Master's thesis

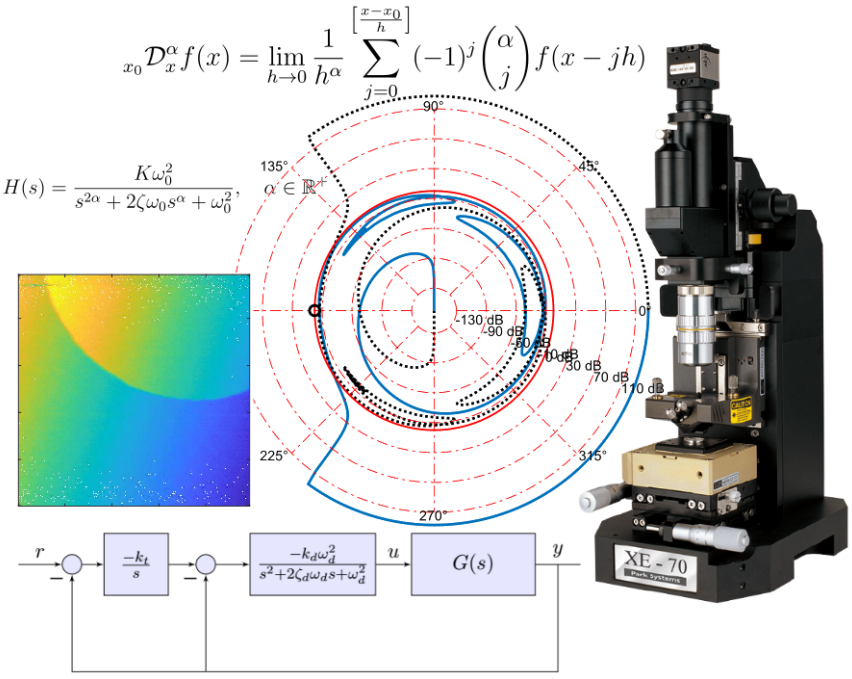
NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Andreas Hanssen Moltumyr

Control in Atomic Force Microscopy: A Fractional Order Approach

Fractional-order positive position feedback in nanopositioning

Master's thesis in Engineering Cybernetics
Supervisor: Jan Tommy Gravdahl, ITK
Co-supervisor: Michael Remo Palmén Ragazzon, ITK
June 2019



NTNU
Norwegian University of Science and Technology

Andreas Hanssen Moltumyr

Control in Atomic Force Microscopy: A Fractional Order Approach

Fractional-order Positive Position Feedback in Nanopositioning, Fractional-order Logarithmic Nyquist Diagram

Master's thesis in Engineering Cybernetics
Supervisor: Jan Tommy Gravdahl, ITK
Co-supervisor: Michael Remo Palmén Ragazzon, ITK
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

In memory of Anna Laura Fredheim Hanssen

Abstract

This thesis is devoted to the investigation of fractional-order control in nanopositioning. The main goal being to test and find out if the theory of fractional calculus could be used to improve tracking performance for nanopositioning systems. To this end, a set of fractional-order controllers have been designed and implemented for the control of the piezoelectric actuated lateral positioning system of an Atomic Force Microscope (AFM). An AFM being a system which can capture the topology of surfaces at the micro- to nanometer scale.

The well-known PID controller and the less known Positive Position Feedback (PPF) controller with tracking, have been used as a basis for this study and have been augmented with fractional-order integrals and derivatives, and tuned with an optimization based, experimental method. The experimental tuning method uses the genetic algorithm, a heuristic optimization method, to find good controller parameters, while ensuring stability through automatic evaluation of the well-known Nyquist stability criterion. The method has been termed experimental because it is not known to the author whether the Nyquist criterion is valid for fractional-order transfer functions or not. Despite this uncertainty, the method has shown promise and has been able to optimize both fractional-order and integer-order PID and PPF controllers. The resulting controllers have been tested both in simulations with MATLAB, and experimentally on a commercial AFM system with the help of MATLAB and dSpace. Oustaloup filter approximations have been used for the realization of fractional-order integrals and derivatives in the controllers.

Results show that PPF controllers with integral tracking is much better at damping the resonance modes than standard PID control and can achieve higher bandwidth. The results also indicate that fractional-order integral tracking with a fractional-order between one and two can remove steady state error when tracking ramp-like signals. On the other hand, a fractional-order below one leads to a steady state error that increases slowly with time. Apart from these observations, the introduction of fractional-orders into the controllers cannot be said to have increased the tracking performance much, when compared to their regular integer-order variants.

In addition to the main results, a MATLAB function for the plotting of logarithmic Nyquist diagrams for fractional-order transfer functions has been created. This function can prove invaluable for stability analysis of fractional-order transfer functions given that the Nyquist criterion is valid for such systems.

Sammendrag

Denne avhandlingen er viet til undersøkelsen av fraksjonell-ordens kontroll innen nanoposisjonering. Hovedmålet har vært å teste og finne ut om teorien bak fraksjonell kalkulus kan bli brukt til å forbedre følging av referansesignaler i nanoposisjoneringssystemer. Til dette formål har et sett med fraksjonell-ordens kontrollere blitt utviklet og implementert for å kontrollere det horisontale, piezoelektrisk styrte posisjoneringstrinnet til et atomkraftmikroskop (AFM). Hvor et atomkraftmikroskop er et system som kan måle topologien til en overflate på mikro- til nanometer nivå.

Den svært kjente PID kontrolleren og den mindre kjente positiv-posisjons-tilbakekoblings (PPF) kontrolleren med referanseføling, har blitt brukt som basis for denne studien og har blitt supplert med fraksjonell-ordens integrasjon og derivasjon, og tuning med en optimaliseringsbasert, eksperimentell metode. Den eksperimentelle tuningsmetoden bruker en genetisk algoritme, en heuristisk optimaliseringsmetode, til å finne gode kontrollparametere, samtidig som stabilitet er sikret gjennom automatisk evaluering av det godt kjente stabilitetskriteriet til Nyquist. Metoden er betegnet som eksperimentell siden det ikke er kjent for forfatteren om Nyquist sitt stabilitetskriterium er gyldig for fraksjonell-ordens transfer funksjoner eller ikke. Likevel, på tross av denne usikkerheten har metoden vist lovende resultater og har vært i stand til å optimalisere både fraksjonell-ordens og heltall-ordens PID og PPF kontrollere. De resulterende kontrollerene har blitt testet både gjennom simulering med MATLAB, og eksperimentelt på et kommersielt AFM system med hjelp av MATLAB og dSpace. Oustaloup filter-approksimasjoner har blitt brukt for å realisere de fraksjonell-ordens integrasjonene og derivasjonene i kontrollerene.

Resultater viser at PPF kontrollerene med integral følging av referanser er mye bedre til å dempe resonanstoppene enn standard PID kontroll, og kan oppnå høyere båndbredde. Resultatene indikerer også at fraksjonell-ordens integral følging med en fraksjonell-orden mellom en og to er i stand til å fjerne stasjonært avvik ved følging av rampelignende signaler. På den annen side vil en fraksjonell-orden under en lede til et stasjonært avvik som øker sakte med tiden. Bortsett ifra disse observasjonene går det ikke an å si at introduksjonen av fraksjonell-orden i kontrollerene har ført til forbedret følging av referansesignaler, når en sammenligner med de vanlige variantene med heltallsorden.

I tillegg til hovedresultatene, kan det legges til at en MATLAB funksjon for plotting av et logaritmisk Nyquist diagram for fraksjonell-ordens transferfunksjoner har blitt produsert. Denne funksjonen kan vise seg å bli verdifull for stabilitetsanalyse av fraksjonell-ordens transferfunksjoner hvis det viser seg at Nyquists stabilitetskriterium holder for slike systemer.

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

MSc thesis assignment

Name of the candidate: Andreas Moltumyr
Subject: Engineering Cybernetics
Title: Control in atomic force microscopy: a fractional order approach

Background:

Something about fractional systems, nanopositioning, afm, projectwork

Tasks:

1. Literature study on fractional order systems (include highlights from [1])
2. Literature study on tracking of raster patterns, with focus on AFM application
3. FO controller design for tracking of raster patterns taking time delay into account
4. Perform experiments with the control design from 3., and compare with the standard solution, on the positioner in the AFM. Compare both tracking performance in the time domain, and quality of acquired AFM images
5. If time, investigate other optimization methods than [1] for System ID.
6. Depending on progress, aim for writing a scientific paper, based on items 1-4 above

To be handed in by: 27/6-2019
Co-supervisor: Dr. Michael Ragazzon

Jan Tommy Gravdahl
Professor, supervisor

[1] A. Moltumyr, project work, ITK, NTNU, 2018

Figure 1: MSc assignment text.

Preface

I remember back at high school where I first learnt the basics of calculus. We were told that the square root of negative numbers did not exist. Starting life at university, this "fact" was crushed with the introduction of complex numbers, and I remember a tingling feeling of going beyond the known. Looking back, I realize that my teachers in high school were only protecting us from some of the complexity of mathematics. Anyway, one year ago, when I was introduced to the idea of non-integer derivatives and integrals in the last year of my msc. degree, I got the same sensation of going beyond the known. So, I delved into this special branch of mathematics without a second thought.

After one year, looking at the theory of fractional calculus, I'm still contemplating whether there is something society can really gain from this peculiar mathematical theory. Maybe it is just a way to overcomplicate things. Only the future knows...

I would especially like to thank Prof. Jan Tommy Gravdahl for the regular meetings, continued support and for supplying me with a fully fledged and already set up AFM nano-positioning lab, so that I could get right to the task at hand. I would also like to give special thanks Post Doc. Michael R. P. Ragazzon for lots of help and advice in addition to introduction sessions to the AFM setup at the nano-lab at ITK.

Big thanks to my fellow students and friends around Trondheim. You have all contributed to making my life at NTNU interesting and enjoyable. Special thanks to my friends from Malvik who have kept up with me, despite my many bad jokes, kept me sane and given me much needed breaks from the studies.

Lastly, I would like to thank my parents, my siblings and the rest of my family for granting me a good childhood and a safe foundation that have enabled me to stretch out and reach where I am today.

Andreas Hanssen Moltumyr
Trondheim, June 2019

Contents

Nomenclature

u	Plant input
y	Plant output
r	Reference signal
$\mathcal{D}^{\alpha}(\cdot)$	Fractional-order derivative or integral
${}_{x_0}\mathcal{D}_x^{\alpha}(\cdot)$	Fractional-order derivative or integral starting at x_0 and running to x
$f^{(n)}(\cdot)$	n-th derivative of a function $f(\cdot)$
$\lceil \cdot \rceil$	Round up to nearest integer (Ceiling)
z^j	Time shift operator
$\binom{\alpha}{\beta}$	Binomial coefficient
$\Gamma(\alpha)$	Gamma Function
$(\gamma)_k$	Rising factorial
$\mathcal{L}\{\cdot\}$	Laplace transform
$\mathcal{E}_{\alpha,\beta}^{\gamma}(z)$	Mittag-Leffler function with three variables
j	Imaginary unit
$G(s)$	Plant model transfer function
$C(s)$	Controller transfer function
$L(s)$	Open-loop transfer function
$S(s)$	Sensitivity transfer function
$T(s)$	Complementary sensitivity transfer function
BW	Bandwidth
$\|\|S(s)\|\|_{\infty}$	Maximum value of sensitivity function
ISE	Integral of Squared Error
Ψ	Phase margin
Δk	Gain margin
SPM	Scanning Probe Microscopy
STM	Scanning Tunnelling Microscopy
AFM	Atomic Force Microscopy
PSPD	Position Sensitive Photo Detector
PZT	Lead Zirconate Titanate

GL	Grünwald-Letnikov
RL	Riemann-Liouville
rhp	right half plane
LTI	Linear Time-Invariant
PID	Proportional Integral Derivative
PPF	Positive Position Feedback
FO	Fractional-Order
FOS	Fractional-Order System
FOCS	Fractional-order Control System
FO-PID	Fractional-Order PID
FO-PPF	Fractional-Order PPF
IO-PID	Integer-order PID
IO-PPF	Integer-order PPF

Chapter 1

Introduction

Scanning probe microscopy (SPM) is a set of techniques that have enabled the human species to observe details on a sub-nano meter level. Using thin probes that can be precisely positioned through nanopositioning techniques and moved around in a scanning pattern, the bane of visible light microscopes, namely the diffraction limit, have been surpassed. SPM techniques have therefore played a part in allowing the human species to make new discoveries in the field of chemistry and molecular biology, while at the same time continue the miniaturization of electronics. Which in turn have revolutionized the world of mankind and brought on the Age of Information.

The most renowned SPM techniques are scanning tunnelling microscopy (STM) and atomic force microscopy (AFM). They both make active use of techniques like feedback control during interrogations of surface properties and differs only in the physical way this is done. While the STM makes use of the concept of quantum tunnelling to measure surface topology the AFM technique can measures surface topology through the physical interaction forces experienced between probe tip and the surface of a sample.

Compared to other microscopy techniques like the broad class of optical microscopy techniques, the SPMs do have several drawbacks. One of them is the total time it takes to acquire a good quality image, which can be on the order of minutes. These long image acquisition times is a result of the limitations in scanning speed, which again is limited by the inherent properties of the mechanical system, like low damping and adverse vibrational dynamics. Because of these dynamical properties, feedback control has become a vital and necessary part of high-performance SPM systems, and simple controllers perform quite well.

However, being humans, we get tired of waiting, and image acquisition times on the order of minutes are deemed too long. As a result, the scientific community have performed several investigations on the topic of SPM and control, with the hopes of being able to speed up the image acquisition times. This thesis joins the investigations on SPM and control with an attempt at controlling the lateral positioning stage of an AFM system through the use of fractional-order control theory and the theory of fractional-order derivatives and integrals.

Fractional-order control systems (FOCS) is a topic in the field of control which have received increasing interest over the last few years. With the use of mathematical theory from fractional calculus, new types of controllers with fractional-order derivatives and fractional-order integrals can be designed. Opening up a whole new world in the field of control where new kinds of tuning parameters appear. It can also be mentioned that a few systems have been shown to exhibit some form of fractional-order dynamics [1–3], and it is reasonable to think that these systems can be

controlled in a better way with fractional-order control systems than with integer-order control systems. Historically, fractional derivatives and integrals dates back to a letter correspondence between Leibniz and L'Hopital in 1695 [4]. So, the thought of fractional calculus and the earliest mathematical results is starting to get quite old.

In traditional control theory, only integer-order derivatives and integer-order integrals have been used. Looking back at the vast number of different controllers and applications that have sprung fourth, one can conclude that the mathematical theory of integer-order calculus have been sufficient in case of control. And no real need for fractional-order derivatives and integrals on a general level have existed. Nevertheless, research into fractional-order control systems have been on the rise. Some of the reasons why the theory of fractional calculus has lied somewhat undisturbed by the control community up until now can probably be explained by the theory's inherent complexity and the fact that there have always been other promising methods or techniques to pursue. The emergence of computers and the steadily increase in available processing power have also made it easier to do research into fractional-order control systems.

There are still a lot of uncharted territory when it comes to the use of fractional-order control systems in the literature. From the authors point of view, it is still not clear if the theory of fractional-order control systems has anything to offer when compared to integer-order control systems, when thinking of fast, accurate and robust control. In terms of stability analysis of fractional-order systems a few results have emerged. The most widely known revolves around checking the locations of the poles of a so-called commensurate-order transfer function [5]. This technique is however found to have its limits. Apart from that, the author misses a good treatment concerning the possible validity and use of Nyquist's stability criterion for fractional-order systems in the literature. If the Nyquist criterion is valid for fractional-order systems it would enable easier stability analysis of feedback control for fractional-order systems, making it easier to design fractional-order controllers.

In this thesis the author's work on developing and testing a set of fractional-order controllers for the control of the lateral axes of an AFM system is presented. The main goal is to investigate if fractional-order control can allow for higher scanning frequencies and shorter image acquisition time than with the use of integer-order control. The idea is that a fractional-order controller can have better damping properties, allowing for a higher closed-loop bandwidth. Another goal is to compare integer-order control and fractional-order control and see if there is anything to gain from introducing fractional-order derivatives and integrals into control. What are the costs of using fractional-order control? For the community to best direct its efforts, there is a need to know.

To this end, two integer-order controllers have been chosen as a basis for the study. These are the proportional integral derivative (PID) controller and the positive position feedback (PPF) controller with integral tracking. By switching out the regular integer-order derivatives and integrals in these controllers with fractional-order derivatives and integrals, we can create fractional-order PID (FO-PID) controllers and fractional-order PPF (FO-PPF) controllers. One FO-PID controller and three different FO-PPF controllers will be studied and compared with the regular integer-order PID controller and the regular PPF controller, which we respectively can abbreviate to IO-PID and IO-PPF. Testing and comparisons of the controllers will be done both in simulations and experimentally on the lateral nanopositioner of an AFM system.

Tuning of fractional-order controllers turns out to be a challenging task. Maybe more so than for regular integer-order controllers, given the added parameters and the not so simple stability analysis. Controllers like the FO-PID controller have been studied quite extensively in the

literature, and as a result, a few tuning guidelines or tuning rules of thumb exists for this controller [6]. Apart from that, optimization is the most frequent method for tuning of fractional-order controllers. As for stability analysis of fractional-order systems, the most frequent method is to check the locations of the poles as mentioned earlier. This can however lead to regular polynomials of an immensely high order, of which, finding the roots is no trivial task.

Therefore, in order to tune the FO-PID and FO-PPF controllers, as well as the IO-PID and IO-PPF controllers, an optimization based tuning algorithm for fractional-order controllers have been developed with MATLAB. The method uses the Genetic Algorithm, a heuristic population-based optimization algorithm to search for good regulator parameters based on a given objectivity function and a model of the plant. A purely frequency domain based objectivity function is used. To make sure that the regulator parameters will lead to a stable closed loop system, an experimental Nyquist criterion based algorithm to check stability for fractional-order systems have been developed. This stability algorithm is used as a constraint for the optimization problem.

To support the development with the tuning method a function for the plotting of an experimental logarithmic Nyquist diagram for fractional-order systems was developed. The plot can be used for visualizing stability for closed-loop fractional-order systems. The idea and inspiration for such a plot came from the logarithmic-amplitude polar diagram developed for regular integer-order systems by Trond Andresen [7]. Using a logarithmic axis for the magnitude in the plot, the whole frequency response curve can be inspected without the need to zoom in and out, simplifying the job of visually checking stability.

In addition to the design and testing of fractional-order controllers on the AFM system, a stability investigation of a simple fractional-order system with two fractional-order poles will be presented. A stability conjecture related to this system will be stated in addition to a demonstration of the conjecture. Hopefully the reader will find this interesting.

It should be noted that the general validity of the stability analysis part of the tuning method presented in this thesis have not been proven to work on a general basis. The author has not been successful in locating strong evidence either in favour of or against the general validity of the Nyquist criterion in the literature. The author urges therefore the wider control community to study this problem, so that the stability of fractional order systems can be determined conclusively, including those proposed in this work.

1.1 Outline

This thesis is organized as follows. In chapter 2 the principle of Atomic Force Microscopy (AFM) is introduced and explained.

Secondly, in chapter 3 we will review some of the mathematical theory in the field of fractional-order systems and control. The most used and best known definitions of fractional derivatives and integrals are given. Fractional-order differential equations and fractional-order transfer functions will be introduced and some results linking the s-domain to the time domain will be given.

In chapter 4 we will take a look at the frequency response and stability of a type of fractional-order systems. Two simple fractional-order systems and some of their properties will be discussed. A stability conjecture is presented and accompanied by a lengthy demonstration motivating the conjecture.

In chapter 5 the proposed tuning method for fractional-order controllers is explained in more detail. Starting with system identification of the plant that should be controlled and moving on to an explanation of the fractional-order controllers that will be studied. The use of the genetic algorithm for parameter optimization is explained, before the optimization problem is formulated. At the end of the chapter the method used for stability assessment with automatic evaluation of the Nyquist criterion is explained with code snippets, before the implemented Logarithmic-amplitude polar diagram and the custom script for generation of AFM images from raw data are presented in the last two sections of the chapter.

In chapter 6 the results from the simulation testing of the different controllers are presented. Performance criteria, step response plots, bode diagrams and logarithmic Nyquist diagrams are all used in an attempt at comparing the performance of the different controllers.

In chapter 7 the results from the experimental testing of different fractional-order controllers on the lateral axes of the commercial AFM system is presented. AFM images and trajectory plots are shown in order to presented the performance of the different controllers. A comparison between the theorized and experimental closed-loop frequency responses are also shown.

Finally the results and the method are discussed in chapter 8. Before conclusions from the work are drawn in chapter 9.

1.2 Contributions

1. A collection of useful mathematics from the field of fractional-order systems and control that should be of value for people new to the field.
2. Experimental results on the subject of using fractional-order controllers to control the lateral motion of an AFM system.
3. A comparison between fractional-order and integer-order controllers.
4. An adaptive step method for fast frequency response calculations of fractional-order and integer-order transfer functions (`astep_fotf_freqresp.m`).
5. An experimental method for tuning of fractional-order controllers based on genetic algorithm optimization and Nyquist's stability theorem.
6. A method for automatic stability assessment calculations of closed-loop systems based on Nyquist stability criterion (`assess_stability.m`).
7. A MATLAB function for the plotting of a logarithmic-amplitude polar diagram for fractional-order systems as well as integer-order systems (`nyqlog_fotf.m`). Useful in stability assessment and to get a feel for some of the differences between integer- and non-integer-order linear systems.
8. A custom function for the generation of AFM images from raw X, Y and Z scan data (`print_AFM_image.m`).

1.3 Note to Assignment Text

See figure 1 for assignment text. The focus of the candidate throughout the work with this MSc thesis have been to test out fractional-order controllers on an Atomic Force Microscope (AFM) system. The design and implementation of such FO controllers did however turn out to be more challenging than first expected. Much time and effort were therefore invested in the development of an optimization-based method for FO controller tuning that would yield stable controllers. As a result of this, task (2.) and (5.) were not treated, while task (6.) were pushed into the future after a conversation with supervisor. Time delay was not studied in (3.).

This lead, however, to the development of an experimental method for the tuning of fractional-order controllers, a fast and efficient script for the calculation of frequency responses for fractional-order transfer functions, as well as integer-order ones, and an experimental function for the plotting of logarithmic-amplitude polar diagrams for fractional-order transfer functions. In addition to the treatment of task (1.), (3.) and (4.).

1.4 Notation

In this thesis we will use the notion fractional-order and non-integer-order somewhat interchangeably, to mean derivatives or integrals of any real numbered order ($\alpha \in \mathbb{R}$). The term fractional-order is most likely a traditional name from the mathematical branch called fractional calculus, where historically, a derivative of fractional order, i.e. $\alpha = \frac{a}{b}$, were considered first. Later the study of irrational-orders where added, making the original name somewhat imprecise and obsolete.

The term fractional polynomial will be used in this thesis to refer to a polynomial on the form

$$p(s) = a_n s^{\alpha_n} + a_{n-1} s^{\alpha_{n-1}} + \ldots + a_1 s^{\alpha_1} + a_0 s^{\alpha_0}. \tag{1.1}$$

In [5] the name pseudo-polynomial has been used for such polynomials. The name fractional-pole and fractional-zero will be used for fractional-polynomials on the form

$$p(s) = (s^{\alpha} - a). \tag{1.2}$$

A fractional-polynomial on the form (1.2) is called a fractional-pole if it lies in the denominator of a fractional-order transfer function and will be called a fractional-zero if it lies in the nominator.

Chapter 2

Microscopy and the Atomic Force Microscope

This chapter is mainly devoted to the introduction and explanation of the Atomic Force Microscope. An interesting and inspiring piece of engineering. However, we will start the chapter by lightly discussing some of the other microscopy techniques like optical, electron and scanning tunnelling microscope in section 2.1. Mentioning some of the drawbacks of the different techniques and motivate the emergence and need for atomic force microscopes. Then, AFM will be presented and explained in 2.2, 2.3 and 2.4. The chapter is ended with an example scan conducted with a Park Systems XE-70 AFM in section 2.5.

2.1 Short Overview of Microscopy

The invention of the optical microscope around 1590 has had a huge impact on how we perceive the world today. The ability to suddenly see things that could not be seen with the naked eye, sparked light in the life and physical sciences. The field of microscopy revolutionized a lot of fields like anatomy and biology, while seeding the seed of life for a lot of new sciences, like histology, microbiology and medicine, to mention a few. The optical microscope has, since its discovery, been continuously improved. Today you find light microscopes with a magnification of up to 1000 times. Above this magnification, light diffraction start to appear, and the consequence of light diffraction is that magnifications above the limit of 1000 times (given visible light with a bandwidth of about 380nm to 740nm) will not increase the amount of details that can be measured or seen. Abbe diffraction limit [8, 9] says that the minimum resolvable distance, given visible light, is

$$d = \frac{\lambda}{2\mathrm{NA}}, \tag{2.1}$$

where λ is the wavelength of the light shining through the lenses of the microscope and NA is the numerical aperture of the optical microscope. In other words: d is the minimum distance needed between two objects for the two objects to be perceived as two objects and not one object. To take an example, visible light has wavelengths in the area of 400nm (violet) to 700nm (red), and with a numerical aperture value of 1.0, the minimum resolvable distance is 200nm for violet light and 350nm for red light. So the diffraction limit is effectively setting a lower limit for what can be seen through an optical microscope under visible lighting, no matter how

sophisticated and perfect the lenses are. Using electromagnetic waves with a lower wavelength than visible light, like ultraviolet light and x-rays, makes it possible to reduce the resolvable distance even more, and therefore increase the resolution, but UV and X-ray microscopes are expensive. And the lack of contrast when studying biological organisms and tissues makes them somewhat non ideal and often contrast fluids must be used. The high frequency UV light and X-rays may even damage the sample.

To enable effective observations of details smaller than 200nm several techniques have been developed over the years. In 1933 Ernst Ruska developed what is considered the first electron microscope [10]. A device that utilized the fact that electrons behave as waves with wavelengths smaller than visible light. Effectively making it possible to achieve much higher magnifications than what is possible with visible light. This was the first electron microscope developed which transcended the diffraction limit of visible light. Later, several different electron microscope techniques have been developed. And a resolution better than 50 pico-meter have been achieved with the Scanning Transmission Electron Microscope (STEM) [11]. One drawback of using electrons to view a sample is that a vacuum environment is needed. If not, the electrons would be scattered by the molecules making up the air. An in-depth study of electron microscopes will not be conducted in this thesis, but interested readers should check out [12] and [13].

Another important development in the field of nano-imaging is the Scanning Tunnelling Microscope (STM). The STM was developed in 1981 by Gerd Binnig and Heinrich Röhrer [14], and uses a completely different technique as opposed to the optical and the electron microscopes. The STM uses the quantum mechanical, tunnel effect arising when an extremely thin tip is passed over a surface. The tunnel effect leads to a small current going between the tip and the surface which varies with the distance between the tip and the surface. Measurements of this current can then be used to create images where the pixels of the image is a representation of the height of the sample, measured at different points. The obtainable resolution is so high that individual atoms on the scale of 0.1nm or 1Å (Ångstrøm) can be distinguished. The STM technique shares the same requirement as the electron microscope for a high vacuum environment to function correctly [15]. Another drawback of STM is that the surface needs to be conductive for a tunnelling current to exist. This led to the development of the Atomic Force Microscope, which will be explained in detail in the next section.

In 1986, the Nobel Prize in Physics was awarded and shared between Ernst Ruska, Gerd Binnig and Heinrich Röhrer for their contributions in the field of nano-imaging [16]. Ernst Ruska was awarded the price for the development of the first electron microscope, while Gerd Binnig and Heinrich Röhrer received the prize for the development of the Scanning Tunnelling Microscope.

2.2 Atomic Force Microscopy

The Atomic Force Microscope was invented in 1985 by Gerd Binnig [17], the same person that was instrumental in the development of the STM. The first working version of an AFM system was developed at IBM by Gerd Binnig, Christoph Gerber and Calvin Quate [18] some time later. STM, AFM and other techniques that uses a probe or tip to interrogate or manipulate a surface is collectively called Scanning Probe Microscopy (SPM).

The AFM shares many working principles with the STM, but instead of utilizing the quantum mechanical tunnel effect, the AFM in based on the physical interaction forces between the extremely sharp tip and sample surface. These forces are mainly the van der Waals forces that results in an attractive force between tip and surface, and the electrostatic force that forces

Figure 2.1: Park Systems XE-70 AFM.

the tip and surface apart. These interaction forces can be modelled or approximated by the Lennard-Jones potential [15]

$$F(r) = k_1 \left[-\left(\frac{\sigma}{r}\right)^2 + \frac{1}{30}\left(\frac{\sigma}{r}\right)^8 \right], \tag{2.2}$$

where F is the interaction force between tip and sample and r is the distance between tip and sample. k_1 is a constant depending on the geometry and material of the tip and sample and σ is an interaction parameter.

2.2.1 How does it Work

The general working principle of AFM systems is moving a very thin probe tip, back and forth over a sample surface while measuring the deflection of the arm, or cantilever, holding the probe tip. Then, through measurements of the cantilever deflection and a feedback loop regulating the probe-tip interaction force, the topology of the sample can be found. This process is explained in more detail below. The content of this section is based on [15, 19].

The probe tip, at the end of the cantilever, is usually on the order of several nano meters in diameter and enables detection of texture on the same level. An illustration of the working principle of a XE-70 Park Systems AFM is shown in figure 2.2. It should however be noted that there exist several different mechanical designs for AFM systems, so figure 2.2 will not be representative for all existing AFM systems. AFM system designs varies mostly in terms of actuator types, sensor types and placement of actuators and sensors.

In the middle of figure 2.2 we see the probe tip at the end of the cantilever, which is just a flexible rod. The probe tip hovers slightly above the sample surface in the illustration. It should be stressed that the probe and cantilever is not drawn to scale with the rest of the system. In reality, the cantilever is only a few hundred micrometer in length and the probe tip is usually too small to be seen with the naked eye. The vertical position of the cantilever and probe tip is controlled by a piezoelectric actuator, while the deflection of the cantilever is measured with laser and a Position Sensitive Photo Detector (PSPD).

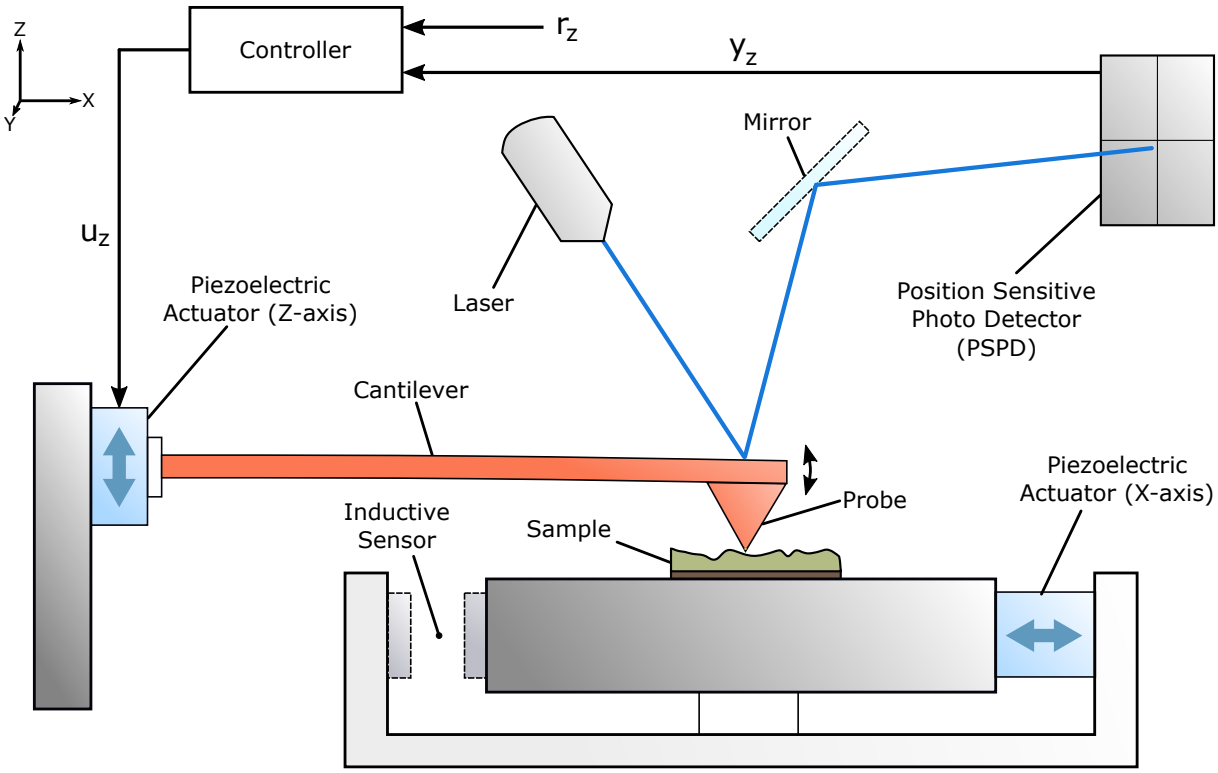


Figure 2.2: Principle illustration of an AFM system. Inspiration taken from [15].

In the AFM configuration shown in 2.2 the vertical positioning (Z-axis) of the cantilever is decoupled from the horizontal positioning (X-axis and Y-axis into the paper plane) of the sample. A point is made of this because this configuration makes it possible to decouple vertical (Z) control from horizontal (X,Y) control. Reducing the complexity of the control system, in addition to lowering signal and noise crosstalk between the axes.

2.2.2 Z-axis: Modes of Operation

Several different operating modes exist for AFM imaging. The two most commonly used modes are called static and dynamic. Static mode is used at the end of this thesis and will therefore be explained here. Dynamic mode, however, is not used and will not be explained here. Interested readers are therefore refereed to [15] for a brief explanation.

2.2.3 Z-axis: Static Mode

Closing the vertical control loop by inputting the PSPD sensor signals y_z to a controller system and feeding the controller output signal u_z to the Z-axis piezoelectric actuator, the interaction force between the tip and surface can be controlled to a given reference value r_z. While the interaction force between tip and surface is kept constant, information about the sample topology can be read from the control signal being feed to the piezoelectric actuator u_z. However, the u_z control signal unit in normally voltage, so an appropriate scaling needs to be done to get topology data in nanometer. This scaling factor is usually found through calibration routines where calibration samples with known topology is scanned.

2.2.4 XY-plane: Scanning Motion

To be able to scan an area of the sample, the sample needs to be moved around in the horizontal plane (XY-plane). To this end, two orthogonal sets of piezoelectric actuators with co-located inductive sensors are often used. Only the piezoelectric actuator and inductive sensor for the x-axis is shown in figure 2.2. Connecting the two sensor-actuator sets (one for X-axis and one for Y-axis) with a feedback loop and a regulator, a 2D position trajectory can be tracked with high precision. Enabling a scanning motion for the probe head relative to the surface. Giving a triangle wave reference for one axis and a staircase reference for the other axis, a simple 2D scanning trajectory like the one below can be generated.

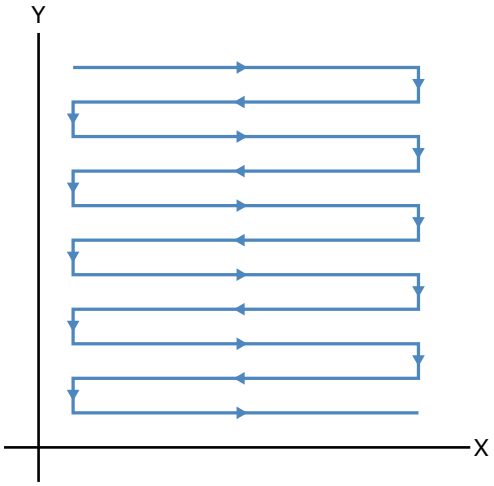


Figure 2.3: Simple 2D scanning trajectory.

2.3 Piezoelectric Actuators

Piezoelectric actuators are the most commonly used actuator in nanopositioning. They can provide frictionless motion, high force and have a high resolution, making them ideal actuators for precise positioning on a micro- to nanometer scale. Piezoelectric actuators can be made from a lot of different materials, where Lead Zirconate Titanate ($Pb[Zr_xTi_{1-x}]O_3$ with $0 \leq x \leq 1$), also known as PZT, is the most common material in use today [20].

Piezoelectric materials have the inherent ability that physical tress generates an electric potential across the material. This is called the direct piezoelectric effect. But there is also the inverse piezoelectric effect that can be utilized to create actuators. By applying a electric potential across the piezoelectric actuator the material will expand or contract proportionally with the applied voltage. The displacement is usually quite small. But by layering the material in so-called piezoelectric stacks the displacement of the material can be increased.

A challenge with piezoelectric actuators is that of creep. Creep is a non-linear effect which can be observed as a slow creeping motion after a voltage step has been applied [21]. An example of creep can be observed in figure 2.4. The figure shows a step response of an AFM system with FO-PPF feedback control. This closed-loop system will be presented in detail later in the thesis. At time equal 0.28 s the controller can no longer hold the reference because the creep effect starts to become too strong and the input voltage signal of the piezo actuator has reached saturation. The effect of creep is influencing the system all the way from the start of the step,

but because of integral action in the controller the controller manages to hold the reference all the way until control signal saturation at time equal 0.28 s.

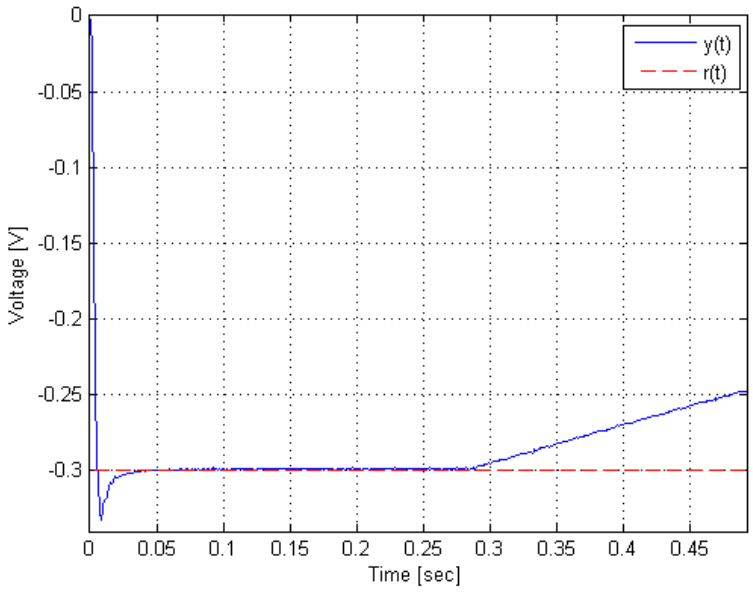


Figure 2.4: Step response of closed-loop system with FO-PPF regulator presented later in the thesis. The effect of creep, due to actuator saturation, can be observed from time equal 0.28 s an onward.

Hysteresis in another non-linear effect of piezoelectric actuators [15]. A system showing hysteresis will have a state that is dependent on the history of the system. Therefore, the input to output map cannot be said to be one-to-one. For a piezoelectric actuator this means that you can't know the displacement of the material merely by looking at the current input voltage potential. The knowledge of the whole history of input voltages are needed in order to know the displacement. However, hysteresis is a phenomenon that have been studied in detail, and a lot of models exist. The most common hysteresis model is called the classical Preisach hysteresis model [15]. Hysteresis will not be considered further in this thesis.

Another issue with piezoelectric actuators are the lightly damped resonance modes which limits the achievable bandwidth [15]. This can be understood as how fast the system can respond to changes in actuation. This is one of the main limiting effects that limit the maximum reasonable scanning frequency, or probe tip speed, that the AFM can be operated with.

2.4 Advantages of AFM

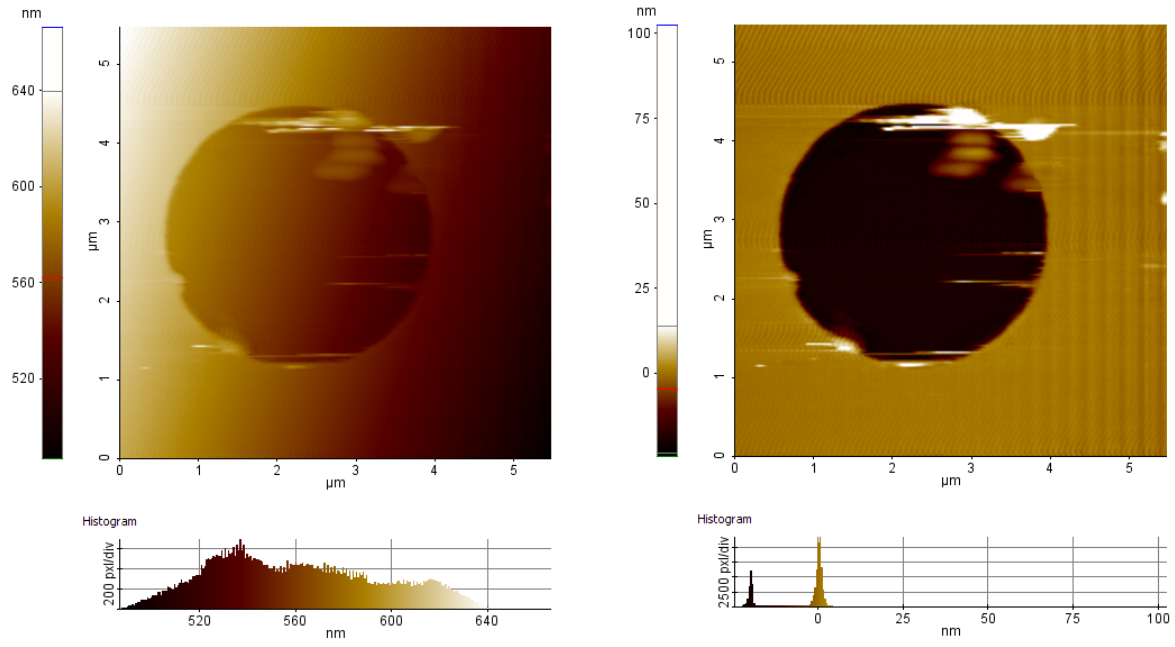
The AFM technique can measure the surface properties of almost any material on a nanometer scale, be it metal or biological tissue. This can be done without destroying the sample in the process given that the material is sufficiently flat, i.e. on the scale of 10 µm.

One of the advantages of AFM as opposed to STM and electron microscopes is that the technique can be used in regular air and liquids, and does not need a vacuum environment during scanning. This simplifies some of the scanning process. And apart from STM, AFM can be used on insulating surfaces, effectively increasing the set of surfaces that can be measured.

The AFM techniques are not limited to the measurement of physical topography. By coating the probe tip in different substances, other interaction forces like electrical and magnetic forces can be introduced into the sum of interaction forces between probe tip and surface. In this way the AFM can be enabled to measure both electrical and magnetic properties of a sample.

2.5 XE-70 AFM Image Capture

Several test image scans were conducted with the Park Systems XE-70 AFM at the lab. This was done for illustration purposes and to get first-hand experience with the system. One of the scans are shown in figure 2.5. This image was generated with the use a stock software program called XEI. The scanned surface was a calibration sample with holes and groves. The raw image is shown to the left, while to the right, a post processed version of the same image is shown. In the post processing the inclination or slope of the sample have been removed. In 2.5b, we see a round hole with a depth of about 20nm, and a radius of about 3.5 µm. In addition, we see some white spots which is most likely the result of dust particles on the surface.



(a) (b)

Figure 2.5: Test scan conducted with a XE-70 AFM from Park Systems.

Chapter 3

Fractional-Order Calculus for Control

This chapter aims at giving the reader an introduction to some of the mathematical results on the subject of fractional calculus with a focus on fractional-order transfer functions. Most of the theory presented here originates from the book *Fractional-order Control Systems* [5] written by Xue Dingyü, and published in 2017. A similar and related overview on the topic of fractional-order calculus was presented by the author of this thesis in a proceeding report [22].

3.1 Grünwald-Letnikov Definition

$$ {}_{x_0}\mathcal{D}_x^{\alpha} f(x) = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\left[\frac{x-x_0}{h}\right]} (-1)^j \binom{\alpha}{j} f(x - jh). \tag{3.1} $$

Equation (3.1) is known as the Grünwald-Letnikov (GL) definition of fractional derivatives. It is one of the most used definitions alongside the Riemann-Liouville (RL) and the Caputo definitions for calculation of fractional, or non-integer, derivatives. Parameter α is the order of differentiation and can be either an integer value or a non-integer value, that is to say $\alpha \in \mathbb{R}$. Often, when expressing fractional-order derivatives the unified fractional-order integro-differential operator [5], written like ${}_{x_0}\mathcal{D}_x^{\alpha}(\cdot)$, is used in place of $\frac{\mathrm{d}^n}{\mathrm{d}x^n}(\cdot)$ to distinguish between the non-integer/fractional-order derivatives and the normal integer order derivatives. It is also used to point out that the formula also holds for integrals by using negative values for α. The unified fractional-order integro-differential operator is defined as:

$$ {}_{x_0}\mathcal{D}_x^{\alpha} = \begin{cases} \frac{\mathrm{d}^n}{\mathrm{d}x^n} f(x), & \alpha > 0, \\ f(x), & \alpha = 0, \\ \int_{x_0}^{x}, & \alpha < 0. \end{cases} \tag{3.2} $$

From (3.1) it can be seen that the GL definition of fractional derivatives rely on the current and all the past values of the function under fractional-order integro-differentiation, starting from x_0. Assuming that $f(x) = 0$ for all $x < x_0$. This is often marked by a prescript on the fractional-order integro-differential operator, ${}_{x_0}\mathcal{D}_x^{\alpha}(\cdot)$. Symbol $[\cdot]$ is used to denote rounding to nearest integer value. The binomial coefficients can be calculated with

$$\binom{\alpha}{j} = \frac{\Gamma(\alpha+1)}{\Gamma(j+1)\Gamma(\alpha-j+1)}, \tag{3.3}$$

where the Gamma function $\Gamma(x)$ is given by

$$\Gamma(x) = \int_0^\infty e^{-t}\, t^{x-1} \mathrm{d}t. \tag{3.4}$$

An alternative and equivalent form of (3.1), presented in [5], can be seen below.

$${}_{x_0}\mathcal{D}_x^\alpha f(x) = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{j=0}^{\frac{x-x_0}{h}} w_j f(x-jh). \tag{3.5}$$

In this alternative form the w_j coefficients can be calculated by the recursive formula

$$w_0 = 1, \quad w_j = \left(1 - \frac{\alpha+1}{j}\right) w_{j-1}. \tag{3.6}$$

From (3.5) it becomes apparent that the GL definition can be seen as a weighted sum of past function values. More on this in [22]. A note on how the Grünwald-Letnikov definition can be derived from the definition of the derivative can be found in appendix A.

3.2 Riemann-Liouville Definition

The Riemann-Liouville definition [5] is another way to calculate the fractional-order integral and derivative of a function. It is based on Cauchy's formula for repeated integration [23] and generalized to non-integer numbers α by switching out the factorial, $(n-1)!$, with the gamma function $\Gamma(\alpha)$, (3.4). The Riemann-Liouville fractional-order derivative, for $\alpha > 0$ and $n = \lceil \alpha \rceil$, is defined as

$${}_{x_0}\mathcal{D}_x^\alpha f(x) = \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_x^{x_0} \frac{f(\tau)}{(x-\tau)^{1+\alpha-n}} \mathrm{d}\tau. \tag{3.7}$$

The Riemann-Liouville fractional-order integral, for $\gamma > 0$, is defined as

$${}_{x_0}\mathcal{D}_x^{-\alpha} f(x) = \frac{1}{\Gamma(\alpha)} \int_x^{x_0} \frac{f(\tau)}{(x-\tau)^{1-\alpha}} \mathrm{d}\tau. \tag{3.8}$$

3.3 Caputo Definition

The third most used definition of fractional-order integrals and derivatives is the Caputo definition [5]. The Caputo fractional-order derivative, for $\alpha > 0$ and $n = \lceil \alpha \rceil$, is defined as

$${}_{x_0}\mathcal{D}_x^\alpha f(x) = \frac{1}{\Gamma(n-\alpha)} \int_{x_0}^x \frac{f^{(n)}(\tau)}{(x-\tau)^{1+\alpha-m}} \mathrm{d}\tau. \tag{3.9}$$

The Caputo fractional-order integral, for $\gamma > 0$, is defined as

$${}_{x_0}\mathcal{D}_x^{-\gamma} f(x) = \frac{1}{\Gamma(\gamma)} \int_{x_0}^x \frac{f(\tau)}{(x-\tau)^{1-\gamma}} \mathrm{d}\tau. \tag{3.10}$$

3.4 Fractional-order Systems in Laplace Domain

The Riemann-Liouville definition of a fractional-order derivative (3.7) has the equivalent Laplace transform definition

$$\mathcal{L}\left\{\mathcal{D}_t^\alpha f(t)\right\}(s) = s^\alpha F(s) - \sum_{k=0}^{(\lceil\alpha\rceil-1)} s^k \left.\mathcal{D}_t^{\alpha-k-1} f(t)\right|_{t=0}. \tag{3.11}$$

As with the time domain version α is restricted to positive values.

Looking at the sum of initial values on the right hand side of (3.11), we see that initial values of fractional-order derivatives has to be specified. As the meaning of a non-integer derivative of a function is not intuitive, this definition can be hard to utilize given initial values different from zero.

The Caputo definition of a fractional-order derivative (3.9) on the other hand, has the equivalent Laplace transform definition

$$\mathcal{L}\left\{\mathcal{D}_t^\alpha f(t)\right\}(s) = s^\alpha F(s) - \sum_{k=0}^{(\lceil\alpha\rceil-1)} s^{\alpha-k-1} \left.f^{(k)}(t)\right|_{t=0}. \tag{3.12}$$

Comparing (3.11) and (3.12) we observe that the Laplace transform of the Riemann-Liouville and the Caputo definition for fractional-order derivatives are quite similar. The only difference is that the Caputo definition contains only integer-order initial values. This makes it easier to handle non-zero initial values with the Caputo version.

If all the initial values are zero, the two definitions (3.11) and (3.12) collapse to the same simplified form

$$\mathcal{L}\left\{\mathcal{D}_t^\alpha f(t)\right\}(s) = s^\alpha F(s). \tag{3.13}$$

3.5 Fractional-order Differential Equations

A general and linear differential equation can be expressed as

$$\begin{aligned} a_n y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \ldots + a_1 \dot{y}(t) + a_0 y(t) \\ = b_m u^{(m)}(t) + b_{m-1} y^{(m-1)}(t) \ldots + b_1 \dot{u}(t) + b_0 u(t), \end{aligned} \tag{3.14}$$

where $x^{(k)}(t)$ should be interpreted as the k-th derivative of x with respect to t. Now, using the unified fractional-order integro-differential operator defined in (3.2) a similar general and linear fractional-order differential equation can be expressed as

$$\begin{aligned} a_n \mathcal{D}^{\eta_n} y(t) + a_{n-1} \mathcal{D}^{\eta_{n-1}} y(t) + \ldots + a_1 \mathcal{D}^{\eta_1} y(t) + a_0 \mathcal{D}^{\eta_0} y(t) \\ = b_m \mathcal{D}^{\gamma_m} u(t) + b_{m-1} \mathcal{D}^{\gamma_{m-1}} u(t) + \ldots + b_1 \mathcal{D}^{\gamma_1} u(t) + b_0 \mathcal{D}^{\gamma_0} u(t), \end{aligned} \tag{3.15}$$

where $\{\eta_i\}$ and $\{\gamma_i\}$ is the order of the fractional-order derivatives.

Now, it turns out that fractional-order differential equations on the form (3.15) can be solved analytically in much the same way that regular differential equations can be solved by using Laplace transformation, partial fraction decomposition and inverse Laplace transformation. But

before illustrating this for fractional-order differential equations a new family of functions must be introduced.

In the solutions of regular linear differential equations, the exponential function plays a vital role. However, in the solving of fractional-order linear differential equations a related family of functions takes on the same level of importance. This family of functions have been termed the Mittag-Leffler family.

A general Mittag-Leffler function with three variables [5] is shown below.

$$\mathcal{E}_{\alpha,\beta}^{\gamma}(z) = \sum_{k=0}^{\infty} \frac{(\gamma)_k}{\Gamma(\alpha k + \beta)} \frac{z^k}{k!}. \tag{3.16}$$

Here, $(\gamma)_k$ is the rising factorial, defined as

$$(\gamma)_k = \gamma(\gamma+1)(\gamma+2)\cdots(\gamma+k-1) = \frac{\Gamma(k+\gamma)}{\Gamma(\gamma)}. \tag{3.17}$$

Another name of a rising factorial is Pochhammer symbol. But since this name have been used on both rising and falling factorials, care should be taken when using this name. When $\gamma = 1$, the rising factorial equals the well-known factorial function, $(1)_k = k!$.

It can be mentioned that there exist other variants on the Mittag-Leffler function with one, two and four variables as well. And as the reader might have guessed from the introductory discussion, the Mittag-Leffler family of functions is indeed a generalization of the well-known exponential function. In fact, the exponential function can be seen as a part of this family. This can be seen if we set $\alpha = 1$, $\beta = 1$ and $\gamma = 1$ in equation (3.16) as have been done below.

$$\mathcal{E}_{1,1}^{1}(z) = \sum_{k=0}^{\infty} \frac{(1)_k}{\Gamma(1\cdot k+1)} \frac{z^k}{k!} = \sum_{k=0}^{\infty} \frac{k! z^k}{\Gamma(k+1)\, k!} = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(k+1)} = \sum_{k=0}^{\infty} \frac{z^k}{k!} = e^z. \tag{3.18}$$

Now, with the knowledge of Mittag-Leffler functions, the analytic solution to fractional-order differential equations can be explained. This is done in the next section.

3.6 Fractional-order Transfer Functions

A general differential equation on the form (3.14) can be expressed on transfer function form as

$$\frac{y(s)}{u(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0}, \tag{3.19}$$

if initial values are set to zero. Here, m and n are positive integer values and b_j and a_i the coefficients of the transfer function.

By using the Riemann-Liouville or Caputo definition for fractional-order derivatives in s-domain, i.e. (3.11) or (3.12), a similar transfer function to (3.19) can be obtained for general fractional-order differential equation on the form (3.15).

$$\frac{y(s)}{u(s)} = \frac{b_m s^{\gamma_m} + b_{m-1} s^{\gamma_{m-1}} + \ldots + b_1 s^{\gamma_1} + b_0 s^{\gamma_0}}{a_n s^{\eta_n} + a_{n-1} s^{\eta_{n-1}} + \ldots + a_1 s^{\eta_1} + a_0 s^{\eta_0}}. \tag{3.20}$$

Here, m and n are the numbers of different fractional-order parts in nominator and denominator, respectively. γ_j and b_j are the corresponding fractional orders and coefficients of the numerator, while η_i and a_i are the corresponding fractional orders and coefficients of the denominator. Here we can assume that $\gamma_m \geq \gamma_{m-1} \geq \cdots \geq \gamma_1 \geq \gamma_0$ and $\eta_n \geq \eta_{n-1} \geq \cdots \geq \eta_1 \geq \eta_0$ without loss of generality.

After a partial fraction expansion of the fractional-order transfer function (3.20), the transfer function can be written on the form

$$G(s) = \sum_{i=1}^{N} \sum_{j=1}^{m_i} \frac{r_{ij}}{(s^\alpha + p_i)^j}, \tag{3.21}$$

where r_{ij} are complex values, the pseudo poles, p_i, are complex values. N is the number of distinct poles, while m_i is the multiplicity of the i-th pseudo pole p_i.

Now, using a neat result of irrational transfer functions, presented in [5], namely

$$\mathcal{L}^{-1}\left\{\frac{s^{\alpha\gamma-\beta}}{(s^\alpha + a)^\gamma}\right\} = t^{\beta-1}\mathcal{E}^{\gamma}_{\alpha,\beta}\left(-at^\alpha\right), \tag{3.22}$$

equation (3.21) can be transformed back to the time domain with the use of the Mittag-Leffler function of three variables, (3.16).

By choosing different values for α, β and γ in (3.22), we can get time domain expressions for many of the Laplace expressions that normally appear during partial fraction expansions. To illustrate, a short list of inverse-Laplace transformations for some irrational transfer functions are shown below.

$$\mathcal{L}^{-1}\left\{\frac{1}{s^\alpha + a}\right\} = t^{\alpha-1}\mathcal{E}^{1}_{\alpha,\alpha}(-at^\alpha), \tag{3.23}$$

$$\mathcal{L}^{-1}\left\{\frac{1}{(s^\alpha + a)^\gamma}\right\} = t^{\alpha\gamma-1}\mathcal{E}^{\gamma}_{\alpha,\alpha\gamma}(-at^\alpha), \tag{3.24}$$

$$\mathcal{L}^{-1}\left\{\frac{1}{s^\beta}\right\} = \frac{t^{\beta-1}}{\Gamma(\beta)}, \tag{3.25}$$

$$\mathcal{L}^{-1}\left\{\frac{s^\beta}{(s^\alpha + a)^\gamma}\right\} = t^{(\alpha\gamma-\beta-1)}\mathcal{E}^{\gamma}_{\alpha,(\alpha\gamma-\beta)}(-at^\alpha). \tag{3.26}$$

So, in theory, it is possible to calculate analytical solutions to fractional-order transfer functions with known inputs like step and impulse. The challenging part of this procedure however, is to find the pseudo poles of the fractional-order transfer function, so that equation (3.20) can be rewritten on the form (3.21). Another thing to mention is the computational complexity of the Mittag-Leffler family of functions (3.16). To do an exact evaluation of the function, an infinite sum will have to be calculated in most cases. This is infeasible, so in practice a finite truncated sum can be used as an approximation.

3.7 Calculation and Realization Techniques

This section is an excerpt from earlier work conducted by the author in [22]. It has been reproduced here for ease of access for the reader, for reference purposes and to make the thesis somewhat more complete.

All the fractional-order derivative definitions are hard to realize in practise because of the infinite sums. The Grünwald-Letnikov definition is the most suited one of the three for practical implementation, however the growing number of function values that must be taken into account as time goes on makes it hard to use in most cases. This has led to the proposal of several approximation techniques for the calculation of fractional derivatives. In the following subsections the most prominent techniques are presented. More techniques are presented in [5].

3.7.1 Truncated Grünwald-Letnikov Approximation

One method that is proposed in [24] and [5] is to use, not the Grünwald-Letnikov definition for fractional-order derivative, but a truncated version of it. where only the last L function values are used. This technique is referred to as "short memory principle" or "short-memory effect". Since the w_j coefficients in (3.6) decrease quite rapidly towards zero for $j > \alpha + 1$ this turns out to work quite well for low h and high L.

A truncated approximated version of (3.5) and (3.6) is shown in (3.27) and (3.28). This is a good approximation given high L, small h and not a too small α.

$$ {}_{x_0}\mathcal{D}_x^\alpha f(x) \approx {}_{(x-L)}\mathcal{D}_x^\alpha f(x) = \frac{1}{h^\alpha} \sum_{j=0}^{L} w_j f(x - jh), \tag{3.27} $$

$$ w_0 = 1, \quad w_j = \left(1 - \frac{\alpha + 1}{j}\right) w_{j-1}, \tag{3.28} $$

where $j = 1, 2, \ldots, L - 1$.

The error of this method is, according to [5]

$$ \Delta(x) = \left| {}_{x_0}\mathcal{D}_x^\alpha f(x) - {}_{(x-L)}\mathcal{D}_x^\alpha f(x) \right| \leq \frac{ML^{-\alpha}}{|\Gamma(1-\alpha)|}, \tag{3.29} $$

given that $f(x) \leq M$.

It is possible to calculate how big L needs to be to ensure $\Delta(x) < \epsilon$, with the following inequality

$$ L \geq \left(\frac{M}{\epsilon |\Gamma(1-\alpha)|} \right)^{1/\alpha}. \tag{3.30} $$

3.7.2 Oustaloup Filter Approximation

The Oustaloup filter is named after the French Professor Alain Oustaloup. It is based on work done by Oustaloup in [25]. The method evolves around approximating the fractional-order derivative s^α in frequency domain by several integer-order pole-zero pairs. In a bode diagram, s^α with $0 < \alpha < 1$ is a straight line with an increase of 20α dB/dec. This line can be approximated with a set of asymptotes that alternate between 0 dB/dec and 20 dB/dec. These asymptotes

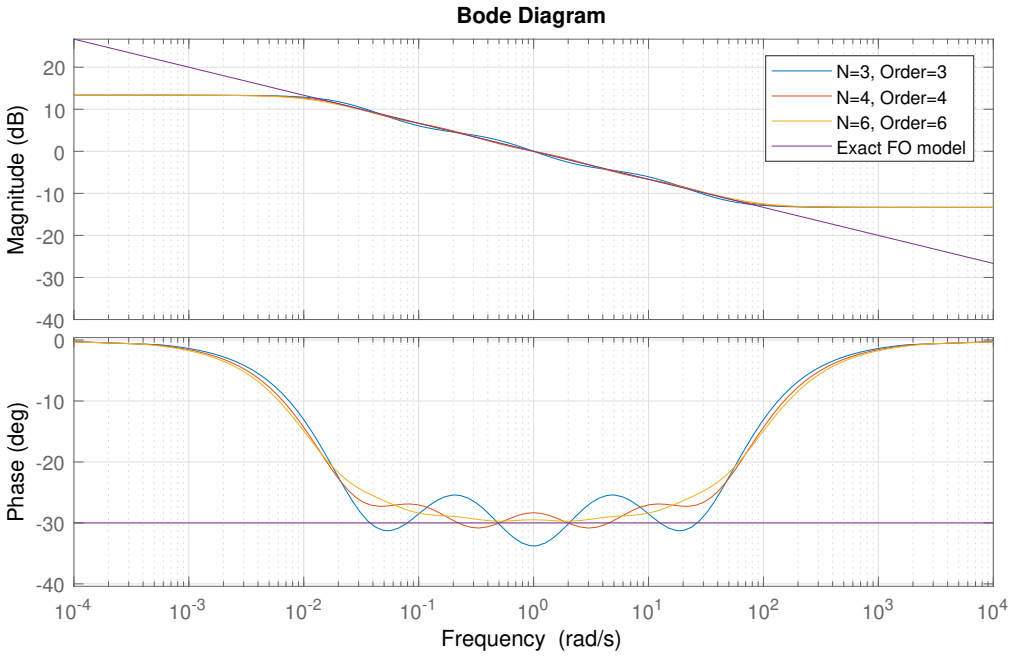


Figure 3.1: Oustaloup approximations of $G(s) = \frac{1}{s^{1/3}}$.

are created by several integer-order pole-zero pairs where the poles and zeros are given specific values according to the set of equations (3.31) - (3.35). Petráš gives a good presentation to Oustaloup filters in [26].

$$s^{\gamma} \approx K \prod_{k=1}^{N} \frac{s + \omega_k'}{s + \omega_k}, \tag{3.31}$$

$$\omega_k' = \omega_b \, \omega_u^{(2k-1-\gamma)/N}, \tag{3.32}$$

$$\omega_k = \omega_b \, \omega_u^{(2k-1+\gamma)/N}, \tag{3.33}$$

$$K = \omega_h^{\gamma}, \tag{3.34}$$

$$\omega_u = \sqrt{\frac{\omega_h}{\omega_b}}, \tag{3.35}$$

given $k = 1, 2, \ldots, N$. Where γ is the fractional derivative order, ω_b and ω_h is respectively, the lower and upper frequency bounds of a frequency area where the approximation is valid, and N is the number of pole-zero pairs used in the approximation, usually called the order of the filter. The number of computations and the accuracy of the approximation increases with increasing N.

Figure 3.1 shows the Oustaloup filter approximations of $G(s) = \frac{1}{s^{1/3}}$ for three different values of n. The corresponding rational transfer functions are shown in (3.36) - (3.38). $\omega_1 = 0.01\,\text{rad/s}$ and $\omega_n = 100\,\text{rad/s}$ have been chosen.

$$H_1(s) = \frac{0.2154s^3 + 8.119s^2 + 13.54s + 1}{s^3 + 13.54s^2 + 8.119s + 0.2154}, \tag{3.36}$$

$$H_2(s) = \frac{0.2154s^4 + 11.11s^3 + 52.04s^2 + 23.94s + 1}{s^4 + 23.94s^3 + 52.04s^2 + 11.11s + 0.2154}, \tag{3.37}$$

$$H_3(s) = \frac{0.2154s^6 + 16.46s^5 + 222.8s^4 + 625s^3 + 371.7s^2 + 45.8s + 1}{s^6 + 45.8s^5 + 371.7s^4 + 625s^3 + 222.8s^2 + 16.46s + 0.2154}. \tag{3.38}$$

3.7.3 Matsuda-Fuji Filter Approximation

Another method is the Matsuda-Fujii filter approximation as described in [5]. The algorithm finds a rational function approximation $H(s)$ for an irrational transfer function $F(s)$. This is done with the continued fraction technique. The filter design procedure is as follows:

1. Define a frequency vector ω_i for $i = 1, 2, \ldots, n$
2. Compute $F(j\omega_i)$ for $i = 1, 2, \ldots, n$, the frequency response of $F(s)$.
3. Compute $v_1(\omega_k) = |f(j\omega_k)|$ for $k = 1, 2, \ldots, n$. An initial sequence.
4. Compute $v_{k+1}(\omega_j) = \frac{\omega_j - \omega_k}{v_k(\omega_j) - v_k(\omega_k)}$ for $j = 1, 2, \ldots, n$.
5. Extract $a_k = v_k(\omega_k)$ for $k = 1, 2, \ldots, n$.
6. Compose the rational approximation $H(s)$ according to (3.39) from the coefficients a_k and ω_k for $k = 1, 2, \ldots, n$.

$$H(s) = a_1 + \cfrac{s - \omega_1}{a_2 + \cfrac{s - \omega_2}{a_3 + \cfrac{s - \omega_3}{a_4 + \cdots}}}. \tag{3.39}$$

Figure 3.2 shows the Matsuda-Fujii filter approximations of $G(s) = \frac{1}{s^{1/3}}$ for three different values of n. The corresponding rational transfer functions are shown in (3.40) - (3.42). $\omega_1 = 0.01\,\text{rad/s}$ and $\omega_n = 100\,\text{rad/s}$ have been chosen. The other ω_j have been placed with equal logarithmic distance between ω_1, ω_n and each other. The relationship between the order of the Matsuda-Fujii filter and the coefficient n is Order $= \lceil (n-1)/2 \rceil$. Odd n gives a transfer function with relative degree zero, while even n gives a transfer function with relative degree equal -1. Therefore, it is recommended to choose an odd n.

$$H_1(s) = \frac{0.1488s^3 + 10.47s^2 + 18.4s + 1}{s^3 + 18.4s^2 + 10.47s + 0.1488}, \tag{3.40}$$

$$H_2(s) = \frac{0.1303s^4 + 16.23s^3 + 92.93s^2 + 39.18s + 1}{s^4 + 39.18s^3 + 92.93s^2 + 16.23s + 0.1303}, \tag{3.41}$$

$$H_3(s) = \frac{0.1056s^6 + 28.98s^5 + 636s^4 + 2076s^3 + 1153s^2 + 102.3s + 1}{s^6 + 102.3s^5 + 1153s^4 + 2076s^3 + 636s^2 + 28.98s + 0.1056}. \tag{3.42}$$

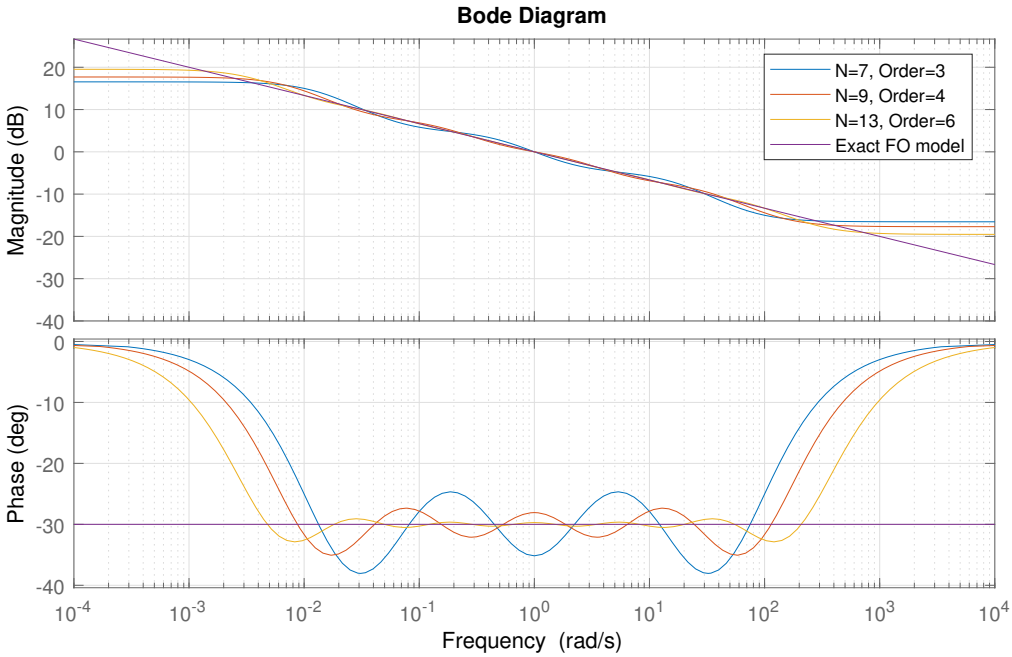


Figure 3.2: Matsuda-Fujii approximations of $G(s) = \frac{1}{s^{1/3}}$.

Chapter 4

Frequency and Stability Analysis of Simple Fractional-order Systems

4.1 Frequency Response of Fractional-order Systems

Switching s with $j\omega$ in a standard integer-order transfer function yields the so-called frequency response of the transfer function. The frequency response of transfer functions are much used for filter shaping and can also be used to assess stability of systems through Nyquist stability criterion. The notion of frequency response can easily be extended to fractional-order systems when noted that

$$(j\omega)^\alpha = \omega^\alpha e^{j\frac{\pi}{2}\alpha} = \omega^\alpha \cos\left(\frac{\pi}{2}\alpha\right) + j\omega^\alpha \sin\left(\frac{\pi}{2}\alpha\right). \tag{4.1}$$

A consequence of this is that fractional-order systems can have poles and zeros contributing phase changes other than multiples of 90 deg when looking at the argument of the frequency response and magnitude slopes other than multiples of 20 dB/decade when looking at the magnitude of the frequency response. In fact, if α is the fractional-order of a pole or zero, then the phase change, contributed by the pole or zero, can be found to be 90α deg, while the change in magnitude slope can be found to be 20α dB/decade.

For a simple fractional-order system with gain K and a fractional-pole with fractional-order α and coefficient a, that is

$$h_0(s) = \frac{K}{s^\alpha - a}, \tag{4.2}$$

the following equations for the magnitude and phase response of the system can be found

$$20\log_{10}|h_0(j\omega)| = 20\log_{10}(K) - 10\log_{10}\left|\omega^{2\alpha} - 2\omega^\alpha a\cos\left(\frac{\pi}{2}\alpha\right) + a^2\right|, \tag{4.3}$$

$$\angle h_0(j\omega) = \arctan\left(\frac{\sin\left(\frac{\pi}{2}\alpha\right)}{\cos\left(\frac{\pi}{2}\alpha\right) - \frac{a}{\omega^\alpha}}\right). \tag{4.4}$$

It's hard to see what the phase change and magnitude slope change of system (4.2) is based on the equations for the magnitude and phase of the system. Therefore, a bode plot with three different values for α is shown below.

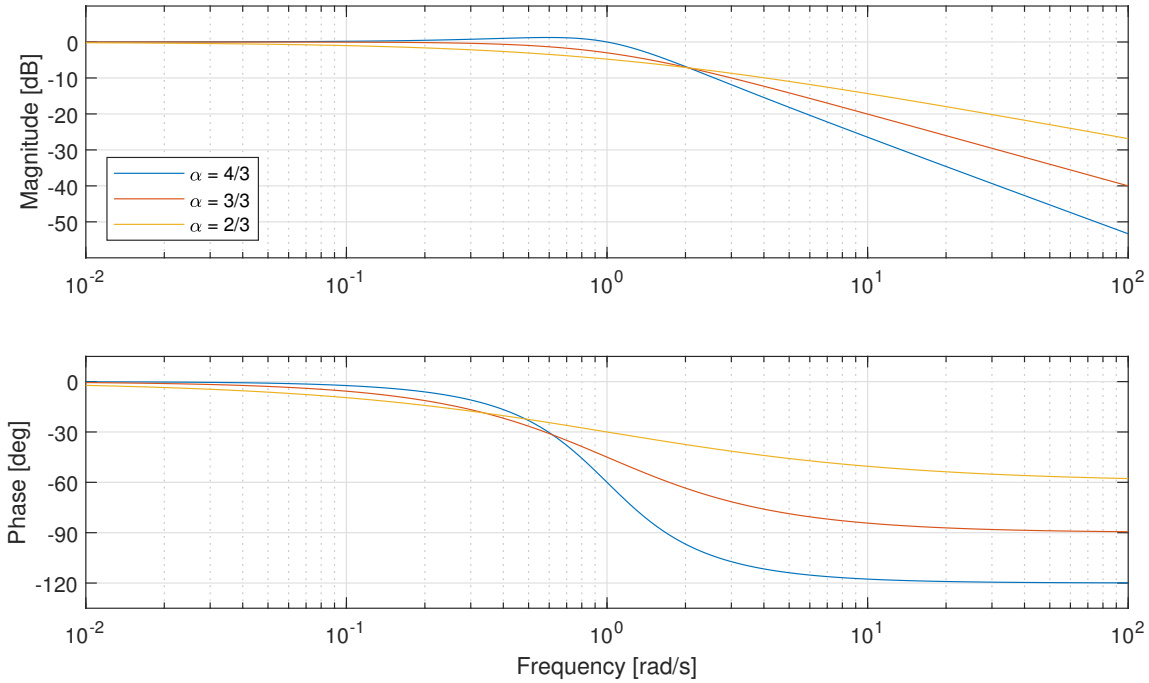


Figure 4.1: Bode diagram of simple fractional pole system (4.2) with different fractional-orders.

Table 4.1: Magnitude slope changes and phase changes for (4.2) given different fractional-orders.

α	Magnitude slope change [dB/decade]	Phase change [deg]
$\frac{4}{3}$	-26.67	$-60°$
$\frac{3}{3}$	-20.00	$-90°$
$\frac{2}{3}$	-13.33	$-120°$

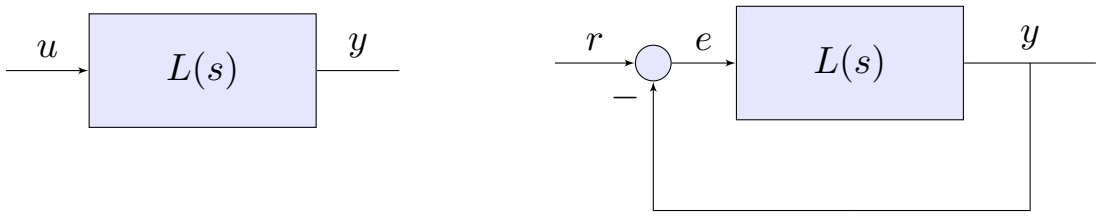
Inspecting the graphs in figure 4.1 we find phase changes and magnitude slope changes as seen in table 4.1. They match what was stated earlier in this section, and we conclude

Phase and magnitude slope change for fractional-order systems

Fractional-poles and fractional-zeros on the form $(s^\alpha - a)$ contributes with a total phase change of $\pm 90\alpha\,\text{deg} = \pm\frac{\pi}{2}\alpha\,\text{rad}$ and a magnitude slope change of $\pm 20\alpha\,\text{dB/decade}$.

4.2 Nyquist Stability Criterion

Using Nyquist stability criterion, it is possible to conclude whether an open-loop system, $L(s)$, will be closed-loop stable after a unity negative feedback have been added by inspection of a Nyquist plot. The number of open-loop unstable poles or in other words the number of poles in the right half plane is also needed to be able to conclude. This is a quite useful result for stability analysis of linear systems and is often used to make sure that a system is stable after a feedback connection have been made. The proceeding explanation of the criterion is based on [27].



(a) Open-loop

(b) Closed-loop

Figure 4.2: Standard system block diagrams.

It is common knowledge in control theory that a linear time-invariant system $L(s)$ will have a stable response if all the poles of $L(s)$ lies in the left half plane. When a unity negative feedback is added as seen in figure 4.2b, the closed-loop system $T(s)$ may no longer be stable, since the poles have changed. In this case Nyquist's stability criterion can be used to check stability of the closed-loop system.

The open-loop system seen in figure 4.2a can be expressed in transfer function form as

$$L(s) = \frac{n(s)}{d(s)}, \tag{4.5}$$

while the closed-loop system seen in figure 4.2b has the transfer function

$$T(s) = \frac{L(s)}{1 + L(s)} = \frac{n(s)}{d(s) + n(s)}. \tag{4.6}$$

Here, $n(s)$ in the nominator of the open-loop system and $d(s)$ is the denominator of the open-loop system.

$$1 + L(s) = 1 + \frac{n(s)}{d(s)} = \frac{d(s) + n(s)}{d(s)}. \tag{4.7}$$

Now, consider the special transfer function shown in (4.7), it becomes apparent that the nominator and denominator polynomials of (4.7) are the denominator of the closed-loop system (4.6) and the denominator of the open-loop system (4.5), respectively. Therefore, a way to think about (4.7) is that it contains information of both the open-loop poles and the closed-loop poles of the system, i.e. it contains a connection linking the poles together. This is one of the key points of the Nyquist criterion.

$$1 + L(s) = \frac{d(s) + n(s)}{d(s)} = c\frac{(s - \lambda_n)(s - \lambda_{n-1}) \cdots (s - \lambda_2)(s - \lambda_1)}{(s - \rho_m)(s - \rho_{m-1}) \cdots (s - \rho_2)(s - \rho_1)}. \tag{4.8}$$

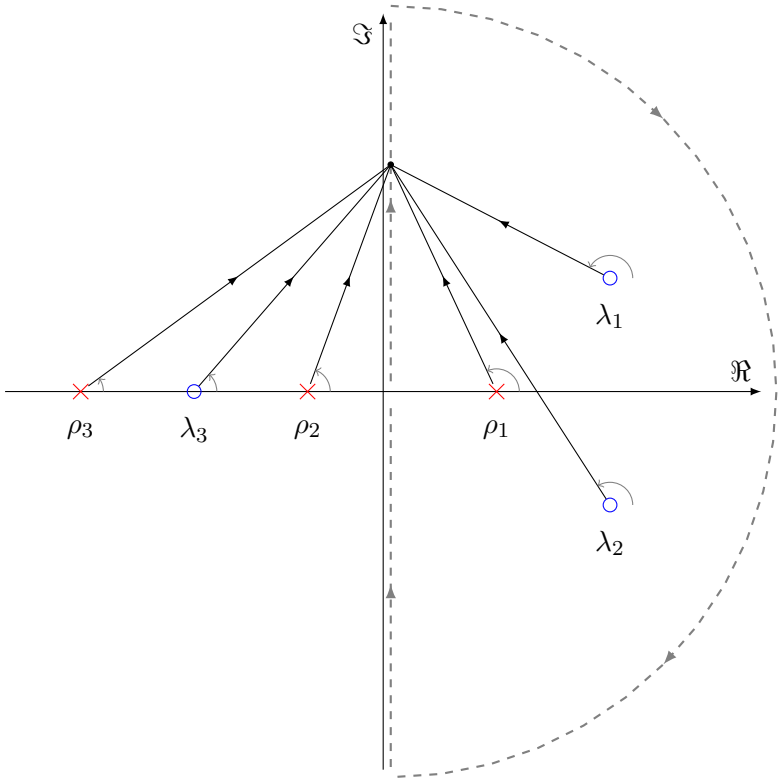


Figure 4.3: Poles (red) and zeros (blue) in the complex plane. $n = 3$ and $m = 3$. Vectors illustrating rotation contributions from zeros and poles is shown in addition to the integration path used to enclose the whole of the right half plane.

Moving on, we start by writing (4.7) on a general form with n zeros and m poles as in (4.8), in addition to a plot of the poles and zeros in the complex plane, with $n = 3$ and $m = 3$, as in figure 4.3. The function $1 + L(s)$ can be seen upon as a mapping from a complex plane $s \in \mathbb{C}$ to another complex plane $1 + L(s) \in \mathbb{C}$.

The argument of the complex function (4.8) can easily be calculated. We find that

$$\angle\left(1 + L(s)\right) = \arg(c) + \sum_{k=0}^{n} \arg(s - \lambda_k) - \sum_{k=0}^{m} \arg(s - \rho_k), \tag{4.9}$$

and realizing that the argument of $1 + L(s)$ is the sum of the arguments from each of the poles and zeros. This motivates the drawing of angles and vectors from each pole and zero to an arbitrary point that indicates the complex variable $s = \sigma + j\omega$, as can be seen in figure 4.3.

Now, to the real ingenuity of the Nyquist criterion. If we let the complex variable s run through a closed path around the right half plane by having s moving along the entire imaginary axis from $s = -j\omega$ to $s = j\omega$ and then following the path of a half circle from $s = j\infty$ to $s = -j\infty$, and then integrate one time around the closed loop, we observe something quite interesting with regards to the zeros and poles. All the poles and zeros in the right half plane will contribute to $\angle\left(1 + L(s)\right)$ with a full rotation in either a clockwise direction or in a counter clockwise direction, i.e. a change in angle of $\pm 360°$ or $\pm 2\pi$. All the poles and zeros in the left half plane will however contribute with zero rotations, or zero change in angle, after $\angle\left(1 + L(s)\right)$ have been integrated one time around the closed path shown in figure 4.3.

Zeros of (4.8), which are the poles of the closed-loop system (4.6) will contribute with a positive rotation (counter-clockwise rotation), while the poles of (4.8), which are the poles of the open-loop system (4.5) will contribute with a negative rotation (clockwise rotation). By summing the individual rotation contributions from the right half plane poles and zeros, it is therefore possible to deduce the difference in number of right half plane poles for the open-loop and closed-loop system. This alone is not enough to conclude whether the closed-loop system or open-loop systems are stable. But if the number of right half plane poles is known for either of them it's possible to know the number of right half plane poles for the other one by use of Nyquist's stability criterion as just explained. Nyquist's Stability Criterion can be summarized as follows

Nyquist's Stability Criterion

Let N_n be the number of right half plane (rhp) poles for the closed-loop system (or the number of rhp zeros in $1 + L(s)$),
and let N_p be the number of rhp poles for the open-loop system (or the number of rhp poles in $1 + L(s)$). Then we have that

$$\Delta\angle\left(1 + L(s)\right) = -2\pi\left(N_n - N_p\right), \tag{4.10}$$

where $\Delta\angle\left(1 + L(s)\right)$ is the sum of the argument of $1 + L(s)$ when integrated around a closed loop path enclosing the whole of the right half complex plane, as shown in figure 4.3.

Some quick notes accompanying the criterion. Usually a stable closed-loop system is wished for. In this case $N_n = 0$, and $\Delta\angle\left(1 + L(s)\right) = 2\pi N_p$ must be fulfilled. So the number of counter-clockwise rotations of $\angle\left(1 + L(s)\right)$ must equal the number right half plane poles for the open-loop system N_p, for the closed-loop system to be stable.

Usually when using (4.10) to assess stability of a closed loop system $\angle L(s)$ is plotted and the number of rotations around the $(-1, 0)$ point is checked instead of plotting $\angle\left(1 + L(s)\right)$ and then checking rotations around $(0, 0)$.

4.3 Fractional-order two-pole system

A well-known standard representation of a second-order linear system is

$$H(s) = \frac{K}{(s - \lambda_1)(s - \lambda_2)} = \frac{K\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}. \tag{4.11}$$

Normally $\omega_0 > 0$. This transfer function needs to satisfy $\zeta > 0$ for the transfer function to be stable.

We now consider a fractional-order modified variant of the same transfer function.

$$H(s) = \frac{K}{(s^\alpha - \lambda_1)(s^\alpha - \lambda_2)} = \frac{K\omega_0^2}{s^{2\alpha} + 2\zeta\omega_0 s^\alpha + \omega_0^2}, \tag{4.12}$$

where α is the fractional order of the two fractional-poles. It should be noted that the two fractional-poles can have different fractional orders, but here we will consider a simple variant where the two fractional-poles have the same fractional order.

Through calculations and simulations, it can be shown that $\zeta > 0$ is no longer a sufficient and necessary condition for stability. Through simulations and by looking at the theory the author has come to the following simple result for the stability of a system with two fractional-poles and with the same fractional order.

Conjecture: Stability of system with two fractional-poles and the same fractional order

For a fractional-order system given by the irrational transfer function (4.12) with $\omega_0 > 0$ to be stable, the following must be satisfied.

$$\zeta > -\cos\left(\frac{\pi}{2}\alpha\right) \quad \text{and} \quad 0 < \alpha < 2. \tag{4.13}$$

At the best of the author's ability, no mention of this result has been found in the literature, but the simplicity of the result may indicate that it is already known by some in the field of fractional-order control theory. Since the author has not been able to formulate a sufficient proof the result is stated as a conjecture. Nevertheless, an attempt is made at motivating the result below, through a mix of theory and simulation.

4.3.1 Inquiry Through Nyquist Criterion

In [5], Xue writes: "In the original Nyquist theorem, there was no assumption that G(s) is a rational integer-order transfer function. Therefore, the theorem should be valid for fractional-order, or even irrational systems." Therefore, according to Xue, it should be possible to use Nyquist's stability criterion even on irrational systems with open loop transfer functions like

$$L(s) = \frac{n(s)}{d(s)} = \frac{\left(s^{\delta_m} - \lambda_m\right)\left(s^{\delta_{m-1}} - \lambda_{m-1}\right)\cdots\left(s^{\delta_1} - \lambda_1\right)\left(s^{\delta_0} - \lambda_0\right)}{\left(s^{\gamma_n} - \rho_n\right)\left(s^{\gamma_{n-1}} - \rho_{n-1}\right)\cdots\left(s^{\gamma_1} - \rho_1\right)\left(s^{\gamma_0} - \rho_0\right)}. \tag{4.14}$$

Apart from Xue's comment, the author has found two articles linking Fractional-order systems and Nyquist stability criterion. In [28] the Nyquist stability criterion was applied to a class of linear time-invariant distributed parameter feedback systems. A form of Nyquist stability analysis of a transfer function with $\sqrt{s}$ in place of s was done. In [29], bounded input bounded output stability of a set of irrational transfer functions has been investigated in terms of Nyquist, focusing on branch points of fractional-order transfer functions. An extension of the Nyquist criterion for a class of irrational transfer functions is proposed as well.

Now to explain some of the author's own investigation into the area of Fractional-order transfer functions and Nyquist stability criterion. When using Nyquist's stability criterion, the difference in number of poles and zeros in the right half plane of the test function $1 + L(s)$ is found by first adding a closed path that covers the whole of the open right half plane. Then the change in argument by moving one time around the path is found, before applying (4.10) and concluding with respect to stability. In practice the frequency response is calculated in order to do part of this analysis. Now on to the idea. ->

Looking at a simple integer-order system with one pole and a simple fractional-order system with one fractional-pole,

$$h_1(s) = \frac{1}{s+a}, \tag{4.15}$$

and

$$h_2(s) = \frac{1}{s^\alpha + a}. \tag{4.16}$$

If we now think about taking the frequency response of $h_2(s)$ and substitute $s = j\omega$ in (4.16), we get

$$h_2(s = j\omega) = \frac{1}{j^\alpha \omega^\alpha + a}. \tag{4.17}$$

Now, instead of regularly inserting $s = j\omega$ into $h_1(s)$, we instead insert $s = (j\omega)^\alpha$, and find

$$h_1(s = (j\omega)^\alpha) = \frac{1}{j^\alpha \omega^\alpha + a}. \tag{4.18}$$

Comparing (4.17) with (4.18), we realize that calculating the frequency response of (4.16) should be the same as calculating an alternate frequency response $s = (j\omega)^\alpha$ of (4.15).

So, instead of looking at the fractional-pole as a polynomial with several branches and solutions, when calculating the frequency response for use with Nyquist, we can look at the fractional-pole as a regular pole, but use another alternative way of calculating the frequency response by using $s = (j\omega)^\alpha$ instead of $s = j\omega$.

As a result, it should be possible to treat the fractional-pole as a regular pole if the integration path illustrated in figure 4.4 is used instead of the regular one shown in figure 4.3.

Now, for the two fractional-pole system (4.12), treating it as a regular system, we find

$$\lambda_{1,2} = \begin{cases} \omega_0 \left(-\zeta \pm j\sqrt{1-\zeta^2} \right), & \zeta < 1, \\ \omega_0 \left(-\zeta \pm \sqrt{\zeta^2 - 1} \right), & \zeta > 1. \end{cases} \tag{4.19}$$

The altered integration path can now be defined as

$$s = (j\omega)^\alpha = \omega^\alpha \cos\left(\frac{\pi}{2}\alpha\right) + j\omega^\alpha \sin\left(\frac{\pi}{2}\alpha\right). \tag{4.20}$$

By comparing (4.19) and (4.20) a condition for the poles to exist outside of the closed path shown in figure 4.4 can be made.

For the poles to lie outside of the contour, the real part of the poles must satisfy

$$\Re(\lambda) = -\omega_0 \zeta < \omega^\alpha \cos\left(\frac{\pi}{2}\alpha\right), \tag{4.21}$$

where ω can be any value. Choosing $\omega = \sqrt[\alpha]{\omega_0}$, and remembering that $\omega_0 > 0$, we find that

$$-\zeta < \cos\left(\frac{\pi}{2}\alpha\right), \tag{4.22}$$

where we can divide by -1 on both sides and flip the less than sign to get

$$\zeta > -\cos\left(\frac{\pi}{2}\alpha\right), \tag{4.23}$$

which is the proposed conjecture presented in section 4.3.

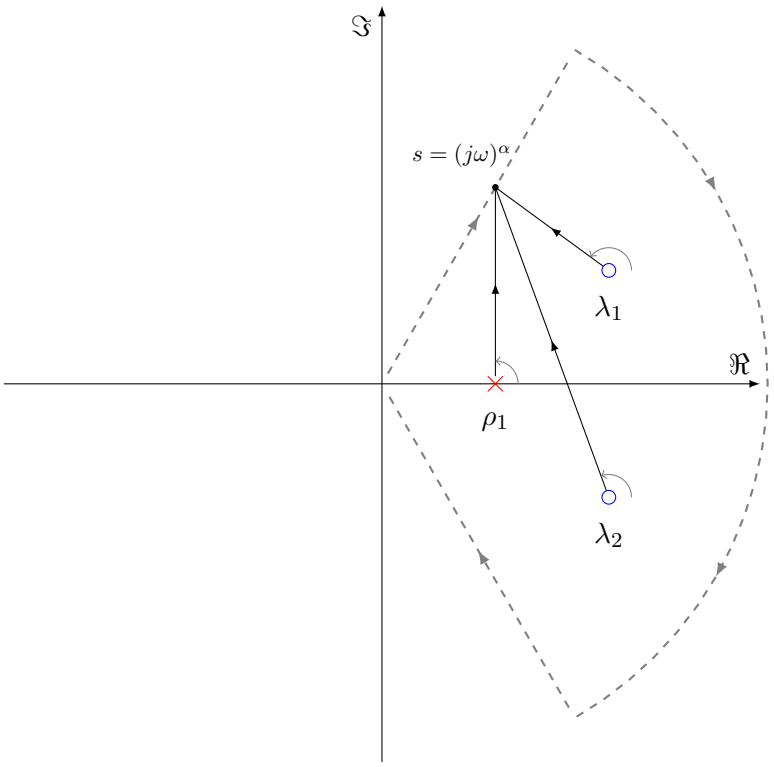


Figure 4.4: Showing the complex plane, a set of poles and zeros and a slightly altered integration path.

4.3.2 Inquiry Through Simulation

In order to sort of verify the conjecture in section 4.3, a simple step response simulation of the fractional-order transfer function (4.12) was set up with MATLAB and Simulink. Like the other simulations of fractional-order systems presented in this thesis, the `Fractional-order transfer function` block from the `fotf` toolbox was used to simulate a fractional-order transfer function. The Oustaloup filter approximation was used for realization of the fractional-order derivatives. More on this technique in section 3.7.2.

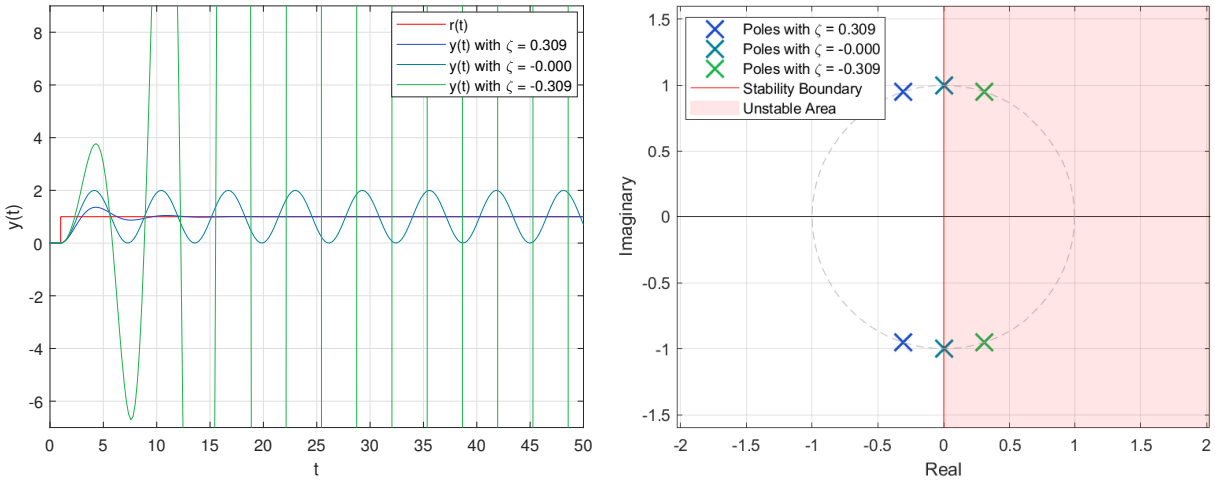
To make the simulation accurate a high Oustaloup filter order of 31 was used. The interested frequency range of the filter was set to $[\omega_b, \omega_h] = [10^{-5}, 10^5]$ rad/s. To make things simple, the coefficient ω and K were set to one.

Below, simulation results for α equal to 1.0, 1.3 and $\frac{\pi}{4}$ for different values of ζ can be seen. In addition to a step response plot, a plot showing the location of the poles and theorized stability boundaries were created in order to better explain the findings. For each α, the two fractional-order pole system was simulated three times with different damping factor ζ. By using the conjecture (4.12), the first of the three simulations was done with a ζ that should make the system stable (blue), i.e. $\zeta > -\cos(\frac{\pi}{2}\alpha)$. The second was done with a ζ on the theorized stability boundary (blue-green), i.e. $\zeta = -\cos(\frac{\pi}{2}\alpha)$. And the last with a ζ that should make the system unstable (green), i.e. $\zeta < -\cos(\frac{\pi}{2}\alpha)$.

By inspection of figure 4.5a, 4.6a and 4.7a we observe that the blue graphs are all showing damped oscillations (the systems are stable), the blue-green graphs show oscillations with zero damping (the systems are marginally stable), while the green graphs show oscillations with negative gain, or amplification above one (the systems are unstable).

Looking at the plots of the poles in figure 4.5b, 4.6b and 4.7b in connecting with the step responses, we see the correctness in the drawing of the stability boundary and unstable areas. For $\alpha = 1.0$, which are the regular order in integer-order transfer functions, we see that the stability boundary is correctly drawn along the imaginary axis. For the cases of $\alpha = 1.3$ and $\alpha = \frac{\pi}{4}$ we see that the stability boundaries has been moved by a rotation about the origin. It should be highlighted that for $\alpha = \frac{\pi}{4}$, which is less than one, we observe that we now can have a stable fractional-order system with fractional-poles in the right half plane. Which, in itself, is an interesting result.

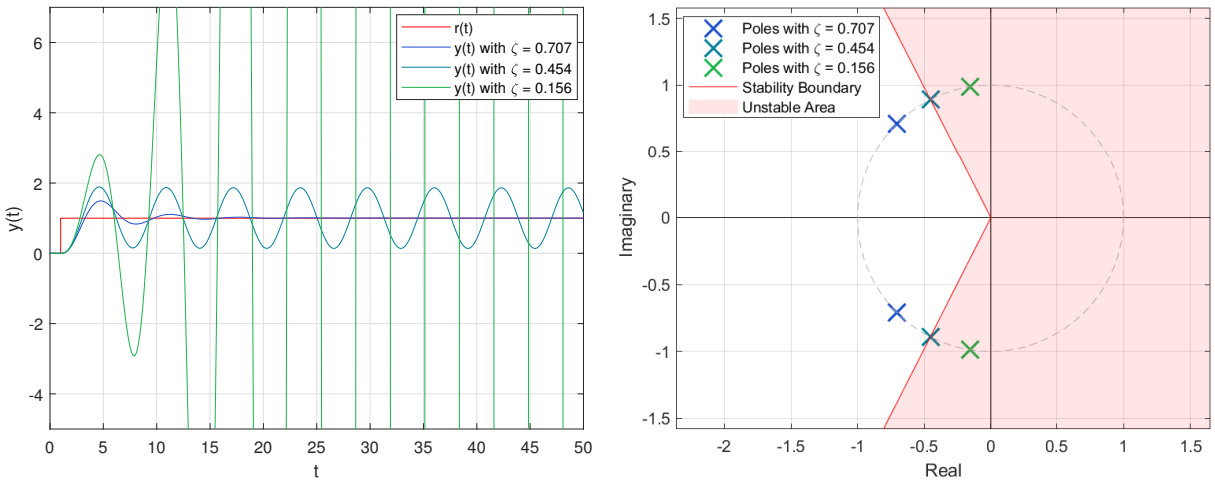
Rounding off this inquiry, it is clear that the simulation results are all in line with the proposed conjecture (4.12).



(a) Step responses of $H(s)$ for different ζ.

(b) Poles of $H(s)$ for different ζ.

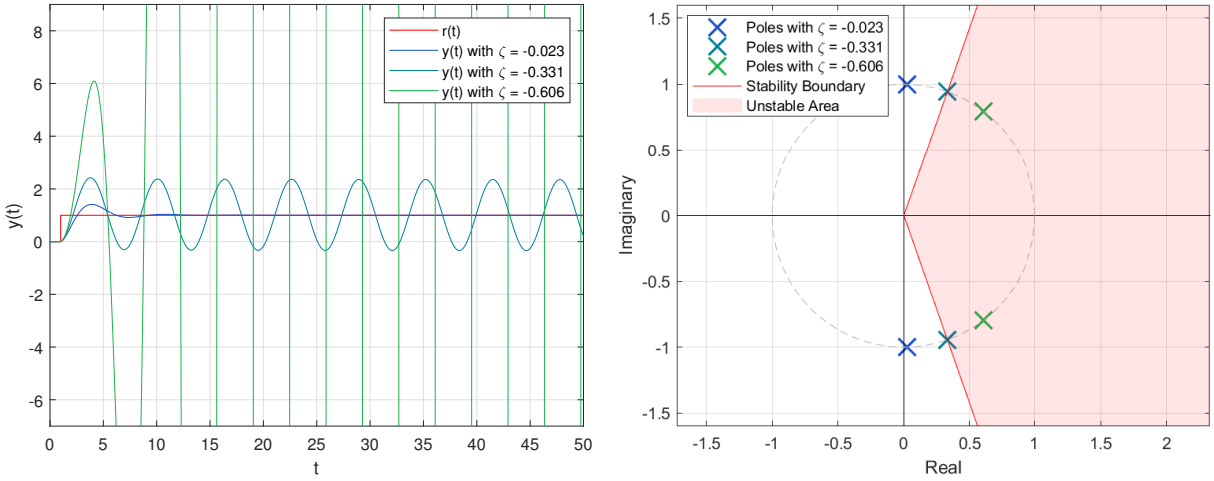
Figure 4.5: Stability inquiry of $H(s)$ through simulation. $\alpha = 1.0$.



(a) Step responses of $H(s)$ for different ζ.

(b) Poles of $H(s)$ for different ζ.

Figure 4.6: Stability inquiry of $H(s)$ through simulation. $\alpha = 1.3$.



(a) Step responses of $H(s)$ for different ζ.

(b) Poles of $H(s)$ for different ζ.

Figure 4.7: Stability inquiry of $H(s)$ through simulation. $\alpha = \frac{\pi}{4} \approx 0.7854$.

Chapter 5

Lateral Control of AFM: A Fractional-order approach

5.1 Short Survey

In recent years a lot of articles have been written on the topic of fractional calculus for use in control [6, 30–35]. Well known control schemes, like the PID, have been taken and modified with the complex theory of fractional calculus with the aim of creating better controllers. Articles are reporting on performance increases like higher bandwidth [36], but the increases are relatively small when compared to their integer-order variant.

A lot of the effort in the field of fractional-order control systems have been directed at the fractional-order PID controller as a natural extension of the well-known and well used PID controller [5, 6, 30–32, 37], but other fractional-order modified controllers have also been studied.

In [38] a fractional-order Positive Position Feedback (FO-PPF) compensator have been used in Active Vibration Damping of a rectangular carbon fibre composite plate with free edges. A PPF controller is a feedback compensator used for vibration damping with a second-order filter [39]. A block diagram of the scheme can be seen in figure 5.1. Here $0 < g < 1$ for stability. The carbon fibre composite plate was controlled by Macro Fibre Composite (MFC) transducers and vibration measurements were performed by a Laser Doppler Vibrometer (LDV). The FO-PPF controller was found to require less actuation voltage and seemed promising in reducing spillover effects from uncontrolled modes.

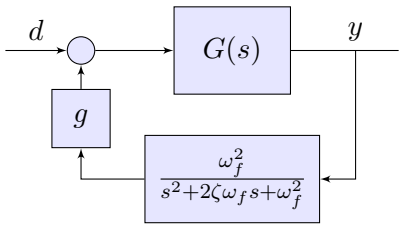


Figure 5.1: Block diagram of PPF compensator Scheme.

In [36], San-Millan et.al. augmented the Integral Resonant Controller (IRC) with a fractional-order integrator in the outer tracking loop and reports improved bandwidth on the control of a nanopositioner with co-located sensor-actuator dynamics and piezoelectric stack actuators.

In earlier works done at the ITK nanopositioning lab a H-infinity controller was developed for

the XE-70 AFM system. Order reduction was performed on the controller in an attempt at getting the controller to run faster on a real-time computer [15].

In [40] a comparison of several low-order control laws, like Integral Resonant Control (IRC), Positive Position Feedback (PPF), Integral Force Feedback (IFF) and passive shunt-damping, were conducted. The controllers were applied to a nanopositioner system and a comparison was done based on different criteria like system bandwidth and Root Mean Square Error (RMSE). Regulator parameters were found through a Nelder-Mead optimization approach.

The goal of this chapter is to present the author's work on developing and testing a set of fractional-order controllers for the lateral motion control of an AFM nanopositioning system. With the tuning method developed we will try to answer questions like: Do fractional-order controllers have any benefits over standard controllers used for AFM today? Can fractional-order controllers give us better performance in terms of bandwidth and damping of resonant modes when compared to integer-order controllers?

Taking inspiration from [38] and [40] several fractional-order variants on the PPF controller is studied, in addition to a fractional-order variant of the PID controller. Controller tuning is done through optimization in the frequency domain, and special focus has been put in the development of an algorithm for automatic stability assessment of a type of fractional-order systems, based on the Nyquist criterion.

5.2 Outline of Chapter

In section 5.3 a transfer function model of the AFM's lateral dynamics will be obtained through system identification. Section 5.4 and 5.5 presents the control structure and the regulators that are going to be tested. Section 5.6 will motivate and introduce the experimental tuning method developed and used in this thesis. An introduction and explanation of the genetic optimization algorithm is given in section 5.8, before the optimization problem is formulated in 5.9. The chapter is ended with an explanation of four MATLAB scripts that have been designed during the work with this thesis (5.10, 5.11, 5.12 and 5.13.).

5.3 System Identification of AFM xy-Scanner

A system identification of the AFMs lateral motion was conducted to obtain a model that could be used during regulator design and tuning. Frequency response data was obtained with the use of the signal analyzer SR780, in fast Fourier transform mode. The SR780s White noise generator was used as an input signal during the analysis. The resulting frequency response was averaged over 2000 samples to get a good representation of the system dynamics and reduce the impact of noise on the frequency response.

Several transfer function models, based on the frequency response data, were found through the use of MATLAB's system identification toolbox (`systemIdentification`). Models with varying accuracy and degree were obtained. This was done to have exact, computational heavy models for verification and less exact, computational lightweight models for calculations during regulator optimization. Frequency responses for two of the identified transfer function models are shown in figure 5.4 along with the experimental frequency response.

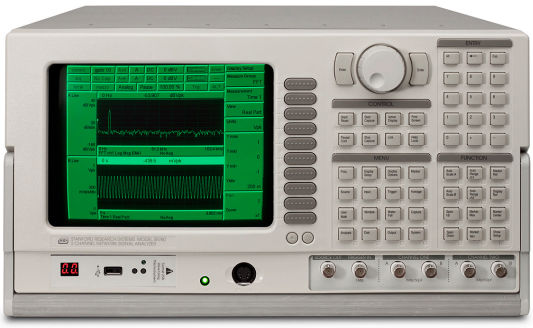


Figure 5.2: SR780 Signal analyser used for capturing the system frequency response.

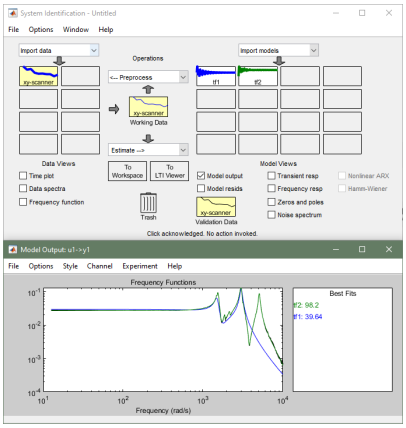


Figure 5.3: Screenshot of MATLAB System Identification Toolbox GUI used in finding approximate transfer functions from frequency response data

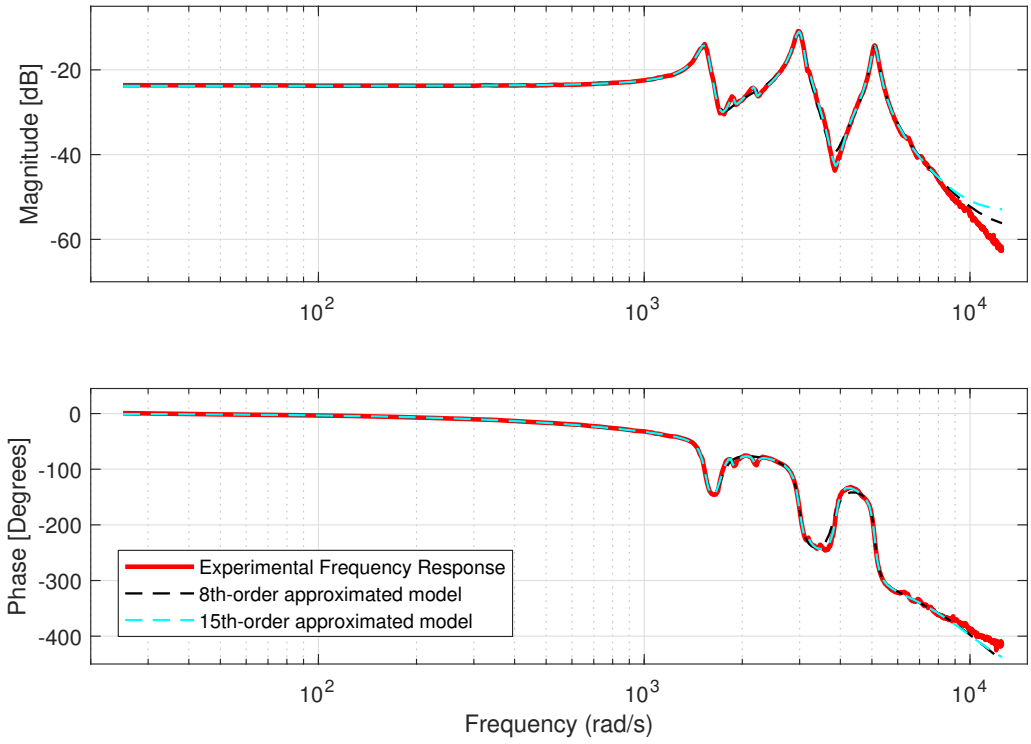


Figure 5.4: Frequency response of identified transfer functions vs. experimental frequency response data.

5.4 Control Structure and Design

Control of the xy-scanner dynamic of the AFM identified in section 5.3 is to be done through feedback control. The standard control structure shown in figure 5.5 is to be used in this report. Where $G(s)$ is the AFMs lateral motion dynamic, $C(s)$ is the forward controller transfer function and $F(s)$ is the backward controller transfer function.

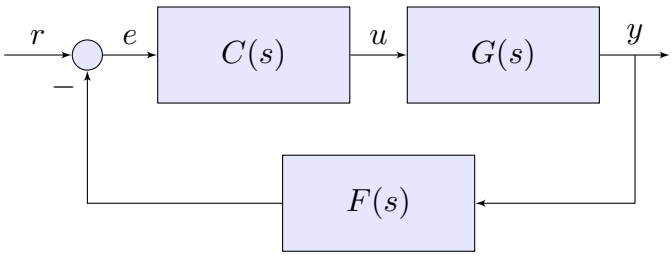


Figure 5.5: General control structure.

The open-loop transfer function $L(s)$ can be expressed as

$$L(s) = F(s)C(s)G(s), \tag{5.1}$$

while the sensitivity function $S(s)$ and complementary sensitivity function $T(s)$, respectively is expressed as

$$S(s) = \frac{e(s)}{r(s)} = \frac{1}{1 + L(s)} = \frac{1}{1 + F(s)C(s)G(s)}, \tag{5.2}$$

and

$$T(s) = \frac{y(s)}{r(s)} = \frac{C(s)G(s)}{1 + L(s)} = \frac{C(s)G(s)}{1 + F(s)C(s)G(s)}. \tag{5.3}$$

5.5 Controllers to Test and Analyse

The main focus of this work has been to test the effects and possible benefits of introducing fractional-order derivatives and integrals into controllers. To that end, two low-order controllers, often used in the literature in the control of nano-positioning applications, have been chosen as a basis and is to be augmented with fractional-order calculus. The chosen controller schemes are the PID-controller and the Positive Position Feedback (PPF) scheme for reference tracking. Their structure and some possible ways to augment the controllers with fractional-order calculus is treated in this section.

5.5.1 Fractional-order PPF

Positive Position Feedback (PPF) is used in vibration damping and can be combined with an integral control loop to provide damping and tracking. The control structure can be seen in figure 5.6.

The PPF scheme mentioned in [40] is comprised of two feedback loops. Here we will adopt the same notation as used in [40]. The inner feedback loop consists of the damping controller

$$C_d(s) = \frac{-k_d\omega_d^2}{s^2 + 2\zeta_d\omega_d s + \omega_d^2}. \tag{5.4}$$

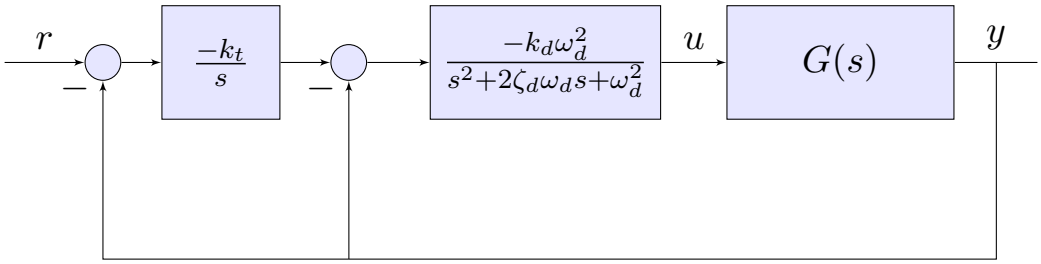


Figure 5.6: PPF damping and tracking control scheme.

This controller acts as a lowpass filter. The goal is to dampen the second, third, and so forth, resonant modes. Making the system resilient to out of bandwidth noise, also called spillover. The outer control loop is called a tracking controller and is given by the transfer function

$$C_t(s) = \frac{-k_t}{s}. \tag{5.5}$$

The tracking controller should enable good reference tracking of the nanopositioner. It should be noted that this controller scheme is indeed stable, even though the controllers C_t and C_d have negative gain. As long as k_t, k_d, ω_d, $\zeta_d > 0$.

Now, there are several ways that this control scheme can be changed or augmented to use fractional-order integrators and derivatives. The two most apparent ways, and the two ways that will be studied here, are

1. Introduce a fractional-order integrator in the tracking controller in-place of the first-order, integer-order integrator (Similar to what have been done in [36]).

2. Change s with s^β in the denominator of the damping controller (As done in [38]).

Table 5.1: PPF based fractional-order regulators

Name	C_t	C_d
PPF	$\frac{k_t}{s}$	$\frac{k_d\omega_d^2}{s^2+2\zeta_d\omega_d s+\omega_d^2}$
FO-PPF1	$\frac{k_t}{s^\alpha}$	$\frac{k_d\omega_d^2}{s^2+2\zeta_d\omega_d s+\omega_d^2}$
FO-PPF2	$\frac{k_t}{s}$	$\frac{k_d\omega_d^2}{s^{2\beta}+2\zeta_d\omega_d s^\beta+\omega_d^2}$
FO-PPF3	$\frac{k_t}{s^\alpha}$	$\frac{k_d\omega_d^2}{s^{2\beta}+2\zeta_d\omega_d s^\beta+\omega_d^2}$

Based on this, table 5.1 of different PPF based fractional-order controllers can be created. In the rest of this thesis, the names given in column one of table 5.1 will be used to refer to the different PPF based regulators.

The regulator structure in figure 5.6 can be changed to the standard feedback structure in figure 5.5 through the following equations

$$C(s) = C_t(s)C_d(s), \tag{5.6}$$
$$F(s) = 1 + C_t(s)^{-1}. \tag{5.7}$$

5.5.2 Fractional-order PID

Proportional Integral Derivative (PID) regulator is one of the most used regulators because of its simple form and its inherent ability to achieve both high bandwidth and low stationary deviation. As mentioned at the start of the chapter, the PID controller have been the base starting point for a lot of research into fractional-order control systems.

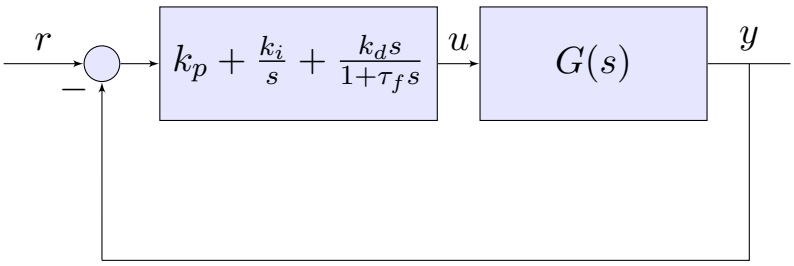


Figure 5.7: PID control scheme.

In this thesis the control structure in figure 5.7 will be used for PID controllers. This form can easily be brought to the form of figure 5.5 by setting

$$C(s) = C_{pid}(s), \tag{5.8}$$
$$F(s) = 1. \tag{5.9}$$

A PID-controller can be expressed in various, almost equivalent forms. Here, we will use the form in (5.10) where a first order low-pass filter have been added to the D-part of the regulator so that $C(s)$ is a proper transfer function.

$$C_{PID}(s) = k_p + \frac{k_i}{s} + \frac{k_d s}{1+\tau_f s} = \frac{(k_p\tau_f + k_d)s^2 + (k_p + k_i\tau_f)s + k_i}{s(s+\tau_f)}. \tag{5.10}$$

The PID controller can be augmented with fractional-orders in a lot of different ways, and a lot of different fractional-order PID controllers have been studied in the literature. Here, we will focus on the following form in addition to the regular integer-order PID controller:

$$C_{FOPID}(s) = k_p + \frac{k_i}{s^{\mu_i}} + \frac{k_d s^{\mu_i}}{1+\tau_f s^{\mu_i}} = \frac{(k_p\tau_f + k_d)s^{(\mu_i+\mu_d)} + k_p s^{\mu_i} + k_i\tau_f s^{\mu_d} + k_i}{\tau_f s^{(\mu_i+\mu_d)} + s^{\mu_i}}. \tag{5.11}$$

5.6 Automatic Tuning of Controllers

Tuning of fractional-order regulators is not a trivial task. A lot of effort has been put into tuning of fractional-order controllers in the literature. A handful tuning rules have emerged for specific controllers like the fractional-order PID controller (FO-PID) [6, 41]. While much of the research has been approaching the tuning problem with optimization techniques [36, 42–44]. This thesis joins the second approach to tuning, namely through optimization.

Introducing fractional-order derivatives or integrators into the linear transfer function models makes the controller tuning optimization a highly non-linear problem. And it is believed that a lot of local optimal points exists in the controller parameter search space. At the start of the work presented in this thesis, traditional optimization methods like interior-point, active-set and Trust-Region-Reflective were tried out by using MATLAB and the function `fmincon` and `GlobalSearch` in the controller tuning. They were found to converge to local minima and/or not able to satisfy the stability constraint imposed on the problem. Therefore, other optimization techniques were considered. First the exhaustive grid search algorithm `MultiStart` in MATLAB's global optimization toolbox were tried out. But given the number of parameters to optimize and the vast search space this method was found to be too time consuming.

Moving forward, biology inspired, heuristic optimization techniques [45, 46] were studied in the hopes that they could be a better fit for the problem of tuning a fractional-order controller. The use of such heuristic optimization techniques for tuning of fractional-order control systems is not a new idea. The particle swarm optimization technique (PSO) has been used in the tuning of a fractional-order PID controller in [47], while the Genetic algorithm (GA) and the Big Bang Big Crunch algorithm (BBBC) have been used in the tuning of the same fractional-order PID controller in [43].

In the end, the genetic algorithm implementation in MATLAB's global optimization toolbox was tried out and it was found adequate for the controller optimization to be conducted. Thus, the genetic algorithm was chosen as the optimization method to use. It is highly likely that other heuristic optimization algorithms like (but not limited to) Particle Swarm Optimization (PSO), Simulated Annealing (SA) and the Gravitational Search Algorithm (GSO) would work just as good.

5.7 Stability Constraint

When using optimization to look for good controller parameters it is necessary to constrain the search to stable solutions, so as not to find unstable solutions. For integer-order linear systems the eigenvalues or poles are usually studied to this end, and for non-linear systems a Lyapunov function can be found and analysed.

This thesis is only considering fractional-order linear systems in the s-domain so the most intuitive is to calculate and study the values of the closed loop poles. However, it turns out that calculating the roots of the denominator polynomial of a fractional-order closed loop system is no easy feat.

In the work with this thesis a script for automatic calculation of Nyquist diagram encirclements around -1 point, called `assess_stability`, have been developed. The author does not currently know about other scripts doing the same but given the number of engineers and scientists in the field of control, and the sheer number of publications, it is quite likely that similar scripts do exist. The `assess_stability` function is explained in section 5.11.

A script for the calculation of the frequency response of a fractional-order transfer function with adaptively changing frequency step-length, called `astep_fotf_freqresp`, has also been implemented. And it serves as one of the underlying ways to calculate the frequency response used in `assess_stability`. This makes the calculation of the frequency response of a fractional transfer function fast and adaptive. It should manage to capture the important dynamics in the frequency response and at the same time reduce the number of calculated points to a somewhat minimum of what is required. The function `astep_fotf_freqresp` is explained in section 5.10.

This adaptive stepping frequency response calculation function is also the backbone of the created function `nyqlog_fotf` which calculates the frequency response of a fractional-order transfer function given as a `fotf` object and plots a logarithmic Nyquist diagram of the frequency response, making stability assessment of fractional-order linear systems a piece of cake given you know the number of unstable poles in the open-loop system. The `nyqlog_fotf` function is explained in section 5.12.

5.8 The Genetic Algorithm (ga)

The genetic algorithm (GA) is a population based, stochastic algorithm designed to replicate the natural selection found in nature. The algorithm starts out with a randomly generated population, where each individual consists of a set of values that are referred to as the individual's genetic material. Through the course of the algorithm, the genetic material of the population will change from one generation to the next by mimicking the mechanisms observed in biological evolution like natural selection, crossover-reproduction and mutations.

The goal is to improve the fitness score calculated from a given objectivity function and not violate the given constraints. In the context of regulator tuning and optimization, the genetic material of an individual is a given set of regulator parameters.

It should be noted that GA is a heuristic optimization method, and because of that, no guarantee of finding a global optimal solution is given. On the other hand, controllers do not usually need global optimal parameters to be functional or useful. Therefore, the use of these types of optimization techniques is usually fine when tuning complex low order controllers.

The genetic algorithm implemented in MATLAB's global optimization toolbox [48] is used in this thesis as mentioned above. It is called `ga` for short. This section only aims at giving the reader an introduction and a feel for the genetic algorithm. The main parts of the algorithm will be commented on below and the different policies as to how selection, crossover and mutation work is going to be quickly explained. For an explaination of all the different fitness scaling, selection, crossover and mutation policies, or for a more in-depth explaination of some parts of the algorithm, refer to the global optimization toolbox manual [48].

It was earlier stated that the genetic algorithm is a population-based method. And as such it consists of individuals of a new generation, which are referred to as children of the previous generation, are created in three ways:

1. **Elite Children** - The N best individuals from the previous generation is added to the new generation. Ensures that the best solution that have been found will always survive and be moved over to the next generation.

2. **Crossover children** - The genetic material of two individuals in the previous generation is mixed to create a child in the new generation. Individuals with a genetic material scoring high usually have a greater chance at being picked as parents.

3. **Mutation Children** - The genetic material of a single individual is randomly manipulated to create a new individual in the new generation. Mutations adds diversity to the populations and enables the algorithm to search in unexplored areas of the search space.

5.8.1 Fitness Scaling Policy

Each individual of a generation is given a score through the calculation of the objectivity function. But before parents of the next generation can be chosen the scores of the individuals needs to be compared and scaled to each other to create fitness scores. To this end, several different comparison methods exists.

For the regulator optimization the rank fitness scaling policy (`fitscalingrank`) have been chosen. Using the rank fitness scaling policy, the individuals are first sorted in descending order based on the objectivity function score to create a ranking list. Then they are awarded a fitness score equal to

$$S(r) = \frac{1}{\sqrt{r}}, \tag{5.12}$$

where r is the rank of the individual. This fitness scaling policy was chosen over other policies because it was able to handle objectivity function scores of infinity. Which is a possible return value because of the way the stability constraint is implemented.

5.8.2 Selection Policy

The selection option of the genetic algorithm specifies how individuals of one generation is chosen as parents for the individuals of the next generation. Several standard policies are implemented in the `ga` algorithm.

The stochastic uniform (`selectionstochunif`) selection policy is used in the regulator optimization. Using this selection policy, a line is created by combining a line segment from each individual. The line segments have a length equal to the fitness score of the corresponding individual. Then the algorithm moves along the line with equal steps and for each step it takes it uses that individual as a parent for either a new crossover child or a mutation child in the next generation. The effect of this selection policy is that better fitness scoring individuals are more likely to have their genetic material pass on to the next generation than individuals scoring low.

5.8.3 Elite Options

The number of elite children, i.e. number of individuals in a generation that will have their genetic material moved over to the next generation, can be freely set by the `EliteCount` option of the `ga` algorithm. An elite count of 5% of the population size have been used when tuning regulators in this work, and it works fairly well.

5.8.4 Crossover Options

The number of crossover children in a generation, i.e. number of individuals in a generation that have genetic material that is a mix of the genetic material of two parents, is specified as

a fraction of the total population size with the option `CrossoverFraction`. In this work a crossover fraction of $60 - 80\%$ have been used.

There are usually many ways the genetic material of two parents can be combined to create the genetic material of a child. Several different crossover policies exist in MATLAB's `ga` algorithm. In this work the `crossoverscattered` policy have been used. This policy consists of doing a coin toss for each of the values in the genetic material. If heads, take the value of the first parent and give to the child, if tails, take the value of the second parent and copy that value to the child. Effectively giving the child a random combination of the values of the parents.

5.8.5 Mutation Options

The number of mutation children in a generation is $N_{mutation} = N_{population} - N_{elite} - N_{crossover}$, i.e. the rest of the population after elite children and crossover children have been removed.

Like crossover policies, there exist many ways of randomly alter the genetic material of one individual through mutation. The mutation policy used in this work has been the adaptive feasible mutation policy (`mutationadaptfeasible`). It has been used primarily because it is the only `ga` algorithm implemented mutation policy that supports bounded problems. The adaptive feasible policy works by randomly generating a direction and a step length for a vector composed by the genetic material and then alter the genetic material with that vector, while at the same time satisfying bounds and linear constraints.

5.9 Formulation of the Optimization Problem

The following optimization problem was formulated for tuning of fractional-order and integer-order regulators presented in section 5.5.1 and 5.5.2. The objectivity function $f(x)$ was formulated based on [40] and [36]. Boundaries on the controller parameters x were introduced to reduce the size of the search space and force the open-loop system to remain stable. In addition, a set of non-linear constraints were included to enforce closed-loop stability of the system.

$$\min_x f(x) \quad \text{s.t.} \begin{cases} c_{eq}(x) = 0, \\ c_1(x) \leq 0, \\ c_2(x) \leq 0, \\ x_{lower} \leq x \leq x_{upper}. \end{cases} \tag{5.13}$$

Two different objectivity functions were tried out and used in the optimization in this work. The first objectivity function, $f_1(x)$, aims at making the magnitude of the closed-loop frequency response, or complementary sensitivity function, $T(s)$ as flat as possible around the 0-dB line. The objectivity function can be expressed mathematically as

$$\begin{aligned} f_1(x) &= || \, 1 - |T(j\omega, x)| \, ||_2 \\ &= \left(\sum_{k=0}^{N} \left(1 - |T(j\omega[k], x)|\right)^2 \right)^{\frac{1}{2}}, \end{aligned} \tag{5.14}$$

where it is presented in a simple norm variant and on the discrete form used in implementation. N is the number of frequency samples used in the evaluation, while $w[k]$ for $k = 1, 2, \ldots, N$ is a non-decreasing vector of frequency values used in the evaluation of $T(j\omega, x)$.

The second objectivity function, $f_2(x)$, is an extension of the first objectivity function with the additional objective of maximizing system bandwidth ω_{bw}. The objectivity function can be expressed mathematically as

$$\begin{aligned} f_2(x) &= (1-\gamma)\left|\left|\frac{1-|T(j\omega,x)|}{\omega_{bw}}\right|\right|_2 - \gamma\omega_{bw} \\ &= (1-\gamma)\frac{1}{N_{bw}}\left(\sum_{k=0}^{N_{bw}}(1-|T(j\omega[k],x)|)^2\right)^{\frac{1}{2}} - \gamma\omega[N_{bw}], \end{aligned} \tag{5.15}$$

where N_{bw} is an index into the frequency vector where the bandwidth is found and $\gamma \in [0, 1]$ is a parameter used in the weighting of the two sub-objectives, namely flat closed-loop response and max bandwidth. The bandwidth of the system is defined as the point where the magnitude of $T(s)$ falls below the -6 dB line for the first time when moving from $\omega = 0\,\text{rad/s}$ to $\omega = \infty\,\text{rad/s}$.

Moving on to comment on the objectivity constraints and boundaries. No linear constraints were added to the problem, only a set of non-linear constraints. The non-linear constraints will be explained shortly, but first some comments on the boundaries. As mentioned above, boundaries on x were introduced to reduce the size of the search space and force the open-loop system to remain stable. For most of the controller parameters, this amounted to forcing the parameters to stay above zero. In addition, the parameters representing fractional-orders were forced to stay below two, in order to keep the open-loop system stable.

As mentioned, a set of non-linear constraint were added to the problem to enforce closed loop system stability during the optimization. These constraints were calculated with the help of the `assess_stability` function written by the author. The `assess_stability` function is bases on automatic calculation of the Nyquist stability criterion and is presented in section 5.11. By using a flag (`isStable`) which tells whether the system is stable or not, and a set of lower and upper stability gain margins (GM_{low} and GM_{high}) that is returned by the `assess_stability` function, the following equality and inequality constraints are formulated for the optimization problem

$$c_{eq}(x) = b(1 - \text{isStable}) = 0, \tag{5.16}$$

$$c_1(x) = \Delta k - \text{GM}_{\text{high}} \leq 0, \tag{5.17}$$

$$c_2(x) = \text{GM}_{\text{low}} + \Delta k \leq 0, \tag{5.18}$$

where b is a large positive constant making it costly for the system to be unstable, and Δk is the stability gain margin in decibel that the optimized system should at least have. A wanted stability gain margin of 6 dB for Δk was used for all runs of the optimization tuning method, and gave fairly good results.

The implementation of the objectivity functions and the stability constraints can be found in listing C.2 and C.3 of appendix C, respectively.

5.10 Adaptive Frequency Response Calculation (`astep_fotf_freqresp.m`)

At first, when starting to develop the optimization-based tuning method that is described in this thesis, the `fotf/bode` function in MATLAB overloaded for fractional order systems developed by Xue [5] was used to calculate frequency responses. This function takes a fractional-order transfer function described by a `fotf` object (from the `fotf` toolbox developed by Xue) and an array of frequencies where the transfer function should be evaluated. To evaluate the transfer function and get a sufficiently smooth graph, a huge number of frequency values must be calculated in order to pick up all the fast-changing dynamics in frequency domain. For checking stability through Nyquist's criterion during the tuning process, the frequency response of the transfer function needs to be calculated many times over. And the exhaustive calculation that is needed with the `fotf/bode` function was found to be too slow.

Therefore, a MATLAB function for faster and smarter calculation of the frequency response, called `astep_fotf_freqresp`, was developed. The function works by adaptively changing the step size between consecutive frequency values where the frequency response values shall be calculated. The most important aspects and steps of the functions are explained below. The complete code can be found in listing C.6 in appendix C.

The Steps of the Function

1.) Function interface
The `astep_fotf_freqresp` function takes a fractional-order transfer function of the `fotf` object type to calculate the frequency response from, and an optional set of parameters `w_low` and `w_high` to specify a frequency range for the calculation if this is desired. The function returns the frequency response in both complex form (`re_out` and `im_out`) and phase-magnitude form (`mag_out` and `phase_out`) in addition to an associated array of frequencies (`w_out`). The returned `phase_out` array is given in degrees and is not restricted to the $[-180°, 180°]$ domain. The returned `mag_out` array is given in decibel.

The function can be invoked with or without the `w_low` and `w_high` parameters. If only a transfer function is given, the interesting frequency range for calculation is set to $[0, \infty]$.

2.) Initial calculations
One of the first things that happen in the function is the extraction of the transfer function coefficients and fractional-orders from the `fotf` object by the means of the `fotf/fotfdata` function.

These system parameters are then used to find the frequency response magnitude and phase at the frequency limits $w \to 0$ and $w \to \infty$.

$$\lim_{\omega \to 0} |L(j\omega)| = \left| \frac{b_0 (j\omega)^{\beta_0}}{a_0 (j\omega)^{\alpha_0}} \right|, \tag{5.19}$$

$$\lim_{\omega \to 0} \angle L(j\omega) = \angle \left(\frac{b_0 (j\omega)^{\beta_0}}{a_0 (j\omega)^{\alpha_0}} \right). \tag{5.20}$$

$$\lim_{\omega\to\infty} |L(j\omega)| = \left| \frac{b_m(j\omega)^{\beta_m}}{a_n(j\omega)^{\alpha_n}} \right|, \tag{5.21}$$

$$\lim_{\omega\to\infty} \angle L(j\omega) = \angle \left(\frac{b_m(j\omega)^{\beta_m}}{a_n(j\omega)^{\alpha_n}} \right). \tag{5.22}$$

The idea is as follows, with $w \to 0$, only the part of the nominator polynomial and the part of the denominator polynomial with the lowest orders is contributing. They are the dominating terms of the fractional-order transfer function and decides the values of the magnitude and phase at the lower limit. With $w \to \infty$, the part of the nominator and denominator with the greatest orders are dominating the transfer function.

3.) Main calculation loops
The frequency response is calculated in two rounds. A parameter `startpoint_w` is used to split the total frequency interval $\omega \in [0, \infty]$ into two. The parameter can be freely set. The frequency response for frequencies in the range `startpoint_w` to infinity is calculated first. Then the frequency response for the frequency range `startpoint_w` to zero is calculated. In both the loops the calculation is started from `startpoint_w` and then moves towards zero or infinity. The two calculation loops are almost identical. The content of each loop can be roughly divided into three parts that will be explained below.

3a.) Frequency response calculation at a point
The frequency response of the fractional-order transfer function is calculated at a point ω with the helper function `calc_freq_resp` at each iteration of the loop. This calculation is done by using the extracted coefficients and fractional-orders from step 2.

3b.) Validation of the calculated point and adaptive change in frequency step
The feasibility of the newly calculated frequency response value at the point ω is checked. For the total calculated frequency response to be a smooth curve two conditions must be meet:

i. The difference in phase (or angle), Δd, between two consecutive lines, were one is drawn between the two previously calculated points and the other is drawn between the previous point and the newly calculated point, must not be above a predefined limit.

ii. The change in magnitude since the last calculated frequency response value must not too great relative to the current magnitude value.

These conditions can be expressed in equation form like

$$\Delta \mathrm{d}_j = |\operatorname{atan2}(\mathrm{im}_j - \mathrm{im}_{j-1},\, \mathrm{re}_j - \mathrm{re}_{j-1}) - \operatorname{atan2}(\mathrm{im}_{j-1} - \mathrm{im}_{j-2},\, \mathrm{re}_{j-1} - \mathrm{re}_{j-2})\ |, \tag{5.23}$$

$$\Delta \mathrm{d}_j < \mathrm{d1}, \tag{5.24}$$

$$\mathrm{d2} < \Delta \mathrm{d}_j, \tag{5.25}$$

and

$$10\log_{10}\left[(\mathrm{re}_j - \mathrm{re}_{j-1})^2 + (\mathrm{im}_j - \mathrm{im}_{j-1})^2\right] < 10\log_{10}\left[\mathrm{re}_j^2 + \mathrm{im}_j^2\right] - \mathrm{y}, \tag{5.26}$$

where re_j and im_j are the real and imaginary part of the newly calculated frequency value and re_{j-1}, im_{j-1}, re_{j-2} and im_{j-2} are the real and imaginary part of the two previously calculated and validated points. The limit values for the first condition is defined as d1 = $\frac{\pi}{180}$d and d2 = $\frac{\pi}{180}$(360-d), where `d` [degrees] is a configuration parameter that can be manually set inside the script in order to decrease or increase the area of acceptable new frequency values. `y` [dB] is a similar configuration parameter for the second condition. (5.24) is used to check that the change in phase between two consecutive line segments is less than `d1`. The additional check in (5.25) is used to handle the possibility of angle wrapping.

If both of these conditions are not satisfied, the newly calculated frequency response value is thrown away and the frequency response of the transfer function at a new frequency point, closer in frequency to the previously accepted value, is calculated. This new frequency point is found by first changing the step length variable $\Delta\omega$ and then add the new step length value to the previously accepted frequency value, in order to find a new ω where the frequency response of the transfer functions can be calculated, and were the likelihood of the point passing validation has grown. $\Delta\omega_{\text{new}}$ and ω_{new} are calculated as follows when a frequency response value is rejected:

$$\Delta\omega_{\text{new}} = \frac{\Delta\omega}{\alpha}, \tag{5.27}$$

$$\omega_{\text{new}} = \omega_{\text{prev}} + \Delta\omega_{\text{new}}. \tag{5.28}$$

The parameter $1 < \alpha$ is a configurable constant that is used to change the frequency step variable $\Delta\omega$.

However, if one of the conditions are satisfied the newly calculated frequency response value is accepted and saved, and the algorithm moves on to the calculation of a new value. When a frequency response value is accepted the step length $\Delta\omega$ is increased. This is done in order for the algorithm to speed up in areas where there is less change in phase and magnitude. The new values for $\Delta\omega$ and ω after the acceptance of a point is

$$\Delta\omega_{\text{new}} = \alpha\,\Delta\omega, \tag{5.29}$$

and

$$\omega_{\text{new}} = \omega_{\text{prev}} + \Delta\omega_{\text{new}}. \tag{5.30}$$

The change of step is done through multiplication and division in order to quickly change the magnitude of the $\Delta\omega$ variable and then quickly react to rapid changes in the frequency response.

In addition to the above main conditions, an additional check for floating-point relative accuracy in ω had to be added. This prevents the frequency step value $\Delta\omega$ from having a value that would be considered as zero when $\Delta\omega$ is added to ω. Which would completely halt the algorithm.

3c.) Loop termination check

The third and last part of the main loops checks if the frequency calculation is done. In case a frequency interval has been specified when calling the function the algorithm just checks if the frequency limit have been reached, and terminates in the event that it has. In case a frequency interval has not been specified, the algorithm assumes that $\omega \to 0$ and $\omega \to \infty$ are the end points and looks for a termination pattern. From the initial calculations, we know the phase and magnitude values of the frequency response when $\omega \to 0$ or $\omega \to \infty$. Namely (5.19) and (5.20), or (5.21) and (5.22). By comparing the ”end dynamics” with the current response

and frequency the stopping criteria explained below can be used to make it probable that the algorithm have stepped through all the important dynamics and have come sufficiently close to the theoretical ”end dynamics”. The word probable is used here because the method can be fooled by the calculated frequency response to end too soon. Not stepping through all the important dynamics. This can usually be fixed by changing the value `startpoint_w`, but the user needs to be aware of this little subtlety of the algorithm to get correct behaviour and a frequency response covering all the relevant dynamics.

Now, to the stopping criteria. We will only explain it for the case of $\omega \to 0$, but the same is done for $\omega \to \infty$. First the algorithm checks if the phase of the newly calculated frequency response value is within the plus/minus 5 degree area around the $\lim_{\omega\to 0} \angle L(j\omega)$ value, which is the approximated phase value at $\omega = 0$. If it is, then a magnitude sum variable is increased with the difference in magnitude between the current magnitude value and the last calculated and saved magnitude value. If not, the magnitude sum variable is reset to zero. So, as long as the phase of newly calculated values keeps within the bounds, the magnitude sum variable is increased further. And when this value turns greater than a specific configuration parameter, that is set to $\pm$30dB during the initialization part, the algorithm concludes that it is approaching the ”end dynamics”, and the frequency calculation loop is exited.

The general thought is that if the frequency response has had the approximately same phase angle ($\pm$5degrees), around the theoretical phase value $\lim_{\omega\to 0} \angle L(j\omega)$, over a sufficiently long enough rise or fall (decided by the value of $\lim_{\omega\to 0} |L(j\omega)|$) in magnitude, then it is probably approaching the ”end dynamics” and have already stepped through all the important dynamics.

In cases where the dominating nominator and denominator parts of the transfer functions have the same orders at $\omega \to 0$ or at $\omega \to \infty$, the end limit frequency responses have finite magnitudes. This is handled somewhat different than when $\omega \to 0$ or $\omega \to \infty$ leads to a magnitude tending towards zero or infinity. In these cases, the algorithm checks if it has moved sufficiently close to the ”end dynamics”, and exits the frequency response calculation loop if it has.

4.) Adding points for $\omega \to 0$ and $\omega \to \infty$
After the two quite similar main loops have finished, some pre-processing of the frequency response vectors are done. Among other things, points for $\omega = 0$ and $\omega = \infty$ are added to the data.

5.) Returning the calculated frequency response
After the post processing in step 4, the frequency response data from the two intervals $\omega \in [0, \texttt{startpoint\_w}]$ and $\omega \in [\texttt{startpoint\_w}, \infty]$ are combined before returning.

Final Remarks

The code for the function can be found in listing C.6 in appendix C. We end the explanation of the `astep_fotf_freqresp` function with a note on pros and contras of the algorithm:

Pros: The algorithm does have a lot of overhead, but compared to an exhaustive frequency response calculation method like the overloaded `fotf/bode` function in the `fotf` toolbox [5], it needs to calculate relatively few frequency response values, making it superior in terms of speed, while at the same time guaranteeing sufficient resolution in areas where the phase and magnitude might change quickly relative to the frequency.

Cons: Since the frequency response values are not evenly spaced in frequency this response

is not suitable for the use in cost functions that should weight the different values equally in frequency. A pre-processing of the data where additional points are added or the points are scaled relative to the step length between the frequencies, could make it work for this type of calculations as well.

5.11 Stability Assessment with Nyquist (`assess_stability.m`)

Checking stability of linear systems are usually done by looking at the eigenvalues or poles of the system. If the system is of a sufficiently large order, and the denominator of the system's transfer function is a large polynomial then explicitly calculating the poles of the system might prove difficult. Even using functions like MATLAB's `roots` function might be troublesome because it may take a long time to calculate, or the result may be inaccurate.

Usually during regulator optimization, system stability needs to be enforced. If not, the controller parameters found will almost always be that of an unstable controller. Something we usually do not want. If the closed loop system is of high order, it can prove quite challenging or even infeasible to check stability by looking at the poles of the system and ensure that they have a real part less than or equal to zero. Finding the roots of fractional polynomials are even trickier than finding the roots of a normal integer polynomial. So, finding the poles of fractional-order transfer functions ends up being a quite challenging feat. And using functions like MATLAB's `roots` ends up taking too much time for it to be an effective stability check in a complex optimization problem.

Another much used way to check stability of closed loop systems is through Nyquist's stability criterion, presented in section 4.2. By knowing the number of unstable poles in the open-loop system and knowing the number and directions of encirclements of $\angle h_0(j\omega)$ around the point $(-1, 0)$, the number of closed loop poles can be found.

In this section, a MATLAB function for automatic calculation of gain margins for a fractional-order transfer function is presented. The function was termed `assess_stability` and uses the `fotf` object from the fotf toolbox [5] to represent a fractional-order transfer function. The function is based on the Nyquist stability criterion and can automatically find the areas of gains that can stabilize a closed loop system, if such a stabilizing gain exists that is. When calling the function, the number of open-loop unstable poles must be given for the function to calculate stability correctly.

The Steps of the Function

The function can be roughly divided into eight steps. The steps will be quickly explained below. For more in-depth details of the function, the script implementing the function can be found in appendix C in listing C.5.

1.) Calculation of frequency response
Firstly, a vector of frequency response values are calculated with the `astep_fotf_freqresp` function explained in section 5.10. This gives a sufficiently smooth frequency response curve that captures the relevant frequency dynamics of the analysed system, while at the same time not taking too long to evaluate.

2.) Finding real-axis intersection points
Secondly, looking at the frequency response in the complex plane, the points where the frequency response curve intersects the real axis is found or approximated through linear interpolation.

Moving along the frequency response vector, all the point where the imaginary part switches sign from one value to the next is found. Then doing a linear interpolation between the two points on each side of the real axis, a good approximation for the real-axis crossing point is found.

3.) Dividing the real axis into segments
Now, a set of N real axis intersection points have been found. Sorting the points in terms of their real value in ascending order and calling the points for x_i, we now have $x_1 < x_2 < \cdots < x_{N-1} < x_N$.

The real axis can now be divided into $N + 1$ intervals or segments, where the first interval stretches from $-\infty$ to x_1, the second interval from x_1 to x_2 and so on. All the way to x_N to ∞.

4.) Counting number of encirclements around segments
Proceeding, the algorithm now adds the two extra points $lim_{\omega\to 0^+}|L(j\omega)|$ and $lim_{\omega\to\infty}|L(j\omega)|$ to the vector of intersection points and sorts the point in ascending order with respect to frequency. Moving along the sorted vector of intersection points with the added information of real axis crossing direction (up or down), the number of encirclements around each of the real-axis segments can be found.

5.) Searching for stable regions with the Nyquist criterion
Now, going through the list containing the number of encirclements for each of the real-axis intervals we search for numbers matching the number of open-loop poles given as argument to the function. This step can be seen upon as applying the Nyquist stability criteria formula (4.10) for each of the segments or intervals. Finding the segments where the point (-1,0) can be located in order to have a closed-loop stable system.

If no interval with a matching encirclement number exist, the system can't be stabilized by just changing the gain, and the method fails to yield results. However, if one or more matching encirclement numbers exist the associated interval(s) are deemed stable and saved for further analysis.

6.) Calculation of gains which stabilize the closed-loop system
Now, for each of the stable intervals found, the distance from both the interval start point and the interval end point and to the (-1,0) point is calculated. These distances can be seen upon as the gain margins of the system, and based on their signs and values, the stability of the analyzed system with a unit negative feedback can be decided.

7.) Returning results
Lastly the function returns information regarding stability of the analyzed system with a unit negative feedback. This comes in the form of a flag which is one for stable closed-loop system and zero if the closed-loop variant of the given open-loop system is unstable. In addition, sets of gain margins that can be used to stabilize the system is returned. These gain margins are used in the constraint formulation of the proposed tuning method.

5.12 Fractional-order Logarithmic Nyquist Diagram (`nyqlog_fotf.m`)

A script for the plotting of a Logarithmic Nyquist diagram, for fractional-order transfer functions was implemented by the author of this thesis in order to visualize the stability analysis of fractional-order systems. The script can be found in listing D.1 in appendix D. The idea and inspiration for the diagram was taken from Trond Andresen [7], who also gave the diagram the name "Logarithmic-Amplitude Polar diagram".

In the event that Nyquist's stability criterion is valid for the kind of fractional-order transfer functions discussed in this thesis, the diagram can be used to check closed-loop stability of some fractional-order system by counting the number of encirclements around the point $(-1, 0)$ and applying Nyquist's stability criterion (4.10).

The script uses the `astep_fotf_freqresp` function presented in section 5.10 to evaluate the frequency response and then plots this in a polar coordinate system where the distance from the origin is on a logarithmic scale. This way of plotting the frequency response with a logarithmic magnitude makes stability considerations and inspections of the graph simpler than it would have been if done with a regular Nyquist diagram with standard real and imaginary axes. The reason for this is the possible span in magnitude for different frequency values, which results in a need to zoom in and out in a regular Nyquist diagram in order to see all the possible loops and consider the full dynamics of the system. In a logarithmic-amplitude polar diagram, however, there is no need to zoom in and out in order to see all the loops and all the dynamics, simplifying the inspection process.

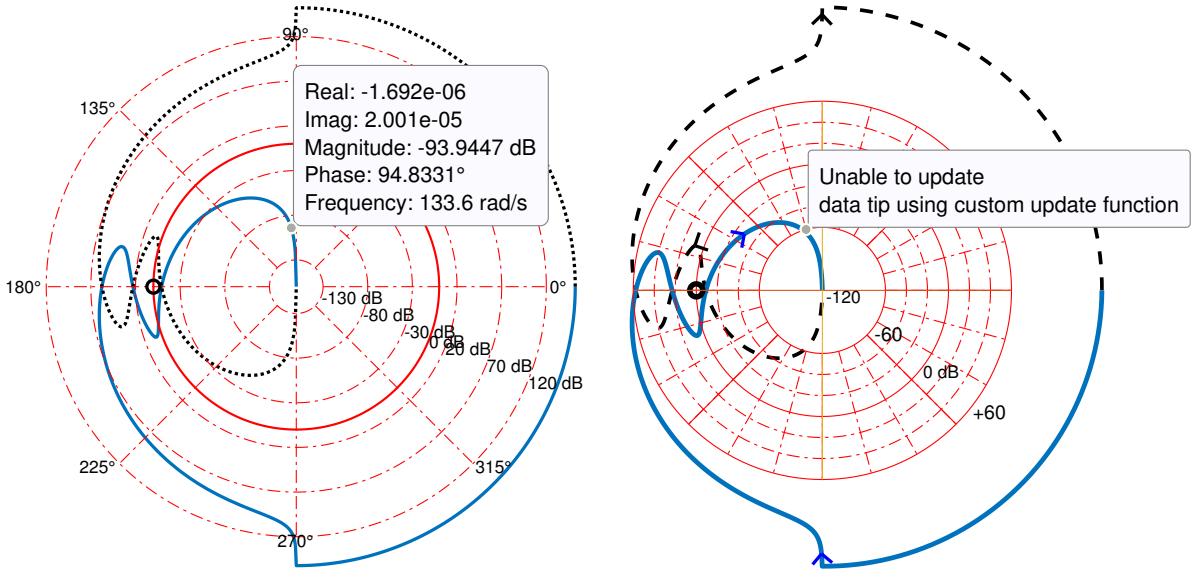
The axes of the diagram scales adaptively to the frequency response and in the event that $\lim_{\omega\to 0} |L(j\omega)|$ and/or $\lim_{\omega\to\pm\infty} |L(j\omega)|$ is equal to ∞ or 0 the frequency response and inverse frequency response are connected with semi-circles at infinity or connected at zero in order to close the loop. Easing the overall stability analysis. In addition, a graph tooltip callback function was implemented to enable the inspection of the frequency response data on the graph. The tooltip shows the frequency in radians per second and the frequency response data in both real-imaginary and magnitude-phase formats.

Comparison of `nyqlog_fotf` and `nyqlog`

In figure 5.8, two logarithmic-amplitude polar diagrams (logarithmic Nyquist diagrams) of the transfer function (5.31) have been plotted. The diagram to the left is plotted with the `nyqlog_fotf` function that were implemented during the work with this thesis, while the diagram to the left is plotted with the `nyqlog` function written by Trond Andresen [7, 49].

$$G(s) = \frac{200(1+3s)(1+2s)}{s(1+50s)(1+10s)(1+0.5s)(1+0.1s)}. \tag{5.31}$$

This comparison in figure 5.8 shows the agreement of the two diagrams and the correctness of the `nyqlog_fotf` function implemented in this thesis. The test transfer function (5.31) used here is from [7] and is a regular integer-order transfer function. This shows that the `nyqlog_fotf` function can handle regular integer-order transfer function as well as fractional-order ones. In figure 5.8a we can see the added functionality of a tooltip which shows information about the frequency response at the given point. Another thing to note is that arrows showing the direction of increasing frequency have yet to be added for the `nyqlog_fotf` function.



(a) Plotted with `nyqlog_fotf` function.

(b) Plotted with `nyqlog` function.

Figure 5.8: Comparison of Logarithmic-Amplitude polar diagrams.

5.13 Custom AFM Image Generation Function (`print_AFM_image.m`)

In order to create images from the experimental data captured from the AFM during controller testing, a custom function was written in MATLAB. The function takes the captured xyz data as well as the xy scan trajectory data and creates a 2D image where the pixel values represent the z-axis height of the sample.

A usual way to create these images is to walk along a vector of z-axis samples and then assign the values of these samples to the different pixels of the image based on the time stamp and the knowledge of the scan trajectory at the different samples. This is a fairly straightforward and simple scheme and gives good results at low scan frequencies. But at higher scan frequencies the contents of the image may be distorted due to the error between the actual probe position and the reference signal. This error is due to the positioning system having a limited bandwidth.

The custom function created and used in this thesis does the image generation in a slightly different way. Instead of relying on the time vector and the knowledge of the reference, the function uses the actual captured lateral xy-position to assign the z-axis samples to the different pixels of the image. For each set of sampled xyz values a binary search is made, first in the x-direction and then in the y-direction. Based on this the z value can be assigned to the most correct pixel in the image.

A topology image generated by this method can be seen in figure 5.9. It should be noted that the axes in the image represent distance but are labelled with the raw voltage measurements from the sensors of the AFM system. A simple conversion factor for the conversion of voltage to distance could have been implemented but was not prioritized during the work.

This image generation method requires more data from the system than the other method. But by using the measured positions and not the positions of the reference trajectory, the distortions mentioned above is removed. However, this way of generating the image may lead to an image where some of the pixels have not been coloured. This is because there exists a possibility that no z-axis value will fall into the area associated with a specific pixel. However, if one assumes

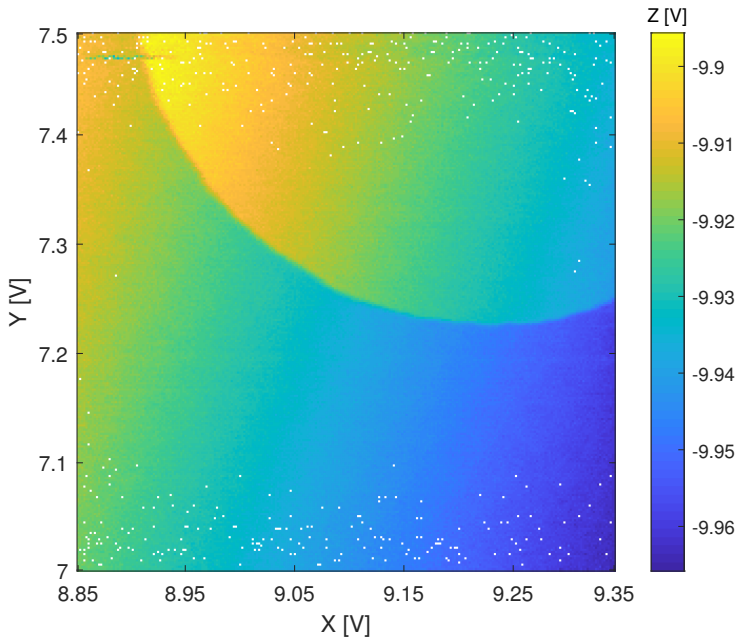


Figure 5.9: An image generated by `print_AFM_image.m`.

that the sample is sufficiently smooth and continuous, the missing data or uncoloured pixels can be fixed by interpolation.

Chapter 6

Simulation Results

The goal of the work presented in this thesis is to assess the usefulness of fractional-order controllers on the lateral axes of an AFM system. The idea is that a fractional-order controller can have better damping properties, allowing for a higher closed-loop bandwidth. In addition, it is interesting to compare the performance of integer-order controllers and fractional-order controllers in order to find out if fractional-order control systems are worth investigating further.

After several attempts at optimizing the IO-PID, IO-PPF, FO-PID and the three FO-PPF controllers with the method explained in chapter 5, the seemingly best controller of each type was selected for simulation testing and experimental testing. This selection was done primarily by looking at the step responses and choosing the ones with lowest settling time and least overshoot.

First we will present the results from the closed-loop simulations with different controllers in section 6.2 of this chapter. Then the experimental testing on the lateral motion stage of the commercial XE-70 AFM system from Park Systems is presented in chapter 7.

6.1 Simulation Setup

The controllers were tested through simulation with MATLAB and Simulink. The `Fractional-order transfer function` block from the `fotf` toolbox [5] was used in order to simulate the fractional-order transfer functions in Simulink. Variable-step solvers were used.

In section 6.2 each controller is presented with a step response plot, a bode diagram and a logarithmic Nyquist diagram. In addition, five evaluation criteria were calculated for each of the controllers in an attempt at making comparison between the controllers easier. The evaluation criteria are explained to some extend below.

Evaluation Criteria

Bandwidth (BW) of the different regulators was calculated at the point where the magnitude of the frequency response dropped below -3 dB for the first time.

Max of Sensitivity Function ($||S(s)||_\infty$) was calculated so as to assess the max gain of external disturbance on the system. The sensitivity function is defined as

$$S(s) = \frac{1}{1 + L(s)}, \tag{6.1}$$

where $L(s)$ is the open-loop transfer function (5.1) presented in section 5.4.

Integral of Squared Error (ISE) of a step response was calculated to check the error between the reference and the response. ISE of a continuous signal is defined as

$$\text{ISE} = \int_0^T e(\tau)^2 \mathrm{d}\tau = \int_0^T \left(r(\tau) - y(\tau)\right)^2 \mathrm{d}\tau, \tag{6.2}$$

while the ISE of a discontinuous signal can be defined in the same manner as

$$\text{ISE} = \Delta t \sum_{k=0}^{N} e[k]^2 = \Delta t \sum_{k=0}^{N} \left(r[k] - y[k]\right)^2, \qquad \Delta t = \frac{T}{N}. \tag{6.3}$$

Here, T is the end time, N is the number of samples and Δt is the step in time between samples. Furthermore, r is the reference signal, y is the response signal and $e = r - y$.

The discontinuous variant, i.e. (6.3), is used for the calculations presented in this chapter because of the discontinuous nature of finite step simulations.

Phase Margin (Ψ) is calculated by the `assess_stability` function, and is the angle between the negative real axis and a line from origin passing through the first intersection where the frequency response of the open-loop system $L(s)$ crosses the 0-dB line when moving in a counter-clockwise direction from the negative real axis. The frequency at this point is usually denoted by ω_c. The phase margin is often calculated according to

$$\Psi = \angle L(j\omega_c) + 180^\circ. \tag{6.4}$$

If the frequency response of $L(s)$ crosses the 0-dB line several times, then it can be argued that there exist several phase margins for the system. In this case the smallest of the phase margins is the correct one.

Gain Margin (Δk) is calculated by the `assess_stability` function in the same manner as the phase margin. The gain margin is the gain that can be added to an already stable system, with smallest absolute value calculated in decibel, that will place the system on the limit of stability. Gain margin is often calculated according to

$$\Delta k = \frac{1}{|L(j\omega_{180})|}, \tag{6.5}$$

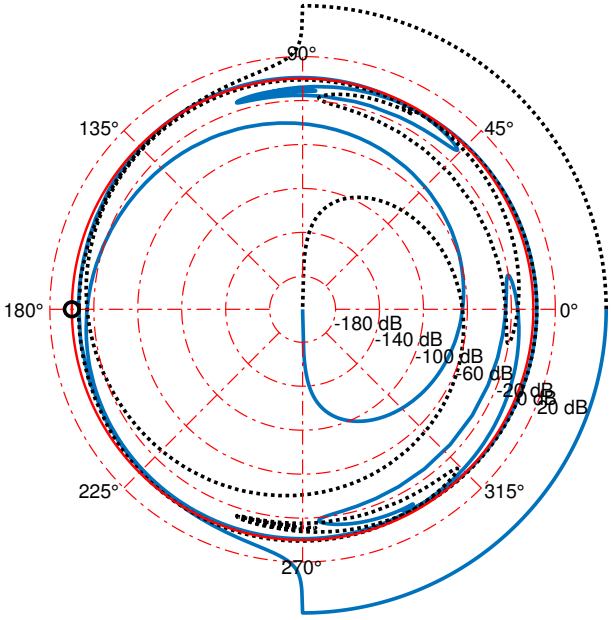
where ω_{180} is the frequency where the phase of the closed-loop system frequency response crosses a multiple of 180°.

6.2 Results

6.2.1 Regular PPF

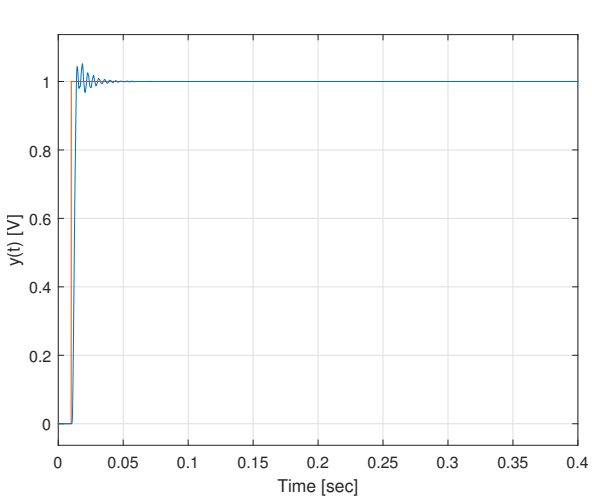
(a) Regulator parameters

k_t	k_d	ζ_d	ω_d
935.8	4.888	0.738	$8.342 \cdot 10^3$

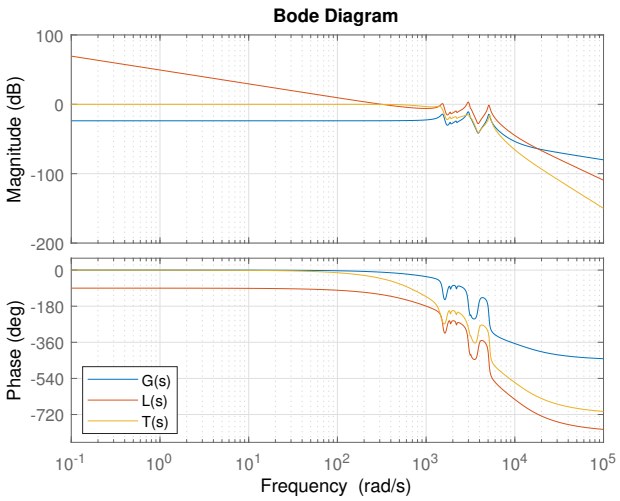


(b) Logarithmic Nyquist diagram

Figure 6.1: Regulator optimization results for regular PPF.



(a) Step responses



(b) Bode plots

Figure 6.2: Regulator optimization results for regular PPF.

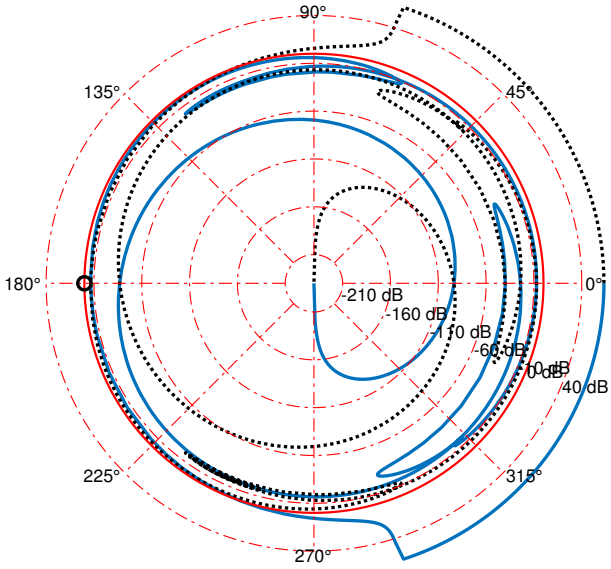
Table 6.1: Evaluation criteria for regular PPF regulator.

BW	$\|\|S(s)\|\|_\infty$	ISE	Ψ	Δk
941.5 rad/s	6.036 dB	$1.898 \cdot 10^{-3}$	54.38°	5.999 dB

6.2.2 FO-PPF version 1

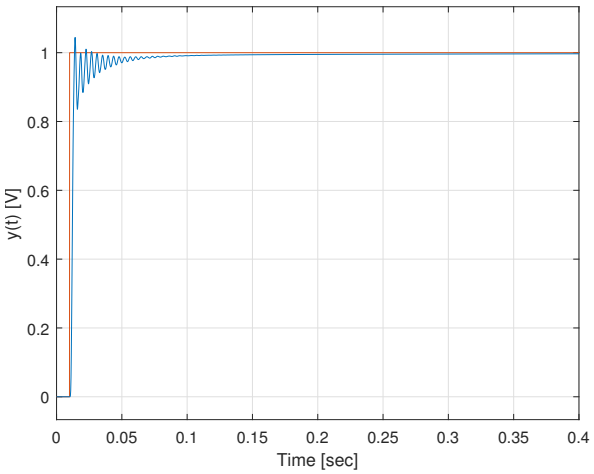
(a) Regulator parameters

k_t	α_t	k_d	ζ_d	ω_d
867.9	0.805	2.3831	3.031	$9.26 \cdot 10^3$

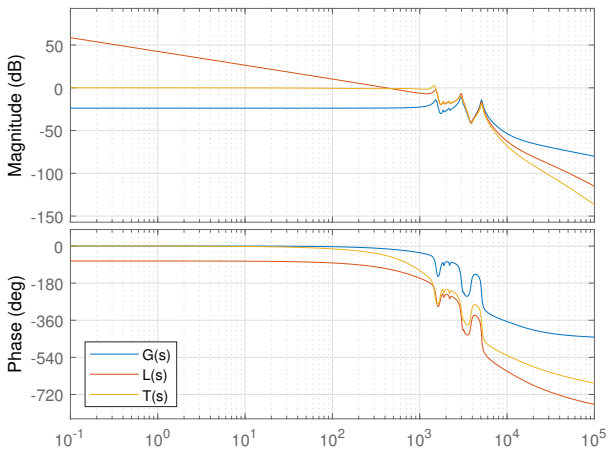


(b) Logarithmic Nyquist diagram

Figure 6.3: FO-PPF_ 1 regulator optimization results



(a) Step responses



(b) Bode plots

Figure 6.4: FO-PPF_ 1 regulator optimization results

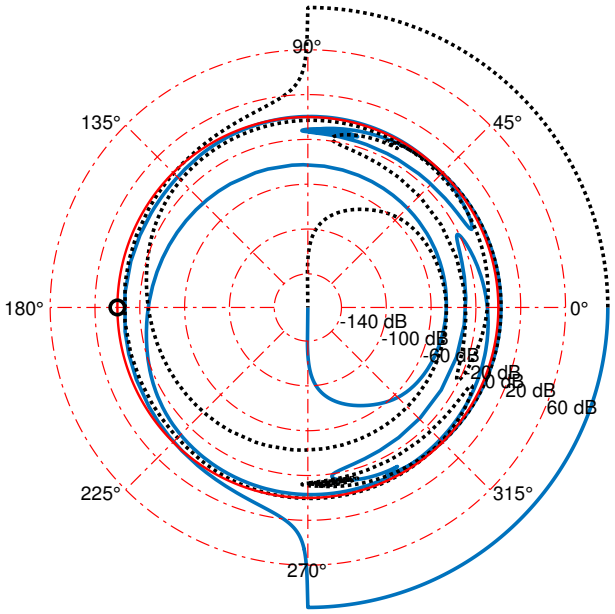
Table 6.2: Evaluation criteria for `FO-PPF_1` regulator

BW	$\|\|S(s)\|\|_\infty$	ISE	Ψ	Δk
1.541×10^3 rad/s	6.506 dB	$1.882 \cdot 10^{-3}$	68.61°	6.578 dB

6.2.3 FO-PPF version 2

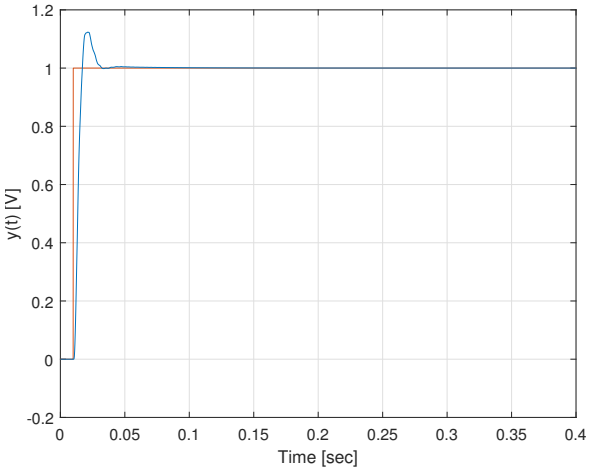
(a) Regulator parameters

k_t	k_d	ζ_d	ω_d	β_d
353.6	8.492	4.308	501.3	0.509

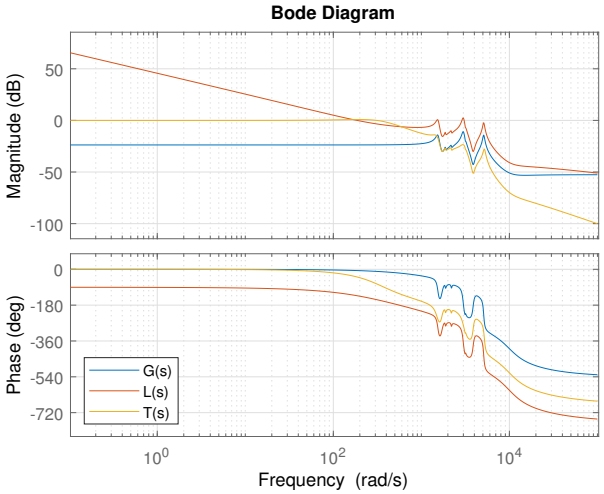


(b) Logarithmic Nyquist diagram

Figure 6.5: FO-PPF_ 2 regulator optimization results



(a) Step responses



(b) Bode plots

Figure 6.6: FO-PPF_ 2 regulator optimization results

Table 6.3: Evaluation criteria for FO-PPF_ 2 regulator

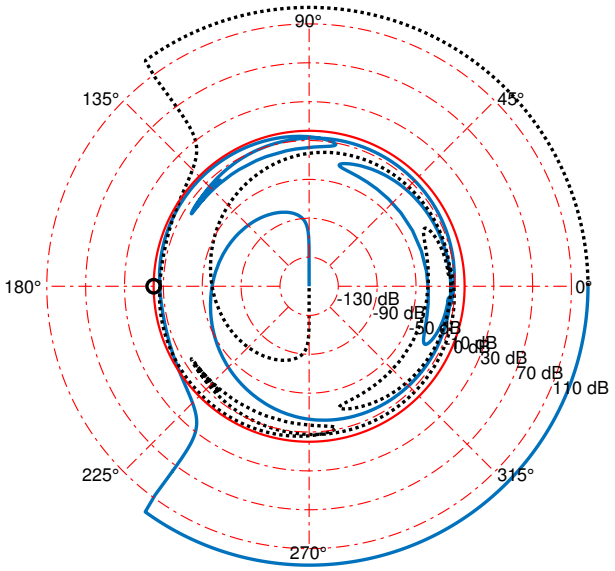
BW	$\|\|S(s)\|\|_\infty$	ISE	Ψ	Δk
455.8 rad/s	6.576 dB	$2.803 \cdot 10^{-3}$	45.98°	6.005 dB

6.2.4 FO-PPF version 3

(a) Regulator parameters

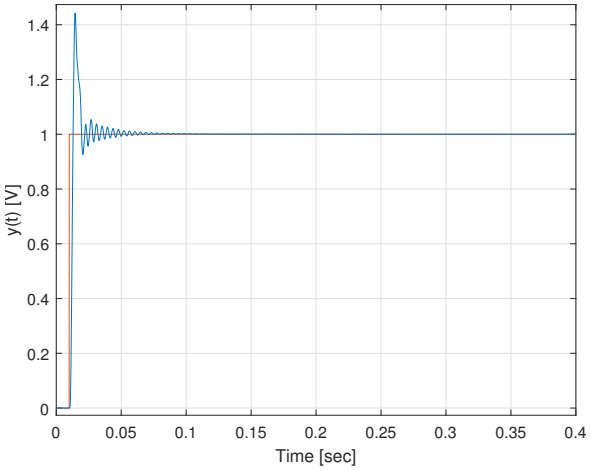
k_t	α_t	k_d
$2.01 \cdot 10^5$	1.405	0.5162

ζ_d	ω_d	β_d
1.047	$8.68 \cdot 10^5$	0.010

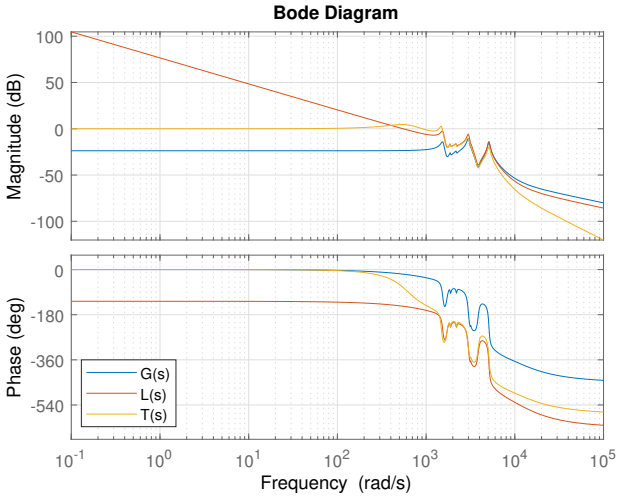


(b) Logarithmic Nyquist diagram

Figure 6.7: FO-PPF_ 3 regulator optimization results



(a) Step responses



(b) Bode plots

Figure 6.8: FO-PPF_ 3 regulator optimization results

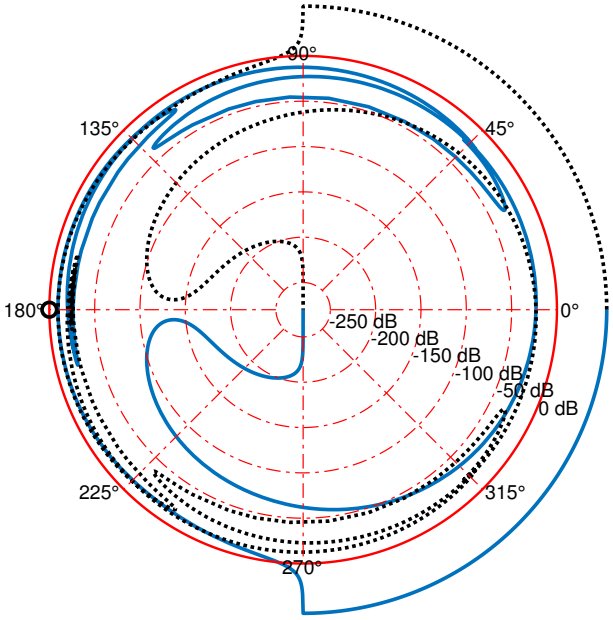
Table 6.4: Evaluation criteria for FO-PPF_ 3 regulator

BW	$\|\|S(s)\|\|_\infty$	ISE	Ψ	Δk
1.54×10^3 rad/s	7.334 dB	$2.130 \cdot 10^{-3}$	33.11°	6.173 dB

6.2.5 Regular PID

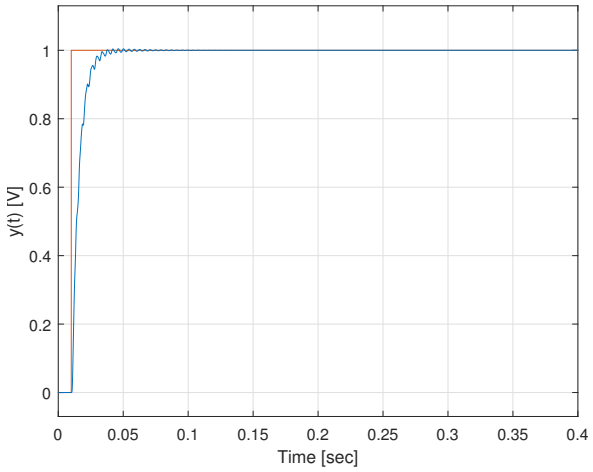
(a) Regulator parameters

k_p	k_i	k_d	τ_f
$1.00 \cdot 10^{-4}$	$2.59 \cdot 10^{3}$	1.553	$9.50 \cdot 10^{3}$

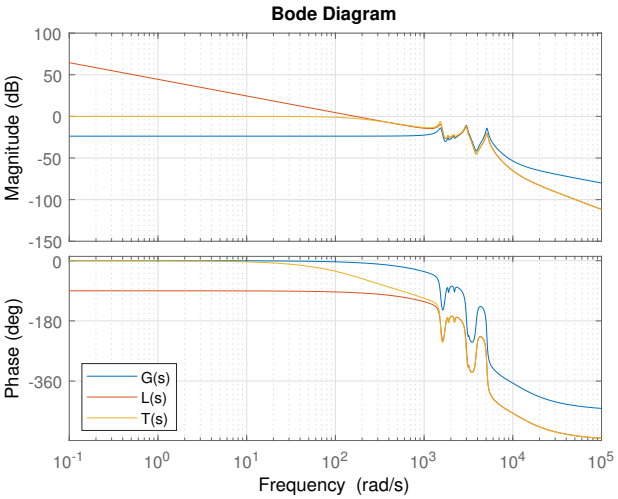


(b) Logarithmic Nyquist diagram

Figure 6.9: Regulator optimization results for regular PID.



(a) Step responses



(b) Bode plots

Figure 6.10: Regulator optimization results for regular PID.

Table 6.5: Evaluation criteria for regular PID regulator.

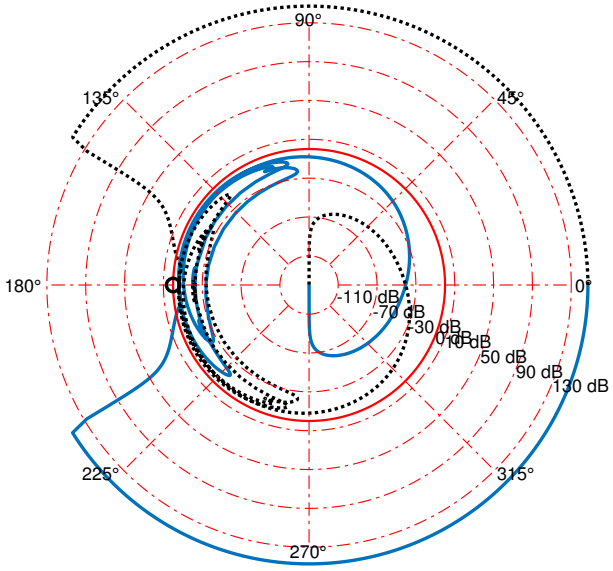
BW	$\|\|S(s)\|\|_\infty$	ISE	Ψ	Δk
187.4 rad/s	3.532 dB	$3.300 \cdot 10^{-3}$	83.90°	9.547 dB

6.2.6 FO-PID

(a) Regulator parameters

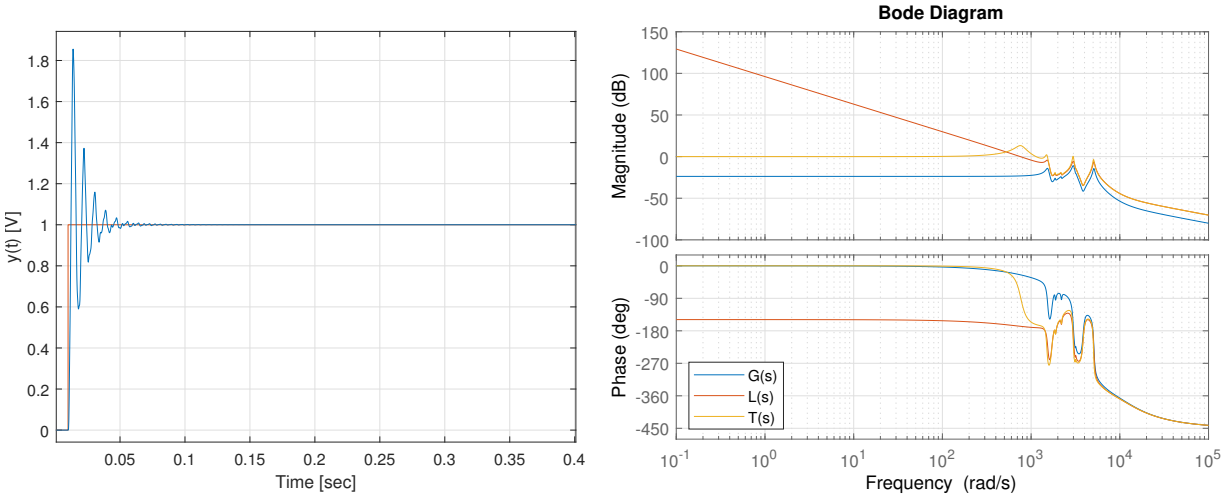
k_p	k_i	k_d
2.071	$9.851 \cdot 10^5$	$5.512 \cdot 10^3$

μ_i	μ_d	τ_f
1.655	1.951	$5.451 \cdot 10^3$



(b) Logarithmic Nyquist diagram

Figure 6.11: Regulator optimization results for FO-PID.



(a) Step responses

(b) Bode plots

Figure 6.12: Regulator optimization results for FO-PID.

Table 6.6: Evaluation criteria for FO-PID regulator.

BW	$\|\|S(s)\|\|_\infty$	ISE	Ψ	Δk
1.545×10^3 rad/s	13.42 dB	$3.247 \cdot 10^{-3}$	12.61°	5.999 dB

Chapter 7

Experimental Results

The experimental testing of the IO-PID, IO-PPF, FO-PID and the three FO-PPF controllers on the lateral motion stage of the commercial XE-70 AFM system from Park Systems is presented in in section 7.2 of this chapter. But, first some information on the experimental setup is presented in section 7.1.

7.1 Experimental Setup

Figure 7.1 shows an overview of the most important system parts and connections in the experimental setup. MATLAB and Simulink were used for design and implementation of the XY-controllers, while ControlDesk was used for monitoring of the signals and regulators. The Simulink models were compiled down to code and loaded onto a dSpace real-time controller system on an external computer. The dSpace controller system was in turn connected to the XE-70 AFM through a set of control boards with coaxial cable interfaces.

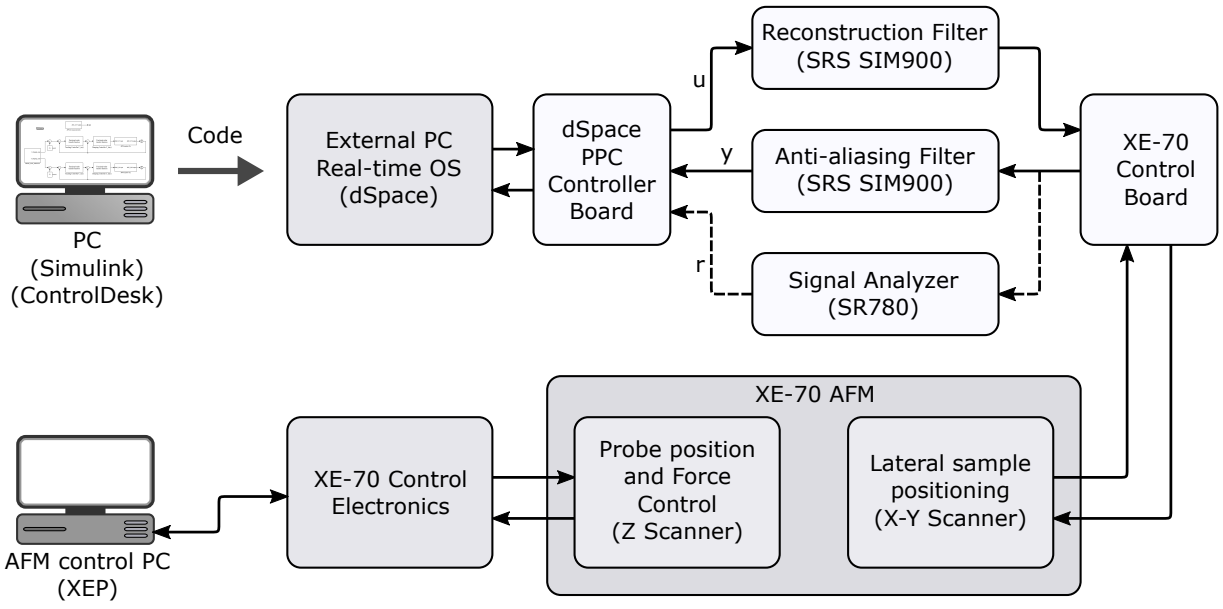


Figure 7.1: Experimental setup.

The standard XE-70 control system for Z-axis force and position control of the probe head, was used in parallel with the experimental lateral control systems under testing, to enable surface scanning. This was possible because of the inherent axis decoupling property of the XE-70 system, which is explained in section 2.2.1.

A set of Butterworth low-pass filters with a cut-off frequency at 10 kHz were connected between the dSpace controller board and the AFM controller board to serve as so-called anti-aliasing and reconstruction filters. A SR780 signal analyzer was also connected, as shown in figure 7.1, during closed loop frequency response analysis. A top level view of the Simulink model used for PPF and FO-PPF control is shown in figure 7.2. A similar model was used for PID and FO-PID control.

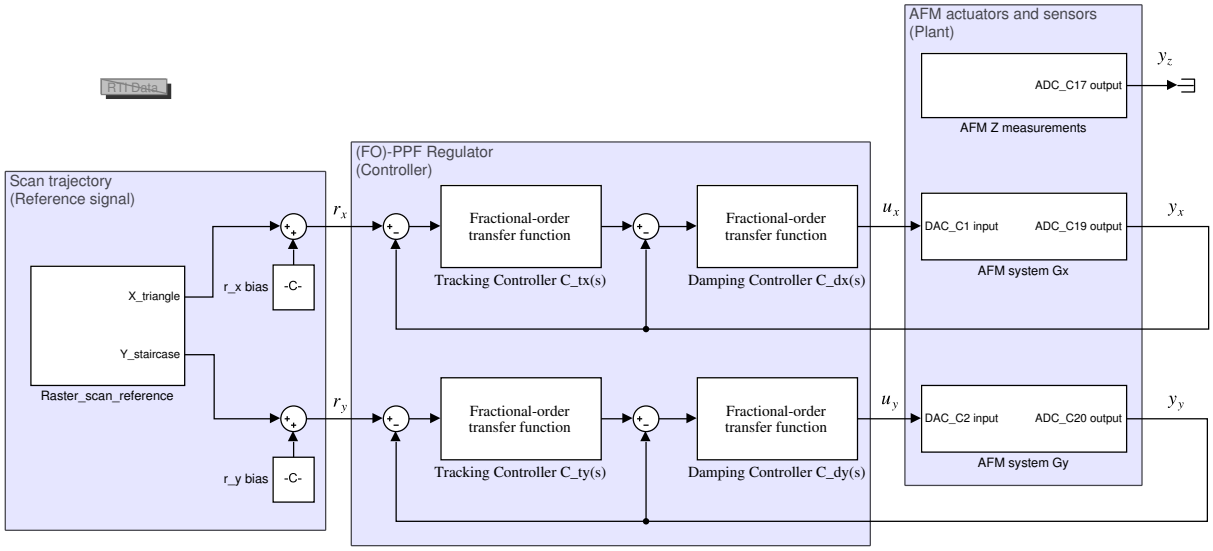


Figure 7.2: Simulink diagram of PPF controller model used for AFM image scanning.

The fixed-step size solver ode1 (Euler) with a step size of around 6.0 µs was found to work for the calculation of the controllers. Moreover, an Oustaloup filter order of 5 was used in the approximation of terms like s^α. As for Oustaloup filter frequency intervals, an interval of $[\omega_b, \omega_h] = [10^{-2}, 10^4]$ was found to work well for the FO-PPF tracking controller and an interval of $[\omega_b, \omega_h] = [10^{-2}, 10^2]$ rad/s was found to work for the FO-PPF damping controller. For the fractional-order PID controller an interval $[\omega_b, \omega_h] = [10^{-2}, 10^4]$ rad/s was found to give stable control.

It was decided to have a resolution of 256x256 pixels on the captured images. This led to 256 *back and forth* scan lines for the AFM. In addition, two different scanning frequencies were used to make comparisons between the controllers easier. The frequencies used were 5 Hz and 30 Hz. To reduce the strain on the xPC that did both control the lateral axes and capture data, data decimation was employed. With a scanning frequency of 5 Hz a data decimation of 1:40 was used, while a data decimation of 1:6.6 was used with the scanning frequency of 30 Hz. The reason for using these specific decimation ratios was to capture enough data per line of pixels and to capture approximately the same amount of data for the two different scanning speeds, making comparisons of the images and controller performances easier.

Number of samples per scan line can be calculated from

$$\text{samples/scanLine} = \frac{1}{\text{scanFrequency} \cdot \text{sampleTime} \cdot \text{decimation}}. \tag{7.1}$$

This leads to respectively 833.3 and 841.8 captured data point per scan line, or row of pixels, for a scanning frequency of 5 Hz and 30 Hz. Which is over three times the number of pixels on a row.

For the test images a sample surface of silicon is used. The sample has several circular indentations with a height of 20 nm and a radius of 3.5 µm. It is the same sample surface used in section 2.5.

A fairly low actuator driving voltage (−10 V to 0 V) was used in the testing of the lateral motion controllers. This was done to keep inside of the recommended voltage levels of the XE-70 control board. Since the piezoelectric actuator extends and contracts as a function of the applied voltage, the use of low voltage leads to a reduced scan area. For the controller tests conducted in this thesis, this was found acceptable.

In addition to the image scans, a frequency response analysis of the closed loop systems was conducted. The analysis was done in similar fashion as the one conducted for system identification purposes in section 5.3.

Notes on realization of Fractional-order models

To realize the fractional-order controllers, a `fractional-order transfer function`block from the `fotf` toolbox [5] created by Xue is used. This transfer function block approximates the fractional-order integrators and derivatives with an Oustaloup filter. The filter order and the frequency interval where the approximation is valid can be freely set. The Oustaloup filter approximation method is explained closer in section 3.7.2.

It should be emphasized that the *fractional-order transfer function block* is compiled down to a regular linear time-invariant state space model before the controller is run on the external computer. Terms in the transfer function on the form s^α is substituted by Oustaloup filter, which are basic LTI transfer functions. Leading to a total controller transfer function that is also just an LTI transfer function.

It should also be mentioned that substitution of integer-order transfer functions for s^α can make the approximated fractional-order controller quite big in terms of order. The corresponding state space model will then become quite large, leading to a computational heavy regulator. Besides, increasing the Oustaloup filter order with the goal of increasing the approximation accuracy, the model complexity will increase as well. As the dimension of the state space model increases, the computational power that is needed will also increase. Oustaloup based, high accuracy fractional-order controllers are therefore hard to realize on *standard* real-time computers.

7.2 Results

7.2.1 Regular PPF

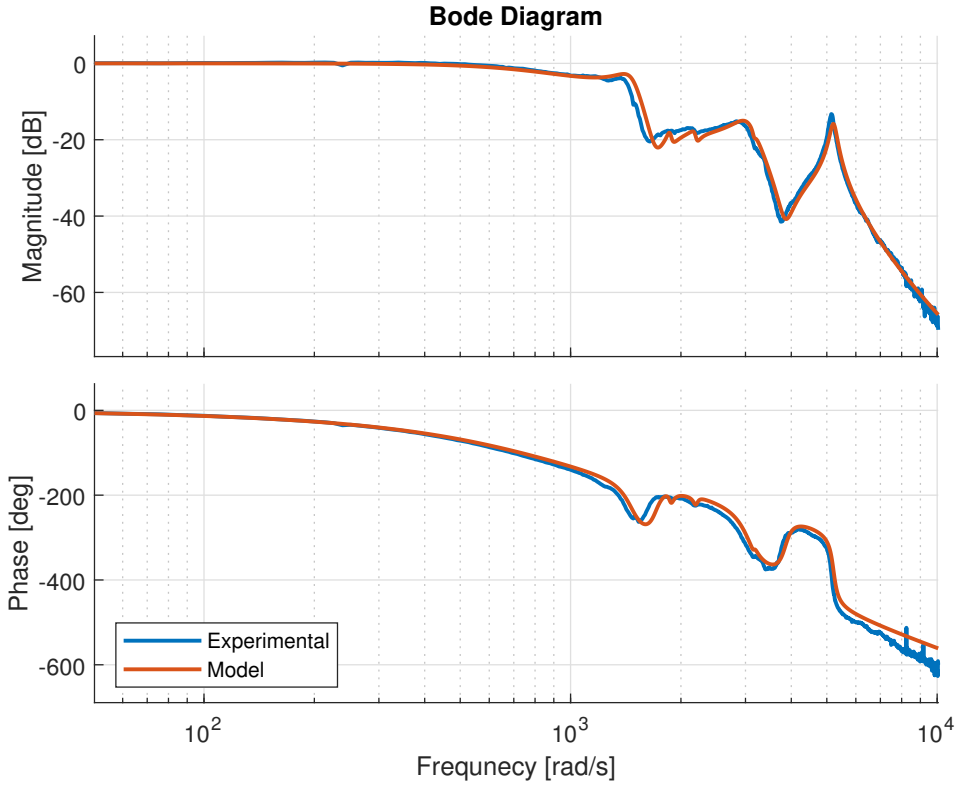


Figure 7.3: Closed-loop frequency response of a PPF controlled AFM lateral positioning system. Experimental frequency response obtained with SR780 plotted against the frequency response of the simulation model.

Table 7.1: Regular PPF controller parameters.

k_t	k_d	ζ_d	ω_d
935.8	4.888	0.738	$8.342 \cdot 10^3$

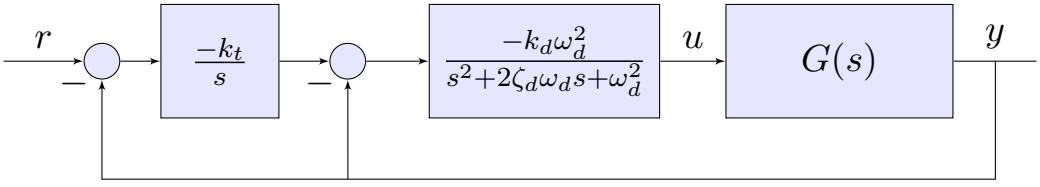
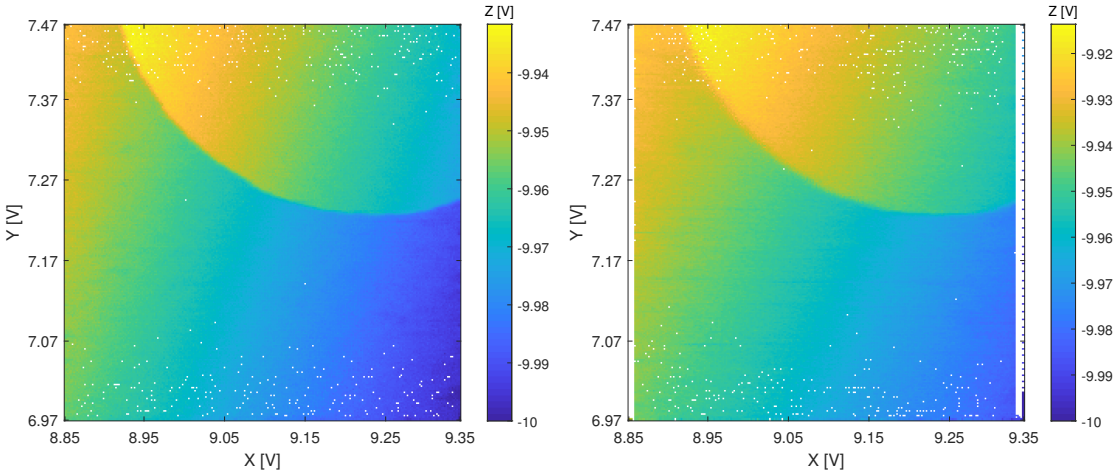
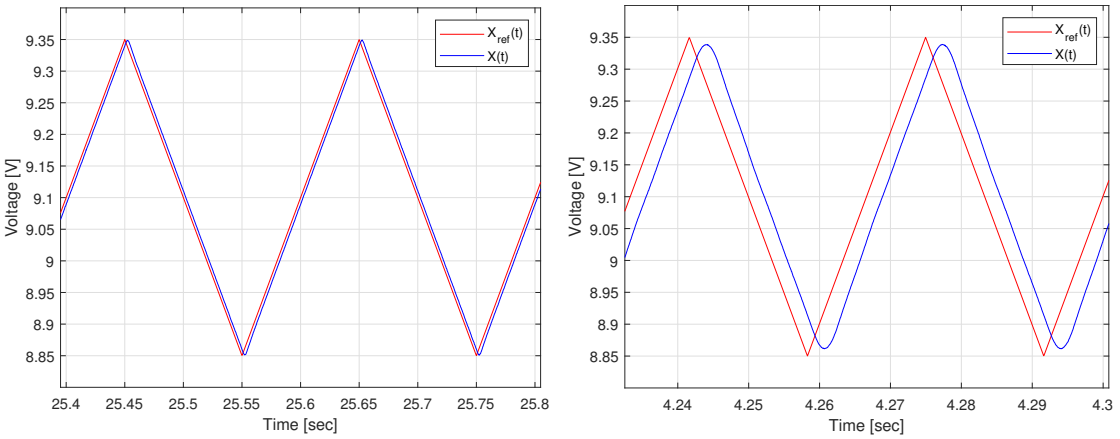


Figure 7.4: Regular PPF damping and tracking control block diagram.



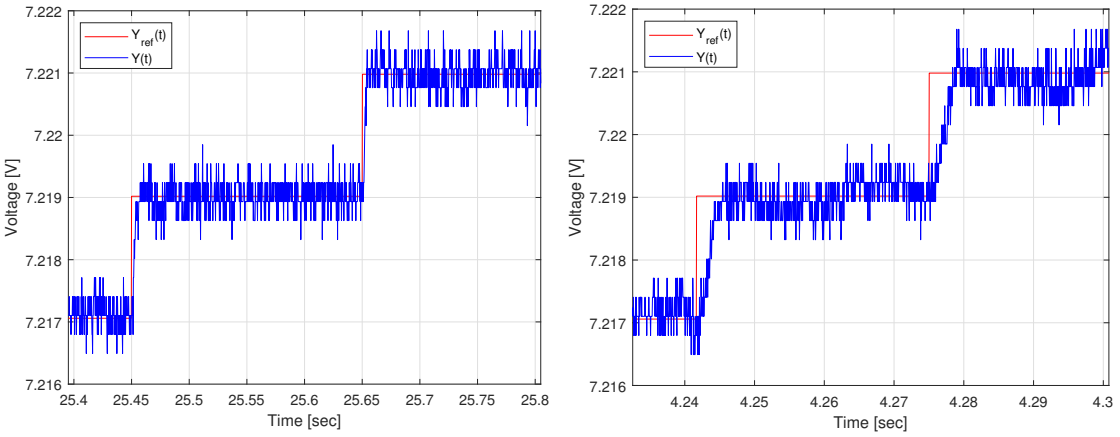
(a) 5 Hz (b) 30 Hz

Figure 7.5: AFM images with PPF controlled lateral position. Size: 256x256.



(a) 5 Hz (b) 30 Hz

Figure 7.6: PPF controlled X-axis tracking.



(a) 5 Hz (b) 30 Hz

Figure 7.7: PPF controlled Y-axis tracking.

7.2.2 FO-PPF version 1

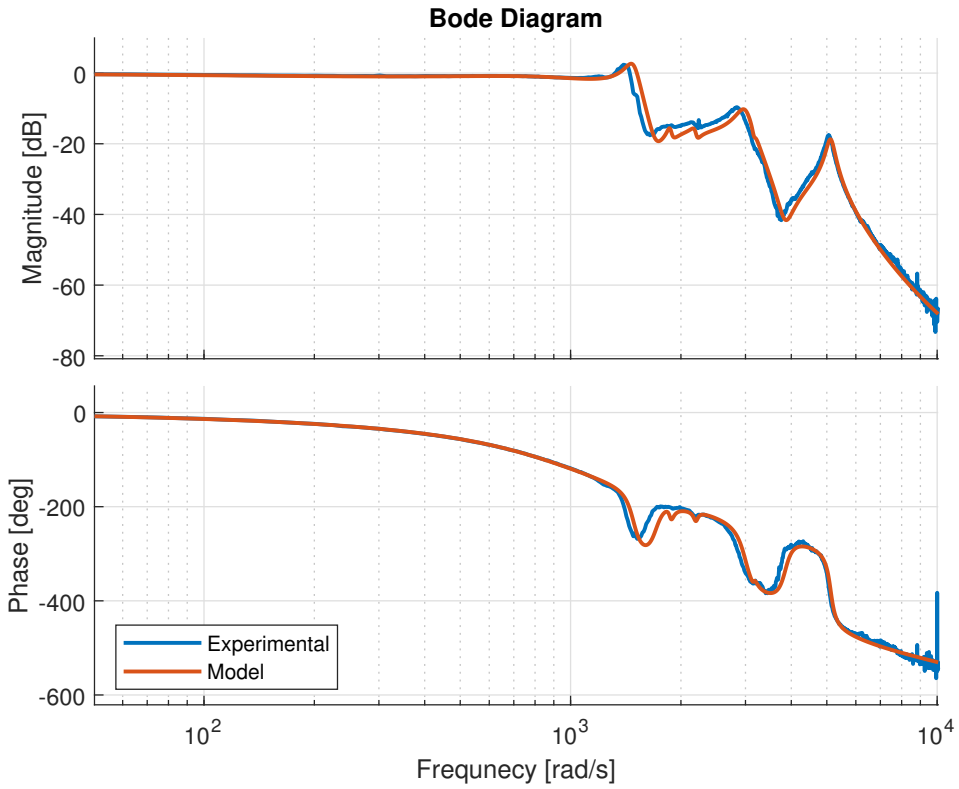


Figure 7.8: Closed-loop frequency response of a FO-PPF version 1 controlled AFM lateral positioning system. Experimental frequency response obtained with SR780 plotted against the frequency response of the simulation model.

Table 7.2: FO-PPF version 1 controller parameters.

k_t	α_t	k_d	ζ_d	ω_d
867.9	0.805	2.3831	3.031	$9.26 \cdot 10^3$

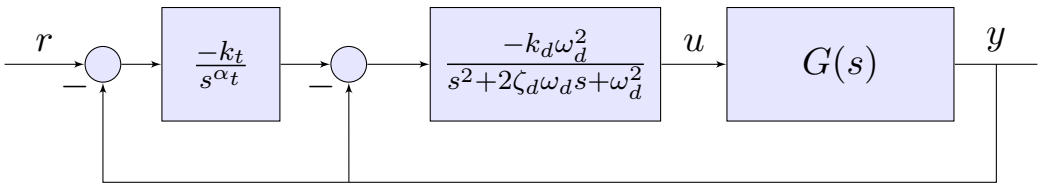
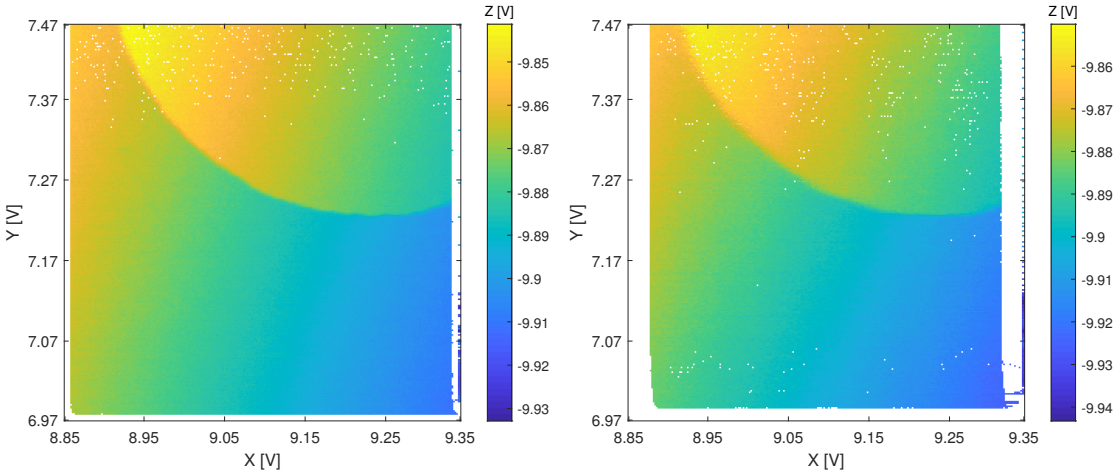
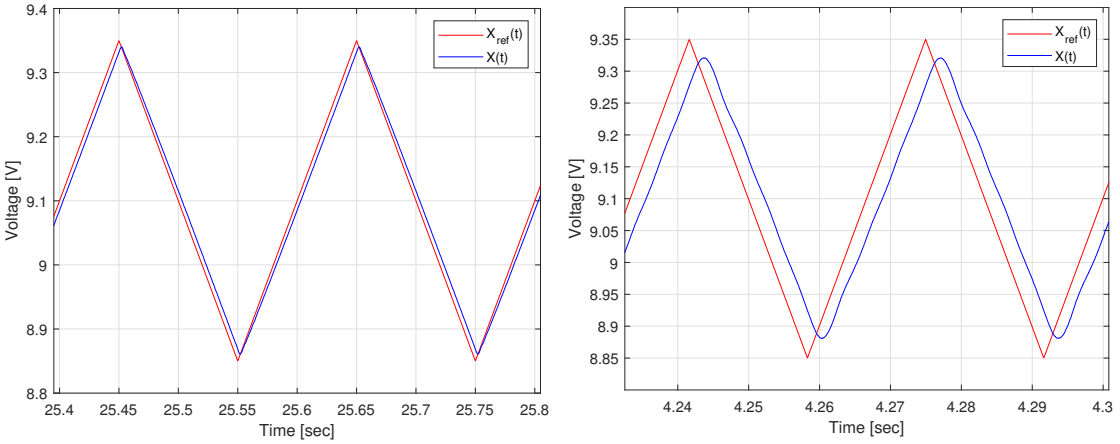


Figure 7.9: FO-PPF version 1 damping and tracking control block diagram.



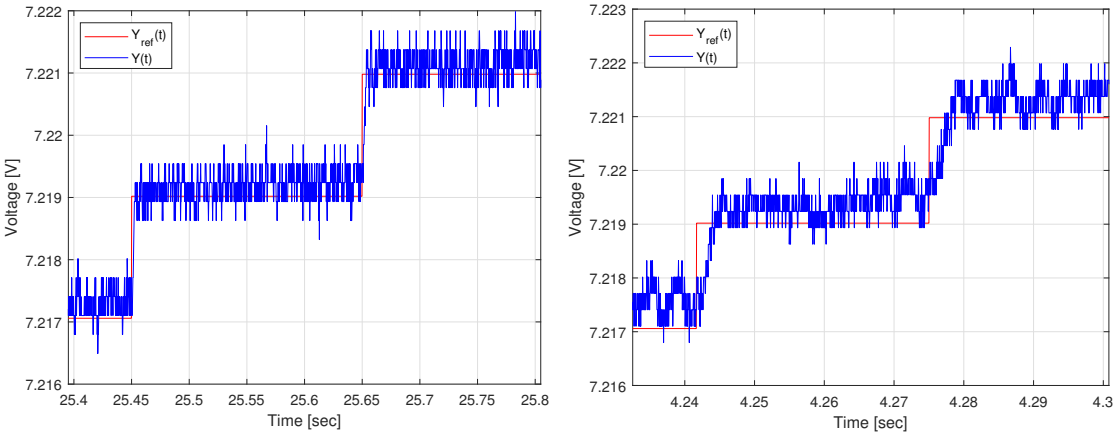
(a) 5 Hz (b) 30 Hz

Figure 7.10: AFM images with FO-PPF version 1 controlled lateral position. Size: 256x256.



(a) 5 Hz (b) 30 Hz

Figure 7.11: FO-PPF version 1 controlled X-axis tracking.



(a) 5 Hz (b) 30 Hz

Figure 7.12: FO-PPF version 1 controlled Y-axis tracking.

7.2.3 FO-PPF version 2

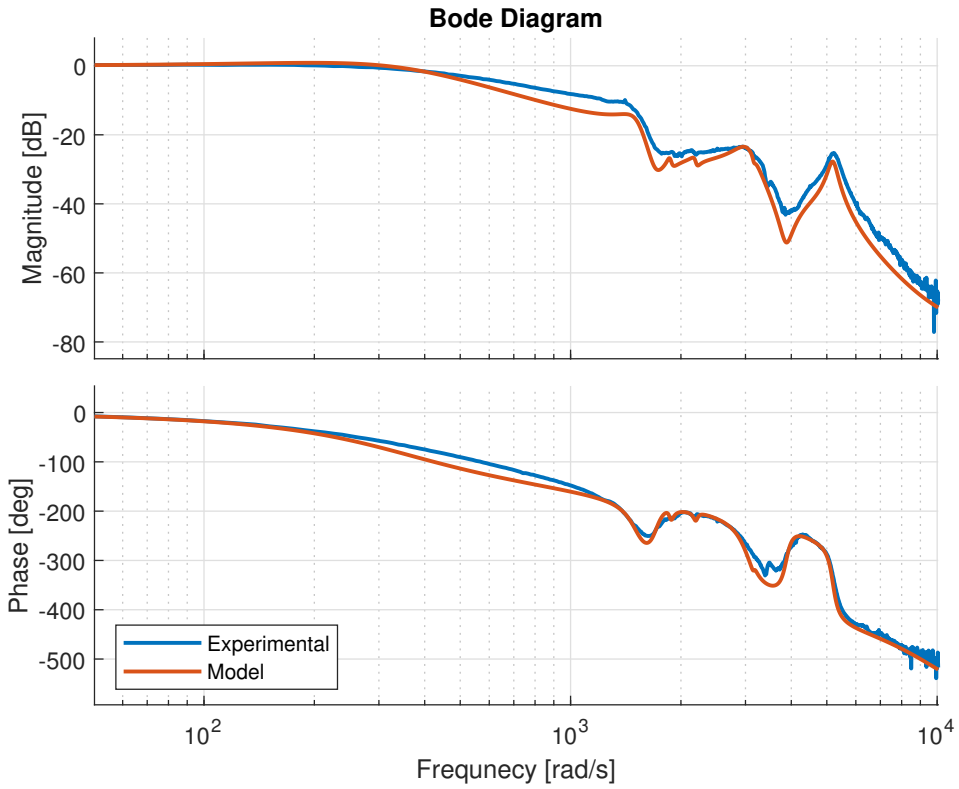


Figure 7.13: Closed-loop frequency response of a FO-PPF version 2 controlled AFM lateral positioning system. Experimental frequency response obtained with SR780 plotted against the frequency response of the simulation model.

Table 7.3: FO-PPF version 2 controller parameters.

k_t	k_d	ζ_d	ω_d	β_d
353.6	8.492	4.308	501.3	0.509

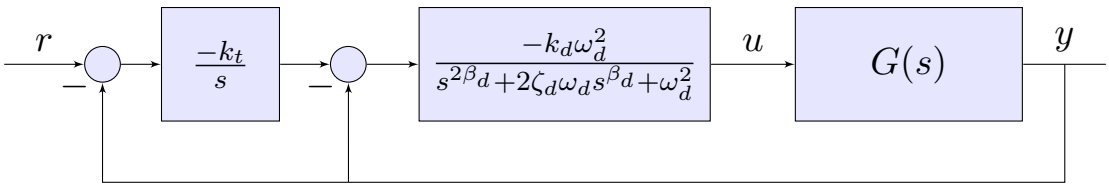
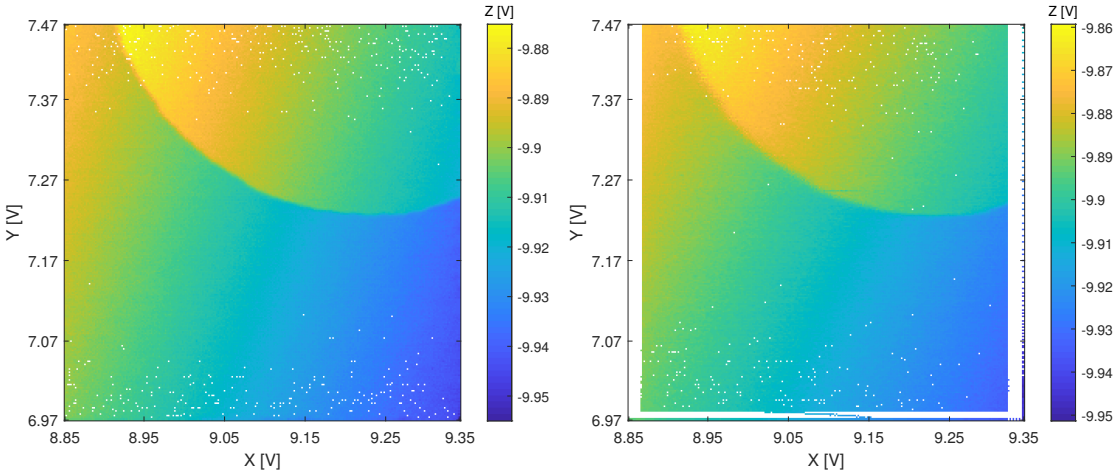
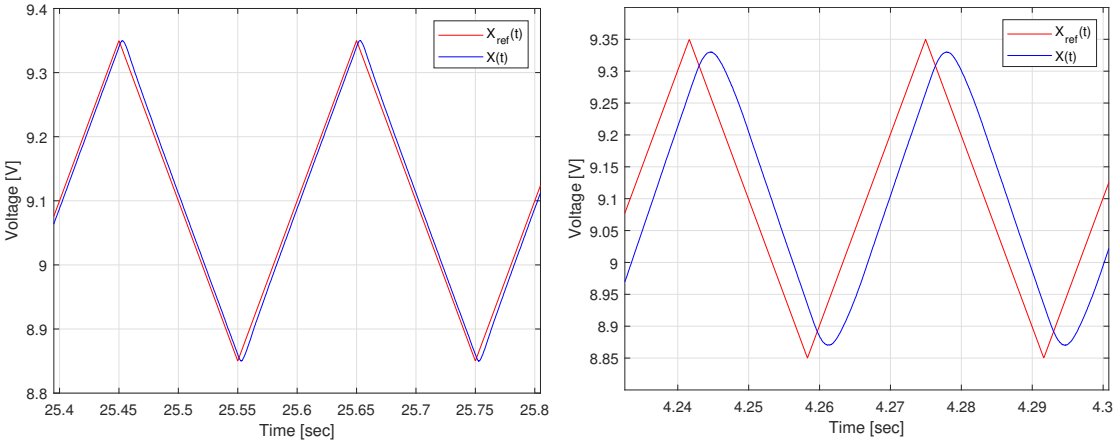


Figure 7.14: FO-PPF version 2 damping and tracking control block diagram.



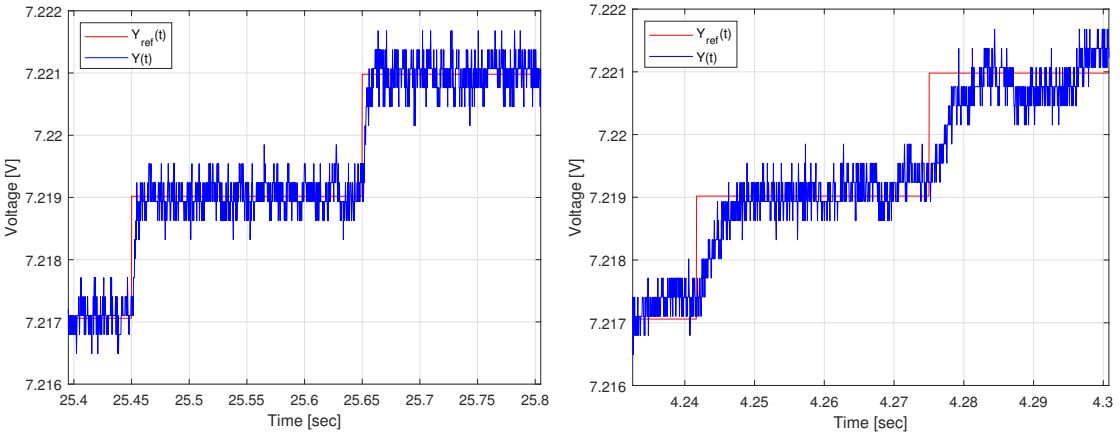
(a) 5 Hz (b) 30 Hz

Figure 7.15: AFM images with FO-PPF version 2 controlled lateral position. Size: 256x256.



(a) 5 Hz (b) 30 Hz

Figure 7.16: FO-PPF version 2 controlled X-axis tracking.



(a) 5 Hz (b) 30 Hz

Figure 7.17: FO-PPF version 2 controlled Y-axis tracking.

7.2.4 FO-PPF version 3

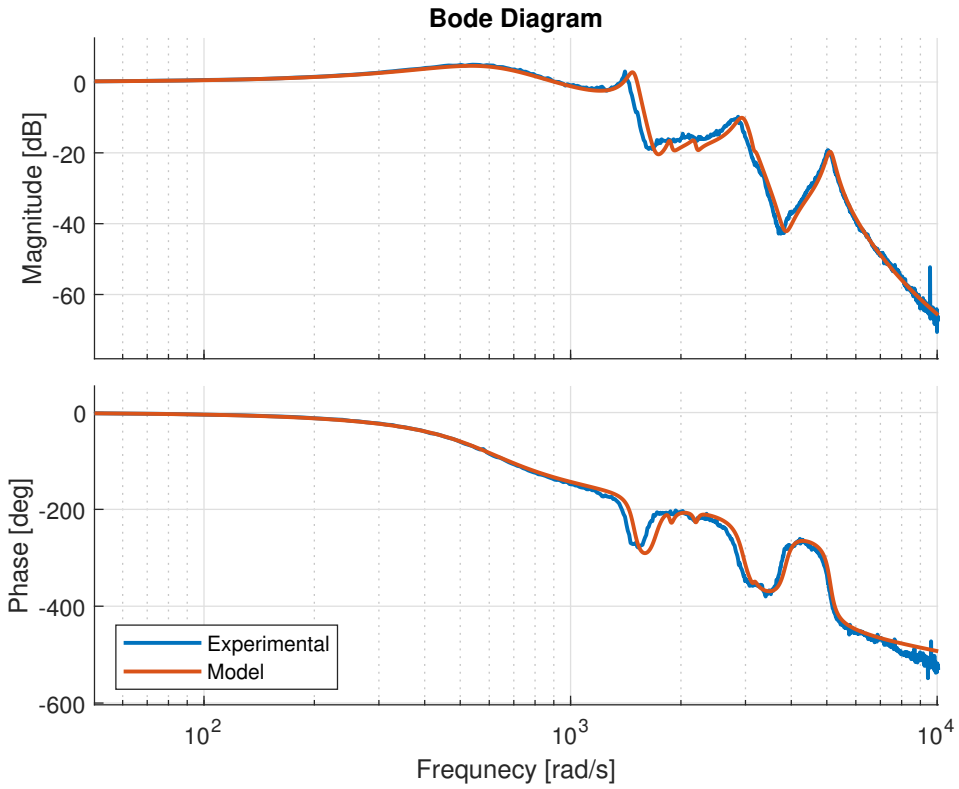


Figure 7.18: Closed-loop frequency response of a FO-PPF version 3 controlled AFM lateral positioning system. Experimental frequency response obtained with SR780 plotted against the frequency response of the simulation model.

Table 7.4: FO-PPF version 3 controller parameters.

k_t	α_t	k_d	ζ_d	ω_d	β_d
$2.01 \cdot 10^5$	1.405	0.5162	1.047	$8.68 \cdot 10^5$	0.010

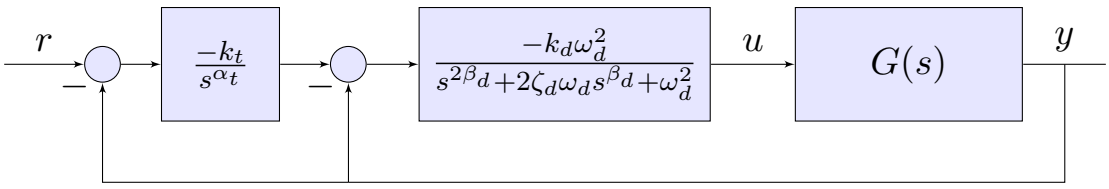
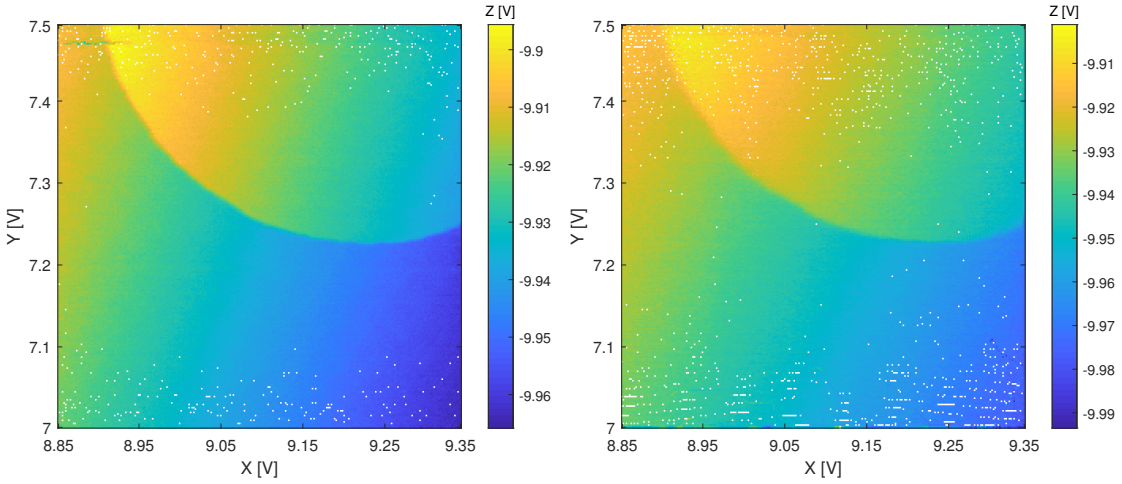
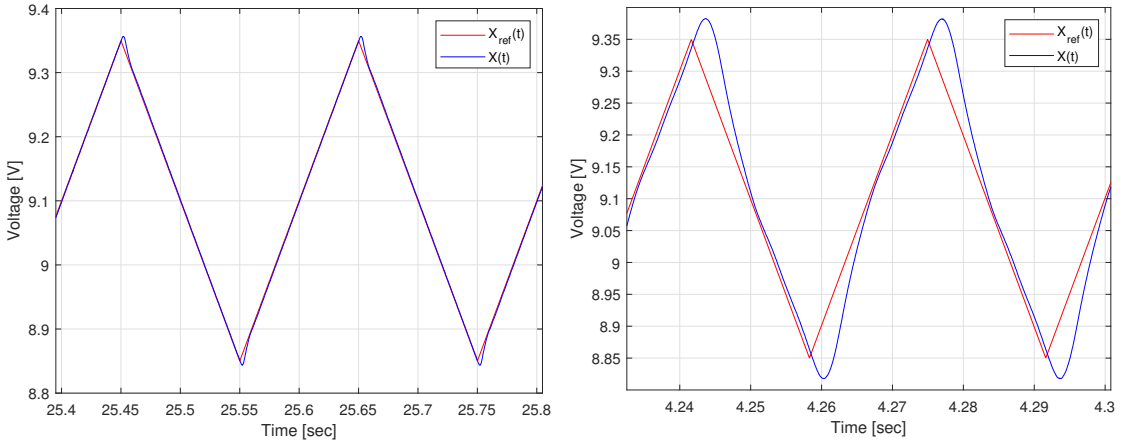


Figure 7.19: FO-PPF version 3 damping and tracking control block diagram.



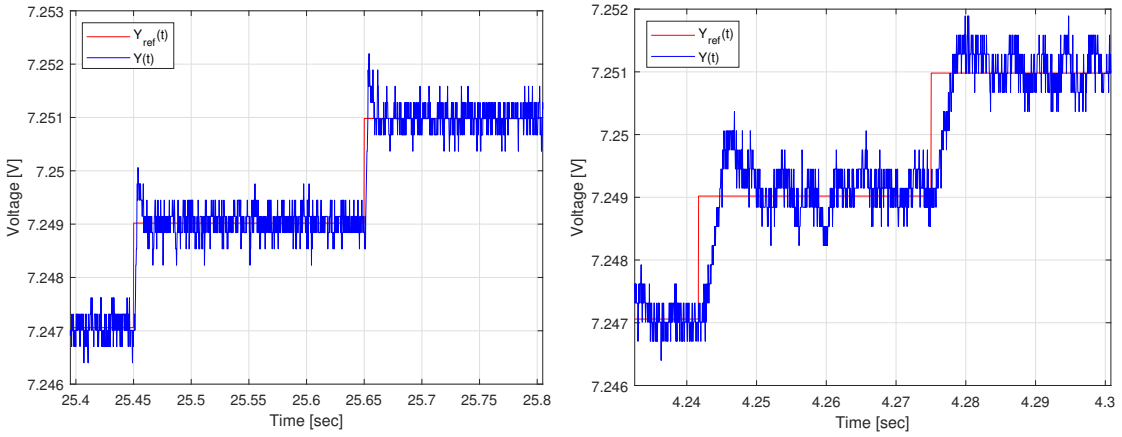
(a) 5 Hz (b) 30 Hz

Figure 7.20: AFM images with FO-PPF version 3 controlled lateral position. Size: 256x256.



(a) 5 Hz (b) 30 Hz

Figure 7.21: FO-PPF version 3 controlled X-axis tracking.



(a) 5 Hz (b) 30 Hz

Figure 7.22: FO-PPF version 3 controlled Y-axis tracking.

7.2.5 Regular PID

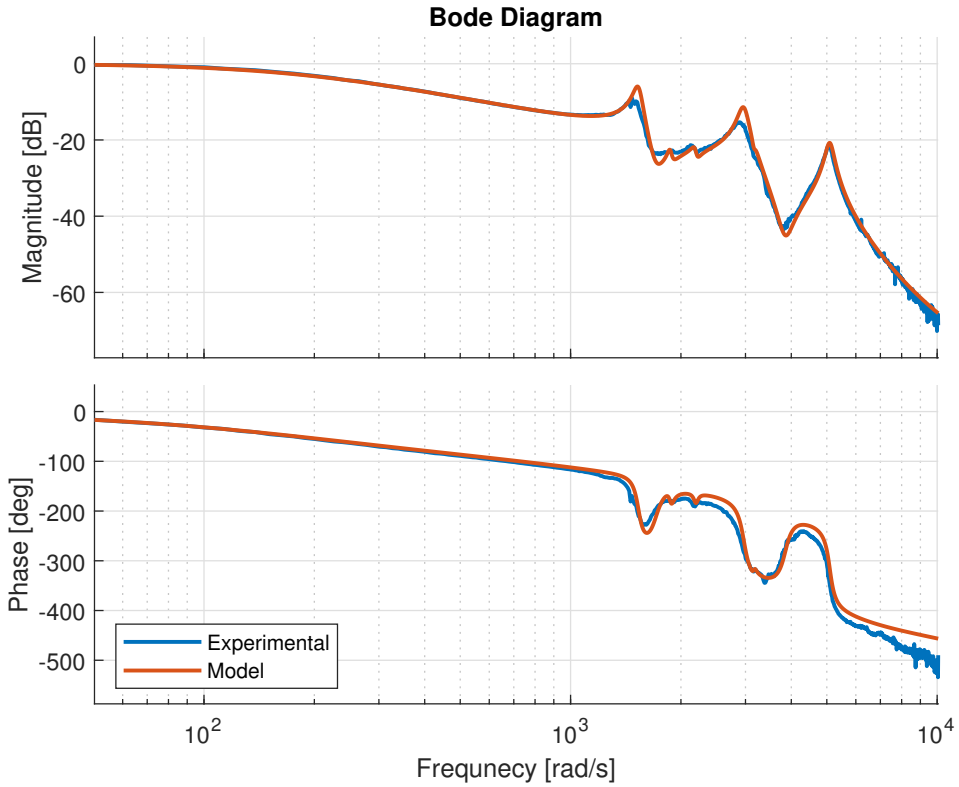


Figure 7.23: Closed-loop frequency response of a PID controlled AFM lateral positioning system. Experimental frequency response obtained with SR780 plotted against the frequency response of the simulation model.

Table 7.5: Regular PID controller parameters.

k_p	k_i	k_d	τ_f
$1.00 \cdot 10^{-4}$	$2.59 \cdot 10^3$	1.553	$9.50 \cdot 10^3$

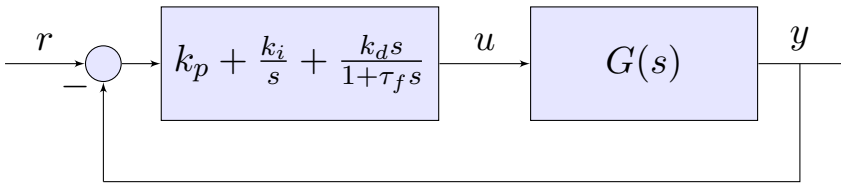
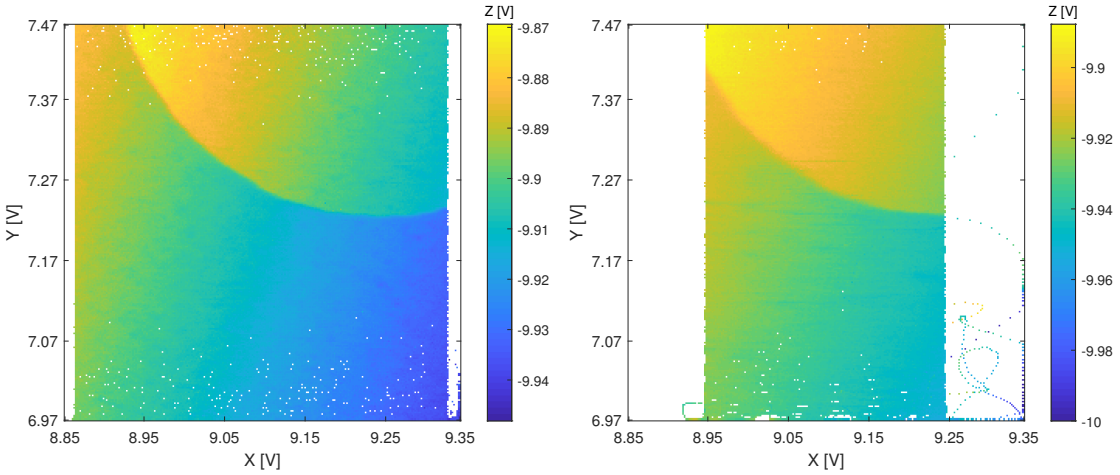
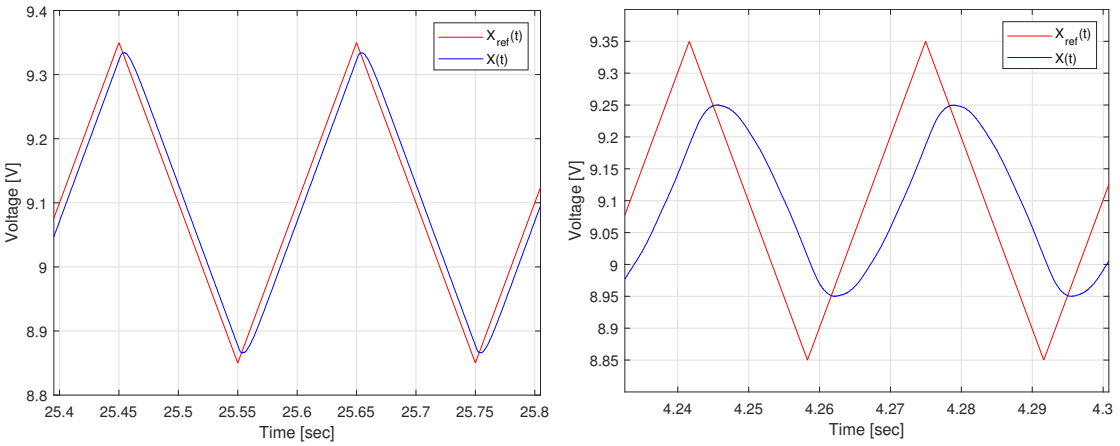


Figure 7.24: Regular PID control block diagram.



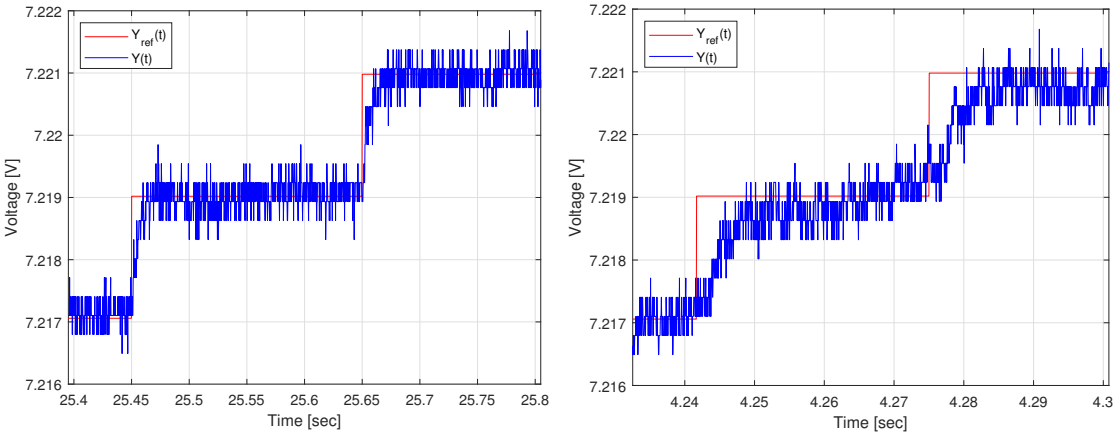
(a) 5 Hz (b) 30 Hz

Figure 7.25: AFM images with PID controlled lateral position. Size: 256x256.



(a) 5 Hz (b) 30 Hz

Figure 7.26: PID version 3 controlled X-axis tracking.



(a) 5 Hz (b) 30 Hz

Figure 7.27: PID controlled Y-axis tracking.

7.2.6 FO-PID

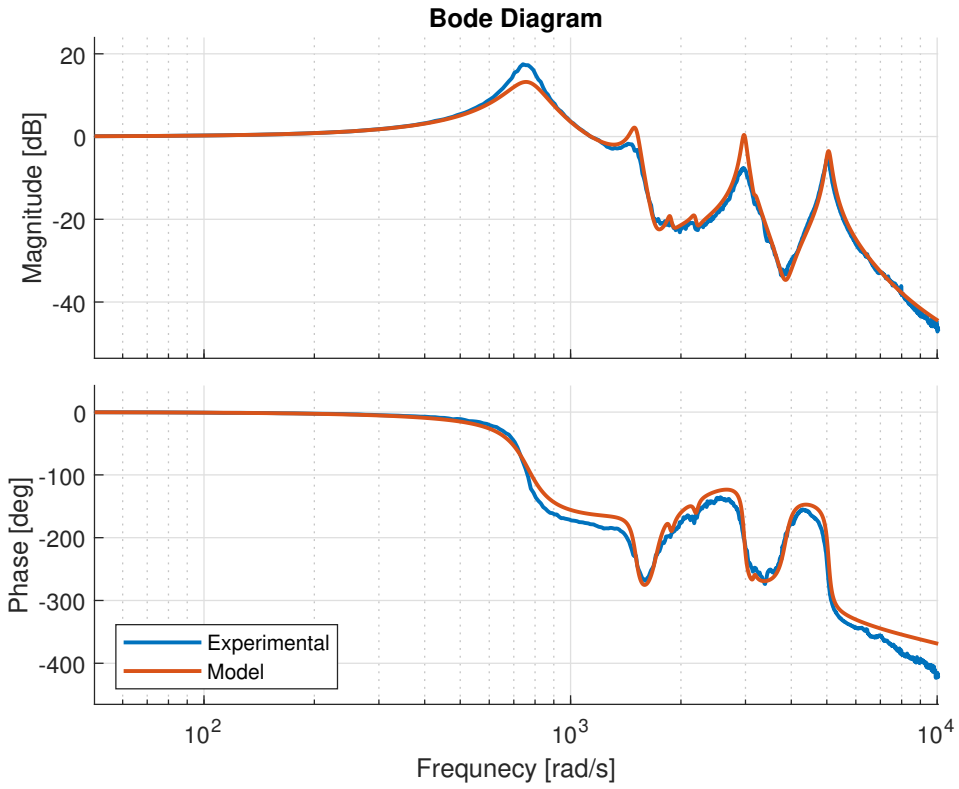


Figure 7.28: Closed-loop frequency response of a FO-PID controlled AFM lateral positioning system. Experimental frequency response obtained with SR780 plotted against the frequency response of the simulation model.

Table 7.6: FO-PID controller parameters.

k_p	k_i	k_d	μ_i	μ_d	τ_f
2.071	$9.851 \cdot 10^5$	$5.512 \cdot 10^3$	1.655	1.951	$5.451 \cdot 10^3$

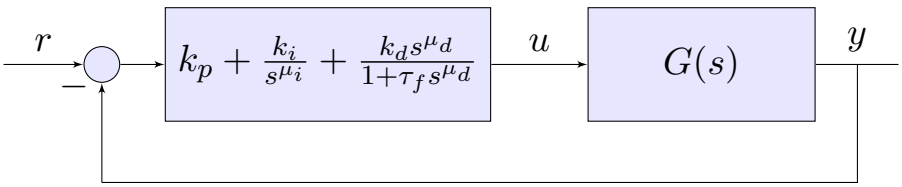


Figure 7.29: FO-PID control block diagram.

Unfortunately, two FO-PID controllers, one for X axis control and one for Y axis control, were too demanding for the real-time computer at the lab, and no combination of controller sample rate and fractional-order approximation configurations resulting in a stable system were found.

7.3 Validity of Gain Margins

A check of the validity of the gain margins of the different closed-loop systems was done in order to check if the implemented tuning method and stability analysis were correct. This check was done by increasing the controller gain little by little towards the gain margin to see if the system started to show signs of continuous oscillations. This check was first done in simulations with MATLAB and then conducted on the real AFM system.

In the end, the results were quite conclusive. All the closed-loop systems started to show continuous oscillations when a gain equal to Δk plus/minus 5% was applied. This happened in both simulation and testing on the real AFM system. This shows that the gain margin calculated by the `assess_stability` function is most likely correct.

7.4 Study of Required Sampling Frequency for Stable Control

A small study of required step size needed for stable closed-loop behaviour was conducted. This was done both in simulations and on the experimental system. The goal was to check if the fractional-order controllers demanded higher sampling frequency than the integer-order ones. A table summing up the results in seen below.

The fixed-step ode1 (Euler) solver was used for both simulation and experimental testing. In simulation an Oustaloup filter order of 21 and an interesting frequency interval of $[\omega_b, \omega_h] = [10^{-5}, 10^5]$ were used. In experimental testing an Oustaloup filter order of 9 and an interesting frequency interval of $[\omega_b, \omega_h] = [10^{-2}, 10^4]$ were used. The Oustaloup filter order in simulations was chosen in order to have a very accurate simulation. The Oustaloup filter order used in experiments needed to be lower than in simulations for the controllers to be able to run in real-time. The same goes for the interesting frequency intervals of the filters.

Table 7.7: Required time step size in seconds needed for stable closed-loop behaviour with different controllers.

	PID	FO-PID	PPF	FO-PPF_1	FO-PPF_2	FO-PPF_3
Simulation	$8.8 \cdot 10^{-6}$	$6.7 \cdot 10^{-6}$	$8.3 \cdot 10^{-6}$	$9.1 \cdot 10^{-6}$	$1.3 \cdot 10^{-6}$	$8.8 \cdot 10^{-6}$
Experimental	$4.5 \cdot 10^{-3}$	-	$1.7 \cdot 10^{-4}$	$3.7 \cdot 10^{-5}$	$7.3 \cdot 10^{-6}$	$5.9 \cdot 10^{-4}$

We can observe that the demands for low step size are higher in simulation than in experiments. In experiments on the AFM system the two integer-order controllers had the lowest required sampling frequency, while the FO-PID controller had a too high demand to be able to run in real-time on the xPC.

7.5 Observations

A gradient moving from top-left to right-bottom can be observed in all the scanned images. This is a result of the sample surface not being orthogonal to the AFM cantilever and probe during the scanning of the surface. The slope can be removed through image processing, but we chose not to do so when presenting these results.

Looking closely at the images it seems like the images scanned with low scan frequency (5 Hz) has a sharper edge between the flat surface and the circular indentations than what can be observed in the images scanned with a higher frequency (30 Hz).

Chapter 8

Discussion

8.1 Simulation Results

Looking at the simulation results of the optimized controllers in section 6, specifically the step responses seen in figure 6.2a, 6.4a, 6.6a, 6.8a, 6.10a and 6.12a, we can see that the regular PPF controller and the fractional-order PPF version 2 controller have achieved a settling time of almost half that of regular PID and FO-PID controllers. This is most likely a result of the damping controller which manages to dampen the three resonance modes more than what the PID and FO-PID controllers can manage. This enables a higher bandwidth for the systems while still damping the resonance modes sufficiently. This shows the somewhat superior performance of the PPF controller on a system with high resonant modes compared to that of the PID controller.

Taking a closer look at the bandwidth of the different controllers, it seems like all the regulators with a fractional-order integrator, i.e. the FO-PID controller and the FO-PPF version 1 and 3 controllers, have achieved higher bandwidth than the ones without fractional-order integrators. All the three regulators have achieved a bandwidth of around 1.54×10^3 rad/s which is noticeably higher than what the other controllers have achieved. However, looking at the gain at the 1st, 2nd and 3rd modes, or peaks in the bode diagrams, which hast been summarized in the table below, we realize that the controllers with fractional-order integrators are not damping the first resonant modes, but rather amplifying them with about 2.5 dB. Leading to the oscillations in the transient area of the step response plots.

Table 8.1: Controller gains at the different resonant modes.

Controller	1st Mode Gain (~1450 rad/s)	2nd Mode Gain (~2950 rad/s)	3rd Mode Gain (~5100 rad/s)	New peak introduced by regulator
PPF	−2.77 dB	−15.0 dB	−15.9 dB	-
FO-PPF_1	2.65 dB	−10.2 dB	−18.7 dB	-
FO-PPF_2	−14.0 dB	−23.5 dB	−27.8 dB	0.85 dB (197 rad/s)
FO-PPF_3	2.74 dB	−10.1 dB	−19.7 dB	4.56 dB (534 rad/s)
PID	−5.98 dB	−11.4 dB	−20.8 dB	-
FO-PID	2.16 dB	0.383 dB	−3.52 dB	13.2 dB (755 rad/s)

By studying the gains of the resonant modes of the fractional-order PID controller in table 8.1, we can see that the first and second modes are amplified a bit while the third mode is hardly damped. What is more, the controller has created a new peak at frequency 755 rad/s with an amplification of 13.2 dB, which is really bad. The result of this new peak can be seen in figure 6.12a where heavy oscillations can be seen in the transient part of the step response.

Another thing to comment on with regards to table 8.1 is the superior damping of the resonant modes performed by the FO-PPF_2 controller when compared to the others. Looking at the step response of the controller in figure 6.6a we hardly see any oscillations at all, just an overshoot of about 13% in the transient part. Similarly, to the FO-PID controller the FO-PPF_2 controller have also created a new peak. This peak is however tiny in comparison with a gain of 0.85 dB. It seems like this peak harmonizes with the overshoot when we notice that the frequency of the peak is 197 rad/s and that this amounts to a period of 0.032 seconds. which is of the same order as the period of the low frequency overshoot observed in 6.6a. From this, we can conclude that the introduction of fractional-order into the damping controller of the PPF scheme seems to have had a positive effect in terms of reducing oscillations, when compared to the regular integer-order PPF.

Now, looking at the response of the FO-PPF_3 controller which has both a fractional-order integrator in the tracking controller part and a fractional-order effect in the damping controller part, we can start to wonder why this controller shows a somewhat worse response than the FO-PPF_2 controller. If α_t is set to 1 in the FO-PPF_3, then we have the exact same controller structure as that of the FO-PPF_2 controller. Which means that the FO-PPF_3 controller should at least be able to perform as good as the FO-PPF_2 controller. Now, this revelation leads to the conclusion that the most optimal parameters for the FO-PPF_3 controller have not been found. Furthermore, we can start to theorize that there most likely exists better objectivity functions for this problem than the objectivity functions stated in section 5.9 and used in this study. Another objectivity function that has another optimal point that is more in line with our expectations of what is good performance for this system. Therefore, a search for a new objectivity function should be conducted. One that better catches the essence of what is meant by “good performance”.

8.2 Experimental Testing on AFM System

Looking at the images generated from the experimental testing of different controllers for the lateral motion of the AFM system, we see little difference for low scanning frequency (5 Hz). The only real difference that can be spotted is the inability of the FO-PPF_1 (figure 7.10a) and the PID (figure 7.25a) controllers to scan all the way out on the edges in the x-direction. This inability to scan out on the edges becomes even more apparent when looking at the images taken with a scanning frequency of 30 Hz.

Looking at the images scanned at high speed we see that the FO-PPF_1 (figure 7.10b) and the PID (figure 7.25b) controllers are doing even worse than before. Especially the PID controller, that only manages to scan an area of about 60% of the height of the reference signal. Looking at a zoomed in plot of the x-axis tracking graph in figure 7.26b we can observe the same. The PPF controller and the FO-PPF_3 controller on the other hand, is able to keep scanning all the way out on the edges even for a scanning frequency of 30 Hz, and tracks the reference signals in a much better way as can be seen in figure 7.5b, 7.20b, 7.6b and 7.21b.

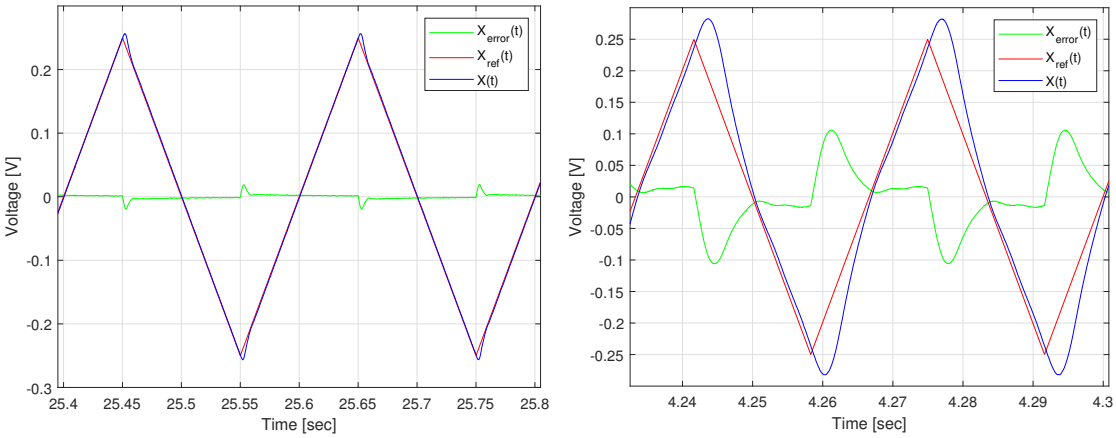
Something else that should be commented on with regards to the AFM images are the white pixels that can be seen. The white pixels indicate that data is missing and is a result of

how the images have been rendered from the raw data with the custom MATLAB function `print_AFM_image`, explained in section 5.13. Because the measured x and y position data have been used to place the measured z height data in the most correct pixels, some pixels may end up with zero data samples, and they are rendered white.

In the bottom right corner of figure 7.25b we see several pixels that seem somewhat misplaced and form several loops. This is just sampled data from the moment that the controller is turned on and the initial movement to the start position of the scan and can be ignored. Ideally these samples could have been removed in the `print_AFM_image` function, but no time was allocated to fixing this.

Looking at the sum of images taken with both high and low scan frequency and with different lateral motion controllers as well as thinking about the rendering method used in `print_AFM_image`, it seems this method of rendering the experimental data removes some of the distortions that can be observed in AFM images generated by for example the commercial XEP program when scanning with high frequency, like squashed circles and the likes. However, based on these images which only shows a little part of the circles, it is hard to conclude.

Now, looking at the x-axis tracking graphs for the FO-PPF_3 controller, figure 7.21a and 7.21b, we observe that this controller do not show a "steady-state" like error when tracking the triangle wave shaped reference signal, like the rest of the controllers do. In figure 7.21a we see that after the reference signal have reached the top, the position signal overshoots, like the rest of the controllers, but the FO-PPF_3 controller as opposed to the other controllers, manages to recover and close the gap to the reference signal. The same tracking behaviour can be seen in figure 7.21b, the only difference is the amount of overshoot. Below, a set of figures showing the x-axis tracking error with the FO-PPF_3 controller for 5 Hz and 30 Hz can be seen. The $X(t)$ and $X_{ref}(t)$ graphs have been shifted to oscillate around zero in order to bring them to the same level as the error graph $X_{error}(t)$ and make the connection between the graphs more apparent.

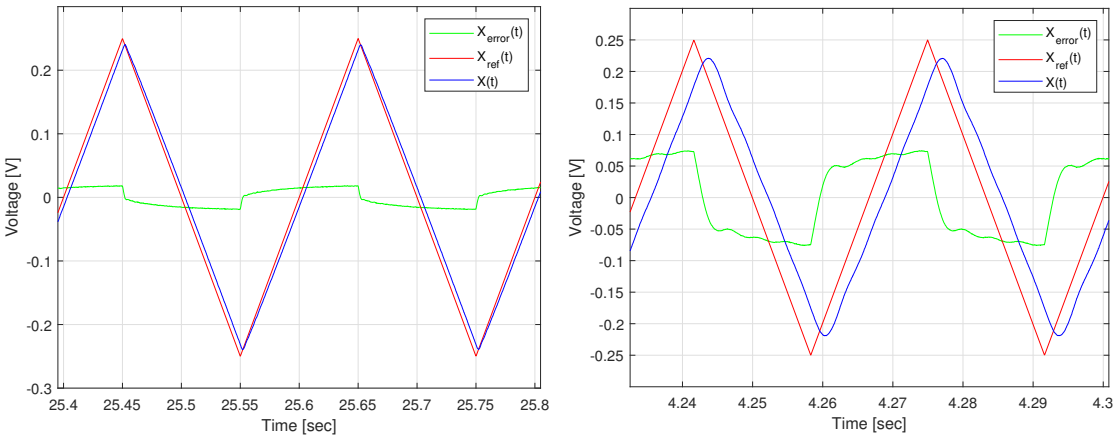


(a) 5 Hz (b) 30 Hz

Figure 8.1: X-axis tracking error with FO-PPF version 3 controller. $X(t)$ and $X_{ref}(t)$ is shifted to oscillate around zero.

Looking at the x-axis tracking graphs of the similar FO-PPF_1 controller (figure 7.11a and 7.11b), which only has fractional-order in the tracking controller, we observe the somewhat inverse result from that of the FO-PPF_3 controller. Looking at figure 7.11a we clearly see an increase in the error between $X(t)$ and $X_{ref}(t)$ while moving from the top to the bottom, or from the bottom to the top of a ramp signal. The same can be observed in 7.11b. It is safe

to say that this particular behaviour is not wanted, and that the behaviour of the FO-PPF_3 controller is a lot better. Below the x-axis tracking error with FO-PPF_1 controller is shown.



(a) 5 Hz (b) 30 Hz

Figure 8.2: X-axis tracking error with FO-PPF version 1 controller. $X(t)$ and $X_{ref}(t)$ is shifted to oscillate around zero.

The author is somewhat convinced that the behaviour explained above has something to do with the fractional-order integrator in the FO-PPF_1 and FO-PPF_3 controllers. Looking at the value of the fractional-order parameter α_t in the FO-PPF_1 controller in table 6.3a, we find that $\alpha_t = 0.805$. Doing the same for the FO-PPF_3 controller, we get $\alpha_t = 1.405$ from table 6.7a. It is known that a double integrator is necessary for exact tracking of ramp like signals [50]. But the double integrator has a negative impact on the closed-loop system's phase characteristic, making a double integrator controller system hard to implement. In light of this, the behaviour of the FO-PPF_3 presented may suggest that a fractional-order integrator with an order between one and two can be a better fit for the tracking of ramp-like signals, compared to a double integrator, when the stability properties of the plant becomes an issue. A more in-depth study of this should therefore be conducted.

Moving on to the staircase reference signal tracking in the y-direction, we clearly see the quantization done by the analogue to digital converter. This is a result of bad utilization of the dynamic area of the ADC. The ADCs have a dynamic area of $-10\,\text{V}$ to $10\,\text{V}$ and for a scan with 256 scan lines about $0.5\,\text{V}$ of this area is used. In hindsight it is clear that additional signal conditioning should have been performed between the dSpace controller board and the XE-70 control board. However, the resolution of the ADC seems to have been just enough to be able to scan with 256 scan lines in an area amounting to $0.5\,\text{V}$ x $0.5\,\text{V}$.

Now, looking at the bode diagrams of the closed-loop frequency response for each of the six controllers. We observe that the experimental frequency response and the frequency response of the MATLAB model on the most part agrees. The worst discrepancy can be found in figure 7.28 for the optimized FO-PID controller, where the new top introduced by the controller is a lot higher than the modelled. Leading to a gain of almost 20 dB at $755\,\text{rad/s}$ instead of the theorized 13.2 dB. This is one of the reasons why the optimized FO-PID controller could not be used for scanning. The author is quite certain that there exist good sets of parameters for the FO-PID controller for the control of the lateral motion of the AFM. The problem is that the proposed tuning method and objectivity function have not been able to find them and find them fast enough.

8.3 Required Sampling Frequencies

From section 7.4 and table 7.7 we observe that a higher sampling frequency was needed in the simulations than for the experimental testing, for the closed-loop systems to be stable. At first it was thought that the simulation of the plant is the reason for the low sample step times, but comparing the upper Oustaloup interesting frequency intervals used in simulation and experimental testing, another theory is that the added decade of frequencies in the simulation case is demanding higher sampling frequencies. So, the Oustaloup filter interesting frequency intervals seem to influence the needed sampling frequency.

Another thing that can be observed is that the integer-order controllers, i.e. PID and PPF controllers seem to have a lower requirement for sampling frequency. This suggests that fractional-order controllers require higher sampling frequencies than integer-order controllers. When also thinking about that the Oustaloup filters introduce a lot of additional states that must be calculated at each sample, it can be concluded that fractional-order controllers are more calculation intensive. If calculation of fractional-order derivatives and integrals with high accuracy is needed, and the dynamics of the system has a high frequency, we are going to need high performance dedicated hardware in order to realize fractional-order transfer functions with the use of Oustaloup filters. Therefore, fractional-order controllers appear to have a high implementation cost in terms of processing power.

8.4 About the Tuning Method

Two different objectivity functions were tried out and used with the controller optimization method presented in this thesis. They are presented in section 5.9. Both are of a frequency domain type, and therefore only uses information about the frequency response. As the above discussion points out, it should be possible to formulate a better objectivity function. One that better catches what is meant with good performance. Perhaps even with faster convergence than the ones used. In addition to frequency domain type objectives, time domain objectives should also be possible to add to the presented method. But if the time domain objective relies on doing a simulation for each step of the optimization algorithm, the addition of such may increase the calculation time for each step. It should be mentioned that time domain objectives were tested out in early stages of this work, but it was dropped because of the increased calculation time and the fact that pure frequency domain objectives were found to work quite well.

In hindsight it appears like a simpler LTI model for the lateral motion stage could have been used. For example, a model where only the first two resonant modes were included would have reduced the transfer function model order, leading to less calculations during calculation of the frequency response and somewhat faster controller tuning.

Although all tests of the proposed controller tuning method have been conducted on open-loop stable systems, the method is yet to be tested on an open-loop unstable system. But, given that the use of Nyquist stability criterion applies to fractional-order systems of the kind treated in this thesis, the author feels quite certain that the proposed tuning method should also be valid for such cases. However, the frailty of the MATLAB code implementing the tuning method, leads the author to doubt that it will work with the current setup.

Regarding the use of the genetic algorithm. The GA and the utilized configuration of the method seems to work fine for the tuning of fractional-order controllers done in this work. There might however exist better policy functions for the different parts of the algorithm that would for

example enable faster convergence, but the search for such policies have been outside the scope of the thesis and have not been treated.

An observation regarding the GA that was made is that the mutation mechanism seems to be the key factor for driving the regulator tuning forward. By halting the genetic algorithm and looking at the population from time to time during controller tuning, the author is convinced that the random search done by the mutation mechanism is the main force driving the cost function down and the optimization forward.

It should again be made clear that the author is somewhat uncertain as to the stability analysis part of the optimization method and the general validity of the way it has been used for transfer functions with pseudo polynomials in the nominator and denominator, also referred to as fractional-order transfer functions. The results obtained indicate that there is some validity to the method, but the author is uncertain if this is just a lucky edge case or if the method is valid on a general level.

8.5 Comparison of FO and IO Control

Now, if we try to compare the system response with fractional-order controllers with the system response with integer-order controllers, based on both simulation results and experimental result, we cannot conclude that the introduction of fractional-orders have led to signification performance improvements. For example, comparing control with PPF and FO-PPF_2, looking at the simulation step responses in figure 6.2a and 6.6a we can see more oscillations in the step response for the regular PPF than for the FO-PPF_2 and the FO-PPF_2 seems to have a higher overshoot than the PPF, but in terms of settling time, they perform the same.

Considering the apparent high demands of processing power and the complexity of designing and tuning the fractional-order controllers, we cannot say to have gained anything new from the fractional-order control systems when compared to a similar integer-order control system. All-in-all it seems like fractional-order systems is not worth it.

8.6 On the Use of Oustaloup Filter for Regulator Realization

The way the fractional-order derivatives and integrals have been implemented with Oustaloup filters raises the question whether there actual is something to gain from the theory of fractional-order control systems. If one looks at the models of the controllers that are actually used in this thesis, one only sees a high number of linear differential equations or difference equations. The only fractional-order about the controllers, that can actually be said to exist, is the inherent structure created by the Oustaloup filter approximations. An inherent structure of lots of poles and zeros that together creates a frequency response approximating that of a theoretical fractional-order derivative or integral.

In the light of this, one might theorize that high-order controllers like the ones found through methods like H-infinity control or model reference control, have the ability to mimic the fractional-order approximations created by an Oustaloup filter. Given that the methods “find” this to be beneficial in terms of the applied objectivities.

The real issue with fractional-order integrals and derivatives for control seems to be the fact that they cannot be realized exactly using a computer. This is because an exact realization would require a state space model with infinity many states. Something which is infeasible to

implement on a computer with finite memory and finite processing power. On the other hand, one can argue that a finite approximation is in many cases good enough.

In light of this it seems like fractional-order controllers realized through Oustaloup filters have a serious problem. The fact that the controllers in the end must be approximated and implemented with pure integer-order transfer functions or linear state space models with high orders, raises the question whether implemented controllers can actually be called fractional-order or not Without a better way of realizing these controllers than by approximations, the question can be raised whether fractional-order calculus can actually bring something new to the established field of control? The author is not yet certain.

8.7 Further Work and Possible Room for Improvements

Through the work with this thesis it has become apparent that the proposed tuning method for fractional-order systems, as well as integer-order systems, needs to be more thoroughly tested. Specifically, the way that stability for fractional-order systems has been calculated. And in addition, the theoretical foundation for the stability analysis of fractional-order systems with Nyquist's stability criterion should be thoroughly investigated, as the author has not been able to bring forth sufficient evidence for the validity of this.

Other objectivity functions, both in frequency domain and in time domain, should be tested with the method to see if they can improve the performance of the controllers found by the proposed tuning method.

In the event that stability analysis for fractional-order systems through the use of Nyquist stability criterion is validated, the fractional-order logarithmic Nyquist diagram presented in section 5.12 and the associated code could prove quite useful. Therefore, the correctness of the MATLAB code used for the creation of this diagram, as well as the correctness of the diagram itself should be validated.

In addition, it should be mentioned that manual testing shows that there is room for improvements in the functions `nyqlog_fotf` and `astep_fotf_freqresp` in terms of performance. For instance, some transfer functions are not rendered smoothly enough. Changing the calculation start point in `astep_fotf_freqresp` can fix this, but that requires manual intervention and trial and error. An additional minor feature that could be added to the `nyqlog_fotf` function is adding arrows to the curve, pointing in the direction of increasing frequency, easing visual inspection. Another thing is the possibility to configure the plot through passing of key-value pairs to the function, in the same way that configuration of the `ga` and a lot of other MATLAB functions are done.

As mentioned is section 8.2, the usefulness of fractional-order integrals for the tracking of ramp-like signals should be investigated. This may lead to better tracking performance of such signals for nanopositioning applications.

Chapter 9

Conclusion

The main goal of the thesis has been to test the applicability of fractional-order control theory on the nanopositioner of an AFM system, and to check if the use of such controllers can lead to performance improvements over standard solutions like PID-control. To that end, a set of fractional-order controllers have been designed for the control of the lateral positioning stage of an atomic force microscope. A PPF controller and a PID controller were augmented with the theory of fractional-order calculus and tuned by an experimental optimization method based on the genetic algorithm and Nyquist's stability criteria. The Oustaloup filter approximation technique was used to implement the fractional-order controllers. Testing and comparisons of the controllers have been performed both in simulations and experimentally on a commercial AFM system.

In addition to the fractional-order control experiments on the AFM, an introduction to the field of microscopy, AFM, and fractional-order control systems has been given. Furthermore, the developed tuning script and inquiries into the use of Nyquist's stability criterion for fractional-order systems, presented in this thesis, shows a clear lack of results linking the topic of Nyquist stability criterion and the topic of fractional-order systems in the literature. If the Nyquist stability criterion is shown to be valid for the kind of fractional-order transfer functions discussed in this thesis, the tools and methods developed in this thesis is sure to be of use in the field of fractional-order control.

Experiments from the testing of fractional-order controllers show that integral tracking controllers with fractional-order behave somewhat differently than the regular single integral tracking controllers. With a fractional-order integral of order $0 < \alpha < 1$, the controllers seem to show an inability to remove the tracking error when tracking ramp-like signals, in-fact the error increases over time. However, with a fractional-order integral of order $1 < \alpha < 2$ the controllers seem able to close the error gap. This is in line with theory. Furthermore, the use of fractional-order in the PPF damping controller seems to have some positive effect on the damping of the resonance modes, leading to less oscillations in the transient area of a step response, when compared to the regular PPF damping controller. The results also show the superiority of the PPF controller scheme in terms of bandwidth when compared to a regular PID controller. Apart from these observations, no significant performance improvements were found in introducing fractional-order integrals and derivatives, when compared to the regular integer-order variants of the PPF and PID controllers.

Lastly, regarding realization of fractional-order controllers through Oustaloup filters, several intricate questions were raised. Can fractional-order controllers actually be implemented with these filters? Can fractional-order controllers implemented through Oustaloup filters really offer something new to the field of control?

Appendix A

Note on the Grünwald-Letnikov Definition

The following note on the Grünwald-Letnikov definition is a refined version of the explaination given in [22] by the author of this thesis.

This note aims at explaining how the GL definition for fractional-order derivatives and integrals can be derived from the well-known definition of the derivative. From introductory mathematics courses on calculus it is well known that the definition of the integer-order derivative is

$$\frac{\mathrm{d}}{\mathrm{d}x}f(x) = \lim_{h\to 0}\frac{f(x)-f(x-h)}{h}. \tag{A.1}$$

By self-substitution, i.e. substitute $f(x)$ in (A.1) with the expression for $\frac{\mathrm{d}}{\mathrm{d}x}f(x)$, we get

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}f(x) = \lim_{h\to 0}\frac{f(x)-2f(x-h)+f(x-2h)}{h^2}. \tag{A.2}$$

If we now substitute (A.1) into (A.2) and keep doing this a few times, as is shown below, a distinct pattern for the n-th derivative emerge.

$$\begin{aligned}
\frac{\mathrm{d}^3}{\mathrm{d}x^3}f(x) &= \lim_{h\to 0}\frac{f(x)-3f(x-h)+3f(x-2h)-f(x-3h)}{h^3},\\
\frac{\mathrm{d}^4}{\mathrm{d}x^4}f(x) &= \lim_{h\to 0}\frac{f(x)-4f(x-h)+6f(x-2h)-4f(x-3h)+f(x-4h)}{h^4},\\
&\vdots\\
\frac{\mathrm{d}^n}{\mathrm{d}x^n}f(x) &= \lim_{h\to 0}\frac{1}{h^n}\sum_{j=0}^{n}(-1)^j\binom{n}{j}f(x-jh).
\end{aligned} \tag{A.3}$$

A closer look at the coefficients in front of all the $f(\cdot)$ terms, on the right hand side, reveals the numbers in Pascal's triangle in addition to a continuous change of sign.

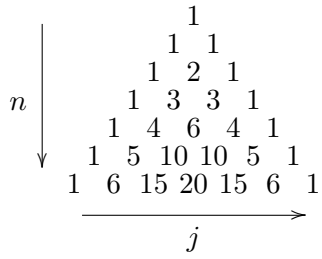


Figure A.1: Pascal's triangle

$$\binom{n}{j} = \frac{n!}{j!(n-j)!} \tag{A.4}$$

The coefficients in Pascal's triangle can be represented by the binomial coefficient. This makes it possible to write the n-th order derivative as a weighted sum of earlier function values, i.e. write it on the form (A.3).

Introducing the shift operator

$$z^j f(x) = f(x+j), \tag{A.5}$$

and using the binomial identity

$$(1+y)^n = \sum_{j=0}^{n} \binom{n}{j} y^j, \tag{A.6}$$

allows the rewriting of (A.3) in the following way:

$$\begin{aligned}
\frac{\mathrm{d}^n}{\mathrm{d}x^n} f(x) &= \lim_{h\to 0} \frac{1}{h^n} \sum_{j=0}^{n} (-1)^j \binom{n}{j} f(x-jh), \\
&= \lim_{h\to 0} \frac{1}{h^n} \sum_{j=0}^{n} (-1)^j \binom{n}{j} z^{-jh} f(x), \\
&= \lim_{h\to 0} \frac{1}{h^n} \sum_{j=0}^{n} \left[\binom{n}{j} (-z^{-h})^j \right] f(x), \\
&= \lim_{h\to 0} \frac{1}{h^n} (1-z^{-h})^n f(x), \\
\frac{\mathrm{d}^n}{\mathrm{d}x^n} f(x) &= \lim_{h\to 0} \left(\frac{1-z^{-h}}{h} \right)^n f(x).
\end{aligned} \tag{A.7}$$

The fractional derivative can now be introduced by substitution of the integer n with the real number α into (A.7). We get

$$\frac{\mathrm{d}^\alpha}{\mathrm{d}x^\alpha} f(x) = \lim_{h\to 0} \left(\frac{1-z^{-h}}{h} \right)^\alpha f(x). \tag{A.8}$$

The pseudo polynomial in (A.8) can now be expanded with the use of the equation for the binomial series, a generalization of the binomial identity:

$$(1+y)^{\alpha} = \sum_{j=0}^{\infty} \binom{\alpha}{j} y^j. \tag{A.9}$$

We get

$$\begin{aligned}
\frac{\mathrm{d}^{\alpha}}{\mathrm{d}x^{\alpha}} f(x) &= \lim_{h\to 0} \left(\frac{1 - z^{-h}}{h} \right)^{\alpha} f(x), \\
&= \lim_{h\to 0} \frac{1}{h^{\alpha}} \left(1 - z^{-h} \right)^{\alpha} f(x), \\
&= \lim_{h\to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\infty} \left[\binom{\alpha}{j} (-z^{-h})^j \right] f(x), \\
&= \lim_{h\to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\infty} (-1)^j \binom{\alpha}{j} z^{-jh} f(x), \\
\frac{\mathrm{d}^{\alpha}}{\mathrm{d}x^{\alpha}} f(x) &= \lim_{h\to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\infty} (-1)^j \binom{\alpha}{j} f(x - jh).
\end{aligned} \tag{A.10}$$

Now, assuming that $f(x) = 0$ for $x < x_0$ and taking advantage of the fact that $\lim_{h\to 0} \frac{x - x_0}{h} \to \infty$, the equation (A.10) can be written on the form:

Grünwald-Letnikov definition for fractional-order derivatives (Form 1)

$$ {}_{x_0}\mathcal{D}_x^{\alpha} f(x) = \lim_{h\to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\frac{x-x_0}{h}} (-1)^j \binom{\alpha}{j} f(x - jh) \tag{A.11}$$

If we define $w_j = (-1)^j \binom{\alpha}{j}$ and calculate the ratio between w_j and w_{j-1} for any $j = 0, 1, 2, \ldots, \infty$, we find that

$$\begin{aligned}
\frac{w_j}{w_{j-1}} &= \frac{(-1)^j \binom{\alpha}{j}}{(-1)^{j-1} \binom{\alpha}{j-1}} = (-1) \cdot \frac{\Gamma(\alpha+1)}{\Gamma(j+1)\Gamma(\alpha-j+1)} \cdot \frac{\Gamma((j-1)+1)\Gamma(\alpha-(j-1)+1)}{\Gamma(\alpha+1)}, \\
&= (-1) \cdot \frac{1}{j\Gamma(j)\Gamma(\alpha-j+1)} \cdot \frac{\Gamma(j)(\alpha-j+1)\Gamma(\alpha-j+1)}{1} = -\frac{\alpha-j+1}{j}, \\
&= 1 - \frac{\alpha+1}{j},
\end{aligned}$$

and

$$\begin{aligned}
w_0 &= (-1)^0 \cdot \binom{\alpha}{0} = (-1)^0 \cdot \frac{\Gamma(\alpha+1)}{\Gamma(0+1)\Gamma(\alpha-0+1)} = \frac{\Gamma(\alpha+1)}{\Gamma(1)\Gamma(\alpha+1)}, \\
&= \frac{1}{\Gamma(1)} = \frac{1}{1} = 1.
\end{aligned}$$

Therefore an iterative scheme to find the weighting coefficients $\{w_j\}$ for $j = 0, 1, 2, \ldots, \infty$ based on the degree *alpha* is

$$w_0 = 1, \quad w_j = \left(1 - \frac{\alpha + 1}{j}\right).$$

So, equation (A.11) can be rewritten in an equivalent form:

Grünwald-Letnikov definition for fractional-order derivatives (Form 2)

$$ {}_{x_0}\mathcal{D}_x^{\alpha} f(x) = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\frac{x - x_0}{h}} w_j f(x - jh) \tag{A.12}$$

$$w_0 = 1, \quad w_j = \left(1 - \frac{\alpha + 1}{j}\right) w_{j-1} \tag{A.13}$$

Appendix B

Content of Attached Zip Folder

A directory tree showing the most important files and folders of the delivery zip can be found below. A short explanation of the different files and folders can be found on the next few pages.

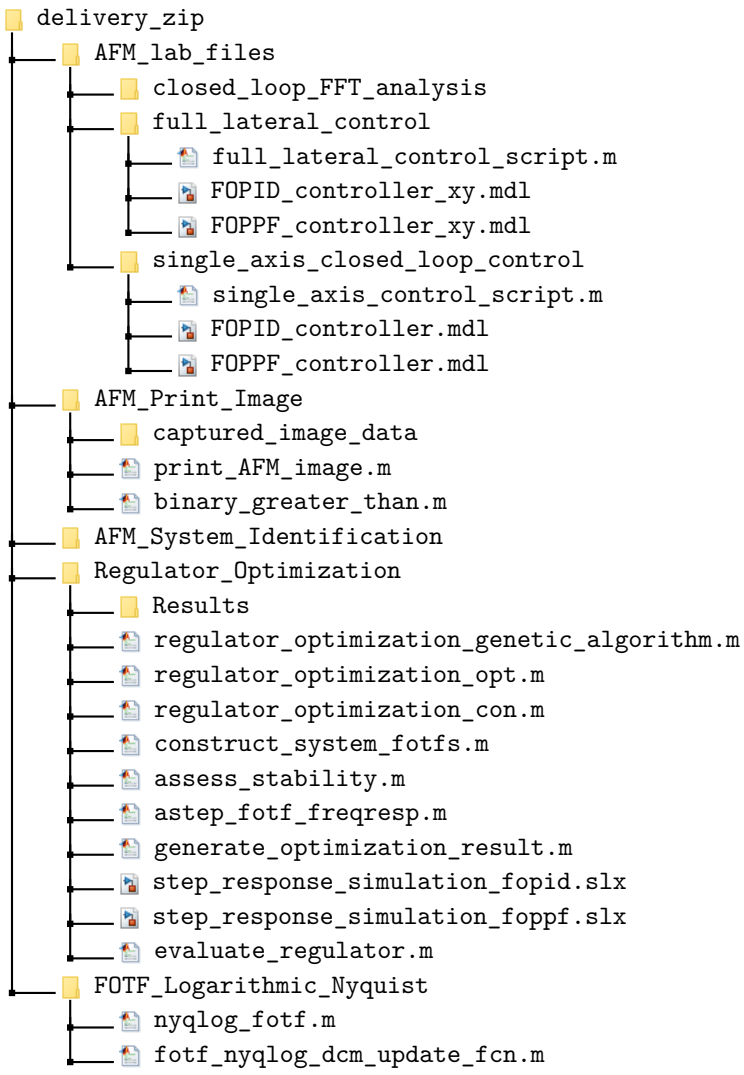


Figure B.1: Content of attached zip-file.

AFM_lab_files/closed_loop_FFT_analysis
Folder containing frequency response data of closed-loop system, captured with SR780, with different controllers in addition to plots of the data.

full_lateral_control
Folder containing files used at the AFM lab for testing the different controllers and capturing image scan data. `FOPPF_controller_xy.mdl` and `FOPID_controller_xy.mdl` are the Simulink models that are compiled down to code that is later run on the xPC.
`full_lateral_control_script.m` is used to choose regulator and configure scanning trajectory, and must be run before the Simulink models are compiled. `FOPPF_controller_xy.mdl` implements all the PPF based controllers, while `FOPID_controller_xy.mdl` implements the PID and the FO-PID controllers.

single_axis_closed_loop_control
Folder contains files used at the AFM lab for testing the different controllers on a single axis. Supports step and triangle wave reference tracking, as well as tracking an analogue reference signal from the dSpace controller board for use in closed-loop frequency response analysis with the SR780. `FOPPF_controller.mdl` and `FOPID_controller.mdl` are the Simulink models that are compiled down to code that is later run on the xPC. `full_lateral_control_script.m` is used to choose regulator and configure scanning trajectory, and must be run before the Simulink models are compiled. `FOPPF_controller.mdl` implements all the PPF based controllers, while `FOPID_controller.mdl` implements the PID and the FO-PID controllers.

AFM_Print_Image/captured_image_data
Folder containing the AFM data captured with the `full_lateral_control` files at the lab.

print_AFM_image.m
Script used for the generation of 2D images from the data in `captured_image_data` folder. Uses the `binary_greater_than` function to place each z sample value in the most correct pixel.

AFM_System_Identification
Folder containing SR780 captured frequency response data of AFM lateral positioning stage, identified transfer functions identified by toolbox `systemIdentification` and plots of data and transfer functions.

Regulator_Optimization/Results
Folder containing optimization results from the controller tuning method with main file `regulator_optimization_genetic_algorithm.m`.

regulator_optimization_genetic_algorithm.m
The main regulator tuning file used to start controller optimization tuning. Contains different configuration options like:

- Choice of regulator to tune.
- Choice of plant model used in the tuning of the regulator.
- Set the boundaries of the optimization problem.
- Different configuration options for the genetic algorithm.
- Chose if the optimization should start with the population in the workspace variable `POPULATION` or randomly generate a new one.

regulator_optimization_opt.m
Objectivity function used by regulator_optimization_genetic_algorithm.m.

regulator_optimization_con.m
Non-linear constraint function used by regulator_optimization_genetic_algorithm.m. Stability constraints are implemented in this function by a call to the assess_stability.m function.

construct_system_fotfs.m
Helper function used to construct a fractional-order transfer function (fotf object) of the system from the plant transfer function, regulator identifier and optimization variables.

assess_stability.m
Function used to automatically evaluate the Nyquist stability criterion and find gain value intervals where the closed-loop system is stable. Calls the astep_fotf_freqresp function in order to evaluate the systems frequency response.

astep_fotf_freqresp.m
Function used for evaluating the frequency response of the given fractional-order transfer function (fotf object) in an adaptive way that makes sure all the relevant dynamics are captured, while at the same time avoiding an exhaustive evaluation of the relevant frequency interval.

generate_optimization_result.m
Script used for stability analysis, plotting of relevant graphs and automatic saving of results from regulator optimization tuning runs done by the regulator_optimization_genetic_algorithm.m script to the Regulator_Optimization/Results folder. This script must be manually run after the regulator_optimization_genetic_algorithm.m script have terminated.

step_response_simulation_(fopid/foppf).slx
Simulink models used by the generate_optimization_result.m script to simulate the systems response to a reference step. The models use Oustaloup filters for the realization of fractional-order derivatives and integrals.

evaluate_regulator.m
Function for adding additional evaluation criteria to the optimization results generated by the generate_optimization_result.m. This function calculates bandwidth, Integral of Squared Error (ISE) of a simulated step response and max of sensitivity function. Must be manually called on results in the Regulator_Optimization/Results folder.

FOTF_Logarithmic_Nyquist/nyqlog_fotf.m
Function for plotting of a logarithmic Nyquist graphs from a tf object or fotf object. The function uses the astep_fotf_freqresp function for calculation of the frequency response.

fotf_nyqlog_dcm_update_fcn.m
Graph tooltip callback function used to display data values in the logarithmic Nyquist graph when a point on the Nyquist curve is selected.

Appendix C

Experimental Fractional-order Controller Tuning Scripts

Listing C.1: `regulator_optimization_genetic_algorithm.m`

```
%% Load Plant Model
ident_tfs = load('../Identified_transfer_functions/srs_data_tfs.mat');
G = ident_tfs.tf9;
G = fotf(G);

%% Choice of controller/regulator to control plant, and optimize
%regulator_type = 'PPF';
%regulator_type = 'FO-PPF_1';
%regulator_type = 'FO-PPF_2';
%regulator_type = 'FO-PPF_3';
regulator_type = 'FO-PID';
%regulator_type = 'PID';

%% Regulator Options
switch regulator_type
    case 'PPF'
        num_unstable_poles_open_loop = 0;
        %                   [   k_t,    k_d,  zeta_d,  omega_d]
        lower_bound_x = [     1,      1,       0,   1*10^3];
        upper_bound_x = [1*10^3, 1*10^3,       5,   3*10^4];
    case 'FO-PPF_1'
        num_unstable_poles_open_loop = 0;
        %                   [   k_t, alpha_t,    k_d,  zeta_d,  omega_d]
        lower_bound_x = [     1,    0.01,      1,       0,   5*10^2];
        upper_bound_x = [1*10^4,     2.0, 1*10^3,      10,   2*10^4];
    case 'FO-PPF_2'
        num_unstable_poles_open_loop = 0;
        %                   [   k_t,    k_d, zeta_d, omega_d,   beta_d]
        lower_bound_x = [     1, 1*10^-3,    0.1,  5*10^2,     0.01];
        upper_bound_x = [1*10^5,  1*10^3,      5,  1*10^6,     1.99];
    case 'FO-PPF_3'
        num_unstable_poles_open_loop = 0;
        %                   [    k_t, alpha_t,     k_d, zeta_d, omega_d, beta_d]
        lower_bound_x = [1*10^-2,    0.9,  1*10^-3,    0.1,  1*10^0,   0.01];
        upper_bound_x = [ 1*10^6,   1.99,  1*10^3,     10,  1*10^6,   1.99];
    case 'FO-PID'
        num_unstable_poles_open_loop = 0;
        %                   [    k_p,     k_i,     k_d, mu_i,  mu_d,    tau_f]
        lower_bound_x = [1*10^-2,  1*10^0, 1*10^-2, 0.01,  0.01,   1*10^-2];
        upper_bound_x = [ 1*10^2,  1*10^8,  1*10^2, 1.99,  1.99,   1*10^4];
    case 'PID'
        num_unstable_poles_open_loop = 0;
        %                   [    k_p,     k_i,     k_d,    tau_f]
        lower_bound_x = [1*10^-8, 1*10^0, 1*10^0,  1*10^-2];
```

```
45          upper_bound_x = [ 1*10^2,  1*10^4,  1*10^4,   1*10^6];
46      otherwise
47          error('Choose a supported regulator or implement support for new regulator');
48  end
49
50  %% Set Objectivity and Constraint functions
51  objective_function  = @(x) regulator_optimization_opt(x,G,regulator_type);
52  constraint_function = @(x) ...
        regulator_optimization_con(x,G,regulator_type,num_unstable_poles_open_loop);
53
54  %% Genetic Algorithm (GA) optimization Options
55  population_size       = 100;
56
57  % Reproduction Options: Controls number of elites, crossover and mutation
58  %                       individuals in next generation.
59  num_elite_individuals = ceil(0.05*population_size);
60  crossover_fraction    = 0.6;
61
62  % Stopping Conditions
63  max_num_generations   = 10;
64
65  max_stall_generations = 10;
66  function_tolerance    = 1e-20;
67
68  fitness_limit         = -inf;
69
70  time_limit            = inf;
71
72  stall_time_limit      = inf;
73
74  % Use previous end population as start population
75  use_prev_pop_as_start_pop = true;
76
77  % Use a previously saved random seed to enable repeatability
78  use_previous_rng_state = false;
79
80  %%
81  if use_prev_pop_as_start_pop == true
82      initial_population = POPULATION;
83  else
84      initial_population = [];
85  end
86  initial_population_range = [lower_bound_x; upper_bound_x];
87
88  ga_options = optimoptions(@ga,...
89                  'Display', 'diagnose',...
90                  'PlotFcn', {@gaplotbestf},...
91                  'PlotInterval', 1,...
92                  'PopulationSize',           population_size,...
93                  'InitialPopulationMatrix',  initial_population,...
94                  'InitialPopulationRange',   initial_population_range,...
95                  ...
96                  'FitnessScalingFcn',        @fitscalingrank,...
97                  'SelectionFcn',             @selectionstochunif,...
98                  'CrossoverFcn',             @crossoverscattered,...
99                  'MutationFcn',              @mutationadaptfeasible, ...
100                 'EliteCount',               num_elite_individuals,...
101                 'CrossoverFraction',        crossover_fraction,...
102                 'NonlinearConstraintAlgorithm', 'auglag',...
103                 ...
104                 'MaxGenerations',           max_num_generations,...
105                 'MaxStallGenerations',      max_stall_generations,...
106                 'FunctionTolerance',        function_tolerance,...
107                 'FitnessLimit',             fitness_limit,...
108                 'MaxTime',                  time_limit,...
109                 'MaxStallTime',             stall_time_limit...
110                 );
111
112 %% Run Optimization with Genetic Algorithm
113 if use_previous_rng_state == true
114     rng(rng_saved_state);
```

```
115 end
116 rng_saved_state = rng;
117
118 length_x = length(lower_bound_x);
119
120 start_optimization = tic;
121 [x_optimal,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES] = ga(...
122                                     objective_function, length_x,...
123                                     [],[],[],[],...
124                                     lower_bound_x,upper_bound_x,...
125                                     constraint_function,...
126                                     ga_options);
127 optimization_elapsed_time = toc(start_optimization);
```

Listing C.2: `regulator_optimization_opt.m`

```
1  function y = regulator_optimization_opt(x, G, reg_type)
2      %% Construct closed-loop system transfer function
3      [~, T, ~] = construct_system_fotfs(G, reg_type, x);
4
5      %% Calculate frequency responses
6      w_low  = -1;
7      w_high = 4;
8      N      = 1000;
9      W = logspace(w_low, w_high, N);
10
11     Z_T   = freqresp(1i*W, T);
12     T_MAG = 20.*log10(abs(Z_T));
13
14     %% Objectivity choice
15     objectivity_to_use = 2;
16
17     %% Objectivity function calculation
18     switch objectivity_to_use
19         case 1
20             T_vec = 1-abs(squeeze(T_MAG));
21             objective = norm(T_vec, 2);
22         case 2
23             obj_weight = 0.1; % [0-1] Weight hight bandwidth against
24                               % flat closed-loop response up to bandwidth.
25             bw_defined_at_mag_dB = -6;
26
27             [bw, idx] = find_bandwidth(T_MAG, W, bw_defined_at_mag_dB);
28             T_vec = 1-abs(squeeze(T_MAG(1:idx)));
29             obj1 = norm(T_vec/idx, 2);
30             obj2 = -bw;
31             objective = (1-obj_weight)*obj1 + obj_weight*obj2;
32         otherwise
33             error(['Objectivity function nr. ', ...
34                 num2str(objectivity_to_use), 'does not exist']);
35     end
36
37     %% Return objective value
38     y = objective;
39 end
40
41 function [bandwidth, index] = find_bandwidth(T_mag_dB, W, magnitude_dB_at_bandwidth)
42     mag_dB_at_bw = magnitude_dB_at_bandwidth;
43
44     bw  = W(1); idx = 1;
45     for j = 1:length(T_mag_dB)-1
46         if (T_mag_dB(j) <= abs(mag_dB_at_bw) && T_mag_dB(j) >= -abs(mag_dB_at_bw))
47             bw  = W(j);
48             idx = j;
49         else
50             break;
51         end
52     end
53
```

```
54      bandwidth = bw;
55      index     = idx;
56  end
```

Listing C.3: `regulator_optimization_con.m`

```
1   function [c, ceq] = regulator_optimization_con(x, G, reg_type, Np)
2       %% Construct open-loop system transfer function
3       L = construct_system_fotfs(G, reg_type, x);
4
5       %% Check regulator open loop stability so to adjust needed encirclement number Np
6       % N_p: Needed encirclements in nyquist diagram for closed-loop stability
7       switch reg_type
8           case 'FO-PPF_2'
9               zeta_d = x(3);
10              beta_d = x(5);
11              if zeta_d > -cos(pi/2*beta_d)
12                  N_p = Np;
13              else
14                  N_p = Np + 2;
15              end
16          case 'FO-PPF_3'
17              zeta_d = x(4);
18              beta_d = x(6);
19              if zeta_d > -cos(pi/2*beta_d)
20                  N_p = Np;
21              else
22                  N_p = Np + 2;
23              end
24          otherwise
25              N_p = Np;
26      end
27
28      %% Evaluate Stability (Nyquist stability criterion approach)
29      w_low  = 10^-10;
30      w_high = 10^5;
31      [isStable,~,GMs_dB,~,~,~] = assess_stability(L,N_p,{w_low,w_high});
32
33      %% Calculate Equality Conditions (Stability)
34      c_eq1 = 100*(1 - isStable);
35
36      ceq = [c_eq1];
37
38      %% Calculate Inequality Conditions (Stability)
39      gain_margin_db = 6;
40
41      if size(GMs_dB,2) ~= 0
42          c_ineq1 = gain_margin_db - GMs_dB(2,1);
43          c_ineq2 = GMs_dB(1,1) - gain_margin_db;
44      else
45          c_ineq1 = -realmax;
46          c_ineq2 =  realmax;
47      end
48
49      if c_ineq1 == inf
50          c_ineq1 = realmax;
51      elseif c_ineq1 == -inf
52          c_ineq1 = -realmax;
53      end
54      if c_ineq2 == inf
55          c_ineq2 = realmax;
56      elseif c_ineq2 == -inf
57          c_ineq2 = -realmax;
58      end
59
60      c = [c_ineq1, c_ineq2];
61  end
```

Listing C.4: `construct_system_fotfs.m`

```
function [L, T, So, misc] = construct_system_fotfs(G_plant, reg_type, reg_param)
    s = fotf('s');

    %% Input guard
    if isa(G_plant, 'tf')
        G_plant = fotf(G_plant);
    elseif ~isa(G_plant, 'fotf')
        error(['Unsupported type ', class(G_plant), ' of argument G']);
    end

    %% Extract Regulator Parameters from x based on regulator
    x = reg_param;

    switch reg_type
        case 'PPF'
            k_t = x(1);
            k_d = x(2); zeta_d  = x(3); omega_d = x(4);
        case 'FO-PPF_1'
            k_t = x(1); alpha_t = x(2);
            k_d = x(3); zeta_d  = x(4); omega_d = x(5);
        case 'FO-PPF_2'
            k_t = x(1);
            k_d = x(2); zeta_d  = x(3); omega_d = x(4); beta_d = x(5);
        case 'FO-PPF_3'
            k_t = x(1); alpha_t = x(2);
            k_d = x(3); zeta_d  = x(4); omega_d = x(5); beta_d = x(6);
        case 'FO-PID'
            k_p = x(1); k_i  = x(2); mu_i  = x(4);
            k_d = x(3); mu_d = x(5); tau_f = x(6);
        case 'PID'
            k_p = x(1); k_i = x(2); k_d = x(3); tau_f = x(4);
        otherwise
            error('Choose a supported regulator or implement support for new regulator');
    end

    %% Construct Regulators and Define System-Interconnection
    % G - Plant transfer function
    % C - Forward Controller transfer function
    % F - Backward controller transfer function
    % L - Open-loop transfer function
    switch reg_type
        case 'PPF'
            C_t = -k_t/s;
            C_d = -(k_d*omega_d^2)/(s^2 + 2*zeta_d*omega_d*s + omega_d^2);

            C = C_t*C_d;
            F = 1 + C_t^(-1);
            L = F*G_plant*C;

            misc = [C_t, C_d];
        case 'FO-PPF_1'
            C_t = -k_t/(s^alpha_t);
            C_d = -(k_d*omega_d^2)/(s^2 + 2*zeta_d*omega_d*s + omega_d^2);

            C = C_t*C_d;
            F = 1 + C_t^(-1);
            L = F*G_plant*C;

            misc = [C_t, C_d];
        case 'FO-PPF_2'
            C_t = -k_t/s;
            C_d = -(k_d*omega_d^2)/(s^(2*beta_d) + 2*zeta_d*omega_d*s^(beta_d) + ...
                omega_d^2);

            C = C_t*C_d;
            F = 1 + C_t^(-1);
            L = F*G_plant*C;

            misc = [C_t, C_d];
        case 'FO-PPF_3'
```

```
70              C_t = -k_t/(s^alpha_t);
71              C_d = -(k_d*omega_d^2)/(s^(2*beta_d) + 2*zeta_d*omega_d*s^(beta_d) + ...
                    omega_d^2);
72
73              C = C_t*C_d;
74              F = 1 + C_t^(-1);
75              L = F*G_plant*C;
76
77              misc = [C_t, C_d];
78          case 'FO-PID'
79              C = k_p + k_i/(s^mu_i) + (k_d*s^(mu_d))/(1+tau_f*s^mu_d);
80
81              L = G_plant*C;
82              misc = [C];
83          case 'PID'
84              C = k_p + k_i/s + (k_d*s)/(1+tau_f*s);
85
86              L = G_plant*C;
87              misc = [C];
88          otherwise
89              error(['reg_type = ', reg_type, ' is not supported! If it should be ...
                    supported then add it!']);
90      end
91
92      %% Calculate Sensitivity and Complementary Sensitivity functions
93      switch nargout
94          case 1
95              % Allready finished so do nothing
96          case {2,3,4}
97              So = (1+L)^(-1);
98              T  = G_plant*C*So;
99          otherwise
100             error('Number of output arguments not supported')
101     end
102
103 end
```

Listing C.5: `assess_stability.m`

```
1  function [isStable, GMs, GMs_dB, neg_sgn_GMs_dB, PMs, data] = ...
       assess_stability(G,Np,w,re,im)
2  % assess_stability(G,Np,w) - Gives info about the stability of the
3  % transfer function G. Np is the number of poles in right half plane (rhp)
4  % of G. w is an array of frequencies used when calculating the frequency
5  % response of G during the stability assessment.
6  % assess_stability(G,Np,{w1,w2}) - Instead of giving an array of
7  % frequencies a lower (w1) and upper (w2) value of interesting frequencies
8  % can be given and the assess_stability will adaptively choose a set of
9  % frequencies to calculate the frequency respose.
10 % assess_stability(G,Np,w,re,im) - Another option is to give a set of
11 % frequencies and the associated frequency response in re and im.
12
13 switch nargin
14     case 2
15         [REAL,IMAG,~,~,w] = astep_fotf_freqresp(G);
16     case 3
17         %% Calculate frequency response when arguments re and im are not given
18         if isa(w,'cell') && length(w) == 2
19             w1 = w{1};
20             w2 = w{2};
21             if w1 > w2
22                 error('When passing w = {w1,w2}, w1 must be greater than w2');
23             end
24             [REAL,IMAG,~,~,w] = astep_fotf_freqresp(G, w1, w2);
25         else
26             H = bode(G,w);
27             [REAL,IMAG,w] = nyquist(H);
28             REAL = squeeze(REAL);
29             IMAG = squeeze(IMAG);
```

```
30          end
31      case 5
32          REAL = re;
33          IMAG = im;
34      otherwise
35          error('Number of arguments unsupported')
36  end
37
38  %% 1.) Find real axis crossings
39  [re_x_val, w_x_val, x_dir] = find_zero_crossings(REAL,IMAG,w);
40
41  %% 2.) Add data for w = 0 and w = inf
42  [a,an,b,bn,~] = fotfdata(G);
43
44  [an_lowest,  an_lowest_index]  = min(an);
45  [bn_lowest,  bn_lowest_index]  = min(bn);
46  [an_highest, an_highest_index] = max(an);
47  [bn_highest, bn_highest_index] = max(bn);
48
49  % Check if system is improper, biproper or strictly proper
50  % Calculate value at w = 0
51  w_x_val_w_zero  = 0;
52  if an_lowest > bn_lowest
53      re_x_val_w_zero = sign(b(bn_lowest_index))*sign(a(an_lowest_index))*inf; % Can be ...
            positive or negative, has to check this
54  elseif an_lowest < bn_lowest
55      re_x_val_w_zero = 0;
56  else
57      re_x_val_w_zero = b(bn_lowest_index)/a(an_lowest_index);
58  end
59
60  % Calculate value at w = inf
61  w_x_val_w_inf  = inf;
62  if an_highest > bn_highest
63      % Strictly proper
64      re_x_val_w_inf = 0;
65  elseif an_highest < bn_highest
66      % Improper
67      re_x_val_w_inf = sign(b(bn_highest_index))*sign(a(an_highest_index))*inf;
68  else
69      % Biproper
70      re_x_val_w_inf = b(bn_highest_index)/a(an_highest_index);
71  end
72
73  if isempty(x_dir)
74      dPsi_w_zero = calc_phase_change(G,0);
75      dPsi_w_inf  = calc_phase_change(G,inf);
76      dPsi_w_ninf  = calc_phase_change(G,-inf);
77  else
78      x_dir_w_zero = -x_dir(1);
79      x_dir_w_inf  = -x_dir(end);
80  end
81
82  re_x_val = [re_x_val_w_zero, re_x_val, re_x_val_w_inf];
83  w_x_val  = [ w_x_val_w_zero,  w_x_val,  w_x_val_w_inf];
84  x_dir    = [   x_dir_w_zero,    x_dir,    x_dir_w_inf];
85
86  %% 3.) Calculate encirclements
87  encircs = zeros(1,length(w_x_val)+1);
88
89  % Create lookup table to map frequency order with real axis crossing order
90  lutbl = zeros(4,length(re_x_val));
91  lutbl(1,:) = 1:length(re_x_val);
92  lutbl(2,:) = re_x_val;
93  lutbl(3,:) = w_x_val;
94  lutbl(:,:) = sortrows(lutbl(:,:)',2)';
95  lutbl(4,:) = 1:length(re_x_val);
96  lutbl_2(:,:) = sortrows(lutbl(:,:)',1)';
97
98  for j = 1:length(w_x_val)-1
99      % Find which values in encircs that should add rotations
```

```
100     k1 = lutbl_2(4,j);
101     k2 = lutbl_2(4,j+1);
102     if k1 > k2
103         k_values = k2:k1;
104     else
105         k_values = k1:k2;
106     end
107     k_values = k_values(1:end-1);
108
109     % Check real axis crossing direction and if the value from one crossing
110     % is lesser than or greater than the next. Add encirclements according
111     % to these checks.
112     if (x_dir(j) == -1 && re_x_val(j) > re_x_val(j+1)) || ...
113        (x_dir(j) ==  1 && re_x_val(j) < re_x_val(j+1))
114         encircs(k_values+1) = encircs(k_values+1) - 1;
115     elseif (x_dir(j) ==  1 && re_x_val(j) > re_x_val(j+1)) || ...
116             (x_dir(j) == -1 && re_x_val(j) < re_x_val(j+1))
117         encircs(k_values+1) = encircs(k_values+1) + 1;
118     end
119 end
120
121 %% 4.) Find stable regions
122 region_data_encircs           = encircs;
123 region_data_lower_real_value  = [-inf, lutbl(2,1:end)];
124 region_data_higher_real_value = [lutbl(2,1:end),  inf];
125
126 if region_data_lower_real_value(1)  == -inf && ...
127    region_data_higher_real_value(1) == -inf
128     region_data_encircs           = region_data_encircs(2:end);
129     region_data_lower_real_value  = region_data_lower_real_value(2:end);
130     region_data_higher_real_value = region_data_higher_real_value(2:end);
131
132 end
133 if region_data_lower_real_value(end)  == inf && ...
134    region_data_higher_real_value(end) == inf
135     region_data_encircs           = region_data_encircs(1:end-1);
136     region_data_lower_real_value  = region_data_lower_real_value(1:end-1);
137     region_data_higher_real_value = region_data_higher_real_value(1:end-1);
138 end
139
140 stable_regions = [];
141 for j = 1:length(region_data_encircs)
142     if region_data_encircs(j) == Np
143         stable_regions = [stable_regions,...
144                           [region_data_lower_real_value(j);...
145                            region_data_higher_real_value(j)]];
146     end
147 end
148
149 % Return information of the regions and their encirclements as value data
150 data = [region_data_encircs; region_data_lower_real_value; region_data_higher_real_value];
151
152 %% 5.) Check stability based on given Np value
153 make_stable_gain_pairs = [];
154 isStable = false;
155
156 for j = 1:size(stable_regions,2)
157     k1 = stable_regions(1,j);
158     k2 = stable_regions(2,j);
159     e1 = 1/abs(k1);
160     e2 = 1/abs(k2);
161     if sign(k1) == -1 && sign(k2) == -1
162         g_lower = e1;
163         g_upper = e2;
164         make_stable_gain_pairs = [make_stable_gain_pairs, [g_lower; g_upper]];
165     elseif sign(k1) == -1 && sign(k2) == 1
166         g_lower = e1;
167         g_upper = inf;
168         make_stable_gain_pairs = [make_stable_gain_pairs, [g_lower; g_upper]];
169
170         g_lower = -e2;
```

```
171             g_upper = 0;
172             make_stable_gain_pairs = [make_stable_gain_pairs, [g_lower; g_upper]];
173         elseif sign(k1) == 1 && sign(k2) == 1
174             g_lower = -e2;
175             g_upper = -e1;
176             make_stable_gain_pairs = [make_stable_gain_pairs, [g_lower; g_upper]];
177         else
178             if k1 < inf && k2 < inf
179                 error(['k1 > k2, is not possible, ', 'k1 = ', num2str(k1), ', k2 = ', ...
                        num2str(k2)]);
180             else
181
182             end
183         end
184
185         if k1 < -1 && -1 < k2
186             % Sets the isStable output value to true if some of the stable
187             % regions contain the -1 point.
188             isStable = true;
189         end
190     end
191
192     %% 6.) Calculate gain margins
193     GMs_dB          = [];
194     neg_sgn_GMs_dB = [];
195     for j = 1:size(make_stable_gain_pairs,2)
196         if sign(make_stable_gain_pairs(1,j)) == -1 || sign(make_stable_gain_pairs(2,j)) == -1
197             neg_sgn_GMs_dB = [neg_sgn_GMs_dB, 20*log10(abs(make_stable_gain_pairs(:,j)))];
198         else
199             GMs_dB          = [GMs_dB, 20*log10(abs(make_stable_gain_pairs(:,j)))];
200         end
201     end
202
203     GMs = make_stable_gain_pairs;
204
205     %% 7.) Calculate phase margins
206     [phase0, w0] = find_unity_gain_crossings(REAL,IMAG,w);
207
208     phase_margins = sort(180-abs(phase0),'ascend');
209
210     if isStable == true
211         PMs = phase_margins;
212     else
213         PMs = [];
214     end
215
216     end % assess_stability()
217
218     %% Helper functions
219     function [phase0, w0] = find_unity_gain_crossings(re, im, w)
220         unity_gain_crossing_pairs = [];
221         for j = 1:length(re)-1
222             z1 = abs(re(j)  +1i*im(j));
223             z2 = abs(re(j+1)+1i*im(j+1));
224             if (z1 < 1 && z2 > 1) || (z1 > 1 && z2 < 1)
225                 new_unity_gain_crossing_pair = [j; j+1];
226                 unity_gain_crossing_pairs = [unity_gain_crossing_pairs, ...
227                                              new_unity_gain_crossing_pair];
228             end
229         end
230
231         unity_gain_crossing_data = zeros(2,size(unity_gain_crossing_pairs,2));
232         for j = 1:size(unity_gain_crossing_data,2)
233             % Find approximate unity gain crossing through linear interpolation
234             x1 = re(unity_gain_crossing_pairs(1,j));
235             y1 = im(unity_gain_crossing_pairs(1,j));
236             x2 = re(unity_gain_crossing_pairs(2,j));
237             y2 = im(unity_gain_crossing_pairs(2,j));
238
239             x  = (x1+x2)/2;
240             y  = (y1+y2)/2;
```

```
241
242         % Find frequency value at approximate unity crossing point
243         w1 = w(unity_gain_crossing_pairs(1,j));
244         w2 = w(unity_gain_crossing_pairs(2,j));
245
246         w_x = (w1+w2)/2;
247         unity_gain_crossing_data(2,j) = w_x;
248
249         phase = angle(x+1i*y);
250         unity_gain_crossing_data(1,j) = phase;
251     end
252
253     phase0 = (180/pi)*unity_gain_crossing_data(1,:);
254     w0     = unity_gain_crossing_data(2,:);
255 end
256
257
258 function [re0, w0, crossing_direction] = find_zero_crossings(re,im,w)
259     % Find approximate crossings of real axis
260     sign_shift_pairs = [];
261     for i = 1:length(im)-1
262         if (im(i) > 0 && im(i+1) < 0)
263             % Detected falling sign shift (from + to -)
264             new_sign_shift_pair = [i; i+1; -1];
265             sign_shift_pairs = [sign_shift_pairs, new_sign_shift_pair];
266         elseif (im(i) < 0 && im(i+1) > 0)
267             % Detected rising sign shift (from - to +)
268             new_sign_shift_pair = [i; i+1; 1];
269             sign_shift_pairs = [sign_shift_pairs, new_sign_shift_pair];
270         end
271     end
272
273     % Linear interpolation of real axis crossings
274     zero_crossing_data = zeros(3,size(sign_shift_pairs,2));
275     for i = 1:size(zero_crossing_data,2)
276         % Find approximate real axis crossing
277         x1 = re(sign_shift_pairs(1,i));
278         y1 = im(sign_shift_pairs(1,i));
279         x2 = re(sign_shift_pairs(2,i));
280         y2 = im(sign_shift_pairs(2,i));
281
282         x  = x1 - y1*((x2-x1)/(y2-y1));
283         zero_crossing_data(1,i) = x;
284
285         % Find frequency value at crossing point
286         w1 = w(sign_shift_pairs(1,i));
287         w2 = w(sign_shift_pairs(2,i));
288
289         w_x = w1 - y1*((w2-w1)/(y2-y1));
290         zero_crossing_data(2,i) = w_x;
291
292         % Add info about crossing direction
293         zero_crossing_data(3,i) = sign_shift_pairs(3,i);
294     end
295
296     re0 = zero_crossing_data(1,:);
297     w0  = zero_crossing_data(2,:);
298     crossing_direction = zero_crossing_data(3,:);
299 end
300
301
302 function dPsi = calc_phase_change(G,w_val)
303     [a,an,b,bn,~] = fotfdata(G);
304     w = w_val;
305
306     % Calculate real and imag part of nominator and denominator, of fotf G
307     B_mag = b.*(w.^bn);
308     B_theta = pi/2*bn;
309     Br = B_mag*cos(B_theta)';
310     Bi = B_mag*sin(B_theta)';
311
```

```
312     A_mag = a.*(w.^an);
313     A_theta = pi/2*an;
314     Ar = A_mag*cos(A_theta)';
315     Ai = A_mag*sin(A_theta)';
316
317     % Calculate derivative with respect of w for Br, Bi, Ar and Ai
318     dB_mag = b.*bn.*(w.^(bn-1));
319     dBr = dB_mag*cos(B_theta)';
320     dBi = dB_mag*sin(B_theta)';
321
322     dA_mag = a.*an.*(w.^(an-1));
323     dAr = dA_mag*cos(A_theta)';
324     dAi = dA_mag*sin(A_theta)';
325
326     % Find derivative of phase with respect to w
327     F1  = Ar*Br + Ai*Bi;
328     dF1 = dAr*Br + Ar*dBr + dAi*Bi + Ai*dBi;
329     F2  = Ar*Bi - Ai*Br;
330     dF2 = dAr*Bi + Ar*dBi - dAi*Br - Ai*dBr;
331
332     dPsi = (dF2*F1 - F2*dF1)/(F1^2 + F2^2);
333 end
```

Listing C.6: `astep_fotf_freqresp.m`

```
1  function [re_out,im_out,mag_out,phase_out,w_out] = astep_fotf_freqresp(G, w_low, w_high)
2      %% Choose function behavior based on input arguments
3      switch nargin
4          case 1
5              w_start = 0;
6              w_stop  = inf;
7              w_interval_given = false;
8          case 3
9              w_start = w_low;
10             w_stop  = w_high;
11             w_interval_given = true;
12         otherwise
13             error('Number of input arguments not supported');
14     end
15
16     %% Configuration parameters (Tuneable)
17     storage_size = 5000;
18
19     % Precission tolerance of slope change between calculated values in degrees
20     d     = 5;
21
22     % Precission tolerance of step length magnitude compared to point magnitude
23     y     = 12;
24
25     % Multiplicative factor when changing step length up and down
26     alpha = 2;
27
28     % Tolerance relaxation value. Used when the method gets stuck [0,180/d - 1]
29     dr    = 1;
30
31     % Check extra frequency above w at stopping condition to be relatively
32     % sure that the algorithm is not stopped to early. Scale the stopping
33     % w with this value and check phase of that value.
34     extra_stopping_test_w_scale = 100;
35
36     % Change startpoint of w. The algorithm is started two times from this
37     % point. One calculates from startpoint_w and towards inf, while
38     % the other one calculates from startpoint_w and towards zero.
39     startpoint_w                = 100;
40
41     % If the algorithm only shows a circle or line (less than what you
42     % would expect from the given fractional-order transfer function), then
43     % try to change startpoint of w with "startpoint_w".
44
```

```
45      %% Configuration parameters (Static)
46      w_delta_start = 1e-6;
47      deltaAngleTolerance_inDeg_atMagZero      = 5;
48      deltaAngleTolerance_inDeg_atMagInf       = 5;
49      magnitudeLength_at_Zero_constantPhase    = 30;
50      magnitudeLength_at_Inf_constantPhase     = 30;
51      constantEndValue_re_tolerance            = 1e-6;
52      constantEndValue_im_tolerance            = 1e-6;
53      constantEndValue_phase_tolerance_in_deg = 0.1;
54
55      %% Extract coefficients and powers/orders from fotf object G
56      [a,an,b,bn,L] = fotfdata(G);
57
58      %% Calculate magnitude and phase at w = 0 and w = inf
59      if w_interval_given == false
60      [~, an_li] = min(an);
61      [~, bn_li] = min(bn);
62      [~, an_hi] = max(an);
63      [~, bn_hi] = max(bn);
64
65      % Calculate value at w = 0
66      w_zero  = 0;
67      magnitude_at_w_zero = abs(b(bn_li)/a(an_li))*w_zero^(bn(bn_li) - an(an_li));
68      phase_at_w_zero     = wrapToPi(pi*(sign(b(bn_li)) ~= sign(a(an_li))) ...
69                                     + pi/2*(bn(bn_li) - an(an_li)));
70      % Calculate value at w = inf
71      w_inf  = inf;
72      magnitude_at_w_inf = abs(b(bn_hi)/a(an_hi))*w_inf^(bn(bn_hi) - an(an_hi));
73      phase_at_w_inf     = wrapToPi(pi*(sign(b(bn_hi)) ~= sign(a(an_hi))) ...
74                                    + pi/2*(bn(bn_hi) - an(an_hi)));
75
76      % Calculate values used during stop checks
77      switch magnitude_at_w_zero
78          case 0
79              lowerPhase_at_wZero = wrapToPi(phase_at_w_zero - ...
                    (pi/180)*deltaAngleTolerance_inDeg_atMagZero);
80              upperPhase_at_wZero = wrapToPi(phase_at_w_zero + ...
                    (pi/180)*deltaAngleTolerance_inDeg_atMagZero);
81              constPhase_stopLength_in_dB_at_wZero = -magnitudeLength_at_Zero_constantPhase;
82          case inf
83              lowerPhase_at_wZero = wrapToPi(phase_at_w_zero - ...
                    (pi/180)*deltaAngleTolerance_inDeg_atMagInf);
84              upperPhase_at_wZero = wrapToPi(phase_at_w_zero + ...
                    (pi/180)*deltaAngleTolerance_inDeg_atMagInf);
85              constPhase_stopLength_in_dB_at_wZero = magnitudeLength_at_Inf_constantPhase;
86          otherwise
87              if 0 < magnitude_at_w_zero && magnitude_at_w_zero < inf
88                  re_at_wZero = magnitude_at_w_zero*cos(phase_at_w_zero);
89                  im_at_wZero = magnitude_at_w_zero*sin(phase_at_w_zero);
90              else
91                  error('NaN encountered');
92              end
93      end
94      switch magnitude_at_w_inf
95          case 0
96              lowerPhase_at_wInf = wrapToPi(phase_at_w_inf - ...
                    (pi/180)*deltaAngleTolerance_inDeg_atMagZero);
97              upperPhase_at_wInf = wrapToPi(phase_at_w_inf + ...
                    (pi/180)*deltaAngleTolerance_inDeg_atMagZero);
98              constPhase_stopLength_in_dB_at_wInf = -magnitudeLength_at_Zero_constantPhase;
99          case inf
100             lowerPhase_at_wInf = wrapToPi(phase_at_w_inf - ...
                    (pi/180)*deltaAngleTolerance_inDeg_atMagInf);
101             upperPhase_at_wInf = wrapToPi(phase_at_w_inf + ...
                    (pi/180)*deltaAngleTolerance_inDeg_atMagInf);
102             constPhase_stopLength_in_dB_at_wInf = magnitudeLength_at_Inf_constantPhase;
103         otherwise
104             if 0 < magnitude_at_w_inf && magnitude_at_w_inf < inf
105                 re_at_wInf = magnitude_at_w_inf*cos(phase_at_w_inf);
106                 im_at_wInf = magnitude_at_w_inf*sin(phase_at_w_inf);
107             else
```

```
108                 error('NaN encountered');
109             end
110         end
111         constantEndValue_phase_tolerance_in_rad = ...
                (pi/180)*constantEndValue_phase_tolerance_in_deg;
112         end
113
114         %% Storage allocation
115         REAL_LOW          = zeros(1,storage_size);
116         IMAG_LOW          = zeros(1,storage_size);
117         MAG_LOW           = zeros(1,storage_size);
118         PHASE_LOW         = zeros(1,storage_size);
119         OMEGA_LOW         = zeros(1,storage_size);
120         DELTA_OMEGA_LOW   = zeros(1,storage_size);
121         DELTA_PHASE_LOW   = zeros(1,storage_size);
122         DELTA_MAGN_LOW    = zeros(1,storage_size);
123
124         REAL_HIGH         = zeros(1,storage_size);
125         IMAG_HIGH         = zeros(1,storage_size);
126         MAG_HIGH          = zeros(1,storage_size);
127         PHASE_HIGH        = zeros(1,storage_size);
128         OMEGA_HIGH        = zeros(1,storage_size);
129         DELTA_OMEGA_HIGH  = zeros(1,storage_size);
130         DELTA_PHASE_HIGH  = zeros(1,storage_size);
131         DELTA_MAGN_HIGH   = zeros(1,storage_size);
132
133         %% Setting some initial values
134         dd1      = (pi/180)*d;
135         dd2      = (pi/180)*(360-d);
136         d1       = dd1;
137         d2       = dd2;
138
139         j_stuck = 0;
140
141         %% Calculate frequency response for first interval, w = [startpoint_w,inf]
142         j                    = 1;
143         w                    = startpoint_w;
144         w_delta              = w_delta_start;
145         stop_flag            = false;
146         check_end_condition  = false;
147         sum_change_in_mag    = 0;
148
149         while ( ((w_interval_given == false) && (stop_flag ~= true)) || ...
150                 ((w_interval_given == true)  && (w < w_stop))          )
151             % 1.) Calculate frequency response value at w
152             [re,im] = calc_freq_resp(w,a,an,b,bn,L);
153
154             % The first two iterations of the loop is calculated without
155             %  i. change in step length for w
156             % ii. stopping condition check
157             if j <= 2
158                 REAL_HIGH(j)         = re;
159                 IMAG_HIGH(j)         = im;
160                 MAG_HIGH(j)          = 10*log10(re^2 + im^2);
161                 PHASE_HIGH(j)        = atan2(im,re);
162                 OMEGA_HIGH(j)        = w;
163                 DELTA_OMEGA_HIGH(j)  = w_delta;
164                 if j == 2
165                     DELTA_PHASE_HIGH(j) = atan2(im - IMAG_HIGH(j-1), re - REAL_HIGH(j-1));
166                     DELTA_MAGN_HIGH(j)  = 10.*log10((re - REAL_HIGH(j-1))^2 + (im - ...
                            IMAG_HIGH(j-1))^2);
167                 end
168                 j = j + 1;
169                 w_prev = w;
170                 w = w_prev + w_delta;
171             else
172                 % 2.) Check phase difference of the last line segment and the line
173                 % drawn between the previous point and the newly calculated
174                 % point. Also checks the increase in magnitude with
175                 % regards to the magnitude value at the current step.
176                 % If the checked values satisfy the given conditions then
```

```
177             % increase the step size for w and save the frequency response
178             % value. If the conditions are not meet, then decrease step
179             % size for w and try a new step point by starting from the top
180             % of the while loop.
181             DELTA_PHASE_HIGH(j) = atan2(im - IMAG_HIGH(j-1), re - REAL_HIGH(j-1));
182             DELTA_MAGN_HIGH(j)  = 10.*log10((re - REAL_HIGH(j-1))^2 + (im - ...
                    IMAG_HIGH(j-1))^2);
183             if ( (DELTA_MAGN_HIGH(j) > 10*log10(re^2 + im^2) - y) ...
184                 || (     abs(DELTA_PHASE_HIGH(j-1)-DELTA_PHASE_HIGH(j)) > d1 ...
185                     &&  abs(DELTA_PHASE_HIGH(j-1)-DELTA_PHASE_HIGH(j)) < d2) ...
186                 || (w == inf) ) ...
187             && (DELTA_OMEGA_HIGH(j-1) > eps(OMEGA_HIGH(j-1)))
188                 w_delta = w_delta/alpha;
189                 w       = w_prev + w_delta;
190
191                 % The iteration process can get stuck, this
192                 % piece of code prevents it, by slowly increasing the
193                 % acceptable range of phase difference values between the
194                 % subsequent line segments when a "stuck" condition is detected.
195                 if w_delta < (1e-6)*w
196                     j_stuck = j;
197                     d1 = (1+dr)*d1;
198                     d2 = (1-dr)*d2;
199                 end
200             else
201                 REAL_HIGH(j)        = re;
202                 IMAG_HIGH(j)        = im;
203                 MAG_HIGH(j)         = 10*log10(re^2 + im^2);
204                 PHASE_HIGH(j)       = atan2(im,re);
205                 OMEGA_HIGH(j)       = w;
206                 DELTA_OMEGA_HIGH(j) = w_delta;
207                 j = j + 1;
208                 check_end_condition = true;
209
210                 w_delta = w_delta*alpha;
211                 w_prev  = w;
212                 w = w_prev + w_delta;
213
214
215                 if j_stuck ~= 0 && j > j_stuck
216                     % Iteration process managed to continue and
217                     % d1 and d2 is reset
218                     d1 = dd1;
219                     d2 = dd2;
220                     j_stuck = 0;
221                 end
222             end
223         end
224
225         % 3.) Check if enough values have been calculated. If so, exit the loop.
226         % If not, continue from the start of the while loop.
227         if (check_end_condition == true) && (w_interval_given == false)
228             if magnitude_at_w_inf == 0 || magnitude_at_w_inf == inf
229                 if angle_inside(lowerPhase_at_wInf,upperPhase_at_wInf,atan2(im,re))
230                     sum_change_in_mag = sum_change_in_mag + (MAG_HIGH(j-1) - ...
                            MAG_HIGH(j-2));
231                     if sign(constPhase_stopLength_in_dB_at_wInf)*sum_change_in_mag >= ...
                            abs(constPhase_stopLength_in_dB_at_wInf)
232                         % Extra check for the existance of lower w value
233                         % which gives a phase value different than the
234                         % termination phase.
235                         [re_stop_test,im_stop_test] = ...
                                calc_freq_resp(extra_stopping_test_w_scale*w,a,an,b,bn,L);
236                         if (re_stop_test < realmax && re_stop_test > -realmax) && ...
                                (im_stop_test < realmax && im_stop_test > -realmax) && ...
                                ~angle_inside(lowerPhase_at_wInf,upperPhase_at_wInf,...
237                                 atan2(im_stop_test,re_stop_test))
238                             sum_change_in_mag = 0;
239                         else
240                             stop_flag = true;
241                         end
```

```
242                     end
243                 else
244                     sum_change_in_mag = 0;
245                 end
246             else
247                 if (    (abs(REAL_HIGH(j-1) - re_at_wInf) < ...
                        constantEndValue_re_tolerance) ...
248                     && (abs(IMAG_HIGH(j-1) - im_at_wInf) < ...
                            constantEndValue_im_tolerance) ...
249                     && (abs(PHASE_HIGH(j-1) - phase_at_w_inf) < ...
                            constantEndValue_phase_tolerance_in_rad) )
250                     stop_flag = true;
251                 end
252             end
253             check_end_condition = false;
254         end
255     end % while loop
256
257     j_high_stop = j-1;
258
259     %% Calc freq resp for w = [0,startpoint_w]
260     j                   = 1;
261     w                   = startpoint_w;
262     w_delta             = w_delta_start;
263     stop_flag           = false;
264     check_end_condition = false;
265     sum_change_in_mag   = 0;
266
267     while ( ((w_interval_given == false) && (stop_flag ~= true)) || ...
268             ((w_interval_given == true)  && (w > w_start))          )
269         % 1.) Calculate frequency response value at w
270         [re,im] = calc_freq_resp(w,a,an,b,bn,L);
271
272         % The first two iterations of the loop is calculated without
273         %  i. change in step length for w
274         % ii. stopping condition check
275         if j <= 2
276             REAL_LOW(j)         = re;
277             IMAG_LOW(j)         = im;
278             MAG_LOW(j)          = 10*log10(re^2 + im^2);
279             PHASE_LOW(j)        = atan2(im,re);
280             OMEGA_LOW(j)        = w;
281             DELTA_OMEGA_LOW(j) = w_delta;
282             if j == 2
283                 DELTA_PHASE_LOW(j) = atan2(im - IMAG_LOW(j-1), re - REAL_LOW(j-1));
284                 DELTA_MAGN_LOW(j)  = 10.*log10((re - REAL_LOW(j-1))^2 + (im - ...
                        IMAG_LOW(j-1))^2);
285             end
286             j = j + 1;
287             w_prev = w;
288             w = w_prev - w_delta;
289         else
290             % 2.) Check phase difference of the last line segment and the line
291             % drawn between the previous point and the newly calculated
292             % point. Also checks the increase in magnitude with
293             % regards to the magnitude value at the current step.
294             % If the checked values satisfy the given conditions then
295             % increase the step size for w and save the frequency response
296             % value. If the conditions are not meet, then decrease step
297             % size for w and try a new step point by starting from the top
298             % of the while loop.
299             DELTA_PHASE_LOW(j) = atan2(im - IMAG_LOW(j-1), re - REAL_LOW(j-1));
300             DELTA_MAGN_LOW(j)  = 10.*log10((re - REAL_LOW(j-1))^2 + (im - ...
                    IMAG_LOW(j-1))^2);
301             if (DELTA_MAGN_LOW(j) > 10*log10(re^2 + im^2) - y ...
302                 || (    abs(DELTA_PHASE_LOW(j-1)-DELTA_PHASE_LOW(j)) > d1 ...
303                     && abs(DELTA_PHASE_LOW(j-1)-DELTA_PHASE_LOW(j)) < d2) ) ...
304             && (DELTA_OMEGA_HIGH(j-1) > eps(OMEGA_HIGH(j-1)))
305                 w_delta = w_delta/alpha;
306                 w       = w_prev - w_delta;
307
```

```
308                 % The iteration process can get stuck, this
309                 % piece of code prevents it, by slowly increasing the
310                 % acceptable range of phase difference values between the
311                 % subsequent line segments when a "stuck" condition is detected.
312                 if w_delta < (1e-6)*w
313                     j_stuck = j;
314                     d1 = (1+dr)*d1;
315                     d2 = (1-dr)*d2;
316                 end
317             else
318                 REAL_LOW(j)         = re;
319                 IMAG_LOW(j)         = im;
320                 MAG_LOW(j)          = 10*log10(re^2 + im^2);
321                 PHASE_LOW(j)        = atan2(im,re);
322                 OMEGA_LOW(j)        = w;
323                 DELTA_OMEGA_LOW(j) = w_delta;
324                 j = j + 1;
325                 check_end_condition = true;
326
327                 w_delta = w_delta*alpha;
328                 w_prev  = w;
329                 w = w_prev - w_delta;
330
331                 %
332                 if j_stuck ~= 0 && j > j_stuck
333                     % Iteration process managed to continue and
334                     % d1 and d2 is reset
335                     d1 = dd1;
336                     d2 = dd2;
337                     j_stuck = 0;
338                 end
339             end
340         end
341
342         while w < 0
343             w_delta = w_delta/alpha;
344             w       = w_prev - w_delta;
345         end
346
347         % 3.) Check if enough values have been calculated. If so, exit the loop.
348         % If not, continue from the start of the while loop.
349         if (check_end_condition == true) && (w_interval_given == false)
350             if magnitude_at_w_zero == 0 || magnitude_at_w_zero == inf
351                 if angle_inside(lowerPhase_at_wZero,upperPhase_at_wZero,atan2(im,re))
352                     sum_change_in_mag = sum_change_in_mag + (MAG_LOW(j-1) - MAG_LOW(j-2));
353                     if sign(constPhase_stopLength_in_dB_at_wZero)*sum_change_in_mag >= ...
                            abs(constPhase_stopLength_in_dB_at_wZero)
354                         % Extra check for the existance of lower w value
355                         % which gives a phase different than the
356                         % termination phase
357                         [re_stop_test,im_stop_test] = ...
                                calc_freq_resp(1/extra_stopping_test_w_scale*w,a,an,b,bn,L);
358                         if (re_stop_test < realmax && re_stop_test > -realmax) && ...
                                (im_stop_test < realmax && im_stop_test > -realmax) && ...
                                ~angle_inside(lowerPhase_at_wZero,upperPhase_at_wZero,...
359                                 atan2(im_stop_test,re_stop_test))
360                             sum_change_in_mag = 0;
361                         else
362                             stop_flag = true;
363                         end
364                     end
365                 else
366                     sum_change_in_mag = 0;
367                 end
368             else
369                 if (    (abs(REAL_LOW(j-1) - re_at_wZero) < ...
                            constantEndValue_re_tolerance) ...
370                      && (abs(IMAG_LOW(j-1) - im_at_wZero) < ...
                             constantEndValue_im_tolerance) ...
371                      && (abs(PHASE_LOW(j-1) - phase_at_w_zero) < ...
                             constantEndValue_phase_tolerance_in_rad) )
```

```
372                     stop_flag = true;
373                 end
374             end
375             check_end_condition = false;
376         end
377     end % while loop
378
379     j_low_stop = j-1;
380
381     %% Add more points to include important approching zero or inf dynamics
382     if w_interval_given == false
383         min_mag = min([MAG_LOW(j_low_stop:-1:1), MAG_HIGH(1:j_high_stop)]);
384         max_mag = max([MAG_LOW(j_low_stop:-1:1), MAG_HIGH(1:j_high_stop)]);
385
386         % Calculate the wished lower value for zero approching
387         if min_mag == MAG_LOW(j_low_stop) || min_mag == MAG_HIGH(j_high_stop)
388             wished_lower_mag_dB = min_mag;
389         else
390             wished_lower_mag_dB = min_mag - 10;
391         end
392         wished_lower_mag = 10^(wished_lower_mag_dB/20);
393
394         % Calculate the wished upper value for inf approaching
395         if max_mag == MAG_LOW(j_low_stop) || max_mag == MAG_HIGH(j_high_stop)
396             wished_upper_mag_dB = max_mag;
397         else
398             wished_upper_mag_dB = max_mag + 10;
399         end
400         wished_upper_mag = 10^(wished_upper_mag_dB/20);
401
402         % Add extra point at w = 0
403         switch magnitude_at_w_zero
404             case 0
405                 if min_mag ~= MAG_LOW(j_low_stop)
406                     j = j_low_stop + 1;
407                     REAL_LOW(j)  = wished_lower_mag*cos(phase_at_w_zero);
408                     IMAG_LOW(j)  = wished_lower_mag*sin(phase_at_w_zero);
409                     MAG_LOW(j)   = wished_lower_mag_dB;
410                     PHASE_LOW(j) = phase_at_w_zero;
411                     OMEGA_LOW(j) = ...
                            (abs(a(an_li)/b(bn_li))*wished_lower_mag)^(1/(bn(bn_li) - ...
                            an(an_li)));
412                     j_low_stop = j;
413                 end
414             case inf
415                 if max_mag ~= MAG_LOW(j_low_stop)
416                     j = j_low_stop + 1;
417                     REAL_LOW(j)  = wished_upper_mag*cos(phase_at_w_zero);
418                     IMAG_LOW(j)  = wished_upper_mag*sin(phase_at_w_zero);
419                     MAG_LOW(j)   = wished_upper_mag_dB;
420                     PHASE_LOW(j) = phase_at_w_zero;
421                     OMEGA_LOW(j) = ...
                            (abs(a(an_li)/b(bn_li))*wished_upper_mag)^(1/(bn(bn_li) - ...
                            an(an_li)));
422                     j_low_stop = j;
423                 end
424             otherwise
425                 % Do Nothing, the frequency response is good enough as it is
426         end
427
428         % Add extra point at w = inf
429         switch magnitude_at_w_inf
430             case 0
431                 if min_mag ~= MAG_HIGH(j_high_stop)
432                     j = j_high_stop + 1;
433                     REAL_HIGH(j)  = wished_lower_mag*cos(phase_at_w_inf);
434                     IMAG_HIGH(j)  = wished_lower_mag*sin(phase_at_w_inf);
435                     MAG_HIGH(j)   = wished_lower_mag_dB;
436                     PHASE_HIGH(j) = phase_at_w_inf;
437                     OMEGA_HIGH(j) = ...
                            (abs(a(an_hi)/b(bn_hi))*wished_lower_mag)^(1/(bn(bn_hi) - ...
```

```
                        an(an_hi)));
438                     j_high_stop = j;
439                 end
440             case inf
441                 if max_mag ~= MAG_HIGH(j_high_stop)
442                     j = j_high_stop + 1;
443                     REAL_HIGH(j)  = wished_upper_mag*cos(phase_at_w_inf);
444                     IMAG_HIGH(j)  = wished_upper_mag*sin(phase_at_w_inf);
445                     MAG_HIGH(j)   = wished_upper_mag_dB;
446                     PHASE_HIGH(j) = phase_at_w_inf;
447                     OMEGA_HIGH(j) = ...
                            (abs(a(an_hi)/b(bn_hi))*wished_upper_mag)^(1/(bn(bn_hi) - ...
                            an(an_hi)));
448                     j_high_stop = j;
449                 end
450             otherwise
451                 % Do Nothing, the frequency response is good enough as it is
452         end
453     end
454
455     %% Add w = 0 and w = inf points
456     if w_interval_given == false
457         % Add point at w = 0
458         j = j_low_stop + 1;
459         REAL_LOW(j)  = magnitude_at_w_zero*cos(phase_at_w_zero);
460         IMAG_LOW(j)  = magnitude_at_w_zero*sin(phase_at_w_zero);
461         MAG_LOW(j)   = 20*log10(magnitude_at_w_zero);
462         PHASE_LOW(j) = phase_at_w_zero;
463         OMEGA_LOW(j) = 0;
464         j_low_stop = j;
465
466         % Add point at w = inf
467         j = j_high_stop + 1;
468         REAL_HIGH(j)  = magnitude_at_w_inf*cos(phase_at_w_inf);
469         IMAG_HIGH(j)  = magnitude_at_w_inf*sin(phase_at_w_inf);
470         MAG_HIGH(j)   = 20*log10(magnitude_at_w_inf);
471         PHASE_HIGH(j) = phase_at_w_inf;
472         OMEGA_HIGH(j) = inf;
473         j_high_stop = j;
474     end
475
476     %% Return the calculated frequency response
477     re_out    =           [ REAL_LOW(j_low_stop:-1:1),  REAL_HIGH(1:j_high_stop)];
478     im_out    =           [ IMAG_LOW(j_low_stop:-1:1),  IMAG_HIGH(1:j_high_stop)];
479     mag_out   =           [  MAG_LOW(j_low_stop:-1:1),   MAG_HIGH(1:j_high_stop)];
480     phase_out = (180/pi).*[PHASE_LOW(j_low_stop:-1:1), PHASE_HIGH(1:j_high_stop)];
481     w_out     =           [OMEGA_LOW(j_low_stop:-1:1), OMEGA_HIGH(1:j_high_stop)];
482
483 end % astep_fotf_freqresp()
484
485
486 %% Helper functions
487 function [re, im] = calc_freq_resp(w, a, na, b, nb, L)
488     freq_resp_num = b*((1i*w).^nb.');
489     freq_resp_den = a*((1i*w).^na.');
490     freq_resp = freq_resp_num/freq_resp_den;
491     if L > 0
492         freq_resp = freq_resp.*exp(-L*1i*w);
493     end
494
495     re = real(freq_resp);
496     im = imag(freq_resp);
497 end
498
499 function isInside = angle_inside(lower_angle,upper_angle,test_angle)
500     upper_angle = upper_angle - lower_angle;
501     if upper_angle < 0
502         upper_angle = upper_angle + 360;
503     end
504     test_angle = test_angle - lower_angle;
505     if test_angle < 0
```

```
506             test_angle = test_angle + 360;
507         end
508         isInside = (test_angle < upper_angle);
509     end
```

Appendix D

Experimental Logarithmic-Amplitude Polar Diagram for FOS

Listing D.1: `nyqlog_fotf.m`

```
function nyqlog_fotf(G,w1,w2)
    %% Input guard
    if isa(G, 'tf')
        G = fotf(G);
    elseif ~isa(G, 'fotf')
        error(['Argument G has unsupported type ', class(G), '.']);
    end

    switch nargin
        case 1
            [re, im, mag_dB, phase_deg, w] = astep_fotf_freqresp(G);
            if abs(mag_dB(1)) == inf
                re        = re(2:end);
                im        = im(2:end);
                mag_dB    = mag_dB(2:end);
                phase_deg = phase_deg(2:end);
                w         = w(2:end);
            end
            if abs(mag_dB(end)) == inf
                re        = re(1:end-1);
                im        = im(1:end-1);
                mag_dB    = mag_dB(1:end-1);
                phase_deg = phase_deg(1:end-1);
                w         = w(1:end-1);
            end
            phase_rad = (pi/180)*phase_deg;
        case 2
            H = bode(G,w1);
            re        = real(squeeze(H.ResponseData))';
            im        = imag(squeeze(H.ResponseData))';
            mag_dB    = 10.*log10(re.^2 + im.^2);
            phase_rad = atan2(im,re);
            w         = w1;
        case 3
            [re, im, mag_dB, phase_deg, w] = astep_fotf_freqresp(G,w1,w2);
            if abs(mag_dB(1)) == inf
                re        = re(2:end);
                im        = im(2:end);
                mag_dB    = mag_dB(2:end);
                phase_deg = phase_deg(2:end);
                w         = w(2:end);
```

```
            end
            if abs(mag_dB(end)) == inf
                re        = re(1:end-1);
                im        = im(1:end-1);
                mag_dB    = mag_dB(1:end-1);
                phase_deg = phase_deg(1:end-1);
                w         = w(1:end-1);
            end
            phase_rad = (pi/180)*phase_deg;
        otherwise
            error('Number of arguments not supported');
    end

    plot_logarithmic_nyquist(G, re, im, mag_dB, phase_rad, w);
end


%% Helper functions
function plot_logarithmic_nyquist(G,re,im,mag_dB,phase,w)
    for j = 1:length(mag_dB)
        if mag_dB(j) == inf
            if j == 1 && length(mag_dB) > 1
                mag_dB(j) = mag_dB(j+1);
            elseif j == length(mag_dB) && length(mag_dB) > 1
                mag_dB(j) = mag_dB(j-1);
            else
                mag_dB(j) = (mag_dB(j+1) + mag_dB(j-1))/2;
            end
        end
    end

    %% Detect/find resonace peaks in data
    w_resonance_intervals_start = [];
    w_resonance_intervals_end   = [];

    numeric_precission_threshold_gain = 10^3;
    angle_threshold_deg = 10;

    on_resonance_peak = false;
    for j = 1:length(w)-1
        if on_resonance_peak == false && ...
        w(j+1) - w(j) <= numeric_precission_threshold_gain*eps(w(j))
            w_resonance_intervals_start = [w_resonance_intervals_start j];
            on_resonance_peak = true;
        elseif on_resonance_peak == true && ...
        w(j+1) - w(j) > numeric_precission_threshold_gain*eps(w(j))
            on_resonance_peak = false;
            w_resonance_intervals_end = [w_resonance_intervals_end j];
        end
    end

    w_dim_diff_res_int = length(w_resonance_intervals_start) - ...
        length(w_resonance_intervals_end);
    if w_dim_diff_res_int > 0
        w_resonance_intervals_end = [w_resonance_intervals_end, length(w)];
    end

    w_resonance_intervals_start = w_resonance_intervals_start(...
        find(w(w_resonance_intervals_start) ~= 1)); %#ok<FNDSB>
    w_resonance_intervals_end   = w_resonance_intervals_end(...
        find(w(w_resonance_intervals_end) ~= 1)); %#ok<FNDSB>

    w_res_int_start = [];
    w_res_int_end   = [];
    for j = 1:length(w_resonance_intervals_start)
        phs_start = phase(w_resonance_intervals_start(j));
        phs_end   = phase(w_resonance_intervals_end(j));
        phs_diff = distance_between_angles(phs_start, phs_end);
        if phs_diff > pi - (pi/180)*angle_threshold_deg
            w_res_int_start = [w_res_int_start, w_resonance_intervals_start(j)];
            w_res_int_end = [w_res_int_end, w_resonance_intervals_end(j)];
```

```
112         end
113     end
114
115     %% Remove areas around resonance peaks with low numerical precission
116     for j = 1:length(w_res_int_start)
117         idx_start = w_res_int_start(j);
118         idx_end   = w_res_int_end(j);
119
120         re      = [     re(1:idx_start),      re(idx_end:end)];
121         im      = [     im(1:idx_start),      im(idx_end:end)];
122         mag_dB = [mag_dB(1:idx_start), mag_dB(idx_end:end)];
123         phase   = [ phase(1:idx_start),   phase(idx_end:end)];
124         w       = [      w(1:idx_start),       w(idx_end:end)];
125
126         % Change indicies for remaining intervals in w_res_int_(start/end)
127         num_removed_elements = idx_end - idx_start - 1;
128         if j ~= length(w_res_int_start)
129             w_res_int_start(j+1:end) = w_res_int_start(j+1:end) - num_removed_elements;
130         end
131         w_res_int_end(j:end)   = w_res_int_end(j:end) - num_removed_elements;
132     end
133
134     %% Find size of background grid
135     mag_db_min = min_ignInf(mag_dB);
136     mag_db_max = max_ignInf(mag_dB);
137
138     mag_lower_grid_line = 10*floor(mag_db_min/10);
139     mag_upper_grid_line = 10*ceil(mag_db_max/10);
140     if mag_upper_grid_line < 0
141         mag_upper_grid_line = 0;
142     end
143     if mag_lower_grid_line > 0
144         mag_lower_grid_line = 0;
145     end
146
147     % Calculate value at w = 0
148     [a,an,b,bn,~] = fotfdata(G);
149     [~, an_li] = min(an);
150     [~, bn_li] = min(bn);
151     [~, an_hi] = max(an);
152     [~, bn_hi] = max(bn);
153
154     w_zero = 0;
155     mag_at_w_zero   = abs(b(bn_li)/a(an_li))*w_zero^(bn(bn_li) - an(an_li));
156     phase_at_w_zero = wrapToPi(pi*(sign(b(bn_li)) ~= sign(a(an_li))) ...
157                                    + pi/2*(bn(bn_li) - an(an_li)));
158     % Calculate value at w = inf
159     w_inf  = inf;
160     mag_at_w_inf   = abs(b(bn_hi)/a(an_hi))*w_inf^(bn(bn_hi) - an(an_hi));
161     phase_at_w_inf = wrapToPi(pi*(sign(b(bn_hi)) ~= sign(a(an_hi))) ...
162                                   + pi/2*(bn(bn_hi) - an(an_hi)));
163
164     approx_number_of_circles_in_grid = 5;
165     approx_distance_between_circles = (mag_upper_grid_line - ...
        mag_lower_grid_line)/approx_number_of_circles_in_grid;
166
167
168     mag_length_between_lines = 10*floor(approx_distance_between_circles/10);
169     %mag_length_between_lines = 20;
170
171     grid_lines_at_dB = mag_lower_grid_line:mag_length_between_lines:mag_upper_grid_line;
172     if grid_lines_at_dB(1:end) ~= 0
173         grid_lines_at_dB = [grid_lines_at_dB, 0];
174         grid_lines_at_dB = sort(grid_lines_at_dB);
175     end
176
177     center_space_dB = 30;
178
179     min_circ_mag = min(grid_lines_at_dB);
180     nyquist_curve_offset_dB = center_space_dB - min_circ_mag;
181
```

```
182     plot_mag_dB = mag_dB + nyquist_curve_offset_dB;
183
184     re_dB = plot_mag_dB.*cos(phase);
185     im_dB = plot_mag_dB.*sin(phase);
186
187     %% semi-circles at inf should all be placed at different hights for better visual ...
            impression
188     base_inf_offset_dB = 20;
189     step_inf_offset_dB = 10;
190     inf_offset_dB = base_inf_offset_dB;
191
192     %% Add wrap around at inf magnitude for resonance peaks
193     for j = 1:length(w_res_int_start)
194         idx_start = w_res_int_start(j);
195         idx_end   = w_res_int_end(j);
196
197         %Decide on rotation direction with symbolic computation
198         syms w_start w_end;
199         N_w = 100;
200         w_sym_start = [w_start, w_start + (w_end - w_start)/N_w];
201         w_sym_end   = [w_end + (w_start - w_end)/N_w, w_end];
202
203         w_sym_start = subs(w_sym_start, [w_start,w_end], [w(idx_start),w(idx_end)]);
204         w_sym_end   = subs(w_sym_end,   [w_start,w_end], [w(idx_start),w(idx_end)]);
205         [~, ~, ~, G_phase] = calc_freq_resp_symbolic([w_sym_start, w_sym_end], ...
                a,an,b,bn,0);
206
207         phase_start_delta = G_phase(2) - G_phase(1);
208         phase_end_delta   = G_phase(4) - G_phase(3);
209         if phase_start_delta >= 0 && phase_end_delta >= 0
210             dir = 1; % CCW
211             draw_half_circle = true;
212         elseif phase_start_delta <= 0 && phase_end_delta <= 0
213             dir  = -1; % CW
214             draw_half_circle = true;
215         else
216             warning(['Direction of rotation for resonance peak ', ...
217                      'at w = ', num2str(w(idx_start)), ...
218                      ' rad/s can not be decided!', ' Drawing of half '...
219                      'circle at inf have been suppressed for this peak.']);
220             draw_half_circle = false;
221         end
222
223         if draw_half_circle == true
224             from_angle = phase(idx_start);
225             to_angle   = phase(idx_end);
226             angle_step_deg = 1;
227
228             if dir == 1 && to_angle > from_angle
229                 angle_step = (pi/180)*angle_step_deg;
230                 angles_semi_circle = [from_angle:angle_step:to_angle, to_angle];
231             elseif dir == 1 && to_angle < from_angle
232                 angle_step =  -(pi/180)*angle_step_deg;
233                 angles_semi_circle = [from_angle:angle_step:to_angle, to_angle];
234             elseif dir == -1 && to_angle > from_angle
235                 angle_step = -(pi/180)*angle_step_deg;
236                 angles_semi_circle = [from_angle:angle_step:(to_angle-2*pi), ...
                        (to_angle-2*pi)];
237             elseif dir == -1 && to_angle < from_angle
238                 angle_step = -(pi/180)*angle_step_deg;
239                 angles_semi_circle = [from_angle:angle_step:to_angle, to_angle];
240             else
241                 error('Can not decide on rotation');
242             end
243
244             mag_semi_circle = max_ignInf(plot_mag_dB) + inf_offset_dB;
245             inf_offset_dB = inf_offset_dB + step_inf_offset_dB;
246             re_half_circle = mag_semi_circle.*cos(angles_semi_circle);
247             im_half_circle = mag_semi_circle.*sin(angles_semi_circle);
248
249             re_dB  = [ re_dB(1:idx_start), re_half_circle,  re_dB(idx_end:end)];
```

```
250             im_dB  = [ im_dB(1:idx_start), im_half_circle,  im_dB(idx_end:end)];
251
252             w_half_circle = ((w(idx_end)+w(idx_start))/2)*ones(size(angles_semi_circle));
253             uncalculateable_values = NaN(size(angles_semi_circle));
254
255             re     = [    re(1:idx_start), uncalculateable_values,     re(idx_end:end)];
256             im     = [    im(1:idx_start), uncalculateable_values,     im(idx_end:end)];
257             mag_dB = [mag_dB(1:idx_start), uncalculateable_values, mag_dB(idx_end:end)];
258             phase  = [ phase(1:idx_start), uncalculateable_values,  phase(idx_end:end)];
259             w      = [     w(1:idx_start),           w_half_circle,      w(idx_end:end)];
260
261             inserted_vector_length = length(angles_semi_circle);
262         else
263             mag_peak_val = max_ignInf(plot_mag_dB) + inf_offset_dB;
264             inf_offset_dB = inf_offset_dB + step_inf_offset_dB;
265             re_peak_fix_val_start = mag_peak_val.*cos(phase(idx_start));
266             im_peak_fix_val_start = mag_peak_val.*sin(phase(idx_start));
267             re_peak_fix_val_end   = mag_peak_val.*cos(phase(idx_end));
268             im_peak_fix_val_end   = mag_peak_val.*sin(phase(idx_end));
269
270             re_peak_subst = [re_peak_fix_val_start, inf, re_peak_fix_val_end];
271             im_peak_subst = [im_peak_fix_val_start, inf, im_peak_fix_val_end];
272
273             re_dB  = [ re_dB(1:idx_start-1), re_peak_subst,  re_dB(idx_end+1:end)];
274             im_dB  = [ im_dB(1:idx_start-1), im_peak_subst,  im_dB(idx_end+1:end)];
275
276             uncalculateable_values = NaN(size(re_peak_subst));
277             w_peak_subst = w(idx_start).*ones(size(re_peak_subst));
278
279             re     = [    re(1:idx_start), uncalculateable_values,     re(idx_end:end)];
280             im     = [    im(1:idx_start), uncalculateable_values,     im(idx_end:end)];
281             mag_dB = [mag_dB(1:idx_start), uncalculateable_values, mag_dB(idx_end:end)];
282             phase  = [ phase(1:idx_start), uncalculateable_values,  phase(idx_end:end)];
283             w      = [     w(1:idx_start),           w_peak_subst,      w(idx_end:end)];
284
285             inserted_vector_length = length(re_peak_subst);
286         end
287
288         % Change indicies for remaining intervals in w_res_int_(start/end)
289         if j ~= length(w_res_int_start)
290             w_res_int_start(j+1:end) = w_res_int_start(j+1:end) + inserted_vector_length;
291         end
292         w_res_int_end(j:end) = w_res_int_end(j:end) + inserted_vector_length;
293     end
294
295     %% Add point in zero or quarter circle at inf
296     if mag_at_w_zero == 0
297         re_dB  = [0,      re_dB];
298         im_dB  = [0,      im_dB];
299
300         w      = [0,       w];
301         re     = [0,      re];
302         im     = [0,      im];
303         phase  = [NaN,  phase];
304         mag_dB = [-inf, mag_dB];
305
306     elseif mag_at_w_zero == inf
307         if sign(b(bn_li)/a(an_li)) == 1
308             from_angle = 0;
309         else
310             from_angle = pi;
311         end
312         to_angle = phase_at_w_zero;
313         angle_step_deg = 1;
314         if from_angle <= to_angle
315             angle_step = (pi/180)*angle_step_deg;
316         else
317             angle_step = -(pi/180)*angle_step_deg;
318         end
319         angles_quarter_circle = from_angle:angle_step:to_angle;
320         mag_quarter_circle = max_ignInf(plot_mag_dB) + inf_offset_dB;
```

```
321         inf_offset_dB = inf_offset_dB + step_inf_offset_dB;
322         re_quarter_circle = mag_quarter_circle.*cos(angles_quarter_circle);
323         im_quarter_circle = mag_quarter_circle.*sin(angles_quarter_circle);
324         re_dB = [re_quarter_circle, re_dB];
325         im_dB = [im_quarter_circle, im_dB];
326
327         re     = [inf.*cos(angles_quarter_circle),       re];
328         im     = [inf.*sin(angles_quarter_circle),       im];
329         mag_dB = [inf(size(angles_quarter_circle)), mag_dB];
330         phase  = [NaN(size(angles_quarter_circle)),  phase];
331         w      = [zeros(size(angles_quarter_circle)),     w];
332     end
333
334     if mag_at_w_inf == 0
335         re_dB  = [re_dB,      0];
336         im_dB  = [im_dB,      0];
337
338         w      = [w,        inf];
339         re     = [re,         0];
340         im     = [im,         0];
341         phase  = [phase,    NaN];
342         mag_dB = [mag_dB, -inf];
343
344     elseif mag_at_w_inf == inf
345         if sign(b(bn_hi)/a(an_hi)) == 1
346             to_angle = 0;
347         else
348             to_angle = pi;
349         end
350         from_angle = phase_at_w_inf;
351         angle_step_deg = 1;
352         if from_angle <= to_angle
353             angle_step = (pi/180)*angle_step_deg;
354         else
355             angle_step = -(pi/180)*angle_step_deg;
356         end
357         angles_quarter_circle = from_angle:angle_step:to_angle;
358         mag_quarter_circle = max_ignInf(plot_mag_dB) + inf_offset_dB;
359         inf_offset_dB = inf_offset_dB + step_inf_offset_dB;
360         re_quarter_circle = mag_quarter_circle.*cos(angles_quarter_circle);
361         im_quarter_circle = mag_quarter_circle.*sin(angles_quarter_circle);
362         re_dB  = [re_dB, re_quarter_circle];
363         im_dB  = [im_dB, im_quarter_circle];
364
365         re     = [re,       inf.*cos(angles_quarter_circle)];
366         im     = [im,       inf.*sin(angles_quarter_circle)];
367         mag_dB = [mag_dB, inf(size(angles_quarter_circle))];
368         phase  = [phase,  NaN(size(angles_quarter_circle))];
369         w      = [w,       inf(size(angles_quarter_circle))];
370     end
371
372
373     %% Plot Nyquist curve
374     nyquist_curve_pw_handle = plot(re_dB,  im_dB, ...
375                         'Color',     [0, 0.447, 0.741], ...
376                         'LineStyle', '-', ...
377                         'LineWidth', 1.5);
378     hold on;
379
380     nyquist_curve_nw_handle = plot(re_dB, -im_dB, ...
381                                    'Color',     [0, 0, 0], ...
382                                    'LineStyle', ':', ...
383                                    'LineWidth', 1.5);
384     hold off; axis off; axis equal;
385
386     set(nyquist_curve_pw_handle, 'UserData', 'positive_nyquist_curve');
387     set(nyquist_curve_nw_handle, 'UserData', 'negative_nyquist_curve');
388
389     plot_grid(grid_lines_at_dB, center_space_dB);
390
391     fig_handle = gcf;
```

```
392     set(fig_handle, 'Color', [0.98, 0.98, 0.98]);
393
394     %% Add Data Cursor calculation function (Enables readout of values on the curve)
395     dcm_obj = datacursormode(fig_handle);
396     set(dcm_obj, 'UpdateFcn', {@fotf_nyqlog_dcm_update_fcn,...
397                                w, mag_dB, phase, re, im});
398     set(dcm_obj, 'DisplayStyle', 'datatip');
399     set(dcm_obj, 'Enable', 'on');
400
401     title({'Logarithmic Nyquist Diagram'; ' '});
402 end
403
404 function plot_grid(circ_num_and_mag, center_space)
405     min_circ_mag = min(circ_num_and_mag);
406     print_circ_num_and_mag = circ_num_and_mag - min_circ_mag + center_space;
407
408     % Options
409     circle_resolution                   = 1024;
410     num_straight_lines                  = 8;
411     start_offset_degree_straight_lines = 0;
412     magnitude_text_angle                = -22.5;
413     magnitude_text_distance_from_line  = 1;
414     phase_text_distance_from_circle    = 4;
415     text_FontSize                       = 8;
416
417     hold on;
418     % Draw circles at given magnitudes
419     omega = 0:2*pi/circle_resolution:2*pi;
420     for j = 1:length(print_circ_num_and_mag)
421         x = print_circ_num_and_mag(j).*cos(omega);
422         y = print_circ_num_and_mag(j).*sin(omega);
423         circle_handle = plot(x,y, 'r-.');
424         set(circle_handle, 'PickableParts', 'none');
425         if (circ_num_and_mag(j) == 0)
426             set(circle_handle, 'LineStyle', '-');
427             set(circle_handle, 'LineWidth', 1.0);
428         end
429     end
430
431     % Draw straight lines streaching outwards from the origin
432     line_angles = start_offset_degree_straight_lines ...
433                 + ((0:num_straight_lines-1)./num_straight_lines).*360;
434     mag_lower  = min(print_circ_num_and_mag);
435     mag_higher = max(print_circ_num_and_mag);
436     for j = 1:num_straight_lines
437         x = [mag_lower, mag_higher].*cos((pi/180)*line_angles(j));
438         y = [mag_lower, mag_higher].*sin((pi/180)*line_angles(j));
439         line_handle = plot(x,y, 'r-.');
440         set(line_handle, 'PickableParts', 'none');
441     end
442
443     % Draw Magnitude labels for grid
444     for j = 1:length(circ_num_and_mag)
445         x = (print_circ_num_and_mag(j)+magnitude_text_distance_from_line)...
446             .*cos((pi/180)*magnitude_text_angle);
447         y = (print_circ_num_and_mag(j)+magnitude_text_distance_from_line)...
448             .*sin((pi/180)*magnitude_text_angle);
449         txt = [num2str(circ_num_and_mag(j)), ' dB'];
450         text(x, y, txt, 'FontSize', text_FontSize);
451     end
452
453     % Draw Phase labels for grid
454     degree_symbol = char(176);
455     for j = 1:num_straight_lines
456         angle = (pi/180)*line_angles(j);
457         x = (mag_higher+phase_text_distance_from_circle).*cos(angle);
458         y = (mag_higher+phase_text_distance_from_circle).*sin(angle);
459         txt = [num2str(line_angles(j)), degree_symbol];
460         text_handle = text(x, y, txt, 'FontSize', text_FontSize);
461         if 0.05 < cos(angle)
462             set(text_handle, 'HorizontalAlignment', 'left');
```

```
463             elseif cos(angle) < -0.05
464                 set(text_handle, 'HorizontalAlignment', 'right');
465             else
466                 set(text_handle, 'HorizontalAlignment', 'center');
467             end
468         end
469
470         % Draw -1 point (mag=0dB and phase = -180)
471         x = (0+ center_space - min_circ_mag).*cos(pi);
472         y = (0+ center_space - min_circ_mag).*sin(pi);
473         minus_one_point_handle = plot(x,y, 'Color',      [0, 0, 0], ...
474                     'LineStyle', 'none', ...
475                     'Marker',    'o', ...
476                     'LineWidth', 1.5);
477         set(minus_one_point_handle, 'PickableParts', 'none');
478         hold off;
479     end
480
481     function max_val = max_ignInf(val)
482         max_val = -inf;
483         for j = 1:length(val)
484             if val(j) > max_val && val(j)~= inf
485                 max_val = val(j);
486             end
487         end
488     end
489
490     function min_val = min_ignInf(val)
491         min_val = inf;
492         for j = 1:length(val)
493             if val(j) < min_val && val(j)~= -inf
494                 min_val = val(j);
495             end
496         end
497     end
498
499     function diff = distance_between_angles(angle1, angle2)
500         psi = rem(abs(angle1 - angle2),2*pi);
501         if psi > pi
502             diff = 2*pi - psi;
503         else
504             diff = psi;
505         end
506     end
507
508     function [re, im, mag, phase] = calc_freq_resp_symbolic(w, a, na, b, nb, L)
509         syms s;
510         freq_resp_num = b*(s.^nb.');
511         freq_resp_den = a*(s.^na.');
512         freq_resp = freq_resp_num/freq_resp_den;
513         if L > 0
514             freq_resp = freq_resp.*exp(-L*s);
515         end
516
517         freq_resp = subs(freq_resp, s, 1i.*w);
518
519         phase = double(angle(freq_resp));
520         mag   = double(abs(freq_resp));
521         re    = double(real(freq_resp));
522         im    = double(imag(freq_resp));
523     end
```

Listing D.2: fotf_nyqlog_dcm_update_fcn.m

```
1   function txt = fotf_nyqlog_dcm_update_fcn(empt, event_obj, w, mag_dB, phase, real, imag)
2       idx = get(event_obj, 'DataIndex');
3       line_type = get(event_obj.Target, 'UserData');
4
5       degree_symbol = char(176);
```

```
6
7       switch line_type
8           case 'positive_nyquist_curve'
9               txt = {['Real: ', num2str(real(idx), '%0.4g')],...
10                     ['Imag: ', num2str(imag(idx), '%0.4g')],...
11                     ['Magnitude: ', num2str(mag_dB(idx)), ' dB'],...
12                     ['Phase: ',      num2str((180/pi)*phase(idx)), degree_symbol],...
13                     ['Frequency: ', num2str(w(idx), '%0.4g'), ' rad/s']};
14          case 'negative_nyquist_curve'
15              txt = {['Real: ', num2str(real(idx), '%0.4g')],...
16                     ['Imag: ', num2str(-imag(idx), '%0.4g')],...
17                     ['Magnitude: ', num2str(mag_dB(idx)), ' dB'],...
18                     ['Phase: ',      num2str(-(180/pi)*phase(idx)), degree_symbol],...
19                     ['Frequency: ', num2str(-w(idx), '%0.4g'), ' rad/s']};
20          otherwise
21              txt = {};
22      end
23  end
```

Appendix E

Custom Script for AFM Image Plotting

Listing E.1: `print_AFM_image.m`

```
function print_AFM_image(in1, in2, in3, in4, in5, in6)
% print_AFM_image(data_path) - Create AFM image from special data structure by looking ...
    at x
% and y values of a sample and then assign the associated z value to the
% pixel corresponding to the x and y position through a binary search like
% greater than.
%
% print_AFM_image(data_path, save_images) - Create AFM image and also saves
% the figures to file as .fig and .eps if save_images == true.
%
% print_AFM_imega(data_path, save_images, scan_frequency) - By supplying
% knowledge of the scan frequency, two additional graphs showing zoomed in
% version of the X tracking and Y tracking graphs is printed. They are also
% saved to file in the event that save_images == true.

    %% Process input
    switch nargin
        case {1,2,3}
            data_file_name  = in1;
            vars_to_find    = {'X', 'Y'};
            found_scan_data = extract_data_from_file(data_file_name, vars_to_find);
            X     = found_scan_data{2}(2).Data;
            Y     = found_scan_data{2}(4).Data;
            Z     = found_scan_data{2}(1).Data;
            T     = found_scan_data{1}.Data;
            X_ref = found_scan_data{2}(6).Data;
            Y_ref = found_scan_data{2}(7).Data;
        case 4
            X     = in1;
            Y     = in2;
            Z     = in3;
            T     = in4;
            X_ref = in5;
            Y_ref = in6;
        otherwise
            error('Wrong number of arguments!');
    end

    if nargin == 2 || nargin == 3
        if in2 == true
            save_imag = true;
        else
            save_imag = false;
        end
```

```
44      else
45          save_imag = false;
46      end
47
48      %% Configurations
49      num_pixels_x = 256;
50      num_pixels_y = 256;
51
52      %% Plot AFM image
53      start_idx = 1;
54      end_idx   = length(T);
55      X_ref_min = min(X_ref(start_idx:end_idx));
56      X_ref_max = max(X_ref(start_idx:end_idx));
57      Y_ref_min = min(Y_ref(start_idx:end_idx));
58      Y_ref_max = max(Y_ref(start_idx:end_idx));
59
60      nx = num_pixels_x;
61      ny = num_pixels_y;
62
63      image_mtx = zeros(nx,ny);
64      cnt_mtx   = zeros(nx,ny);
65
66      % Find which of the Z values that should color which pixels based on X
67      % and Y values. This approach to imaging uses a binary search'ish
68      % greater than approach.
69      for j = start_idx:end_idx
70          % Find pixel position
71          X_idx = binary_greater_than(X(j), X_ref_min, X_ref_max, nx);
72          Y_idx = binary_greater_than(Y(j), Y_ref_min, Y_ref_max, ny);
73
74          % Points falling outside the image is added to the edges of the
75          % image
76          if X_idx == 0
77              X_idx = 1;
78          end
79          if Y_idx == 0
80              Y_idx = 1;
81          end
82
83          if X_idx == nx+1
84              X_idx = nx;
85          end
86          if Y_idx == ny+1
87              Y_idx = ny;
88          end
89
90          % Add Z value to image
91          image_mtx(X_idx,Y_idx) = image_mtx(X_idx,Y_idx) + Z(j);
92          cnt_mtx(X_idx,Y_idx)   = cnt_mtx(X_idx,Y_idx)   + 1;
93      end
94
95      % Take mean value of each pixel. Pixels that have not been assigned a
96      % single Z value is set to NaN to prevent imagesc to include zero in
97      % the value/color range.
98      mean_image_mtx = zeros(size(image_mtx));
99      for j = 1:size(image_mtx,1)
100         for k = 1:size(image_mtx,2)
101             if cnt_mtx(j,k) ~= 0
102                 mean_image_mtx(j,k) = image_mtx(j,k)./cnt_mtx(j,k);
103             else
104                 mean_image_mtx(j,k) = NaN;
105             end
106         end
107     end
108
109     % Detrend slope of image
110     %mean_image_mtx = detrend2(mean_image_mtx,[2 2], [0.1,99.9]);
111
112     % Create image
113     fig1 = figure;
114     ax1  = axes(fig1);
```

```
115
116     im=imagesc(mean_image_mtx');
117     im.AlphaData = ~(isnan(mean_image_mtx'));
118     set(ax1, 'ydir', 'normal');
119     axis square;
120
121     % Change axes to Voltage values
122     num_Ticks = 6;
123
124     % Change X-axis tick values and label
125     distance_between_ticks_X        = ceil(num_pixels_x/(num_Ticks-1));
126     ax1.XTick = [1, (1:num_Ticks-2).*distance_between_ticks_X, num_pixels_x];
127
128     distance_between_ticks_label_X = (X_ref_max - X_ref_min)/(num_Ticks-1);
129     newTickStr_X = cellstr(num2str(X_ref_min + ...
            (0:num_Ticks-1)'.*distance_between_ticks_label_X, '%2.3g'));
130     ax1.XTickLabel = newTickStr_X;
131     xlabel('X [V]');
132
133     %Change Y-axis tick values and label
134     distance_between_ticks_Y        = ceil(num_pixels_y/(num_Ticks-1));
135     ax1.YTick = [1, (1:num_Ticks-2).*distance_between_ticks_Y, num_pixels_y];
136
137     distance_between_ticks_label_Y = (Y_ref_max - Y_ref_min)/(num_Ticks-1);
138     newTickStr_Y = cellstr(num2str(Y_ref_min + ...
            (0:num_Ticks-1)'.*distance_between_ticks_label_Y, '%2.3g'));
139     ax1.YTickLabel = newTickStr_Y;
140     ylabel('Y [V]');
141
142     % Add colorbar
143     clbar = colorbar;
144     clbar_label = 'Z [V]';
145     set(get(clbar,'Title') ,'String',clbar_label);
146
147     %% Plot X,Y,Z data vs x,y trajectory
148     if nargin == 1 || nargin == 2 || nargin == 3
149         fig2 = figure;
150         subplot(3,1,1);
151         plot(T, X_ref, 'r'); hold on;
152         plot(T, X, 'b'); hold off; grid on;
153         ylabel('Voltage [V]');
154         xlabel('Time [sec]');
155         legend('X_{ref}(t)', 'X(t)');
156         x_plot_mid_val  = (max(X_ref)+min(X_ref))/2;
157         x_plot_diff_val = (max(X_ref)-min(X_ref))/2;
158         ylim([x_plot_mid_val - 1.2*x_plot_diff_val, x_plot_mid_val + 1.2*x_plot_diff_val]);
159
160         subplot(3,1,2);
161         plot(T, Y_ref, 'r'); hold on;
162         plot(T, Y, 'b'); hold off; grid on;
163         ylabel('Voltage [V]');
164         xlabel('Time [sec]');
165         legend('Y_{ref}(t)', 'Y(t)');
166         y_plot_mid_val = (max(Y_ref)+min(Y_ref))/2;
167         y_plot_diff_val = (max(Y_ref)-min(Y_ref))/2;
168         ylim([y_plot_mid_val - 1.2*y_plot_diff_val, y_plot_mid_val + 1.2*y_plot_diff_val]);
169
170         subplot(3,1,3);
171         plot(T, Z, 'b'); grid on;
172         ylabel('Voltage [V]');
173         xlabel('Time [sec]');
174         legend('Z(t)');
175     end
176
177     % Triangle tracking and staircase tracking plot
178     if nargin == 3
179         scan_freq = in3;
180         end_time = num_pixels_y/scan_freq;
181
182         zoom_start_T = 0.496;
183         zoom_end_T   = 0.504;
```

```
184
185         fig3 = figure;
186         plot(T, X_ref, 'r'); hold on;
187         plot(T, X, 'b'); hold off; grid on;
188         ylabel('Voltage [V]');
189         xlabel('Time [sec]');
190         legend('X_{ref}(t)', 'X(t)');
191         x_plot_mid_val = (max(X_ref)+min(X_ref))/2;
192         x_plot_diff_val = (max(X_ref)-min(X_ref))/2;
193         ylim([x_plot_mid_val - 1.2*x_plot_diff_val, x_plot_mid_val + 1.2*x_plot_diff_val]);
194         xlim([zoom_start_T*end_time, zoom_end_T*end_time]);
195
196         fig4 = figure;
197         plot(T, Y_ref, 'r'); hold on;
198         plot(T, Y, 'b'); hold off; grid on;
199         ylabel('Voltage [V]'); xlabel('Time [sec]');
200         lgn1 = legend('Y_{ref}(t)', 'Y(t)'); set(lgn1, 'Location', 'northwest');
201         xlim([zoom_start_T*end_time, zoom_end_T*end_time]);
202         axis 'auto y';
203
204         % X error plot
205         fig5 = figure;
206         plot(T, X_ref - X, 'g'); hold on;
207         plot(T, X_ref - x_plot_mid_val, 'r');
208         plot(T, X     - x_plot_mid_val, 'b'); hold off; grid on;
209         ylabel('Voltage [V]'); xlabel('Time [sec]');
210         legend('X_{error}(t)', 'X_{ref}(t)', 'X(t)');
211         ylim([-1.2*x_plot_diff_val, 1.2*x_plot_diff_val]);
212         xlim([zoom_start_T*end_time, zoom_end_T*end_time]);
213     end
214
215     %% Save image and plots if save_imag is true
216     if save_imag == true
217         [data_path,data_name,~] = fileparts(data_file_name);
218
219         image_file_name = fullfile(data_path, [data_name, '_image']);
220         saveas(fig1, [image_file_name, '.fig'], 'fig');
221         saveas(fig1, [image_file_name, '.eps'], 'epsc');
222
223         raw_plot_file_name = fullfile(data_path, [data_name, '_raw_plot']);
224         saveas(fig2, [raw_plot_file_name, '.fig'], 'fig');
225         saveas(fig2, [raw_plot_file_name, '.eps'], 'epsc');
226
227         if nargin == 3
228             triwave_file_name = fullfile(data_path, [data_name, '_triwave']);
229             saveas(fig3, [triwave_file_name, '.fig'], 'fig');
230             saveas(fig3, [triwave_file_name, '.eps'], 'epsc');
231
232             staircase_file_name = fullfile(data_path, [data_name, '_staircase']);
233             saveas(fig4, [staircase_file_name, '.fig'], 'fig');
234             saveas(fig4, [staircase_file_name, '.eps'], 'epsc');
235
236             triwave_error_file_name = fullfile(data_path, [data_name, '_triwave_error']);
237             saveas(fig5, [triwave_error_file_name, '.fig'], 'fig');
238             saveas(fig5, [triwave_error_file_name, '.eps'], 'epsc');
239         end
240     end
241
242 end % print_AFM_image
243
244 %% Helper functions
245 function found_data = extract_data_from_file(file_name, vars_to_find)
246     file_data       = load(file_name);
247     file_data_field = fieldnames(file_data);
248     data = file_data.(file_data_field{1});
249     fields = fieldnames(data);
250
251     found_data = cell(size(vars_to_find));
252     for j = 1:length(fields)
253         for k = 1:length(vars_to_find)
254             str = vars_to_find{k};
```

```
255                 if strcmp(fields{j}, str)
256                     found_data{k} = data.(fields{j});
257                 end
258             end
259         end
260     end
```

Listing E.2: `binary_greater_than.m`

```
1  function X_gt_idx = binary_greater_than(X, v_low, v_high, num_bins)
2  if v_low > v_high
3      error('v_high must be greater than v_low');
4  end
5
6  v_step = (v_high - v_low)/num_bins;
7  v = v_low:v_step:v_high;
8
9  X_gt_idx = bin_gr_than(X, v, 0);
10 end
11
12 function idx = bin_gr_than(X, v, idx_val)
13     if length(v) == 1
14         if X >= v(1)
15             idx = idx_val + 1;
16         else
17             idx = idx_val;
18         end
19         return;
20     end
21
22     idx_v_mid = ceil(length(v)/2);
23
24     if X >= v(idx_v_mid)
25         s = v(idx_v_mid+1:end);
26         idx = bin_gr_than(X, s, idx_val+idx_v_mid);
27         return;
28     else
29         s = v(1:idx_v_mid);
30         idx = bin_gr_than(X, s, idx_val);
31         return;
32     end
33 end
```

Bibliography

[1] D. A. John and K. Biswas. "Electrical equivalent circuit modelling of solid state fractional capacitor". *AEU - International Journal of Electronics and Communications*, 78 (2017). ISSN: 1434-8411.

[2] K. Lazopoulos, D. Karaoulanis, and A. Lazopoulos. "On fractional modelling of viscoelastic mechanical systems". *Mechanics Research Communications*, 78 (2016). ISSN: 0093-6413.

[3] R. Caponetto, S. Graziani, F. Pappalardo, E. Umana, M. Xibilia, and P. D. Giamberardino. "A Scalable Fractional Order Model for IPMC Actuators". *IFAC Proceedings Volumes*, 45 (2) (2012). ISSN: 1474-6670.

[4] S. David, J. López, and E. Pallone. "Fractional order calculus: Historical apologia, basic concepts and some applications". *Revista Brasileira de Ensino de Física*, 33 (2011).

[5] D. Xue. "Fractional-order control systems: Fundamentals and numerical implementations". Fractional calculus in applied sciences and engineering 1. ISBN 978-3-11-049999-5. De Gruyter, 2017.

[6] A. A. Dastjerdi, N. Saikumar, and S. H. HosseinNia. "Tuning guidelines for fractional order PID controllers: Rules of thumb". *Mechatronics*, 56 (2018). ISSN: 0957-4158.

[7] T. Andresen. "A Logarithmic-Amplitude Polar Diagram". *Modeling, Identification and Control*, 22 (2) (2001).

[8] S. G. Lipson. "Optical physics". eng. 3rd ed. Cambridge: Cambridge University Press, 1995. ISBN: 978-0-521-43047-0.

[9] *Diffraction-limited system*. en. 2018. URL: https://en.wikipedia.org/w/index.php?title=Diffraction-limited_system&oldid=869455215 (visited on 2019-03-13).

[10] The Nobel Prize. *Ernst Ruska, Facts*. URL: https://www.nobelprize.org/prizes/physics/1986/ruska/facts/ (visited on 2019-05-31).

[11] R. P. Erni. "Atomic Resolution Imaging with a sub-50 pm Electron Probe". (2009).

[12] T. G. Rochow and E. G. Rochow. "An Introduction to Microscopy by Means of Light, Electrons, X-Rays, or Ultrasound". Boston, MA: Springer US, 1978. ISBN: 978-1-4684-2454-6.

[13] E. Ruska. "The development of the electron microscope and of electron microscopy". *Bioscience Reports*, 7 (8) (1987). ISSN: 1573-4935.

[14] The Nobel Prize. *Gerd Binning, Facts*. URL: https://www.nobelprize.org/prizes/physics/1986/binnig/facts/ (visited on 2019-05-31).

[15] M. R. P. Ragazzon. "Nanopositioning in Atomic Force Microscopes: Robust Control Design, Order Reduction and Numerical Implementability". MA thesis. Norwegian University of Science and Technology, 2013.

[16] The Nobel Prize. *The Nobel Prize in Physics 1986*. URL: https://www.nobelprize.org/prizes/physics/1986/summary/ (visited on 2019-05-31).

[17] G. K. Bennig. "Atomic force microscope and method for imaging surfaces with atomic resolution". U.S. pat. 4724318A. I. B. M. Corp. 1988.

[18] G. Binnig, C. F. Quate, and C. Gerber. "Atomic Force Microscope". *Physical Review Letters*, 56 (9) (1986).

[19] D. Y. Abramovitch, S. B. Andersson, L. Y. Pao, and G. Schitter. "A Tutorial on the Mechanisms, Dynamics, and Control of Atomic Force Microscopes". (2007).

[20] K. Uchino. *Introduction to Piezoelectric Actuators and Transducers*. 2003. URL: https://www.researchgate.net/publication/235110841_Introduction_to_Piezoelectric_Actuators_and_Transducers (visited on 2019-06-22).

[21] A. A. Eielsen. "Topics in Control of Nanopositioning Devices". Norwegian University of Science, Technology, Faculty of Information Tech-

nology, Mathematics, and Electrical Engineering, Department of Engineering Cybernetics, 2012. ISBN: 978-82-471-3949-3.

[22] A. H. Moltumyr. "Fractional-Order Modeling of an Atomic Force Microscope". Project report. Department of Engineering Cybernetics, NTNU Norwegian University of Science and Technology, 2018.

[23] Millennium Mathematics Project, University of Cambridge. *Fractional Calculus II*. URL: https://nrich.maths.org/1369 (visited on 2019-05-24).

[24] C. A. Monje, Y. Chen, B. M. Vinagre, D. Xue, and V. Feliu-Batlle. "Fractional-order Systems and Controls: Fundamentals and Applications". Advances in Industrial Control. London: Springer-Verlag, 2010. ISBN: 978-1-84996-334-3.

[25] A. Oustaloup, F. Levron, B. Mathieu, and F. M. Nanot. "Frequency-band complex noninteger differentiator: characterization and synthesis". *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47 (1) (2000). ISSN: 1057-7122.

[26] I. Petráš. "Fractional-Order Nonlinear Systems - Modelling, Analysis and Simulation". Higher Education Press, Beijing, 2011.

[27] J. G. Balchen. "Reguleringsteknikk". In collab. with B. A. Foss, T. Andresen, and NTNU Department of Engineering Cybernetics. 5th edition. Trondheim: Department of Engineering Cybernetics, Norwegian University of Science and Technology, 2003. ISBN: 978-82-471-5147-1.

[28] N. Öztürk and A. Uraz. "An Extension of the Nyquist Criterion to a Class of Distributed Parameter Systems". (1982).

[29] A. Trächtler. *On BIBO stability of systems with irrational transfer function*. 2016. URL: https://www.researchgate.net/publication/301854944_On_BIBO_stability_of_systems_with_irrational_transfer_function (visited on 2019-05-29).

[30] A. Dabiri, B. P. Moghaddam, and J. A. T. Machado. "Optimal variable-order fractional PID controllers for dynamical systems". *Journal of Computational and Applied Mathematics*. Modern Fractional Dynamic Systems and Applications, MFDSA 2017 339 (2018). ISSN: 0377-0427.

[31] P. D. Mandić, T. B. Šekara, M. P. Lazarević, and M. Bošković. "Dominant pole placement with fractional order PID controllers: D-decomposition approach". *ISA Transactions*, 67 (2017). ISSN: 0019-0578.

[32] I. R. Birs, C. I. Muresan, O. Prodan, S. C. Folea, and C. Ionescu. "Structural vibration attenuation using a fractional order PD controller designed for a fractional order process". *IFAC-PapersOnLine*. 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control 51 (4) (2018). ISSN: 2405-8963.

[33] J. Fei and C. Lu. "Adaptive fractional order sliding mode controller with neural estimator". *Journal of the Franklin Institute*, 355 (5) (2018). ISSN: 0016-0032.

[34] G. Sun, L. Wu, Z. Kuang, Z. Ma, and J. Liu. "Practical tracking control of linear motor via fractional-order sliding mode". *Automatica*, 94 (2018). ISSN: 0005-1098.

[35] A. Guefrachi, S. Najar, M. Amairi, and M. Aoun. "Tuning of Fractional Complex Order PID Controller**This work was supported by the Ministry of the Higher Education and Scientific Research in Tunisia." *IFAC-PapersOnLine*. 20th IFAC World Congress 50 (1) (2017). ISSN: 2405-8963.

[36] A. San-Millan, V. Feliu-Batlle, and S. S. Aphale. "Fractional order implementation of Integral Resonant Control – A nanopositioning application". *ISA Transactions*. Fractional Order Signals, Systems, and Controls: Theory and Application 82 (2018). ISSN: 0019-0578.

[37] A. Tepljakov. "Fractional-order Modeling and Control of Dynamic Systems". Springer Theses. Springer International Publishing, 2017. ISBN: 978-3-319-52949-3.

[38] L. Marinangeli, F. Alijani, and S. H. HosseinNia. "Fractional-order positive position feedback compensator for active vibration control of a smart composite plate". *Journal of Sound and Vibration*, 412 (2018). ISSN: 0022-460X.

[39] J. L. FANSON and T. K. CAUGHEY. "Positive position feedback control for large space structures". *AIAA Journal*, 28 (4) (1990). ISSN: 0001-1452.

[40] A. A. Eielsen, M. Vagia, J. T. Gravdahl, and K. Y. Pettersen. "Damping and Tracking Control Schemes for Nanopositioning". *IEEE/ASME Transactions on Mechatronics*, 19 (2) (2014). ISSN: 1083-4435.

[41] A. Chevalier, C. Francis, C. Copot, C. M. Ionescu, and R. De Keyser. "Fractional-order PID design: Towards transition from state-of-art to state-of-use". *ISA Transactions*, 84 (2019). ISSN: 0019-0578.

[42] L. Kumar, P. kumar, Satyajeet, and D. Narang. "Tuning of Fractional Order PI D Controllers using Evolutionary Optimization for PID Tuned Synchronous Generator Excitation System". *IFAC-PapersOnLine*. 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control PID 2018 51 (4) (2018). ISSN: 2405-8963.

[43] G. Altintas and Y. Aydin. "Optimization of Fractional and Integer Order PID Parameters using Big Bang Big Crunch and Genetic Algorithms for a MAGLEV System". *IFAC-PapersOnLine*. 20th IFAC World Congress 50 (1) (2017). ISSN: 2405-8963.

[44] C. A. Monje, B. M. Vinagre, V. Feliu, and Y. Chen. "Tuning and auto-tuning of fractional order controllers for industry applications". *Control Engineering Practice*, 16 (7) (2008). ISSN: 0967-0661.

[45] D. Floreano and C. Mattiussi. "Bio-Inspired Artificial Intelligence, Theories, Methods, and Technologies". Cambridge, Massachusetts, London, England: The MIT Press, 2008. ISBN: 978-0-262-06271-8.

[46] M. Nazari-Heris, B. Mohammadi-Ivatloo, and G. B. Gharehpetian. "A comprehensive review of heuristic optimization algorithms for optimal combined heat and power dispatch from economic and environmental perspectives". *Renewable and Sustainable Energy Reviews*, 81 (2018). ISSN: 1364-0321.

[47] B. Khalfa and C. Abdelfateh. "Optimal tuning of fractional order P D µA controller using Particle Swarm Optimization algorithm". *IFAC-PapersOnLine*. 20th IFAC World Congress 50 (1) (2017). ISSN: 2405-8963.

[48] *Matlab Global Optimization Toolbox User's Guide*. MathWorks. 2019. URL: https://se.mathworks.com/help/pdf_doc/gads/gads_tb.pdf (visited on 2019-03-21).

[49] T. Andresen. *Nyquist plot with logarithmic amplitudes - File Exchange - MATLAB Central*. URL: https://se.mathworks.com/matlabcentral/fileexchange/7444 (visited on 2019-06-21).

[50] M. Altaher, D. Russell, and S. S. Aphale. "A dual-loop tracking control approach to precise nanopositioning". *Journal of Vibration and Control*, 25 (3) (2019). ISSN: 1077-5463.

Andreas Hanssen Moltumyr

Control in Atomic Force Microscopy: A Fractional Order Approach

NTNU
Norwegian University of
Science and Technology