Iterative Methods for High Precision Motion Control with Application to a Wafer Scanner System

Author
Stearns, Hoday Margaret

Publication Date
2011

Peer reviewed|Thesis/dissertation
Iterative Methods for High Precision Motion Control with Application to a Wafer Scanner System

by

Hoday Margaret Stearns

A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Engineering - Mechanical Engineering in the Graduate Division of the University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Roberto Horowitz
Professor Ronald Fearing

Spring 2011
Iterative Methods for High Precision Motion Control with Application to a Wafer Scanner System

Copyright 2011
by
Hoday Margaret Stearns
Abstract

Iterative Methods for High Precision Motion Control with Application to a Wafer Scanner System

by

Hoday Margaret Stearns

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Advances in photolithography are one of the key driving factors in the continuing expansion in capacity and decrease in cost of semiconductors. Extending this trend into the future necessitates the development of next-generation lithography technologies in order to overcome the fundamental challenges of improving critical dimension and overlay control, and lowering the total cost-of-ownership. As feature sizes become smaller and smaller, performance requirements for wafer scanner machines will become more stringent; with regards to motion control, requirements for the wafer stage include sub-nanometer positioning precision under high scan velocities and accelerations. Advanced control algorithms are needed to meet these requirements in the face of disturbances such as vibrations, noise, force ripple and friction, as well as model uncertainty.

This dissertation focuses on using the repetitiveness of the stage’s motion in the photolithography process to improve control precision. Similar to many manufacturing processes, the step-and-scan motion used to expose a wafer is very repetitive, on a die-to-die and also wafer-to-wafer level. By using data gathered from past runs, the control effort for future runs may be improved, thereby exploiting the repetitiveness of the process to increase control precision.

In this research, iterative learning control (ILC) and iterative feedback tuning (IFT) were applied to reduce tracking error of the wafer stage. In ILC, a feedforward control signal for the system is incrementally adjusted to achieve better tracking performance using error signals from previous runs. ILC is an attractive method for high-precision control because of its simplicity and data-based nature. In this research, ILC algorithm design specifically for attenuating high frequency vibrations is investigated. Through careful design of the ILC update law, fast learning convergence and small final error is achieved. One drawback of ILC is that a feedforward signal learned through ILC is only applicable to the training trajectory; learning must be restarted when the trajectory is changed. A method is presented for making ILC results applicable to any trajectory within a class of scan trajectories; this
is accomplished by using ILC as a training method for feedforward signal patterns. In IFT, controller parameters are fine-tuned incrementally using only data collected in experimental runs. IFT is applied to tune fixed-structure feedforward, feedback, and force-ripple compensator controllers. The performance of IFT is also compared with ILC in the context of iterative methods for designing feedforward control. All results are verified through computer simulations and experiments done on a wafer stage testbed system.
# Contents

List of Figures iv

List of Tables viii

1 Introduction 1
   1.1 Photolithography 1
   1.2 Repetitive processes 3
   1.3 Contributions 4
   1.4 Organization 5

2 Introduction of Wafer Stage System 7
   2.1 Hardware 7
      2.1.1 Stage Mechanical Description 7
      2.1.2 Laser Measurement System 9
      2.1.3 Linear Motor 11
      2.1.4 Digital Controller 12
   2.2 Basic Controller Design 13
   2.3 Identification of Wafer Stage 14
   2.4 Force Ripple 16
   2.5 Input shaping 19
      2.5.1 Polynomial Spline Trajectory Designs 20
      2.5.2 FIR pre-filter trajectory design 21
   2.6 Chapter Summary 27

3 Iterative Learning Control 28
   3.1 ILC formulation 29
   3.2 ILC analysis frameworks 30
      3.2.1 Lifted domain framework 31
      3.2.2 Frequency domain framework 32
   3.3 ILC Analysis 33
      3.3.1 Stability Analysis 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.2</td>
<td>3.3.2 Asymptotic Performance Analysis</td>
<td>34</td>
</tr>
<tr>
<td>3.3.3</td>
<td>3.3.3 Transient Performance Analysis</td>
<td>36</td>
</tr>
<tr>
<td>3.3.4</td>
<td>3.3.4 Robustness Analysis</td>
<td>37</td>
</tr>
<tr>
<td>3.4</td>
<td>3.4 ILC design methods</td>
<td>38</td>
</tr>
<tr>
<td>3.4.1</td>
<td>3.4.1 PD-type ILC</td>
<td>38</td>
</tr>
<tr>
<td>3.4.2</td>
<td>3.4.2 Inverse model ILC</td>
<td>38</td>
</tr>
<tr>
<td>3.4.3</td>
<td>3.4.3 Quadratic Optimal ILC</td>
<td>39</td>
</tr>
<tr>
<td>3.5</td>
<td>3.5 Summary</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>4 ILC design for vibration reduction</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>4.1 Suppression of non-repetitive disturbances at low frequency</td>
<td>44</td>
</tr>
<tr>
<td>4.1.1</td>
<td>4.1.1 Disturbance observer</td>
<td>44</td>
</tr>
<tr>
<td>4.1.2</td>
<td>4.1.2 Experimental results</td>
<td>46</td>
</tr>
<tr>
<td>4.2</td>
<td>4.2 Repetitive vibrations at high frequency</td>
<td>46</td>
</tr>
<tr>
<td>4.2.1</td>
<td>4.2.1 Vibration model</td>
<td>47</td>
</tr>
<tr>
<td>4.3</td>
<td>4.3 Proposed ILC schemes for vibration reduction</td>
<td>53</td>
</tr>
<tr>
<td>4.3.1</td>
<td>4.3.1 ILC 2: P-type with 250 Hz cutoff Q filter</td>
<td>56</td>
</tr>
<tr>
<td>4.3.2</td>
<td>4.3.2 ILC 3: P-type with notch Q filter</td>
<td>57</td>
</tr>
<tr>
<td>4.3.3</td>
<td>4.3.3 ILC 4: P-type with dynamic notch Q filter</td>
<td>57</td>
</tr>
<tr>
<td>4.3.4</td>
<td>4.3.4 ILC 5: Notch L filter</td>
<td>57</td>
</tr>
<tr>
<td>4.3.5</td>
<td>4.3.5 ILC 6: Dynamic notch L filter</td>
<td>57</td>
</tr>
<tr>
<td>4.3.6</td>
<td>4.3.6 ILC 7: Frequency-shaped L filter</td>
<td>57</td>
</tr>
<tr>
<td>4.3.7</td>
<td>4.3.7 ILC 8: Model-inverse L filter</td>
<td>58</td>
</tr>
<tr>
<td>4.3.8</td>
<td>4.3.8 Discussion</td>
<td>58</td>
</tr>
<tr>
<td>4.3.9</td>
<td>4.3.9 Experimental Results</td>
<td>59</td>
</tr>
<tr>
<td>4.4</td>
<td>4.4 Chapter Summary</td>
<td>63</td>
</tr>
<tr>
<td>5</td>
<td>5 ILC for Scan Trajectories</td>
<td>64</td>
</tr>
<tr>
<td>5.1</td>
<td>5.1 Method Description</td>
<td>66</td>
</tr>
<tr>
<td>5.1.1</td>
<td>5.1.1 Scan trajectory construction</td>
<td>66</td>
</tr>
<tr>
<td>5.1.2</td>
<td>5.1.2 Synthesis of feedforward signal for new trajectory</td>
<td>67</td>
</tr>
<tr>
<td>5.1.3</td>
<td>5.1.3 Method of calculating base feedforward signal</td>
<td>69</td>
</tr>
<tr>
<td>5.1.4</td>
<td>5.1.4 Additional Considerations</td>
<td>69</td>
</tr>
<tr>
<td>5.2</td>
<td>5.2 Simulation</td>
<td>70</td>
</tr>
<tr>
<td>5.3</td>
<td>5.3 Experiment</td>
<td>71</td>
</tr>
<tr>
<td>5.4</td>
<td>5.4 Chapter Summary</td>
<td>74</td>
</tr>
<tr>
<td>6</td>
<td>6 Iterative Controller Tuning</td>
<td>78</td>
</tr>
<tr>
<td>6.1</td>
<td>6.1 IFT Method</td>
<td>79</td>
</tr>
<tr>
<td>6.1.1</td>
<td>6.1.1 Output-related signals</td>
<td>80</td>
</tr>
<tr>
<td>6.1.2</td>
<td>6.1.2 The Three IFT Experiments</td>
<td>81</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Input-related signals</td>
<td>82</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Discussion</td>
<td>82</td>
</tr>
<tr>
<td>6.2</td>
<td>IFT tuning of feedforward controller</td>
<td>82</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Methodology</td>
<td>82</td>
</tr>
<tr>
<td>6.3</td>
<td>Convergence Analysis and Similarity to ILC</td>
<td>84</td>
</tr>
<tr>
<td>6.4</td>
<td>Controller Tuning Results</td>
<td>85</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Simulation</td>
<td>85</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Experiment</td>
<td>85</td>
</tr>
<tr>
<td>6.5</td>
<td>Force Ripple Compensation</td>
<td>90</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Methodology</td>
<td>90</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Experiment</td>
<td>92</td>
</tr>
<tr>
<td>6.6</td>
<td>IFT Tuning of Feedback Controller</td>
<td>93</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Controller Tuning Methodology</td>
<td>93</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Simulation Results</td>
<td>94</td>
</tr>
<tr>
<td>6.7</td>
<td>Comparison of two Iterative Feedforward Tuning Methods: ILC and IFT</td>
<td>96</td>
</tr>
<tr>
<td>6.7.1</td>
<td>Comparison</td>
<td>96</td>
</tr>
<tr>
<td>6.7.2</td>
<td>Discussion</td>
<td>96</td>
</tr>
<tr>
<td>6.8</td>
<td>Simultaneous tuning of ILC and IFT</td>
<td>98</td>
</tr>
<tr>
<td>6.8.1</td>
<td>Algorithm</td>
<td>100</td>
</tr>
<tr>
<td>6.8.2</td>
<td>Simulation</td>
<td>100</td>
</tr>
<tr>
<td>6.8.3</td>
<td>Experiment</td>
<td>101</td>
</tr>
<tr>
<td>6.9</td>
<td>Chapter Summary</td>
<td>103</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>106</td>
</tr>
<tr>
<td>7.1</td>
<td>Issues Addressed</td>
<td>106</td>
</tr>
<tr>
<td>7.2</td>
<td>Future Research</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>108</td>
</tr>
</tbody>
</table>
## List of Figures

1.1 Wafer scanner diagram .................................................. 2
2.1 Two stage wafer scanner prototype experimental setup ................. 8
2.2 Control loop diagram ...................................................... 8
2.3 Optical assembly in experimental setup .................................. 10
2.4 Measurement system upgrade ........................................... 11
2.5 Control loop structure during system identification ..................... 14
2.6 Diagram of stage model .................................................. 14
2.7 Experimental frequency response data from input $r$ to output $y$ .......... 15
2.8 Experimental frequency response data from input $u$ to output $y$ .......... 16
2.9 Verification of physical model in time domain .......................... 17
2.10 Verification of model based on frequency response in time domain ....... 18
2.11 Tracking error caused by force ripple disturbance ..................... 19
2.12 Schematic of step-and-scan trajectory for wafer scanning ............... 20
2.13 Plot of position, velocity, acceleration, jerk, and snap of second order trajectory 22
2.14 Plot of position, velocity, acceleration, jerk, and snap of third order trajectory 23
2.15 Plot of position, velocity, acceleration, jerk, and snap of fourth order trajectory 24
2.16 Comparison of velocity plot of three trajectories during acceleration phase 25
2.17 Comparison of velocity error for the designed trajectories: Close-up on acceleration part .................................................. 25
2.18 Reference trajectory velocity during acceleration part ................... 26
2.19 Close-up of velocity error during acceleration part ..................... 26
3.1 Structure for ILC formulation .......................................... 30
4.1 Tracking error from two runs. The 150 Hz vibration is the same between runs, while the phase of the 20 Hz vibration is different. ................ 42
4.2 System structure for consideration of disturbance observer. ............... 44
4.3 Feedback control system structure with disturbance observer ............. 44
4.4 Tracking error with and without disturbance observer. It is seen that using disturbance observer reduces error. .......................... 46
4.5 Disturbance estimate produced by the disturbance observer. ............. 46
4.6 Frequency response data of closed loop (r → y) in red overlayed with vibration model closed loop frequency response. .......................... 48
4.7 Frequency response data of plant only (u → y) in blue overlayed with vibration model frequency response. ................................. 49
4.8 Comparison of closed loop Bode plots using simple plant model and vibration model. ......................................................... 50
4.9 Tracking error of vibration model in simulation vs. experiment data. .... 50
4.10 Tracking error of vibration mode vs. simple model in simulation. ...... 51
4.11 Tracking error after 3 ILC iterations with Q filter with 250 Hz cutoff: vibration model in simulation vs. experimental data. ............. 51
4.12 Tracking error after 3 ILC iterations with Q filter with 250 Hz cutoff: vibration model vs. simple model in simulation ...................... 52
4.13 Bode plot of ILC stability transfer function for simple and vibration models. ................................................................. 52
4.14 Tracking error after 3 ILC iterations with Q filter with 100 Hz cutoff: vibration model in simulation vs. experimental data. ............. 53
4.15 Tracking error after 3 ILC iterations with Q filter with 100 Hz cutoff: vibration model vs. simple model in simulation ...................... 53
4.16 Tracking error comparison of ILC types for vibration – simulation reduction 54
4.17 Tracking error comparison of ILC types for vibration reduction – simulation – magnification of acceleration part in Figure 4.16 ............. 55
4.18 Peak error vs. ILC iteration number – comparison of ILC types ........ 55
4.19 RMS error vs. ILC iteration number – comparison of ILC types ........ 56
4.20 Bode plot of ILC stability and monotonic transfer function ............. 59
4.21 Bode plot of ILC asymptotic error transfer function ...................... 60
4.22 Comparison of tracking error for vibration compensation ILC schemes – Experimental results. Note that the inverse-model ILC is unstable, and Ptype with 250 Hz cutoff has learning transient. ......................... 60
4.23 Comparison of tracking error for vibration compensation ILC schemes – Experimental results, magnification of acceleration part. ......... 61
4.24 Comparison of RMS tracking error for different ILC schemes designed for vibration compensation. Tracking error norm reduction over learning iterations is shown. Experimental results. ......................... 62
4.25 Comparison of maximum magnitude of tracking error for different ILC schemes designed for vibration compensation. Tracking error norm reduction over learning iterations is shown. Experimental results. ......................... 63

5.1 When the reference trajectory changes, the ILC input signal must be modified, through re-learning, or other methods. .............................. 64
5.2 Construction of a second-order scan trajectory ............................. 66
5.3 Trajectory 1 ........................................................................... 71
5.4 ILC input for trajectory 1 .................................................. 72
5.5 Comparison of ILC basic signal calculated from data and predicted from model impulse response. The signals are almost identical .................................................. 72
5.6 Trajectory 2 ................................................................. 72
5.7 Calculated and learned ILC input for trajectory 2. The signals are almost identical. ................................................................. 73
5.8 Tracking error for trajectory 2. The tracking error for after 5 iterations of ILC and using the synthesized feedforward input are almost identical. ........ 73
5.9 Tracking error for trajectory 2, magnification of Fig. 5.8. The tracking error for after 5 iterations of ILC and using the synthesized feedforward input are almost identical. .................................................. 74
5.10 ILC basic signal ........................................................... 75
5.11 Calculated and learned ILC input for trajectory 2. The signals are almost identical. ................................................................. 75
5.12 Tracking error for trajectory 2 ............................................. 76
5.13 Tracking error for trajectory 2, magnification of Figure 5.12 .......... 76

6.1 Controller structure for IFT tuning ........................................ 79
6.2 Block diagram showing system controller topology .................. 83
6.3 Cost as a function of $\rho_1 \rho_2$ ........................................... 86
6.4 Reference trajectory for wafer stage ..................................... 86
6.5 Minimization of cost function (sum of squared error) in simulation over 10 cycles of tuning ................................................................. 86
6.6 Convergence of feedforward parameters in simulation .............. 87
6.7 Reduction of error after 10 cycles of tuning in simulation .......... 87
6.8 Minimization of cost function (sum of squared error) in experiment over 10 cycles of tuning ................................................................. 88
6.9 Reduction of error after 10 cycles of tuning in experiment .......... 88
6.10 Convergence of feedforward parameters in experiment with initial values $\rho_1 = 0 \rho_2 = 0$ .................................................. 88
6.11 Convergence of feedforward parameters in experiment with initial values $\rho_1 = 0.4417 \rho_2 = 0.6000$. .................................................. 89
6.12 Comparison of two feedforward controllers: 1. using parameters from iterative tuning vs. 2. using parameters corresponding to inverse of plant obtained through system identification .................................................. 89
6.13 Block Diagram with Force Ripple ....................................... 91
6.14 Convergence of Feedforward Parameters in Experiment .......... 92
6.15 Reduction of Error After 10 Cycles of Tuning in Experiment ...... 92
6.16 Iterative Minimization of Cost Function in Experiment ............ 93
6.17 Cost function decreases with tuning of PID controller in simulation .................................................. 94
6.18 Cost function decreases with tuning of PID controller ............... 95
6.19 Cost function decreases with tuning of PID controller ....................... 95
6.20 Comparison of Error ................................................................. 97
6.21 Comparison of Error - Magnification during scan phase .................. 98
6.22 Comparison of Feedforward Signal ............................................. 99
6.23 Block diagram ................................................................. 101
6.24 Reduction of error after ILC and IFT tuning ................................. 101
6.25 ILC feedforward signal after 10 iterations of tuning .................... 102
6.26 IFT feedforward signal after 10 iterations of tuning .................... 102
6.27 Convergence of parameters in IFT feedforward controller ............. 102
6.28 Reduction of error after ILC and IFT tuning in experiment ............ 103
6.29 ILC feedforward signal after 10 iterations of tuning in experiment .. 103
6.30 IFT feedforward signal after 10 iterations of tuning in experiment .. 104
6.31 Convergence of parameters in IFT feedforward controller in experiment ... 104
List of Tables

1.1 Wafer stage motion requirements .................................................. 3
2.1 Laser Measurement System Specifications ...................................... 10
2.2 Amplifier specifications ................................................................. 11
2.3 Motor specifications ................................................................. 11
2.4 Parameters for designing trajectory ............................................... 21
2.5 Comparison of step times and settle times for tracking error for different trajectories ......................................................... 21
4.1 Disturbance and vibration categorization for the wafer stage system 43
4.2 Tested ILC Types ............................................................................. 54
Acknowledgments

I am indebted to many people who shaped my experience over the last 5 years along my journey towards the Ph.D.

First, I would like to express my thanks towards my research advisor, Professor Masayoshi Tomizuka, who graciously accepted me into his lab when I was still only an undergraduate and didn’t know anything, and for patiently watching over me through the years as I learned how to become a better researcher. Because of Professor Tomizuka, I was given the precious opportunity to study at the MSC lab in Berkeley, where I could work with many of the brightest students in the field from all around the world. I am also grateful towards Professor Horowitz and Professor Fearing for serving on my dissertation committee and carefully editing my dissertation drafts.

I would like to thank the engineers at Nikon, whose comments were vital in shaping the direction of my group’s research. I also appreciate the interest in our project from the engineers at Agilent. I also thank the U.C. Discovery grant for sponsoring our project. I would also like to thank Dr. Falcon at National Instruments for her support in acquiring needed equipment. Personally, I would also like to thank the National Science Foundation and the Achievement Rewards for College Scientists Foundation for sponsoring my fellowship, making it possible for me to continue my studies here.

The most influential part of my academic life has been discussions with my labmates in the MSC lab. I would like to express my appreciation for their research guidance and immeasurable support and encouragement, and my admiration at their enthusiasm. Special thanks are extended to the precision motion control group, including Sandipan Mishra and Benjamin Fine for their mentorship and guidance, and to Emma Yu and William Yeh for their help and companionship.

Finally, I would like to thank my family for their support and care, and for their sacrifices making it possible for me to be here.
Chapter 1

Introduction

1.1 Photolithography

Semiconductors have become a ubiquitous part of the modern world, found in many products we use everyday, such as computers, consumer electronics, automobiles, measurement and test devices, space systems, and data processing equipment. The semiconductor industry has experienced steady growth in the last few years with demand for semiconductor products.

The semiconductor manufacturing process has many complex steps, and extracting the maximum possible performance from each subprocess is vitally important to stay competitive in today’s semiconductor industry. This thesis concerns one subprocess in particular, photolithography, the process of transferring the circuit pattern from a mask onto the silicon wafer. Photolithography involves preparing the silicon wafer, coating it with photoresist, transferring the pattern onto the resist, etching, and removing the photoresist. A light source, commonly an excimer laser AR-F laser (DUV, deep ultraviolet lithography), is used to transfer a pattern from a mask (“reticle”) onto a chemical photoresist covering the silicon wafer. The light is passed through a lens reducer system before exposing the photoresist, and the exposed areas harden, creating the circuit. The mask pattern represents one die, and is repeated many times on the wafer. In addition, many exposures are needed to create patterns and layers on a single die, so a single wafer may go through the lithography process multiple times. Advances in photolithography can be said to be a major driving force behind the continuation of Moore’s law. Advances are made through improvements in the resolution achievable in lithography, both by decreasing the wavelength (through new light source technology), and by increasing numerical aperture of the reducer system (through lens technology or immersion lithography). The resolution improvement is what allows smaller and smaller feature sizes on the wafer.

With regards to the lithography process, the International Technology Roadmap for Semiconductors (ITRS) [17] has identified three challenges: critical dimension (CD) control,
overlay control, and low total cost-of-ownership. CD is defined as a measure of the width of a particular feature within a given pattern. The 2011 target set by the ITRS is the 22 nm node. For good overall circuit performance, CD needs to be precisely controlled in all aspects: wafer-to-wafer, die-to-die, and within die. Overlay refers to the process of building layers in the wafer by multiple scanning over the same section. Due to the fact that misalignment between current and underlying layers can cause short circuits and connection failures, which in turn impact fabrication yield and profit margins, overlay control becomes increasingly important. This is particularly relevant as the adoption of double patterning becomes widespread. State-of-the-art lithography in the semiconductor manufacturing industry requires CD 3-sigma between 2-5nm and overlay accuracies between 1-3nm. To minimize cost, the scanning process in lithography must be done at a high velocity rate to maintain high throughput while still meeting the CD control and overlay requirements.

In the projection-based lithography method, the pattern transfer is often done using an opto-mechanical device called a wafer stepper or a wafer scanner, a diagram of which is shown in Figure 1.1. The mask is placed on the reticle stage, and the silicon wafer on the wafer stage. The light is passed through a slit before the mask. During exposure, the reticle stage travels in the x-direction ("scan") perpendicular to the slit direction, and the wafer stage travels in the opposite direction for only 1/4 the distance; this increases the relative movement. After the exposure of one die, the wafer stage repositions the wafer under the

Figure 1.1: Wafer scanner diagram
Table 1.1: Wafer stage motion requirements

<table>
<thead>
<tr>
<th>Process</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage Velocity</td>
<td>2 m/s (reticle)</td>
</tr>
<tr>
<td>Synchronization accuracy</td>
<td>&lt; 10nm</td>
</tr>
<tr>
<td>Positioning accuracy</td>
<td>&lt; 1nm</td>
</tr>
<tr>
<td>Overlay error</td>
<td>&lt; 3 nm</td>
</tr>
</tbody>
</table>

mask so that another area can be exposed (“step”). The wafer stage is able to move in both x- and y-directions.

Because the feature sizes created on the wafer are of such small dimension (nanometer scale), it is crucial that positions of the stages be controlled to an even higher degree of precision [12]. As higher resolution technologies are developed in the future, it is necessary for the stage motion control accuracy to keep up. Additionally, the stages must be able to control position with high accuracy at the same time as moving at high speeds, to preserve high throughput of the machines. The motion specifications are listed in Table 1.1. Advanced control algorithms are needed to achieve such high control precision.

1.2 Repetitive processes

The repetitive nature of the step and scan motion can be exploited to achieve high precision control. In fact, repetitive motions are common in many robotic manufacturing processes. In such cases, information collected from past runs of the process, or “iterations”, can be used to improve the performance in future runs. Some main strategies for using repetitiveness include:

- **Iterative Learning Control** - Iterative Learning Control (ILC) applies to systems that repetitively perform the same task and that stop and restart between tasks, so that they have identical initial conditions and finite time-length. In ILC, the feedforward signal is incrementally adjusted with each trial of the task to reduce error. ILC has found wide application to many industrial robots such as CNC machines, injection-molding machines, and wafer scanners [8] [9], [11] [21].

- **Repetitive Control** - Repetitive Control (RC) is similar to ILC, except it is designed for systems that continuously repeat the same task (periodic), so that the initial condition of each repetition is not necessarily the same, leading to different analysis [20]. RC application is commonly found in track following disk drives [7], and contour tracking for CNC machines [30] [31].

- **Iterative Feedback Tuning** - Iterative Feedback Tuning (IFT) is a data-driven method of automatically adjusting controller parameters (rather than a signal) be-
tween iterations \([14], [15], [16]\). The IFT algorithm iteratively updates the controller parameters so as to minimize a certain cost function. Normally, such an optimization would require a complete model of the plant and a full-order controller, but in IFT, the optimization is performed iteratively by using an estimate of the gradient of the cost function that is calculated based on data collected from experiment iterations. The IFT method has been previously used to tune controllers of systems such as chemical plants, DC-servo systems \([14]\), two-mass torsional motor systems \([22]\), two-mass spring systems with friction \([13]\), and wafer stages \([32]\).

1.3 Contributions

This dissertation investigates the use of iterative methods for improving positioning accuracy of a wafer stage. There are three main contributions: 1.) the application of ILC for high precision control of systems with vibration, 2.) making ILC tuning results applicable to multiple trajectories, and 3.) force ripple disturbance compensation through model parameter tuning.

First, the application of ILC for high precision control of systems with vibrations is investigated. First we consider whether or not the vibration is iteration-invariant, and the frequency range. It is well-known that ILC works well for reducing error caused by disturbances which are the same from iteration-to-iteration, but nonrepetitive disturbances may hurt the performance. In the case that the vibration is low frequency, a disturbance observer may be used to compensate. As for the second consideration, if the repetitive vibration is of high frequency, it is found that ILC monotonic convergence and good converged performance are difficult to achieve together. Several design methods for ILC algorithms especially for high-frequency vibration reduction are proposed, tested, and compared.

Next, a method for making ILC tuning results applicable to multiple trajectories is developed and tested. After ILC tuning iterations are carried out, the result is a single feedforward signal, whose application is used to reduce tracking error, but may only be used for the same trajectory as used during the learning iterations. If high precision tracking is desired even with a different trajectory, the learning iterations must be restarted to learn the new feedforward signal from scratch. In this thesis, a method is developed to make the result of ILC tuning extendable to any scan trajectory, with consideration restricted to polynomial spline scan trajectories of limited velocity and acceleration. The method involves using ILC as a training method to find a good feedforward signal; then based on known information about the trajectory parameters and the LTI nature of the system, a "base feedforward signal pattern" is extracted. The base signal is recombined in order to construct feedforward signals to work with other trajectories.

Third, this thesis presents a method of force ripple disturbance compensation, through the use of a nonlinear feedforward controller compensator. The controller structure is set, based on a nonlinear model of the force ripple. Then, the IFT method is used to tune
the controller parameters in order to minimize tracking error; the parameters represent unknown coefficients in the force ripple model. The improvement of tracking error via IFT is compared with that of ILC. While it is true that ILC achieves good performance in many kinds of problems, ILC signals are only applicable for the same trajectory, while IFT-tuned controllers can be used for different trajectories. Based on the advantages of each method, we then propose a novel method of simultaneously tuning an ILC feedforward signal and IFT feedforward controller with the end result of good performance obtained from ILC while still ending up with a tuned controller to be used with different trajectories.

1.4 Organization

This thesis is organized as follows.

In Chapter 2, the wafer stage experimental testbed system is introduced. All of the components which make up the position control feedback loop, including the stage hardware, laser interferometer sensor, linear motors, and electronic hardware on which the controller is implemented, are described in detail. The basic feedback and feedforward controller structures are laid out. Next, a transfer function basic model is identified and tested. Finally, design methods for the step and scan trajectory are presented, and trajectory shaping methods for vibration reduction are applied and tested.

In Chapter 3, we introduce the foundations for understanding ILC. First, a basic learning scheme is presented. Characteristics of ILC schemes such as stability, monotonic convergence, and asymptotic performance are discussed and analyzed from the point of view of two frameworks: the lifted domain, and the frequency domain. Several design methods of ILC algorithms are also mentioned.

In Chapter 4, we investigate ILC scheme design specifically for vibration reduction. The use of a disturbance observer for reduction of non-repeating disturbances together with the use of ILC for repeating disturbances is shown. Moreover, several design methods for ILC for reducing repetitive vibrations in high frequency are proposed and examined.

In Chapter 5, a method of generalizing ILC results to different scan trajectories is presented. Scan trajectory construction is discussed, followed by the method of constructing feedforward signals from scan trajectories from parts. The method’s performance is supported by simulation and experiment results.

In Chapter 6, we introduce IFT theory. Then we show how this method can be applied to tune feedforward and feedback controller parameters with the goal of minimizing the norm of error. We also show that this method can be used to identify coefficients of sinusoidal harmonics to be used in a force ripple compensator. We follow the IFT results by showing a comparison with ILC in the context of comparing two iterative feedforward tuning methods. We also illuminate parallels between the ILC update law and IFT iterative search, with regards to convergence. Also, a scheme for simultaneous ILC and IFT learning is presented. In each section, simulation and experimental results are presented to verify the effectiveness
of the methods.
Chapter 2

Introduction of Wafer Stage System

In this chapter, we introduce a wafer stage testbed system for experimental verification of precision control methods. A picture of the stage is shown in Fig. 2.1. This testbed setup is meant to imitate one axis of the wafer stage part of a full industrial wafer scanner. First, we will describe the physical hardware and software needed to control the stage. Following the description of the physical hardware, we present models of the wafer stage for the purposes of control and simulation. Next, we present a basic feedback and feedforward controller, which provide the benchmarks for comparison of the advanced control algorithms designed in the following chapters. Finally, we will describe typical scan trajectories for photolithography application and their design methods.

2.1 Hardware

In order to control the motion of the wafer stage, the testbed system includes many hardware components, interconnected as shown in Fig. 2.2. First, for measuring the stage’s position, a laser interferometer system is used. Then, the sensor data is input to the digital controller, which is implemented on a Field-Programmable Gate Array (FPGA) and real-time PC and written in LabVIEW graphical programming language. Finally, the wafer stage is moved by a force applied by a linear permanent magnet motor (LPMM). Each of the individual components will be described in further detail in the following subsections.

2.1.1 Stage Mechanical Description

The wafer stage system in Fig. 2.1 is composed of two moveable parts: a single-axis linear stage (wafer stage) which is coupled to a movable countermass. The wafer stage is essentially an aluminum platform that is moveable in one direction along two guide rails, and is floated on an air bearing to reduce friction. The moveable countermass is included in the system to keep the center of momentum of the entire system at a fixed point when the wafer
Figure 2.1: Two stage wafer scanner prototype experimental setup

Figure 2.2: Control loop diagram
stage is moving. The stator of the linear motor, (the permanent magnet rail), is mounted on the countermass, and the mover of the linear motor (the magnet coils) is mounted under the stage. Therefore, when a current is put through the motor, a force is generated to move the stage in one direction. The equal and opposite reaction force acts on the countermass, which is mounted on a linear bearing, moving it slightly in the other direction. The overall center of gravity of the system remains the same. As a result, there is no resulting force transmitted to the base, which decreases vibrations in the support structure which would transfer to the stage. The countermass is supported by a pair of linear roller bearings, and is actuated by a second linear motor coupled between the countermass and base. Position is measured by a coarse linear optical encoder. However, since the purpose of the countermass is to act only as a momentum balancer, we are not interested in its precision tracking, so a weak PID controller is sufficient for the countermass.

In 2008, an additional stage was added to the setup to imitate the reticle stage. The lower stage represents the wafer stage, and the upper represents the reticle stage. The support structure for the base of the upper stage is built on top of the base of the lower stage in order to study vibrational coupling of the two during motion. The support structure was designed so that its resonant vibration frequency is higher than the desired system bandwidth. The hardware, sensors, motors, and electronics for the two stages are identical.

2.1.2 Laser Measurement System

The wafer stage position is measured by a laser interferometer position measurement system. A complete laser measurement system is a combination of a laser source, beam directing optics (mirrors, beam splitter, shown in Figure 2.3), measurement optics (interferometer), optical receiver, and axis electronics. The laser measurement system was upgraded in 2009. The following provides a detailed description of the new system along with a brief comparison with the capabilities of the previous system.

Laser interferometry measures the linear displacement of a system and works by measuring the change in frequency of a laser beam due to its reflecting off a moving object, due to the Doppler effect. A moving target causes a doppler shift in the reflecting laser beam due to its velocity. A receiver can then compare the frequency of the laser from the original laser source (reference beam) and reflected laser (measurement beam). The slight difference in frequency between the beams will cause a change in the superimposed laser beams’ intensity, which the receiver measures. To guide the laser beam from the laser head to the moving stage, a system of adjustable mirrors and beam splitter is used. Critical to laser interferometry, a half-mirror is used to pass both the reflected and original beams back to the receiver. In addition, the entire optics hardware setup is mounted on top of a vibration isolation table to prevent vibrations from the building floor to transmit to the optics hardware, affecting the laser beam travel path and degrading the measurement accuracy.

Using the measured laser signal, the axis board electronics compute the position data in the form of a 32-bit number and stores it in a data register. The signal is sampled at a rate
Table 2.1: Laser Measurement System Specifications

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agilent E1709A Remote High Performance Receiver, 5517F Laser Head</td>
<td></td>
</tr>
<tr>
<td>N1231B Three-Axis High Performance Laser Board with External Sampling</td>
<td></td>
</tr>
<tr>
<td>Maximum velocity</td>
<td>2 m/s</td>
</tr>
<tr>
<td>Resolution</td>
<td>0.15 nm</td>
</tr>
<tr>
<td>Resolution extension</td>
<td>x1024</td>
</tr>
<tr>
<td>Data Update Rate</td>
<td>20 MHz</td>
</tr>
<tr>
<td>Number axes</td>
<td>3</td>
</tr>
</tbody>
</table>

of 20 MHz, and processing time takes 700 ns, so that although position data is available every 50 ns, there is a 700 ns data age. The position data registers are accessible in one of two ways: through software interface (the PCI bus) or hardware interface (four connectors of DIO pins on the top of the card). The hardware mode is chosen due to higher speeds. An interface printed circuit board was designed and created to connect the axis board hardware connectors to the controller. Measurement system specifications are summarized in Table 2.1.

The upgraded measurement system has greatly improved the tracking precision possible due to its higher sampling rate and greater resolution. The current sampling rate of 20 MHz is a factor of 10 times higher than the previous measurement system, reducing position errors due to sample jitter. Additionally, the current resolution of 0.15 nm is 30 times finer than the previous resolution of 5 nm. The resulting improvement in tracking precision is demonstrated in Fig. 2.4.
2.1.3 Linear Motor

The stage and countermass are driven by a pair of linear permanent magnet motors (LPMMs). LPMMs are popular for high-precision high-speed motion control applications because their low inertia, mechanical simplicity, and low friction allow for fast and smooth movements [29]. Linear motors work just like their rotary counterparts: by passing an electric current through the motor coil in a magnetic field, a force is produced. We use a Trilogy 310-3 three-phase motor (epoxy-core, I-shaped top mounted) for the stage, and the smaller Trilogy 310-1 motor for the countermass.

The motors are driven by a Trust Automation TA320 motor driver / power amplifier, with power supplied from a Matsusada RE60-20. The TA320 produces motor currents for each of the three coils as specified. In our setup, the motor phase commutation is done in software. Two of the three motor current signals are provided to the amplifier, and the amplifier computes the third one independently. The motor and amplifier specifications are summarized in Tables 2.2 and 2.3.

### Table 2.3: Motor specifications

<table>
<thead>
<tr>
<th>Trilogy i-Force Ironless Linear Motor 310-3A WD1 or WD4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Force constant</td>
<td>40.9 N/A</td>
</tr>
<tr>
<td>Peak force</td>
<td>600 N</td>
</tr>
<tr>
<td>Continuous force</td>
<td>133.9 N</td>
</tr>
</tbody>
</table>
2.1.4 Digital Controller

The controller hardware consists of three main subsystems: FPGA, real-time controller, and host computer.

A National Instruments FPGA (NI PXI-7831R Reconfigurable IO FPGA for PXI bus) is used for interfacing to all sensors and motors. FPGAs have several advantages over microprocessors including speed of execution and truly parallel processing, because program logic is implemented in hardware. Disadvantages include the time necessary to recompile code to load on the FPGA, limited space on the FPGA, and difficulty of floating point calculations. The PXI-7831R can be reprogrammed through the LabVIEW FPGA module, which is very similar from a user’s point of view to standard LabVIEW. The FPGA program for the wafer stage includes several loops running in parallel: 1. An encoder quadrature counters for monitoring the encoder signals of the countermass; 2. A safety loop, 3. A main timed loop which integrates the functions of reading interferometer position registers, controlling PID loop timing, and sending motor signals to motor amplifiers. In the third loop, the laser position is first read, then an interrupt signal is generated. When the real-time host controller receives the interrupt, the controller calculates the PID controller output for the next cycle, and acknowledges the interrupt. Upon receiving the acknowledgment, the FPGA reads the control output, stores it, and waits for the next loop cycle to send the output to the motors simultaneously as the sensor is being read. The order is important because it reduces timing uncertainty between the different components (sensor, FPGA loop, real-time loop).

The feedback controller is implemented on a National Instruments real-time target PXI-8176 communicating with the FPGA through PXI bus with the control software programmed in LabVIEW. The main control loop (PID controller) is implemented here, The real-time target handles initialization of the position sensors, reading and writing data files, Real-time target is where sensor data conversions to physical units are made and PID control is done. Data is also saved here, and communication with the host computer.

The third component of the controller is the host computer, communicating with the RT controller through the network. The host computer is a Windows PC with LabVIEW loaded. The code development is done on the host computer and uploaded to the RT target at runtime. While the controller is running, the host computer GUI is linked to the RT target over the network, allowing us to monitor the experiment and to change controller parameters in real-time.

Communication with the host computer is done through network. At the host computer, the code is written, and the user can control the system through a GUI.
2.2 Basic Controller Design

This section describes the structure of the feedback and feedforward controllers. Good feedback controller design is necessary for stabilizing systems, improving tracking performance, and making performance robust to disturbances (such as gravity, random disturbances, etc.) However, because feedback control by definition uses sensor information feedback to correct for errors, it is limited to being reactive. When a setpoint change is needed, the feedback controller takes some time to correct it. On the other hand, feedforward controllers use a-priori known information about the desired trajectory and plant dynamics to predict the necessary controller action. For high-precision motion control applications, both well-designed feedback and feedforward controllers are needed.

The control loop structure is shown in Figure 2.5. For the feedback controller, a standard PID controller is used. The structure of the controller is given by

\[ C(s) = k_p \left( 1 + \frac{k_i}{s} + k_ds \right). \]

where \( s \) is the Laplace variable, and \( k_p, k_i, \) and \( k_d \) are controller parameters. For implementation, the above continuous-time PID controller was discretized using the backwards-difference approximation \( s = \frac{1-z^{-1}}{T_s} \), giving

\[ C(z^{-1}) = k_p \left( 1 + k_i \frac{T_s}{1 - z^{-1}} + k_d \frac{1 - z^{-1}}{T_s} \right) \]

where \( T_s = 0.0004 \text{s} \) is the sampling time, and \( z^{-1} \) denotes one step delay. The PID controller parameters were hand-tuned, resulting in the \( k_p = 30000, k_i = 2, \) and \( k_d = 0.012 \). In actual implementation, the derivative term was first passed through a low-pass filter (4-tap FIR filter) to reduce noise from sensor measurement quantization and sensor noise. In the remainder of this thesis, the feedback PID controller was fixed, while other control structures were implemented as an "add-on" feature.

The feedforward controller is designed to approximate the inverse plant model identified later in 2.3, or

\[ F(s) = a_1s^2 + a_2s. \]

In actual implementation, the following structure is used

\[ u_F(k) = a_1 a(k) + a_2 v(k) \]

where \( v(k) \) and \( a(k) \) are the sampled velocity and acceleration trajectories. In implementation, the velocity and acceleration trajectories are calculated beforehand at the same time the reference trajectory is designed, and then sampled. This has the effect of saving calculation time in the control loop at runtime, and also avoiding errors caused by approximating derivatives after sampling.
2.3 Identification of Wafer Stage

This section describes the identification of a model of the wafer stage for simulation and control design purposes. Two methods for identification, physical modeling and sine sweep frequency response estimation, were used.

A simple physical model of the stage is shown in Fig. 2.6. The wafer stage was modeled as a mass with viscous damping:

\[ mi\ddot{y} + b\dot{y} = F \]

where \( m \) is the stage mass, \( b \) is viscous friction coefficient, \( F \) is force applied by the linear motor, and \( y \) is the linear displacement (in meters) of the stage relative to the base. Coulomb friction and other non-viscous friction forces were considered negligible since the stage is supported on an air bearing, so only viscous friction is included. Furthermore, the force exerted by the motors was modeled as being proportional to the voltage output by the controller, or \( F = ku \), where \( u \) is the controller output voltage, and \( k \) includes amplifier and motor gains. This results in the transfer function model

\[ P(s) = \frac{k}{ms^2 + bs} \]  

where the input is considered to be voltage input to the motor amplifier (units of volts) and output is position of the stage in meters.

The values of \( k, m, \) and \( b \) were measured or computed directly. The moving part of the stage was weighed and mass \( m \) was determined to be 5.3 kg. The hardware gain \( k \) was computed to be 11.79 N/V based on the gains shown in Tables 2.2 and 2.3. The friction
coefficient $b$ was determined to be 7.2 from step response experiments. The resulting transfer function is

$$
P(s) = \frac{11.79}{5.3s^2 + 7.2s}
$$

In addition to the physical model, the frequency response was also measured from a sine sweep experiment. A series of sinusoidal input signals of different frequencies were given to the system. The amplitude gain and phase shift were measured by comparing the discrete fourier transforms of the output and input signals. The frequency response was identified in the range from 0.1 Hz to 1000 Hz. Between 0.1 Hz and 1 Hz, data was obtained at intervals of 0.1 Hz; between 1 Hz and 200 Hz, data was collected at every 1 Hz; and between 200 Hz and 1000 Hz, data was collected for every 10 Hz. Identification was done in closed-loop with the default PID controller, as shown in Fig. 2.5. The results are plotted in Figs. 2.7 and 2.8. Fig. 2.7 is the direct plot of the frequency response from $r$ to $y$ and Fig. 2.8 is the response from $u$ to $y$. From the graph, the bandwidth of the closed-loop system is determined to be around 100 Hz.

Using the data from Fig. 2.8, a transfer function of the form in Eq. 2.1 was fitted to the data using a linear least-squares algorithm. The Matlab command `invfreqs` solves the
Figure 2.8: Experimental frequency response data from input $u$ to output $y$

The two models in Equations 2.2 and 2.3 were then tested to verify their ability to predict the response of the actual system. The performances of the two models at reproducing the tracking response of the system to the standard scan trajectory were compared. In Figure 2.9, the simulated output of the first transfer function model (the model identified from measuring physical parameters). In Figure 2.10, the same is plotted for the second transfer function model (the model identified from frequency response). It is seen that the physical model predicts actual plant response more accurately for this particular scan trajectory.

### 2.4 Force Ripple

One complication arises from *force ripple*, a disturbance force that occurs in linear permanent magnet motors. Force ripple is a periodic disturbance force which varies according to the relative stator and mover position of the linear motor. The effect of the force ripple
Comparison of error profiles

Figure 2.9: Verification of physical model in time domain.
Figure 2.10: Verification of model based on frequency response in time domain.
can be seen in Figure 2.11. It is caused by irregularity in the magnet position on the stator, irregularity in the coil shapes inside the mover, and inaccuracy of commutation. In some linear motors, the motor coils have an iron core, which makes the force ripple even stronger due to the magnetic attraction ("cogging force"); however the motor used in this setup has an epoxy core so this effect is negligible. So although the plant was modeled as an LTI, actually there is a small nonlinearity, in the form of a position-varying gain.

Force ripple disturbance is modeled as a nonlinear function of the stage position and controller voltage output, given by the following equation

\[ F_{\text{ripple}}(y) = u(y) \left[ \sum_{k=1}^{N} a_k \sin(k\omega_0 y) + \sum_{k=1}^{N} a_k \cos(k\omega_0 y) \right] \]

where \( y \) is stage position, \( F_{\text{ripple}}(y) \) is the force caused by the force ripple disturbance, \( u \) is the controller output voltage, and \( \omega_0 = \frac{2\pi}{p} \) where \( p \) is the pitch of the permanent magnets.

### 2.5 Input shaping

In this section, we present the design of the reference trajectory used for wafer scanning. In the typical operation of an industrial wafer scanner, the wafer stage performs a step and scan motion – the stage scans at constant velocity for the length of one die in the x-axis direction, then the stage steps to the next die in the y-axis direction and repeats the constant-velocity scan back in the opposite direction (See Figure 2.12 for reference).

In the x-axis, a typical scanning motion of the wafer stage consists of a short acceleration to a desired scan velocity, a constant velocity scan over some distance, and then a deceleration to rest, then a short wait time while the stage is stepped in the y-direction. This is followed by the same motion in the return direction. Therefore the wafer stage scanning motion can be considered as an example of point-to-point motion in terms of velocity. It is desired for the stage to reach the constant scan velocity in as short a time as possible while obeying constraints such as maximum allowable acceleration imposed by hardware or safety factors. However, depending on the frequency content of the trajectory, structural vibrations may be
excited if the acceleration of the stage is not done smoothly. These two design goals must be simultaneously considered when creating trajectories for point-to-point motions.

There are several approaches to designing trajectories for point-to-point motion [26]. The most basic way is to design trajectories as polynomial splines, keeping in mind the constraints on maximum position, velocity, acceleration [19]. To achieve more smoothness in the motion trajectory, further constraints on jerk (the derivative of acceleration) and snap (derivative of jerk) can be added. Other approaches include using a lowpass filter to eliminate vibration-causing high-frequency components of the trajectory implemented as a prefilter, or to use the verisign method [10]. Another approach is to pass a designed trajectory through an FIR prefilter with imaginary axis zeros placed so as to cancel out peaks in the plant response [26], [27]. This has the effect of breaking up a step reference into series of smaller steps so that the induced vibrations destructively cancel. The filter coefficients are determined based on the natural frequency and damping ratio of the vibrations.

In this section, the basic design of polynomial spline trajectories with acceleration limits, jerk limits, or snap limits is presented. Vibration reduction performance is compared with the FIR filter method of input shaping.

### 2.5.1 Polynomial Spline Trajectory Designs

We designed and compared three different polynomial spline trajectories: Limited-acceleration trajectory, limited-jerk trajectory, and limited-derivative-of-jerk trajectories. Because the limited-acceleration trajectory can be represented as a piecewise continuous function of second-order polynomials, we call it "Second Order". Similar holds for the limited-jerk and limited-snap trajectories so we call them "Third Order" and "Fourth Order". The parameters used for trajectory design are shown in table 2.4. For each trajectory order, a trajectory is uniquely determined by this set of constraints. From the constraints, the tra-
Table 2.4: Parameters for designing trajectory

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum travel (m)</td>
<td>0.25</td>
</tr>
<tr>
<td>Maximum velocity (m/s)</td>
<td>0.1875</td>
</tr>
<tr>
<td>Maximum acceleration (m/s²)</td>
<td>5</td>
</tr>
<tr>
<td>Maximum jerk (m/s³)</td>
<td>300</td>
</tr>
<tr>
<td>Maximum snap (m/s⁴)</td>
<td>50000</td>
</tr>
</tbody>
</table>

Table 2.5: Comparison of step times and settle times for tracking error for different trajectories

<table>
<thead>
<tr>
<th></th>
<th>Second Order</th>
<th>Third Order</th>
<th>Fourth Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step time (s)</td>
<td>0.0372</td>
<td>0.0540</td>
<td>0.0600</td>
</tr>
<tr>
<td>Settle time (s)</td>
<td>0.0428</td>
<td>0.0236</td>
<td>0.0228</td>
</tr>
<tr>
<td>Sum of step and settle (s)</td>
<td>0.0800</td>
<td>0.0776</td>
<td>0.0828</td>
</tr>
</tbody>
</table>

The designed reference trajectories are shown in Figs. 2.13, 2.14, and 2.15. For each trajectory, the position, velocity, acceleration, jerk, and snap are also shown stacked vertically for comparison. The jerk for the second order trajectory and derivative of jerk for both second and third orders should be infinite in theory, but due to discretization and approximation of derivative as backwards difference, the value is not infinite, but still very large. Fig. 2.16 shows a comparison of the velocity profile during the acceleration to scan velocity. Notice that the second order trajectory reaches scan velocity the soonest, but that its plot has sharp corners, while the fourth order trajectory although is the smoothest, takes the longest to reach scan velocity.

The trajectories were tested on the experimental wafer stage system under a default PID controller. The velocity error is plotted in Fig. 2.17. It can be seen that the second-order trajectory has the fastest rise time, but that it excites the most oscillations which make it take longer for error to settle within bounds. The rise time and settle time and their sum are compared quantitatively in Table 2.5. It is confirmed that second order has the fastest rise time but slowest settle time, and that fourth order has fastest rise time but slowest settle time, but the total of the two is the smallest for the third order trajectory.

2.5.2 FIR pre-filter trajectory design

Next we designed and tested a trajectory passed through a FIR pre-filter for reducing vibrations. The frequency of the vibrations is around 160 Hz, and has damping ratio of 0.53, so we chose a 3-tap FIR filter with appropriate coefficients to cancel this out. A plot of the reference trajectory velocity of original trajectory (second order) and after it has been
Figure 2.13: Plot of position, velocity, acceleration, jerk, and snap of second order trajectory
Figure 2.14: Plot of position, velocity, acceleration, jerk, and snap of third order trajectory
Figure 2.15: Plot of position, velocity, acceleration, jerk, and snap of fourth order trajectory
Figure 2.16: Comparison of velocity plot of three trajectories during acceleration phase

Figure 2.17: Comparison of velocity error for the designed trajectories: Close-up on acceleration part
passed through the filter is shown in Fig. 2.18. Notice it has been delayed a little, but not as much as third or fourth order trajectories. The trajectory was applied to the closed-loop system and the resulting velocity error is shown in Fig. 2.19. We can see that the shaped trajectory has much lower residual vibration than the original one.

Figure 2.18: Reference trajectory velocity during acceleration part

Figure 2.19: Close-up of velocity error during acceleration part
2.6 Chapter Summary

This chapter gives important context for understanding the subsequent chapters, and provides motivation for the development of precision control methods presented later. In this chapter, we introduced the prototype wafer stage experimental setup. The hardware components were described and shown as part of the completed feedback control loop. The basic feedback controller algorithm was also introduced. Next, we showed one possible model for the plant as well as presenting experimental identification data. Finally, the design of the step and scan trajectory for the stage was introduced.
Chapter 3
Iterative Learning Control

Iterative learning control (ILC) is a control method for improving the performance of systems performing a repetitive task. ILC is based on the idea that the performance of a system executing a repetitive task can be improved by learning from previous executions or trials. In essence, the learning aspect of ILC is similar to how humans learn to perfect a task through practice and repetition.

For a motion control system, the goal of ILC is to synthesize the best input signal to the system to drive the tracking error to zero. This is done by repetitively adjusting the control signal each iteration based on the measured plant output of previous iterations. In this way, information learned from previous runs is used to incrementally improve system performance each run.

ILC is one of several types of learning control strategies, including adaptive control, or neural networks. Similarities between these strategies is that they all use feedback information to adjust system parameters or signals to achieve better performance. These strategies may all be applied to systems with unknown disturbances. The main difference of ILC is that it is formulated and applied for systems executing a repetitive task, it adjusts a feed-forward signal rather than controller parameters, and learning is carried out in terms of an entire iteration time interval. ILC is also very similar to repetitive control (RC), the main difference being that ILC systems are assumed to be stopped and reset between iterations, while RC systems run continuously.

ILC is an attractive option due to several important advantages. First of all, ILC is simple to design, implement, and analyze. Furthermore, designing an effective ILC requires very little plant model knowledge and disturbance knowledge. Even when faced with such limited knowledge, ILC is often able to achieve a very high degree of tracking precision. In fact, it has been found that on the wafer stage test system, the best possible performance achievable is reached with ILC, over other control methods including model-based feedforward, adaptive control, or disturbance observer, demonstrating that ILC is a very powerful tool for achieving precision motion tracking. In addition, ILC does not upset the feedback loop stability because it adjusts a feedforward signal, instead of feedback controller parameters. Also, the
The progress of the learning can be monitored by examining the computed ILC signal between the iterations, so that we are able to immediately interrupt the learning process if it seems that it’s diverging. ILC also has wide potential for application, since repetitive tasks are common in robotic manufacturing processes.

The aim of this chapter is to introduce the fundamentals of ILC and to set up a framework for its understanding. ILC will be introduced from the perspective of mechanical systems. First, we will introduce the problem formulation and present a basic ILC control law. We will also comment on characteristics of a good ILC controller. Next, we will briefly describe several methods for designing the ILC control law. Following these presentations, we will consider important properties of ILC control laws such as convergence and steady-state performance, conducting the analysis from the point of view of two frameworks (the lifted domain and frequency domain). The results of this chapter may be found in greater detail in references [1] [6] [23] [24] [20].

3.1 ILC formulation

In this section, we will introduce ILC for a general feedback control system. Consider the system described in the block diagram shown in Fig. 3.1. The systems $P$ and $C$ are assumed to be discrete-time, linear time-invariant systems with proper rational transfer functions. The closed-loop system is assumed to be asymptotically stable. $e_j$ is the tracking error signal, defined in the usual way as

$$e_j(k) = r(k) - y_j(k).$$

where $r$ is the reference trajectory, and $y_j$ is the measured output. The subscript $j$ indicates the iteration number, and $k$ is the discrete time index. $u_j^{ILC}$ is the ILC control signal. $r$ is the desired trajectory which is the same from iteration to iteration. $d_j$ is an exogeneous disturbance signal which includes both repeated disturbances (invariant from iteration to iteration), as well as nonrepeated disturbances. It is assumed that the system operates for a finite duration of time, so that all the discrete signals are finite length of $N$ samples, i.e. $k \in 0...N - 1$. Also, it is assumed that the initial conditions of the system are the same from iteration to iteration.

The output relation of this system is given by

$$y_j(k) = \frac{P(q)C(q)}{1 + P(q)C(q)}r_j(k) + \frac{P(q)C(q)}{1 + P(q)C(q)}u_j^{ILC}(k) + \frac{P(q)}{1 + P(q)C(q)}d_j(k)$$

where $q$ is the discrete time shift operator. Here, the notation $P(q)$ represents an operator that is a rational function of polynomials of $A$ and $B$ of $q$, or $P(q) = B(q)/A(q)$. The input-output relationship $y(k) = B(q)/A(q)u(k)$ represents a difference equation $A(q)y(k) =$
$B(q)u(k)$. $P(z) = B(z)/A(z)$ is the z-domain transfer function. Similar holds for the operator $C(q)$. The above equation may be rewritten
\begin{equation}
y_j(k) = T_r(q)r_j(k) + T_u(q)u_j^{ILC}(k) + T_d(q)d_j(k)
\end{equation}
with appropriately defining $T_r, T_u$ and $T_d$.

In this formulation, the goal of the ILC is to synthesize the best ILC feedforward signal $u_j^{ILC}$ such that the plant follows the desired trajectory, or in other words for tracking error $e$ to become as close to zero as possible. This is accomplished by iteratively adjusting the control signal $u_j^{ILC}$ based on the output error from the previous iteration:
\begin{equation}
u_{j+1}^{ILC}(k) = f(u_j^{ILC}(0) \ldots u_j^{ILC}(N-1), e_j(0) \ldots e_j(N-1)).
\end{equation}
A commonly used linear update law is given by
\begin{equation}
u_{j+1}^{ILC}(k) = Q(q)(u_j^{ILC}(k) + L(q)e_j(k + m)).
\end{equation}
In this case, the task of ILC design becomes how to best design the update law, with the design variables of $L$ (known as the learning filter), $Q$ (known as the $Q$ filter), and $m$ (known as the ILC delay).

In achieving the ultimate goal of reduced tracking error, several factors must be taken into consideration. The designed ILC method should ultimately reduce the tracking error as much as possible (low steady-state error). It is also desirable that the algorithm reduces the error as fast as possible (high speed of convergence; good transient behavior). Furthermore, the algorithm performance should be robust against nonrepetitive disturbances and model uncertainties. At times, these three design considerations may pose a trade-off. These considerations will be discussed in more detail, but first we will present two frameworks for the analysis of ILC.

### 3.2 ILC analysis frameworks

For analyzing properties of the ILC scheme, there are two main complementary approaches: analysis in the lifted domain, and in the frequency domain. In the lifted domain
framework, the system signals of the finite-time task are sampled at the chosen sample interval, so that the analysis deals with finite-length discrete-time signals. These finite-length discrete-time signals are stacked into vectors where each element is a sampled time instant of the signal. Analysis is carried out in terms of vectors of signals and matrices of Markov parameters. The analysis of an ILC system in the lifted domain may be thought of as analysis of a discrete state-space system, where the ILC iteration index becomes the state-space system time index, and the ILC system’s time-sampled error signal becomes the state-space system’s states. On the other hand, in frequency domain analysis, the task length is assumed to be infinite so that the Z-transform may be applied. Analysis is carried out in terms of Z-domain transfer functions.

Later, we will use both frameworks to analyze important properties of ILC systems including convergence, steady-state error, and transient behavior.

3.2.1 Lifted domain framework

Consider the ILC system given by Equations 3.2, 3.1, and 3.3. In the ILC system output equation (Equation 3.2), the rational function operators may be alternately represented (by dividing the numerator by the denominator) as an infinite series of Markov parameters. For example, we have

\[ T_u(q) = t_{u1}(q^{-1}) + t_{u2}(q^{-2}) + t_{u3}(q^{-3}) + \ldots \]

Here, we have assumed that the delay of the system \( T_r \) is 1, so that \( t_{u1} \) is the first non-zero Markov parameter. Similar holds for \( T_r \) and \( T_d \). Also recall that the system is executing a finite time-length task with signals sampled at even time intervals, so that we have \( N \) samples of all signals. Stacking the samples into vectors of length \( N \), and arranging the Markov parameters into matrices, Equation 3.2 can be written as

\[
\begin{bmatrix}
  y_j(1) \\
  y_j(2) \\
  \vdots \\
  y_j(N)
\end{bmatrix}
= \begin{bmatrix}
  t_{r1} & 0 & \ldots & 0 \\
  t_{r2} & t_{r1} & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  t_{r,N} & t_{r,N-1} & \ldots & t_{r1}
\end{bmatrix}
\begin{bmatrix}
  r(0) \\
  r(1) \\
  \vdots \\
  r(N-1)
\end{bmatrix}
+ \begin{bmatrix}
  0 & \ldots & 0 & 0 \\
  t_{u1} & t_{u1} & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  t_{u,N} & t_{u,N-1} & \ldots & t_{u1}
\end{bmatrix}
\begin{bmatrix}
  u_{jILC}^r(0) \\
  u_{jILC}^r(1) \\
  \vdots \\
  u_{jILC}^r(N-1)
\end{bmatrix}
+ \begin{bmatrix}
  0 & \ldots & 0 & 0 \\
  t_{d1} & t_{d1} & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  t_{d,N} & t_{d,N-1} & \ldots & t_{d1}
\end{bmatrix}
\begin{bmatrix}
  d_j(0) \\
  d_j(1) \\
  \vdots \\
  d_j(N-1)
\end{bmatrix}
\] (3.4)
Note that the signal $u^{ILC}$ runs from sample index 0 to $N - 1$, while $y$ runs from 1 to $N$; this is because of the assumed one-step delay. Then, defining the lifted signal vectors as

$$u_j^{ILC} := \begin{bmatrix} u_j^{ILC}(0) & u_j^{ILC}(1) & \cdots & u_j^{ILC}(N - 1) \end{bmatrix}^T$$

and similarly for the other signals $e_j(k)$, $d_j(k)$, $r(k)$, $y_j(k)$, and defining the lifted system matrices $T_r$, $T_u$, and $T_d$ appropriately, the above equation may be rewritten as

$$y_j = T_r r_j + T_u u_j^{ILC} + T_d d_j$$

(3.5)

and Equation 3.1 may be rewritten as

$$e_j = r - y_j.$$  (3.6)

Equations 3.5 and 3.6 are known as the ILC system lifted representation.

Similarly, the learning algorithm (Equation 3.3) can also be written in the lifted representation as

$$u_{j+1} = Q(u_j + L e_j).$$  (3.7)

In the Toeplitz matrices $Q$ and $L$, the elements below the diagonal correspond to the causal part of the filter, and those above to the noncausal part. Note that it is possible to implement filters with a noncausal part for $Q$ and $L$ because the data for the entire signals of $u_j$ and $e_j$ are available at the end of iteration $j$. On the other hand, the physical system is strictly causal so the matrices $T_r$, $T_u$ and $T_d$ have all zeros above the diagonal.

### 3.2.2 Frequency domain framework

Alternatively, ILC representation and analysis is also possible through the frequency domain (z domain) framework.
Consider the ILC system of Equations 3.2 and 3.1. If it is assumed that the signal lengths are infinite \((N = \infty)\) (or treated as periodic), then the z-transform can be applied to these equations to yield

\[
Y_j(z) = T_r(z)R(z) + T_u(z)U^{ILC}_j(z) + T_d(z)D_j(z) \tag{3.8}
\]

\[
E_j(z) = R(z) - Y_j(z). \tag{3.9}
\]

Similarly, the ILC update equation 3.3 may be represented in the z-domain as

\[
U^{ILC}_{j+1}(z) = Q(z) \left( U^{ILC}_j(z) + z^m L(z) E(z) \right). \tag{3.10}
\]

Together, the above three equations are known as the z-domain ILC representation and are used for frequency-domain analysis.

Although the assumption about infinite-length time is not accurate, the approximation is reasonable if the signal length is long compared to system time constants.

### 3.3 ILC Analysis

#### 3.3.1 Stability Analysis

Once important concern when designing an ILC system is whether or not the learning algorithm will converge. Since the ILC control input is computed recursively, care must be taken to ensure that the ILC control input (and all other system signals) do not grow uncontrollably over the learning iterations. In other words, it is desired that the ILC signals be bounded or converge. In this section, the definition for ILC stability is introduced, along with a condition for stability.

**Definition 1. Asymptotic stability for ILC**

The ILC system in Eqs. 3.2, 3.1, and 3.3 is asymptotically stable if \(\exists \bar{u} \in \mathbb{R} \) such that

\[
|u^{ILC}_j(k)| \leq \bar{u} \quad \forall k \in 0...N - 1, j \in \mathbb{N}
\]

and

\[
\exists u^{ILC}_\infty \in \mathbb{R}^N \quad \text{such that} \quad u^{ILC}_\infty(k) = \lim_{j \to \infty} u^{ILC}_j(k) \quad \forall k \in 0...N - 1.
\]

**Theorem 1. Condition for Stability (Lifted Domain)**

The ILC system in Eqs. 3.3 and 3.2 is asymptotically stable if and only if

\[
\rho(Q(I_N - LT_u)) < 1
\]

Here \(\rho(\cdot)\) represents the spectral radius of a matrix.
Proof. From Eqs. 3.7, 3.6, and 3.5,
\[
\begin{align*}
u_{j+1}^{ILC} &= Q(u_j^{ILC} + L e_j) \\
&= Q(u_j^{ILC} + L(r - y_j)) \\
&= Q(u_j^{ILC} + L(r - (T_r r + T_u u_j^{ILC} + T_d d_j))) \\
&= Q(I - LT_u)u_j^{ILC} + QL(I - T_r)r - QLT_d d_j
\end{align*}
\]

The above system is asymptotically stable if and only if
\[
\max_i |Q(I_N - LT_u)| < 1.
\]

Theorem 2. Condition for Stability (z Domain)
The ILC system in Eqs. 3.8, 3.9 and 3.10 is asymptotically stable if
\[
\|Q(z)(1 - z^m L(z)T_u(z))\|_{\infty} < 1
\]

(Note this is is a sufficient condition only, not an equivalent condition like in the lifted domain case). In the above, \(\|T(z)\|_{\infty}\) is defined as \(\sup_{\omega \in [0, 2\pi]} |T(e^{j\omega})|\).

Proof. From above,
\[
\begin{align*}
U_{j+1}^{ILC}(z) &= Q(z)(U_j^{ILC}(z) + z^m L(z)E(z)) \\
&= Q(z)(U_j^{ILC}(z) + z^m L(z)(R(z) - Y_j(z))) \\
&= Q(z)(U_j^{ILC}(z) + z^m L(z)\left(\left(1 - T_r(z)R(z) + T_u(z)U_j^{ILC}(z) + T_d(z)D(z)\right)\right)) \\
&= Q(z)(1 - z^m L(z)T_u(z))U_j^{ILC}(z) + Q(z)z^m L(z)(1 - T_r(z))R(z) + \ldots \\
&\quad \ldots Q(z)z^m L(z)T_d(z)D(z)
\end{align*}
\]

A sufficient condition for the convergence of \(U^{ILC}\) in the above equation is
\[
\sup_{\omega \in [0, 2\pi]} |Q(e^{j\omega})(1 - L(e^{j\omega})T(e^{j\omega}))| < 1
\]

3.3.2 Asymptotic Performance Analysis

Earlier, it was stated that the goal of ILC is to reduce the tracking error as much as possible. Therefore, one way to characterize the performance of the ILC system is with the converged value of the tracking error. In this section, we will present formulae for the asymptotic error, and comment on the effect of repetitive and nonrepetitive disturbances on
this value, as well as comment on ILC scheme design considerations to make this value as low as possible.

Theorem 3. **Asymptotic error in lifted-domain** If the ILC system in Eqs. 3.7, 3.6, and 3.5 is asymptotically stable, then the asymptotic error is

$$e_{\infty} = (I - T_u(I - Q(I - LT_u))^{-1}QL)(I - T_r)r - T_d d$$

Proof. It can be shown that $e_{\infty} = (I - Q(I - LT_u))^{-1}QL((I - T_r)r - T_d d)$. Then

$$e_{\infty} = (1 - Q(z)) R(z) - T_d(z) D(z).$$

Theorem 4. **Asymptotic error in frequency-domain** If the ILC system in Eqs. 3.10 and 3.2 is asymptotically stable, then the asymptotic error is

$$E_{\infty} = \frac{1 - Q(z)}{1 - Q(z)(1 - z^m L(z)T_u(z))} ((1 - T_r(z)) R(z) - T_d(z) D(z)).$$

Proof. First, form the recursive error equation:

$$E_{j+1} = R - Y_{j+1}$$

$$= R - T_r R - T_u U_{j+1} - T_d D_{j+1}$$

$$= (1 - T_r) R - Q T_u U_j - z^m T_u Q L E_j - T_d D_{j+1}$$

$$= (1 - T_r) R - Q (Y_j - T_r R - T_d D_j) - z^m T_u Q L E_j - T_d D_{j+1}$$

$$= (1 - T_r) R + Q (I - Y_j) - z^m T_u Q L E_j + Q T_d D_j - T_d D_{j+1}$$

Then to find asymptotic error, set $E_j = E_{j+1} = E_{\infty}$ and solve. Also, assume disturbance is the same so $D_j = D_{j+1} = D$.

$$E_{\infty} = (1 - T_r)(1 - Q) R + Q (I - z^m L T_u) E_{\infty} + (Q - 1) T_d D$$

$$= \frac{1 - Q}{1 - Q(1 - z^m L T_u)} ((1 - T_r) R - T_d D)$$
Based on the formula for asymptotic error above, we can comment on the contributions of the reference trajectory and the disturbance on the final converged error. First, ignoring the disturbance, and noticing that $E_o := (1 - T_r)R$ is the tracking error in the zero iteration before learning, we see that

$$E_\infty = \frac{1 - Q}{1 - Q(1 - z^m LT_u)} E_o.$$  

This is the ratio of reduction of tracking error from the first iteration. From this equation, it is seen that a necessary and sufficient condition for zero tracking after learning (assuming that neither $T_u$ or $L$ is zero) is

$$Q = 1.$$  

We can also conclude that $Q$ must be close to 1 in frequency bands where it is desired to decrease $E_o$.

Next, let us examine the effect of repetitive disturbance on the asymptotic error. Ignoring the effect of the reference, the asymptotic error is

$$E_\infty = \frac{1 - Q}{1 - Q(1 - z^m LT_u)} (-T_d D).$$  

In mechanical systems, $T_d$ is typically a lowpass filter, so in order to decrease disturbance contribution across all frequency bands, the bandwidth of the lowpass $Q$ filter should be higher than $T_d$.

### 3.3.3 Transient Performance Analysis

Earlier, the asymptotic properties (stability and asymptotic performance) of ILC were discussed. However, it may be argued that transient performance of ILC is just as important. Transient performance of ILC concerns the growth or reduction in ILC error in the first few iterations, whereas ILC stability is concerned with the ILC error as the iteration number goes to infinity. It is possible that although an ILC algorithm is ultimately stable, the error grows in the first several iterations before finally converging to the asymptotic value. Such growth is undesirable, because the initial divergence of the ILC input may cause physical damage to the system and also may be confused with instability. Here, we introduce a condition for the monotonic convergence of the ILC scheme, in other words, a condition so that the error is sure to decrease in each iteration.

**Definition 2. Monotonic Convergence in Frequency Domain** Consider the ILC system in Eqs. 3.10 and 3.8, 3.9. The system is said to be monotonically convergent with convergence rate $\gamma_1$ if

$$\|E_{j+1}(z) - E_\infty(z)\|_\infty < \gamma_1 \|E_j(z) - E_\infty(z)\|_\infty \quad \forall j \in 0, 1, \ldots$$
Theorem 5. **Condition for Monotonic Convergence in the Frequency Domain**

The ILC system is monotonically convergent with rate $\gamma_1$ if

$$\gamma_1 := \|Q(z)(1 - z^m L(z))T_u(z)\|_\infty < 1.$$ 

**Proof.**

$$\|E_\infty - E_{j+1}\|_\infty = \left\| \frac{1 - Q}{1 - Q(1 - z^m L T_u)} (1 - T_r) R - (1 - T_r)(1 - Q) R - Q(1 - z^m L T_u) E_j \right\|_\infty$$

$$= \left\| \frac{Q(1 - z^m L T_u)}{1 - Q(1 - z^m L T_u)} (1 - Q)(1 - T_r) R - Q(1 - z^m L T_u) E_j \right\|_\infty$$

$$\leq \|Q(1 - z^m L T_u)\|_\infty \left\| \frac{1 - Q}{1 - Q(1 - z^m L T_u)} (1 - T_r) R - E_j \right\|_\infty$$

$$= \|Q(1 - L T_u)\|_\infty \|E_\infty - E_j\|_\infty$$

Applying the definition, the above is monotonically convergent if and only if $\|Q(1 - L T_u)\|_\infty < 1$. \qed

Definition 3. **Monotonic Convergence in Lifted Domain** Consider the ILC system in Eqs. 3.3 and 3.2, 3.1. The system is said to be monotonically convergent if

$$\|e_{j+1}(k) - e_\infty(k)\| < \|e_j(k) - e_\infty(k)\| \quad \forall j \in 0, 1, \ldots$$

Theorem 6. **Condition for Monotonic Convergence in the Lifted Domain** The ILC system is monotonically convergent in the Euclidean norm if

$$\tilde{\sigma}(T_u Q(1 - L T_u) T_u^{-1}) < 1.$$ 

where $\tilde{\sigma}(A)$ is the maximum singular value of matrix $A$.

### 3.3.4 Robustness Analysis

Another consideration in the analysis of ILC systems is whether the ILC system is robust. In other words, it is desirable that ILC stability, good transient performance, and good steady-state error may all be achieved even under some degree of plant model uncertainty, and also nonrepetitive disturbances (such as measurement noise). First define the uncertainty model as

$$T_u(q) = \hat{T}_u(q)(1 + W(q)\Delta(q))$$

where $\hat{T}_u(q)$ is the nominal system model, $W(q)$ is known, stable, and typically highpass, and $\Delta(q)$ is unknown but stable with $\|\Delta(z)\|_\infty < 1$. 

Theorem 7. **Robustness of Monotonic Convergence in Frequency Domain** The ILC system is robustly monotonically convergent if

\[
|W(z)|_{z = e^{i\theta}} \leq \frac{1 - |Q(z)(1 - z^m L(z) T_u(z))|}{|Q(z)z^m L(z) T_u(z)|} \bigg|_{z = e^{i\theta}} \forall \theta \in [-\pi, \pi]
\]

Then, the Q-filter lowpass filter cutoff must be lowered to increase the robustness. Thus, there is a tradeoff between asymptotic error, which requires a high Q filter cutoff, and robustness, which requires a low cutoff.

### 3.4 ILC design methods

There are several methods for ILC scheme design. The difference in the methods is in the choice of the learning filter \( L \) and Q-filter \( Q \) of Equation 3.3.

#### 3.4.1 PD-type ILC

PD-type ILC is the simplest form of ILC. In PD-ILC, the learning signal is the sum of a proportional error term and difference error term, so that the update law may be written

\[
u_{j+1}^{ILC}(k) = Q(q)(u_j^{ILC}(k) + \alpha e_j(k + 1) + \beta(e_j(k + 1) - e_j(k))).
\]  

Advantages of PD-type ILC are that it is easily tunable through the parameters \( \alpha \) and \( \beta \), and that it does not require a plant model.

A subset of PD-type ILC is P-type ILC, where \( \beta \) is set to zero. Then, the frequency-domain ILC monotonic convergence condition becomes

\[
\|Q(z)(1 - z\alpha T_u)\|_{\infty} < 1.
\]

In the above formula, \( \alpha \) is the tuning parameter which influences mostly the rate of convergence, and \( Q \) influences the converged error value. For mechanical systems stabilized in a feedback loop, normally \( 0 < \alpha < 1 \) gives good performance. For such systems, typically the frequency response of \( T_u \) has a lowpass filter. Then, if \( \alpha = 1 \) is used, the left side of the above inequality automatically becomes small at low frequencies, implying that transient performance is good at low frequencies. Also, if the Q filter bandwidth is made low, then the inequality is small at high frequencies as well. Thus, for many mechanical systems, even the simple P-type ILC scheme can achieve fast convergence and low asymptotic error.

#### 3.4.2 Inverse model ILC

Inverse model ILC uses a model of the inverse plant in the update equation. The ILC update law is

\[
u_{j+1}^{ILC}(k) = Q(q)(u_j^{ILC}(k) + \hat{T}_{u}^{-1} e_j(k + 1))
\]
where $\hat{T}_u^{-1}$ is the nominal plant model inverse. Theoretically, if the nominal system model used in the above equation is equal to the real system, then the learning converges in only one iteration, and furthermore, without a Q-filter, it converges to zero tracking error. Compared with P-type ILC, the learning is faster, but it is less robust than P-type, mainly because $\hat{T}_u^{-1}$ typically has high gain at high frequencies.

### 3.4.3 Quadratic Optimal ILC

In quadratic optimal ILC (Q-ILC), the $Q$ and $L$ filters are determined by minimizing a quadratic cost function of the error for the next iteration. If the following quadratic cost function is defined,

$$J_{j+1}(u_{j+1}^{ILC}) = e_{j+1}^T Q e_{j+1} + (u_{j+1}^{ILC})^T R u_{j+1}^{ILC} + (u_{j+1}^{ILC} - u_j^{ILC})^T S (u_{j+1}^{ILC} - u_j^{ILC})$$

where $Q$ is a positive definite matrix and $R$ and $S$ are positive semidefinite, then it can be easily calculated that the optimal ILC control effort for the next iteration, $u_{j+1}$, is a linear combination of the input and error in the current iteration, or

$$u_{j+1}^{ILC*} = Q^*(u_j^{ILC*} + L^* e_j)$$

where the optimal $Q$ filter and learning filters are given by

$$Q^* = (T_u^T Q T_u + S + R)^{-1} (T_u^T Q T_u + S)$$

$$L^* = (T_u^T Q T_u + S)^{-1} T_u^T Q.$$

### 3.5 Summary

In this chapter, we gave a basic introduction to ILC as a method to iteratively improve the performance of systems executing a repetitive process. We presented two frameworks for representing and analyzing ILC, the lifted framework and frequency-domain framework. The lifted framework is attractive because analysis is done with finite-length vectors, and the stability and performance conditions are less strong compared to the frequency domain analysis. The lifted framework also easily handles LTV system analysis. However, if the task length is long, then the computational burden of large matrices becomes too heavy. On the other hand, design in the frequency domain is easy because many familiar frequency-domain ideas can be used, but it is limited to LTI ILC design.

We also commented on ILC scheme performance considerations, including stability, performance, transient performance, and robustness. Through the analysis, it was found that there is a tradeoff between several of these characteristics, such as transient performance vs. asymptotic performance, transient performance vs. robustness, and robustness to nonrepetitive disturbances. In general, it may be said that aggressive learning (high-gain learning...
filter) may have fast learning convergence, but poor robustness to plant variations or non-repetitive disturbances, and the opposite is true of cautious, slower learning.

There are many research issues remaining for ILC, including: understanding ILC performance and robustness tradeoff, and generalizing ILC results to different trajectories. It is hoped that the contents of this chapter serve as a good groundwork for understanding the remaining chapters of this thesis, in which we tackle some of these.
Chapter 4

ILC design for vibration reduction

In the previous chapter, ILC was introduced as a method for improving tracking error in systems repeatedly performing the same task. It was seen that ILC is effective for reducing errors caused by the trajectory shape and also external disturbances that are the same from iteration to iteration. In this chapter, we will shift the focus to designing ILC for systems affected by vibrations.

In the wafer stage system, there are several sources of vibrations and disturbances. Based on experience, we are aware of two main vibrations affecting the wafer stage system: one occurring at 20 Hz, and another at 150 Hz (see Figure 4.1). The 20 Hz vibration can be seen at the beginning and end in the tracking error for the scan trajectory, and has a different phase for each run. It is thought to come from an external source (such as table vibration). The 150 Hz vibration, which occurs during the stage acceleration, is unchanging from run to run, and thought to come from unmodeled dynamics somewhere in the stage structure (specifically, from the adjustable mirror mount, which uses springs), since it only appears during stage acceleration. In addition to sinusoidal vibrations, there are also other non-sinusoidal disturbances affecting the wafer stage, such as force ripple, cable tug forces, and measurement noise.

The disturbances and vibrations just described can be categorized in terms of low frequency vs. high frequency, and repetitive in each run vs. nonrepetitive, as summarized in Table 4.1. First, we consider ILC performance on repetitive vs. nonrepetitive disturbances. ILC is highly effective to reduce errors from disturbances that are repetitive from iteration to iteration. However, ILC is not as effective against disturbances which are different in each iteration. One example of such a disturbance is an external vibration; although the frequency might be the same between iterations, the phase would change. If the ILC scheme is not designed with care, using ILC on a system with significant nonrepetitive disturbances may hurt rather than help performance. In some cases, the nonrepetitive disturbances can be separated from the repetitive disturbances in frequency band, or by time interval of occurrence.

Next, we consider ILC performance for high frequency vs. low frequency disturbances,
Figure 4.1: Tracking error from two runs. The 150 Hz vibration is the same between runs, while the phase of the 20 Hz vibration is different.

given that the disturbance is repetitive. ILC is highly effective for compensating for low frequency disturbances, but it is more difficult to have improvement against high frequency disturbances. Typically, the closed loop transfer function is well known at low frequencies, and more uncertain at high frequencies. This makes it difficult to design ILC algorithms which are stable and have good transient convergence and asymptotic performance at high frequencies. A common approach is to use the Q-filter bandwidth to turn off learning at high frequencies, but the drawback is that no error reduction in the high frequency region is possible. If the Q filter bandwidth is increased past the disturbance frequency, but ILC wasn’t designed well, then it is possible that large error growth will occur due to a bad learning transient.

In light of our final goal of improving tracking error as much as possible, we will investigate methods of ILC design to reduce tracking error caused by the disturbances and vibrations discussed before. Our proposed method is to use a disturbance observer together with a well-designed ILC. The disturbance observer is mainly for reducing the error caused by nonrepetitive disturbances in low frequencies (since ILC is capable of handling well the repetitive disturbances of low frequency). A disturbance observer is a model-based observer for estimating disturbance and is implemented in the feedback loop. Because it is imple-
Table 4.1: Disturbance and vibration categorization for the wafer stage system

<table>
<thead>
<tr>
<th>Range</th>
<th>Repetitive</th>
<th>Non-repetitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low frequency</td>
<td>Force ripple (&lt; 20Hz)</td>
<td>Table vibration (20 Hz)</td>
</tr>
<tr>
<td>High frequency</td>
<td>Vibration modes in plant dynamics (150 Hz)</td>
<td>Sensor noise, other (? Hz)</td>
</tr>
</tbody>
</table>

mented in the feedback loop, a disturbance observer is capable to handle disturbances that are both repetitive and non-repetitive from run to run. We will use a disturbance observer to compensate for the 20 Hz vibration, while ILC is used for 150 Hz.

As mentioned above, the second component of the proposed method is to use a well-designed ILC, especially for reducing the vibration error (which is repetitive) at high frequencies. As we will demonstrate later, we found that a basic P-type ILC is inadequate to reduce the 150 Hz vibrations. To reduce 150 Hz vibrations, the Q filter bandwidth must be set higher than this frequency, but this resulted in the vibration becoming worse, because the ILC was either unstable or not monotonically convergent at this frequency. We can avoid this transient growth by tuning the Q filter cutoff to be lower than 150 Hz, but then it is not possible for ILC to reduce the error at this frequency. Therefore, we found that the ILC learning filter L and Q filter must be designed intelligently if we are to reduce this vibration, or even just to avoid amplifying it.

Several methods for ILC filter design for reducing vibrations in the high frequency region are proposed and compared. The P-type ILC with low-cutoff and high-cutoff Q filters are tested as a starting point. From here, we tested a Q filter with a notch at the vibration frequency, and also an LTV version of the notch [3] [5] [4]. We also tested changing the L filter shape, by augmenting with a notch filter, by using a time-varying filter [36] or by using a second-order transfer function for L. Finally, we also compared the performance with that of an inverse-model ILC design. Throughout this section, the focus is on heuristic methods of ILC design, because we assume that no accurate model of the plant up to high frequencies is available. The heuristic designs can be tuned based on observations from the time-domain data, such as error vibration frequency and damping ratio. The results will be first demonstrated through simulations in which a fictitious vibration model is used, just to demonstrate the principles. Then, the effectiveness of the methods on the physical test system are demonstrated.
4.1 Suppression of non-repetitive disturbances at low frequency

As stated earlier, ILC cannot be used to reduce the effect of nonrepetitive disturbances; therefore, a different control strategy is needed. We introduce a disturbance observer scheme, and present experimental results showing the reduction of both repetitive and nonrepetitive disturbances at low frequency.

4.1.1 Disturbance observer

The basic idea of a disturbance observer is to estimate an unknown disturbance through plant input and output data and a plant model, and then to compensate for the disturbance by subtracting the disturbance estimate from the control input. In this way, the disturbance’s effect is negated.

Consider the system shown in Figure 4.2. The system is affected by a disturbance $d$. The output of the plant becomes

$$Y(s) = P(s)[U(s) + D(s)].$$
Therefore, a suitable estimate of the disturbance can be given by

\[ D_{est}(s) = P_n^{-1}(s)Y(s) - U(s). \]

where \( P_n(s) \) is a model of the plant. The disturbance estimate should first be passed through a low-pass filter \( Q(s) \) before injecting back into the feedback loop, for the purpose of robustness against model uncertainty and sensor noise.

\[ D_{est}^*(s) = Q(s) \left[ P_n^{-1}(s)Y(s) - U(s) \right]. \]

Then the disturbance estimate can be subtracted from the feedback controller output before it is applied to the plant. The structure of the DOB is shown in the block diagram in Figure 4.3. With the disturbance observer in place, the closed-loop transfer function of the overall system becomes

\[
Y = \frac{PC}{1 + PC + Q\Delta} R + \frac{P(1 - Q)}{1 + PC + Q\Delta} D
\]

where \( \Delta \) indicates the multiplicative uncertainty of the plant model \( P_n \),

\[ P(s) = P_n(s)(1 + \Delta(s)). \]

It is easily seen that if the plant model \( P_n \) is equal to the plant \( P \) (in otherwords, \( \Delta = 0 \)), then the transfer function from reference to output is unchanged. In addition, if the Q filter is set to 1 across all frequency ranges, then the transfer function from disturbance to output becomes zero, and perfect disturbance rejection is achieved. Realistically, however, there will be mismatch in the plant model especially at high frequencies, so Q must be designed to be a lowpass filter with cutoff below the frequency range of uncertainty. Thus, the product \( Q\Delta \) can be made to be small across all frequencies, so that the transfer function from reference to output remains reasonably unchanged. Also, disturbance rejection will be achieved up to the Q filter cutoff frequency.

The disturbance observer was implemented for the wafer stage system. The inverse plant model was implemented with a first order difference approximation for the derivative in discretizing the plant model. Also, a delay of 1 was added to the \( u \) term to make it realizable. The Q filter was designed to be a simple lowpass filter

\[ Q^{DOB}(s) = \left( \frac{\omega_c}{s + \omega_c} \right)^2 \]

The cutoff frequency of the Q filter was chosen to be \( \omega_c = 2\pi 80 \text{ rad/s} \), which is higher than the disturbance frequency (20 Hz) but lower than plant model uncertainty region (higher than approximately 130 Hz). A second order filter was chosen, resulting in -40 dB/dec rolloff, in order to balance the 40 dB/dec rollup of the inverse plant model. The filter was discretized with the Tustin transformation.
4.1.2 Experimental results

Using a DOB resulted in lower tracking error, especially the part of the error caused by the force ripple disturbance and the 20 Hz vibration, as desired. The results of the disturbance observer are shown in Figures 4.4 and 4.5. The 20 Hz vibration error reduction is visible at the beginning and ends of the trajectory while the stage is at rest (RMS error was reduced 26.3 % during this part of the trajectory), and the force ripple error reduction is visible in the scan parts of the trajectory (RMS error was reduced 28.2 % during this part).

![Figure 4.4](image1.png)

**Figure 4.4:** Tracking error with and without disturbance observer. It is seen that using disturbance observer reduces error.

![Figure 4.5](image2.png)

**Figure 4.5:** Disturbance estimate produced by the disturbance observer.

4.2 Repetitive vibrations at high frequency

Unlike low frequency disturbances, the high frequency vibrations affecting the wafer stage system are more challenging to compensate for. In the remainder of this chapter, the
focus will be on the reduction of tracking error from high frequency vibrations. In order to understand the source of the vibrations, a new plant model, including a vibration model, is proposed. Later, we propose several strategies for ILC design for vibration reduction. The strategies are tested on the experimental system and results are compared.

4.2.1 Vibration model

In Chapter 2, a simple model for the wafer stage system was introduced. This model considered the stage dynamics to behave as a mass with viscous friction system. However, it is known that the actual behavior of the system is more complex than can be captured by a simple second-order transfer function, due to factors such as structural vibration modes and actuator bandwidth. Indeed, the experimental data collected supports this; the tracking error from the simulated system using second-order plant model does not exhibit vibration, while the real system does.

In order to create a more accurate plant model which can reproduce the vibration phenomenon, it was deemed necessary to carry out system identification again. The new Bode plots of frequency response data are shown in Figures 4.6 and 4.7, which are the closed loop frequency response and plant-only frequency response data, respectively. The complex dynamics in mid to high frequency ranges are seen in these plots, unlike the previous identification results in Chapter 2. Next, a model was fitted to the frequency response data in Figure 4.7 of the open-loop plant. The previous simple plant model was modified by multiplying by two notch filters:

\[
P_{\text{vib}}(s) = \frac{k}{ms^2 + bs} \cdot \frac{s^2 + \frac{d_1}{c_1}2\pi \omega_1 s + (2\pi \omega_1)^2}{s^2 + \frac{d_1}{c_1}2\pi \omega_1 s + (2\pi \omega)^2} \cdot \frac{s^2 + \frac{d_2}{c_2}2\pi \omega_2 s + (2\pi \omega_2)^2}{s^2 + \frac{d_2}{c_2}2\pi \omega_2 s + (2\pi \omega_2)^2}
\]  

(4.1)

Here, \( k \), \( m \), and \( b \) are assumed the same as in Chapter 2. The parameters in the notch part were tuned, resulting in \( \omega_1 = 170 \), \( d_1 = 0.8 \), \( c_1 = 10 \), \( \omega_2 = 290 \), \( d_2 = 0.8 \), and \( c_2 = 5 \). The frequency responses of the model are overlayed with the experimental frequency response data in Figures 4.6 and 4.7. Although the model isn’t perfectly accurate, the fitted model still reproduces the vibration behavior of interest.

The closed loop Bode plot of the system with the modified plant model including vibrations is plotted against the Bode plot of the original model for comparison in Figure 4.8.

It is seen that by augmenting the simple plant model in the way shown above, the new model’s output will be closer to the experimental system’s output, in terms of vibrations in the tracking error, as seen in Figure 4.9. Also, the simulated tracking error of the vibration model is compared against the simple model in Figure 4.10, where it is seen that the output of the simple model has no vibrations.

Next, we will show how the presence of the unmodeled vibration dynamics causes problems in the performance of ILC. Since the frequency of the vibration in the error is 150 Hz,
Figure 4.6: Frequency response data of closed loop \( (r \rightarrow y) \) in red overlayed with vibration model closed loop frequency response.
Figure 4.7: Frequency response data of plant only \( (u \rightarrow y) \) in blue overlayed with vibration model frequency response.
Figure 4.8: Comparison of closed loop Bode plots using simple plant model and vibration model.

Figure 4.9: Tracking error of vibration model in simulation vs. experiment data.
a first guess would be to attempt to reduce the tracking error by using a simple P-type ILC scheme with Q filter low pass filter cutoff frequency higher than the error frequency (we have chosen a 250 Hz cutoff frequency). However, after 3 iterations of ILC, the vibration error is made worse, as seen in Figure 4.11. This effect is worse in the experimental system than in simulation, the difference being due to inaccurate modeling. If iterations are carried out further, the vibration error becomes larger before eventually reducing. So, although this ILC system is stable, the transient behavior is unacceptably bad. In Figure 4.12, the tracking error after 3 ILC iterations for the simulated simple model and vibration model are compared. It is seen that simple P-type ILC works well for the nominal plant, but for the case with the modified plant, using ILC amplifies the vibration. The error converges fast and to a low value in the case of the nominal plant, but for the case of modified plant, the error becomes oscillatory.

Recall that in Chapter 3, a sufficient condition for both monotonic and asymptotic convergence was that the magnitude of the transfer function $Q(z)(1 - z^{-m}L(z)T_u(z))$ be less than 1 across all frequencies. The Bode plot of this function is shown in Figure 4.13. From
Figure 4.12: Tracking error after 3 ILC iterations with Q filter with 250 Hz cutoff: vibration model vs. simple model in simulation

inspection, it is seen that the magnitude comes very close to 1 at the frequency of 150 Hz, while the simple plant model ILC stability transfer function is small across all frequencies.

Figure 4.13: Bode plot of ILC stability transfer function for simple and vibration models.

It may be noted that the problem of vibration amplification can be avoided by lowering the Q filter cutoff to below the frequency of vibration, but the best tracking error can not be achieved if this is done. In Figures 4.14 and 4.15, the tracking error after 3 ILC iterations where the Q filter cutoff frequency is 100 Hz is shown. It is seen in both cases that the tracking error during the acceleration part still remains high.
4.3 Proposed ILC schemes for vibration reduction

As brought to attention in the previous section, a P-type ILC scheme with a low pass filter for Q filter has difficulty in treating the high frequency vibrations present in the wafer stage test system. In this section, we investigate ILC design methods for reduction of these high frequency vibrations. The designed ILC scheme should lower tracking error as much as possible, with particular focus placed on reducing vibrations excited during the stage acceleration. To this end, we will investigate and compare several types of ILC design to be described in detail in the following section. We define 8 different ILC schemes for comparison, listed in Table 4.2 for reference. The methods are tested on both the vibration model in simulation and the experimental wafer stage test system. Finally, effectiveness and implementation complexity are discussed.

The emphasis here is placed on heuristic methods of ILC design. One of the main advantages of ILC is that it allows achieving a very high degree of tracking precision through repeated trials, rather than through accurate modeling. We assume that an accurate plant model is unavailable, especially in the high frequency regions of the vibration in question. Therefore, it is desirable that the ILC may be designed easily based on insights easily obtained from time plots of tracking error data, such as vibration frequency and decay rate.
Table 4.2: Tested ILC Types

<table>
<thead>
<tr>
<th>L filter</th>
<th>Q filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. scalar</td>
<td>low pass 100 Hz cutoff</td>
</tr>
<tr>
<td>2. scalar</td>
<td>low pass 250 Hz cutoff</td>
</tr>
<tr>
<td>3. scalar</td>
<td>low pass 250 Hz cutoff with 150 Hz notch filter</td>
</tr>
<tr>
<td>4. scalar</td>
<td>low pass 250 Hz cutoff with 150 Hz dynamic notch</td>
</tr>
<tr>
<td>5. 150 Hz notch filter</td>
<td>low pass 250 Hz cutoff</td>
</tr>
<tr>
<td>6. 150 Hz dynamic notch filter</td>
<td>low pass 250 Hz cutoff</td>
</tr>
<tr>
<td>7. frequency shaped</td>
<td>low pass 250 Hz cutoff</td>
</tr>
<tr>
<td>8. model inverse</td>
<td>low pass 250 Hz cutoff</td>
</tr>
</tbody>
</table>

Figure 4.16: Tracking error comparison of ILC types for vibration – simulation reduction
Figure 4.17: Tracking error comparison of ILC types for vibration reduction – simulation – magnification of acceleration part in Figure 4.16

Figure 4.18: Peak error vs. ILC iteration number – comparison of ILC types
The first ILC scheme in consideration is P-type ILC with a scalar learning filter and low-pass Q filter. This ILC is included as a benchmark for comparison of the other types. We have chosen $L = 1$ and $Q = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}$, where $a_1 = \frac{-2 + \omega_c T_s}{2 + \omega_c T_s}$, $b_0 = b_1 = \frac{\omega_c T_s}{2 + \omega_c T_s}$, and $\omega_c = 2\pi 100$, which is a zero-phase low-pass filter with 100 Hz cutoff.

The performance of this ILC scheme is shown in Figures 4.16 and 4.17, compared against the other ILC types to be discussed next. As expected, the peak error of this ILC type is the highest of all the schemes considered, due to the low cutoff of the Q filter.

4.3.1 ILC 2: P-type with 250 Hz cutoff Q filter

The second tested ILC scheme is the same as the first, except that the Q filter bandwidth has been increased to 250 Hz, which is higher than the vibration frequency. This ILC scheme causes large transient error growth, in the form of magnified vibration. The length of the vibration tail (time it takes for vibration to decay after the acceleration) becomes long. The effect is much worse in the experimental setup than can be seen in this simulation graph, as will be shown in the next section.
4.3.2 ILC 3: P-type with notch Q filter

Based on the observation that the vibration is amplified using ILC 2, a zero-phase notch filter

\[ Q_n = \frac{s^2 + \frac{d}{2} 2\pi \omega_n s + (2\pi \omega_n)^2}{s^2 + \frac{1}{c^2} 2\pi \omega_n s + (2\pi \omega_n)^2} \cdot \frac{(-s)^2 + \frac{d}{2} 2\pi \omega_n (-s) + (2\pi \omega_n)^2}{(-s)^2 + \frac{1}{c^2} 2\pi \omega_n (-s) + (2\pi \omega_n)^2} \]

with notch at the vibration frequency was added to the Q filter. For digital implementation, the causal part (first term only) was discretized using a prewarped Tustin transformation; then the discrete signal was filtered through this filter, time-reversed before filtering again, and finally time-reversed again. This results in a zero-phase filtering operation.

As seen in Figure 4.16, this had the desired effect of making the transient vibration decay faster.

It must be noted that the performance of the notch filter depends greatly on the tuning of the parameters \( d, c, \) and \( \omega_n \).

4.3.3 ILC 4: P-type with dynamic notch Q filter

It was noticed that adding the notch filter to the Q filter as in ILC 3 had the undesirable effect of increasing the peak error during acceleration larger than in ILC 2. Therefore, we proposed to turn the notch filter off during the acceleration, and turn it on only when needed, in other words immediately after the acceleration phase, in order to damp vibrations. As seen in Figure 4.16, and comparing ILC 4 with ILC 3, this approach was effective.

4.3.4 ILC 5: Notch L filter

Since vibrations arise from complex and unknown plant dynamics rather than an exogenous disturbance source, adding a notch filter to the L filter seems logical. The filter parameters were tuned to get the best performance. Results are shown in Figure 4.17. ILC 5 has better performance than ILC 3.

4.3.5 ILC 6: Dynamic notch L filter

Similar to using an switched on/off notch filter in the Q filter (ILC 4), the same idea was applied to the L filter. As seen in Figure 4.17, a similar effect was observed: the vibration was suppressed while the peak error in acceleration was reduced.

4.3.6 ILC 7: Frequency-shaped L filter

A frequency-shaped L filter was also tested. This filter is given by

\[ L(s) = \left( \frac{(2\pi \omega)^2}{s^2 + 2\zeta (2\pi \omega) s + (2\pi \omega)^2} \right)^{-1} \cdot \frac{\omega_c^2}{(s + \omega_c)^2}. \]
It is the inverse of a second-order transfer function, multiplied by a low-pass filter to make the transfer function proper and realizable.

The parameters are picked by inspecting the vibration in the tracking error and calculating the vibration’s natural frequency and damping ratio (by measuring roughly the vibration’s frequency, and relative decay of magnitude of vibration peaks).

The reason that this filter was proposed is that the shape is similar to the notch L filter, except that the phase lag is less at the center frequency. Comparing the performance of the frequency-shaped L filter and notch L filter in 4.17; it is shown that the frequency-shaped L filter reduces tracking error further.

4.3.7 ILC 8: Model-inverse L filter

Lastly, a model-based ILC scheme was tested, where \( L \) is chosen as the inverse of the transfer function \( T_u = \frac{P_{vib}C_{i}}{P_{vib}C_{i} + 1} \), and \( P_{vib} \) is as defined in Eq. 4.1. From inspecting 4.17, it can be seen that the model-inverse ILC design has the best performance of all the schemes.

4.3.8 Discussion

The performance of the proposed ILC schemes in terms of a tracking error time-plot was already compared in Figures 4.16 and 4.17. In addition, the reduction of the magnitude of tracking error vs. ILC iteration is plotted in Figures 4.18 and 4.19.

The previous results may be better understood by frequency domain analysis of the ILC scheme. Recall that in Chapter 3, we presented a condition for monotonic convergence of the tracking error in ILC,

\[
\|Q(z)(1 - z^mL(z)T_u(z))\|_\infty < 1.
\]

Also, the smaller the magnitude of this transfer function, the faster the rate of convergence. The Bode plot of this transfer function for each of the discussed ILC types (except for the time-varying types) are shown in Figure 4.20. The P-type ILC with 100 Hz cutoff Q filter has a relatively high magnitude and thus has slow convergence. Also, the P-type ILC with 250 Hz cutoff Q filter has a magnitude that reaches very close to 1 at the 150 Hz frequency, which is undesirable. On the other hand, as was expected, the model inverse ILC scheme has the lowest magnitude across a wide frequency range, and resulted in the fastest convergence. Also, the frequency-shaped L filter ILC scheme has a low magnitude especially in the frequency region near 150 Hz, and as a result, had fast convergence.

Similarly, we may also consider the asymptotic error in frequency analysis. Previously in Chapter 3, it was shown that the asymptotic error is given by

\[
E_\infty = \frac{1 - Q(z)}{1 - Q(z)(1 - z^mL(z)T_u(z))}E_0.
\]
This transfer function tells us how much it is possible to reduce the final tracking error through ILC. It is desirable that the magnitude of this transfer function is low across all frequencies. In Figure 4.21, the Bode plot of this function is shown for all of the discussed ILC types. The inverse-model ILC and frequency-shaped L filter ILC have the lowest magnitudes, implying that they have the highest potential to reduce tracking error. This conclusion is also supported by the ILC simulation results shown earlier.

![Bode Diagram](image)

Figure 4.20: Bode plot of ILC stability and monotonic transfer function

### 4.3.9 Experimental Results

Finally, the proposed ILC schemes will be verified on the experimental setup, followed by a discussion and conclusion. In Figure 4.22, the tracking error after 20 iterations of learning is plotted for each of the types of ILC described in the previous section. The acceleration portion is shown in magnification in Figure 4.23. The experimental results have several important differences from the simulation results, which we will next discuss.

From these results, several observations may be made. ILC 1, P-type ILC with 100 Hz cutoff leaves the highest peak error, the same as in simulation. Increasing the Q-filter cutoff to 250 Hz (ILC 2) reduces the peak error greatly, but causes the 150 Hz vibration to be magnified (after the first acceleration part only). The vibration amplitude dies down with successive learning iterations, but the decay rate was slow. The 150 Hz vibration was amplified greatly in the experimental data, although this problem was not seen as much in the simulation, probably due to inaccurate modeling. Next, the effect of adding a notch filter into the Q filter (ILC 4) can also be seen from these graphs. The Q filter cuts
Figure 4.21: Bode plot of ILC asymptotic error transfer function

Figure 4.22: Comparison of tracking error for vibration compensation ILC schemes – Experimental results. Note that the inverse-model ILC is unstable, and P-type with 250 Hz cutoff has learning transient.
Figure 4.23: Comparison of tracking error for vibration compensation ILC schemes – Experimental results, magnification of acceleration part.
down vibration, but has the undesired effect of increasing peak error during acceleration. Turning the Q filter on and off (ILC 5) during desired portions of the trajectory gives better performance. Similar conclusions apply to adding a notch filter to the learning filter (ILC 6 and 7). However, it is seen that the best performance is achieved by the frequency-shaped learning filter (ILC 7). Lastly, the inverse-model based ILC had problems with 150 Hz vibration amplification, although in simulation it did not. The mismatch must again be due to the modeling inaccuracy.

![Figure 4.24: Comparison of RMS tracking error for different ILC schemes designed for vibration compensation. Tracking error norm reduction over learning iterations is shown. Experimental results.](image)

When evaluating an ILC scheme performance, not only is it important to consider the best achievable tracking error after learning is finished, but it is also important to consider the speed of learning convergence. In Figures 4.24 and 4.25, the magnitude of tracking errors (measured in terms of 2-norm and maximum norm) over learning iteration is compared for each of the proposed ILC types. The frequency-shaped learning has the fastest convergence speed and lowest steady-state tracking error.

Based on the results of the simulation and experimental data, the recommended method of ILC design for vibration reduction is the frequency shaped L filter (ILC 7). In the simulation, it was seen that the model-inverse ILC design (ILC 8) performed the best out of all the tested types. However, obtaining the model required a time-consuming system identification experiment and modeling process. Furthermore, the model-inverse ILC design was not convergent in experiments because of modeling inaccuracy. On the other hand, the frequency-shaped L filter ILC achieved the second-best performance in simulation and
Figure 4.25: Comparison of maximum magnitude of tracking error for different ILC schemes designed for vibration compensation. Tracking error norm reduction over learning iterations is shown. Experimental results.

first-best in experiment. In addition, the order of the L filter is much lower for the frequency-shaped than the model-based (2 compared to 6) making it easier to implement digitally.

4.4 Chapter Summary

We have found that the frequency-shaped L filter performed the best at high-frequency precision tracking and for reducing vibration. The advantages of the methods described above is that enable the design of an ILC scheme for precision tracking up to high frequencies, even into frequency ranges where plant model is uncertain, based on only observations made from experimental data about vibration frequency and damping ratio. Thus, the proposed ILC schemes may be implemented even without a detailed knowledge of the plant model. One of the biggest strengths of using ILC is that it enables achieving high precision tracking for mechanical systems even without an accurate model. The proposed methods preserve this advantage.
Chapter 5

ILC for Scan Trajectories

As demonstrated in the previous section, ILC provides good performance improvement for stable systems executing a repetitive task. In fact, for the wafer stage, we have found that ILC yields superior performance to that of inverse-model based feedforward control, adaptive feedforward controller, DOB, etc. With ILC, we were able to achieve tracking performance up to the limit imposed by sensor noise.

However, one drawback of ILC not applicable to the aforementioned methods is the limitation of an ILC signal to a single trajectory. If the trajectory is changed, the learning iterations must be redone. The problem is illustrated in Figure 5.1. In many manufacturing applications, motion profiles such as point-to-point motion, or scan motions, are very common. Sometimes, engineers might want to modify slightly the trajectory, such as changing the scan velocity, scan length or time, maximum acceleration, etc. If we limit our consideration to trajectories within a particular class, perhaps it is possible to find a way to make one ILC result applicable to all other trajectories in the class.

Figure 5.1: When the reference trajectory changes, the ILC input signal must be modified, through re-learning, or other methods.
In this chapter, we propose a method for generalizing a learned ILC signal for LTI systems to apply to other trajectories for a class of scan trajectories. In particular, we consider a class of time-optimal acceleration-limited scan trajectories (second-order polynomial splines), for example as shown in Figure 2.14. These trajectories consist of an acceleration from rest to a constant scanning velocity, scanning at the constant velocity over a certain distance, and finally a deceleration back to rest. However, the proposed method is easily extendable to third-order or fourth-order trajectories shown in Chapter 2 with small modifications.

In the proposed method, the ILC learning iterations need only be carried out once, with any scan trajectory of the class used as the learning trajectory. Then, the learned ILC signal is used as a base signal to re-calculate a suitable ILC feedforward signal to use with any other new trajectory of the class. The calculated ILC input signal will then result in low tracking error, without the need to repeat learning iterations on the new trajectory. In essence, the proposed method involves first the analysis of a learned ILC input signal to break it down to basic components, followed by the synthesis of a new ILC input signal specifically for the new trajectory, using those basic components. In other words, ILC is used as a feedforward signal training method.

Several researchers have also previously tackled the problem of generalizing ILC input signals to different trajectories. Previous research has included modifying ILC signals for trajectories differing in magnitude and time-scale. In [34] [33], a scheme for time-scale and magnitude-scale adjustment of ILC signals applied to nonlinear systems was presented. In [18], the researchers used ILC to create feedforward signals for an underwater robot performing the same task at different speeds, using a time-scale transformation. This research differs from [18], because not only does the proposed method allow for changes in the time-scale, but also any changes in trajectory shape, such as acceleration limit, scan distance, scan velocity, etc. In [2], ILC was first performed using many different training tasks, and the results stored in a data-bank of learned ILC signals. Then for future runs of new trajectories, the ILC signal for the most similar trajectory found in the databank is used as a starting point. However, the performance of this scheme is limited by the similarity of the new trajectory to those previously stored in the data bank.

In may be noted that the function of the synthesized ILC input trajectory is very similar to the output of a feedforward controller. The proposed method has several advantages, though. The performance of a feedforward controller is limited by knowledge of the inverse plant model, i.e. plant transfer function order, and parameters. However, since the proposed method is based on ILC, it is entirely data-based, with no modeling of the plant or disturbances necessary. Secondly, the proposed method results in a feedforward signal that is rich: The synthesized feedforward signal generated by this method captures higher order dynamics that are hard to do in simple transfer function models, which are used in creating feedforward controllers. Therefore the performance attainable from this method is expected to be better than that possible from most model-based feedforward controllers.
5.1 Method Description

5.1.1 Scan trajectory construction

The construction of a second-order scan trajectory may be understood from looking at Figure 5.2. The trajectory consists of two parts: acceleration parts at the beginning and end, and a constant velocity scan part in the middle. In designing a second-order scan trajectory, the acceleration limit, velocity limit, and scan distance are first specified (the start time ($t_{dead}$) may also be specified if a nonzero start time is desired). Then the time-optimal trajectory which satisfies these limits is uniquely determined, and is given by a polynomial spline function with acceleration time length $t_{acc}$ and scan time length $t_{scan}$. The continuous-time trajectory is first found analytically, and then sampled to get the discrete-time trajectory for digital control.

In the bottom graph of Figure 5.2, notice that the acceleration profile can be thought of
as the superposition of four scaled and time-shifted step functions:

\[ a(t) = \bar{a} \left( u_{\text{step}}(t - t_1) - u_{\text{step}}(t - t_2) - u_{\text{step}}(t - t_3) + u_{\text{step}}(t - t_4) \right) \]

where the time-shift quantities are given by \( t_1 = t_{\text{dead}} \), \( t_2 = t_{\text{dead}} + t_{\text{acc}} \), \( t_3 = t_{\text{dead}} + t_{\text{acc}} + t_{\text{scan}} \), \( t_4 = t_{\text{dead}} + 2t_{\text{acc}} + t_{\text{scan}} \), and the scalar \( \bar{a} \) is the peak acceleration. Here, \( u_{\text{step}} \) denotes the step function. Also notice that the position profile is the double integral of this, or

\[ r(t) = \bar{a} \int_{0}^{t} \int_{0}^{T} (u_{\text{step}}(\tau - t_1) - u_{\text{step}}(\tau - t_2) - u_{\text{step}}(\tau - t_3) + u_{\text{step}}(\tau - t_4)) d\tau dT. \]

When this reference is given to the feedback system of continuous-time LTI plant \( P \) and controller \( C \), and assuming that the system is not affected by disturbances or noise, then the tracking error is

\[ E = \frac{1}{1 + PC} R \]

Since \( \frac{1}{1 + PC} \) is LTI, then it is apparent that the tracking error resulting from any second-order scan trajectory is also a superposition of signals of the form of the step response of \( \frac{1}{1 + PC} \).

### 5.1.2 Synthesis of feedforward signal for new trajectory

Now, consider the converged ILC input signal \( u^{ILC*}(t) \), obtained after many iterations of learning. Since the standard ILC algorithm is linear, the signal \( u^{ILC*}(t) \) will also be a superposition of step responses of some transfer function.

Using the ILC system definitions and notation from Chapter 3, we can easily determine the converged ILC input. Here, we assume the system is subject to no disturbances or noise, and consider only the effect of the reference trajectory on the ILC input. Combining Equations 3.8, 3.9, and 3.10, we obtain the recursive formula for the ILC input

\[ U_{j+1}^{ILC} = Q(U_j^{ILC} + z^m L(R - (T_r R - T_u U_j^{ILC}))). \]

Taking the limit of both sides as \( j \to \infty \) and defining \( U^{ILC*} = \lim_{j \to \infty} U_j^{ILC} \), we obtain

\[ U^{ILC*} = \frac{Qz^m L(1 - T_r)}{1 - Q(1 - z^m L T_u)} R. \]

So, \( u^{ILC*}(t) \) is a superposition of shifted and scaled signals which are the step responses of the transfer function \( T_{ILC} := \bar{a} \frac{Qz^m L (1 - T_r)}{1 - Q(1 - z^m L T_u)}. \) This is expressed by

\[ u^{ILC*}(t) = \bar{a} \left( u(t - t_1) - u(t - t_2) - u(t - t_3) + u(t - t_4) \right) \]

where

\[ u(t) = L^{-1} \left( \frac{1}{s^3} \frac{Qz^m L(1 - T_r)}{1 - Q(1 - z^m L T_u)} \right). \]
Hereafter, we will refer to $\bar{u}(t)$ as the base feedforward signal. It is simple to calculate $\bar{u}(t)$ from $u^{ILC*}(t)$ based on Eq. 5.1. The method will be explained in more detail later.

As a side note, it is worth mentioning that if the ILC input achieves perfect tracking, in other words when $Q = 1$, then

$$U^{ILC*} = \frac{1 - T_r}{T_u} R = \frac{1}{PC} R$$

(compare to Equation 5.1.2), and that then the base ILC signal becomes

$$\bar{u}(t) = \mathcal{L}^{-1} \left( \frac{1}{s^3} \frac{1}{PC} \right). \quad (5.3)$$

(compare to Equation 5.2). Note that it is not intended that $\bar{u}(t)$ be obtained from Equation 5.3. This equation is presented to show the relation of $\bar{u}(t)$ to $P$ and $C$ under the ideal condition that all assumptions at the beginning of this subsection be satisfied. It is better for $\bar{u}(t)$ to be calculated based on data, as detailed in the next section, rather than based on a model as in Equation 5.3, because of model inaccuracy.

Once $\bar{u}(t)$ is obtained, it is simple to use it to construct feedforward signals for other scan trajectories. Consider a second scan trajectory $r'$, which is characterized by acceleration limit $\bar{a}'$ and step times $t'_1,t'_2,t'_3$, and $t'_4$. Then, with referring to Equation 5.1, we can see the feedforward $u^{FF}(t)$ signal given by

$$u^{FF}(t) = \bar{a}' (\bar{u}(t - t'_1) - \bar{u}(t - t'_2) - \bar{u}(t - t'_3) + \bar{u}(t - t'_4)) \quad (5.4)$$

should be the same signal as the converged ILC signal for $r'$. Therefore, applying $u^{FF}(t)$ to the system for trajectory $r'$ should yield the same (low) tracking error as if ILC iterations had been carried out.

In summary, the method involves

1. Design the first scan trajectory $r$.

2. Apply ILC to the system with scan trajectory $r$ until a satisfactory level of tracking accuracy is achieved; save the final ILC input signal, $u^{ILC*}(t)$.

3. From $u^{ILC*}(t)$, extract the base feedforward signal $\bar{u}(t)$.

4. Design the new scan trajectory $r'$.

5. For the new scan trajectory $r'$, calculate the new feedforward signal by Eq. 5.4, and apply to the system.
5.1.3 Method of calculating base feedforward signal

In the previous section, we mentioned that the base feedforward signal is calculated from the converged ILC signal through Eq. 5.1. Here, we describe the methods in more detail. There are several ways to obtain the base feedforward signal $\overline{u}$, depending on how accurate it is desired to be known, and the number of trial runs willing to be carried out.

The base feedforward signal may be calculated recursively in the time intervals between $t_1, t_2, t_3,$ and $t_4$, based on Equation 5.1. First, let $t_Q$ be the expected length of the acausal part of the base feedforward signal; a good way to choose a value of $t_Q$ is to set it equal to the length of the non-zero part of the learned ILC signal before the start of the movement at $t_1$. Also, let $t_f$ be the time-length of the trajectory. Then,

\[
\overline{u}(t) = \frac{1}{\Delta} u^{ILC*}(t + t_1), \quad t \in [-t_Q, t_{acc} - t_Q) \\
\overline{u}(t) = \frac{1}{\Delta} u^{ILC*}(t + t_2) + \overline{u}(t - t_{acc}), \quad t \in [t_{acc} - t_Q, t_{acc} + t_{scan} - t_Q) \\
\overline{u}(t) = \frac{1}{\Delta} u^{ILC*}(t + t_3) + \overline{u}(t - t_{acc}) + \overline{u}(t - t_{acc} - t_{scan}), \quad t \in [t_{acc} + t_{scan} - t_Q, 2t_{acc} + t_{scan} - t_Q) \\
\overline{u}(t) = \frac{1}{\Delta} u^{ILC*}(t + t_4) + \overline{u}(t - t_{acc}) + \overline{u}(t - t_{acc} - t_{scan}) - \overline{u}(t - 2t_{acc} - t_{scan}), \quad t \in [2t_{acc} + t_{scan} - t_Q, t_f - t_1]
\]

This gives us the signal $\overline{u}(t)$ from time $-t_Q$ to $t_f - t_1$. However, because the computation for $\overline{u}(t)$ at later times depends on the result from earlier times, noise is added to itself, and the signal becomes noisier and noisier at later times. Three ways to circumvent this problem are to use a low-pass filter on the estimate of signal $\overline{u}(t)$, or to use curve fitting to fit a smooth curve (such as an exponential) to the tail end of $\overline{u}(t)$, or to average $u^{ILC*}(t)$ from multiple trials. As for curve fitting, the approximation will not pose a large problem, because the most important information of $\overline{u}(t)$ is contained shortly before and after $t = 0$, and this is also where the magnitude of $\overline{u}(t)$ was the biggest for our test system.

5.1.4 Additional Considerations

In actual implementation, there are several complications to the method as presented above.

Disturbances

One major problem is the presence of disturbances. First of all, the presence of disturbances causes problems in identifying $\overline{u}$ from $u^{ILC*}$, because then $u^{ILC*}$ will contain a part that compensates for error caused by the disturbances, as well as the part of error caused by the trajectory. Secondly, $\overline{u}$ as described above cannot reduce the part of the tracking error of the new trajectory caused by the disturbances; it can only be used to compensate for the part of the error due to the reference trajectory.
There are three methods to work around the first problem to identify \( \bar{u} \) more accurately: using a disturbance observer (DOB), using a model-based feedforward compensator for the disturbances, or approximation by curve fitting. If a DOB is used, the effect of the disturbance on the tracking error, and also on the learned ILC signal, will be greatly reduced. This way, the contribution of disturbances to the learned ILC signal is separated from the contribution of the reference trajectory. A second way would be to use a model-based feedforward compensator for the disturbance. Similar to the effect of a DOB, this would remove the effect of the disturbance from the learned ILC signal; however, performance depends on the accuracy of the disturbance model parameters. The third way is to approximate \( \bar{u} \) by a function fitted to the data for \( \bar{u} \), rather than using the data itself. Based on experiential knowledge of the system, we may have an idea of which features of \( \bar{u} \) are important to keep, and which features are due to the disturbances, which we want to smooth out or eliminate.

A future avenue of research is to develop a method to separate the part of the ILC learned signal due to the trajectory from the part due to the disturbances. Such a method would also need to be able to predict how the disturbances themselves or the learned ILC signal changes when the trajectory changes.

**Discrete Implementation**

Another consideration is that when the method is implemented on a discrete system, the acceleration start and stop times which define the scan trajectory, \( t_1, t_2, t_3, \) and \( t_4 \), must coincide with the sampling instances. If the trajectory was designed such that one of these times (especially \( t_1 \) or \( t_2 \)) falls between two sampling instances, then it is difficult to identify \( \bar{u} \) well. The inaccuracy is especially significant during the beginning of \( \bar{u} \), because the slope is very large at first (as illustrated later). An ad-hoc solution would be to use interpolation (linear, or higher order) to estimate the values of \( u^{ILC}_* \) between the sample instances.

**Training Trajectory Selection**

A third consideration is that some trajectories are "easier" to use as training trajectories than others. Trajectories with longer acceleration times \( t_{acc} \) allow the beginning part of \( \bar{u} \) to be identified with more accuracy. It is also best to use trajectories affected by less disturbances, for example trajectories with faster scan velocities may be less affected by nonlinear friction, but if the velocity is too fast, force ripple effect becomes larger. Also, \( \bar{u} \) can only be identified up to time \( t_f - t_1 \), therefore it is better to use trajectories with longer duration as the training trajectory.

**5.2 Simulation**

The method is demonstrated on an example in simulation. First, an ILC feedforward signal for the trajectory (trajectory 1) shown in Figure 5.3 is obtained after 5 iterations
of learning. The resulting ILC signal is shown in Figure 5.4. Then, the basic feedforward signal is extracted, and plotted in Figure 5.5. Note that there is an acausal part to the base feedforward signal; this is due to the forward time shift of the ILC algorithm, along with the zero-phase Q filter. The basic feedforward signal is also compared with the simulated impulse response of $\frac{1}{PC_s}$ in Figure 5.5, and shows close agreement.

Next, we desire to find a suitable feedforward signal to use with a new trajectory (trajectory 2) shown in Figure 5.6. Note that trajectory 2 is longer than trajectory 1; in this case we used extrapolation from curve fitting to extend the length of $\bar{u}(t)$ past the part calculable from data. The basic feedforward signal is used to construct the feedforward signal, and is shown in Figure 5.7. It is plotted against the signal obtained after 5 iterations of learning on trajectory 2, showing close agreement. The tracking error resulting from applying this feedforward to the system is shown in Figure 5.8, and a magnification of the acceleration part in 5.9. In this figure, it is seen that the feedforward signal designed by this method achieves very similar performance to ILC.

![Trajectory 1](image)

Figure 5.3: Trajectory 1

### 5.3 Experiment

The method is also tested on the experimental system. However, in the wafer stage test system, the force ripple disturbance poses a complication. Force ripple error causes a component of the ILC learned signal which is separate from the part due to the reference trajectory. Also, it is difficult to predict how the force ripple shape will change with the stage’s position and velocity, since the available model is not so accurate.

As described earlier, there are three methods (DOB, model-based feedforward disturbance compensator, and curve-fitting) to work around this problem. We used a DOB in combination with curve-fitting. The DOB greatly reduced the effect of the force ripple on the ILC input, but not completely. Later, curve fitting was used to smooth the $\bar{u}$ signal.
Figure 5.4: ILC input for trajectory 1

Figure 5.5: Comparison of ILC basic signal calculated from data and predicted from model impulse response. The signals are almost identical

Figure 5.6: Trajectory 2
Figure 5.7: Calculated and learned ILC input for trajectory 2. The signals are almost identical.

Figure 5.8: Tracking error for trajectory 2. The tracking error for after 5 iterations of ILC and using the synthesized feedforward input are almost identical.
Figure 5.9: Tracking error for trajectory 2, magnification of Fig. 5.8. The tracking error for after 5 iterations of ILC and using the synthesized feedforward input are almost identical.

From experience, we expect that the vibrations in ILC signal due to plant dynamics will die down very fast, we expected that the tail end of $\tilde{u}$ should be smooth, and any ripples would have been caused by the force ripple, so we used an exponential function fit to the data to approximate the tail end (See Figure 5.10).

The experimental results are shown in Figures 5.10 through 5.13. The same trajectory as used in the simulation was used for the feedforward signal learning (shown in Figure 5.3). Figure 5.10 shows the basic feedforward signal which is calculated from the learned ILC signal for trajectory 1. The second scan trajectory is shown in Figure 5.6 (trajectory 2). Figure 5.11 shows the synthesized feedforward signal for trajectory 2, compared with the learned ILC signal, had the ILC learning iterations been carried out. The two signals are very close. This feedforward signal is applied to the system with trajectory 2. Finally, the resulting tracking error is shown in Figure 5.12, a magnification of which is shown in Figure 5.13. It can be seen that the synthesized feedforward input greatly reduces tracking error, and reduces it almost as much as ILC does. The small difference is thought to be due to the force ripple problem, and the DOB not being able to completely compensate for it. Also, in the figure it is shown that the synthesized feedforward input reduces tracking error more than the inverse-model based feedforward controller. In conclusion, the proposed method is an effective feedforward signal generator.

5.4 Chapter Summary

We proposed a method for using ILC as a feedforward signal training method. The method is confined to a class of second-order scan trajectories. In the proposed method, ILC is carried out as normal on a training trajectory from the class. Then using knowledge of the trajectory, a base feedforward signal is identified from the learned ILC signal, and used
Figure 5.10: ILC basic signal

Figure 5.11: Calculated and learned ILC input for trajectory 2. The signals are almost identical.
Figure 5.12: Tracking error for trajectory 2

Figure 5.13: Tracking error for trajectory 2, magnification of Figure 5.12
as a base pattern to synthesize feedforward signals for other trajectories belonging to the same class. The method was successful to reduce tracking errors for other scan trajectories, especially errors during the acceleration phase of scan trajectories. The method performed better than inverse-model based feedforward control, and only slightly worse than ILC.

In this chapter, the method was demonstrated on a class of scan trajectories in particular, but this method may be applied within classes of any kind of trajectories, as long as the trajectories within the class are all constructed from the superposition of time-shifted and scaled base elements.
Chapter 6

Iterative Controller Tuning

Up until now, in this thesis we have been considering the application of ILC to improve the error tracking of precision motion systems. In this chapter, we will shift our focus to a method of iterative controller tuning for systems performing the same task repeatedly. In ILC, the idea is to use repetitive trials of the same task to synthesize a feedforward signal, but the iterative controller tuning fine-tunes controller parameters through repetitive trials.

In this chapter, we introduce the Iterative Feedback Tuning (IFT) method for controller tuning. IFT is a data-driven iterative method for tuning of controller parameters. The controller parameters are tuned in a gradient-based search to minimize a chosen cost function, usually the sum of squared tracking error. The gradient is estimated from system output data (such as tracking error), and the controller parameters are adjusted between runs. IFT can be used to tune both feedforward and feedback controller parameters.

ILC and IFT are similar in that they are both data-based, model-free tuning methods. This makes ILC and IFT easy to apply in practice. In both ILC and IFT, the learning or tuning is done through repeated trials of the same task. On the other hand, there are some important differences between the two. The main motivation for using iterative controller tuning, rather than ILC, or in addition to it, is that the tuned controller may be used with many different reference trajectories (although it is optimal only for the training trajectory), whereas the tuned ILC signal can only be used for one. The ultimate ILC performance and convergence speed depends on the designed algorithm, such as Q and L filters, while the ultimate IFT performance depends on the optimization algorithm step size and search direction, and also on the complexity of controller structure to be tuned.

In this chapter, first the theory of the IFT method is presented. Then we will apply the method to tune the feedforward controller of the wafer stage system, demonstrating through both simulation and experiment. Next, we will introduce the force ripple disturbance, propose a force ripple feedforward compensator structure, and apply IFT to tune the controller parameters. The effect of the force ripple compensator is shown through experiments. We will also compare the performance of ILC and IFT experimentally, commenting on performance metrics such as ability to reduce tracking error due to accelerations and force ripple,
and the noisiness of the control effort signal.

Lastly, we present an original method for simultaneous application of IFT and ILC. The feedforward controllers may be tuned at the same time as ILC learning is carried out, so that separate learning iterations are not needed for each, saving time. The final result of this method is a well-tuned feedforward controller, which handles errors due to trajectory-dependent factors (accelerations, viscous friction) and disturbances with known structure, such as force ripple, in addition to an ILC signal which handles errors due to remaining factors (unmodeled plant dynamics, unmodeled disturbances). Then the tuned controller may be carried over to other trajectories. In designing the proposed method, care is taken that simultaneous tuning the controller and ILC feedforward signal will not cause over-compensation of errors. Finally, the simultaneous tuning method is demonstrated through experiment.

6.1 IFT Method

This section describes the general IFT controller tuning method. Consider the 2-DOF controller structure in Figure 6.1. From the block diagram, the plant equation is

$$y(k) = P(q)u(k) + v(k)$$

and control law is

$$u(k) = C_\rho(r(k) - y(k)) + F_\rho r(k).$$

Here, $y$ represents plant measured output, $u$ is control input, and $v$ is zero-mean weakly stationary random process. $k \in 1...N$ is the discrete time index, and $q$ is the shift operator. $P$ represents a discrete-time LTI plant, and $C_\rho(q)$ and $F_\rho(q)$ represent LTI feedback and feedforward controllers, parameterized by some parameters $\rho$. Hereafter, the argument of $q$ will be omitted for compactness of notation.
Define the cost function

\[
J(\rho) := \frac{1}{2N} E \left[ \sum_{k=1}^{N} (e(k))^2 + \lambda (u(k))^2 \right]
\]

(6.1)

where \( e(k) := r(k) - y(k) \). The objective is to find the values of \( \rho \) that minimize this cost function.

The above cost function is minimized through an iterative gradient-based search, using the following formula

\[
\rho_{i+1} = \rho_i - \gamma_i R_i^{-1} \frac{\partial J(\rho)}{\partial \rho} (\rho_i)
\]

(6.2)

where \( i \) is iteration number, and \( R_i \) and \( \gamma_i \) are design parameters used to influence algorithm convergence, where \( R_i \) is a positive-definite matrix, and \( \gamma_i \) is a scalar to control step size.

The iterative search method above is implementable if the gradient \( \frac{\partial J(\rho)}{\partial \rho} \) is known. Here, it will be shown how the gradient may be computed from signals taken from experimental runs. Taking the partial derivative of the cost function with respect to controller parameters \( \rho \), the gradient may be written as

\[
\frac{\partial J(\rho)}{\partial \rho} = \frac{1}{N} E \left[ \sum_{k=1}^{N} e(k) \frac{\partial y(k)}{\partial \rho} + \lambda u(k) \frac{\partial u(k)}{\partial \rho} \right]
\]

(6.3)

The signals \( e(k) \) and \( u(k) \) are measurable, but the quantities \( \frac{\partial y(k)}{\partial \rho} \) and \( \frac{\partial u(k)}{\partial \rho} \) are harder since there is no accurate plant model so we don’t know how \( y(k) \) or \( u(k) \) depend on \( \rho \). However, these quantities can be computed from the data from a special set of experiments. The calculation methods for \( \frac{\partial y(k)}{\partial \rho} \) (output-related signals) and \( \frac{\partial u(k)}{\partial \rho} \) (input-related signals) are described in the next two subsections.

### 6.1.1 Output-related signals

In order to compute \( \frac{\partial y}{\partial \rho} \), we first obtain a closed-form expression for \( y(k) \):

\[
y(k) = \frac{P(C + F)}{1 + PC} r(k) + \frac{1}{1 + PC} v(k).
\]

Taking the partial derivative, we have

\[
\frac{\partial y}{\partial \rho} = (1 + PC) P \left( \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right) - P(C + F) P \frac{\partial C}{\partial \rho} \frac{1}{1 + PC^2} r + \frac{-P \frac{\partial C}{\partial \rho}}{(1 + PC)^2} v
\]

\[
= P \frac{1 + PC}{1 + PC} \left( \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right) r - (C + F) P^2 \frac{\partial C}{\partial \rho} \frac{1}{(1 + PC)^2} r - \frac{P}{(1 + PC)^2} \frac{\partial C}{\partial \rho} v
\]

\[
= \left( \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right) \frac{1}{C} Tr - \frac{\partial C}{\partial \rho} (C + F) \frac{1}{C^2} T^2 r - \frac{\partial C}{\partial \rho} \frac{1}{C} C^2 T S v
\]

(6.4)
where $S := \frac{1}{1+PC}$ and $T := \frac{PC}{1+PC}$. In this equation, $C, F, \frac{\partial C}{\partial \rho}$, and $\frac{\partial F}{\partial \rho}$ are all known. The filtered signals $Tr$ and $T^2r$ are unknown, but can be obtained by running experiments on the closed-loop system as explained next.

### 6.1.2 The Three IFT Experiments

Notice that the formula for finding $\frac{\partial y}{\partial \rho}$ involves $Tr$ and $T^2r$, which are double-filtering of the reference through the system. This suggests that the output of the system must be put through the system a second time. Therefore, it is proposed that in each iteration of IFT tuning, there will be 3 experiments performed, during which the controller is fixed, but three different references are used:

\[
\begin{align*}
  r^1 &= r \\
  r^2 &= y^1 \\
  r^3 &= r
\end{align*}
\]

\[
\begin{align*}
  y^1 &= \frac{1}{C + F}(C + F)Tr + Sv^1 \\
  y^2 &= \frac{(C+F)^2}{C^2}T^2r + Sv^2 + \frac{(C+F)}{C}TSv^1 \\
  y^3 &= \frac{1}{C}(C + F)Tr + Sv^3. \\
\end{align*}
\] (6.5)

Here, the superscript refers to the experiment number. Notice that the output signal from the first experiment is used as the reference for the second experiment. Now we will show how to represent the gradient in terms of $y^1, y^2,$ and $y^3$. Define the gradient estimate

\[
est \left[ \frac{\partial y}{\partial \rho} \right] := \frac{1}{C + F} \left( \left[ \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right] y^3 - \left[ \frac{\partial C}{\partial \rho} \right] y^2 \right).\] (6.6)

Note that $\nest \left[ \frac{\partial y}{\partial \rho} \right]$ defined in Equation 6.6 is a perturbed version of $\frac{\partial y}{\partial \rho}$ as calculated in Equation 6.4, due to noises $v^2$ and $v^3$. To see how the noise affects the gradient estimate, notice that $\nest \left[ \frac{\partial r}{\partial \rho} \right]$, can be written in terms of $\frac{\partial y}{\partial \rho}$ as

\[
\nest \left[ \frac{\partial y}{\partial \rho} \right] = \frac{1}{C + F} \left( \left[ \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right] \left[ \frac{C + F}{C} Tr + Sv^3 \right] \ldots \right.
\]

\[
\ldots - \left[ \frac{\partial C}{\partial \rho} \right] \left[ \frac{(C+F)^2}{C^2} T^2r + \frac{C + F}{C} TSv^1 + Sv^2 \right]
\]

\[
\left. \right) = \left[ \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right] \frac{1}{C} Tr - \left[ \frac{\partial C}{\partial \rho} \right] \frac{C + F}{C^2} T^2r \ldots \]

\[
\ldots - \left[ \frac{\partial C}{\partial \rho} \right] \frac{1}{C} TSv^1 + \left[ \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right] \frac{1}{C + F} Sv^3 - \left[ \frac{\partial C}{\partial \rho} \right] \frac{1}{C + F} Sv^2
\]

\[
\nest \left[ \frac{\partial y}{\partial \rho} \right] = \left[ \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right] \frac{1}{C + F} Sv^3 - \left[ \frac{\partial C}{\partial \rho} \right] \frac{1}{C + F} Sv^2.
\]

However, even if the estimate of $\frac{\partial y}{\partial \rho}$ is perturbed, the estimate of the gradient is still unbiased if we make the reasonable assumption that $v^1, v^2,$ and $v^3$ are zero-mean and independent.
6.1.3 Input-related signals

Now we will show how to obtain $\frac{\partial u}{\partial \rho}$ from system signals. The development of this section will follow that of the previous section closely. The closed-form expression for $u$ is

$$u = \frac{C + F}{1 + PC} r - \frac{C}{1 + PC} v.$$

Then the gradient is

$$\frac{\partial u}{\partial \rho} = \left[ \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right] Sr - \frac{C}{C} \left[ \frac{\partial C}{\partial \rho} \right] C S v R - \left[ \frac{\partial C}{\partial \rho} \right] S (1 - T) v.$$

Notice that the input signals of the three ILC experiments in Equations 6.5 are

$r_1 = r$

$u_1 = (C + F) Sr - CS v_1$

$r_2 = y_i$

$u_2 = \frac{1}{C} (C + F)^2 S \alpha R + \frac{1}{C} (C + F) S^2 v_1 - CS v_2$

$r_3 = r$

$u_3 = (C + F) Sr - CS v_3$.

Substituting in signals defined previously, we can form the estimate of the gradient as

$$\text{est} \left[ \frac{\partial u}{\partial \rho} \right] := \frac{1}{C + F} \left( \left[ \frac{\partial C}{\partial \rho} + \frac{\partial F}{\partial \rho} \right] u_3 - \left[ \frac{\partial C}{\partial \rho} \right] u_2 \right).$$

(6.8)

6.1.4 Discussion

In summary, in each iteration of IFT, three experiments are necessary as outlined in Eq. 6.5. Then compute $\text{est} \left[ \frac{\partial y}{\partial \rho} \right]$ from Eq. 6.6 and $\text{est} \left[ \frac{\partial u}{\partial \rho} \right]$ from Eq. 6.8. Next, gradient is computed as in Eq. 6.3 and finally parameter $\rho$ is updated as in Eq. 6.2.

Notice that of the three experiments in Eq. 6.5, although experiments 1 and 3 use identical references, they are both necessary so that the noise does not cause a bias in Eq. 6.3 when the terms are multiplied, assuming that the noises are independent.

6.2 IFT tuning of feedforward controller

6.2.1 Methodology

We applied IFT to iteratively tune the feedforward controller of the wafer stage. In this section, we explain the derivation of the tuning method and show simulation and experimental results.

The block diagram of the wafer stage control system is shown in Figure 6.2. $C(s)$ represents the feedback controller, a PID controller, which we keep fixed during the feedforward tuning. $F(s)$ is the feedforward controller which we wish to tune,

$$F(s) = \rho_1 s^2 + \rho_2 s$$
where $\rho_1$ is the parameter that corresponds to mass and $\rho_2$ corresponds to the damping coefficient. The controller is discretized as

$$F(z) = \rho_1 \left( \frac{2}{T} \frac{1-z^{-1}}{1+z^{-1}} \right)^2 + \rho_2 \left( \frac{2}{T} \frac{1-z^{-1}}{1+z^{-1}} \right)$$

where $T$ is the sampling time.

The objective of controller tuning is to find the values of the parameters $\rho_1$ and $\rho_2$ such that a cost function is minimized. For the cost function, we have chosen

$$J(\rho) = \sum_{k=1}^{N} e(k)^2$$

where $e(k) = r(k) - y(k)$ is the error, and $\rho = [\rho_1 \ \rho_2]^T$ is a vector of controller parameters.

The values for the controller parameters are updated iteratively in a gradient-based search based on the error profile from the previous iteration. The update equation is

$$\rho^{k+1} = \rho^k - \gamma R^{-1} \frac{\partial J}{\partial \rho} \bigg|_{\rho^k}$$

where $\rho^k$ is the parameter estimate in the $k$th iteration, $\gamma$ is a scalar parameter to control step size, $R$ is a matrix to modify the search direction, and $\frac{\partial J}{\partial \rho}$ is the gradient of the cost function with respect to controller parameters evaluated at the present value of $\rho$.

The gradient is estimated by calculations using experimental data. The derivation of the formula for the gradient estimate is as follows. Repeating Equation 6.9, the cost function $J(\rho)$ is given by

$$J(\rho) = \sum_{k=1}^{N} (r(k) - y(k))^2.$$

Therefore the gradient is

$$\frac{\partial J}{\partial \rho} = -2 \sum_{k=1}^{N} (r(k) - y(k)) \frac{\partial y}{\partial \rho}(k).$$
In the above equation, the unknown quantity that must be calculated is $\frac{\partial y}{\partial \rho}$. Referring to the block diagram in Figure 6.2, the output $y(k)$ is written as

$$y = \frac{P(F + C)}{1 + PC}r$$

which gives

$$\frac{\partial y}{\partial \rho} = \frac{P}{1 + PC} \frac{\partial F}{\partial \rho}r$$

$$= \frac{\partial F}{\partial \rho} \frac{1}{C} \frac{1}{1 + PC}y_{noFF}$$

where $y_{noFF}$ denotes the output of the system when no feedforward controller is applied.

### 6.3 Convergence Analysis and Similarity to ILC

In this section, we will demonstrate the similarity of the IFT parameter update law with an Iterative Learning Control (ILC) update law and derive a condition for convergence of the parameters using ILC convergence analysis techniques.

A general ILC-form parameter update law can be written as

$$\rho^{j+1} = \rho^j + L e^j$$

where $L$ is some learning function, and $j$ is iteration index. We will show that the IFT parameter update law can be put in this form. Consider the update equation in the gradient decent, Eq. 6.10:

$$\rho^{j+1} = \rho^j - \gamma R^{-1} \frac{\partial J}{\partial \rho} \bigg|_{\rho^j}.$$ 

Substituting in the formula for $\frac{\partial J}{\partial \rho}$ from Eq. 6.11, we have

$$\rho^{j+1} = \rho^j + 2\gamma R^{-1} \sum_{k=1}^{N} e^j(k) \frac{\partial y}{\partial \rho}(k) \bigg|_{\rho^j}$$

$$= \rho^j + 2\gamma R^{-1} \sum_{k=1}^{N} e^j(k) \frac{\partial F}{\partial \rho} \bigg|_{\rho^j} \frac{1}{C} y_{noFF}(k).$$

Using lifted notation, we can define the vectors $e^j \in \mathbb{R}^{N \times 1}$ and $y_{noFF}^j \in \mathbb{R}^{N \times 1}$ from the appropriate signals, and the matrices $C \in \mathbb{R}^{N \times N}$ and $\frac{\partial F}{\partial \rho_1}, \frac{\partial F}{\partial \rho_2} \in \mathbb{R}^{N \times N}$ from the appropriate
filters. Then the above equation can be written

$$\rho^{j+1} = \rho^j + 2\gamma R^{-1} \left[ \frac{\partial F}{\partial \rho_1} C^{-1} y_{noFF}^j \frac{\partial F}{\partial \rho_2} C^{-1} y_{noFF}^j \right]^T e^j$$

$$= \rho^j + Le^j.$$  

The IFT scheme provides one choice of learning matrix $L$ that is constant from one iteration to the next.

### 6.4 Controller Tuning Results

#### 6.4.1 Simulation

The controller tuning algorithm was first implemented in simulation. The model of the plant used for simulation is the same as given in Chapter 2. Therefore we expect the feedforward parameters $\rho_1$ and $\rho_2$ to converge to 0.4417 and 0.6000, respectively.

The shape of the objective function $J(\rho)$ is shown in Figure 6.3. The shape of $J(\rho)$ depends on the particular reference trajectory used for tuning. The objective function achieves its minimum when $\rho_1$ and $\rho_2$ reach their true values corresponding to the true plant parameters, as expected. It can be seen that the value of the objective function depends more heavily on the $\rho_1$ parameter than on $\rho_2$ for our particular choice of trajectory.

The controller tuning algorithm was applied to the stage for ten iterations. The reference trajectory used for tuning is a scan and return motion and is shown in Figure 6.4. Figure 6.5 shows that the value of the objective function decreases with each iteration as desired. The sum of squared error decreased from $5.30 \times 10^{-9}$ to $4.26 \times 10^{-11}$. The convergence of the feedforward parameters is shown in Figure 6.6. The parameters appear to converge to their expected values (values from the inverse of the plant model). The error profile after ten cycles of controller tuning is compared with the profile before any tuning in Figure 6.7. It is seen that the peak error during acceleration is reduced from $7.15 \times 10^{-6}$ m to $1.13 \times 10^{-6}$ m.

#### 6.4.2 Experiment

Next we applied the iterative tuning algorithm experimentally to the wafer stage. The wafer stage was placed in a feedback loop with a PID controller to ensure stability while the feedforward controller was being tuned. Figure 6.8 is a plot of the cost function at each iteration. The figure shows that the sum of the square of the error decreased from $1.61 \times 10^{-6}$ to $5.13 \times 10^{-9}$ m$^2$ at the tenth iteration while the most significant reduction took place at the second iteration. Figure 6.9 shows a comparison of the error profile in the 1st and 10th iterations. It can be seen that the peak error during acceleration has decreased from $7.49 \times 10^{-5}$ m to $5.98 \times 10^{-6}$ m. The plot of the feedforward parameters over each iteration is shown in Figure 6.10. Both parameters converge.
Figure 6.3: Cost as a function of $\rho_1\rho_2$

Figure 6.4: Reference trajectory for wafer stage

Figure 6.5: Minimization of cost function (sum of squared error) in simulation over 10 cycles of tuning
Figure 6.6: Convergence of feedforward parameters in simulation

Figure 6.7: Reduction of error after 10 cycles of tuning in simulation
Figure 6.8: Minimization of cost function (sum of squared error) in experiment over 10 cycles of tuning

Figure 6.9: Reduction of error after 10 cycles of tuning in experiment

Figure 6.10: Convergence of feedforward parameters in experiment with initial values $\rho_1 = 0, \rho_2 = 0$
Figure 6.11: Convergence of feedforward parameters in experiment with initial values $\rho_1 = 0.4417$ $\rho_2 = 0.6000$.

Figure 6.12: Comparison of two feedforward controllers: 1. using parameters from iterative tuning vs. 2. using parameters corresponding to inverse of plant obtained through system identification.
Theoretically, the best performance should be achieved when the feedforward controller is the inverse of the plant. However, it is interesting to note that the parameters in Figure 6.10 converge to values different from the expected (i.e. values from the inverse of the plant model obtained through system identification). Even if the controller tuning is started with the parameters’ initial values set to the expected values, after tuning, the parameters still converge to the same values as before (compare Figures 6.10 and 6.11). Using the new values of the parameters in the feedforward controller results in lower sum of squared error than using the expected values as seen in Figure 6.12. Therefore, applying iterative tuning is still beneficial even when a model of the plant already exists, possibly because the plant model always has inaccuracies and the plant changes slightly over time.

6.5 Force Ripple Compensation

One implementation issue arises due to disturbances caused by force ripple of the linear motor [25], [35].

As explained in Chapter 2, force ripple can be modeled as

\[ F_{\text{ripple}}(y) = u(y) \left[ \sum_{k=1}^{N} a_k \sin(k\omega_0 y) + \sum_{k=1}^{N} a_k \cos(k\omega_0 y) \right] \tag{6.12} \]

where \( y \) is stage position, \( F_{\text{ripple}}(y) \) is the force caused by the force ripple disturbance, \( u \) is the controller output voltage, and \( \omega_0 \) is the basic frequency of the ripple which is known beforehand.

The presence of force ripple adversely affects the convergence of controller parameters. During experimental tuning, we reduced the effect of the force ripple on the gradient calculation by approximating the error profile by a smooth polynomial during segments where force ripple is present.

However, instead of ignoring the effect of force ripple, now we try to cancel it out using additive feedforward control. We will show that the same iterative controller tuning method used earlier can be used to tune the parameters of the force ripple compensation feedforward controller. This method is still effective even when applied to eliminate nonlinear force ripple.

6.5.1 Methodology

We investigated the use of feedforward methods to compensate for force ripple. In this section we present the iterative tuning method which we apply to tune the force ripple compensator controller. The derivation is very similar to that presented in Section 6.2.

We desire to cancel out the force ripple by additive feedforward control of the form

\[ F_{fr}(t) = k_c v(t) \left[ \sum_{k=1}^{N} a_k \sin(k\omega_0 r(t)) + \sum_{k=1}^{N} a_k \cos(k\omega_0 r(t)) + \gamma_0 + \gamma_1 r(t) \right] \]
where \( r(t) \) is reference position, \( v(t) \) is reference velocity, and \( k_c \) is a constant. The reason for approximating \( y(t) \) and \( u(t) \) in Equation 6.12 by \( r(t) \) and \( k_c v(t) \) is so that the feedforward data \( F_f \) can be computed offline and remains constant from run to run.

The objective of the iterative tuning is to tune the coefficients \( a_k \) and \( b_k \) to minimize a cost function which is a quadratic function of tracking error, the same as in Eq. 6.9. The gradient of the cost function with respect to controller parameters is similarly identical to Eq. 6.11. However, now we must recalculate the gradient \( \frac{\partial y}{\partial \rho} \).

From the block diagram in Figure 6.13, we see that with the addition of feedforward force ripple compensation, the output is given by

\[
y(t) = \frac{PC}{1+PC}r(t) + \frac{PF}{1+PC}r(t) + F_{\text{ripple}} + \frac{P}{1+PC}k_c v(t) \left[ \sum_{n=1}^{N} a_k \sin(k\omega_o n) + \sum_{n=1}^{N} b_k \cos(k\omega_o n) \right].
\]

Taking the partial derivatives with respect to \( a_k \) and \( b_k \), we have

\[
\frac{\partial y}{\partial a_k} = \frac{P}{1+PC}k_c v(t) \sin(k\omega_o n), \quad \frac{\partial y}{\partial b_k} = \frac{P}{1+PC}k_c v(t) \cos(k\omega_o n).
\]

However, assuming that the force ripple disturbances are at low frequencies, we can approximate \( \frac{PC}{1+PC} \approx 1 \), so we will use simply

\[
\frac{\partial y}{\partial a_k} = \frac{1}{C} k_c v(t) \sin(k\omega_o n), \quad \frac{\partial y}{\partial b_k} = \frac{1}{C} k_c v(t) \cos(k\omega_o n)
\]

for the partial derivative of \( y \). Using this approximation will eliminate the need to do a preliminary experiment to calculate \( \frac{P}{1+PC}k_c v(t) \sin(k\omega_o n) \) for each harmonic frequency \( k\omega_o \) of the force ripple.
Next, the feedforward force ripple compensator parameters were tuned for the wafer stage. The results are shown in Figures 6.14, 6.15, and 6.16. The $\infty$-norm of the error during constant velocity scan phase decreased from $1.51 \times 10^{-6}$ m to $6.33 \times 10^{-7}$ m, and the sum of squared error decreased from $1.98 \times 10^{-9}$ to $1.52 \times 10^{-10}$.

The residual force ripple seen in Figure 6.15 is thought to be a result of the inaccuracy of the approximation of the disturbance signal from the nonlinear force ripple model with the additive feedforward control signal $F_{fr}(t)$. The DC-offset and slowly varying trend in the error profile is a result of the disturbance forces from the cables connected to the wafer stage, including power supply cables and sensor cables.

Figure 6.14 shows the convergence of parameters $a_k$, $b_k$. The oscillations in the estimates of parameters $\gamma_0$, $\gamma_1$, which correspond to the cable force disturbance, may be reduced by better modeling of cable forces, or by choosing a smaller step-size in the optimization algorithm.

6.5.2 Experiment

Figure 6.14: Convergence of Feedforward Parameters in Experiment

Figure 6.15: Reduction of Error After 10 Cycles of Tuning in Experiment
6.6 IFT Tuning of Feedback Controller

In this section we will consider the IFT tuning of the PID controller for the wafer stage. First, we will explain the derivation of the methodology and compare with the case of tuning feedforward controller. Next, we will present simulation results, and finally present experimental results.

6.6.1 Controller Tuning Methodology

Consider the system shown in Fig. 6.2. The objective is to tune the feedback controller $C$. During this tuning, we will consider $F$ to be zero.

Define the performance objective function to be the same as that defined in Equation 6.1, then the gradient is the same as in 6.3. Previously, it was seen that $\frac{\partial y}{\partial \rho}$ and $\frac{\partial u}{\partial \rho}$ are given by Equations 6.6 and 6.8. If we set $F = 0$ in these equations, then they simplify to

$$\text{est} \left[ \frac{\partial y}{\partial \rho} \right] := \frac{1}{C + F} \left( \left[ \frac{\partial C}{\partial \rho} \right] y^3 - \left[ \frac{\partial C}{\partial \rho} \right] y^2 \right)$$

$$\text{est} \left[ \frac{\partial u}{\partial \rho} \right] := \frac{1}{C + F} \left( \left[ \frac{\partial C}{\partial \rho} \right] u^3 - \left[ \frac{\partial C}{\partial \rho} \right] u^2 \right).$$

The PID controller structure being used is

$$C = k_p \left( 1 + k_i \frac{T_s}{1 - z^{-1}} + k_d \frac{1 - z^{-1}}{T_s} \right).$$

Therefore, if we let $\rho = [ k_p \ k_i \ k_d ]^T$, the gradient $\frac{\partial C}{\partial \rho}$ is

$$\frac{\partial C}{\partial \rho} = \begin{bmatrix} 1 + k_i \frac{T_s}{k_p T_s} + k_d \frac{1 - z^{-1}}{T_s} \\ \frac{1 - z^{-1}}{k_p T_s} \\ k_p \frac{1 - z^{-1}}{T_s} \end{bmatrix}.$$
Now we have all the pieces needed for tuning of $C$. Notice that IFT tuning of feedback controllers requires a double-filtering of the reference trajectory through the closed-loop system whereas tuning of feedforward controller only requires single-filtering. This is why the second experiment must be carried out to obtain $y^2$ but $y^2$ was not needed in feedforward tuning, only $y^1$.

### 6.6.2 Simulation Results

The iterative controller tuning method described in the previous section was applied in simulation to a model of the stage for 20 iterations. The matrix $R$ was chosen such that the step size in the optimization algorithm of each parameter was constant. In Fig. 6.17, it is shown that the cost function decreases. Fig. 6.18 shows the evolution of the parameters. Fig. 6.19 shows that the error profile was improved.

Several implementation issues were encountered in tuning the feedback controller. One is that parameter $k_i$ tended to zero or even negative values. We added an artificial disturbance at the input to the plant during the simulation to ensure integral action remains. Another is that tuning the feedback controller is much harder than feedforward because the shape of the optimization surface is rough and the Hessian is ill-conditioned. This is why we constrained the step-size of each optimization parameter to be constant. Otherwise, the parameter values would change much in the first iteration but not move much in the second and afterwards.
Figure 6.18: Cost function decreases with tuning of PID controller

Figure 6.19: Cost function decreases with tuning of PID controller
6.7 Comparison of two Iterative Feedforward Tuning Methods: ILC and IFT

In this section, we will compare two methods of iteratively tuning feedforward control: ILC and IFT [28]. We will compare the performances of the two in terms of reduction of error in acceleration and scan phases and control effort. We follow with a discussion regarding the results of the comparison and consider the merits of each scheme.

6.7.1 Comparison

An IFT feedforward controller and ILC feedforward signal were tuned experimentally and performance is compared in this section. The controller topology is same as shown in Fig. 6.2 and reference trajectory is same as in Section 6.2. During both sets of experiments, the feedback controller was held to be constant. The performance of ILC and IFT is compared in the following graphs. The results for ILC are shown after 9 cycles of learning, and IFT for after 10. Fig. 6.20 shows a comparison of the tracking error of both methods. From this figure, we can see that ILC is much more effective in eliminating error during the acceleration phase and also peak error during the constant velocity scan phase. Fig. 6.21 shows a magnification of the error during the tracking phase. ILC is also more effective at eliminating the effect of force ripple. However, the feedforward signal produced from ILC was the noisiest, as seen in Fig. 6.22.

6.7.2 Discussion

We will now discuss the advantages and disadvantages of each method. ILC and IFT both do not require a plant model to implement; however, the performance of IFT is limited by the choice of feedforward controller structure. Since we only chose a second order feedforward controller, the IFT signal failed to completely eliminate the transient error due to acceleration as seen in Fig. 6.20. Better performance may have been possible if we had chosen a higher-order transfer function structure for the inverse model of the plant. Also, the chosen model of force ripple did not have enough complexity (harmonics) for IFT to completely compensate for force ripple, as shown in Fig. 6.21. In addition, ILC is good at compensating for disturbances that repeat from iteration to iteration, but this kind of disturbance causes bias in the computation of the gradient in IFT so causes problems in convergence. On the other hand, noise does not directly enter into the IFT signal but does in ILC, as seen in Fig. 6.22. In addition, whereas ILC results in a tuned feedforward signal, IFT results in a tuned controller. This controller is applicable to many trajectories, while the ILC signal is only usable for the same trajectory used in tuning. The reference trajectory can be changed even while tuning in IFT.
Figure 6.20: Comparison of Error
6.8 Simultaneous tuning of ILC and IFT

In the previous section, we saw that ILC and IFT each have unique advantages. ILC’s performance generally surpasses that of IFT, but the tuned signal is limited for use on only one trajectory. An IFT-tuned controller is useable for a variety of trajectories. Therefore, it would be advantageous if the two methods were combined: the IFT controller could compensate for transients caused by changes in set-point, and the ILC signal could compensate for repetitive disturbances. Furthermore, it would be more efficient if the two methods could be tuned at once instead of requiring a separate set of iterations for each.

This section describes a novel method for combining ILC and IFT tuning. In each iteration, the ILC signal is updated as normal, but the IFT cost function is modified so that the IFT input attempts to minimize the ILC input of the same iteration. The result is a tuned feedforward signal with good performance for the present trajectory, along with a tuned feedforward controller that can be taken and used for different trajectories.
Figure 6.22: Comparison of Feedforward Signal
6.8.1 Algorithm

Consider the block diagram shown in Fig. 6.23. For IFT, define a new cost function

\[ J(\rho) = \sum_{n=0}^{N} (u_{ILC}^{k+1}(n))^2 \]

where \( n \) indicates sample number in time domain and \( k \) indicates iteration number. \( \rho^k \) of iteration \( k \) is calculated based on minimizing the norm of \( u_{ILC}^{k+1} \). We define the ILC update law as

\[ u_{ILC}^{k+1} = Q(u_{ILC}^k + Le^k) \]

where \( L \) is a learning gain chosen appropriately for stability, and \( Q \) is a low-pass filter. Then \( J(\rho) \) is equal to

\[ J(\rho) = \sum_{n=0}^{N} (Q(u_{ILC}^k + Le^k))^2. \]

Therefore the gradient is

\[
\frac{\partial J}{\partial \rho} = 2 \sum (Q(u_{ILC}^k + Le^k))QL\frac{\partial e^k}{\partial \rho} \\
= 2 \sum (Q(u_{ILC}^k + Le^k))QL\frac{-P}{1 + PC} \frac{\partial F}{\partial \rho r} \\
= -2 \sum (Q(u_{ILC}^k + Le^k))QL\frac{\partial F}{\partial \rho} \frac{1}{C} y^1 \\
= -2 \sum (u_{ILC}^{k+1})QL\frac{\partial F}{\partial \rho} \frac{1}{C} y^1.
\]

The term

\[
\frac{\partial F}{\partial \rho} \frac{1}{C} y^1
\]

is the same as in Section 6.2. The update of \( \rho \) is carried out as normal as in Eq. 6.10.

6.8.2 Simulation

The performance of this scheme is first verified in simulation. The results of simulation are shown in Figs. 6.24 through 6.27. Figure 6.24 shows that this scheme reduced error due to both changes in the reference setpoint (during the acceleration phase) and force ripple disturbances (during the constant scan phase). The breakup of the feedforward signal into ILC and IFT components is shown in Figs. 6.25 and 6.26. It is seen that \( u_{IFT} \) mainly compensates for errors due to accelerations in the trajectory, while \( u_{ILC} \) mainly compensates...
for force ripple, which is impossible to compensate for under the current chosen feedforward controller structure. Lastly, Fig. 6.27 shows that the tuned parameters in the feedforward controller converged to values close to what was found in Section 6.2. This scheme successfully accomplishes the reasons for simultaneous ILC/IFT tuning.

6.8.3 Experiment

The previous section’s results were experimentally verified on the wafer stage system. The results are shown in Figs. 6.28 through 6.31. In Fig. 6.28, we can see that the norm of the error was greatly reduced, and especially the effects of force ripple and error caused by acceleration are reduced. Fig. 6.29 shows a comparison of the ILC input obtained after 10 iterations of tuning when ILC is used alone, and when ILC is tuned together with IFT tuning of the controller. Also, Fig. 6.30 shows the output of the feedforward controller that was tuned with IFT simultaneously to ILC. From looking at Fig. 6.30 and Fig. 6.29, it can be seen that the IFT controller makes a signal that compensates for the error caused by trajectory, and the ILC tuning makes a signal that compensates for the error caused by
Figure 6.25: ILC feedforward signal after 10 iterations of tuning

Figure 6.26: IFT feedforward signal after 10 iterations of tuning

Figure 6.27: Convergence of parameters in IFT feedforward controller
disturbances such as force ripple, gravity, and cable tug. From comparing the two signals in Fig. 6.29, it is seen that the IFT controller together with ILC resulted in an ILC signal with lower 2-norm, as desired in the minimization problem in the setup of the problem. The reduction in the norm comes mainly from removing the large peaks in the ILC signal during acceleration, because this is now handled by the output of the feedforward controller instead. Finally, Fig. 6.31 shows that the feedforward controller parameters appear to converge to the same values found in tuning in Section 6.2, giving confidence that the algorithm is working as desired.

6.9 Chapter Summary

In this chapter we introduced the controller tuning method IFT. IFT was used to tune parameters for the wafer stage inverse-model based feedforward controller, and also structured force ripple feedforward compensator. IFT was compared with ILC performance in terms of maximum reduction of tracking error, and noisiness of control input. Finally, a
Figure 6.30: IFT feedforward signal after 10 iterations of tuning in experiment

Figure 6.31: Convergence of parameters in IFT feedforward controller in experiment
method of tuning feedforward controllers at the same time with ILC learning iterations was presented.
Chapter 7

Conclusion

This dissertation investigated methods for high-precision motion control of systems, specifically systems that perform a repetitive task. The repetition may be exploited to learn information from previous runs, in order to improve future performance. In particular, in this thesis, we focus on using learning to better design both feedforward signals, and controller parameters.

7.1 Issues Addressed

The following questions were explored:

*How do you reduce vibrations in repetitive systems?*

We found that the vibration reduction method should be matched with the vibration type (external or unmodeled dynamics, repetitive or nonrepetitive, high frequency or low frequency). In particular, for vibrations that are repetitive from iteration to iteration, ILC may be used to reduce the error; however, care must be taken in designing the ILC learning law. It was found that the learning filter design is important, especially the gain and phase of the filter at the vibration frequency. Also, it was shown that time-varying filters can be used to achieve both vibration reduction high-bandwidth tracking.

*How do you achieve high precision tracking when the reference trajectory changes?*

How do you generalize ILC learned feedforward signals to other trajectories? We found that it is possible to modify ILC learned signals to work with other trajectories, even if the plant model is not known at all, by knowing the trajectory structure and the relation of the training trajectory to the new trajectory. This was demonstrated on a class of scan trajectories in particular; perhaps it is possible for other trajectory types as well.

*How do you tune controllers to achieve precision tracking in repetitive systems?*
The IFT method was used to tune feedforward controllers, including an inverse-model based feedforward controller, a nonlinear force ripple feedforward compensator, and PID controller gains.

7.2 Future Research

The investigations made for this thesis brought up many additional questions, which would make interesting directions for further research.

With regards to the research into ILC design for vibration reduction, we found that there is a tradeoff between the convergence rate and the converged value of tracking error, when only P-type is used, and that sometimes it is possible to achieve both by careful design of the ILC law. This was shown through empirically through experimental data; however, precise analysis of this tradeoff is an interesting direction for future research.

With regards to the research into generalizing ILC feedforward signals to all scan trajectories, the biggest issue is to include a way to separate those parts of the ILC feedforward signal due to the reference, and those parts due to the disturbances, and furthermore to figure out how to modify the part of the ILC feedforward signal due to disturbances when new trajectories are used. This requires understanding the characteristics of the disturbance, and how it is affected by the trajectory. The implementation in this thesis used a DOB, but if the method can be developed to handle disturbances, we can avoid this and also achieve better tracking performance for the new trajectory. In addition, another direction of research is to extend the method to other types of trajectories, not only scan motions.

As for the research on automated feedforward controller tuning, further investigation is needed on how increasing the controller complexity for tuning of the parameters can decrease tracking error. Also, investigation of the optimization algorithm, such as ways to speed the convergence, or better choose the step size and search direction, are useful.
Bibliography


