



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Nanopositioning in Atomic Force Microscopes

Robust Control Design, Order Reduction and  
Numerical Implementability

**Michael Remo Palmén**  
**Ragazzon**

Master of Science in Engineering Cybernetics

Submission date: June 2013

Supervisor: Jan Tommy Gravdahl, ITK

Co-supervisor: Arnfinn Aas Eielsen, ITK

Norwegian University of Science and Technology  
Department of Engineering Cybernetics



## Abstract

This thesis can be divided into two overall topics. The first concerns control design for nanopositioning devices, and specifically an Atomic Force Microscope (AFM). The second topic concerns the implementability issues involved with running a complex controller in real-time especially for a stiff system with fast modes.

A robust  $\mathcal{H}_\infty$  multiple-input multiple-output (MIMO) controller is designed for the lateral positioning stage of an AFM. This controller is compared to a  $\mathcal{H}_\infty$  single-input single-output (SISO) controller based on independent axis design and a PID controller. We consider especially how much benefit there is to a more complex MIMO controller compared to independent axis controllers in terms of cross-coupling gains between the lateral axes. This is important to consider in a scanning application where any cross-coupling can be detrimental to the resulting image. Experiments are performed for this purpose. The results show that the differences in terms of cross-coupling reduction is negligible compared to noise and disturbances. This favors the choice of independent axis controllers over MIMO controllers because of their simpler design and increased implementability. In terms of dampening the resonant gain, the two  $\mathcal{H}_\infty$  controllers are shown to perform considerably better than the PID controller.

A model-based controller such as the  $\mathcal{H}_\infty$  MIMO controller can quickly become complex and may be difficult to run in real-time on hardware with limited computational power. We show how to find the maximum step-size for numerical stability of a given controller using an explicit Runge-Kutta (ERK) solver as is commonly used in real-time applications. We also consider model reduction on the controller and how this affects the required step-size and how much it reduces the computational complexity. We have shown that the 18th order  $\mathcal{H}_\infty$  MIMO controller could be reduced to a 10th order controller without any significant reduction in performance or stability, which resulted in a 46.7% reduction in execution time partly because the order reduction enabled us to use a simpler solver type.

## Sammendrag

Denne oppgaven er delt opp i to overordnede temaer. Det første temaet omhandler kontrolldesign for nanoposisjoneringsenheter, spesifikt for atomkraftmikroskop. Det andre temaet omhandler utfordringer rundt implementering av komplekse kontrollere for et sanntids-system, spesielt der system er stivt med hurtige moduser.

En robust  $\mathcal{H}_\infty$  fler-inngang fler-utgang (MIMO) kontroller er designet for sideveis bevegelse av en plattform for et atomkraftmikroskop. Denne kontrolleren er sammenlignet med en  $\mathcal{H}_\infty$  en-inngang en-utgang (SISO) kontroller, samt en PID kontroller. Vi betrakter spesielt gevinsten ved å bruke en mer kompleks MIMO-kontroller sammenlignet med en SISO-kontroller i forhold til krysskoblings-forsterkningen mellom de to sideveis aksene. Dette er spesielt viktig i skanne-applikasjoner hvor krysskoblingen kan føre til ugunstige bilder. Eksperimenter er utført for å undersøke dette. Resultatene viser at forskjellene i redusert krysskobling er neglisjerbar i forhold til støy og forstyrrelser. Dette favoriserer valget av SISO-kontrollere over MIMO-kontrollere da disse har et enklere design og høyere implementerbarhet. Imidlertid er de to  $\mathcal{H}_\infty$  kontrollerne mye bedre til å dempe ut resonans-forsterkningen.

En modellbasert kontroller slik som  $\mathcal{H}_\infty$  MIMO-kontrolleren kan fort bli kompleks, noe som kan resultere i at den blir vanskelig å implementere på maskinvare med begrenset regnekapasitet. Vi viser hvordan man kan finne den lengste skritt-lengden for numerisk stabilitet av en gitt kontroller for en numerisk løser av typen eksplisitt Runge-Kutta. Vi viser også hvordan modellreduksjon utført på kontrolleren endrer ytelse og stabilitet. I tillegg undersøker vi hvordan dette endrer øvre grense på skritt-lengden og hvor stor reduksjon vi får i form av regnekompleksitet. Vi har vist at den 18. ordens originale  $\mathcal{H}_\infty$  MIMO-kontrolleren kan bli redusert til en 10. ordens kontroller uten nevneverdige forskjeller i ytelse og stabilitet. Dette resulterte i en 46,7% reduksjon i regnetiden delvis fordi modellreduksjonen tillot oss å bruke en enklere numerisk løser.



## MSc thesis assignment

Name of the candidate: Michael Remo Palmén Ragazzon  
Subject: Engineering Cybernetics  
Title: Nanopositioning in Atomic Force Microscopes: Robust control design, order reduction and numerical implementability

### **Background**

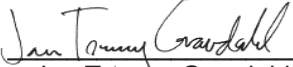
Nanopositioning is positioning of some device with nanometer accuracy. This kind of performance is impossible without feedback control, and as such, control systems design is a key enabling technology for many nanoscience applications and in particular nanopositioning. The atomic force microscope (AFM) is one of the foremost tools for imaging, measuring and manipulation at the nanometer scale. In this work, experiments are to be done at the nanopositioning lab at the Department of Engineering Cybernetics and its Park Systems XE-70 AFM.

### **Assignment:**

1. Present the AFM and the nanopositioning problem
2. Design  $\mathcal{H}_\infty$  control laws for the nanopositioning system. Consider especially any cross-coupling between the x- and y-axes of the positioning stage.
3. Compare experimentally, on the departments AFM, the performance of the controllers with the performance of a PID-control law.
4. Design of control laws by using  $\mathcal{H}_\infty$  methods tends to lead to high order control laws. Investigate the possibility of controller order reduction for increased implementability.
5. Based on the main results of the thesis, write a paper to be submitted to the IFAC World Congress

To be handed in by: 10/6-2013

Trondheim, 23.01.2013

  
Jan Tommy Gravdahl  
Professor, supervisor



# Preface

*Science is the best thing that humanity has ever come up with. And if it isn't, then science will fix it.* —Bill Nye

I must admit, the word nano gives me a tickling sensation. There are so many claims to how much good nanotechnology can do for our species, yet dystopian-minded people aren't in lack of considering end-of-the-world scenarios involving nanobots. Perhaps it may be the Russian roulette of the world, nevertheless, I am glad to have been given the opportunity to contribute something to the field, however small the contributions may be.

I would especially like to thank Prof. Jan Tommy Gravdahl for the regular meetings and continued support. Also thanks to Post Doc. Arnfinn Aas Eielsen for giving me lots of advice and help along the way. Thanks to both of you for trusting me with all the equipment at the nano-lab (I swear I didn't ruin anything).

A big thanks goes out to my fellow students and friends whom all have made the last five years a fun and interesting part of my life. My studies here at NTNU have taught me a great deal of knowledge, and I'm looking forward to learn more.

Lastly, I would like to thank my parents, my brother and the rest of my family. This thesis is the result of all the things you have taught me.

Michael Remo Palmén Ragazzon  
Trondheim, June 2013





# Contents

<b>Abstract</b>	<b>i</b>
<b>Preface</b>	<b>v</b>
<b>Nomenclature</b>	<b>xv</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Outline . . . . .	2
1.2. Notation . . . . .	3
<b>2. Atomic Force Microscopy</b>	<b>5</b>
2.1. Operating Modes . . . . .	7
2.1.1. Static mode . . . . .	7
2.1.2. Dynamic mode . . . . .	8
2.2. Piezoelectric Actuator . . . . .	10
2.2.1. Hysteresis and Creep . . . . .	11
2.3. Challenges with AFM . . . . .	13
2.4. Sample Scan . . . . .	13
<b>3. Robust Control Theory</b>	<b>15</b>
3.1. MIMO Feedback System . . . . .	15
3.2. MIMO Robust Stability . . . . .	18
3.2.1. Robust Stability Condition . . . . .	18
3.2.2. Multiplicative Output Uncertainty . . . . .	19
3.2.3. Specifying the Weight $W$ . . . . .	21
3.3. Control Design Objectives . . . . .	21
3.4. PID Controller . . . . .	23
3.4.1. Tuning PID Controller . . . . .	23
3.5. $\mathcal{H}_\infty$ Control . . . . .	24
3.5.1. Procedure for Mixed Sensitivity $\mathcal{H}_\infty$ Control . . . . .	25
3.5.2. Other $\mathcal{H}_\infty$ Control Approaches . . . . .	26

<b>4. Control Strategies for Lateral Motion</b>	<b>27</b>
4.1. Feedforward Control . . . . .	28
4.1.1. Creep and Hysteresis Compensation . . . . .	28
4.1.2. Inversion Techniques . . . . .	29
4.2. Feedback Control . . . . .	29
4.2.1. PID . . . . .	29
4.2.2. $\mathcal{H}_\infty$ Loop Shaping . . . . .	30
4.2.3. $\mathcal{H}_\infty$ Mixed Sensitivity . . . . .	30
4.3. Other Control Approaches . . . . .	30
<b>5. Comparison of Control Laws on Cross-Coupling Effects</b>	<b>31</b>
5.1. Experimental Setup . . . . .	32
5.1.1. Frequency Domain Response . . . . .	33
5.2. System Identification . . . . .	34
5.2.1. Frequency Response $\hat{G}(\omega)$ and Fitted Model $G(s)$ . . . . .	34
5.2.2. Time Delay . . . . .	35
5.2.3. Uncertainty weight . . . . .	36
5.3. Controller Design . . . . .	37
5.3.1. PID controller . . . . .	37
5.3.2. $\mathcal{H}_\infty$ Mixed Sensitivity Controllers . . . . .	38
5.4. Implementation Details and Issues . . . . .	41
5.5. Results . . . . .	42
5.6. Observations . . . . .	48
5.7. Conclusion . . . . .	50
<b>6. Control Order Reduction</b>	<b>51</b>
6.1. Background Theory . . . . .	52
6.1.1. Balanced Realization . . . . .	52
6.1.2. Reduction Problem Formulation . . . . .	53
6.1.3. Method 1: Model Truncation . . . . .	54
6.1.4. Method 2: Model Residualization . . . . .	55
6.1.5. Other Methods . . . . .	55
6.2. Reduction of Controller . . . . .	56
6.3. Simulation Results . . . . .	56
6.3.1. Observations . . . . .	60
6.4. Experimental Results . . . . .	60
6.4.1. Observations . . . . .	62
6.5. Conclusion . . . . .	62
<b>7. Computational Complexity after Control Reduction</b>	<b>63</b>
7.1. Simulation Results . . . . .	63
7.1.1. Observations . . . . .	64
7.2. Experimental Results . . . . .	68
7.2.1. Observations . . . . .	68
7.3. Conclusion . . . . .	69

---

<b>8. Numerical Stability of Controller</b>	<b>71</b>
8.1. Explicit Runge-Kutta Methods . . . . .	72
8.2. Stability of Explicit Runge-Kutta Methods . . . . .	73
8.3. Runge-Kutta Stability of Linear Systems . . . . .	77
8.4. Determining Maximum Step-Size . . . . .	77
8.5. Observations . . . . .	79
8.6. Conclusion . . . . .	80
<b>9. Conclusions</b>	<b>83</b>
9.1. Future Work . . . . .	84
<b>A. Definitions and Properties</b>	<b>87</b>
A.1. $\mathcal{H}_2$ and $\mathcal{H}_\infty$ Norm . . . . .	87
A.2. Hankel Norm and Hankel Singular Value . . . . .	88
A.3. Butcher Tableau for Dormand-Prince Methods . . . . .	89
<b>B. Contents of Attached Zip-file</b>	<b>91</b>
B.1. File Descriptions . . . . .	91
<b>C. Matlab Code</b>	<b>93</b>
C.1. h_inf_mixed_sensitivity.m . . . . .	93
C.2. identification.m . . . . .	94
C.3. Phi.m . . . . .	96
C.4. rk_stability.m . . . . .	98
C.5. test_bench_raster_pattern.m . . . . .	99
C.6. test_bench_reduced_controllers.m . . . . .	100
<b>D. Paper</b>	<b>103</b>
<b>Bibliography</b>	<b>113</b>



# List of Figures

2.1. Principle of operation of AFM and STM . . . . .	6
2.2. A typical AFM device setup . . . . .	7
2.3. Tip-sample interaction force . . . . .	9
2.4. AFM Amplitude Modulated mode . . . . .	10
2.5. Piezo actuators . . . . .	11
2.6. Hysteresis and creep . . . . .	12
2.7. Scan image using a Park Systems XE-70 AFM . . . . .	14
3.1. Feedback control system . . . . .	16
3.2. $M\Delta$ -structure used for robust stability analysis . . . . .	19
3.3. Feedback system with multiplicative output uncertainty . . . . .	20
3.4. General control configuration . . . . .	24
3.5. General control configuration of the mixed sensitivity $\mathcal{H}_\infty$ problem. . . . .	25
4.1. Combined feedforward and feedback controller schemes . . . . .	28
5.1. Scanning motion in SPM . . . . .	32
5.2. Park Systems XE-70 . . . . .	33
5.3. Experimental Setup Overview . . . . .	34
5.4. Plant frequency response and fitted model . . . . .	35
5.5. Identification of time delay . . . . .	36
5.6. Robustness fit, $W(s)$ and $\hat{W}(\omega)$ . . . . .	37
5.7. Sensitivity $\sigma(S)$ and complementary sensitivity $\sigma(T)$ for the three controllers . . . . .	40
5.8. Simulink diagram of the controller implementation . . . . .	41
5.9. Comparison of experimental closed loop frequency response . . . . .	43
5.10. Comparison of $\sigma(\hat{T})$ versus $\sigma(T)$ for each controller. . . . .	44
5.11. Scanning motion output at 10 Hz. . . . .	45
5.12. Scanning motion output at 100 Hz. . . . .	46
6.1. Simulink model for simulation of step-response. . . . .	57
6.2. Reduced controller order properties . . . . .	57
6.3. Closed-loop frequency response $\sigma(S)$ , $\sigma(T)$ versus $\sigma(S_r)$ , $\sigma(T_r)$ . . . . .	58

---

6.4.	Simulated step-response in reference signal on the $x$ -axis, reduced vs original controller . . . . .	59
6.5.	Experimental complementary sensitivity $\sigma(\hat{T}_r)$ for the reduced order controllers . . . . .	61
6.6.	Experimental step-response in reference signal on the $x$ -axis, reduced vs original controller . . . . .	61
7.1.	Simulation using an implicit Runge-Kutta method . . . . .	65
8.1.	Stability region of various ERK methods . . . . .	76
8.2.	Illustration of the trade-off between several factors for implementability	81

# List of Tables

3.1. Feedback control notation . . . . .	18
5.1. Summary of weighting transfer functions . . . . .	39
5.2. Bandwidth, robustness, and implementation comparison between the three controllers . . . . .	40
5.3. Controller implementation details . . . . .	42
5.4. Performance comparison of the three controllers . . . . .	47
6.1. Hankel singular values of the balanced realization of the controller . .	56
7.1. Simulation time and stability with different controller model order, solver types and step times (part 1/2) . . . . .	66
7.2. Simulation time and stability with different controller model order, solver types and step times (part 2/2) . . . . .	67
7.3. Average Task Execution Time with different controller model order and solver types . . . . .	69
8.1. Selection of explicit Runge-Kutta methods with their corresponding Butcher tableau and stability function . . . . .	75
8.2. Eigenvalues of original controller and 10th order reduced controller . .	78
8.3. Maximum step-size of a given explicit Runge-Kutta method for the various controllers . . . . .	79
A.1. Signal norms and transfer function norms for two types of input signals	88
A.2. Butcher tableau for Dormand-Prince (erk5) method [19] . . . . .	89
A.3. Butcher tableau for Dormand-Prince (erk8) method [46] . . . . .	90





# Nomenclature

$G$	nominal plant model
$G_d$	disturbance model
$G_p$	perturbed plant model
$K$	controller
$L$	loop function
$S$	sensitivity function (p.16)
$T$	complementary sensitivity function (p.16)
$\hat{G}$	experimentally obtained frequency response of plant
$\hat{T}$	experimentally obtained closed-loop frequency response
$d$	disturbance signal
$e$	tracking error
$n$	measurement noise
$r$	reference signal
$u$	plant input
$y$	plant output
$\bar{\sigma}(G)$	maximum singular value of $G$ (p.17)
$\underline{\sigma}(G)$	minimum singular value of $G$ (p.17)
$\sigma(G)$	maximum and minimum singular value of $G$ (p.17)
$\dot{x}$	time-derivate of $x$ , i.e. $\frac{dx}{dt}$

---

$\lambda(A)$	eigenvalues of $A$
$\ u\ _2$	2-norm of vector $u$
$\ G\ _\infty$	$\mathcal{H}_\infty$ norm of transfer function $G$ (p.87)
$(A, B, C, D)$	state-space model with matrices $A, B, C, D$ (p.52)
$\triangleq$	equal by definition
$\omega_{bS}$	bandwidth of $S$ (p.22)
$\omega_{bT}$	bandwidth of $T$ (p.22)
erk5	an explicit Runge-Kutta solver of order 5
$\text{std}(x)$	standard deviation of $x$ -signal
ADC	analog-to-digital converter
AFM	Atomic Force Microscopy
DAC	digital-to-analog converter
ERK	explicit Runge-Kutta
iff	if and only if
MIMO	multiple-input multiple-output
ODE	ordinary differential equation
SISO	single-input single-output
SPM	Scanning Probe Microscopy
STM	Scanning Tunneling Microscopy
TET	task execution time

# Chapter 1

## Introduction

The broad field of nanotechnology involves manipulation of matter down to the atomic scale. This topic is considered within a broad range of scientific fields such as organic chemistry, molecular biology, and semiconductor physics. The applications range from self-assembly of molecules, hard-disk drive systems, to development of new materials. One of the fundamental technologies of the field is Scanning Probe Microscopy (SPM). This is a branch of microscopy that involves using a physical probe which scans over the topography of the sample in a raster pattern. The term SPM covers many microscopy techniques such as Atomic Force Microscopy (AFM) and Scanning Tunneling Microscopy (STM), both of which can achieve resolutions down to the atomic scale. We will present AFM to more detail in this thesis.

The term nanopositioning refers to position control of devices down to atomic resolutions. An important application in nanopositioning is control of AFMs. The control problems of such microscopes are divided into two general domains in the literature. One deals with lateral control, while the other deals with vertical control. Positioning in the lateral ( $xy$ -) axes is commonly achieved by using piezoelectric actuators. Such actuators have the ability to vary their length based on the applied voltage, and is dominantly used in nanopositioning applications. They can be challenging to control due to nonlinearities such as hysteresis and creep. Other challenges that are common within nanopositioning include lightly damped vibration modes giving rise to high resonant gains, and large model uncertainties. Additionally, the cross-coupling between axes may adversely affect performance and result in badly scanned images.

In this thesis we will present control design for the lateral axes of an AFM. We will design three different feedback controllers and implement them on a commercial AFM. Our primary goal is to investigate the differences in cross-coupling gain using the respective controllers. In the literature on nanopositioning, assumptions are often made that the cross-coupling gains are low enough such that a controller can be designed for one axis at a time. We will carry out these experiments to investigate how much the performance is affected by compensating for the cross-coupling gains by using multiple-input multiple-output (MIMO) control. The most complex controller

to be designed is a  $\mathcal{H}_\infty$  MIMO mixed sensitivity controller. The second is similarly a  $\mathcal{H}_\infty$  mixed sensitivity controller, but is designed for one axis at a time and does not take cross-coupling gains into account. Lastly, a simple PID controller is designed for each axis. Every controller will be implemented and several experiments run to investigate the differences between them.

The controllers just described can be represented by continuous-time state-space models. For a real-time implementation however, the model is solved at discrete time-steps using a fixed step-size. Many popular solver types are based on the family of explicit Runge-Kutta (ERK) methods. These solvers become unstable if the step-size is too large. At the same time, the complexity of a controller running on hardware with limited computational power puts a lower limit on the step-size the hardware needs to perform the necessary calculations. This is important to consider especially for stiff systems such as our nanopositioning application. We will discuss how to reduce the lower limit and at the same time make sure the system is numerically stable such that it becomes implementable.

A model-based  $\mathcal{H}_\infty$  controller has a tendency to become computationally complex. In order to reduce the computational complexity such that it becomes easier to implement, we will perform model order reduction on the  $\mathcal{H}_\infty$  MIMO controller. We will investigate how the stability and performance is affected at various reduced orders, and estimate the resulting reduction in computational complexity. By reducing the computational complexity the lower limit on the step-size is decreased.

Lastly we perform an analytical approach to gain some insight into the numerical stability problems of a real-time implementation of a controller. We investigate the stability properties of some ERK methods, and how to determine the maximum step-size for a given controller. The effect of performing model order reduction on the controller is also discussed in terms of affecting the upper limit of the step-size.

## 1.1. Outline

In [chapter 2](#) we present the working principles of Atomic Force Microscopy (AFM), including control in the vertical axis. Although we will not implement such a controller, we present this matter for the sake of completeness such that the reader has a better sense of understanding of AFM.

Next, in [chapter 3](#), we present background theory necessary for designing and analyzing the controllers in later chapters. The characteristics of a MIMO feedback system are presented and the condition of robust stability in the presence of multiplicative output uncertainty is given. Additionally, an introduction to  $\mathcal{H}_\infty$  controllers including the mixed sensitivity method is presented.

In [chapter 4](#) we give an overview of the current literature on lateral control within the topic of nanopositioning.

Chapters 5–8 can be considered the main contributions of this thesis. In [chapter 5](#) we investigate the differences on cross-coupling between a  $\mathcal{H}_\infty$  MIMO controller,  $\mathcal{H}_\infty$

SISO controller, and PID controller. It gives details on the experimental setup, model identification and controller design. Experimental results are provided and discussed.

In [chapter 6](#) we examine the effect of reducing the model order of the  $\mathcal{H}_\infty$  MIMO controller. Some background theory on model reduction is provided, then the stability and performance at various reduced orders is discussed.

The effect of model order reduction on the computational complexity is investigated in [chapter 7](#) which is important to consider for implementability. Both simulation and experimental results are provided.

An analysis of numerical stability for a selection of ERK methods applied to a controller is presented in [chapter 8](#). The trade-off between step-size, controller complexity, solver complexity, eigenvalues of the controller, and hardware performance for a real-time controller implementation is discussed.

Finally, [chapter 9](#) provides some concluding remarks on the results of this thesis, with some suggestions for future work.

## 1.2. Notation

Transfer functions are denoted by upper-case symbols, and often their dependency on the Laplace-variable  $s$  is omitted for ease of notation. Frequency domain data such as gathered from experimental data will be denoted by a hat, e.g.  $\hat{G}(\omega)$  or just  $\hat{G}$  for simplicity. Closed-loop transfer functions such as  $S$  and  $T$  is found using the identified nominal plant  $G$  and some controller  $K$ . Their experimentally measured analogues are again denoted by a hat, i.e.  $\hat{S}$  and  $\hat{T}$ . Also see the nomenclature on page [xvi](#).

Sometimes we use the term “SISO controller” or “independent axis controller” where we actually mean “controller based on independent axis design” for convenience. Independent axis design means that the controller is designed for one axis at a time and does not consider cross-couplings in a multidimensional plant.



## Chapter 2

# Atomic Force Microscopy

Atomic Force Microscopy (AFM) is a tool for imaging, measuring, and manipulating matter at high resolutions down to the nanometer scale. It is one of several techniques within the category of Scanning Probe Microscopy (SPM). As the name suggests, SPM uses a small probe which scans across the target surface. In many cases the probe is controlled to follow a certain path in the  $xy$ -plane, while the  $z$ -axis position along the probe is adjusted such that the tip follows the surface contour as illustrated in [Figure 2.1](#). The material presented in this chapter is primarily based on [1, 30, 31, 47, 50].

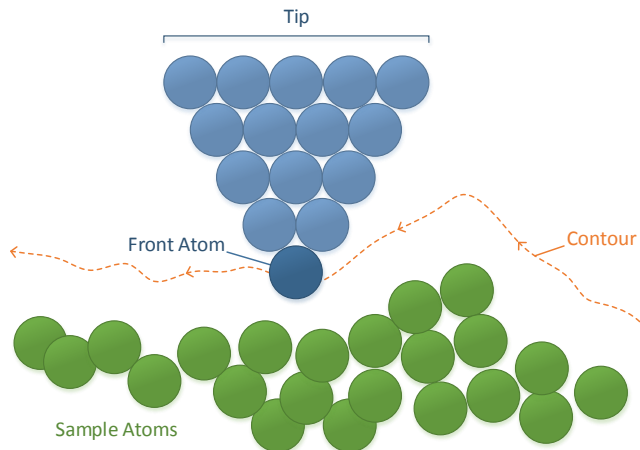
Scanning Tunneling Microscopy (STM) was the first type of SPM developed in 1981, with the first results published by Binnig et al. [7]. It is based on the quantum tunneling effect, which allows electrons to flow between the tip of the probe and the sample material. The current resulting from the electron flow is a function of the tip distance from the sample material. The current is kept at a constant value as the probe scans over the sample. By keeping the current constant, the distance is also constant, and then one can create a height map of the sample by recording the  $xyz$ -position of the tip as it scans the surface. STM was the first device capable of imaging individual atoms, and after its invention it quickly contributed to solving several problems within surface science such as the structure of the Si(111)-(7x7) surface, ultimately awarding the inventors with the Nobel Prize in Physics in 1986. A drawback with STM is that the samples need to be conductive so only metals and semiconductors can be investigated, additionally, most scans require an ultra-high vacuum (UHV) environment to work sufficiently. Thus the inventors tried to find a new method without these drawbacks, resulting in the Atomic Force Microscope.

AFM was first introduced in Binnig et al. [8], and builds on many of the principles of STM. Instead of tunneling electrons across to the sample, the tip is placed so close to the sample atoms that they are essentially in contact. This produces interaction forces between the probe and sample material. The probe is mounted on a cantilever, which is deflected when there is any interaction force, see [Figure 2.2](#) for an illustration. This small deflection is commonly measured by using a laser pointing at the cantilever,

which is reflected up to a photodetector. Other less used methods include optical interferometry, and the capacitance method. The topography of the sample can be determined by the deflection of the cantilever as the probe scans the surface.

AFM has the capability to image in vacuum, ambient air, or in liquids. Not only can it find the topology of the surface, but it is also possible to measure other characteristics such as magnetic properties, and electrostatic forces down to pN scales. AFM has also been used to manipulate objects for lithography, nanomanipulation, and nanoassembly. Like the STM, an AFM can achieve atomic resolutions, but this has only been possible in UHV environments.

There are primarily two modes of operation in AFM, *static* and *dynamic*. In static mode, the applied force on the sample is kept constant by varying the  $z$ -position of the probe. As with STM, by scanning the probe in the  $xy$ -plane, and recording the resulting  $z$ -position, a topographic map can be created of the sample. In dynamic mode, the cantilever is actuated and oscillates up and down. This interaction will change the amplitude and frequency of the cantilever vibration which can be measured and used in a feedback loop for control of the vertical  $z$ -position. These operating modes will be studied in more detail in the next section.



**Figure 2.1:** Principle of operation of both AFM and STM. In STM electrons are tunneled between the tip and sample, and the resulting current is kept constant. In AFM static mode, the tip-sample contact force is kept constant.



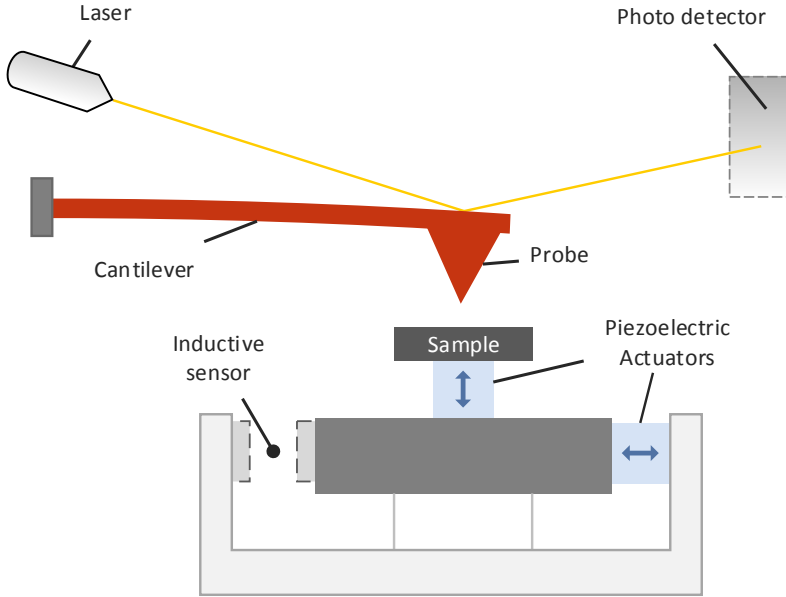


Figure 2.2: A typical AFM device setup

## 2.1. Operating Modes

In AFM, we are measuring the deflection of the cantilever, but ultimately we are interested in finding information about the sample material such as topography. Determining the topography from the cantilever deflection depends on the operating mode of the AFM, but usually involves some form of feedback loop. We will now present two commonly used operating modes for AFM in more detail.

### 2.1.1. Static mode

Consider the case of stationary conditions such that no scanning is performed and sample conditions is constant, then the cantilever deflection is proportional to the force between the tip and sample. With a rectangular cantilever, the displacement  $d$  at the end of the cantilever is [6]

$$d = k_p F(r) \quad (2.1)$$

where the constant  $k_p$  depends on the dimensions and material of the cantilever and  $F(r)$  is the tip-sample interaction force which depends on the distance between the tip and sample  $r$ . The interaction force between the tip and sample is nonlinear, and can be approximated by the Lennard-Jones potential [1]

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

where  $\sigma$  is an interaction parameter,  $r$  is the distance between the tip and sample,  $k_1$  is a constant which depends on the geometry and material of the tip and sample, and  $F(r)$  represents the interaction force between a flat sample and spherical tip. The equation is plotted in [Figure 2.3a](#) for some parameters. As can be seen, at larger distances (relatively speaking), there is an attractive force between the tip and sample, which results from van der Waals forces. As the distance is reduced, the repulsive electrostatic force becomes dominant, and the net force becomes repulsive.

The cantilever displacement  $d$  can be measured so we could theoretically solve for  $r$  in (2.1)-(2.2) to find the distance between tip and sample. However, the parameters of the equations are to a large extent unknown. The problem of finding the topography of the sample is instead solved by using a feedback loop from measurements of the absolute position of the sample. By keeping the tip-sample force  $F(r)$  constant, the distance is also kept constant. Now, the changes in absolute position of the sample along the  $z$ -axis represents the topography. A simple P-controller could be used for feedback. The reference setpoint for the force is commonly set to the beginning of the attractive region.

We haven't yet considered the impact of the force from the deflection of the cantilever when it is pushed away from its resting position. Although the cantilever dynamics can be quite complex, at stationary conditions we can approximate the cantilever force  $F_c$  by Hooke's Law

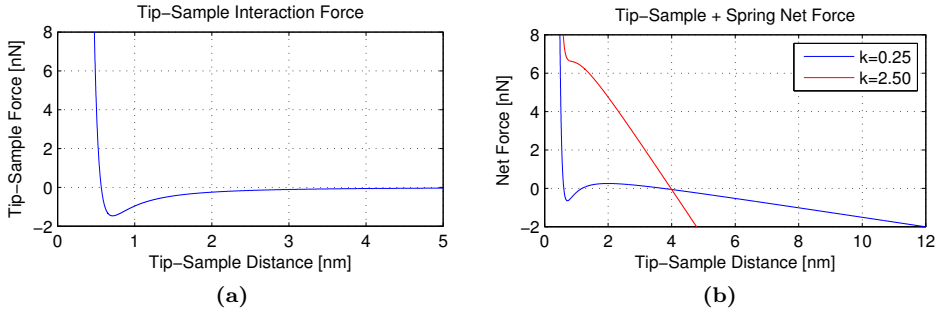
$$F_c = -kd \tag{2.3}$$

where  $k$  is the spring constant and  $d$  is the deflection from rest position. The sum of the tip-sample and spring force is plotted in [Figure 2.3b](#) for two different values of  $k$ . Note that the softer spring has two stable points where the net force is zero. This represent a problem because the tip may end up jumping between the two points if we have disturbances, changing parameters, or quick changes in topography as we scan the surface. In the literature, this is called the *jump-to-contact* phenomena [50]. This can be solved by using a stiffer spring (cantilever) as evident from [Figure 2.3b](#), since we only have one stable point for the stiff spring. But using a stiffer spring requires more sensitive deflection measurement equipment as the cantilever will deflect less for a given tip-sample force, so there is a trade-off between stability and accuracy here.

These problems motivate the investigation into an oscillating cantilever. An oscillating cantilever will virtually stiffen the spring at the point of largest force gradient, so we can easily avoid the *snap-to-contact* phenomena.

### 2.1.2. Dynamic mode

The previous section motivates the investigation into the dynamic mode of operation where the cantilever is oscillated at some frequency. There are two different methods under the term dynamic mode. The first one performs oscillations with a small amplitude in the attractive region of the tip-sample force and is often called the true non-contact mode. This mode can be used to reduce the damage to the sample material and tip due to the smaller interaction forces. Alternatively, in the tapping



**Figure 2.3:** (a) Tip-sample interaction force, with  $k_1 = 10^{-9}$ ,  $\sigma = 10^{-9}$ . (b) Tip-sample and cantilever net force with cantilever rest-position at the 4 nm mark for two different spring constants. Positive values are repulsive, while negative values attractive.

mode, the amplitude is much larger, and the tip swipes past the entire attractive region and into the repulsive region of the surface. The rest of the discussion here encompasses both of these schemes.

We can describe a cantilever that oscillates freely as a harmonic oscillator, where the deflection at the end of the cantilever is

$$d(t) = A \cos(2\pi ft + \phi) \quad (2.4)$$

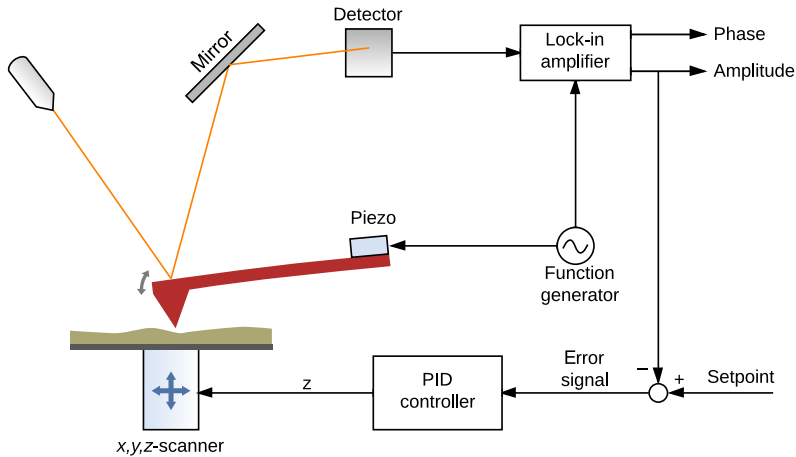
where  $A$  is the amplitude,  $f$  is the resonant frequency, and  $\phi$  is the phase. In tapping mode, the cantilever will be excited by forces every time the tip comes close to the sample (once each period). Because the force is nonlinear, the oscillation of the cantilever becomes anharmonic, and it can no longer be described by a single sine. Trying to interpret the response on  $d(t)$  to the interaction force is non-trivial. Again, as with the static mode, our only option is to use a feedback controller to control the  $z$ -axis position such that we have a constant interaction force.

There are primarily two different feedback schemes in use, the amplitude modulated (AM) and frequency modulated (FM) modes. These work by oscillating the cantilever around its resonant frequency using an actuator. When the tip interacts with the sample, the amplitude, phase and frequency will change depending on which side the resonant peak the oscillations are occurring. In AM mode, the amplitude is used as a feedback signal to the vertical  $z$ -actuator of the AFM as illustrated in [Figure 2.4](#). By keeping the amplitude constant in a feedback loop, the tip-sample interactions are also constant and thus the height over the surface is maintained. By recording the  $xyz$ -position over time, we can then generate an image of the topography.

FM mode is somewhat more complicated as it requires another feedback circuit for driving the cantilever with constant amplitude. If the amplitude is kept constant, the changes in frequency can be used as the feedback signal to the  $z$ -actuator. This mode requires very accurate instrumentation to detect such small changes to the frequency,

and is therefore for the most part only used in vacuum operating conditions where this is easier.

The phase can also be used as a feedback signal, but it is more complicated than using the amplitude. There has also been research into exploiting the higher-orders of the Fourier expansion of the anharmonic oscillations to plot more information about the sample such as in [34].



**Figure 2.4:** AFM Amplitude Modulated mode. The cantilever is oscillated near its resonant frequency. These oscillations will be detected by the photo detector, and the lock-in amplifier demodulates the signal to provide the signal amplitude and phase. The amplitude is used for feedback by a PID controller. Based on [50].

## 2.2. Piezoelectric Actuator

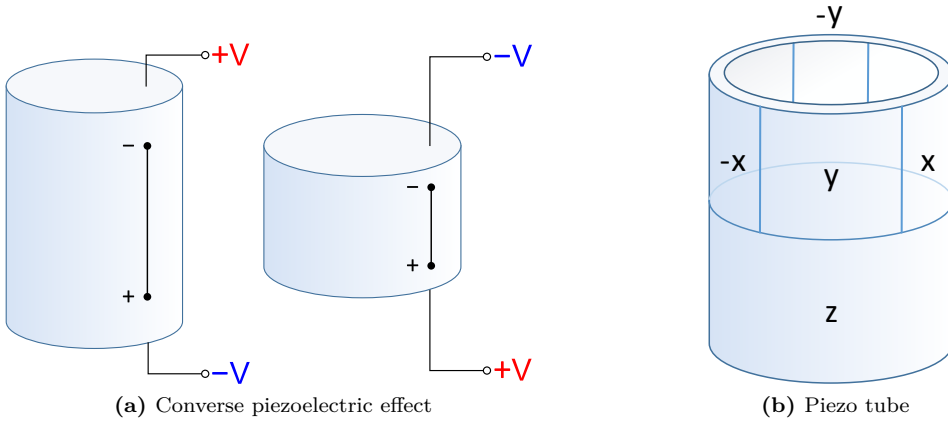
AFM is almost exclusively controlled by piezoelectric actuators. These actuators have very high resolution only limited by the instrumentation used. Additionally, they provide high force and frictionless motion. When a voltage is applied to the piezoelectric element, it will expand or contract along the poling axis. This is called the converse piezoelectric effect illustrated in Figure 2.5a.

As reported by [1], the most common actuator in AFM is a three degree-of-freedom piezo tube as shown in Figure 2.5b. It is actuated in the  $x$  and  $y$  directions by four electrodes placed around the cylinder, while the  $z$  direction is actuated by inner electrodes. A problem with the tube scanner is that the  $z$ -height varies when applying a voltage in the  $x$ - and  $y$ -direction since the axes are coupled.

Another approach is to use piezo stack actuators which only have one axis of contraction and expansion. By placing one such actuator for each axis, the lateral  $xy$ -axes and vertical  $z$ -axis becomes completely decoupled. The lateral axes are usually still

slightly coupled through a flexure setup, but far less than by using a piezo tube. Additionally, one can achieve a larger scan area with this setup.

Issues in control of piezoelectric actuators include phenomena such as creep and hysteresis nonlinearities. Additionally, they exhibit a lightly damped response to vibrations. This is a problem because it limits the bandwidth of the positioning device. High frequency reference signals will excite the vibration modes, which makes control difficult. Hysteresis and creep can be effectively compensated for if we have an accurate model of their behavior.



**Figure 2.5:** (a) Converse piezoelectric effect on a stack actuator shown by applying a voltage  $V$ . (b) Piezo tube actuator.

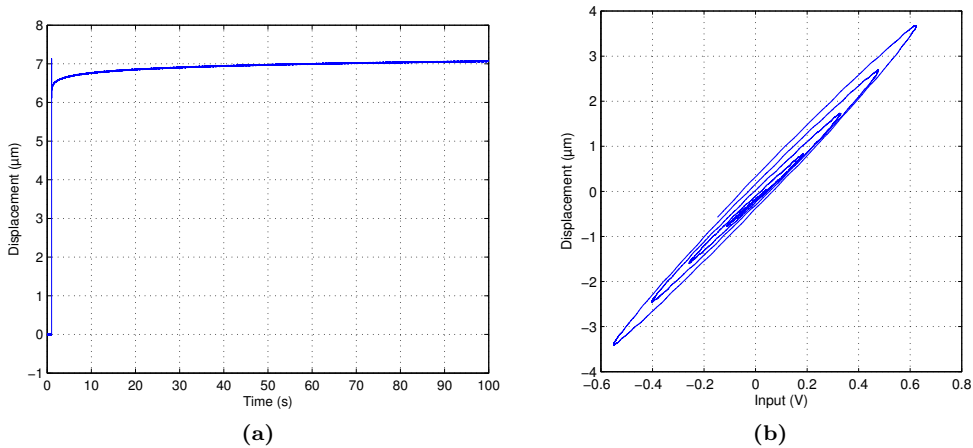
### 2.2.1. Hysteresis and Creep

Hysteresis and creep is important to consider in nanopositioning applications, as it is possibly the most dominant non-linear effect of a piezo-actuated system. We will describe these effects and some common models for them here.

Creep in piezoelectric transducers can lead to significant loss of precision in open-loop control. It is a rate-dependent phenomenon that stems from the remnant polarization changes after a voltage has been applied on the piezo element. This results in a slow creep with a time constant often in the order of minutes as seen in [Figure 2.6a](#). There are two common models in the literature [17]. The first model gives the displacement  $w(t)$  as

$$w(t) = w_o \left( 1 + \gamma \log \frac{t}{t_0} \right) \quad (2.5)$$

where  $t_0$  is the time at which the effect is apparent,  $w_0$  is the actuator displacement at time  $t_0$ , and the creep rate  $\gamma$  is a fixed constant which can be found from experimental data. Another model uses a superposition of low-pass filters with a feedthrough term,



**Figure 2.6:** The effect of creep (a) can be seen in the step-response as a slowly increasing value after the initial step. The effect of hysteresis (b) can be seen when plotting the output against the input signal, in a linear system we would get a straight line, but instead we get an ellipse. Data has been gathered experimentally on one of the lateral axes of the Park Systems XE-70 AFM.

which can be written as

$$G_c(s) = \frac{1}{k_0} + \sum_{i=1}^m \frac{1}{d_i s + k_i} \quad (2.6)$$

where  $k_0$ ,  $k_i$  are the spring constants and  $d_i$  are the damping constants. The advantage of the latter model is the ease of obtaining an inverted linear model, i.e.  $G_c^{-1}(s)$ , which can be used in the control design.

Hysteresis compensation is a more complex topic, but very well researched. Hysteresis is a nonlinear effect which means to “lag behind”, but must not be confused with phase lag which is a linear effect. It is often defined as a *rate-independent* effect as opposed to creep. Rate-independent means that it is not affected by the rate of input variations. In many ways it looks like creep, and they are not independent of each other, variations to the creep model can affect the hysteresis response and vice versa. When plotting the resulting displacement over an applied voltage, the hysteresis will show itself as an ellipse as seen in [Figure 2.6b](#). Hysteresis depends not only on the current value of the input, but also on the history of previous input values, as such it is said have internal memory. The most commonly used model of hysteresis is the classical Preisach hysteresis model [17].

### 2.3. Challenges with AFM

Although AFM has seen a wide variety of applications and been widely adopted, some of the operating challenges involved can be discouraging for some users.

**Slow scan-rates** The time it takes a commercial AFM to perform an image scan is often in the order of minutes. The slow scan-rates primarily come from the control challenges involved such as from the vibrations dynamics and nonlinearities in the piezo-elements.

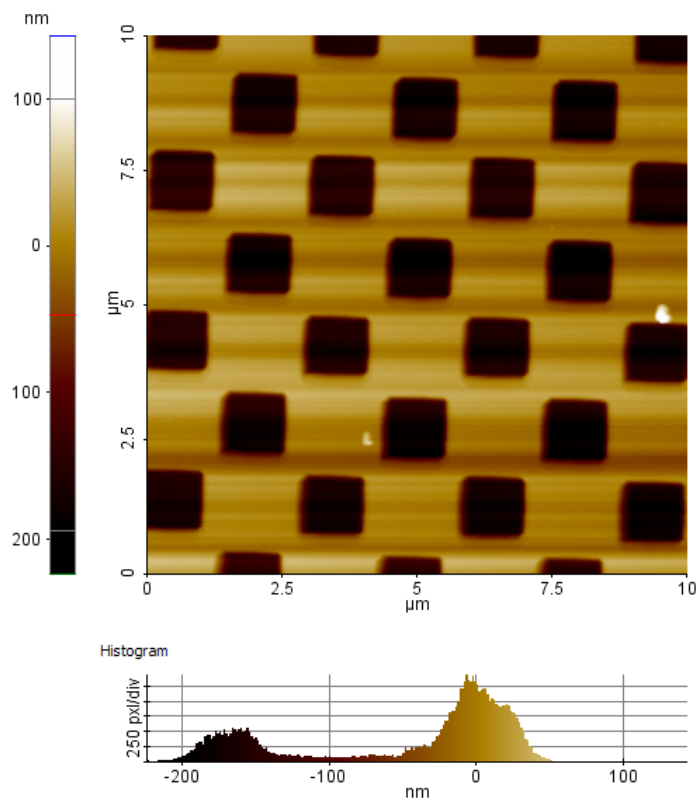
**Non-repeatability** Even when using the same parameters, results can vary from scan to scan. This may be due to damage to the probe tip, variations in temperature, or other disturbances. Additionally, changing cantilever or sample will produce different results if not properly calibrated.

**Ease of use** There are a lot of things that can go wrong when operating an AFM, such as when damaging the probe tip which produces artifacts on the image. Also, it tends to involve tweaking with control parameters to make it work correctly. Thus AFM imaging often requires a highly skilled operator.

**High maintenance** Because of limited durability of the probes, AFM requires higher maintenance than traditional microscopes.

### 2.4. Sample Scan

We performed an image scan on the AFM we have used throughout this thesis shown in [Figure 2.7](#). This image was gathered using stock software and control on a sample material with small grooves. We can see that the grooves are around 200 nm deep, and around 1  $\mu\text{m}$  wide. We can also see some small dust particles that have gathered on the surface (white).



**Figure 2.7:** Scan image using a Park Systems XE-70 AFM



## Chapter 3

# Robust Control Theory

Later in this thesis we will design several control laws and implement these for lateral control of an Atomic Force Microscope (AFM). This chapter will provide some background theory for the control laws we will employ, as well as theory needed to study the performance and stability of the system.

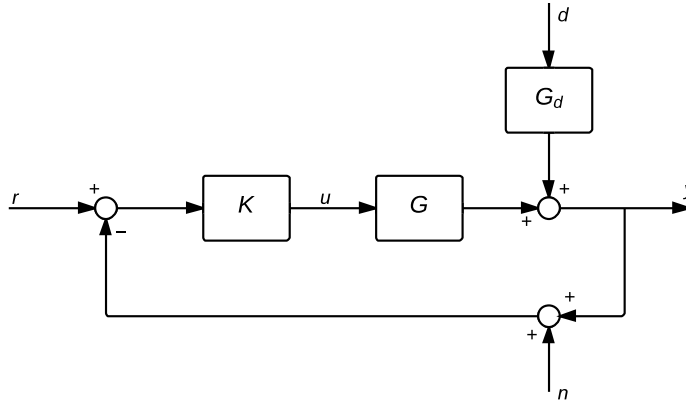
Control of AFM can be challenging because the operating conditions varies strongly over time, from interactions with sample material and other unmodeled disturbances. It is necessary that a controller is stable and performs well in all conceivable conditions. This is the primary task of robustness theory and will be presented here. Additionally, because of the importance of cross-coupling between the lateral axes of an AFM, we will base our theory on multiple-input multiple-output (MIMO) systems. A MIMO controller is able to compensate for such cross-coupling gains.

We start by introducing notation and a description of a feedback system. This allows us to properly define the conditions under which we have robust stability for a given controller. Then we will continue with presenting a variety of controllers starting with the traditional PID controller and then more modern control strategies based on  $\mathcal{H}_\infty$  control.

The theory in this chapter is based on [21, 55, 59].

### 3.1. MIMO Feedback System

This section will present notation and several important characteristics for a MIMO feedback system. Closed-loop characteristics such as the sensitivity and complementary sensitivity functions are introduced, later on we will discuss how these functions relate to tracking performance and stability of the closed loop system. Additionally, we will compare this to classical measurements of stability such as gain margin and phase margin. We will introduce the concept of singular values, a useful tool for analysis of MIMO systems.



**Figure 3.1:** Feedback control system

**Closed Loop System** Consider the block diagram in [Figure 3.1](#) of the feedback control system we will be studying with symbols as described in [Table 3.1](#). We will describe and make some observations on the closed-loop properties of this system. The plant can be written as

$$y = Gu + G_d d \quad (3.1)$$

as can be seen from the block diagram, and the plant input is given by

$$u = K(r - y - n) \quad (3.2)$$

Inserting this into [\(3.1\)](#), we can find the closed loop system description

$$\begin{aligned} y &= GK(r - y - n) + G_d d \\ (I + L)y &= Lr - Ln + G_d d \\ y &= (1 + L)^{-1} Lr - (1 + L)^{-1} Ln + (1 + L)^{-1} G_d d \end{aligned} \quad (3.3)$$

where we have used the definition  $L \triangleq GK$ , called the loop function. Note that the order of the matrices is important as matrix multiplication in general is not commutative. Let us introduce some additional definitions for the closed-loop system

$$S \triangleq (I + L)^{-1} \quad (3.4)$$

$$T \triangleq (I + L)^{-1} L \quad (3.5)$$

where  $S$  is the *sensitivity function*, and  $T$  the *complementary sensitivity function*. Notice that the property

$$S + T = I \quad (3.6)$$

always holds. This can immediately be seen from the definitions. The closed loop system [\(3.3\)](#) can now be written as

$$y = Tr + SG_d d - Tn \quad (3.7)$$

This equation is important to consider when designing a controller  $K$  whose task it is to make  $y$  track  $r$  as close as possible. It gives us some hints on the desired shapes of  $S$  and  $T$  as will be discussed in [section 3.3](#).

**Stability** One of the fundamental tasks of control theory is to design a controller which provides stability to the closed-loop system. For a single-input single-output (SISO) loop function  $L(s)$ , stability is often determined using one of the following statements [55].

- The system is stable if and only if the poles of the closed-loop system, i.e. the roots of  $1 + L(s) = 0$ , all lie in the open left half plane (LHP).
- The Nyquist stability criterion states that the Nyquist plot of  $L(s)$  needs to make as many counter-clockwise encirclements of the critical point -1 as there are number of poles in the right-half plane (RHP) of  $L(s)$ .

Similar conditions exist for MIMO systems. In classical control theory, stability is often specified in terms of gain margin and phase margin on  $L$ . These measurements give some indication of the closeness of  $L(j\omega)$  to the critical point -1, which is a good indication of robustness considering the Nyquist criterion. The actual closeness to the critical point can be found by using the  $\mathcal{H}_\infty$  norm of  $S$ , specifically [55]

$$\text{minimum distance from } L(j\omega) \text{ to } -1 = \|S\|_\infty^{-1} \quad (3.8)$$

where the notation  $\|\cdot\|_\infty$  refers to the  $\mathcal{H}_\infty$  norm as defined in [appendix A.1](#). We can also find a bound on the gain and phase margin by considering the peak of  $\bar{\sigma}(S)$  (defined next), so the flatness of this function is sometimes used as a measurement of robustness.

**Singular Values** For single-input single-output (SISO) systems, the gain from input to output is independent of the input magnitude since we have for a sine input with angular frequency  $\omega$

$$\frac{|y(\omega)|}{|u(\omega)|} = \frac{|G(j\omega)u(\omega)|}{|u(\omega)|} = |G(j\omega)| \quad (3.9)$$

This does not easily generalize to the MIMO case. One reasonable approach is to use the 2-norm on the input and output signal, but it turns out unlike the gain in the SISO case that this is not a constant value at a given frequency  $\omega$ . There is an additional degree of freedom in the input *direction*, by this we refer to the normalized vector with unit length, i.e.  $\hat{u} \triangleq u / \|u\|_2$ . A variety of measures have been proposed to represent something equivalently to gain, but for the MIMO case. The most important is possibly the maximum and minimum singular value which takes the 2-norm of the input and output signals and the worst-case and best-case directions,

$$\bar{\sigma}(G) \triangleq \max_{u \neq 0} \frac{\|Gu\|_2}{\|u\|_2} = \max_{\|u\|_2=1} \|Gu\|_2 \quad (3.10)$$

$$\underline{\sigma}(G) \triangleq \min_{u \neq 0} \frac{\|Gu\|_2}{\|u\|_2} = \min_{\|u\|_2=1} \|Gu\|_2 \quad (3.11)$$

which is constant for a given frequency  $\omega$ . When plotting MIMO transfer functions in the frequency domain we will plot both of these values in the same plot using the notation  $\sigma(G)$  to signify this. There are several other norms that serve a similar purpose, such as the Frobenium norm, sum norm, and maximum row/column norm. For more details, we refer to Skogestad and Postlethwaite [55].

**Table 3.1:** Feedback control notation

Symbol	Dimension	Description
$G$	$n \times m$	nominal plant model
$G_d$	$n \times n_d$	disturbance model
$K$	$m \times n$	controller
$r$	$n$	reference signal
$u$	$m$	plant input
$y$	$n$	plant output
$d$	$n_d$	disturbances
$n$	$n_n$	measurement noise

## 3.2. MIMO Robust Stability

The primary concern of robustness is to provide stability not only for the nominal plant  $G$ , but also in the presence of uncertainties. A simplistic approach to robustness is to only consider properties such as gain margin, phase margin, and maximum magnitude of the sensitivity function. From these properties, we can get a sense of how large deviation from the model the system can handle before instability occurs. A more rigorous approach to robustness is to make a complete specification of possible uncertainties. There are several ways to specify this, such as parameter uncertainty and plant model structure uncertainties. The set of plant models with such uncertainty is denoted  $G_p$ , and called the perturbed plant. After the uncertainty is specified we can investigate the stability for all model perturbations.

The goal of *robust stability* (RS) can be formulated as

$$\text{RS} \stackrel{\text{def}}{\Leftrightarrow} \text{System stable } \forall L_p$$

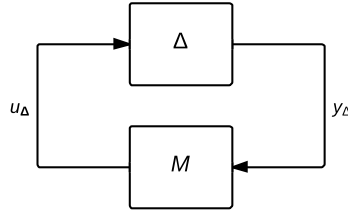
where the perturbed loop function is defined as  $L_p \triangleq G_p K$ .

### 3.2.1. Robust Stability Condition

The uncertainty perturbations are represented by a block-diagonal matrix  $\Delta$  where each element  $\Delta_i$  is a specific source of uncertainty such as parametric uncertainty, multiplicative output uncertainty etc. We define the uncertainties such that

$$\|\Delta\|_\infty \leq 1 \tag{3.12}$$

Note that we do not allow *any*  $\Delta$  that satisfies (3.12), because it must satisfy the *structure* that we have defined upon it, such as a block-diagonal matrix. If we have



**Figure 3.2:**  $M\Delta$ -structure used for robust stability analysis

no structure on  $\Delta$ , that is all elements can take any value, it is called *unstructured*. For analysis of robust stability (RS) the system must be transformed into the  $M\Delta$  structure [Figure 3.2](#) such that the uncertainty is “pulled out” of the closed-loop system.

For robust stability analysis we introduce a new value called the *structured singular value*, denoted  $\mu$ . The definition is

$$\mu(M)^{-1} \triangleq \min_{\Delta} \{ \bar{\sigma}(\Delta) \mid \det(I - M\Delta) = 0 \text{ for structured } \Delta \} \quad (3.13)$$

which in short can be described as finding the smallest structured  $\Delta$  that makes the matrix  $I - M\Delta$  singular. We can now state the robust stability condition taken verbatim from Skogestad and Postlethwaite [[55](#)] who also provides a proof.

### Robust stability for block-diagonal perturbations

Assume that the nominal system  $M$  and the perturbations  $\Delta$  are stable. Then the  $M\Delta$ -system in [Figure 3.2](#) is stable for all allowed perturbations with  $\bar{\sigma}(\Delta) \leq 1$ ,  $\forall \omega$ , if and only if

$$\mu(M(j\omega)) < 1, \quad \forall \omega \quad (3.14)$$

Note that in the case of an unstructured  $\Delta$ , we have  $\mu(M) = \bar{\sigma}(M)$ . In fact, it is shown in Skogestad and Postlethwaite [[55](#)] that if we have an unstructured  $\Delta$  the necessary and sufficient condition for robust stability ([3.14](#)) is reduced to

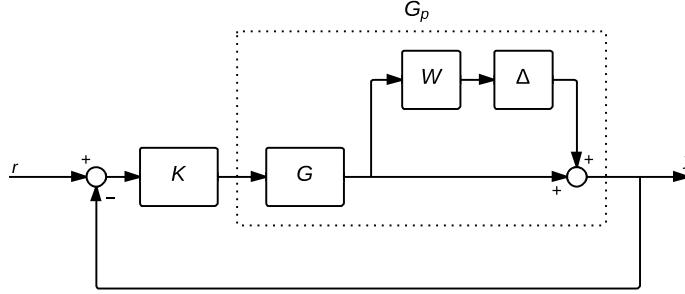
$$\text{RS} \Leftrightarrow \|M\|_{\infty} < 1 \quad (3.15)$$

with otherwise equivalent assumptions.

### 3.2.2. Multiplicative Output Uncertainty

There are many classes of structured uncertainties, and we will focus on one of them called multiplicative output uncertainty. This can be seen in [Figure 3.3](#), mathematically we have

$$G_p = (I + \Delta W)G \quad (3.16)$$



**Figure 3.3:** Feedback system with multiplicative output uncertainty

where  $W$  is the weighting, a fixed stable scalar transfer function. We assume that  $\Delta$  is unstructured which simplifies our calculations. Now that we have properly specified a set of perturbed plants in a mathematically convenient way (3.16), we want to find the conditions for which the system is stable for all perturbations.

We begin by transforming the feedback system with multiplicative output uncertainty in Figure 3.3 into the  $M\Delta$ -formulation to see when (3.15) is satisfied. This is done by finding the closed-loop transfer function between the input and output of  $\Delta$ . We have

$$\begin{aligned} y &= y_{\Delta} - GK y \\ (I + GK) y &= y_{\Delta} \\ y &= (I + GK)^{-1} y_{\Delta} \end{aligned} \quad (3.17)$$

where  $u_{\Delta}$  is the input signal to  $\Delta$  and  $y_{\Delta}$  is the output signal from  $\Delta$ . Furthermore

$$\begin{aligned} u_{\Delta} &= -WGK y \\ u_{\Delta} &= -WGK (I + GK)^{-1} y_{\Delta} \end{aligned} \quad (3.18)$$

$$u_{\Delta} = M y_{\Delta} \quad (3.19)$$

where the last two equations give us

$$\begin{aligned} M &= -WGK (I + GK)^{-1} \\ &= -WT \end{aligned} \quad (3.20)$$

Inserting (3.20) into (3.15) gives us necessary and sufficient condition for robust stability,

$$\text{RS} \Leftrightarrow \|WT\|_{\infty} < 1 \quad (3.21)$$

Next we will describe how to find  $W$  given a plant model  $G$  and a specified set of expected perturbations assuming they can be described as multiplicative output uncertainty.

### 3.2.3. Specifying the Weight $W$

Let us consider that we have a specification of all the possible perturbed plants that we would like to show robustness against, denoted by the set  $\Pi$ . We would like to find a weighting function  $W$  such that  $G_p$  in (3.16) complies with all the models in  $\Pi$ . Some algebra is needed to find a solution to this problem,

$$\begin{aligned} G_p &= (I + \Delta W)G \\ \Delta W &= (G_p - G)G^{-1} \\ |W(j\omega)| &\geq \bar{\sigma}((G_p(j\omega) - G(j\omega))G^{-1}(j\omega)) \quad \forall \omega, G_p \in \Pi \end{aligned} \quad (3.22)$$

where the last inequality comes from considering that  $\|\Delta\|_\infty < 1$ . For convenience we introduce the notation

$$\hat{W}(\omega) \triangleq \max_{G_p \in \Pi} \bar{\sigma}((G_p(j\omega) - G(j\omega))G^{-1}(j\omega)) \quad (3.23)$$

Now the weighting function  $W(s)$  can be found as any transfer function that satisfies

$$|W(j\omega)| \geq \hat{W}(\omega), \quad \forall \omega \quad (3.24)$$

which is easily seen by considering (3.22) and (3.23).

In summary, we start by selecting a nominal model  $G(s)$ . The physical plant will never behave exactly like this model, so specify a set  $\Pi$  of possible plants that the real plant may behave like. This can be specified either from experiments, or from other knowledge of the system. Then, let  $G_p$  take on all of these possible plants and solve for  $\hat{W}$  in (3.23). Finally specify a transfer function  $W(s)$  such that (3.24) is satisfied. If additionally (3.21) is satisfied for some controller being considered, it means our system is robustly stable for all  $G_p \in \Pi$ .

## 3.3. Control Design Objectives

The overall objective in control theory can be stated as providing inputs to a system such that the outputs behave with desired response. There are several considerations to take into account as we will see here. Let us restate the closed loop system (3.7) for convenience,

$$y = Tr + SG_d d - Tn \quad (3.25)$$

Let us also define the tracking error as

$$e \triangleq y - r \quad (3.26)$$

By studying [Figure 3.1](#) we can find the transfer function from  $r$  to  $e$  to be

$$e = Sr \quad (3.27)$$

There are a few things to note by considering (3.25) and (3.27). Since we want  $y$  to track  $r$ , we can see that this is accomplished by having  $T = I$ . To reduce the

tracking error we would like  $S = 0$ , which is convenient considering the constraint  $S + T = I$ . However, to attenuate the measurement noise  $n$  we would like  $T$  to be as small as possible. These conflicting goals can be handled by observing that each goal dominates in some frequency domain, while is negligible in other domains. The measurement noise is usually most prominent at high frequencies, while the tracking goal and disturbances are usually given at lower frequencies.

This discussion provides some insight into the desired shapes of  $S$  and  $T$ . The magnitude of  $T$  should start at 0 dB for good tracking performance, but above some frequency point we want  $T \ll 0$  dB to attenuate measurement noise. Since  $S + T = I$ , the shape of  $S$  must start such that  $S \ll 0$  dB and increase until it flattens out at around the same frequency point.

The goals can be summarized as follows when also considering control effort and multiplicative output uncertainty. The list is restated from Skogestad and Postlethwaite [55, p.342].

1. For disturbance rejection, make  $\bar{\sigma}(S)$  small
2. For noise attenuation make  $\bar{\sigma}(T)$  small
3. For reference tracking make  $\bar{\sigma}(T) \approx \underline{\sigma}(T) \approx 1$
4. For input usage (control energy) reduction make  $\bar{\sigma}(KS)$  small
5. For robust stability in the presence of multiplicative output perturbation, make  $\bar{\sigma}(T)$  small

A useful measurement of the closed-loop characteristics is *bandwidth* which in short tells us the frequency range the controller is able to track sufficiently well. We will introduce two different methods of measuring bandwidth on a closed-loop system, one depends on  $S$  while the other depends on  $T$ .

#### Definition: Closed-loop bandwidth

The bandwidth of  $S$ , stated as  $\omega_{bS}$ , is the lowest frequency where  $\bar{\sigma}(S)$  crosses the -3 dB mark from below.

The bandwidth of  $T$ , stated as  $\omega_{bT}$ , is the lowest frequency where  $\underline{\sigma}(T)$  crosses the -3 dB mark from above.

A lot of information about the system can thus be gathered by considering the plots of  $\sigma(S)$  and  $\sigma(T)$ . As well as the point at which  $S$  and  $T$  rises and falls, the flatness of these functions are also considered to be good measurements of the performance and robustness of the system. For these reasons, plots are provided of  $\sigma(S)$  and  $\sigma(T)$  in several cases throughout the thesis.



## 3.4. PID Controller

Thus far we have concerned ourselves with providing a framework such that we can specify a controller and plant, and analyze for stability and performance. The rest of the chapter will present control laws both from traditional and modern control design. We start by introducing the classical PID controller.

A PID controller can be written on the form

$$K_{PID} = K_p + \frac{1}{s}K_i + K_d s \quad (3.28)$$

where  $K_p$ ,  $K_i$ , and  $K_d$  are the proportional (P), integral (I), and derivative (D) constant parameters respectively.

A PID controller considers only one axis at a time. For MIMO systems, we can use one such controller for each axis if the cross-coupling gains are sufficiently small. For a 2-by-2 plant, such a controller can be written in matrix form as

$$K = \begin{bmatrix} K_{PID,1} & 0 \\ 0 & K_{PID,2} \end{bmatrix} \quad (3.29)$$

where  $K_{PID,1}$  and  $K_{PID,2}$  are of the form (3.28) possibly with different parameters.

### 3.4.1. Tuning PID Controller

A PID controller is made up of three constants that need to be properly tuned for the closed-loop system to behave well in terms of stability and performance. The most common approach of tuning the PID controller is possibly the Ziegler-Nichols method. This is an experimental method where no previous knowledge about the model has to be known. We will provide a quick summary of the method here.

#### Ziegler-Nichols method

Consider the controller in (3.28). Set  $K_i = K_d = 0$ , then increase  $K_p$  from zero until the system oscillates with a constant amplitude. This value of  $K_p$  is called the ultimate gain  $K_u$ , and the oscillation period is  $T_u$ . Set the PID parameters such that

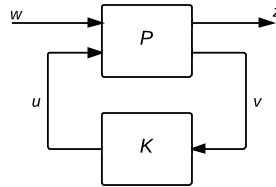
$$K_p = 0.6K_u, \quad K_i = 2\frac{K_p}{T_u}, \quad K_d = 0.125K_p T_u \quad (3.30)$$

Variations to these parameters are available for different performance characteristics, see e.g. McCormack and Godfrey [40].

For a lightly damped system, such as the nanopositioning stage, the system becomes unstable with even small proportional gains. This makes the use of Ziegler-Nichols

method impractical. Instead we will tune the controller by shaping the loop transfer function  $L = GK$  of the system as well as the closed-loop transfer functions  $S$  and  $T$ . This requires a model of the system, but the advantage is that we can immediately determine the nominal stability properties. In general, for good closed-loop performance, we want the loop to have high gain at low frequencies and small gains at high frequencies. For a lightly damped system, we emphasize the use of the integral part of the PID controller to dampen the resonant peak of the plant.

### 3.5. $\mathcal{H}_\infty$ Control



**Figure 3.4:** General control configuration

Traditional control strategies are mainly concerned with shaping the loop  $L$  of the feedback system by optimizing properties such as gain margin and phase margin. The term  $\mathcal{H}_\infty$  control is used for a variety of control strategies, but they all have in common that they try to optimize for some  $\mathcal{H}_\infty$  norm as is defined in [appendix A.1](#).

Consider the control configuration in [Figure 3.4](#), and let  $N$  be the transfer function from  $w$  to  $z$ , then the general  $\mathcal{H}_\infty$  optimal control problem is to find all stabilizing controllers  $K$  which minimize

$$\|N\|_\infty \quad (3.31)$$

An optimal solution can be challenging to find, but iterating on suboptimal solutions can be solved efficiently e.g. by the algorithm presented by Doyle et al. [20]. In order to use this approach, we need to find a suitable  $P$  to describe our objective. This problem formulation is very general and many problems can be reduced to it, we will now present one such formulation called mixed sensitivity  $\mathcal{H}_\infty$  control.

#### Mixed sensitivity $\mathcal{H}_\infty$ control problem

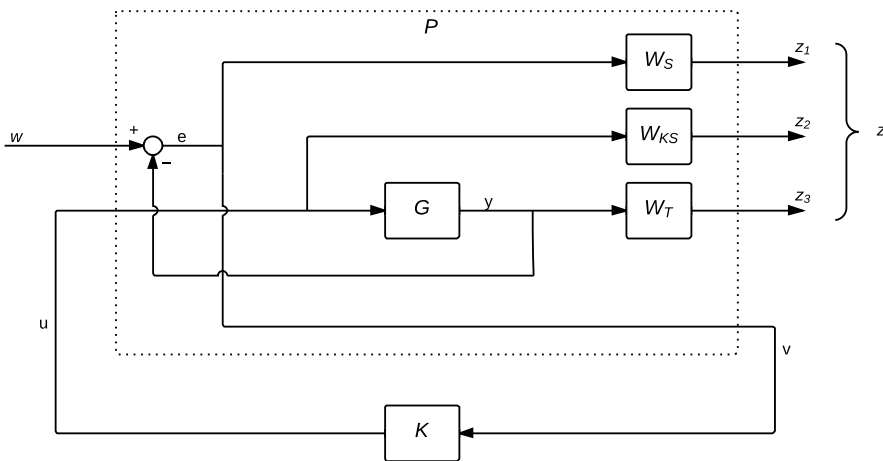
Given a plant  $G$ , find the controllers  $K$  which solves

$$\min_K N(K) = \left\| \begin{array}{c} W_S S \\ W_T T \\ W_{KS} K S \end{array} \right\|_\infty \quad (3.32)$$

where  $W_S$ ,  $W_T$ , and  $W_{KS}$  are user-defined scalar transfer functions for weighting the respective closed-loop functions.

The function  $KS$  is the closed-loop transfer function from  $r$  to  $u$ , thus minimizing this can be seen as reducing the control effort or input energy. We would like the various closed-loop functions to be small or large in different frequency ranges, which is the reason for having frequency dependent weighting functions. The discussion in [section 3.3](#) can be used as a reference for designing these functions. The absolute value of each of the weightings does not matter as long as they are scaled similarly, i.e. multiplying all of them by a constant does not change the optimization problem. Often one of the elements of the problem are omitted, that is either  $S$ ,  $T$ , or  $KS$ . This can be convenient to avoid having too many options to tune.

To solve the mixed sensitivity problem, the problem must be converted to the general  $\mathcal{H}_\infty$  optimization problem. The block diagram of the resulting structure can be seen in [Figure 3.5](#) which is adapted from [56].



**Figure 3.5:** General control configuration of the mixed sensitivity  $\mathcal{H}_\infty$  problem.

### 3.5.1. Procedure for Mixed Sensitivity $\mathcal{H}_\infty$ Control

The procedure for obtaining a robustly stable  $\mathcal{H}_\infty$  mixed sensitivity controller can be summarized as follows, assuming the uncertainty can be well described by multiplicative output uncertainty. Assume we already have a plant model  $G$  identified using experimental data  $\hat{G}(\omega)$ .

1. Calculate  $\hat{W}(\omega)$  in (3.23), and find a  $W(s)$  satisfying the condition (3.24)
2. Construct the weightings in the mixed sensitivity optimization problem (3.32)
3. Solve the optimization problem using a tool such as Matlab with the command `mixsyn`
4. Verify that the robust stability condition (3.21) is satisfied and that the closed-loop characteristics are reasonable by considering the discussion in [section 3.3](#)

### 3.5.2. Other $\mathcal{H}_\infty$ Control Approaches

The mixed sensitivity control strategy is a powerful tool for obtaining a robustly stable controller. However, because of the various weightings each with their own parameters, it can sometimes be difficult to tune.

For more flexibility in the control design, a method based on Glover and McFarlane [32] often called  $\mathcal{H}_\infty$  loop-shaping design or the Glover-McFarlane method can be utilized. It is based on classical loop shaping, and can possibly employ an existing controller designed for a given system from traditional control theory. This method then shapes the loop using an algorithm to make it more robustly stable by taking into account multiplicative perturbation to the nominal plant model. One advantage to this method is that it requires no tuning, and the algorithm is relatively simple. One could for example start by designing a PID controller using standard methods, and then run this algorithm to make the system more robust.

Another method is signal based  $\mathcal{H}_\infty$  control. The goal here is to find a controller which optimizes for some signal such as the 2-norm of the error signal. By considering stochastic disturbance and noise signals, we acquire in its simplest form a more traditional Linear Quadratic Gaussian (LQG) controller. We can additionally put frequency dependent weightings on the various signals and a description of the uncertainty model. This is not a standard  $\mathcal{H}_\infty$  optimization problem, but can be solved using  $\mu$ -synthesis [55]. However, this has not been solved for the general case, and when a solution does exist, it often results in a high-order controller. Additionally, the problem often does not always converge and the problem can be difficult to formulate. For these reasons this method has not been widely employed.

## Chapter 4

# Control Strategies for Lateral Motion

In the topic of nanopositioning a distinction is often made between control in the lateral direction (along  $x$ - and  $y$ -axis) and vertical direction (along  $z$ -axis). We can see that this is useful for instance in the case of Atomic Force Microscopy (AFM). Here the sample is moved in the lateral direction, while the probe is moved up and down in the vertical axis using very different control schemes. In this chapter we will present some of the literature on the topic of control in the lateral directions. We will give a brief introduction on the various types of controllers utilized in the literature on nanopositioning, from very simple ones to more complex.

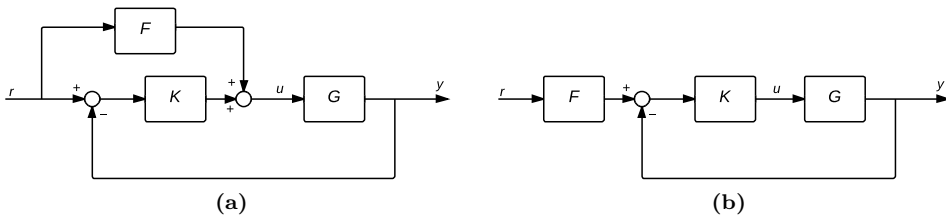
The performance of a controller is usually measured in terms of resolution, tracking performance, and bandwidth. Resolution is important for being able to image down to the smallest scales, possibly down to atomic scales. Good tracking performance is also important as the probe scans over the surface so the resulting image does not become corrupted. The bandwidth provides a limit to how fast the probe can scan a given area and is important for capturing the state of a sample with fast changing conditions, as well as the patience and time consumption of the operator.

Usually linear control strategies are employed since the system is dominantly linear. Nonlinear effects such as creep and hysteresis are often associated with the use of piezoelectric actuators, and some control schemes compensate specifically for this.

Traditional nanopositioning control has to account for (1) hysteresis, (2) creep, and (3) vibration. The vibration dynamics results in large magnitude resonance peaks, which tends to give low gain margin because of the rapid phase-drop at around the peak. The difficulty is amplified by large model uncertainties. Uncertainties can originate both from the mechanical system itself and also the varying conditions of a Scanning Probe Microscopy (SPM) application. The system response usually varies with different setpoints and with wear over time. Additionally, the sample will affect the mass of the system, and the interaction between the probe and sample will also

affect the response. Cross-coupling between the axes needs also to be taken into account, even when employing different positioners for different axes because they will ultimately all be connected through the sample.

We will divide the presentation of control strategies into the two schemes, feedforward control, and feedback control. In general the feedforward scheme can achieve high resolution and high bandwidth, while the feedback scheme can give good tracking. The feedback controller is limited in its bandwidth because it needs to guarantee robust stability, and additionally the controller increases the noise level in the system. The feedforward controller on the other hand does not need to worry about stability and is not affected by sensor noise, but will not give good tracking performance because of unmodeled system characteristics and disturbances. By combining the two types as shown in Figure 4.1, we can get the performance of a feedforward controller and accuracy of the feedback controller. It is however useful to separate them when investigating their individual properties. A survey of control issues in the field of nanopositioning can be found in [17].



**Figure 4.1:** Combined feedforward and feedback controller schemes. In (a) the input  $u$  is augmented with a feedforward signal, in (b) the reference trajectory is filtered. If the reference trajectory is known in advance  $F$  can possibly be an acausal filter.

## 4.1. Feedforward Control

The simplest form of controller is a feedforward signal where the reference signal is multiplied with the DC-gain of system from input to output. This scheme is very limited in terms of accuracy and bandwidth, because higher harmonics of the reference signal will excite the resonance peaks of the system. Additionally effects such as creep and hysteresis are ignored. This motivates the usage of more complex control schemes. A review of feedforward control in nanopositioning can be found in [13].

### 4.1.1. Creep and Hysteresis Compensation

Creep and Hysteresis is important to consider when using piezoelectric actuators as is used in most AFM applications. There are several models of hysteresis and creep proposed in the literature as seen in section 2.2. Once a model is established, an adaptive method can be used to estimate the parameters. Then a feedforward

compensation can be made by inverting the estimated models, see some examples on this in [16, 23, 42]. For a more thorough review of hysteresis compensation, see [17].

### 4.1.2. Inversion Techniques

Inversion techniques can be employed to reduce mechanical vibrations of the system. If the mechanical system can be approximated by a minimum-phase transfer function  $G(s)$  a feedforward signal can easily employ the inverted model  $G^{-1}(s)$  ideally for perfect tracking. In practice however, there will be model and parameter uncertainties, as well as other disturbances which will affect the tracking performance. Usage of these inversion techniques can be found in [14, 15]. Some methods also modifies the trajectory to avoid input saturation and excitation of undesirable frequencies such as [27, 39, 53], which can be viewed as an acausal filter on the reference signal. A challenge with the acausal filter is that the reference trajectory will need to be specified before-hand. Dealing with this issue, where the trajectory is only partially known, is done by [60]. This paper also discusses dealing with non-minimum-phase systems which is challenging because we can't invert the model directly as this will lead to unstable internal modes in the controller. This issue is also dealt with in [51].

## 4.2. Feedback Control

Although feedforward control can achieve high resolution and high bandwidth, many applications also require good tracking performance. A feedback controller is necessary to achieve this. We usually distinguish between model-based and fixed-order, fixed-structure control. We will start by presenting controllers in the latter category, such as a PID controller. Then we will present model-based  $\mathcal{H}_\infty$  control.

### 4.2.1. PID

The simplicity of a PID controller has made it the default choice in feedback control. For lateral control in nanopositioning, we can find this controller employed in [1, 52]. A PID controller is very limited in bandwidth because of the lightly damped nature of the positioner, which makes it quickly become unstable at higher gains. Secondly, the integrator part of the controller is prone to wind-up due to saturation. There are however methods to overcome this, such as presented in [25]. One possible way to increase bandwidth of a PID controller, is to augment it with notch-filters to reduce the sharp resonant peaks [43, 44]. Other control schemes can also be used to dampen the resonant peak of the system, before applying a more traditional control law [28, 41, 45].

In scanning applications with a triangle reference signal, the controller is often augmented with an additional integrator making what is often designated as a PIID controller. This is used to be able to asymptotically track ramp signals, i.e. the flanks of the triangle signal often used in raster scan applications.

### 4.2.2. $\mathcal{H}_\infty$ Loop Shaping

Because of the changing nature of the nanopositioning stage, a feedback controller will need to ensure that it is stable for all variations to the system. A PID controller does not give very good robustness properties, but it is possible to robustify the controller using the  $\mathcal{H}_\infty$  loop shaping technique. Traditional loop-shaping methods or simple controllers such as PID can be used at first, and then by applying the method a controller will be generated which stabilizes the system for a given family of perturbed plants. This method has been applied to nanopositioning devices by [36, 54]. The robustness may come at some cost to the performance.

### 4.2.3. $\mathcal{H}_\infty$ Mixed Sensitivity

$\mathcal{H}_\infty$  control has the advantage that it shapes the closed-loop characteristics of the system, thus we don't have to deal with finding a reasonable loop function manually as in traditional loop-shaping methods. Additionally, it is formulated in such a way that it is convenient to adjust the weightings such that we can guarantee robust stability. This control scheme has been used extensively in the topic of nanopositioning [36, 48, 49, 51, 52, 54, 58].

## 4.3. Other Control Approaches

Many applications of nanopositioning use a repetitive reference signal such as the raster pattern in imaging. This is very well suited for a repetitive control approach which has been implemented by [3, 23]. It works by embedding a model of the periodic signal in the control loop. Such an approach can be combined with another feedback scheme and does not affect the closed-loop stability.

Iterative Learning Control is a more computationally expensive method, but can be efficient against repeating disturbances and/or repeating reference trajectories [9, 10, 33, 37].

Adaptive control theory such as model reference adaptive control (MRAC) has also been studied such as in [23, 24]. Lyapunov based robust adaptive control has been studied in [5].

Nanopositioning can be considered a good test-case for a large part of control theory because of the difficulties due to the nonlinearities, vibration dynamics, uncertainties, and varying conditions involved. As we have seen, this has made nanopositioning a well-studied field with a vast amount of control strategies, and this chapter only touches on some of the available literature on the subject.



## Chapter 5

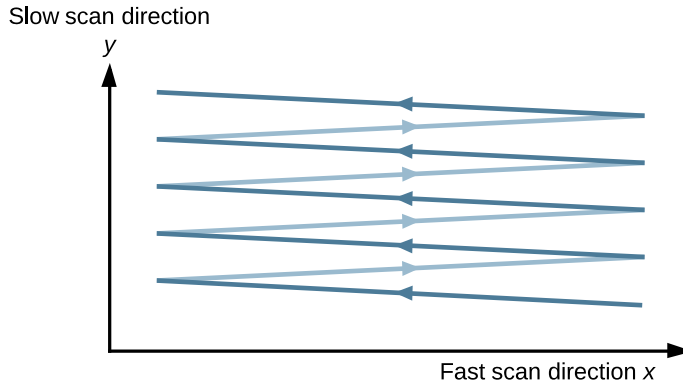
# Comparison of Control Laws on Cross-Coupling Effects

Many nan positioning applications has a coupling between the axes especially in the lateral directions, but sometimes also in the vertical axis. This cross-coupling can be of high importance in applications such as scanning probe microscopy (SPM). This can be understood by considering the scanning motion of the microscope which has one fast direction and one slow direction, as illustrated in [Figure 5.1](#). It is important that the fast scanning direction does not move the probe in the slow direction as this could position the probe over the wrong scan-line in the  $y$ -direction and produce artifacts in the resulting image. Thus any cross-coupling between  $x$  and  $y$  should be as small as possible, especially considering that the fast scan direction speed can be of several orders of magnitude larger than the slow direction.

In the literature, control laws are often designed based on independent axis design where control laws are designed for one axis at a time such as in [\[54, 58\]](#). In this approach the cross-coupling gains are considered to be small compared to the same-axis coupling and just treated as a disturbance. The advantages of using independent axis design is that the control design becomes simpler and enables more traditional control laws. Additionally, a MIMO controller tends to become more computationally complex such that it is more difficult to implement for real-time control. The primary concern of this chapter will be to investigate how much there is to gain by using a MIMO approach for a typical nan positioning application even when the cross-coupling gain is small, and whether independent axis design can be justified.

We will design a  $\mathcal{H}_\infty$  MIMO controller for the lateral positioning of an Atomic Force Microscope (AFM). An equivalent  $\mathcal{H}_\infty$  SISO controller is also designed based on independent axis design. By comparing these two control laws, we can determine how large the differences are particularly in terms of cross-coupling. The two  $\mathcal{H}_\infty$  controllers are also compared to a simple PID controller. The three controllers are experimentally tested on a commercial AFM, and the results are discussed both in terms of cross-coupling reduction and same-axis performance with emphasis on the

former. The purpose is to be able to give better insight to the question of whether the extra complexity both in terms of control design and computational complexity of a MIMO based controller is worth the performance benefits.



**Figure 5.1:** A common scanning motion in SPM. The  $x$ -axis must move at a much faster rate than the  $y$ -axis. Any cross-coupling from the  $x$ -axis to the  $y$ -axis may adversely affect the resulting image.

## 5.1. Experimental Setup

All experiments are done using a commercial AFM of the type Park Systems XE-70, pictured in [Figure 5.2](#). In this device, the sample is placed on a parallel kinematic 2d flexure scanner for motion in the horizontal  $xy$ -plane. Motion along the vertical  $z$ -axis is done by moving the cantilever itself using a flexure guided setup. In such a setup the coupling between the vertical axis and the lateral axes is none at all, except for small forces via the sample. However, the two lateral axes will be slightly coupled due to the flexure setup.

The signals from the AFM are routed to an electronic processing and controller box that is available for the microscope. As well as having its own controller circuits, it provides access to analog measurements from the sensors. It can also receive external signals for manual control of the AFM's actuators which we will use to control the piezoelectric elements.

See a schematic overview of the setup in [Figure 5.3](#). The controllers are implemented in a Simulink model, which is compiled and transferred to a dedicated computer, an xPC, which runs a real-time operating system. The xPC performs the numerical calculations, and is externally connected to a digital-to-analog converter (DAC) and an analog-to-digital converter (ADC). These input-output signals are run through anti-aliasing and reconstruction filters, which are constructed as low-pass filters with a bandwidth of less than half the sample frequency.

For our purpose, we have overridden control of the piezo-actuators in the  $x$ - and  $y$ -axes. This is connected to a PiezoDrive PDL200, a linear voltage amplifier. The input into this amplifier is considered as the input to the system. The voltage output from the distance sensors in the  $x$ - and  $y$ -axes located on the AFM is used as the output of the system.

### 5.1.1. Frequency Domain Response

Several of the measurements gather frequency response data of the system, both in open-loop and closed-loop with the controllers. These are done using a Stanford Research Systems SR780 frequency analyzer. For the open-loop case, it is connected directly to the inputs and outputs of the system. In the closed-loop case, the device is connected to the xPC which uses the output signal from the SR780 as the reference signal.

The SR780 was set up to provide a white noise source signal. The device then performs a fast Fourier transform of the output signal from the system to obtain the frequency response. This approach was chosen over other methods such as a swept sine signal, which is a single sine that increases its frequency over time. Such a signal may cause damage because of the lightly damped poles of the system which will generate very high gains around the resonance frequencies. Additionally, one avoids the transient effects from the frequency-changing sine signal.



**Figure 5.2:** Park Systems XE-70

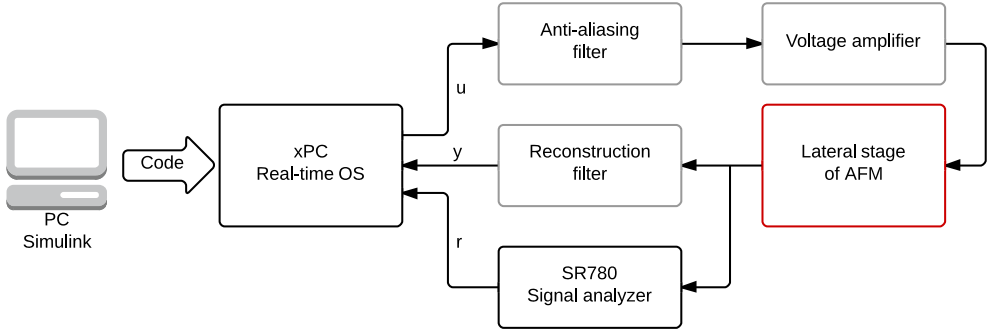


Figure 5.3: Experimental Setup Overview

## 5.2. System Identification

When designing the controllers, especially for the  $\mathcal{H}_\infty$  controller, we need a model of the system. Such a model is represented by the transfer function matrix  $G(s)$ , a 2-by-2 matrix, one dimension for each axis of the platform. The purpose of this section is to identify such a model of the system.

In order to obtain a system model, we will start by getting a frequency response using the SR780 device. This gives us  $\hat{G}(\omega)$ , the experimental analogue to  $G(s)$ . Each element of the 2-by-2 matrix  $\hat{G}(\omega)$  contains a set of (frequency, complex response) pairs. The complex response value can be converted to magnitude and phase, and plotted in a bode plot. Using tools available in Matlab, a transfer function approximation to the experimental data can be obtained using any desired degree, to obtain a nominal plant  $G(s)$ . Keep in mind that any such function will only be an approximation of the linear response of the real system.

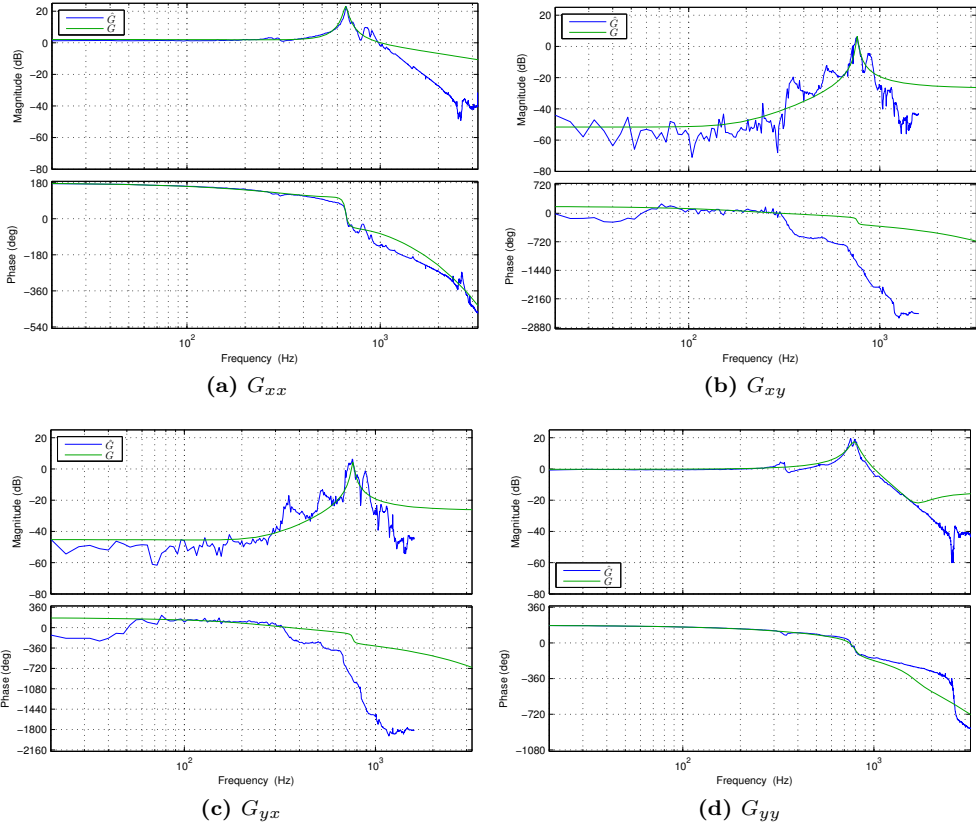
We gathered a total of three different sets of frequency responses at different set-points, amplitudes and at different times. These additional responses are used to make sure the system is robustly stable for all expected variations to the system response.

### 5.2.1. Frequency Response $\hat{G}(\omega)$ and Fitted Model $G(s)$

One of the frequency responses is plotted in Figure 5.4 together with the fitted models. The transfer functions were fitted using the Matlab function `tfest` on the experimental data. The diagonal elements of  $G(s)$  were approximated by a third-degree transfer function, while the off-diagonal elements were approximated by a second-degree function. We tried to keep the order as low as possible to reduce the complexity of the resulting controller. The nominal plant model was found as

$$G(s) = e^{-4.58e-04s} \begin{bmatrix} \frac{-5924s^2 - 1.709e07s - 9.878e10}{s^3 + 4703s^2 + 1.82e07s + 7.806e10} & \frac{-0.04567s^2 + 69.72s - 6.043e04}{s^2 + 104.5s + 2.29e07} \\ \frac{-0.04705s^2 + 89.17s - 1.253e05}{s^2 + 134.1s + 2.288e07} & \frac{-8708s^2 + 2.618e07s - 9.214e11}{s^3 + 3.865e04s^2 + 4.379e07s + 9.44e11} \end{bmatrix}$$

where the exponential term represents the time delay between input and output, see next section on how this was identified. We can see that the phase starts at  $180^\circ$  which means that the system has an inverse response, i.e. positive inputs give negative outputs and vice versa. This is just the sign convention of our raw data, and we decided not to change it for simplicity.



**Figure 5.4:** Experimental frequency response of the system  $\hat{G}(\omega)$  for both axes including cross-terms, and the corresponding fitted models  $G(s)$

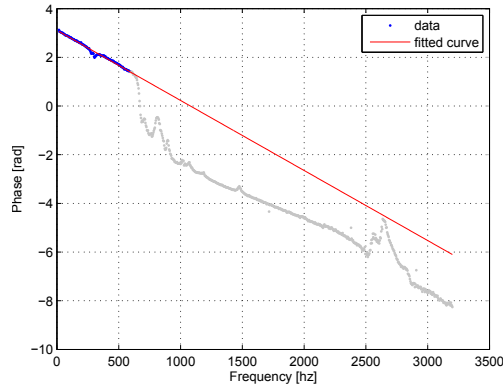
### 5.2.2. Time Delay

We experienced that there was a notable time-delay between input and output of the plant. A possible source of this time delay may be due to the displacement sensor on the AFM's  $xy$ -scanner. It is convenient to know the time delay as this is important for stability, and our general understanding of the system. Additionally, it can let us correct the phase of a system because the time-delay affects the phase plot.

A time delay will present itself as a linear reduction of the phase as a function of frequency. Thus, we may find the time delay of the system between input and output

by taking a look at the phase plot of the elements of  $\hat{G}$ . By assuming that the change in phase at lower frequencies is dominated by the time delay and other sources of phase change is close to zero, we find that the time delay is proportional to the slope at the start of the phase plot. This can be justified by considering the phase plot of a system with a high resonant peak where the phase starts dropping rapidly at around the resonance frequency, while only slowly decreases up until this point. See [Figure 5.5](#) where we have used linear regression to find the slope of the line. The time delay is found to be

$$T_d = 4.58 \times 10^{-4} \text{ s}$$



**Figure 5.5:** Identification of time delay. We are assuming that the linear reduction of the phase at the beginning is purely caused by the time delay. The slope of the fitted line (red) is then proportional to the time delay. Only data points up to 636 Hz (blue) are used for the linear regression.

### 5.2.3. Uncertainty weight

We want to make sure that the system is robustly stable for variations in the system responses and inaccuracies in the model fits. To do this, we must first calculate  $\hat{W}(\omega)$  as in (3.23) for our experimentally gathered frequency responses, i.e.

$$\hat{W}(\omega) = \max_{\hat{G} \in \Pi} \bar{\sigma} \left( \left( \hat{G}(\omega) - G(j\omega) \right) G^{-1}(j\omega) \right) \quad (5.1)$$

using all three of the gathered frequency responses  $\hat{G}(\omega) \in \Pi$ . Then we will fit the output uncertainty weighting  $W(s)$  such that  $|W(j\omega)| > \hat{W}(\omega)$  for all  $\omega$  in accordance with (3.24).

The weighting was fitted by the transfer function

$$W(s) = 3.8254 \frac{(s + 210)(s + 1850)}{(s + 2400)(s + 3200)} \quad (5.2)$$

which is plotted together with  $\hat{W}$  in Figure 5.6. We can see that  $\hat{W}$  has some spikes, e.g. one at around 300-400 Hz. This is because the low-order model fit does not pick up the changes in gain around this frequency range, as evident in Figure 5.4. A higher-order model fit could possibly pick up these changes and reduce the peaks of  $\hat{W}$ , and thus allow us to more accurately fit  $W$  around  $\hat{W}$ . This will of course cost more in terms of complexity, so ultimately it comes down to a trade-off between complexity and accuracy. Both of these will in the end affect the performance of the final controller. When the controllers are designed in the next section, we can prove robust stability of the closed-loop system by showing that  $\|WT\|_\infty < 1$  as explained in section 3.2 where  $T$  is the closed-loop complementary sensitivity function.

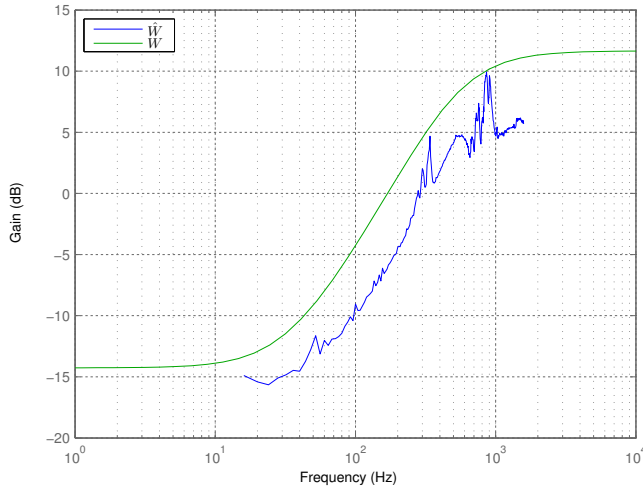


Figure 5.6: Robustness fit,  $W(s)$  and  $\hat{W}(\omega)$

## 5.3. Controller Design

This section will present the design of the PID- and the two  $\mathcal{H}_\infty$  controllers. We will explain the reasoning behind how they were designed and present the closed-loop characteristics resulting from the controllers based on the identified model  $G$ .

### 5.3.1. PID controller

Two PID controllers were designed, one for the  $x$ -axis, and one for the  $y$ -axis. They were designed by looking at the frequency response of the open-loop and closed-loop transfer functions, as well as some experimentation. The constants for the two axes are set to:

$$\begin{aligned} x\text{-axis:} \quad & K_p = -5 \times 10^{-3}, \quad K_i = -370, \quad K_d = -2.5 \times 10^{-5} \\ y\text{-axis:} \quad & K_p = -5 \times 10^{-3}, \quad K_i = -450, \quad K_d = -2.5 \times 10^{-5} \end{aligned}$$

We can see that the system is dominated by the integral term, this is because the resonant peak of the system needs to be dampened out sufficiently. In fact, the proportional and derivative term could have been eliminated completely which would result in only small changes to the closed-loop performance.

Additionally, a low-pass filter was added to the derivative term to make the controller proper, with a time constant equal to the sample time. The resulting controller transfer function  $K$  can then be written as

$$K(s) = \begin{bmatrix} \frac{-0.555s^2 - 495s - 9.25e06}{s^2 + 2.5e04s} & 0 \\ 0 & \frac{-0.555s^2 - 575s - 1.125e07}{s^2 + 2.5e04s} \end{bmatrix} \quad (5.3)$$

The closed-loop sensitivity and complementary sensitivity function is shown in [Figure 5.7a](#).

### 5.3.2. $\mathcal{H}_\infty$ Mixed Sensitivity Controllers

We will design two  $\mathcal{H}_\infty$  controllers, one MIMO version and one SISO version. We will use the same weighting functions for both of them so that they are more easily comparable. We will also design the weighting functions such that the controllers are similar to the PID controller in terms of bandwidth.

In [section 3.5](#) the  $\mathcal{H}_\infty$  mixed sensitivity problem is formulated as

$$\min_K N(K) = \left\| \begin{array}{c} W_S S \\ W_T T \\ W_{KS} K S \end{array} \right\|_\infty \quad (5.4)$$

where  $W_S$ ,  $W_T$ , and  $W_{KS}$  are user-defined weightings, we will now explain how these were chosen for our control design. We have chosen  $W_S$  to be a first-order transfer function of the form

$$W_S = \frac{\frac{1}{M_S}s + \omega_b}{s + A\omega_b} \quad (5.5)$$

where  $\omega_b$  is a desired bandwidth, the scalar  $M_S$  is the inverse gain, and  $A$  is a small constant which exists to avoid an integrator which makes the mixed sensitivity problem ill-conditioned. Increasing  $M_S$  will allow large values of  $S$  at high frequencies more. It can be thought of as the relative weighting on  $S$  from those on  $T$  and  $KS$ , allowing a more undesirable  $S$  while making the other closed-loop functions more desirable.

We have chosen  $W_T$  to be equal to  $W$  in (5.2). The reasoning behind this is to make the controller more easily robustly stable. From (3.21) we have that the system is robustly stable iff  $\|WT\|_\infty < 1$ . Selecting  $W_T = W$  shapes the controller to more easily comply with this criteria.

Finally, the weighting  $W_{KS}$  is chosen as a first order high-pass filter of the form

$$W_{KS} = \frac{1}{M_{KS}} \frac{s}{s + \omega_b} \quad (5.6)$$



where  $M_{KS}$  is the inverse gain, and  $\omega_b$  again is the desired bandwidth, possibly the same as in  $W_S$ . This will punish very fast input variations applied to the actuators. A commonly used physical interpretation of this is the reduction of energy usage by the controller.

After some experimentation, we ended up with the values  $\omega_b = 2\pi \cdot 70$ ,  $A = 10^{-4}$ ,  $M_s = 1.2$ , and  $M_{KS} = 1.2$ . The resulting weighting functions are summarized in [Table 5.1](#).

The  $\mathcal{H}_\infty$  mixed sensitivity problem was solved using the Matlab function `mixsyn`, the complete code can be seen in [appendix C.1](#). For the MIMO controller we used the nominal model matrix  $G$  as input to the function, which resulted in a state-space model controller with 18 states. For the SISO controller we designed two controllers, each of them only calculated using the corresponding axis model, i.e.  $G_{xx}$  for the  $x$ -axis controller and  $G_{yy}$  for the  $y$ -axis controller. Each of them resulted in a controller with 7 states, and they were combined for a total number of 14 states.

The resulting sensitivity and complementary sensitivity functions  $S$ , and  $T$  for all three controllers are plotted in [Figure 5.7](#). We can observe that the PID controller has two notches in both  $S$  and  $T$  at around 665 Hz and 770 Hz. The first corresponds with the frequency of the peak magnitude of  $G_{xx}$  while the second corresponds with the peaks of  $G_{xy}$ ,  $G_{yx}$ , and  $G_{yy}$ . The PID controller does not take these peaks of the model into account and instead it simply dampens everything over its bandwidth. The  $\mathcal{H}_\infty$  SISO controller has that same second notch at 770 Hz. This is probably from the fact that it doesn't take into account the peak in the cross-coupling axis, but is effective at damping out the resonant peak of the same axis, which it has a model of. The  $\mathcal{H}_\infty$  MIMO controller has none of these notches, which indicates that it effectively takes the cross-coupling effects into account since it contains a model of the cross-coupling as well.

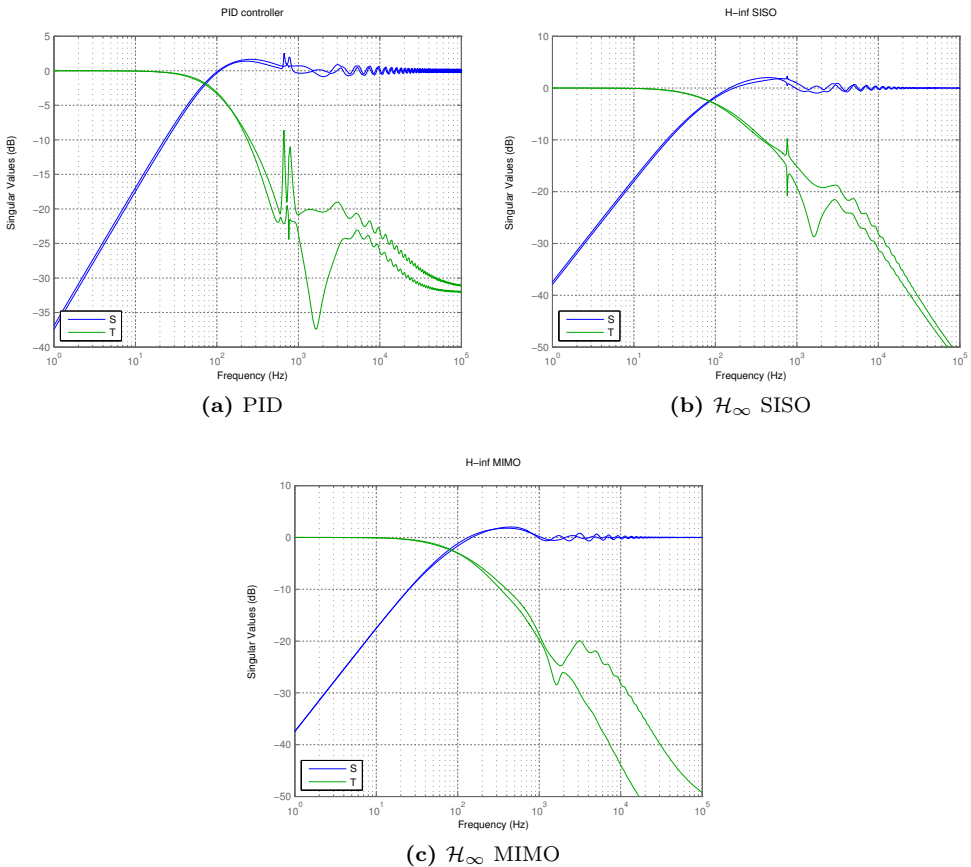
The bandwidth of all three controllers, measured on both  $S$  and  $T$ , is given in [Table 5.2](#). We have used the definition of bandwidth as defined in [section 3.3](#). The table shows us that the closed-loop system is robustly stable because  $\|WT\|_\infty < 1$  for both of the  $\mathcal{H}_\infty$  controllers. This is not the case for the PID controller, so in fact we can not guarantee that this controller is stable for all perturbations of the system.

**Table 5.1:** Summary of weighting transfer functions

$W_S(s)$	$\frac{0.8333s + 439.8}{s + 0.04398}$
$W_T(s)$	$3.8254 \frac{(s + 210)(s + 1850)}{(s + 2400)(s + 3200)}$
$W_{KS}(s)$	$0.8333 \frac{s}{s + 439.8}$

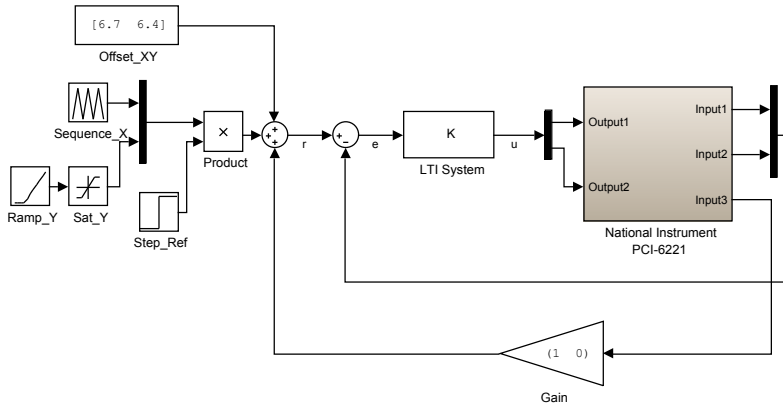
**Table 5.2:** Bandwidth, robustness, and implementation comparison between the three controllers. TF = transfer function, SS = state space.

	$\omega_{bS}$ [Hz]	$\omega_{bT}$ [Hz]	$\ WT\ _\infty$	Implementation
PID	58.0	93.1	1.073	4th order TF
$\mathcal{H}_\infty$ SISO	75.6	96.4	0.9938	14th order SS
$\mathcal{H}_\infty$ MIMO	69.8	98.6	0.6717	18th order SS



**Figure 5.7:** Sensitivity  $\sigma(S)$  and complementary sensitivity  $\sigma(T)$  for the three controllers. The identified plant  $G(s)$  with time-delay has been used in the calculations.

## 5.4. Implementation Details and Issues



**Figure 5.8:** Simulink diagram of the controller implementation. The bottom gain enables external reference signal, such as from the SR780. The source blocks on the left provides an optional raster pattern reference signal.

Three different controllers were designed in the previous section, and we will now explain some of the details and associated issues experienced with implementing and running these on the experimental equipment.

The controllers were added as either state-space or transfer function models to a Simulink diagram as seen in [Figure 5.8](#). This model was compiled and transferred to the xPC where it was executed.

The  $\mathcal{H}_\infty$  controllers gave us some issues. The controller wouldn't run in the first implementations because the xPC issued "CPU overload"-errors. The controller was too complex to be solved in real-time at the specified step-size. We reduced the step-size until we didn't get the error anymore, but this resulted in an unstable controller because of the large steps. We found that by reducing the bandwidth of the controller (by changing the mixed sensitivity weightings), and increasing the solver complexity we could finally get a working and stable controller. The PID controller did not give us the same amount of problems.

At first we had some issues with noise, but this was reduced by introducing an anti-aliasing and reconstruction filter in the loop.

Since the displacement sensor gives us a value in voltage, we would like to convert this to distance. The relationship between sensor voltage output and distance was found by running the AFM at some specified amplitude using the included software and measuring the resulting voltage from the sensor. The relationship was found to be linear, so the problem was reduced to a simple linear regression between distance and voltage output.

See a summary of the implementation details in [Table 5.3](#).

**Table 5.3:** Controller implementation details

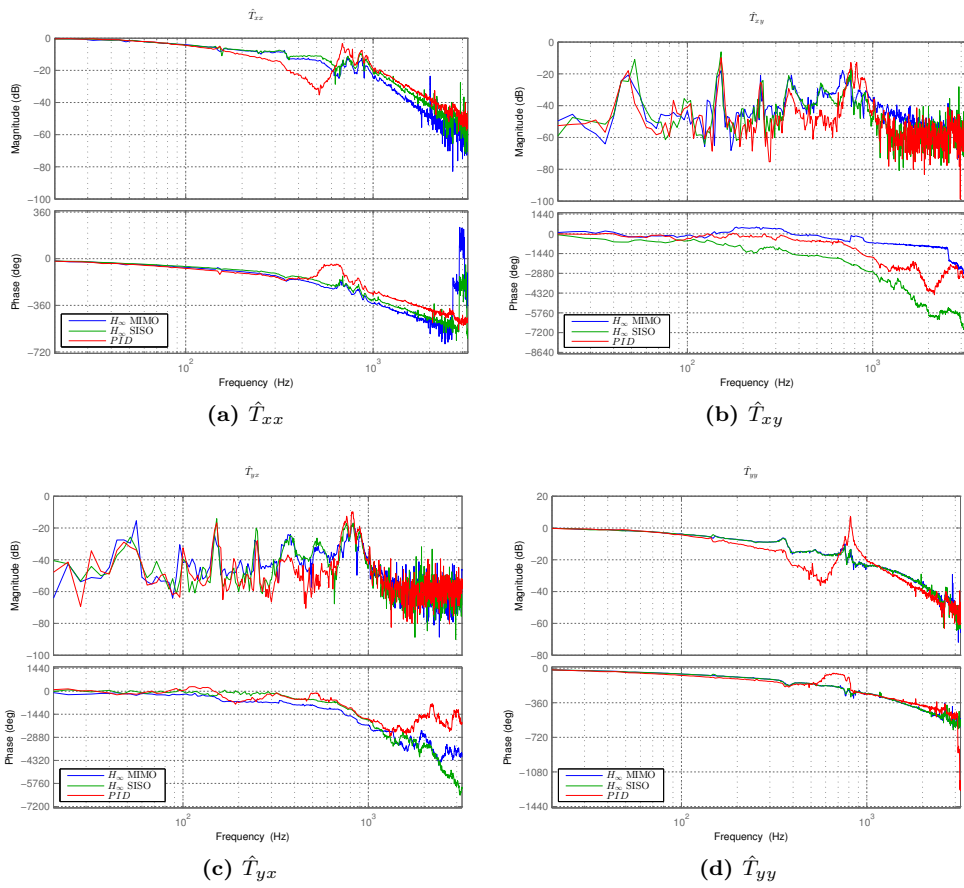
Step-size	$4 \times 10^{-5}$ s
Solver type	ode8 (Dormand-Prince)
Anti-aliasing filter type	2nd order Butterworth (low-pass)
Anti-aliasing filter bandwidth	10 kHz
Reconstruction filter type	2nd order Butterworth (low-pass)
Reconstruction filter bandwidth	10 kHz
Displacement/sensor voltage-ratio	4.66 $\mu\text{m}/\text{V}$

## 5.5. Results

We measured the closed-loop frequency responses  $\hat{T}_{xx}$ ,  $\hat{T}_{xy}$ ,  $\hat{T}_{yx}$ , and  $\hat{T}_{yy}$  for each of the controllers using the SR780 device. The results from these is plotted in [Figure 5.9](#). We have plotted  $\sigma(\hat{T})$  together with  $\sigma(T)$  for each controller so we can easily compare the expected response with the actual response in [Figure 5.10](#).

Finally, we ran the positioner in a raster pattern typically used in imaging applications with all three controllers. We scanned over an area of  $2 \times 2 \mu\text{m}$  at various different frequencies. We have plotted the results for 10 Hz and 100 Hz in [Figure 5.11](#) and [Figure 5.12](#) respectively. Here we can see the movement along the  $x$ -axis,  $y$ -axis, as well as an  $xy$ -plot of the scan.

The standard deviation in the error signal from all scans are given in [Table 5.4](#). Standard deviation was chosen instead of real mean square because this would be dominated by the stationary error, which is less interesting for our purposes. Only the middle 60% of the scanning range is considered when measuring the standard deviation, and only samples when moving in the positive direction of the  $x$ -axis.



**Figure 5.9:** Comparison of experimental closed loop frequency response,  $\hat{T}$ , for both controllers and along both axes.

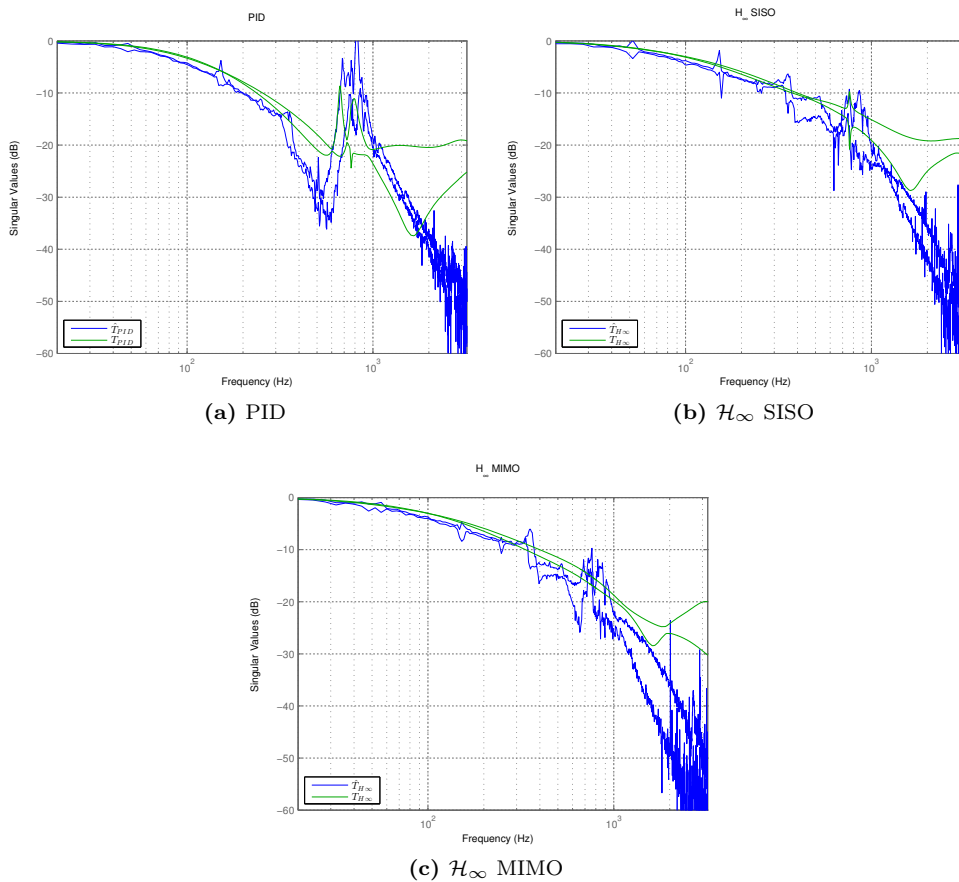
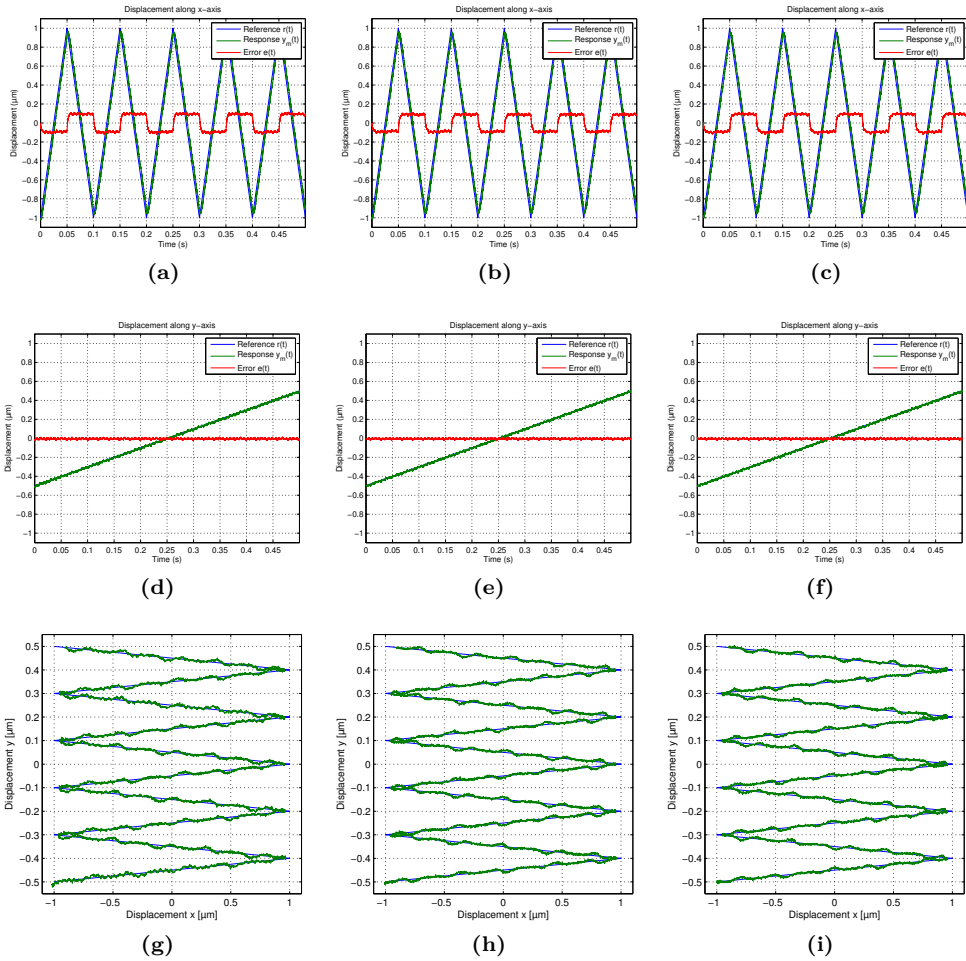
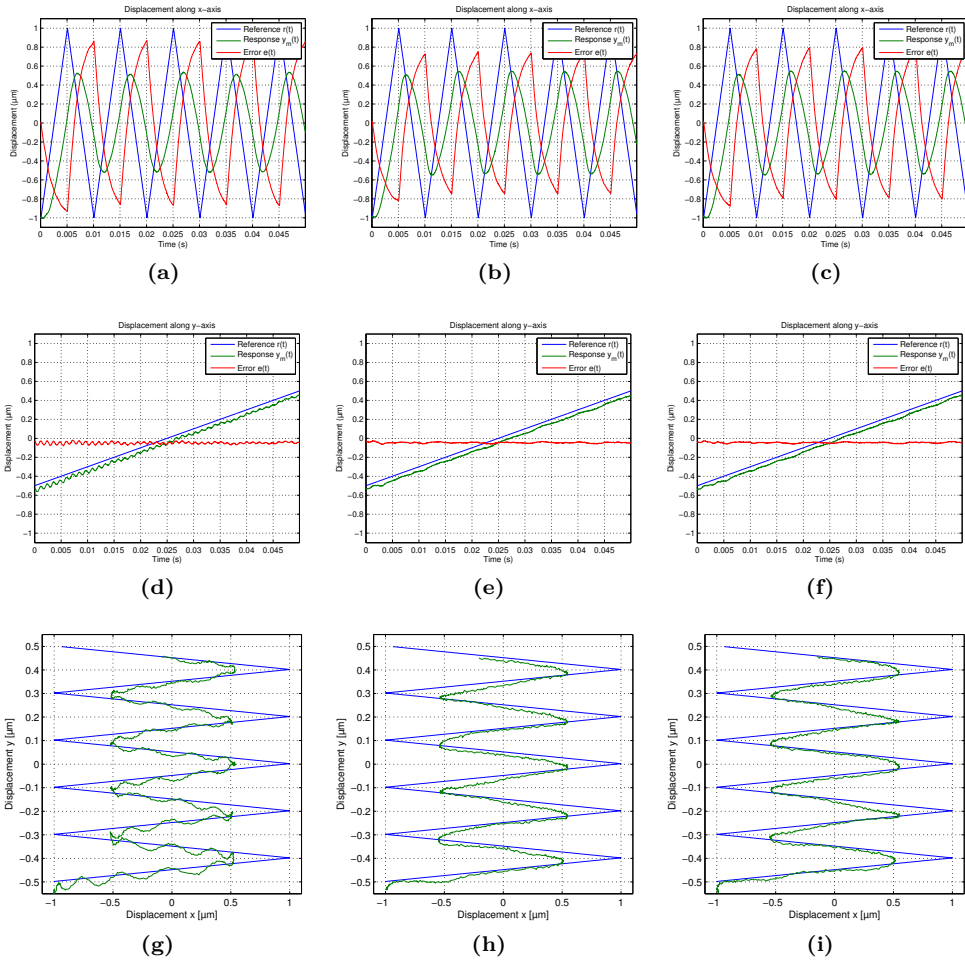


Figure 5.10: Comparison of  $\sigma(\hat{T})$  versus  $\sigma(T)$  for each controller.



**Figure 5.11:** Scanning motion output at 10 Hz. Left column PID-controller, middle column  $\mathcal{H}_\infty$  SISO, right column  $\mathcal{H}_\infty$  MIMO. First row gives the displacement along the  $x$ -axis, the second row the displacement along the  $y$ -axis, and the last row the  $xy$ -postion.



**Figure 5.12:** Scanning motion output at 100 Hz. Left column PID controller, middle column  $\mathcal{H}_\infty$  SISO, right column  $\mathcal{H}_\infty$  MIMO. First row gives the displacement along the  $x$ -axis, the second row the displacement along the  $y$ -axis, and the last row the  $xy$ -position.



**Table 5.4:** Performance comparison of the three controllers, with a triangle reference signal on the  $x$ -axis and a constant forward speed on the  $y$ -axis. The values in the  $\text{std}(x)$  columns are colored gray for readability.

	PID		$\mathcal{H}_\infty$ SISO		$\mathcal{H}_\infty$ MIMO	
Rate	$\text{std}(x)$	$\text{std}(y)$	$\text{std}(x)$	$\text{std}(y)$	$\text{std}(x)$	$\text{std}(y)$
[Hz]	[nm]	[nm]	[nm]	[nm]	[nm]	[nm]
10	7.283	5.870	6.574	5.248	7.098	5.360
20	8.121	5.606	10.36	5.723	10.75	5.832
50	102.9	6.885	93.95	3.704	96.68	5.900
100	288.3	8.729	224.4	7.913	243.5	3.607
200	424.6	14.16	363.6	13.12	404.0	11.70
400	422.3	59.58	468.0	53.54	475.5	53.53
662	412.0	122.1	429.3	99.92	405.9	101.3
760	434.1	165.3	447.9	117.7	412.4	122.6

## 5.6. Observations

We introduced this chapter with the primary goal of investigating the differences in cross-coupling performance of a SISO controller and a MIMO controller, and whether we could justify independent axis design. In addition to discussing this, we will also make some observations on the more general differences between the PID controller and  $\mathcal{H}_\infty$  controllers.

**Same-axis performance** For the same-axis couplings, it is clear from  $\hat{T}_{xx}$  and  $\hat{T}_{yy}$  in Figure 5.9 that although it has roughly the same bandwidth, the PID controller is inferior to the  $\mathcal{H}_\infty$  controllers. Especially along the  $y$ -axis, it has a very large gain at around the resonance frequency. This was even higher than expected when the PID controller was designed, and is probably because of varying conditions of the characteristics of the system. The  $\mathcal{H}_\infty$  controllers tackle these changes more gracefully, and have a much flatter response along the entire frequency range.

**Cross-coupling performance** We can see that the open-loop gain of  $\hat{G}_{xy}$  peaks at over 6 dB in Figure 5.4. This is pretty substantial compared to the low-frequency gain of  $\hat{G}_{xx}$  at  $\sim 1$  dB. If we ran a triangle reference signal using a simple feedforward controller, this could result in a large error on the perpendicular axis due to the cross-coupling gain being activated by the higher harmonics of the triangle signal.

The closed-loop cross-coupling gains  $\hat{T}_{xy}$  and  $\hat{T}_{yx}$  however have been damped significantly for all the controllers as seen in Figure 5.9 staying below -10 dB at around the resonant peak. At lower frequencies on the other hand, the closed-loop gains has some significant peaks which are larger than the open-loop gains at the same frequency. These were unexpected and from some other runs, it seems that the gains of these peaks varied a lot with different set-points and at different times. So these differences are thought to be due to varying conditions from when the experiments were run.

In Table 5.4 we can see the standard deviation in the error on the  $x$ - and  $y$ -axis signals for various reference frequencies. We will mostly pay attention to the  $y$ -axis (slow axis) because the reference signal along this axis is linear, so any cross-coupling from the  $x$ -axis should be easily noticeable. We can see that the controllers are close to identical at the 10 Hz triangle reference, perhaps with a slight disadvantage to the PID controller. This trend continues roughly through all the frequencies. The two  $\mathcal{H}_\infty$  controllers perform identical for all practical purposes. The small differences that are seen are so small that they could be from the different noise suppression characteristics. Especially for the PID controller, we can see that it has some large spikes in  $\hat{T}$  at higher frequencies which mean it will be worse at suppressing sensor noise at these frequencies compared to the  $\mathcal{H}_\infty$  controllers.

It is clear from the closed-loop cross-coupling performance that all controllers, including the ones based on independent axis design, dampens the cross-coupling gains very effectively. We can see that the cross-coupling resonant peaks are located at

high frequencies, whereas at low frequencies the cross-coupling is nearly zero. Since our controllers don't operate at such high frequencies they don't seem to excite the high gain cross-coupling frequencies, which results in the SISO controllers operating as well as the MIMO controller. This shows that the usage of independent axis design is justified, and perhaps even advantageous because of the complexities involved with MIMO control.

**Experimental data vs. nominal model** The notches found on the analytical  $\sigma(S)$  and  $\sigma(T)$  in Figure 5.7 indicates that the  $\mathcal{H}_\infty$  MIMO controller might perform slightly better with its flatter response. But this performance is not noticeable in the experiments. This is possibly because the variations and noise in the response of the physical plant is larger than these notches anyway. Additionally, the inaccuracies from the model fit are also more significant than these notches.

When looking at  $\sigma(\hat{T})$  in Figure 5.10 we can see that all the controllers roughly follow their respective model  $\sigma(T)$  especially at lower frequencies. At higher frequencies, the PID controller has the largest error which is expected. But the  $\mathcal{H}_\infty$  controllers also have some errors with rises and falls, some of which can be attributed to the simplified model fit. For instance in Figure 5.4 at around 300 - 500 Hz, we can see that  $G_{yy}$  diverges pretty significantly from  $\hat{G}_{yy}$  with a fast rise and fall. This rise and fall is also found in  $\hat{T}_{yy}$ , but not seen in  $T$  because we did not model it in. If our goal was to have a controller with high bandwidth as possible, this could become significant as the rise and fall would ruin the flatness of  $\hat{T}$  at these frequencies. Thus in general a controller with higher bandwidth will need to be more complex.

**Closing remarks** Part of the question we set out to answer was whether or not we can ignore the cross-coupling properties of the system. In the open-loop response, cross-coupling is over 6 dB at the largest, which is substantial. Although this occurs at higher frequencies, higher-order harmonics of the reference signal could possibly excite this frequency range. Additionally, we have to consider the commonly applied scanning motion where the  $x$ -axis has a much higher amplitude than the  $y$ -axis, so even small cross-coupling gains may produce high levels of noise in the  $y$ -axis. So in feedforward control, one should be careful about ignoring cross-coupling effects.

The nanopositioner platform we have used in these experiments is for a commercial AFM, and is marketed to have very low cross-coupling. There are certainly platforms with much worse characteristics in terms of cross-coupling, so these will have a stronger case for the choice of a MIMO controller.

From the scanning motion in Figure 5.11 we can clearly see that there is a stationary error. Our goal was not to create a best performing controller, so this was beyond our scope to deal with. But we will mention that this is usually solved in one of two ways. For the PID controller, it is common to add another integral term, resulting in a PIID controller or just PII. Such an integrator could also be added to the  $\mathcal{H}_\infty$  controllers, but this does not fit naturally into the design flow of it. The second method is to augment the feedback controller with a feedforward signal.

Some high-frequency components are clearly visible in the error on  $x$  and  $y$  for all the controllers as seen in Figure 5.11 and Figure 5.12. Some simulations with white noise added to the feedback signal seem to replicate this error. This indicates that it is caused by sensor noise which gets integrated by the controller possibly reinforced by the time delay, although feedback stops it from drifting far.

## 5.7. Conclusion

For the same-axis performance the  $\mathcal{H}_\infty$  controllers are both superior to the PID controller as expected. They compensate for the resonant peaks of the system much better. However, there are no significant differences between the two  $\mathcal{H}_\infty$  controllers.

In terms of cross-coupling reduction, the  $\mathcal{H}_\infty$  MIMO controller seemed to give a slight advantage over the  $\mathcal{H}_\infty$  SISO controller for our system in the analytical results. This small advantage however was negligible compared to noise and disturbances as shown by the experimental results. Even the simple PID controller seems to dampen out the dominant cross-coupling effects from the open-loop response just as well as the other two. Thus, our experiments show that the added complexity from a MIMO controller, both in terms of the control design and computational complexity, does not provide adequate benefits compared to an independent axis design.

We may still expect different results under one of the following two conditions:

1. *Using a controller with higher bandwidth.* The errors in our model ( $T$  vs  $\hat{T}$ ) was to a large extent due to the inaccuracies in the low-order model fit. In a controller with higher bandwidth, these inaccuracies will become more decisive for the performance since they will not be dampened out as much. This can be dealt with by using a higher-order controller and taking into account the cross-coupling effects. Although, at some point the variations in the model response and nonlinearities will dominate the model fit accuracy.
2. *With a system that has worse cross-coupling characteristics.* The system we used has a particularly low cross-coupling gain, this will not be the case for all devices.

## Chapter 6

# Control Order Reduction

In the previous chapter we designed three different control laws for real-time implementation on the experimental setup. The most complex controller, the  $\mathcal{H}_\infty$  MIMO approach, resulted in a controller with 18 states. Some issues were experienced while implementing this controller as the computational complexity was very demanding for our hardware. The selection of solver type and step-size was crucial for being able to run such a complex controller without overloading the hardware and at the same time providing a step-size small enough for numerical stability.

In this chapter we will investigate the possibilities of reducing the complexity of the controller such that it becomes easier to implement. In the literature on nanopositioning, model order reduction is extensively used to simplify the controller [18, 35, 38, 51]. Details on how this is performed and the implications for performance and stability is often omitted in the literature. We will show how to perform such reduction and the effects of this on stability and performance at various reduced orders, starting with presenting theory on model order reduction. In the next chapter we will extend this analysis by also investigating the effect on computational complexity. We will use the  $\mathcal{H}_\infty$  MIMO controller from the previous chapter and base our discussion around the results of reducing this controller. Both simulation and experimental results are provided for this purpose.

Essentially, there are three paths to a low-order controller if we start with a higher-order model of the plant using model-based control laws such as  $\mathcal{H}_\infty$  control. The first method is to reduce the plant model first, and then design a controller. The second method is to design a high-order controller first and then perform model reduction on the controller at the end. The third method is to use some other direct method from plant model to controller. It is argued by Anderson [2] that the first method should not be employed as the approximations you do as a first step gets carried through all the calculations. We will start with the high-order  $\mathcal{H}_\infty$  MIMO controller and reduce this directly, although it could be argued that we should have provided an even higher-order model fit when first designing our controllers.

A few words about notation. The closed-loop characteristics  $S$  and  $T$  uses the iden-

tified  $G$  and the  $\mathcal{H}_\infty$  MIMO controller  $K$  as found in the previous chapter. The term  $K_r$  may be used as a collective term for all the reduced controllers, or a controller of a specific order. The usage should be obvious from the context. The closed-loop characteristics for the reduced order controller  $K_r$  are denoted  $S_r$  and  $T_r$ , and uses the same identified plant model  $G$ . Throughout the chapter we use the term (model) order, which is the number of states in a *minimal realization*, sometimes called the McMillan degree.

## 6.1. Background Theory

We will present two different approaches to model order reduction based on material by Skogestad and Postlethwaite [55] and Zhou et al. [59]. Our primary purpose is to provide background material in order to reduce the order of the controller from the previous chapter. The theory presented here can be used for any transfer function whether it is a plant or a controller. But when reducing a controller it is important to also consider the changes made to the closed-loop characteristics, not just the controller itself.

We will start by explaining what a balanced realization means and how we can transform a transfer function into such a form. Once we have a balanced realization of the model, we will show two different methods of model order reduction, namely *model truncation* and *model residualization*. We will explain the differences between them and briefly mention other methods which exist in the literature. Later in this chapter, we will apply this theory on one of the controllers from the previous chapter to investigate how performance and stability is affected after model reduction.

### 6.1.1. Balanced Realization

A *realization* is the process of transforming a transfer function  $G(s)$  into state-space form, i.e. the set of equations

$$\dot{x} = Ax + Bu \tag{6.1}$$

$$y = Cx + D \tag{6.2}$$

which will be denoted by  $(A, B, C, D)$ . A realization is *minimal* if it has the minimal number of states with the same input-output behavior as the transfer function. The process of realization is non-unique and can take many forms, one of which is the balanced realization. We will quote the definition from Skogestad and Postlethwaite [55].

**Definition: Balanced realization**

Let  $(A, B, C, D)$  be a minimal realization of a stable, rational transfer function  $G(s)$ , then  $(A, B, C, D)$  is called balanced if the solutions to the following Lyapunov equations

$$AP + PA^T + BB^T = 0 \quad (6.3)$$

$$A^T Q + QA + C^T C = 0 \quad (6.4)$$

are  $P = Q = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n) \triangleq \Sigma$ , where  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n > 0$ .

The  $P$  and  $Q$  matrices are often called the controllability and observability *Gramians* respectively. Since they are equal, a balanced system can be said to be as controllable as it is observable. The  $\sigma_i$ 's are called the *ordered Hankel singular values* of  $G(s)$  as defined in [appendix A.2](#). The Hankel singular values are closely related to the  $\mathcal{H}_\infty$  norm with the relationship [55]

$$\max_i \sigma_i \leq \|G(s)\|_\infty \leq 2 \sum_{i=1}^n \sigma_i \quad (6.5)$$

In Matlab this can be performed using the command `balreal` which requires the control systems toolbox. This returns both the new balanced realization as well as the Hankel singular values.

### 6.1.2. Reduction Problem Formulation

Given a model  $G(s)$ , the overall goal of model reduction is to find a new lower order model  $G_r(s)$  such that the two models are close in some regard. There are several ways to formulate this closeness, one reasonable objective is to use the  $\mathcal{H}_\infty$  norm of the error such that

$$\|G(s) - G_r(s)\|_\infty \quad (6.6)$$

becomes as small as possible. For reduction of controllers, it is also reasonable to consider the closed-loop error

$$\|T(s) - T_r(s)\|_\infty \quad (6.7)$$

Both model truncation and model residualization methods work by removing states from the balanced realization of the model  $G(s)$ . Thus it is convenient to formulate the state-space equations for the original model and resulting reduced model which we will do next.

Consider the balanced realization  $(A, B, C, D)$  of  $G(s)$  partitioned such that the  $n$ -dimensional state vector  $x$  is parted into  $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$  where  $x_2$  is the state vector which we

would like to remove, of dimension  $n - k$ . Let the state-space equations for the system be given as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u \quad (6.8)$$

$$y = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + D \quad (6.9)$$

Our objective is to reduce this model to the state-space model of order  $k$  given as

$$\dot{x}_1 = A_r x_1 + B_r u \quad (6.10)$$

$$y = C_r x_1 + D_r \quad (6.11)$$

while keeping the input-output relationship as close as possible to the original model. With this formulation, we only need to find reasonable candidates for the  $A_r, B_r, C_r,$  and  $D_r$  matrices while keeping the objective (6.6) and/or (6.7) in mind. The model truncation and model residualization method each provides their own solution to this.

### 6.1.3. Method 1: Model Truncation

The model truncation method solves the model reduction problem simply by removing the  $x_2$  states and all terms related to this vector from (6.8)-(6.9). The resulting reduced state-space model ( $A_r, B_r, C_r, D_r$ ) of order  $k$  is given by the matrices

$$A_r = A_{11}$$

$$B_r = B_1$$

$$C_r = C_1$$

$$D_r = D$$

which can be inserted into the state-space equations (6.10)-(6.11).

We need to know how well the reduced model approximates the original. Using the properties of the Hankel singular values, we can find an upper bound on the  $\mathcal{H}_\infty$  error norm (6.6) given by [55]

$$\|G(s) - G_r(s)\|_\infty \leq 2 \sum_{i=k+1}^n \sigma_i \quad (6.12)$$

where  $G_r(s)$  is the reduced model of order  $k$ . This tells us that the  $\mathcal{H}_\infty$  error norm will be less than twice the sum of the discarded Hankel singular values.

One problem with the model truncation method is that it can change the steady-state solution of the system, or the so-called DC-gain. This leads us to the next method which tries to deal with this problem.



### 6.1.4. Method 2: Model Residualization

The second reduction method we will present tries to deal with the DC-gain offset that comes from performing model truncation. In model residualization we focus on making the steady-state solution unchanged. This is done by setting  $\dot{x}_2 \equiv 0$  and then performing some algebraic substitutions to remove the  $x_2$  state vector from (6.8)-(6.9). It can be shown that this gives the reduced model [55]

$$\begin{aligned} A_r &= A_{11} - A_{12}A_{22}^{-1}A_{21} \\ B_r &= B_1 - A_{21}A_{22}^{-1}B_2 \\ C_r &= C_1 - C_2A_{22}^{-1}A_{21} \\ D_r &= D - C_2A_{22}^{-1}B_2 \end{aligned}$$

This model will have the same solution as the original model for  $\dot{x} = 0$ , and it also has the same upper bound on the  $\mathcal{H}_\infty$  error norm as the model truncation method as given in (6.12). However, it will perform worse than the model truncation method at high-frequencies. In general, the choice between the two methods thus mainly depends on whether it is more important to maintain accuracy at high or low frequencies.

### 6.1.5. Other Methods

We have presented two methods which are fairly easy to implement. However, modifications to these methods as well as other schemes exist as well. The most notable is possibly the optimal Hankel norm approximation, which attempts at minimizing the Hankel norm of the error. The Hankel norm is defined in [appendix A.2](#). It can be shown that this method provides an upper bound on the  $\mathcal{H}_\infty$  norm [59]

$$\|G(s) - G_r(s)\|_\infty \leq \sum_{i=k+1}^n \sigma_i \quad (6.13)$$

that is, half the upper bound of the model truncation and residualization method.

Consider the case where the the state-space representation is given in a diagonal canonical form instead of the balanced form, such that the eigenvalues of the model is given by the diagonal elements of the  $A$ -matrix. Using the model truncation method on this realization will effectively remove the desired eigenvalues from the model. This can be useful if large eigenvalues are unwanted, but may more easily result in an unstable model.

We have only considered stable transfer functions, if we have unstable transfer functions we need different approaches, some methods are given in [55]. Another method is a modification to the model truncation method by providing weighting in the frequency domain [26].

## 6.2. Reduction of Controller

In the previous section we presented theory on model order reduction. Now we will make use of this theory by performing model reduction on the  $\mathcal{H}_\infty$  MIMO controller from [chapter 5](#). We will show how this is done using Matlab.

First, we find the balanced realization of the controller  $K(s)$  using the command `balreal` which returns the Hankel singular values as well as the balanced realization model. Next the model reduction is performed using the `modred` command. This command supports both the model truncation and residualization method, we chose to use the latter to maintain accuracy in the low-frequency region.

The model reduction process using Matlab is summarized with the following code which requires the control system toolbox, and is performed for each desired order from 2 to 17:

```
% Given original controller K, and desired order new_order
[K_b, sigma] = balreal(K);
elim = [ zeros(new_order, 1); ones(size(sigma, 1) - new_order, 1) ];
K_r = modred(K_b, logical(elim));
```

The Hankel singular values  $\sigma_i$  for each state in  $K_b$  as returned by `balreal(K)` is given in [Table 6.1](#).

**Table 6.1:** Hankel singular values of the balanced realization of the controller,  $\sigma_i$

1)	5.436e+03	7)	6.799e-02	13)	1.324e-02
2)	4.227e+03	8)	6.049e-02	14)	4.091e-03
3)	3.307e-01	9)	3.885e-02	15)	4.073e-03
4)	2.775e-01	10)	1.977e-02	16)	1.400e-03
5)	1.246e-01	11)	1.514e-02	17)	1.044e-03
6)	7.628e-02	12)	1.430e-02	18)	5.722e-05

## 6.3. Simulation Results

The first tests were done in simulations because we did not want to damage the physical equipment which may happen if some of the controllers turn out to be unstable. We used the Simulink model shown in [Figure 6.1](#).

We generated reduced order controllers  $K_r$  of order 2 to 17, and simulated step-responses using the Simulink model shown in [Figure 6.1](#). The step was made on the  $x$ -axis while the  $y$ -axis was kept at zero, and we measured the output on both axes. The results from order 7, 8, 9, 10 are plotted in [Figure 6.4](#). Results from order larger

than 10 were nearly indistinguishable from the 10th order results, and results from order lower than 7 were unstable, thus we omitted these plots.

We found the closed-loop  $\mathcal{H}_\infty$  error norm between the reduced and original controllers  $\|T - T_r\|_\infty$ , and plotted it in a bar-plot seen in Figure 6.2a. Additionally, the robust stability measure  $\|WT_r\|_\infty$  is given in Figure 6.2b which must be less than one for robust stability per (3.21). The frequency response of the closed-loop functions  $S_r$ ,  $T_r$  have been plotted in Figure 6.3 against  $S$ ,  $T$  for the 9th and 10th order controllers. These two controllers were chosen because of the notably large differences between them.

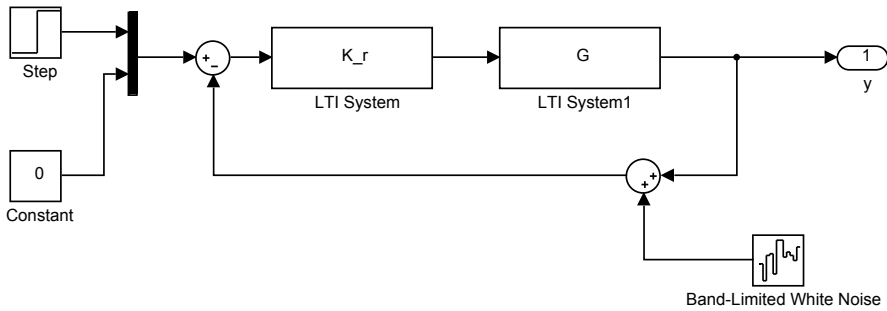


Figure 6.1: Simulink model for simulation of step-response.

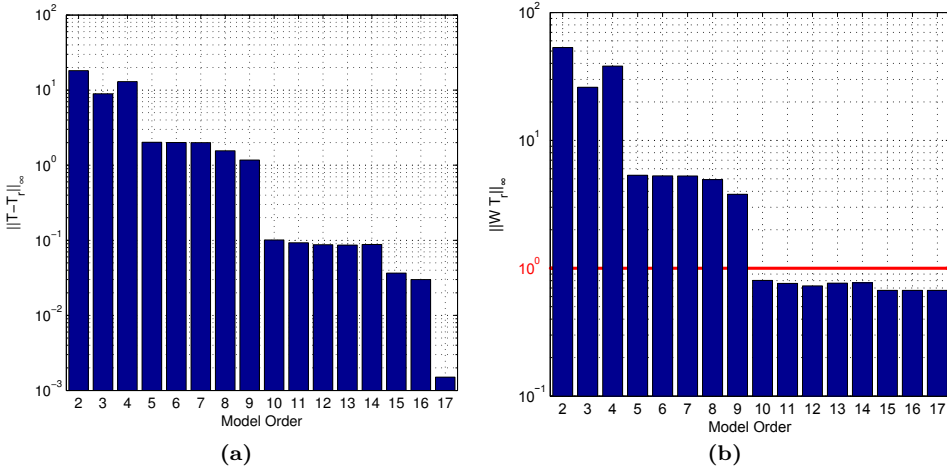
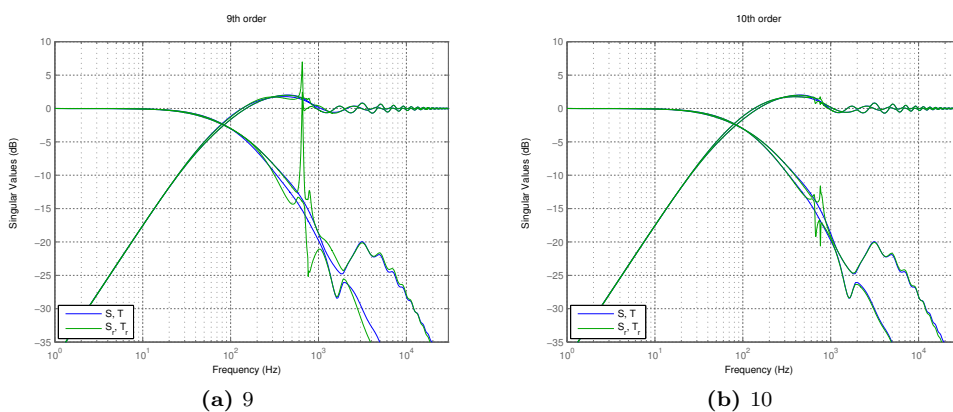
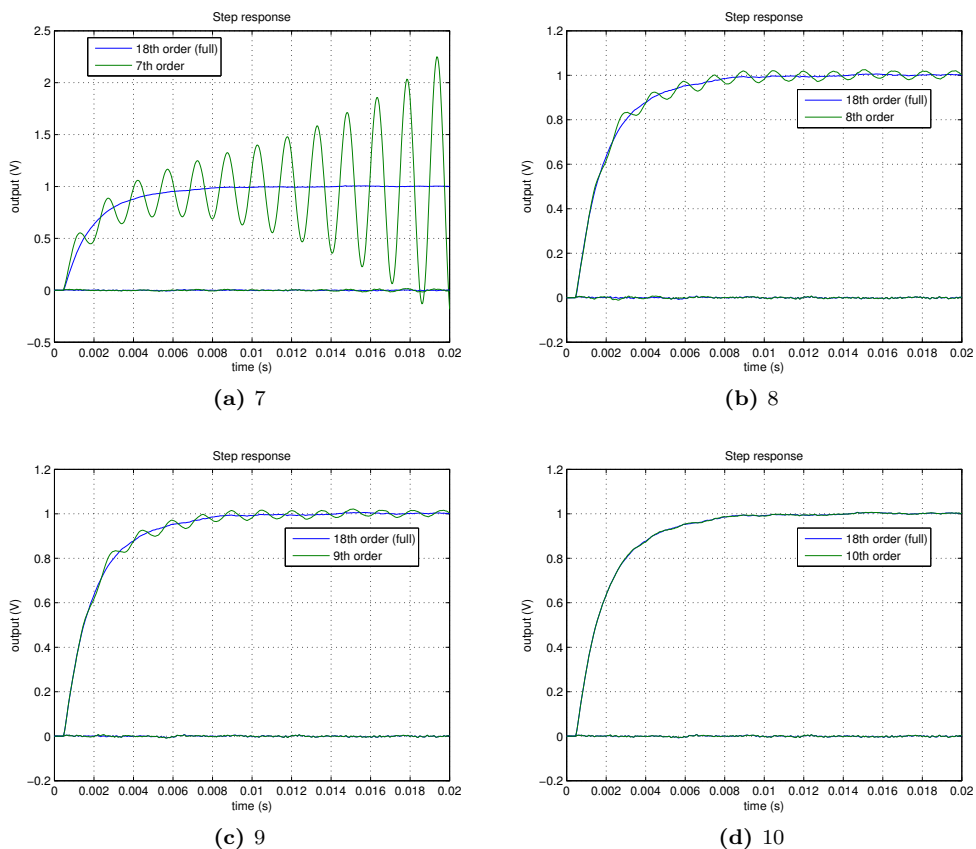


Figure 6.2: Reduced controller order properties. (a) Closed-loop error  $\|T - T_r\|_\infty$ . (b) Robustness  $\|WT_r\|_\infty$ , must be  $< 1$  for robust stability (marked by red).



**Figure 6.3:** Closed-loop frequency response  $\sigma(S)$ ,  $\sigma(T)$  versus  $\sigma(S_r)$ ,  $\sigma(T_r)$  for (a) 9th order controller and (b) 10th order controller.



**Figure 6.4:** Simulated step-response in reference signal on the  $x$ -axis, reduced vs original controller. Shows output from both  $x$ -axis and  $y$ -axis. The caption number states the order of the controller employed.

### 6.3.1. Observations

On the logarithmic error norm plot in [Figure 6.2a](#), we can clearly see that there are significant drops between a few of the orders, such as from 4→5, 9→10 and 16→17. The error changes relatively little in-between these drops. Since we would like a controller with as low order as possible while maintaining the performance characteristics, we are inclined to select one of the orders after such a drop, i.e. 5, 10, or 17. We can see the effect of the drops on the closed-loop frequency response between the controllers of order 9 and 10 as plotted in [Figure 6.3](#). Where the order 10 controller has only some small errors at around the resonance frequency, the order 9 controller has huge spikes which would lead to noise amplification at this frequency, tracking errors, as well as possibly instability.

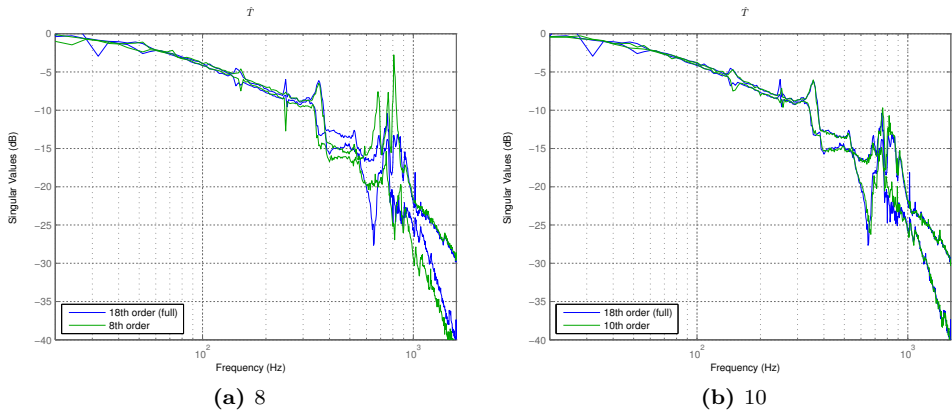
From [Figure 6.2b](#) we can see that only controller order 10 and higher are robustly stable with  $\|WT_r\|_\infty < 1$ . Lower orders can not guarantee robust stability, so in a practical implementation we should avoid using these.

The previous discussion clearly favors choosing the 10th order controller as it provides robust stability with little error. This choice is further reinforced by considering the step responses as shown in [Figure 6.4](#). The 10th order controller gives nearly indistinguishable results to the original controller, while the 8th and 9th order controllers shows some oscillatory behavior. The 7th order model is unstable, so we clearly want to avoid it.

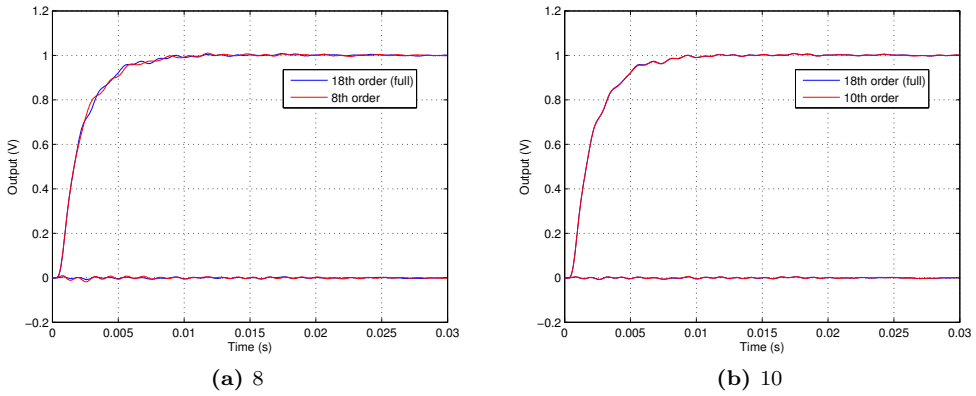
## 6.4. Experimental Results

In the previous section we ran simulations of the various reduced controllers. The primary reason for this was to avoid running unstable controllers on the experimental equipment as well as the ease of setup. However, experimental tests are important to determine real-world results which may be different due to unmodeled dynamics, noise, and other disturbances.

We used the same setup as described in [section 5.1](#). The 8th order and 10th order controllers were implemented and compared to the original controller. We ran the same step-response as in the previous section, and recorded the closed-loop frequency response  $\hat{T}_r$  using the SR780 device. The frequency response is plotted in [Figure 6.5](#), and the step-response is plotted in [Figure 6.6](#). A new closed-loop frequency response of the original controller was gathered, such that the conditions were similar for when the original and reduced controllers were run. Because of the large uncertainties inherent in the nanopositioning platform the closed-loop response from [chapter 5](#) will look slightly different.



**Figure 6.5:** Experimental complementary sensitivity  $\sigma(\hat{T}_r)$  for the reduced order controllers versus the original controller  $\sigma(\hat{T})$ . (a) 8th order, and (b) 10th order.



**Figure 6.6:** Experimental step-response in reference signal on the  $x$ -axis, reduced vs original controller. Shows output from both  $x$ -axis and  $y$ -axis. (a) 8th order, and (b) 10th order.

### 6.4.1. Observations

We can see that the frequency response from the 10th order controller is nearly indistinguishable from the original controller in [Figure 6.5](#). The 8th order controller has some larger spikes rising up to -2.76 dB at 808 Hz, but is not as bad as in the simulations.

The step-response plotted in [Figure 6.6](#) shows that the 10th order model performs just as well as the original controller. The 8th order model shows some differences compared to the original, although the differences are smaller than in the simulations and it is not clear from this plot whether the original or reduced performs better. We can see that the reduced 8th order has slightly worse cross-coupling oscillations at the start of the step.

Considering the previous discussion, we can conclude that the 10th order controller performs nearly as good as the original controller and the 8th order model performs slightly worse, but better than in the simulations.

## 6.5. Conclusion

The purpose of this chapter was to investigate feasibility of performing model order reduction to reduce the complexity of a controller. We based our discussion around the results of reducing the 18th order  $\mathcal{H}_\infty$  MIMO controller from [chapter 5](#). We showed that the controller could be reduced to a 10th order model with negligible impact on performance and still maintain robust stability. The 8th and 9th order controllers resulted in noticeably slightly worse performance, and did not achieve robust stability, although they were nominally stable in our simulations and experiments. The 7th order controller and lower models were nominally unstable.

As we have seen, the order of the controller can be significantly reduced with only small changes in performance and stability. In the next chapter we will investigate how much this order reduction means in terms of computational complexity and ultimately for the implementability of the controller.



## Chapter 7

# Computational Complexity after Control Reduction

In the previous chapter we showed the performance and stability impact of reducing the model order of the controller to various orders. In this chapter we will investigate what the reduction of the controller order means in terms of computational complexity. In a practical implementation this is important to consider because the experimental setup is limited in terms of hardware performance and needs to perform in real-time. This puts a lower limit on the step-size we can implement a certain controller with on a given device, any lower than this and the hardware can't keep up with the required calculations. At the same time, a step-size needs to be small enough such that the system is stable and performs as desired. This gives an upper limit on the step-size. If the two limits don't intersect we can not implement the controller.

We will measure computational complexity in two distinct ways, first in the desktop environment and then in the experimental environment. By first simulating different modes in the desktop environment, we can find which of them are unstable so we can avoid implementing them on the experimental model. Additionally, we will be able to test stability at a wider range of step-sizes that the real-time setup can't handle.

### 7.1. Simulation Results

There is no simple method to measure computational complexity because the actual performance will vary with different hardware setups, and does not necessarily scale linearly with clock frequency. Different operations may take different number of clock cycles on different architectures. Even so, by comparing similar models we can get an indication of the complexity. We will do this by measuring the time it takes to perform a simulation while changing one variable at a time. Simulation time was recorded by running the Simulink model in [Figure 6.1](#) with different controller order, solver type, and step-size. We recorded the time it took to simulate 1 s, the results are

given in [Table 7.1](#) and [Table 7.2](#). The different solvers are explicit Runge-Kutta (ERK) methods of various orders, denoted `erk[1-8]` where the number represents the solver order. Additionally, we have performed the simulations on the only fixed-step implicit Runge-Kutta method available in Simulink which is denoted `ode14x`. The outputs from a simulation using this solver and the full-order original controller is plotted in [Figure 7.1](#) to demonstrate how the solutions sometimes explode using this implicit method. All solvers are performed with a fixed step-size because this is necessary for a real-time implementation, and will thus better represent the computational complexity for such an application.

Some notable comments about the data. The simulations were run on a desktop computer and may be affected by various background tasks and operative system conditions. The simulation runs both the controller  $K_r$  and the model of the plant  $G$ , therefore reducing the order of  $K_r$  will not be as effective as if we only implemented the controller like we would in a practical implementation. Also remember that this test does not take accuracy and performance into account, only stability. Ultimately we would like to choose the combination with the lowest computational complexity (i.e. lowest simulation time) while maintaining acceptable accuracy and stability.

### 7.1.1. Observations

In general, the data in [Table 7.1](#) and [Table 7.2](#) reveals that shorter step-size and more complex solver types increases both numerical stability and computational complexity as should be expected. Controllers of order 7 and lower were nominally unstable, so there is no data for these modes.

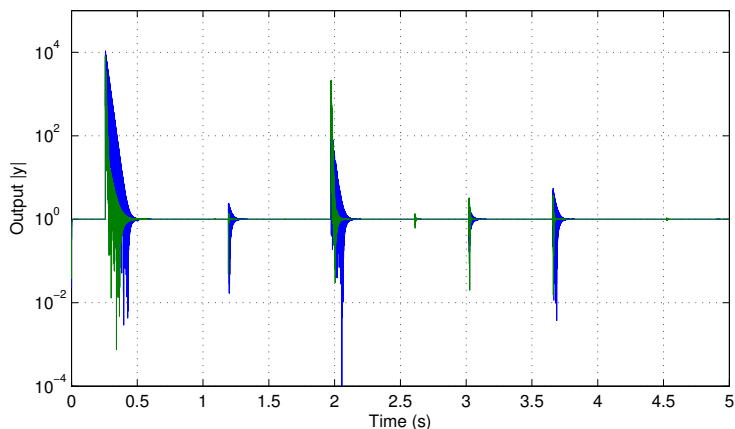
An interesting observation from the data is that the lower-order controllers are stable with less complex solver types than higher-order controllers. This means that by reducing the order of the controller, not only do we gain a computationally easier problem from fewer states, but also from reducing the solver complexity. We speculate that this may be because the model reduction tends to reduce the fastest modes, i.e. the most negative eigenvalues of the controller. These eigenvalues will result in numerical instability with simpler solver type if we reduce the step-size sufficiently, while more complex solver types are stable for larger regions in general.

Simulation time increases about linearly with step-size, which is expected since the same calculations needs to be performed each step. The primary purpose of performing the calculations at various step-sizes is to see at which step-size a certain mode becomes unstable. This will also be valuable in the analysis in the next chapter.

With increasing controller order, the simulation time tends to increase steadily. There are some exceptions however which may be due to statistical variances, or small numerical differences. The effect of increased controller order on the simulation time doesn't seem to be very large compared to the solver complexity. But we are also simulating the plant model which does not have any benefit from the controller reduction. Thus there is a high constant overhead to the simulation time which is not reduced by varying the controller order. On the other hand, this overhead is reduced

when changing to a simpler solver type because this also simplifies the calculations of the plant model.

The reasons for providing the plot of a simulation using the implicit Runge-Kutta (IRK) solver in Figure 7.1 is to demonstrate the numerical explosions that we observed with this method. We will not further investigate the reasons for these explosions, but we mention it as a precaution to using this method for a real-time implementation. These effects were not observed using the explicit Runge-Kutta methods.



**Figure 7.1:** Outputs from a simulation using the implicit Runge-Kutta method `ode14x`, showing how the outputs explodes at certain intervals. Reference signal is set to one.

**Table 7.1:** Simulation time and stability with different controller model order, solver types and step times (part 1/2). The 18th order model is the original non-reduced controller. All values are in seconds. Dash (-) represents instability.

Order	$h = 100 \mu\text{s}$					
	erk1	erk2	erk3	erk5	erk8	ode14x
$\leq 7$	-	-	-	-	-	-
8	-	-	-	-	0.448	1.949
9	-	-	-	-	-	2.094
10	-	-	-	-	-	2.071
11	-	-	-	-	-	2.249
12	-	-	-	-	-	2.334
13	-	-	-	-	-	2.423
14	-	-	-	-	-	2.344
15	-	-	-	-	-	2.563
16	-	-	-	-	-	2.589
17	-	-	-	-	-	2.767
18	-	-	-	-	-	2.671

Order	$h = 40 \mu\text{s}$					
	erk1	erk2	erk3	erk5	erk8	ode14x
$\leq 7$	-	-	-	-	-	-
8	-	0.281	0.345	0.543	0.995	4.681
9	-	-	0.346	0.549	1.036	5.245
10	-	-	0.349	0.542	1.038	5.146
11	-	-	0.350	0.552	1.029	5.622
12	-	-	0.350	0.549	1.078	5.572
13	-	-	-	-	1.093	5.952
14	-	-	-	-	1.100	5.879
15	-	-	-	-	1.116	6.265
16	-	-	-	-	1.114	6.314
17	-	-	-	-	1.123	6.819
18	-	-	-	-	1.075	6.504

**Table 7.2:** Simulation time and stability with different controller model order, solver types and step times (part 2/2)

Order	$h = 10 \mu\text{s}$					
	erk1	erk2	erk3	erk5	erk8	ode14x
$\leq 7$	-	-	-	-	-	-
8	-	0.897	1.140	1.905	3.800	18.754
9	-	0.892	1.170	1.995	3.824	20.653
10	-	0.900	1.199	2.014	3.930	20.470
11	-	0.920	1.191	2.019	3.967	22.360
12	-	0.922	1.202	2.068	4.002	22.252
13	-	0.918	1.220	2.037	4.172	23.240
14	-	0.921	1.218	2.041	4.130	22.719
15	-	0.940	1.217	2.069	4.170	24.869
16	-	0.923	1.225	2.125	4.305	25.524
17	-	0.953	1.255	2.131	4.364	27.339
18	-	0.935	1.197	2.070	4.158	25.919

Order	$h = 1 \mu\text{s}$					
	erk1	erk2	erk3	erk5	erk8	ode14x
$\leq 7$	-	-	-	-	-	-
8	5.713	8.185	10.870	18.839	37.730	184.648
9	5.739	8.337	10.886	19.060	38.869	205.038
10	5.705	8.572	11.129	19.421	39.233	206.224
11	5.560	8.446	11.230	19.692	38.209	220.674
12	5.753	8.550	11.262	19.827	39.899	222.089
13	5.716	8.488	11.335	20.209	39.543	233.898
14	5.861	8.605	11.320	19.655	40.000	228.107
15	5.857	8.556	11.239	19.892	42.131	255.043
16	5.768	8.879	11.521	20.254	41.245	252.640
17	6.010	8.751	11.781	21.251	41.637	267.837
18	5.799	8.513	11.326	19.699	40.628	255.363

## 7.2. Experimental Results

The simulation time presented in the previous section provides an indication of the complexity of the model, but it has several shortcomings as discussed. In this section we will implement the stable modes from the previous section on the experimental setup and measure the average task execution time (TET). This value is provided by the xPC operative system and is described as follows by the xPC User's Guide [57]

“ [Average task execution time] is an average of the measured CPU times, in seconds, to run the model equations and post outputs during each sample interval. Task execution time is nearly constant, with minor deviations due to cache, memory access, interrupt latency, and multirate model execution. ”

TET should not depend on step-size, but will depend on the controller order and complexity of the solver. If the TET takes longer than the step-size, the operative system will stop all execution and report a “CPU overload” error. To be able to run the model we would then have to increase the step-size, but at some point the controller will become unstable as can be seen from the data in the previous section.

We used the experimental setup as described in [section 5.1](#). A very simple Simulink model was used with only the controller  $K_r$  and necessary input-output connections. All simulations were run with a step-size of  $40 \mu\text{s}$  as this was about the limit the xPC hardware could handle most of the modes without overloading. The exception is the modes using the ode14x solver, which was too demanding performance-wise to be implementable on our hardware except for the 8th order controller. The results are given in [Table 7.3](#).

### 7.2.1. Observations

Compared to the simulations, we can see that reducing the controller order is more effective in the experimental setup in terms of execution time. This can possibly to a large extent be attributed to that we are only running the controller in this setup and not the plant model, so we have a lower constant overhead.

We can see that by reducing the controller to 8th order, we achieve a reduction in TET by 49.0% from 20.11 to  $10.25 \mu\text{s}$ , partly because of the reduced solver complexity. For the 10th order model we have a reduction of 46.7%, from 20.11 to  $10.71 \mu\text{s}$  with the ode3 solver.

The PID controller has a lower TET than any of the other controllers. However, it is not robustly stable. The controller with the lowest value of TET with robust stability is the 10th order reduced controller using the ode3 solver. The  $\mathcal{H}_\infty$  SISO controller executes slightly faster than the 10th order reduced with the same solver, but this advantage is lost because it needs a more complex solver type for stability.

**Table 7.3:** Average task execution time (TET) with different controller model order and solver types. Dash (-) not tested. (x) CPU overload.

Order	Average TET [ $\mu$ s]					
	erk1	erk2	erk3	erk5	erk8	ode14x
$\leq 7$	-	-	-	-	-	-
8	-	10.25	10.43	11.10	13.64	26.72
9	-	-	10.55	11.43	14.26	x
10	-	-	10.71	11.64	14.99	x
11	-	-	10.89	11.91	15.79	x
12	-	-	11.07	12.36	16.79	x
13	-	-	-	-	17.68	x
14	-	-	-	-	18.82	x
15	-	-	-	-	19.61	x
16	-	-	-	-	20.91	x
17	-	-	-	-	22.61	x
18	-	-	-	-	20.11	x
PID	-	9.95	9.98	10.13	11.12	14.91
$\mathcal{H}_\infty$ SISO	-	-	-	-	14.58	x

We can also see that there seems to be a significant constant overhead also on these experimental results. We would expect the PID controller to be even faster compared to the  $\mathcal{H}_\infty$  controllers if there was no overhead, so some time seems to be spent on other tasks. This can possibly be attributed to the input-output signaling to and from the DAC/ADC.

It is also interesting to note that the execution time does not strictly decrease with increased controller order, e.g. the 17th to 18th order controller. By inspecting the state-space model of each of these controllers, we notice that the 18th order controller has a lot more zero-valued elements. We speculate that the compiler simplifies the arithmetic on these elements.

### 7.3. Conclusion

In this chapter we have estimated the reduction in computational complexity of reducing the model order of the  $\mathcal{H}_\infty$  MIMO controller from [chapter 5](#). This has been done both in a simulated environment and in an experimental environment.

By reducing the controller from 18th order to 10th order, not only is the model computationally simpler itself, but we also achieved numerical stability with a less

complex solver type at the same step-size. With the combined benefit of a less complex controller, and less complex solver, we achieved a 46.7% reduction in task execution time on the experimental equipment. This comes at almost no cost on the performance of the controller, and the number would possibly have been even better if there was no constant overhead to the calculations.

The 14th order  $\mathcal{H}_\infty$  SISO controller is similar to the 10th order reduced controller in terms of performance, and is slightly faster computationally using the same solver. But the 10th order reduced controller can achieve stability with a less complex solver type, which in the end makes it execute faster.



## Chapter 8

# Numerical Stability of Controller

The previous two chapters investigated the feasibility of performing model order reduction on a controller, and showed how much this means in terms of computational complexity. This was done in order to make the controller more easily implementable on a real-time system. We could see that the controller became unstable at certain step-sizes for a given controller and solver type. This chapter will perform a more analytical approach when it comes to the stability of numerical methods such that we can possibly predict which modes are stable for a given controller a priori.

Controllers can be described using continuous-time state-space models. For a real-time implementation however, the model is solved at discrete time-steps using a fixed step-size. Many popular solver types are based on the family of explicit Runge-Kutta (ERK) methods, which can become unstable if the step-size is too large. At the same time, the complexity of the controller running on hardware is limited in terms of computational power so we can't set an arbitrary small step-size.

These problems are especially significant for *stiff systems* with *fast modes*. Stiffness is not a rigidly defined property, but is often understood as a large spread of eigenvalues. Another characteristic is that the stability of the numerical solutions can become an issue and is generally more important to consider than the accuracy of the solution [11]. For a system with mechanically high bandwidth, we need a controller with fast modes to perform control at large frequencies, which means that the step-size also needs to be small enough to perform control at these high frequencies. The nanopositioning device used in this thesis could be considered stiff, as is shown by the large spread of eigenvalues in Table 8.2. There is also a physical interpretation of this, because the lateral platform of the AFM is a lightly damped system with high resonant frequencies. The controller dampens out the resonant peak which requires it to operate at a frequency at least equivalent to this peak. Thus our complex  $\mathcal{H}_\infty$  controller has fast modes which give large negative eigenvalues. Additionally there are small eigenvalues to give an integral-like effect which in summary means that the controller has a large spread of eigenvalues. Note that by the expression “eigenvalues of the controller” we mean the eigenvalues of the  $A$ -matrix in the state-space representation of the controller.

We start this chapter by defining the family of explicit Runge-Kutta methods. Then we will show how the stability of the numerical solution for a given solver type depends on the step-size and eigenvalues of the controller. Finally we present how to find the maximum step-size for a given controller and solver type, and present these results for several of the controllers presented earlier in this thesis.

## 8.1. Explicit Runge-Kutta Methods

Let us consider the ordinary differential equation (ODE)

$$\dot{y} = f(t, y) \quad (8.1)$$

where  $t$  is the time variable. A numerical solver estimates the solution to this equation at discrete time-steps. For a fixed-step solver we denote the step-size by  $h$ . Controllers such as presented in the previous chapters can be represented by a system of ODEs, as evident from the state-space equations (6.1)-(6.2). The literature on numerical methods is vast, and we will consider the commonly used explicit Runge-Kutta (ERK) methods as used in our experiments.

### Definition: Explicit Runge-Kutta methods

The family of explicit Runge-Kutta (ERK) methods can be written as [22]

$$y_{n+1} = y_n + \sum_{i=1}^s b_i k_i \quad (8.2)$$

where  $s$  describes the number of stages of the Runge-Kutta method and

$$\begin{aligned} k_1 &= hf(t_n, y_n) \\ k_2 &= hf(t_n + c_2 h, y_n + a_{21} k_1) \\ k_3 &= hf(t_n + c_3 h, y_n + a_{31} k_1 + a_{32} k_2) \\ &\vdots \\ k_s &= hf(t_n + c_s h, y_n + a_{s1} k_1 + a_{s2} k_2 + \dots + a_{s,s-1} k_{s-1}) \end{aligned}$$

where the coefficients  $a_{ij}$ ,  $b_i$ , and  $c_i$  are specified for a given ERK method.

Note that there are certain constraints on the coefficients for it to be a valid Runge-Kutta method, e.g. the sum of  $b_i$  must equal 1. The coefficients are often arranged in a Butcher tableau

0					
$c_2$	$a_{21}$				
$c_3$	$a_{31}$	$a_{32}$			
$\vdots$	$\vdots$		$\ddots$		
$c_s$	$a_{s1}$	$a_{s2}$		$a_{s,s-1}$	
	$b_1$	$b_2$	$\cdots$	$b_{s-1}$	$b_s$

and makes up the matrices  $A, b, c$  such that the tableau can be written as

$$\frac{c}{\quad} \left| \begin{array}{c} A \\ b^T \end{array} \right.$$

## 8.2. Stability of Explicit Runge-Kutta Methods

In order to analyze the numerical stability of a system solved by an ERK method, we will start by introducing a simple differential equation which as we will see can be used to determine the numerical stability of the state-space equations for a controller.

Let us consider the scalar test system

$$\dot{y} = \lambda y \tag{8.3}$$

where  $\lambda$  can possibly take on complex values. A solver applied to this system takes the discrete state  $y_n$  to the next time step  $y_{n+1}$  with step-size  $h$ ,

$$\begin{aligned} y_{n+1} &= \Phi(h\lambda)y_n \\ &= [\Phi(h\lambda)]^n y_0 \end{aligned} \tag{8.4}$$

where  $\Phi(h\lambda)$  is called the stability function. It is evident that (8.4) is stable, i.e.  $|y_n| \leq c < \infty \forall n \geq 0$ , if and only if

$$|\Phi(h\lambda)| \leq 1 \tag{8.5}$$

All solvers we will consider has such a stability function, and the region of stability, i.e. the region of the complex plane where (8.5) is satisfied, varies between each solver type.

**Example 1** *Euler's Method applied to (8.3) gives*

$$\begin{aligned} y_{n+1} &= y_n + f(t_n, y_n) \\ &= y_n + h(\lambda y_n) \\ &= (1 + h\lambda) y_n \end{aligned} \tag{8.6}$$

thus  $\Phi(h\lambda) = 1 + h\lambda$ , which is stable in the region  $\{z \in \mathbb{C} \mid |1 + z| < 1\}$ , in other words the unit circle with center  $-1$ .  $\triangle$

Before we find an expression for the stability function of ERK methods, let us introduce the concept of solver order. The next definition is taken verbatim from Egeland and Gravdahl [22].

**Definition: Order  $p$  of a one-step solver method**

A one-step method is of order  $p$  if  $p$  is the smallest integer such that  $e_{n+1} = O(h^{p+1})$ . If the numerical solution  $y_{n+1}$  satisfies the equation

$$y_{n+1} = y_n + hf(y_n, t) + \dots + \frac{h^p}{p!} \frac{d^{p-1}f(y_n, t_n)}{dt^{p-1}} + O(h^{p+1}) \quad (8.7)$$

then  $e_{n+1} = O(h^{p+1})$ , and it follows that the method is of order  $p$ .

The order of an ERK method can be found by considering the coefficients  $A, b, c$ , although the details will be omitted here. We can now state the stability function of ERK methods.

The stability function of an explicit Runge-Kutta method with coefficients  $A$  and  $b$  is given by [22]

$$\Phi(z) = \det(I - zA + z\mathbf{1}b^T) \quad (8.8)$$

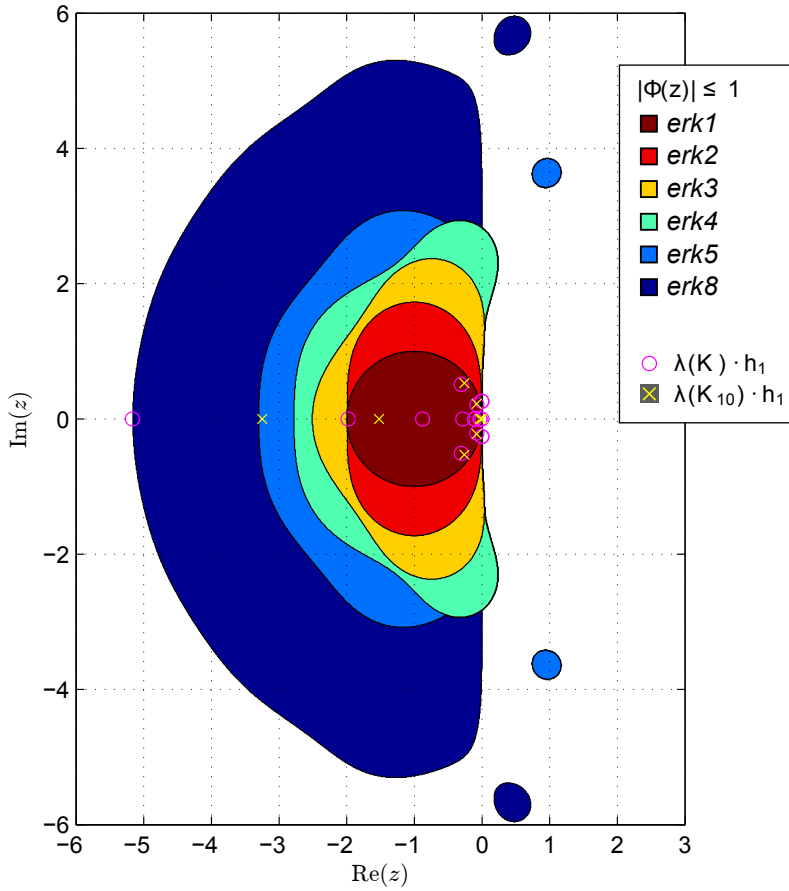
where  $\mathbf{1}$  is a column vector of one-elements. In fact for an ERK method of order  $p = s$  this simplifies to

$$\Phi(z) = 1 + z + \dots + \frac{z^p}{p!} \quad (8.9)$$

although only ERK of order 1–4 can have  $p = s$ , higher order requires more stages such as the fifth order which needs a minimum of six stages [12]. The Butcher tableau of some popular ERK methods is given in Table 8.1. These methods are reckoned to be the same as the fixed-step solver methods available in Simulink. The stability region of these methods is plotted in Figure 8.1 together with some of the eigenvalues of two controllers considered in previous chapters. These will be discussed later.

**Table 8.1:** Selection of explicit Runge-Kutta methods with their corresponding Butcher tableau and stability function

Solver name (order)	Butcher tableau	Stability func. $\Phi(z)$
Euler (erk1)	$\begin{array}{c c} 0 & \\ \hline & 1 \end{array}$	$1 + z$
Heun (erk2)	$\begin{array}{c cc} 0 & & \\ \hline 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$	$1 + z + \frac{z^2}{2}$
Bogacki–Shampine (erk3) Note: 4 stages, but order 3	$\begin{array}{c ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{3}{4} & 0 & \frac{3}{4} & \\ \hline 1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} \\ \hline & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & 0 \end{array}$	$1 + z + \frac{z^2}{2} + \frac{z^3}{6}$
Runge-Kutta (erk4)	$\begin{array}{c ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & 0 & \frac{1}{2} & \\ \hline 1 & 0 & 0 & 1 \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$	$1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}$
Dormand-Prince (erk5)	See Butcher tableau in <a href="#">appendix A.3</a> .	
Dormand-Prince (erk8)	Stability function found by evaluating (8.8)	



**Figure 8.1:** Stability region of the ERK methods in Table 8.1. *Magenta circles:* Eigenvalues of the full order controller  $K$  scaled by the maximum step-size achieving stability for erk8, denoted  $h_1$ . *Yellow X's:* Eigenvalues of the reduced tenth order controller  $K_{10}$  scaled by the same step-size. Note that  $K_{10}$  is stable for erk5, so the order reduction has made it possible to use a simpler solver type.

### 8.3. Runge-Kutta Stability of Linear Systems

We have stated the stability functions for several ERK methods for the scalar test system. But we are ultimately interested in the stability of an explicit Runge-Kutta method applied to our controllers which are larger state-space models. In this section, we will show how to find the stability of the solvers applied to a linear system of ODEs.

**Theorem 1.** *Consider the system*

$$\dot{y} = Ay \tag{8.10}$$

where  $A$  is an  $n \times n$  diagonalizable matrix with eigenvalues  $\lambda_1, \dots, \lambda_n$ . Let us apply a Runge-Kutta method to this system. Then the RK method has a stable (asymptotically stable) fixed point at the origin if and only if the same method has a stable (asymptotically stable) fixed point for

$$\dot{x} = \lambda_i x \quad \forall i \in [1, \dots, n] \tag{8.11}$$

This is a standard result in theory on numerical methods, see e.g. Ascher and Petzold [4]. We have used wording similar to [29] which is also used for the next corollary.

**Corollary 2.** *Consider a Runge-Kutta method applied to the system  $\dot{y} = Ay$ . Then the origin is stable for the numerical method with step-size  $h$  if and only if*

$$|\Phi(h\lambda_i)| \leq 1, \quad \forall i \in [1, \dots, n] \tag{8.12}$$

where  $\lambda_i$  are the eigenvalues of  $A$ .

In other words, if all the eigenvalues of  $A$  are within the region of stability by satisfying (8.5) for a given solver and step-size  $h$ , then the solver applied to (8.10) is stable. This gives us a tool to check for the required maximum step-size for a given controller and solver as will be discussed next.

### 8.4. Determining Maximum Step-Size

We will now discuss how to find the maximum step-size for a given controller and solver type. In the previous sections we showed that a solver applied to a controller is stable iff it's stable for all its eigenvalues individually. We can thus find a maximum step-size by starting with a large value of  $h$  and reducing it until (8.5) is satisfied for all the eigenvalues of the controller. This may be slow, but can be sped up by doing a binary search as in Algorithm 8.1. Note that this may not give correct results if the stability region is not convex as viewed from the origin, i.e. every point on the edge of the stability region must be visible from the origin<sup>1</sup>. This seems to be the case for the stability regions plotted in Figure 8.1 if we only consider the left half plane.

<sup>1</sup>We use the word convex rather roughly here, this is not the exact mathematical definition.

We have used this method to find the maximum step-size  $h_{max}$  for the controllers designed in [chapter 5](#) as well as the reduced order controllers from [chapter 6](#) for various solvers. The results are given in [Table 8.3](#). The Matlab code used to generate the data for this table as well as the stability region plot is given in [appendix C.4](#).

The eigenvalues of the  $\mathcal{H}_\infty$  MIMO controller is plotted in [Figure 8.1](#) (magenta circles) together with the stability regions of the solvers. They have been scaled by their maximum step-size achieving stability for `erk8`, i.e.  $h_1 = 54.62 \mu\text{s}$ . The eigenvalues of the reduced 10th order controller is also plotted (yellow x's) at the same step-size. The eigenvalues of the two controllers are listed in [Table 8.2](#) for reference.

---

**Algorithm 8.1** Binary search for  $h_{max}$

---

```

L = 0
U = ( $\gg$  1)
while (U - L) >  $\epsilon$  do
    h = (U - L)/2 + L
    if  $|\Phi(h\lambda_i)| \leq 1 \forall i$  then
        L = h
    else
        U = h
    end if
end while
return L

```

---

**Table 8.2:** Eigenvalues of original controller  $K$  and 10th order reduced controller  $K_{r10}$ , sorted by absolute value

$\lambda(K) \times 10^4$					
1)	-9.4590	7)	-0.0033 + 0.4793i	13)	-0.2044
2)	-3.6243	8)	-0.0033 - 0.4793i	14)	-0.2003
3)	-1.6074	9)	-0.0085 + 0.4777i	15)	-0.0548
4)	-0.5644 + 0.9322i	10)	-0.0085 - 0.4777i	16)	-0.0495
5)	-0.5644 - 0.9322i	11)	-0.1395 + 0.3820i	17)	-0.0000
6)	-0.5228	12)	-0.1395 - 0.3820i	18)	-0.0000
$\lambda(K_{r10}) \times 10^4$					
1)	-7.5155	5)	-0.1557 + 0.3664i	9)	-0.0000
2)	-2.8710	6)	-0.1557 - 0.3664i	10)	-0.0000
3)	-0.5239 + 1.1179i	7)	-0.1016		
4)	-0.5239 - 1.1179i	8)	-0.0376		



**Table 8.3:** Maximum step-size of a given explicit Runge-Kutta (ERK) method for the various controllers presented in the previous chapters as well as the nominal plant model. Larger values are generally better because they are stable at higher step-sizes.

Order	$h_{max}$ [ $\mu$ s]					
	erk1	erk2	erk3	erk4	erk5	erk8
7 (unst.)	32.18	71.33	95.74	100.8	127.2	203.6
8	58.37	58.37	73.34	81.29	96.51	150.8
9	37.54	37.54	47.17	52.28	62.07	96.99
10	33.61	33.61	42.22	46.8	55.56	86.82
11	39.70	39.7	49.87	55.28	65.63	102.5
12	24.96	32.66	41.03	45.48	54.00	84.37
13	2.821	21.55	27.07	30.01	35.63	55.67
14	2.818	19.11	24.01	26.62	31.60	49.38
15	2.843	21.73	27.30	30.26	35.92	56.13
16	2.948	22.64	28.44	31.53	37.43	58.48
17	2.918	21.18	26.61	29.49	35.01	54.71
18 (full)	2.910	21.14	26.56	29.45	34.96	54.62
PID	80.00	80.00	100.5	111.4	132.3	206.7
$\mathcal{H}_\infty$ SISO	21.45	21.45	26.95	29.87	35.46	55.41
$G(s)$	4.564	52.42	65.85	73.00	86.66	135.4

## 8.5. Observations

In [Figure 8.1](#) we can see that the most negative eigenvalue is on the edge of the stability region. If the step-size was set any higher than this the eigenvalue would move outside (to the left of) the stability region and make the system unstable. The eigenvalues of the 10th order reduced controller have moved to the right causing them all to lie within the stability region of erk5. Thus, by reducing the model order we have made it possible to use a less complex solver method or alternatively a larger step-size.

The same phenomena can be observed in [Table 8.3](#). The reduced controllers are generally stable for higher step-sizes as they decrease in order, but not strictly so, e.g. order 15 to 14.

In the previous chapter we ran several simulations for various controllers and solvers at different step-sizes primarily to give a measure of the computational complexity. But it also turned out that several of the modes became unstable for large step-sizes.

We can compare these results (page 66-67) to the values in Table 8.3 to see if the unstable modes match up with our analysis. Note that since we simulated both the controller and nominal plant  $G(s)$  in a feedback loop, the mode must be stable for both of these.

The results match up very well. For example, the only stable controller with  $h_{max} \geq 100 \mu\text{s}$  is the 8th order controller with erk8 solver, and we can see in Table 7.1 that this is the only stable mode for  $h = 100 \mu\text{s}$ . The results also match up for  $h = 40 \mu\text{s}$  and actually reveals that the instability for 8th order controller with erk1 is due to the plant model  $G(s)$  and not the controller which should be stable at this mode. Because we assumed it was unstable, we did not test it in the task execution time experiment.

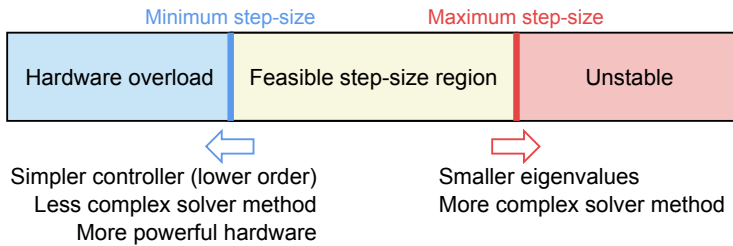
## 8.6. Conclusion

The results of this chapter gives us an insight into to the reason for the instability experienced in chapter 5 and chapter 7. We have shown that numerical stability properties can be well predicted and the results are confirmed by comparison to the stability of the simulations in the previous chapter.

As we have seen, the eigenvalues of the controller are one of the decisive factors for stability of an applied explicit Runge-Kutta method. Hence, it is important to consider how controller reduction changes the eigenvalues, especially for a fast and stiff system such as our lateral positioning platform of an AFM. The eigenvalues of our controller tended to become smaller (in absolute value) with reduced orders, but this need not be the case. One should be careful when performing model reduction and possibly verify that the eigenvalues are within the stable region of the solver considered. If the solver stability becomes a problem, one could consider a different model reduction method, such as removing the eigenvalues directly from a diagonal canonical realization of the controller.

We have also seen how increased solver order increases the stability region, but at the same time it increases the execution time. This will ultimately be a trade-off between moving the lower limit (due to computation time) and the upper limit (for stability) of the step-size, as illustrated in Figure 8.2. If the unstable and hardware-overload regions overlap, it is impossible to implement the controller. Then one must either design a computationally simpler controller possibly via model reduction, a controller with smaller eigenvalues, use more powerful hardware, or try to change the solver method.

Note that we have not considered the accuracy of the solver methods. This is because we have assumed that the system is stiff, and stiff systems are characterized such that instability occurs before accuracy becomes an issue.



**Figure 8.2:** Illustration of the trade-off between step-size, controller complexity, solver complexity, eigenvalues of the controller, and hardware performance for a real-time controller implementation.



## Chapter 9

# Conclusions

This thesis has presented control challenges within the topic of nanopositioning devices. The challenges are primarily due to nonlinearities such as hysteresis and creep of piezoelectric actuators, lightly-damped vibration dynamics, and large uncertainties. We have designed several control laws which tries to overcome these challenges and implemented them for real-time control of the lateral ( $xy$ -) axes of an Atomic Force Microscope (AFM). We have also discussed the implementability issues for complex controllers running on hardware with limited computational power.

Because of the high uncertainties and disturbances involved with nanopositioning, we have emphasized robustness in the control law design. A robustly stable  $\mathcal{H}_\infty$  mixed sensitivity MIMO controller was designed. Such a model-based control law effectively reduces the resonant peaks arising due to the lightly damped dynamics of the system. This controller was compared to a  $\mathcal{H}_\infty$  SISO version of the same controller (one for each axis) with similar design criteria, and a simple PID controller. We focused especially on the comparison in cross-coupling gains between the two lateral axes. In AFMs an image scan is often performed using a raster pattern, any cross-coupling can then be detrimental to the resulting image. The PID controller was shown to provide much worse closed-loop characteristics than the other controllers as the resonant peaks were not properly damped and got excited by the reference signal. In terms of cross-coupling gains however,  $\mathcal{H}_\infty$  MIMO controller was found to give only marginally better theoretical results than the others, and the differences were negligible compared to the noise and disturbance level of the system. So in practice there were no significant differences between the three controllers in terms of cross-coupling gain, and all controllers seemed to dampen the cross-coupling significantly. Thus, the added complexity of a MIMO control strategy does not justify the small benefits achieved.

We suspect that we may get different results under two conditions. First, by designing a controller with higher bandwidth, as this will not dampen out the resonant frequencies including the cross-gains as much. Secondly, by implementing on a system with worse cross-coupling characteristics than the AFM we performed experiments on.

The implementability issues involved with running a complex controller such as the  $\mathcal{H}_\infty$  MIMO controller were also discussed. Such a controller can be represented by a continuous-time state-space model, but for an actual implementation the ordinary differential equations needs to be solved at discrete time-steps. Many popular solver types are based on the family of explicit Runge-Kutta (ERK) methods. These solvers become unstable if the step-size is too large. At the same time, the complexity of a controller running on hardware with limited computational power puts a lower limit on the step-size the hardware needs to perform the necessary calculations.

If the lower limit on the step-size (due to computation time) and the upper limit (for stability) do not overlap, it will be impossible to implement the controller. Then one must either design a computationally simpler controller, a controller with smaller (in absolute value) eigenvalues, use more powerful hardware, or try to change the solver method. We have shown both how to reduce the lower limit by performing model reduction on the controller, as well as analyzing the controller to find the upper limit.

To reduce the computational complexity of a controller and thus make it more easily implementable, we showed how to perform model reduction on it. We carried out model reduction on our most complex  $\mathcal{H}_\infty$  MIMO controller, and showed that it could be reduced from 18th order to 10th order without any significant impact on performance or stability. With reduced controller order we achieved stability using a less complex solver type. So not only do we have reduced computational complexity from the lower order itself, but the solver also is less computationally demanding. This combined benefit resulted in a 46.7% reduction in task execution time. Controllers of lower order than 10th were not robustly stable.

Lastly, we presented a more analytical approach to determine the numerical stability of an ERK method applied to a controller. We used this to show how to determine the maximum step-size of a given controller and solver type, and determined that the placement of the eigenvalues of the controller was decisive for stability. We performed the calculations on our  $\mathcal{H}_\infty$  MIMO controller and observed that the results complied with the instability experienced in simulations.

## 9.1. Future Work

In the cross-coupling experiments we limited the bandwidth of the controllers both to make it implementable on our experimental setup, and secondly to be able to compare it to a PID controller of similar performance. We postulated that a MIMO controller will be more important for the cross-coupling gains at larger bandwidths. This could be determined by new experiments using larger bandwidth on the controllers. The PID controller can be dropped as this will not be able to achieve higher bandwidth due to the resonant gain. It would also be interesting to investigate the differences of MIMO control versus SISO control on a system with worse cross-coupling characteristics.

A further extension to the cross-coupling experiments would be to find an expression for how much is to gain by using a MIMO approach compared to a SISO approach

in terms of cross-coupling. Such an expression would depend on the plant's inherent cross-coupling gains as well as the desired controller bandwidth.

Since we have shown that the eigenvalues of the controller is central to the stability of the numerical solver, it would be advantageous to have a model order reduction method that reduces the eigenvalues of the controller as much as possible with a limit to the allowable error. That is, design a model reduction method which solves

$$\min_{K_r} |\lambda(K_r)| \quad \text{subject to } \|T - T_r\|_\infty < \epsilon \quad (9.1)$$

for some value of  $\epsilon$ .

An implicit Runge-Kutta (IRK) method is stable for all controllers with eigenvalues in the left half plane. Yet, we experienced in our simulations that the solutions to the IRK method available in Simulink exploded at certain intervals. Nevertheless, it would be interesting to further analyze the stability and feasibility of using a fixed-step IRK method for real-time implementation.

We have only considered continuous-time models in our numerical stability analysis. Some implementations use discrete-time models. Future studies could investigate the stability properties of such models and discuss this in relationship to model order reduction and computational complexity.

Another topic of future work which could be considered is to find how a different solver affects the minimum step-size versus maximum step-size. Consider [Figure 8.2](#), if a more complex solver method increases the max. step-size more than it increases the min. step-size, it would be advantageous to switch to this solver in terms of numerical stability. An expression for this trade-off would involve the controller order and the constant overhead of the numerical calculations.

Lastly, it would be interesting to see how the controllers designed in this thesis compares to the stock controller on the AFM. This could be done by implementing our controller and performing an imaging task.





# Appendix A

## Definitions and Properties

### A.1. $\mathcal{H}_2$ and $\mathcal{H}_\infty$ Norm

Although with its somewhat intimidating name, the  $\mathcal{H}_\infty$  norm has in fact a simple definition.

#### Definition: $\mathcal{H}_\infty$ norm

The  $\mathcal{H}_\infty$  norm of a proper transfer function  $G(s)$  is

$$\|G(s)\|_\infty \triangleq \max_{\omega} |G(j\omega)| \quad (\text{A.1})$$

So the  $\mathcal{H}_\infty$  norm is simply the peak value of the magnitude of  $G(j\omega)$ . Note that if  $G(s)$  is a matrix this definition is insufficient, in this case we have

$$\|G(s)\|_\infty \triangleq \max_{\omega} \bar{\sigma}(G(j\omega)) \quad (\text{A.2})$$

where  $\bar{\sigma}(\cdot)$  is as defined on page 17. The expression has a mathematical origin, where the  $\mathcal{H}$  stands for the *Hardy space*, and the  $\infty$  comes from the notion that we can pick out the maximum magnitude by raising the signal to the power of infinity, i.e.

$$\max_{\omega} |G(j\omega)| = \lim_{p \rightarrow \infty} \left( \int_{-\infty}^{\infty} |G(j\omega)|^p d\omega \right)^{1/p} \quad (\text{A.3})$$

By considering this equation we can extend the concept to define a  $\mathcal{H}_2$  norm.

**Definition:  $\mathcal{H}_2$  norm**

The  $\mathcal{H}_2$  norm of a strictly proper transfer function  $G(s)$  is

$$\|G(s)\|_2 \triangleq \left( \frac{1}{2\pi} \int_{-\infty}^{\infty} |G(j\omega)|^2 d\omega \right)^{1/2} \quad (\text{A.4})$$

The relationship between the norm of a time domain signal and a transfer function norm is worth noting. For a stable and strictly proper transfer function  $G(s)$  with inputs  $u(t)$  and outputs  $y(t)$ , we have the relationships given in [Table A.1](#). We can see that the  $\mathcal{H}_2$  norm is equal to the 2-norm of the output-signal from impulse response, and that a sine-input is infinite. In fact, the  $\mathcal{H}_2$  norm must be strictly proper else it will become infinite [55]. Additional details and proofs for the table is given by Doyle et al. [21].

**Table A.1:** Signal norms and transfer function norms for two types of input signals, with  $\delta(t)$  being the unit impulse function.

	$u(t) = \delta(t)$	$u(t) = \sin(\omega_0 t)$
$\ y(t)\ _2$	$\ G(s)\ _2$	$\infty$
$\ y(t)\ _\infty$	$\ G(s)\ _\infty$	$ G(j\omega_0) $

## A.2. Hankel Norm and Hankel Singular Value

The Hankel norm aims to provide a measure of how much past input values into the stable system  $G(s)$  can affect the output in the future. We have used the definition from Skogestad and Postlethwaite [55].

**Definition: Hankel norm**

Consider a stable transfer function  $G(s)$ , input signal  $u(t)$ , and output signal  $y(t)$ , then the Hankel norm of  $G(s)$  is

$$\|G(s)\|_H \triangleq \max_{u(t)} \frac{\sqrt{\int_0^\infty \|y(\tau)\|_2^2 d\tau}}{\sqrt{\int_{-\infty}^0 \|u(\tau)\|_2^2 d\tau}} \quad (\text{A.5})$$

One way to think about it is to consider a given amount of input energy, then the input signal is shaped such that we maximize the amount of energy at the output after we stop applying any more input.

The closely related Hankel singular values of  $G(s)$  are central in the model order reduction theory.

**Definition: Hankel singular values**

The Hankel singular values of a stable transfer function  $G(s)$  are

$$\sigma_i \triangleq \sqrt{\lambda_i(PQ)}, \quad i = 1, \dots, n \quad (\text{A.6})$$

where  $\lambda_i$  is the positive  $i$ th eigenvalue of  $PQ$ , and  $P, Q$  are the solutions to the Lyapunov equations

$$AP + PA^T + BB^T = 0 \quad (\text{A.7})$$

$$A^T Q + QA + C^T C = 0 \quad (\text{A.8})$$

for a minimal state-space realization  $(A, B, C, D)$  of  $G(s)$ .

The Hankel singular values are called *ordered* if we have  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n > 0$ .

It can be shown that the Hankel norm  $\|G(s)\|_H$  is equal to the largest of the Hankel singular values  $\sigma_i$ . It is also closely related to the  $\mathcal{H}_\infty$  norm with the relationship [55]

$$\|G(s)\|_H \equiv \max_i \sigma_i \leq \|G(s)\|_\infty \leq 2 \sum_{i=1}^n \sigma_i \quad (\text{A.9})$$

### A.3. Butcher Tableau for Dormand-Prince Methods

The Butcher tableaus for the explicit Runge-Kutta methods Dormand-Prince erk5 and erk8 are given in Table A.2 and Table A.3 respectively.

**Table A.2:** Butcher tableau for Dormand-Prince (erk5) method [19]

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0



## Appendix B

# Contents of Attached Zip-file

Suggested execution order to generate the various controllers and run a simulation:

1. IDENTIFICATION.M
2. UNCERTAINTY\_WEIGHTING.M
3. PID\_CONTROLLER.M
4. H\_INF\_MIXED\_SENSITIVITY.M
5. CONTROLLER\_SIM.SLX

### B.1. File Descriptions

#### Pdf files

**thesis.pdf** Digital copy of this thesis.

**paper.pdf** Digital copy of the paper written based on this thesis.

#### Matlab files

**export\_plot.m** Function used to save plots generated in other files.

**h\_inf\_mixed\_sensitivity.m** Generates the  $\mathcal{H}_\infty$  MIMO and  $\mathcal{H}_\infty$  SISO controller.

**identification.m** Identifies the nominal plant model  $G$  based on the frequency response data.

**Phi.m** Returns the stability function value  $\Phi(z)$  for the explicit Runge-Kutta methods in [Table 8.1](#).

**pid\_controller.m** Generates the PID controller.

**plot\_xpc\_data.m** Generates several of the plots used in the thesis, using data from the experiments.

**rk\_stability.m** Finds the maximum stable step-size for a given controller and generates the plot of the stability regions for several explicit Runge-Kutta solvers.

**test\_bench\_raster\_pattern.m** Runs experiments with raster pattern reference signal at several frequencies. Saves and plots the results.

**test\_bench\_reduced\_controller.m** This code generates the reduced order controllers, runs a simulation on all of them using different solver types and step-sizes, records the time spent on each simulation, then saves and plots the results.

**uncertainty\_weighting.m** Uses the experimental data and nominal plant  $G$  to find an appropriate uncertainty weighting  $W$ .

## Simulink files

**controller\_sim.slx** Simulink model for simulation of the various controllers.

**controller\_xpc.slx** Simulink model for implementation of the various controllers on the xPC hardware.

**controller\_xpc\_stripped.slx** Simulink model for implementation of any controller on the xPC hardware, stripped of all unnecessary blocks and safety checks. Used in the computational complexity experiment for more accurate timing results.

## Data files

The following folders contains frequency response data from the SR780 device, and outputs from scanning experiments. All data files are provided in .MAT file-type which can be loaded into Matlab.

**data/01 - Plant Response  $\hat{G}_{\text{hat}}$ /** Contains the three frequency responses gathered of the plant in the four directions  $\hat{G}_{xx}, \hat{G}_{xy}, \hat{G}_{yx}, \hat{G}_{yy}$ .

**data/02 - PID/** Contains closed-loop frequency response of the PID controller in the four directions  $\hat{T}_{xx}, \hat{T}_{xy}, \hat{T}_{yx}, \hat{T}_{yy}$ , as well as outputs from scanning experiments.

**data/03 - Hinf SISO/** Contains closed-loop frequency response of the  $\mathcal{H}_{\infty}$  SISO controller in the four directions  $\hat{T}_{xx}, \hat{T}_{xy}, \hat{T}_{yx}, \hat{T}_{yy}$ , as well as outputs from scanning experiments.

**data/04 - Hinf MIMO/** Contains closed-loop frequency response of the  $\mathcal{H}_{\infty}$  MIMO controller in the four directions  $\hat{T}_{xx}, \hat{T}_{xy}, \hat{T}_{yx}, \hat{T}_{yy}$ , as well as outputs from scanning experiments.

## Appendix C

# Matlab Code

Some of the code used to run the simulations, experiments and generate the plots is given here. See the zip-file attachment for additional files.

### C.1. h\_inf\_mixed\_sensitivity.m

```
1 % Generates the Hinf MIMO and Hinf SISO controllers
2 %
3 %% Bandwidth
4 M=1.2;
5 wb=70*2*pi;
6 A=1e-4;
7 Wp=tf([1/M wb], [1 wb*A]);
8 Wu=0;
9 Wu=tf([1/1.2 0], [1 wb]);
10
11 %Find H-inf controller
12 [K,ghinf,GAM] = mixsyn(G_nodelay,Wp,Wu,W_O);
13
14 [K_SISO11,ghinf,GAM] = mixsyn(G_nodelay(1,1),Wp,Wu,W_O);
15 [K_SISO22,ghinf,GAM] = mixsyn(G_nodelay(2,2),Wp,Wu,W_O);
16
17 K_SISO = [K_SISO11, 0; 0, K_SISO22];
18
19 L = G*K;
20 T = (eye(2) + L)\L;
21 S = eye(2) - T;
22
23 L_SISO = G*K_SISO;
24 T_SISO = (eye(2) + L_SISO)\L_SISO;
25 S_SISO = eye(2) - T_SISO;
26
27 h=figure(9);
28 hb=sigmaplot(S,T);
29 setoptions(hb,'FreqUnits','Hz');
30 legend1 = legend('S','T','Location','SouthWest');
31 title('H-inf MIMO');
32 xlim([1,1e5]);
33 ylim([-50,10]);
34 grid on
```

```

35
36 export_plot('experiment_03_MIMO_h_inf_MIMO_sensitivity.pdf');
37
38 hb=sigmaplot(S_SISO,T_SISO);
39 setoptions(hb,'FreqUnits','Hz');
40 legend1 = legend('S','T','Location','SouthWest');
41 title('H-inf SISO');
42 xlim([1,1e5]);
43 ylim([-50,10]);
44 grid on
45
46 export_plot('experiment_03_MIMO_h_inf_SISO_sensitivity.pdf');
47
48
49 norm(pade(W_O*T_PID,5),Inf)
50 norm(pade(W_O*T_SISO,5),Inf)
51 norm(pade(W_O*T,5),Inf)

```

## C.2. identification.m

```

1  %% Identifies the nominal plant model G based on the frequency ...
   response data
2
3  close all
4  clear
5  clc
6
7  h=figure(9);
8
9  %% Simulation variables
10
11  Dt_step = 4e-5;
12
13
14  %% --- System identification ---
15  bodeOpt = bodeoptions;
16  bodeOpt.FreqUnits = 'Hz';
17  bodeOpt.PhaseMatching = 'on';
18  bodeOpt.PhaseMatchingFreq = 100;
19  bodeOpt.PhaseMatchingValue = 0;
20  bodeOpt.YLim = {[ -80,25];[-360,0]}; % {maglimits; phaselimits}
21  bodeOpt.YLimMode = {'manual';'auto'};
22
23  %% Time delay
24  load('data/01 - Plant Response G_hat/Run 1/GXX.MAT');
25
26  f = gxx(5:end,1);
27  d = gxx(5:end,2);
28  p = phase(gxx(5:end,2));
29
30  Gxx_experimental = frd(d,f,'FrequencyUnit','Hz');
31
32  % Time delay
33  l = fit(f(5:156),p(5:156),'poly1');
34
35  plot(l,f,p);
36  xlabel('Frequency [hz]'); ylabel('Phase [rad]');
37  grid on
38
39  export_plot('experiment_03_MIMO_Gxx_delay.pdf');
40
41  Td = -l.p1/(2*pi);

```



```

42
43
44 %% Gxx
45 np = 3;
46 nz = 2;
47
48 wpass = [20,140; 358,370; 540,548; 666,672; 792,794; 834,846]*2*pi;
49 opt = tfestOptions('Focus',wpass);
50 Gxx = tfest(Gxx_experimental,np,nz,Td,opt);
51
52 bode(Gxx_experimental,Gxx,bodeOpt);
53 childHnd=get(h, 'Children'); axes1=childHnd(3);
54 legend1 = legend(axes1,'$\hat{G}$', '$G$', 'Location', 'NorthWest');
55 set(legend1, 'Interpreter', 'latex');
56 title('');
57 xlim([20 3200]);
58 grid on
59
60 export_plot('experiment_03_MIMO_Gxx_id.pdf');
61
62 %% Gyy
63 load('data/01 - Plant Response G_hat/Run 1/GYY.MAT');
64
65 f = gyy(5:end,1);
66 d = gyy(5:end,2);
67 p = phase(gyy(5:end,2));
68
69 Gyy_experimental = frd(d,f,'FrequencyUnit','Hz');
70
71
72 % Use same time delay as from Gxx: Td
73
74 % System identification
75 np = 3;
76 nz = 2;
77
78 wpass = [20,140; 358,370; 540,548; 666,672; 752,821; 1100,1108]*2*pi;
79 opt = tfestOptions('Focus',wpass);
80 Gyy = tfest(Gyy_experimental,np,nz,Td,opt);
81
82 bode(Gyy_experimental,Gyy,bodeOpt);
83 childHnd=get(gcf, 'Children'); axes1=childHnd(3);
84 legend1 = legend(axes1,'$\hat{G}$', '$G$', 'Location', 'SouthWest');
85 set(legend1, 'Interpreter', 'latex');
86 title('');
87 xlim([20 3200]);
88 grid on
89
90 export_plot('experiment_03_MIMO_Gyy_id.pdf');
91
92
93
94 %% Gxy
95 load('data/01 - Plant Response G_hat/Run 1/GXY.MAT');
96
97 f = gxy(5:end,1);
98 d = gxy(5:end,2);
99 p = phase(gxy(5:end,2));
100
101 Gxy_experimental = frd(d,f,'FrequencyUnit','Hz');
102
103 % Use same time delay as from Gxx: Td
104
105 % System identification
106 np = 2;
107 nz = 2;
108
109 wpass = [10,300; 756,760;]*2*pi;
110 opt = tfestOptions('Focus',wpass);

```

```

111 Gxy = tfest(Gxy_experimental,np,nz,Td,opt);
112
113 bode(Gxy_experimental,Gxy,bodeOpt);
114 childHnd=get(gcf, 'Children'); axes1=childHnd(3);
115 legend1 = legend(axes1, '$\hat{G}$', '$G$', 'Location', 'NorthWest');
116 set(legend1, 'Interpreter', 'latex');
117 title('');
118 xlim([20 3200]);
119 grid on
120
121 export_plot('experiment_03_MIMO_Gxy_id.pdf');
122
123
124 %% Gyx
125 load('data/01 - Plant Response G_hat/Run 1/GYX.MAT');
126
127 f = gyx(5:end,1);
128 d = gyx(5:end,2);
129 p = phase(gyx(5:end,2));
130
131 Gyx_experimental = frd(d,f, 'FrequencyUnit', 'Hz');
132
133 % Use same time delay as from Gxx: Td
134
135 % System identification
136 np = 2;
137 nz = 2;
138
139 wpass = [10,300; 768,772;]*2*pi;
140 opt = tfestOptions('Focus',wpass);
141 Gyx = tfest(Gyx_experimental,np,nz,Td,opt);
142
143 bode(Gyx_experimental,Gyx,bodeOpt);
144 childHnd=get(gcf, 'Children'); axes1=childHnd(3);
145 legend1 = legend(axes1, '$\hat{G}$', '$G$', 'Location', 'NorthWest');
146 set(legend1, 'Interpreter', 'latex');
147 title('');
148 xlim([20 3200]);
149 grid on
150
151 export_plot('experiment_03_MIMO_Gyx_id.pdf');
152
153
154 %% Wrap it up
155
156 G = [Gxx, Gxy; Gyx, Gyy];
157 G_nodelay = pade(tf(G),0);

```

### C.3. Phi.m

```

1 function output = Phi( z, p )
2 %Returns the stability function value of z for some explicit ...
   Runge-Kutta
3 %methods of order p. Valid order values: [1:5, 8]
4
5 if any( p == 1:4 )
6     %Any ERK with p == s
7     output = 1;
8     for i=1:p
9         output = output + z.^i ./ factorial(i);
10    end
11 elseif p == 5

```

```

12     %Dormand-Prince 5
13     A = [0 0 0 0 0 0 0;...
14           1/5 0 0 0 0 0 0;...
15           3/40 9/40 0 0 0 0 0;...
16           44/45 -56/15 32/9 0 0 0 0;...
17           19372/6561 -25360/2187 64448/6561 -212/729 0 0 0;...
18           9017/3168 -355/33 46732/5247 49/176 -5103/18656 0 0;...
19           35/384 0 500/1113 125/192 -2187/6784 11/84 0
20     ];
21     b = [35/384 0 500/1113 125/192 -2187/6784 11/84 0]';
22
23     PhiERK = @(z) det( eye(size(A)) - z*A + z*(ones(size(b,1),1)*b') );
24
25     output = zeros(size(z));
26     for i=1:size(z,1)
27         for j = 1:size(z,2)
28             output(i,j) = PhiERK( z(i,j) );
29         end
30     end
31
32     elseif p == 8
33         %Dormand-Prince 8
34         A = [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;...
35             1.0 / 18.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;...
36             1.0 / 48.0, 1.0 / 16.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;...
37             1.0 / 32.0, 0, 3.0 / 32.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;...
38             5.0 / 16.0, 0, -75.0 / 64.0, 75.0 / 64.0, 0, 0, 0, 0, 0, 0, ...
39             0, 0, 0;...
40             3.0 / 80.0, 0, 0, 3.0 / 16.0, 3.0 / 20.0, 0, 0, 0, 0, 0, ...
41             0, 0;...
42             29443841.0 / 614563906.0, 0, 0, 77736538.0 / 692538347.0, ...
43             -28693883.0 / 1125.0e+6, 23124283.0 / 18.0e+8, 0, 0, 0, ...
44             0, 0, 0;...
45             16016141.0 / 946692911.0, 0, 0, 61564180.0 / 158732637.0, ...
46             22789713.0 / 633445777.0, 545815736.0 / 2771057229.0, ...
47             -180193667.0 / 1043307555.0, 0, 0, 0, 0, 0;...
48             39632708.0 / 573591083.0, 0, 0, -433636366.0 / 683701615.0, ...
49             -421739975.0 / 2616292301.0, 100302831.0 / 723423059.0, ...
50             790204164.0 / 839813087.0, 800635310.0 / 3783071287.0, 0, ...
51             0, 0, 0;...
52             246121993.0 / 1340847787.0, 0, 0, -37695042795.0 / ...
53             15268766246.0, -309121744.0 / 1061227803.0, -12992083.0 / ...
54             490766935.0, 6005943493.0 / 2108947869.0, 393006217.0 / ...
55             1396673457.0, 123872331.0 / 1001029789.0, 0, 0, 0;...
56             -1028468189.0 / 846180014.0, 0, 0, 8478235783.0 / ...
57             508512852.0, 1311729495.0 / 1432422823.0, -10304129995.0 ...
58             / 1701304382.0, -48777925059.0 / 3047939560.0, ...
59             15336726248.0 / 1032824649.0, -45442868181.0 / ...
60             3398467696.0, 3065993473.0 / 597172653.0, 0, 0;...
61             185892177.0 / 718116043.0, 0, 0, -3185094517.0 / ...
62             667107341.0, -477755414.0 / 1098053517.0, -703635378.0 / ...
63             230739211.0, 5731566787.0 / 1027545527.0, 5232866602.0 / ...
64             850066563.0, -4093664535.0 / 808688257.0, 3962137247.0 / ...
65             1805957418.0, 65686358.0 / 487910083.0, 0, 0;...
66             403863854.0 / 491063109.0, 0, 0, -5068492393.0 / ...
67             434740067.0, -411421997.0 / 543043805.0, 652783627.0 / ...
68             914296604.0, 11173962825.0 / 925320556.0, -13158990841.0 ...
69             / 6184727034.0, 3936647629.0 / 1978049680.0, -160528059.0 ...
70             / 685178525.0, 248638103.0 / 1413531060.0, 0, 0;...
71     ];
72     b = [14005451.0 / 335480064.0, 0, 0, 0, -59238493.0 / ...
73         1068277825.0, 181606767.0 / 758867731.0, 561292985.0 / ...
74         797845732.0, -1041891430.0 / 1371343529.0, 760417239.0 / ...
75         1151165299.0, 118820643.0 / 751138087.0, -528747749.0 / ...
76         2220607170.0, 1.0 / 4.0]';
77
78     PhiERK = @(z) det( eye(size(A)) - z*A + z*(ones(size(b,1),1)*b') );
79
80     output = zeros(size(z));

```

```

53     for i=1:size(z,1)
54         for j = 1:size(z,2)
55             output(i,j) = PhiERK( z(i,j));
56         end
57     end
58 end
59
60
61 end

```

## C.4. rk\_stability.m

```

1  %Finds the maximum stable step-size for a given controller and ...
   generates
2  %the plot of the stability regions for several explicit Runge-Kutta
3  %solvers.
4
5  % Assumes controller K is already generated
6  for k_r_orders = 10:17
7      % Create the reduced controller
8      new_order = k_r_orders;
9      [K_bal,g] = balreal(K);
10     elim = [ zeros(new_order,1); ones(size(g,1)-new_order,1) ];
11     K_r = modred(K_bal,logical(elim));
12
13     K_eig = sort(eig(K), 'descend');
14     K_r_eig = sort(eig(K_r), 'descend');
15
16     % Binary search to find the maximum step-size
17     erk_orders = [1:5,8];
18     max_step_sizes = zeros(size(erk_orders));
19
20     for i = 1:size(erk_orders,2)
21         low = 0;
22         high = 1;
23         epsilon = 1e-9;
24
25         while (high - low) > epsilon
26             h = (high - low) / 2 + low;
27             z = h * K_eig;
28             if abs( Phi(z,erk_orders(i)) ) <= 1
29                 low = h;
30             else
31                 high = h;
32             end
33         end
34
35         max_step_sizes(1,i) = low;
36     end
37
38     display(regexprep( num2str(max_step_sizes*1e6,4), ...
39         '([0-9])\s+([0-9])', '$1\t$2'))
40 end
41 %% plot
42 [re, im] = meshgrid(-6.0:.01:3, -6:.01:6);
43 z = complex(re,im);
44 ineq = zeros(size(z));
45 figure(9); hold off;
46
47 for i=erk_orders
48     ineq = ineq + (ineq | (abs(Phi(z,i))<=1));

```

## C.5 test\_bench\_raster\_pattern.m

```
49 end
50
51 [C,h] = contourf(re,im, ineq, [1,2,3,4,5,5.5]) ;
52
53 grid on;
54 xlabel('Re($z$)', 'Interpreter', 'latex');
55 ylabel('Im($z$)', 'Interpreter', 'latex');
56
57 hold on;
58 re = real(K_eig*max_step_sizes(end));
59 im = imag(K_eig*max_step_sizes(end));
60 plot(re,im, 'om');
61 re = real(K_r_eig*max_step_sizes(end));
62 im = imag(K_r_eig*max_step_sizes(end));
63 plot(re,im, 'xy');
64 daspect([1 1 1])
65 hold off
66
67 export_plot('experiment_07_rk_stability_K_18_eigenvalues.pdf');
```

## C.5. test\_bench\_raster\_pattern.m

```
1 %% This code runs the AFM platform using the simulink model ...
   controller_xpc
2 % with a raster pattern reference signal at the frequencies ...
   specified in
3 % all_freq, then saves the data and plots the results.
4
5 % Initialize
6 tg = xpc('TargetPC1');
7 tg.load('controller_xpc');
8
9 all_freq = [1 10 20 50 100 200 300 400 662 760];
10
11 for ScanRate=all_freq
12
13 LengthPerVolt = 4.664678428174072; % um/V
14 Offset = [7.4 6.8];
15
16 setparam(tg, getparamid(tg, 'Offset_XY', 'Value'), Offset); % V
17
18 ScanWidth = 2; % um
19 NumLines = 10 - 5*(ScanRate < 10);
20
21 LineLength = ScanWidth / NumLines;
22 ScanSpeedY = LineLength * ScanRate; % um/s
23 ScanWidthV = ScanWidth / LengthPerVolt; % V
24 ScanRampSlopeYV = ScanSpeedY / LengthPerVolt; % V
25 SimTime = ScanWidth / ScanSpeedY + 1/ScanRate;
26
27 setparam(tg, getparamid(tg, 'Step_Ref', 'Time'), 1/ScanRate); % ...
   Delay scanning by one period to stabilize
28 setparam(tg, getparamid(tg, 'Sequence_X/Constant', 'Value'), ...
   1/ScanRate); % Scan Period
29 setparam(tg, getparamid(tg, 'Sequence_X/Look-Up Table1', ...
   'InputValues'), [0 .5 1]/ScanRate); % Time
30 setparam(tg, getparamid(tg, 'Sequence_X/Look-Up Table1', 'Table'), ...
   [0 ScanWidthV 0]); % Output
31
32 setparam(tg, getparamid(tg, 'Ramp_Y/Step', 'After'), ...
   ScanRampSlopeYV );
33
```

```

34 set(tg, 'StopTime', SimTime);
35
36 disp( ['Simulation time: ', num2str(SimTime), ' s'] );
37
38 tg.start();
39
40 while( ~strcmp(tg.status, 'stopped') )
41     pause(0.25);
42 end
43
44 % Plot
45
46 start_index = floor((1/ScanRate)/Dt_step);
47
48 td = tg.TimeLog(start_index:end)-tg.TimeLog(start_index);
49 xd = tg.OutputLog(start_index:end,[1 3]) * LengthPerVolt; % um
50 yd = tg.OutputLog(start_index:end,[2 4]) * LengthPerVolt; % ...
51     Assuming um/V value is same as for x (should check this)
52 exd = xd(:,2)-xd(:,1) + sum(xd(:,2),1)/size(xd,1);
53 eyd = yd(:,2)-yd(:,1) + sum(yd(:,2),1)/size(yd,1);
54
55 %save(['..\02 - PID\Scanning\data_',int2str(ScanRate),'hz.mat'], ...
56     'td', 'xd', 'yd');
57 %save(['..\03 - Hinf ...
58     SISO\Scanning\data_',int2str(ScanRate),'hz.mat'], 'td', 'xd', 'yd');
59 %save(['..\04 - Hinf ...
60     MIMO\Scanning\data_',int2str(ScanRate),'hz.mat'], 'td', 'xd', 'yd');
61
62 plot(td,xd,td,yd,td,exd,'r',td,eyd);
63
64 plot(xd,yd);
65 set(gca,'PlotBoxAspectRatio',[1 1 1]);
66 grid on;
67 xlabel('Displacement x [um]'); ylabel('Displacement y [um]');
68
69 std(exd)
70 std(eyd)
71 end

```

## C.6. test\_bench\_reduced\_controllers.m

```

1 % This code generates the reduced order controllers, runs a ...
2 simulation on
3 % all of them using different solver types and step-sizes, records ...
4 the time
5 % spent on each simulation, then saves and plots the results.
6
7 new_order_it = 3:18;
8 solver_type_it = {'ode1','ode2','ode3','ode5','ode8','ode14x'};
9 step_time_it = [1e-4, 1e-5, 4e-5, 1e-6];
10
11 resSimTime = zeros(size(new_order_it,2), size(solver_type_it,2), ...
12     size(step_time_it,2));
13 resRMSx = Inf*ones(size(new_order_it,2), size(solver_type_it,2), ...
14     size(step_time_it,2));
15 resRMSy = Inf*ones(size(new_order_it,2), size(solver_type_it,2), ...
16     size(step_time_it,2));
17 resDT = zeros(size(new_order_it,2),1);
18 resWT = zeros(size(new_order_it,2),1);

```

```

14 resEig = zeros(size(new_order_it,2),18);
15
16 for new_order_index = ...
17     1:size(new_order_it,2);%[7,10,13,17,18];%[4,7,9,10]%2:17
18     % Step response full order controller
19     K_r = K;
20     simout = sim('controller_xpc','StopTime', '.02');
21     y = simout.get('yout');
22     t = simout.get('tout');
23
24     % Generate reduced order controller
25     new_order = new_order_it(new_order_index);
26     if new_order == 18
27         K_r = K;
28     else
29         [K_bal,g] = balreal(K);
30         elim = [ zeros(new_order,1); ones(size(g,1)-new_order,1) ];
31         K_r = modred(K_bal,logical(elim));
32
33         L_r = G*K_r;
34         T_r = (eye(2) + L_r)\L_r;
35         S_r = eye(2) - T_r;
36
37         dT = T-T_r;
38         resDT(new_order_index,1) = norm(pade(dT,5),inf);
39         resWT(new_order_index,1) = norm(pade(W_O*T_r,5),inf);
40         eigVals = eig(K_r);
41         resEig(new_order_index,1) = eigVals(1);
42         disp(['dT: ',num2str( norm(dT,inf), '%3.2e' )]);
43     end
44
45     eigVals = sort(eig(K_r))
46     resEig(new_order_index,1:size(eigVals,1)) = eigVals;
47
48     solver_type = char(solver_type_it(1));
49     set_param('hinf_ms_controller_stripped','Solver',solver_type);
50
51     h=figure(9);
52     set(h,'DefaultAxesColorOrder',[0 0 1;0 .5 0;0 0 1;0 .5 0])
53     hb=sigmaplot(S,'b',S_r,'g',T,'b',T_r,'g');
54     setoptions(hb,'FreqUnits','Hz'); legend('S, T','S_r, ...
55         T_r','Location','SouthWest');
56     title([num2str(new_order),'th order']); xlim([1,3e4]); ...
57         ylim([-35,10]); grid on;
58     export_plot(['experiment_04_S_T_order_', ...
59         num2str(new_order),'.pdf']);
60
61     Step response reduced order controller
62     simout = sim('sim_closed_loop','StopTime', '.02');
63     y_r = simout.get('yout');
64     t_r = simout.get('tout');
65
66     plot(t,y(:,1),t_r,y_r(:,1)); % step in r1
67     hold on;
68     plot( t,y(:,2),t_r,y_r(:,2));
69     hold off;
70     legend('18th order (full)',[num2str(new_order),'th order'], ...
71         'Location','Best'); xlabel('time (s)'); ylabel('output (V)'); ...
72     grid on;
73     title(['Step response']);
74     export_plot(['experiment_04_step_sim_order_', ...
75         num2str(new_order),'.pdf']);
76
77     % Run simulation for various solvers and step time
78     for solver_type_index = 1:size(solver_type_it,2);
79         solver_type = solver_type_it(solver_type_index);
80         for step_time_index = 1:size(step_time_it,2);
81             step_time = step_time_it(step_time_index);

```

```

76     % Simulation
77     Dt_step = step_time;
78     disp(['Order: ', num2str(new_order), ' Solver: ...
79         ', char(solver_type), ' StepTime: ', num2str(Dt_step)]);
80     %set_param('sim_closed_loop','Solver','ode14x');
81     try
82         t0 = tic;
83         simout = sim('sim_closed_loop','StopTime', ...
84             '.2','Solver',char(solver_type));
85         elapsedTime = toc(t0);
86         y = simout.get('yout');
87
88         resSimTime(new_order_index, solver_type_index, ...
89             step_time_index) = elapsedTime;
90         resRMSx(new_order_index, solver_type_index, ...
91             step_time_index) = rms(y(:,1));
92         resRMSy(new_order_index, solver_type_index, ...
93             step_time_index) = rms(y(:,2));
94
95         t = simout.get('tout');
96         plot(t,y);
97         hold on
98
99         disp(['RMS x: ', num2str( rms( y(:,1) ) ), ' y: ...
100             ', num2str( rms( y(:,2) ) )]);
101     catch err
102         disp(['Unstable Error:',err.identified]);
103     end
104     disp(resSimTime(new_order_index, solver_type_index, ...
105         step_time_index));
106
107     end
108     end
109
110     hold off;
111     set_param('hinf_ms_controller_stripped','Solver',char(solver_type));
112
113     %%
114     GAM
115     norm(pade(W_0*T,5),Inf)
116
117     %%
118     figure1 = figure(6);
119     axes1 = axes('Parent',figure1,'YScale','log','YMinorTick','on',...
120         'YMinorGrid','on',...
121         'XTick',2:17);
122     xlim(axes1,[1.5 17.5]); box(axes1,'on');
123     grid(axes1,'on');
124     hold(axes1,'all');
125     xlabel('Model Order'); ylabel('||T-T_r||_\infty');
126     set(gcf,'Position',[680 558 460 420]);
127
128     bar(new_order_it,resDT,'BaseValue',0.001);
129     export_plot('experiment_04_hinf_error_norm_bar.pdf',figure1);
130
131     %%
132     figure1 = figure(6);
133     axes1 = axes('Parent',figure1,'YScale','log','YMinorTick','on',...
134         'YMinorGrid','on',...
135         'XTick',2:17);
136     xlim(axes1,[1.5 17.5]); box(axes1,'on');
137     grid(axes1,'on');
138     hold(axes1,'all');
139     xlabel('Model Order'); ylabel('||W T_r||_\infty');
140     set(gcf,'Position',[680 558 460 420]);
141
142     bar(new_order_it,resWT,'BaseValue',.1);
143     export_plot('experiment_04_W_x_T_robustness.pdf',figure1);

```



## Appendix D

# Paper

The next pages contain a paper written based on the main contents of this thesis. The paper will be submitted to the IFAC World Congress 2014 at a later date possibly with modifications.

# $\mathcal{H}_\infty$ Reduced Order Control for Nanopositioning: Numerical Implementability

Michael R. P. Ragazzon\* Arnfinn A. Eielsen\*\*  
J. Tommy Gravdahl\*\*\*

\* *Department of Engineering Cybernetics, Norwegian University of  
Science and Technology, Trondheim, Norway,  
(e-mail: ragazzon@stud.ntnu.no)*

\*\* *Department of Engineering Cybernetics, Norwegian University of  
Science and Technology, Trondheim, Norway,  
(e-mail: eielsen@itk.ntnu.no)*

\*\*\* *Department of Engineering Cybernetics, Norwegian University of  
Science and Technology, Trondheim, Norway,  
(e-mail: Tommy.Gravdahl@itk.ntnu.no)*

---

**Abstract:** A robust  $\mathcal{H}_\infty$  multiple-input multiple-output (MIMO) controller is designed and implemented for the lateral stage of an Atomic Force Microscope (AFM). Such a model-based controller can quickly become complex and may be difficult to run in real-time on hardware with limited computational power. Handling this difficulty is the main topic of this paper. The resulting controller can be considered to be stiff, which is characterized by a large spread of eigenvalues. Continuous-time systems running in real-time are usually solved using explicit Runge-Kutta (ERK) methods, which easily becomes unstable for stiff systems. We show how small the time-step for a given controller needs to be for a selection of ERK methods. We also consider model reduction on the controller and how this affects the required step-size and how much it reduces the computational complexity. We have shown that the original 18th order  $\mathcal{H}_\infty$  controller could be reduced to a 10th order controller without any significant reduction in performance or stability, which resulted in a 46.7% reduction in execution time partly because the order reduction enabled us to use a simpler solver type.

---

## 1. INTRODUCTION

Atomic Force Microscopy (AFM) is a tool capable of studying matter down to the atomic scale. This has made it one of the fundamental tools within the field of nanotechnology. Control of the lateral stage of an AFM has been shown to be challenging because of several reasons, including non-linearities such as (1) hysteresis and (2) creep, (3) lightly-damped vibration dynamics, and (4) large uncertainties. For high performance under such conditions, model-based controllers has been widely employed in the literature such as  $\mathcal{H}_\infty$  controllers (Schitter et al., 2001; Salapaka et al., 2002; Salapaka and Sebastian, 2003; Schitter and Stemmer, 2004; Ladjal et al., 2009; Yong et al., 2010). Such controllers tends to become complex in terms of computational complexity. Because of the high uncertainties and the non-linearities it is important to consider robustness in nanopositioning applications. This topic has been studied in Salapaka et al. (2002); Sebastian and Salapaka (2005); Ladjal et al. (2009).

The majority of the literature on nanopositioning seems to perform control design in the continuous time domain as opposed to the discrete-time  $z$ -domain. Such controllers can be described using continuous-time state-space models which we will base our discussion around. For a real-time implementation however, the model is solved at discrete

time-steps using a fixed step-size. Many popular solver types are based on the family of explicit Runge-Kutta (ERK) methods. These solvers become unstable if the step-size is too large. At the same time, the complexity of the controller running on hardware with limited computational power puts a lower limit on the step-size the hardware needs to perform the necessary calculations. Thus we have both a lower and an upper limit on the step-size determined by various factors. For a controller to be implementable we need the limits to intersect. We will discuss some of these factors in this paper and what we can do about them.

To control a system with a mechanically high bandwidth, we need to have high bandwidth on the control loop as well. Thus, the step-size needs to be sufficiently small. On hardware with limited computational power, we often need to simplify the model such that it becomes feasible. The most widely used method to reduce the computational complexity of a controller is to perform model reduction on an already existing controller. Model reduction aims to keep the input-output behavior as close as possible to the original model while removing states from a state-space representation of the controller. Model reduction has been used extensively, some approaches performs reduction on the plant model (Dong et al., 2007; Lee and Salapaka, 2009). Using a model-based approach, this results in a

controller with less complexity. Another approach is to perform reduction after synthesis using a high-order model plant (Schitter and Stemmer, 2004; Kuiper and Schitter, 2012). The discussion of whether to reduce the plant model and then perform model reduction, or perform reduction on the controller is treated in Anderson and Liu (1989); Anderson (1992), where it is generally concluded that reduction should be performed as a last step in the control design process. Even if the system is already implementable, there is an advantage of reducing the complexity, because we can then run the system on a smaller step-size which reduces the overall noise floor of the system.

In this paper, we will base our discussion around a  $\mathcal{H}_\infty$  controller which is designed for the lateral positioning of a commercial AFM. The controller is designed to be robustly stable for a given description of plant uncertainty. We will present equations for how to determine the solver stability of a controller and the required maximum step-size for a variety of ERK methods. Additionally, we will show the effect of model reduction on the complex controller and how this affects the solver stability and computational complexity. This paper is based to a large extent on Ragazzon (2013).

The paper is organized as follows. In Section 2, the experimental set-up is explained and identified model of the plant is found. In section 3, we present the control law design. In Section 4 we present solver stability, specifically for some ERK methods. Section 5 describes the model reduction of the controller. Section 6 gives experimental results of closed-loop characteristics and execution time. The results are discussed in Section 7. Finally, some conclusions are drawn in Section 8.

## 2. SYSTEM IDENTIFICATION

### 2.1 Device Description

All experiments are done using a commercial AFM of the type Park Systems XE-70. In this device, the sample is placed on a parallel kinematic 2d flexure scanner for motion in the horizontal  $xy$ -plane. Motion along the vertical  $z$ -axis is completely decoupled and not regarded for our purposes. The signals from the AFM are routed to an electronic processing and controller box that comes with the microscope. As well as having its own controller circuits, it provides access to analogue measurements from the sensors. It can also receive external signals for manual control of the AFM’s actuators which we will use to control the piezoelectric elements.

See a schematic overview of the setup in Fig. 1. The controllers are implemented in a Simulink model, which is compiled and transferred to a dedicated computer, an xPC, which runs a real-time operating system. The xPC performs the number crunching, and is externally connected to a DAC and ADC. These input-output signals are run through anti-aliasing and reconstruction filters, which are constructed as low-pass filters with a bandwidth of less than half the sample frequency.

For our purpose, we have overridden control of the piezo-actuators in the  $x$ - and  $y$ -axes. This is connected to

a “PiezoDrive PDL200”, a linear voltage amplifier. The input into this amplifier is considered as the input to the system. The voltage output from the distance sensors in the  $x$ - and  $y$ -axes located on the AFM is used as the output of the system.

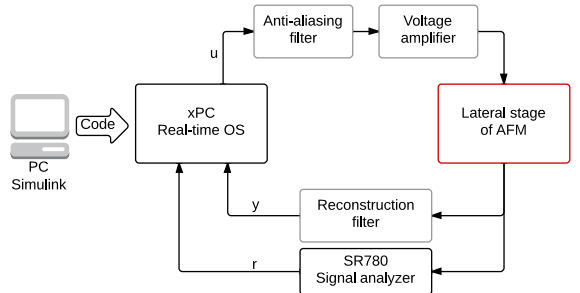


Fig. 1. Block diagram of the experimental setup for the closed-loop system. For the plant frequency response the SR780 device is connected directly to  $u$ .

### 2.2 Frequency Response and Model Fit

The lateral positioning stage of the AFM is considered to be dominantly linear, therefore the system can be described by its frequency response. The system has two inputs  $u_1, u_2$  and two outputs  $y_1, y_2$ , along the  $x$ - and  $y$ -axis respectively. The frequency response of the plant  $G(s)$  was gathered using a Stanford SR780 frequency analyser using a white noise source signal. One of several gathered frequency responses is plotted in Fig. 2 together with the fitted models. The transfer functions were fitted using the Matlab function `tfest` on the experimental data. The diagonal elements of  $G(s)$  were approximated by a third-degree transfer function, while the off-diagonal elements were approximated by a second-degree function. The nominal plant model was found as shown in (1) at the top of the next page.

The exponential term represents the time delay between input and output. A time delay will present itself as a linear reduction of the phase as a function of frequency. Thus, we may find the time delay of the system between input and output by taking a look at the phase plot of the elements of  $\hat{G}$ . By assuming that the change in phase at lower frequencies is dominated by the time delay, and other sources of phase change is close to zero, we can deduce that the time delay is proportional to the slope at the start of the phase plot. This is how we identified the time delay  $T_d = 4.58 \times 10^{-4}$  s.

We can see that the phase starts at  $180^\circ$  which means that the system has an inverse response, i.e. positive inputs give negative outputs and vice versa. This is just the sign convention of our raw data, and we decided not to change it for simplicity.

We can observe that the off-diagonal elements of  $G(s)$  are relatively small compared to the diagonal elements, this indicates that the two axes are physically well decoupled. This indicates that the system is well suited for independent control of the axes where the cross-coupling is not considered. However, we will treat the system as a

$$G(s) = e^{-4.58e-04s} \begin{bmatrix} \frac{-5924s^2 - 1.709e07s - 9.878e10}{s^3 + 4703s^2 + 1.82e07s + 7.806e10} & \frac{-0.04567s^2 + 69.72s - 6.043e04}{s^2 + 104.5s + 2.29e07} \\ \frac{-0.04705s^2 + 89.17s - 1.253e05}{s^2 + 134.1s + 2.288e07} & \frac{-8708s^2 + 2.618e07s - 9.214e11}{s^3 + 3.865e04s^2 + 4.379e07s + 9.44e11} \end{bmatrix} \quad (1)$$

single multiple-input multiple-output (MIMO) plant and design a single more complex controller rather than two independent slightly simpler controllers.

### 2.3 Robust Stability and Uncertainty Weighting

Since the system has large uncertainties and inaccuracies in the model fits, we need to make sure it is robustly stable for a specified set of perturbations of the plant. We chose to model the uncertainties as multiplicative output uncertainty. The perturbed plant is described as

$$G_p = (I + \Delta W)G \quad (2)$$

where  $\Delta$  is the uncertainty variable with  $\|\Delta\|_\infty \leq 1$  and  $W$  is a specified weighting transfer function. The block-diagram of the feedback system is plotted in Fig. 3.

For a given controller  $K$  the robust stability (RS) condition for the described set of perturbations is (Skogestad and Postlethwaite, 2007)

$$\text{RS} \Leftrightarrow \|WT\|_\infty < 1 \quad (3)$$

where  $T \triangleq (I + GK)^{-1}GK$  is the complementary sensitivity function. Similarly we have the sensitivity function  $S \triangleq (I + GK)^{-1}$ .

To find a suitable  $W$  that fits the uncertainties in our system, we can record a set of plant frequency responses  $\hat{G} \in \Pi$ . Then we can guarantee robust stability for at least all of these responses by finding a  $W$  such that  $|W(j\omega)| > \hat{W}(\omega)$  where (Skogestad and Postlethwaite, 2007)

$$\hat{W}(\omega) \triangleq \max_{\hat{G} \in \Pi} \bar{\sigma} \left( \left( \hat{G}(\omega) - G(j\omega) \right) G^{-1}(j\omega) \right) \quad (4)$$

and  $\bar{\sigma}(\cdot)$  is the maximum singular value. Other equations exist for different perturbation descriptions such as input multiplicative uncertainty or additive uncertainty.

We recorded three different frequency responses at different set-points and input amplitudes, and fitted the calculated  $\hat{W}$  by the transfer function

$$W(s) = 3.8254 \frac{(s + 210)(s + 1850)}{(s + 2400)(s + 3200)} \quad (5)$$

which is plotted together with  $\hat{W}$  in Fig. 4.

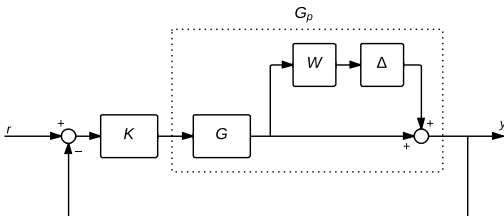


Fig. 3. Feedback system with a multiplicative output uncertainty

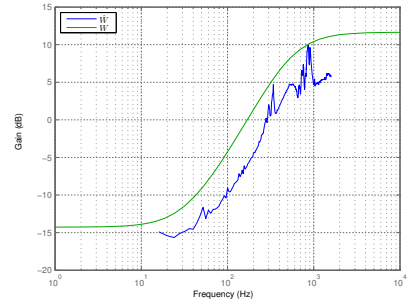


Fig. 4. Robustness fit,  $W(s)$  and  $\hat{W}(s)$

## 3. CONTROL LAW DESIGN

This section will present the design of the  $\mathcal{H}_\infty$  controller. We will explain the choice of weightings for the mixed-sensitivity problem used to synthesize the controller based on the identified model  $G$ .

The  $\mathcal{H}_\infty$  mixed sensitivity problem can be formulated as

$$\min_K N(K) = \left\| \begin{bmatrix} W_S S \\ W_T T \\ W_{KS} K S \end{bmatrix} \right\|_\infty \quad (6)$$

where  $W_S$ ,  $W_T$ , and  $W_{KS}$  are user-defined weightings.

The sensitivity function  $S$  is the closed-loop transfer function from  $r$  to  $e \triangleq y - r$ . Therefore we want this to be as small as possible, especially in the bandwidth we would like to achieve effective control. Thus, within the desired bandwidth we want  $W_S$  to be large, so it was chosen as a first-order filter with large gains at low frequencies and low gains at high frequencies.

The complementary sensitivity function  $T$  is the closed-loop transfer function from  $r$  to  $y$ . Thus, we would like this to be close to one within the desired bandwidth for good tracking behavior. For higher frequencies we would like it to be as small as possible to attenuate measurement noise. Thus  $W_T$  is chosen to be small at low frequencies and large at high frequencies. Since  $WT$  is a measurement of the robust stability we have chosen  $W_T = W$  to form the system to become more easily robustly stable.

Finally, we have the weighting  $W_{KS}$ . In fact  $KS$  is the closed-loop transfer function from  $r$  to  $u$ , so it describes the control effort for a given reference signal. We want to punish high frequencies of  $u$  since this means a large energy usage by the controller, and we know that the system won't respond to very high frequencies. Thus  $W_{KS}$  is modeled as a high-pass filter. The resulting weighting functions are summarized in Table 1.

The problem was solved using the Matlab function `mixsyn`. This resulted in a controller with 18 states. For the sake of comparison later on, we also designed a similar controller using independent axis design, i.e. one  $\mathcal{H}_\infty$  controller for each axis, as well as a simple PID controller. The

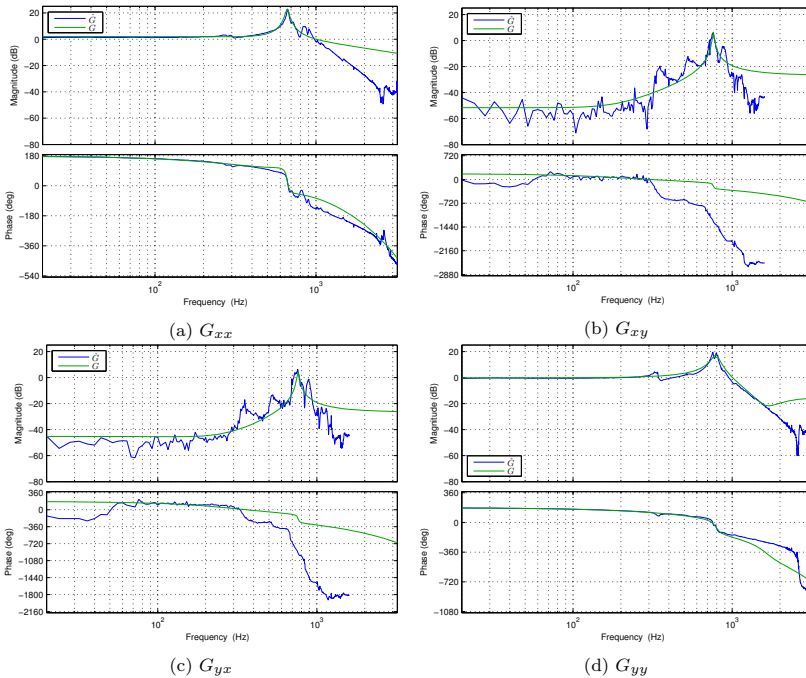


Fig. 2. Experimental frequency response for both axes including cross-terms, and the corresponding fitted models, i.e. the elements of  $G(s)$  and  $\hat{G}(j\omega)$ .

bandwidth of all three controllers as well as the robustness properties and model order is given in Table 2. The table shows us that the closed-loop system is robustly stable because  $\|WT\|_\infty < 1$  for both of the  $\mathcal{H}_\infty$  controllers. This is not the case for the PID-controller, so in fact we can not guarantee that this controller is stable for all perturbations of the system.

Table 1. Summary of weighting transfer functions

$W_S(s)$	$\frac{0.8333s + 439.8}{s + 0.04398}$
$W_T(s)$	$3.8254 \frac{(s + 210)(s + 1850)}{(s + 2400)(s + 3200)}$
$W_{KS}(s)$	$0.8333 \frac{s}{s + 439.8}$

Table 2. Bandwidth and robustness comparison between the three controllers. TF = transfer function, SS = state space.

	$\omega_{b,S}$ [Hz]	$\omega_{b,T}$ [Hz]	$\ WT\ _\infty$	Implementation
PID	58.0	93.1	1.073	4th order TF
$\mathcal{H}_\infty$ SISO	75.6	96.4	0.9938	14th order SS
$\mathcal{H}_\infty$ MIMO	69.8	98.6	0.6717	18th order SS

#### 4. SOLVER STABILITY

We will consider the case where a controller is represented by a continuous-time state-space model. A real-time implementation of such a model will use a solver to perform

the necessary integration steps at fixed discrete time intervals, denoted by the step-size  $h$ . A complex controller will require a larger step-size because of limited computational power in the hardware, while at the same time an increased step-size can make the solver unstable. In this section we will see that the solver stability depends on the eigenvalues of the controller, the step-size, as well as the solver type.

Let us consider the scalar test system

$$\dot{y} = \lambda y \quad (7)$$

which is applied to a solver taking the discrete state  $y_n$  to the next time step  $y_{n+1}$  with step-size  $h$ ,

$$y_{n+1} = \Phi(h\lambda)y_n \quad (8)$$

$$= [\Phi(h\lambda)]^n y_0 \quad (9)$$

where  $\Phi(h\lambda)$  is called the stability function. It is evident that (9) is stable, i.e.  $|y_n| \leq c < \infty \forall n \geq 0$ , if and only if

$$|\Phi(h\lambda)| \leq 1 \quad (10)$$

All solvers we will consider have such a stability function, and the region of stability, i.e. the region of the complex plane where (10) is satisfied, varies between each solver type.

*Example 1:* Euler's Method applied to (7) gives

$$\begin{aligned} y_{n+1} &= y_n + h(\lambda y_n) \\ &= (1 + h\lambda) y_n \end{aligned} \quad (11)$$

thus  $\Phi(h\lambda) = 1 + h\lambda$ , which is stable in the region  $\{z \in \mathbb{C} \mid |1 + z| < 1\}$ , in other words the unit circle with center -1.  $\triangle$

### 4.1 Runge-Kutta Methods

The family of explicit Runge-Kutta (ERK) methods can be written as

$$y_{n+1} = y_n + \sum_{i=1}^s b_i k_i \quad (12)$$

where  $s$  describes the number of stages of the Runge-Kutta method and

$$\begin{aligned} k_1 &= hf(t_n, y_n) \\ k_2 &= hf(t_n + c_2 h, y_n + a_{21} k_1) \\ k_3 &= hf(t_n + c_3 h, y_n + a_{31} k_1 + a_{32} k_2) \\ &\vdots \end{aligned}$$

$k_s = hf(t_n + c_s h, y_n + a_{s1} k_1 + a_{s2} k_2 + \dots + a_{s,s-1} k_{s-1})$  where the coefficients  $a_{ij}$ ,  $b_i$ , and  $c_i$  are elements of  $A$ ,  $b$ , and  $c$  respectively, which are specified by a given ERK solver. Note that Euler's method is a first order ERK method.

### 4.2 Stability of Explicit Runge-Kutta Methods

The stability function of ERK methods are given by (Egeland and Gravdahl, 2002)

$$\Phi(z) = \det(I - zA + z\mathbf{1}b^T) \quad (13)$$

where  $\mathbf{1}$  is a column vector of one-elements. It can be shown that this can be simplified for an ERK method of order  $p = s$  to (Hairer and Wanner, 1996)

$$\Phi(z) = 1 + z + \dots + \frac{z^p}{p!}$$

which is only possible for methods of order up to 4. Higher order ERK methods need more stages  $s$  than the order  $p$ . We have plotted the stability region of a selection of ERK methods in Fig. 5. Specifically the region of erk1-erk4 with  $p = s$ , Dormand-Prince 5 (erk5) and Dormand-Prince 8 (erk8). The last two with coefficients taken from Dormand and Prince (1980); Prince and Dormand (1981). The stability functions of these methods are reckoned to be the same as for the fixed-step solver methods available in Simulink.

### 4.3 Linear System

We have given the stability methods for several ERK methods for the scalar test system. In this section, we will show that the stability of the solvers applied to a linear system of ordinary differential equations (ODE) is given by its eigenvalues.

*Theorem 1.* Consider the system

$$\dot{y} = Ay \quad (14)$$

where  $A$  is an  $n \times n$  diagonalizable matrix having eigenvalues  $\lambda_1, \dots, \lambda_n$ . Let us apply an explicit Runge-Kutta method to this system. Then the ERK method has a stable point at the origin if and only if the same method has a stable point for

$$\dot{z} = \lambda_i z \quad \forall i \in [1, \dots, n]$$

△

This is a standard result in numerical methods theory, see e.g. Ascher and Petzold (1998). We have used a wording similar to Frank (2008) which is also used for the next corollary.

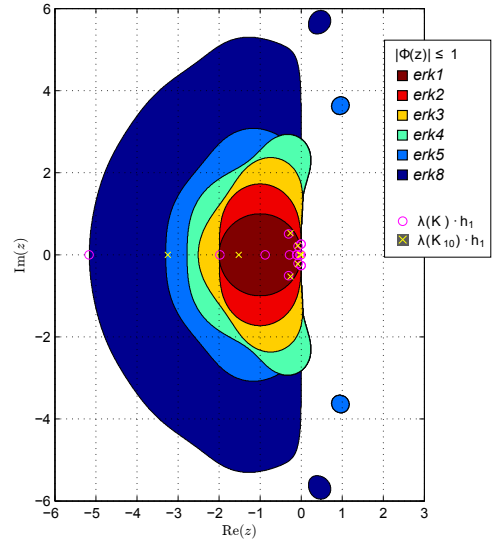


Fig. 5. Stability region of a variety of ERK methods of order 1-5 and 8. *Magenta circles*: Eigenvalues of the full order controller  $K$  scaled by the maximum step-size achieving stability for erk8. *Yellow X's*: Eigenvalues of the reduced tenth order controller  $K_{10}$  scaled by the same step-size. Note that  $K_{10}$  is stable for erk5.

*Corollary 2.* Consider a Runge-Kutta method with stability function  $\Phi(z)$  applied to the system  $\dot{y} = Ay$ . Then the origin is stable for the numerical method with step-size  $h$  if and only if

$$|\Phi(h\lambda_i)| \leq 1, \quad \forall i \in [1, \dots, n]$$

where  $\lambda_i$  are the eigenvalues of  $A$ . △

In other words, if all the eigenvalues of  $A$  are within the region of stability for a given solver at a specific step-size  $h$ , then the solver applied to (14) is stable. This gives us a tool to check for the required maximum step-size for a given controller and solver.

## 5. CONTROL ORDER REDUCTION

### 5.1 Model Reduction Theory

There exist several methods to perform model reduction Obinata and Anderson (2001), most widely used is possibly balanced residualization. Here the controller is first transformed to a balanced realization. A realization is said to be balanced if the controllability and observability *Gramians* are equal, thus a balanced model can be said to be as observable as it is controllable. The model states are ordered by decreasing Hankel singular values to form a state-space model  $(A, B, C, D)$ . The last states are removed and the system is transformed such that

$$\left[ \begin{array}{cc|c} A_{11} & A_{12} & B_1 \\ A_{21} & A_{22} & B_2 \\ \hline C_1 & C_2 & D \end{array} \right] \Rightarrow \left[ \begin{array}{cc|c} A_{11} - A_{12}A_{22}^{-1}A_{21} & B_1 - A_{21}A_{22}^{-1}B_2 \\ \hline C_1 - C_2A_{22}^{-1}A_{21} & D - C_2A_{22}^{-1}B_2 \end{array} \right]$$

The DC-gain of the system is maintained using this method, at some cost to the accuracy in the faster modes.

If we instead of a balanced realization used a canonical modal realization where the  $A$ -matrix is diagonal with elements equal to the eigenvalues, and performed the same reduction technique we can essentially remove the system eigenvalues of choice. This may be useful if some of the eigenvalues are larger than wanted, but can also result in large errors and possibly instability. Other methods include the truncation method which is more accurate at high frequencies at the cost of low frequencies, the optimal Hankel norm method, and using Linear Matrix Inequalities.

### 5.2 Results of Control Reduction

The designed controller  $K$  was first transformed to a balanced realization using the Matlab command `balreal`. The Hankel singular values for  $K$  are given in Table 3 which gives an idea of the error to be expected from removing each state.

We performed model reduction on  $K$  by model residualization to several new controllers  $K_r$  of order 2 to order 17. This was done in Matlab with the command `modred`. The closed-loop  $\mathcal{H}_\infty$  error norm on  $T - T_r$ , where  $T_r$  is the complementary sensitivity of the reduced controller, for each reduced controller is shown in Fig. 6a. We can see that there are significant drops specifically between order 4-5, 9-10, and 16-17. The error changes relatively little in-between these drops. Since we would like a controller with as low order as possible while maintaining the performance characteristics, we are inclined to select one of the orders after such a drop, i.e. 5, 10, or 17. The robustness norm is shown in Fig. 6b where we can see that only controller order 10 and higher are robustly stable with  $\|WT_r\|_\infty < 1$ . The previous discussion clearly favors choosing the 10th order controller as it provides robust stability with little error. This choice is further reinforced by considering the simulated step responses as shown in Fig. 7. The 10th order controller gives nearly indistinguishable results to the original controller, while the 8th and 9th order controllers shows some oscillatory behavior. The 7th order model is unstable, so we clearly want to avoid it.

Table 3. Hankel singular values of the balanced realization of the controller,  $\sigma_i$

1)	5.436e+03	7)	6.799e-02	13)	1.324e-02
2)	4.227e+03	8)	6.049e-02	14)	4.091e-03
3)	3.307e-01	9)	3.885e-02	15)	4.073e-03
4)	2.775e-01	10)	1.977e-02	16)	1.400e-03
5)	1.246e-01	11)	1.514e-02	17)	1.044e-03
6)	7.628e-02	12)	1.430e-02	18)	5.722e-05

### 5.3 Eigenvalues and Maximum Step-Size

We have previously shown that the stability of an explicit Runge-Kutta method applied to a state-space model depends on the eigenvalues of the  $A$ -matrix and the step-size. Thus, for a given controller we can find the maximum step-size needed for stability. The maximum step-size for controller  $K$  was found to be  $54.62 \mu\text{s}$  using the `erk8` solver. The eigenvalues scaled by step-size can be seen in Fig. 5 (magenta circles) which are seen to lie within the stability region of `erk8`. It is not stable for `erk5` as the eigenvalues

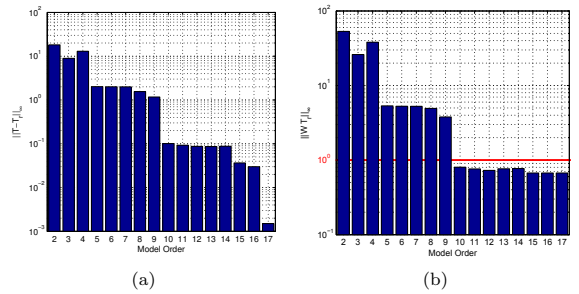


Fig. 6. Reduced controller order properties. (a) Closed-loop error  $\|T - T_r\|_\infty$ . (b) Robustness  $\|WT_r\|_\infty$ , must be  $< 1$  for robust stability (marked red).

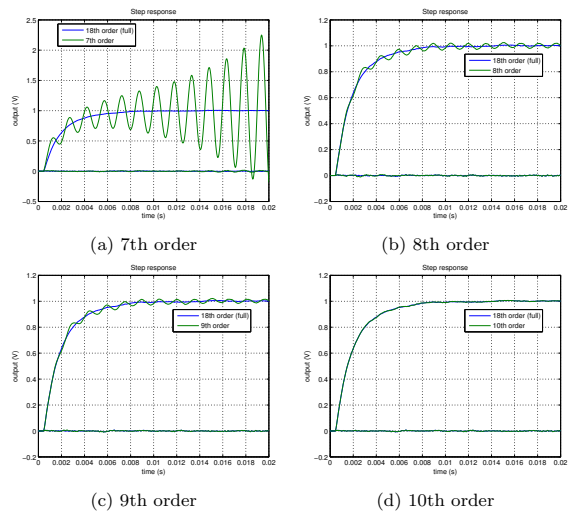


Fig. 7. Simulated step-response in reference signal on the  $x$ -axis, reduced vs original controller. Shows output from both  $x$ -axis and  $y$ -axis.

are outside the stability region for this solver. The eigenvalues of the reduced controller  $K_{10}$  has also been plotted (yellow x's) for the same step-size and we can see how the controller reduction has affected the eigenvalues. We can see that they have become smaller which has resulted in the controller becoming stable even for the `erk5` solver. So not only does model reduction reduce the computational complexity of the controller, but it can also enable us to use a simpler solver type or alternatively a larger step-size.

The maximum step-size for a variety of controllers and solver types are shown in Table 4. Note that the maximum step-size does not strictly increase with lower model-orders, since the model residualization method used does not necessarily reduce the eigenvalues. Other reduction methods could be considered if this is critical, such as methods directly removing states with large eigenvalues.

## 6. EXPERIMENTAL RESULTS

Experiments were performed for two reasons. The first was to see how well the reduced order controllers performed

Table 4. Maximum step-size for a given explicit Runge-Kutta (ERK) method for the various reduced controllers as well as some simpler controllers and the nominal plant model. Larger values are generally better because they are stable at higher step-sizes.

Order	$h_{max}$ [ $\mu$ s]					
	erk1	erk2	erk3	erk4	erk5	erk8
7 (unst.)	32.18	71.33	95.74	100.8	127.2	203.6
8	58.37	58.37	73.34	81.29	96.51	150.8
9	37.54	37.54	47.17	52.28	62.07	96.99
10	33.61	33.61	42.22	46.8	55.56	86.82
11	39.70	39.7	49.87	55.28	65.63	102.5
12	24.96	32.66	41.03	45.48	54.00	84.37
13	2.821	21.55	27.07	30.01	35.63	55.67
14	2.818	19.11	24.01	26.62	31.60	49.38
15	2.843	21.73	27.30	30.26	35.92	56.13
16	2.948	22.64	28.44	31.53	37.43	58.48
17	2.918	21.18	26.61	29.49	35.01	54.71
18 (full)	2.910	21.14	26.56	29.45	34.96	54.62
PID	80.00	80.00	100.5	111.4	132.3	206.7
$\mathcal{H}_{\infty SISO}$	21.45	21.45	26.95	29.87	35.46	55.41
$G(s)$	4.564	52.42	65.85	73.00	86.66	135.4

compared to our simulations. The second was to record the *average task execution time* (TET), the time it takes the hardware to perform calculations from one time-step to the next. This can be considered a measurement of the computational complexity of the controller, and is a lower limit on the step-size. Any lower than this and the hardware will not be able to meet its deadline and exit with a ‘‘CPU overload’’ error.

The experiments were performed in the setup shown in Fig. 1. The closed loop frequency response of the original controller  $K$  compared to the reduced controllers  $K_{10}$  and  $K_8$  is given in Fig. 8. The average TET for the various controllers and solver types are given in Table 5. The system was run with step-size  $h = 40 \mu$ s, and only the modes that are stable at this step-size as can be seen from Table 4 was tested, i.e.  $h_{max} \geq 40 \mu$ s.

## 7. DISCUSSION

### 7.1 Model Reduction and Performance

The model reduction showed us that the original 18th order controller could be reduced to a 10th order model with no noticeable difference in the simulated step-response or the experimental closed-loop frequency response. Additionally, it was shown to maintain robust stability, thus it is a very viable controller choice. In terms of the impact on computational complexity, we can see that we have reduction of 25.5%, from 20.11 to 14.99  $\mu$ s if we use the erk8 solver for both controllers.

From Table 4 we can see that the 10th order controller can run using the erk3 solver at a step-size of  $h = 40 \mu$ s, while the full  $\mathcal{H}_{\infty}$  MIMO controller needs erk8 for stability at this step-size. By choosing erk3 for the 10th order controller we can see from Table 5 that this reduces the execution time to 10.71  $\mu$ s, or a 46.7% reduction from the original controller.

Table 5. Average Task Execution Time (TET) with different controller model order and solver types which gives an indication on the computational complexity. Step-size  $h = 40 \mu$ s. Dash (-) unstable, not tested. (x) CPU overload.

Order	Average TET [ $\mu$ s]					
	erk1	erk2	erk3	erk5	erk8	ode14x
$\leq 7$	-	-	-	-	-	-
8	-	10.25	10.43	11.10	13.64	26.72
9	-	-	10.55	11.43	14.26	x
10	-	-	10.71	11.64	14.99	x
11	-	-	10.89	11.91	15.79	x
12	-	-	11.07	12.36	16.79	x
13	-	-	-	-	17.68	x
14	-	-	-	-	18.82	x
15	-	-	-	-	19.61	x
16	-	-	-	-	20.91	x
17	-	-	-	-	22.61	x
18	-	-	-	-	20.11	x
PID	-	9.95	9.98	10.13	11.12	14.91
$\mathcal{H}_{\infty SISO}$	-	-	-	-	14.58	x

We tried to run the controller with the implicit solver *ode14x* (*Extrapolation*) supplied with Simulink, but it was found to be such computationally demanding that we were only able to run it with the 8th order controller in addition to the PID controller. Additionally, the solutions was found to explode at times in our simulations so this solver was not further considered.

It is also interesting to note that the execution time does not strictly decrease with increased controller order, e.g. the 17th to 18th order controller. By inspecting the state-space model of each of these controllers, we notice that the 18th order controller has a lot more zero-valued elements. We speculate that the compiler simplifies the arithmetic on these elements.

### 7.2 Eigenvalues and Stability

As we have seen, the eigenvalues of the controller are one of the decisive factors for stability of an applied ERK method. Hence, it is important to consider how controller reduction changes the eigenvalues, especially for a fast and stiff system such as our lateral positioning platform of an AFM. The eigenvalues of our controller tended to become smaller with reduced orders, but this need not be the case. One should be careful when performing model reduction and possibly verify that the eigenvalues are within the stable region of the solver considered. If the solver stability becomes a problem, one should consider a different model reduction method, such as removing the eigenvalues directly from a canonical modal realization of the controller.

We have also seen how increased solver order increases the stability region, but at the same time it increases the execution time. This will ultimately be a trade-off between moving the lower limit (due to computation time) and the upper limit (for stability) of the step-size, as illustrated in Fig. 9. Halving the step-size usually doubles the computational complexity (per unit of time), while



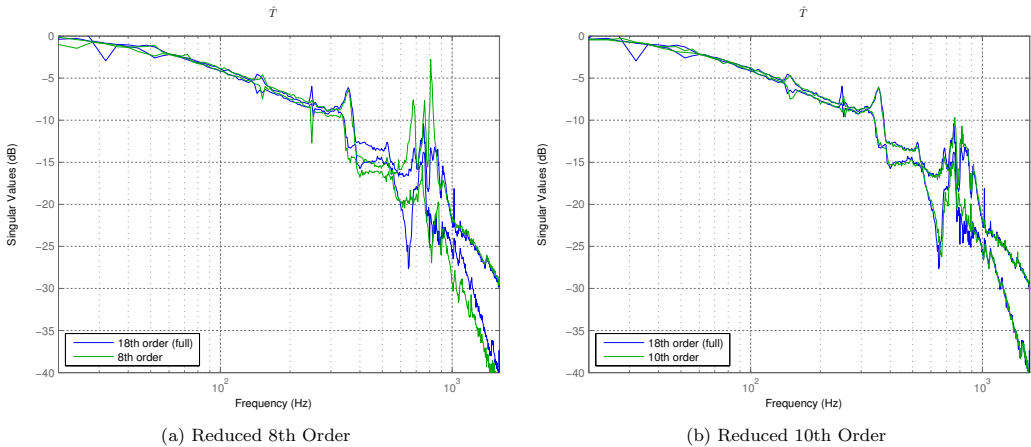


Fig. 8. Singular values of the closed-loop  $\sigma(\hat{T})$  for the 8th and 10th order reduced controller versus the original controller.

increasing the solver complexity is harder to predict, but will generally depend on the controller order.

Note that we have not considered the accuracy of the solver methods. This is because we have assumed that the system is stiff, and stiff systems are characterized such that instability comes before any accuracy problems.

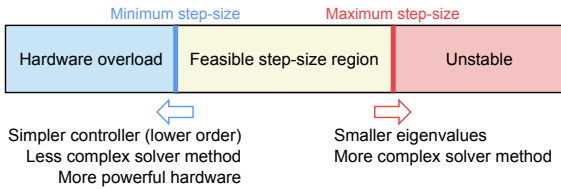


Fig. 9. Illustration of the trade-off between step-size, controller complexity, solver complexity, eigenvalues of the controller, and hardware performance for a real-time controller implementation.

### 8. CONCLUSION

This paper concerns itself with some practical issues for a controller running in real-time. Since the capability of hardware is limited in terms of computational performance, a complex controller can become difficult to implement with a step-size small enough for stability. This paper tries to achieve two goals, (1) to show how to determine the maximum step-size a controller requires for numerical stability, and (2) to show what can be done to reduce the computational complexity of an already existing controller with a focus on model reduction.

To show how this can be done, a robustly stable  $\mathcal{H}_\infty$  controller was designed for a nanopositioning device. We showed that the numerical stability of an explicit Runge-Kutta method is determined by the eigenvalues of the controller. Model reduction was performed on the controller and the reduced controllers were compared in terms of performance and stability both in simulations and experiments which showed that the 18th order controller could be reduced to a 10th order model without any significant

reduction in performance or stability. After the reduction process, the largest eigenvalues were reduced in size so the system also became numerically stable for even simpler solver types which allow further reduction of the computational requirement.

An experiment was run to determine the change in computational complexity by recording the execution time of the various controllers. The reduction to a 10th order model, as well as the possibility of using a simpler solver type resulted in a 46.7% reduction in task execution time.

### REFERENCES

Anderson, B.D.O. (1992). Controller Design : Moving from Theory to Practice. *IEEE Control Systems Magazine*, 16–25.

Anderson, B.D.O. and Liu, Y. (1989). Controller reduction: concepts and approaches. *Automatic Control, IEEE Transactions on*, 34(8).

Ascher, U. and Petzold, L. (1998). *Computer methods for ordinary differential equations and differential-algebraic equations*.

Dong, J., Salapaka, S.M., and Ferreira, P.M. (2007). Robust MIMO control of a parallel kinematics nanopositioner for high resolution high bandwidth tracking and repetitive tasks. *Decision and Control, 2007. 46th IEEE Conference on*, 4495–4500.

Dormand, J. and Prince, P. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1), 19–26.

Egeland, O. and Gravdahl, J.T. (2002). *Modeling and Simulation for Automatic Control*.

Frank, J. (2008). Stability of Runge-Kutta Methods (Lecture notes for course "Numerical Modelling of Dynamical Systems").

Hairer, E. and Wanner, G. (1996). *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*.

Kuiper, S. and Schitter, G. (2012). Model-based feedback controller design for dual actuated atomic force microscopy. *Mechatronics*, 22(3), 327–337.

- Ladjal, H., Hanus, J.L., and Ferreira, A. (2009). H<sub>inf</sub> robustification control of existing piezoelectric-stack actuated nanomanipulators. *2009 IEEE International Conference on Robotics and Automation*, 3353–3358.
- Lee, C. and Salapaka, S.M. (2009). Fast Robust Nanopositioning: A Linear-Matrix-Inequalities-Based Optimal Control Approach. *Mechatronics, IEEE/ASME Transactions on*, 14(4), 414–422.
- Obinata, G. and Anderson, B.D.O. (2001). *Model reduction for control system design*. Springer-Verlag New York, Inc.
- Prince, P. and Dormand, J. (1981). High order embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*.
- Ragazzon, M.R.P. (2013). *Nanopositioning in Atomic Force Microscopes: Robust Control Design, Order Reduction and Numerical Implementability*. Master’s thesis, Norwegian University of Science and Technology.
- Salapaka, S., Sebastian, a., Cleveland, J.P., and Salapaka, M.V. (2002). High bandwidth nano-positioner: A robust control approach. *Review of Scientific Instruments*, 73(9), 3232.
- Salapaka, S. and Sebastian, A. (2003). Control of the nanopositioning devices. *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, 3(December), 3644–3649.
- Schitter, G., Menold, P., Knapp, H.F., Allgöwer, F., and Stemmer, A. (2001). High performance feedback for fast scanning atomic force microscopes. *Review of Scientific Instruments*, 72(8), 3320.
- Schitter, G. and Stemmer, a. (2004). Identification and Open-Loop Tracking Control of a Piezoelectric Tube Scanner for High-Speed Scanning-Probe Microscopy. *IEEE Transactions on Control Systems Technology*, 12(3), 449–454.
- Sebastian, A. and Salapaka, S.M. (2005). Design methodologies for robust nano-positioning. *IEEE Transactions on Control Systems Technology*, 13(6), 868–876.
- Skogestad, S. and Postlethwaite, I. (2007). *Multivariable feedback control: analysis and design*. Wiley-Interscience, 2 edition edition.
- Yong, Y., Liu, K., and Moheimani, S. (2010). Reducing cross-coupling in a compliant XY nanopositioner for fast and accurate raster scanning. *Control Systems Technology, IEEE Transactions on*, 18(5), 1172–1179.

# Bibliography

- [1] D. Y. Abramovitch, S. B. Andersson, L. Y. Pao, and G. Schitter. A Tutorial on the Mechanisms, Dynamics, and Control of Atomic Force Microscopes. In *2007 American Control Conference*, pages 3488–3502. IEEE, 2007. ISBN 1-4244-0988-8.
- [2] B. D. O. Anderson. Controller Design : Moving from Theory to Practice. *IEEE Control Systems Magazine*, pages 16–25, 1992.
- [3] U. Aridogan, Y. Shan, and K. K. Leang. Design and Analysis of Discrete-Time Repetitive Control for Scanning Probe Microscopes. *Journal of Dynamic Systems, Measurement, and Control*, 131(6):061103, 2009.
- [4] U. Ascher and L. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. 1998.
- [5] S. Bashash and N. Jalili. Robust Adaptive Control of Coupled Parallel Piezo-Flexural Nanopositioning Stages. *Mechatronics, IEEE/ASME Transactions on*, 14(1):11–20, 2009.
- [6] B. Bhushan and O. Marti. Scanning Probe Microscopy - Principle of Operation, Instrumentation, and Probes. In B. Bhushan, editor, *Springer Handbook of Nanotechnology*, pages 573–617. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-02524-2.
- [7] G. Binnig, H. Rohrer, C. Gerber, and E. Weibel. Surface Studies by Scanning Tunneling Microscopy. *Phys. Rev. Lett.*, 49(1):57–61, 1982.
- [8] G. Binnig, C. F. Quate, and C. Gerber. Atomic Force Microscope. *Phys. Rev. Lett.*, 56(9):930–933, 1986.
- [9] D. A. Bristow, M. Tharayil, and A. G. Alleyne. A survey of iterative learning control. (June):96–114, 2006.
- [10] D. a. Bristow, J. Dong, A. G. Alleyne, P. Ferreira, and S. Salapaka. High bandwidth control of precision motion instrumentation. *The Review of scientific instruments*, 79(10):103704, 2008.
- [11] J. Butcher. Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics*, 125(December 2000):147–158, 2000.

- 
- [12] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2003. ISBN 9780471967583.
- [13] G. M. Clayton, S. Tien, K. K. Leang, Q. Zou, and S. Devasia. A Review of Feed-forward Control Approaches in Nanopositioning for High-Speed SPM. *Journal of Dynamic Systems, Measurement, and Control*, 131(6):061101, 2009.
- [14] D. Croft and S. Devasia. Vibration compensation for high speed scanning tunneling microscopy. *Review of Scientific Instruments*, 70(12):4600, 1999.
- [15] D. Croft, S. Stilson, and S. Devasia. Optimal tracking of piezo-based nanopositioners. *Nanotechnology*, 10(2):201–208, 1999.
- [16] D. Croft, G. Shedd, and S. Devasia. Creep, hysteresis, and vibration compensation for piezoactuators: atomic force microscopy application. *American Control Conference, 2000. Proceedings of the 2000*, 3(June):2123–2128, 2000.
- [17] S. Devasia, E. Eleftheriou, and S. O. R. Moheimani. A Survey of Control Issues in Nanopositioning. *IEEE Transactions on Control Systems Technology*, 15(5):802–823, 2007.
- [18] J. Dong, S. M. Salapaka, and P. M. Ferreira. Robust MIMO control of a parallel kinematics nano-positioner for high resolution high bandwidth tracking and repetitive tasks. *Decision and Control, 2007. 46th IEEE Conference on*, pages 4495–4500, 2007.
- [19] J. Dormand and P. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.
- [20] J. Doyle, K. Glover, P. Khargonekar, and B. Francis. State-space solutions to standard  $H_2$  and  $H_\infty$  control problems. *IEEE Transactions on Automatic Control*, 34(8):831–847, 1989.
- [21] J. Doyle, B. Francis, and A. Tannenbaum. *Feedback control theory*. 1992.
- [22] O. Egeland and J. T. Gravdahl. *Modeling and Simulation for Automatic Control*. 2002.
- [23] A. A. Eielsen. *Topics in Control of Nanopositioning Devices*. PhD thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics, 2012.
- [24] A. A. Eielsen and J. T. Gravdahl. Adaptive control of a nanopositioning device. *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 5065–5072, 2012.
- [25] A. A. Eielsen, M. Burger, J. T. Gravdahl, and K. Y. Pettersen. PI2-Controller Applied to a Piezoelectric Nanopositioner Using Conditional Integrators and Optimal Tuning. *18th IFAC World Congress, Proceedings of the*, pages 1–6, 2011.
- [26] D. Enns. Model reduction with balanced realizations: An error bound and a frequency weighted generalization. *Decision and Control, 1984. The 23rd IEEE Conference on*, (December):127–132, 1984.

- [27] A. Fleming and A. Wills. Optimal Periodic Trajectories for Band-Limited Systems. *IEEE Transactions on Control Systems Technology*, 17(3):552–562, 2009.
- [28] A. J. Fleming, S. S. Aphale, and S. O. R. Moheimani. A New Method for Robust Damping and Tracking Control of Scanning Probe Microscope Positioning Stages. *IEEE Transactions on Nanotechnology*, 9(4):438–448, 2009.
- [29] J. Frank. Stability of Runge-Kutta Methods (Lecture notes for course "Numerical Modelling of Dynamical Systems"), 2008. URL <http://homepages.cwi.nl/~jason/Courses/numwisk/ch10.pdf>. Online; accessed June 7, 2013.
- [30] R. Garcia and R. Perez. Dynamic atomic force microscopy methods. *Surface science reports*, 47, 2002.
- [31] F. Giessibl. Advances in atomic force microscopy. *Reviews of modern physics*, 2003.
- [32] K. Glover and D. McFarlane. Robust stabilization of normalized coprime factor plant descriptions with  $H_\infty$ -bounded uncertainty. *Automatic Control, IEEE Transactions on*, 34(8):821–830, 1989.
- [33] B. E. Helfrich, C. Lee, D. A. Bristow, X. Xiao, J. Dong, A. Alleyne, S. M. Salapaka, and P. M. Ferreira. Combined  $H_\infty$ -feedback and iterative learning control design with application to nanopositioning systems. *IEEE Transactions on Control Systems Technology*, 18(2):336–351, 2010.
- [34] R. Hillenbrand, M. Stark, and R. Guckenberger. Higher-harmonics generation in tapping-mode atomic-force microscopy: Insights into the tip-sample interaction. *Applied Physics Letters*, 76(23):3478, 2000.
- [35] S. Kuiper and G. Schitter. Model-based feedback controller design for dual actuated atomic force microscopy. *Mechatronics*, 22(3):327–337, 2012.
- [36] H. Ladjal, J.-L. Hanus, and A. Ferreira.  $H_\infty$  robustification control of existing piezoelectric-stack actuated nanomanipulators. *2009 IEEE International Conference on Robotics and Automation*, pages 3353–3358, 2009.
- [37] K. K. Leang and S. Devasia. Design of hysteresis-compensating iterative learning control for piezo-positioners: Application to atomic force microscopes. *Mechatronics*, 16(3-4):141–158, 2006.
- [38] C. Lee and S. M. Salapaka. Fast Robust Nanopositioning: A Linear-Matrix-Inequalities-Based Optimal Control Approach. *Mechatronics, IEEE/ASME Transactions on*, 14(4):414–422, 2009.
- [39] Y. Li and J. Bechhoefer. Feedforward control of a closed-loop piezoelectric translation stage for atomic force microscope. *The Review of scientific instruments*, 78(1):013702, 2007.
- [40] A. McCormack and K. Godfrey. Rule-based autotuning based on frequency domain identification. *IEEE Transactions on Control Systems Technology*, 6(1): 43–61, 1998.

- 
- [41] S. O. R. Moheimani. A Survey of Recent Innovations in Vibration Damping and Control Using Shunted Piezoelectric Transducers. *11(4):482–494*, 2003.
- [42] B. Mokaberi and A. Requicha. Compensation of scanner creep and hysteresis for AFM nanomanipulation. *Automation Science and Engineering, IEEE Transactions on*, 5, 2008.
- [43] H. Numasato and M. Tomizuka. Settling control and performance of a dual-actuator system for hard disk drives. *IEEE/ASME Transactions on Mechatronics*, 8(4):431–438, 2003.
- [44] Y. Okazaki. A micro-positioning tool post using a piezoelectric actuator for diamond turning machines. *Precision Engineering*, 12(3):151–156, 1990.
- [45] H. R. Pota, S. O. R. Moheimani, and M. Smith. Resonant controllers for smart structures. *Smart Materials and Structures*, 11(1):1–8, 2002.
- [46] P. Prince and J. Dormand. High order embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 1981.
- [47] O. Sahin, C. Quate, O. Solgaard, and F. Giessibl. Higher Harmonics and Time-Varying Forces in Dynamic Force Microscopy. In B. Bhushan, editor, *Springer Handbook of Nanotechnology*, pages 711–729. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-02524-2.
- [48] S. Salapaka and A. Sebastian. Control of the nanopositioning devices. *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, 3(December):3644–3649, 2003.
- [49] S. Salapaka, a. Sebastian, J. P. Cleveland, and M. V. Salapaka. High bandwidth nano-positioner: A robust control approach. *Review of Scientific Instruments*, 73(9):3232, 2002.
- [50] A. Schirmeisen, B. Anczykowski, H. Hölscher, and H. Fuchs. Dynamic Modes of Atomic Force Microscopy. In B. Bhushan, editor, *Springer Handbook of Nanotechnology*, pages 731–761. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-02524-2.
- [51] G. Schitter and a. Stemmer. Identification and Open-Loop Tracking Control of a Piezoelectric Tube Scanner for High-Speed Scanning-Probe Microscopy. *IEEE Transactions on Control Systems Technology*, 12(3):449–454, 2004.
- [52] G. Schitter, P. Menold, H. F. Knapp, F. Allgöwer, and A. Stemmer. High performance feedback for fast scanning atomic force microscopes. *Review of Scientific Instruments*, 72(8):3320, 2001.
- [53] G. Schitter, P. J. Thurner, and P. K. Hansma. Design and input-shaping control of a novel scanner for high-speed atomic force microscopy. *Mechatronics*, 18(5-6):282–288, 2008.
- [54] A. Sebastian and S. M. Salapaka. Design methodologies for robust nano-positioning. *IEEE Transactions on Control Systems Technology*, 13(6):868–876, 2005.

- [55] S. Skogestad and I. Postlethwaite. *Multivariable feedback control: analysis and design*. Wiley-Interscience, 2 edition edition, 2007. ISBN 978-0470011683.
- [56] The MathWorks. Matlab documentation: mixsyn. URL <http://www.mathworks.se/help/robust/ref/mixsyn.html>. Online; accessed 11 May 2013.
- [57] The MathWorks. xPC Target For Use with Real-Time Workshop - User's Guide (Version 2), 2003.
- [58] Y. Yong, K. Liu, and S. Moheimani. Reducing cross-coupling in a compliant XY nanopositioner for fast and accurate raster scanning. *Control Systems Technology, IEEE Transactions on*, 18(5):1172–1179, 2010.
- [59] K. Zhou, J. Doyle, and K. Glover. *Robust and optimal control*. Prentice Hall Upper Saddle River, NJ, 1996.
- [60] Q. Zou and S. Devasia. Preview-Based Optimal Inversion for Output Tracking: Application to Scanning Tunneling Microscopy. *IEEE Transactions on Control Systems Technology*, 12(3):375–386, 2004.

