction of the steepest descent $d = -\nabla J$ can be scaled. By using a $n \times n$ nonsingular matrix Γ_1, so that $\Gamma = \Gamma_1 \Gamma_1^T$, and a vector $\bar{\theta} \in \Re^n$ given by

$$\Gamma_1 \bar{\theta} = \theta. \tag{4.8}$$

The minimization problem from equation 4.3 becomes

$$\begin{aligned} &\text{minimize } \bar{J}(\bar{\theta}) \triangleq J(\Gamma_1 \bar{\theta}) \\ &\text{subject to } \bar{\theta} \in \Re^n \end{aligned} \tag{4.9}$$

If $\bar{\theta}^*$ is a minimum of $\bar{J}$, the vector $\theta^* = \Gamma_1 \bar{\theta}^*$ is a minimum of J (Ioannou and Sun, 2012; Leer, 2016). The steepest descent for the minimization problem (equation 4.9) is then given by

$$\bar{\theta}_{k+1} = \theta_k - \lambda \nabla \bar{J}(\bar{\theta}_k) \tag{4.10}$$

which can be written as

$$\theta_{k+1} = \theta_k - \lambda \Gamma_1 \Gamma_1^T \nabla J(\theta_k) \tag{4.11}$$

since

$$\nabla \bar{J}(\bar{\theta}) = \frac{\partial J(\Gamma_1 \bar{\theta})}{\partial \bar{\theta}} = \Gamma_1^T \nabla J(\theta), \quad \Gamma_1 \bar{\theta} = \theta$$

and to obtain the scaled version of the steepest descent algorithm, $\Gamma = \Gamma_1 \Gamma_1^T$

$$\theta_{k+1} = \theta_k - \lambda \Gamma \nabla J(\theta_k) \tag{4.12}$$

which can be written in continuous-time as

$$\dot{\theta} = -\Gamma \nabla J(\theta) \tag{4.13}$$

(Ioannou and Sun, 2012; Leer, 2016).

Estimation Error

Now that $J(\theta)$ is minimized with respect to θ by using the gradient method, the next step is to develop the estimation error. The parametric model (equation 4.2) can be written on the form

$$w = \theta^{*T} \phi \tag{4.14}$$

since θ^* is constant, and where $\phi = W(s)\psi$. By using equation 4.14 the estimate $\widehat{w}$ of w is generated at time t as

$$\widehat{w} = \theta^T \phi \tag{4.15}$$

where $\theta(t)$ is the estimate of θ^* at time t. With this at hand the normalized estimation error ϵ is developed as

$$\epsilon = \frac{w - \widehat{w}}{m^2} = \frac{w - \theta^T \phi}{m^2} \tag{4.16}$$

where $m^2 = 1 + n_s^2$ and n_s^2 is the normalizing signal designed so that

$$\frac{\phi}{m} \in \mathcal{L}_\infty \tag{4.17}$$

(Ioannou and Sun, 2012; Leer, 2016).

Instantaneous Cost Function

The instantaneous cost function is a simple quadratic function given by

$$J(\theta) = \frac{\epsilon^2 m^2}{2} = \frac{(w - \theta^T \phi)^2}{2m^2}. \tag{4.18}$$

From equation 4.13 the minimization of $J(\theta)$ is generated, and the only unknown part of this equation is the term $\nabla J(\theta)$, because $\Gamma = \Gamma^T > 0$ is referred to as the adaptive gain and scaling matrix

$$\nabla J(\theta) = -\frac{(w - \theta^T \phi)\phi}{m^2} = -\epsilon\phi \tag{4.19}$$

then the adaptive law for generating $\theta(t)$ is given by

$$\dot{\theta} = \Gamma\epsilon\phi \tag{4.20}$$

which is referred to as the *gradient algorithm* (Ioannou and Sun, 2012; Leer, 2016).

4.1.2 Parameter Estimator

The method described in section 4.1.1 gives every equation that is needed to make an on-line parameter estimator using the gradient method with instantaneous cost for the system 4.37−4.40. For simplicity, the equations needed will be repeated below

$$\hat{w} = \theta^T \phi \tag{4.21}$$

$$\epsilon = \frac{w - \hat{w}}{m^2} \tag{4.22}$$

$$\dot{\theta} = \Gamma\epsilon\phi \tag{4.23}$$

$$m^2 = 1 + n_s^2 = 1 + \alpha\phi^T\phi \tag{4.24}$$

θ is the parameter estimate vector, and α, Γ are positive constants. Theorem 4.3.2 in (Ioannou and Sun, 2012) gives independence of the boundedness of the signal vector ϕ and ϕ is persistently exciting (PE), so that $\theta(t)$ converges exponentially to θ^*. Proof is found in (Ioannou and Sun, 2012; Leer, 2016).

4.2 Least-Squares

This old method can be dated back to the eighteenth century where Gauss used it to determine the orbit of the plants. Along with the gradient method (see section 4.1), the least-squares are an on-line parameter estimation and the adaptive laws are developed using normalization (Ioannou and Sun, 2012). The basic idea of the least-squares are described in (Ioannou and Sun, 2012) as:

> ”Fitting a mathematical model to a sequence of observed data by minimizing the sum of the squares of the difference between the observed and computed data. In doing so, any noise or inaccuracies in the observed data are expected to have less effect on the accuracy of the mathematical model.”

The unknown parameters in this method are given in a linear form, which makes it easy to apply and analyze. This apply that the parametric model is linear and is given in equation 4.14.

This method have the estimate $\hat{w}$ of w and the normalized estimation error

$$\hat{w} = \theta^T \phi, \quad \epsilon = \frac{w - \hat{w}}{m^2} = \frac{w - \theta^T \phi}{m^2} \tag{4.25}$$

which is generated the same way as in section 4.1, and where $m^2 = 1 + n_s^2$, $\theta(t)$ is the estimate of θ^* at time t, and m satisfies $\phi/m \in \mathcal{L}_\infty$. The equations known as the continuous-time recursive least-squares algorithm with forgetting factor

$$\dot{P} = \beta P - P\frac{\phi\phi^T}{m^2}P, \quad P(0) = P_0 = Q_0^{-1} \tag{4.26}$$

$$\dot{\theta} = P\epsilon\phi \tag{4.27}$$

where P is the covariance matrix, β is a positive constant, ϵ,m^2 defined above and ϕ is the signal vector. The derivations of equations 4.26-4.27 can be found in (Ioannou and Sun, 2012). The stability properties to the least-squares algorithm depends on the value of β, which is known as the forgetting factor in equation 4.26.

In section 4.3, the method used is the least-squares with forgetting factor, but it is also PE. The stability properties is described in (Ioannou and Sun, 2012) as:

> **Corollary 4.3.2** If n_s, $\phi \in \mathcal{L}_\infty$ and ϕ is PE then the recursive least-squares algorithm with forgetting factor $\beta > 0$ given by (4.26) and (4.27) guarantees that P, $P^{-1} \in \mathcal{L}_\infty$ and that $\theta(t)$ converges exponentially to θ^*.

This method is appropriate when the main objective is parameter estimation of stable plants with parameter convergence (Ioannou and Sun, 2012).

4.3 Base Model

In this thesis, the model from (Ragazzon et al., 2016) will be used as a base. The model consists of cantilever-sample dynamics and a parameter estimator (figure 4.1). The cell model consists of springs and dampers, and the goal is to get a good estimate which mimics that of a real biological cell and its mechanics. Most of this section can be found in (Ragazzon et al., 2016).

4.3.1 Cantilever-Sample Dynamics

The cantilever-sample dynamics consists of three parts; cantilever dynamics, tip geometry and sample force. These parts are supposed to provide a dynamical description of a cantilever interaction with a general viscoelastic sample material (Ragazzon et al., 2016).

Figure 4.2 shows that the best way to measure is when the position of the tip of the AFM is given in the coordinate system (x,y,z). As mentioned earlier, the model consists of spring-damper elements in the xy-axis. These elements can be compressed in z-direction, which gives a coordinate system of (x,y,z). Fig.2 in (Ragazzon et al., 2016) describes the interaction between the cantilever and the cell. From the description, the vertical Z position is given by the cantilever deflection, D, and the cantilever base position, U.

$$Z = U - D. \tag{4.28}$$

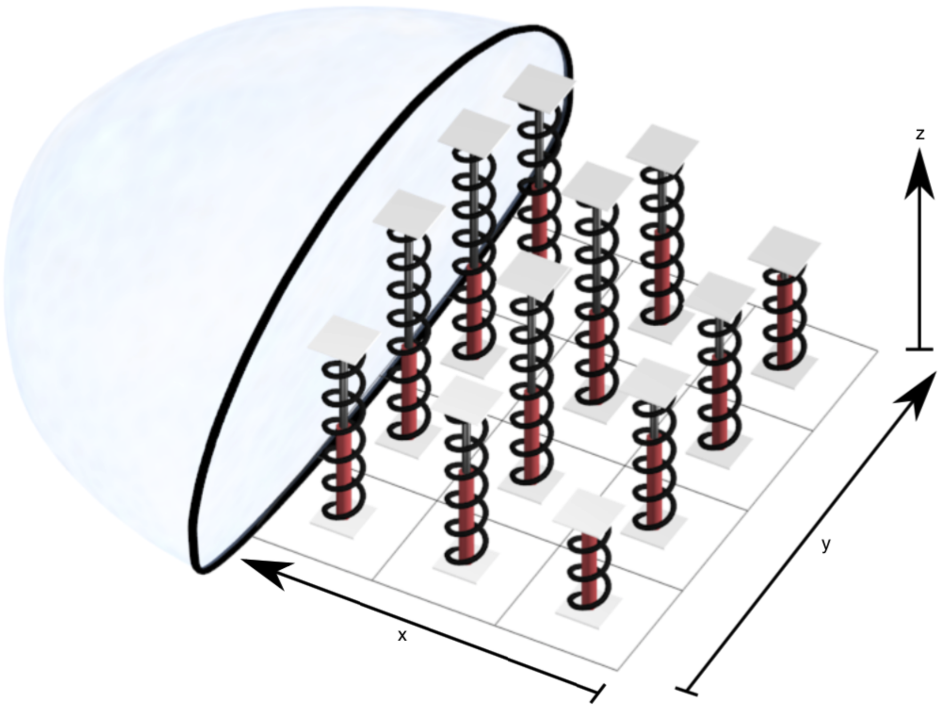

Figure 4.2: Biological cell modeled by spring-damper elements from (Ragazzon et al., 2016).

Cantilever Dynamics

The cantilever dynamics is a spring-damper system, and is approximated from its first resonance mode.

$$\begin{aligned} M\ddot{Z} &= KD + C\dot{D} + F_{sample} \\ &= K(U - Z) + C(\dot{U} - \dot{Z}) + F_{sample} \end{aligned} \tag{4.29}$$

where M is the effective mass of the cantilever, K is the cantilever spring constant, C is the cantilever damper constant, and F_{sample} is the force from the sample acting on the cantilever (Ragazzon et al., 2016).

Tip Geometry

From equation 4.28 the cantilever tips position determines the deflection and the motion of each individual spring element. The tip has a spherical shape with radius R. This gives a relationship between the tip and the ith sample elements position, z_i, and the velocity, $\dot{z}_i$, (Ragazzon et al., 2016).

$$z_i = Z - \sqrt{R^2 - (X - x_i)^2 - (Y - y_i)^2} \tag{4.30}$$

$$\dot{z}_i = \dot{Z} \tag{4.31}$$

where $\dot{X}$, $\dot{Y}$ is assumed to be zero when the tip is in contact with the sample, and ($\dot{x}_i$,$\dot{y}_i$) is the lateral position of the ith sample element (Ragazzon et al., 2016).

Sample Force

There is a rest position, z_i^0, for the ith spring-damper element and this represents the sample topography at the given lateral position (x_i,y_i). z_i from equation 4.30 gives the new position when the cantilever tip is pressed onto the sample and compresses the elements. The force generated by this is according to Hooke's law

$$F_{ki} = k_i \bar{z}_i \tag{4.32}$$

where k_i is the spring constant of the ith element, and

$$\bar{z}_i \triangleq z_i - z_i^0 \tag{4.33}$$

is the deflection of the ith sample element (Ragazzon et al., 2016). The same goes for the damping force

$$F_{ci} = c_i \dot{z}_i \tag{4.34}$$

where c_i is the damping constant. By combining equation 4.32 and 4.34 the total force acting from the sample on the tip is given as

$$F_{sample} = \sum_{i \in W} F_{ki} + F_{ci} \tag{4.35}$$

where $W = W(X, Y, Z)$ (Ragazzon et al., 2016), which is the active set of sample elements on which the tip is in contact with

$$W = \{i : \bar{z}_i < 0 \land (X - x_i)^2 + (Y - y_i)^2 < R^2\} \tag{4.36}$$

(Ragazzon et al., 2016).

Parametric System Model

Have the system on the form

$$w = \theta^{*T} \phi \tag{4.37}$$

where w is the signal scalar, $\theta^* = [c_j, k_j]^T$ is parameter vector and ϕ is the signal vector. From (Ragazzon et al., 2016) the system has the form

$$w = \frac{1}{\Lambda(s)}\left[(Cs+K)u - (Ms^2+Cs+K)d\right] \tag{4.38}$$

$$\phi = \left[\frac{s\bar{z}}{\Lambda(s)}, \frac{\bar{z}}{\Lambda(s)}\right] \tag{4.39}$$

$$\theta = [c_j, k_j]^T \tag{4.40}$$

with the adaptive law given in equations 4.26-4.27, and repeated below

$$\begin{aligned} \dot{\theta} &= P\epsilon\phi \\ \dot{P} &= \begin{cases} \beta P - P\frac{\phi\phi^T}{m^2}P, & \text{if } \|P(t)\| \leq R_0 \\ 0, & \text{otherwise} \end{cases} \\ P(0) &= P_0 \end{aligned} \tag{4.41}$$

(Ragazzon et al., 2016).

Method Properties

The least-square with forgetting factor have the same properties as the pure least-square with covariance resetting given by theorem 4.3.5 from (Ioannou and Sun, 2012). The theorem is given below for simplicity

(i) $\epsilon, \epsilon n_s, \theta, \dot{\theta} \in \mathcal{L}_\infty$.

(ii) $\epsilon, \epsilon n_s, \dot{\theta} \in \mathcal{L}_2$.

(iii) If $n_s, \phi \in \mathcal{L}_\infty$ and ϕ is *PE* then $\theta(t)$ converges exponentially to θ^*.

The properties for equation 4.41 given by theorem 4.3.5 from (Ioannou and Sun, 2012), are similar for those of the gradient method with instantaneous cost in subsection 4.1.2 (Ragazzon et al., 2016).

Chapter 5

Parallel Computing

In parallel computing, a problem is solved by using multiple compute nodes. The given problem is broken into discrete parts which can be solved concurrently. These parts are then transformed into a series of instructions that can be executed independently on different compute nodes. This differs from serial computing, where a problem is broken into discrete series of instructions. These instructions are sequentially executed, and there can only be one instruction executed at a given moment of time. Also, they are executed on a single CPU (central processing unit) (Barney et al., 2010).

Solving complex problems tend to be time consuming. By carefully dividing the problem into smaller parts, these parts can be executed in parallel on several compute nodes, taking advantage of often highly specialized hardware (internal network, storage capabilities and high speed computing power). Different parts of the problem can even be solved with different problem solvers (Barney et al., 2010).

Parallel computing really shines when it comes to calculating complex systems, i.e. weather systems, orbital movements, particle collisions, DNA sequencing etc. Theses type of machines are expensive and therefore mostly used in science, engineering and large scale industrial and commercial companies (Barney et al., 2010).

5.1 Vilje

Vilje is a HPC (High-Performance Computing) computer procured by NTNU together with met.no and UNINETT Sigma. It is used for different purposes: numerical weather prediction, research, master thesis, etc. The name Vilje comes from Vilje or Vili, which is the brother of Odin from Norse Mythology (NOTUR, 2016). Vilje is a computer with 1404 nodes, where every node has 2 eight core CPU and 32 GB RAM.

In (Leer, 2016), the simulation time was high, ending with Matlab crashing due to local storage exhaustion when the simulation ran for too long. By using Vilje with it's parallel computing abilities this is no longer be a problem. Also, it is possible to run several simulations in parallel.

Figure 5.1: The HPC Vilje at NTNU.

Figure 5.2: Vilje at NTNU.

5.1.1 Matlab on Vilje

When using parallel computing with Matlab, the problem was not that the code needed to be changed, but to make it usable on Vilje. In order to run Matlab programs on Vilje, several changes to the original code has been made.

Several problems relating to different versions of Matlab on Vilje occurred. The Matlab version R2014a is the most stable version to run on Vilje. However, several models and scripts for the Simulink model were written in Matlab version R2015a, thus will not run on in Vilje's Matlab environment due to non working backward compatibility. In order to fix this, Matlab version R2016a was installed on Vilje. However, too many errors were encountered, forcing the team to uninstall the version. Therefore, one had to wait for version R2016b to come out before running the program with parallel computing. Still, there were problems running the simulation on Vilje. So, a quick fix was made for using the new version R2016b. By running a Matlab program that displayed ”donothing” first, it worked.

5.1.2 Running a Program on Vilje

A bash script was used to initiate simulation runs on Vilje. The following commands had to be included in the start of the bash script

Listing 5.1: Basic needed commands

```
#!/bin/bash
#
#PBS -N jobname
#PBS -A accountname
#PBS -l select=1:ncpus=32:ompthreads=16
#PBS -l walltime=01:00:00
#PBS -q workq
#
cd $PBS_O_WORKDIR
```

In the code 5.1, all the commands are basic features that is needed to run a program on Vilje. These commands represent the job name, which account is running the program, a message to the queue system (PBS), maximum run time before the job is shut down and that it will be added to the queue.

Listing 5.2: Commands when using Matlab

```
#Basic needed commands
module load matlab/R2016b

matlab -nodisplay -nosplash -r matlabFile
```

Code listing 5.2 shows the commands needed to load the Matlab program onto Vilje, as well as a description of what Matlab file to run ("ViljeJob.m" in appendix B). The simulation could then be started by running the bash script through terminal access using the qsub command, as shown in listing 5.3.

Listing 5.3: Run Program From Command Line

```
qsub filename.sh
```

5.1.3 Challenges with Vilje

There have been some challenges with using Vilje. In the beginning, the main problem was to make the code usable on Vilje. Vilje do not support GUI (graphical user interface), which means that every part of the code containing any form of GUI had to be removed. In this process, another problem was discovered. The Matlab version which the simulations run on differs from the most stable version to run on Vilje. The Matlab version problem was described in detail in subsection 5.1.1.

In theory, there is possible to run up to twenty jobs on Vilje at the same time, but when dealing with programs that requires a large amount of storage it is important not to use up the assigned storage per user. Due to possible storage problems, the maximum number of jobs was set to seven.

Before the quick fix (described in subsection 5.1.1), there was a period of time where it was not possible to use Vilje, due to Matlab version problems. However, after the quick fix was introduced, there were no problems running simulations on Vilje.

In November, there was an internal error on Vilje and large update that made it unable to use Vilje in these periods.

5.2 Ways to Analyse the Results When Using Parallel Computing

When using parallel computing, several jobs are started simultaneously. This means that there will be multiple results, and these results need to be analysed. Therefore, some method(s) need be to used to determine the quality of these results. Three methods were chosen, bias, relative tolerance and the rate of convergence. The theory till these methods can be found in subsections 2.3.2-2.3.4.

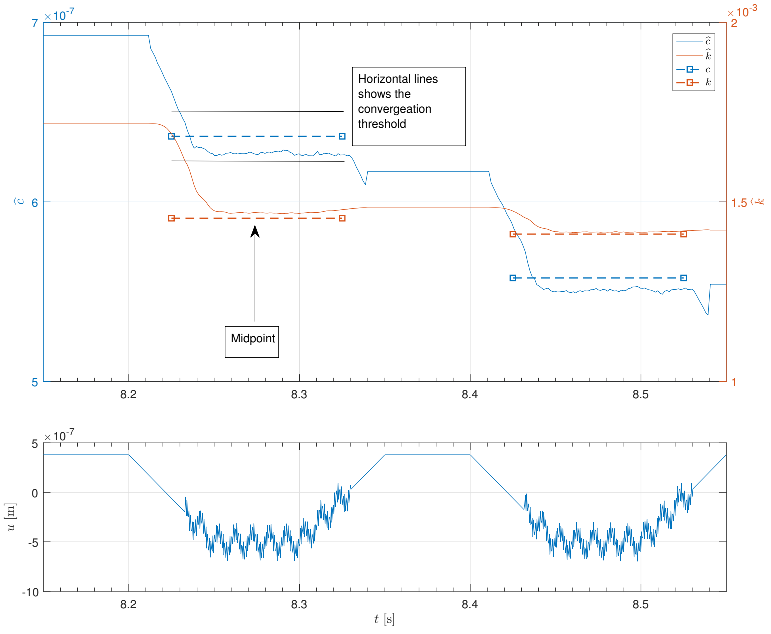

Figure 5.3: Shows how the analysing tools is defined in section 5.2.

5.2.1 Bias

When implementing a procedure based on the theory presented in subsection 2.3.2 in Matlab, some assumptions were made. Firstly, the bias is found by taking the mean over a estimated sequence. Secondly, when looking at figure 5.3, there are four time intervals defining the real values at the displayed time period. The sequences will be defined from the midpoint to the end on each of these time intervals. Then, equation 2.34 is used to find the bias.

5.2.2 Relative Tolerance

The relative tolerance (described in subsection 2.3.3), gives the percentage difference between the real and estimated values (see equation 2.36). This method uses the same assumptions as in subsection 5.2.1, because equation 2.36 uses the bias (in the equation known as the error, ϵ).

5.2.3 Rate of Convergence

The methods described in the two previous subsections 5.2.1-5.2.2, will only give an answer to how good the estimations are from a defined part of the time intervals. A small bias indicates a good estimation, which correspond to a small relative tolerance. When the bias and relative tolerance is satisfactory, but observing from the figure that the estimated values uses long time to reach the real values. A new analysing tool are introduced to handle problems were this occurs. Therefore, the rate of convergence is introduced and found. The theory is described in subsection 2.3.4.

To implement this in Matlab, some assumptions have to be made. The starting points of the measurement is given in figure 5.3 and the end points will be when the estimation has reached 95 $\%$ of the real values. The real values exist for $0.1s$ each. This means the results in rate of convergence will be less than $0.1s$. Therefore, when analysing the result it will be easier to see how many percent of $0.1s$ it takes for the estimated parameter to converge to 95$\%$ of the real values. This is given as

$$RoC_{\%} = \frac{RoC}{0.1} \cdot 100\% \tag{5.1}$$

where $RoC_{\%}$ is the rate in percent, RoC is the rate of convergence in seconds (Walpole et al., 1993). However, when comparing two results it can be interesting to find the increase or decrease in rate of convergence in percent. Here, equation 2.36 can be used. The difference will be that ϵ will be the deviation between the two results and a will be $0.1s$.

5.2.4 Matlab Implementation

The implementations of the bias, relative tolerance and the rate of convergence in Matlab is given in pseudo code below, while the Matlab script is given in appendix B

Listing 5.4: Pseudo code for finding the bias, relative tolerance and rate of convergence.label

```
%% Find the bias, relative tolerance and the rate of
    convergence for the different tests
Retrive results from logout
t1__, t2__  % Real values defined in time [s] [1x2]
c1_, c2_, k1_, k2_ %the real damper and spring values
t_ % time axis given in 300 steps
theta_ = interp1(t, theta, t_); % theta_=[c_hat k_hat];
c_hat, k_hat % Estimated damper and spring values

%% Finding the bias
% Find the indices on the time axis where the real values
    lies
idx1 = find(t_>=t1__(1) & t_<=t1__(2));
idx2 = find(t_>=t2__(1) & t_<=t2__(2));

% Values in c_hat and k_hat that correspond to the indices
    in t1__ and t2__
c_hat1 = c_hat(idx1(1):idx1(end));
c_hat2 = c_hat(idx2(1):idx2(end));
k_hat1 = k_hat(idx1(1):idx1(end));
k_hat2 = k_hat(idx2(1):idx2(end));

% Calculate the bias from the middle of the real values
% Find c_hat and k_hat for these values
c_hat1_mid = c_hat1(end/2:end);
c_hat2_mid = c_hat2(end/2:end);
k_hat1_mid = k_hat1(end/2:end);
k_hat2_mid = k_hat2(end/2:end);

% Bias
bias_c1 = c1_ - mean(c_hat1_mid);
bias_c2 = c2_ - mean(c_hat2_mid);
bias_k1 = k1_ - mean(k_hat1_mid);
bias_k2 = k2_ - mean(k_hat2_mid);

%% Relative tolerance (bias/real value *100%)
rel_error_c1 = bias_c1/c1_ *100;
rel_error_c2 = bias_c2/c2_ *100;
rel_error_k1 = bias_k1/k1_ *100;
rel_error_k2 = bias_k2/k2_ *100;

%% Finding the rate of convergence
tid1, tid2 %time the real values exist

% The estimated values converge between
```

```
conv_c1 = [c1_105 c1_95]; %epsilon_c1
conv_c2 = [c2_105 c2_95]; %epsilon_c2
conv_k1 = [k1_105 k1_95]; %epsilon_k1
conv_k2 = [k2_105 k2_95]; %epsilon_k2

% Indices where the estimated values are between 95% and
    105% of real value
c1_idx=find(c_hat1>=conv_c1(1) & c_hat1<=conv_c1(2));
c2_idx=find(c_hat2>=conv_c2(1) & c_hat2<=conv_c2(2));
k1_idx=find(k_hat1>=conv_k1(1) & k_hat1<=conv_k1(2));
k2_idx=find(k_hat2>=conv_k2(1) & k_hat2<=conv_k2(2));

% Time the estimated values crosses epsilon
t_eps_c1 = tid1(c1_idx(1));
t_eps_c2 = tid2(c2_idx(1));
t_eps_k1 = tid1(k1_idx(1));
t_eps_k2 = tid2(k2_idx(1));

% Rate of Convergence
RoC_c1 = t_eps_c1-t1__(1);
RoC_c2 = t_eps_c2-t2__(1);
RoC_k1 = t_eps_k1-t1__(1);
RoC_k2 = t_eps_k2-t2__(1);
```

Chapter 6

Cantilever Dynamics

The cantilever dynamics have great impact on the force and image measurements of the AFM (see chapter 3). Therefore are these properties very important in the interaction between the sample and the tip. There are many factors that interfere with these properties: Shape and material of the tip, where the AFM is operated (air, gases, vacuum, liquid, etc.) and the dimensions of the cantilever. These properties are typically based on theoretical estimates or have only been made for the supposedly representative sample (Hutter and Bechhoefer, 1993).

The task is to find an equation for the cantilever dynamics and transforming it into a form that is suitable for implementation of various adaptive estimation methods presented in (Ioannou and Sun, 2012).

6.1 Cantilever Dynamic Equation

Using equation 4.29 for the spring-damper system from (Ragazzon et al., 2016), which is repeated below

$$\begin{aligned} M\ddot{Z} &= KD + C\dot{D} + F_{sample} \\ &= K(U - Z) + C(\dot{U} - \dot{Z}) + F_{sample} \end{aligned} \tag{6.1}$$

with the following relationship

$$Z = U - D \Rightarrow \dot{Z} = \dot{U} - \dot{D} \tag{6.2}$$

it is possible to identify the following parameters by on-line parameter estimation:

K	=	spring constant
C	=	damper constant
M	=	effective mass of cantilever

Assuming to know the cantilever base position, U, and the cantilever deflection, D. These two can be used to find the vertical position of the cantilever tip, Z, using equation 6.2(Ragazzon et al., 2016). The only difficulty with U and D is that they have unknown scaling. They are in volt ($[V]$), not meter. The scaling constant can be found with some simple experiments in the nanolab, and that is why they are assumed to be known. Equation 6.1 need to be rewritten into linear-in-parameter form (equation 4.21):

$$w = \theta^{*T}\phi \tag{6.3}$$

By using the Laplace transform of derivatives (Kreyszig, 2010), equation 6.1 from (Ragazzon et al., 2016) becomes

$$M[s(Z - Z(0)) - Z'(0)] - \frac{F_{sample}}{s} = K\frac{D}{s} + C(D - D(0)) \tag{6.4}$$

then rearranging it

$$\frac{F_{sample}}{s} = M[s(Z - Z(0)) - Z'(0)] - K\frac{D}{s} - C(D - D(0)) \tag{6.5}$$

The system is now on the form (equation 6.3)

$$w' = \frac{F_{sample}}{s} \tag{6.6}$$

$$\theta = [M, K, C]^T \tag{6.7}$$

$$\phi = \left[s\left(Z - Z(0)\right) - Z'(0), -\frac{D}{s}, -\left(D - D(0)\right) \right]^T \tag{6.8}$$

The important part in equation 6.5 is F_{sample}, which is defined as the total force acting from the sample on the tip.

6.2 Defining Parameters for Simulation

When lowering the cantilever down slowly towards the sample, the total force acting from the sample on the tip can be assumed to be equal to Hooke's law (equation 3.1), which means that F_{sample} becomes

$$F_{sample} = F = -k_c\delta_c = -k_c\bar{z}_c \tag{6.9}$$

(Cappella and Dietler, 1999). The cantilever deflection z_c can be measured, which means that only unknown parameter is equation 6.2 is the cantilever spring constant, k_c. Typical values for this parameter varies between 0.01 N/m and 50 N/m, which is the variation between soft and hard cantilevers. A good assumption for k_c is

$$k_c = 0.2 \quad N/m. \tag{6.10}$$

A stable filter of relative degree 1 is added on each side of equation 6.5. The degree is chosen because s appear in w'. The filter makes all signals proper and avoids pure differentiation. The chosen filter

$$\frac{1}{\Lambda(s)} = \frac{1}{\tau_1 s + 1} \tag{6.11}$$

where τ_1 is the tunable positive filter constant. This gives the system

$$\frac{w^{'}}{\Lambda(s)} = [M, K, C] \left[\frac{s(Z - Z(0)) - Z'(0)}{\Lambda(s)}, \frac{-\frac{D}{s}}{\Lambda(s)}, \frac{-(D - D(0))}{\Lambda(s)} \right]^T \tag{6.12}$$

and on the form (equation 6.3)

$$\begin{aligned} w &= \frac{1}{\Lambda(s)} \left(\frac{F_{sample}}{s} \right) \\ \phi &= \left[\frac{s\,(Z - Z(0)) - Z'(0)}{\Lambda(s)}, \frac{1}{s} \frac{-D}{\Lambda(s)}, \frac{-\,(D - D(0))}{\Lambda(s)} \right]^T \\ \theta &= [M, K, C]^T \end{aligned} \tag{6.13}$$

Before the system can be simulated, an input signal that is PE have to be chosen for the system (equation 6.6-6.8).

Chapter 7

Simulation Results

The simulations in this chapter are split into experiments. These experiments consist of multiple jobs, where the name job comes from running simulations on Vilje (section 5.1).

When simulating the model and using the analysing tools described in section 5.2, the results will consist of four solutions per analysing tool. That means that the bias, the relative tolerance and the rate of convergence will together produce twelve result sets per simulation. The bias will produce small numbers (e.g. 6.4532e-09) which will not be listed in this chapter, because of the difficulty of relating the number to the real value. Instead, the relative tolerance will be listed, because it gives the deviation between the two in percent. Also, the relative tolerance will only consist of two decimals in this chapter, while in the tables in appendix A it will be given with four decimals. The rate of convergence gives the amount of time in seconds it takes for the system to converge to the real values. The bias can be found in the tables in appendix A.

7.1 Simulation Setup

When starting the simulation, the setup will be the same for all the experiments. The setup consist of parameters that can be tuned in order to improve the result. These parameters are: gain matrix (Γ), the tap period, simulation solver, relative tolerance in the simulation solver, the cantilever oscillation period ($T0$), the cantilever oscillation frequency ($f0$), the tunable positive constant in the filter (τ). The starting setup is given in table 7.1. The simulations were performed on my own computer and on Vilje (see section 5.1) with Matlab version R2016b.

The gain matrix consist of two diagonal elements, where the first diagonal element is the gain on the damper constant, while the second diagonal element is the gain on the spring constant.

Simulation solver	$ode45$
Gamma, Γ	$1e-0 \cdot diag([10e8 \quad 10e18])$
Tap period	0.2
Relative Tolerance (sim. solver)	$1.00e-09$
τ	$0.1 \cdot T0$
T0	$1/f0$
f0	$2.00e04$

Table 7.1: The setup when starting the simulations.

The different experiments in this chapter will conduct changes on the parameters in table 7.1. Every change will be described in the relevant experiment.

7.2 Continuing Work From Final Year Assignment

The results from the Final Year Assignment (Leer, 2016) differed from (Ragazzon et al., 2016), because there was a large difference between the two methods (least square with forgetting factor and the gradient method using instantaneous cost) when it came to converging towards the real values, c and k.

7.2.1 Overview

In (Leer, 2016), the main task was to compare the convergence result between two different adaptive estimation methods, least square with forgetting factor and the gradient method using instantaneous cost, on a mathematical model of a biological cell and AFM (figure 4.1). The results from the least square method is given in (Ragazzon et al., 2016) and the convergence result is given as a recap in Figure 7.1. Using the same model from (Ragazzon et al., 2016), the same gain, Γ, and only changing the on-line parameter estimator from least square with forgetting factor to the gradient method using instantaneous cost. The result is given in Figure 7.2. The best result from (Leer, 2016) is given in Figure 7.3. The different gains, Γ, and on-line parameter estimation method in the Figure 7.1- 7.3 is given in Table 7.2.

Figure	**Gamma,** Γ	**On-Line Parameter Estimation Method**
7.1	$1e-0 \cdot diag([10e8 \quad 10e18])$	Least square with forgetting factor
7.2	$1e-0 \cdot diag([10e8 \quad 10e18])$	Gradient method using instantaneous cost
7.3	$1e-0 \cdot diag([8e7 \quad 15e19])$	Gradient method using instantaneous cost

Table 7.2: An overview of the different Γ in Figure 7.1- 7.3.

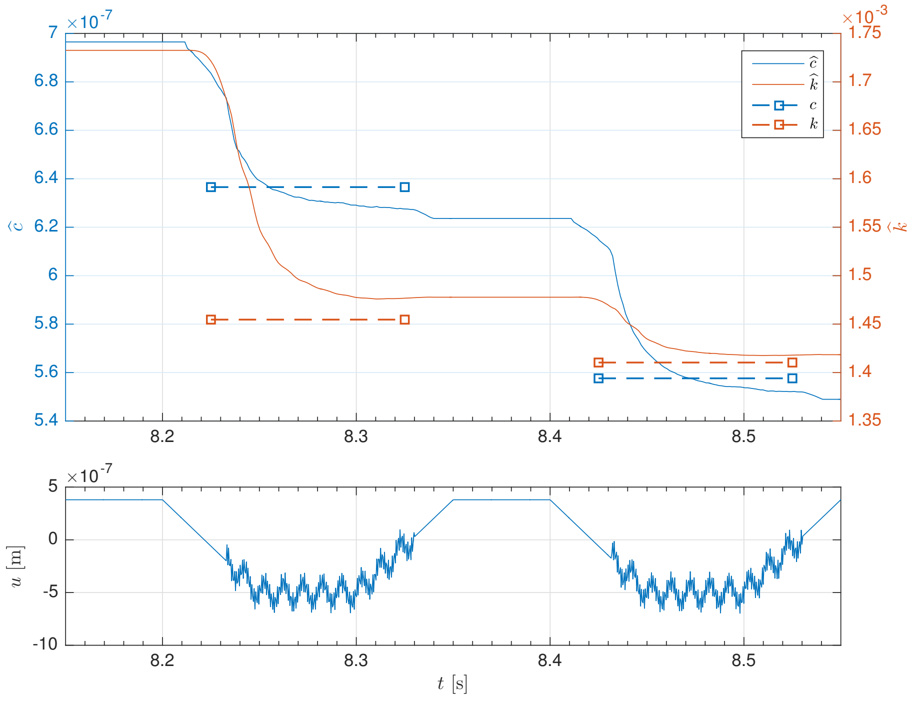

Figure 7.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the least square with forgetting factor, and the cantilever position input u over time. This figure is a result from (Ragazzon et al., 2016).

7.2.2 Tuning the Gain Matrix

The setup when starting the simulations is given in table 7.1. When tuning in (Leer, 2016), the gain on the damper constant was decreased while the spring constant was increased. Here, the opposite is tried, because of bad results in (Leer, 2016). First, by only decreasing the spring constant and later tune from that result. The first change, gave the Figure 7.4. It is an okay estimate for $\widehat{c}$, while the $\widehat{k}$ never reaches the real spring constant, k. When continuing tuning, the end result is given in figure 7.5 and figure 7.6. The different tuning cases for the gain matrix, Γ, is given in table A.1. The related figures to table A.1 are given in appendix C1.

7.2.3 Comparing the Gradient Method with the Least-Square Method

The best result with the gradient method using instantaneous cost is given in figure 7.6, and the gain matrix used is given as Test 15 in table A.1, is repeated below

$$\begin{aligned}\Gamma =&\ \ 1e-0\cdot diag([5e8 \quad 10e14]) \\ =&\begin{bmatrix}5.0\cdot 10^{8} & 0\\ 0 & 1.0\cdot 10^{15}\end{bmatrix}\end{aligned} \tag{7.1}$$

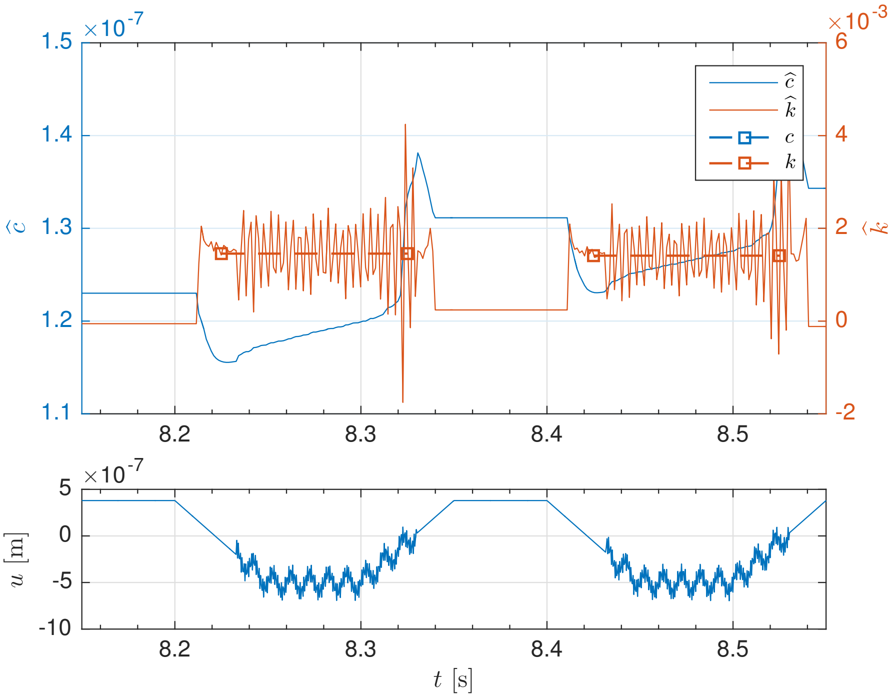

Figure 7.2: The estimates of $\widehat{k}$ and $\widehat{c}$ with the gradient method using instantaneous cost, and the cantilever position input u over time. It was the first simulation using the gradient method with the same Γ as in (Ragazzon et al., 2016).

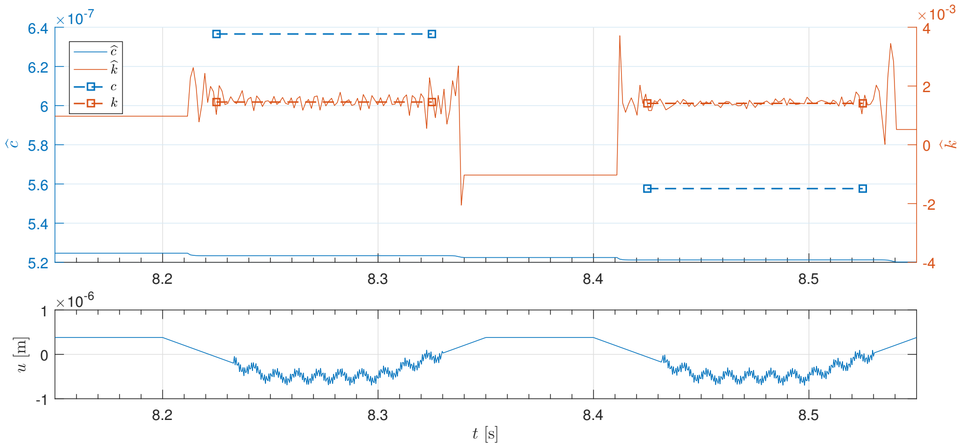

Figure 7.3: The estimates of $\widehat{k}$ and $\widehat{c}$ with the gradient method using instantaneous cost, and the cantilever position input u over time. This is the best result from the Final Year Project (Leer, 2016).

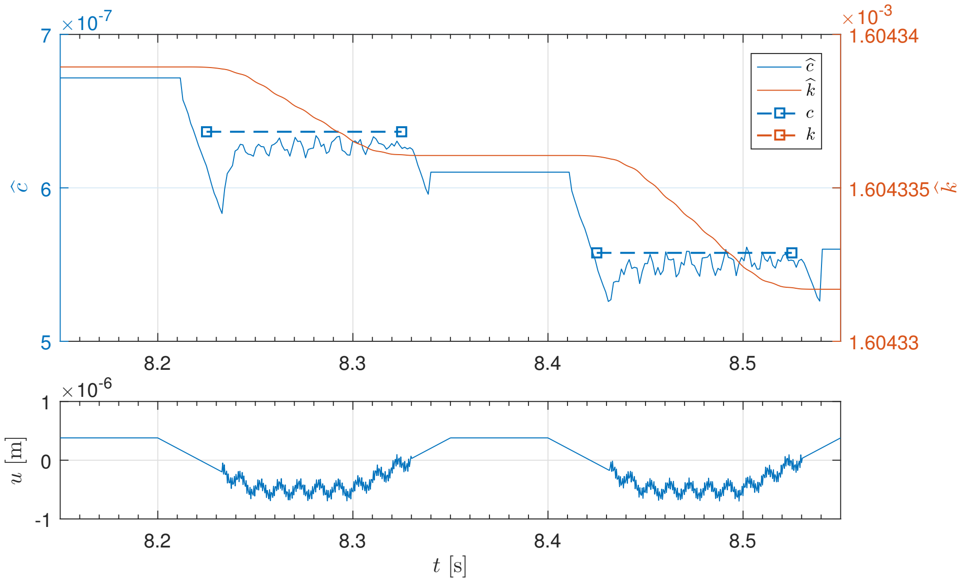

Figure 7.4: The estimates of $\widehat{k}$ and $\widehat{c}$ with the gradient method using instantaneous cost, and the cantilever position input u over time.

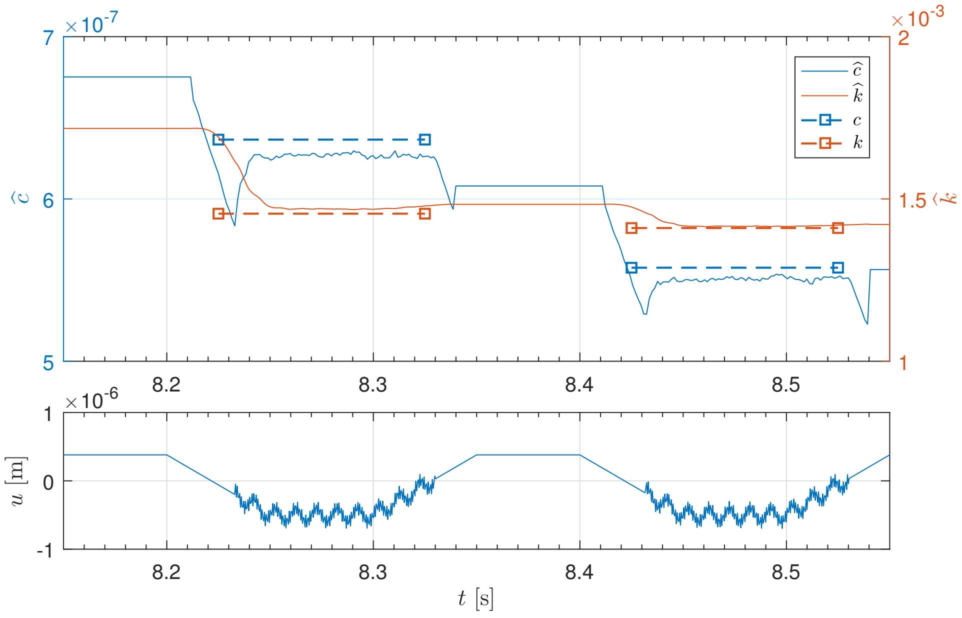

Figure 7.5: The estimates of $\widehat{k}$ and $\widehat{c}$ with the gradient method using instantaneous cost, and the cantilever position input u over time. Gamma10 from table7.1

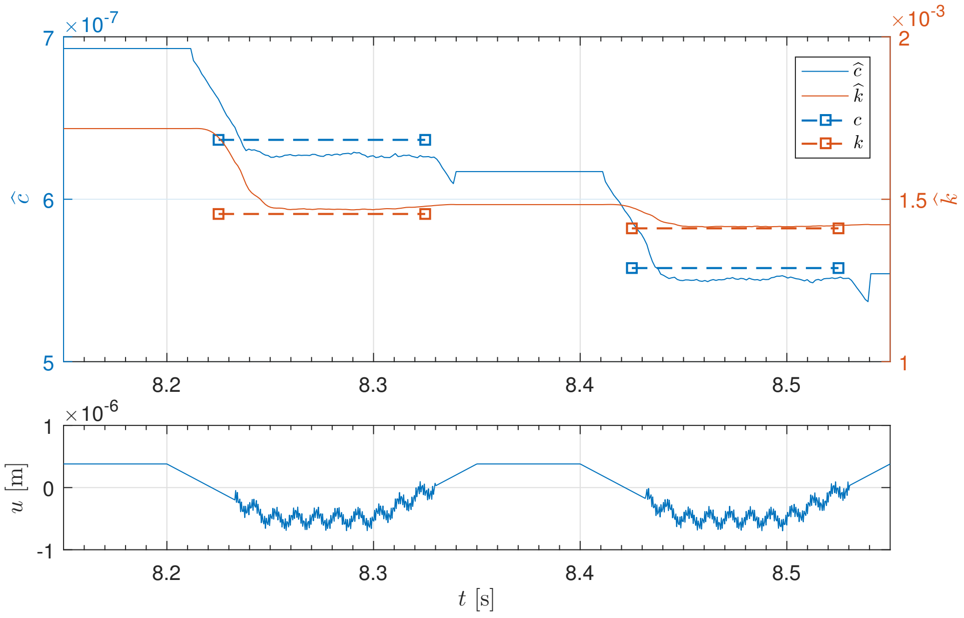

Figure 7.6: The estimates of $\widehat{k}$ and $\widehat{c}$ with the gradient method using instantaneous cost, and the cantilever position input u over time. Test 15 from table A.1

This gain matrix replaces the gain matrix given in the simulation setup (table 7.1) in the Simulink model using the least square with forgetting factor as the on-line parameter estimation method. Here, the gain matrix is used as initial condition in the integration of $\dot{P}$ to P, which is given in equation 4.26.

By comparison, figure 7.1 and figure 7.7 are quite similar. Therefore, the relative tolerance and the rate of convergence (see subsection 5.2.4 for Matlab impementation) are used for further analysis. The results from the two analysing tools are given in table 7.3.

	Relative tolerance (%)				**Rate of convergence**			
Figure	RT c1	RT c2	RT k1	RT k2	RoC c1	RoC c2	RoC k1	RoC k2
7.1	1.14	0.68	-1.76	-0.58	0.0093	0.014	0.032	5.78e-04
7.7	1.14	0.67	-1.77	-0.58	0.0093	0.014	0.032	5.78e-04

Table 7.3: The relative tolerance and the rate of convergence in Figure 7.1 and Figure 7.7.

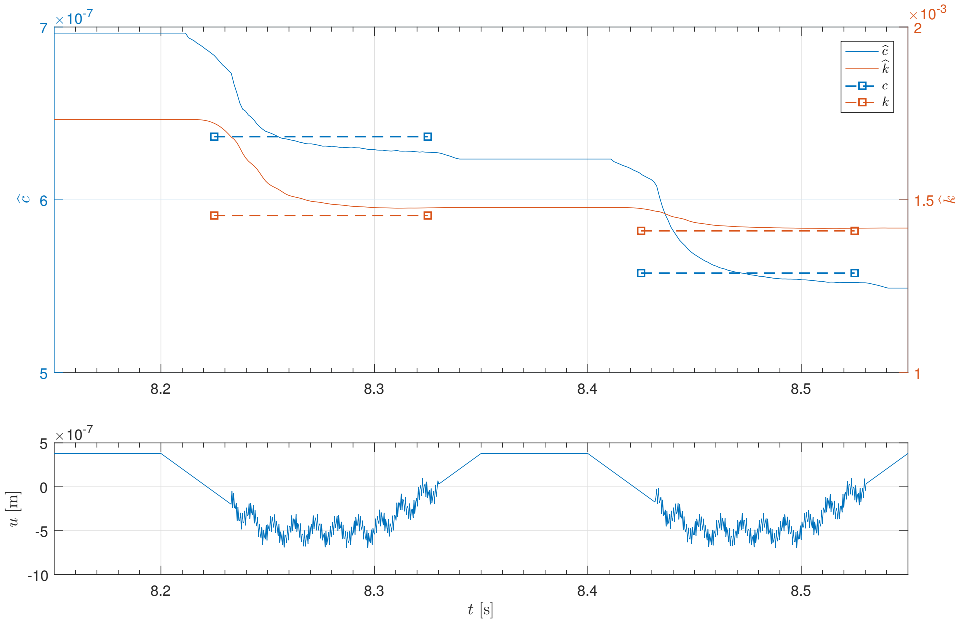

Figure 7.7: The estimates of $\widehat{k}$ and $\widehat{c}$ using the least square with forgetting factor, and the cantilever position input u over time. This is the result with Γ equal to equation 7.1.

7.3 Parallel Computing

The simulations conducted here, are split into different experiments, where every experiment consists of multiple jobs. Each experiment represent one specific change in the system. The changes affects the system and the on-line parameter estimation method.

When starting to use Vilje and parallel computing a new simulation setup is given, hence to the gain matrix found in equation 7.1. The new simulation setup is given in table 7.4 The

Simulation solver	$ode45$
Gamma, Γ	$1e-0 \cdot diag([5e8 \quad 10e14])$
Tap period	0.2
Relative Tolerance (sim. solver)	$1.00e-09$
τ	$0.1 \cdot T0$
T0	$1/f0$
f0	$2.00e04$

Table 7.4: The simulation setup when starting simulations with parallel computing.

analysing tools described in section 2.3 will be used as a simple procedure for analysing the performance of the system when dealing with multiple results.

7.3.1 Changing the Simulation Solver

It is possible that the result from subsection 7.2.2 can be improved by using a different simulation solver. The first experiment consist of six jobs with simulation setup equal to the parameters in table 7.4, except the gain matrix which is equal to equation 7.1.

Experiment 1	
Job	**Simulation solver**
1	$ode45$
2	$ode15s$
3	$ode23$
4	$ode23s$
5	$ode23t$
6	$ode113$

Table 7.5: Experiment 1: Changes the simulation solver. Mathematical description of these solvers are found in section 2.2.

The results from experiment 1 (table 7.5) is given as relative tolerance and rate of convergence in table 7.6 and the corresponding figures are found in appendix C2.

	Relative tolerance ($\%$)				**Rate of convergence ([s])**			
Job	RT c1	RT c2	RT k1	RT k2	RoC c1	RoC c2	RoC k1	RoC k2
1	1.51	1.19	-1.10	-0.43	0.0013	5.78e-04	0.016	5.78e-04
2	1.47	1.21	-1.11	-0.43	0.0013	5.78e-04	0.016	5.78e-04
3	1.52	1.21	-1.10	-0.43	0.0013	5.78e-04	0.016	5.78e-04
4	1.56	1.25	-1.10	-0.43	0.0146	5.78e-04	0.016	5.78e-04
5	1.44	1.14	-1.09	-0.43	0.012	5.78e-04	0.016	5.78e-04
6	1.52	1.21	-1.09	-0.42	0.0013	5.78e-04	0.016	5.78e-04

Table 7.6: The results corresponding to Experiment 1 (table 7.5), given as the relative tolerance and the rate of convergence.

7.3.2 Combining Changes

In this subsection, changes will be made on parameters effecting the model and the simulation solver. The results found in subsection 7.3.1 will be used to experiment with other parameter changes combined with simulation solvers.

Simulation Solver and Relative Tolerance

In this experiment, the simulation setup is equal to table 7.4, except for the simulation solver and the relative tolerance in the solver (section 2.2). Experiment 2 consist of six jobs and is given in table 7.7.

Experiment 2		
Job	**Simulation solver**	**Relative tolerance** (in sim. solver)
1	$ode45$	$1.00e-07$
2	$ode15s$	$1.00e-07$
3	$ode23$	$1.00e-07$
4	$ode23s$	$1.00e-07$
5	$ode23t$	$1.00e-07$
6	$ode113$	$1.00e-07$

Table 7.7: Experiment 2: Changes the simulation solver and the relative tolerance in the simulation solver. Mathematical description of these solvers are found in section 2.2.

	Relative tolerance $(\%)$				**Rate of convergence ([s])**			
Job	RT c1	RT c2	RT k1	RT k2	RoC c1	RoC c2	RoC k1	RoC k2
1	1.51	1.19	-1.10	-0.43	0.0013	5.78e-04	0.016	5.78e-04
2	1.46	1.22	-1.10	-0.43	0.0013	5.78e-04	0.016	5.78e-04
3	1.52	1.21	-1.10	-0.43	0.0013	5.78e-04	0.016	5.78e-04
4	1.56	1.25	-1.10	-0.43	0.0146	5.78e-04	0.013	5.78e-04
5	1.44	1.14	-1.09	-0.43	0.012	5.78e-04	0.016	5.78e-04
6	1.52	1.21	-1.09	-0.43	0.0013	5.78e-04	0.016	5.78e-04

Table 7.8: The results corresponding to Experiment 2 (table 7.7), given as the relative tolerance and the rate of convergence.

The corresponding figures to experiment 1 if found in appendix C3.

Simulation Solver and Tap Period

The simulation setup for this experiment is given in table 7.4, where the simulation solver and the tap period have been changed. The tap period is the time it takes for the tip on the cantilever to tap into the sample and go back into start position (section 3). When changing the tap period, it will allow the solvers to get more time for estimation and giving them time to converge toward the real values.

Experiment 3 is given in table 7.9 and consist of six jobs with corresponding results given as relative tolerance and rate of convergence in table 7.10. The corresponding figures to experiment 3 if found in appendix C4.

Experiment 3		
Job	**Simulation solver**	**Tap period**
1	$ode45$	0.205
2	$ode15s$	0.205
3	$ode23$	0.205
4	$ode23s$	0.205
5	$ode23t$	0.205
6	$ode113$	0.205

Table 7.9: Experiment 3: Changes the simulation solver and the tap period.

	Relative tolerance (%)				**Rate of convergence ([s])**			
Job	RT c1	RT c2	RT k1	RT k2	RoC c1	RoC c2	RoC k1	RoC k2
1	1.49	1.19	-1.08	-0.42	0.0093	0.0086	0.024	5.78e-04
2	1.47	1.23	-1.08	-0.42	0.0039	0.0086	0.024	5.78e-04
3	1.50	1.21	-1.07	-0.42	0.0013	0.0046	0.024	5.78e-04
4	1.57	1.25	-1.08	-0.42	0.0013	5.78e-04	0.024	5.78e-04
5	1.41	1.16	-1.07	-0.42	0.0013	5.78e-04	0.024	5.78e-04
6	1.88	1.51	-1.47	-0.96	0.024	5.78e-04	0.0079	0.022

Table 7.10: The results corresponding to Experiment 3 (table 7.9).

7.3.3 Optimize the Gain Matrix

After finding a good estimate for the gain matrix, Γ, in section 7.2. It was important to try to optimize the gain matrix when gaining access to tools like Vilje (section 5.1). Then it was possible to run multiple simulations at the same time and run simulations with large gain matrix. By running multiple jobs in parallel, small changes could be made on the two diagonal elements in the gain matrix. This made the optimization quicker, than running these jobs in sequence.

The simulation setup was similar to table 7.4. These simulations were split into two experiments. Experiment 4 consisted of changing the gain on the damper constant (table 7.11), while experiment 5 changed the gain on the spring constant (table 7.13).

Experiment 4	
Job	Γ (gain matrix)
1	$1e-0 \cdot diag([6e7 \quad 10e14])$
2	$1e-0 \cdot diag([7e7 \quad 10e14])$
3	$1e-0 \cdot diag([8e7 \quad 10e14])$
4	$1e-0 \cdot diag([9e7 \quad 10e14])$
5	$1e-0 \cdot diag([1e8 \quad 10e14])$
6	$1e-0 \cdot diag([2e8 \quad 10e14])$
7	$1e-0 \cdot diag([3e8 \quad 10e14])$
8	$1e-0 \cdot diag([4e8 \quad 10e14])$
9	$1e-0 \cdot diag([6e8 \quad 10e14])$
10	$1e-0 \cdot diag([7e8 \quad 10e14])$

Table 7.11: Experiment 4: Shows the fine tuning of the damper element in the gain matrix, Γ. The spring constant remains the same, while the damper element changes.

The results from experiment 4 (table 7.11) is given in relative tolerance and rate of convergence in table 7.12. Examining the results in experiment 4, shows a local minimizer. Although, the rate of convergence is unsatisfactory at this minimizer. Therefore, job 8 in results will be used as the optimal gain on the damper constant. The reason for this is the rate of convergence. It is $0.0187s$ faster than the rate of convergence in the local minimizer in job 3 ($0.02s$). Given in percentage, job 8 has a rate of convergence 18.7% quicker than job 3 (equation 5.1).

In experiment 4 we found the optimized gain on the damper constant, which means that optimized gain on the spring constant have to be determined. This will be done in experiment 5. The same method for finding the gain on the damper constant will be used. Keeping the gain on the damper constant equal to the one found in job 8 and vary the gain on the spring constant around the best spring constant gain found in subsection 7.2.2. Before starting experiment 5 the gain matrix is given as

$$\begin{aligned} \Gamma = &\ 1e-0 \cdot diag([4e8 \quad 10e14]) \\ = &\begin{bmatrix} 4.0 \cdot 10^8 & 0 \\ 0 & 1.0 \cdot 10^{15} \end{bmatrix} \end{aligned} \tag{7.2}$$

Experiment 5 is given in table 7.13, and the corresponding results in relative tolerance and rate of convergence are found in table 7.14.

Analysing the relative tolerance and rate of convergence in table 7.14, job 4 produces the best overall results, because of the large variations in relative tolerance from job 5 to job 13. Job 1 produced the best results in rate of convergence. However, the relative tolerance is

	Relative tolerance (%)				**Rate of convergence ([s])**			
Job	RT c1	RT c2	RT k1	RT k2	RoC c1	RoC c2	RoC k1	RoC k2
1	-0.59	-1.19	-1.12	-0.45	0.02	0.0367	0.016	5.78e-04
2	-0.28	-0.54	-1.11	-0.44	0.0213	0.0313	0.0163	5.78e-04
3	0.02	-0.08	-1.11	-0.44	0.02	0.0273	0.016	5.78e-04
4	0.28	0.25	-1.11	-0.44	0.02	0.0247	0.016	5.78e-04
5	0.50	0.49	-1.11	-0.44	0.0186	0.022	0.016	5.78e-04
6	1.38	1.15	-1.10	-0.43	0.012	0.0126	0.016	5.78e-04
7	1.49	1.19	-1.08	-0.42	0.0093	0.0086	0.024	5.78e-04
8	1.50	1.19	-1.10	-0.43	0.0013	0.0059	0.016	5.78e-04
9	1.51	1.19	-1.10	-0.43	0.0013	5.78e-04	0.016	5.78e-04
10	1.51	1.19	-1.10	-0.43	0.0013	5.78e-04	0.016	5.78e-04

Table 7.12: The results corresponding to Experiment 4 (table 7.11).

Experiment 5	
Job	Γ (gain matrix)
1	$1e-0 \cdot diag([4e8 \quad 6e16])$
2	$1e-0 \cdot diag([4e8 \quad 6e15])$
3	$1e-0 \cdot diag([4e8 \quad 2e15])$
4	$1e-0 \cdot diag([4e8 \quad 1e15])$
5	$1e-0 \cdot diag([4e8 \quad 9e14])$
6	$1e-0 \cdot diag([4e8 \quad 8e14])$
7	$1e-0 \cdot diag([4e8 \quad 7e14])$
8	$1e-0 \cdot diag([4e8 \quad 6e14])$
9	$1e-0 \cdot diag([4e8 \quad 5e14])$
10	$1e-0 \cdot diag([4e8 \quad 4e14])$
11	$1e-0 \cdot diag([4e8 \quad 3e14])$
12	$1e-0 \cdot diag([4e8 \quad 2e14])$
13	$1e-0 \cdot diag([4e8 \quad 1e14])$

Table 7.13: Experiment 5: Shows the fine tuning of the spring element in the gain matrix, Γ. The damper constant remains the same, while the spring element changes.

worse and the estimates oscillates around the real values (see figure 7.8). These oscillations does not appear in job 4 (figure 7.9). This means that the optimized gain matrix is equal to equation 7.2.

	Relative tolerance (%)				**Rate of convergence ([s])**			
Job	RT c1	RT c2	RT k1	RT k2	RoC c1	RoC c2	RoC k1	RoC k2
1	2.09	1.70	-1.09	-0.37	0.0013	0.0059	0.0013	5.78e-04
2	1.51	1.20	-1.18	-0.45	0.0026	0.0073	0.0039	5.78e-04
3	1.50	1.20	-1.14	-0.44	0.0026	0.0059	0.0106	5.78e-04
4	1.50	1.19	-1.10	-0.43	0.0013	0.0059	0.016	5.78e-04
5	1.14	0.68	-1.76	-0.85	0.0093	0.014	0.032	5.78e-04
6	1.14	0.67	-1.76	-0.85	0.0093	0.014	0.032	5.78e-04
7	1.14	0.67	-1.76	-0.85	0.0093	0.014	0.032	5.78e-04
8	1.14	0.67	-1.76	-0.85	0.0093	0.014	0.032	5.78e-04
9	1.14	0.67	-1.76	-0.85	0.0093	0.014	0.032	5.78e-04
10	1.14	0.67	-1.76	-0.85	0.0093	0.014	0.032	5.78e-04
11	1.14	0.67	-1.76	-0.85	0.0093	0.014	0.032	5.78e-04
12	1.14	0.67	-1.76	-0.85	0.0093	0.014	0.032	5.78e-04
13	1.48	1.19	-4.38	-1.43	0.0013	0.0059	0.0641	0.022

Table 7.14: The results corresponding to Experiment 5 (table 7.13).

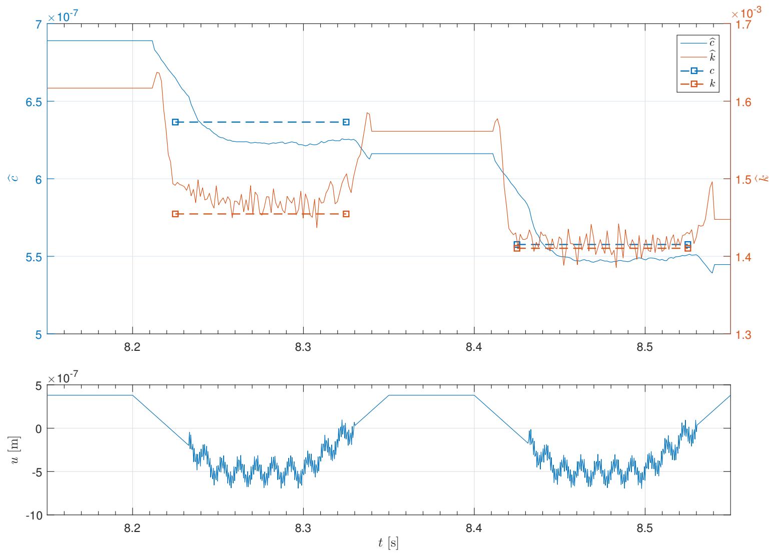

Figure 7.8: The estimates of $\widehat{c}$ and $\widehat{k}$ using the gradient method, and the cantilever position input u over time, with gain matrix equal to job 1 in table 7.13. The oscillations on the estimated spring constant, $\widehat{k}$.

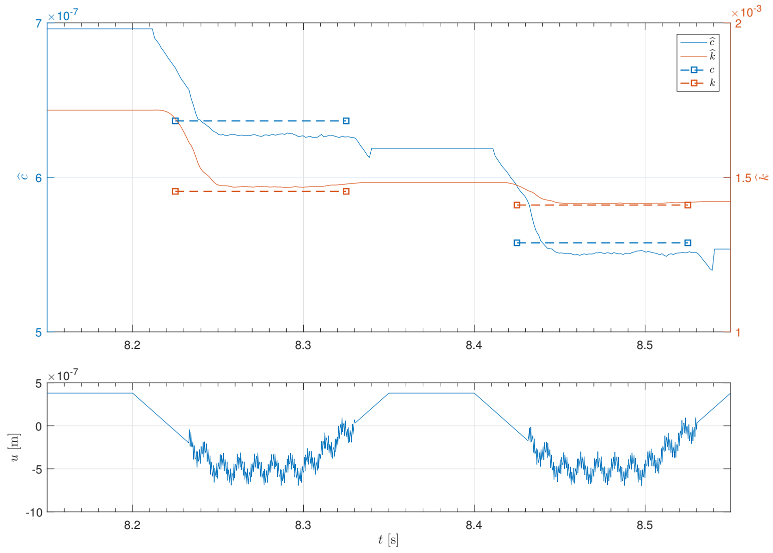

Figure 7.9: The estimates of $\widehat{c}$ and $\widehat{k}$ using the gradient method, and the cantilever position input u over time, with gain matrix equal to job 1 in table 7.13. The estimate of $\widehat{k}$ does not have oscillations.

The corresponding figures to table 7.11 and table 7.13 are found in appendix C5 and appendix C6.

7.3.4 Optimized Gain Matrix and Changing the Tunable Positive Filter Constant

The optimized gain matrix is found (equation 7.2), which means that the results can be improved by changing the tunable positive filter constant, τ. The filter is given as

$$\frac{1}{\Lambda(s)} = \frac{1}{\tau^2 s^2 + 2 \cdot \tau s + 1} \tag{7.3}$$

where τ is the tunable positive filter constant. In equation 7.3, the term in the denominator is a transfer function (section 2.4). Studying this term

$$\Lambda(s) = \tau^2 s^2 + 2 \cdot \tau s + 1 \tag{7.4}$$

it is found that τ is equal to $1/\omega_0$ in equation 2.41 and K, ξ is equal to 1. Changing τ will effect the properties of the system. The same will happen if $T0$ or $f0$ is changed, since τ is defined from them. The simulation setup is equal to the parameters in table 7.4. These are given in table 7.15 as experiment 6 and consist of five jobs.

The results from experiment 6 is given in table 7.16 as relative tolerance and rate of convergence. The corresponding figures to table 7.15 is found in appendix C8.

Experiment 7		
Job	Γ (gain matrix)	τ
1	$1e-0 \cdot diag([1e8 \quad 6e14])$	$0.05 \cdot T0$
2	$1e-0 \cdot diag([1e8 \quad 6e14])$	$0.10 \cdot T0$
3	$1e-0 \cdot diag([1e8 \quad 6e14])$	$0.15 \cdot T0$
4	$1e-0 \cdot diag([1e8 \quad 6e14])$	$0.17 \cdot T0$
5	$1e-0 \cdot diag([1e8 \quad 6e14])$	$0.20 \cdot T0$

Table 7.15: Experiment 6: Best gain matrix with different tunable positive filter constant, τ.

	Relative tolerance (%)				**Rate of convergence ([s])**			
Job	RT c1	RT c2	RT k1	RT k2	RoC c1	RoC c2	RoC k1	RoC k2
1	1.49	1.18	-1.10	-0.43	0.0013	0.0059	0.016	5.78e-04
2	1.50	1.19	-1.10	-0.43	0.0013	0.0059	0.016	5.78e-04
3	1.15	0.68	-1.76	-0.58	0.0093	0.014	0.032	5.78e-04
4	1.16	0.69	-1.76	-0.58	0.0093	0.014	0.032	5.78e-04
5	1.16	0.69	-1.76	-0.58	0.0093	0.014	0.032	5.78e-04

Table 7.16: The results corresponding to experiment 6 (table 7.15).

7.3.5 Optimized Gain Matrix with Different Simulation Solvers

After finding the optimized gain matrix, the properties of the system might be changed. This can be discovered by trying different simulation solvers. The simulation setup is equal to the parameters in table 7.1, except the gain matrix and the simulation solver. These are given in table 7.17 as experiment 7 and consist of six jobs.

Experiment 7		
Job	Γ (gain matrix)	**Simulation solver**
1	$1e-0 \cdot diag([4e8 \quad 10e14])$	$ode45$
2	$1e-0 \cdot diag([4e8 \quad 10e14])$	$ode15s$
3	$1e-0 \cdot diag([4e8 \quad 10e14])$	$ode23$
4	$1e-0 \cdot diag([4e8 \quad 10e14])$	$ode23s$
5	$1e-0 \cdot diag([4e8 \quad 10e14])$	$ode23t$
6	$1e-0 \cdot diag([4e8 \quad 10e14])$	$ode113$

Table 7.17: Experiment 6: Best gain matrix with different simulation solvers.

The results from experiment 7 is given as relative tolerance and rate of convergence in table 7.18. The figures corresponding to table 7.17 is found in appendix C7.

	Relative tolerance (%)				**Rate of convergence ([s])**			
Job	RT c1	RT c2	RT k1	RT k2	RoC c1	RoC c2	RoC k1	RoC k2
1	1.50	1.19	-1.10	-0.43	0.0013	0.0059	0.016	5.78e-04
2	1.48	1.21	-1.10	-0.43	0.0013	0.0019	0.016	5.78e-04
3	1.51	1.21	-1.10	-0.43	0.0013	5.78e-4	0.016	5.78e-04
4	1.56	1.25	-1.10	-0.43	0.0013	5.78e-4	0.016	5.78e-04
5	1.44	1.14	-1.09	-0.43	0.0013	5.78e-4	0.016	5.78e-04
6	1.51	1.21	-1.10	-0.42	0.0013	5.78e-4	0.016	5.78e-04

Table 7.18: The results corresponding to experiment 7 (table 7.17).

Chapter 8

Discussion

8.1 Cantilever Dynamics

By using equation 4.29 and Laplace transform of derivatives (Kreyszig, 2010), it was possible to find an equation the described the system in a linear-in-parameter form. There are three unknown parameters, K, C and M, that need to be estimated, while others had to be defined. The total force acting from the sample on the tip, the cantilever spring constant and the filter were found. The system was written into a linear-in-parameter form, which means that there are multiple parameter estimation method that can be chosen to solve this problem.

8.2 Continuing Work From Final Year Assignment

8.2.1 Tuning the Gain Matrix

Figures 7.4-7.6 shows changes in the estimated damper constant, $\widehat{c}$. The estimated damper constant oscillates around the real damper constant, c (see figure 7.4). When the gain on the damper constant is decreases, the oscillation on the estimated damper constant becomes smoother. In (Ioannou and Sun, 2012), an increase in the gain matrix will make the system faster, but it will create more noise since the noise is multiplied with the gain matrix. In figure 7.4, the oscillations is seen as noise in the system. Also, the rate of convergence decreases when the gain on the damper constant decreases. On the other hand, the difference between the estimated spring constant, $\widehat{k}$, in Figure 7.4 and Figure 7.5 is large. The limits on the y-axis in Figure 7.4 is small compared to the limits in Figures 7.5-7.6. By increasing the gain on the estimated spring constant improves as it almost reaches the real value of c.

8.2.2 Comparing the Two On-Line Parameter Estimation Methods

Comparing the Least-Square and Gradient Method

The relative tolerance and the rate of convergence given in table 8.1 shows that the there are a difference between the least-square method with forgetting factor and the gradient method using instantaneous cost. There are small deviations in the relative tolerance varying from 0.15% to 0.66%. The deviation in the rate of convergence varies from $0s$ to $0.016s$, which corresponds to an increase of 16% in the rate of convergence (subsection 5.2.3).

The difference between the two methods lies in the adaptive laws, since the systems have the same form (see equations 4.37-4.40). The adaptive laws for the two methods is repeated below, first the gradient method,

$$\dot{\theta} = \Gamma\epsilon\phi$$

then the least-square method

$$\begin{aligned}\dot{\theta} &= P\epsilon\phi \\ \dot{P} &= \beta P - P\frac{\phi\phi^T}{m^2}P, \quad P(0) = P_0\end{aligned}$$

The least-square method performs an extra computation for each time state opposed to the gradient method. The extra computation changes the least-square algorithm from a nonrecursive to a recursive one (Ioannou and Sun, 2012). The convergence properties of the gradient method and least-square method is the same (Ioannou and Sun, 2012), which means that the results shall in theory be equal. However, they are not equal (table 8.1). Deviations on the rate of convergence have more impact on the system than the relative tolerance, which can be seen in figure 7.6 and figure 7.7. This means that the gradient method has the best performance between the two parameter estimation methods.

Least-square with Two Different Γ

Table 8.2 shows that the deviation between the least-square method with Γ given in equation 7.1 and the least-square method with Γ equal to equation 7.1. Only the relative tolerance is taking into account, because the rate of convergence is the same for both Γs when studying the results in table 7.3. The deviation between the to results is small considering that the results are given in percent. The largest deviation is 0.0045%.

In (Ragazzon et al., 2016), Γ represented the initial value of P, P_0, found in the adaptive law for the least-square method (see equations 4.26-4.27). This means that the two results are with different initial values for P (equation 4.26). Since the maximum deviation is 0.0045%, the two results can be said to be equal. That means the initial value, P_0, does not affect the end results in this case.

	Relative tolerance (%)			
Figure	RT c1	RT c2	RT k1	RT k2
7.6	1.5066	1.1934	-1.1026	-0.4330
7.7	1.1429	0.6750	-1.7646	-0.5788
Deviation	0.3637	0.5184	0.6620	0.1458
	Rate of convergence ([s])			
Figure	RoC c1	RoC c2	RoC k1	RoC k2
7.6	0.0013	5.7839e-04	0.016	5.7839e-04
7.7	0.0093	0.014	0.032	5.7839e-04
Deviation	0.0080	0.0134	0.0160	0

Table 8.1: Absolute value of the deviation in the relative tolerance and the rate of convergence between the results from Figure 7.1 and Figure 7.7.

	Relative tolerance (%)			
Figure	RT c1	RT c2	RT k1	RT k2
7.1	1.1425	0.6745	-1.7652	-0.5833
7.7	1.1429	0.6750	-1.7646	-0.5788
Deviation	0.0004	0.0005	0.0006	0.0045

Table 8.2: Absolute value of the deviation in the relative tolerance between the results from Figure 7.6 and Figure 7.7.

8.3 Parallel Computing

When studying the results, there are two analysing tools used, relative tolerance and rate of convergence (subsection 5.2). Each of these produce four results, eight in total. These result corresponds to the relative tolerance and the rate of convergence defined in subsection 5.2.3. The relative tolerance between the real and estimated values for the damper constant and the spring constant, and the rate of convergence for the estimated values (damper and spring constant) towards the real values. The relative tolerance is given with four decimals in appendix A for more accuracy.

8.3.1 Changing the Simulation Solver

Simulation solvers have mathematical differences (see section 2.2), which means that they effect the results. The produced results are given as relative tolerance and rate of convergence in table 7.6. Studying these results, there is found that the there are small variations between the solvers. The largest variation in the relative tolerance is 0.1269%. Although, the rate of convergence varies for two solvers ($ode23s$ and $ode23t$) in the first value for the damper constant ($RoC\ c1$). The remaining solvers have equal rate of convergence. This makes a considerable difference in the results, because the rate of convergence increases with $0.0107s$ ($ode23s$) and $0.0133s$ ($ode23t$). The two results correspond to an increase in rate of convergence of 10.7% and 13.3%. If comparing the two analysing tools, small changes in the rate of convergence have more impact on the system than the relative tolerance. The two solvers produces an increase of minimum 10.7% in the rate of convergence are stiff solvers (see subsection 2.2.2): $ode23s$ and $ode23t$ (Egeland and Gravdahl, 2002; Shampine and Reichelt, 1997). The remaining solvers ($ode45$, $ode15s$, $ode23$, $ode113$) produce similar results with a maximum variation of 0.1269% in the relative tolerance.

8.3.2 Combining Changes

Simulation Solver and Relative Tolerance

The relative tolerance in the simulation solver has an impact on the simulation (section 2.2), because it determines the error relative to the size of each state (MatlabWorks, 2016a). Therefore, when increasing the relative tolerance in the simulation solver from $1.00e-09$ to $1.00e-07$ the produced results can change. By allowing the error relative to the size of each state to increase, the simulation time decreases and it might affect the system. The results is given in relative tolerance and rate of convergence (see subsection 5.2) and are found in table 7.8.

There are variations when studying the relative tolerance in experiment 2 (table 7.7). For $RT\ c1$ (relative tolerance), the largest variation between the solvers is 0.127%. For $RT\ c2$, it is 0.1067% (see subsection 5.2.3). For $RT\ k1$, it is 0.0123%, while it is 0.0046% for $RT\ k2$. The variations in relative tolerance between the simulation solvers are small, greatest at 0.1067%. When looking at the rate of convergence, the results are equal except in two solvers ($ode23s$ and $ode23t$). The results for $ode23s$ is job 4 in experiment 2, and for $ode23t$ it is job 5 in experiment 2. In both cases the rate of convergence is equal to the other solvers for $RoC\ c2$ (rate of convergence) and $RoC\ k2$. The rate of convergence increases to $0.0146s$ for $RoC\ c1$ and decreases to $0.013s$ for $RoC\ k1$ in $ode23s$. Compared to the other solvers at these points, $ode23s$ takes $0.0133s$ longer time to converge in $RoC\ c1$ and $0.003s$ shorter time to converge. These numbers correspond to an increase of 13.3% and a decrease of 3% in the rate of convergence. $ode23t$ has the equal rate of convergence as the other simulation solvers, except for $RoC\ c1$, which is equal to $0.012s$ instead of $0.0013s$ as the others. This means that it has $0.0107s$ longer convergence time, which correspond to 10.7% (equation 5.1).

Comparing the results from experiment 1 (table 7.6) and experiment 2 (table 7.8), there is a small deviation between the same simulation solvers. The largest deviation is 0.0157% in the relative tolerance, while the rate of convergence is equal except one result in $RoC\ k1$ for $ode23s$ in experiment 2. This value is $0.003s$ lower than the same value in experiment 1.

When changing the simulation solver and the relative tolerance in the solver, the stiff solvers, $ode23s$ and $ode23t$ have an unsatisfactory performance on the system. This is due to the fact that the rate of convergence increases with minimum 10%. The remaining solvers ($ode45$, $ode15s$, $ode23$, $ode113$) have a desirable performance with an estimation leading to low relative tolerance and rate of convergence.

Based on experiments, results show that the relative tolerance in the simulation solvers should have been set to the default value first, $1e-03$. This would yield a better starting point, before testing what value that worked best with the solvers. Another point is that not only should the relative tolerance in the solvers be changed, but also the absolute tolerance. The reason for this is that the relative and absolute tolerance determine the acceptable error. The two tolerances defines the acceptable error as

$$e_i \leq \max(rtol \times |x_i|, atol_i)$$

where $rtol$ is the relative tolerance, x_i is the ith state and $atol_i$ is absolute tolerance in the ith state. The absolute tolerance in the solvers is defined by (MatlabWorks, 2016a) as:

> Absolute tolerance is a threshold error value. This tolerance represents the acceptable error as the value of the measured state approaches zero. The default value for the absolute tolerance is $1e-06$.

In this thesis, only the relative tolerance was changed, even though the relative tolerance and the absolute tolerance depend on each other.

Simulation Solver and Tap Period

In experiment 3, job 4 ($ode23s$) have the best relative tolerance (table 7.10) for the estimated values. The rate of convergence shows that job 4 ($ode23s$) and job 5 ($ode23t$) have the best performance (table 7.10). Since $ode23s$ have the best performance for both the relative tolerance and the rate of convergence. It is the best solver for this case where the tap period is increased by $0.005s$. The rest of the solvers have quite similar performance. Although, $ode113$ (job 6), have a large increase in the relative tolerances and the most varying rate of convergence.

By comparing experiment 3 and experiment 1, there are small deviation in the relative tolerance between each solver, except for $ode113$. In percentage points,
The rate of convergence for $ode113$ increases in three out of four cases. When studying

	RT c1	RT c2	RT k1	RT k2
Percentage Point	0.3647%	0.2952%	2.5575%	0.5344%

Table 8.3: The relative tolerance given in percentage points between experiment 3 and experiment 1 for $ode113$.

the results in table 7.10, the two best performances are two of the stiff solvers ($ode23s$ and $ode23t$) (subsection 2.2.2).

Changing the tap period to 0.205 and then change to simulation solvers was one way to execute experiment 3. However, if the tap period had been varied around 0.2 (see simulation setup, table 7.4), it would be possible to analyse the system in another way.

8.3.3 Optimizing the Gain Matrix

In experiment 4 (table 7.11) and experiment 5 (table 7.13), a special case occurs when analysing at the results. In both, only one of the diagonal elements in the gain matrix were changed. There is a clear change in the relative tolerance in the result. When changing the value in either direction, the relative tolerance increases.

Changing the Gain on the Damper Constant

When looking at the relative tolerances, $RT\ c1$ and $RT\ c2$, in table 7.12 the smallest relative tolerance is found in job 3. On either side of this job, the relative tolerance increases, which means that there exists a *local minimizer*. In (Nocedal and Wright, 2006), a local minimizer is described as:

> A point x^* is a local minimizer if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{N}$.

where x^* in this case is the gain matrix, Γ.

The rate of convergence increases as the 1^{st} element in the gain matrix decreases. However, if the 1^{st} element increases, the rate of convergence decreases. This can be seen in $RoC\ c1$ and $RoC\ c2$ in table 7.12 when comparing to the gain matrices (Γ) in table 7.11.

Changing the Gain on the Spring Constant

When changing the gain on the spring constant, the same phenomenon occurs as when changing the gain on the damper constant. By looking at the relative tolerances $RT\ k1$ and $RT\ k2$ in table 7.14, job 4 had the lowest relative tolerances. On either side of this job the relative tolerances increased. This means that there exist a local minimizer for the spring constant as well. The spring constant varies less when changing the gain on the spring opposed to changing the gain on the damper constant. There are smaller changes between the results than for the damper constant. However, the relative tolerance decreases for the damper constant from job 5 to job 12. Whereas, the relative tolerance for the spring constant increases at these jobs. The percentage points for the relative tolerance for the damper constant is given in table 8.4. Negative percentage points means that there are decreasing, while positive means that there are an increasing between job 4 and jobs 5-12 (the average, equation 2.3.1).

The rate of convergence increases from job 4 to job 5, except for $RoC\ k2$ which is equal. However, the rate of convergence is equal for jobs 5-12. This means that there is an increase in $RoC\ c1$, $RoC\ c2$ and $RoC\ k1$ between there jobs. For $RoC\ c1$, the rate of convergence increased with $0.008s$, which correspond to 8% slower rate of convergence. For $RoC\ c2$ and $RoC\ k1$, the rate of convergence increased with $0.0081s$ and $0.016s$,

	RT c1	RT c2	RT k1	RT k2
Percentage Point	-0.36%	-0.54%	0.66%	0.42%

Table 8.4: The relative tolerance given in percentage points between job 4 and jobs 5-12 in experiment 5.

corresponding to 8.1% and 16% (see subsection 5.2.3 and equation 5.1 for these calculations). Even though there were small changes in the relative tolerance, the percentage increase from job 5 to job 12 indicate that job 4 produces the best result.

In (Ioannou and Sun, 2012), it is described that the rate of convergence decreases when the gain matrix increases, and vice versa. When studying the rate of convergence in table 7.12 and table 7.14 and compare them to the gain matrices in experiment 4 (table 7.11) and experiment 5 (table 7.13). The phenomenon described in (Ioannou and Sun, 2012) appear in experiment 4 and experiment 5.

As described above, when the two elements in the gain matrix are increased, the rate of convergence will decrease (Ioannou and Sun, 2012). However, the gain matrix is multiplied with the noise in the system, which means that the noise will increase (Ioannou and Sun, 2012) when increasing the gain elements. The noise is seen as oscillations in the estimated parameters around the real values. This can be seen in figure C.44 by comparing it to figure C.47. There are few variations in relative tolerance and rate of convergence between the two figures as seen in table 7.14. The small variations in the relative tolerance from experiment 6, an extra job was added. This was done to conclude that it actually existed a local minimizer. However, this was not done in experiment 5, because the variations were clearer. Therefore, it is not possible to see the same oscillations in the figures from experiment 4 (appendix C5).

Local Minimizer

There exist a local minimizer when changing both parameters in the gain matrix. Although, the definition of a local minimizer given above can sometimes imply that it is a *weak local minimizer*. This means that one cannot know if it is an outright winner in its neighborhood. When a point a the outright winner it is called a *strict local minimizer*, which is defined in (Nocedal and Wright, 2006) as:

> A point x^* is a strict local minimizer (also called strong local minimizer) if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) < f(x)$ for all $x \in \mathcal{N}$ with $x \neq x^*$.

It is not possible to call the gain matrix given in equation 7.2 for a strict local minimizer, because there have not been executed enough tests to conclude this. There is extremely small numbers in the results given in section 7.3. Therefore, it is possible that there exist another combination with the two diagonal elements in the gain matrix that gives the same results as in equation 7.2. Since it is not possible to conclude that the gain matrix found is a strong local minimizer, it implies that it cannot be a *global minimizer* either. A global minimizer is defined in (Nocedal and Wright, 2006) as:

A point x^* is a global minimizer if $f(x^*) \leq f(x)$ for all x,

where x ranges over all of $\mathbb{R}^n$. The reason for not being able to say it can be global is that we only have knowledge of the behavior of the system at a small part, which in this case is local knowledge.

Optimized Gain Matrix and Changing the Tunable Positive Filter Constant

The tunable positive filter constant, τ, was varied in experiment 6 (table 7.15), where job 2 is the same as in the simulation setup (table 7.4). The results are given in relative tolerance and rate of convergence (table7.16). Analysing the results, there are a clear distinction in both relative tolerance and rate of convergence between job 2 and job 3. The relative tolerance is studied in detail before the rate of convergence.

Job 1-2 have a maximum variation at 0.0112% (RT $c2$) and a minimum at 0.0003% (RT $k2$), and is therefore assumed equal. The mean (subsection 2.3.1) between the different relative tolerances are taken to use as comparison to the mean between job 3-5. The mean of jobs 1-2 are given in table 8.5. The maximum variation between job 3-5 is 0.008% (RT $c1$), while the minimum is 0.0003% (RT $c1$,$c2$). As for job1 2, the mean between job 3-5 is found and given in table 8.6. From the new relative tolerances (RT) found above, there is found a decrease in RT $c1$ (0.339%) and RT $c2$ (0.5005%), and an increase in RT $k1$ (0.6634%) and RT $k2$ (0.1448%). This means that the deviation in the relative tolerance for experiment 6 have little impact on which τ works best for the system performance.

	RT c1	RT c2	RT k1	RT k2
Mean	1.4950%	1.1883%	-1.1008%	-0.4333%

Table 8.5: The mean given in percentage is job 1-2 in experiment 6.

	RT c1	RT c2	RT k1	RT k2
Mean	1.1560%	0.6878%	-1.7642%	-0.5781%

Table 8.6: The mean given in percentage is job 1-2 in experiment 6.

When studying the rate of convergence, the jobs are separated into the same two groups as for the relative tolerance, job 1-2 and job 3-5. In the two groups separately, the rate of convergence is similar. However, for RT $k2$ the rate of convergence is equal for every job in experiment 7. There is an increase in RoC $c1$ ($0.008s$), RoC $c2$ ($0.0081s$) and RoC $k1$ ($0.016s$) from jobs 1-2 to jobs 3-5. This correspond an increase of 8%, 8.1% and 16% in rate of convergence (subsection 5.2.3).

The increase of the tunable positive filter constant (τ) makes the performance of the system decrease. This can be seen in the rate of convergence for these jobs (3-5). The best results are when τ is less or equal to $0.1 \cdot T0$, where $T0$ is defined in table 7.4. The reason for this is found when studying the transfer function in equation 7.4. In (Balchen et al., 2003), an increase in ω_0 will resolve in a quicker system, while decreasing ω_0 will make the system slower. ω_0 increases as τ decreases, which shows in the rate of convergence in table 7.16. Because of the small differences in the relative tolerance in job 1-2, there is not possible to decide which one have the best performance. However, Both the relative tolerance and the rate of convergence decreases from job 1 to job 2. Transfer function theory found in (Balchen et al., 2003), states that continuing the decrease in τ will resolve in a increase in ω_0 which is equal to $1/\tau$ in the transfer function 2.41.

τ is described as

$$\tau = 0.1 \cdot T0 = 0.1 \cdot \frac{1}{f0} \tag{8.1}$$

This means that the properties of the system will not only change when changing τ, but also when changing $T0$ and $f0$. When $T0$ and $f0$ is given as in the simulation setup (table 7.4), a decrease in $f0$ will make $T0$ increase and τ increase, while ω_0 decrease. If $f0$ increases, $T0$ and τ decreases, while ω_0 increases.

Optimized Gain Matrix with Different Simulation Solvers

The system properties might have changed and therefore it can have changed from a nonstiff to a stiff system. This is easily checked when trying different simulation solvers on the system. This resulted in experiment 7 (table 7.17) with the corresponding relative tolerance and rate of convergence found in table 7.18. When studying the results for the relative tolerance, it is discovered that there are small variations here. Because of earlier findings on the importance in rate of convergence, it is studied in detail before the relative tolerance.

Studying the rate of convergences in experiment 7, all results are equal except one results from $RoC\ c2$. Job 1 ($0.0059s$) and job 2 ($0.0019s$) have a higher rate of convergence than the rest of the jobs ($5.78e-4s$. In terms of rate of convergence, $ode45$ (job 1) have the least satisfactory performance before $ode15s$ (job 2). $ode23$, $ode23s$, $ode23t$ and $ode113$ have the best performance in terms of rate of convergence.

The small deviations between the relative tolerances in experiment 7, makes it difficult to conclude which of the solvers that have the best performance. However, because of the rate of convergence $ode45$ and $ode15s$ can be excluded. The largest deviations in the relative tolerances are

$$\text{RT c1} = 0.1253\%, \quad \text{RT c2} = 0.1070\%, \quad \text{RT k1} = 0.0108\%, \quad \text{RT k2} = 0.0115\%.$$

Non of the remaining solvers stand out with best performance, because they all vary. However, a special case is that there are both nonstiff and stiff solvers remaining.

8.4 Working With Vilje

Vilje was a big part of this thesis. The problems described in subsection 5.1.3 made the progress slow in the beginning. However, when it was possible to run multiple simulations at the same time the usage of Vilje became positive. The reason for this is that it is possible to run simulations at night, which means that there are multiple results to the next day.

Another point is that now that version R2016b is installed on Vilje, it is possible to use the Matlab command *parpool* on Vilje.

Chapter 9

Conclusion

In this chapter we will conclude this thesis. We have described and implemented a set of analysing tools that were able to analyse the simulation results given when running simulations in parallel. Also, a set of equations for were derived for the cantilever dynamics and given in linear-in-parameter form.

9.1 Achievements

This thesis is based on the problem descriptions

- Continuing the work from (Leer, 2016). Finding a gain matrix for the gradient method which provides a good estimate.
- Learning to use Vilje and run a single simulation before trying to run multiple simulations in parallel.
- Investigate the cantilever dynamics and discovering if it is possible to use an adaptive parameter estimator to estimate the damper and spring constant in the cantilever.

This thesis has focused on the use of parameter estimation on a mathematical model of a biological cell and AFM as a base in the simulations. We have reviewed different parameters that effect the properties of the model, simulation solvers and a set of analysing tools. Based on the findings, the chosen analysing tools are described and implemented in section 5.2. The parameters chosen to be tuned are found in table 7.1 and the simulation methods are described in section 2.2. In addition, the cantilever dynamics of the AFM was reviewed and studied. The findings are found in section 3.2 and chapter 6.

9.1.1 Continuing Work From Final Year Assignment

Multiple experiments were conducted with the gradient method using instantaneous cost as parameter estimator. First, we continued the work from (Leer, 2016), where the gain matrix was tuned. The results gave a gain matrix that made the estimated parameters converge to the real values.

Then, the new gain matrix was used to see if there was a deviation in the results between two simulations with the least-square method, but with different initial value, P_0 (=gain matrix). There were no deviation between the rate of convergence, but there were maximum deviation of 0.0045% in the relative tolerances.

Concluding Remarks

By successfully finding a gain matrix that made the estimated parameters converge to the real values, and comparing this result with the result using the least-square method found in the base model (section 4.3). The theory stated in subsection 4.1 and subsection 4.2, gives the two parameter estimation methods the same convergence properties. However, the gradient method gave a faster rate of convergence than the least-square method.

The simulations of least-square with different initial value can be assumed equal, which means that the initial value did not effect the end result in this case.

9.1.2 Experiments on Vilje

Using Vilje and parallel computing (section 5.1), multiple simulations ran simultaneously. This open the opportunity for optimization of the gain matrix and other parameters (table 7.4), while determining if the system is nonstiff or stiff (section 2.2). In addition to determine if it was useful running parallel computing with Vilje.

In experiment 1, the solvers produced similar results, except for two stiff solvers (see subsection 2.2.2), $ode23s$ and $ode23t$. The remaining solvers are three nonstiff solvers ($ode45$,$ode23$,$ode113$) and one stiff solver ($ode15s$). Each of the remaining solvers have variations in the relative tolerance. This means that there is not one specific solver that produced the best result.

The results from experiment 2, shows small deviation between the solvers in relative tolerance. The largest deviation is at 0.1067%. However, when analysing the rate of convergence, job 4 ($ode23s$) and job 5 ($ode23t$) varies from the other solvers. There is an increase in $RoC\ c1$ of 13.3% and a decrease in $RoC\ k1$ of 3% for $ode23s$, while there is a increase in rate of convergence of RoC 10.7% for $ode23t$.

Experiment 3 changes the rate of convergence when comparing to experiment 1. The nonstiff solvers have become slower, while the stiff solvers have a decrease in the rate of convergence.

The optimize gain matrix was found (equation 7.2) by changing each diagonal element separately, and two local minimizers (see subsection 8.3.3) were discovered. However, the second depended on the first.

By finding the optimal gain matrix, it was possible to try and optimize the other parameters in the simulation setup (table 7.4). The filter is a transfer function with properties defined in section 2.4. This indications that by decreasing the tunable positive filter constant, τ, the system will become faster. Experiment 6 was about optimizing the system even more by decreasing τ.

After changing parameters from the simulation setup (table 7.4), a change in the properties of the system can occur. Early in the simulations, the simulation solver was changed to check if the system was nonstiff or stiff. Therefore, experiment 7 consisted of running the optimized parameter found earlier with different simulation solvers.

Concluding Remarks

All the remaining solvers in experiment 1 are nonstiff and one is stiff solver which performs well with nonstiff problems, the system with the given parameters can be said to be nonstiff (see subsection 2.2.2).

Experiment 2 shows that the system performs best when nonstiff solvers are used (see subsection 2.2.2). This means that the system is nonstiff for these properties. Also, if comparing experiment 1 to experiment 2 the results are almost identical. The largest deviation in relative tolerance is at 0.0157%, while the rate of convergence is equal for all except that $RoC\ k1$ for $ode23s$ which converges $0.003s$ faster in experiment 2.

Changing the tap period in experiment 3, changed the system behavior. With these parameters, the system have become stiff (subsection 2.2.2).

Experiment 4 and experiment 5 found two local minimizers. The number of experiments made it impossible to conclude if the second local minimizer (equation 7.2) was a strict local minimizer or a global local minimizer.

Experiment 6, showed that the tunable filter constant, τ, decides the properties of the system. This is because τ is equal to ω_0 in equation 2.41 (section 2.4).

No conclusion could be drawn from experiment 7, since the results were varying and consisted of both stiff and nonstiff solvers.

9.1.3 Cantilever Dynamics

The unknown parameters in the cantilever dynamics described in the linear-in-parameter form (equation 6.13) can be estimated using an adaptive parameter estimation method. Before the system can be implemented, an input signal that is PE need to be found.

9.2 Using Vilje

Though we encountered problems using Vilje (subsection 5.1.3) in the beginning, it was rewarding in the later stages of this thesis when everything worked. By using Vilje, we were able to run more simulations than if not.

9.3 Future Work

In this section, issues for future work is presented.

- Continue to optimize the system with changing the parameters in the simulation setup (table 7.4) or other parameters that could effect the properties of the system. The relative tolerance could be set to default value and changed from there along with changing the absolute tolerance. The same goes for changing the tap period, and T_0 and f_0 in τ.
- Make the system into an optimization problem. When trying to optimize the system, it was difficult to know how the parameters should be changed. First, finding which optimization algorithm could be used. Second, trying to optimize one parameter before increasing the number.
- Find the input signal that is PE will for the cantilever dynamics. Implement the model and simulate.
- Try to remove the bias by adding another estimation parameter. This is the indention depth, and by adding this the system changes into a bilinear model. This will make the estimator more complex.

Bibliography

Abramovitch, D., Andersson, S., Pao, L., Schitter, G., 2007. A tutorial on the mechanisms, dynamics, and control of atomic force microscopes. In: Proceedings of the 2007 American Control Conference. New York City, USA, pp. 3488–3502.

Ashino, R., Nagase, M., Vaillancourt, R., 2000. Behind and beyond the matlab ode suite. Computers & Mathematics with Applications 40 (4), 491–512.

Balchen, J. G., Fjeld, M., Solheim, O. A., 2003. Reguleringsteknikk. Tapir.

Bao, G., Suresh, S., 2003. Cell and molecular mechanics of biological materials. Nature materials 2 (11), 715–725.

Barney, B., et al., 2010. Introduction to parallel computing. Lawrence Livermore National Laboratory 6 (13), 10.

Binnig, G., Quate, C. F., Gerber, C., 1986. Atomic force microscope. Physical Review Letters 56 (9), 930–933.

Butt, H.-J., Cappella, B., Kappl, M., 2005. Force measurements with the atomic force microscope: Technique, interpretation and applications. Surface science reports 59 (1), 1–152.

Campbell, N. A., Reece, J. B., Urry, L. A., Cain, M. L., Wasserman, S. A., Minorsky, P. V., Jackson, R. B., 2015. Biology - A Global Approach, 10th Edition. Pearson.

Cappella, B., Dietler, G., 1999. Force-distance curves by atomic force microscopy. Surface science reports 34 (1), 1–104.

Egeland, O., Gravdahl, J. T., 2002. Modeling and simulation for automatic control. Vol. 76. Marine Cybernetics Trondheim, Norway.

Grenoble, J. F. U., 1999. ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb. URL http://www.obs.ujf-grenoble.fr/scci/logiciels/matlab61/help/techdoc/ref/ode23.html

Haase, K., Pelling, A. E., 2015. Investigating cell mechanics with atomic force microscopy. Journal of The Royal Society Interface 12 (104), 20140970.

Hutter, J. L., Bechhoefer, J., 1993. Calibration of atomic-force microscope tips. Review of Scientific Instruments 64 (7), 1868–1873.

Ioannou, P. A., Sun, J., 2012. Robust Adaptive Control. Dover Publications, Inc.

Iversen, K., 2015. Biological cell models and atomic force microscopy: A literature review. Master's thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics.

Kreyszig, E., 2010. Advanced engineering mathematics. John Wiley & Sons.

Kuznetsova, T. G., Starodubtseva, M. N., Yegorenkov, N. I., Chizhik, S. A., Zhdanov, R. I., 2007. Atomic force microscopy probing of cell elasticity. Micron 38 (8), 824–833.

Leer, K. B., May 2016. Biological cell models and atomic force microscopy: Parameter estimation. Final year project assignment, Norwegian University of Science and Technology.

MatlabWorks, 2016a. Choose a solver.
URL `https://se.mathworks.com/help/simulink/ug/types-of-solvers.html`

MatlabWorks, 2016b. Choose an ode solver.
URL `https://se.mathworks.com/help/matlab/math/choose-an-ode-solver.html`

Nocedal, J., Wright, S., 2006. Numerical optimization. Springer Science & Business Media.

NOTUR, 2016. About vilje.
URL `https://www.hpc.ntnu.no/display/hpc/About+Vilje`

Ragazzon, M. R., Gravdahl, J. T., 2016. Imaging topography and viscoelastic properties by constant depth atomic force microscopy. In: Control Applications (CCA), 2016 IEEE Conference on. IEEE, pp. 923–928.

Ragazzon, M. R., Gravdahl, J. T., Pettersen, K. Y., Eielsen, A. A., 2015. Topography and force imaging in atomic force microscopy by state and parameter estimation. In: 2015 American Control Conference (ACC). IEEE, pp. 3496–3502.

Ragazzon, M. R., Vagia, M., Gravdahl, J. T., 2016. Cell mechanics modeling and identification by atomic force microscopy. IFAC-PapersOnLine 49 (21), 603–610.

Sader, J. E., Chon, J. W., Mulvaney, P., 1999. Calibration of rectangular atomic force microscope cantilevers. Review of Scientific Instruments 70 (10), 3967–3969.

Sader, J. E., Larson, I., Mulvaney, P., White, L. R., 1995. Method for the calibration of atomic force microscope cantilevers. Review of Scientific Instruments 66 (7), 3789–3798.

Shampine, L., 1973. Local extrapolation in the solution of ordinary differential equations. Mathematics of Computation 27 (121), 91–97.

Shampine, L. F., Gordon, M. K., 1975. Computer solution of ordinary differential equations.

Shampine, L. F., Reichelt, M. W., 1997. The matlab ode suite. SIAM journal on scientific computing 18 (1), 1–22.

Sokolov, I., 2007. Atomic force microscopy in cancer cell research. Cancer Nanotechnology, 1–17.

Suresh, S., 2007. Biomechanics and biophysics of cancer cells. Acta Materialia 55 (12), 3989–4014.

Tortonese, M., 1997. Cantilevers and tips for atomic force microscopy. IEEE engineering in medicine and biology magazine 16 (2), 28–33.

Walpole, R. E., Myers, R. H., Myers, S. L., Ye, K., 1993. Probability and statistics for engineers and scientists. Vol. 5. Macmillan New York.

Wilson, R., Bullen, H., n.d. Basic theory - atomic force microscopy, note on AFM from Northern Kentucky University. Accessed online 15-March-2016.

Appendix A

Tables

Continuing changing Γ from Final Year Assignment

Test	Γ	**Short summary**
1	$1e-0 \cdot diag([10e8 \quad 10e8])$	$\widehat{c}$ good with some oscillation, can't see k
2	$1e-0 \cdot diag([10e9 \quad 10e8])$	$\widehat{c}$ okay with some oscillation, can't see k
3	$1e-0 \cdot diag([10e11 \quad 10e8])$	much oscillation on $\widehat{c}$, can't see k
4	$1e-0 \cdot diag([10e2 \quad 10e8])$	can't see k or c
5	$1e-0 \cdot diag([10e6 \quad 10e8])$	can't see k or c
6	$1e-0 \cdot diag([10e12 \quad 10e8])$	much oscillation on $\widehat{c}$, can't see k
7	$1e-0 \cdot diag([10e8 \quad 10e12])$	$\widehat{c}$ good with some oscillation, can't see k
8	$1e-0 \cdot diag([10e10 \quad 10e12])$	$\widehat{c}$ okay, can't see k
9	$1e-0 \cdot diag([10e8 \quad 10e13])$	$\widehat{k}$,$\widehat{c}$ good, some oscillation on $\widehat{c}$
10	$1e-0 \cdot diag([10e8 \quad 10e14])$	very good results on both $\widehat{k}$ and $\widehat{c}$
11	$1e-0 \cdot diag([10e9 \quad 10e14])$	good, but not as good as test 10
12	$1e-0 \cdot diag([10e7 \quad 10e14])$	okay, but a little slow
13	$1e-0 \cdot diag([10e6 \quad 10e14])$	$\widehat{k}$ okay, can't see c
14	$1e-0 \cdot diag([5e7 \quad 10e14])$	almost the same as test 12
15	$1e-0 \cdot diag([5e8 \quad 10e14])$	best result till now

Table A.1: An overview over how Γ was tuned.

Parallel Computing

Changing Simulation Solver

	Bias (real value - estimated value)			
Job	Bias c1	Bias c2	Bias k1	Bias k2
1	9.5901e-09	6.6547e-09	-1.6039e-05	-6.1070e-06
2	9.3571e-09	6.7283e-09	-1.6074e-05	-6.1072e-06
3	9.6480e-09	6.7333e-09	-1.5977e-05	-6.0498e-06
4	9.9482e-09	6.9445e-09	-1.6053e-05	-6.0934e-06
5	9.1403e-09	6.3480e-09	-1.5892e-05	-6.1229e-06
6	9.6598e-09	6.7528e-09	-1.5890e-05	-5.9592e-06

	Relative tolerance (%)			
Job	RT c1	RT c2	RT k1	RT k2
1	1.5066	1.1934	-1.1026	-0.4330
2	1.4700	1.2066	-1.1050	-0.4330
3	1.5157	1.2075	-1.0983	-0.4289
4	1.5628	1.2453	-1.1035	-0.4320
5	1.4359	1.1384	-1.0925	-0.4341
6	1.5175	1.2110	-1.0923	-0.4225

	Rate of convergence ([s])			
Job	RoC c1	RoC c2	RoC k1	RoC k2
1	0.0013	5.7839e-04	0.016	5.7839e-04
2	0.0013	5.7839e-04	0.016	5.7839e-04
3	0.0013	5.7839e-04	0.016	5.7839e-04
4	0.0146	5.7839e-04	0.016	5.7839e-04
5	0.0120	5.7839e-04	0.016	5.7839e-04
6	0.0013	5.7839e-04	0.016	5.7839e-04

Table A.2: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the test in table 7.5.

Combining Changes

Simulation Solver and Relative Tolerance

	Bias (real value - estimated value)			
Job	Bias c1	Bias c2	Bias k1	Bias k2
1	9.5901e-09	6.6547e-09	-1.6039e-05	-6.1070e-06
2	9.3095e-09	6.8160e-09	-1.6070e-05	-6.1100e-06
3	9.6451e-09	6.7320e-09	-1.6061e-05	-6.0577e-06
4	9.9488e-09	6.9431e-09	-1.6053e-05	-6.0936e-06
5	9.1403e-09	6.3480e-09	-1.5892e-05	-6.1219e-06
6	9.6692e-09	6.7458e-09	-1.5899e-05	-6.0623e-06
	Relative tolerance (%)			
Job	RT c1	RT c2	RT k1	RT k2
1	1.5066	1.1934	-1.1026	-0.4330
2	1.4625	1.2223	-1.1048	-0.4332
3	1.5152	1.2072	-1.1041	-0.4295
4	1.5629	1.2451	-1.1035	-0.4321
5	1.4359	1.1384	-1.0925	-0.4341
6	1.5190	1.2097	-1.0930	-0.4298
	Rate of convergence ([s])			
Job	RoC c1	RoC c2	RoC k1	RoC k2
1	0.0013	5.7839e-04	0.016	5.7839e-04
2	0.0013	5.7839e-04	0.016	5.7839e-04
3	0.0013	5.7839e-04	0.016	5.7839e-04
4	0.0146	5.7839e-04	0.013	5.7839e-04
5	0.0120	5.7839e-04	0.016	5.7839e-04
6	0.0013	5.7839e-04	0.016	5.7839e-04

Table A.3: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the test in table 7.7.

Simulation Solver and Tap Period

	Bias (real value - estimated value)			
Job	Bias c1	Bias c2	Bias k1	Bias k2
1	9.4568e-09	6.6448e-09	-1.5671e-05	-5.9512e-06
2	9.3732e-09	6.8711e-09	-1.5697e-05	-5.9443e-06
3	9.5528e-09	6.7315e-09	-1.5613e-05	-5.8972e-06
4	9.9651e-09	6.9834e-09	-1.5687e-05	-5.9401e-06
5	8.9448e-09	6.4597e-09	-1.5553e-05	-5.9601e-06
6	1.3578e-08	9.5880e-09	2.5863e-05	-1.3919e-05

	Relative tolerance (%)			
Job	RT c1	RT c2	RT k1	RT k2
1	1.4856	1.1916	-1.0773	-0.4220
2	1.4725	1.2322	-1.0791	-0.4215
3	1.5007	1.2072	-1.0733	-0.4181
4	1.5655	1.2523	-1.0784	-0.4212
5	1.4052	1.1584	-1.0692	-0.4226
6	1.8822	1.5062	-1.4652	-0.9569

	Rate of convergence ([s])			
Job	RoC c1	RoC c2	RoC k1	RoC k2
1	0.0093	0.0086	0.024	5.7839e-04
2	0.0039	0.0086	0.024	5.7839e-04
3	0.0013	0.0046	0.024	5.7839e-04
4	0.0013	5.7839e-04	0.024	5.7839e-04
5	0.0013	5.7839e-04	0.024	5.7839e-04
6	0.0240	5.7839e-04	0.0079	0.022

Table A.4: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the test in table 7.9.

Optimize the Gain Matrix

Damper Constant

	Bias (real value - estimated value)			
Job	Bias c1	Bias c2	Bias k1	Bias k2
1	-3.7364e-09	-6.6212e-09	-1.6234e-05	-6.3085e-06
2	-1.7774e-09	-3.0224e-09	-1.6199e-05	-6.2492e-06
3	9.6142e-11	-4.5903e-10	-1.6167e-05	-6.2082e-06
4	1.7582e-09	1.3988e-09	-1.6140e-05	-6.1798e-06
5	3.1780e-09	2.7498e-09	-1.6117e-05	-6.1600e-06
6	8.7718e-09	6.4284e-09	-1.6042e-05	-6.1125e-06
7	9.4568e-09	6.6448e-09	-1.5671e-05	-5.9512e-06
8	9.5585e-09	6.6577e-09	-1.6037e-05	-6.1080e-06
9	9.6095e-09	6.6525e-09	-1.6041e-05	-6.1064e-06
10	9.6249e-09	6.6512e-09	-1.6043e-05	-6.1060e-06

	Relative tolerance (%)			
Job	RT c1	RT c2	RT k1	RT k2
1	-0.5870	-1.1874	-1.1160	-0.4473
2	-0.2792	-0.5440	-1.1136	-0.4431
3	0.0151	-0.0823	-1.1114	-0.4402
4	0.2762	0.2508	-1.1095	-0.4382
5	0.4993	0.4931	-1.1080	-0.4368
6	1.3780	1.1528	-1.1028	-0.4334
7	1.4856	1.1916	-1.0773	-0.4220
8	1.5016	1.1939	-1.1025	-0.4331
9	1.5096	1.1930	-1.1027	-0.4330
10	1.5120	1.1928	-1.1028	-0.4329

Table A.5: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the test in table 7.11.

	Rate of convergence ([s])			
Job	RoC c1	RoC c2	RoC k1	RoC k2
1	0.02	0.0367	0.016	5.7839e-04
2	0.0213	0.0313	0.0163	5.7839e-04
3	0.02	0.0273	0.016	5.7839e-04
4	0.02	0.0247	0.016	5.7839e-04
5	0.0186	0.022	0.016	5.7839e-04
6	0.0120	0.0126	0.016	5.7839e-04
7	0.0093	0.0086	0.0240	5.7839e-04
8	0.0013	0.0059	0.016	5.7839e-04
9	0.0013	5.7839e-04	0.016	5.7839-04
10	0.0013	5.7839e-04	0.016	5.7839e-04

Table A.6: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the test in table 7.11. Continuing to table A.5.

Spring Constant

	Bias (real value - estimated value)			
Job	Bias c1	Bias c2	Bias k1	Bias k2
1	1.3314e-08	9.5190e-09	-1.5879e-05	-5.1708e-06
2	9.6447e-09	6.7409e-09	-1.7154e-05	-6.3708e-06
3	9.5774e-09	6.6577e-09	-1.6592e-05	-6.2667e-06
4	9.5585e-09	6.7515e-09	-1.6037e-05	-6.1080e-06
5	7.2750e-09	3.7638e-09	-2.5667e-05	-8.1572e-06
6	7.2749e-09	3.7637e-09	-2.5666e-05	-8.1559e-06
7	7.2748e-09	3.7636e-09	-2.5665e-05	-8.1544e-06
8	7.2747e-09	3.7635e-09	-2.5665e-05	-8.1526e-06
9	7.2746e-09	3.7633e-09	-2.5664e-05	-8.1503e-06
10	7.2744e-09	3.7631e-09	-2.5560e-05	-8.1474e-06
11	7.2741e-09	3.7628e-09	-2.5660e-05	-8.1435e-06
12	7.2738e-09	3.7624e-09	-2.5658e-05	-8.1478e-06
13	9.4423e-09	6.7571e-09	-6.3760e-05	-2.0100e-06

	Relative tolerance (%)			
Job	RT c1	RT c2	RT k1	RT k2
1	2.0916	1.7070	-1.0916	-0.3666
2	1.5151	1.2029	-1.1792	-0.4517
3	1.5046	1.1956	-1.1406	-0.4443
4	1.5016	1.1939	-1.1025	-0.4331
5	1.1429	0.6750	-1.7645	-0.5784
6	1.1428	0.6749	-1.7644	-0.5783
7	1.1428	0.6749	-1.7644	-0.5782
8	1.1428	0.6749	-1.7643	-0.5780
9	1.1428	0.6749	-1.7642	-0.5779
10	1.1428	0.6748	-1.7641	-0.5777
11	1.1427	0.6748	-1.7640	-0.5774
12	1.1427	0.6747	-1.7638	-0.5770
13	1.4833	1.1876	-4.3831	-1.4252

Table A.7: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the test in table 7.13.

	Rate of convergence ([s])			
Job	RoC c1	RoC c2	RoC k1	RoC k2
1	0.0013	0.0059	0.0013	5.7839e-04
2	0.0026	0.0073	0.0039	5.7839e-04
3	0.0026	0.0059	0.0106	5.7839e-04
4	0.0013	0.0059	0.016	5.7839e-04
5	0.0093	0.014	0.032	5.7839e-04
6	0.0093	0.014	0.032	5.7839e-04
7	0.0093	0.014	0.032	5.7839e-04
8	0.0093	0.014	0.032	5.7839e-04
9	0.0093	0.014	0.032	5.7839e-04
10	0.0093	0.014	0.032	5.7839e-04
11	0.0093	0.014	0.032	5.7839e-04
12	0.0093	0.014	0.032	5.7839e-04
13	0.0013	0.0059	0.0641	0.022

Table A.8: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the test in table 7.13. Continuing to table A.7.

Optimized Gain Matrix and Changing the Tunable Positive Filter Constant

	Bias (real value - estimated value)			
Job	Bias c1	Bias c2	Bias k1	Bias k2
1	9.4896e-09	6.5950e-09	-1.5990e-05	-6.1132e-06
2	9.5585e-09	6.6577e-09	-1.6037e-05	-6.1080e-06
3	7.3342e-09	3.8149e-09	-2.5664e-05	-8.1545e-06
4	7.3556e-09	3.8330e-09	-2.5662e-05	-8.1530e-06
5	7.3851e-09	3.8579e-09	-2.5660e-05	-8.1505e-06
	Relative tolerance (%)			
Job	RT c1	RT c2	RT k1	RT k2
1	1.4908	1.1827	-1.0992	-0.4334
2	1.5016	1.1939	-1.1025	-0.4331
3	1.1522	0.6841	-1.7643	-0.5782
4	1.1555	0.6874	-1.7642	-0.5781
5	1.1602	0.6918	-1.7640	-0.5779
	Rate of convergence ([s])			
Job	RoC c1	RoC c2	RoC k1	RoC k2
1	0.0013	0.0059	0.016	5.7839e-04
2	0.0013	0.0059	0.016	5.7839e-04
3	0.0093	0.014	0.032	5.7839e-04
4	0.0093	0.014	0.032	5.7839e-04
5	0.0093	0.014	0.032	5.7839e-04

Table A.9: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the Experiment 8 in table 7.15.

Optimized Gain Matrix with Different Simulation Solvers

	Bias (real value - estimated value)			
Job	Bias c1	Bias c2	Bias k1	Bias k2
1	9.5585e-09	6.6577e-09	-1.6037e-05	-6.1080e-06
2	9.4516e-09	6.7419e-09	-1.6063e-05	-6.1115e-06
3	9.6215e-09	6.7357e-09	-1.5955e-05	-6.0505e-06
4	9.9363e-09	6.9493e-09	-1.6051e-05	-6.0947e-06
5	9.1384e-09	6.3526e-09	-1.5894e-05	-6.1236e-06
6	9.6429e-09	6.7570e-09	-1.5938e-05	-5.9614e-06

	Relative tolerance (%)			
Job	RT c1	RT c2	RT k1	RT k2
1	1.5016	1.1939	-1.1025	-0.4331
2	1.4848	1.2090	-1.1042	-0.4333
3	1.5115	1.2079	-1.0968	-0.4290
4	1.5609	1.2462	-1.1034	-0.4321
5	1.4356	1.1392	-1.0926	-0.4342
6	1.5149	1.2117	-1.0956	-0.4227

	Rate of convergence ([s])			
Job	RoC c1	RoC c2	RoC k1	RoC k2
1	0.0013	0.0059	0.016	5.7839e-04
2	0.0013	0.0019	0.016	5.7839e-04
3	0.0013	5.7839e-04	0.016	5.7839e-04
4	0.0013	5.7839e-04	0.016	5.7839e-04
5	0.0013	5.7839e-04	0.016	5.7839e-04
6	0.0013	5.7839e-04	0.016	5.7839e-04

Table A.10: The results given as the bias, the relative tolerance (four decimals) and the rate of convergence corresponding to the Experiment 7 in table 7.17.

Appendix B

Code

Code from Vilje

Listing B.1: Run Parallel Jobs on Vilje

```
#!/bin/bash
##################################
#
# Matlab job
#
##################################
#
#PBS -N parCompJob1
#PBS -A ntnu265
#PBS -l select=1:ncpus=32:ompthreads=16
#PBS -l walltime=05:00:00
#PBS -q workq
#

cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=16

module load matlab/R2016b

matlab -nodisplay -nosplash -r donothing
matlab -nodisplay -nosplash -r ViljeJob1
```

For description of the different part of the code B.1, see subsection 5.1.2.

Listing B.2: Do Nothing file

```
display('donothing')
```

Matlab Code

BiasAndConvRate.m

Listing B.3: Finding the bias, relative tolerance and rate of convergence

```
%% Find the bias, relative tolerance and the rate of
    convergence for the different tests

%clear all
%clc

% Load .mat file
%load('logsout')

% Retrive the results from the logout file
t = logsout.get('theta').Values.Time;
theta = logsout.get('theta').Values.Data/p.
    num_elements_under_tip;
u =  logsout.get('U').Values.Data;
x =  logsout.get('X').Values.Data;
y =  logsout.get('Y').Values.Data;
Fk =  logsout.get('Fk').Values.Data(:);
Fc =  logsout.get('Fc').Values.Data(:);
kj =  logsout.get('kj').Values;
cj =  logsout.get('cj').Values;
D =  logsout.get('D').Values.Data;
Z =  logsout.get('Z').Values.Data;
height =  logsout.get('height').Values.Data;
t_ends = [t(1) t(end)];

% Real value
t1__ = [8.225 8.325]; % between these two times the first
    real values exist
t2__ = [8.425 8.525]; % between these two times the second
    real values exist

c1_ = mean(interp1(cj.Time, cj.Data(:), t1__)); % first
    average real damper value between 8.225 and 8.325
c2_ = mean(interp1(cj.Time, cj.Data(:), t2__)); % second
    average real damper value between 8.425 and 8.525

k1_ = mean(interp1(kj.Time, kj.Data(:), t1__)); % first
    average real spring value between 8.225 and 8.325
k2_ = mean(interp1(kj.Time, kj.Data(:), t2__)); % second
    average real spring value between 8.425 and 8.525
```

```

% The estimated values over the whole figure
t_ = linspace(8.15, 8.54999, 300);
theta_ = interp1(t, theta, t_);

% Estimated damper values
c_hat = theta_(:,1);
% Estimated spring values
k_hat = theta_(:,2);

%% Finding the bias
% Find the bias by taking the average over the estimated
   parameters from
% the middle of the real values to the end. Then taking
   this value minus
% the real value.

% Find the indices on the time axis where the real values
   lies
idx1 = find(t_>=t1__(1) & t_<=t1__(2));
idx2 = find(t_>=t2__(1) & t_<=t2__(2));

% Values in c_hat and k_hat that correspond to the indices
   in time1 and time2
c_hat1 = c_hat(idx1(1):idx1(end));
c_hat2 = c_hat(idx2(1):idx2(end));

k_hat1 = k_hat(idx1(1):idx1(end));
k_hat2 = k_hat(idx2(1):idx2(end));

% Calculate the bias from the middle of the real values
% Find c_hat and k_hat for these values
c_hat1_mid = c_hat1(end/2:end);
c_hat2_mid = c_hat2(end/2:end);

k_hat1_mid = k_hat1(end/2:end);
k_hat2_mid = k_hat2(end/2:end);

% Bias
bias_c1 = c1_ - mean(c_hat1_mid);
bias_c2 = c2_ - mean(c_hat2_mid);

bias_k1 = k1_ - mean(k_hat1_mid);
bias_k2 = k2_ - mean(k_hat2_mid);

```

```
%% Relative tolerance
% Relative tolerance (%) (bias/real value *100%)
rel_error_c1 = bias_c1/c1_ *100;
rel_error_c2 = bias_c2/c2_ *100;

rel_error_k1 = bias_k1/k1_ *100;
rel_error_k2 = bias_k2/k2_ *100;

%% Finding the rate of convergence

tid1 = t_(idx1(1:end));
tid2 = t_(idx2(1:end));

% 95% of the wanted values
c1_95 = c1_+(c1_*0.05);
c2_95 = c2_+(c2_*0.05);

k1_95 = k1_+(k1_*0.05);
k2_95 = k2_+(k2_*0.05);

% The estimated values converge between
conv_c1 = [c1_*0.95 c1_95]; %epsilon_c1
conv_c2 = [c2_*0.95 c2_95]; %epsilon_c2

conv_k1 = [k1_*0.95 k1_95]; %epsilon_k1
conv_k2 = [k2_*0.95 k2_95]; %epsilon_k2

% Indices where the estimated values are 95% or more of the
    real values
c1_idx=find(c_hat1>=conv_c1(1) & c_hat1<=conv_c1(2));
c2_idx=find(c_hat2>=conv_c2(1) & c_hat2<=conv_c2(2));

k1_idx=find(k_hat1>=conv_k1(1) & k_hat1<=conv_k1(2));
k2_idx=find(k_hat2>=conv_k2(1) & k_hat2<=conv_k2(2));

% Time the estimated values crosses epsilon
t_eps_c1 = tid1(c1_idx(1));
t_eps_c2 = tid2(c2_idx(1));

t_eps_k1 = tid1(k1_idx(1));
t_eps_k2 = tid2(k2_idx(1));

% Rate of Convergence
RoC_c1 = t_eps_c1-t1__(1);
RoC_c2 = t_eps_c2-t2__(1);
```

```

RoC_k1 = t_eps_k1-t1__(1);
RoC_k2 = t_eps_k2-t2__(1);
```

ViljeJob.m

Listing B.4: Matlab code on Vilje

```
%% Simulating the model and extract the results in Vilje
    with Instantaneous Cost
% Job number 1

% Simulating

p = init();
% Change p.Gamma and other variables from init.m

%Starting the simulation
sim('cell_boop_xy');

% Saving variables in .mat file
save('ResVar1.mat')
```

init.m

Listing B.5: init.m

```matlab
function p = init()
% Initialize parameters

p = struct;

%p.SimNumPeriods = 500;

p.f0 = 20e3; % 10e3 cantilever osc. freq. [Hz]
p.zeta = 1/200;
p.M = 0.0075*9/20*125e-6*30e-6*4e-6*2330*100;
%p.LJ_sigma = 3.41e-10; %theta_2
%p.LJ_k_1 = -2/3*pi^2*20e-21*(2.5e-3*1e-10^-3)^2*3.41e
    -10^4*200e-10;

p.K = p.M*(p.f0*2*pi)^2;
p.C = 2*p.zeta*sqrt(p.M*p.K);

p.R = 100e-9;%6.4516e-08;% 50e-9;
%p.d0 = 300e-9;
p.X0 = 0e-9;
p.Y0 = 0e-9;
zi_max = 0.25e-6;

p.U0 = zi_max + p.R + 30e-9;
%p.u0 = p.R - 4e-7 - 100e-9;
p.Z0 = p.U0;

%p.omega0 = p.f0*2*pi;
%p.LJ_c_1 = p.LJ_k_1 * p.LJ_sigma^2;
%p.LJ_c_2 = 1/30 * p.LJ_k_1 * p.LJ_sigma^8;
%p.Spp = 2000; % Samples per cantilever period
%p.phi = 15*pi/180; % Measured signal phase [rad]

p.T0 = 1/p.f0; % cantilever oscillation period [s]
%p.Tend = p.SimNumPeriods * p.T0;
%p.Ts = p.T0 / p.Spp; % sample time [s]
%p.fs = 1/p.Ts;
%p.Soffset = round(p.phi*p.Spp/(2*pi)); % Num. samples to
    offset signal


p.num_taps = 10; % per axis (for a total of num_taps^2)
```

```
p.tap_period = 0.1; % Tapping period
%p.tap_period = 0.300;
p.tap_moving_time = 0.05;
p.x_scan_period =  p.num_taps * p.tap_period;

p.Tend = p.x_scan_period * p.num_taps;


% Make random functions repeatable
rng('default');
rng(15);


%k = 0.05; % Spring constant [N/m]
k_B = 1.3806488e-23; %Boltzmann constant [m2 kg s-2 K-1]
T = 293.15; % Temperature [K]

%p.A = 10e-9; % Deflection amplitude [m]

p.w_z_sqr = k_B / p.K * T; % Thermal noise (Hutter and
    Bechhoefer, 1993)
%p.w_a_sqr = (0.05*p.A)^2;



N = 32;      % Num. elements per axis
L = 1e-6;     % length of cell [m]
dx = L/(N-1);
xi = -p.R:dx:L+p.R;
dy = L/(N-1);
yi = -p.R:dy:L+p.R;
p.num_elements_under_tip = (pi/4)*(2*p.R)^2/(dx*dy);

[Xi,Yi] = meshgrid(xi,yi);

E = 5e3;       % Typical Young's modulus [Pa]
A0 = dx*dy;     % Contact area / element area [m^2]
H0 = 1e-9;%0.5e-6;   % Typical cell/element height [m]
k0 = E*A0/H0/4; % Typical spring constant [N/m]

xic = (Xi-L/2)/2;
yic = (Yi-L/2)/2;
Sphere = max(sqrt(L^2 - xic.^2 - 1.2*yic.^2)*1e7-9.5, 0)*2;
InvSphere = 0.9*(1-Sphere).^3+0.1;

```

```
Nk = imgaussfilt(randn(size(Xi)),4);
Nz = -imgaussfilt(randn(size(Xi)),8).*Sphere;  rng(6);
Nc = imgaussfilt(randn(size(Xi)),6).*Sphere;% - 0.5*Sphere;
%surf(xic, yic, Nc);

z_unscaled = (-sin(1*pi*Xi/L)-0.5*sin(3.5*pi*Xi/L) -0.6*sin
    (3*pi*Yi/L)+0.5*cos(1.0*pi*Yi/L-pi*0.35)-1.0*sin(4*pi*Xi
    .*Yi/L^2)+3 + 15.3*Nz).*Sphere;
k_unscaled = -(Nk+.5).*(Sphere+1.5);%(Nk.*Sphere +
    InvSphere);
c_unscaled = Nc;

% Scale to min and max values
p.xi = xi; p.dx = dx; p.L = L;
p.yi = yi; p.dy = dy;
p.zi0 = scale(z_unscaled, 0, zi_max);
p.ki = scale(k_unscaled, k0*0.5, k0*2);
p.ci = scale(c_unscaled, 4e-7, 8e-7);

figure(1);clf;
%surf(xi, yi, p.zi0); axis equal;
surf(xi, yi, p.ki); hold on; imagesc(xi, yi, p.ki);
%surf(xi, yi, p.ci); hold on; imagesc(xi, yi, p.ci);%
    imagesc(xi, yi, p.ci);
% xlabel('x'); ylabel('y');

%plot(p.xi,p.zi0); grid on;legend('z_0');
%semilogy(p.xi,p.zi0,p.xi,p.ki,p.xi,p.ci); legend('z_0','k
    ','c'); grid on;



% Parameter identification
p.filter_tau = 0.1*p.T0; % also used for d/dt filter 0.03
%p.filter_tau = 0.01*p.T0;

p.theta0 = [p.ci(1)*p.num_elements_under_tip; p.ki(1)*p.
    num_elements_under_tip];
%p.Gamma = 1e-0*diag([10e8 10e18]); %real gamma matrix (2e8
     Estimator gain)
p.Gamma = 1e-0*diag([4e8 10e14]);
p.alpha = 1e5;
p.beta = 3e1;%1e2;
p.R0 = 40*norm(p.Gamma); % 20
p.Lambda = [p.filter_tau^2 2*p.filter_tau 1];
```

```

[A, B, C, D] = tf2ss([p.M 0 0], [p.filter_tau^2 2*p.
    filter_tau 1]);
p.f_A = A;
p.f_B = B;
p.f_C = C;
p.f_D = D;

% %% Asserts
% assert(mod(p.Spp, 4)==0, ['Samples per period should be
    divisible by 4
% '...
%     'to make sure cosine signal is exactly 90deg out of
    phase from sine']);
%

end
```

InstantaneousCost.m

Listing B.6: InstantaneousCost.m

```
function [theta_dot, w_hat] = fcn(w, phi, theta, p)
%#codegen

w_hat = theta' * phi;

m_sqr = 1 + p.alpha*(phi')*phi;

epsilon = (w - w_hat) / m_sqr;

theta_dot = p.Gamma * epsilon * phi;

end
```

run.m

Listing B.7: run.m

```
close all;
clear;
clc;
p = init();
close all;
%tic;
%logsout = sim('cell_boop_xy');
%toc;
```

plot_short_time.m

Listing B.8: Plot the estimated values

```
t = logsout.get('theta').Values.Time;
theta = logsout.get('theta').Values.Data/p.
    num_elements_under_tip;
u =  logsout.get('U').Values.Data;
x =  logsout.get('X').Values.Data;
y =  logsout.get('Y').Values.Data;
Fk =  logsout.get('Fk').Values.Data(:);
Fc =  logsout.get('Fc').Values.Data(:);
kj =  logsout.get('kj').Values;
cj =  logsout.get('cj').Values;
D =  logsout.get('D').Values.Data;
Z =  logsout.get('Z').Values.Data;
height =  logsout.get('height').Values.Data;
t_ends = [t(1) t(end)];



%%
% First, find time indices just before raising cantilever
im = []; % Measurement indices
i = 0; n = 0;
for t1 = t'
    i = i + 1;
    if t1 > p.tap_period*n + p.tap_period-2*p.
        tap_moving_time;
        n = n + 1;
        im = [im i];
    end
end

%% Time dependent theta plot

figure(1); clf;
subplot(3,1,[1 2]);

t_ = linspace(8.15, 8.54999, 300);
theta_ = interp1(t, theta, t_);

[ax,h1,h2] = plotyy(t_,theta_(:,1), t_, theta_(:,2));
grid on;
xlabel(ax(1),'$t$ [s]') % label x-axis
ylabel(ax(1),'$\widehat{c}$','Interpreter','latex') % label
```

```
     left y-axis
ylabel(ax(2),'$\widehat{k}$','Interpreter','latex') % label
     right y-axis

ax1 = ax(1);
ax2 = ax(2);

ax1.XTick = 8.2:0.1:8.5;
ax1.XMinorTick = 'on';

t__ = [8.225 8.325];% 8.45 8.5];
%t__ = [1.225 1.325];
k_ = interp1(kj.Time, kj.Data(:), t__);
c_ = interp1(cj.Time, cj.Data(:), t__);
h3 = line(t__, c_, 'Parent', ax(1), 'Color', h1.Color, '
    LineWidth', 1, 'Marker', 's', 'LineStyle', '--');
h4 = line(t__, k_, 'Parent', ax(2), 'Color', h2.Color, '
    LineWidth', 1, 'Marker', 's', 'LineStyle', '--');


t__ = [8.425 8.525];
%t__ = [1.425 1.525];
k_ = interp1(kj.Time, kj.Data(:), t__);
c_ = interp1(cj.Time, cj.Data(:), t__);
line(t__, c_, 'Parent', ax(1), 'Color', h1.Color, '
    LineWidth', 1, 'Marker', 's', 'LineStyle', '--');
line(t__, k_, 'Parent', ax(2), 'Color', h2.Color, '
    LineWidth', 1, 'Marker', 's', 'LineStyle', '--');

legend([h1 h2 h3 h4], '$\widehat{c}$','$\widehat{k}$','$c$'
    ,'$k$');

subplot(3,1,3);

t_ = linspace(t_(1), t_(end), 1000);
u_ = interp1(t, u, t_);
plot(t_,u_);
grid on;
xlabel('$t$ [s]','Interpreter','latex') % label x-axis
ylabel('$u$ [m]','Interpreter','latex') % label left y-axis
ax1 = gca;
ax1.XTick = 8.2:0.1:8.5;
ax1.XMinorTick = 'on';


```

```

%toTikz('theta_time');
```

Appendix C

Figures From the Simulation

C1 Changing the Gain Matrix

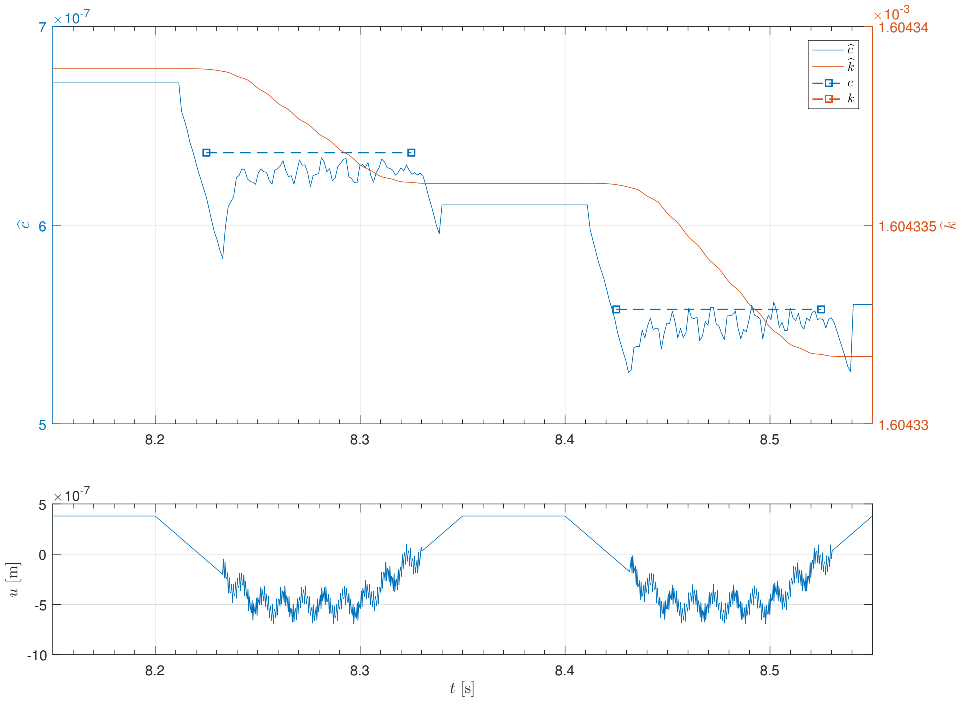

Figure C.1: Test 1 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

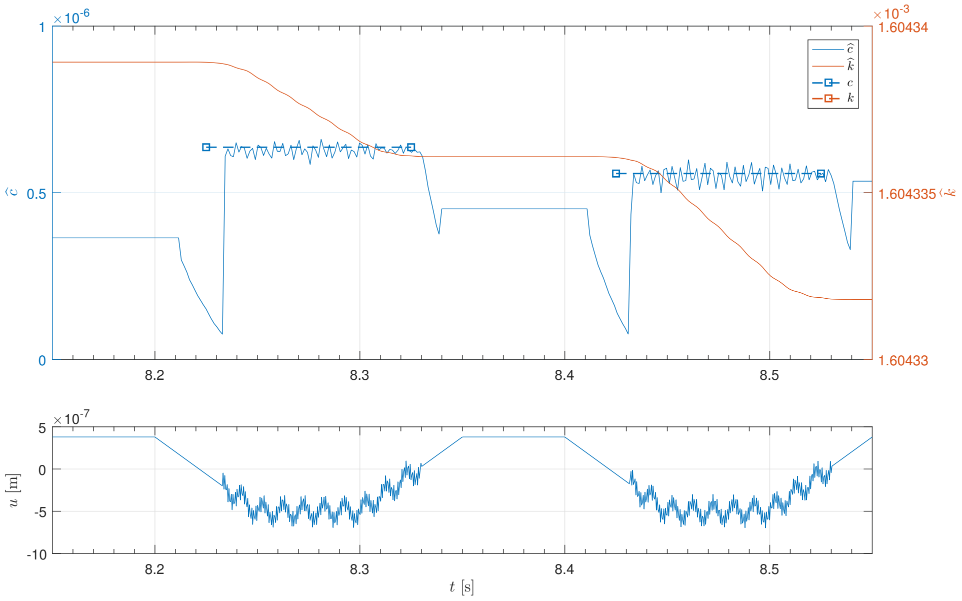

Figure C.2: Test 2 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

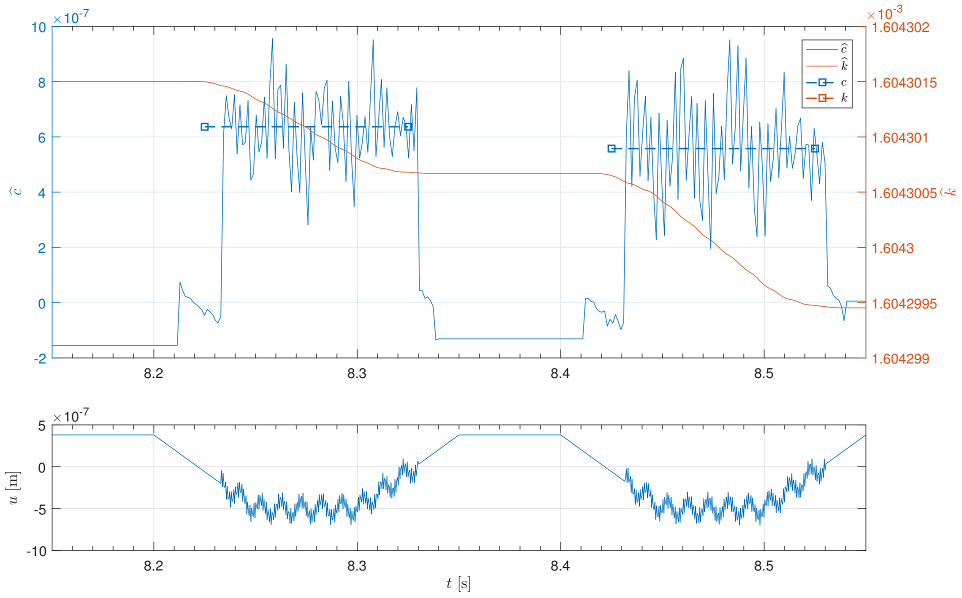

Figure C.3: Test 3 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

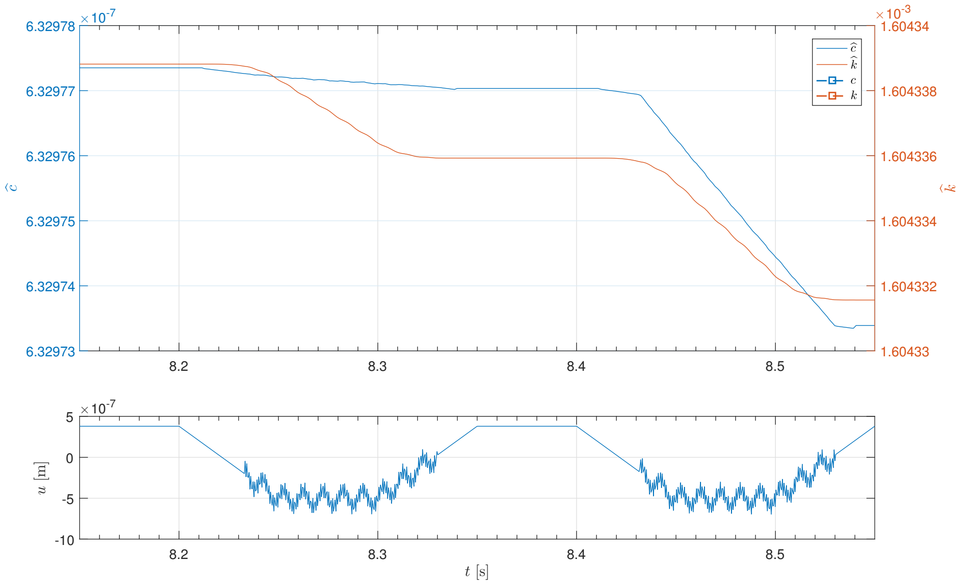

Figure C.4: Test 4 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

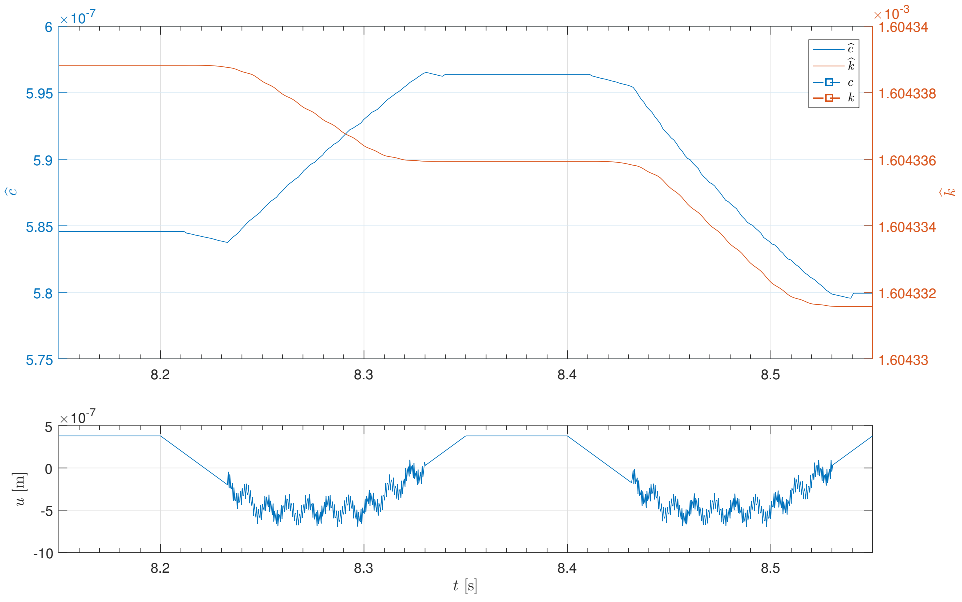

Figure C.5: Test 5 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

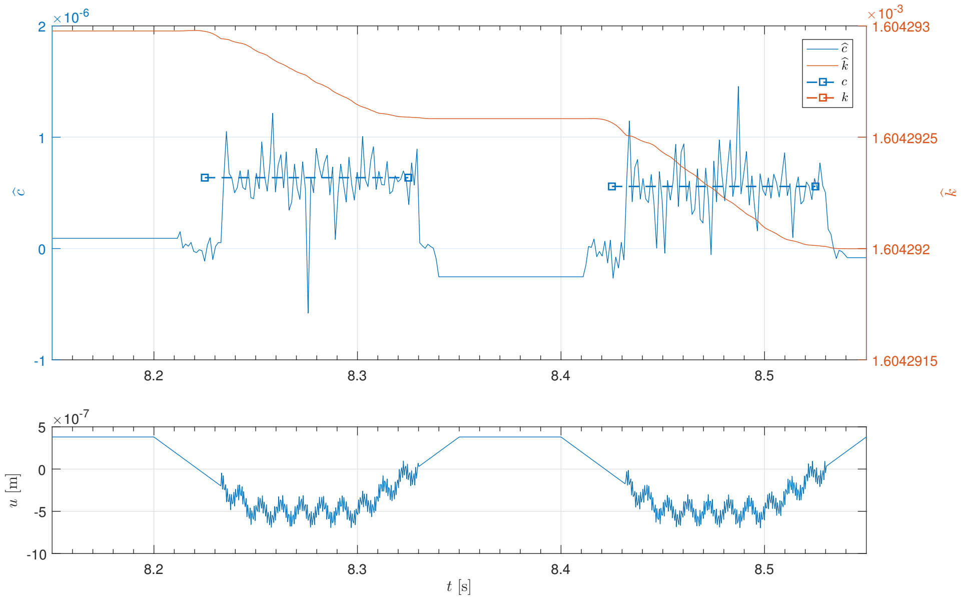

Figure C.6: Test 6 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

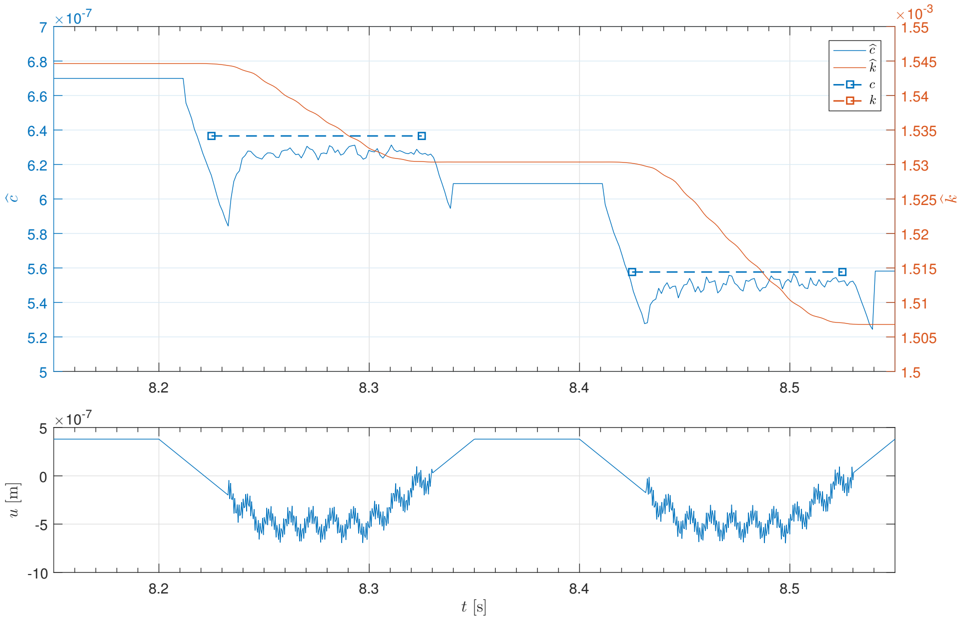

Figure C.7: Test 7 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

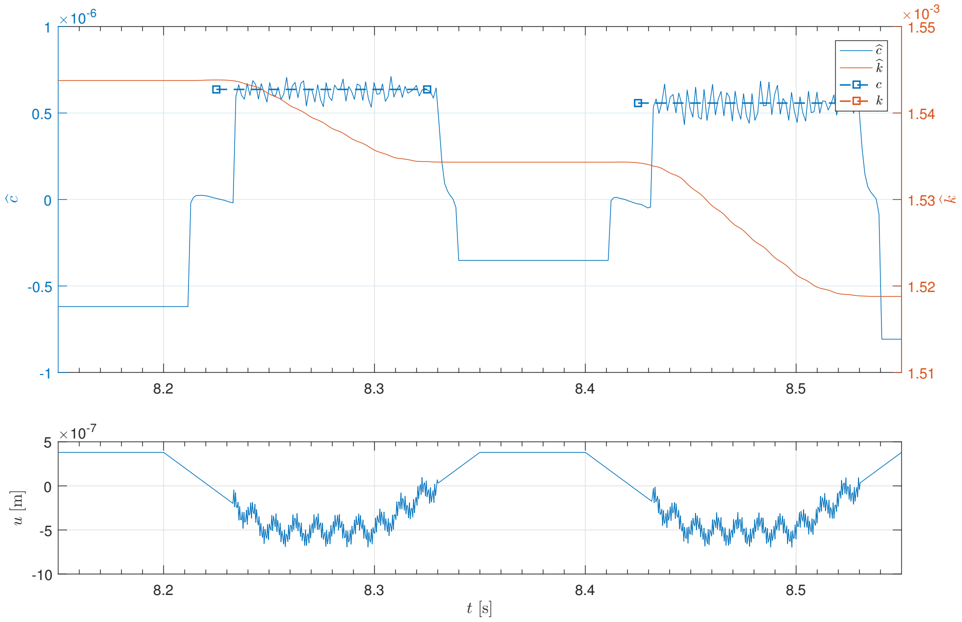

Figure C.8: Test 8 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

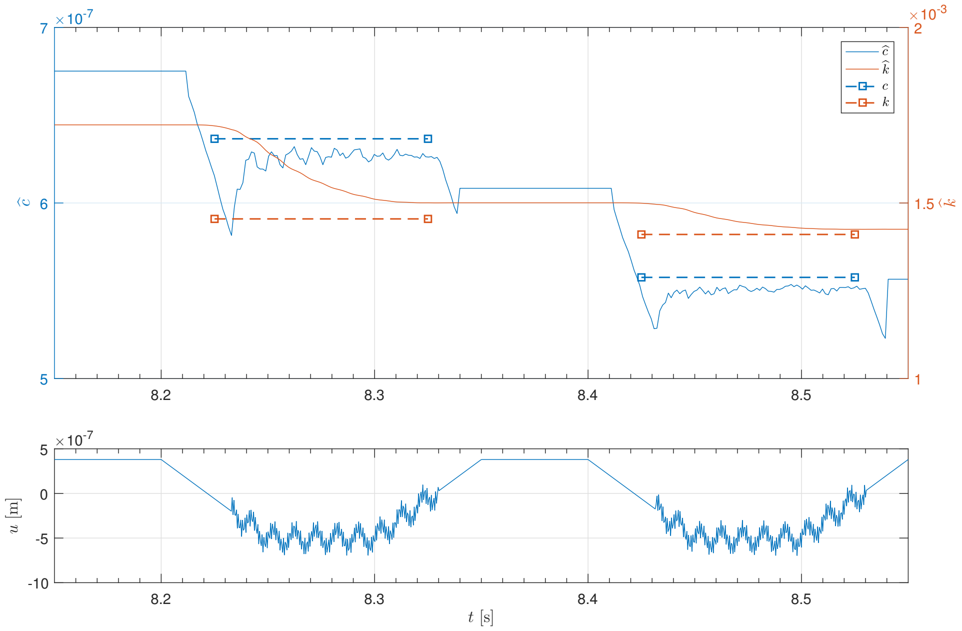

Figure C.9: Test 9 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

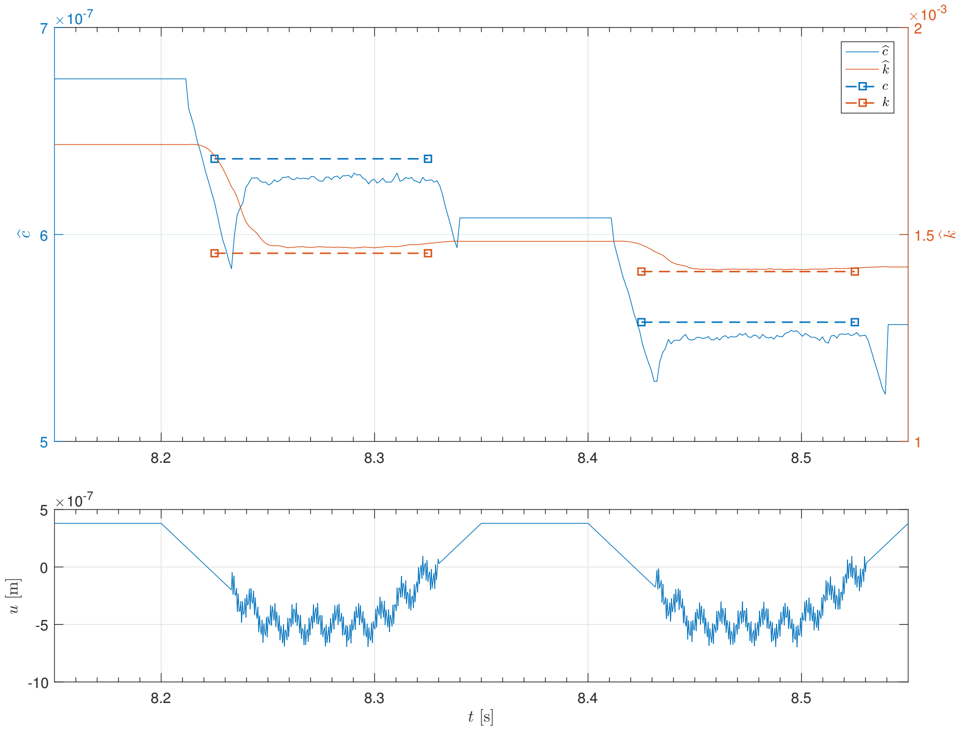

Figure C.10: Test 10 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

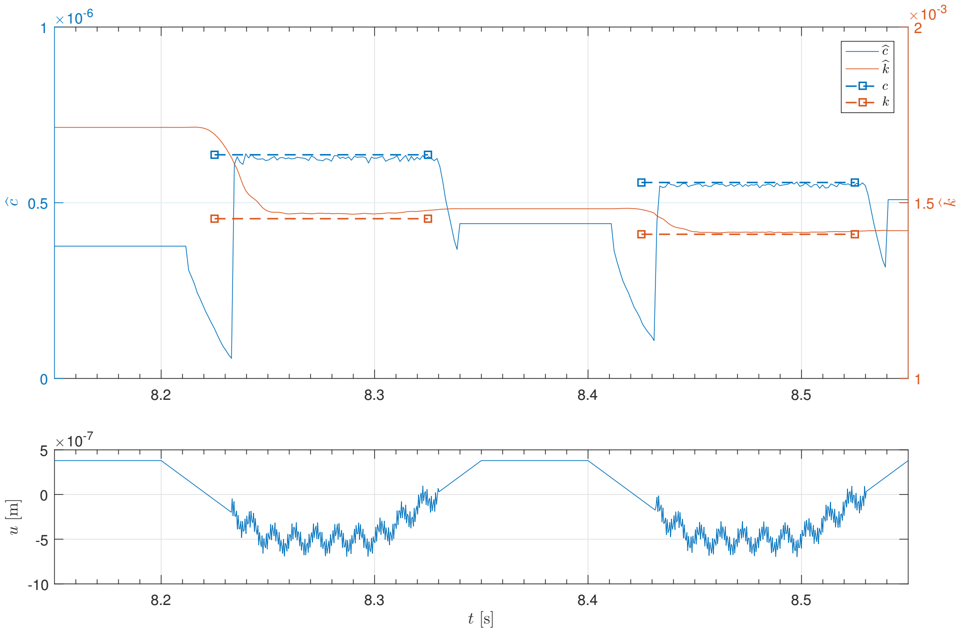

Figure C.11: Test 11 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

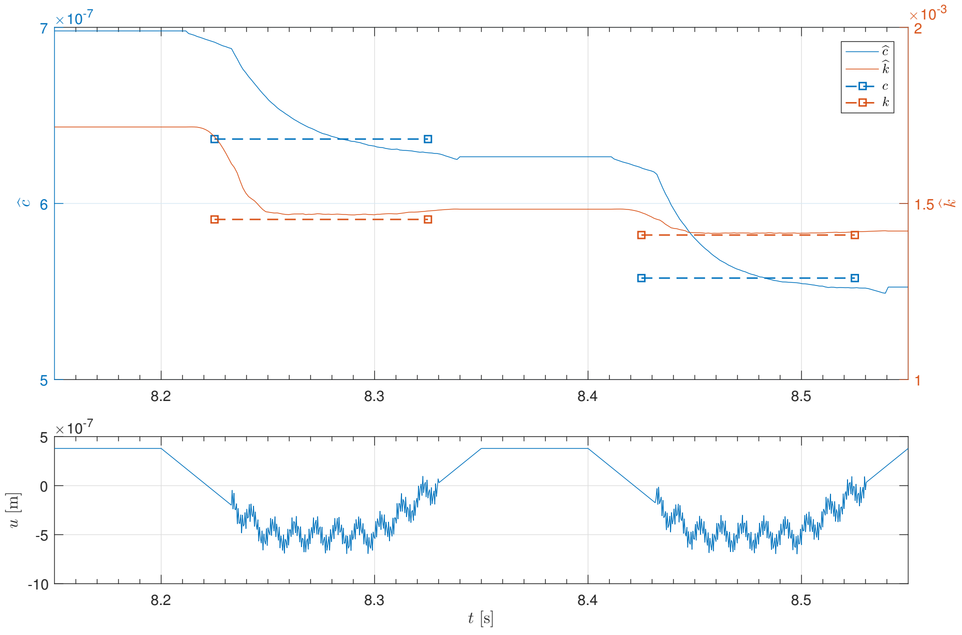

Figure C.12: Test 12 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

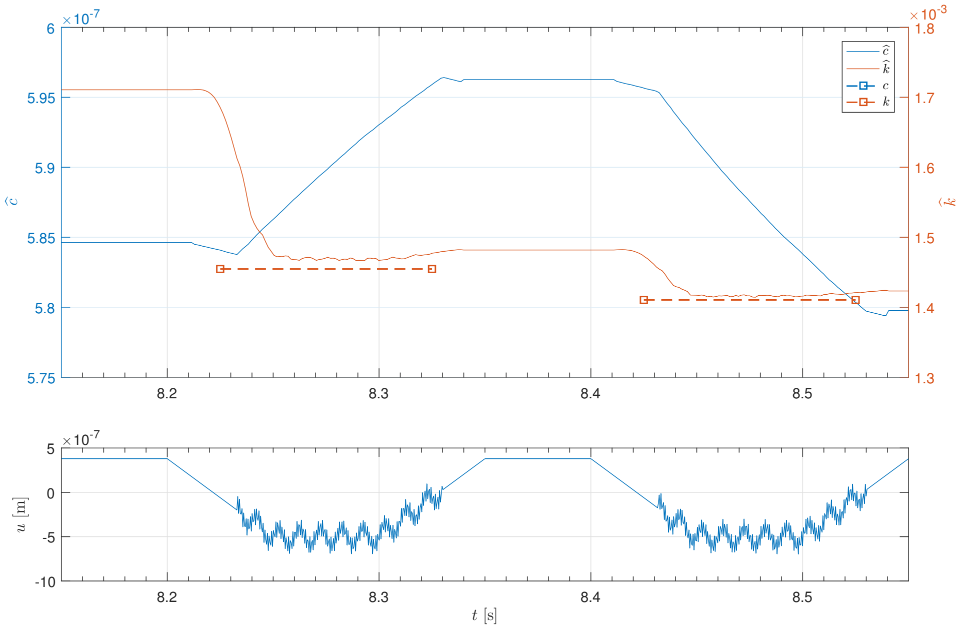

Figure C.13: Test 13 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

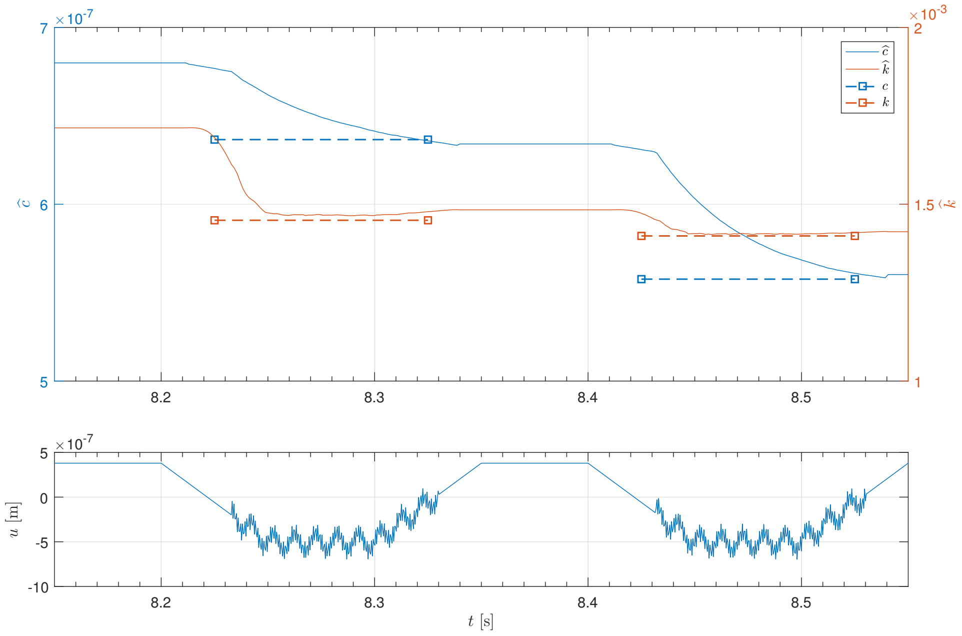

Figure C.14: Test 14 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

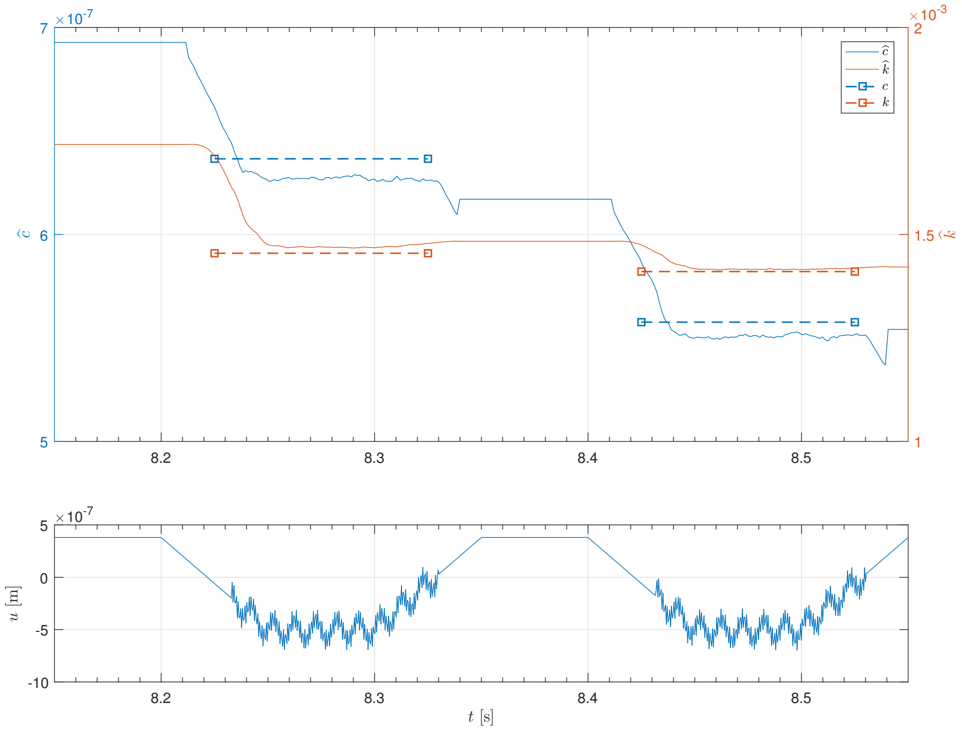

Figure C.15: Test 15 in table A.1: The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

C2 Changing the Simulation Solver

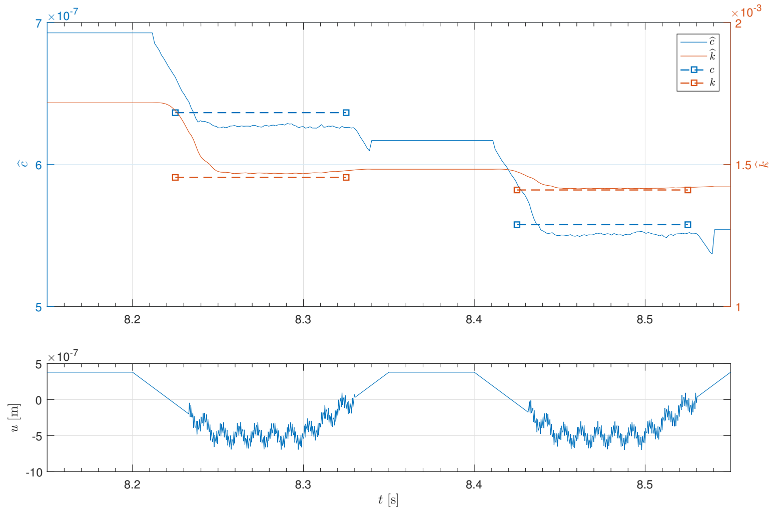

Figure C.16: Experiment 1, Job 1 (table 7.5): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

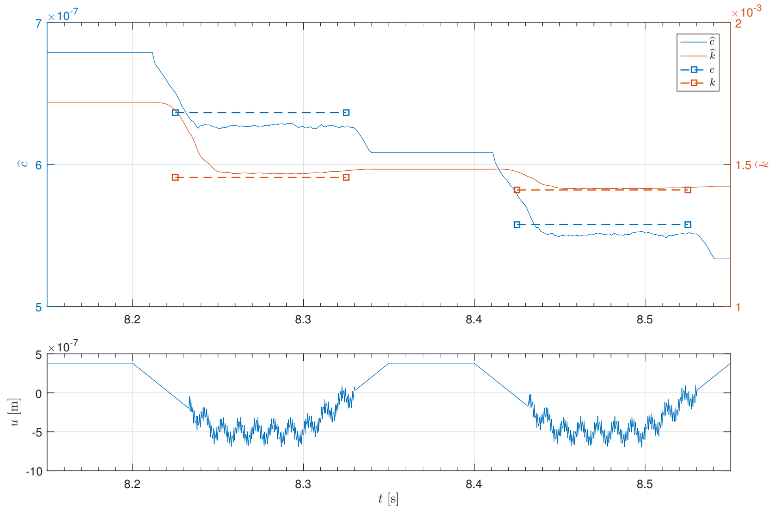

Figure C.17: Experiment 1, Job 2 (table 7.5): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

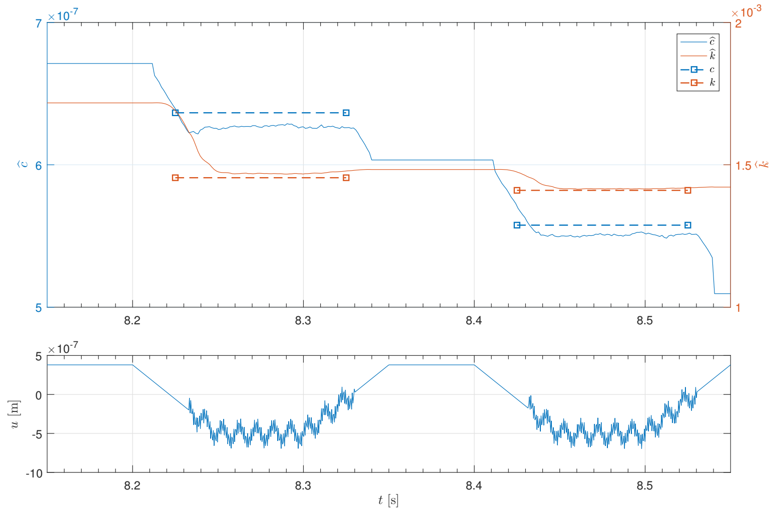

Figure C.18: Experiment 1, Job 3 (table 7.5): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

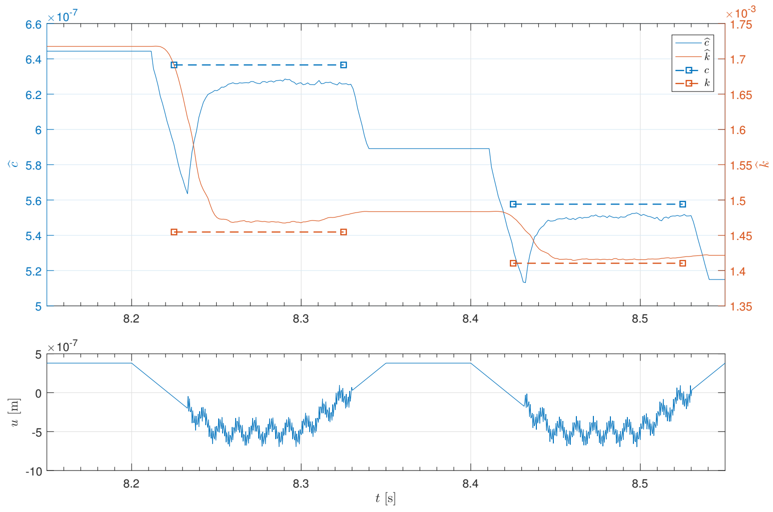

Figure C.19: Experiment 1, Job 4 (table 7.5): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

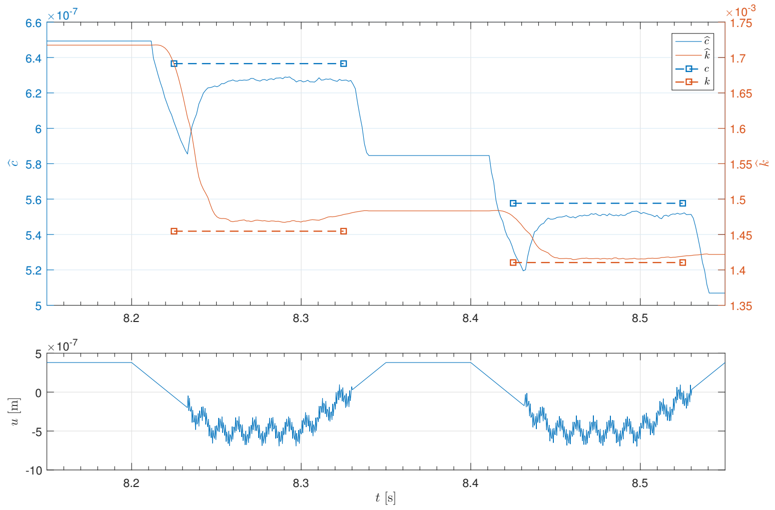

Figure C.20: Experiment 1, Job 5 (table 7.5): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

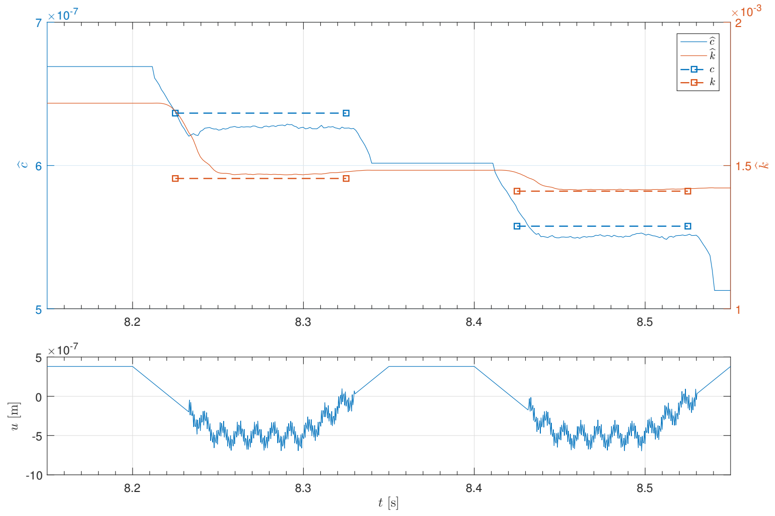

Figure C.21: Experiment 1, Job 6 (table 7.5): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

C3 Simulation Solver and the Relative Tolerance

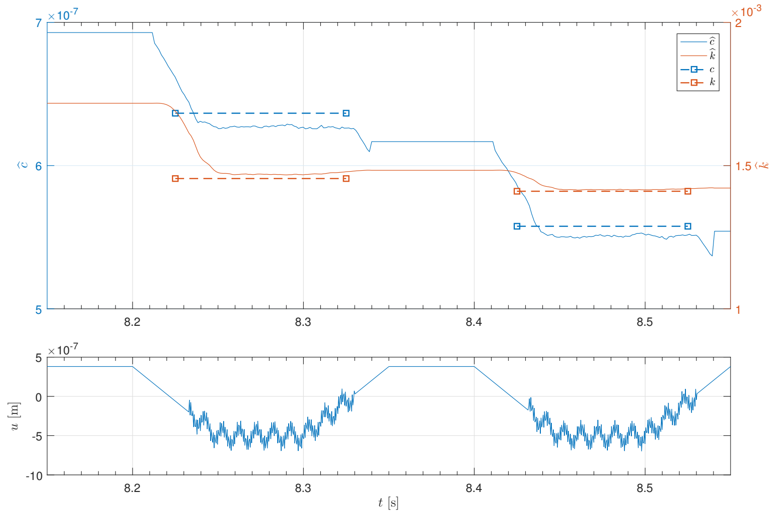

Figure C.22: Experiment 2, Job 1 (table 7.7): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

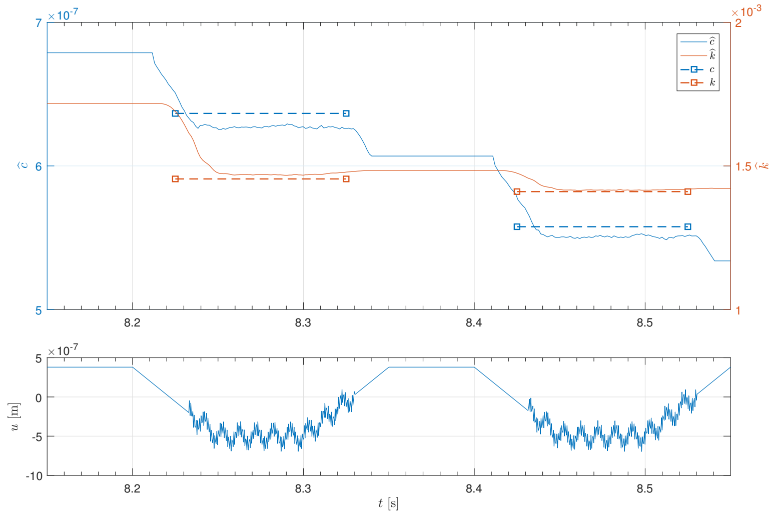

Figure C.23: Experiment 2, Job 2 (table 7.7): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

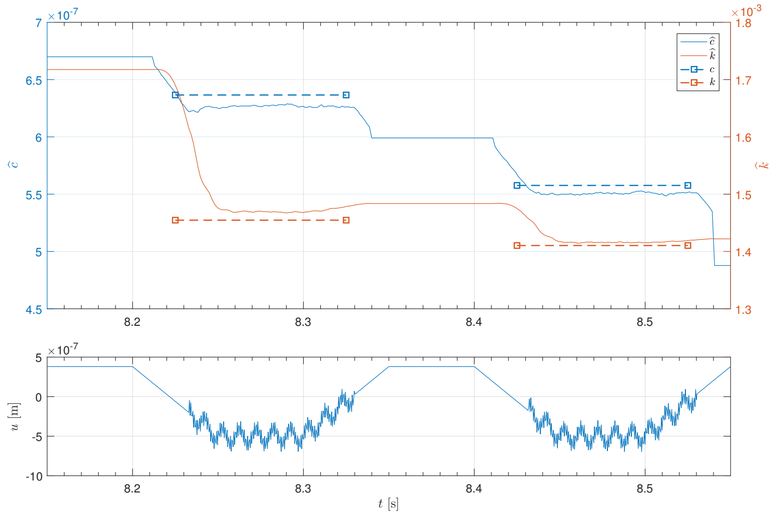

Figure C.24: Experiment 2, Job 3 (table 7.7): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

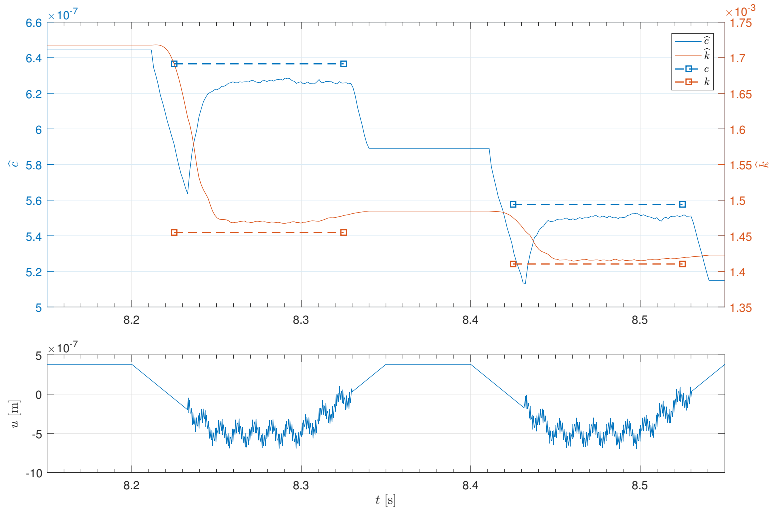

Figure C.25: Experiment 2, Job 4 (table 7.7): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

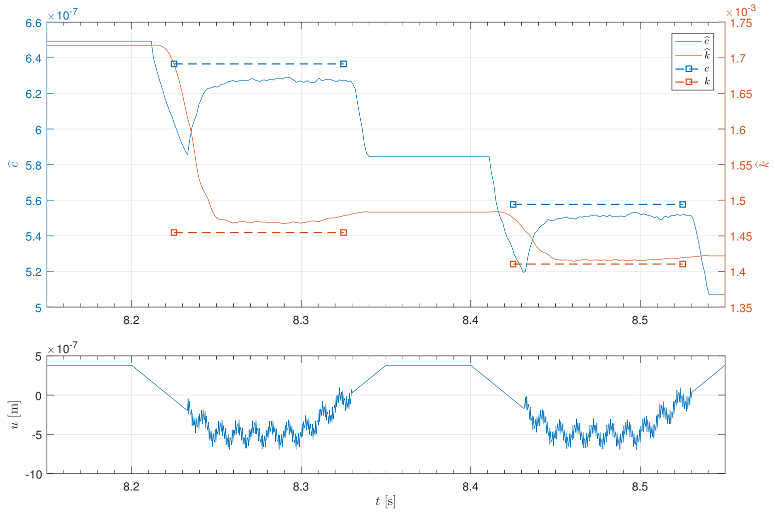

Figure C.26: Experiment 2, Job 5 (table 7.7): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

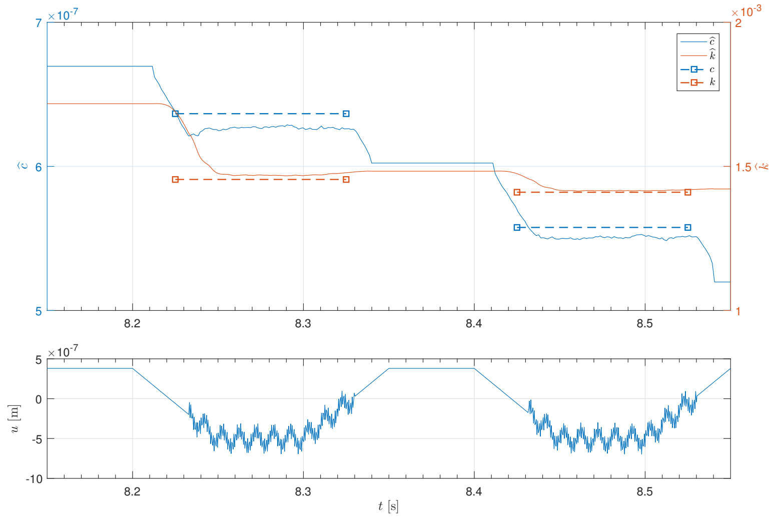

Figure C.27: Experiment 2, Job 1 (table 7.7): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

C4 Simulation Solver and Tap Period

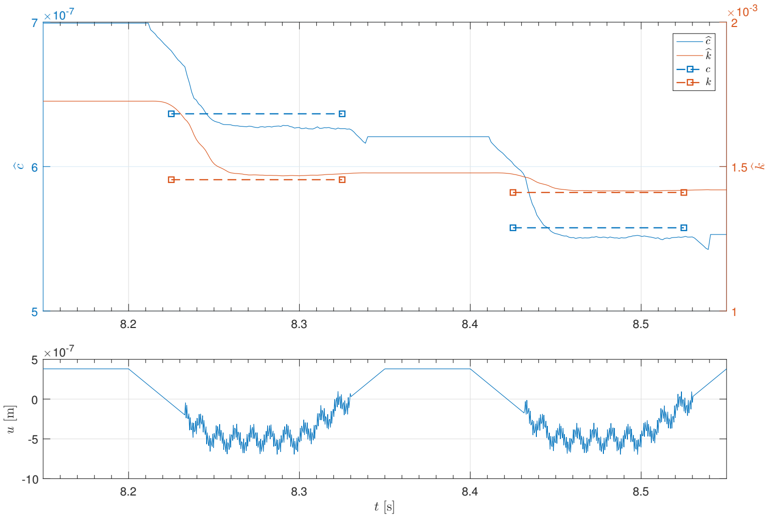

Figure C.28: Experiment 3, Job 1 (table 7.9): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

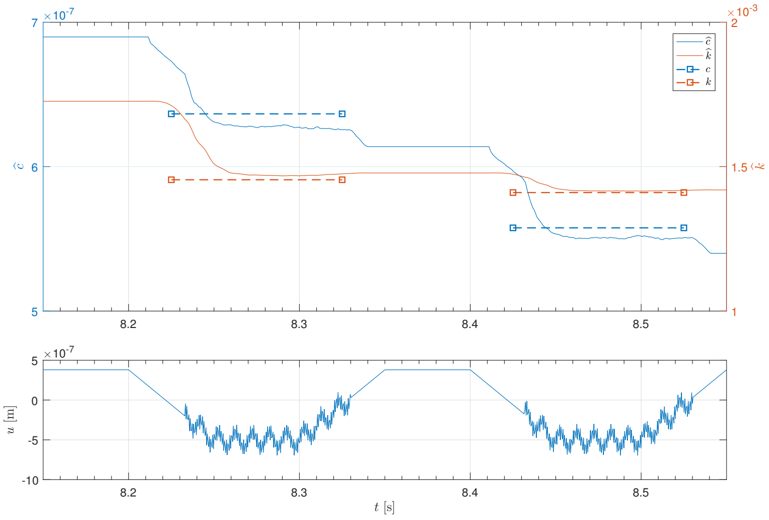

Figure C.29: Experiment 3, Job 2 (table 7.9): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

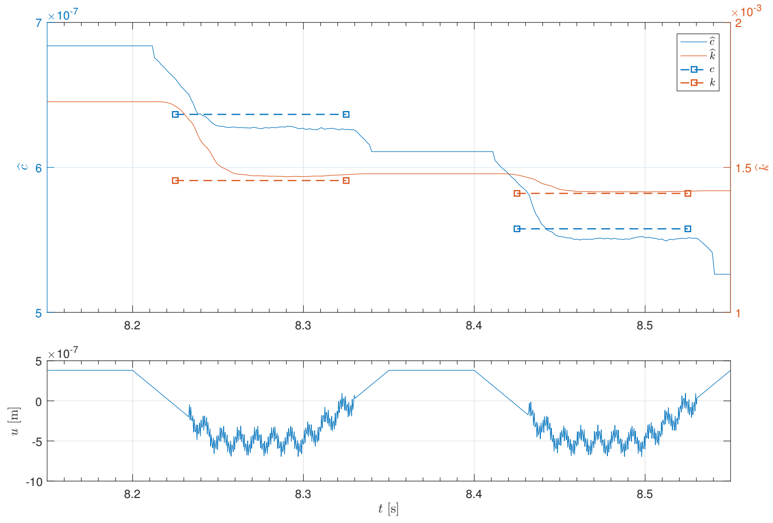

Figure C.30: Experiment 3, Job 3 (table 7.9): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

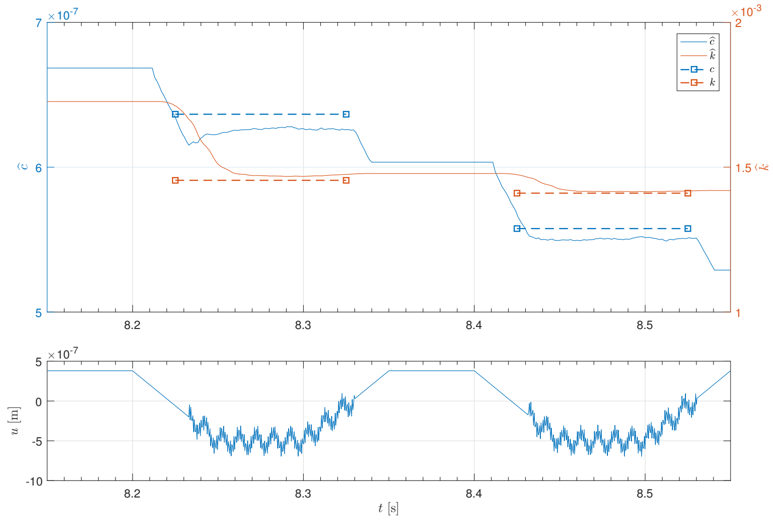

Figure C.31: Experiment 3, Job 4 (table 7.9): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

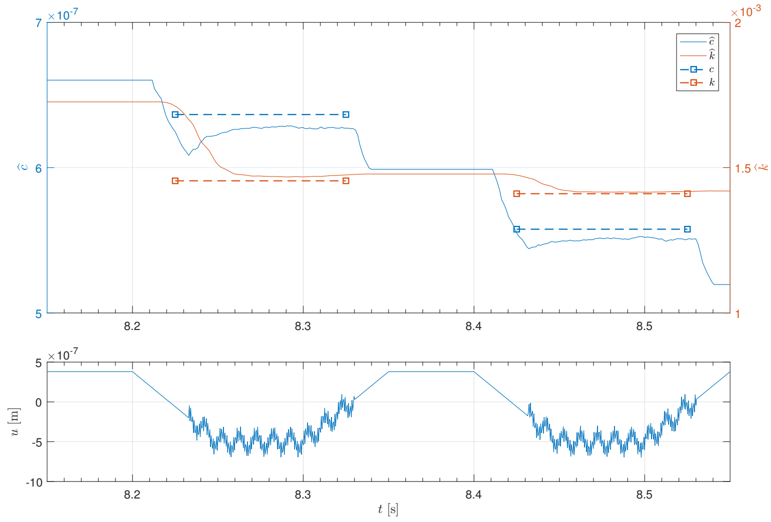

Figure C.32: Experiment 3, Job 5 (table 7.9): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

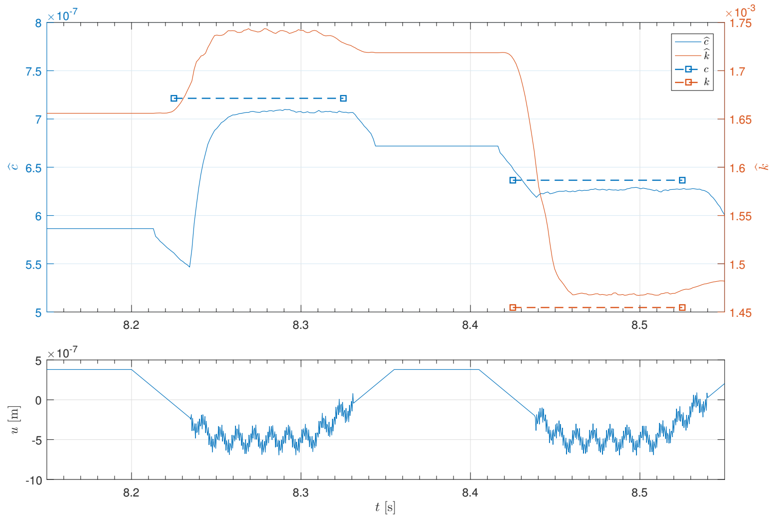

Figure C.33: Experiment 3, Job 6 (table 7.9): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

C5 Changing the Damper Constant In the Gain

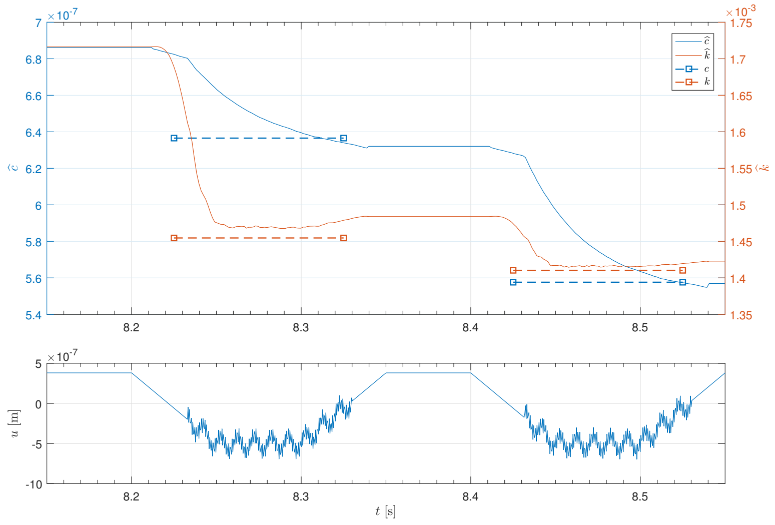

Figure C.34: Experiment 4, Job 1 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

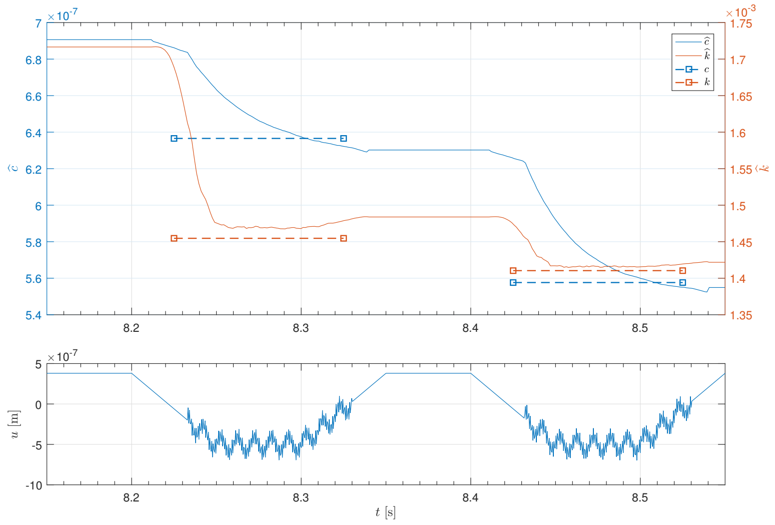

Figure C.35: Experiment 4, Job 2 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

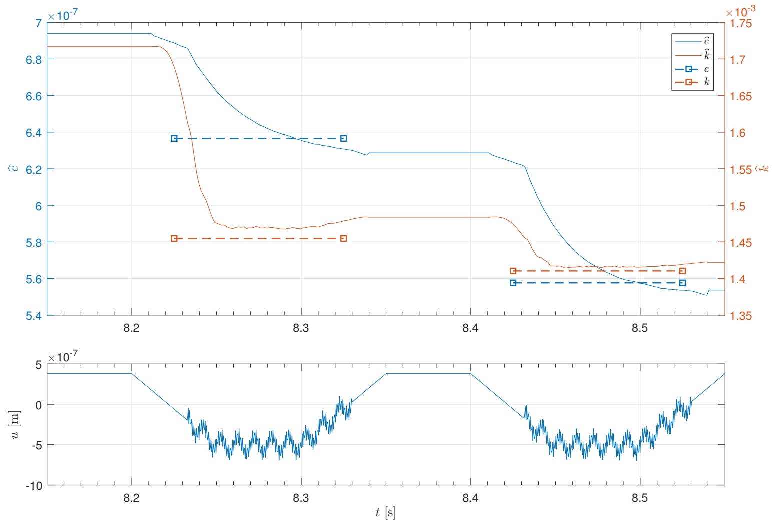

Figure C.36: Experiment 4, Job 3 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

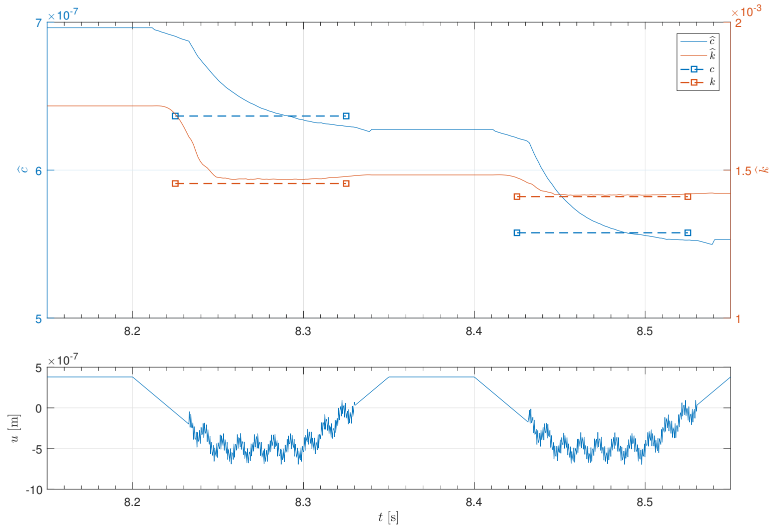

Figure C.37: Experiment 4, Job 4 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

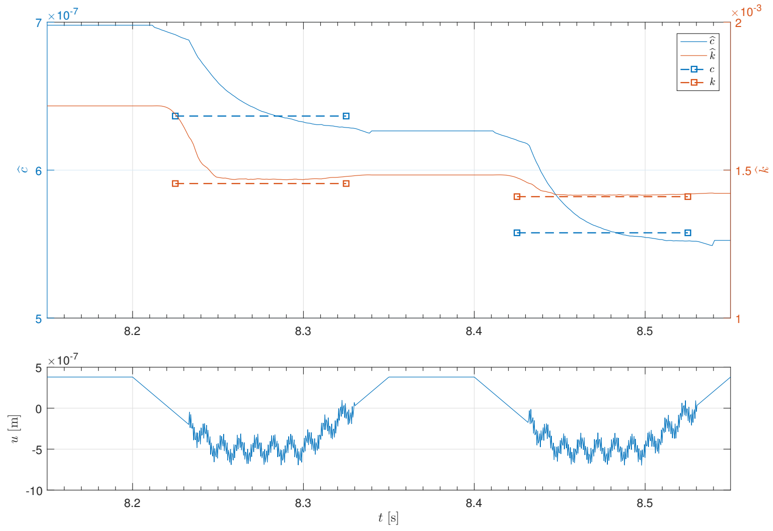

Figure C.38: Experiment 4, Job 5 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

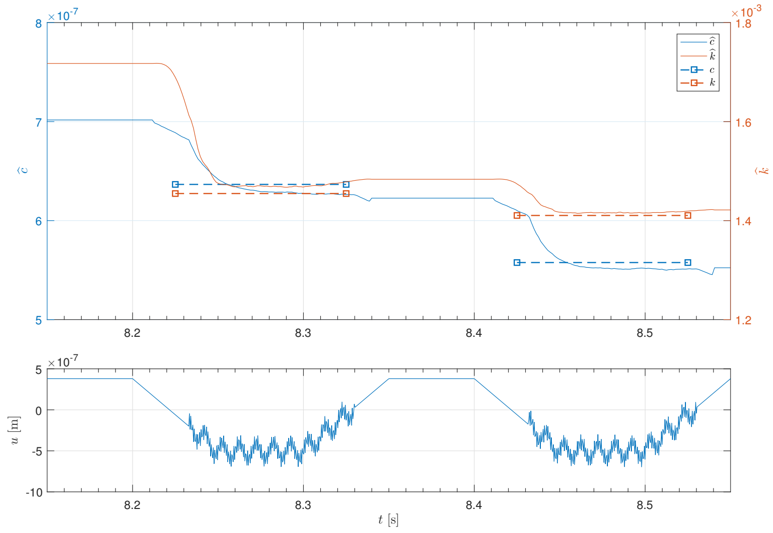

Figure C.39: Experiment 4, Job 6 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

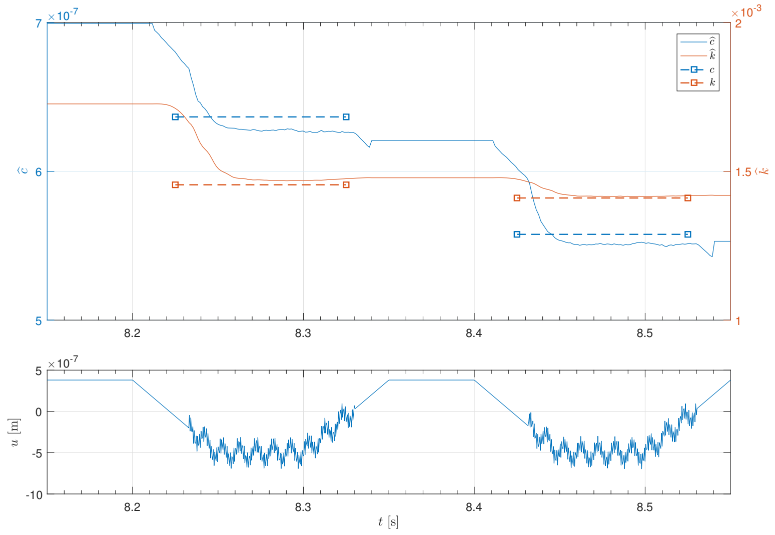

Figure C.40: Experiment 4, Job 7 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

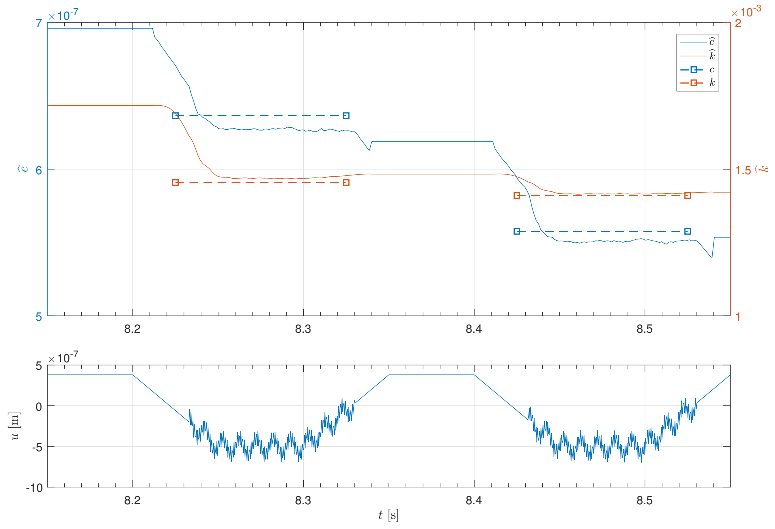

Figure C.41: Experiment 4, Job 8 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

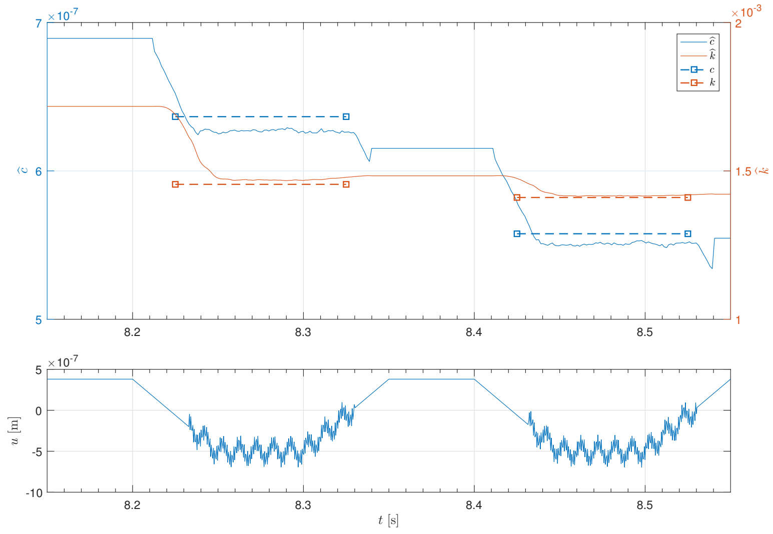

Figure C.42: Experiment 4, Job 9 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

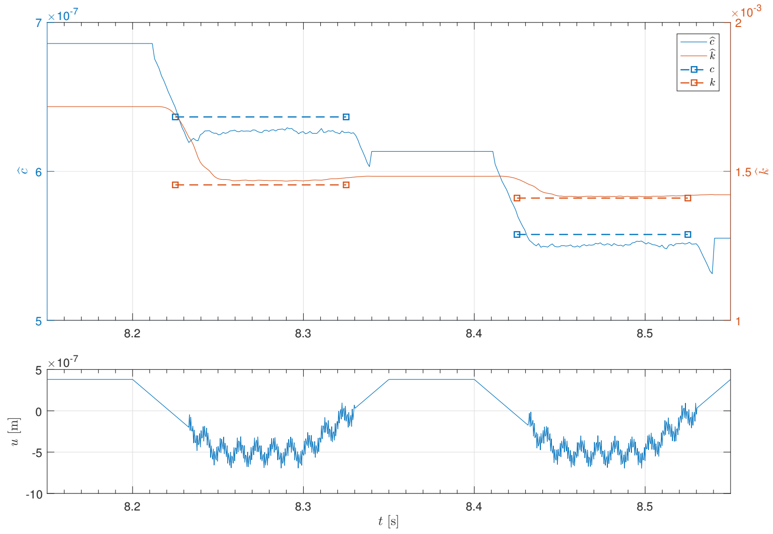

Figure C.43: Experiment 4, Job 10 (table 7.11): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

C6 Changing the Spring Constant In the Gain

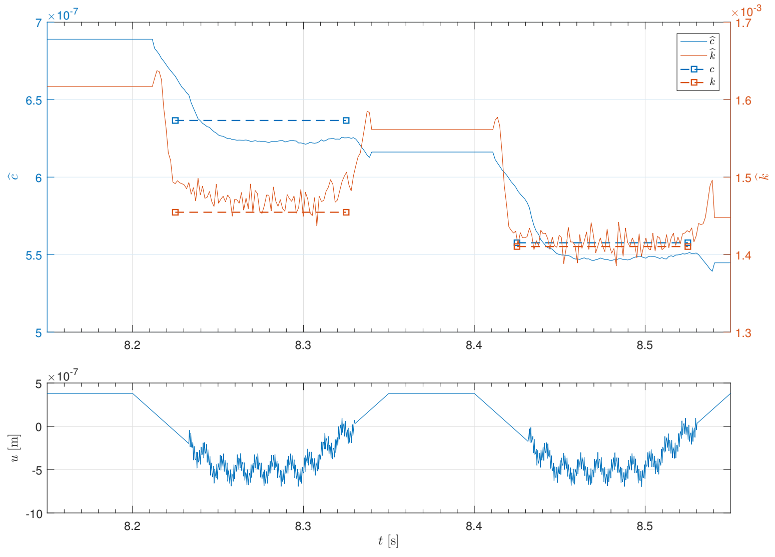

Figure C.44: Experiment 5, Job 1 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

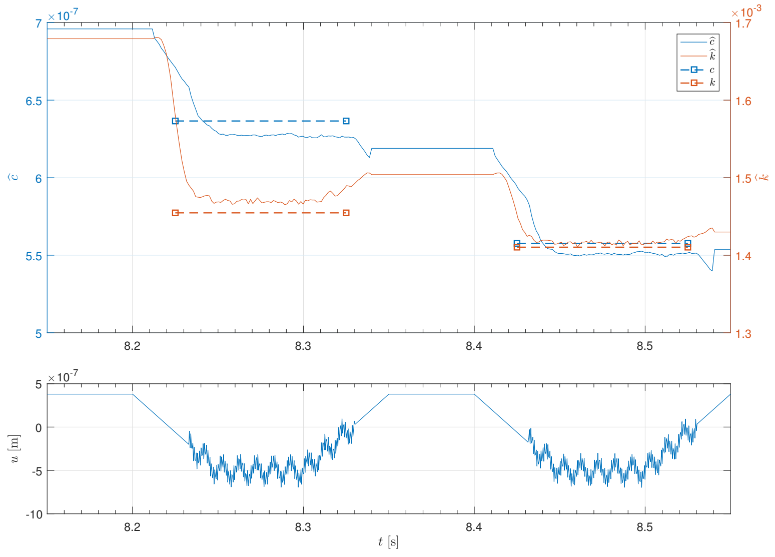

Figure C.45: Experiment 5, Job 2 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

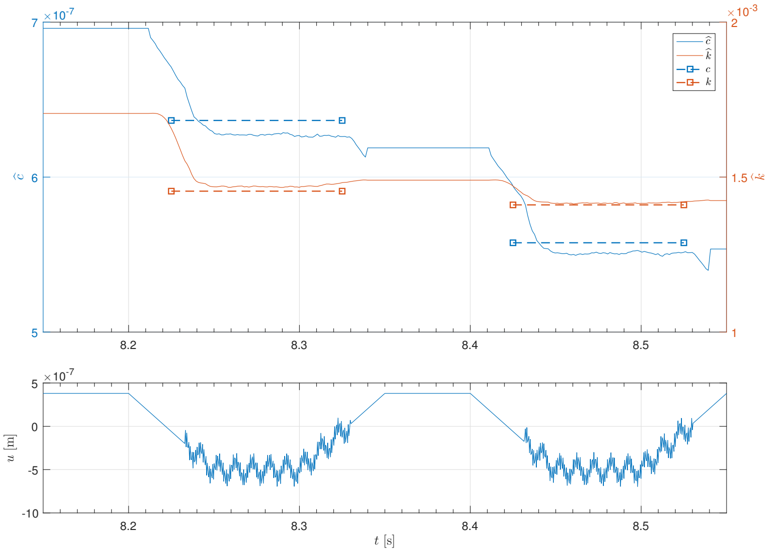

Figure C.46: Experiment 5, Job 3 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

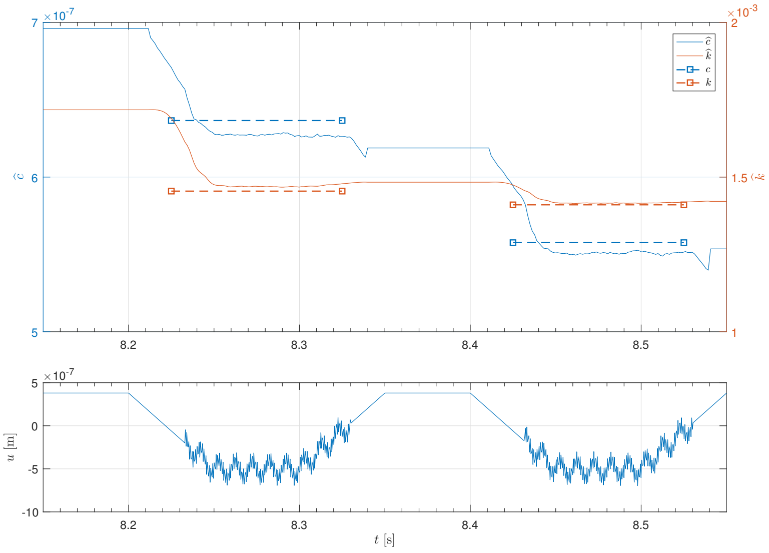

Figure C.47: Experiment 5, Job 4 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

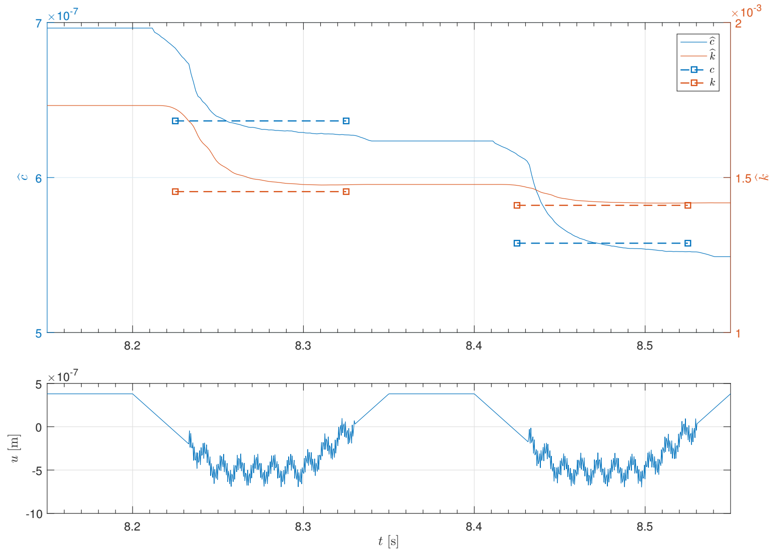

Figure C.48: Experiment 5, Job 5 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

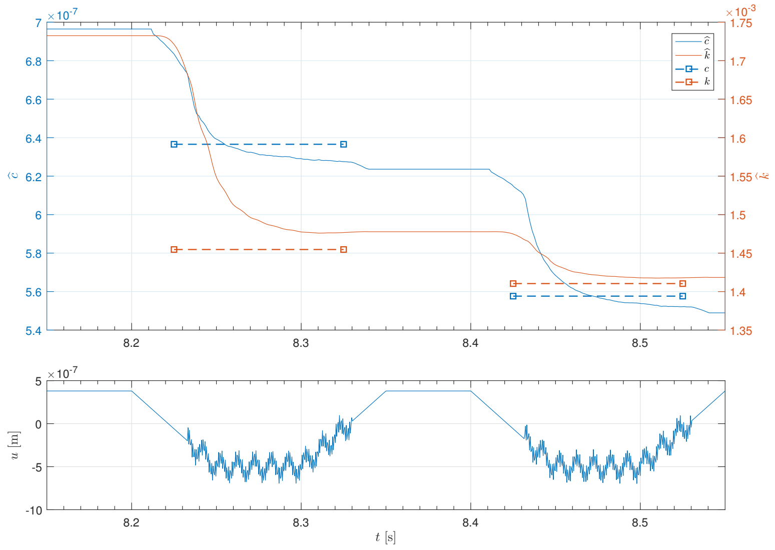

Figure C.49: Experiment 5, Job 6 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

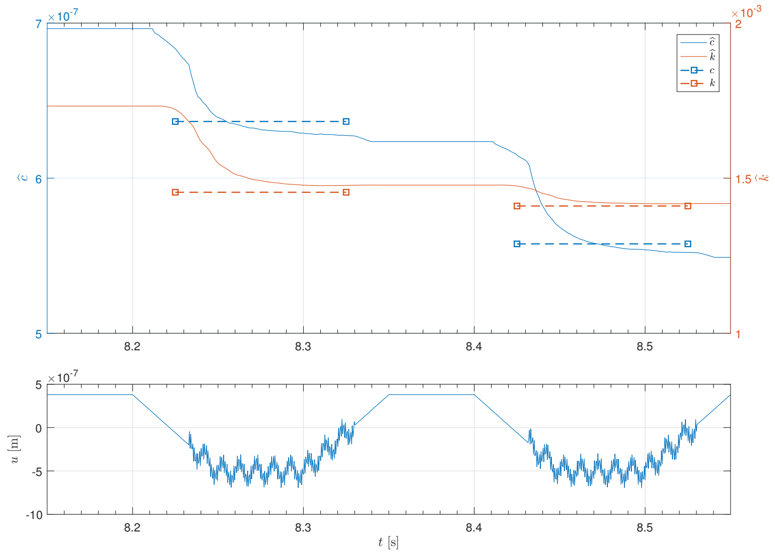

Figure C.50: Experiment 5, Job 7 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

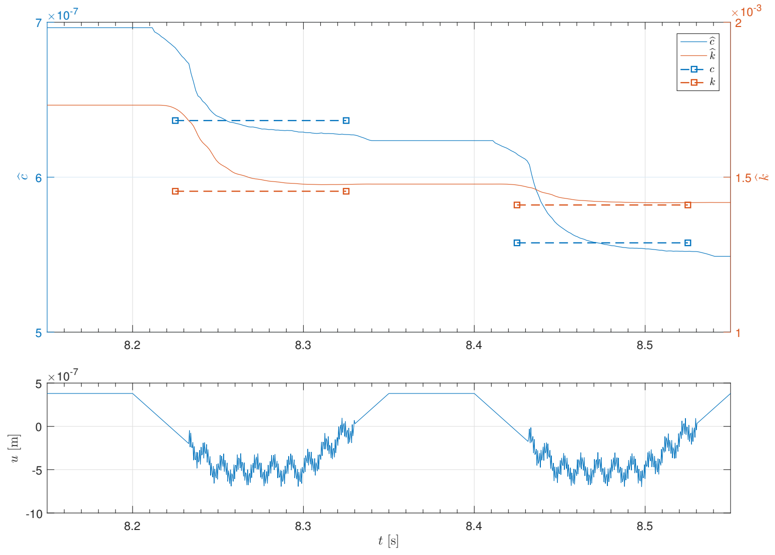

Figure C.51: Experiment 5, Job 8 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

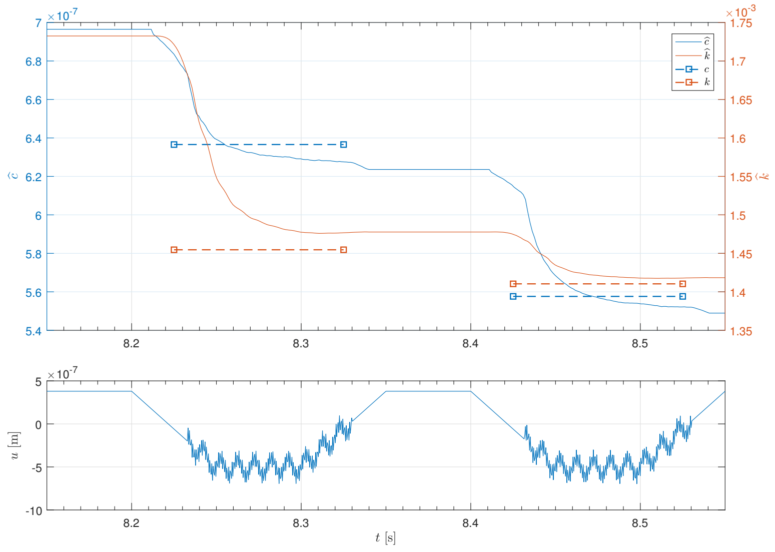

Figure C.52: Experiment 5, Job 9 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

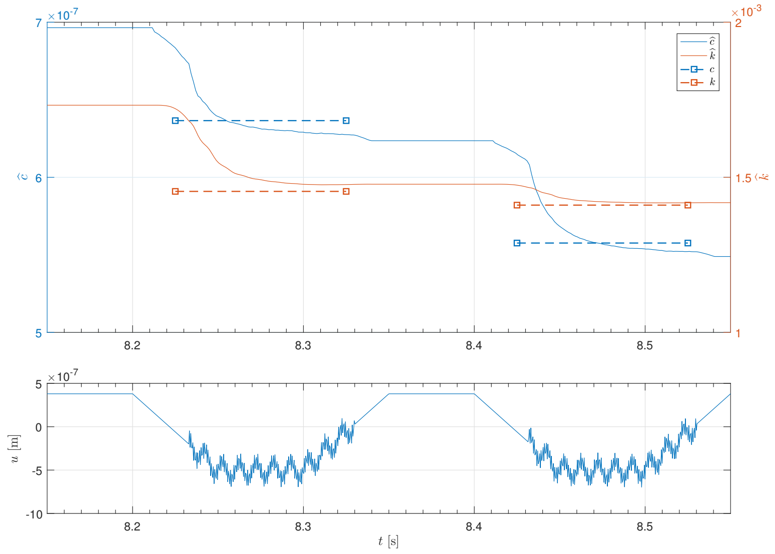

Figure C.53: Experiment 5, Job 10 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

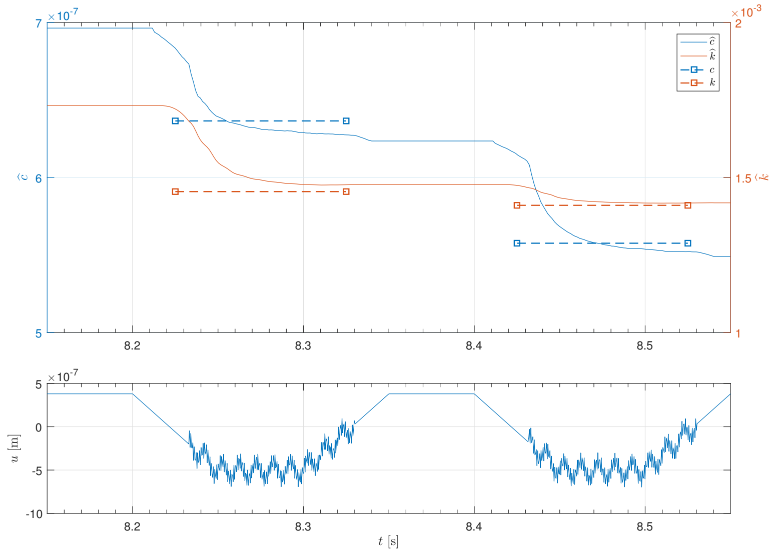

Figure C.54: Experiment 5, Job 11 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

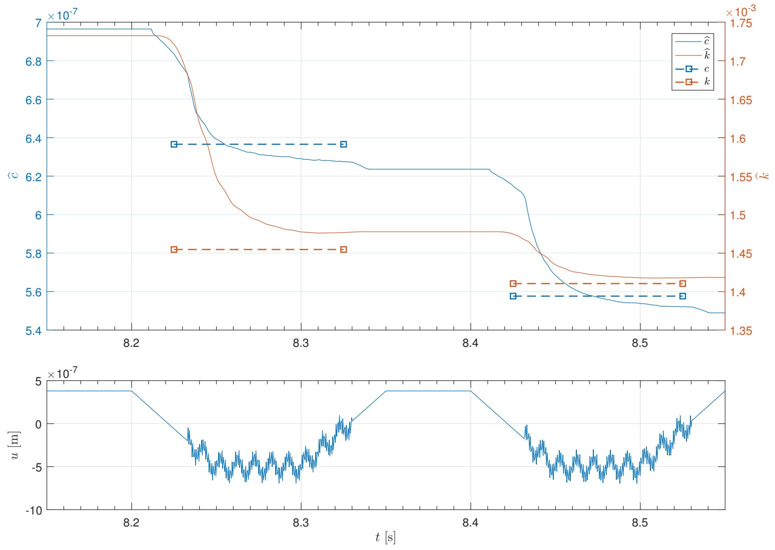

Figure C.55: Experiment 5, Job 12 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

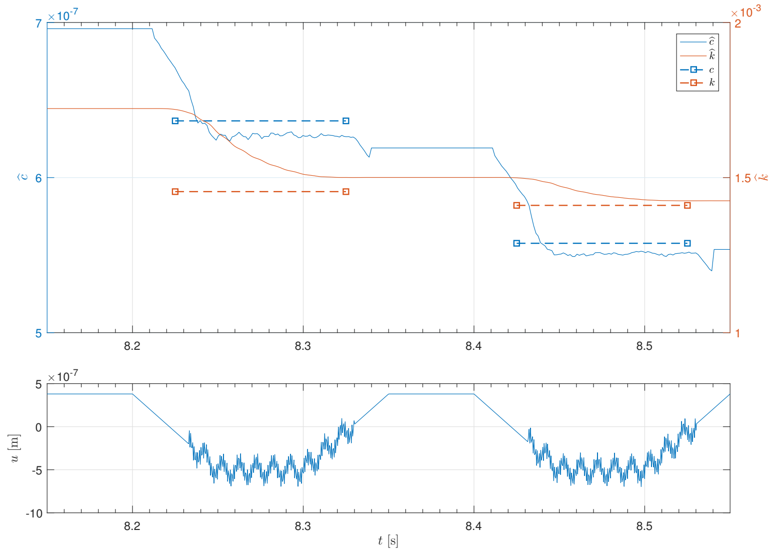

Figure C.56: Experiment 5, Job 1 (table 7.13): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

C7 Optimized Gain Matrix with Different Tunable Filter Constants

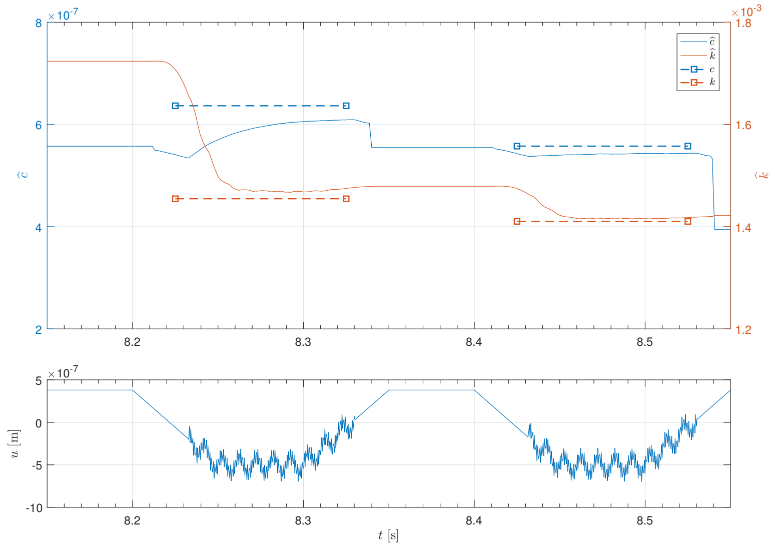

Figure C.57: Experiment 6, Job 1 (table 7.15): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

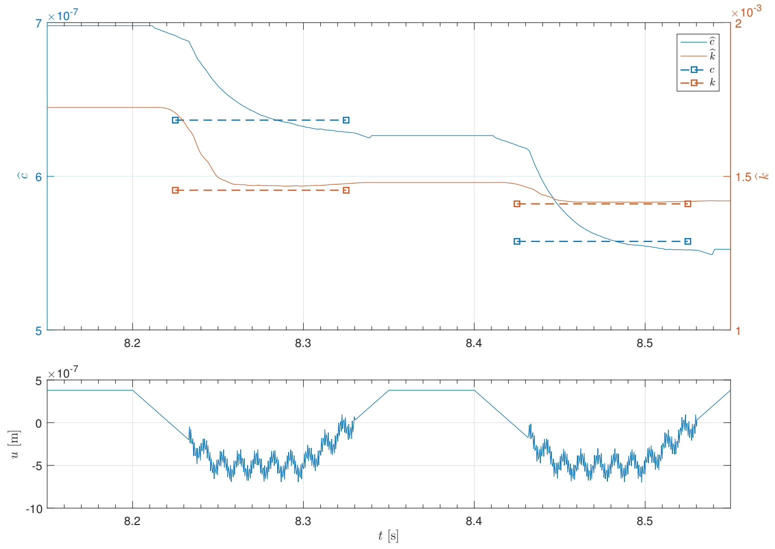

Figure C.58: Experiment 6, Job 2 (table 7.15): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

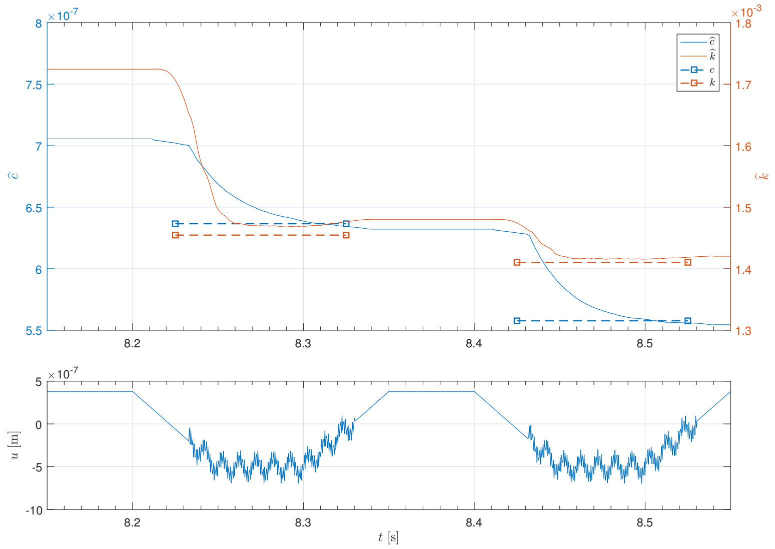

Figure C.59: Experiment 6, Job 3 (table 7.15): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

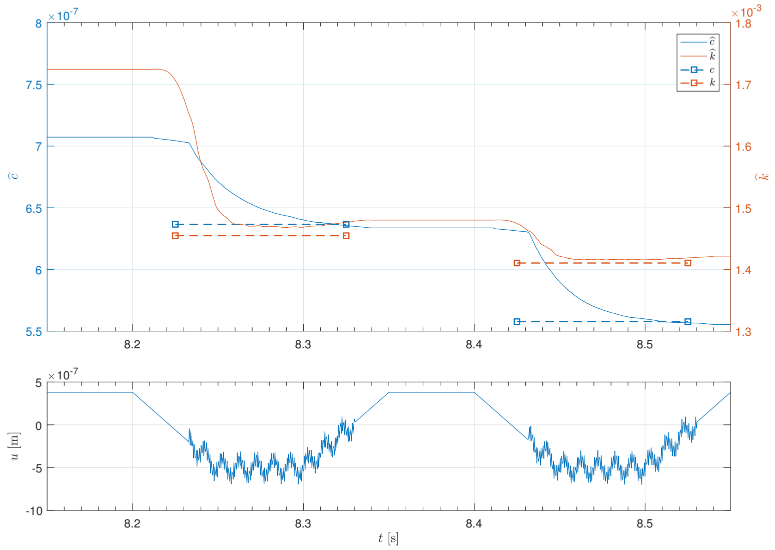

Figure C.60: Experiment 6, Job 4 (table 7.15): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

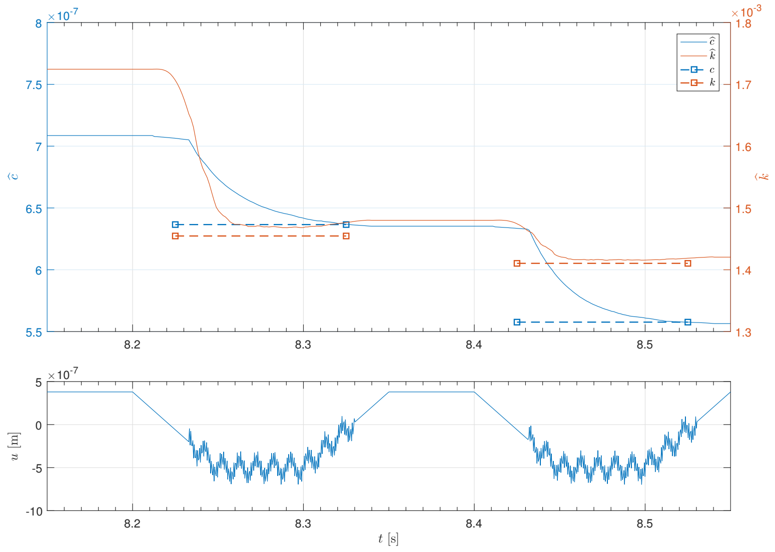

Figure C.61: Experiment 6, Job 5 (table 7.15): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

C8 Optimized Gain Matrix with Different Simulation Solvers

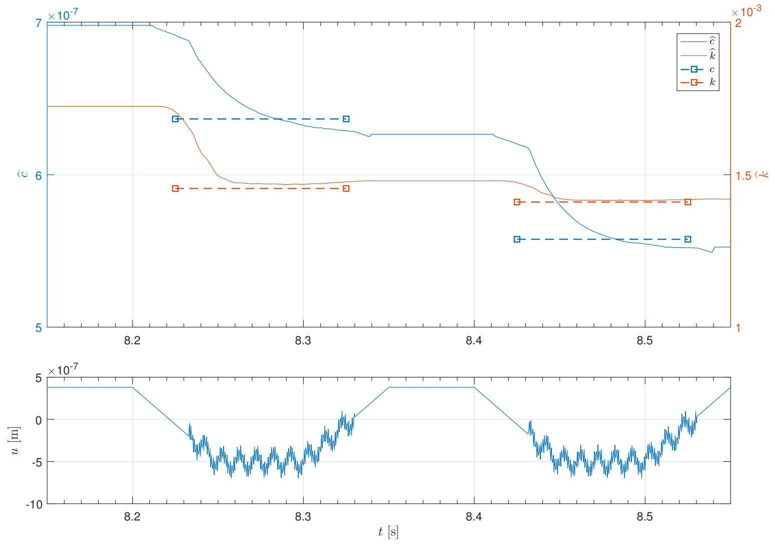

Figure C.62: Experiment 7, Job 1 (table 7.17): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

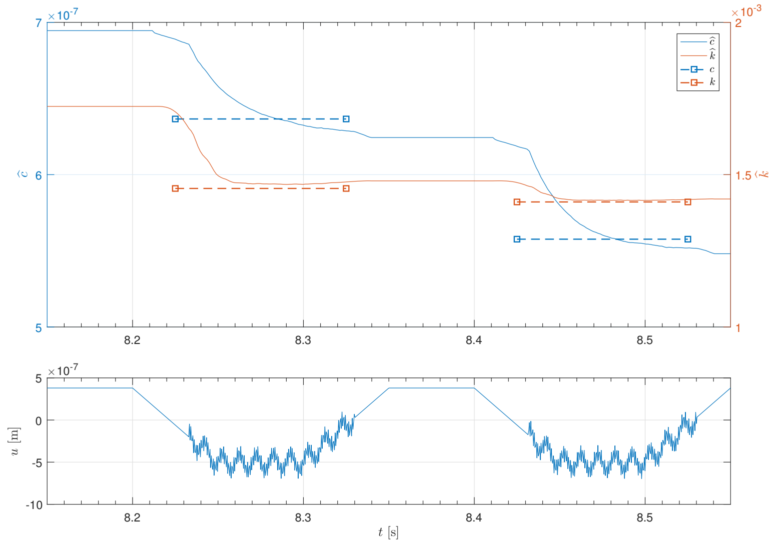

Figure C.63: Experiment 7, Job 2 (table 7.17): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

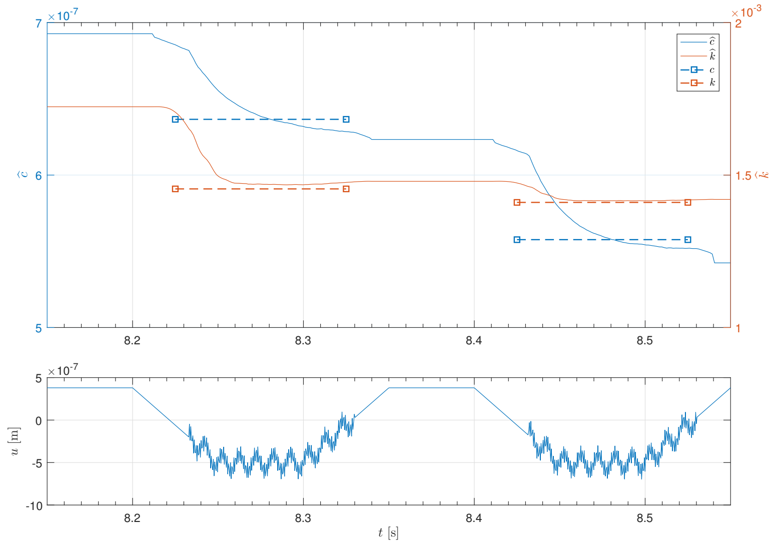

Figure C.64: Experiment 7, Job 3 (table 7.17): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

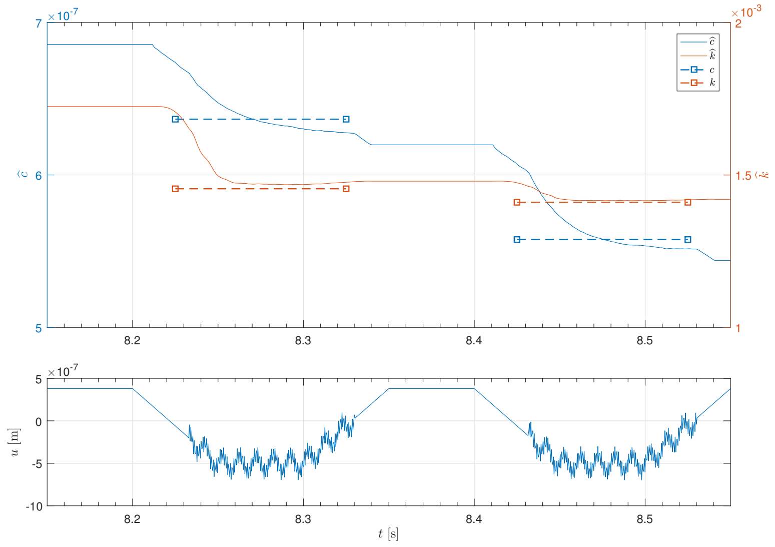

Figure C.65: Experiment 7, Job 4 (table 7.17): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

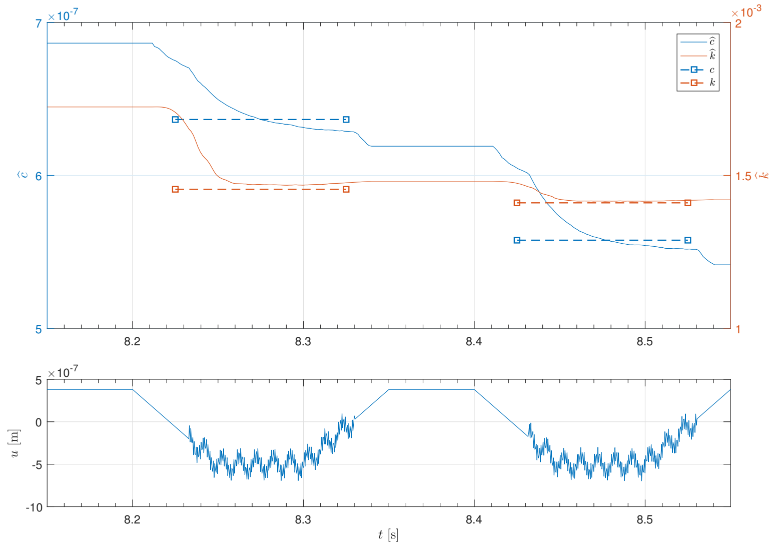

Figure C.66: Experiment 7, Job 5 (table 7.17): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.

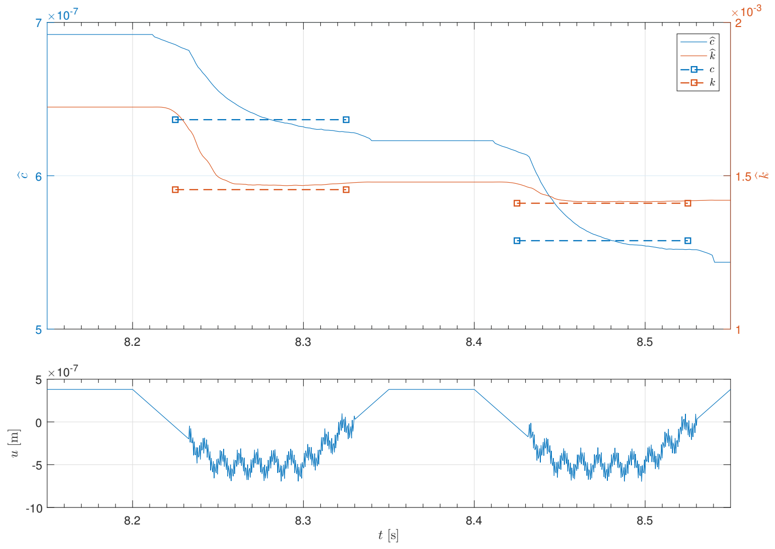

Figure C.67: Experiment 7, Job 6 (table 7.17): The estimates of $\widehat{k}$ and $\widehat{c}$ using the gradient method using instantaneous cost, and the cantilever position input u over time.