

© 2010 by Kira Louise Barton. All rights reserved.

PRECISION COORDINATION AND MOTION CONTROL OF MULTIPLE
SYSTEMS VIA ITERATIVE LEARNING CONTROL

BY

KIRA LOUISE BARTON

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Professor Andrew Alleyne, Chair
Professor Placid Ferreira
Professor Geir Dullerud
Associate Professor Srinivasa Salapaka
Professor Lucy Pao, University of Colorado at Boulder

Abstract

In today's engineering world, many emerging applications ranging from manufacturing to the autonomous vehicle industry require coordination of multiple systems. Traditional approaches for controlling these systems often neglect the underlying coupling in the application. To stay at the forefront of these fields requires the development of innovative approaches to new challenges. The research in this dissertation focuses on designing novel control strategies for coordinated applications. Electrohydrodynamic jet (E-jet) printing, an example of an emerging micro/nano-manufacturing process with applications in biotechnology and flexible electronics, requires multiple systems to work in a coordinated manner to achieve a desired objective. Repetitive execution of processes such as E-jet can be harnessed to achieve high performance. Iterative learning control (ILC) is an adaptive control technique for improving process performance in systems that execute a task repetitively. This research simultaneously exploits the repetitiveness and inherent coupling of the desired outcome by applying a coordinated ILC approach to processes such as E-jet. The versatility of this approach will be demonstrated through applications ranging from robotics to emerging manufacturing processes.

To Alex and our little family

Acknowledgments

I would like to acknowledge the different contributions to this dissertation from the many people who have played an integral part. First and foremost I would like to thank my advisor, Professor Andrew Alleyne. His drive, determination, and continuous support throughout my PhD have been a constant source of inspiration and encouragement. He has pushed me to set higher goals for myself, forcing me to evaluate what I want in life and how to go about achieving it. I would also like to thank my committee members, Prof. Placid Ferreira, Prof. Geir Dullerud, Prof. Srinivasa Salapaka, and Prof. Lucy Pao from the University of Colorado at Boulder for their time, knowledgeable discussions, and insightful comments. I would like to specifically thank Prof. Doug Bristow and Prof. Sandipan Mishra for their invaluable contributions to my understanding of ILC and our joint work on the Electrohydrodynamic jet printing process through the Center for Nanoscale Chemical-Electrical-Mechanical-Manufacturing Systems (Nano-CEMMS). I would also like to thank David Hoelzle for his contribution to the work on dissimilar dynamics presented in Appendix J.

I would like to acknowledge that the funding for this research was provided by the National Science Foundation (NSF) Nano-CEMMS Center under Awards DMI 0328162 and CMMI 0749028. Additional educational opportunities were sponsored in part by an NSF International Research and Education in Engineering (IREE) travel grant and an American Society of Mechanical Engineers (ASME) Graduate Teaching Fellowship. Through the NSF IREE travel grant, I participated in a five month study abroad opportunity at the Eindhoven University of Technology (TU/e)

in The Netherlands during the Fall Semester 2007. As an ASME Graduate Teaching Fellow, I had the unique opportunity to be the instructor for an undergraduate senior / first-year graduate student level controls course during the Fall Semester 2009. I am incredibly grateful for my many amazing opportunities and thank all of the resources that contributed to my graduate studies.

Through my experiences at TU/e in The Netherlands, I would like to thank Prof. Maarten Steinbuch and Prof. Okko Bosgra for their guidance and support, and Dr. Jeroen van de Wijdeven for the in-depth discussions on ILC and his contribution to the coupled norm optimal framework presented in Chapter 5.

I would also like to thank and acknowledge the lively research discussions, collaborative environment, encouragement, support, and friendships from both past and present ARG members. The group has played a pivotal role in my experience and success at the University of Illinois and I look forward to continuing these rewarding relationships in the future.

Family has always played an important role in my life and I would like to thank my family and friends for being so supportive and understanding during this time of my life. Lastly, I would like to thank Alex for being who he is, the most encouraging, considerate, and loving person I know.

Table of Contents

List of Tables	xi
List of Figures	xii
List of Abbreviations	xxii
List of Symbols	xxiii
Chapter 1 Introduction	1
1.1 Manufacturing	2
1.1.1 Emerging Manufacturing Technology	3
1.1.2 E-jet Printing Capabilities	5
1.2 Control Objective	7
1.3 Research Objective	8
Chapter 2 Traditional Control Approaches - A Literature Review .	10
2.1 1-D Control Strategies	11
2.2 2-D Control Strategies	13
2.2.1 Iterative Learning Control	13
2.2.2 Cross-Coupled ILC	15
2.2.3 Shape Tracking	18
2.3 3-D Control Strategies	19
Chapter 3 Iterative Learning Control	24
3.1 System Setup	26
3.2 Frequency Domain Design	29
3.3 Time Domain Design	30
3.3.1 Nominal Convergence	31
3.3.2 Robust convergence	32
3.3.3 Performance	34
3.3.4 Tuning Guidelines for Time-Invariant Weighting Matrices . . .	36
3.4 Servo System Example: Norm-Optimal Design	37

Chapter 4	Coupling and Coordination of Multiple Systems	44
4.1	Contour Error	44
4.2	Formation Control	46
4.2.1	Unit-Centered Reference Approach	46
4.2.2	Leader Reference Approach	48
4.2.3	Neighbor Reference Approach	49
4.2.4	Mapping Position Tracking to Formation Errors	50
Chapter 5	2-D Coupled Iterative Learning Control	52
5.1	Introduction	52
5.2	Background	54
5.3	Time-varying Weighting Matrices	56
5.3.1	Motivation	56
5.3.2	Design Methodology	58
5.4	Design Example 1: Time-Varying Q Weighting Matrix	61
5.4.1	Design Step 1	61
5.4.2	Design Step 2	63
5.4.3	Design Step 3	64
5.4.4	Simulation Results	67
5.5	Design Example 2: Time-Varying S Weighting Matrix	70
5.5.1	Motivation	71
5.5.2	Weighting Matrix Design	72
5.5.3	Simulation Results	73
5.6	Design Example 3: Time-Varying R Weighting Matrix	75
5.6.1	Motivation	76
5.6.2	Weighting Matrix Design	77
5.6.3	Simulation Results	78
5.7	Experimental Validation of Time-Varying Q ^{tv}	81
5.8	Discussion	82
Chapter 6	3-D Coordinated Iterative Learning Control	85
6.1	Introduction	85
6.2	Coordinated Weighting Matrices	87
6.2.1	Individual System Control Design	87
6.2.2	Coupled Agent Control Design	88
6.2.3	Formation Control Design	89
6.2.4	Convergence and Stability Analysis	92
6.3	Design and Implementation	92
6.3.1	Design Methodology	93
6.4	Simulation Example	96
6.4.1	System Set-up	96
6.4.2	Implementation and Results	98

Chapter 7	3-D Coordinated ILC: Experimental System	104
7.1	System Description	105
7.2	Kinematics	107
7.2.1	Forward Kinematics	108
7.2.2	Inverse Kinematics	110
7.2.3	Jacobian Matrix	111
7.3	Dynamics	112
7.3.1	Euler-Lagrange	112
7.3.2	Single Leg Dynamic Model	113
7.3.3	Experimental System Modeling	115
Chapter 8	3-D Coordinated ILC: Experimental Validation	119
8.1	Experimental Set-up	119
8.1.1	System Set-up	119
8.1.2	Trajectory Design	122
8.1.3	Case Studies	125
8.2	Coordinated Learning Controller	126
8.2.1	Case 1 Design	128
8.2.2	Case 2 Design	129
8.2.3	Case 3 Design	131
8.3	Experimental Results	132
8.3.1	Implementation	132
8.3.2	Raster Trajectory Results	133
8.3.3	Spiral Trajectory Results	136
8.3.4	Formation Tracking in the Presence of Model Uncertainty and Exogenous Disturbances	140
8.4	Conclusions	143
Chapter 9	Conclusion and Future Direction	150
9.1	Additional Applications	152
9.1.1	Autonomous Vehicles	152
9.1.2	Robotics	153
9.1.3	Business Modeling	155
9.2	Future Theoretical Development	155
9.2.1	Task Based Learning	155
9.2.2	Time-Varying Objective Weighting Gains	156
9.2.3	Hierarchical Approach	157
9.2.4	Design Benefits of Structured versus Unstructured Uncertainties	158
9.2.5	Performance and Robustness Analysis of the Series versus Par- allel Design Architecture	159
Appendix A	Coefficients for the μ-RD Y-axis Plant (5.12) and Con- troller (5.13) Models	161
Appendix B	Coefficients for Simulation Example in Chapter 6 . . .	162

Appendix C	MATLAB Code for Solving Forward Kinematics Problem	163
C.1	Spherical Mapping Program	163
C.2	Coordinate Mapping Program	168
Appendix D	Derivation of Simple System Dynamics for Single Leg Link from Chapter 7	169
Appendix E	Bode Diagrams for Systems 2 and 3 for System ID from Chapter 7	172
Appendix F	MATLAB Code to Build Jacobian	174
Appendix G	Experimental Results for the PKM setup from Chapter 7	176
G.1	Raster Trajectory	176
G.2	Spiral Trajectory	180
G.3	Spiral Trajectory with Agent 1 Subject to Model Uncertainty and Force Disturbance	183
Appendix H	MATLAB Code: Simple 2-D ILC Design Example . .	187
H.1	2D Weighting Matrix Design	187
H.2	2D Simulation Program	192
H.3	Gaussian Filter Program	196
H.4	Simulink Model	197
Appendix I	MATLAB Code: Simple 3-D ILC Design Example . . .	198
I.1	MatLAB Code for Controller Design	198
I.1.1	Case 1	204
I.1.2	Case 2	205
I.1.3	Case 3	207
I.1.4	Case 4	209
I.1.5	Lifted Matrices	211
I.1.6	Simulation Models	212
I.2	Control and Error Signals	213
I.3	Simulation Results	214
Appendix J	2-D ILC Design for Dissimilar Systems	218
J.1	Introduction	218
J.2	Class of Systems	220
J.3	Coupling of Multiple Dissimilar Systems	223
J.4	Norm Optimal ILC	227
J.5	System Setup	230
J.6	Results	236
J.7	Concluding Remarks and Future Direction	238

References	242
Vita	252

List of Tables

7.1	Key Parameters for Leg Linkage Description	107
7.2	Design Variables for Leg Linkage Control and Implementation	107
7.3	Variables for Single Leg Dynamics	115
8.1	PID Gains for the Stabilizing Feedback Controller	121
8.2	Weighting Matrix Gains for Case 1	129
8.3	Weighting Matrix Gains for Case 2	129
8.4	Weighting Matrix Gains for Case 3 with the Raster Trajectory	131
8.5	Weighting Matrix Gains for Case 3 with the Spiral Trajectory	131
I.1	Converged Control Signals U_x and U_y	213
I.2	Converged Error Signals E_x and E_y	213
I.3	Converged Error Signals E_d and E_h	214

List of Figures

1.1	Schematic of the E-jet printing set-up including nozzle and ink chamber, air supply for back pressure, conducting substrate, and translation and tilting stages.	4
1.2	Block "I" printed using the desktop E-jet system. Image was printed from a nozzle diameter of 5 μm resulting in printed droplets with an average measured diameter of 2.8 μm . Typical ink jet droplets with a 20 μm diameter have been superimposed on the printed image for comparison purposes. . .	5
1.3	Block diagram of the parallel ILC process.	8
1.4	Block diagram of the series ILC process.	8
2.1	Traditional control approaches in terms of dimensional analysis. Note that the axes describe the degrees of freedom for the different approaches. . . .	11
2.2	Image of a CNC process. Note that coupling the movements of the XYZ axes may result in better path following and the construction of a more precise part.	12
2.3	Koren Variable Gain CCC Design [1]	13
2.4	Image of semi-conductor fabrication in which wafer scanning is an important requirement. This is an example of a 2-D application in which the repetitive nature of the process enables the use of learning control schemes such as Iterative Learning Control.	14
2.5	Image of a μ -Robotic Deposition system used for contoured applications such as the fabrication of bone scaffolding. The repetitive patterns and functional tasks combined with the contoured patterns associated with this process make it well suited for Cross-Coupled Iterative Learning Control. .	16
2.6	Block diagram of the combined CCILC and ILC design process.	17
2.7	Examples of 2-D cooperative tracking. A) Formation of planes illustrating coordinated flying patterns. B) A school of fish displaying cooperative maneuvering.	19
2.8	Formation tracking techniques described in terms of a group objective: formation-center reference, leader-reference, and neighbor-reference. Note that in all three cases, the individual systems maintain their own objectives independently from the group objective.	20

2.9	Examples of 3-D applications in which multiple systems perform a coordinated task repetitively. A) Common manufacturing application, known as pick n' place. B) Coordinated agrosience practices are a critical component of increased efficiency in food production. C) Search and rescue tasks can benefit from cooperative learning techniques.	21
2.10	Figure identifying the key elements in a coordinated learning controller. The elements are associated with specific control strategies as indicated by the large 'X'. Note that the proposed control scheme addresses all of the key elements.	23
3.1	ILC process. As the number of iterations increases, the feedforward time domain control signal is determined and the error signal is minimized . . .	25
3.2	Bode plot of the servo system in (3.25)	38
3.3	Triangle reference signal and system output using feedback controller. . .	39
3.4	Error, feedback control, and ILC time-series using norm-optimal ILC. Note how the feedback control signal is slowly being replaced by the ILC feedforward signal with increasing iteration.	40
3.5	RMS converged error values for various \mathbf{Q} weighting matrices.	41
3.6	RMS converged error and monotonic convergence values for various \mathbf{S} weighting matrices.	41
3.7	RMS error values for varying \mathbf{R} weighting matrices. Note the longer iteration range with increasing value of r	42
3.8	RMS error values for varying \mathbf{R} weighting matrices with trial varying noise. Note the longer iteration range.	42
3.9	Comparison of the effects of the weighting matrix design on performance, robustness to model uncertainty and disturbances, and convergence. . . .	43
4.1	Trajectory illustrating contour and individual axis errors. Linearized coupling gains at point in time (k) can be used to simplify the derivation of the contour error.	45
4.2	Formation-center formation tracking technique. Note that each agent determines the group formation center based on the positions of the other agents.	47
4.3	Leader reference formation tracking technique. Note that each agent, aside from the lead agent, determines its position from the lead agent.	48
4.4	Neighbor reference formation tracking technique. Note that each agent determines its position from neighboring agents.	49
4.5	Example formation set-up with three individual multi-axis agents. Note that the planar formation errors are defined in terms of horizontal formation errors $\{E_{d1}, E_{d2}, E_{d3}\}$ and vertical formation errors $\{E_{h1}, E_{h2}, E_{h3}\}$	50
5.1	Experimental results comparing a traditional feedback controller versus a coupled ILC design implemented on a micro-Robotic Deposition system. Results taken from [2].	55
5.2	Time-varying Weighting Matrix Design Methodology.	60

5.3	Raster trajectory containing linear sections (B), contoured sections (A,C), high acceleration sections ($t_1 - t_6$), and low acceleration sections.	64
5.4	Initial contour tracking errors without the use of learning. Individual axis errors show a similar trend in peak locations for initial tracking errors. . .	65
5.5	Alternating gains $\gamma_Q(k)$ and $1 - \gamma_Q(k)$ to switch between weighting individual versus coupled error signals.	66
5.6	Profile for the diagonal elements in the weighting matrix Σ_Q . Notice that the gain is increased in the locations corresponding to the high acceleration sections ($t_1 - t_6$).	67
5.7	Image of the multi-axis robotic testbed used in this work.	68
5.8	Comparison of RMS contour errors for feedback, norm optimal control using time-invariant $(\gamma_Q, 1 - \gamma_Q, \sigma_Q)$ gains, and norm optimal control using time-varying $(\gamma_Q(k), 1 - \gamma_Q(k), \sigma_Q(k))$ gains [Simulation].	69
5.9	Trajectory tracking comparison of the norm optimal controllers with time-invariant and time-varying weighting gains. Notice that the controller using time-varying gains produces tighter tolerances around the corners as a result of increasing the gain at specific locations [Simulation].	70
5.10	Tracking performance for the y-axis. Notice the reduction in the error resulting from switching the weighting gains from $(\gamma_Q(k) = 0, 1 - \gamma_Q(k) = 1)$ to $(\gamma_Q(k) = 1, 1 - \gamma_Q(k) = 0)$ [Simulation].	71
5.11	Simulated illustration of a nominal x-axis plant model with two examples of potential resonance shifting in the plant dynamics.	73
5.12	Reference trajectory in which a raster occurs at two distinct locations with the second position corresponding to a location with increased model uncertainty.	74
5.13	Heuristically determined time-varying gains $\sigma_S(k)$. Notice that the gain is increased in the location at which the dynamics are uncertain. The uncertainty in the dynamics leads to an increase in model uncertainty.	75
5.14	Normalized contour errors for systems with position dependent dynamics [Simulation].	76
5.15	Raster trajectory in which the process undergoes intermittent on/off modes. The on/off transitions lead to an external stochastic disturbance that is repeated at predictable intervals.	78
5.16	Time-varying weighting matrix gains $\sigma_R(k)$ for the raster trajectory in Figure 5.15. Notice how the gain increases at the locations corresponding to the on/off modes which have led to the introduction of external stochastic disturbances.	79
5.17	Normalized contour errors for systems with time or position dependent stochastic disturbances. Variances of the converged error signals have been included in the figure. Note that the presence of a large magnitude stochastic disturbance signal results in a reduction in the overall performance [Simulation].	80

5.18	Comparison of RMS contour errors for Feedback, norm optimal control using time-invariant weighting gains and norm optimal control using time-varying weighting gains [Experimental]	82
5.19	Trajectory tracking comparison of norm optimal controllers using time-invariant and time-varying weighting gains. Notice that the controller using time-varying weighting gains produces tighter tolerances around the corners as a result of increasing the gain $\sigma_Q(k)$ at specific locations. [Experimental]	83
6.1	Design methodology used to determine the weighting matrices for the optimal learning controllers.	93
6.2	Frequency response from the x-axis for each individual agent. Y-axis responses are assumed to be similar.	97
6.3	Raster trajectory applied to each 2-DOF agent.	98
6.4	Example formation for a 3 agent coupled MIMO system	100
6.5	Agent 1 x-axis tracking errors for the three tracking cases.	101
6.6	Formation error \mathbf{E}_{d2} for trajectory versus formation tracking.	102
6.7	Height formation errors for the formation tracking case. \mathbf{E}_{h2} is not derived from Agent 1 and therefore converges to much smaller values.	103
7.1	Three-DOF parallel kinematic mechanism used for experimental validation of the coordinated ILC design. The system is a Novint Falcons [3]. A schematic of one of the leg links, l_1 , can be seen in Figure 7.3.	105
7.2	Example group formation of three of the 3-DOF parallel kinematic mechanisms used for experimental validation.	106
7.3	Schematic of an individual leg link for the Falcon presented in Figure 7.1. Note that the leg orientation corresponds to the falcon being placed in a vertical position with the end effector pointing upward. While the example is for the first leg, l_1 , the remaining linkages for this system, as well as the other systems in a group formation, are assumed to be identical.	106
7.4	Single link dynamic model. This image is used to determine a simple, linearized model for each individual leg link.	114
7.5	Bode diagram of the experimental data and the identified system model for the dynamic relationships between $\theta_{1,i,1}$ and the input torque to motors 1-3. Bode plots for the leg links in systems 2 and 3 show similar experimental results and can be seen in Appendix E. The dynamic similarities within the systems enables identical system models to be used for each leg link. . . .	118
8.1	Experimental testbed with three 3-DOF PKM systems. The PKM systems are Novint Falcons. Note the orientation of the falcons. While arbitrary, they have been arranged to create a specific formation. This formation should be maintained during formation tracking with respect to the location of the end effectors.	120
8.2	Raster reference trajectory for experimental testing. Note that the original design is a planar raster in the XY plane.	122

8.3	Raster reference trajectory for experimental testing mapped to the corresponding angle inputs for motors 1-3. The XY cartesian trajectory was mapped to $\theta_{1,m}$ - $\theta_{13,m}$ using the Jacobian matrix in (F).	124
8.4	3-D image of the revised raster reference trajectory for experimental testing. The cartesian inputs were derived by mapping the angle input to cartesian inputs using the MATLAB command interx . Small changes due to the Jacobian mapping matrix resulted in a nonplanar raster trajectory. Note that the z-axis movements are significantly smaller than the x- or y-axis movements.	124
8.5	Spiral reference trajectory for experimental testing. Note that the original design is a planar spiral in the XY plane.	125
8.6	Spiral reference trajectory for experimental testing mapped to the corresponding angle inputs for motors 1-3. The XY cartesian trajectory was mapped to $\theta_{1,m}$ - $\theta_{13,m}$ using the Jacobian matrix in (F).	126
8.7	3-D image of the revised spiral reference trajectory for experimental testing. The modified cartesian inputs were derived by mapping the angle input to cartesian inputs using the MATLAB command interx . Small changes due to the Jacobian mapping matrix resulted in a nonplanar raster trajectory. Note that the z-axis label indicates much smaller vertical movements than in the XY plane.	127
8.8	Block diagram of the implementation scheme used for experimental testing of the system illustrated in Figure 8.1	133
8.9	Normalized x - and y -axis position tracking RMS errors for system 1. Note that focusing on trajectory tracking results in the lowest converged RMS error. Modifying the controller for equal weighting on the two objectives improves the position tracking, resulting in nearly equivalent performance capabilities. Additional weighting combinations may result in decreased tracking performance. Note that there is a tradeoff between decreased RMS formation and trajectory tracking errors.	135
8.10	Normalized E_{d3} and E_{h3} formation tracking RMS errors. Note that focusing on formation tracking results in the lowest converged RMS error. Modifying the controller for equal weighting on the two objectives improves the horizontal formation tracking, but only minimally for this trajectory and these systems. Additional performance benefits can be obtained by weighting the formation objective more heavily. Note, however, that there is a tradeoff between decreased RMS formation and trajectory tracking errors.	136
8.11	Y-axis trajectory tracking RMS errors for systems 1-3. Note that despite different initial RMS values, all three systems result in approximately the same converged RMS tracking error.	137
8.12	XY position tracking for agent 1. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.	138

8.13	Contoured representation of the formation tracking capability of the multi-agent system from Figure 8.1. Desired formation is given in red, while the gray mesh represents the relative area over which the formation tracks for all weighting on the trajectory objective.	139
8.14	Contoured representation of the formation tracking capability of the multi-agent system from Figure 8.1. Desired formation is given in red, while the gray mesh represents the relative area over which the formation tracks for all weighting on the formation objective.	140
8.15	Normalized position tracking RMS errors for system 1. Modifying the controller for dual weighting on the two objectives improves the position tracking performance over the performance using formation control. This results in almost equivalent performance capabilities as compared to the trajectory tracking objective.	141
8.16	Normalized E_{d3} and E_{h3} formation tracking RMS errors. Note that focusing on formation tracking results in the lowest converged RMS error. Applying equal weighting on the two objectives may improve the formation tracking as compared to trajectory tracking, but only minimally for this trajectory and these systems.	142
8.17	Y-axis trajectory tracking RMS errors for systems 1-3. Note that despite different initial RMS values, all three systems result in approximately the same converged RMS tracking error for the spiral trajectory.	143
8.18	XY position tracking for agent 1. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.	144
8.19	Contoured representation of the formation tracking capability of the multi-agent system from Figure 8.1. Desired formation is given in red, while the gray mesh represents the relative area over which the formation tracks for all weighting on the trajectory objective.	145
8.20	Contoured representation of the formation tracking capability of the multi-agent system from Figure 8.1. Desired formation is given in red, while the gray mesh represents the relative area over which the formation tracks for all weighting on the formation objective.	145
8.21	Image of the experimental system with model and force perturbations introduced to system 1.	146
8.22	Normalized x - and y -axis position tracking RMS errors for system 1 when subject to model uncertainty and a force disturbance.	146
8.23	Normalized E_{d3} and E_{h3} formation tracking RMS errors when agent 1 is subject to model uncertainty and force disturbance.	147
8.24	XY position tracking for agent 1 when subject to model uncertainty and a force disturbance. Note that when using a controller focused on the formation objective, the position tracking must be sacrificed in order to ensure accurate formation tracking. Although the tracking varies slightly along the spiral for the trajectory objective control scheme, a more aggressive disturbance signal would results in additional performance degradation. . .	148

8.25	Contoured representation of the formation tracking capability of the multi-agent system when agent 1 is perturbed. Desired formation is in red, the gray mesh represents the relative area over which the formation tracks for a trajectory tracking controller.	149
8.26	Contoured representation of the formation tracking capability of the multi-agent system when agent 1 is perturbed. Desired formation is in red, the gray mesh represents the relative area over which the formation tracks for a formation tracking controller.	149
9.1	Examples of autonomous vehicle applications in which multiple systems perform a coordinated task repetitively. A) Surveying or searching task using multiple autonomous underwater vehicles. B) Coordinated agrosience practices are a critical component of increased efficiency in food production. C) Search and rescue tasks can benefit from repetitive cooperative learning techniques.	153
9.2	Examples of robotic applications in which multiple systems perform a coordinated task repetitively. A) Assembly line robots performing pick & place task. B) Multiple biomedical robots performing a test operation. C) A team of robots collecting information about satellite motions and dynamics using vision sensors (http://robots.mit.edu/projects/jaxa/index.html). . .	154
9.3	Illustration of a task based learning approach. Note that the different tasks may consist of a combination of both individual and group objectives. . .	156
9.4	Illustration of a hierarchical approach to designing coordinated learning controllers for multi-agent systems with a large number of systems. . . .	158
D.1	Single link dynamic model. This image is used to determine a simple, linearized model for each individual leg link.	169
E.1	Bode diagram of the experimental data and the identified system model for the dynamic relationships between $\theta_{1,i,2}$ and the input torque to motors 1-3. The dynamic similarities within the systems enables identical system models to be used for each leg link.	172
E.2	Bode diagram of the experimental data and the identified system model for the dynamic relationships between $\theta_{1,i,3}$ and the input torque to motors 1-3. The dynamic similarities within the systems enables identical system models to be used for each leg link.	173
G.1	Normalized x - and y -axis position tracking RMS errors for system 2. Modifying the controller for equal weighting on the two objectives results in nearly equivalent performance capabilities as compared to trajectory tracking.	176
G.2	Normalized x - and y -axis position tracking RMS errors for system 3. Modifying the controller for equal weighting on the two objectives results in nearly equivalent performance capabilities as compared to trajectory tracking.	177

G.3	Normalized E_{d1} and E_{d2} formation tracking RMS errors. Modifying the controller for equal weighting on the two objectives improves the horizontal formation tracking over the trajectory tracking objective, but only minimally for this trajectory and these systems.	177
G.4	Normalized E_{h1} and E_{h2} formation tracking RMS errors. Note that focusing on formation tracking results in the lowest converged RMS error. Modifying the controller for equal weighting on the two objectives has a very minimal impact on the vertical formation errors for this trajectory and these particular PKM systems.	178
G.5	XY position tracking for agent 2. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.	178
G.6	XY position tracking for agent 3. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.	179
G.7	Normalized x - and y -axis position tracking RMS errors for system 2. . .	180
G.8	Normalized x - and y -axis position tracking RMS errors for system 3. . .	180
G.9	Normalized E_{d1} and E_{d2} formation tracking RMS errors. Modifying the controller for equal weighting on the two objectives does not significantly modify the horizontal formation tracking over the trajectory tracking objective in this case.	181
G.10	Normalized E_{h1} and E_{h2} formation tracking RMS errors. Note that focusing on formation tracking results in the lowest converged RMS error. Modifying the controller for equal weighting on the two objectives has a very minimal impact on the vertical formation errors for this case.	181
G.11	XY position tracking for agent 2. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.	182
G.12	XY position tracking for agent 3. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.	182
G.13	Normalized x - and y -axis position tracking RMS errors for agent 2 when agent 1 is subject to model uncertainty and disturbances.	183
G.14	Normalized x - and y -axis position tracking RMS errors for agent 3 when agent 1 is subject to model uncertainty and disturbances.	184
G.15	Normalized E_{d1} and E_{d2} formation tracking RMS errors when agent 1 is subject to model uncertainty and force disturbance.	184
G.16	Normalized E_{h1} and E_{h2} formation tracking RMS errors when agent 1 is subject to model uncertainty and force disturbance.	185
G.17	XY position tracking for agent 2. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.	185

G.18	XY position tracking for agent 3. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.	186
H.1	Simulink model for simulating the 2D time-varying tracking problem. Note that the model contains several subsystems within the basic block diagram.	197
I.1	Simulink model for simulating the formation tracking problem. Note that the model contains several subsystems within the basic block diagram. . .	212
I.2	Simulink model for deriving the trajectory dependent coupling gains. Note that the model contains a few subsystems within the basic block diagram.	212
I.3	Normalized RMS x- and y-axis position errors for agent 1. Note that the y-axis is logarithmic, while the x-axis is linear.	214
I.4	Normalized RMS x- and y-axis position errors for agent 2. Note that the y-axis is logarithmic, while the x-axis is linear.	215
I.5	Normalized RMS x- and y-axis position errors for agent 3. Note that the y-axis is logarithmic, while the x-axis is linear.	215
I.6	Normalized RMS E_{d3} and E_{h3} formation errors. Note that the y-axis is logarithmic, while the x-axis is linear.	216
I.7	Normalized RMS E_{d1} and E_{d2} formation errors. Note that the y-axis is logarithmic, while the x-axis is linear.	216
I.8	Normalized RMS E_{h1} and E_{h2} formation errors. Note that the y-axis is logarithmic, while the x-axis is linear.	217
J.1	2D trajectory illustrating contour (ε) and individual errors (e_1, e_2) for two individual axes. These errors are defined with respect to the desired position (y_{1r}, y_{2r}) and the actual position (y_1, y_2) of a system defined with respect to the (Axis – 1 , Axis – 2) coordinate frame. Linearized coupling gains ($c_1(k, \theta), c_2(k, \theta)$) at point in time (k) with respect to the tangent angle (θ) can be used to simplify the derivation of the contour error.	223
J.2	Subplot 1: Example complementary sensitivity plot for a dynamically slow and fast system. Note the frequencies of interest lie between the cutoff frequencies of the two systems, respectively. In this frequency range the fast system can compensate for performance limitations from the slower system. Subplot 2: Weighting filters designed using the fast and slow system from subplot 1. Note that W_{fast} is calculated by dividing the complementary sensitivity of the fast system by the complementary sensitivity of the slow system. W_{slow} is set to one to reflect no additional weighting on the slow system.	224

J.3	Block diagram of a two-axis MIMO system in which the two independent SISO axes are coupled together via CCILC. $P_{(\cdot)}$ represents the plant sensitivity function defined as $\frac{G_{(\cdot)}}{1+G_{(\cdot)}k_{(\cdot)}}$, where $G_{(\cdot)}$ is the open-loop axis model and $k_{(\cdot)}$ is a feedback controller used to stabilize the open-loop axis. Note that $P_{(\cdot)}$ is different from the complementary sensitivity function, $T_{(\cdot)}$, defined as $\frac{G_{(\cdot)}k_{(\cdot)}}{1+G_{(\cdot)}k_{(\cdot)}}$. In this example, the fast and slow system descriptions are associated with Axis-1 and Axis-2, respectively.	227
J.4	Multi-axis robotic testbed with extrusion system included. Note that the design example used in this paper only couples together the extrusion system and the y-axis positioning system.	231
J.5	Extrusion system for material deposition	232
J.6	Diagram of the desired fabricated structure and the corresponding reference trajectories. Reference trajectories for the two axes are the desired flowrate, q_r , and desired y-axis position, y_r . Position reference is shown in terms of axis velocity, $v_r(k) = (y_r(k) - y_r(k-1))/0.001$, where 0.001 is the sample time.	233
J.7	Coupling gains used in the derivation of the contour error. Note that the vectors have been filtered using a Gaussian filter with a 3 Hz bandwidth in order to ensure the ability to force compensatory action from the y-axis. . .	234
J.8	Weighting filters used to compensate for the dissimilar dynamics between the positioning stage and the extrusion system. Note that only the fast system requires a weighting filter, W_y , as demonstrated by a weighting filter of 1 for the slow system, W_q	235
J.9	Y-axis output for the Nominal ILC and CCILC cases. Axes coupling forces additional dynamics in the response to compensate for extrusion system inadequacies.	238
J.10	Input signals for the extrusion system for the Nominal ILC and CCILC cases. Despite fairly consistent input signals that are within the input signal constraints, the CCILC controller results in much sharper material start and stop features as illustrated in Fig. J.11	239
J.11	Contour plot of the output signals for the μ RD system, along with experimental images of the start positions (k=1000) for the Nominal ILC and CCILC cases. Note the improved sharpness of the deposited material for the CCILC case. The scale bar is 0.5 mm.	239
J.12	Extrusion trials of the Nominal ILC and CCILC depositing two adjoined lines of material. Adjoining location denoted by white dashed line. CCILC minimizes material overlap by depositing a structure with sharp material starts and stops.	240

List of Abbreviations

ILC	Iterative Learning Control.
SISO	Single Input Single Output.
MIMO	Multiple Input Multiple Output.
CCC	Cross-Coupled Control.
CCILC	Cross-Coupled Iterative Learning Control.
μ -RD	micro-Robotic Deposition.
LTI	Linear Time-Invariant.
LTV	Linear Time-Varying.
RMS	Root Mean Square.
PD	Proportional Derivative.
PKM	Parallel Kinematic Mechanisms.
DOF	Degrees of Freedom.
PID	Proportional-Integral-Derivative
PMC	Precision Motion Control.

List of Symbols

$k = 0, \dots, N - 1$	Discrete time index.
$j = 0, 1, \dots$	Iteration index.
ε	Contour error.
$\mathbf{Q}, \mathbf{S}, \mathbf{R}$	Nominal weighting matrices for norm optimal; typically diagonal.
$\mathbf{C} : c_{(\cdot)}$	Gains defining the coupling of individual signals.
$\mathbf{\Gamma 1}, \mathbf{\Gamma 2} : \gamma_{(\cdot)}$	Gain on individual versus coupled axis signals.
$\mathbf{Q}_i, \mathbf{S}_i, \mathbf{R}_i$	Individual agent weighting matrices.
$\mathbf{B1}, \mathbf{B2} : b_{(\cdot)}$	Gain on individual versus coupled agent tracking.
$\mathbf{K} : \kappa_{(\cdot)}$	Gains defining the coupling of individual agents.
$\mathbf{Q}_{fc}, \mathbf{S}_{fc}, \mathbf{R}_{fc}$	Formation center weighting matrices.
$\mathbf{Q}_{ia}, \mathbf{S}_{ia}, \mathbf{R}_{ia}$	Matrices combining all individual agent matrices.
$\bar{\mathbf{Q}}, \bar{\mathbf{S}}, \bar{\mathbf{R}}$	Matrices for trajectory tracking of MIMO system.
$\mathbf{X1}, \mathbf{X2} : \chi_{(\cdot)}$	Gain on trajectory versus formation tracking.
$\mathbf{F} : f_{(\cdot)}$	Gains mapping the individual position errors to formation errors.
$\mathbf{\Sigma} : \sigma_{(\cdot)}$	Overall gain on $e_{j+1}, u_{j+1}, \delta u_{j+1}$.
$\hat{\mathbf{Q}}, \hat{\mathbf{S}}, \hat{\mathbf{R}}$	Combined matrices used in cost function.

Chapter 1

Introduction

Cooperative behavior defines the coordination of two or more systems directed towards a common goal which is mutually beneficial. The introduction of cooperative behavior is often at the expense of an individual achievement, thereby compromising the individual objective for the benefit of the group. Cooperative behavior has often been observed in nature for group tasks such as foraging [4], transportation [5], and migration [6, 7]. Based on cooperative examples in nature, multi-system coordination has been applied to autonomous vehicles [8], robotics [9, 10], and navigation [11].

Many of these multi-system applications implement a decentralized feedback control approach for coordinated control [12]. Feedback control responds to current events without any anticipatory reaction or response. Contrary to this approach, there are many examples in nature of learned behavior, in which the designated group or individuals incorporate information from past experiences into the present task in order to improve cooperative performance of a designated task [13, 14]. The concept of learning is not unique to nature. A feedforward control approach known as Iterative Learning Control (ILC) has been successfully applied to applications in robotics [15], manufacturing [16], and chemical processing [17]. In these applications, ILC was implemented to minimize the trajectory tracking errors of the system through iterative updates to the control signal. Although there are a few recent papers in which ILC has been applied to the coordination problem of performing a mutually beneficial task such as a group formation [18, 19], the ability to minimize tracking and coordination in a single control framework has not been addressed.

One of the unique advantages of working with multi-agent systems, in which the desired output is coupled in the form of a particular formation, is the ability to focus on the coordination of these agents as a group objective, in addition to the individual objective of trajectory tracking. This research seeks to address the current gap in coordination by developing a novel control methodology for precision coordination and motion control of multiple systems which perform the same task repetitively. While the requirement of repeatability may appear limiting at first, there are many systems in which repeatability plays a crucial role. Examples of multiple systems coupled through a common outcome which perform the same task repetitively can be found in manufacturing [15,20,21], target tracking [22], and agricultural applications such as crop dusting and spraying [23]. The research in this dissertation focuses on applications in manufacturing.

1.1 Manufacturing

There are many things in life that fall into the category of essential elements in life. Some of the more obvious items are food, water, and health. A few less obvious include energy, communication, transportation, security, and manufacturing. These elements are the backbone of any society, ensuring that the community continues to thrive. A critical component of any healthy system is the ability to adapt to changes. For many years the United States led the way in manufacturing. The reality is that things have been changing over the last 20+ years as the traditional manufacturing industry transitions out of the US. One of the driving forces for this transition stems from the ability to significantly reduce production costs for these traditional methods by moving to alternative locations. In order to maintain a leadership position in this field, the focus has turned towards high-value manufacturing which requires a combination of innovation and enhanced capabilities. Innovative manufacturing means

producing parts in novel ways, while enhanced capabilities requires the modification of existing methods to enable complex, high-resolution, high-throughput functionality. High-value manufacturing is particularly important for emerging fields such as biotechnology and nanotechnology.

1.1.1 Emerging Manufacturing Technology

Current trends in the fields of electronics, bioengineering and microelectromechanical systems are leading to increased demands for high-resolution manufacturing capabilities. Electrohydrodynamic jet (E-jet) printing is a technique that uses electric field induced fluid flows through microcapillary nozzles to create devices in the micro/nano-scale range [24]. Because of the ability to print high-resolution ($< 10\mu\text{m}$) droplets and lines with a range of inks, E-jet printing has shown tremendous promise for applications such as printing metallic (Ag) interconnects for printed electronics [25], bio-sensors [24, 26], etc.

The first patented use of E-jet printing came from the Natural Imaging Corporation under US Patent 5838349 claimed by D. H. Choi and I. R. Smith (1998). The printer and printing process detailed in this patent were designed to dispense different colored ink droplets into uniform patterns on a substrate. While these methods easily surpassed the 2-D printing capabilities of ink jet printers at that time, droplet resolution, ink variations, and potential applications for E-jet printing were not fully addressed. In January 2009, the University of Illinois was granted a patent (WO2009/011709) for high-resolution E-jet printing for manufacturing systems. The research detailed in this patent focused on using the E-jet process to print high-resolution patterns or functional devices (e.g. electrical or biological sensors) in the sub-micron range. The patterning of wide ranging classes of inks in diverse geometries, as well as printed examples of functional circuits and sensors demonstrating the diverse applications of E-jet printing are provided in [24]. In addition to a

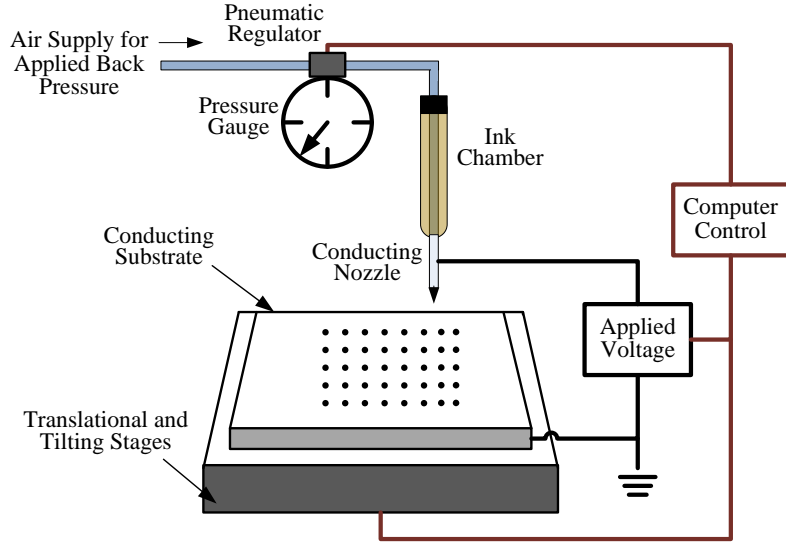


Figure 1.1: Schematic of the E-jet printing set-up including nozzle and ink chamber, air supply for back pressure, conducting substrate, and translation and tilting stages.

wide ranging class of liquids, this process has been used to deposit suspensions containing particulates such as zirconia, DNA, and silver nanoparticles as demonstrated in [26–28]. Along with the ability to print electrical and biological sensors, these suspensions can be used to fabricate 3D structures without supporting material as demonstrated in [29].

Figure 1.1 provides a schematic of the E-jet printing process. The main elements for E-jet printing include an ink chamber, controlled pressure supply, glass nozzle tip, substrate, and positioning system. The printing conditions are controlled through the back pressure (air applied to the nozzle), the offset height from the nozzle tip to the substrate, and the applied voltage potential between a conducting nozzle tip and substrate. Changes in back pressure, stand-off height, and applied voltage affect the size and frequency of the droplets. These changes result in different jetting modes (e.g. pulsating, stable jet, e-spray) which can be used to achieve various printing requirements.

Figure 1.2 presents an example E-jet printed image. For additional details re-

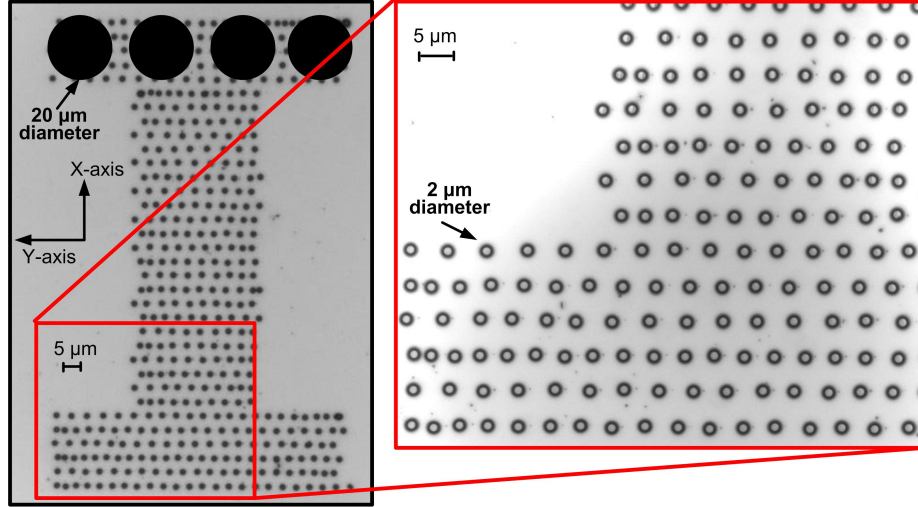


Figure 1.2: Block "I" printed using the desktop E-jet system. Image was printed from a nozzle diameter of $5\text{ }\mu\text{m}$ resulting in printed droplets with an average measured diameter of $2.8\text{ }\mu\text{m}$. Typical ink jet droplets with a $20\text{ }\mu\text{m}$ diameter have been superimposed on the printed image for comparison purposes.

garding the E-jet process or printing system, the readers are referred to [24, 30–33].

1.1.2 E-jet Printing Capabilities

The printing results from Figure 1.2 demonstrate typical E-jet printing capabilities. The natural dynamics of the process, described in the scaling law from [30], determine the jetting frequency and droplet diameter simultaneously. Because the system is sensitive to process variability in offset height and voltage potential, typical printing results contain substantial variability even in the presence of fairly simplistic printing requirements (e.g. printing frequencies of $1 - 5\text{ Hz}$, combined with rastered trajectory following across flat substrates). As the advantages of E-jet printing become more apparent (e.g. the potential for purely additive operations, the ability to directly print biological materials, maskless lithography), enhanced process capabilities such as contoured trajectory following and independent droplet size and delivery frequency become critical. Additionally, as with any manufacturing process, throughput rates

(in this case, printing speeds) and process robustness are key decision parameters in the adoption of the process. Therefore, to fully realize the capability of the E-jet printing process, the following limitations need to be addressed.

- 1) Materials: In order to transition the E-jet process to a printing system in the vein of a bioprinter, the material diversity needs to continue to be explored, particularly biological materials.
- 2) Printing reliability: Relying on the natural pulsating of the ink results in substantial variability in the jetting frequency and droplet diameter.
- 3) Printing speeds: Typical printing frequencies of 1 – 5 Hz have traditionally been used to maintain printing resolution. However, high throughput requires high-speed printing which necessitates a significant increase in printing frequency.
- 4) Contour tracking: Current practice follows a rastered trajectory regardless of the desired printed pattern. Future applications may require contour tracking capabilities, including the ability to print on a contoured surface.
- 5) Coordinated printing: In addition to increasing the printing frequency, the system throughput can be greatly enhanced through coordinated printing of multiple printheads. Furthermore, printing consistency can be improved by synchronizing droplet deposition with the positioning of the tilt and translational stages.
- 6) Repeatability: As with many manufacturing processes, the ability to reliably reproduce a pattern is critical for high yields. Run-to-run control algorithms such as Iterative Learning Control can provide substantial performance improvement if the operating conditions vary repetitively in every run of the process.

Although all of the items listed in 1) – 6) describe important elements in the transition of this process from a research tool into a viable manufacturing process, the

research in this dissertation seeks to address the coupled components, namely contour tracking and coordinated printing. These particular features have been chosen because they represent common manufacturing challenges that exist in many different applications. Additionally, the approach used in this research will capitalize on the repeatability of the process through the use of an Iterative Learning Control scheme.

1.2 Control Objective

The control objectives in this research can be broken down into two main categories with respect to the manufacturing design objectives. In the first scenario, contoured trajectory tracking requires *coupled* movements between two independent axes. For example, consider the E-jet application presented in Subsection 1.1.1 where contour tracking would ensure that the individual x - and y -axes of the positioning system are coupled through a contoured reference trajectory. Now extend this concept to multiple systems performing individual tasks, while simultaneously trying to achieve a defined cooperative objective. Returning to the E-jet example, the x and y positioning system would be *coordinated* with the z -axis printhead in order to maintain a constant offset height between a contoured substrate and the printhead. Note that in this example the printhead defines a separate system from the positioning system. These two examples illustrate the overall concept of controlling multiple systems.

As stated previously, the control framework presented in this research utilizes a learning approach known as Iterative Learning Control [15]. ILC can be combined with existing feedback control schemes in two distinct architectures, parallel and series. Figures ?? and ?? present the two different control architectures. As can be seen in Figure ??, the parallel approach directly alters the control signal to the plant. This architecture may be more intuitive to control designers due to the similarity to other feedforward control approaches in which the feedforward signal adds directly

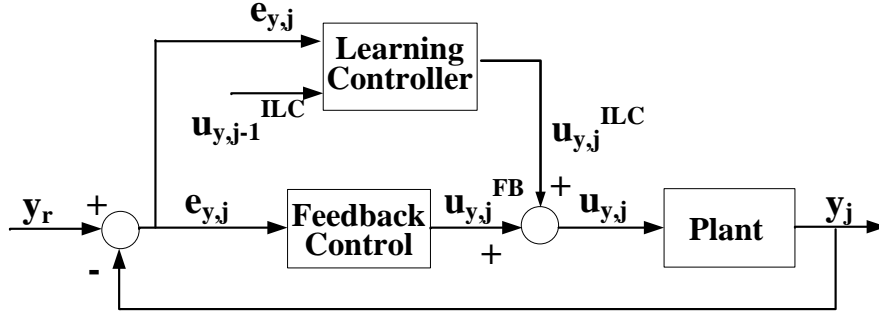


Figure 1.3: Block diagram of the parallel ILC process.

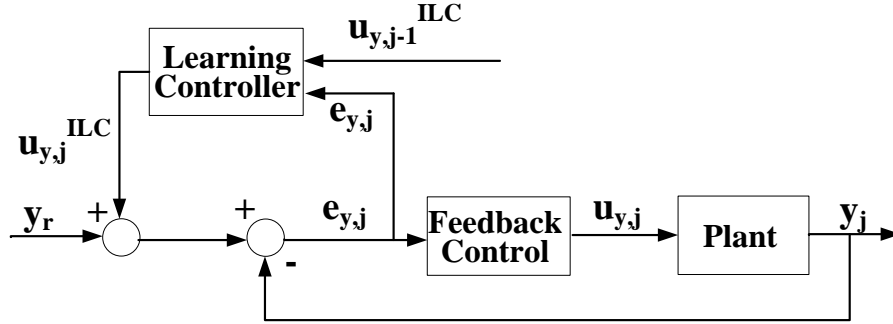


Figure 1.4: Block diagram of the series ILC process.

to the feedback signal for improved tracking performance. In the series design illustrated in Figure ??, the ILC signal directly alters the reference signal to the system. This set-up is particularly useful when adding ILC to preexisting systems with commercial controllers that do not allow access to existing control signals. The desired approach is generally determined based on the individual system configuration. To demonstrate the versatility of the design framework presented in this research, both control architectures will be used in the dissertation.

1.3 Research Objective

The primary objective of the research presented in this dissertation is to introduce a novel control methodology for precision coordination and motion control of multiple systems which perform the same task repetitively. More specifically, this work

will focus on 1) the development of a generalized structure for an iterative learning controller which incorporates a coupled approach to improving the performance of multiple systems, and 2) a description of a design methodology for generating this coupled iterative learning controller. The remainder of the dissertation is as follows. Chapter 2 provides a brief literature review of work related to this research. Chapter 3 introduces Iterative Learning Control including the formulation of a typical cost function and guidelines for tuning the weighting matrices for the norm optimal framework. Although a brief discussion of both the frequency and time domain ILC formats will be provided, the work in this dissertation focuses on the time domain norm optimal approach. A description of a few techniques for coupling multiple systems is provided in Chapter 4. Chapters 5 and 6 present the novel norm optimal framework and design methodology which will be used to develop coupled and coordinated learning controllers. Simulation and experimental results for learning controllers designed to satisfy varying design objectives will also be given in these chapters. Chapter 7 discusses the application of these novel control approaches to parallel kinematic mechanisms (PKM), including a system description and dynamic model. Chapter 8 presents experimental results from implementing the coordinated learning controller on an experimental setup using the PKM systems described in Chapter 7. Chapter 9 provides concluding remarks and a brief discussion on additional applications of these control schemes.

Chapter 2

Traditional Control Approaches - A Literature Review

The research in this dissertation focuses on the development of a method for improving the precision coordination and motion control of MIMO systems that execute the same task repetitively. A norm optimal framework is used to design optimal learning operators based on specific design objectives. This framework provides a straightforward approach for adapting traditional coupling and coordination approaches into novel learning control strategies. Traditional control approaches for the contour and coordination problems described in Section 1.2 can be defined in terms of a dimensional analysis approach illustrated in Figure 2.1 (note that these are not spatial dimensions).

The 1-D problem, with time along the horizontal axis, refers to a single multi-axis system that executes a task one time. While traditional feedback control schemes can be applied, the presence of a multi-axis system enables the use of a specific feedback control strategy known as cross-coupled control (CCC). There are two 2-D design problems requiring very different control approaches. The 2-D problem considering iteration and time refers to a multi-axis system that executes a single task multiple times. Iterative learning control is a natural control design for these systems. The 2-D scenario with multiple systems performing a task one time falls into the category of shape or formation tracking problems. Lastly, the 3-D problem is a combination of the two 2-D cases, in which the three axes are time, iteration and number of individual systems. Traditional control strategies for addressing these multidimensional systems are discussed briefly in the following sections.

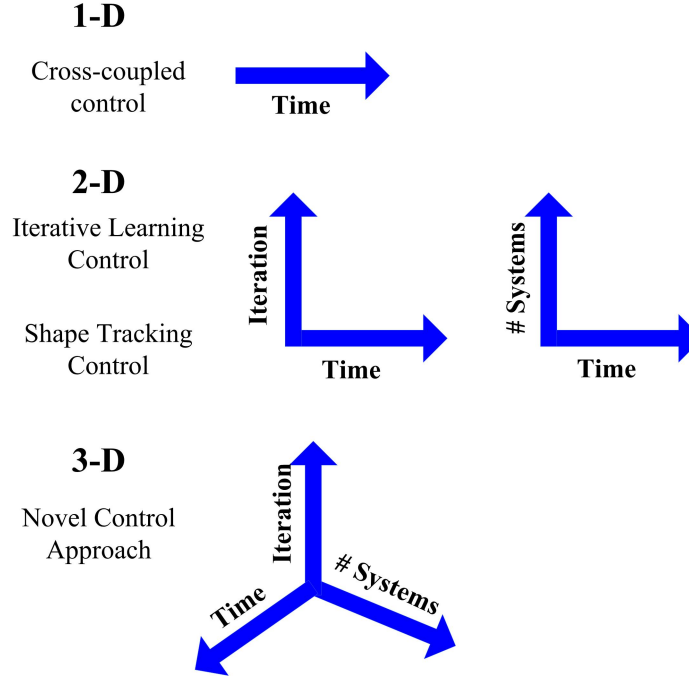


Figure 2.1: Traditional control approaches in terms of dimensional analysis. Note that the axes describe the degrees of freedom for the different approaches.

2.1 1-D Control Strategies

This section presents a brief overview of feedback cross-coupled control (CCC). CCC is a control technique that focuses on improving the contour tracking of multi-axis systems by coupling the individual axis tracking errors of single-input single-output (SISO) systems together and applying a controller to the combined signal. The control input is then fed back into the individual axes, thereby making the control into a single axis dependent on the performance of the other axes. This technique is particularly useful for applications in which the ability to track a desired path is more important than being in a specific location within the system's state space at a specified point in time.

Figure 2.2 provides an example of a multi-axis system in which the ability to track a desired trajectory or path is critical for the successful completion of the desired task. The CNC process depicted in this image shows a common process in which the



Figure 2.2: Image of a CNC process. Note that coupling the movements of the XYZ axes may result in better path following and the construction of a more precise part.

coupled movement of the individual XYZ axes may ensure accurate patterning.

CCC was originally introduced by Koren around 1980 [34]. In this paper, he introduced the notion of defining contour error as a function of multiple individual error signals, applying a controller to this combined error signal, and feeding the new signal back into the respective systems. His work was followed by a few key papers from Kulkarni and Srinivasan [35], [36]. In [36], a new CCC design was introduced which separated the contour error into two distinct signals for the x - and y -axis, respectively. The following year Koren published an article describing a new variable gain CCC design [1]. In this modified design, a single controller was applied to the contour error and then the updated signal was decomposed into two axial components by multiplying the signal by terms known as coupling gains. This additional step ensured that the contour error correction was executed in the proper direction. Figure 2.3 provides a block diagram of the classic feedback CCC design introduced in [1].

The main disadvantage of this approach comes from the derivation of the contour

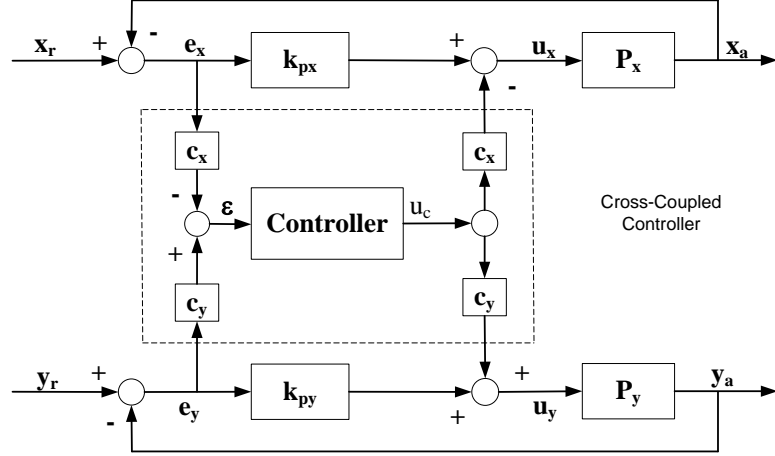


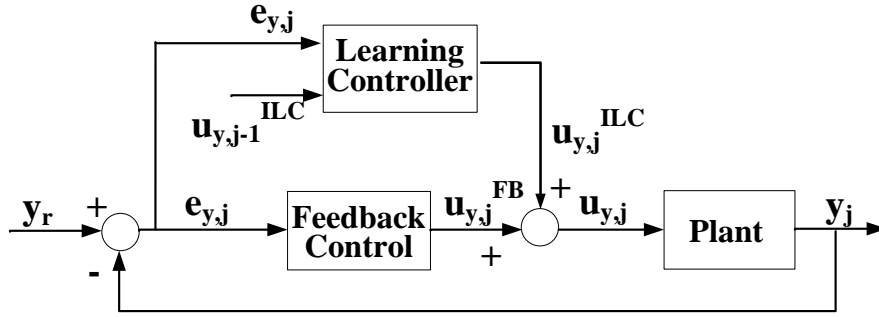
Figure 2.3: Koren Variable Gain CCC Design [1]

error. The mapping from the individual axis errors to the contour error is trajectory dependent. Therefore, the more complex the trajectory is, the more complicated the mapping becomes. A means for simplifying this process was developed in [2] and is described briefly in Subsection 2.2.2.

2.2 2-D Control Strategies

2.2.1 Iterative Learning Control

Iterative learning control (ILC) is a performance enhancing control technique that is implemented on systems which repeat the same task. When a system executes the same task multiple times, the controller is able to optimize a feedforward control signal in order to minimize the time domain error signals. This approach has been shown to be successful in several manufacturing applications [16], with performance improvements commonly reported to be several orders of magnitude, measured by root mean square (RMS) or maximum error, as compared to those systems' feedback controllers. A basic block diagram representation of the parallel ILC control architecture was presented in Figure ?? and is reprinted here. Note that j represents the



Block diagram of the parallel ILC process.

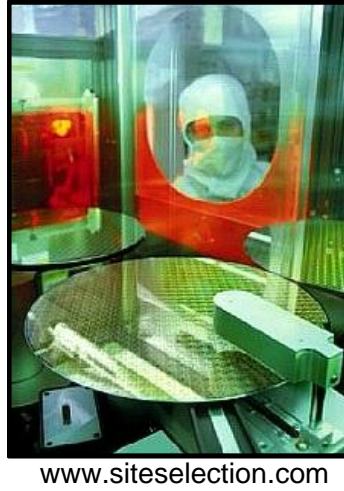


Figure 2.4: Image of semi-conductor fabrication in which wafer scanning is an important requirement. This is an example of a 2-D application in which the repetitive nature of the process enables the use of learning control schemes such as Iterative Learning Control.

iteration number.

Figure 2.4 shows an image of a semi-conductor fabrication process. Like many manufacturing processes, semi-conductor fabrication requires the use of repetitive motions in the form of wafer scanning. The tracking performance of this type of motion can be improved through the use of learning techniques such as Iterative Learning Control.

Although ILC was originally developed for robotics [15], it has been shown to improve the performance of a much broader range of manufacturing and chemical systems. Examples included CNC machine tools, wafer stage motion, injection mold-

ing machines, aluminum extruders, cold rolling mills, induction motors, rapid thermal processing, and semibatch chemical reactors (see references in [37–39]). As potential nontraditional applications for ILC have become more apparent (e.g. training mechanism for open-loop systems [40], identification procedure for system dynamics [41], non-identical repetitive applications [42]), research in the area has continued to increase. Many of the current key investigators, along with their respective areas of focus, are listed in Figure 2.10.

Chapter 3 provides an overview of ILC, focusing on the key elements and approaches utilized in this research. For more information, the interested reader is referred to the many books and surveys on ILC [37–39, 43–46].

2.2.2 Cross-Coupled ILC

In this subsection, a specific form of multi-input multi-output (MIMO) ILC is presented. Cross-coupled ILC (CCILC) combines feedback CCC with ILC into a learning control design which focuses on minimizing the contour tracking of a given system. Contoured trajectories require coordinated positioning between two or more axes. In these systems, shifting the focus from individual axis tracking to contour tracking may result in a final outcome that more closely resembles the desired trajectory. Note that in order to place more emphasis on contour tracking, enhanced performance for the individual axis tracking may need to be compromised.

In Figure 2.5, a multi-axis positioning system is combined with an extrusion system for micro-Robotic Deposition (μ -RD). The objective of this process is to fabricate 3-dimensional parts for a range of applications such as bone scaffolding [47, 48].

The general ILC approach for precision motion control (PMC) of contoured trajectories on multi-axis repetitive systems is to implement individual ILC controllers on each axis. These controllers are designed to minimize individual axis errors. This approach works well for trajectories that lie within the bandwidth of the individual

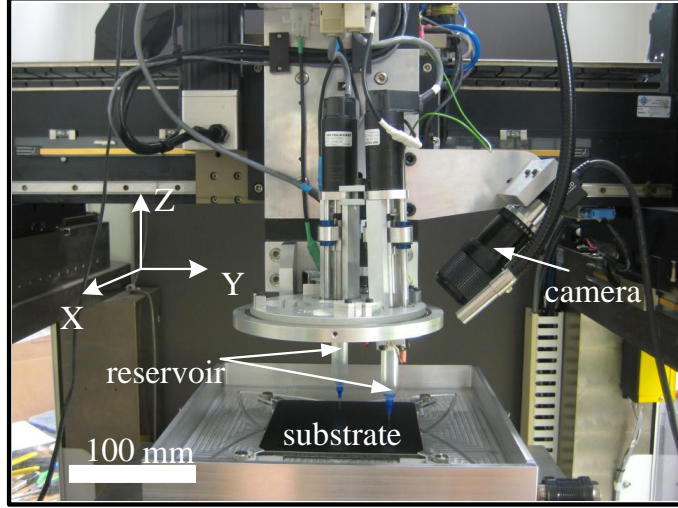


Figure 2.5: Image of a μ -Robotic Deposition system used for contoured applications such as the fabrication of bone scaffolding. The repetitive patterns and functional tasks combined with the contoured patterns associated with this process make it well suited for Cross-Coupled Iterative Learning Control.

systems. However, for trajectories that contain frequency content outside the bandwidth of the individual systems, the performance of the multi-axis system may begin to degrade. Previous work in combining servomechanism control and contour control for two-axis systems required the use of feedback CCC [49, 50].

In [2, 51], a novel ILC control design which enables the control designer to focus on contour tracking was introduced. [2] presented a frequency based combined CCILC and ILC control scheme. The CCILC design focused on contour tracking, while the ILC design focused on individual tracking errors. One of the challenges in combining the two learning controllers emerged when trying to emphasize one learning design over the other. The optimal learning controller depends on several factors, such as the system requirements, the reference trajectory, and the design objectives. The goal of the work in [51] was to reformat the combined learning controller into a norm optimal framework. The norm optimal framework provides a standard approach for focusing on contour tracking, which results in a more intuitive design approach. The weighting approach of this framework also enabled one to focus on individual axis

tracking, contour tracking, or a combination of the two as determined by the control designer.

Figure 2.6 illustrates a multi-axis system with combined CCILC and ILC controllers. Note that the norm optimal framework utilized in [51] enabled the control designer to choose from either an ILC design, CCILC design, or some combination of the two.

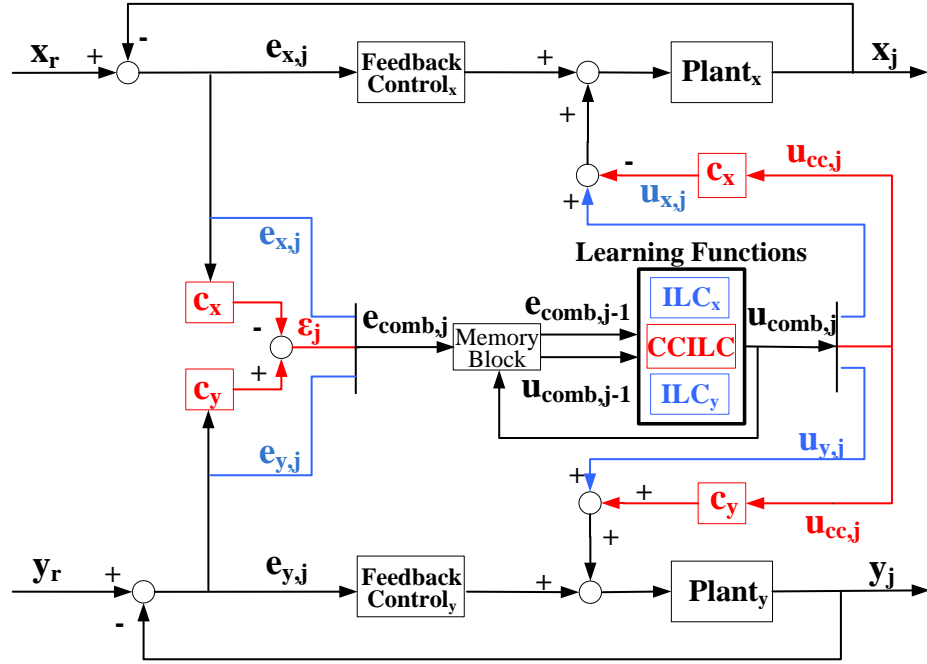


Figure 2.6: Block diagram of the combined CCILC and ILC design process.

Lastly, [52] provided a generalized norm optimal framework and design methodology for coupling the individual axis signals of a multi-axis system for individual or coupled control design with respect to the error signals, the control signals, and the change in control signals. The framework presented in this paper was an extension of previous work from the authors detailed in [51]. The time-varying weighting matrix design presented in [52] enables the controller to take trajectory, position-dependent dynamics, and time-varying stochastic disturbances into consideration when designing an optimal learning controller.

2.2.3 Shape Tracking

The final section on 2-D control techniques focuses on coordinating multiple systems in order to maintain a desired group formation or shape. The formation-tracking approach has been inspired by observations from nature such as flocking [7], schooling [53], and foraging [4]. In nature, the studied behavior often incorporates multiple objectives such as maintaining a desired velocity, while simultaneously performing a given task (i.e. foraging) and maintaining a certain spatial orientation with neighboring systems [53]. Extending the concept of multiple objectives, researchers have expanded the notion of multi-system coordination to a variety of applications.

The notion of a multi-objective design approach is very common in coordination control, although the objectives differ depending on the application. Many of the applications involve the dual objectives of target tracking and formation keeping [54–56], as well as navigation and collision or hazard avoidance [11, 57].

Figure 2.7 presents two common examples of shape or formation tracking. As can be seen from these images, the close proximity of the other agents within the multi-agent system ensures that cooperative maneuvering is critical to the successful execution of a desired task. While airplane formations can rely on sight as well as vocal communications to maintain formation, schools of fish must follow more subtle means of communication such as flow patterns and the slight shifting of neighboring fish [58].

One of the central concepts in formation tracking is the method of communication amongst the individual systems. The mode of communication, as well as the internal structure for how this information is used and disseminated amongst the group is highly dependent on the system and the application. Taking the concepts from nature, Balch and Arkin [11] introduced the now standard behavior-based terminology: unit-center (or formation-center) reference, leader-reference, and neighbor-reference

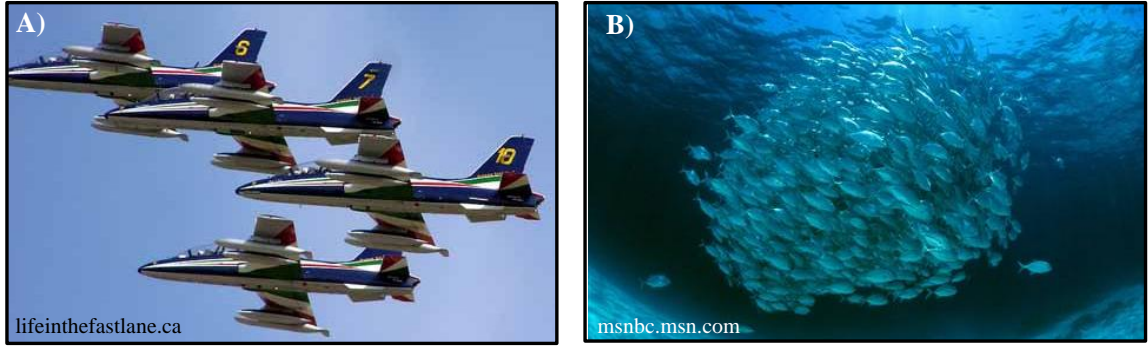


Figure 2.7: Examples of 2-D cooperative tracking. A) Formation of planes illustrating coordinated flying patterns. B) A school of fish displaying cooperative maneuvering.

formations. Figure 2.8 presents a depiction of the three formation approaches. Detailed descriptions of these approaches are provided in Chapter 4 Section 4.2.

While these formation methods are still commonly used in practice today, additional methods began appearing in recent years. In 2007, Porfiri et al [54] introduced a virtual leader to help compensate for systems with potentially limited or sparse communication. Morgan and Schwartz [59] decomposed a large swarm problem into multiple subswarm problems in which the subsets function independently, applying their desired reference formations individually, while simultaneously ensuring collision avoidance with the other subsets. D’Andrea and Dullerud [12] applied a decentralized control approach to a similar, yet slightly modified problem in which the individual systems were interconnected. A review of some standard strategies and challenges in formation control can be found in [60].

2.3 3-D Control Strategies

This last section addresses what has been identified as a current gap in the literature. The 3-D problem combines all of the elements from the 2-D examples into a single design problem. By combining the approaches described in the previous sections, this dissertation work focuses on the development of a novel approach for controlling

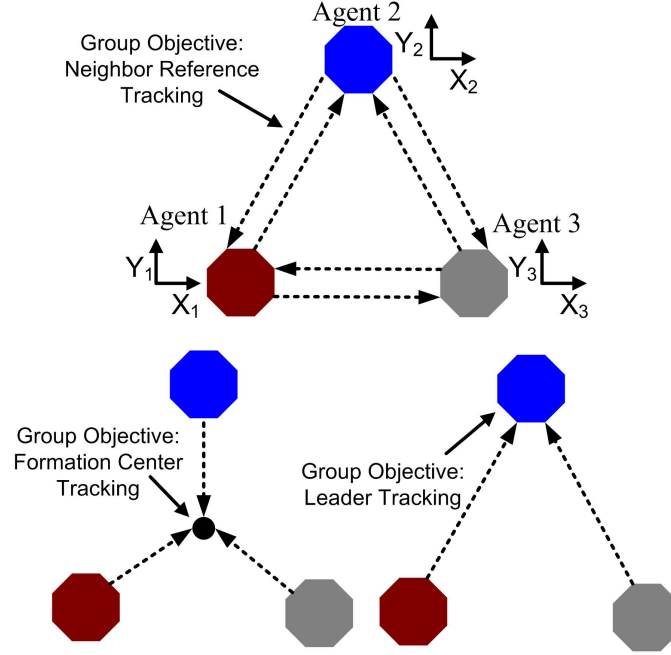


Figure 2.8: Formation tracking techniques described in terms of a group objective: formation-center reference, leader-reference, and neighbor-reference. Note that in all three cases, the individual systems maintain their own objectives independently from the group objective.

multiple systems that execute a task repetitively.

As mentioned in Subsection 2.2.3, there are several different types of applications that require coordinated movements from multiple systems in order to achieve a specific task. In many of these applications, the performance of the process would benefit from utilizing the repetitive nature of the task or application. Figure 2.9 presents examples of 3-D applications in which a multi-agent system performs a task repetitively. These examples stem from a broad range of areas including manufacturing, agricultural, and search and rescue applications.

There has been some initial work in using ILC to improve the performance of multi-agent systems. Wu et al [18] combined a formation tracking feedback controller with a trajectory tracking learning controller for coordinated control of satellites. Ahn and Chen [19] introduced the use of ILC for off-line control tuning to achieve relative formations amongst multiple agents. This approach showed simulations with

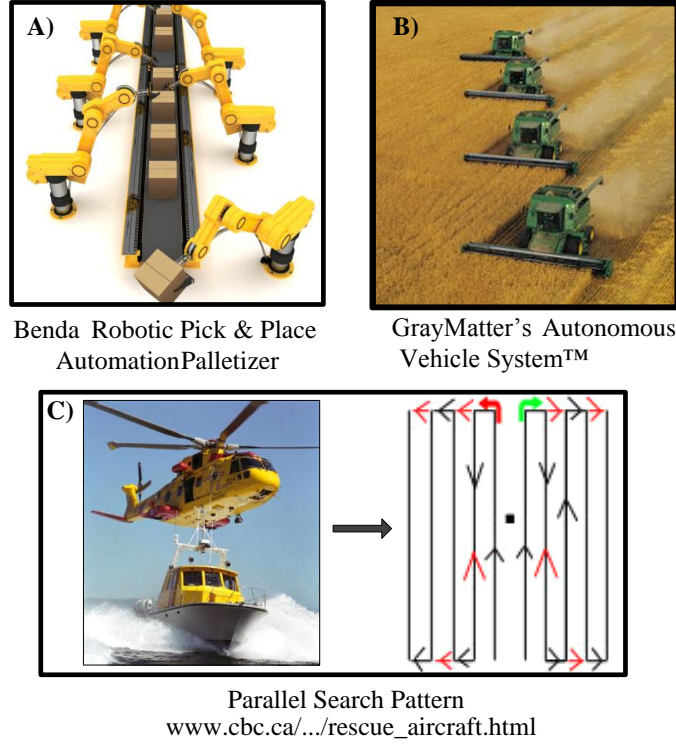


Figure 2.9: Examples of 3-D applications in which multiple systems perform a coordinated task repetitively. A) Common manufacturing application, known as pick n' place. B) Coordinated agrosience practices are a critical component of increased efficiency in food production. C) Search and rescue tasks can benefit from cooperative learning techniques.

large transient growth followed by individual trajectory tracking convergence (ergo formation tracking convergence) in the iteration domain.

One limitation from these two approaches is the inability to design the trajectory and formation tracking control schemes simultaneously. One of the advantages of the proposed control approach presented in this work is a framework and design methodology which enables the control designer to emphasize trajectory tracking, formation tracking, or a combination of the two in a single controller. This approach is the first control framework to not only combine the design for individual trajectory and formation tracking control, but to incorporate the ability to determine the formation tracking approach (e.g. formation-center, leader, or neighbor-reference) through the selection of different gains within the design.

Each of the different control approaches presented in this chapter, contain certain

desirable qualities. These qualities have been listed in Figure 2.10 and paired with specific control schemes. As can be seen in Figure 2.10, the proposed control scheme addresses all of the key elements.

	Contour Error	Coordi nation	Learning	Simple Design	Flexible Design	Modular Design
Major Contributors: Cross-Coupled Control 1. Y. Koren – original design, variable gain CCC 2. Chuang and Liu – adaptive CCC 3. Kulkarni and Srinivasan – servomechanisms 4. Yeh and Hsu – biaxial CCC 5. Tang and Landers – hierarchical optimal CCC 6. Chen, Tilbury and Ulsoy 7. Chiu and Tomizuka – feed drive systems	X					
Major Contributors: Iterative Learning Control 1. Arimoto – original design 2. Amann, Owens, de Roover, Steinbuch, Bosgra – norm optimal ILC 3. Norrlof and Gunnarsson – convergence property 4. Bien and Xu – overview book on ILC 5. Rogers – 2-D analysis, reference input design 6. Longman – stability and robustness, transients 7. Moore and Chen – review book on ILC 8. Bristow and Alleyne – frequency domain, Q-filter designs 9. Saab – stochastic ILC 10. Xu and Tan – nonlinear ILC 11. Tomizuka and Tsao – ILC for wafer scanners			X	X		
Major Contributors: Formation Tracking 1. Dullerud and D’Andrea – distributed control of heterogeneous systems 2. Morgansen and Tsukamaki – collision avoidance for unicycle-type vehicles, target tracking 3. Porfiir, Roberson, Stilwell – formation control 4. Balch and Arkin – behavior based formation 5. Morgan and Schwartz – coordinated control laws 6. Chen and Wang – survey of formation control 7. Lee – Teleoperation over the internet		X				X
Major Contributors: Formation ILC 1. Ahn and Chen – leader/follower coordination 2. Wu, Poh, and Xu – satellite formation tracking		X	X			X
Major Contributors: Coupled & Coordinated ILC 1. Kira Barton	X	X	X	X	X	X

Figure 2.10: Figure identifying the key elements in a coordinated learning controller. The elements are associated with specific control strategies as indicated by the large 'X'. Note that the proposed control scheme addresses all of the key elements.

Chapter 3

Iterative Learning Control

The research in this dissertation focuses on the development of a method for improving the precision coordination and control of MIMO systems that execute the same task repetitively. The repetitive nature of these systems enables a controller to *learn* from previous iterations and modify the control input for improved tracking performance [61]. Iterative learning control (ILC) is an adaptive feedforward (with respect to the time domain) control method which minimized tracking errors through iterative updates to the control signal [37, 44]. The learning process converges anywhere from a few to tens of iterations, depending on the design of the algorithm. The learning process is illustrated in Figure 3.1.

The essential caveat in ILC is system repeatability, meaning that multiple repetitions of the trajectory or task yield nearly identical error signals. Although this may appear to be very restrictive, many practical systems are highly repetitive. This is particularly true in manufacturing systems, as will be discussed shortly. Along with the stipulation of repeatability, there are additional system requirements that must be met in order to utilize an iterative learning scheme:

- A1) The reference operation must be discontinuous.
- A2) Initial conditions on each iteration are identical.
- A3) Any external disturbance repeats identically on each iteration.
- A4) The system has a stable or stabilizable plant.

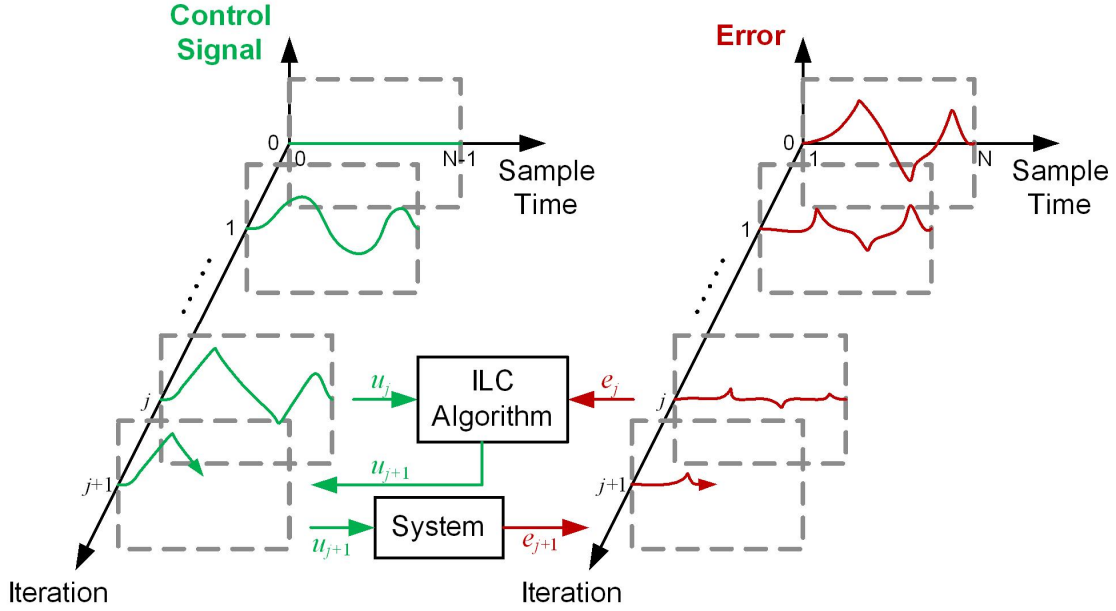


Figure 3.1: ILC process. As the number of iterations increases, the feedforward time domain control signal is determined and the error signal is minimized

The first assumption requires the desired reference trajectory to contain discrete start and stop actions. This provides the system with a finite length reference signal that can be repeated. Although assumptions A2 and A3 are rarely strictly true in practice, iteration-to-iteration variation has a major impact on system performance. This is discussed in more detail in Subsection 3.3.3. The last assumption guarantees stability in the time domain, although it says nothing about stability in the iteration domain.

This chapter will focus on summarizing the main ideas in ILC, specifically those ideas associated with the time domain design approach, and presenting convergence and performance analysis for weighting matrix design in the norm optimal framework. The remainder of this chapter is as follows. The class of systems addressed in this research, along with a basic ILC algorithm, are introduced in Section 3.1. Two design frameworks are presented in this Chapter. The first framework is frequency domain design which will be covered briefly in Section 3.2. Section 3.3 will contain a more detailed presentation of the time domain design which focuses on the norm

optimal framework, as well as convergence and performance analysis results. Section 3.4 provides simulation results illustrating the norm optimal weighting matrix design approach.

3.1 System Setup

In this paper we consider linear, causal, discrete-time MIMO systems, P , given as

$$P \triangleq \begin{cases} x_j(k+1) = A(k)x_j(k) + B(k)u_j(k) \\ \hat{y}_j(k) = C(k)x_j(k) + D(k)u_j(k), \end{cases} \quad (3.1)$$

$$y_j(k) = \hat{y}_j(k) + y_o(k) + d_j(k) \quad (3.2)$$

where $k = 0, 1, \dots, N-1$ is the discrete time index, $j = 0, 1, \dots$ is the iteration index, $u_j(k) \in \mathbb{R}^{q_i}$ is the control, $y_j(k) \in \mathbb{R}^{q_o}$ is the output, $y_o(k) \in \mathbb{R}^{q_o}$ is iteration-invariant, $d_j(k) \in \mathbb{R}^{q_o}$ corresponds to stochastic (iteration-varying) external disturbances, $x_j(k) \in \mathbb{R}^n$ are system states, and $(A(k), B(k), C(k), D(k))$ are appropriately sized iteration-invariant real-valued matrices. It is assumed that $x_j(0) = x_o$ for all j , and note that $y_o(k)$ can be used to capture iteration-invariant initial-condition responses [61], feedback control [62], and external disturbances. As illustrated by the matrices, $(A(k), B(k), C(k), D(k))$, P is defined as time-varying over a single profile, but iteration-invariant from trial-to-trial. In the lifted-domain [63, 64], the discrete-time behavior of the system is represented by its convolution matrix \mathbf{P} using impulse response data $H_{i,j}(k)$, (J.3).

$$\mathbf{P} = \begin{bmatrix} H_{0,0} & & 0 \\ \vdots & \ddots & \\ H_{N-1,0} & \cdots & H_{N-1,N-1} \end{bmatrix} \quad (3.3)$$

For MIMO linear time-varying (LTV) systems, $H_{i,j}(k)$ contains the impulse response from each of the q_i inputs to each of the q_o outputs and can be derived using the time-varying matrices in (J.1),

$$H_{i,j} : \begin{cases} D(i), & i = j \\ C(i)A(i-1)A(i-2)\dots A(j+1)B(j), & i > j. \end{cases} \quad (3.4)$$

Given $H_{i,j}(k) \in \mathbb{R}^{q_o \times q_i}$, system $\mathbf{P} \in \mathbb{R}^{Nq_o \times Nq_i}$ is a lower triangular matrix with a block Toeplitz structure. While the approach presented in this research is for LTV systems, the same design process can be applied to linear time-invariant (LTI) systems. In the case of LTI systems, $H_{i,j}$ is of the form,

$$H_{i,j} : \begin{cases} D, & i = j \\ CA^{i-j-1}B, & i > j. \end{cases} \quad (3.5)$$

Many system models contain some form of model uncertainty. To address this in the controller design, assume that the true system \mathbf{P}_t corresponds to the nominal model \mathbf{P} plus an uncertainty Δ_P : $\mathbf{P}_t = \mathbf{P}(I + \Delta_P)$, with the multiplicative uncertainty $\Delta_P = \mathbf{W}\Delta$ and $\|\Delta\|_{i2} \leq 1$.

During trial j , system \mathbf{P}_t maps the input signal u_j to the measured output signal y_j , i.e., $\mathbf{y}_j = \mathbf{P}_t \mathbf{u}_j + \mathbf{d}_j$, with \mathbf{u}_j , \mathbf{y}_j , and \mathbf{d}_j defined in (3.6), (3.7), and (3.8)

respectively.

$$\mathbf{u}_j = \begin{bmatrix} u_j^T(0) & u_j^T(1) & \cdots & u_j^T(N-1) \end{bmatrix}^T \quad (3.6)$$

$$\mathbf{y}_j = \begin{bmatrix} y_j^T(0) & y_j^T(1) & \cdots & y_j^T(N-1) \end{bmatrix}^T \quad (3.7)$$

$$\mathbf{d}_j = \begin{bmatrix} d_j^T(0) & d_j^T(1) & \cdots & d_j^T(N-1) \end{bmatrix}^T, \quad (3.8)$$

$$\begin{aligned} \text{with } u_j^T(k) &= \begin{bmatrix} u_j^1(k) & \cdots & u_j^{q_i}(k) \end{bmatrix} \\ y_j^T(k) &= \begin{bmatrix} y_j^1(k) & \cdots & y_j^{q_o}(k) \end{bmatrix} \\ \text{and } d_j^T(k) &= \begin{bmatrix} d_j^1(k) & \cdots & d_j^{q_o}(k) \end{bmatrix}. \end{aligned}$$

Here we adopt a widely used norm optimal ILC update law [64, 65]

$$\mathbf{u}_{j+1} = \mathbf{L}_u \mathbf{u}_j + \mathbf{L}_e \mathbf{e}_j \quad (3.9)$$

with

$$\mathbf{e}_j = \mathbf{y}_r - \mathbf{y}_j, \quad (3.10)$$

where \mathbf{y}_r is the reference signal and is assumed iteration invariant. In (3.9), \mathbf{L}_u and \mathbf{L}_e are solutions to a quadratic optimization problem detailed shortly in Section 3.3. These lifted matrices are generally non-causal [66], time-invariant linear operators on the control and error signals, respectively. The non-causality leads to full lifted matrices rather than the lower triangular toeplitz form of the system \mathbf{P} . For the purpose of exposition, (3.9) can be rewritten as:

$$\begin{aligned} \mathbf{u}_{j+1} &= \mathbf{L}_u \mathbf{u}_j + \mathbf{L}_e \mathbf{e}_j = \mathbf{Q}(\mathbf{u}_j + \mathbf{L}_e \mathbf{e}_j) \\ \text{where } \mathbf{L}_u &= \mathbf{Q}, \mathbf{L}_e = \mathbf{Q}\mathbf{L}. \end{aligned} \quad (3.11)$$

The update law presented in (3.11) is analogous to the common frequency domain algorithm of the form,

$$u_{j+1} = Q(z)(u_j(k) + L(z)e_j(k)) \quad (3.12)$$

where the mixture of time-domain signal and frequency-domain filters is a standard abuse of notation. The Q -filter is a lowpass filter used to limit the learning bandwidth to provide robustness to the system. The L -filter, also known as the learning filter, is designed to maximize the learnable bandwidth and convergence rate.

Although the lifted representation from (3.11) is very useful in design and analysis, it is not necessary, or often times desirable, to use the lifted representation in implementation because \mathbf{L}_u and \mathbf{L}_e (or \mathbf{Q} and \mathbf{L}) are large matrices depending on the iteration length, the sampling rate, and the number of individual systems within the overall design. The benefits and challenges associated with each design framework are discussed in the following sections.

3.2 Frequency Domain Design

The frequency domain design framework uses classical analysis tools such as Nyquist and Bode plots to design learning controllers [67]. For many controls engineers, these tools provide a more intuitive approach to designing controllers for practical use. The key assumption in this framework is that the iteration is infinite. This is impractical in real world applications, resulting in an estimation of performance. While this is an approximation, it enables the use of convergence and performance analysis techniques, such as the use of the z-transform, which simplify the design approach. An additional advantage to the frequency domain approach is that many conventional learning filters use tunable designs such as proportional and proportional-derivative

controllers; thereby requiring very little a priori knowledge of the system.

There have been numerous papers and books which focus on design and analysis techniques using the frequency domain framework. The research in this dissertation considers time-varying systems and therefore is focused on time-domain design, which is the natural framework for time-varying systems. For further information regarding frequency domain design the interested reader is referred to [37,68] and the references therein.

3.3 Time Domain Design

In the previous section, a norm optimal ILC control algorithm was presented in (3.9). This controller results from a quadratic optimization problem, [69]. In this problem, we want to minimize an objective \mathcal{J} , with \mathcal{J} corresponding to the sum of weighted norms of the error $\|\mathbf{e}_{j+1}\|_Q$, the command signal $\|\mathbf{u}_{j+1}\|_S$, and the rate of change of the command signal $\|\mathbf{u}_{j+1} - \mathbf{u}_j\|_R$, as shown in (3.13).

$$\mathcal{J} = \mathbf{e}_{j+1}^T \mathbf{Q} \mathbf{e}_{j+1} + \mathbf{u}_{j+1}^T \mathbf{S} \mathbf{u}_{j+1} + (\mathbf{u}_{j+1} - \mathbf{u}_j)^T \mathbf{R} (\mathbf{u}_{j+1} - \mathbf{u}_j). \quad (3.13)$$

$(\mathbf{Q}, \mathbf{R}, \mathbf{S})$ are symmetric positive definite matrices with a common form given as $(\mathbf{Q}, \mathbf{R}, \mathbf{S}) \triangleq (q\mathbf{I}, r\mathbf{I}, s\mathbf{I})$. Note that in some cases $(\mathbf{Q}, \mathbf{R}, \mathbf{S})$ may be positive semi-definite matrices, as long as $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R}$ is positive definite.

Applying the substitution $\mathbf{e}_{j+1} = \mathbf{e}_j - \mathbf{P}(\mathbf{u}_{j+1} - \mathbf{u}_j)$ and differentiating \mathcal{J} with respect to \mathbf{u}_{j+1} results in the following relationship.

$$\frac{\partial \mathcal{J}}{\partial \mathbf{u}_{j+1}} = \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{u}_{j+1} + \mathbf{S} \mathbf{u}_{j+1} + \mathbf{R} \mathbf{u}_{j+1} - \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{u}_j - \mathbf{P}^T \mathbf{Q} \mathbf{e}_j - \mathbf{R} \mathbf{u}_j. \quad (3.14)$$

Setting the derivative from (3.14) equal to zero and rearranging the solution, yields the norm optimal ILC controller from (3.9) with respect to the weighting matrices

$(\mathbf{Q}, \mathbf{R}, \mathbf{S})$ and the plant \mathbf{P} ,

$$\begin{aligned}\mathbf{u}_{j+1} &= \mathbf{L}_u \mathbf{u}_j + \mathbf{L}_e \mathbf{e}_j \\ \mathbf{L}_u &= (\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R})^{-1} (\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{R}) \\ \mathbf{L}_e &= (\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R})^{-1} \mathbf{P}^T \mathbf{Q}.\end{aligned}\tag{3.15}$$

Note that for $(\mathbf{L}_u, \mathbf{L}_e)$ to ensure convergence we require $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R}$ to be positive definite, as will be shown in Section 3.3.1.

Although this ILC control strategy is relatively well known [65], there has been relatively little documentation on how to tune $(\mathbf{Q}, \mathbf{R}, \mathbf{S})$ [70,71]. Therefore, the following sections derive tuning guidelines by studying the properties of the ILC controlled system with respect to nominal convergence, robust convergence, and performance.

3.3.1 Nominal Convergence

In the following subsection, nominal convergence is explored for the nominal plant model \mathbf{P} . Given the ILC controller (3.15) and the system dynamics $\mathbf{y}_j = \mathbf{P} \mathbf{u}_j$ (with $\mathbf{y}_o = 0, \mathbf{d}_j = 0$), the trial domain dynamics can be given by

$$\mathbf{u}_{j+1} = (\mathbf{L}_u - \mathbf{L}_e \mathbf{P}) \mathbf{u}_j + \mathbf{L}_e \mathbf{y}_r.\tag{3.16}$$

For this system to be asymptotically stable, the spectral radius $\max_i |\lambda_i(\mathbf{L}_u - \mathbf{L}_e \mathbf{P})| < 1$ for $i = 1, 2, \dots, Nq_o$ where $\lambda(\cdot)$ is the eigenvalue of (\cdot) . For most practical systems, asymptotic stability is not a strong enough condition. ILC systems that are asymptotically stable may experience large transients in the iteration domain prior to convergence [61]. Monotonic convergence is a stronger stability requirement than asymptotic stability and minimizes the possibility of transient growth.

For monotonic convergence [44], consider the control input \mathbf{u}_j as $j \rightarrow \infty$.

$$\lim_{j \rightarrow \infty} \mathbf{u}_j \triangleq \mathbf{u}_\infty = (\mathbf{I} - \mathbf{L}_u + \mathbf{L}_e \mathbf{P})^{-1} \mathbf{L}_e \mathbf{y}_r \quad (3.17)$$

Rearranging (3.17) to solve for $\mathbf{L}_e \mathbf{y}_r$, substituting that into (3.16), and reorganizing the variables results in the following equation,

$$\mathbf{u}_\infty - \mathbf{u}_{j+1} = (\mathbf{L}_u - \mathbf{L}_e \mathbf{P}) \cdot (\mathbf{u}_\infty - \mathbf{u}_j). \quad (3.18)$$

Monotonic convergence of the control input is determined by taking the 2-Norm of (3.18)

$$\|\mathbf{u}_\infty - \mathbf{u}_{j+1}\|_2 \leq \eta \|\mathbf{u}_\infty - \mathbf{u}_j\|_2, \quad (3.19)$$

where monotonicity requires that $\eta = \|\mathbf{L}_u - \mathbf{L}_e \mathbf{P}\|_{i2} < 1$, where $\|\mathbf{A}\|_{i2} = \bar{\sigma}(\mathbf{A})$ and $\bar{\sigma}(\mathbf{A})$ is the largest singular value of \mathbf{A} . Note that $\max_i |\lambda_i(\mathbf{A})| \leq \|\mathbf{A}\|_{i2}$. Furthermore, the value of η gives the rate of convergence for the control signal as defined by (3.19).

For the norm optimal ILC controller, we have $\mathbf{L}_u - \mathbf{L}_e \mathbf{P} = (\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R})^{-1} \mathbf{R}$. As a result, convergence is guaranteed for any symmetric positive semi-definite $(\mathbf{Q}, \mathbf{R}, \mathbf{S})$ with $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R}$ positive definite. Note that the convergence speed strongly depends on \mathbf{R} . For $\|\mathbf{R}\|_{i2} = 0$ deadbeat control is achieved and as $\|\mathbf{R}\|_{i2} \rightarrow \infty$ the convergence speed approaches zero.

3.3.2 Robust convergence

In this subsection, we consider the true system \mathbf{P}_t to correspond to the nominal model \mathbf{P} plus an uncertainty Δ_P : $\mathbf{P}_t = \mathbf{P}(I + \Delta_P)$, with the multiplicative uncertainty $\Delta_P = \mathbf{W}\Delta$ and $\|\Delta\|_{i2} \leq 1$ as defined in Section 3.1. As a result, the requirement for

robust convergence is given by

$$\|\mathbf{L}_u - \mathbf{L}_e \mathbf{P}_t\|_{i2} < 1 \quad (3.20)$$

$$\Rightarrow \max_{\Delta} \|(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R})^{-1} (\mathbf{R} - \mathbf{P}^T \mathbf{Q} \mathbf{P} \Delta_P)\|_{i2} < 1.$$

Lemma 1 Consider (3.20) with $\|\mathbf{R}\| = 0$. Then a sufficient condition for robust convergence is given by $\|(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S})^{-1} \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{W}\|_{i2} < 1$.

Proof 1 Follows directly from (3.20) and the inequality:

$$\|(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S})^{-1} \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{W} \Delta\|_{i2} \leq \|(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S})^{-1} \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{W}\|_{i2} \|\Delta\|_{i2}.$$

Lemma 2 Consider (3.20) with $\|(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S})^{-1} \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{W}\|_{i2} < 1$, and assume $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S}$ symmetric and positive definite. Then robust convergence is guaranteed for all $\mathbf{R} = r\mathbf{I}$, $r \in \mathbb{R} \geq 0$.

Proof 2 With $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S}$ a symmetric positive definite matrix, its singular value decomposition equals $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} = \mathbf{U} \Sigma \mathbf{U}^T$ with \mathbf{U} a unitary matrix and Σ diagonal and of full rank with diagonal elements σ_i . Furthermore, with $\mathbf{Z} \triangleq (\mathbf{U} \Sigma \mathbf{U}^T)^{-1} \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{W}$ and $\|(\mathbf{U} \Sigma \mathbf{U}^T)^{-1} \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{W}\|_{i2} \triangleq \alpha < 1$ we have $\|\mathbf{Z}\|_{i2} = \alpha < 1$. Therefore:

$$\begin{aligned} & \max_{\Delta} \|(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R})^{-1} (\mathbf{R} - \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{W} \Delta)\|_{i2} \\ &= \max_{\Delta} \|(\mathbf{U} \Sigma \mathbf{U}^T + r\mathbf{I})^{-1} (r\mathbf{I} + \mathbf{P}^T \mathbf{Q} \mathbf{P} \mathbf{W} \Delta)\|_{i2} \\ &= \max_{\Delta} \|(\mathbf{U} \Sigma \mathbf{U}^T + r\mathbf{I})^{-1} (r\mathbf{I} + \mathbf{U} \Sigma \mathbf{U}^T \mathbf{Z} \Delta)\|_{i2} \\ &= \max_{\Delta} \|(\Sigma + r\mathbf{I})^{-1} (r\mathbf{U}^T + \Sigma \mathbf{U}^T \mathbf{Z} \Delta)\|_{i2} \\ &\leq \|(\Sigma + r\mathbf{I})^{-1} (r\mathbf{I} + \alpha \Sigma)\|_{i2} \\ &= \max_i \frac{\alpha \sigma_i + r}{\sigma_i + r} \\ &< 1, \forall r \in \mathbb{R} \geq 0. \end{aligned}$$

From Lemma 2, we can conclude that the design parameter $\mathbf{R} = r\mathbf{I}$ does *not* influence the robust convergence properties of the ILC controlled system. \mathbf{S} on the other hand, should be designed such that the robust convergence condition in Lemma 1 holds. Similar statements and conclusions have been provided in [72].

3.3.3 Performance

To study performance, we study the steady state error $\mathbf{e}_{ss} \triangleq \lim_{j \rightarrow \infty} \mathbf{e}_j \triangleq \mathbf{e}_\infty$. The existence of $\mathbf{e}_\infty < \infty$ implies a convergent ILC controlled system (3.16). However, with $(\mathbf{L}_u, \mathbf{L}_e)$ requiring $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R}$ to be positive definite, convergence is guaranteed, as given in Subsection 3.3.1.

The steady state error is derived from the steady state command signal \mathbf{u}_{ss} (3.21). If the controller is asymptotically stable and $(\mathbf{I} - \mathbf{L}_u + \mathbf{L}_e \mathbf{P})$ is nonsingular, then the steady state control can be found as

$$\begin{aligned} \mathbf{u}_{ss} \triangleq \lim_{j \rightarrow \infty} \mathbf{u}_j \triangleq \mathbf{u}_\infty &= (\mathbf{I} - \mathbf{L}_u + \mathbf{L}_e \mathbf{P})^{-1} \mathbf{L}_e \mathbf{y}_r \\ \mathbf{u}_\infty &= (\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S})^{-1} \mathbf{P}^T \mathbf{Q} \mathbf{y}_r. \end{aligned} \quad (3.21)$$

With (3.21), $\mathbf{e}_\infty = \mathbf{y}_r - \mathbf{P} \mathbf{u}_\infty$, and $\mathbf{d}_j(k) = 0$ the steady state error is given by,

$$\lim_{j \rightarrow \infty} \mathbf{e}_j \triangleq \mathbf{e}_\infty = (\mathbf{I} - \mathbf{P}(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S})^{-1} \mathbf{P}^T \mathbf{Q}) \mathbf{y}_r. \quad (3.22)$$

From (3.22), we can now conclude the following: the smallest possible error, considered optimal performance in the ILC literature, requires $\|\mathbf{S}\|_{i2} = 0$ and hence $\mathbf{P}^T \mathbf{Q} \mathbf{P}$ to be positive definite. Furthermore, \mathbf{e}_∞ in (3.22) is not a function of \mathbf{R} , and hence performance is not a function of convergence speed in the absence of external stochastic disturbances.

In order to extend performance aspects of norm optimal ILC by including trial

varying or stochastic disturbances \mathbf{d}_j , consider $\mathbf{e}_j = \mathbf{y}_r - \mathbf{P}\mathbf{u}_j - \mathbf{d}_j$. Substituting \mathbf{e}_j into (3.16) yields the iteration domain closed loop update law with stochastic disturbances,

$$\mathbf{u}_{j+1} = (\mathbf{L}_u - \mathbf{L}_e \mathbf{P})\mathbf{u}_j + \mathbf{L}_e \mathbf{y}_r - \mathbf{L}_e \mathbf{d}_j. \quad (3.23)$$

If \mathbf{d}_j is bounded, then the asymptotic stability condition becomes the bounded-input, bounded-output stability condition for stochastic disturbances. Note that \mathbf{d}_j is filtered by \mathbf{L}_e in (3.23), and thus one would expect the stochastic disturbance sensitivity to decrease when the gain \mathbf{L}_e is reduced. From (3.11) minimizing $\|\mathbf{L}_e\|_{i2}$ requires changes to \mathbf{S} , \mathbf{R} , or a combination of the two. Additionally, as was shown in [73], the influence of stochastic disturbances can be reduced by reducing the convergence speed. Given that convergence speed is highly dependent on \mathbf{R} and yet \mathbf{R} does not affect robust convergence or nominal performance, it is the natural candidate for reducing stochastic disturbance sensitivity.

Note that the steady state solution for \mathbf{e}_j is a function of \mathbf{d}_j for any given learning filter.

$$\mathbf{e}_\infty = (\mathbf{I} - \mathbf{P}(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S})^{-1} \mathbf{P}^T \mathbf{Q})\mathbf{y}_r + (\mathbf{I} + \mathbf{P}(\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + 2\mathbf{R})^{-1} \mathbf{P}^T \mathbf{Q})\mathbf{d}_j. \quad (3.24)$$

Hence, after convergence the error \mathbf{e}_∞ will continue to fluctuate. In general, a larger $\|\mathbf{R}\|_{i2}$ will result in smaller fluctuations in \mathbf{e}_∞ . In the presence of stochastic disturbances, a compromise must be made between having a large $\|\mathbf{R}\|_{i2}$ to minimize disturbance effects, while also maintaining acceptable learning rates through a small $\|\mathbf{R}\|_{i2}$.

3.3.4 Tuning Guidelines for Time-Invariant Weighting Matrices

Based on the previous subsections, the following design tuning guidelines [74] for norm-optimal ILC control are given. The guidelines are design heuristics which can be used as a starting point for weighting matrix design. Similar to other design techniques (i.e. Ziegler-Nichols tuning rules [67]), the initial selection of the weighting matrices may not result in a controlled system which exhibits the desired design criteria. In this situation, the tuning may need to be refined over multiple designs.

Note that selection of (\mathbf{S}, \mathbf{R}) should occur before a sequence of trials, *not* between trials. This minimizes the possibility of selecting a combination of (\mathbf{S}, \mathbf{R}) weighting matrices which may result in an undesirable performance or even unstable system. This also allows one to accurately determine the effect of the variation in the (\mathbf{S}, \mathbf{R}) weighting matrices on the system. These guidelines are most easily implemented using common $q\mathbf{I}, s\mathbf{I}, r\mathbf{I}$ diagonal type real-valued scalar gains. The guidelines are given as follows.

- s1) Design \mathbf{Q} : \mathbf{Q} corresponds to the desired weighting of the error. Generally let $\mathbf{Q} = \mathbf{I}$ for uniform weighting of the individual axis errors.
- s2) Design \mathbf{S} : The actual system dynamics will not usually be perfectly captured by the system model. Thus, \mathbf{S} must be designed such that the system is robustly monotonically convergent. Start with an \mathbf{S} yielding $\|\mathbf{S}\|_{i2} \approx 0.01\|\mathbf{P}\|_{i2}$. Note, the critical design parameter is the size of $\|\mathbf{S}\|_{i2}$ relative to the size of $\|\mathbf{P}\|_{i2}$, where the magnitude of $\|\mathbf{P}\|_{i2}$ is related to system uncertainty through $\mathbf{P}\Delta_P$. Subsequently reduce $\|\mathbf{S}\|_{i2}$ until the system diverges. Set $\|\mathbf{S}\|_{i2} = 2 \cdot \|\mathbf{S}\|_{i2}^{min}$ to allow for a safety factor of 2.
- s3) Design \mathbf{R} : When stochastic disturbances are present, steady state error fluctua-

tions will occur. Start with $\|\mathbf{R}\|_{i2} = 0$ and increase $\|\mathbf{R}\|_{i2}$ until the steady state error fluctuations are within desired bounds, or the root mean square (RMS) error does not decrease anymore.

- s4) Iterate process: If ILC performance does not meet design specifications, one may need to go as far back as system identification to determine a smaller model uncertainty. Repeat design steps s1) - s3) until the ILC performance is within desired convergence and performance requirements.

These tuning guidelines provide a general approach for designing a norm optimal learning controller using time-invariant weighting matrices. The relationships between the $(\mathbf{Q}, \mathbf{S}, \mathbf{R})$ weighting matrices and nominal convergence, robust convergence, and performance criteria for a given system are maintained irrespective of the use of time-invariant or time-varying weighting matrices.

3.4 Servo System Example: Norm-Optimal Design

To illustrate the effects of varying $\mathbf{Q}, \mathbf{S}, \mathbf{R}$, this subsection provides performance results for various combinations of the weighting matrices, where $\mathbf{Q} = q\mathbf{I}$, $\mathbf{S} = s\mathbf{I}$, and $\mathbf{R} = r\mathbf{I}$. The different weighting matrices are applied to the following servo system example [68].

Consider the discrete-time system,

$$G(z) = \frac{z - 0.5}{(z - 1)(z - 0.925)} \quad (3.25)$$

whose Bode plot is shown in Figure 3.2. A discrete-time system is used in this example because ILC requires a storage of signals, which is generally done for discrete-time

systems. In (3.25), G represents a servo-positioning system with viscous friction. Assume that the system is stabilized with a proportional feedback controller,

$$C(z) = 0.425. \quad (3.26)$$

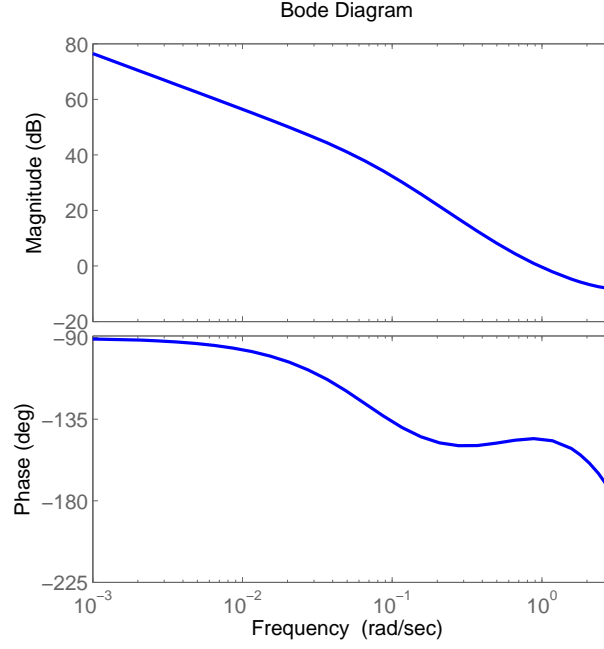


Figure 3.2: Bode plot of the servo system in (3.25)

The desired output trajectory is the triangle wave (illustrated in Figure 3.3),

$$y_d(k) = \begin{cases} k/200 & 0 \leq k \leq 200 \\ 2 - k/200 & 201 \leq k \leq 400, \end{cases} \quad (3.27)$$

where $k = 0, 1, \dots, 400$ is the time index. The triangle wave is used, for example, in scanning operations where the forward and backward motions are scanning at the same velocity.

Figure 3.4 shows the error, feedback control signal, and feedforward control signal for $\{\mathbf{Q}, \mathbf{S}, \mathbf{R}\} = \{\mathbf{I}, \mathbf{I}, \mathbf{I}\}$. These results illustrate how the feedback signal is slowly

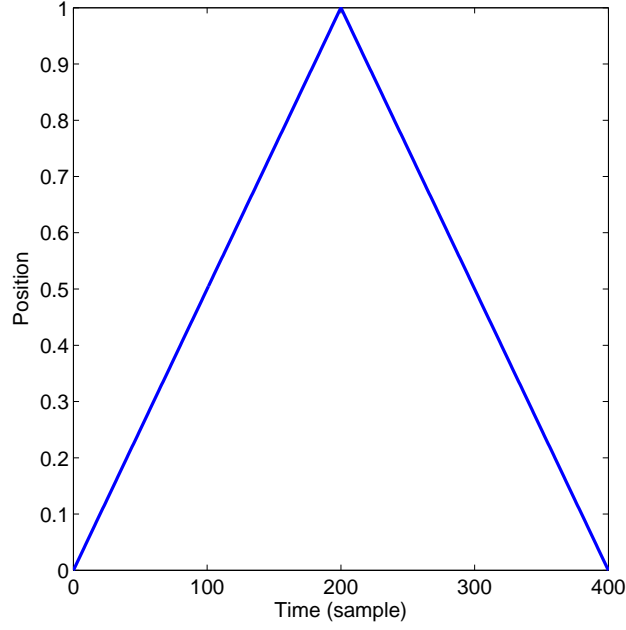


Figure 3.3: Triangle reference signal and system output using feedback controller.

replaced by the ILC feedforward signal with increasing iteration.

To determine the effect of q on the error, s and r are held constant at $\{s, r\} = \{1, 1\}$, while the value of q is varied as $\{1, 0.5, 0.1, 0.001\}$. Figure 3.5 illustrates that the smaller the value of q , the larger the value of the root mean square (RMS) converged tracking error. Varying s affects performance and convergence. While a smaller s results in smaller converged RMS error, robustness to model uncertainty requires a larger s value. Holding $\{q, r\}$ constant at $\{1, 1\}$, respectively, Figure 3.6 illustrates how decreasing s results in a decrease in the RMS converged error.

Lastly, r can be increased to minimize trial varying disturbances, at the expense of convergence rate. Holding $\{q, s\}$ constant at $\{1, 1\}$, respectively, Figure 3.7 shows how the convergence rate increases as r increases. Figure 3.8 illustrates the effects on the RMS error of adding white noise $\{(mean, var) = (0, 1e-6)\}$ to the system. As r increases, the effect of the noise on the system is decreased, as demonstrated by a decrease in converged RMS error.

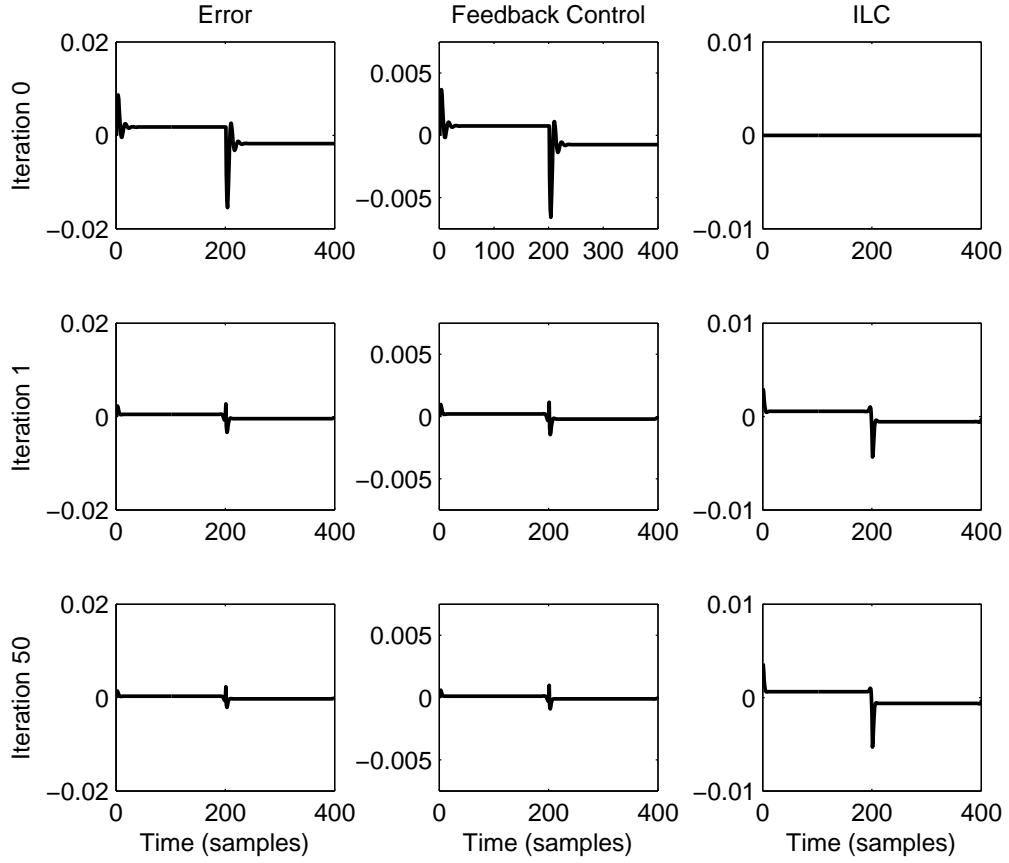


Figure 3.4: Error, feedback control, and ILC time-series using norm-optimal ILC. Note how the feedback control signal is slowly being replaced by the ILC feedforward signal with increasing iteration.

Based on the guidelines provided in Subsection 3.3.4 and the performance results presented in the current section using the servo system example, the impact of varying the norm optimal weighting matrices on the performance, robustness, and convergence of a system has been summarized in Figure 3.9.

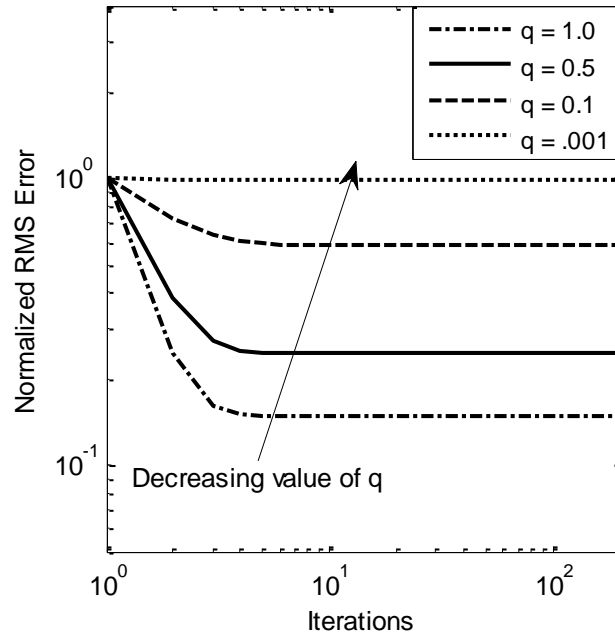


Figure 3.5: RMS converged error values for various \mathbf{Q} weighting matrices.

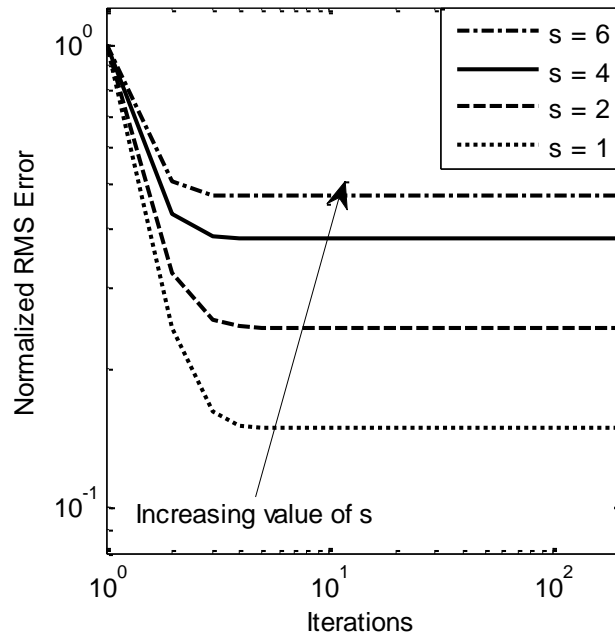


Figure 3.6: RMS converged error and monotonic convergence values for various \mathbf{S} weighting matrices.

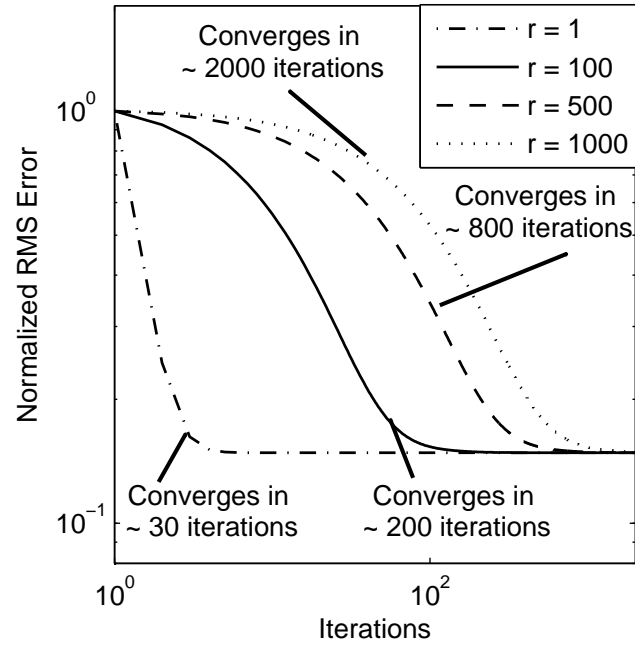


Figure 3.7: RMS error values for varying \mathbf{R} weighting matrices. Note the longer iteration range with increasing value of r .

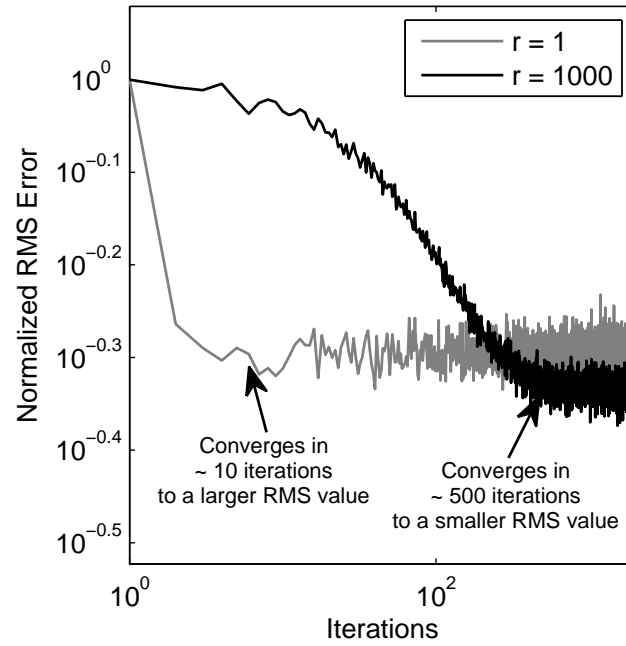


Figure 3.8: RMS error values for varying \mathbf{R} weighting matrices with trial varying noise. Note the longer iteration range.

	Nominal Convergence	Robust Convergence	Nominal Performance	Performance with disturbances
Q	✗	✗	$Q = I$	✗
S	✗	$\ S\ _{i2} \rightarrow \infty$ Robust to all unmodelled dynamics	$\ S\ _{i2} \rightarrow 0$ minimize converged error	✗
R	$\ R\ _{i2} \rightarrow 0$ deadbeat control	✗	✗	$\ R\ _{i2} \rightarrow \infty$ minimize error fluctuations

Figure 3.9: Comparison of the effects of the weighting matrix design on performance, robustness to model uncertainty and disturbances, and convergence.

Chapter 4

Coupling and Coordination of Multiple Systems

The focus of this work is to improve the coordination and control of systems that perform the same task multiple times. The previous chapter introduced the learning control technique employed in this work. In addition to determining the control strategy, one must define coordination of multiple systems for the purpose of this research. This chapter presents two approaches for combining multiple systems. The first section focuses on the cross-coupled design scenario defined in Subsections 2.1 and 2.2.2. The second section identifies some traditional approaches for coordinating multiple systems based on the 2-D approach known as shape tracking presented in Subsection 2.2.3. While this approach is based on a 2-D design, it will be useful for the 3-D control scheme presented in Chapter 6.

4.1 Contour Error

When coupling multiple independent systems or agents, one may couple these agents through the desired coordinated output of the combined MIMO system. For agents consisting of two or more individual axes, an additional error component known as the contour error can be identified, as illustrated in Figure 4.1. Contour errors for a general class of multi-axis systems can be defined with respect to the individual error signals, e_1, e_2, \dots, e_{q_o} , and trajectory dependent gains known as coupling gains [34, 36], $c_1(\theta, k), c_2(\theta, k), \dots, c_{q_o}(\theta, k)$, where k is the time interval from $k = 0, 1, \dots, N - 1$, θ is the instantaneous angle of a line tangential to the reference trajectory at a specific

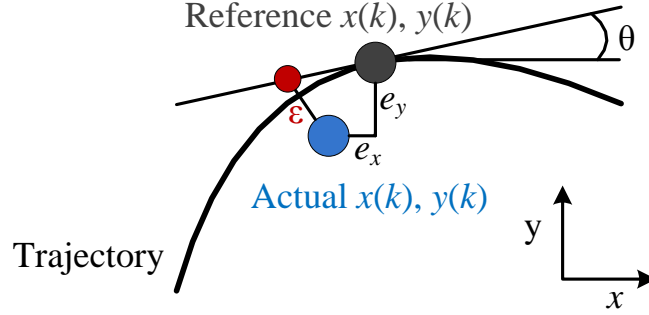


Figure 4.1: Trajectory illustrating contour and individual axis errors. Linearized coupling gains at point in time (k) can be used to simplify the derivation of the contour error.

point in time k with respect to the horizontal axis of the coordinate system, and $1, 2, \dots, q_o$ are the individual uncoupled axes. Mathematically, for two axes this can be shown as [34]

$$\varepsilon(k) = c_1(\theta, k) \cdot e_1(k) + c_2(\theta, k) \cdot e_2(k) \quad (4.1)$$

$$\varepsilon(k) = C(\theta, k) \cdot e(k). \quad (4.2)$$

Linearized coupling gains for the 2 degree-of-freedom (DOF) example illustrated in Figure 4.1 have the following format

$$c_1(\theta, k) = -\sin \theta(k); c_2(\theta, k) = \cos \theta(k). \quad (4.3)$$

For the trajectory shown in Figure 4.1, e_1 and e_2 correspond to e_x and e_y , respectively. For systems with multiple degrees of freedom, a coordinate transformation from the extended axes to a planar trajectory may need to be employed. For these systems, the derivation of the coupling gains would be significantly more complicated and incorporate multiple angles in addition to θ . Note that the use of trajectory-dependent coupling gains leads to a time-varying controller.

One of the main challenges associated with contour tracking control comes from determining the appropriate coupling gains. As the desired trajectory becomes more

complex, the derivation of the coupled output (i.e. the contour error) becomes more complicated. Assuming a repetitive process, one of the advantages of using a *learning* approach is that a coupled learning controller designed to maintain contour tracking can be developed using linearized coupling gains [2]. Any discrepancies between the linearized gains and the actual derivation of the gains will be learned and compensated for by the controller.

4.2 Formation Control

Contour tracking describes a technique for coupling the output trajectory from multiple axes for a given system. An approach for coupling multiple individual systems, known as formation control, ensures shape or formation-keeping among multiple agents. In this technique, there are two distinct design objectives; maintain the formation of multiple agents, and minimize individual axis tracking errors for each independent agent. The general approach to formation control includes decoupling the individual system objective from the group objective. In this manner, an optimal solution for accomplishing each objective can be obtained. This approach enables the group to maintain a set shape or formation, while also enabling each system to accomplish its desired task independently.

There are three general techniques for determining a coordinated system's formation position: unit- or formation-centered reference, leader reference, and neighbor reference [11].

4.2.1 Unit-Centered Reference Approach

In the unit- or formation-centered approach, each agent independently calculates a group formation center by averaging the xyz positions of all the other agents. The individual agents then determine and maintain their own formation and tracking

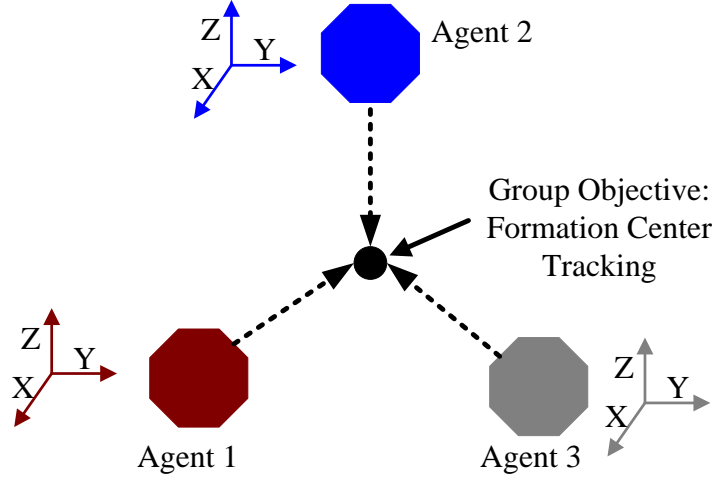


Figure 4.2: Formation-center formation tracking technique. Note that each agent determines the group formation center based on the positions of the other agents.

positions relative to the calculated center. Figure 4.2 illustrates the formation-center approach.

This approach has been shown to be successful for tasks such as target tracking. Klein et al [55] describes four subproblems of target tracking that fit into the formation-center approach: constant velocity matching, dynamic velocity matching, centroid target tracking of position and velocity, and position target tracking without velocity matching controls. In all four cases, the objective of each agent is to achieve a specific task such that the group centroid is able to follow the desired target. The resulting controller achieves target tracking, as long as the group centroid tracks the desired task and the individual agents stay near the target. One of the main challenges of this approach is that each system must receive regular updates from all of the other systems. If one system loses communication, the formation center will be miscalculated and the group formation will break down.

4.2.2 Leader Reference Approach

In leader reference control, each agent determines its formation position with respect to the lead agent. The lead agent determines the desired trajectory and does not attempt to maintain the group formation. The other agents within the system are responsible for formation maintenance. Figure 4.3 shows an example of the leader reference approach.

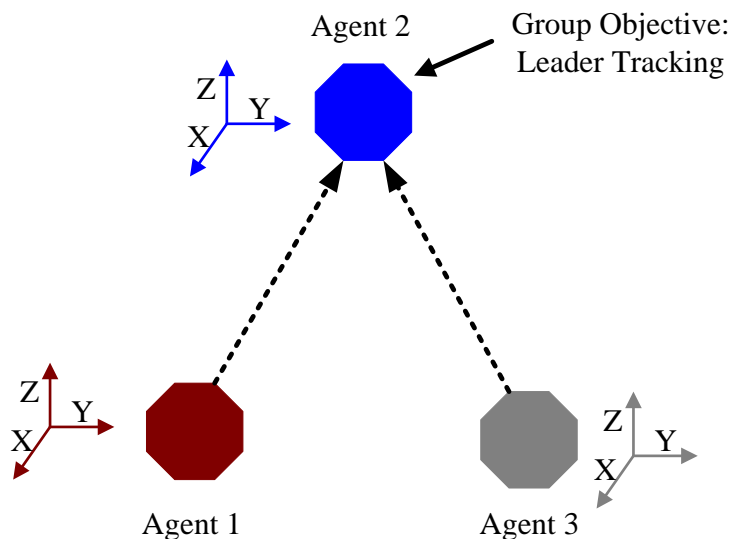


Figure 4.3: Leader reference formation tracking technique. Note that each agent, aside from the lead agent, determines its position from the lead agent.

As with the formation-center approach, one of the key challenges in formation tracking is maintaining formation control in the presence of potentially limited or sparse communication. Porfiri et al address this limitation in [54] by introducing a virtual leader. The virtual leader is used to predict the position of a lead agent when communication has been lost. Communication also becomes an issue as the number of agents increases, particularly between the lead agent and other agents separated by large distances. Morgan and Schwartz [59] present an approach in which they decompose a large swarm problem into several subswarm problems. These subswarms are independent from each other, only allowing communication within the group.

Each subswarm assigns a lead agent, thereby creating several small leader-reference formation tracking systems.

4.2.3 Neighbor Reference Approach

The last approach, illustrated in Figure 4.4, is termed neighbor reference control and requires that each agent maintain a position relative to other neighboring agents. Unlike the other two approaches, this approach only requires communication between a few neighboring agents. This approach has often been studied in Nature with respect to fish schooling [53], birds flocking [7], and insects foraging [4]. In all of these examples, the behavior of the individual is dictated by neighboring individuals. Morgan and Schwartz [59] apply this approach for collision avoidance in their subswarm approach. While communication is generally restricted with respect to other agents within the subswarm, perimeter agents can communicate with neighboring agents in other subswarms to ensure collision avoidance.

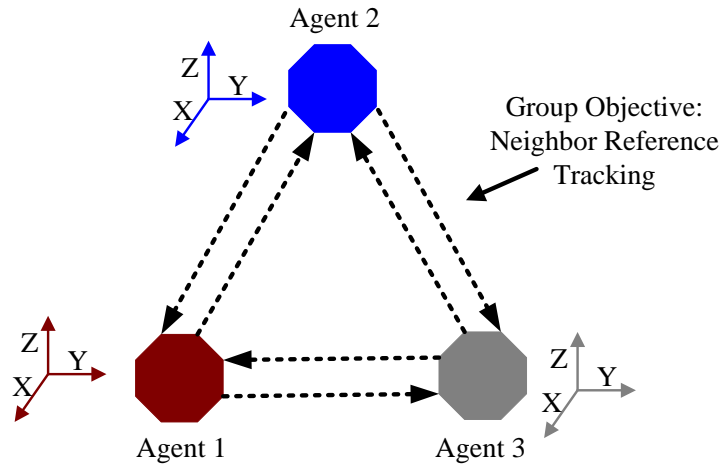


Figure 4.4: Neighbor reference formation tracking technique. Note that each agent determines its position from neighboring agents.

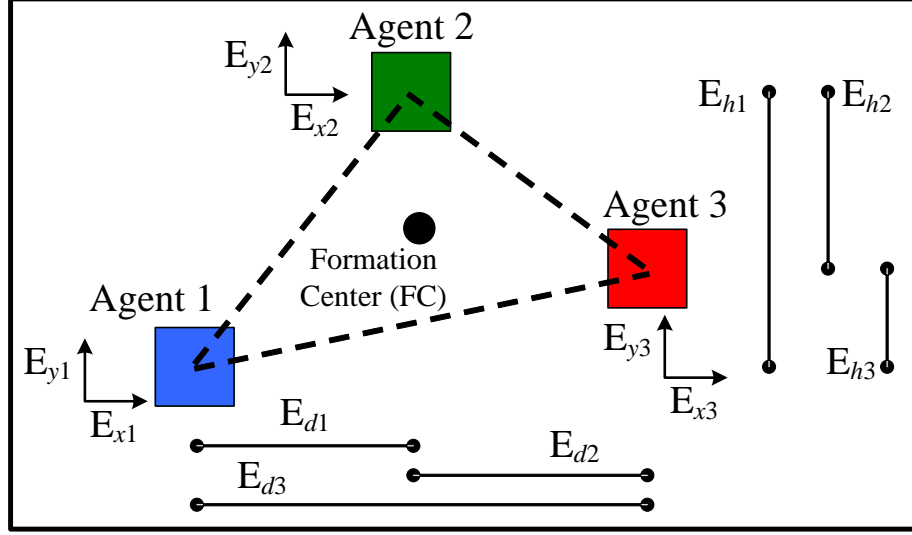


Figure 4.5: Example formation set-up with three individual multi-axis agents. Note that the planar formation errors are defined in terms of horizontal formation errors $\{E_{d1}, E_{d2}, E_{d3}\}$ and vertical formation errors $\{E_{h1}, E_{h2}, E_{h3}\}$.

4.2.4 Mapping Position Tracking to Formation Errors

One of the key assumptions with formation control is the ability to map the position tracking errors to the formation errors. This mapping enables one to use a single framework for addressing the two distinct design objectives. To illustrate the process of deriving the mapping matrix, three dual-axis agents with planar position and formation tracking are presented in Figure 4.5.

The three individual agents from Figure 4.5 are coupled through the formation objective defined in terms of the horizontal formation errors, E_{d1}, E_{d2}, E_{d3} , and the vertical formation errors, E_{h1}, E_{h2}, E_{h3} . Redefining the formation error signals in terms of the $\{x, y\}$ position errors for the independent agents can be accomplished through a mapping matrix F defined in (4.4). The reader should note that this mapping matrix is not unique. The size is unique for a bijective map. However, if

you choose the error definitions differently, the matrix entries will change.

$$\begin{aligned}
\begin{bmatrix} E_{d1}(k) \\ E_{d2}(k) \\ E_{d3}(k) \\ E_{h1}(k) \\ E_{h2}(k) \\ E_{h3}(k) \end{bmatrix} &= \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} E_{x1}(k) \\ E_{y1}(k) \\ E_{x2}(k) \\ E_{y2}(k) \\ E_{x3}(k) \\ E_{y3}(k) \end{bmatrix} \\
E_{d,h}(k) &= F \cdot E_{(x,y)}(k)
\end{aligned} \tag{4.4}$$

The mapping of the position to the formation errors results in a non-symmetric matrix in which only certain combinations of the individual position tracking errors are weighted to ensure a desired formation or shape. These gains are time-invariant for MIMO systems in which the same reference trajectory is applied to each individual agent.

Chapter 5

2-D Coupled Iterative Learning Control

Coupled ILC is a 2-D control approach for multi-axis systems that perform the same task repetitively. From the dimensional analysis approaches illustrated in Figure 2.1, this control scheme is applied to 2-D problems with time along one axis and iteration along the second axis. This chapter presents the 2-D coupled ILC technique including background information, the derivation of a 2-D design framework for coupled ILC, and simulation results validating this control method.

5.1 Introduction

In Section 3.3 a norm optimal design framework based on a quadratic optimization problem was presented. Current applications of norm optimal ILC have generally been for systems in which the unmodelled dynamics and external disturbances occur throughout the time period, yielding linear time-invariant (LTI) learning controllers [75]. However, in many systems there are position- or time-varying dynamics that affect the performance or robustness of the system at different times throughout a single iteration. For these types of systems, it is beneficial to consider a time-varying controller which enables one to focus on the different position or time dynamics independently. Focusing on individual dynamics at different times throughout the iteration may result in a final outcome that not only improves tracking control but is more robust to time-varying disturbances.

This chapter focuses on the design of time-varying iterative learning controllers in

the norm optimal framework. The objective of these controllers is to address time-varying dynamics and disturbances that affect the performance and/or robustness of a system that repeats the same process iteration after iteration. Design and analysis for the time-varying learning controllers uses the lifted domain since that is a natural domain for iterative processes and is well suited for time-varying systems.

The norm optimal ILC framework in the lifted domain is designed to minimize a quadratic optimization problem through the selection of weighting matrices targeting performance and robustness criteria [76]. The general norm optimal ILC approach for PMC on repetitive systems has been previously used to implement time-invariant weighting matrices that are designed to satisfy various constraints [71, 75, 77]. For example, a norm optimal control design for minimum and non-minimum phase systems is developed in [75]. In this paper, the use of time-invariant diagonal weighting matrices enables the control algorithm in the lifted domain to correspond to a filtering operation in the frequency domain. In [71], time-invariant weighting matrices tailored for process control applications which exhibit time and/or position dependent dynamics are introduced. Although these time-invariant weighting matrices have been shown to improve performance and robustness for process control systems which exhibit time and/or position dependent behavior [71], limiting the performance of the learning controller to the worst-case dynamics may result in a more conservative controller and a decrease in system performance.

Previous work has been done to address the deficiencies of time-invariant weighting matrices through the design of time-varying weighting matrices for specific applications, [73, 77, 78]. In [77], critical and non-critical sections of a given trajectory are weighted separately using a time-varying weighting matrix design. [2] introduces the use of time-varying weighting matrices for enhanced contour tracking, while [73] implements a time-varying weighting matrix to address robustness issues in systems subjected to high frequency trajectory tracking. While some work has been done on

improving the trajectory design using ILC [79], this work focuses on trajectories for which the critical design criterion is to follow the given trajectory. On the basis of previous work in the area of norm optimal learning controllers using time-invariant and time-varying weighting matrices, the authors have identified two key issues: 1) development of a generalized structure for time-varying weighting matrices, and 2) description of a design methodology for designing time-varying weighting matrices.

This chapter seeks to address issues 1) and 2) by presenting the general format and design of time-varying weighting matrices using the norm optimal framework. The weighting matrices are then implemented on a model of a multi-axis robotic testbed.

5.2 Background

The idea of coupled ILC was derived as an extension of the more traditional coupled feedback design approach known as cross-coupled control [34] (described in Section 2.1). Cross-coupled control provided an intuitive approach for minimizing the coupled tracking errors of a multi-axis system defined as the contour error, (4.2). While this approach provided a means for minimizing errors along the trajectory, one of the main challenges in this approach came from deriving the coupling gains used in the definition of the contour error.

Work in [2] presented the first learning controller designed to minimize tracking errors along the trajectory. Using the *learning* advantage of this approach, linearized coupling gains simplified the control design while improving the contour tracking performance of the system as illustrated in Figure 5.1.

Initial work in the area of coupled ILC focused on implementing a simple proportional derivative (PD) controller in the frequency domain [51]. PD control was selected in an effort to simplify the design for use and implementation of a combined

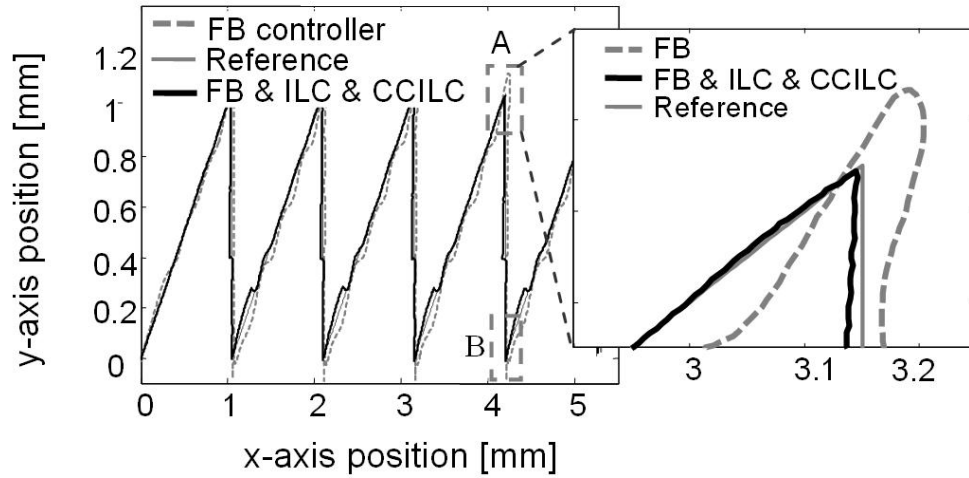


Figure 5.1: Experimental results comparing a traditional feedback controller versus a coupled ILC design implemented on a micro-Robotic Deposition system. Results taken from [2].

individual axis ILC and coupled ILC controller when little or no plant information was available. PD control does not require *a priori* information about the system, thereby enabling this control technique to be implemented directly on many different types of systems with only the control gains needing to be tuned. Additionally, PD control is useful for practicing servo engineers because they are used to tuning different gains.

While this approach worked well when combined with individual axis learning controllers, the PD control design was not adequate for improving the contour tracking performance of the system when used without individual learning controllers on each independent axis. In order to implement a single controller which minimized contour errors, an optimized control design needed to be employed. Given the time-varying nature of the trajectory dependent coupling gains, an optimized time-domain design approach was a straightforward choice since the time-domain is the natural framework for time-varying designs.

5.3 Time-varying Weighting Matrices

5.3.1 Motivation

As Section 3.3 stated, the weighting matrices are generally of the form $(\mathbf{Q}, \mathbf{R}, \mathbf{S}) \triangleq (q\mathbf{I}, r\mathbf{I}, s\mathbf{I})$. While this approach works well for time-invariant unmodelled dynamics and external disturbances, in many manufacturing systems the disturbances, dynamics, and tracking errors are time and position dependent. For these systems, time-varying weighting matrices of the form $(\mathbf{Q}^{tv}, \mathbf{S}^{tv}, \mathbf{R}^{tv})$ are better able to address specific design requirements at specific time locations throughout the trajectory. Time-varying weighting matrices result in a modified optimization cost function,

$$\mathcal{J} = \mathbf{e}_{j+1}^T \mathbf{Q}^{tv} \mathbf{e}_{j+1} + \mathbf{u}_{j+1}^T \mathbf{S}^{tv} \mathbf{u}_{j+1} + (\mathbf{u}_{j+1} - \mathbf{u}_j)^T \mathbf{R}^{tv} (\mathbf{u}_{j+1} - \mathbf{u}_j), \quad (5.1)$$

where the generalized structures of $(\mathbf{Q}^{tv}, \mathbf{S}^{tv}, \mathbf{R}^{tv})$ for multi-axis systems are given below,

$$\mathbf{Q}^{tv} = \boldsymbol{\Sigma}_Q \cdot [\boldsymbol{\Gamma} \mathbf{1}_Q + \boldsymbol{\Gamma} \mathbf{2}_Q \cdot \mathbf{C}_Q^T \mathbf{C}_Q] \quad (5.2)$$

$$\mathbf{S}^{tv} = \boldsymbol{\Sigma}_S \cdot [\boldsymbol{\Gamma} \mathbf{1}_S + \boldsymbol{\Gamma} \mathbf{2}_S \cdot \mathbf{C}_S^T \mathbf{C}_S] \quad (5.3)$$

$$\mathbf{R}^{tv} = \boldsymbol{\Sigma}_R \cdot [\boldsymbol{\Gamma} \mathbf{1}_R + \boldsymbol{\Gamma} \mathbf{2}_R \cdot \mathbf{C}_R^T \mathbf{C}_R]. \quad (5.4)$$

In (5.2)–(5.4), the $\mathbf{C}_{(\cdot)}$ matrices contain the terms used to define coupling between the individual signals of a MIMO system. The coupling can be determined with respect to the tracking profile, the physical system, or to user defined relationships between the signals. For example, \mathbf{C}_Q corresponds to the coupling matrix used to define contour error with respect to the individual axis errors, (4.2), as a function of the reference trajectory. The matrices $\boldsymbol{\Gamma} \mathbf{1}_{(\cdot)}$ and $\boldsymbol{\Gamma} \mathbf{2}_{(\cdot)}$ refer to the amount of weighting applied to the coupled or individual signals, respectively. These matrices are of the

forms provided in (5.5) and (5.6), where the inner block diagonal matrices are shown for a 2 DOF system.

$$\mathbf{\Gamma 1}_{(\cdot)} = \begin{bmatrix} \begin{bmatrix} \gamma_{(\cdot)}(1) & 0 \\ 0 & \gamma_{(\cdot)}(1) \end{bmatrix} & & 0 \\ & \ddots & \\ 0 & & \begin{bmatrix} \gamma_{(\cdot)}(N) & 0 \\ 0 & \gamma_{(\cdot)}(N) \end{bmatrix} \end{bmatrix}, \quad (5.5)$$

$$\mathbf{\Gamma 2}_{(\cdot)} = \begin{bmatrix} \begin{bmatrix} 1 - \gamma_{(\cdot)}(1) & 0 \\ 0 & 1 - \gamma_{(\cdot)}(1) \end{bmatrix} & & 0 \\ & \ddots & \\ 0 & & \begin{bmatrix} 1 - \gamma_{(\cdot)}(N) & 0 \\ 0 & 1 - \gamma_{(\cdot)}(N) \end{bmatrix} \end{bmatrix}. \quad (5.6)$$

As can be seen from (5.5) and (5.6), the individual elements in $\mathbf{\Gamma 1}_{(\cdot)}$ and $\mathbf{\Gamma 2}_{(\cdot)}$ are directly related. The gain $\gamma_{(\cdot)}(k)$ is used to determine what portion of the overall weighting is applied to the individual and coupled signals, respectively. From (5.5) and (5.6), $(\gamma_{(\cdot)}(k) = 1)$ refers to all of the weighting being applied to the individual signals, while $(\gamma_{(\cdot)}(k) = 0)$ results in only the coupled signals being weighted. The gain matrix $\mathbf{\Sigma}_{(\cdot)}$ determines the overall weighting on the error signals, control signals, or change in control signals and is of the form shown in (5.7). Note that the inner

diagonal matrix is illustrated for a 2 DOF system.

$$\mathbf{\Sigma}_{(\cdot)} = \begin{bmatrix} \begin{bmatrix} \sigma_{(\cdot)}(1) & 0 \\ 0 & \sigma_{(\cdot)}(1) \end{bmatrix} & & & 0 \\ & \ddots & & \\ & & \begin{bmatrix} \sigma_{(\cdot)}(N) & 0 \\ 0 & \sigma_{(\cdot)}(N) \end{bmatrix} & \\ 0 & & & \end{bmatrix}. \quad (5.7)$$

The gains $(\sigma_Q(k), \sigma_S(k), \sigma_R(k))$ are similar to the gains (q, s, r) from the time-invariant form of the weighting matrices $(\mathbf{Q}, \mathbf{R}, \mathbf{S}) \triangleq (q\mathbf{I}, r\mathbf{I}, s\mathbf{I})$ in that they weight the different components in the cost function. Therefore, the tuning rules identified in Section 3.3.4 can be used to design the gains $(\sigma_Q(k), \sigma_S(k), \sigma_R(k))$, respectively. Note that while the gain matrix $\mathbf{\Sigma}_{(\cdot)}$ could be absorbed into the gain matrices $\mathbf{\Gamma}\mathbf{1}_{(\cdot)}$ and $\mathbf{\Gamma}\mathbf{2}_{(\cdot)}$, it is kept separate in this paper to add an additional design component which enables more direct intra element weighting within the cost function.

Each of the design elements in (5.2)–(5.4) offers a means of weighting different aspects of the error signals, control signals, and change in control signals. A four step design methodology providing details on each of the individual terms in (5.2) is given in Subsection 5.3.2. While this methodology is presented for the \mathbf{Q}^{tv} weighting matrix, the same process can be applied to the design of the \mathbf{S}^{tv} and \mathbf{R}^{tv} weighting matrices.

5.3.2 Design Methodology

When it comes to designing time-varying weighting matrices for multi-axis systems it is important to maintain design flexibility to allow for a variety of system dynamics, applications, and external environments. The format presented in (5.2)–(5.4) provides the framework for considering: the coordination between the signals $\mathbf{C}_{(\cdot)}$, the

importance of the individual versus coupled signals $\mathbf{\Gamma 1}_{(\cdot)}$ and $\mathbf{\Gamma 2}_{(\cdot)}$, and the weighting on the signals as a whole $\mathbf{\Sigma}_{(\cdot)}$. The design procedure given below and detailed in Figure 5.2 presents a four step methodology for designing a time-varying \mathbf{Q}^{tv} weighting matrix.

- s1) Design \mathbf{C}_Q : \mathbf{C}_Q corresponds to the desired coupling between the individual error signals. Generally, for the weighting on the error signals, let $\varepsilon = \mathbf{C}_Q \cdot \mathbf{e}$ where \mathbf{C}_Q contains the coupling gains determined from the reference trajectory (4.3).
- s2) Design $\mathbf{\Gamma 1}_Q$ and $\mathbf{\Gamma 2}_Q$: As mentioned previously, $\mathbf{\Gamma 1}_{(\cdot)}$ and $\mathbf{\Gamma 2}_{(\cdot)}$ refer to the weighting gain matrices applied to the coupled and individual error signals, respectively. Set $\gamma_Q(k) = 1$ in (5.5) and (5.6) at the discrete times when time and position synchronization are critical. Set $\gamma_Q(k) = 0$ when the synchronization between time and position is not critical and the emphasis is on contour tracking.
- s3) Design $\mathbf{\Sigma}_Q$: $\mathbf{\Sigma}_Q$ provides an overall weighting on the error. The design of $\mathbf{\Sigma}_Q$ should follow the tuning guidelines presented in Section 3.3.4 with respect to the correlation between \mathbf{Q} and performance. Generally, set $\sigma_Q(k) = 1$ for baseline tracking and increase the gain in the locations where more emphasis on trajectory tracking is required, i.e. in the corners of rastered trajectories. Continue increasing the gain until the system performance begins to display transient behavior. Set $\sigma_Q(k) = \frac{1}{2} \cdot \sigma_Q^{max}(k)$ to allow for a safety factor of 2.
- s4) Iterate process: Evaluate the performance of the controlled system. If it does not meet design specifications, repeat design steps s1) - s3) until the ILC performance is within desired convergence and performance requirements.

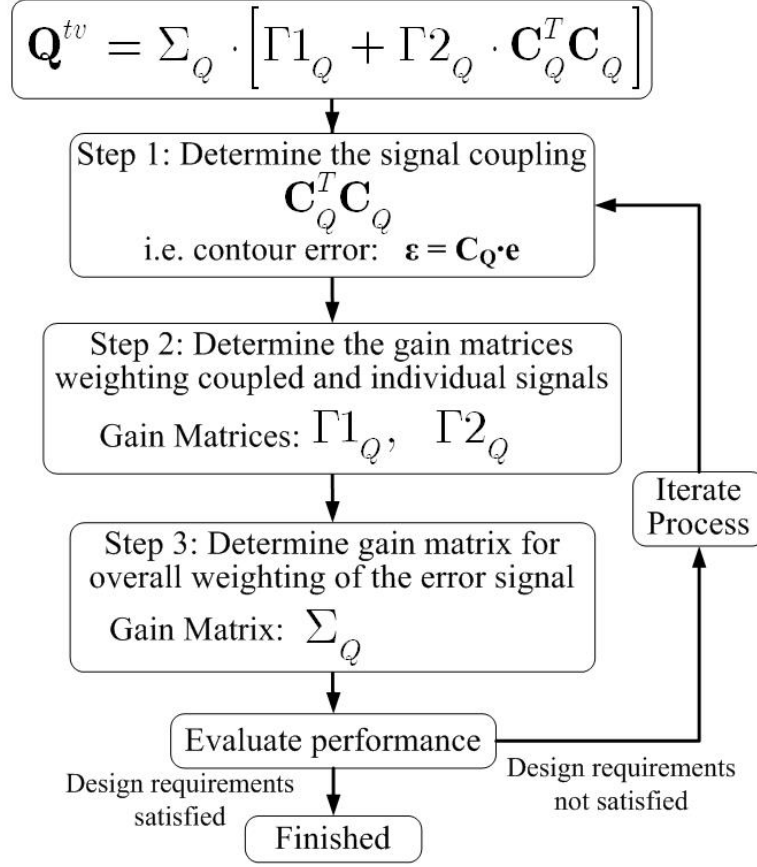


Figure 5.2: Time-varying Weighting Matrix Design Methodology.

As mentioned previously, this type of design approach can be applied when designing the \mathbf{S}^{tv} and \mathbf{R}^{tv} weighting matrices with respect to the control signals or change in control signals, respectively. Time or position dependent weighting matrix designs allow the controller to maximize the performance and robustness of the system without being overly conservative or becoming unstable. The next three Sections 5.4-5.6 demonstrate the design methodology presented in Figure 5.2 for generating time-varying weighting matrices of the form given in (5.2)–(5.4).

In order to validate the performance and robustness improvements attributed to the design of \mathbf{Q}^{tv} , \mathbf{S}^{tv} , and \mathbf{R}^{tv} weighting matrices, the next three sections also include results obtained from simulating time-varying norm optimal learning controllers (5.8)

with models of a multi-axis robotic testbed.

$$\begin{aligned}
\mathbf{u}_{j+1} &= \mathbf{L}_{\mathbf{u}}^{tv} \mathbf{u}_j + \mathbf{L}_{\mathbf{e}}^{tv} \mathbf{e}_j \\
\mathbf{L}_{\mathbf{u}}^{tv} &= (\mathbf{P}^T \mathbf{Q}^{tv} \mathbf{P} + \mathbf{S}^{tv} + \mathbf{R}^{tv})^{-1} (\mathbf{P}^T \mathbf{Q}^{tv} \mathbf{P} + \mathbf{R}^{tv}) \\
\mathbf{L}_{\mathbf{e}}^{tv} &= (\mathbf{P}^T \mathbf{Q}^{tv} \mathbf{P} + \mathbf{S}^{tv} + \mathbf{R}^{tv})^{-1} \mathbf{P}^T \mathbf{Q}^{tv}.
\end{aligned} \tag{5.8}$$

5.4 Design Example 1: Time-Varying \mathbf{Q} Weighting Matrix

The previous section presented the basic format and design methodology for developing norm optimal learning controllers using time-varying weighting matrices. This section focuses on improving trajectory tracking through a time-varying design of the \mathbf{Q} weighting matrix. As shown in (3.13), \mathbf{Q} applies weighting to the error signals, thereby influencing the tracking performance directly. The \mathbf{Q} weighting matrix is time-varied based on the reference trajectory and the initial error signals without learning, i.e. iteration $j = 0$. Additionally, the time-varying format of the weighting matrix (5.2) introduces a method for weighting the coordination of the error signals through the use of the \mathbf{C}_Q matrix. For \mathbf{Q}^{tv} , \mathbf{C}_Q corresponds to the C matrix given in the definition for contour error (4.2), as will be illustrated in the following subsection.

5.4.1 Design Step 1

Using the general time-varying weighting matrix form given in (5.2), there are three key steps to designing a time-varying \mathbf{Q}^{tv} weighting matrix. The first step requires the derivation of the coupling matrix, \mathbf{C}_Q . For the error element of the cost function (5.1), the coupling of the individual error signals comes in the form of an additional error component known as the contour error introduced in Section 4.1. Applying the

lifted approach to the definition given in (4.2) results in the lifted form of contour error,

$$\varepsilon = \mathbf{C} \cdot \mathbf{e}, \quad (5.9)$$

where \mathbf{C} is a lifted matrix containing the trajectory dependent, time-varying coupling gains. The term $\mathbf{C}_Q^T \mathbf{C}_Q$ used in (5.2) can now be written as,

$$\mathbf{C}_Q^T \mathbf{C}_Q = \begin{bmatrix} C^T(\theta, 0)C(\theta, 0) & & 0 \\ & \ddots & \\ 0 & & C^T(\theta, N-1)C(\theta, N-1) \end{bmatrix}, \quad (5.10)$$

where $C^T(\theta, k)C(\theta, k)$ for a 2 DOF system is defined as,

$$C^T(\theta, k)C(\theta, k) = \begin{bmatrix} c_1(\theta, k)c_1(\theta, k) & c_1(\theta, k)c_2(\theta, k) \\ c_2(\theta, k)c_1(\theta, k) & c_2(\theta, k)c_2(\theta, k) \end{bmatrix}. \quad (5.11)$$

Using linearized coupling gains for the 2 DOF system, $\mathbf{C}_Q^T \mathbf{C}_Q$ results in a matrix which is time-varying and block diagonal. A weighting matrix of this form only weights certain combinations of individual error signals rather than each signal equally. By only weighting certain combinations, the system is free to generate different combinations of individual axis errors which minimize the contour errors, while potentially increasing the individual axis errors. This process can be described as effectively decoupling position and time for each individual axis tracking task in order to focus on minimizing the contour tracking errors. [78] illustrates the enhanced trajectory tracking capabilities that result from the use of $\mathbf{C}_Q^T \mathbf{C}_Q$.

With $\mathbf{C}_Q^T \mathbf{C}_Q$ defined in (5.10), the next step in the design process is to determine the gains $\gamma_Q(k)$ and $(1 - \gamma_Q(k))$ for all k which refer to the weighting gains applied to the contour or individual axis tracking, respectively. While generally constant, these gains can be varied throughout the trajectory using shaping criteria based on

the reference trajectory.

5.4.2 Design Step 2

In order to explore the performance benefits of a time-varying \mathbf{Q} weighting matrix, we consider a rastered reference trajectory shown in Figure 5.3. This type of trajectory is commonly used in atomic force microscopy (AFM), as well as other manufacturing systems which require sharp transitions between signals. Sections *A* and *C* of Fig 5.3 correspond to locations where the learning controller focuses on minimizing contour tracking by relinquishing position and time synchronization for each axis, while in section *B* the learning controller is designed to improve individual axis tracking and reestablish position and time synchronization [2]. The high acceleration transition points, identified using circles on Figure 5.3, correspond to locations within the trajectory where the demanding acceleration requirements result in increased trajectory tracking errors as illustrated in Figure 5.4 [80]. These areas indicate potential opportunities for large $\sigma_Q(k)$ gains to provide improved tracking capabilities as compared to small $\sigma_Q(k)$ gains. This will be explored in the third design step.

Consider the trajectory in Figure 5.3. The use of the matrix $\mathbf{C}_Q^T \mathbf{C}_Q$ to enable weighting on the coupled error signals has been shown to result in the most improved contour or trajectory tracking performance for rasters, such as sections *A* and *C* [78]. Focusing on the individual axis errors by selecting the gain $\gamma_Q(k) = 1$ for the gain matrices $\mathbf{\Gamma 1}_Q$ and $\mathbf{\Gamma 2}_Q$ in (5.2) can be used to reestablish position and time synchronization by focusing on minimizing the individual axis errors during the linear sections. In order to maximize the tracking performance, while maintaining position and time synchronization at the start of each raster, the gains are selected as $\gamma_Q(k) = 0$ and $1 - \gamma_Q(k) = 1$ in sections *A* and *C*, while the gains $\gamma_Q(k) = 1$ and $1 - \gamma_Q(k) = 0$ are used in section *B*.

To facilitate smooth transitions between the sections, time-varying vectors con-

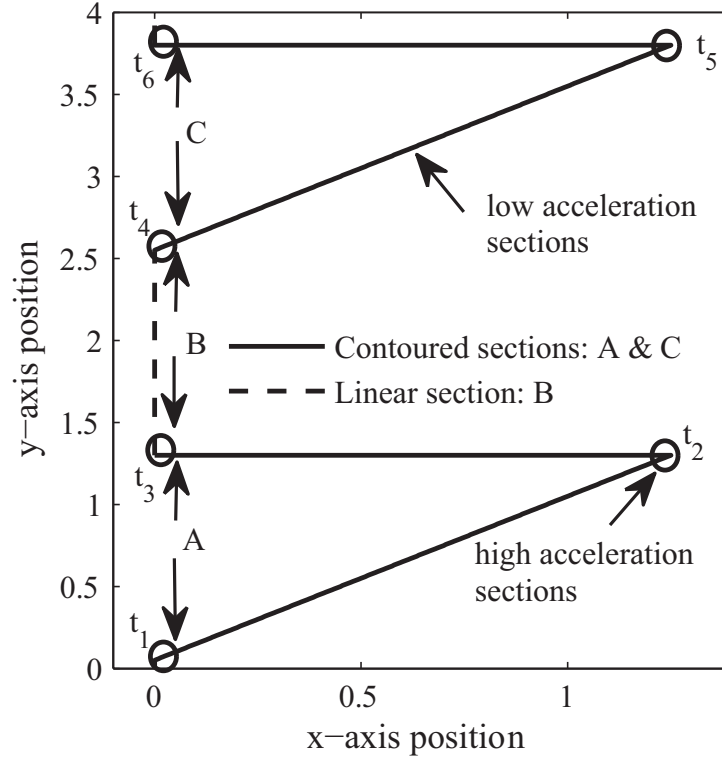


Figure 5.3: Raster trajectory containing linear sections (B), contoured sections (A,C), high acceleration sections ($t_1 - t_6$), and low acceleration sections.

taining $\gamma_Q(k)$ and $1 - \gamma_Q(k)$ for $k = 1, \dots, N - 1$ are filtered using a lowpass Gaussian filter with a bandwidth of 15 Hz. Although any lowpass filter type could be used, in this work we use a Gaussian filter because it is a symmetric filter in which the filter coefficients can be defined with respect to the bandwidth. This results in gain vectors of the form shown in Figure 5.5.

5.4.3 Design Step 3

While the gains $\gamma_Q(k)$ and $1 - \gamma_Q(k)$ focus on weighting the coupled versus individual error signals for a given trajectory, the third step in the design process focuses on establishing weighting on the error signal ($\mathbf{e}_{j+1}(k)$) as a whole. The overall weighting on the error depends on the initial error signal. The locations where the error signal is large correspond to locations in the trajectory that challenge the performance of

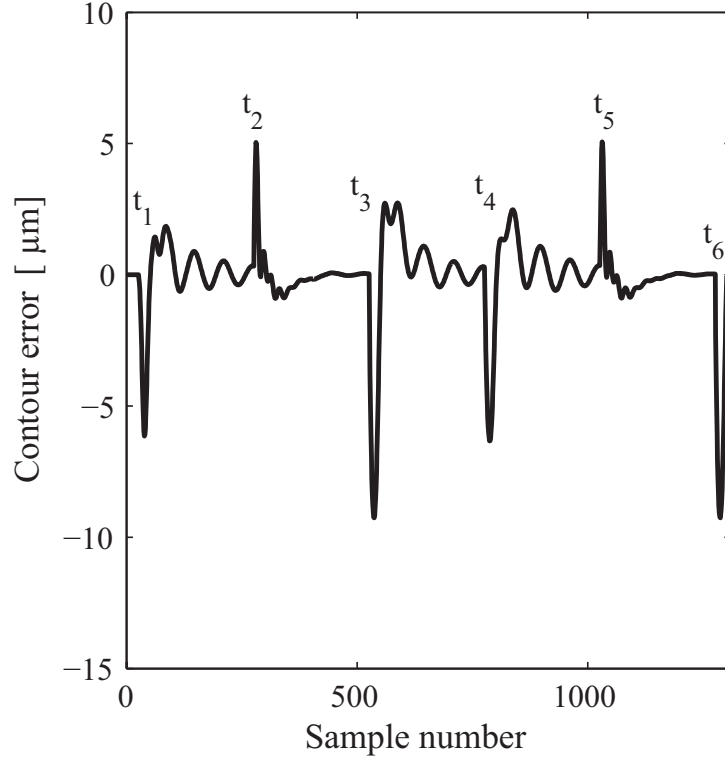


Figure 5.4: Initial contour tracking errors without the use of learning. Individual axis errors show a similar trend in peak locations for initial tracking errors.

the system, i.e. the high acceleration sections ($t_1 - t_6$). For the trajectory given in Figure 5.3, the high and low acceleration components of the reference trajectory can be addressed using the Σ_Q gain matrix.

In a time-invariant norm optimal control design, the scalar weighting σ_Q represents constant performance weighting on the error throughout the entire iteration. Figure 5.4 clearly indicates the locations where increased emphasis on the error (larger σ_Q gains) may result in better trajectory tracking. From this information, the locations where the $\sigma_Q(k)$ gain will be increased from 1 to 30 have been identified as t_1 , t_2 , t_3 , t_4 , t_5 , and t_6 . As with the previous time-varying weighting gains, the transitions between high and low gains are smoothed out using a lowpass Gaussian filter with a 15 Hz bandwidth. The modified time-varying $\sigma_Q(k)$ profile is given in Figure 5.6.

Following general protocol, the baseline value for the overall weighting gain was

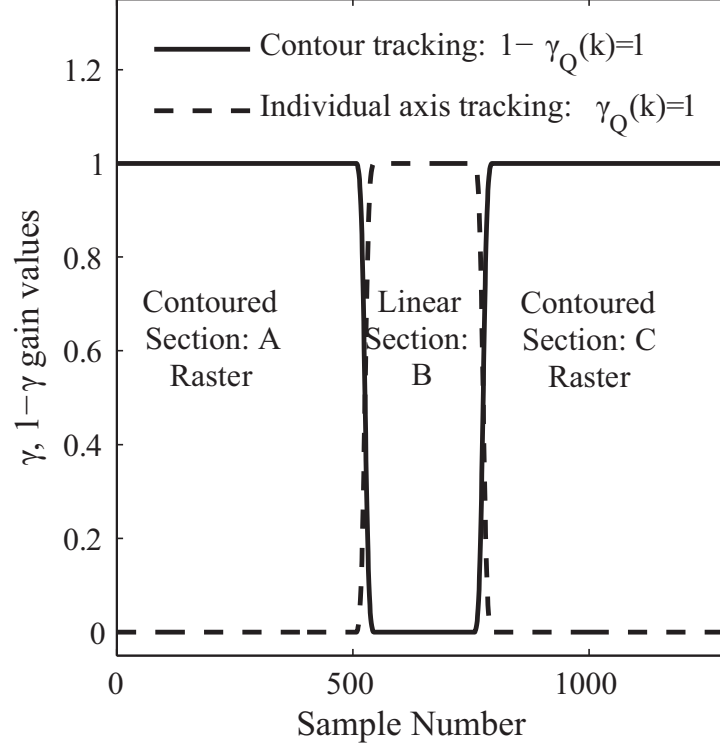


Figure 5.5: Alternating gains $\gamma_Q(k)$ and $1 - \gamma_Q(k)$ to switch between weighting individual versus coupled error signals.

set to $\sigma_Q(k) = 1$. The overall weighting gains $\sigma_S(k)$ and $\sigma_R(k)$ were designed with respect to this baseline value; therefore, increasing $\sigma_Q(k)$ disrupts the relationship between these gains and care must be taken to ensure the controller meets performance and robustness requirements. While the trajectory tracking performance generally improves with the use of time-varying weighting matrices, there may be a tradeoff between tracking in the low versus high acceleration sections. The value of the gain during the high acceleration sections was chosen by increasing the value until the simulation performance of the system began to display transient behavior, and then reducing the gain to provide for a safety factor on the real system. For this system we chose a safety factor of 2.

Combining the three design steps results in a time-varying weighting matrix that addresses individual versus coupled axis errors, as well as overall error performance

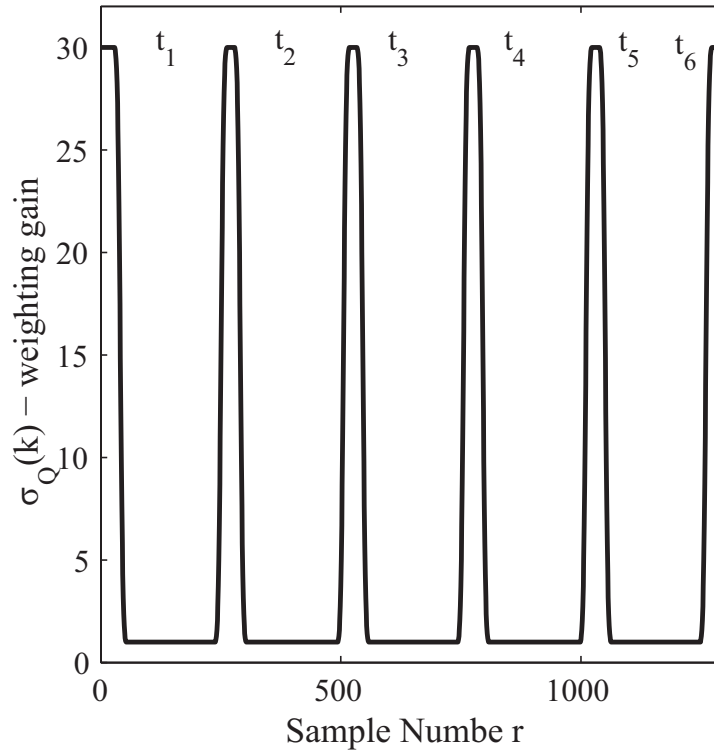


Figure 5.6: Profile for the diagonal elements in the weighting matrix Σ_Q . Notice that the gain is increased in the locations corresponding to the high acceleration sections ($t_1 - t_6$).

requirements.

5.4.4 Simulation Results

In this subsection we apply the time-varying weighting matrix derived from combining the three design steps described in Subsections 5.4.1, 5.4.2, and 5.4.3 on a model of the multi-axis robotic testbed illustrated in Figure 5.7. A parallel ILC architecture (Figure ??) was used in this example. For simulation purposes, 1-kHz sampled dynamic models of the x and y axes, along with stabilizing feedback controllers, were developed in [20]. Numerical values identified for the plant models along with controller coefficients can be found in Appendix A.

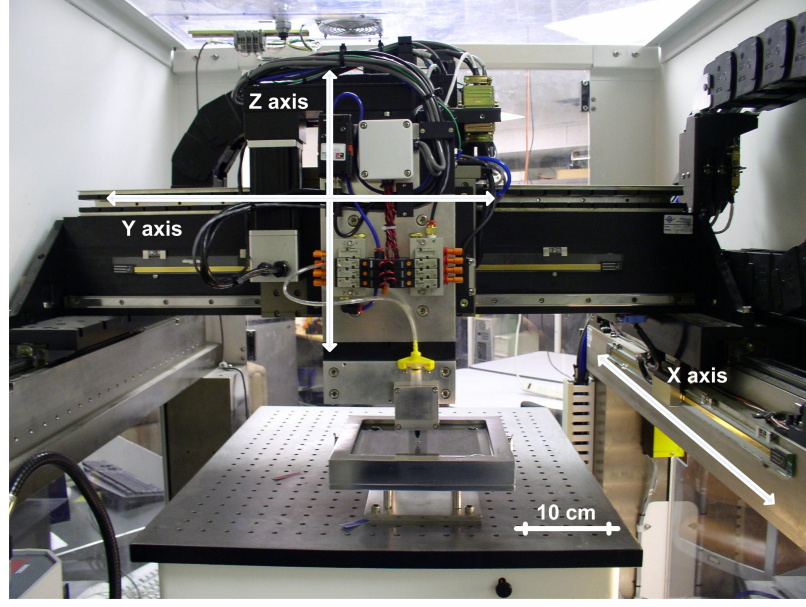


Figure 5.7: Image of the multi-axis robotic testbed used in this work.

$$P_i(z) = \frac{K(z + \alpha_{i1})(z^2 - \alpha_{i2}z + \alpha_{i3})(z^2 - \alpha_{i4}z + \alpha_{i5})}{(z - \beta_{i1})(z - 1)(z^2 - \beta_{i2}z + \beta_{i3})(z^2 - \beta_{i4}z + \beta_{i5})}, \quad i = x, y. \quad (5.12)$$

$$\text{feedback controller} \triangleq k_{pi}(z) = \frac{k(z - \alpha_{i1})(z - \alpha_{i2})(z - \alpha_{i3})}{(z - \beta_{i1})(z - \beta_{i2})(z - \beta_{i3})}, \quad i = x, y. \quad (5.13)$$

The following results were obtained using the stabilized dynamic models from (5.12) and (5.13), the reference trajectory from Fig 5.3 ($N = 1300$), and the norm optimal controllers (3.11) with \mathbf{Q}^{tv} replacing \mathbf{Q} . Using the tuning guidelines from section 3.3.4 and designing for the multi-axis system of Figure 5.7, the scalar gains for $\mathbf{S} = s\mathbf{I}$ and $\mathbf{R} = r\mathbf{I}$ were heuristically chosen as ($s = 1e^{-2}, r = 2e^{-2}$).

Figure 5.8 illustrates the effect of using time-varying weighting gains, $\gamma_Q(k)$, $1 - \gamma_Q(k)$, and $\sigma_Q(k)$ on the RMS contour error, as compared to basic feedback control and a norm optimal controller with time-invariant gains ($\gamma_Q = 0, 1 - \gamma_Q = 1, \sigma_Q = 1$). Figure 5.9 shows the improvement in the contour tracking at the corners as a result of high $\sigma_Q(k)$ weighting gains at these particular locations. Figure 5.10 demonstrates

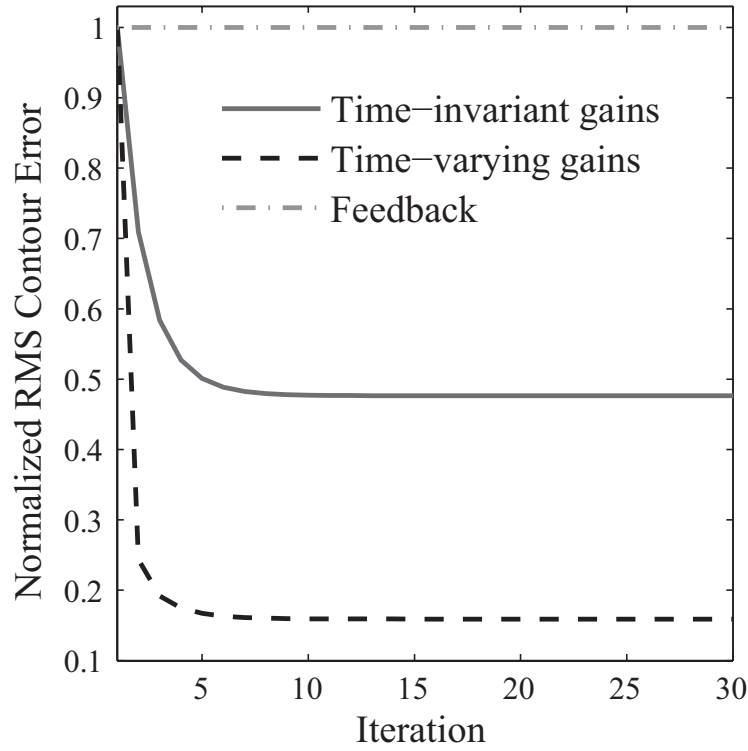


Figure 5.8: Comparison of RMS contour errors for feedback, norm optimal control using time-invariant $(\gamma_Q, 1 - \gamma_Q, \sigma_Q)$ gains, and norm optimal control using time-varying $(\gamma_Q(k), 1 - \gamma_Q(k), \sigma_Q(k))$ gains [Simulation].

the improvement in the individual y-axis tracking, and therefore enhanced position and time synchronization, at the locations where the controller switches from focusing on contour tracking to individual axis tracking by changing the weighting gains from $(\gamma_Q(k) = 0, 1 - \gamma_Q(k) = 1)$ to $(\gamma_Q(k) = 1, 1 - \gamma_Q(k) = 0)$.

Figures 5.8-5.10 clearly indicate the performance improvements obtained by implementing a norm optimal learning controller using time-varying weighting matrix gains, $\gamma_Q(k)$, $1 - \gamma_Q(k)$, and $\sigma_Q(k)$, in simulation.

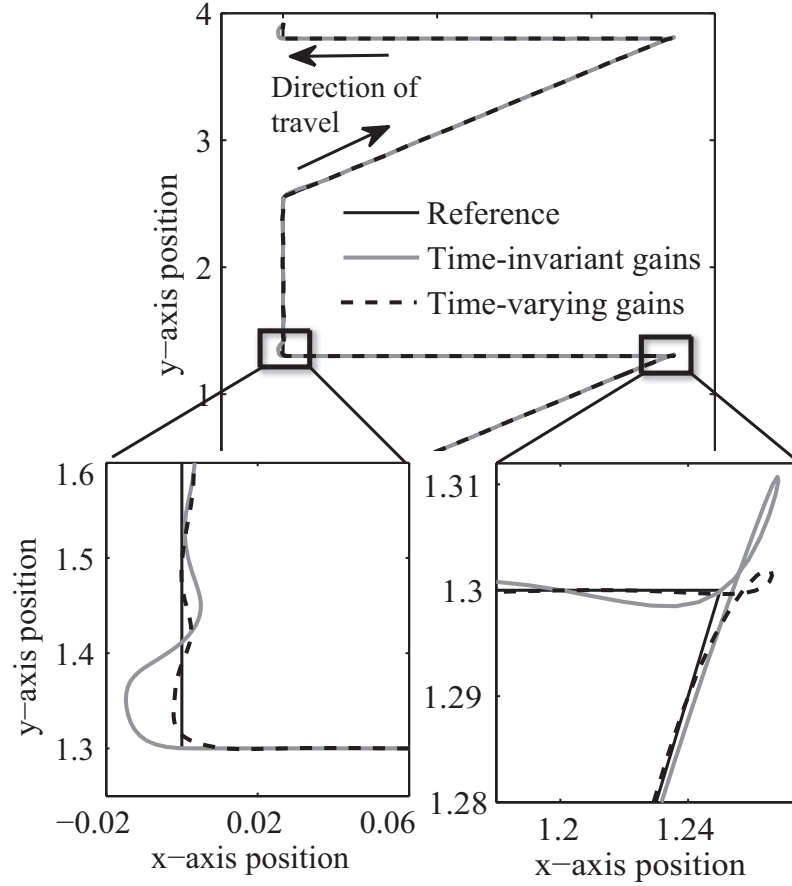


Figure 5.9: Trajectory tracking comparison of the norm optimal controllers with time-invariant and time-varying weighting gains. Notice that the controller using time-varying gains produces tighter tolerances around the corners as a result of increasing the gain at specific locations [Simulation].

5.5 Design Example 2: Time-Varying \mathbf{S}

Weighting Matrix

The previous section presented a technique for designing a time-varying weighting matrix for performance benefits. An equally important aspect in control design is ensuring robustness of the controller. This section focuses on implementing a time-varying \mathbf{S} weighting matrix in order to provide robustness in the presence of position dependent dynamics.

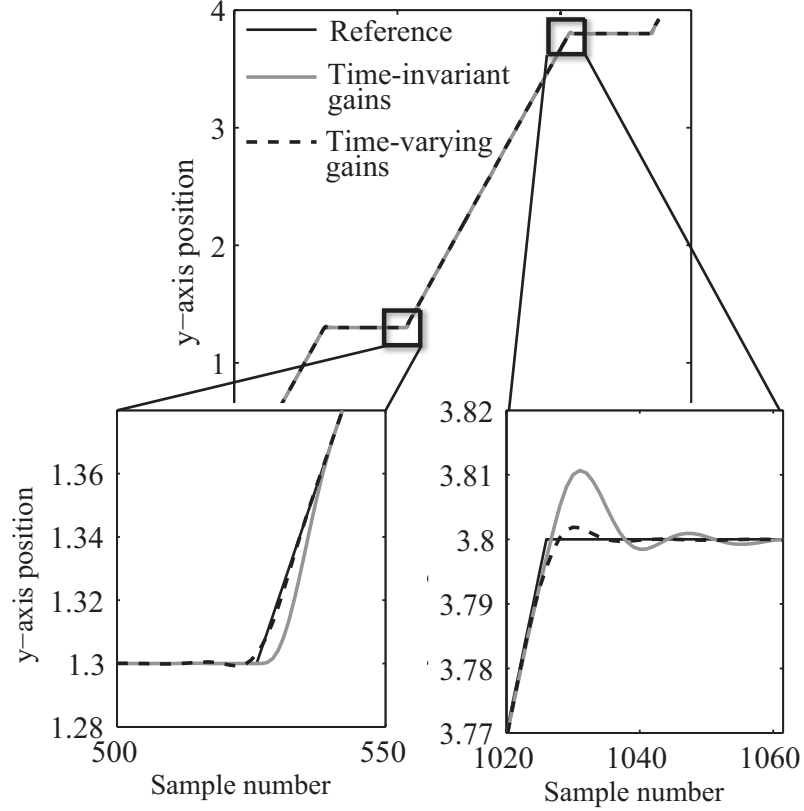


Figure 5.10: Tracking performance for the y-axis. Notice the reduction in the error resulting from switching the weighting gains from $(\gamma_Q(k) = 0, 1 - \gamma_Q(k) = 1)$ to $(\gamma_Q(k) = 1, 1 - \gamma_Q(k) = 0)$ [Simulation].

5.5.1 Motivation

Using analysis provided in [72,78] it can be shown that the \mathbf{S} weighting matrix should be designed to ensure robust monotonic convergence in the presence of model uncertainty. Assuming a weighting matrix of the form $\mathbf{S} = s\mathbf{I}$, the weighting gain s provides constant weighting for uniform model uncertainty. However, in some applications, the dynamics are position dependent [81]. For applications which extend into locations with different dynamics, a time-varying weighting matrix of the form provided in (5.3) enables the controller to adequately address the model uncertainty at each location. As with the design of \mathbf{Q}^{tv} , \mathbf{S}^{tv} has the versatility to consider coordination of the control signals, as well as each individual signal separately.

In many manufacturing applications, the system contains position dependent dy-

namics [82–84]. Often times these differences in dynamics are greatest at the edge of the system workspace [85]. System identification generally occurs in the center of the workspace, resulting in dynamics models which are less certain at the outer limits of the workspace. For these systems, increasing the value of the $\sigma_S(k)$ gain at the locations with more model uncertainty provides more robustness against the position-varying dynamics.

Consider a multi-axis system with potential x -axis position dependent dynamics illustrated in Figure 5.11 [80, 82]. Resonance shifting in any axis will have similar results. For this example, the system resonances are shifted depending on the position of the axis during the trajectory. For these types of systems, time-varying designs which enable the controller to compensate for model uncertainty due to position shifting dynamics at specific time locations in the input signal are a reasonable choice. The design of a time-varying \mathbf{S}^{tv} weighting matrix is described in the following subsection.

5.5.2 Weighting Matrix Design

Without loss of generality, the weighting matrix design in this section is focused on individual control signals. Therefore, design step 1 is not necessary because design step 2 sets $\gamma_S(k) = 1$ and $1 - \gamma_S(k) = 0$ for all $k = 1, 2, \dots, N - 1$.

Assume a continuous reference trajectory in which the system performs a task at one location, moves to a different location for an additional task, and then returns to the start location. The objective is to design a time-varying weighting matrix gain $\sigma_S(k)$ that increases in value at the locations where the position dynamics have shifted and therefore the model contains some additional uncertainty. Figure 5.12 gives an example of such a trajectory, while Figure 5.13 illustrates the heuristically determined time-varying weighting matrix gain vector $\sigma_S(k)$ for $k = 1, \dots, N - 1$ associated with this trajectory and uncertain system dynamics. The transitions between the high

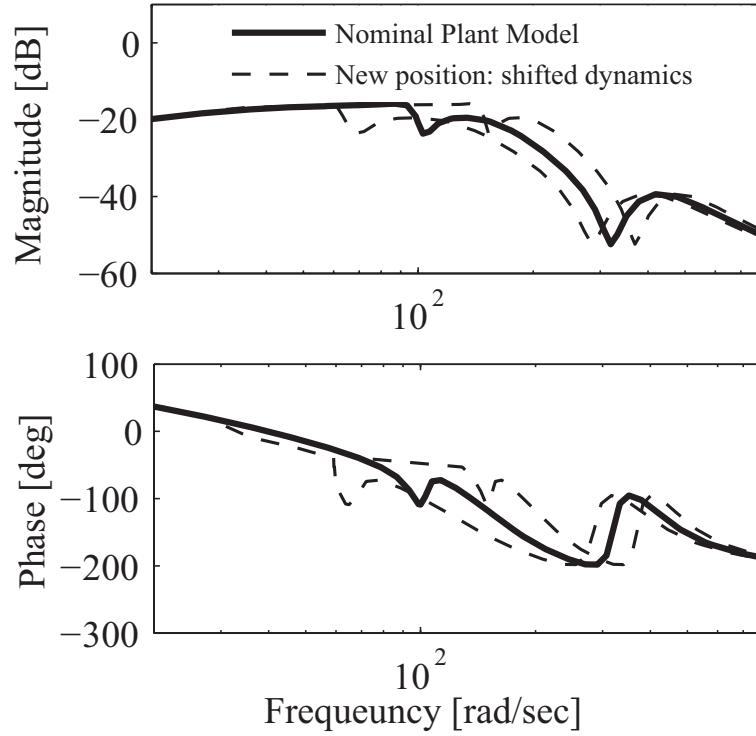


Figure 5.11: Simulated illustration of a nominal x-axis plant model with two examples of potential resonance shifting in the plant dynamics.

and low values for the gain have been smoothed using a lowpass Gaussian filter with a bandwidth of 5 Hz. The weighting matrix gains $\sigma_S(k)$ chosen heuristically in this example should satisfy (3.20) for a given uncertainty, so that the system is robustly monotonically convergent in the presence of the unmodelled dynamics.

5.5.3 Simulation Results

This subsection implements the time-varying \mathbf{S} weighting matrix design from Figure 5.13 to address position dependent dynamics. The system was subjected to a multiplicative uncertainty which mimics position dependent dynamics with a high degree of uncertainty at the position corresponding to x, y from $N = 800$ to $N = 1200$. The uncertainty function can be found in Appendix B. The \mathbf{Q}^{tv} and \mathbf{R} weighting matrices were set to $(\gamma_Q(k) = 0, 1 - \gamma_Q(k) = 1, \sigma_Q(k) = 1)$ for $k = 1, 2, \dots, N - 1$ and $\mathbf{R} = r\mathbf{I}$

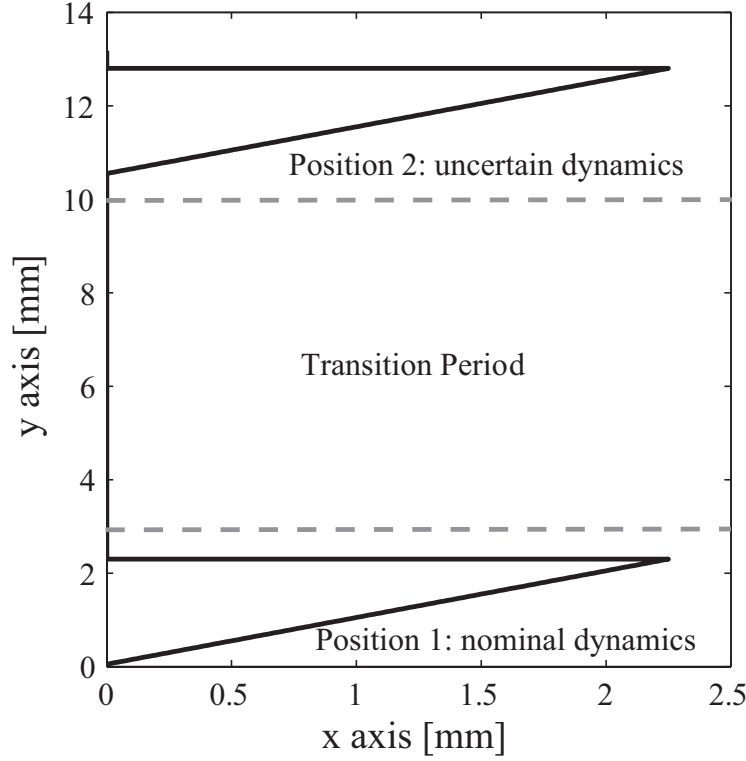


Figure 5.12: Reference trajectory in which a raster occurs at two distinct locations with the second position corresponding to a location with increased model uncertainty.

with a heuristically chosen time-invariant scalar gain $r = 2e^{-2}$, respectively. Combining the time-varying gains $\sigma_S(k)$ from Figure 5.13 and the \mathbf{Q}^{tv} and \mathbf{R} matrices defined above, a time-varying norm optimal learning controller was designed. Using this time-varying learning controller, along with the reference trajectory from Figure 5.12 ($N = 1200$), the following results were obtained.

Figure 5.14 presents the normalized RMS contour errors for norm optimal controllers designed using high time-invariant σ_S , low time-invariant σ_S , and time-varying $\sigma_S(k)$ gain values. The time-invariant gains ($\gamma_S = 1, 1 - \gamma_S = 0, \sigma_S = \text{constant}$) in \mathbf{S}^{tv} are equivalent to using the gain s in the time-invariant format $\mathbf{S} = s\mathbf{I}$. As the figure illustrates, low time-invariant σ_S gain values in the presence of position dependent model uncertainty result in an unstable system, while high time-invariant σ_S gain values produce a stable system that converges to a larger RMS contour error than

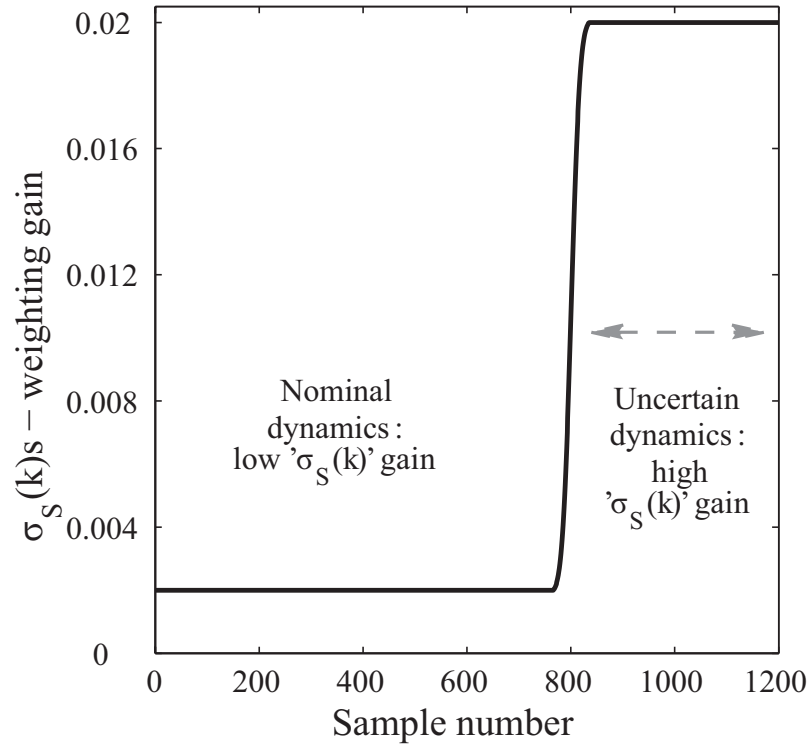


Figure 5.13: Heuristically determined time-varying gains $\sigma_S(k)$. Notice that the gain is increased in the location at which the dynamics are uncertain. The uncertainty in the dynamics leads to an increase in model uncertainty.

the time-varying $\sigma_S(k)$ design. These results indicate how the use of a time-varying \mathbf{S}^{tv} weighting matrix results in a more robust system which converges to lower RMS contour errors in the presence of position dependent dynamics.

The next section presents a design example for a time-varying \mathbf{R} weighting matrix.

5.6 Design Example 3: Time-Varying \mathbf{R} Weighting Matrix

The previous section presented a technique for designing a time-varying weighting matrix for robustness to position dependent dynamics. This section focuses on implementing a time-varying weighting matrix in order to maintain robustness and

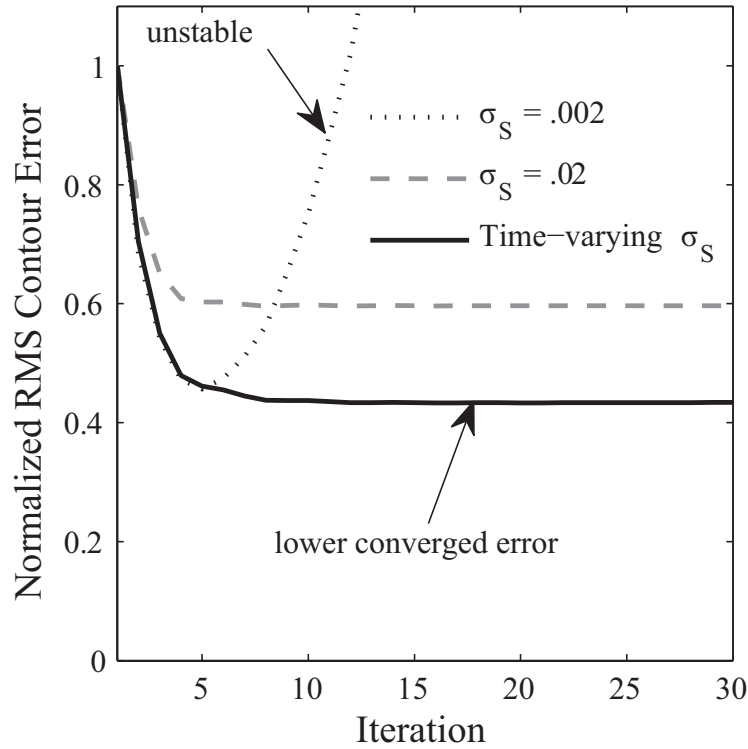


Figure 5.14: Normalized contour errors for systems with position dependent dynamics [Simulation].

performance in the presence of position and time dependent external stochastic disturbances or noise.

5.6.1 Motivation

In this subsection, we consider performance in the presence of external stochastic disturbances or noise. As is shown in [73], the influence of stochastic disturbances can be minimized by reducing the convergence speed. In [78] the dominating factor in convergence speed was shown to be the \mathbf{R} weighting matrix. While a constant weighting gain r in $\mathbf{R} = r\mathbf{I}$ provides consistent influence on the effect of stochastic disturbances, many applications include external disturbances and noise that change depending on time or position. For these cases, designing a time-varying weighting

matrix of the form illustrated in (5.4) results in a more robust controller that is capable of handling different types of disturbances without requiring overly long convergence times. The design of a time-varying \mathbf{R}^{tv} weighting matrix is demonstrated in the following subsection.

5.6.2 Weighting Matrix Design

Without loss of generality, the design example for \mathbf{R}^{tv} presented here only considers individual signals, rather than coordination between the signals. Therefore, design step 1 can be skipped as a result of design step 2 being simplified to setting $\gamma_R(k) = 1$ and $1 - \gamma_R(k) = 0$ for $k = 1, 2, \dots, N - 1$.

Consider a MIMO system in which an unknown stochastic disturbance occurs at a specific location or time during a given trajectory. An example could be a spot welding or laser cutting application where external electromagnetic interferences occur at discrete locations due to the on/off modes for these processes [86]. In applications which require mass production, the on/off modes and subsequently the locations of the time and position varying external disturbances repeat each iteration. While the occurrence of these disturbances can be predicted, the stochastic nature of the signal does not allow the system to *learn* the disturbances from iteration to iteration. If a time-invariant controller is designed too aggressively, the presence of the external stochastic disturbances may cause the converged error signal to fluctuate drastically, thus reducing the performance of the system. However, if a more conservative time-invariant controller is used, the system may experience long convergence times. For these types of systems with discrete external disturbances, a time-varying \mathbf{R}^{tv} weighting matrix design enables the controller to handle the time-dependent disturbances at specific times without forcing the controller to be overly conservative or too aggressive throughout the trajectory.

Figure 5.15 shows a raster trajectory in which the system is subject to some

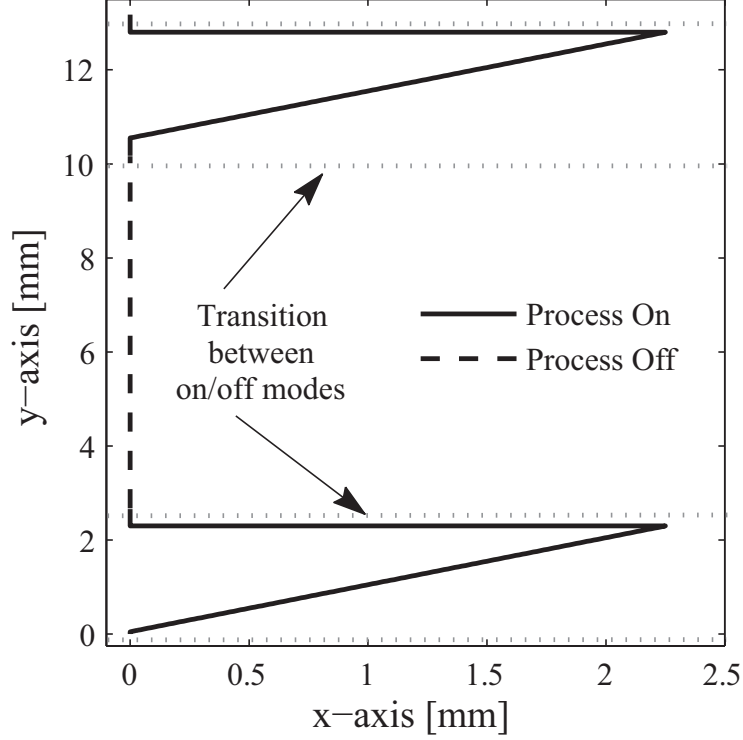


Figure 5.15: Raster trajectory in which the process undergoes intermittent on/off modes. The on/off transitions lead to an external stochastic disturbance that is repeated at predictable intervals.

process on/off mode, which introduces an external interference, four times during a single time period. Figure 5.16 presents the time-varying weighting matrix gain $\sigma_R(k)$ associated with the given trajectory.

The time-varying weighting matrix gains $\sigma_R(k)$ are designed heuristically to ensure robustness and performance in the presence of external stochastic disturbances, while maintaining a reasonable convergence rate η . The transition between high and low gains is filtered using a lowpass Gaussian filter with a 15 Hz bandwidth.

5.6.3 Simulation Results

To validate system robustness and performance in the presence of position dependent stochastic disturbances, the time-varying \mathbf{R}^{tv} weighting matrix designed in the pre-

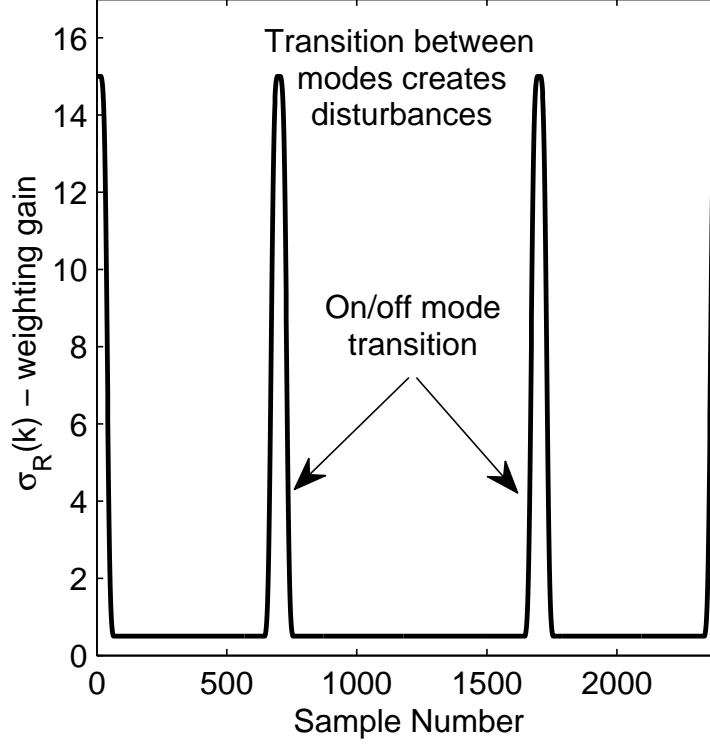


Figure 5.16: Time-varying weighting matrix gains $\sigma_R(k)$ for the raster trajectory in Figure 5.15. Notice how the gain increases at the locations corresponding to the on/off modes which have led to the introduction of external stochastic disturbances.

vious subsection is implemented on a model of the robotic testbed in Figure 5.7. For this example, a Gaussian white noise disturbance was introduced to the simulation at the specific position intervals which corresponded with the on/off locations depicted in Figure 5.15. In order to determine the performance and robustness benefits of a time-varying \mathbf{R}^{tv} weighting matrix, the other two weighting matrices were set to time-invariant gains ($\gamma_Q = 0, 1 - \gamma_Q = 1, \sigma_Q = 1$) for \mathbf{Q}^{tv} and the heuristically chosen time-invariant scalar gain $s = 5e^{-1}$ for $\mathbf{S} = s\mathbf{I}$. Using these \mathbf{Q}^{tv} and \mathbf{S} weighting matrices, along with the time-varying gains $\sigma_R(k)$ from Figure 5.16 to design a time-varying norm optimal controller and the reference trajectory from Figure 5.15 ($N = 1200$), the following results were obtained.

Figure 5.17 presents the normalized RMS contour errors for norm optimal con-

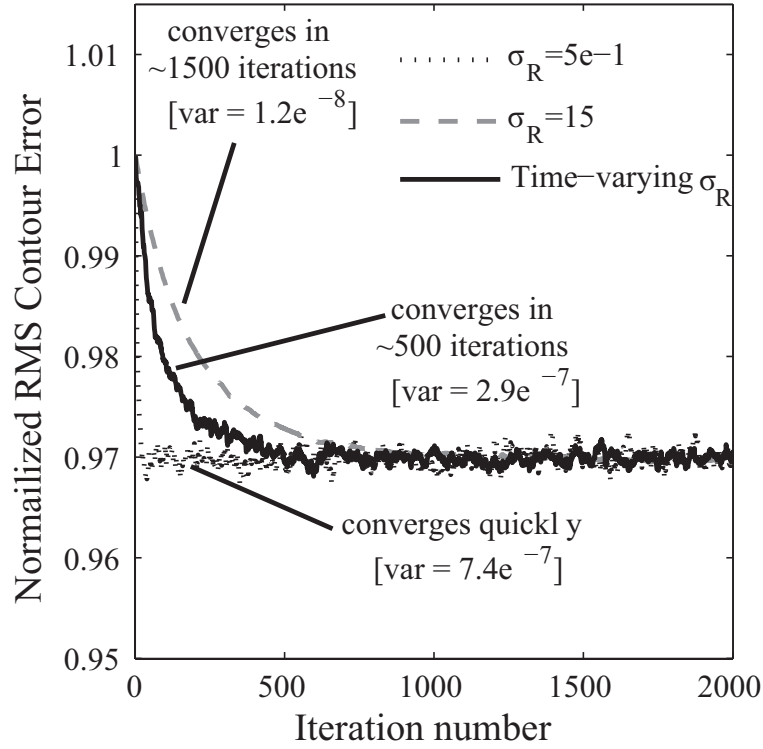


Figure 5.17: Normalized contour errors for systems with time or position dependent stochastic disturbances. Variances of the converged error signals have been included in the figure. Note that the presence of a large magnitude stochastic disturbance signal results in a reduction in the overall performance [Simulation].

trollers designed using high time-invariant σ_R , low time-invariant σ_R , and time-varying $\sigma_R(k)$ gain values. Note that due to the presence of stochastic external disturbances a more conservative \mathbf{S} weighting matrix was designed to ensure convergence. The increase in the gain s resulted in a reduction in overall performance. As the figure illustrates, low time-invariant σ_R gain values in the presence of time or position dependent external disturbances result in a system which exhibits a highly fluctuating converged error signal, while high time-invariant σ_R gain values produce a more conservative system with a very slow convergence rate. The time-varying $\sigma_R(k)$ gain value design results in a system that converges to an error signal with smaller fluctuations than low σ_R at a faster convergence rate than high σ_R . These results indicate how the use of a time-varying \mathbf{R}^{tv} weighting matrix results in a more robust

system with a faster convergence rate and a less oscillatory converged error signal in the presence of time or position dependent stochastic disturbances.

5.7 Experimental Validation of Time-Varying \mathbf{Q}^{tv}

Sections 5.4-5.6 presented simulation results for different cases of the general time-varying weighting matrix designs. Here we present experimental results for the particular time-varying case where the original norm optimal design is made time-varying by changing the cost function to include time-varying weighting on the error signal. The simulation results from Section 5.4 are validated by implementing norm optimal learning controllers using time-invariant and time-varying gains $(\gamma_Q, 1 - \gamma_Q, \sigma_Q)$ on the experimental testbed from Figure 5.7. For this particular system, the need for time-varying \mathbf{S}^{tv} and \mathbf{R}^{tv} weighting matrices is not present. Analogous to the simulation results, the norm optimal controller using time-varying gains results in the most improved tracking performance as illustrated in Figure 5.18 and Figure 5.19.

In Figure 5.18, a norm optimal learning controller using time-varying $(\gamma_Q(k), 1 - \gamma_Q(k), \sigma_Q(k))$ weighting gains produces the lowest normalized RMS contour tracking errors as compared to a norm optimal controller with time-invariant gains and a Feedback controller with a 32% reduction from the norm optimal controller using time-invariant to time-varying gains. The trajectory tracking performance improvements resulting from this reduction in RMS contour error can be seen in Figure 5.19. These results indicate how time-varying $\gamma_Q(k)$, $1 - \gamma_Q(k)$, and $\sigma_Q(k)$ weighting gains result in a controller with more precise tracking for this particular trajectory.

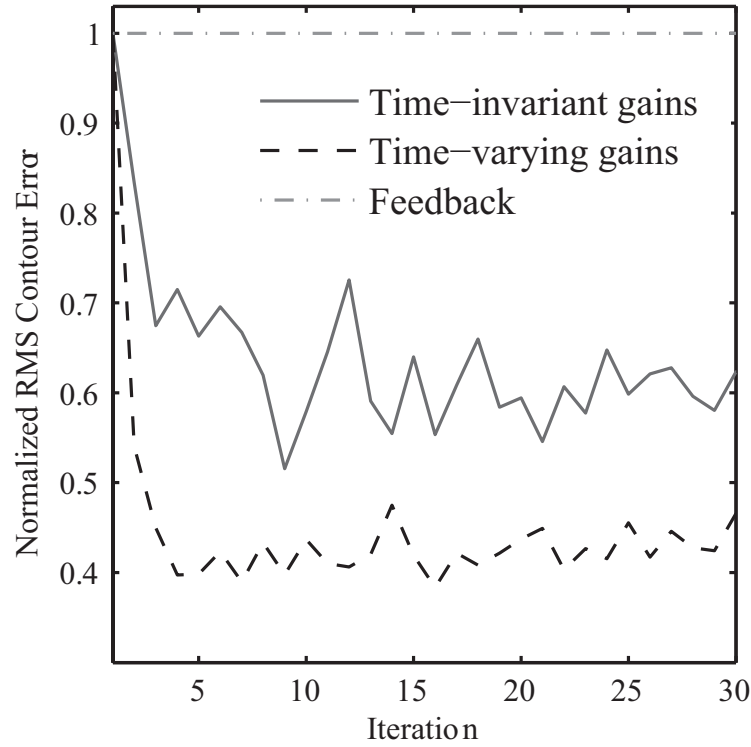


Figure 5.18: Comparison of RMS contour errors for Feedback, norm optimal control using time-invariant weighting gains and norm optimal control using time-varying weighting gains [Experimental]

5.8 Discussion

This chapter presented time-varying weighting matrix designs for coupled ILC controllers for multi-axis systems. Explicit design steps demonstrate that time-varying weighting matrices provide a means for improving both performance and robustness of a given system.

Using the four step tuning guidelines and the four step time-varying weighting matrix design approach detailed in the chapter, norm optimal learning controllers using time-invariant and time-varying weighing gains $(\gamma_{(\cdot)}(k), 1 - \gamma_{(\cdot)}(k), \sigma_{(\cdot)}(k))$ were designed for comparison in simulation on a multi-axis robotic testbed. Simulation and experimental results showed that a norm optimal controller with time-varying gains in the \mathbf{Q}^{tv} weighting matrix improves the trajectory tracking performance of a

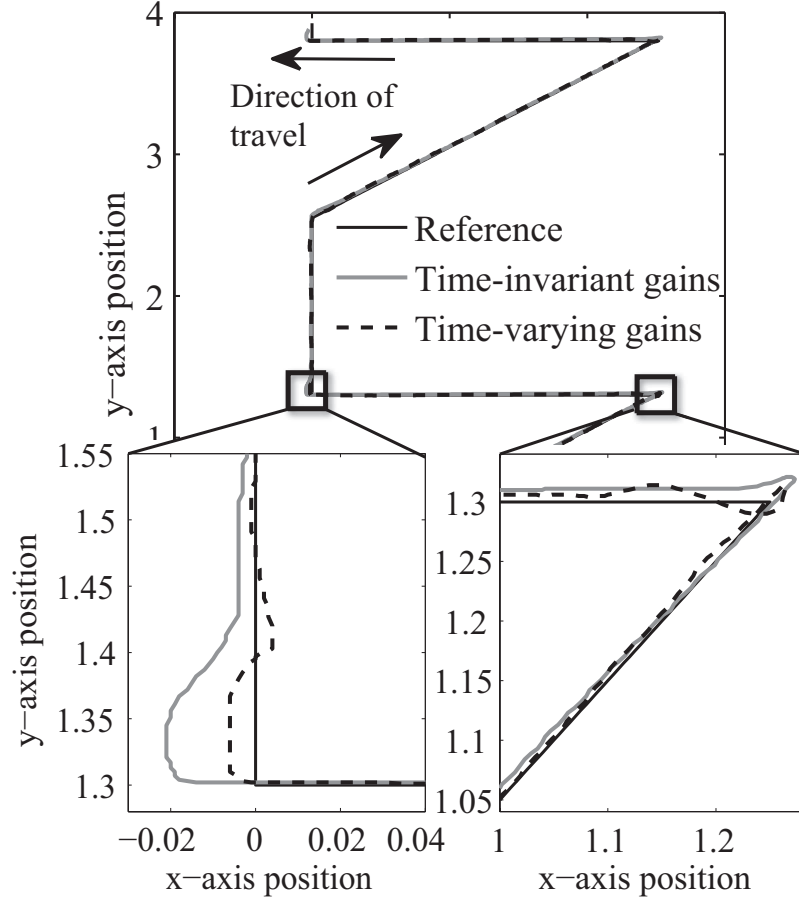


Figure 5.19: Trajectory tracking comparison of norm optimal controllers using time-invariant and time-varying weighting gains. Notice that the controller using time-varying weighting gains produces tighter tolerances around the corners as a result of increasing the gain $\sigma_Q(k)$ at specific locations. [Experimental]

MIMO system over a norm optimal design using time-invariant gains $\gamma_Q, 1 - \gamma_Q, \sigma_Q$. Simulation results for the \mathbf{S}^{tv} weighting matrix, which focuses on robustness issues, illustrated that an \mathbf{S}^{tv} weighting matrix with time-invariant $\gamma_S = 1, 1 - \gamma_S = 0$ and time-varying $\sigma_S(k)$ gains stabilizes a system with dynamic uncertainty, while time-invariant σ_S weighting gains result in either larger converged errors or an unstable system. Finally, a time-varying \mathbf{R}^{tv} weighting matrix with time-invariant $\gamma_R = 1, 1 - \gamma_R = 0$ and time-varying $\sigma_R(k)$ gains resulted in a system that converged to an error signal with less fluctuations and a faster convergence rate in the presence of position and time-varying external stochastic disturbances, as compared to time-

invariant σ_R weighting gain designs.

The results presented in this chapter demonstrate the performance and robustness benefits obtained from applying coupled learning controllers to problems with planar manufacturing robots in which both axes can be characterized as similar systems; having similar dynamics and identical hardware. However, there are many repetitive applications in which dynamically dissimilar systems cooperate to pursue a primary performance objective. Work in [87] introduced a novel framework to couple dynamically dissimilar systems while applying ILC, showing the ability to noncausally compensate for a slow system with a fast system. In this framework, performance requirements for a primary objective can more readily be achieved by emphasizing an underutilized fast system instead of straining a less-capable slow system. The controller is applied to a micro-Robotic Deposition (μ -RD) manufacturing system to coordinate a slow extrusion system axis and a fast positioning system axis to pursue the primary performance objective, dimensional accuracy of a fabricated part. A more detailed presentation of this material can be found in Appendix J.

Chapter 6

3-D Coordinated Iterative Learning Control

Previous work in Chapter 5 introduced a coupled norm optimal ILC design which reformats the general norm optimal framework to enable the controller to focus on improving the trajectory tracking performance and robustness of a multi-axis system. The objective of the work presented in this chapter is to extend the norm optimal design strategies to include individual and group objectives in an effort to improve trajectory tracking performance and group formation coordination through the use of modified weighting matrices in the norm optimal framework. The generalized structure for the modified framework is given in the following sections.

6.1 Introduction

Multiple system control is comprised of a multitude of independent systems, which are coupled together through a common desired outcome. As the drive to enhance efficiency and positioning from the macro to the nano-scale increases, the ability to improve system coordination and precision motion control becomes more critical. This chapter presents a method for improving precision coordination and motion control of multiple multi-input multi-output (MIMO) systems that execute the same task repetitively. Examples of multiple systems coupled through a common outcome which perform the same task repetitively can be found in manufacturing applications [15, 20, 21], surveying [88, 89], and agricultural applications such as crop dusting and spraying, [23].

A general approach for controlling the positioning of multiple systems is to implement individual controllers on each independent agent. However, one of the unique advantages of working with MIMO systems in which the desired output is coupled in the form of a particular formation is the ability to focus on the coordination of these agents as a group objective, in addition to the individual objectives of trajectory tracking. The formation-keeping approach has been inspired by observations from nature, such as flocking or schooling. Majority of the work in this area has focused on developing optimal and robust feedback controllers in the time domain [11, 54, 55, 90]. More recent approaches have included the use of Iterative Learning Control [18, 19]. However, most methods which enable one to focus on tracking and coordination of a combined MIMO system require different control design strategies depending on the formation and the objectives of the combined system. Previous results in [2] demonstrated performance improvements obtained through the use of a controller which coupled individual systems through the error signals. This coupled approach makes the control input, and thereby the system output, dependent on the performance of the other systems. The work presented in this chapter seeks to extend the idea of coupling multiple systems through a common desired output and formation.

The primary objective of the chapter is to present a novel control methodology for precision coordination and motion control of multiple systems which perform the same task repetitively. More specifically, this chapter will focus on 1) the development of a generalized structure for an iterative learning controller which incorporates a coordinated approach to improving the performance of multiple systems, and 2) a description of a design methodology for generating this coordinated iterative learning controller.

6.2 Coordinated Weighting Matrices

Recall from Chapter 3 that the objective of the norm optimal framework is to minimize an objective \mathcal{J} ,

$$\mathcal{J} = \mathbf{e}_{j+1}^T \mathbf{Q} \mathbf{e}_{j+1} + \mathbf{u}_{j+1}^T \mathbf{S} \mathbf{u}_{j+1} + (\mathbf{u}_{j+1} - \mathbf{u}_j)^T \mathbf{R} (\mathbf{u}_{j+1} - \mathbf{u}_j),$$

where the weighting matrices $\{\mathbf{Q}, \mathbf{S}, \mathbf{R}\}$ are designed to satisfy the existence and convergence requirement $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R} > 0$.

An essential part of the design process involves determining the weighting matrices $\{\mathbf{Q}, \mathbf{S}, \mathbf{R}\}$. In this section, the format of the weighting matrices is modified to enable one to focus on individual position tracking objectives, a group formation objective, or some combination of the two.

6.2.1 Individual System Control Design

As stated in Chapter 3, the weighting matrices are generally of the form $\{\mathbf{Q}, \mathbf{R}, \mathbf{S}\} \equiv \{q\mathbf{I}, r\mathbf{I}, s\mathbf{I}\}$. While this approach works well for time-invariant unmodelled dynamics and external disturbances, previous work in [51] demonstrated performance and robustness improvements from implementing time-varying weighting matrices aimed at addressing time and position dependent disturbances, dynamics, and tracking errors. Applying the design framework described in [51] and Chapter 5, a norm optimal \mathbf{Q} weighting matrix for independent multi-axis agents can be defined as,

$$\mathbf{Q}_{1,2,\dots,p} = [\mathbf{\Gamma} \mathbf{1}_Q + \mathbf{\Gamma} \mathbf{2}_Q \cdot \mathbf{C}_Q^T \mathbf{C}_Q], \quad (6.1)$$

where $1, 2, \dots, p$ identifies the individual multi-axis agent within the combined MIMO system. Note that norm optimal \mathbf{S} and \mathbf{R} matrices for independent agents would be of the same form as $\mathbf{Q}_{1\dots p}$.

In (6.1), the \mathbf{C}_Q matrix corresponds to the coupling matrix used to define contour error with respect to the individual axis errors as a function of the reference trajectory. The matrices $\mathbf{\Gamma}\mathbf{1}_Q$ and $\mathbf{\Gamma}\mathbf{2}_Q$ refer to the amount of weighting applied to the coupled or individual signals, respectively. These matrices are of the form provided in (6.2), where the inner block diagonal matrices are shown for a 2 DOF system and the gains are defined as $\gamma_1(k) = \gamma(k)$ and $\gamma_2(k) = 1 - \gamma(k)$, respectively.

$$\mathbf{\Gamma}\mathbf{j} = \begin{bmatrix} \begin{bmatrix} \gamma_j(1) & 0 \\ 0 & \gamma_j(1) \end{bmatrix} & & 0 \\ & \ddots & \\ 0 & & \begin{bmatrix} \gamma_j(N) & 0 \\ 0 & \gamma_j(N) \end{bmatrix} \end{bmatrix}_{j=1,2} \quad (6.2)$$

The gain $\gamma(k)$ is used to determine the weighting applied to the individual and coupled signals, respectively. From (6.2), $(\gamma(k) = 1)$ refers to all of the weighting being applied to the individual signals, while $(\gamma(k) = 0)$ results in only the coupled signals being weighted.

6.2.2 Coupled Agent Control Design

After all the agent weighting matrices have been determined, the individual trajectory tracking objective, in terms of a coupled versus individual agent approach, needs to be addressed. Using a similar form to that presented in 6.2.1, the norm optimal weighting matrix for coupled versus individual system error tracking can be defined,

$$\bar{\mathbf{Q}} = [\mathbf{B}\mathbf{1}_Q \cdot \mathbf{Q} + \mathbf{B}\mathbf{2}_Q \cdot \mathbf{K}_Q^T \mathbf{Q}_{FC} \mathbf{K}_Q]. \quad (6.3)$$

Note that $\bar{\mathbf{S}}$ and $\bar{\mathbf{R}}$ are matrices of the same form as $\bar{\mathbf{Q}}$. In (6.3), \mathbf{Q} is a diagonal

matrix containing the individual multi-axis agent weighting matrices ($\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_p$) along the diagonal, while \mathbf{Q}_{FC} is a nominal weighting matrix of the form provided in (6.1) designed for formation-center tracking. Matrices $\mathbf{B1}$ and $\mathbf{B2}$ are used to turn off/on the weighting on the individual and coupled agent tracking, respectively. Setting ($\mathbf{B1} = \mathbf{I}, \mathbf{B2} = \mathbf{0}$) results in individual agent tracking, whereas coupled formation-center tracking is achieved by setting ($\mathbf{B1} = \mathbf{0}, \mathbf{B2} = \mathbf{I}$). Alternative formation designs may be achieved through specific designs of $\mathbf{B1}$ with $\mathbf{B2} = \mathbf{0}$.

\mathbf{K}_Q is a non-square matrix containing the time-invariant gains used to define the formation center with respect to the individual agents within the combined MIMO system and is of the form shown in (6.4), where each agent is assumed to have 2 DOF.

$$\mathbf{K} = \left[\begin{bmatrix} \kappa_1(1) & & 0 \\ & \ddots & \\ 0 & & \kappa_1(N) \end{bmatrix} \dots \begin{bmatrix} \kappa_p(1) & & 0 \\ & \ddots & \\ 0 & & \kappa_p(N) \end{bmatrix} \right] \quad (6.4)$$

Recall that $1, 2, \dots, p$ identifies the individual agents. The diagonal elements are given as,

$$\kappa_{(\cdot)}(k) = \begin{bmatrix} \kappa x(k) & 0 \\ 0 & \kappa y(k) \end{bmatrix} \quad (6.5)$$

6.2.3 Formation Control Design

The final element in the modified norm optimal framework is an additional component in the design of the weighting on the error signals enabling formation or shape tracking. As discussed previously, there exist applications in which the ability to maintain a specific formation (or group objective) may outweigh the individual tracking objective. For these systems, it is important to have the ability to vary the weighting on the individual versus group objective within the controller design. A weighting

matrix capable of independent objective weighting is given in (6.6).

$$\hat{\mathbf{Q}} = \Sigma_Q \cdot [\mathbf{X1}_Q \cdot \bar{\mathbf{Q}} + \mathbf{X2}_Q \cdot \mathbf{F}^T \mathbf{F}] \quad (6.6)$$

In (6.6), \mathbf{F} contains the gains defining the formation errors with respect to the individual agent tracking errors as presented earlier in Chapter 3. This results in a non-symmetric matrix (see (4.4)) in which only certain combinations of the individual tracking errors are weighted to ensure a desired formation or shape. These gains are time-invariant for MIMO systems in which the same reference trajectory is applied to each individual agent.

Similar to the matrices $\mathbf{B1}$ and $\mathbf{B2}$, $\mathbf{X1}$ and $\mathbf{X2}$ are directly related to each other through their diagonal elements $(\chi_1(1), \dots, \chi_p(N))$ and $(1-\chi_1(1), \dots, 1-\chi_p(N))$, which are used to determine the ratio of weighting on the individual objective versus the group objective. Generally $\chi_{(\cdot)}(k)$ has a value other than 0 or 1 in order to weight a combination of individual and group objectives.

The diagonal matrix Σ_Q (6.7) is used to determine the overall gain on the error signals with respect to the control and change in control signals. Similarly, Σ_S and Σ_R in (6.8) and (6.9) describe the overall gain on the control signals and change in control signals, respectively.

$$\Sigma_Q = \begin{bmatrix} \begin{bmatrix} \sigma_{Q1}(1) & & 0 \\ & \ddots & \\ 0 & & \sigma_{Qp}(1) \end{bmatrix} & & 0 \\ & \ddots & \\ & & \begin{bmatrix} \sigma_{Q1}(N) & & 0 \\ & \ddots & \\ 0 & & \sigma_{Qp}(N) \end{bmatrix} \end{bmatrix} \quad (6.7)$$

$$\hat{\mathbf{S}} = \mathbf{\Sigma}_S \cdot \bar{\mathbf{S}} \quad (6.8)$$

$$\hat{\mathbf{R}} = \mathbf{\Sigma}_R \cdot \bar{\mathbf{R}}. \quad (6.9)$$

The gains $(\sigma_{Q(\cdot)}(k), \sigma_{S(\cdot)}(k), \sigma_{R(\cdot)}(k))$ are similar to the gains (q, s, r) from the original form of the norm optimal weighting matrices in (3.13) in that they weight the different components of the cost function. Therefore, the tuning rules presented in Subsection 3.3.4 can be used to design the gains $(\sigma_{Q(\cdot)}(k), \sigma_{S(\cdot)}(k), \sigma_{R(\cdot)}(k))$, respectively.

Each of the design elements in 6.2.1 - 6.2.3 offers a means of weighting different aspects of the error signals, control signals and change in control signals. A multi-step design methodology providing details on the individual terms in the weighting matrices is provided in the next section. In order to validate the capabilities of the modified weighting matrix design, Section 6.4 presents results obtained from simulating modified norm optimal learning controllers, (6.10), with generic models of stable second order systems.

$$\begin{aligned} \mathbf{u}_{j+1} &= \hat{\mathbf{L}}_{\mathbf{u}} \mathbf{u}_j + \hat{\mathbf{L}}_{\mathbf{e}} \mathbf{e}_j \\ \hat{\mathbf{L}}_{\mathbf{u}} &= (\mathbf{P}^T \hat{\mathbf{Q}} \mathbf{P} + \hat{\mathbf{S}} + \hat{\mathbf{R}})^{-1} (\mathbf{P}^T \hat{\mathbf{Q}} \mathbf{P} + \hat{\mathbf{R}}) \\ \hat{\mathbf{L}}_{\mathbf{e}} &= (\mathbf{P}^T \hat{\mathbf{Q}} \mathbf{P} + \hat{\mathbf{S}} + \hat{\mathbf{R}})^{-1} \mathbf{P}^T \hat{\mathbf{Q}}. \end{aligned} \quad (6.10)$$

Note that the modified learning controllers are derived from an updated cost function featuring the 3-D coordinated weighting matrix designs presented in (6.6), (6.8), and (6.9).

$$\mathcal{J} = \mathbf{e}_{j+1}^T \hat{\mathbf{Q}} \mathbf{e}_{j+1} + \mathbf{u}_{j+1}^T \hat{\mathbf{S}} \mathbf{u}_{j+1} + (\mathbf{u}_{j+1} - \mathbf{u}_j)^T \hat{\mathbf{R}} (\mathbf{u}_{j+1} - \mathbf{u}_j). \quad (6.11)$$

6.2.4 Convergence and Stability Analysis

While the structure and design of the weighting matrices has been changed to allow for time-varying formation and tracking control, the necessary and sufficient condition for monotonic stability of the modified norm optimal learning controllers follows the well-known requirement identified in Chapter 3. This condition states that monotonic convergence [44] of the control input requires that $\eta = \|\hat{\mathbf{L}}_{\mathbf{u}} - \hat{\mathbf{L}}_{\mathbf{e}}\mathbf{P}\|_{i2} < 1$, where $\|\mathbf{A}\|_{i2} = \bar{\sigma}(\mathbf{A})$ and $\bar{\sigma}(\mathbf{A})$ is the largest singular value of \mathbf{A} . For the linear operators provided in (6.10), this corresponds to the following condition,

$$\|\hat{\mathbf{L}}_{\mathbf{u}} - \hat{\mathbf{L}}_{\mathbf{e}}\mathbf{P}\| = \|(\mathbf{P}^T\hat{\mathbf{Q}}\mathbf{P} + \hat{\mathbf{S}} + \hat{\mathbf{R}})^{-1}\hat{\mathbf{R}}\| < 1. \quad (6.12)$$

As a result, convergence is guaranteed for any symmetric positive semi-definite $\{\hat{\mathbf{Q}}, \hat{\mathbf{S}}, \hat{\mathbf{R}}\}$ with $\mathbf{P}^T\hat{\mathbf{Q}}\mathbf{P} + \hat{\mathbf{S}} + \hat{\mathbf{R}}$ positive definite.

6.3 Design and Implementation

Section 4.2 introduced three common approaches for describing coordination control in multiple agents. The framework introduced in the previous chapter provides a novel approach to formation control which includes decoupling the individual agent tracking objective from the group objective within the same framework. In this manner, an optimal solution for accomplishing the desired goals from each objective can be obtained within a single design. This approach enables the group to maintain a set shape or formation, while also enabling each agent to accomplish its desired task independently. Applying this decoupled approach, a three step design methodology for generating optimal learning controllers can be determined (Figure 6.1).

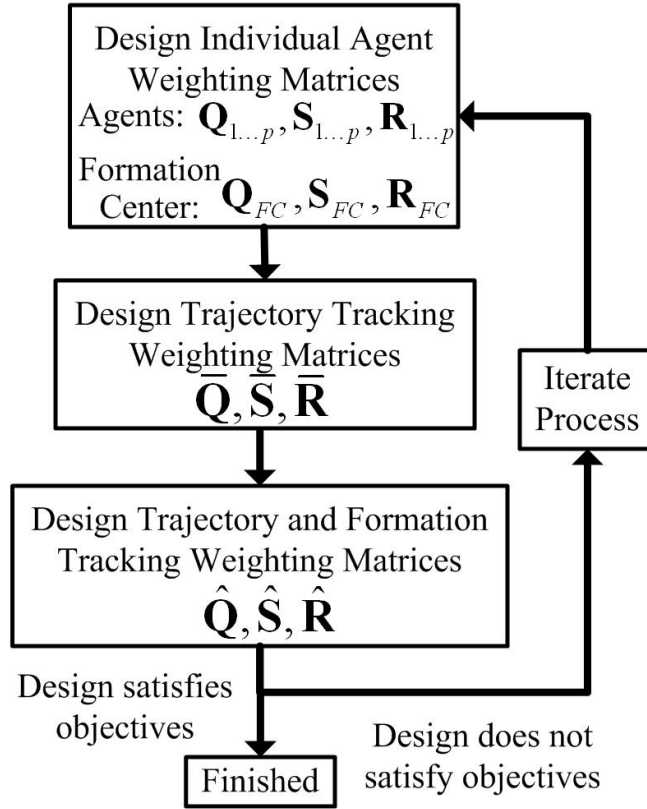


Figure 6.1: Design methodology used to determine the weighting matrices for the optimal learning controllers.

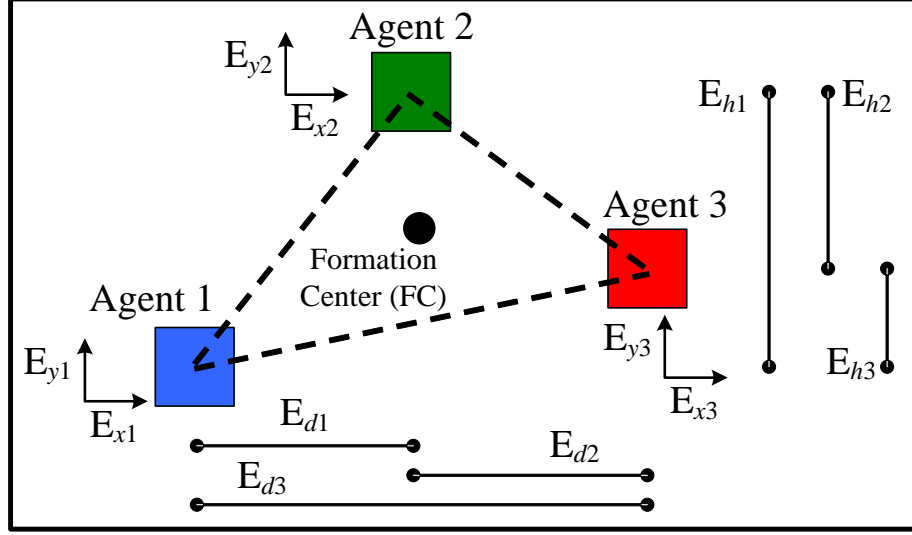
6.3.1 Design Methodology

Following the decoupled approach, the first step involves generating individual optimal weighting matrices $(\mathbf{Q}_{(\cdot)}, \mathbf{S}_{(\cdot)}, \mathbf{R}_{(\cdot)})$ of the form given in (6.1) for each multi-axis agent within the combined MIMO system. While the framework is general enough to allow separate $(\mathbf{Q}_{(\cdot)}, \mathbf{S}_{(\cdot)}, \mathbf{R}_{(\cdot)})$ designs for each multi-axis agent, many combined MIMO systems are comprised of multiple *identical* agents. In this case, a single design may be applied to each individual agent. Along with the individual agent designs, optimal $(\mathbf{Q}_{FC}, \mathbf{S}_{FC}, \mathbf{R}_{FC})$ weighting matrices for the formation center should be determined at this time. Nominal weighting matrices, valid for the formation center as well as the individual agents, will often suffice.

Having designed the weighting matrices for each agent, the individual weighting matrices are combined to form $\{\mathbf{Q}, \mathbf{S}, \mathbf{R}\}$, while the formation center weighting matrices are used to generate the coupled agent approach. The coupling gains $\kappa x(k), \kappa y(k)$ for a 2 DOF agent are calculated as the ratio of the horizontal and vertical components from the agent to the formation center position. For example, the coupling gains for a three agent system are identically given as $\kappa x(k) = \kappa y(k) = 1/3$, for all $k = 1, 2, \dots N$. Finally, diagonal matrices $\mathbf{B1}$ and $\mathbf{B2}$ need to be determined based on the desired tracking objective. The format of $\mathbf{B1}$ can be seen in (6.13), where the individual diagonal elements of $\mathbf{B1}$ and $\mathbf{B2}$ satisfy the relationship, $b1 + b2 = 1$. The following protocol can be used to switch between different trajectory tracking methods: for individual agent tracking set $(\mathbf{B1} = \mathbf{I}, \mathbf{B2} = \mathbf{0})$, for formation center tracking set $(\mathbf{B1} = \mathbf{0}, \mathbf{B2} = \mathbf{I})$, and for leader reference tracking set $b1_1(k) = 1, b1_{(2,\dots,p)}(k) = 0$ for all k in $\mathbf{B1}$ and $\mathbf{B2} = \mathbf{0}$. Note, neighbor reference trajectory tracking is a form of leader reference tracking in which there are multiple *leaders*.

$$\mathbf{B1} = \begin{bmatrix} \begin{bmatrix} b1_1(1) & & 0 \\ & \ddots & \\ 0 & & b1_p(1) \end{bmatrix} & & 0 \\ & \ddots & \\ & & \begin{bmatrix} b1_1(N) & & 0 \\ & \ddots & \\ 0 & & b1_p(N) \end{bmatrix} \end{bmatrix} \quad (6.13)$$

The final step in the design methodology focuses on formation tracking. In (6.6), $\mathbf{X1}$ and $\mathbf{X2}$ weight the individual objective (trajectory tracking) versus the group objective (formation tracking), respectively. Prior to determining these weighting matrices, one must define the desired formation or shape of the combined system. Figure 4.5, reprinted here, illustrates an example formation for a three agent MIMO system.



Example formation for a 3 agent coupled MIMO system

The three individual agents are coupled through the desired formation defined in terms of the lifted formation error signals $\mathbf{E}_{formation} = \{\mathbf{E}_{d1}, \mathbf{E}_{d2}, \mathbf{E}_{d3}, \mathbf{E}_{h1}, \mathbf{E}_{h2}, \mathbf{E}_{h3}\}$.

Redefining the formation error signals in terms of the individual axis errors for the independent agents, $\mathbf{E}_{position} = \{\mathbf{E}_{x1}, \mathbf{E}_{y2}, \mathbf{E}_{x2}, \mathbf{E}_{y2}, \mathbf{E}_{x3}, \mathbf{E}_{y3}\}$, the lifted formation matrix \mathbf{F} (original time-domain matrix, $F(k)$, presented in Subsection 4.2.4) for the 2-D planar system illustrated in 4.5 is given below.

$$\mathbf{E}_{formation} = \mathbf{F} \cdot \mathbf{E}_{position}$$

$$\text{where } \mathbf{F} = \begin{bmatrix} F(1) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & F(N) \end{bmatrix} \quad (6.14)$$

In (6.14), $F(k)$ becomes time-invariant for multi-agent systems in which the same trajectory is applied to each individual agent. A 2-DOF planar time-invariant F for the system presented in 4.5 is given in (6.15). Note that 3-DOF formation tracking would change the matrix dimensions from 6x6 to 9x9. Recall that this matrix is not unique. Identifying new definitions for the formation errors would result in a modified

F mapping matrix.

$$F = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.15)$$

Lastly, the overall gains on the error, control, and change in control signals $(\sigma_{Q(\cdot)}(k), \sigma_{S(\cdot)}(k), \sigma_{R(\cdot)}(k))$ are designed based on the tuning guidelines provided in Subsection 3.3.4. Simulation results are used to evaluate the performance of the optimal learning controllers. If the coupled system performance does not satisfy the desired objectives, the design process can be iterated. The next section presents simulation results for a three agent system.

6.4 Simulation Example

6.4.1 System Set-up

In order to validate the performance of the proposed 3-D coordinated learning control framework, simulation results from a multi servo system example were obtained. Various combinations of the coordinated weighting matrices $\{\hat{\mathbf{Q}}, \hat{\mathbf{S}}, \hat{\mathbf{R}}\}$ were applied to the following servo example.

Consider discrete-time systems of the following form,

$$G(z) = \frac{\alpha \cdot (z + \beta_1)}{(z - \beta_2)(z - \beta_3)} \quad (6.16)$$

In (6.16), G represents a single axis of a multi-axis servo-positioning system with viscous friction. Assume that the axis is stabilized with a proportional feedback

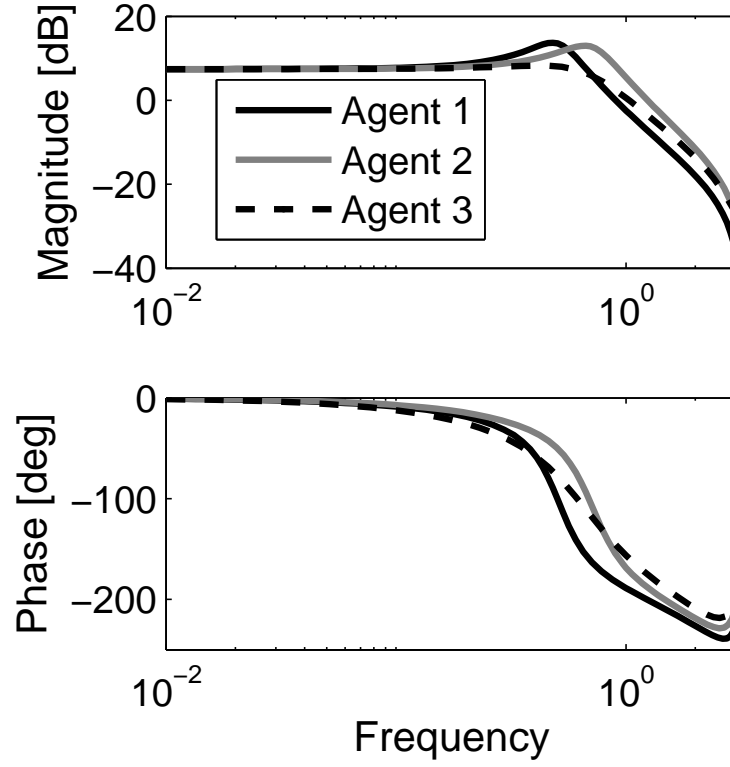


Figure 6.2: Frequency response from the x-axis for each individual agent. Y-axis responses are assumed to be similar.

controller of the form,

$$C(z) = 0.425. \quad (6.17)$$

The example in this section includes three dynamically similar multi-axis systems with slight variations that show up as model uncertainty when using a nominal design for all three systems. A bode plot of the x -axis system response is given in Figure 6.2. Note the slight differences between the individual axes. Similar results can be found for the y -axis system responses. Values for the coefficients in (6.16) are the given in Appendix I.

The output trajectory applied to each individual system is the $\{x, y\}$ rastered trajectory illustrated in Figure 6.3

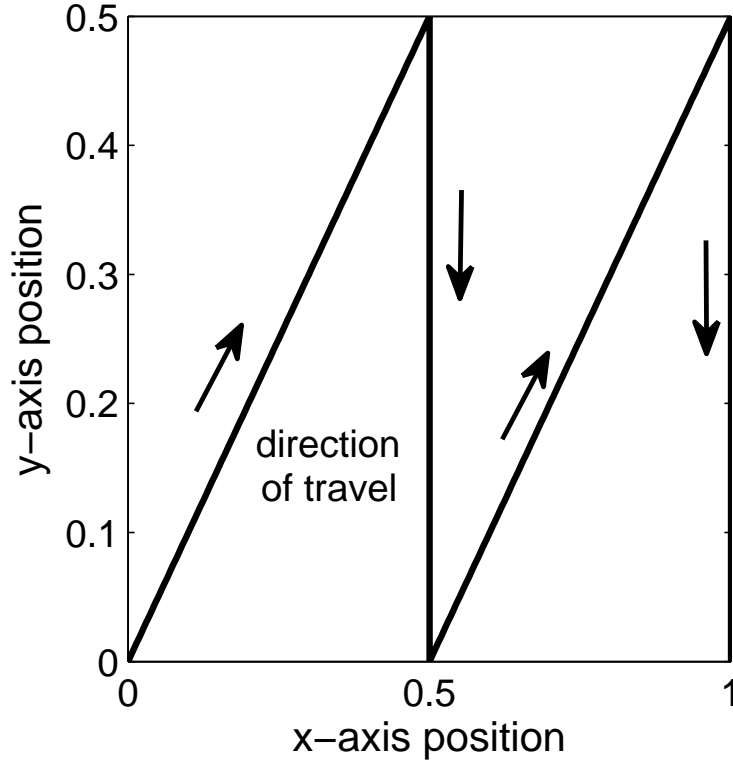


Figure 6.3: Raster trajectory applied to each 2-DOF agent.

6.4.2 Implementation and Results

Using the design methodology from Subsection 6.3.1 and the MIMO system presented in Subsection 6.4.1, norm optimal learning controllers for three different design scenarios were constructed. These scenarios were selected to evaluate the system performance while focusing on the individual trajectory tracking objective, the group formation objective, and a combination of the two using formation center trajectory tracking. The simulation set-up includes the following assumptions: dynamically similar agents enabling nominal $(\mathbf{Q}, \mathbf{S}, \mathbf{R})$ weighting matrices, multiplicative model uncertainty applied to each agent independently, identical raster reference trajectory applied to each agent, and a non-repetitive external disturbance applied only to agent 1. Additionally, although the example systems used in this section were stabilized using feedback control, for implementation with ILC the three stabilized systems were

assumed to be open-loop stable. This arrangement enables the reader to more clearly evaluate the affect of the formation tracking objective on individual trajectory tracking. Experimental results in Chapter 8 will demonstrate the implementation of this control design on a stabilized system with feedback control rather than an open-loop stable system.

Using these assumptions, optimal weighting matrices were designed. In order to simplify the design and evaluation, the same $\hat{\mathbf{S}}$ and $\hat{\mathbf{R}}$ matrices were used for all three cases. The first scenario, which focused on individual agent trajectory tracking, applied zero weighting to formation tracking. The gain selection for this case included $\mathbf{\Gamma}\mathbf{1}_{Q,S,R} = \mathbf{I}$, $\mathbf{B}\mathbf{1}_{Q,S,R} = \mathbf{I}$, and $\mathbf{X}\mathbf{1}_{Q,S,R} = \mathbf{I}$. The second scenario focused on the group formation objective by setting the individual objective gain $\mathbf{X}\mathbf{1}_Q = \mathbf{0}$, thereby negating the design of $\bar{\mathbf{Q}}$. The third case required the calculation of both the \mathbf{K} and \mathbf{F} coupling matrices. As previously described, the formation center coupling matrix for a three agent system sets $\kappa_i(k) = 1/3 \cdot \mathbf{I}(2)$ for all $k = 1, 2, \dots N$ and $i = 1, 2, 3$, while \mathbf{F} is given in (6.14). For equal weighting on the individual and group objectives set $\chi_i(k) = 1/2$ for all $k = 1, 2, \dots N$ and $i = 1, 2, 3$. The overall gains for all three scenarios were selected as $(\sigma_{Qi}(k) = 1, \sigma_{Si}(k) = .005, \sigma_{Ri}(k) = .001)$ for all $k = 1, 2, \dots N$ and $i = 1, 2, 3$, respectively.

Figure 6.4 presents the converged root mean square (RMS) error signals for the individual x-axis of agent 1 (\mathbf{E}_{x1}), the formation width error (\mathbf{E}_{d2}), and the formation height error (\mathbf{E}_{h3}). These signals, along with Figures 6.5–6.7, are used to represent the overall MIMO system performance.

Figures 6.5 and 6.6 indicate the performance trade-offs between individual trajectory tracking and formation tracking. A controller designed to optimize trajectory tracking minimizes the individual axis errors, while indirectly reducing the formation errors. Controllers designed to achieve a group objective result in the lowest formation errors, but very poor trajectory tracking. The coupling of the individ-

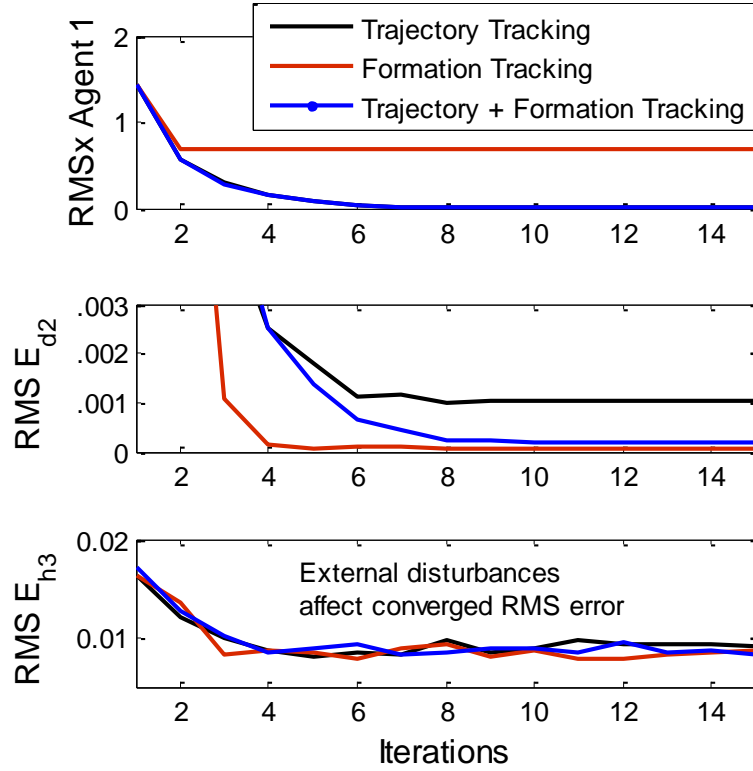


Figure 6.4: Example formation for a 3 agent coupled MIMO system

ual axis errors into the formation errors enables the formation error to tend to zero (Figure 6.6), while the individual axis errors remain large (Figure 6.5). Weighting the tracking objectives equally and applying a formation center trajectory tracking approach, the MIMO system is able to minimize trajectory and formation tracking errors simultaneously.

In addition to formation and trajectory tracking, it is important to determine the effect of external disturbances on the combined system. This effect can be seen in Figure 6.7, in which the three height formation error signals for the formation tracking case are given. The addition of a non-repeating external disturbance signal to agent 1 degrades the formation tracking performance for the combined system as shown in Figure 6.7. These results indicate that formation error signals defined with respect to the individual axis errors of agent 1 will exhibit performance limitations

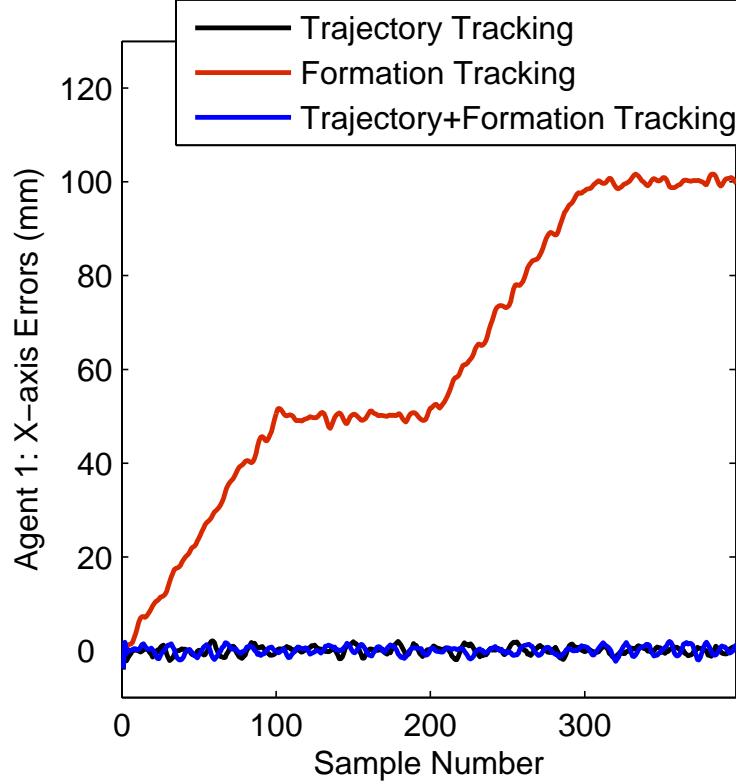


Figure 6.5: Agent 1 x-axis tracking errors for the three tracking cases.

resulting from the applied disturbance signals. The formation error signal \mathbf{E}_{h2} does not demonstrate the same limitations since it is not derived from the error signals of agent 1, thereby allowing the signal to converge to much smaller values.

These results validate the design flexibility of the norm optimal framework presented in Section 6.2. The ability to vary the design objectives, as well as the formation method, through the selection of different gains within a single framework is unique to this approach.

While simulations present an important initial validation of the control design, experimental results demonstrate the robustness of the design to stochastic disturbances and noise within the actual system. The next chapter introduces a multi-agent system consisting of three individual, yet dynamically similar, parallel kinematic systems. A system description, as well as kinematic and dynamic details of the systems

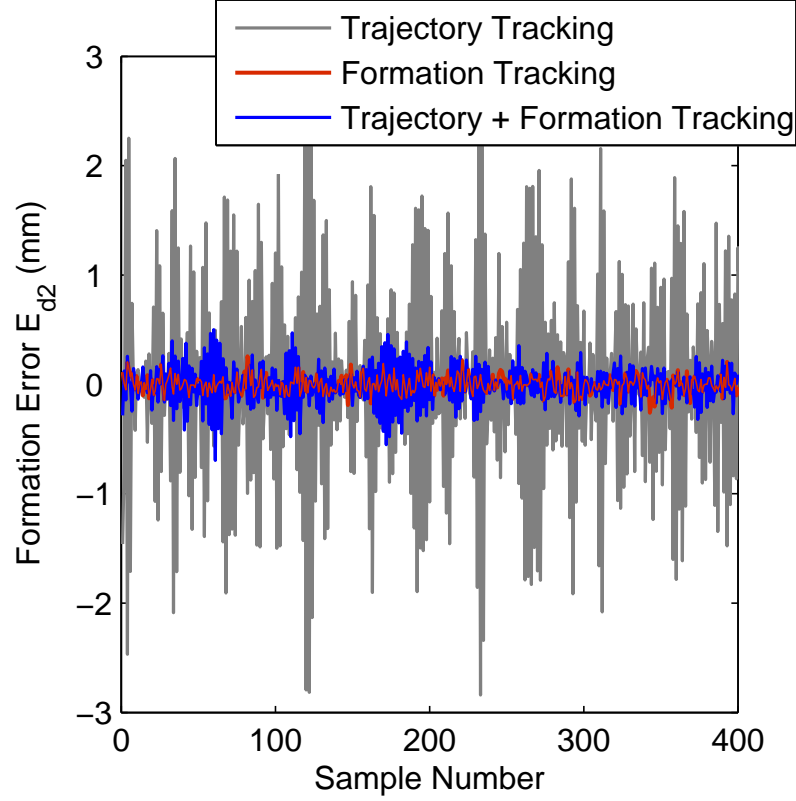


Figure 6.6: Formation error \mathbf{E}_{d2} for trajectory versus formation tracking.

will be provided. Implementation of independent case studies for two different trajectories will be used to evaluate controller design and performance for varying objectives: formation tracking, trajectory tracking, and combined formation and trajectory tracking.

Chapter 7 introduces the system and provides the kinematics and dynamics of the three parallel kinematic mechanisms (PKM). Controller design, as well as the experimental results for the independent case studies will be presented in Chapter 8.

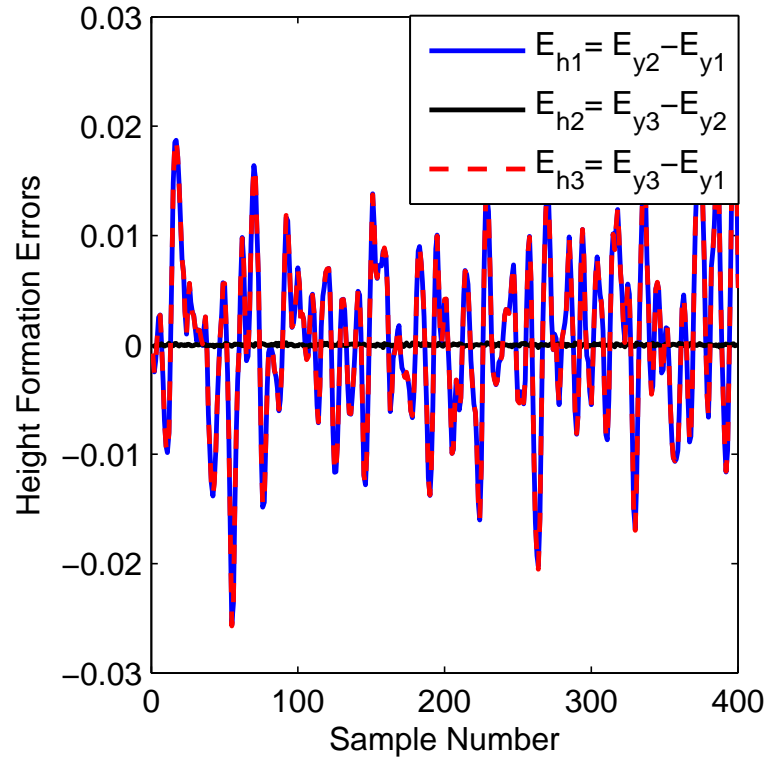


Figure 6.7: Height formation errors for the formation tracking case. \mathbf{E}_{h2} is not derived from Agent 1 and therefore converges to much smaller values.

Chapter 7

3-D Coordinated ILC: Experimental System

A classical design platform noted for exceptional range of motion and motion control is a parallel manipulator with six-degrees of freedom (6-DOF) known as a Stewart Platform. The Stewart platform was originally introduced in [91], although the name comes from the 1966 paper by Stewart [92] in which a hybrid example of the 6-linkage platform was presented. Modifications to the original 6-DOF platform have been developed including a three-degree of freedom (3-DOF) platform [93]. The 3-DOF planar manipulator is a specific type of Stewart platform in which the end effector remains planar while moving along the $\{X, Y, Z\}$ axes. Stewart platform (6-DOF or 3-DOF) parallel manipulators have applications in a diverse range of areas including: manufacturing [94], crane technology [95], underwater research [96], flight simulation [97], satellite dish positioning [98], and orthopedic surgery [99].

This chapter presents a multi-agent system which includes three independent, yet dynamically similar, 3-DOF parallel kinematic mechanisms (PKM). This multi-agent system will be used to experimentally demonstrate the novel coordinated ILC scheme presented in Chapter 6. A full description of the system including kinematics, dynamics, and system modeling are provided in this chapter. The experimental setup, implementation and controller design, and experimental results for several case studies demonstrate the versatility of the novel control methodology and are provided in Chapter 8.

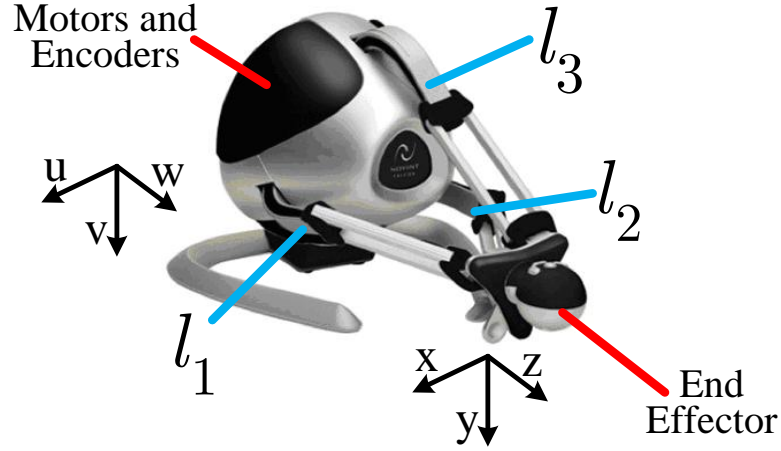


Figure 7.1: Three-DOF parallel kinematic mechanism used for experimental validation of the coordinated ILC design. The system is a Novint Falcons [3]. A schematic of one of the leg links, l_1 , can be seen in Figure 7.3.

7.1 System Description

The 3-DOF PKM used in this experimental set-up is a high-fidelity interactive three-dimensional touch system known as the Novint Falcon [3] (Figure 7.1). While initially introduced as a PC game controller, independent control of the integrated motors and encoders transitions the Falcon from a force-feedback joystick or mouse into a parallel manipulator.

Figure 7.2 presents a schematic of the three system set-up. Each independent system is identified by $\{s_1, s_2, s_3\}$, while the individual leg links are identified as $\{l_1, l_2, l_3\}$, respectively. A schematic of an individual leg link is presented in Figure 7.3. The individual leg link consists of one bottom curved linkage, which is assumed to be a set radius for analysis purposes, and an upper linkage comprised of two parallel bars. The combination of the angles $\{\theta_{1,i,m}, \theta_{2,i,m}, \theta_{3,i,m}\}$ where i is the number of leg links in a system $\{i = 1, 2, 3\}$ and m is the number of individual Falcons in the multi-agent system $\{m = 1, 2, 3\}$, can be used to calculate the position of the end effector in the XYZ world coordinate space. The key parameters from Figure 7.3, as well as

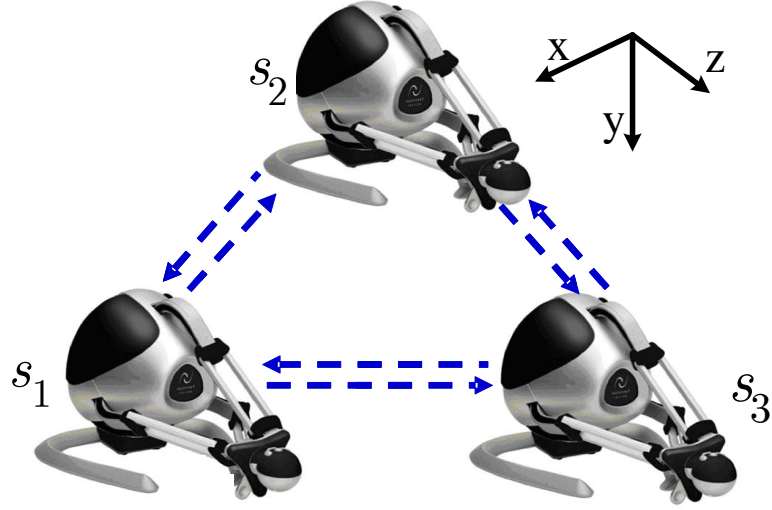


Figure 7.2: Example group formation of three of the 3-DOF parallel kinematic mechanisms used for experimental validation.

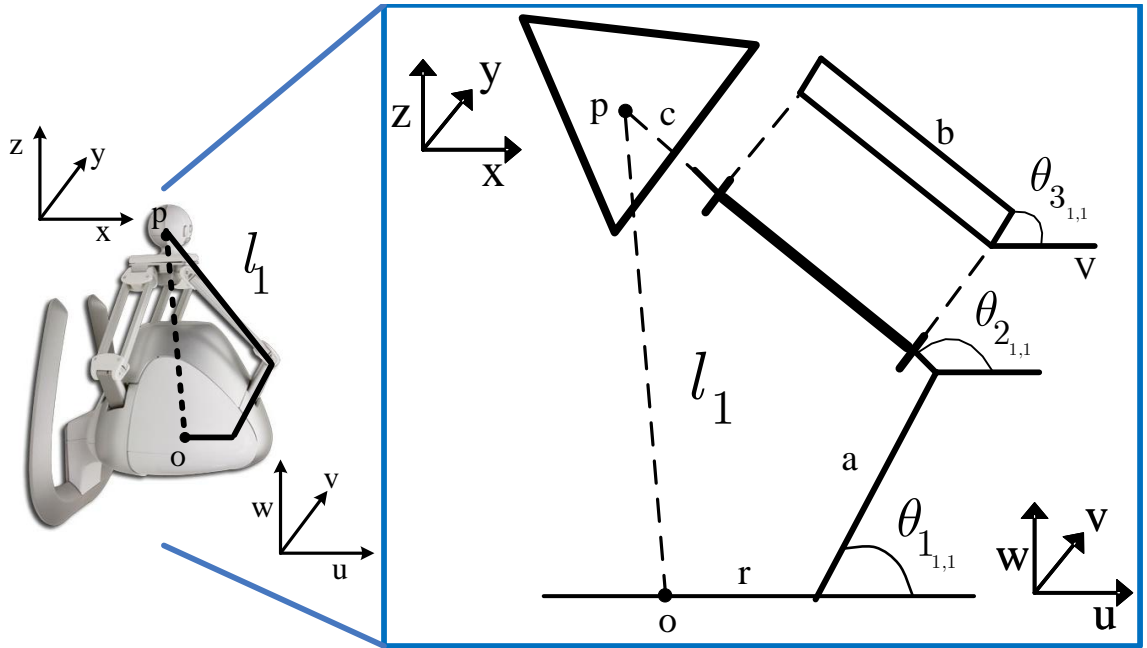


Figure 7.3: Schematic of an individual leg link for the Falcon presented in Figure 7.1. Note that the leg orientation corresponds to the falcon being placed in a vertical position with the end effector pointing upward. While the example is for the first leg, l_1 , the remaining linkages for this system, as well as the other systems in a group formation, are assumed to be identical.

the design variables used for control and implementation, are listed in Tables 7.1 and 7.2.

Table 7.1: Key Parameters for Leg Linkage Description

$i = 1, 2, 3$	number of leg linkages per individual falcon
$m = 1, 2, 3$	number of falcons in the multi-agent system
o	center position on fixed base, origin of $\{x, y, z\}$ world frame at $\{0, 0, 0\}$
r	radius from o to curved linkage axis of rotation: $\{37mm\}$
a	radius for circular path of curved linkage: $\{60mm\}$
b	length of parallel linkage bars: $\{125mm\}$
c	distance from parallel linkage to p : $\{20mm\}$

Table 7.2: Design Variables for Leg Linkage Control and Implementation

$\theta_{1i,m}$	angle between $\{u, w\}$ axes for curved linkage: encoder readings
$\theta_{2i,m}$	angle between $\{u, w\}$ axes for parallel linkage: calculated in MATLAB
$\theta_{3i,m}$	angle between $\{v, w\}$ axes for parallel linkage: calculated in MATLAB
$p_{xi,m}$	position of end effector with respect to x-axis: calculated in MATLAB
$p_{yi,m}$	position of end effector with respect to y-axis: calculated in MATLAB
$p_{zi,m}$	position of end effector with respect to z-axis: calculated in MATLAB
p	center position on end effector: with respect to $\{p_x, p_y, p_z\}$

7.2 Kinematics

The focus of the multi-agent system consists of two objectives: 1) maintaining a specified shape or formation coordinated amongst the individual Falcons, and 2) enabling independent trajectory tracking for each system. The desired formation is originally defined by the initial orientation and placement of the individual systems. This formation should be maintained with respect to the location of the end effector (see Figure 7.2) throughout the given period.

In order to track the position of the end effector, one must determine the relationship between the angles of the bottom leg links and the position of the end effector. Using the angles defined in Figure 7.3, as well as the parameters and variables listed in Tables 7.1 and 7.2, forward and inverse kinematics of the individual systems presented in Figure 7.2 can be determined.

7.2.1 Forward Kinematics

Various methods for deriving forward kinematics for parallel manipulators have been studied in the literature [100–103]. This subsection will provide a brief overview of the method presented in [104] which has been adopted for use in this research.

Forward kinematics define a mapping technique from the bottom leg link angles (illustrated for one leg link in Figure 7.3), $\{\theta_{1,1}, \theta_{12,1}, \theta_{13,1}\}$, to the position of the end effector in the XYZ world coordinate frame. The list of known parameters includes all of the key elements listed in Table 7.1 plus the planar angles, $\{\phi_{1,m} = 0^\circ, \phi_{2,m} = 120^\circ, \phi_{3,m} = 240^\circ\}$ for $m = 1, 2, 3$, that transform the base of the legs from the UVW frame to the XYZ world frame. Note that only the bottom angles, $\{\theta_{1,1}, \theta_{12,1}, \theta_{13,1}\}$, are assumed to be known. This is true for all m agents in the multi-agent system. The upper angles, $\{\theta_{2,i,m}, \theta_{3,i,m}\}$ for all i and m are assumed to be unknown. The desired variables can be listed as $\{p_{x,i,m}, p_{y,i,m}, p_{z,i,m}\}$ for a given link on each individual system, respectively.

Using the diagram presented in Figure 7.3, expressions for the end effector position, $\{p_{u,i,m}, p_{v,i,m}, p_{w,i,m}\}$, in the UVW coordinate frame can be determined for the i^{th} link in the m^{th} system.

$$p_{u,i,m} = a \cos(\theta_{1,i,m}) - c + (b \sin(\theta_{3,i,m})) \cos(\theta_{2,i,m}), \quad (7.1)$$

$$p_{v,i,m} = b \cos(\theta_{3,i,m}), \quad (7.2)$$

$$p_{w,i,m} = a \sin(\theta_{1,i,m}) + (b \sin(\theta_{3,i,m})) \sin(\theta_{2,i,m}) \quad (7.3)$$

The relationship between the end effector position in the UVW coordinate frame and the XYZ coordinate frame is determined using simple transformation matrices with

respect to the planar angles, $\{\phi_{1,m} = 0^\circ, \phi_{2,m} = 120^\circ, \phi_{3,m} = 240^\circ\}$, respectively.

$$\begin{bmatrix} p_{ui,m} \\ p_{vi,m} \\ p_{wi,m} \end{bmatrix} = \begin{bmatrix} \cos(\phi_{i,m}) & \sin(\phi_{i,m}) & 0 \\ -\sin(\phi_{i,m}) & \cos(\phi_{i,m}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{xi,m} \\ p_{yi,m} \\ p_{zi,m} \end{bmatrix} + \begin{bmatrix} -r \\ 0 \\ 0 \end{bmatrix}, \{i = 1, 2, 3\} \quad (7.4)$$

Substituting (7.1)-(7.3) into the equations formed from the transformation matrix in (7.4), rearranging, and simplifying these equations leads to a system of three equations,

$$\begin{aligned} p_{xi,m}^2 + p_{yi,m}^2 + p_{zi,m}^2 + 2[c - r - a \cos(\theta_{1i,m})][p_{xi,m} \cos(\phi_{i,m}) + p_{yi,m} \sin(\phi_{i,m})] \\ - 2a \sin(\theta_{1i,m})p_{zi,m} + a^2 + (r - c)^2 - 2a(c - r) \cos(\theta_{1i,m}) - b^2 = 0, \end{aligned} \quad (7.5)$$

for $i = 1, 2, 3$, with three unknowns, $\{p_{xi,m}, p_{yi,m}, p_{zi,m}\}$. Geometrically, each of these equations defines a sphere. The intersection of these spheres is the location of the end effector and the solution to the forward kinematics problem. The solution to this set of equations can be found in [104].

In addition to solving the problem analytically, a computer program which finds the point of intersection for the three spheres defined by (7.5) was originally written in Maple and has been translated to MATLAB. The program, denoted **interx**, requires input from the user in the form of the radius of each sphere (i.e. 'b' in Figure 7.3) and the location of the center of each sphere denoted by the following vector,

$$\begin{bmatrix} (r + a \cos(\theta_{1i,m})) \cos(\phi_{i,m}) & (r + a \cos(\theta_{1i,m})) \sin(\phi_{i,m}) & a \sin(\theta_{1i,m}) \end{bmatrix}. \quad (7.6)$$

Using this input, the function calculates the intersection of the three spheres, and thus the solution to the forward kinematics problem. The MATLAB code for this function can be found in Appendix C.

7.2.2 Inverse Kinematics

As the complement to the forward kinematics problem, the inverse kinematics solution identifies the mapping from the position of the end effector to a set of joint angles, $\{\theta_{1i,m}, \theta_{2i,m}, \theta_{3i,m}\}$ for the i^{th} leg link on the m^{th} system, that make the position possible. The key angle from this set is $\theta_{1i,m}$ which defines the angle of the bottom leg link with respect to the horizontal axis. This angle shows up in the control of the system with respect to the motor dynamics, as shown in Section 7.3.

Using the mapping identified in (7.4), the position of the end effector in the UVW coordinate frame can be determined with respect to each leg link. Substituting these equations into (7.1)-(7.3) results in a system of nine equations and nine unknowns. Two solutions for $\theta_{3i,m}$ can initially be found from (7.2).

$$\theta_{3i,m} = \pm \cos^{-1}\left(\frac{p_{vi,m}}{b}\right) \quad (7.7)$$

Defining the half-tangent angel of $\theta_{1i,m}$ as $t_{1i,m} = \tan\left(\frac{\theta_{1i,m}}{b}\right)$, a quadratic relationship for $t_{1i,m}$ can be solved to determine $\theta_{1i,m}$.

$$l_{2i}t_{1i,m}^2 + l_{1i}t_{1i,m} + l_{0i} = 0 \quad (7.8)$$

where

$$l_{0i} = p_{wi,m}^2 + p_{ui,m}^2 + 2cp_{ui,m} - 2ap_{ui,m} + a^2 + c^2 - b^2 \sin(\theta_{3i,m})^2 - 2ac \quad (7.9)$$

$$l_{1i} = -4ap_{wi,m} \quad (7.10)$$

$$l_{2i} = p_{wi,m}^2 + p_{ui,m}^2 + 2cp_{ui,m} + 2ap_{ui,m} + a^2 + c^2 - b^2 \sin(\theta_{3i,m})^2 + 2ac \quad (7.11)$$

Having identified $\theta_{1i,m}$ and $\theta_{3i,m}$, $\theta_{2i,m}$ can be calculated by substituting these values into (7.1) and (7.3) and solving for $\theta_{2i,m}$. The solution to these equations results in

two possible orientations for the leg links. The appropriate posture is determined by selecting the orientation with the most reasonable choice of angles based on visual inspection of the given system or the orientation of the angles in the previous time step.

7.2.3 Jacobian Matrix

In addition to the analytical solutions, a simplified method for mapping back and forth between the angles and the end effector position requires the use of a Jacobian transformation matrix. The Jacobian matrix is derived by mapping small changes in the actuated joint angles, $\theta_{1,i,m}$ for $i = 1, 2, 3$, to the end effector positions using the MATLAB function **interx**. The variation from one end effector position to the next over a small angle change is used to calculate the mapping from the angles to the XYZ position. The MATLAB code for deriving the Jacobian can be found in Appendix F. There are a few assumptions that are made when using this approach.

- A1) The reference operation is bounded within a specified work area.
- A2) The mapping is time- and angle-invariant.
- A3) The system, and therefore the mapping, is linear within the work area.

The key assumption is A1, in which the work area is assumed to be small enough such that A2 and A3 will hold. If the work area increases, assumptions A2 and A3 begin to break down, leading to large variations between the Jacobian transformation and the mapping based on the kinematics. One approach for addressing this limitation is to design a Jacobian matrix that is a function of the lower leg link angles, $J(\theta_{1,i,m})$. One of the main advantages of using an angle-varying Jacobian is improved matching between the Jacobian transformation and the kinematics over a larger work area. One major disadvantage of this approach is that an angle-varying Jacobian may

require a controller that is a function of leg link angle. This is particularly true of model-based control designs in which the Jacobian is integrated into the design as part of the system model. A static Jacobian enables the use of a single controller, while an angle-varying Jacobian may lead to multiple control designs as the model changes.

An additional approach to compensate for potential variations is to use a learning controller with the static Jacobian. The use of a *learning* function minimizes the transformation errors through iterative updates to the input signal that compensate for repetitive modeling errors.

7.3 Dynamics

There are a few different strategies for determining the dynamics of the 3-DOF parallel kinematic mechanism. The more complicated, yet dynamically accurate, approach is to use an Euler-Lagrange method in which the nonlinear and coupled aspects of the system are incorporated into a single model design. A simplified method assumes an uncoupled system in which a linearized model for each independent leg link can be determined. While this approach introduces model uncertainty into the formulation, when used in conjunction with learning control, it can often result in a simpler and faster control design. Lastly, the system dynamics can be identified experimentally over a specified range of motion. These approaches are detailed in the following subsections.

7.3.1 Euler-Lagrange

Traditionally, the dynamics of a 3-DOF PKM system include a nonlinear MIMO relationship that accounts for all of the inherent coupling within the system. In this

approach, the Euler-Lagrange equations can be written as [105],

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q), \quad (7.12)$$

where $M(q)$ is a symmetric, positive-definite matrix defined as the inertia matrix, $C(q, \dot{q})$ contains the centrifugal and Coriolis terms of the system, and $G(q)$ is the gravity vector. The argument q contains the three actuator angles, $\{\theta_{11,m}, \theta_{12,m}, \theta_{13,m}\}$ for each independent agent $\{m = 1, 2, \dots\}$ in the combined system, while τ contains the torque inputs for the motors associated with each leg link.

Although this approach provides an analytically correct dynamic relationship, the nonlinearities may lead to a more complicated control scheme. One of the advantages of using ILC is the ability to *learn* out unmodeled dynamics and nonlinearities that repeat from iteration to iteration. This functionality enables the control designer to know very little *a priori* information regarding the plant dynamics. To facilitate a more direct control design process, a simplified uncoupled linearized model is presented in the next subsection.

7.3.2 Single Leg Dynamic Model

The single leg dynamic model is a simplified version of the Euler-Lagrange dynamics modified for use with uncoupled, linearized SISO systems rather than coupled, nonlinear MIMO systems [104]. Figure 7.4 shows the simple single leg link model used in this section.

There are two key assumptions that must be made when using this approach: 1) the combined weight of the end effector and the parallel linkage bars is concentrated at the tip of the single link, and 2) the inherent coupling in the system can be ignored in the single link model. Despite the linearization and simplification to a single link model, the underline structure of the dynamics remains similar to the model presented

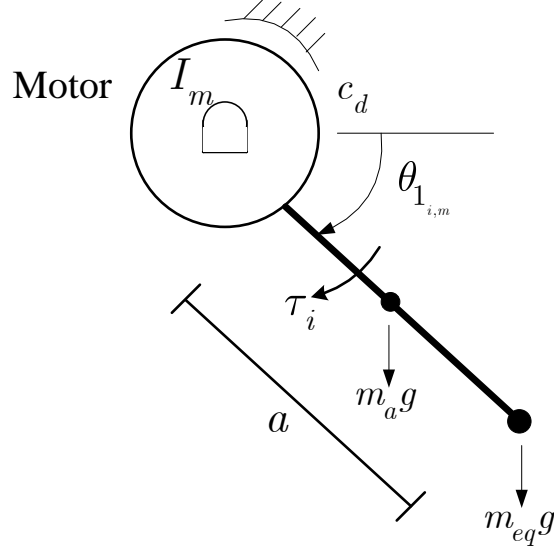


Figure 7.4: Single link dynamic model. This image is used to determine a simple, linearized model for each individual leg link.

in the previous subsection.

$$\tau_{i,m} = \alpha \ddot{\theta}_{1,i,m} + c_d \dot{\theta}_{1,i,m} - \beta \cos \theta_{1,i,m}. \quad (7.13)$$

In (7.13), α contains the inertial elements, c_d is the viscous damping of the motor, and $\beta \cos \theta_{1,i,m}$ is the gravitational term for the specific leg link $\{i = 1, 2, 3\}$ and system $\{m = 1, 2, 3\}$, respectively. Based on the single leg link illustrated in Figure 7.4, α and β from (7.13) are defined as,

$$\alpha = I_m + \frac{1}{3}m_a a^2 + (2m_b + \frac{1}{3}m_c)a^2 \quad (7.14)$$

$$\beta = a \cdot g \cdot (\frac{1}{2}m_a + 2m_b + \frac{1}{3}m_c), \quad (7.15)$$

where the variables in (7.13)-(7.15) are provided in Table 7.3. Derivations for α and β are provided in Appendix D.

The nonlinear single leg dynamics are then linearized about a desired operating point, defined as the initial angle $\theta_{1,i,m,0}$. Substituting this value back into (7.13)

Table 7.3: Variables for Single Leg Dynamics

I_m	actuator rotor inertia
m_a	mass of the single link, concentrated midway along link
m_b	combined mass of the parallel linkage bars, concentrated at end of single link
m_c	mass of the moving platform, concentrated at end of single link
m_{eq}	combined mass at end of single link: $m_{eq} = 2m_b + \frac{1}{3}m_c$
a	length of pendulum in single link model
g	gravity [9.8 m/s ²]
c_d	viscous damping of the motor
τ	applied torque to the motor
$\theta_{1i,m}$	actuator angle with respect to the single link

and solving for $\tau_{i,m,0}$ results in the following linearized relationship for the system dynamics,

$$\hat{\tau}_{i,m} = \alpha \ddot{\theta}_{1i,m} + c_d \dot{\theta}_{1i,m} + \tau_{i,m,0} \theta_{1i,m} \quad (7.16)$$

$$\text{where } \tau_{i,m,0} = -\beta \cos \theta_{1i,m,0}. \quad (7.17)$$

The equation presented in (7.16) defines the linearized dynamics between the link angle and the motor torque for an individual leg link. Using this straightforward approach, a simple PID controller can be designed to stabilize each leg link. As mentioned previously, this approach will introduce unmodeled dynamics into the derivation. An additional approach which seeks to combine the simplicity of the linearized leg link approach with greater accuracy more commonly associated with the full Euler-Lagrange approach is presented in the following subsection.

7.3.3 Experimental System Modeling

A common approach for identifying a model of a system, particularly when the system includes nonlinear and higher-order dynamics, is to experimentally identify a model. While this can be achieved using different methods, one of the most commonly used

approaches is to use a swept-sine measurement technique. Although this approach will result in a linear model, transients within the system can be identified and included in the model. This technique, first introduced by Reed, Hall, and Barker in 1960 [106], uses an excitation signal of a sinusoidal shape whose frequency increases with time. The response to this signal is recorded and then analyzed to determine the system amplitude and phase for each frequency. These values are evaluated using the well known sinusoidal response formula,

$$\begin{aligned} X(t) &= A \sin(\omega(t)) \\ Y(t) &= G(t) \cdot X(t) = |G|A \sin(\omega(t) + \angle G), \end{aligned} \quad (7.18)$$

where $X(t)$ is the input signal, $Y(t)$ is the output signal, A is the amplitude of the input sine wave, ω is the forcing frequency of the sine wave at a given point in time, $|G|$ is the amplitude of the system, and $\angle G$ is the phase of the system. As can be seen from (7.18), the output signal should be a sine wave with the same forcing frequency as the input, but with a different amplitude and phase shift due to the system dynamics.

Applying this technique, a swept sine analysis can be applied to each independent leg link. To determine an experimental model for each independent leg link, the following protocol was applied.

- s1) Generate several sinusoidal input signals of varying frequencies at the desired sampling rate and nominal trajectory length of time.
- s2) Send generated input signals to $\theta_{1,1}$ while holding $\theta_{1,i,m}$ stationary with an input signal of zeros for the remaining leg links on system 1 and all three leg links on systems 2 and 3.
- s3) Save output response of the designated leg link.

- s4) Repeat s2)-s3) for each sinusoidal input of varying frequency.
- s5) Analyze output data using MATLAB to generate a bode plot with the calculated magnitude and phase values at the given frequencies.
- s6) Determine a model that closely fits the experimental data.
- s7) Repeat steps s2)-s6) for $\theta_{12,1}$ and $\theta_{13,1}$.
- s8) Repeat steps s2)-s7) for systems 2 and 3.

Implementing this protocol, a model for the individual leg links was determined. Figure 7.5 shows a bode plot of the experimental data versus the estimated system model for the dynamic relationship between $\theta_{1i,m}$ and the torque input on motor i . The system model was determined to be a second order system of the form provided in (7.19).

$$G(s) = \frac{K \cdot \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \quad (7.19)$$

In (7.19), $\{K = 6\}$ is the system gain, $\{\omega_n = 9\}$ is the natural frequency or spring constant, and $\{\zeta = 0.9\}$ is the damping coefficient. The same model was identified for each leg link on all three systems.

Having identified system models for each leg link, the learning control scheme presented in Chapter 6 can now be implemented on an experimental set-up using the multi-agent 3-DOF PKM system described in this Chapter. The experimental set-up, controller design and implementation, and experimental results are provided in the following chapter.

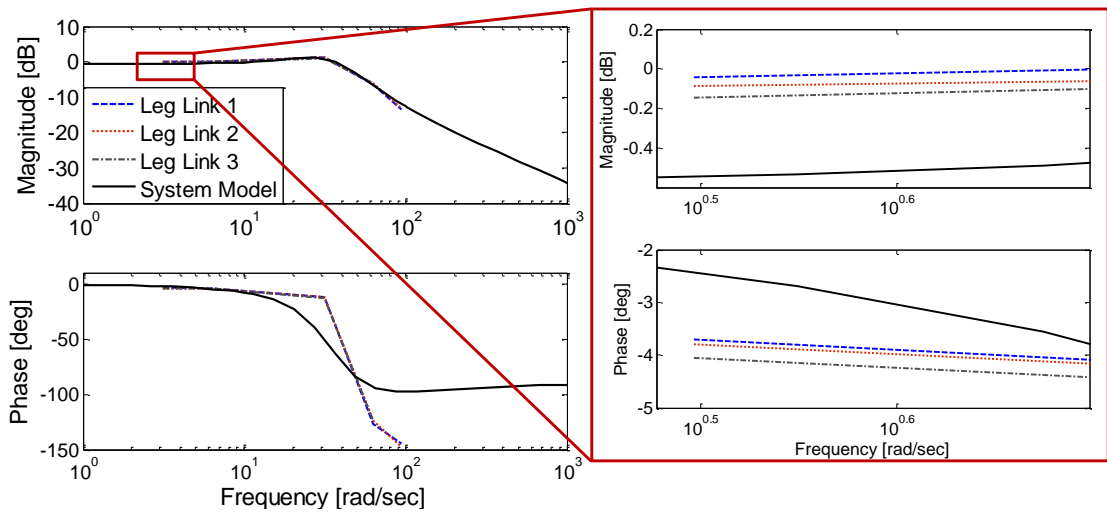


Figure 7.5: Bode diagram of the experimental data and the identified system model for the dynamic relationships between $\theta_{1,i,1}$ and the input torque to motors 1-3. Bode plots for the leg links in systems 2 and 3 show similar experimental results and can be seen in Appendix E. The dynamic similarities within the systems enables identical system models to be used for each leg link.

Chapter 8

3-D Coordinated ILC: Experimental Validation

In order to validate the performance of the 3-D coordinated ILC framework presented in Chapter 6, coordinated learning controllers of the form (6.10) were designed and implemented on an experimental testbed. This chapter presents the experimental set-up, controller design and implementation, and experimental results for various trajectories and design objectives.

8.1 Experimental Set-up

8.1.1 System Set-up

The experimental testbed used in this chapter consists of three dynamically similar 3-DOF PKM systems. Figure 8.1 shows an image of the experimental testbed. Note that while the outward appearance of the three systems may indicate slight differences (such as the appearance of the end effector), the dynamics and therefore the tracking performances of the three systems are very similar.

As can be seen from Figure 8.1, the 3-DOF PKM system, described dynamically in Chapter 7, contains three independent leg links connected to the base on one end and to a single end effector on the other. The position of the end effector is determined by the angular movements of the three leg links. Forward and inverse kinematic descriptions to map from the link angles to the end effector position and vice versa are given in Chapter 7 Section 7.2. As a means of simplifying the inverse kinematics

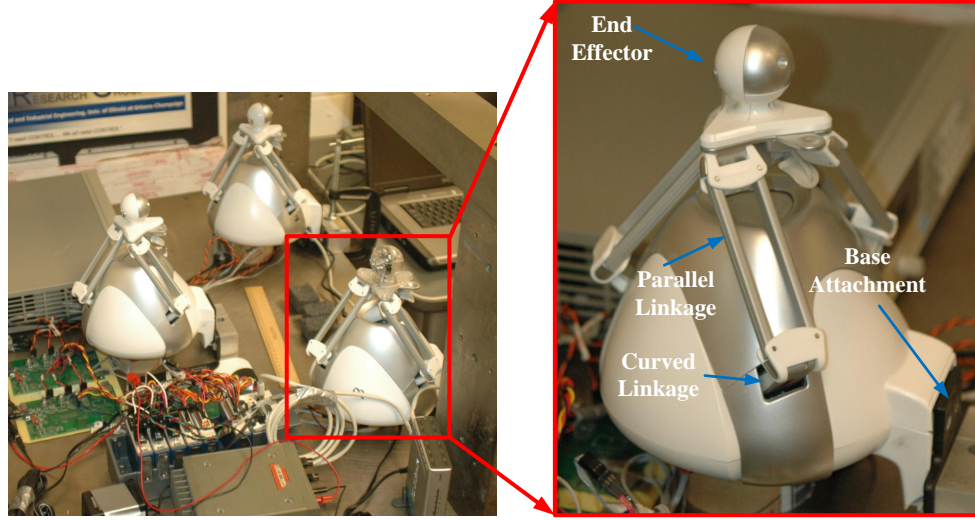


Figure 8.1: Experimental testbed with three 3-DOF PKM systems. The PKM systems are Novint Falcons. Note the orientation of the falcons. While arbitrary, they have been arranged to create a specific formation. This formation should be maintained during formation tracking with respect to the location of the end effectors.

and for use in the coordinated learning controller design, detailed in Section 8.2, a Jacobian mapping matrix was determined. The Jacobian maps the leg link angles $\theta_{1i,m}$ to the end effector position in the XYZ coordinate frame.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = J \cdot \begin{bmatrix} \theta_{11,m} \\ \theta_{12,m} \\ \theta_{13,m} \end{bmatrix}. \quad (8.1)$$

Applying the assumptions stated in Subsection 7.2.3, the calculated Jacobian matrix is presented in (F). Note that the Jacobian must take into account the parameters of the desired trajectories since the matrix should be suitable for the desired work area and angle changes required by the reference trajectories.

$$J = \begin{bmatrix} -44.71 & 22.36 & 22.36 \\ 0 & -38.72 & 38.72 \\ 14.56 & 14.56 & 14.56 \end{bmatrix} \quad (8.2)$$

Table 8.1: PID Gains for the Stabilizing Feedback Controller

k_p	2.300
k_d	0.040
k_i	0.300
t_s	0.005

It is important to note that the Jacobian should to be nonsingular. Singularities occur at the edge of the task space, leading to a singular Jacobian matrix. The Jacobian presented in (F) was determined for a command space oriented in the middle of the task space. For the reference trajectories used in these experimental results, the movement of the end effector was restricted to the central area of the workspace.

Stabilizing Feedback Controller

One of the critical assumptions when implementing ILC is that the plant is stable or stabilizable. Designing an appropriate controller to stabilize the plant is particularly critical for model based learning control techniques such as the norm optimal approach used in this work. Using the experimentally determined plant model from Subsection 7.3.3, a stabilizing PID feedback controller is designed to ensure stability and nominal performance in the time domain. Recall that a single model has been identified for use with all leg links, and therefore identical PID feedback controllers can be used for all systems. The discrete PID controller is of the form,

$$C_{fb} = k_p + k_i \frac{t_s}{z - 1} + k_d \frac{z^{-1} - z^{-2}}{t_s}, \quad (8.3)$$

where k_p is the proportional gain, k_d is the derivative gain, k_i is the integral gain, and t_s is the discrete sampling rate for the system. Values for these gains (shown in Table 8.1) were determined through heuristic tuning on the actual system.

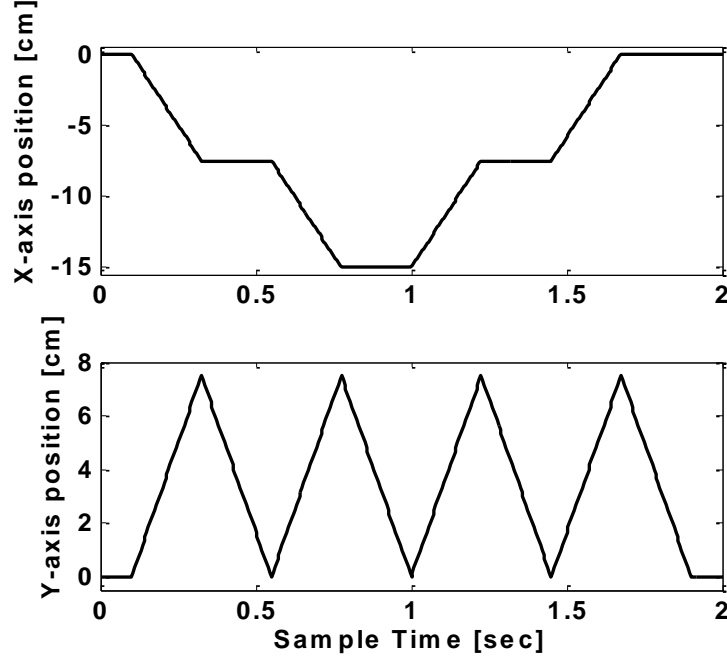


Figure 8.2: Raster reference trajectory for experimental testing. Note that the original design is a planar raster in the XY plane.

8.1.2 Trajectory Design

The performance results presented in this chapter were obtained using the following two reference trajectories. These reference trajectories were selected to demonstrate the performance tracking capabilities of the proposed learning controller design. Identical reference trajectories are applied to each agent in the combined MIMO system.

Raster Trajectory

The first trajectory is a planar raster trajectory in the XY plane with two rasters forward and then two rasters in the reverse direction to return to the initial starting position. The time period of the trajectory is 2 seconds. Figure 8.2 shows the x - and y -axis reference trajectories with respect to time.

Similarly, Figure 8.3 presents the three angle reference inputs corresponding to the planar XY raster. The angle inputs were determined using the inverse of the Jacobian matrix presented in (F). Recall that the Jacobian matrix maps the angle

inputs into cartesian values (8.1). Therefore, the inverse Jacobian maps the cartesian values back into angle inputs. Small mapping errors introduced through the use of the Jacobian matrix were corrected by mapping the angle reference trajectories back into the XYZ cartesian coordinate frame using the MATLAB function **interx**. As stated in Chapter 7, the **interx** function finds the point of intersection for the three spheres. This function requires input from the user in the form of the radius of each sphere (i.e. 'b' in Figure 7.3) and the location of the center of each sphere denoted by the following vector,

$$\begin{bmatrix} (r + a \cos(\theta_{1,i,m})) \cos(\phi_{i,m}) & (r + a \cos(\theta_{1,i,m})) \sin(\phi_{i,m}) & a \sin(\theta_{1,i,m}) \end{bmatrix}. \quad (8.4)$$

Using this input, the function calculates the intersection of the three spheres, and thus the solution to the forward kinematics problem. The MATLAB code for this function can be found in Appendix C.

The modified cartesian reference inputs are provided in Figure 8.4. Note that the raster is no longer planar, but contains some z-axis movement. For a more accurate comparison, the revised reference trajectories illustrated in Figure 8.4 are used to calculate the end effector position tracking errors for use in the ILC update law.

Spiral Trajectory

Similarly to the raster trajectory, the second reference trajectory is originally designed to be a planar trajectory in the XY cartesian plane. The trajectory consists of a spiral pattern starting in the center and spiralling counter-clockwise away from the initial starting point. The spiral designed x - and y -axis positions can be seen in Figure 8.5.

Following the same process as with the raster trajectory, the corresponding angular reference inputs for the spiral trajectory can be derived by using the inverse Jacobian

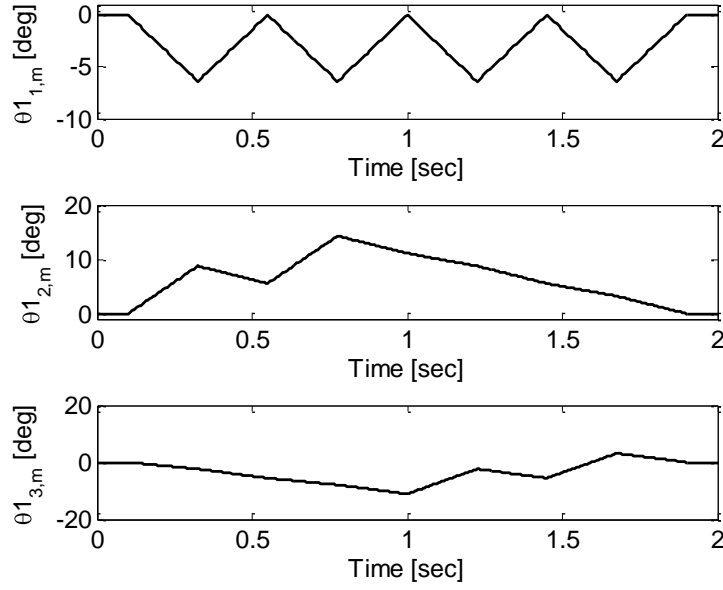


Figure 8.3: Raster reference trajectory for experimental testing mapped to the corresponding angle inputs for motors 1-3. The XY cartesian trajectory was mapped to $\theta_{1,m} - \theta_{3,m}$ using the Jacobian matrix in (F).

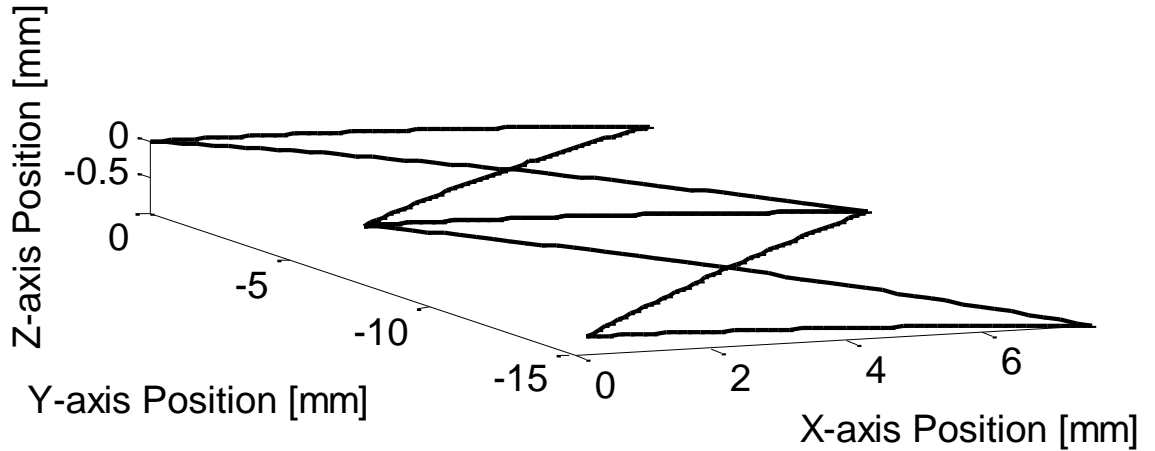


Figure 8.4: 3-D image of the revised raster reference trajectory for experimental testing. The cartesian inputs were derived by mapping the angle input to cartesian inputs using the MATLAB command **interx**. Small changes due to the Jacobian mapping matrix resulted in a nonplanar raster trajectory. Note that the z-axis movements are significantly smaller than the x- or y-axis movements.

matrix. These inputs are shown in Figure 8.6. Using the MATLAB command **interx**, the angle inputs are then mapped back into the XYZ cartesian coordinate frame to

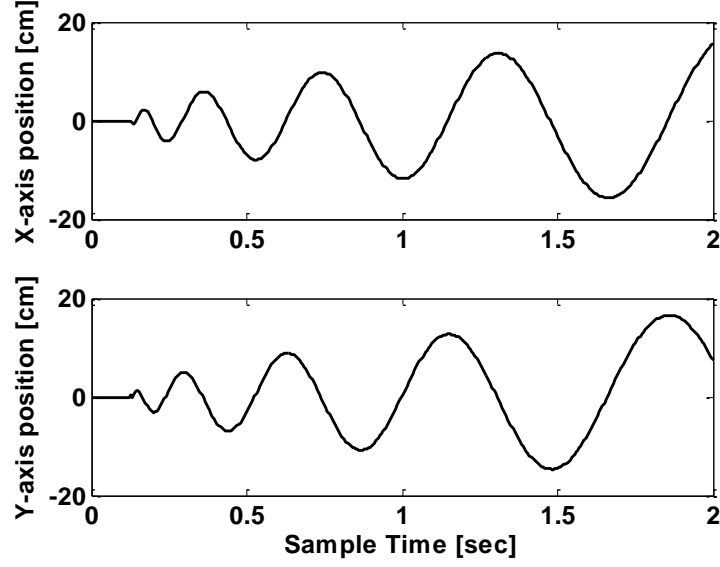


Figure 8.5: Spiral reference trajectory for experimental testing. Note that the original design is a planar spiral in the XY plane.

evaluate any discrepancies between the original design and the actual design based on the input values to the motors. Note that mapping errors from using the Jacobian matrix result in a nonplanar spiral as illustrated in Figure 8.7. The modified XYZ inputs from Figure 8.7 are used in the calculation of the end effector position errors.

8.1.3 Case Studies

Having designed the reference trajectories and stabilized the system, the final step in the experimental set-up is to determine the case studies of interest. In an effort to demonstrate the versatility of the design framework and provide appropriate test examples for comparison purposes, three case studies to be implemented with each of the reference trajectories have been selected.

The first study considers a single design objective, trajectory tracking of the individual agents. In this study, all of the emphasis is placed on trajectory tracking, thereby decoupling the performance of the individual agents from each other. The second study is the exact opposite of the first study. In the second study, all of the

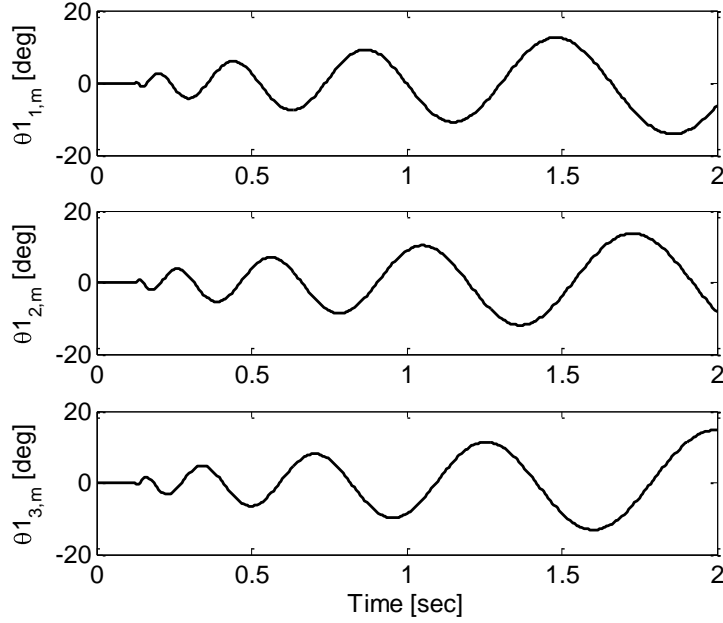


Figure 8.6: Spiral reference trajectory for experimental testing mapped to the corresponding angle inputs for motors 1-3. The XY cartesian trajectory was mapped to $\theta_{11,m} - \theta_{13,m}$ using the Jacobian matrix in (F).

focus is directed towards minimizing the formation tracking errors. This approach requires coordinated movements between the individual agents and will be shown to adversely affect the trajectory tracking performance of the individual systems. Finally, the third case study looks at a combination of the first two studies by weighting the trajectory and formation tracking objectives. The goal of this study is to ensure trajectory and formation tracking improvements simultaneously.

The next section presents the unique learning controller designs for each case study. Note that the same framework will be used for each design, with the differences stemming from the selection of the various gains within the weighting matrices.

8.2 Coordinated Learning Controller

The coordinate learning controller design follows the basic framework presented in Chapter 6 with some slight modifications due to the use of PKM systems. The goal of

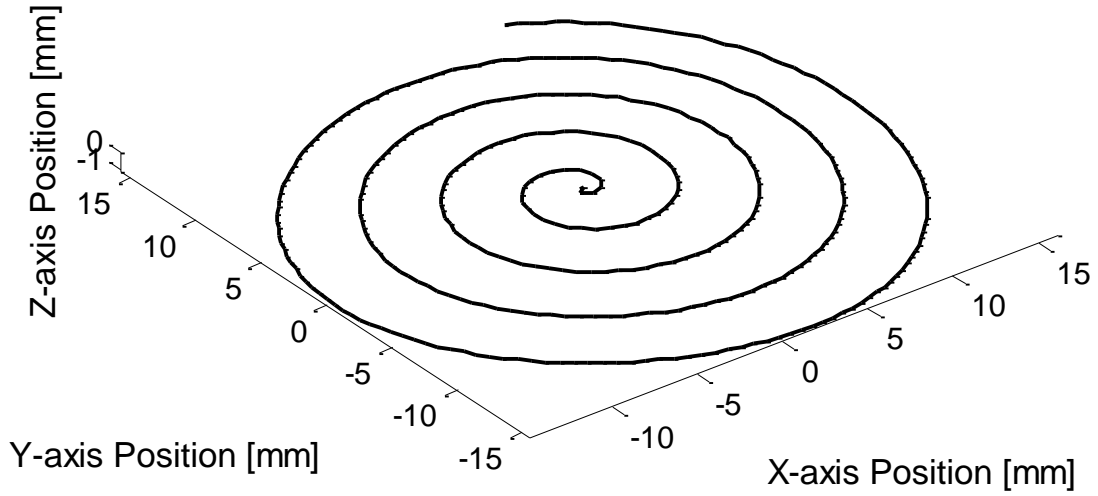


Figure 8.7: 3-D image of the revised spiral reference trajectory for experimental testing. The modified cartesian inputs were derived by mapping the angle input to cartesian inputs using the MATLAB command **interx**. Small changes due to the Jacobian mapping matrix resulted in a nonplanar raster trajectory. Note that the z-axis label indicates much smaller vertical movements than in the XY plane.

a PKM system is to following a particular reference trajectory with respect to the end effector position. However, the control and sensor information in the system relates to the individual leg link motors and angles. Therefore, a transformation between the end effector position and the input/output relationship between the leg link angles and motor torques must be utilized. This relationship, simplified through the use of a Jacobian transformation matrix, is given in (8.1) and (F). Using the Jacobian mapping technique originally presented in Chapter 7, a modified cost function and norm optimal learning controller can be determined.

Rewrite the cost function (6.11) as

$$\mathcal{J} = \bar{\mathbf{e}}_{j+1}^T \hat{\mathbf{Q}} \bar{\mathbf{e}}_{j+1} + \mathbf{u}_{j+1}^T \hat{\mathbf{S}} \mathbf{u}_{j+1} + (\mathbf{u}_{j+1} - \mathbf{u}_j)^T \hat{\mathbf{R}} (\mathbf{u}_{j+1} - \mathbf{u}_j), \quad (8.5)$$

where $\bar{\mathbf{e}}_{j+1}$ is a vector containing the end effector positions in terms of the XYZ carte-

sian coordinate frame and the control signals are the angle inputs to the individual leg links on the PKM. The angular position error of the leg links can be defined as

$$\mathbf{e}_{j+1} = \mathbf{e}_j - \mathbf{P}(\mathbf{u}_{j+1} - \mathbf{u}_j) \quad (8.6)$$

where \mathbf{P} is the dynamic relationship between the motor torque and the leg link arm angle. Using the Jacobian matrix (J) to map from rotation angle to end effector position results in $\bar{\mathbf{e}}_{j+1} = J \cdot \mathbf{e}_{j+1}$.

For learning controller design purposes, the coordinated norm optimal learning controllers can be determined by substituting $\bar{\mathbf{e}}_{j+1} = J \cdot \mathbf{e}_{j+1} = J \cdot (\mathbf{e}_j - \mathbf{P}(\mathbf{u}_{j+1} - \mathbf{u}_j))$ into (8.5), taking the derivative with respect to \mathbf{u}_{j+1} , and rearranging the solution to yield the coordinated norm optimal learning controllers for use on the PKM system.

$$\begin{aligned} \mathbf{u}_{j+1} &= \mathbf{L}_u \mathbf{u}_j + \mathbf{L}_e \bar{\mathbf{e}}_j \\ \mathbf{L}_u &= (\bar{\mathbf{P}}^T \hat{\mathbf{Q}} \bar{\mathbf{P}} + \hat{\mathbf{S}} + \hat{\mathbf{R}})^{-1} (\bar{\mathbf{P}}^T \hat{\mathbf{Q}} \bar{\mathbf{P}} + \hat{\mathbf{R}}) \\ \mathbf{L}_e &= (\bar{\mathbf{P}}^T \hat{\mathbf{Q}} \bar{\mathbf{P}} + \hat{\mathbf{S}} + \hat{\mathbf{R}})^{-1} \bar{\mathbf{P}}^T \hat{\mathbf{Q}}. \end{aligned} \quad (8.7)$$

In (8.7), $\bar{\mathbf{P}} = J \cdot \mathbf{P}$ is the dynamic relationship between the end effector position and the motor torques. Note that the update law is a function of the end effector tracking errors $\bar{\mathbf{e}}_j$.

8.2.1 Case 1 Design

As described in Subsection 8.1.3, the first case study focuses on individual trajectory tracking of the end effectors. This requires independent multi-axis trajectory tracking controllers to be identified for each PKM system. Following the process detailed in Subsection 6.3.1 and illustrated in Figure 6.1, the set of weighting matrix gains selected for individual axis trajectory tracking of each system are presented in Table

Table 8.2: Weighting Matrix Gains for Case 1

Matrix	$\Gamma 1$	$\Gamma 2$	C	$B1$	$B2$	K	$X1$	$X2$	F	Σ
\hat{Q}	I	0	n/a	I	0	n/a	I	0	n/a	$[1.0, 0.8, 1.0] \cdot I$
\hat{S}	I	0	n/a	I	0	n/a	I	0	n/a	$1.0 \cdot I$
\hat{R}	I	0	n/a	I	0	n/a	I	0	n/a	$0.5 \cdot I$

Table 8.3: Weighting Matrix Gains for Case 2

Matrix	$\Gamma 1$	$\Gamma 2$	C	$B1$	$B2$	K	$X1$	$X2$	F	Σ
\hat{Q}	n/a	n/a	n/a	n/a	n/a	n/a	0	I	(8.8)	$[1.0, 0.8, 1.0] \cdot I$
\hat{S}	I	0	n/a	I	0	n/a	I	0	n/a	$1.0 \cdot I$
\hat{R}	I	0	n/a	I	0	n/a	I	0	n/a	$0.5 \cdot I$

8.2. The σ_q , σ_s , and σ_r gains were heuristically tuned on the system to guarantee good performance results while maintaining robustness to model uncertainty and noise. The procedure for heuristic tuning follows the guidelines detailed in Subsection 3.3.4. Note that due to the similarity in plant dynamics, the same set of gains can be selected for each system, with the exception of σ_Q for agent 2. Despite similar dynamics, this system required a slightly more conservative tracking gain.

8.2.2 Case 2 Design

The second case study focuses on the formation objective, which requires coordinated movements amongst the three systems in order to maintain the desired shape. Using a similar approach to the design for case 1, coordinated norm optimal controllers for formation tracking were designed using the following weighting matrix gains (see Table 8.3). Note that once again, the σ_q , σ_s , and σ_r gains were heuristically tuned on the system to guarantee good performance results while maintaining robustness to model uncertainty and noise.

Applying the definitions of the formation errors used in Figure 4.5, the F matrix

used to map the XY cartesian end effector positions into the formation errors, $\mathbf{E}_{d1,d2,d3}$ and $\mathbf{E}_{h1,h2,h3}$, was defined in (6.15) as,

$$\mathbf{F} = F \cdot \mathbf{I}$$

where

$$F = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

As stated previously, identical reference trajectories were applied to each system, therefore F is a time-invariant matrix. Note that the z -axis errors were not included in the construction of F for the \mathbf{F} transformation matrix since the formation errors $\mathbf{E}_{d1,d2,d3}$ and $\mathbf{E}_{h1,h2,h3}$ are not a function of the z -axis. If the formation errors along the z -axis were included in the design objective, the \mathbf{F} mapping matrix would be redefined to include the z -axis errors, and F would change from a 6x6 matrix to a 9x9 matrix, respectively.

As can be seen from Table 8.3, the formation tracking approach is a function of position tracking and therefore only affects the design of the $\hat{\mathbf{Q}}$ weighting matrix. The $\hat{\mathbf{S}}$ and $\hat{\mathbf{R}}$ weighting matrices are designed for individual system control. If one wanted to consider coordinated control signals or rate of change of the control signals, the gains for the $\hat{\mathbf{S}}$ and $\hat{\mathbf{R}}$ weighting matrices would be similar to the gains for $\hat{\mathbf{Q}}$ presented in Table 8.3.

Table 8.4: Weighting Matrix Gains for Case 3 with the Raster Trajectory

Matrix	$\Gamma 1$	$\Gamma 2$	C	$B1$	$B2$	K	$X1$	$X2$	F	Σ
\hat{Q}	I	0	n/a	I	0	n/a	$0.1 \cdot I$	$0.9 \cdot I$	(8.8)	$[1.0, 0.8, 1.0] \cdot I$
\hat{S}	I	0	n/a	I	0	n/a	I	0	n/a	$1.0 \cdot I$
\hat{R}	I	0	n/a	I	0	n/a	I	0	n/a	$0.5 \cdot I$

Table 8.5: Weighting Matrix Gains for Case 3 with the Spiral Trajectory

Matrix	$\Gamma 1$	$\Gamma 2$	C	$B1$	$B2$	K	$X1$	$X2$	F	Σ
\hat{Q}	I	0	n/a	I	0	n/a	$0.5 \cdot I$	$0.5 \cdot I$	(8.8)	$[1.0, 0.8, 1.0] \cdot I$
\hat{S}	I	0	n/a	I	0	n/a	I	0	n/a	$1.0 \cdot I$
\hat{R}	I	0	n/a	I	0	n/a	I	0	n/a	$0.5 \cdot I$

8.2.3 Case 3 Design

The third and final case study focuses on a combination of the first two case studies. In this approach, trajectory tracking and formation tracking are both weighted. This requires a dual design in which individual tracking control is combined with coordinated control in order to improve the trajectory tracking performance, while maintaining the desired shape. The weighting on the two objectives for the combined controller was specific to the desired reference trajectory. The gains were selected to guarantee some performance improvement with the combined controller. Using a similar approach to the design for case 1 and case 2, coordinated norm optimal controllers for combined trajectory and formation tracking were designed using the weighting matrix gains given in Tables 8.4 and 8.5. Note that once again, the σ_q , σ_s , and σ_r gains were heuristically tuned on the system to guarantee good performance results while maintaining robustness to model uncertainty and noise.

Assuming the same derivation for the formation errors, (8.8) defines the mapping matrix from position errors to formation errors. Similar to case 2, the \hat{S} and \hat{R} weighting matrices have not been affected by combining trajectory and formation

tracking into a single $\hat{\mathbf{Q}}$ weighting matrix design.

8.3 Experimental Results

Using the coordinated norm optimal learning controllers designed in Section 8.2, experimental results can be obtained from the testbed in Figure 8.1. The tests consisted of the three different case studies applied to the experimental set-up for the two reference trajectories provided in Subsection 8.1.2. Prior to testing, a few additional implementation details should be addressed.

8.3.1 Implementation

The experimental set-up is run using a realtime LABVIEW c-RIO chassis, while the ILC update is applied in MATLAB. The LABVIEW interface reads in the reference input angles in degrees via a text file. Stabilizing feedback controllers (8.3) ensure nominal tracking performance for the first iteration. Rotary encoders on the motors track the angular positions of the bottom leg links, saving the output data as a text file. Once the iteration is complete, the output data is sent to MATLAB for the ILC update. Note that the output data is in terms of angle position in degrees, whereas the ILC update only considers end effector tracking errors (8.7). Using the MATLAB function **interx**, the output data is changed from degrees to radians and then converted from angle positions to end effector positions in the XYZ coordinate frame. Applying the ILC update law, the new ILC input signal is added to the reference input and saved in a new reference trajectory text file. In order to ensure a smooth signal, the combined reference and ILC update signals are filtered using a Gaussian filter with a 5 Hz bandwidth for the raster trajectory and a 3 Hz bandwidth for the spiral trajectory. The spiral trajectory is a more aggressive control signal, therefore the input signal must be filtered with a lower bandwidth to ensure robust

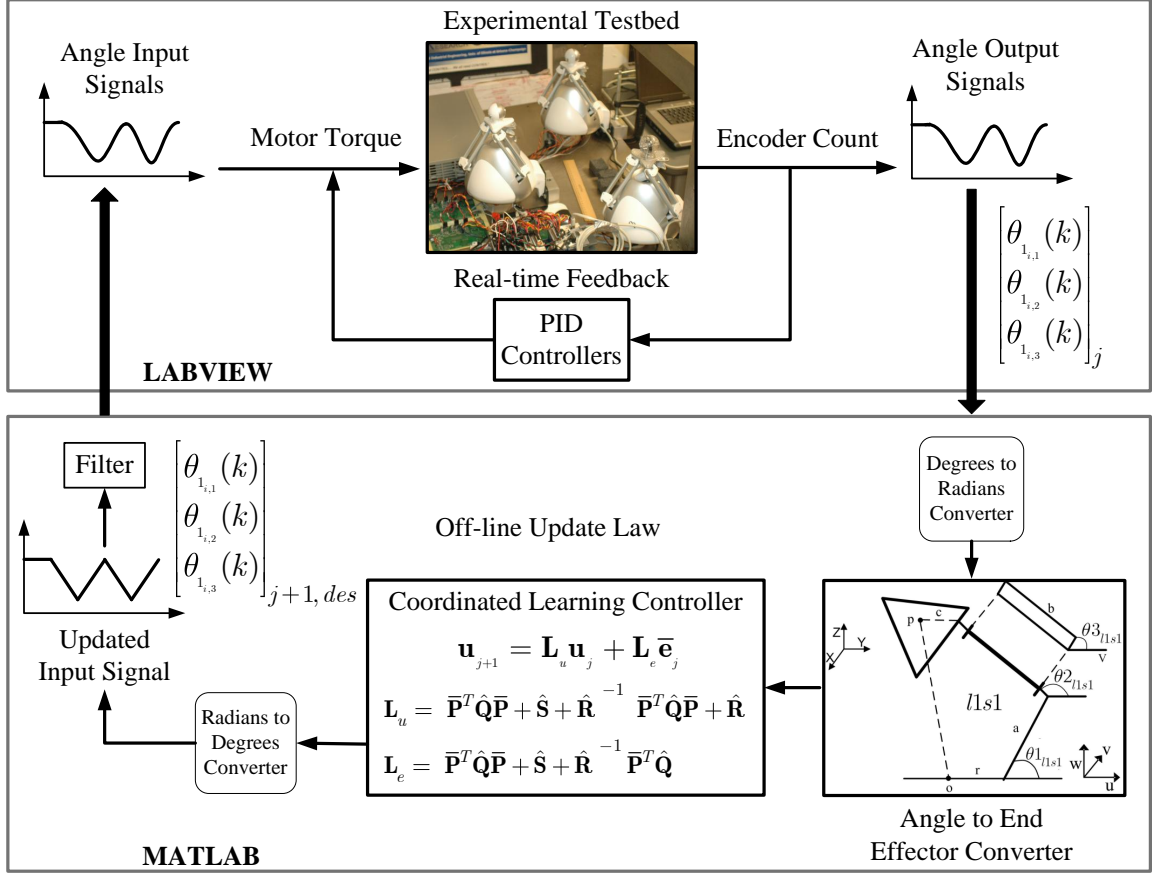


Figure 8.8: Block diagram of the implementation scheme used for experimental testing of the system illustrated in Figure 8.1

convergence of the system. Figure 8.8 presents a block diagram of this process. Note that the series ILC architecture is used with this system.

8.3.2 Raster Trajectory Results

The first set of experimental results comes from the raster trajectory presented in Subsubsection **Raster Trajectory**, Figure 8.4. Applying the same angle input signals (Figure 8.3) to all three systems, for the three case studies described in Subsections 8.2.1-8.2.3, the following performance results were achieved.

Figures 8.9 and 8.10 present normalized RMS tracking errors for the x - and y -axis position errors of system 1 and E_{d3} and E_{h3} formation tracking errors. These

signals show representative performance results for the position and formation tracking errors, respectively. RMS tracking results for the additional x - and y -axis position and $E_{h,d}$ formation errors can be found in Appendix G. As can be seen from these two figures, trajectory tracking leads to the lowest RMS position errors, while formation tracking leads to the lowest RMS formation errors. By setting the gains for the $\hat{\mathbf{Q}}$ weighting matrix to either an all trajectory or all formation tracking objective, the performance in terms of the other objective is generally sacrificed. An alternative approach was considered in the third case scenario in which the two objectives were weighted simultaneously. This led to the performance results marked 'trajectory + formation objective'. Although the weighting for the two objectives was heavy favored towards the formation objective, it only resulted in improved performance results for the horizontal formation tracking. The trajectory tracking and vertical formation tracking errors were relatively unchanged from the trajectory objective case. Additional weighting combinations may lead to more distinct differences between the trajectory and formation errors and can be considered depending on the design objectives and desired performance metrics.

Figure 8.11 was included to illustrate the performance similarities amongst the three individual systems. Although the three systems may start off with different RMS error values, all three result in approximately the same converged RMS error. This similarity in performance enabled the use of identical control designs for each system. Systems with dissimilar dynamics or performance capabilities may warrant distinct controller designs for each independent system.

In addition to the standard method of using RMS error signals to evaluate the performance capabilities of a given controller, it is important to compare the performance capabilities with respect to the actual position and contour tracking. Figure 8.12 compares the XY tracking performance of trajectory versus formation control design for system 1. This figure clearly indicates the trajectory tracking performance

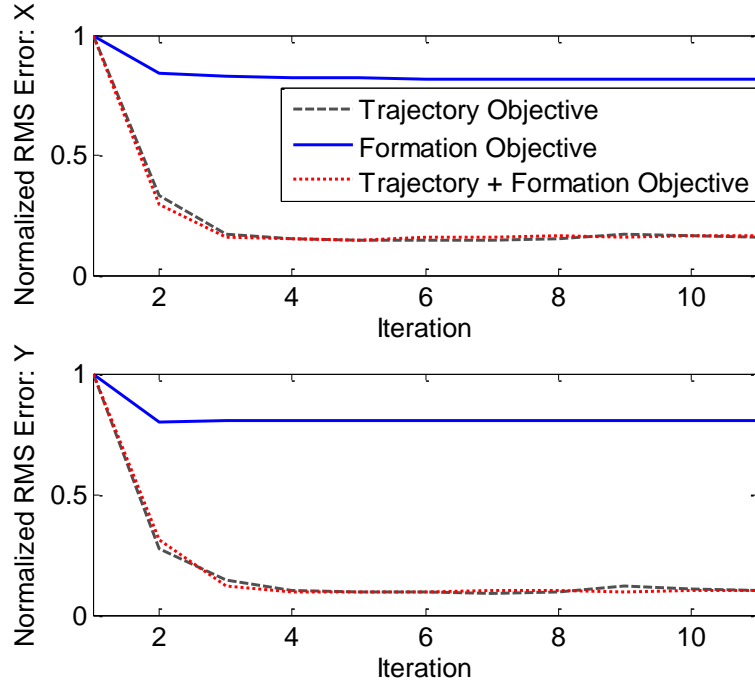


Figure 8.9: Normalized x - and y -axis position tracking RMS errors for system 1. Note that focusing on trajectory tracking results in the lowest converged RMS error. Modifying the controller for equal weighting on the two objectives improves the position tracking, resulting in nearly equivalent performance capabilities. Additional weighting combinations may result in decreased tracking performance. Note that there is a tradeoff between decreased RMS formation and trajectory tracking errors.

degradation that occurs when switching from trajectory to formation tracking control. On the other hand, placing all of the weight on the trajectory tracking objective results in a system which can track the given rastered trajectory fairly accurately.

Figures 8.13 and 8.14 provide contoured representations of the formation tracking capabilities of the multi-agent system for case 1 and case 2, respectively. These figures illustrate the relative area over which the formation tracking occurs, thereby indicating how accurately the multi-agent system is able to maintain the desired shape.

Although the two cases appear quite similar, the formation tracking controller has a slightly smaller area over which it maintains the desired shape. This corroborates the results presented in Figure 8.10 in which the formation tracking controller exhibits

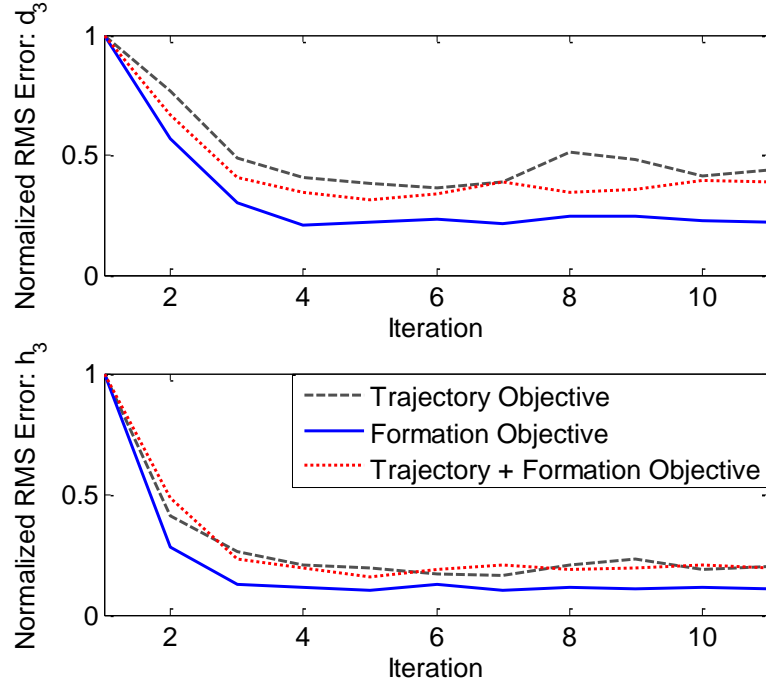


Figure 8.10: Normalized E_{d3} and E_{h3} formation tracking RMS errors. Note that focusing on formation tracking results in the lowest converged RMS error. Modifying the controller for equal weighting on the two objectives improves the horizontal formation tracking, but only minimally for this trajectory and these systems. Additional performance benefits can be obtained by weighting the formation objective more heavily. Note, however, that there is a tradeoff between decreased RMS formation and trajectory tracking errors.

the most improved formation tracking performance.

8.3.3 Spiral Trajectory Results

In order to demonstrate the versatility of the coordinated learning control scheme presented in Chapter 6, experimental results for an additional reference trajectory are presented in this section. Implementing the spiral reference trajectory introduced in Subsubsection **Spiral Trajectory**, the performance capabilities of the three case studies (Subsections 8.2.1-8.2.3) can be evaluated.

Similar to the results presented in Subsection 8.3.2, normalized RMS plots for the system 1 x - and y -axis position and E_{d3} and E_{h3} formation errors illustrate the performance differences between weighting the trajectory versus formation objective,

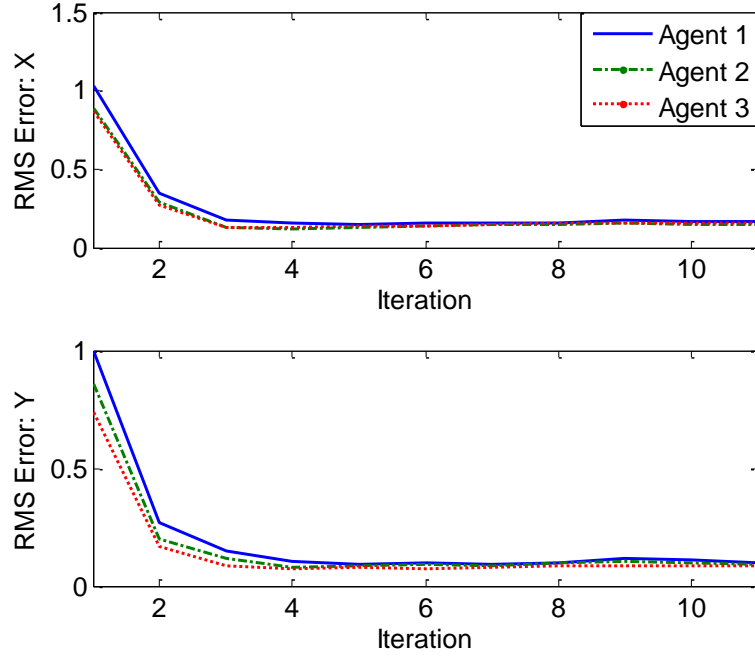


Figure 8.11: Y-axis trajectory tracking RMS errors for systems 1-3. Note that despite different initial RMS values, all three systems result in approximately the same converged RMS tracking error.

Figures 8.15 and 8.16. In Figure 8.15, the RMS position tracking errors for the x - and y -axis have degraded for the formation objective case. This degradation results from the control effort required to minimize the formation tracking errors for this particular trajectory. Given the aggressive start to the reference signal, the trajectory tracking of the three agents must be compromised in order to maintain the desired shape. A combined controller can be used to tradeoff between the formation and trajectory tracking emphasis depending on the design requirements. The combined controller results presented in Figure 8.15 used the modified weighting values in which the weight on trajectory tracking was 0.5, resulting in a weighting gain of 0.5 on the formation objective.

Figure 8.17 illustrates the similarities in terms of performance capabilities for the three individual agents within the multi-agent system by plotting the RMS x - and y -axis error signals for all three systems. These results were achieved using the gain

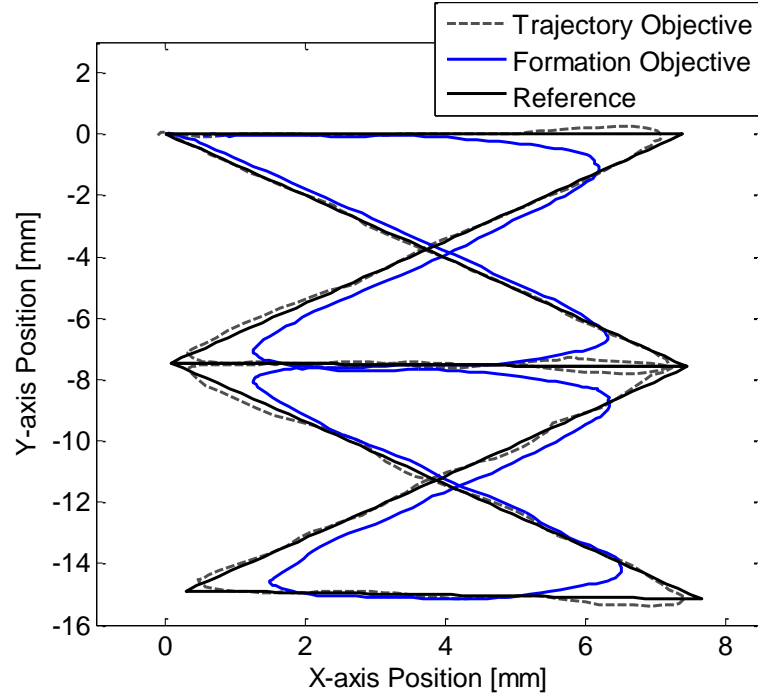


Figure 8.12: XY position tracking for agent 1. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.

selection for case 1 in which all of the weighting is on trajectory tracking.

Figure 8.18 presents the XY tracking performance of system 1 for trajectory versus formation control. As can be seen from the figure, trajectory control yields a system that is better able to follow the desired reference trajectory, despite tracking errors at the start. Formation control focuses on maintaining a desired shape with respect to the other agents in the system, thereby requiring all three systems to settle into a reference trajectory that does not impede the formation tracking performance. This results in poor trajectory tracking as illustrated in this figure.

The last two figures show a comparison of the formation tracking performance of the multi-agent system for trajectory versus formation control. The contoured surface indicates the total area over which the formation travels for a given iteration. The red lines show the desired formation shape. Figure 8.19 illustrates a slightly larger surface area over which the formation spreads as compared to Figure 8.20. This

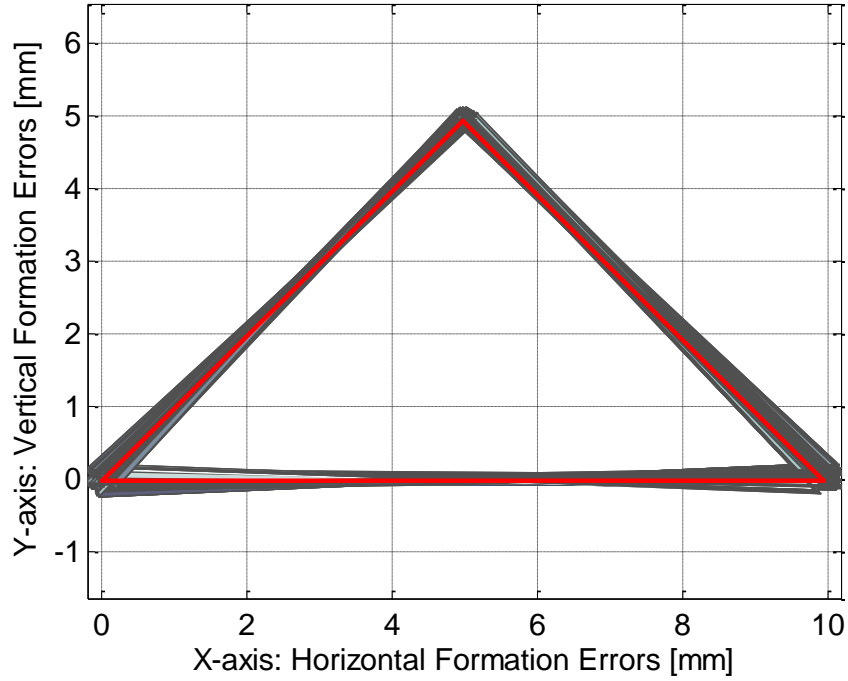


Figure 8.13: Contoured representation of the formation tracking capability of the multi-agent system from Figure 8.1. Desired formation is given in red, while the gray mesh represents the relative area over which the formation tracks for all weighting on the trajectory objective.

difference is due to the small reduction in formation tracking performance resulting from the trajectory tracking controller. Recall from Figure 8.16 that the difference between formation and trajectory tracking is relatively small. Both of these figures show larger surface areas for the trajectory and formation controllers as compared to the contoured plots for the raster trajectory, Figures 8.13 and 8.14. In the raster trajectory, the movements are repeated and contain several sections of straight lines, whereas the spiral trajectory is a more complicated pattern with a constantly changing radii. The repetitive sections and straight lines of the raster trajectory allow the three systems to more easily maintain the desired formation as illustrated in Figures 8.13 and 8.14.

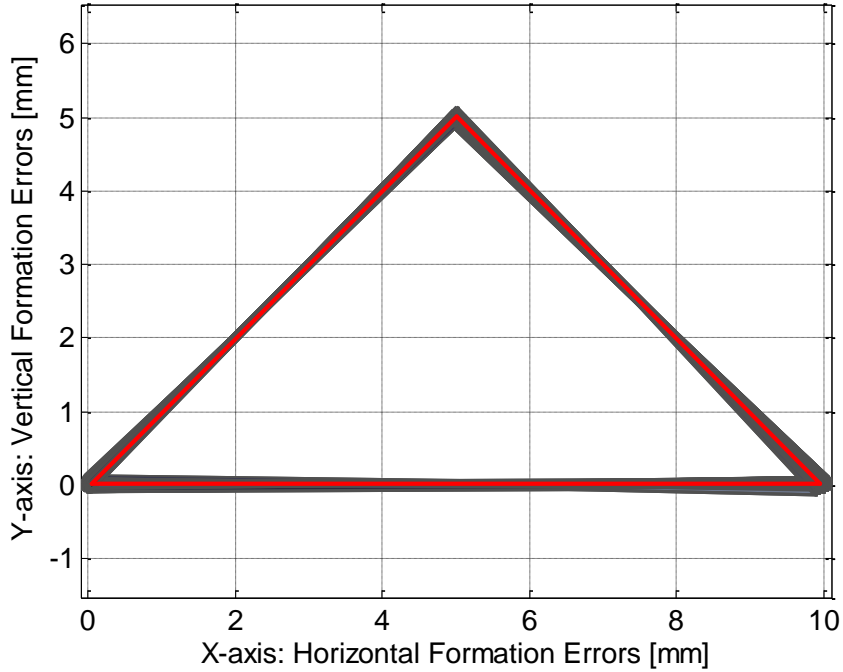


Figure 8.14: Contoured representation of the formation tracking capability of the multi-agent system from Figure 8.1. Desired formation is given in red, while the gray mesh represents the relative area over which the formation tracks for all weighting on the formation objective.

8.3.4 Formation Tracking in the Presence of Model Uncertainty and Exogenous Disturbances

The experimental results presented in Subsections 8.3.2 and 8.3.3 show relatively small differences in the formation tracking performances for trajectory versus formation tracking controllers. The three systems used in the experimental setup show extremely similar system dynamics and performance capabilities, Figures 7.5, 8.11, and 8.17, respectively. This similarity enables the three systems to maintain relatively good formation tracking performance even when the control signals are decoupled from each other.

In an effort to more clearly illustrate the potential differences between trajectory and formation tracking, the formation errors are compared for case 1 and case 2 in the

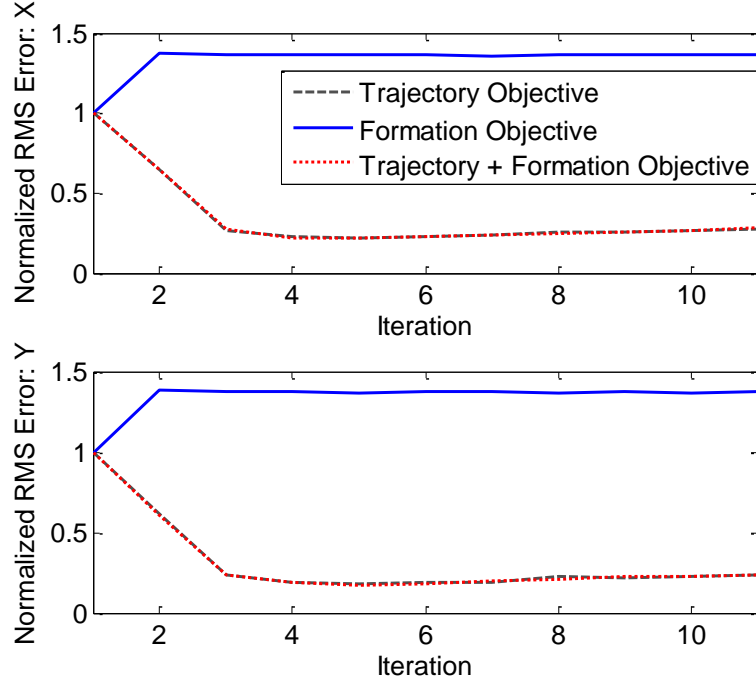


Figure 8.15: Normalized position tracking RMS errors for system 1. Modifying the controller for dual weighting on the two objectives improves the position tracking performance over the performance using formation control. This results in almost equivalent performance capabilities as compared to the trajectory tracking objective.

presence of model uncertainty and exogenous disturbances. The model uncertainty is introduced through the addition of a modified end effector on system 1, creating slightly different mass and inertial elements in the system dynamics. Attaching an elastic band to the modified end effector generates a stochastic force disturbance at various times during the trajectory. These changes are illustrated in Figure 8.21. In addition to perturbing system 1, the tracking gain σ_Q for the three systems was increased to 5, while the bandwidth on the Gaussian filter was changed to 4 Hz for the formation objective case. These changes were designed to push the three systems towards better formation tracking in the presence of the perturbations on system 1. However, the gains for the trajectory objective were not changed as they were already designed for robustness. Applying the spiral trajectory from Figure 8.7, the following experimental results were achieved.

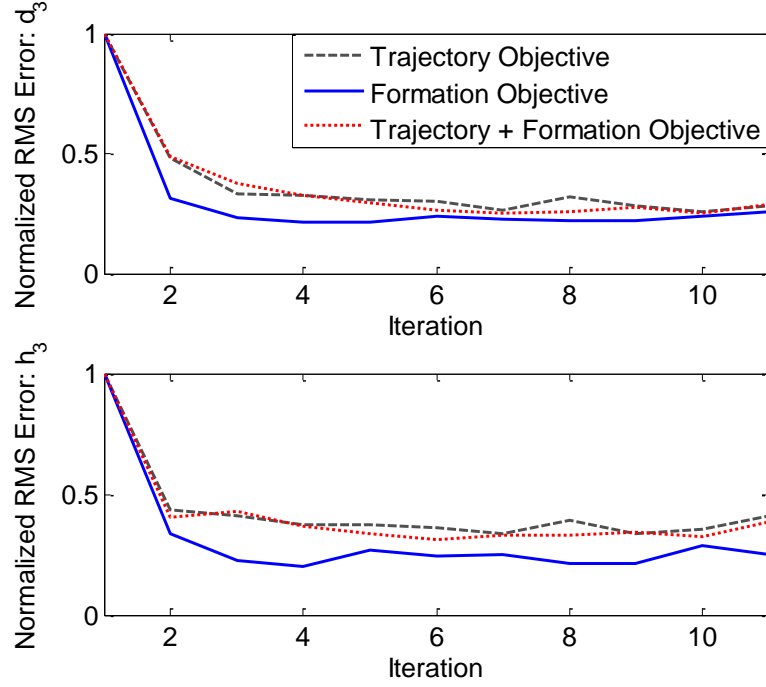


Figure 8.16: Normalized E_{d3} and E_{h3} formation tracking RMS errors. Note that focusing on formation tracking results in the lowest converged RMS error. Applying equal weighting on the two objectives may improve the formation tracking as compared to trajectory tracking, but only minimally for this trajectory and these systems.

Figure 8.22 presents the RMS trajectory tracking errors for system 1. Similar to previous results, the trajectory tracking controller results in the most improved trajectory tracking performance. Figure 8.23, on the other hand, illustrates the formation tracking capabilities of the formation controller as compared to the trajectory tracking control approach. Note that the introduction of the disturbance and model uncertainty degrades the formation tracking performance of the trajectory controller. By switching to formation tracking, the three systems coordinate their tracking performances in order to overcome the uncertainty and disturbances in System 1 and maintain the desired formation.

Figure 8.24, 8.25, and 8.26 present a trajectory and formation tracking perspective of the RMS error results. Figure 8.24 shows the XY tracking performance of the two controllers for agent 1, while Figures 8.25 and 8.26 present a contoured representation

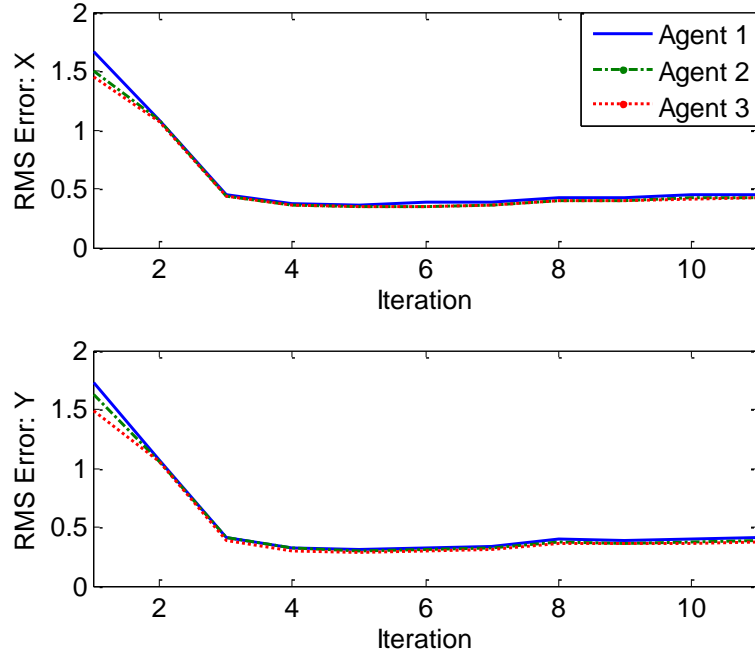


Figure 8.17: Y-axis trajectory tracking RMS errors for systems 1-3. Note that despite different initial RMS values, all three systems result in approximately the same converged RMS tracking error for the spiral trajectory.

of the formation tracking performance. As these figures illustrate, when subject to uncertainty and disturbances, the individual objectives are more closely followed with trajectory tracking, while the desired shape is most closely maintained through coordination of the three agents.

8.4 Conclusions

The experimental results presented in this chapter clearly illustrate the versatility of the coordinated design framework presented in Chapter 6. The performance benefits and tradeoffs presented in Figures 8.9 - 8.26 demonstrate how the trajectory and formation tracking of a multi-agent system is affected by selecting different design objectives. By selecting two diverse trajectories, these results highlight the benefits that exist in this design framework. While the three independent agents used in

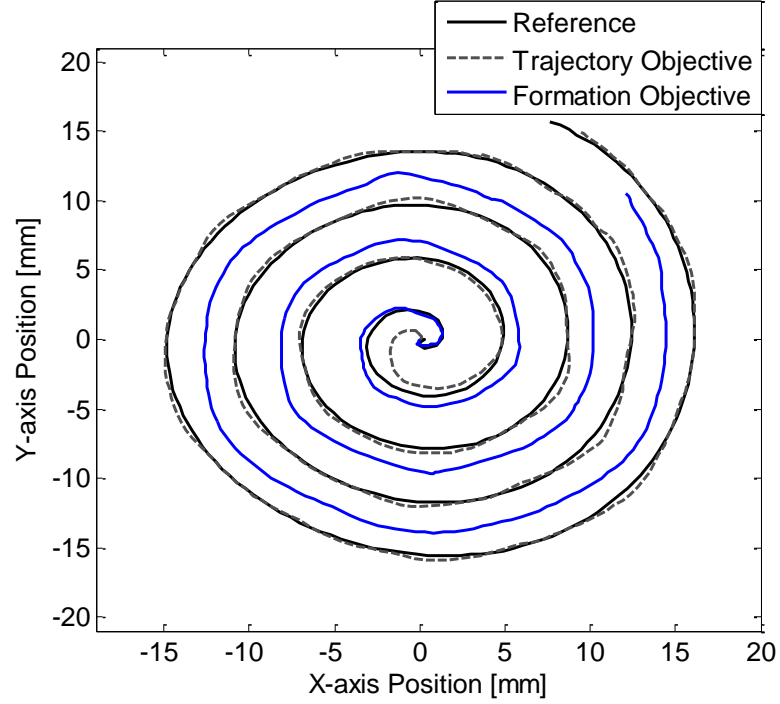


Figure 8.18: XY position tracking for agent 1. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.

this multi-agent systems were similar, thereby enabling a single plant model and 2-D design to be utilized, this approach is versatile enough to be used with dissimilar systems (for a dissimilar system example, please see Appendix J).

The next and last chapter in this dissertation will provide some concluding remarks about the research presented in this dissertation, as well as a brief introduction to some of the future areas that are appropriate for further study. This will include some discussion on additional applications for the coordinated learning control framework presented here, as well as some natural theoretical extensions of the current design.

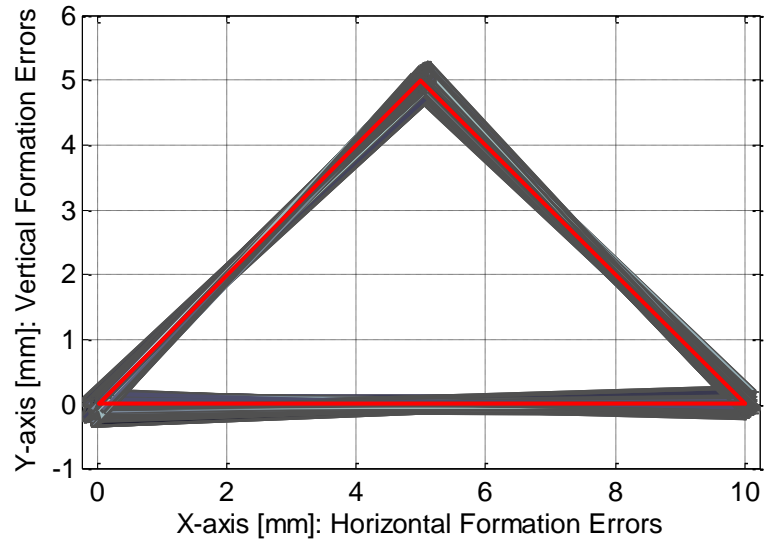


Figure 8.19: Contoured representation of the formation tracking capability of the multi-agent system from Figure 8.1. Desired formation is given in red, while the gray mesh represents the relative area over which the formation tracks for all weighting on the trajectory objective.

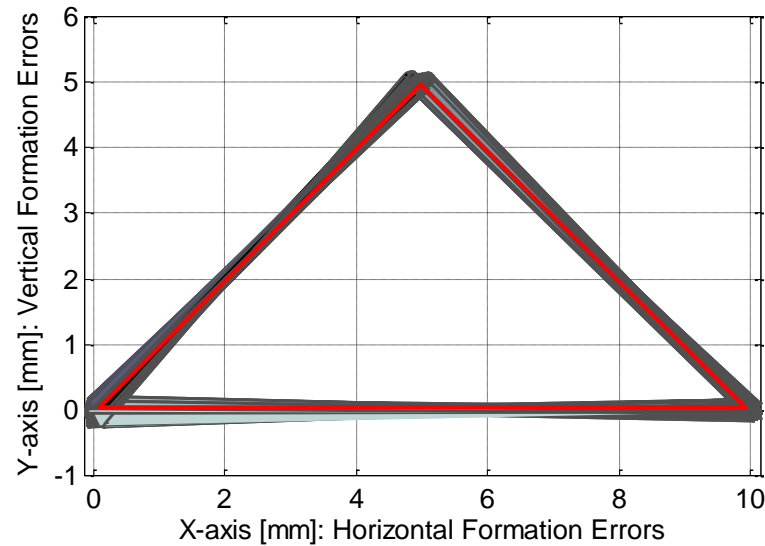


Figure 8.20: Contoured representation of the formation tracking capability of the multi-agent system from Figure 8.1. Desired formation is given in red, while the gray mesh represents the relative area over which the formation tracks for all weighting on the formation objective.

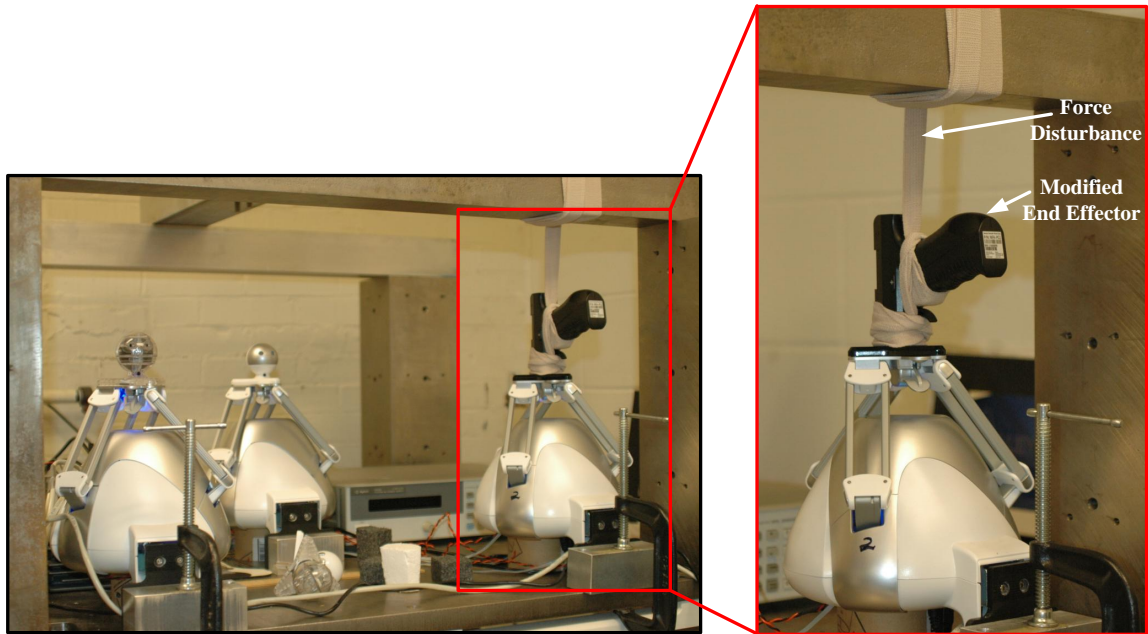


Figure 8.21: Image of the experimental system with model and force perturbations introduced to system 1.

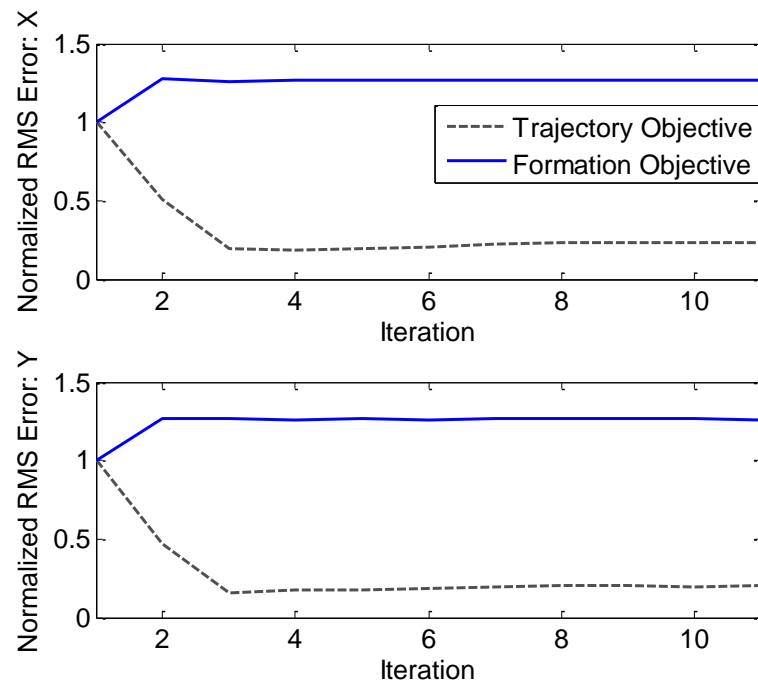


Figure 8.22: Normalized x - and y -axis position tracking RMS errors for system 1 when subject to model uncertainty and a force disturbance.

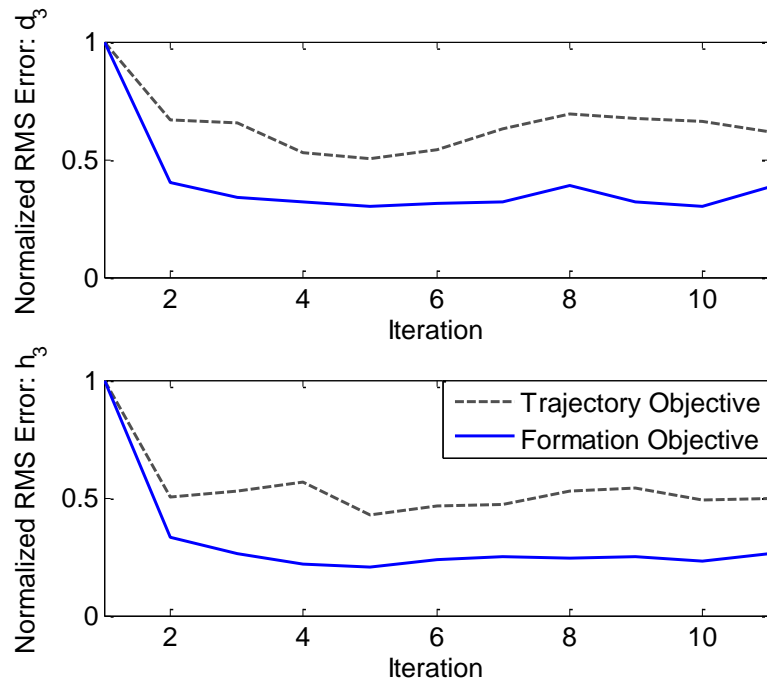


Figure 8.23: Normalized E_{d3} and E_{h3} formation tracking RMS errors when agent 1 is subject to model uncertainty and force disturbance.

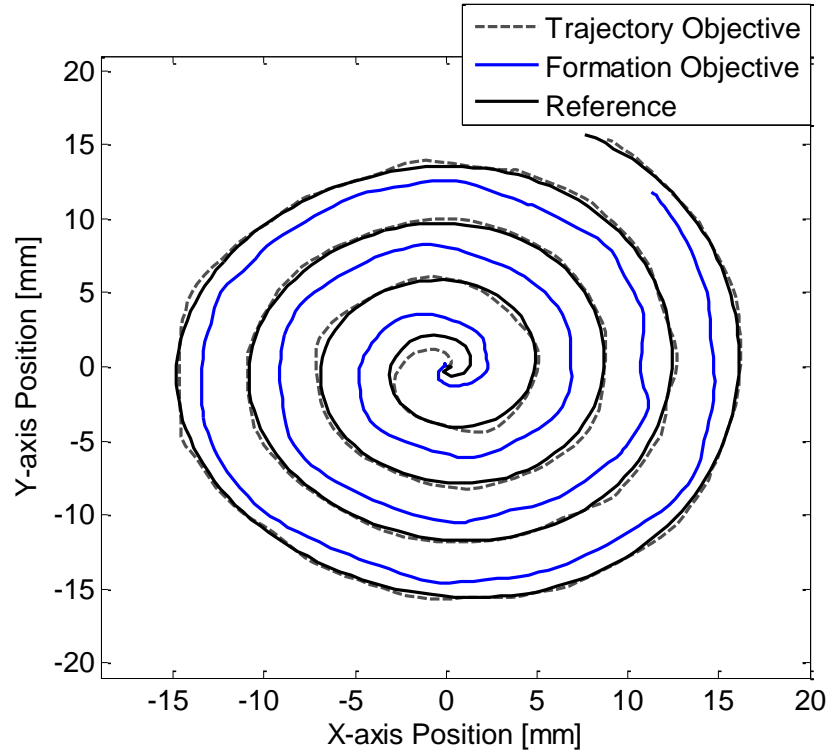


Figure 8.24: XY position tracking for agent 1 when subject to model uncertainty and a force disturbance. Note that when using a controller focused on the formation objective, the position tracking must be sacrificed in order to ensure accurate formation tracking. Although the tracking varies slightly along the spiral for the trajectory objective control scheme, a more aggressive disturbance signal would result in additional performance degradation.

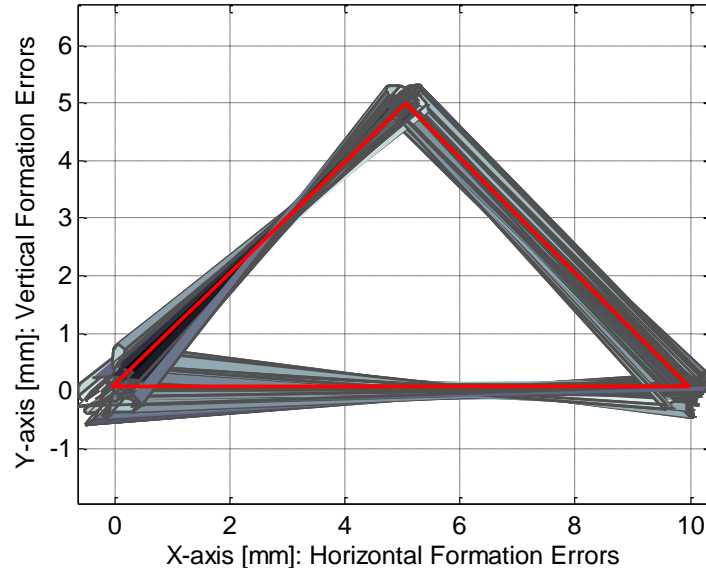


Figure 8.25: Contoured representation of the formation tracking capability of the multi-agent system when agent 1 is perturbed. Desired formation is in red, the gray mesh represents the relative area over which the formation tracks for a trajectory tracking controller.

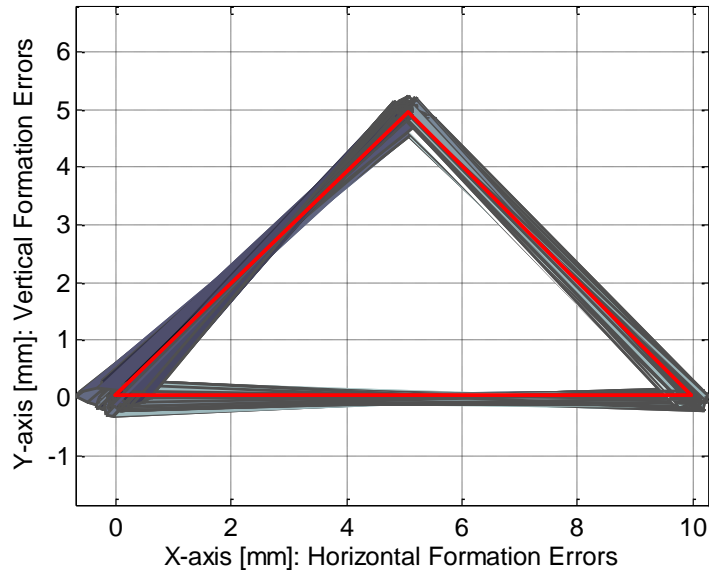


Figure 8.26: Contoured representation of the formation tracking capability of the multi-agent system when agent 1 is perturbed. Desired formation is in red, the gray mesh represents the relative area over which the formation tracks for a formation tracking controller.

Chapter 9

Conclusion and Future Direction

The research presented in this dissertation addresses a current gap in coordinated learning behavior. In multi-agent systems, the two control objectives, individual system tasks and a group task, are often controlled using independent control designs. The focus of this work was to develop a novel framework in which the control design could focus on individual tasks, the group task, or some combination of the two. This framework was formatted into an Iterative Learning Control architecture to take advantage of the repetitive nature of many multi-agent systems. There are several dominant design paradigms in ILC: linear repetitive process design, internal model design, norm optimal design, and frequency-domain design. Of these different approaches, the norm optimal framework uses the lifted domain which is a natural domain for iterative processes and is well suited for time-varying systems. Additionally, norm optimal weighting matrices can be reformatted such that different control objectives are determined through a selection of weighting gains. This flexibility provided a versatile framework for the design of coordinated learning controllers.

Control designs for cooperative behavior were split into two distinct techniques, a 2-D and 3-D approach. The 2-D approach, presented in Chapter 5, coupled the individual axes of a multi-axis system in order to minimize a contoured objective. In addition to the 2-D framework, Chapter 5 included a design methodology and detailed example to demonstrate that time-varying weighting matrices provide a means for improving both performance and robustness of a given system. As illustrated in the simulation results in Sections 5.4 - 5.6, a time-varying weighting matrix approach

enables a controller to address time-varying dynamics and disturbances that affect the performance and/or robustness of a system repeating the same process iteration after iteration. Experimental results from a multi-axis robotic testbed validated these findings.

The 3-D approach, introduced in Chapter 6, combined a 2-D coupled control scheme and a coordinated group objective into a single design framework. In this framework, different control objectives are chosen based on appropriate weighting gain selection. The 3-D design required a combination of the 2-D design methodology presented in Chapter 5 and a 3-D design process. The 3-D process incorporated the desired multi-agent formation, as well as the selection of the control objective. This process was demonstrated by the simulation example provided in Chapter 6 and the experimental testbed presented in Chapters 7 and 8. For the experimental setup in Chapter 8, coordinated movements for three parallel kinematic mechanisms (PKM) were demonstrated for two different reference trajectories. The use of PKM systems introduced an additional degree of complexity into the control design through a transformation requirement from the angular dynamics to the desired end effector position.

The simulation and experimental results for the 2-D and 3-D learning controllers demonstrated the two different design architectures for ILC. The 2-D example used a parallel approach in which the control signal was directly added to the feedback control signal for enhanced performance. A series architecture, utilized for the 3-D experimental system, combined the ILC input with the reference signal. A Gaussian filter was used to smooth the modified reference signal.

While all of the example systems used in this dissertation were homogenous systems, one of the advantages of this framework comes from the flexibility in the control design. The framework is structured such that the design can incorporate systems with similar or dissimilar dynamics, thus enabling this coordinated learning approach

to be applied to a diverse range of applications.

9.1 Additional Applications

The work in this research focused on improving coordination and motion control in manufacturing applications. While this continues to be an important area of research, particularly in emerging manufacturing areas, there are many multi-system applications that could benefit from the coordinated learning control approach presented in this dissertation. A few examples are provided in the following subsections.

9.1.1 Autonomous Vehicles

Autonomous vehicles have long been considered a coordinated control problem. Much of the work in this area has focused on distributed control methods for tracking and obstacle avoidance [8, 55, 57, 107]. Although many of the trajectories may not be strictly repetitive, the actions or tasks that a system performs are frequently repeated. The repetition in the task space provides an opportunity for performance enhancement through the use of Iterative Learning Control.

A method for utilizing ILC for systems with nonrepetitive signals is to learn a series of tasks or formations *a priori* and store those signals in a library of potential input signals. This approach has been used in manufacturing to improve the performance and versatility of a micro-Robotic Deposition process [108]. The signals learned in [108] consisted of a series of deposition tasks generally performed during routine deposition trials. One of the advantages of combining this with the coordinated learning controller is to learn the individual task, as well as group formations and objectives for a variety of options. The learned signals then provide an optimized method for performing a given task with a specified formation and group objective. By combining the learned input signals with a robust feedback controller to guaran-



Figure 9.1: Examples of autonomous vehicle applications in which multiple systems perform a coordinated task repetitively. A) Surveying or searching task using multiple autonomous underwater vehicles. B) Coordinated agrosience practices are a critical component of increased efficiency in food production. C) Search and rescue tasks can benefit from repetitive cooperative learning techniques.

tee performance in the presence of nonrepetitive disturbances, the performance and coordination of the autonomous vehicles may be improved.

This approach can be applied to autonomous land, sea, and air vehicles for a variety of applications such as search and rescue, surveying, and farming. Figure 9.1 shows some example systems in which the desired objective includes a combination of individual and group tasks.

9.1.2 Robotics

Although robotic manufacturing systems are common in industry, the term robotics defines a much larger class of systems. Multi-axis or multi-system robotics describes a wide range of systems such as assembly line robots, satellite robots, and biomedical robots to name a few. In many of these applications, the desired output requires coordination amongst multiple systems.

Figure 9.2 presents three examples of robotic systems that execute coordinated tasks repeatedly. In part A), the pick and place operation is a classic example of an assembly line application in which multiple robots perform a specified task repeatedly. The robots must coordinate their actions with the constantly moving conveyor belt, thereby coupling multiple systems with dissimilar dynamics. In part B), nine research

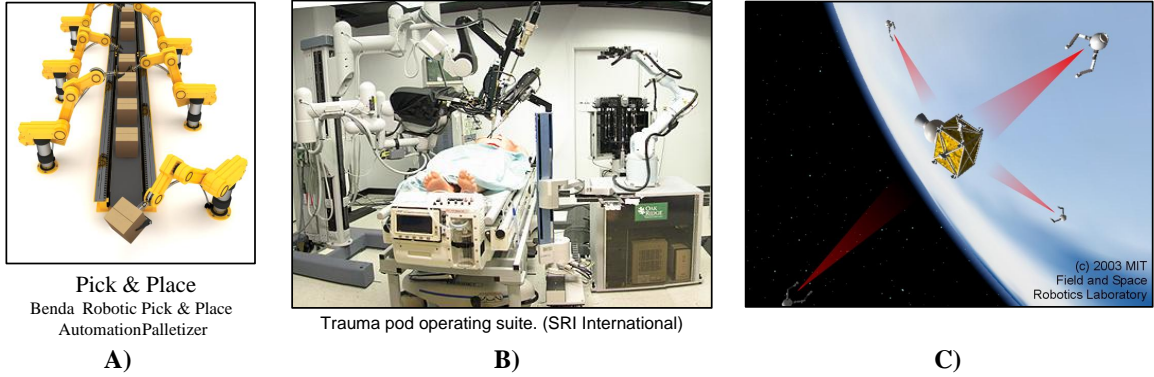


Figure 9.2: Examples of robotic applications in which multiple systems perform a coordinated task repetitively. A) Assembly line robots performing pick & place task. B) Multiple biomedical robots performing a test operation. C) A team of robots collecting information about satellite motions and dynamics using vision sensors (<http://robots.mit.edu/projects/jaxa/index.html>).

teams from around the world simultaneously coordinate the motions of a multi-robot biomedical system (the SRI International surgical robot system titled M7) for surgical operations. Lastly, the third image portrays an image from the Field and Space Robotics Lab at MIT of multiple robots assessing the performance of a satellite using vision-based sensors.

Various control schemes exist for robotic systems, many of which are focused on a single individual or group task [109]. For these control schemes, changing the task from one objective to another may require the design of an entirely different controller. In the coordinated learning framework, the objective can be switched multiple times during a given task through the use of time-varying weighting gains.

In addition to the requirement for coordinated behavior, many robotic applications require precision motion control from task to task. The coordination, combined with precision motion control for a repetitive task makes robotics well suited for the coordinate learning control framework presented in this dissertation.

9.1.3 Business Modeling

In addition to traditional engineering applications, this framework could be applied to business modeling. There are many business applications that require coordinated behavior from multiple divisions or individuals to achieve a mutually beneficial objective. The ability to learn from previous ventures, without the requirement of high fidelity models, could provide a unique approach to enhancing business practices. While the idea of integrating machine learning into business practices is not new [110], current approaches do not provide a coordinated learning approach for handling the inherently coupled nature of business. Implementing a coordinated learning approach could affect overall system efficiency, quality of work, cost, and time; four key concepts in business.

Along with the wide range of potential applications for this control approach, there are a few key limitations in the current design that should be addressed through theoretical advancements. These are addressed in the following section.

9.2 Future Theoretical Development

Many of the future theoretical developments that are natural extensions of this framework were alluded to in the previous subsections. A few of the key areas in which theoretical advancements could greatly extend the current framework include: task based learning, time-varying objectives, a hierarchical approach to classifying multiple systems, design benefits from the use of structured versus unstructured uncertainty, and performance and robustness analysis of the series versus parallel ILC architecture.

9.2.1 Task Based Learning

Task based learning is a process in which optimized control inputs for a series of predefined tasks are learned offline and stored in a library. These signals can then

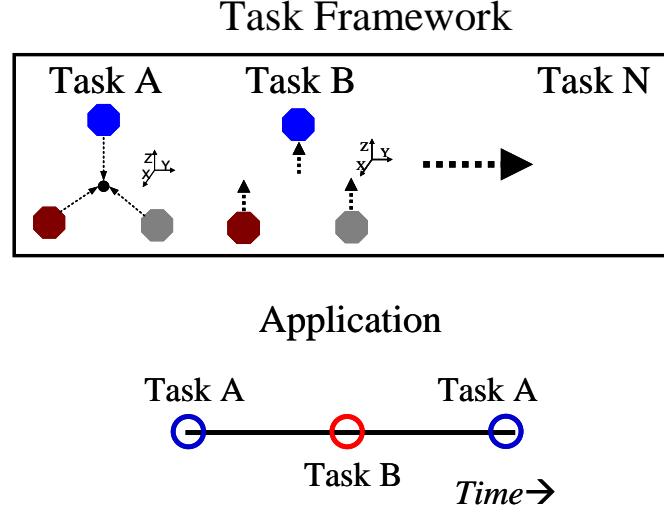


Figure 9.3: Illustration of a task based learning approach. Note that the different tasks may consist of a combination of both individual and group objectives.

be reconfigured into different patterns based on the desired reference input. Previous work in this area [108] focused on learning the task space for a micro-Robotic Deposition system. In [108], the learned tasks include actions common in many deposition applications such as starting, stopping, cornering, and steady-state flow of an extruded material. Extending this approach to a coordinated learning framework, the learned tasks now consist of a compilation of actions to achieve an individual objective, as well as different formation patterns based on group objectives. The combination of learned input signals for individual and group tasks enables a multi-agent system to combine individual and group objectives into a single pattern.

Figure 9.3 illustrates the task based learning process for multi-agent systems. As describes previously, the tasks combine individual and group requirements into a single library of learned signals.

9.2.2 Time-Varying Objective Weighting Gains

A natural extension to the concept of combining individual and group tasks into a single pattern is time-varying objective weighting gains. While time-varying weighting

gains have been illustrated for the 2-D coupled controller in Chapter 5, a time-varying 3-D controller has not yet been introduced. One of the advantages of using a norm optimal framework is that any of the weighting gains can be made time-varying.

Recall the definition of the coordinated $\hat{\mathbf{Q}}$ weighting matrix from (6.6),

$$\hat{\mathbf{Q}} = \Sigma_Q \cdot [\mathbf{X1}_Q \cdot \bar{\mathbf{Q}} + \mathbf{X2}_Q \cdot \mathbf{F}^T \mathbf{F}],$$

where $\mathbf{X1}$ and $\mathbf{X2}$ are directly related to each other through their diagonal elements $(\chi_1(1), \dots, \chi_p(N))$ and $(1 - \chi_1(1), \dots, 1 - \chi_p(N))$ and are used to determine the ratio of weighting on the individual objective versus the group objective. By selecting $\chi_1(1), \dots, \chi_p(N)$ as time-varying, the control objective can be changed from individual to group tracking or some combination of the two during a given trial.

One of the motivations for time-varying the control objective is to exploit the advantages of the different control strategies. For example, if a disturbance occurs during one of the trials, this disturbance will propagate through the combined system when subject to formation tracking. However, the disturbance will only affect the specific system in the case of individual trajectory tracking. Based on the desired control objective, if a disturbance is detected during one of the trials, it may be beneficial for the controller to switch objectives during the disturbance in order to eliminate the propagation of the effect.

9.2.3 Hierarchical Approach

As the number of individual agents increases, the size of the norm optimal matrices may become computationally intractable. In an effort to address this limitation, a hierarchical approach can be applied. The concept behind this method is to simplify the problem by breaking it down into more feasible control spaces. This approach is similar to a subswarm method presented in [59]. By using a hierarchical architecture,

Hierarchical Approach

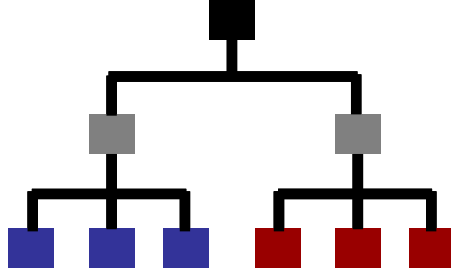


Figure 9.4: Illustration of a hierarchical approach to designing coordinated learning controllers for multi-agent systems with a large number of systems.

the individual agents may be broken down into separate groups and controlled as subpods. The individual subpods can then be considered as individual *agents* of the combined multi-agent system. By redefining the agents in the overall setup, the relative size of the system becomes more manageable. This scheme decreases the size of the norm optimal weighting matrices, thereby minimizing the computational complexity of the control design.

Figure 9.4 shows an illustration of a hierarchical approach. In this image, multiple agents are reassembled into subgroups. These subgroups are then considered individual elements in the overall system.

9.2.4 Design Benefits of Structured versus Unstructured Uncertainties

In addition to performance, robustness is a critical attribute of any learning algorithm. Analysis presented in Chapter 3 Subsection 3.3.2 considered robust convergence requirements for unstructured uncertainties. Unstructured uncertainty is defined as the use of a *full* complex perturbation matrix Δ , where at each frequency any $\Delta(j\omega)$ satisfying $\bar{\sigma}(\Delta(j\omega)) \leq 1$ is allowed [111]. While this approach is commonly used to get a simple uncertainty model, it may result in a more conservative design approach.

Structured uncertainty assumes more knowledge of the plant, thereby enabling a structured perturbation matrix such as a diagonal matrix. Recall that the convolution matrix \mathbf{P} is generally lower triangular block toeplitz.

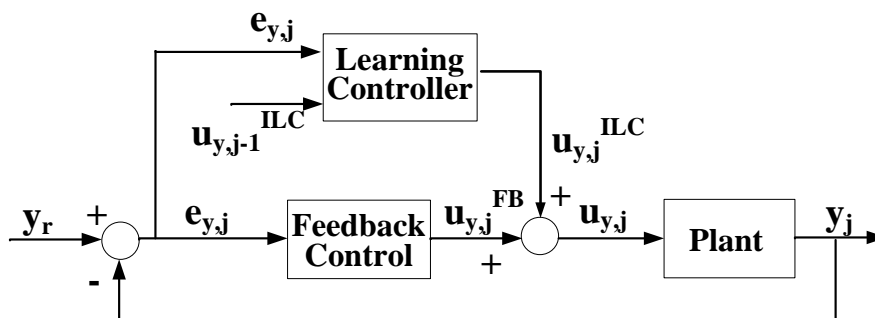
$$\mathbf{P} = \begin{bmatrix} H_{0,0} & & 0 \\ \vdots & \ddots & \\ H_{N-1,0} & \cdots & H_{N-1,N-1}. \end{bmatrix}$$

As long as the Plant is restricted to being noncausal, one may be able to take advantage of the structure in the structured uncertainty to make the design less conservative.

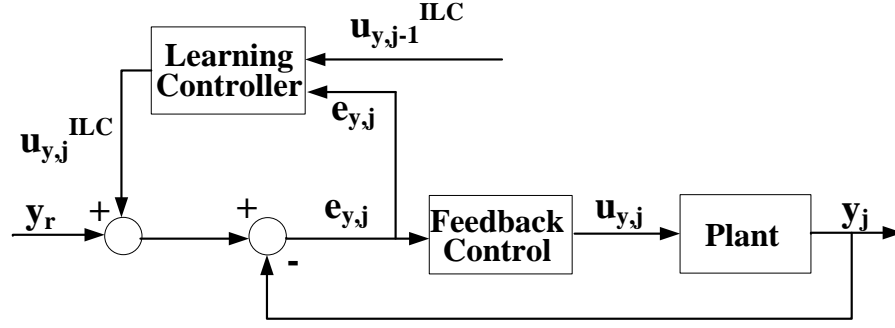
9.2.5 Performance and Robustness Analysis of the Series versus Parallel Design Architecture

The series and parallel architectures (see figures) are well known implementation structures in ILC [37]. The experimental work in this dissertation employed both architectures to demonstrate the versatility of the novel control framework. Although both structures have been used in practice, very little work has focused on analyzing the two approaches for robustness and performance benefits.

The decision to select one option over the other is often determined from the



Block diagram of the parallel ILC process.



Block diagram of the series ILC process.

experimental set-up. Many applications include preexisting systems with commercial controllers that do not allow access to existing control signals. For these systems, one must implement a series approach. For systems which enables access to either the reference or control signals, it would be beneficial to evaluate the performance and robustness benefits associated with the different architectures. These advantages may become more distinct when model uncertainty is present within the system. A novel analytical approach for comparing these two architectures and determining the appropriate design for a specific system and application would greatly benefit the ILC community.

Appendix A

Coefficients for the μ -RD Y-axis Plant (5.12) and Controller (5.13) Models

The following coefficients were used for the simulation example in Chapter 5. The x-axis plant for the μ -RD would have similar dynamics and simulation results.

$$\left[\begin{array}{l} \textit{Symbol} \\ \textit{Num} \quad \alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4 \quad \alpha_5 \\ G_y \quad 0.9963 \quad 1.768 \quad 0.9567 \quad 0.2238 \quad 0.7933 \\ \textit{Den} \quad \beta_1 \quad \beta_2 \quad \beta_3 \quad \beta_4 \quad \beta_5 \\ G_y \quad 0.9972 \quad 1.764 \quad 0.9562 \quad 0.1784 \quad 0.7898 \\ \textit{Gain} \quad K \\ G_y \quad 0.0459 \end{array} \right] \quad (\text{A.1})$$

$$\left[\begin{array}{l} \textit{Symbol} \\ \textit{Num} \quad \alpha_1 \quad \alpha_2 \quad \alpha_3 \\ k_{py} \quad 1.377 \quad 0.9147 \quad 0.776 \\ \textit{Den} \quad \beta_1 \quad \beta_2 \quad \beta_3 \\ k_{py} \quad 1.001 \quad 0.5182 \quad 0.1691 \\ \textit{Gain} \quad K \\ k_{py} \quad 1.5 \end{array} \right] \quad (\text{A.2})$$

Appendix B

Coefficients for Simulation Example in Chapter 6

The following coefficients were used for the simulation example in Chapter 6. The system consisted of three dynamically similar dual-axis systems.

<i>Symbol</i>	<i>Quantity</i>		
<i>Num</i>	α	β_1	
$G1_x(z)$	0.28355	0.8753	
$G1_y(z)$	0.2913	0.8761	
$G2_x(z)$	0.53598	0.7943	
$G2_y(z)$	0.4978	0.7861	
$G3_x(z)$	0.40552	0.7209	
$G3_y(z)$	0.41466	0.7209	(B.1)
<i>Den</i>	β_2	β_3	
$G1_x(z)$	1.00	0.6703	
$G1_y(z)$	1.00	0.6721	
$G2_x(z)$	1.00	.4999	
$G2_y(z)$	1.00	0.4843	
$G3_x(z)$	1.00	0.3719	
$G3_y(z)$	1.00	0.3720	

Appendix C

MATLAB Code for Solving Forward Kinematics Problem

This sections provides the MATLAB code used to solve the forward kinematic problem for the 3 agent PKM experimental set-up used in Chapter 8. The code utilizes an m-file previously designed in Maple known as **interx**. The **interx** program requires input from the user in the form of the radius of each sphere (i.e. 'b' in Figure 7.3) and the location of the center of each sphere denoted by the following vector,

$$\begin{bmatrix} (r + a \cos(\theta_{1i,m})) \cos(\phi_{i,m}) & (r + a \cos(\theta_{1i,m})) \sin(\phi_{i,m}) & a \sin(\theta_{1i,m}) \end{bmatrix}.$$

Using this input, the function calculates the intersection of the three spheres, and thus the solution to the forward kinematics problem.

In addition to the **interx** program, a coordinate transformation mapping is accomplished through a program titled **getposition**, provide in Section C.2. This code derives the end effector position from the experimentally determined leg link angles $\{\theta_{11,m}, \theta_{12,m}, \theta_{13,m}\}$ for each $m = 1, 2, 3$ system.

C.1 Spherical Mapping Program

This program provides the code used to solve the forward kinematic problem for the 3 agent PKM experimental set-up used in Chapter 8.

```
%% Interx program for spherical calculation of potential end effector position
```

```
% Function to calculate intersection of three spheres
% Returns NAN if no intersection or some bad condition encountered
% X1, X2, X3 are vectors of centers of three spheres
% r1, r2, r3 are radii of the three spheres
% pos =0 for lower point and 1 for higher point
% Computation Code calculated in (and copied from) Maple
% For questions/suggestions, contact hrishi.shah2002@gmail.com
% *****
```

```
function result=interx(X1,X2,X3,r1,r2,r3,pos);
if(nargin<7), pos=1; end % default value
x1=X1(1); y1=X1(2); z1=X1(3);
x2=X2(1); y2=X2(2); z2=X2(3);
x3=X3(1); y3=X3(2); z3=X3(3);
% *****
%% convert in coord sys at [x1 y1 z1] oriented same as global
x2=x2-x1; y2=y2-y1; z2=z2-z1;
x3=x3-x1; y3=y3-y1; z3=z3-z1;
```

```
a=(16*y2^2*z3*y3^2*z2*x3*r1^2*x2-4*y2^3*z3*y3*z2*x3*r1^2*x2+4*y2^3*z3*y3*z2*x3...
*x2*r3^2-4*y2*y3^3*z2*x2*z3*r1^2*x3+4*y2*y3^3*z2*x2*z3*r2^2*x3+16*z2*x3^2*x2^2...
*r1^2*y3*y2*z3-4*z2*x3^3*x2*r1^2*y3*y2*z3+4*z2*x3^3*x2*y2*z3*r2^2*y3-4*x2^3*z3...
*x3*y2*r1^2*z2*y3+4*x2^3*z3*x3*y2*r3^2*z2*y3-4*y2*z3*z2^3*y3*x3*r1^2*x2+4*y2...
*z3*z2^3*y3*x3*x2*r3^2+8*y2*z3^2*z2^2*y3*x2*r1^2*x3-4*y2*z3^2*z2^2*y3*x2*r2^2...
*x3+4*x2^2*y3^2*z2*y2^2*z3*x3^2-4*x2^4*z3^2*x3^2*y3*y2-2*z2^2*x3^4*x2^2*y2^2...
+2*z2^2*x3^3*y2^2*r1^2*x2-2*z2^2*x3^3*y2^2*x2*r3^2+2*z2^2*x3^5*x2*y2*y3+2*x2^5...
*z3^2*x3*y3*y2+2*z2^3*y3^2*y2^2*z3*x3^2+2*z2^3*y3^2*x2^2*z3*x3^2+2*z2^3*y3^2...
*x2^2*z3*r1^2-2*z2^3*y3^2*x2^2*z3*r3^2+2*x2^2*y3^4*z2*y2^2*z3-4*x2^2*y3^2*z2^2...
*x3^2*y2^2+2*x2^4*y3^2*z2*z3*x3^2-2*x2^2*y3^3*z2^2*y2*x3^2+2*x2^2*y3^3*z2^2*y2...
*z3^2+2*x2^3*y3^2*z2^2*x3*z3^2+2*x2^2*y3^2*z2*y2^2*z3^2+2*x2^2*y3^3*z2^2*y2...
*r1^2+2*x2^2*y3^2*z2^2*x3^2*r2^2+2*x2^4*y3^2*z2*z3*r1^2-2*x2^4*y3^2*z2*z3*r3^2...
-2*x2^2*y3^3*z2^2*y2*r3^2+2*x2^3*y3^2*z2^2*x3*r1^2-2*x2^3*y3^2*z2^2*x3*r3^2+2...
*y2^4*z3*x3^2*y3^2*z2+2*y2^2*z3*x3^4*z2*x2^2-4*y2^2*z3^2*x3^2*x2^2*y3^2+2*y2^3...
*z3^2*x3^2*z2^2*y3+2*y2^2*z3^2*x3^3*x2*z2^2-2*y2^3*z3^2*x3^2*x2^2*y3+2*y2^3...
*z3^2*x3^2*r1^2*y3+2*y2^2*z3*x3^4*z2*r1^2-2*y2^2*z3*x3^4*z2*r2^2+2*y2^2*z3^2...
*x3^2*x2^2*r3^2-2*y2^3*z3^2*x3^2*r2^2*y3+2*y2^2*z3^2*x3^3*x2*r1^2-2*y2^2*z3^2...
*x3^3*x2*r2^2-2*y2^2*z3^2*y3^2*x2^3*x3-4*y2^4*z3^2*y3^2*x2*x3+2*y2^2*z3^2*y3^2...
*x2^2*r3^2+4*y2^3*z3^2*y3*x2^3*x3+2*y2^5*z3^2*y3*x2*x3+4*y2*y3^3*z2^2*x3^3*x2...
+2*y2*y3^5*z2^2*x3*x2-2*y2^2*y3^2*z2^2*x3^3*x2-4*y2^2*y3^4*z2^2*x3*x2+2*y2^2...
*y3^2*z2^2*x3^2*r2^2-4*z2^2*x3^4*x2^2*y2*y3+2*z2*x3^2*x2^2*y2^2*z3^3+2*z2*x3^2...
*y2^4*z3*r1^2+2*z2^2*x3^2*y2^3*r1^2*y3-2*z2*x3^2*y2^4*z3*r3^2-2*z2^2*x3^2*y2^3...
*r3^2*y3-z2^4*y3^2*x3^2*x2^2-z2^4*y3^2*x3^2*y2^2+2*z2^3*y3^4*x2^2*z3+2*z2^3...
*y3^2*x2^2*z3^3+2*x2^2*y3^5*z2^2*y2-2*x2^2*y3^4*z2^2*y2^2-2*x2^4*y3^2*z2^2*x3^2...
+2*x2^3*y3^2*z2^2*x3^3+2*x2^4*y3^4*z2^2*z3+2*x2^3*y3^4*z2^2*x3+2*x2^2*y3^4*z2^2...
*r2^2-2*y2^4*z3^2*x3^2*y3^2+2*y2^5*z3^2*x3^2*y3+2*y2^4*z3*x3^4*z2+2*y2^2*z3^2...
```

$\begin{aligned}
& *x^3z^3x^2y^2z^3x^4x^2+2y^2z^3x^3x^2+2y^2z^3x^4z^2y^2\ldots \\
& *z^3x^3x^2x^2+2y^2z^3x^2r^3x^2-2y^2z^3x^3x^2+2y^2z^3x^4x^2+2y^2z^3x^3x^2\ldots \\
& *x^2y^2z^3x^4y^3x^2x^2-2y^2z^3x^3x^2+2y^2z^3x^4x^2+2y^2z^3x^3x^2-2y^2z^3x^4x^2\ldots \\
& *z^2x^3x^2-2y^2z^3x^3x^2-2y^2z^3x^4x^2-2y^2z^3x^3x^2-2y^2z^3x^4x^2\ldots \\
& *x^3x^4y^2z^3x^3x^2-2y^2z^3x^4x^2-2y^2z^3x^3x^2-2y^2z^3x^4x^2-2y^2z^3x^3x^2\ldots \\
& *z^3x^4y^3x^2-2y^2z^3x^3x^2-2y^2z^3x^4x^2-2y^2z^3x^3x^2-2y^2z^3x^4x^2+2\ldots \\
& *y^2x^4x^3x^4r^3x^2+2y^2z^3x^4x^3-2y^2x^4x^3x^2r^1x^4-2y^2x^4x^3x^2y^3x^4+2y^2z^3x^3x^2y^3x^3\ldots \\
& -y^2x^4x^3x^2r^3x^4-2y^2z^3x^3x^2y^3x^2-2y^2z^3x^4x^2r^1x^4-2y^2z^3x^4x^2x^4+2y^2z^3x^3x^2x^4\ldots \\
& -y^2x^2x^3x^4r^2x^4-2y^2z^3x^3x^2x^2-2y^2x^4x^3x^4x^2+2y^2x^4x^3x^4r^2x^2+2y^2x^4x^3x^5\ldots \\
& *x^2+2x^2x^4y^3x^5y^2-2x^2x^4y^3x^4y^2+2x^2x^4y^3x^4r^2x^2-2x^2x^4y^3x^4r^1x^4-2x^2x^4y^3x^6\ldots \\
& *y^2+2x^2x^2y^3x^5y^2x^3-2x^2x^2y^3x^4y^2+2x^2x^4y^3x^4r^2x^2-2x^2x^4y^3x^4r^1x^4-2x^2x^4y^3x^2\ldots \\
& *x^3x^2+2x^2x^5y^3x^2x^3x^3-2x^2x^4y^3x^2x^3x^4-2x^2x^4y^3x^2r^3x^4+2x^2x^5y^3x^4x^3-2x^2x^4\ldots \\
& *y^3x^4x^3x^2+2x^2x^4y^3x^4r^3x^2-2y^3x^6x^2x^2-2y^2x^4x^3x^6-2y^2x^6x^3x^4-2x^2x^6y^3x^4-2x^2x^4\ldots \\
& *y^3x^6-2x^2x^3x^2r^1x^2y^2z^3x^3x^2-2x^2y^3x^2r^2x^2x^2z^3x^3x^2+2x^2y^3x^2r^2x^2\ldots \\
& *x^2x^2z^3x^3x^2+2x^2x^4y^3x^2z^2z^3x^3+2y^2r^1x^4z^2x^2y^3x^3x^2+2x^2x^2y^3x^2z^2y^2\ldots \\
& *z^3x^3x^2-8x^2x^2y^3x^3z^2r^1x^2y^2z^3-2x^2x^2y^3x^2z^2y^2z^3x^3x^2-8x^2x^2y^3x^2z^2\ldots \\
& *z^3x^3x^2x^3+2y^2x^2z^3x^3x^2r^1x^2y^3x^2z^2-8y^2x^3x^3x^2r^1x^2z^2y^3-2y^2x^2z^3x^3x^2\ldots \\
& *z^2y^3x^2r^2x^2-8y^2x^2z^3x^3x^3z^2r^1x^2x^2-4y^2x^2z^3x^2y^3x^2x^2z^2x^3-4y^2x^2z^3x^2\ldots \\
& *y^3x^2x^2r^1x^2x^3+4y^2x^2z^3x^2y^3x^2x^2r^2x^3-4y^2x^3x^3x^2z^2x^3x^3x^2-4y^2x^3x^3\ldots \\
& *y^3x^3z^2x^3x^2-4y^2x^3x^3x^3y^3z^2x^3x^2+4y^2x^3x^2y^3x^2x^2z^2x^3-4y^2x^3x^3x^2y^3\ldots \\
& *x^2r^2x^2x^3-4y^2y^3x^3z^2x^2x^3x^3+4y^2y^3x^3z^2x^3x^2z^3x^2-4y^2y^3x^3z^2x^3x^2\ldots \\
& *z^3x^3-4y^2y^3x^3z^2x^3x^2r^3x^2-4y^2x^2y^3x^2z^2x^3r^1x^2x^2+4y^2x^2y^3x^2z^2\ldots \\
& *x^3x^2r^3x^2-4x^2x^2x^3x^2x^2y^2z^3x^2y^3+2x^2x^3x^2x^2y^2z^3x^3x^2-4x^2x^2x^3x^2\ldots \\
& *x^2x^2y^2r^1x^2y^3-2x^2x^3x^2x^2y^2z^3x^3x^2+4x^2x^2x^3x^2x^2y^2r^3x^2y^3-4x^2x^3\ldots \\
& *x^3x^3x^2y^2z^3y^3+4x^2x^2x^3x^3x^2y^2z^3x^2y^3-4x^2x^3x^3x^2x^3y^3y^2z^3-4x^2x^2x^3\ldots \\
& *x^2y^2r^3x^2y^3+4x^2x^3x^3x^2x^3y^2z^2x^3-4x^2x^3x^3x^3x^2y^2z^3-4x^2x^3x^3x^2x^3\ldots \\
& *y^2r^2x^2y^3+2x^2x^2z^3x^3x^2r^1x^2y^3x^2z^2-4x^2x^2z^3x^2x^3x^2r^1x^2y^3y^2-2x^2x^2z^3\ldots \\
& *x^3x^2z^2y^3x^2r^2x^2+4x^2x^2z^3x^2x^3x^2y^2r^2x^2y^3-4y^2z^3x^3z^2x^3y^3x^3x^2+2y^2z^3x^2\ldots \\
& *z^2x^4y^3x^2x^3+2y^2z^3x^4z^2x^2y^3x^3x^2-2r^1x^2y^3x^2z^2x^2z^3x^3x^2-2y^2x^2z^3\ldots \\
& *r^1x^2z^2x^3x^2r^2x^2+2r^1x^4y^3y^2z^3x^2x^2x^3+2x^2x^2x^3x^5y^2x^2+2x^2x^2x^3x^4y^2x^3\ldots \\
& *y^3+2x^2z^3x^3x^2y^2x^4z^3x^3+2x^2x^2x^3x^4y^2x^2r^2x^2-2x^2x^3x^4x^2x^2y^3x^2+2x^2x^5z^3x^2\ldots \\
& *x^3y^3x^2-2x^2x^4z^3x^2x^3x^2y^2x^2-2x^2x^4z^3x^2x^3x^2y^3x^2+2x^2x^4z^3x^2y^3x^3y^2+2x^2x^4\ldots \\
& *z^3x^2y^3x^2r^3x^2-2y^2x^2x^3x^4z^2x^2y^3x^2-2x^2x^3x^2x^2y^3x^4-2x^2x^2z^3x^2y^2x^4\ldots \\
& *x^3x^2-2x^2x^4y^3x^2y^2x^2z^3x^2+2y^2x^2z^3x^3z^2x^3x^2-2x^2x^3x^2y^2r^1x^4x^3x^2-2x^2x^3x^2y^2\ldots \\
& *z^2x^4x^3x^2-2x^2x^3x^2y^2x^2r^2x^4x^3x^2-2x^2x^3x^2y^2x^4x^3x^2z^2+2x^2z^3x^2y^2x^4x^3x^2r^2x^2-2\ldots \\
& *x^3x^4z^2x^2y^2x^2z^3x^2+2x^3x^4z^2x^2y^2x^2r^3x^2-2x^2x^2y^2x^2r^1x^4-2x^2x^2z^2x^2y^2\ldots \\
& *z^3x^4-2x^2x^2y^2x^2r^3x^4-2x^2x^3x^2x^4y^3x^2z^2+2x^2z^3x^2x^4y^3x^2r^2x^2-2x^2x^2x^2\ldots \\
& *r^1x^4y^3x^2-2x^2x^2z^2x^2y^3x^2-2x^2x^2r^2x^4y^3x^2+2y^2x^3x^4r^1x^2y^3-2y^2x^3\ldots \\
& *x^3x^4x^2x^2y^3-2y^2x^3x^3x^4r^2x^2y^3+2y^2x^4x^3x^3r^1x^2x^2-2y^2x^4x^3x^3x^2y^3x^2-2x^2x^2\ldots \\
& *x^3x^2x^2x^2y^3x^2z^3x^2+2x^2x^2x^3x^2x^2y^3x^2r^3x^2-2x^2x^2z^3x^2y^2x^3x^2z^2+2x^2x^2\ldots \\
& *z^3x^2y^2x^2x^3x^2r^2x^2+2x^2x^2y^3x^2y^2x^2z^3x^2r^2x^2+2y^2x^2z^3x^3z^2x^3x^2r^1x^2-2y^2x^2\ldots \\
& *z^3x^3z^2x^3x^2r^2x^2+2x^2x^3x^3x^2y^2x^2z^3r^1x^2-2x^2x^3x^3x^2y^2x^3r^3x^2+2x^2x^2z^3x^3\ldots \\
& *r^1x^2y^3x^2z^2-2x^2x^2z^3x^3z^2y^3x^2r^2x^2+2r^1x^4y^3x^2z^2x^2z^3+2y^2x^2z^3r^1x^4z^2\ldots \\
& *x^3x^2+2x^2x^2z^3y^3x^4r^1x^2z^2+2x^2x^2z^3x^2y^3x^3r^1x^2y^2-2x^2x^2z^3y^3x^4z^2r^2x^2-2\ldots \\
& *x^2x^2z^3x^2y^3x^3y^2r^2x^2+2x^2x^3x^3x^2y^3x^2r^1x^2x^3-2x^2x^3x^3x^2y^3x^2r^2x^3-2y^2x^2\ldots \\
& *z^3x^2z^2x^2y^3x^2x^2-2y^2x^2x^3x^2z^2x^3x^2+2y^2x^2x^3x^2z^2x^3x^2r^3x^2-4y^2\ldots \\
& *z^3x^2z^2x^2y^3x^3x^2r^3x^2-4y^2z^3x^3z^2y^3x^2r^1x^2x^3+4y^2z^3x^3z^2y^3x^2r^2x^3-4\ldots
\end{aligned}$

$$\begin{aligned}
& *r1^4*y3*y2*z3^2*x3*x2-4*r1^2*y3*y2*z3^2*x2*r2^2*x3-4*y2*r1^2*z2^2*y3*x3*x2*r3^2... \\
& +4*r1^2*y3*y2*z3^2*x3*x2*r3^2+4*y2*r1^2*z2^2*y3*x2*z3^2*r2^2*x3+2*z2^2*x3^2*r2^2*y2^2... \\
& *z3^2*r3^2+2*y2^2*z3^2*r2^4*y3*x2*x3+2*y2^2*r3^4*z2^2*y3*x3*x2+4*y2^2*x3*x2*y3^2*r1^2... \\
& *r3^2+4*y2^2*x3*x2*y3^2*r1^2*r2^2-4*y2^2*x3*x2*y3^2*r3^2*r2^2+4*y2^2*x3^2*x2^2*y3... \\
& *r1^2*r2^2+4*y2^2*x3^2*x2^2*y3*r1^2*r3^2-4*y2^2*x3^2*x2^2*y3*r2^2*r3^2-4*y2^2*x3^2*x2^2... \\
& *y3^2*z3^2*r3^2-4*y2^2*x3^2*x2^2*y3^3*r1^2*r2^2-4*y2^2*x3^2*x2^2*y3^3*r1^2*r3^2-4*y2^2*x3^2*x2^2*y3^3... \\
& *z2^2*r2^2-4*y2^2*x3^2*x2^2*y3^3*r1^2*r3^2-4*y2^2*x3^2*x2^2*y3^3*r1^2*r2^2-4*y2^2*x3^2*x2^2*y3^3... \\
& *z2^2*r2^2-4*y2^2*x3^2*x2^2*y3^3*z3^2*r3^2-4*z3^2*y2^2*r1^2*x3^2*z2^2+2*z3^2*y2^2*r1^2... \\
& *x3^2*r2^2+2*z3^2*y2^2*z2^2*x3^2*r2^2+2*x3^2*z2^2*y2^2*z3^2*r3^2+2*x3^2*z2^2*y2^2... \\
& *r1^2*r3^2-4*z3^2*x2^2*r1^2*y3^2*z2^2+2*z3^2*x2^2*r1^2*y3^2*r2^2+2*z3^2*x2^2*z2^2... \\
& *y3^2*r2^2-2*y2^2*x3^2*r1^2*y3^2*r3^2-2*y2^2*x3^2*x2^2*y3^2*r1^2+2*y2^2*x3^2*x2^2*y3^2... \\
& *r3^2-2*y2^2*x3^2*r1^2*r2^2*y3+2*y2^2*x3^2*r3^2*r2^2*y3-2*y2^2*x3^2*r1^2*x2^2*r2^2... \\
& -2*y2^2*x3^2*r1^2*x2^2*y3^2-2*y2^2*x3^2*r1^2*x2^2*r3^2+2*y2^2*x3^2*r2^2*x2^2*y3^2+2... \\
& *y2^2*x3^2*r2^2*x2^2*r3^2+2*y2^2*x3^2*r1^2*y3^2*r2^2+4*y2^2*x3^2*x2^2*y3^2*r2^2+2... \\
& *y2^2*x3^2*r1^2*x2^2*r3^2+4*y2^2*x3^2*x2^2*y3^2*r3^2-8*y2^2*x3^2*r1^2*x2^2*y3-2*x2^2... \\
& *y3^2*r1^2*y2^2*x3^2-2*x2^2*y3^2*r1^2*y2^2*r3^2-2*x2^2*y3^2*y2^2*r1^2*r2^2+2*x2^2*y3^2... \\
& *y2^2*x3^2*r2^2+2*x2^2*y3^2*y2^2*r3^2*r2^2-2*x2^2*y3^2*r1^2*y2^2*x3-2*x2^2*y3^2*r1^2... \\
& *r2^2*x3-2*x2^2*y3^2*r1^2*x3^2*r3^2+2*x2^2*y3^2*y2^2*x3^2*r3^2+2*x2^2*y3^2*r2^2*x3^2*r3^2... \\
& +2*x2^2*y3^2*y2^2*r1^2*r3^2-4*y2^2*z3^2*r2^2*y3^2*z2^2*x3*x2^2*r3^2-4*x2^4*y3^2*r1^2*x3^2+2... \\
& *x2^4*y3^2*r1^2*r3^2+2*x2^2*y3^2*r1^2*x3^2+2*x2^4*y3^2*x3^2*r2^2-2*x2^5*y3^2*x3... \\
& *r3^2-2*x2^3*y3^2*r2^2*x3^2+2*x2^4*y3^2*x3^2*r3^2-y3^2*z2^2*r1^4*x2^2-y3^2*z2^2... \\
& *x2^2*z3^4-y3^2*z2^2*x2^2*r3^4-2*y3^4*z2^2*x2^2*z3^2+2*y3^4*z2^2*x2^2*r3^2+4*y2^2... \\
& *x3^2*x2^3*y3^2+4*y2^2*x3^2*x2^2*y3^2+2*y2^2*x3^2*x2^5*y3^2+2*y2^2*x3^2*x2^5*y3+2*y2^2*x3... \\
& *x2^2*y3^5-4*y2^4*x3^2*x2^2*y3^4+2*y2^5*x3^2*x2^2*y3^3+2*y2^2*x3^2*x2^5*y3-4*y2^2*x3^4*x2^4*y3... \\
& +2*y2^5*x3^2*x2^2*y3+2*y2^2*x3^5*x2^3*y3+2*y2^2*x3^2*x2^3*y3^5+4*y2^2*x3^2*x2^3*y3+4*y2... \\
& *x3^2*x2^3*y3^2-2*y2^4*x3^2*x2^2*r3^2-2*y2^5*x3^2*r3^2*y3+2*y2^4*x3^2*y3^2*r2^2+2... \\
& *y2^2*x3^2*r1^2*y3^2+2*y2^2*x3^2*r1^4*y3-4*y2^4*x3^2*r1^2*y3^2-3*y2^4*x3^2*x2^2... \\
& *y3^2+2*y2^4*x3^2*r1^2*r3^2+2*y2^5*x3^2*r1^2*y3+2*y2^4*x3^2*y3^2*r3^2-2*y2^2*x3^2... \\
& *y3^2*r2^2-y2^2*x3^2*r1^4*y3^2-3*y2^2*x3^2*x2^4*y3^2-y2^2*x3^2*r2^4*y3^2-y2^2*x3^2... \\
& *r1^4*x2^2-3*y2^2*x3^2*x2^2*y3^4-y2^2*x3^2*x2^2*r3^4+2*y2^2*x3^2*r1^4*x2^2+y2^2... \\
& *x3^2*r1^2*x2^3-4*y2^2*x3^4*r1^2*x2^2+2*y2^2*x3^4*r1^2*r2^2+2*y2^2*x3^5*r1^2*x2+2... \\
& *y2^2*x3^4*x2^2*r2^2-2*y2^2*x3^2*x2^3*r3^2-2*y2^2*x3^5*r2^2*x2-3*y2^2*x3^4*x2^2... \\
& *y3^2+2*y2^2*x3^4*x2^2*r3^2+2*x2^4*y3^2*y2^2*r1^2-2*x2^4*y3^2*y2^2*x3^2-2*x2^4*y3^2... \\
& *y2^2*r3^2+2*x2^3*y3^4*r1^2*x3-2*x2^3*y3^4*y2^2*x3-2*x2^3*y3^4*r2^2*x3-2*x2^2*y3^2... \\
& *y2^2*r3^2+2*x2^2*y3^4*y2^2*r2^2+2*x2^2*y3^5*r1^2*y2+2*x2^2*y3^2*r1^4*y2-4*x2^2... \\
& *y3^4*r1^2*y2^2+2*x2^2*y3^4*r1^2*r2^2+2*x2^2*y3^2*y2^2*r1^2+2*x2^2*y3^4*y2^2*r3^2... \\
& -2*x2^2*y3^5*y2^2*r2^2-x2^2*y3^2*y2^2*r1^4-x2^2*y3^2*y2^2*r3^4-x2^2*y3^2*r1^4*x3^2... \\
& -x2^2*y3^2*r2^4*x3^2+2*x2^3*y3^2*r1^4*x3+2*x2^5*y3^2*r1^2*x3+2*x2^2*y3^2*r1^2*x3^2... \\
& *r2^2-8*x2^3*y3^2*r1^2*y2^2*x3+2*y3^2*z2^2*r1^2*x2^2*r3^2+2*y3^2*z2^2*x2^2*z3^2*r3^2... \\
& +4*y2^4*x3^2*y3^2*r3^2+4*y2^2*x3^2*y3^2*z2^2-4*y2^2*x3^2*y3^2*r2^2-4*y2^2*x3^2... \\
& *y3^4*r1^2-4*y2^2*x3^2*y3^2*r1^4+8*y2^2*x3^2*y3^2*r1^2+4*y2^2*x3^2*y3^2*z2^2-4... \\
& *y2^2*x3^2*x2^3*y3^2*r2^2-4*y2^4*x3^2*x2^2*y3^2*r1^2+4*y2^2*x3^2*x2^2*y3^2*z3^2-4*y2^2*x3^2*x2^2... \\
& *y3^2*r3^2+4*y2^2*x3^2*x2^2*y3^2*z3^2-4*y2^2*x3^2*x2^2*y3^2*r3^2+4*y2^2*x3^2*x2^2*y3^4*r2^2+2*y2... \\
& *x3^2*y3^2*r1^4+2*y2^2*x3^2*x2^2*y3^2*r1^4+2*y2^2*x3^2*x2^2*y3^2*z3^4+2*y2^2*x3^2*x2^2*y3^2*r3^4+2... \\
& *y2^2*x3^2*x2^2*y3^2*z2^4+2*y2^2*x3^2*x2^2*y3^2*r2^4+2*y2^2*x3^2*x2^2*y3^2*r1^4+2*y2^2*x3^2*x2^2*y3^2*r1^4... \\
& +2*y2^2*x3^2*x2^2*y3^2*z2^4+2*y2^2*x3^2*x2^2*y3^2*r2^4+2*y2^2*x3^2*x2^2*y3^2*z3^4+2*y2^2*x3^2*x2^2*y3^2*r3^4... \\
& -4*y2^2*x3^2*x2^2*y3^2*r1^4-4*y2^2*x3^2*x2^4*y3^2*r1^2+8*y2^2*x3^2*x2^2*y3^2*r1^2-4*y2^2*x3^4*x2^2...
\end{aligned}$$

```

*y3*r1^2+4*y2*x3^3*x2^3*y3*z2^2-4*y2*x3^3*x2^3*y3*r2^2+4*y2*x3^2*x2^4*y3*r3^2+4*y2^3...
*x3^3*x2*y3*z2^2-4*y2^3*x3^3*x2*y3*r2^2+4*y2*x3^4*x2^2*y3*r2^2+4*y2*x3^3*x2^3*y3*z3^2...
-4*y2*x3^3*x2^3*y3*r3^2+4*y2*x3*x2^3*y3^3*z3^2-4*y2*x3*x2^3*y3^3*r3^2+16*y2^2*x3^2...
*x2^2*y3^2*r1^2);

b=(-z2^3*y3^2-x2^2*y3^2*z2-y2^2*z3*x3^2-y2^2*z3*y3^2+y2^3*z3*y3+y2*y3^3*z2-y2^2*y3^2...
*z2-z2*x3^2*x2^2-z2*x3^2*y2^2+z2*x3^3*x2+x2^3*z3*x3-x2^2*z3*x3^2-x2^2*z3*y3^2+y2*z3...
*z2^2*y3+y2*x3^2*z2*y3+y2*z3^2*z2*y3+z2*x3*x2*y3^2+z2*x3*x2*z3^2+x2*z3*y2^2*x3+x2*z3...
*z2^2*x3+x2^2*y3*y2*z3-y2^2*z3^3-z2^3*x3^2-x2^2*z3^3-r1^2*y3^2*z2-y2^2*z3*r1^2+r1^2...
*y3*y2*z3+y2*r1^2*z2*y3+z2*y3^2*r2^2-z2*x3^2*r1^2+z2*x3^2*r2^2-x2^2*z3*r1^2+x2^2*z3...
*r3^2+y2^2*z3*r3^2-y2*z3*r2^2*y3-y2*r3^2*z2*y3+z2*x3*r1^2*x2-z2*x3*x2*r3^2+x2*z3*r1^2...
*x3-x2*z3*r2^2*x3);

c=(-2*y2*z3*z2*y3-
2*z2*x3*x2*z3+z3^2*y2^2+x3^2*z2^2+z3^2*x2^2+y2^2*x3^2+x2^2*y3^2+y3^2...
*z2^2-2*y2*x3*x2*y3);
%% *****
%% evaluate feasibility of given coordinates
if(a<0||c==0), result=[nan;nan;nan]; return; end % coz c is the denominator & a is under a root
za=-1/2*(b-a^(1/2))/c;
zb=-1/2*(b+a^(1/2))/c;
if(za>zb)
    if(pos), z=za; else z=zb; end
else
    if(pos), z=zb; else z=za; end
end
a=(2*z*z2*x3-2*x2*z3+r1^2*x2-r1^2*x3-x2^2*x3-y2^2*x3-
z2^2*x3+r2^2*x3+x2*x3^2+x2*y3^2+x2*z3^2-x2*r3^2);
b=(-2*y2*x3+2*x2*y3);
if(b==0), result=[nan;nan;nan]; return; end % coz b is the denominator in the expression
y=a/b;
if(x2==0), result=[nan;nan;nan]; return; end
x = 1/2*(r1^2+x2^2-2*y*y2+y2^2-2*z*z2+z2^2-r2^2)/x2;
%% *****
%% convert result back to global
result=[x1;y1;z1;1]+[x;y;z;0];
% disp([x1 y1 z1 x2+x1 y2+y1 z2+z1 x3+x1 y3+y1 z3+z1]);
% disp('Solution'); disp(result);
% disp('Constraints'); % check distances to three centers
% disp((result(1)-X1(1))^2+(result(2)-X1(2))^2+(result(3)-X1(3))^2-r1^2);
% disp((result(1)-X2(1))^2+(result(2)-X2(2))^2+(result(3)-X2(3))^2-r2^2);
% disp((result(1)-X3(1))^2+(result(2)-X3(2))^2+(result(3)-X3(3))^2-r3^2);
%% *****

```

C.2 Coordinate Mapping Program

```
%% Function to get the end effector position from the angles

function position = getposition(theta);
% theta is a column vector with the homed theta positions
% theta(1) is the angle from the bottom link to the horizontal axis for link 1
% theta(2) is the angle from the bottom link to the horizontal axis for link 2
% theta(3) is the angle from the bottom link to the horizontal axis for link 3
% theta(4) is the initial x-axis reference position
% theta(5) is the initial y-axis reference position
% theta(6) is the initial z-axis reference position
% *****
r1 = 37;    % radius of the base circle in mm
r2 = 60;    % radius of the base arm in mm
r3 = 145;   % radius of the top arm in mm (125mm original + 20mm additional length)
xref = theta(4); yref = theta(5); zref = theta(6)
load initpos
xo = t(1); yo = t(2); zo = t(3);
clear t
% *****
%% Base arm positions p1, p2, p3 : in [x;y;z]' format
p1 = [(r1 + r2*cos(theta(1)))*cos(0*pi/180); (r1 + r2*cos(theta(1)))*sin(0*pi/180); r2*sin(theta(1))];
p2 = [(r1 + r2*cos(theta(2)))*cos(120*pi/180); (r1 + r2*cos(theta(2)))*sin(120*pi/180); r2*sin(theta(2))];
p3 = [(r1 + r2*cos(theta(3)))*cos(240*pi/180); (r1 + r2*cos(theta(3)))*sin(240*pi/180); r2*sin(theta(3))];
% *****
%% Use interx to get the position of the tip
t = interx(p1,p2,p3,r3,r3,r3,1);    % radii same for all three arms, "1" is to get the 'top' position
ex = xref-t(1)+xo;
ey = yref-t(2)+yo;
ez = zref-t(3)+zo;
position = [ex;ey;ez];
% *****
```


Appendix D

Derivation of Simple System Dynamics for Single Leg Link from Chapter 7

This section presents the derivation for coefficients α and β in the following equation, originally introduced as (7.13) in Chapter 7 Subsection 7.3.2.

$$\tau_{i,m} = \alpha \ddot{\theta}_{1,i,m} + c_d \dot{\theta}_{1,i,m} - \beta \cos \theta_{1,i,m}.$$

Based on the simple leg model presented in Figure D.1, Newton's method or Lagrangian Dynamics can be applied to determine the equation of motion for the simple model. For this example, the derivation is performed using Lagrangian dynamics.

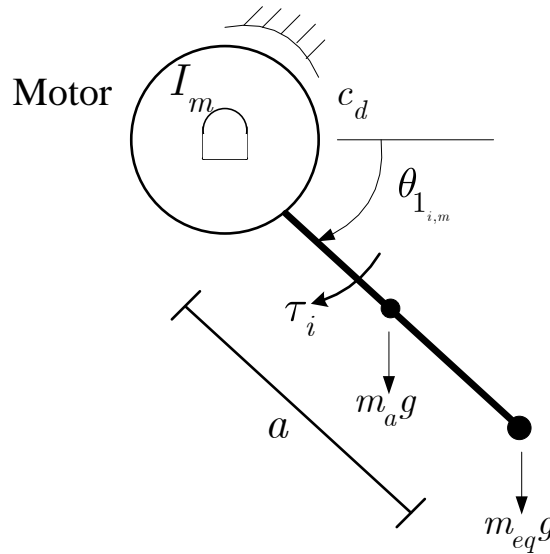


Figure D.1: Single link dynamic model. This image is used to determine a simple, linearized model for each individual leg link.

Consider the well-known Lagrangian equation (ref?),

$$L = T - V, \quad (D.1)$$

where T is the kinetic energy and V contains the potential energy components of the system. For the simple system illustrated in Figure D.1, the kinetic and potential energy were found to be,

$$T = \frac{1}{2}I_m\dot{\theta}^2 + \frac{1}{2}m_a \cdot \left(\frac{a}{2}\right)^2\dot{\theta}^2 + \frac{1}{2}m_{eq}a^2\dot{\theta}^2, \quad (D.2)$$

$$V = m_a \cdot \frac{a}{2} \cdot g \cdot \sin\theta + m_{eq} \cdot a \cdot g \cdot \sin\theta. \quad (D.3)$$

The equation of motion is found by applying the following derivation,

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} + \frac{\partial L}{\partial \theta} = \tau - c_d \cdot \dot{\theta}. \quad (D.4)$$

In D.4, τ is the torque applied to the motor and $c_d \cdot \dot{\theta}$ is the viscous damping friction applied to the system. Note that the variables on the right-hand-side of the equation are the nonconservative forces in the system. Applying the derivative action from (D.4) to the kinetic and potential energy relationships defined in equations (D.2) and (D.3), the following equation of motion for the simple system can be determined.

$$(I_m + \frac{1}{4}m_a a^2 + m_{eq}a^2)\ddot{\theta} - (\frac{1}{2}m_a \cdot g \cdot a + m_{eq} \cdot g \cdot a)\cos\theta = \tau - c_d\dot{\theta} \quad (D.5)$$

$$\alpha\ddot{\theta} + c_d\dot{\theta} - \beta\cos\theta = \tau. \quad (D.6)$$

Replacing the combined mass $m_{eq} = 2m_b + \frac{1}{3}m_c$ results in the definition of α and β

presented in (D.7) and (D.8).

$$\alpha = I_m + \frac{1}{3}m_a a^2 + (2m_b + \frac{1}{3}m_c)a^2 \quad (\text{D.7})$$

$$\beta = a \cdot g \cdot (\frac{1}{2}m_a + 2m_b + \frac{1}{3}m_c). \quad (\text{D.8})$$

Appendix E

Bode Diagrams for Systems 2 and 3 for System ID from Chapter 7

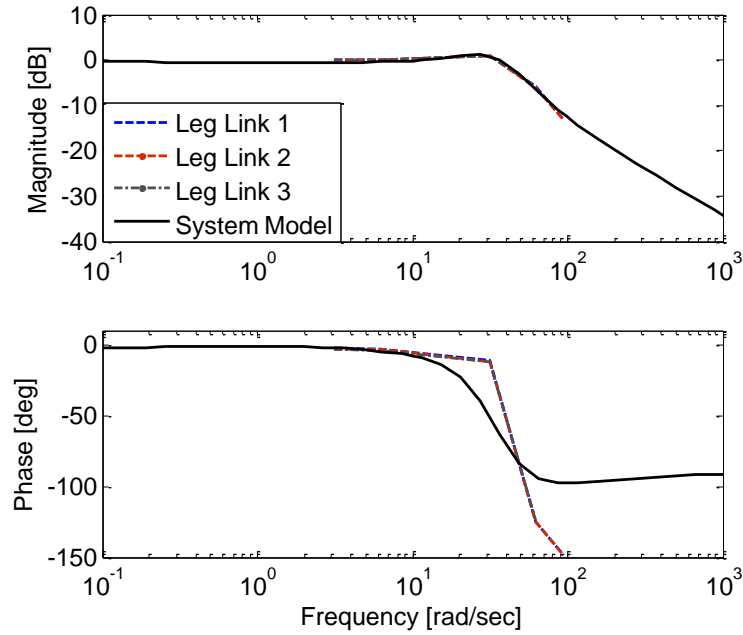


Figure E.1: Bode diagram of the experimental data and the identified system model for the dynamic relationships between $\theta_{1,i,2}$ and the input torque to motors 1-3. The dynamic similarities within the systems enables identical system models to be used for each leg link.

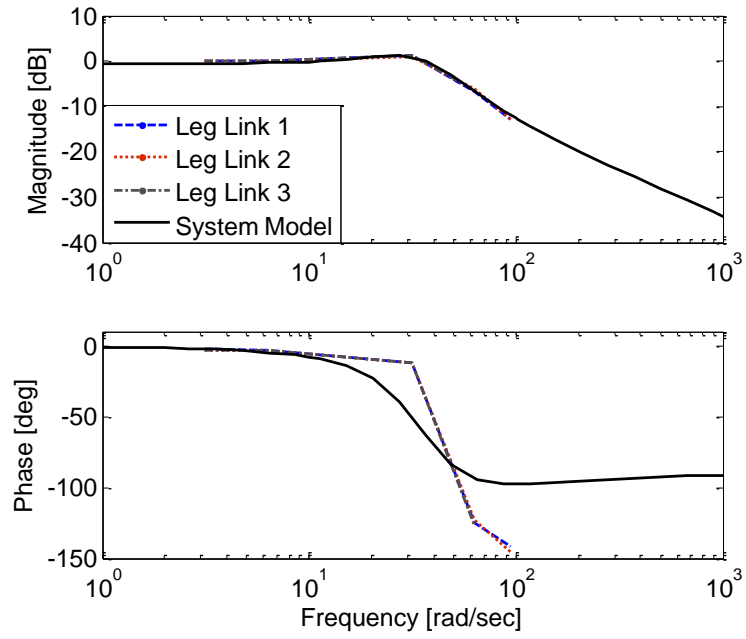


Figure E.2: Bode diagram of the experimental data and the identified system model for the dynamic relationships between $\theta_{1,i,3}$ and the input torque to motors 1-3. The dynamic similarities within the systems enables identical system models to be used for each leg link.

Appendix F

MATLAB Code to Build Jacobian

This section provides the code for building the Jacobian matrix which provides a simplified method for mapping back and forth between the angles $\{\theta_{11,m}, \theta_{12,m}, \theta_{13,m}\}$ and the end effector position. There are a few assumptions that are made when using this approach.

- A1) The reference operation is bounded within a specified work area.
- A2) The mapping is time- and angle-invariant.
- A3) The system, and therefore the mapping, is linear within the work area.

Applying these assumptions (originally stated in Subsection 7.2.3) and utilizing the code presented on the next page, the calculated Jacobian matrix was found to be,

$$J = \begin{bmatrix} -44.71 & 22.36 & 22.36 \\ 0 & -38.72 & 38.72 \\ 14.56 & 14.56 & 14.56 \end{bmatrix}.$$

Note that the Jacobian must take into account the parameters of the desired trajectories since the matrix should be suitable for the desired work area and angle changes required by the reference trajectories.

```
%% Program for finding localized Jacobian from theta angles to cartesian coordinates over a given work area.
% Jacobian assumed to be time and iteration invariant.
```

```
function J = getjacobian(theta, delta_theta);
% theta is a column vector with the homed theta positions
% delta_{theta} is a scalar vector describing the change in angle
% *****
%% Initialize the lengths of the leg links and the Jacobian matrix
r1 = 37; % radius of the base circle in mm
r2 = 60; % radius of the base arm in mm
r3 = 145; % radius of the top arm in mm (125mm original + 20mm additional length)

J = zeros(3,3);
% *****
%% Base arm positions p1, p2, p3 : in [x;y;z]' format
p1 = [(r1 + r2*cos(theta(1)))*cos(0*pi/180); (r1 + r2*cos(theta(1)))*sin(0*pi/180); r2*sin(theta(1))];
p2 = [(r1 + r2*cos(theta(2)))*cos(120*pi/180); (r1 + r2*cos(theta(2)))*sin(120*pi/180); r2*sin(theta(2))];
p3 = [(r1 + r2*cos(theta(3)))*cos(240*pi/180); (r1 + r2*cos(theta(3)))*sin(240*pi/180); r2*sin(theta(3))];
% *****
%% Use interx to get the position of the tip
t = interx(p1,p2,p3,r3,r3,r3,1); % radii same for all three arms, "1" is to get the 'top' position
% *****
%% Perturb with delta_t and obtain first column of Jacobian
theta_p1 = theta + delta_theta*[1;0;0];

p11 = [(r1 + r2*cos(theta_p1(1)))*cos(0*pi/180); (r1 + r2*cos(theta_p1(1)))*sin(0*pi/180); r2*sin(theta_p1(1))];
p21 = [(r1 + r2*cos(theta_p1(2)))*cos(120*pi/180); (r1 + r2*cos(theta_p1(2)))*sin(120*pi/180); r2*sin(theta_p1(2))];
p31 = [(r1 + r2*cos(theta_p1(3)))*cos(240*pi/180); (r1 + r2*cos(theta_p1(3)))*sin(240*pi/180); r2*sin(theta_p1(3))];

t1 = interx(p11,p21,p31,r3,r3,r3,1); % radii same for all three arms, "1" is to get the 'top' position

J(:,1) = (t1(1:3) - t(1:3))/delta_theta; % Compute first column of Jacobian
% *****
%% Perturb with delta_t and obtain second column of Jacobian
theta_p2 = theta + delta_theta*[0;1;0];

p12 = [(r1 + r2*cos(theta_p2(1)))*cos(0*pi/180); (r1 + r2*cos(theta_p2(1)))*sin(0*pi/180); r2*sin(theta_p2(1))];
p22 = [(r1 + r2*cos(theta_p2(2)))*cos(120*pi/180); (r1 + r2*cos(theta_p2(2)))*sin(120*pi/180); r2*sin(theta_p2(2))];
p32 = [(r1 + r2*cos(theta_p2(3)))*cos(240*pi/180); (r1 + r2*cos(theta_p2(3)))*sin(240*pi/180); r2*sin(theta_p2(3))];

t2 = interx(p12,p22,p32,r3,r3,r3,1); % radii same for all three arms, "1" is to get the 'top' position

J(:,2) = (t2(1:3) - t(1:3))/delta_theta; % Compute second column of Jacobian
% *****
%% Perturb with delta_t and obtain third column of Jacobian
theta_p3 = theta + delta_theta*[0;0;1];

p13 = [(r1 + r2*cos(theta_p3(1)))*cos(0*pi/180); (r1 + r2*cos(theta_p3(1)))*sin(0*pi/180); r2*sin(theta_p3(1))];
p23 = [(r1 + r2*cos(theta_p3(2)))*cos(120*pi/180); (r1 + r2*cos(theta_p3(2)))*sin(120*pi/180); r2*sin(theta_p3(2))];
p33 = [(r1 + r2*cos(theta_p3(3)))*cos(240*pi/180); (r1 + r2*cos(theta_p3(3)))*sin(240*pi/180); r2*sin(theta_p3(3))];

t3 = interx(p13,p23,p33,r3,r3,r3,1); % radii same for all three arms, "1" is to get the 'top' position

J(:,3) = (t3(1:3) - t(1:3))/delta_theta; % Compute third column of Jacobian

save Jacobian J delta_theta theta
% *****
```

Appendix G

Experimental Results for the PKM setup from Chapter 7

This appendix presents the supplemental plots for the experimental results presented in Chapter 8.

G.1 Raster Trajectory

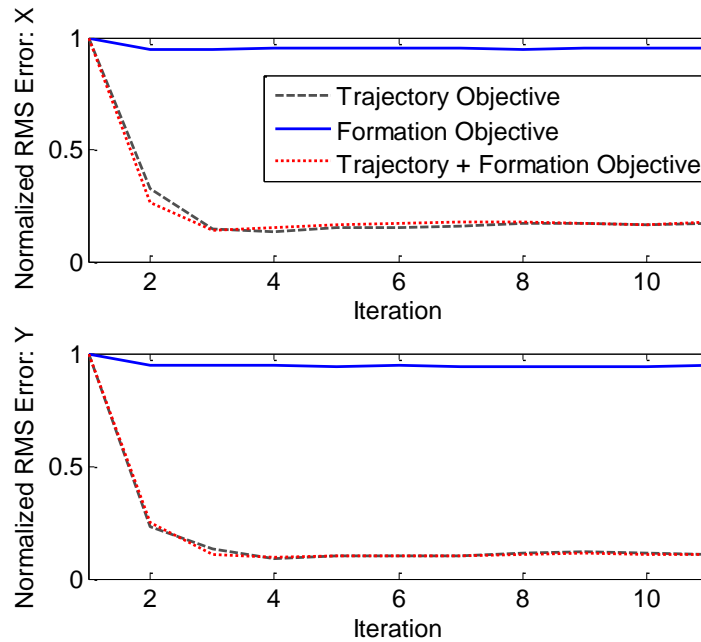


Figure G.1: Normalized x - and y -axis position tracking RMS errors for system 2. Modifying the controller for equal weighting on the two objectives results in nearly equivalent performance capabilities as compared to trajectory tracking.

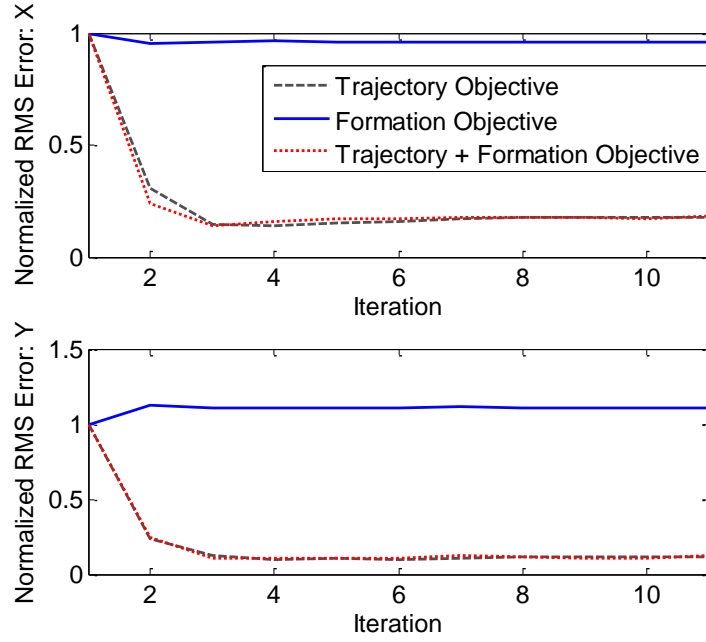


Figure G.2: Normalized x - and y -axis position tracking RMS errors for system 3. Modifying the controller for equal weighting on the two objectives results in nearly equivalent performance capabilities as compared to trajectory tracking.

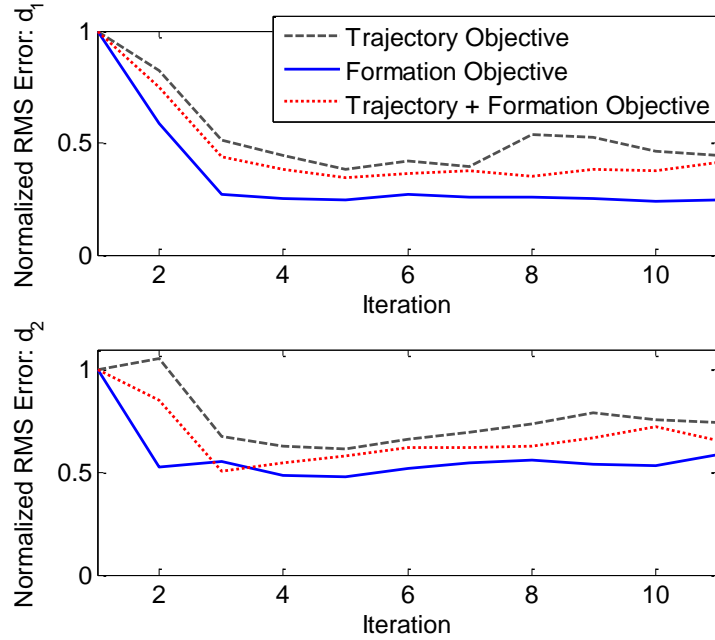


Figure G.3: Normalized E_{d1} and E_{d2} formation tracking RMS errors. Modifying the controller for equal weighting on the two objectives improves the horizontal formation tracking over the trajectory tracking objective, but only minimally for this trajectory and these systems.

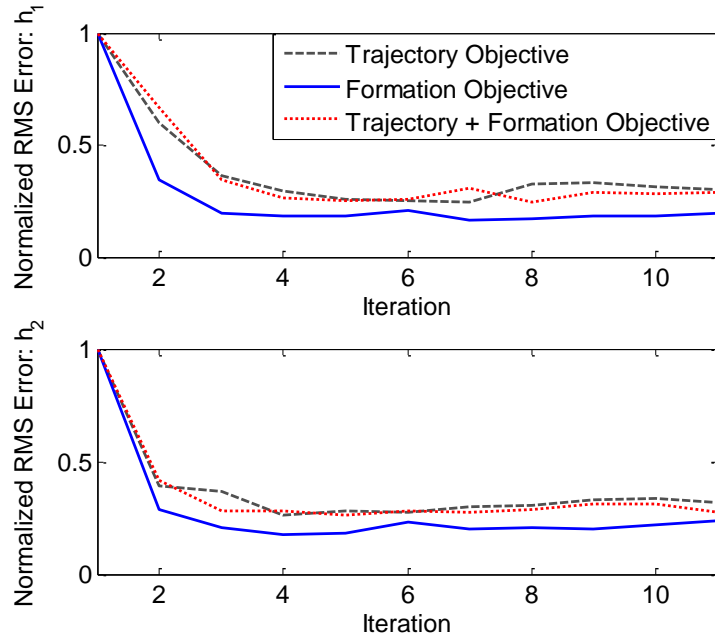


Figure G.4: Normalized E_{h1} and E_{h2} formation tracking RMS errors. Note that focusing on formation tracking results in the lowest converged RMS error. Modifying the controller for equal weighting on the two objectives has a very minimal impact on the vertical formation errors for this trajectory and these particular PKM systems.

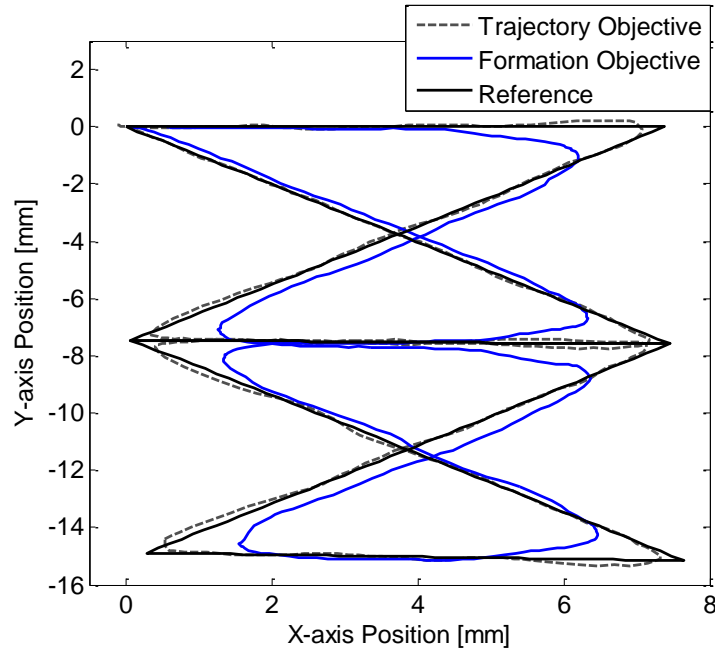


Figure G.5: XY position tracking for agent 2. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.

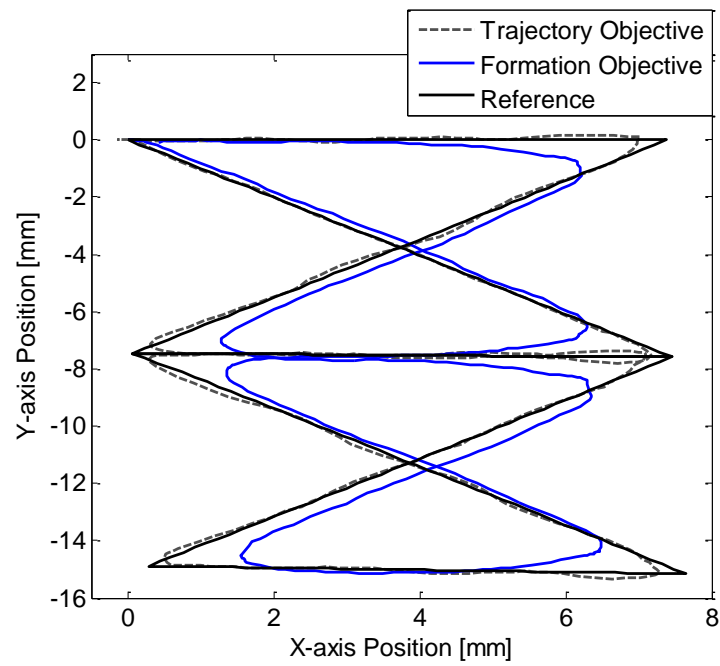


Figure G.6: XY position tracking for agent 3. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.

G.2 Spiral Trajectory

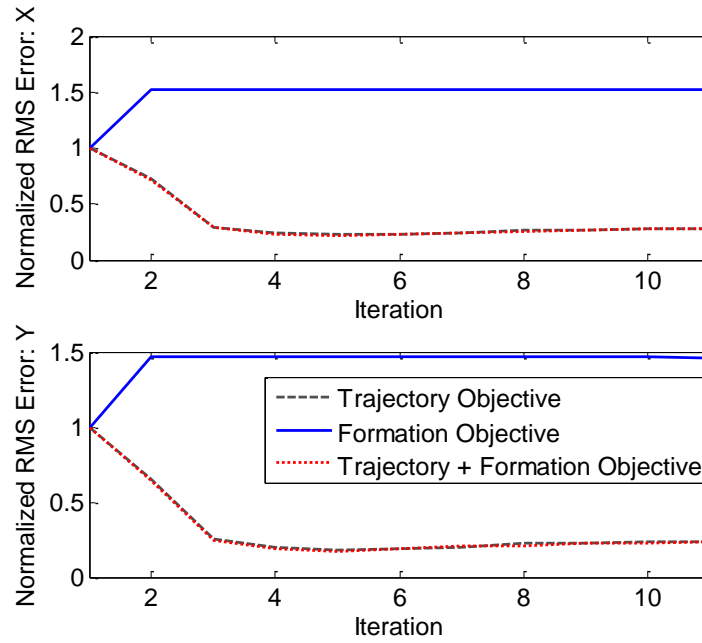


Figure G.7: Normalized x - and y -axis position tracking RMS errors for system 2.

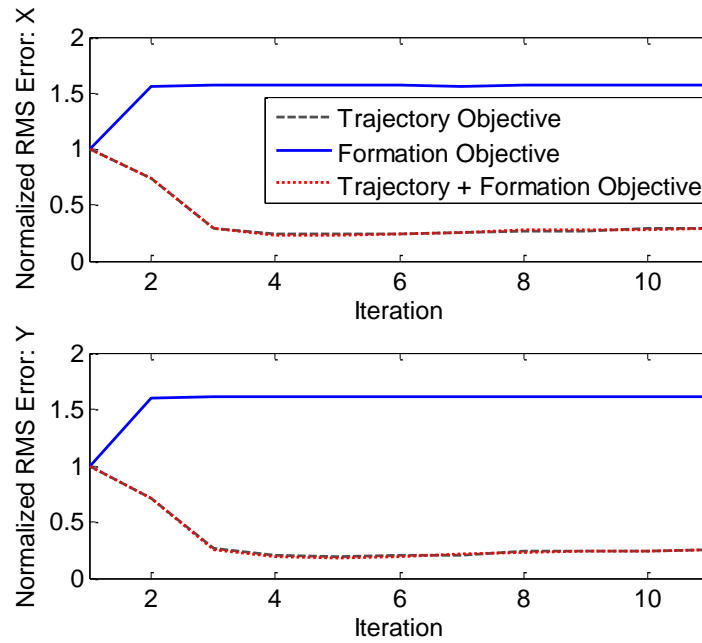


Figure G.8: Normalized x - and y -axis position tracking RMS errors for system 3.

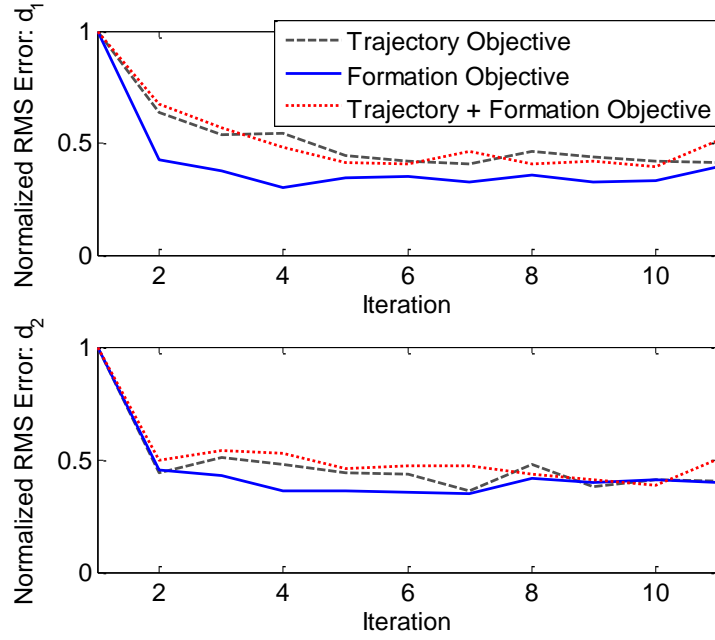


Figure G.9: Normalized E_{d1} and E_{d2} formation tracking RMS errors. Modifying the controller for equal weighting on the two objectives does not significantly modify the horizontal formation tracking over the trajectory tracking objective in this case.

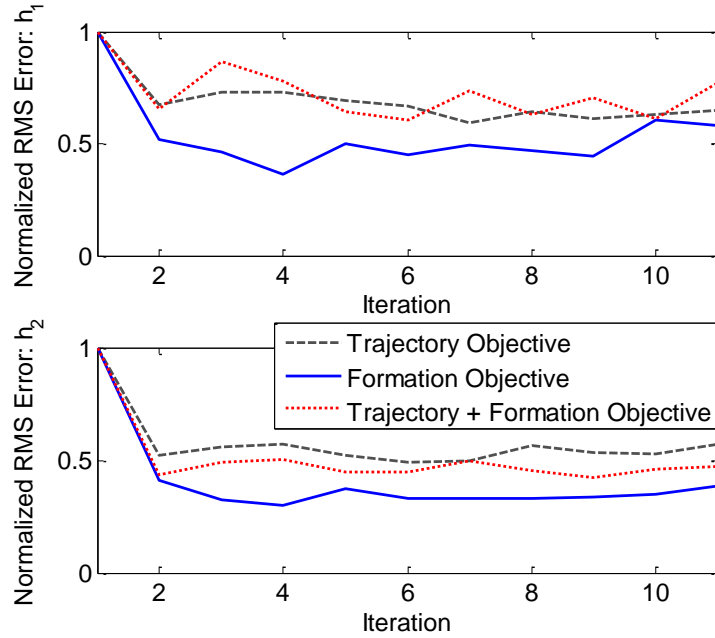


Figure G.10: Normalized E_{h1} and E_{h2} formation tracking RMS errors. Note that focusing on formation tracking results in the lowest converged RMS error. Modifying the controller for equal weighting on the two objectives has a very minimal impact on the vertical formation errors for this case.

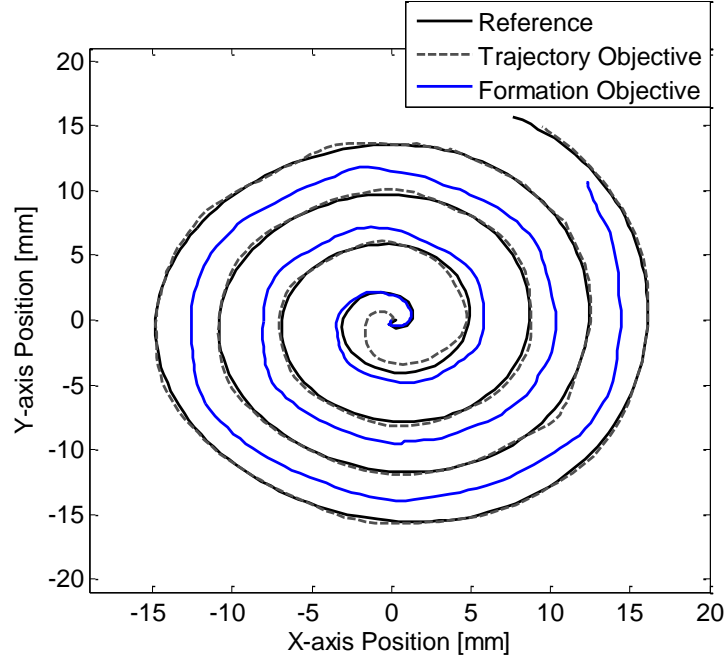


Figure G.11: XY position tracking for agent 2. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.

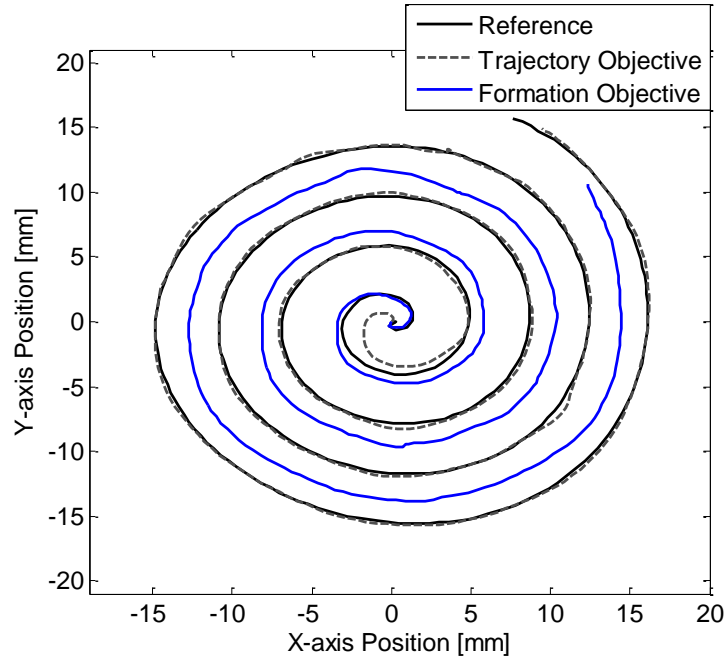


Figure G.12: XY position tracking for agent 3. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.

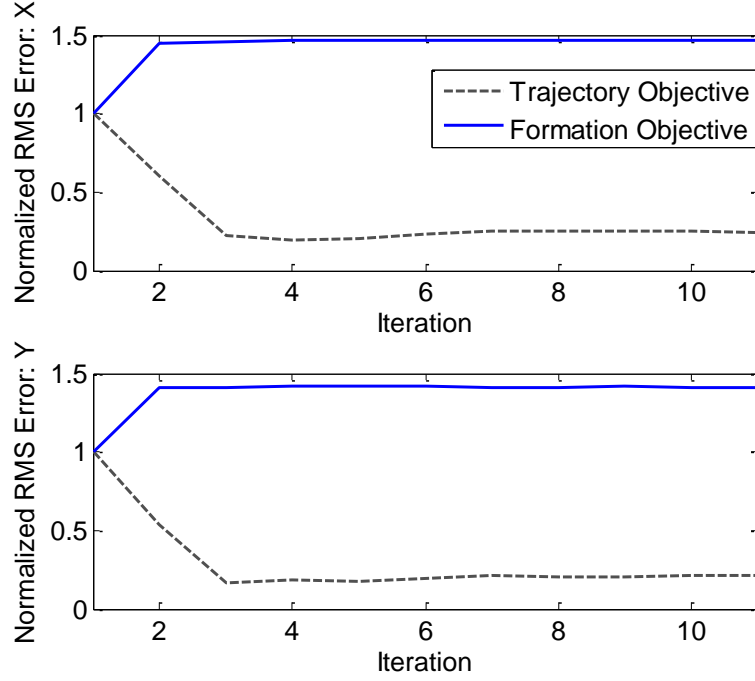


Figure G.13: Normalized x - and y -axis position tracking RMS errors for agent 2 when agent 1 is subject to model uncertainty and disturbances.

G.3 Spiral Trajectory with Agent 1 Subject to Model Uncertainty and Force Disturbance

This section presents experimental results for the 3 PKM system presented in Chapter 8. In these results, agent 1 has been perturbed through a modification to the end effector and the addition of an elastic band. These changes introduced model uncertainty and a force disturbance to agent 1.

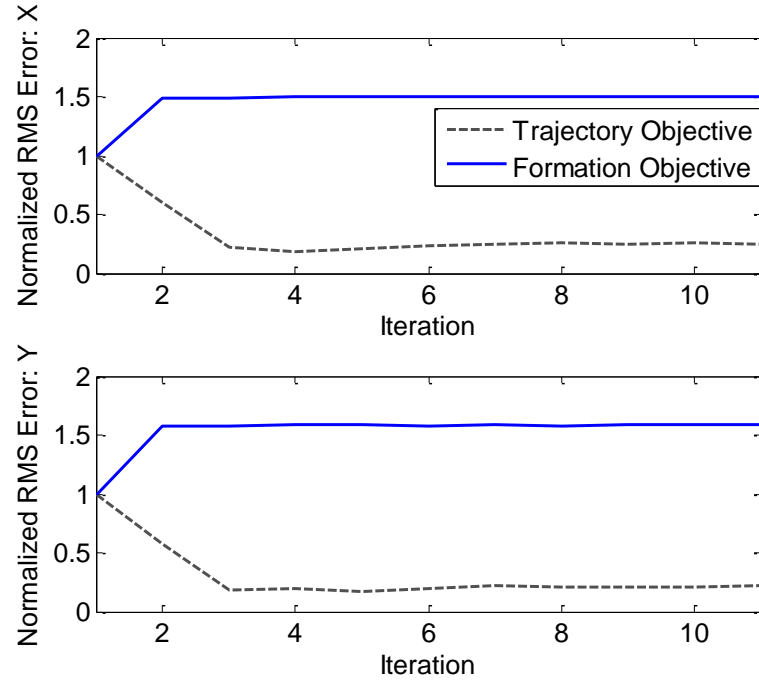


Figure G.14: Normalized x - and y -axis position tracking RMS errors for agent 3 when agent 1 is subject to model uncertainty and disturbances.

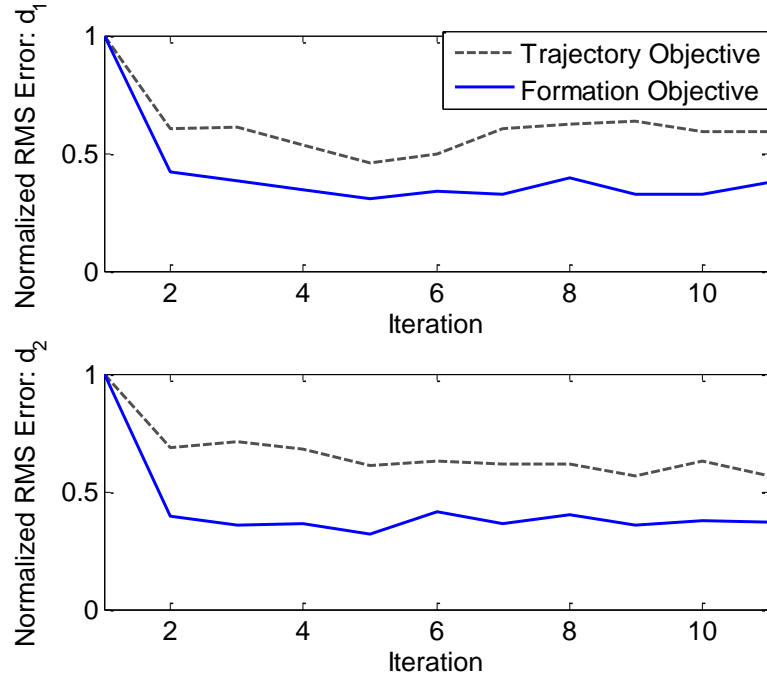


Figure G.15: Normalized E_{d1} and E_{d2} formation tracking RMS errors when agent 1 is subject to model uncertainty and force disturbance.

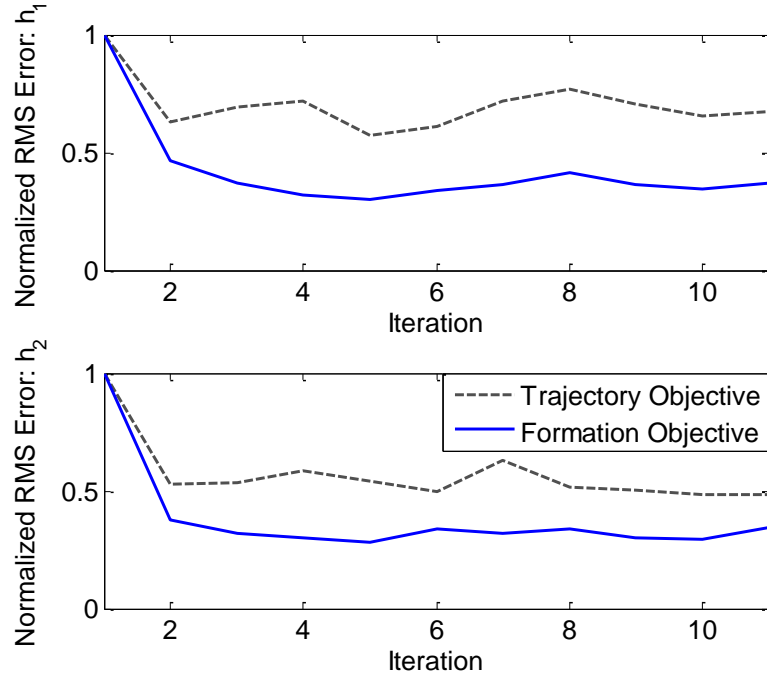


Figure G.16: Normalized E_{h1} and E_{h2} formation tracking RMS errors when agent 1 is subject to model uncertainty and force disturbance.

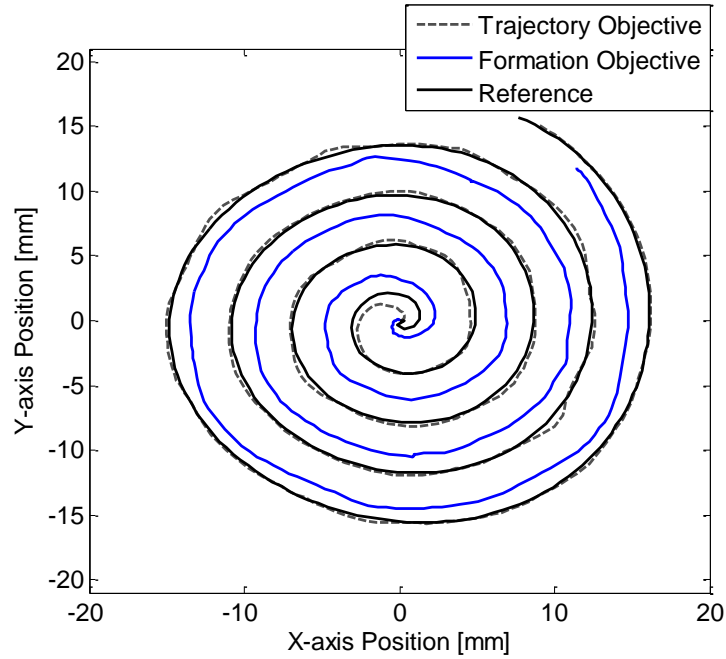


Figure G.17: XY position tracking for agent 2. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.

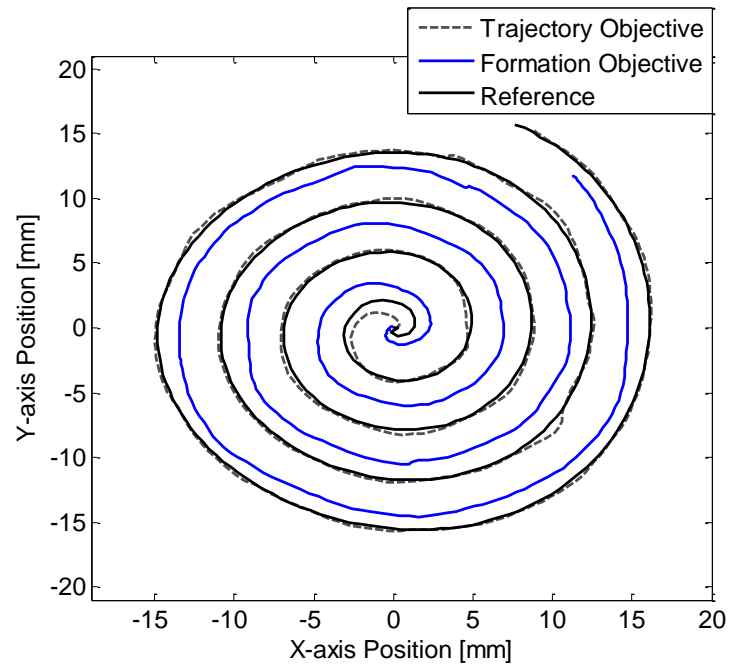


Figure G.18: XY position tracking for agent 3. Note the decrease in performance that occurs as a result of weighting the formation objective rather than the trajectory tracking objective.

Appendix H

MATLAB Code: Simple 2-D ILC Design Example

The goal of this appendix is to present the code for deriving and simulating 2D learning controllers. In particular, this section provides the necessary code for building time-varying weighting matrices. Time-varying weighting matrices (presented in Chapter 5 Section 5.3) enable the control designer to compensate for position or time-dependent disturbances, dynamics, and tracking errors. Using the code provided in the following sections, time-varying weighting matrices can be determined and implemented in simulation or on an experimental testbed.

H.1 2D Weighting Matrix Design

The code in this section enables one to independently design individual weighting matrices for varying system trajectory tracking, model uncertainty, and noise attenuation design requirements.

```

% *** 2D Weighting Matrix Design ***

%% *** Initial Conditions ***
% turning time-varying matrices on and off
clear
clc

switchQ = 0;           % switches between ILC and CCILC
switchTVQ = 1;         % switches between TV-ILC and TV-CCILC
timevaryQ = 0;         % time-vary the Q matrix
timevaryS = 0;         % time-vary the S matrix
timevaryR = 0;         % time-vary the R matrix
ccilc = 0;             % tracking at start location is focused on contour tracking

% *** Weighting matrix gains ***
sigma_q = 1.0;         % gain for nominal Q matrix
bwq = 50;              % high-bandwidth value for enhanced tracking performance
sigma_s = 1;           % gain for nominal S matrix
bws = 10;              % conservative learning, enhanced robustness to model uncertainty
sigma_r = 1;           % gain for nominal R matrix
bwr = 10;              % slower learning for enhanced noise rejection
%*****
%% *** Universal Variables
ts = 1;                % sample time stepsize (seconds)
time = 5;              % length of iteration (seconds)
T = 0:ts:time;         % time-iteration range
L = time/ts;           % number of samples
%*****
%% *** Designing the Switching ILC and CCILC Q Matrices
% Switching between CCILC and ILC
if (switchQ)
    sl = [0 1 3 5];     % switching locations determined from application
    n = length(sl);
    for i = 2:n
        for j = sl(i-1)+1:sl(i)
            if (ccilc)
                if (mod(i,2))==0 % determines if the iteration is even or odd
                    Qa(j) = 0;    % even indices indicate CCILC tracking
                else
                    Qa(j) = sigma_q; % ILC portion of Q matrix
                end
            else
                if (mod(i,2))==0
                    Qa(j) = sigma_q;
                else
                    Qa(j) = 0;
                end
            end
        end
    end
end
end
end

```

```

Qb = 1 - Qa; % CCILC portion of Q matrix
% Creating time vectors and matrices with time-varied ILC and CCILC sections
for i=1:L;
    Qa1(2*i-1) = Qa(i);
    Qa1(2*i) = Qa(i);
    Qb1(2*i-1) = Qb(i);
    Qb1(2*i) = Qb(i);
end
% Filtering the signals for smooth transitions
for i = 1:2*L
    Qam1(i) = GaussFilt(Qa1,10,i,2*L,ts);
end
Qam = diag(Qam1);

for i = 1:2*L
    Qbm1(i) = GaussFilt(Qb1,10,i,2*L,ts);
end
Qbm = diag(Qbm1);

save SwitchingQ Qam Qbm
end
%*****
%% *** Designing the Time-Varying CCILC Matrix
% Time-vary Q
if (timevaryQ)
    sl = [0 1 2 3 4 5]; % locations determined from initial error signals (large error values)
    n = length(sl);
    for i = 2:n
        for j = sl(i-1)+1:sl(i)
            if (mod(i,2))==0 % determines if the iteration is even or odd
                Qb(j) = bwq; % even locations indicate large bandwidth locations
            else
                Qb(j) = sigma_q;
            end
        end
    end
end
for i=1:L;
    Qb1(2*i-1) = Qb(i);
    Qb1(2*i) = Qb(i);
end
Qm = diag(Qb1); % Unfiltered signal for use in switchTVQ
% Filtering the signal for smooth transitions
for i = 1:2*L
    Qbm1(i) = GaussFilt(Qb1,10,i,2*L,ts);
end
Qmf = diag(Qbm1);

save TimeVaryQ Qm Qmf
end

```

```

*****
%% *** Designing the Switching TV-ILC and TV-CCILC Q Matrices
% Switching between TV-CCILC and TV-ILC
if (switchTVQ)
    sl = [0 2 4 5]; % switching locations determined from application
    n = length(sl);
    load TimeVaryQ
    Qa = diag(Qm); % time-varying time vector
    Qb = diag(Qm);
    clear Qm Qmf
    for i = 2:n
        for j = sl(i-1)+1:sl(i)
            if (ccilc)
                if (mod(i,2))==0 % even indices indicate time-varied CCILC tracking
                    Qa1(j) = Qa(j)*0;
                    Qb1(j) = Qb(j)*1;
                else
                    Qa1(j) = Qa(j)*1;
                    Qb1(j) = Qb(j)*0;
                end
            else
                if (mod(i,2))==0
                    Qa1(j) = Qa(j)*1;
                    Qb1(j) = Qb(j)*0;
                else
                    Qa1(j) = Qa(j)*0;
                    Qb1(j) = Qb(j)*1;
                end
            end
        end
    end
    end
    % Filtering the signals for smooth switching
    for i = 1:2*L
        Qam1(i) = GaussFilt(Qa1,15,i,2*L,ts);
    end
    Qam = diag(Qam1);

    for i = 1:2*L
        Qbm1(i) = GaussFilt(Qb1,15,i,2*L,ts);
    end
    Qbm = diag(Qbm1);

    save SwitchingTVQ Qam Qbm
end
% *****
%% *** Designing the Time-Varying S Matrix
% Time-vary S
if (timevaryS)
    sl = [0 1 4 5]; % locations determined from position/time dependent dynamics
    n = length(sl);

```

```

for i = 2:n
    for j = sl(i-1)+1:sl(i)
        if (mod(i,2))==0 % even indices indicate conservative S design
            S1(j) = bws;
            sat(j) = 1; % turns on saturation for simulation results
        else
            S1(j) = sigma_s;
            sat(j) = 0;
        end
    end
end
for i=1:L;
    S2(2*i-1) = S1(i);
    S2(2*i) = S1(i);
end
% Filtering the signal for smooth transitions
for i = 1:2*L
    S3(i) = GaussFilt(S2,5,i,2*L,ts);
end
S = diag(S3);
sat = [sat sat(j)]; sat = [t' sat'];
save TimeVaryS S sat
end
%*****
%% *** Designing the Time-Varying R Matrix
% Time-vary R
if (timevaryR)
    sl = [0 3 5]; % locations determined from noise/disturbance time indices
    n = length(sl);
    for i = 2:n
        for j = sl(i-1)+1:sl(i)
            if (mod(i,2))==0 % even indices indicate conservative R design
                R1(j) = bwr;
                Noise(j) = 1; % noise present in the system
            else
                R1(j) = sigma_r;
                Noise(j) = 0;
            end
        end
    end
    for i=1:L;
        R2(2*i-1) = R1(i);
        R2(2*i) = R1(i);
    end
    % Filtering the signal for smooth transitions
    for i = 1:2*L
        R3(i) = GaussFilt(R2,15,i,2*L,ts);
    end
    R = diag(R3);
    save TimeVaryR R Noise
end
%*****

```

H.2 2D Simulation Program

```
% *** Simulation Program - Simple 2D Example ***

clear
clc

% Load models
load models          % User defined plant models - closed-loop (Tss), plant sensitivity (Pss)
load ModelUncertainty % User defined model uncertainty (Dss) for time-varying S case
load Filter          % User defined filter for model uncertainty (Fss)

% Reference Trajectory
load Trajectory      % User defined reference trajectory
%*****

%% *** Search Variables ***

% Q,S,R Weighting Gains - nominal design
sigma_r = 1;        % R weighting matrix gain wrt Q,S weighting matrices
sigma_s = 1;        % S weighting matrix gain wrt Q,R weighting matrices
sigma_q = 1;        % Q weighting matrix gain wrt S,R weighting matrices
gamma1 = 1;         % ILC - individual axis tracking
gamma2 = 1-gamma1;  % CCILC - contour tracking

% Search options (1 = yes, 0 = no)
calcstab = 0;       % Calculate stability & convergence
simulate = 1;       % Simulate learning and measure converged performance
storedata = 0;      % Write data to a dataset file
switchingq = 0;     % Switches between ILC and CCILC design
switchingtvq = 0;   % Switches between ILC and TV CCILC design
timevaryq = 0;      % time-varying Q matrix: only use all ILC or all CCILC
timevarys = 0;      % time-varying S matrix
timevaryr = 0;      % time-varying R matrix
filter = 0;         % adds a filter to the model uncertainty case (time-varying S)
sat_on = 0;         % turns on control saturation
%*****

%% *** Universal Variables ***
ts = 1;             % sample time stepsize (seconds)
time = 5;           % length of iteration (seconds)
T = 0:ts:time;      % time-iteration range
L = time/ts;        % number of samples
t = [0:ts:time];

% Learning Gains
learn = 1;          % turning on learning

% Initialize saturation
sat = [T' zeros(L+1,1)];
satn = [T' 1-sat(:,2)];

% Simulation variables
MaxIter = 10;       % Number of iterations to run
%*****
```



```

%% *** Datafile Setup ***
if (storedata)
    % Make data file
    mytime=clock;

fname=sprintf('2DDataSet_v2_','%1.2d;%1.2d;%1.2d,%s',mytime(3),mytime(4),mytime(5),date);
clear mytime

    % Initialize Data Set
    DataSet(1).r = 0;
    DataSet(1).s = 0;
    DataSet(1).a = 0;
    DataSet(1).b = 0;
end
%*****
%% *** Stability / Convergence Matrix Setup ***
disp ('Generating matrices for stability calculations...')
load LiftedMatrices % LiftedMatrix m-file
Plant = Plant(1:2*L,1:2*L);
C = C(1:L,1:2*(L));
disp('Matrix generation complete')
%*****
%% *** Generate Weighting Matrices for 2D learning control ***
% Nominal controllers
Qccilc = sigma_q*gamma2*C'*C;
Qilc = sigma_q*gamma1*eye(2*(L));
R = sigma_r*eye(2*(L));
S = sigma_s*eye(2*(L));

if (switchingq)
    load SwitchingQ % Loads filtered Qam and Qbm, switching between ILC and CCILC
    Qccilc = C'*C*Qbm(1:L,1:L);
    Qilc = Qam(1:L,1:L);
elseif (switchingtvq)
    load SwitchingTVQ % Loads filtered Qam and Qbm, switching between TV ILC and CCILC
    Qccilc = C'*C*Qbm(1:L,1:L);
    Qilc = Qam(1:L,1:L);
elseif (timevaryq)
    load TimeVaryQ % Loads Qm = time-varying learning, Qmf = filtered time-varying learning
    Qccilc = gamma2*C'*C*Qmf(1:L,1:L);
    Qilc = gamma1*Qmf(1:L,1:L);
elseif (timevarys)
    load TimeVaryS % Loads S = time-varying matrix, sat = time-varying saturation vector
    S = S(1:L,1:L);
    sat = sat(1:L+1,:);
    satn = [T; (1-sat(:,2))'];
elseif (timevaryr)
    load TimeVaryR % Loads R = time-varying matrix, Noise = time-varying noise vector
    R = R(1:L,1:L);
end

```

```

% Progress message
disp(sprintf('Calculating for data set r=%i, s=%i, a=%i, b=%i',r,s,a,b,))

% *** Calculate Stability ***
% *** Designing Optimal Learning Controllers
% Finding the optimal Learning controller for CCILC and ILC
Le = inv(Plant'*(Qccilc+Qilc)*Plant + S + R)*Plant'*(Qccilc+Qilc);
Lu = inv(Plant'*(Qccilc+Qilc)*Plant + S + R)*(Plant'*(Qccilc+Qilc)*Plant + R);

if (calcstab)
    M = Lu-Le*Plant;
    % Stability calculation
    stability = max(abs(eig(M)));
    % Convergence calculation
    convergence = max(svd(M));

    disp(sprintf(' Stability=%f Convergence=%f',stability,convergence))
end
%*****
%% *** Simulate Learning ***
if (simulate)

    % *** MIMO System ***
    nullsys = ss(tf([0],[1]));
    Plantcomb = [P1ss nullsys; nullsys P2ss];
    Ttotal = [T1ss nullsys; nullsys T2ss];

    % *** initialize E and U ***
    U = [zeros(1,2*L)];
    Ux = [T; zeros(1,L+1)];
    Uy = [T; zeros(1,L+1)];

    % initialize statistics data
    d_xRMS = 0;
    d_yRMS = 0;
    d_cRMS = 0;
    d_xMAX = 0;
    d_yMAX = 0;
    d_cMAX = 0;

    for iter = 0:MaxIter

        % *** Simulate and measure performance ***
        if (timevaryr)
            Noise = [0; Noise'];
            x = rand(1,L+1);
            noise(2,:) = (x-mean(x))/10;
            for i = 1:length(x)
                noise(2,i) = noise(2,i)*Noise(i);
            end
        end
    end
end

```

```

sim('2D_Model',time) % Simulate one iteration
% Update statistics
d_xRMS(iter+1) = RMSx;
d_yRMS(iter+1) = RMSy;
d_cRMS(iter+1) = RMScontour;
d_xMAX(iter+1) = MAXx;
d_yMAX(iter+1) = MAXy;
d_cMAX(iter+1) = MAXcontour;
% *** Learning Update Law ***
for i = 1:L
    Uold(2*i-1) = controlx(i);
    Uold(2*i) = controly(i);
end
for i = 1:L
    Eold(2*i-1) = enewx(i+1);
    Eold(2*i) = enewy(i+1);
end
U = Lu*Uold'+Le*Eold';
Ux(2,:) = [U(1:2:end)]' zeros(1,1)];
Uy(2,:) = [U(2:2:end)]' zeros(1,1)];
end
end
%*****
%% *** Store Data ***
if (storedata)
    % Write results to data structure
    DataSet(cnt).r = r;
    DataSet(cnt).s = s;
    DataSet(cnt).a = a;
    DataSet(cnt).b = b;
    if (calcstab)
        DataSet(cnt).stab = stability;
        DataSet(cnt).conv = convergence;
    end
    if (simulate)
        DataSet(cnt).N = iter;
        DataSet(cnt).rmsx = d_xRMS;
        DataSet(cnt).rmsy = d_yRMS;
        DataSet(cnt).rmsc = d_cRMS;
        DataSet(cnt).maxx = d_xMAX;
        DataSet(cnt).maxy = d_yMAX;
        DataSet(cnt).maxc = d_cMAX;
        DataSet(cnt).exinf = enewx;
        DataSet(cnt).eyinf = enewy;
        DataSet(cnt).ecinf = ContourError;
        DataSet(cnt).uxinf = controlx;
        DataSet(cnt).uyinf = controly;
    end
    % Save Data
    save(fname,'DataSet')
end
%*****

```

H.3 Gaussian Filter Program

```

function wfilt = GaussFilt(w, omega, to, tn, ts)
% This function filters a single location (to) in w and returns it as wfilt.

% w is a vector of data to be filtered
% omega is the frequency of the filter (Hz)
% to is the location to be filtered
% tn is the index of the last data point in w (length of w)
% ts is the sample time
% *****
%% Determines width of Q-Filter: Gaussian Dist. from -Z to Z
Z_RATIO = 2.5762;
% Use Z=1.9604 for 95% of Gauss. Dist.
% Use Z=2.5762 for 99% of Gauss. Dist.
% Use Z=3.2908 for 99.9% of Gauss. Dist.
% Use Z=3.8906 for 99.99% of Gauss. Dist.
% *****
%% Find sigma that gives bandwidth omega
sigma = sqrt(log(2))/(2*pi*omega);

% Find length of q-filter required to meet given Z-Ratio for Gaussian Distribution
N = ceil(Z_RATIO*sigma/ts); % Make N and integer by rounding up
N = min(N, tn/2); % N must be less than tn/2
% *****
%% Filter Convolution
wfilt = w(to);
den = 1; % filter denominator - used to normalize filter
for i=1:N
    gauss = exp(-0.5*(i*ts/sigma)^2);
%    gauss = 0;
    den = den + 2*gauss;

% Filter right side
if (to+i <= tn) % if not past right boundary, then filter here
    wfilt = wfilt + w(to+i)*gauss;
else % otherwise, fold back and filter at new location
    wfilt = wfilt + w(2*tn-to-i+1)*gauss;
end

% Filter left side
if (to-i >= 1) % if not past left boundary, then filter here
    wfilt = wfilt + w(to-i)*gauss;
else % otherwise, fold back and filter at new location
    wfilt = wfilt + w(1-to+i)*gauss;
end
end

wfilt = wfilt / den; % Normalize Q-Filter
% *****

```

H.4 Simulink Model

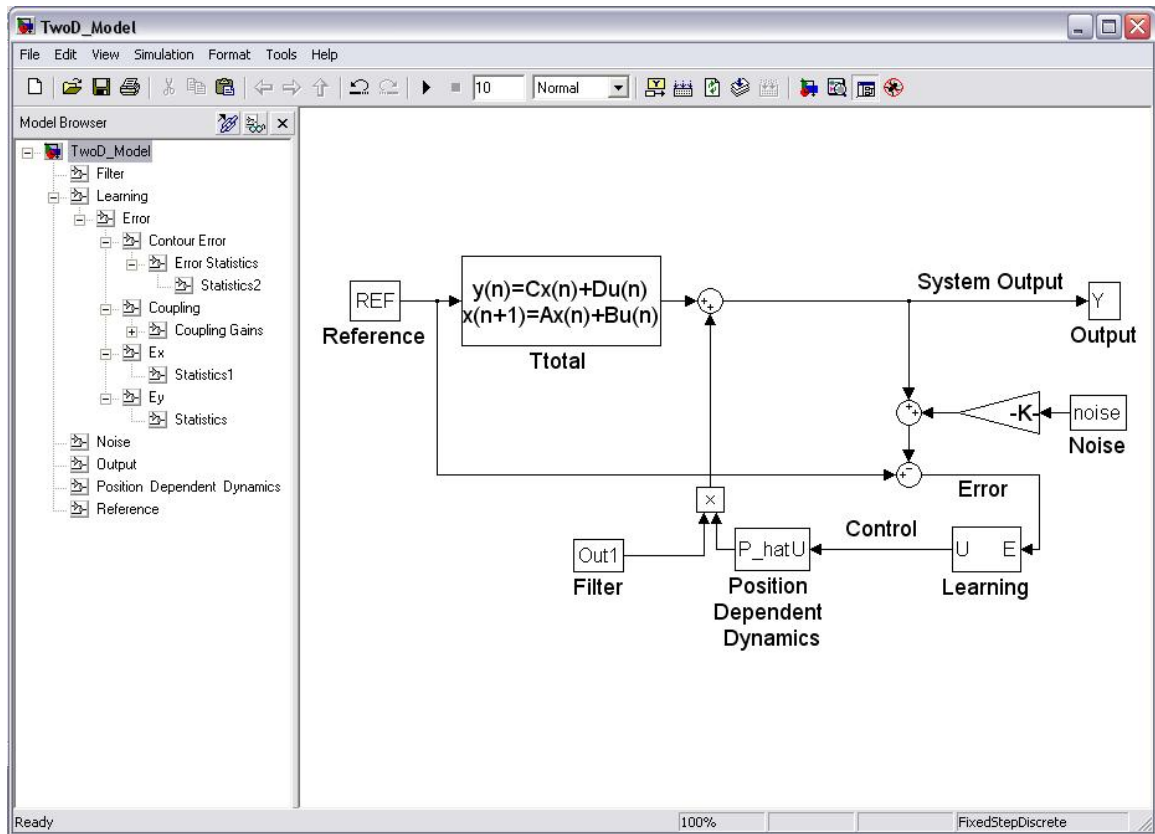


Figure H.1: Simulink model for simulating the 2D time-varying tracking problem. Note that the model contains several subsystems within the basic block diagram.

Appendix I

MATLAB Code: Simple 3-D ILC Design Example

The goal of this section is to present a simple example of how to design and implement a 3-D coordinated ILC controller in simulation including: the MATLAB code to derive the coordinated controller such as the lifted plant and $\{\hat{\mathbf{Q}}, \hat{\mathbf{S}}, \hat{\mathbf{R}}\}$ weighting matrices, a simple Simulink model for simulations, example control and error signals, and simulated results.

Some basic assumptions that were made with respect to this example are included below.

- A1) The systems are assumed to be similar, but not exact.
- A2) Initial conditions on each system are assumed to be zero.
- A3) There are no external disturbances for this example.
- A4) Simple $\{q, r, s\}$ weighting gains are chosen to simplify the controller design and are not optimized for performance.
- A5) The plant models do not include model uncertainty.

I.1 MatLAB Code for Controller Design

This code is used to simulate a 3-D learning controller on models of the 3 agent PKM system presented in Chapter 7.

```

% *** Controller Design Program - Formation and Individual System Tracking ***
clear
clc
% *****
%% *** Search Variables ***
% Search options (1 = yes, 0 = no)
calctab = 0; % Calculate stability & convergence
simulate = 1; % Simulate learning and measure converged performance
storedata = 1; % Write data to a dataset file
case1 = 1; % Individual system tracking
case2 = 0; % Individual trajectory and/or shape tracking
case3 = 0; % Leader ref trajectory and formation shape tracking
case4 = 0; % Formation center trajectory and formation shape tracking
% Simulation variables
MaxIter = 5; % Number of iterations to run
% *****
%% *** Plant models ***
% Three 2-DOF mass + damping systems
p = 3; % number of systems
m = [1.55 1.51 .75 .80 .91 .89]; % mass
b = [.62 .6 .52 .58 .9 .88]; % damping
for i = 1:6
    G(i) = tf([1/m(i)],conv([1 0],[1 b(i)/m(i)]));
    Gz(i) = c2d(G(i),1);
end
% Proportional feedback controller
Cz = tf([.425],[1,1]);

for i = 1:6
    H(i) = feedback(Gz(i),Cz);
end
% *****
%% *** Triangle Trajectory ***
N = 4; % length of iteration (seconds)
t = [0:1:4];
y = [t' zeros(1,length(t))']; x = [t' zeros(1,length(t))'];
for i = 1:N/4
    y(i+1,2) = (2*i)/N; x(i+1,2) = (2*i)/N;
end
for i = N/4+1:N/2
    y(i+1,2) = (2*(N/2-i))/N; x(i+1,2) = 1/2;
end
for i = N/2+1:3*N/4
    y(i+1,2) = (2*(i-N/2))/N; x(i+1,2) = (2*i-N/2)/N;
end
for i = 3*N/4+1:N
    y(i+1,2) = (2*(N-i))/N; x(i+1,2) = 1;
end
% same reference trajectory used for each system
YREF1 = y; XREF1 = x;
YREF2 = y; XREF2 = x;
YREF3 = y; XREF3 = x;

```

```

%*****
%% *** Universal Variables ***
ts = 1;          % sample time stepsize (seconds)
time = N;        % length of iteration (seconds)
T = 0:ts:time;   % time-iteration range
L = time/ts;     % number of samples
t = [0:ts:time];
%*****
%% *** Datafile Setup ***
if (storedata)
    % Make data file
    mytime=clock;

fname=sprintf('FormDataSet,%1.2d;%1.2d;%1.2d,%s',mytime(3),mytime(4),mytime(5),date);
    clear mytime

    % Initialize Data Set
    DataSet(1).c1 = 0;
    DataSet(1).c2 = 0;
    DataSet(1).c3 = 0;
    DataSet(1).c4 = 0;
end
%*****
%% *** Stability / Convergence Matrix Setup ***
disp ('Generating matrices for stability calculations...')
XREF = XREF1;
YREF = YREF1;
[Plant C] = LiftedMatrix_diss(H, XREF, YREF, time); % m-file provided in
subsequent section
disp('Matrix generation complete')
%*****
%% *** Search ***
% Generate Weighting Matrices for LQ Control
if (case1)
    [Q_hat S_hat R_hat] = Run_case1(C, L);
end
if (case2)
    [Q_hat S_hat R_hat] = Run_case2(C, L, p);
end
if (case3)
    [Q_hat S_hat R_hat] = Run_case3(C, L, p);
end

% Designing Optimal Learning Controllers
Le = inv(Plant'*Q_hat*Plant + R_hat + S_hat)*Plant'*Q_hat;
Lu = inv(Plant'*Q_hat*Plant + R_hat + S_hat)*(Plant'*Q_hat*Plant + R_hat);

if (calcstab)

    M = Lu-Le*Plant;

```



```

% Stability calculation
stability = max(abs(eig(M)));

% Convergence calculation
convergence = max(svd(M));

disp(sprintf(' Stability=%f Convergence=%f',stability,convergence))
end
% *****
% *** Simulate Learning ***
if (simulate)

% *** MIMO System ***
nullsys = ss(tf([0],[1]));
Plantcomb = [ss(H(1)) nullsys nullsys nullsys nullsys nullsys;
             nullsys ss(H(2)) nullsys nullsys nullsys nullsys;
             nullsys nullsys ss(H(3)) nullsys nullsys nullsys;
             nullsys nullsys nullsys ss(H(4)) nullsys nullsys;
             nullsys nullsys nullsys nullsys ss(H(5)) nullsys;
             nullsys nullsys nullsys nullsys nullsys ss(H(6))];
[Abar,Bbar,Cbar] = obsvf(Plantcomb.A,Plantcomb.B,Plantcomb.C);
Plantcomb.A = Abar; Plantcomb.B = Bbar; Plantcomb.C = Cbar;
clear Abar Bbar Cbar

% *** initialize E and U ***
U = [zeros(1,2*p*L)];
Ux1 = XREF1';
Uy1 = YREF1';
Ux2 = XREF2';
Uy2 = YREF2';
Ux3 = XREF3';
Uy3 = YREF3';

% initialize statistics data
d_xRMS = [0 0 0];
d_yRMS = [0 0 0];
d_cRMS = [0 0 0];
d_xMAX = [0 0 0];
d_yMAX = [0 0 0];
d_cMAX = [0 0 0];
d_dRMS = [0 0 0];
d_hRMS = [0 0 0];

for iter = 0:MaxIter

% *** Simulate and measure performance ***
sim('FormationShapeModel_diss',time) % Simulate one iteration

% Update statistics
d_xRMS(iter+1,:) = [RMSx1 RMSx2 RMSx3];
d_yRMS(iter+1,:) = [RMSy1 RMSy2 RMSy3];
d_cRMS(iter+1,:) = [RMSc1 RMSc2 RMSc3];

```

```

d_xMAX(iter+1,:) = [MAXx1 MAXx2 MAXx3];
d_yMAX(iter+1,:) = [MAXy1 MAXy2 MAXy3];
d_cMAX(iter+1,:) = [MAXc1 MAXc2 MAXc3];
d_dRMS(iter+1,:) = [RMSd1 RMSd2 RMSd3];
d_hRMS(iter+1,:) = [RMSH1 RMSH2 RMSH3];

% *** Learning Update Law ***
for i = 1:L
    Uold(6*i-5) = controlx1(i);
    Uold(6*i-4) = controly1(i);
    Uold(6*i-3) = controlx2(i);
    Uold(6*i-2) = controly2(i);
    Uold(6*i-1) = controlx3(i);
    Uold(6*i) = controly3(i);
end
for i = 1:L
    Eold(6*i-5) = enewx1(i+1);
    Eold(6*i-4) = enewy1(i+1);
    Eold(6*i-3) = enewx2(i+1);
    Eold(6*i-2) = enewy2(i+1);
    Eold(6*i-1) = enewx3(i+1);
    Eold(6*i) = enewy3(i+1);
end

U = Lu*Uold'+Le*Eold';
% separate into individual signals
Ux1(2,:) = [U(1:6:end)' zeros(1,1)];
Uy1(2,:) = [U(2:6:end)' zeros(1,1)];
Ux2(2,:) = [U(3:6:end)' zeros(1,1)];
Uy2(2,:) = [U(4:6:end)' zeros(1,1)];
Ux3(2,:) = [U(5:6:end)' zeros(1,1)];
Uy3(2,:) = [U(6:6:end)' zeros(1,1)];

end

% Output results
disp(sprintf('xRMS=%0.3e %0.3e %0.3e yRMS=%0.3e %0.3e
%0.3e',d_xRMS(iter,:),d_yRMS(iter,:)));
disp(sprintf('dRMS=%0.3e %0.3e %0.3e hRMS=%0.3e %0.3e
%0.3e',d_dRMS(iter,:),d_hRMS(iter,:)));
end
% *****
% *** Store Data ***
if (storedata)
    % Write results to data structure
    DataSet(1).c1 = case1;
    DataSet(1).c2 = case2;
    DataSet(1).c3 = case3;
    DataSet(1).c4 = case4;
    if (calcstab)
        DataSet(1).stab = stability;
        DataSet(1).conv = convergence;
    end
end

```

```

if (simulate)
    DataSet(1).N = iter;
    DataSet(1).rmsx = d_xRMS;
    DataSet(1).rmsy = d_yRMS;
    DataSet(1).rmsc = d_cRMS;
    DataSet(1).rmsd = d_dRMS;
    DataSet(1).rmsh = d_hRMS;
    DataSet(1).maxx = d_xMAX;
    DataSet(1).maxy = d_yMAX;
    DataSet(1).maxc = d_cMAX;
    DataSet(1).exinf = [enewx1 enewx2 enewx3];
    DataSet(1).eyinf = [enewy1 enewy2 enewy3];
    DataSet(1).edinf = [Ed1 Ed2 Ed3];
    DataSet(1).ehinf = [Eh1 Eh2 Eh3];
    DataSet(1).ecinf = [CE1 CE2 CE3];
    DataSet(1).uxinf = [controlx1 controlx2 controlx3];
    DataSet(1).uyinf = [controly1 controly2 controly3];
end
% Save Data
save(fname,'DataSet')
end

```

I.1.1.1 Case 1

```

% *** Run - Case 1 ***

function [Q_hat S_hat R_hat] = Run_case1(C, L);
% This function returns the (Q_hat,S_hat,R_hat) weighting matrices for individual
% system trajectory tracking with indirect formation shape tracking

% C = lifted coupling gains
% L = number of sample steps in one iteration
% *****
% Gain Values
r1 = 1e-6; r2 = 1e-6; r3 = 1e-6;      % gains for R and S weighting matrices
s1 = 1e-4; s2 = 1e-4; s3 = 1e-4;
a = 1;      % ILC - individual tracking
b = 1-a;    % CCILC - contour tracking
% *****
% Layer 1: Generate Individual System Weighting Matrices (2-D design)
Q1 = eye(2*(L))*a + C'*(eye(L)*b)*C;      % System 1 2-D controller
R1 = eye(2*(L))*r1;
S1 = eye(2*(L))*s1;

Q2 = eye(2*(L))*a + C'*(eye(L)*b)*C;      % System 2 2-D controller
R2 = eye(2*(L))*r2;
S2 = eye(2*(L))*s2;

Q3 = eye(2*(L))*a + C'*(eye(L)*b)*C;      % System 3 2-D controller
R3 = eye(2*(L))*r3;
S3 = eye(2*(L))*s3;
% Building Q,S,R weighting matrices for 3-D controller design
for i = 1:L
    Q(6*i-5:6*i-4,6*i-5:6*i-4)=Q1(2*i-1:2*i,2*i-1:2*i);
    Q(6*i-3:6*i-2,6*i-3:6*i-2)=Q2(2*i-1:2*i,2*i-1:2*i);
    Q(6*i-1:6*i,6*i-1:6*i)=Q3(2*i-1:2*i,2*i-1:2*i);

    S(6*i-5:6*i-4,6*i-5:6*i-4)=S1(2*i-1:2*i,2*i-1:2*i);
    S(6*i-3:6*i-2,6*i-3:6*i-2)=S2(2*i-1:2*i,2*i-1:2*i);
    S(6*i-1:6*i,6*i-1:6*i)=S3(2*i-1:2*i,2*i-1:2*i);

    R(6*i-5:6*i-4,6*i-5:6*i-4)=R1(2*i-1:2*i,2*i-1:2*i);
    R(6*i-3:6*i-2,6*i-3:6*i-2)=R2(2*i-1:2*i,2*i-1:2*i);
    R(6*i-1:6*i,6*i-1:6*i)=R3(2*i-1:2*i,2*i-1:2*i);
end

clear Q1 R1 S1 Q2 R2 S2 Q3 R3 S3 Z
% *****

```

I.1.2 Case 2

```

% *** Run - Case 2 ***

function [Q_hat S_hat R_hat] = Run_case2(C, L, p);
% This function returns the (Q_hat,S_hat,R_hat) weighting matrices for individual system
% trajectory tracking plus direct formation shape tracking

% C = lifted coupling gains
% L = number of sample steps in one iteration
% p = number of systems
% *****
% Gain Values
r1 = 1e-6; r2 = 1e-6; r3 = 1e-6;
s1 = 1e-4; s2 = 1e-4; s3 = 1e-4;
a = 1; % ILC - individual tracking
b = 1-a; % CCILC - contour tracking
% *****
% Layer 1: Generate Individual System Weighting Matrices (2-D design)
Q1 = eye(2*(L))*a + C*(eye(L)*b)*C;
R1 = eye(2*(L))*r1;
S1 = eye(2*(L))*s1;

Q2 = eye(2*(L))*a + C*(eye(L)*b)*C;
R2 = eye(2*(L))*r2;
S2 = eye(2*(L))*s2;

Q3 = eye(2*(L))*a + C*(eye(L)*b)*C;
R3 = eye(2*(L))*r3;
S3 = eye(2*(L))*s3;

for i = 1:L
    Q(6*i-5:6*i-4,6*i-5:6*i-4)=Q1(2*i-1:2*i,2*i-1:2*i);
    Q(6*i-3:6*i-2,6*i-3:6*i-2)=Q2(2*i-1:2*i,2*i-1:2*i);
    Q(6*i-1:6*i,6*i-1:6*i)=Q3(2*i-1:2*i,2*i-1:2*i);

    S(6*i-5:6*i-4,6*i-5:6*i-4)=S1(2*i-1:2*i,2*i-1:2*i);
    S(6*i-3:6*i-2,6*i-3:6*i-2)=S2(2*i-1:2*i,2*i-1:2*i);
    S(6*i-1:6*i,6*i-1:6*i)=S3(2*i-1:2*i,2*i-1:2*i);

    R(6*i-5:6*i-4,6*i-5:6*i-4)=R1(2*i-1:2*i,2*i-1:2*i);
    R(6*i-3:6*i-2,6*i-3:6*i-2)=R2(2*i-1:2*i,2*i-1:2*i);
    R(6*i-1:6*i,6*i-1:6*i)=R3(2*i-1:2*i,2*i-1:2*i);
end

clear Q1 R1 S1 Q2 R2 S2 Q3 R3 S3 Z
% *****
% Layer 2: individual systems, no coupling --> (Q,S,R)

% Layer 3: formation center trajectory + shape tracking
chi1 = 0.5; % trajectory tracking
chi2 = 1-chi1; % shape tracking

```

```

sigma_q = 1;           % overall weighting wrt S,R
sigma_s = 1;           % overall weighting wrt Q,R
sigma_r = 1;           % overall weighting wrt Q,S

F1 = [-1 0 1 0 0 0;    % mapping matrix from position to formation errors
      0 0 -1 0 1 0;
      -1 0 0 0 1 0;
      0 -1 0 1 0 0;
      0 0 0 -1 0 1;
      0 -1 0 0 0 1];
for i = 1:L
    F(6*i-5:6*i,6*i-5:6*i)=F1;
end

Q_hat = eye(length(Q))*sigma_q*chi1*Q + eye(length(Q))*sigma_q*chi2'*F'*F;
S_hat = eye(length(S))*sigma_s*S;
R_hat = eye(length(R))*sigma_r*R;

clear Q_bar S_bar R_bar F
%*****

```

I.1.3 Case 3

```

% *** Run - Case 3 ***

function [Q_hat S_hat R_hat] = Run_case3(C, L, p)
% This function returns the (Q_hat,S_hat,R_hat) weighting matrices for leader reference
% trajectory tracking with direct formation shape tracking

% C = lifted coupling gains
% L = number of sample steps in one iteration
% p = number of systems
% *****
% Gain Values - only need to design for lead system
r = 1;
s = 1;
a = 0;           % ILC - individual tracking
b = 1-a;         % CCILC - contour tracking
% *****
% Layer 1: Generate Leader Weighting Matrices (2-D design)
Q = eye(2*(L))*a + C*(eye(L)*b)*C;
R = eye(2*(L))*r;
S = eye(2*(L))*s;
% *****
% Layer 2: individual leader system, no coupling --> (Q_bar,S_bar,R_bar)
for i = 1:L
    Q_bar(6*i-5:6*i-4,6*i-5:6*i-4)=Q(2*i-1:2*i,2*i-1:2*i);
    Q_bar(6*i-3:6*i-2,6*i-3:6*i-2)=zeros(2);
    Q_bar(6*i-1:6*i,6*i-1:6*i)=zeros(2);

    S_bar(6*i-5:6*i-4,6*i-5:6*i-4)=S(2*i-1:2*i,2*i-1:2*i);
    S_bar(6*i-3:6*i-2,6*i-3:6*i-2)=zeros(2);
    S_bar(6*i-1:6*i,6*i-1:6*i)=zeros(2);

    R_bar(6*i-5:6*i-4,6*i-5:6*i-4)=R(2*i-1:2*i,2*i-1:2*i);
    R_bar(6*i-3:6*i-2,6*i-3:6*i-2)=zeros(2);
    R_bar(6*i-1:6*i,6*i-1:6*i)=zeros(2);
end
clear Q S R
% *****
% Layer 3: formation center trajectory + shape tracking
chi1 = 1/2;      % trajectory tracking
chi2 = 1-chi1;   % shape tracking

sigma_q = 1;     % overall weighting wrt S,R
sigma_s = 1;     % overall weighting wrt Q,R
sigma_r = 1;     % overall weighting wrt Q,S

F1 = [-1 0 1 0 0 0; % mapping matrix from position to formation errors
      0 0 -1 0 1 0;
      -1 0 0 0 1 0;
      0 -1 0 1 0 0;
      0 0 0 -1 0 1;
      0 -1 0 0 0 1];

```

```

for i = 1:L
    F(6*i-5:6*i,6*i-5:6*i)=F1;
end

Q_hat = eye(length(Q_bar))*sigma_q*chi1*Q_bar +
eye(length(Q_bar))*sigma_q*chi2*F'*F;
S_hat = eye(length(S_bar))*sigma_s*S_bar;
R_hat = eye(length(R_bar))*sigma_r*R_bar;

clear Q_bar S_bar R_bar F
%*****

```


I.1.4 Case 4

```

% *** Run - Case 4 ***

function [Q_hat S_hat R_hat] = Run_case2(C, L, p);
% This function returns the (Q_hat,S_hat,R_hat) weighting matrices for formation center
% trajectory tracking with direct formation shape tracking

% C = lifted coupling gains
% L = number of sample steps in one iteration
% p = number of systems
% *****

% Gain Values
r = 1;
s = 1;
a = 0;          % ILC - individual tracking
b = 1-a;        % CCILC - contour tracking
% *****

% Layer 1: Generate Formation Center Weighting Matrices
Q1 = eye(2*(L))*a + C'(eye(L)*b)*C;
R1 = eye(2*(L))*r;
S1 = eye(2*(L))*s;

for i = 1:L
    Q(6*i-5:6*i-4,6*i-5:6*i-4)=Q1(2*i-1:2*i,2*i-1:2*i);
    Q(6*i-3:6*i-2,6*i-3:6*i-2)=Q1(2*i-1:2*i,2*i-1:2*i);
    Q(6*i-1:6*i,6*i-1:6*i)=Q1(2*i-1:2*i,2*i-1:2*i);

    S(6*i-5:6*i-4,6*i-5:6*i-4)=S1(2*i-1:2*i,2*i-1:2*i);
    S(6*i-3:6*i-2,6*i-3:6*i-2)=S1(2*i-1:2*i,2*i-1:2*i);
    S(6*i-1:6*i,6*i-1:6*i)=S1(2*i-1:2*i,2*i-1:2*i);

    R(6*i-5:6*i-4,6*i-5:6*i-4)=R1(2*i-1:2*i,2*i-1:2*i);
    R(6*i-3:6*i-2,6*i-3:6*i-2)=R1(2*i-1:2*i,2*i-1:2*i);
    R(6*i-1:6*i,6*i-1:6*i)=R1(2*i-1:2*i,2*i-1:2*i);
end
% *****

% Layer 2: formation center coupling --> (Q_bar,S_bar,R_bar)
Q_bar = (eye(length(Q))*1/3)'*Q*eye(length(Q))*1/3; % centroid for triangle, E=1/3
S_bar = (eye(length(Q))*1/3)'*S*eye(length(Q))*1/3;
R_bar = (eye(length(Q))*1/3)'*R*eye(length(Q))*1/3;

clear Q S R
% *****

% Layer 3: formation center trajectory + shape tracking
chi1 = 1/2;          % trajectory tracking
chi2 = 1-chi1;       % shape tracking

sigma_q = 1;         % overall weighting wrt S,R
sigma_s = 1;         % overall weighting wrt Q,R
sigma_r = 1;         % overall weighting wrt Q,S

```

```

F1 = [-1 0 1 0 0 0;      % mapping matrix from position to formation errors
      0 0 -1 0 1 0;
      -1 0 0 0 1 0;
      0 -1 0 1 0 0;
      0 0 0 -1 0 1;
      0 -1 0 0 0 1];
for i = 1:L
    F(6*i-5:6*i,6*i-5:6*i)=F1;
end

Q_hat = eye(length(Q_bar))*sigma_q*chi1*Q_bar +
eye(length(Q_bar))*sigma_q*chi2*F'*F;
S_hat = eye(length(S_bar))*sigma_s*S_bar;
R_hat = eye(length(R_bar))*sigma_r*R_bar;

clear Q_bar S_bar R_bar F
%*****

```

I.1.5 Lifted Matrices

```

%% *** Lifted Matrix Calculation ***
function [Plant C] = LiftedMatrix_diss(H, XREF, YREF, time)

% H = transfer function matrix containing the x and y-axis dynamics for each system
% XREF = x-axis reference trajectory
% YREF = y-axis reference trajectory
% time = length of iteration in seconds
% *****

%% *** Universal Variables ***
ts = 1; %sample time stepsize (seconds)
T = 0:ts:time; %time-iteration range
L = time/ts; %number of samples
t = 0:1:time; %time vector
% *****

%% *** Lifted Plant Framework ***
for i = 1:6
    p_imp(i,:) = impulse(H(i),time+ts); % take the impulse of the function
    pp(i,:) = p_imp(i,2:L+1); % drop the zero iteration for time delay
end
% Plant Matrices - size should be {2Np x 2Np} N = time length, p = # systems
% need the matrix to contain increasing time values for each system
for i = 1:L
    Plant1(6*i-5,1) = pp(1,i);
    Plant1(6*i-4,2) = pp(2,i);
    Plant1(6*i-3,3) = pp(3,i);
    Plant1(6*i-2,4) = pp(4,i);
    Plant1(6*i-1,5) = pp(5,i);
    Plant1(6*i,6) = pp(6,i);
end
% Creating a toeplitz Plant matrix
for i = 1:L
    if i < L
        for j = 1:L+1-i
            Plant(6*j-5+6*i-6:6*j+6*i-6,6*j-5:6*j) = Plant1(6*i-5:6*i,1:6);
        end
    else
        Plant(6*i-5:6*i,1:6) = Plant1(6*i-5:6*i,1:6);
    end
end
clear p_imp p Plant1
% *****

% *** Obtaining Coupling Gains ***
sim('CouplingGains',time)

% Coupling Gain Matrix C - size should be (N x 2N)
for i = 1:L
    C(i,2*i-1) = -(Cx(i+1));
    C(i,2*i) = Cy(i+1);
end
clear Cx Cy
% *****

```

I.1.6 Simulation Models

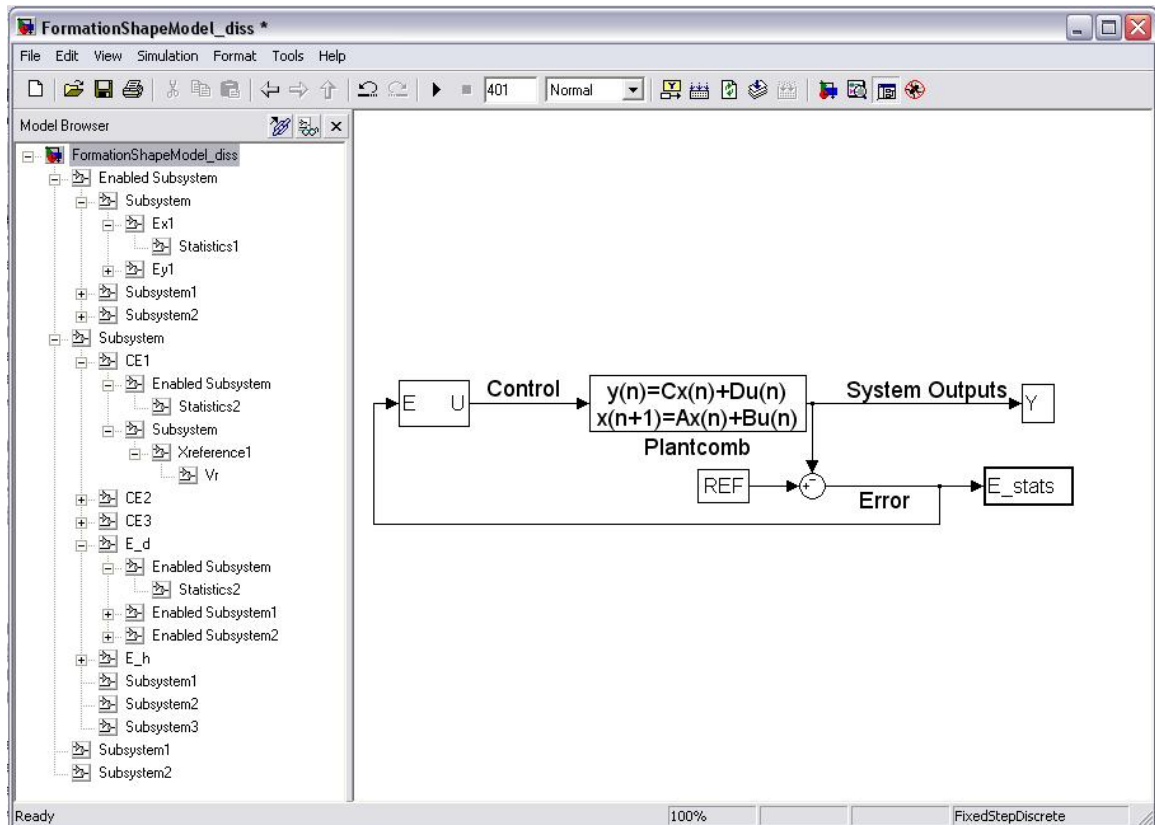


Figure I.1: Simulink model for simulating the formation tracking problem. Note that the model contains several subsystems within the basic block diagram.

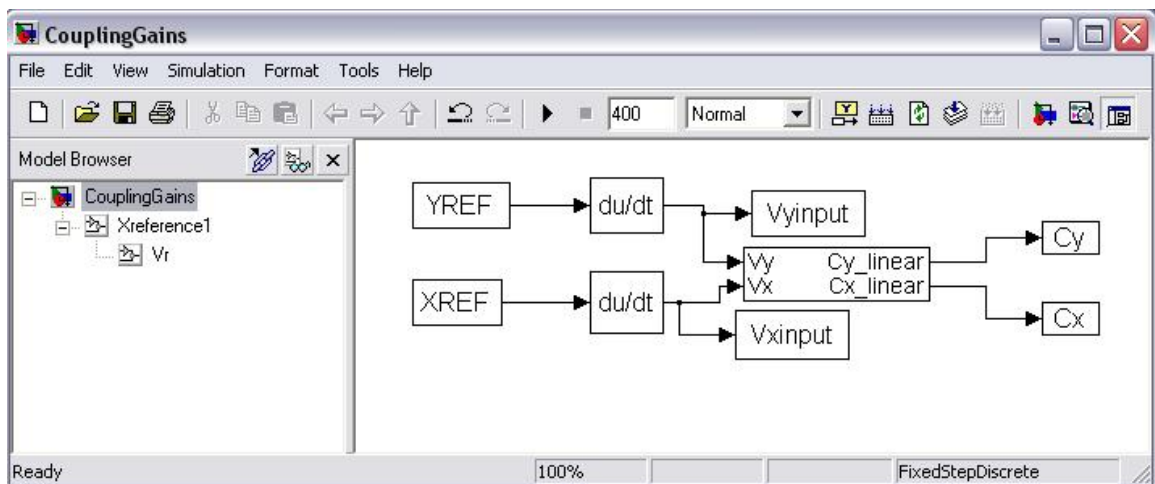


Figure I.2: Simulink model for deriving the trajectory dependent coupling gains. Note that the model contains a few subsystems within the basic block diagram.

I.2 Control and Error Signals

This section presents the control and error signals for a simple example. The signals correspond to a 3-D coordinated learning controller designed for a trajectory tracking objective and were generated using the MATLAB code and simulation models presented in the previous subsections. These converged signals were derived after 5 iterations.

Table I.1: Converged Control Signals U_x and U_y

U_{x1}	time	0	1	2	3	4
	control	1.7259	-2.3916	4.1612	-4.1571	0
U_{x2}	time	0	1	2	3	4
	control	0.9292	-0.9841	2.0876	-1.5258	0
U_{x3}	time	0	1	2	3	4
	control	1.2269	-1.1179	2.3918	-1.5970	0
U_{y1}	time	0	1	2	3	4
	control	1.6569	-3.9662	6.3751	-8.1458	0
U_{y2}	time	0	1	2	3	4
	control	0.9966	-2.0448	3.2467	-3.8187	0
U_{y3}	time	0	1	2	3	4
	control	1.1957	-2.2825	3.4305	-3.9004	0

Table I.2: Converged Error Signals E_x and E_y

E_{x1}	time	0	1	2	3	4
	control	0	0.0106	-0.0087	0.0050	-0.0015
E_{x2}	time	0	1	2	3	4
	control	0	0.0020	-0.0016	0.0010	-0.0003
E_{x3}	time	0	1	2	3	4
	control	0	0.0025	-0.0021	0.0013	-0.0004
E_{y1}	time	0	1	2	3	4
	control	0	0.0173	-0.0148	0.0090	-0.0028
E_{y2}	time	0	1	2	3	4
	control	0	0.0039	-0.0035	0.0022	-0.0008
E_{y3}	time	0	1	2	3	4
	control	0	0.0042	-0.0039	0.0026	-0.0009

Table I.3: Converged Error Signals E_d and E_h

E_{d1}	time	0	1	2	3	4
	control	0	-0.0086	0.0071	-0.0040	0.0012
E_{d2}	time	0	1	2	3	4
	control	0	$0.49e^{-3}$	$-0.50e^{-3}$	$0.37e^{-3}$	$-0.11e^{-3}$
E_{d3}	time	0	1	2	3	4
	control	0	-0.0082	0.0065	-0.0037	0.0011
E_{h1}	time	0	1	2	3	4
	control	0	-0.0134	0.0113	-0.0067	0.0020
E_{h2}	time	0	1	2	3	4
	control	0	0.2810	-0.4113	0.3987	-0.1735
E_{y3}	time	0	1	2	3	4
	control	0	-0.0132	0.0109	-0.0063	0.0019

I.3 Simulation Results

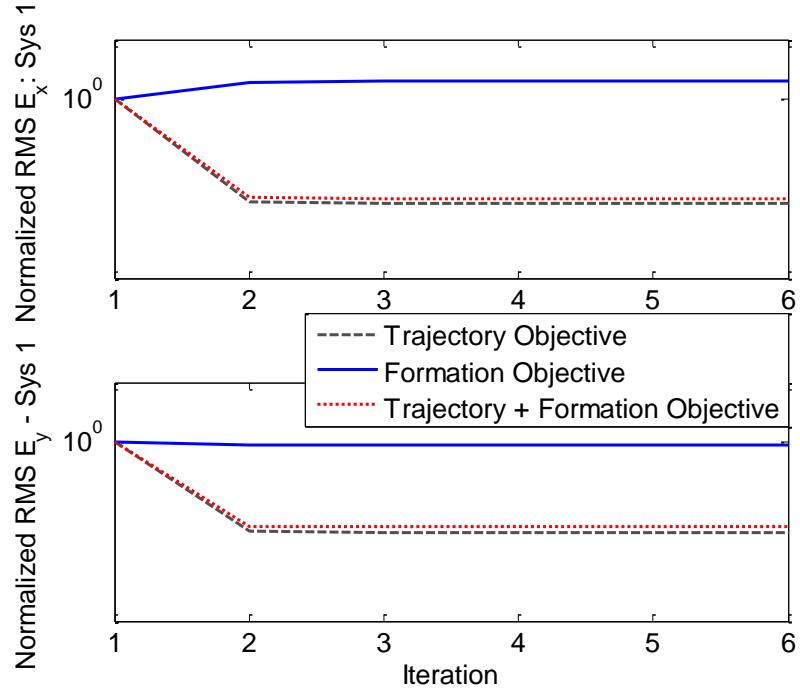


Figure I.3: Normalized RMS x- and y-axis position errors for agent 1. Note that the y-axis is logarithmic, while the x-axis is linear.

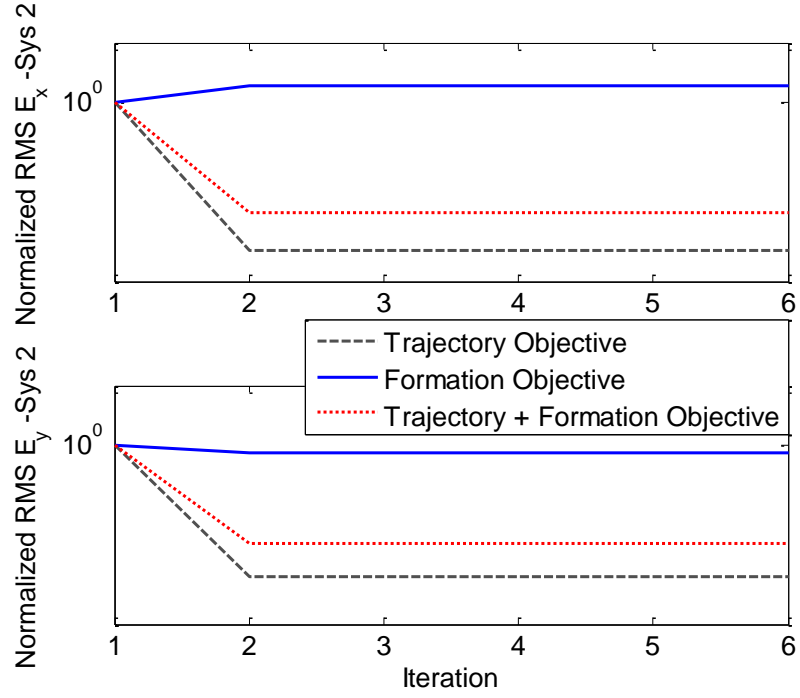


Figure I.4: Normalized RMS x- and y-axis position errors for agent 2. Note that the y-axis is logarithmic, while the x-axis is linear.

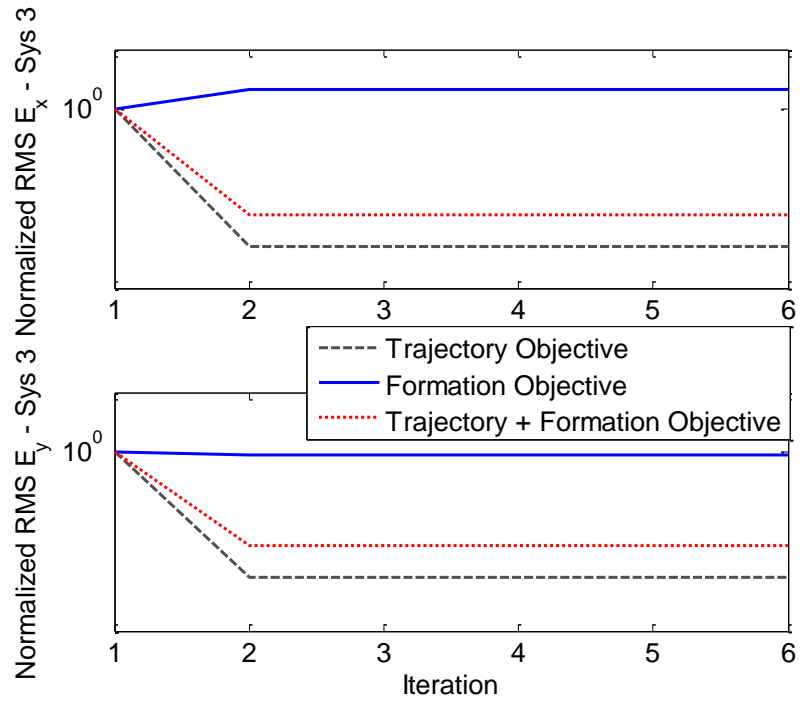


Figure I.5: Normalized RMS x- and y-axis position errors for agent 3. Note that the y-axis is logarithmic, while the x-axis is linear.

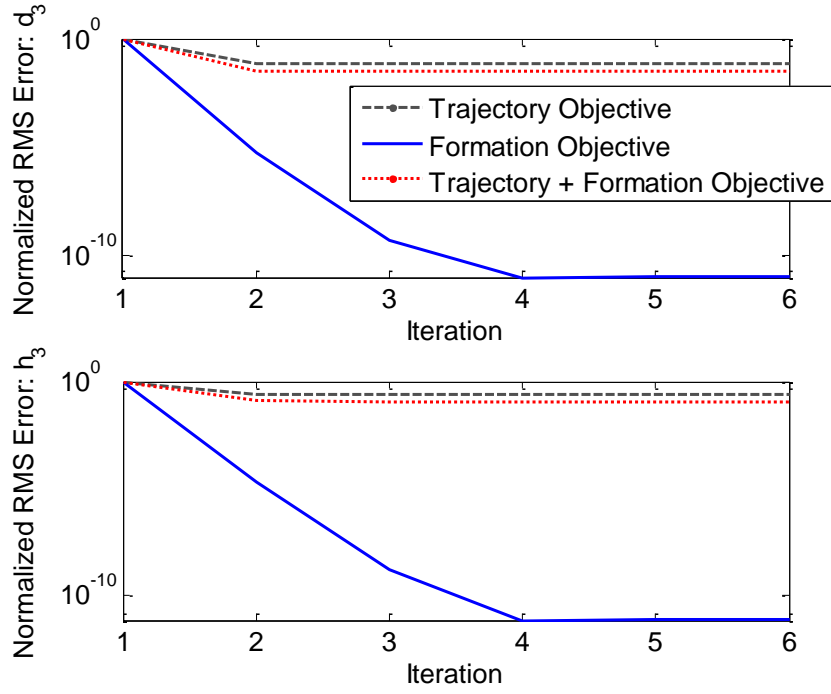


Figure I.6: Normalized RMS E_{d3} and E_{h3} formation errors. Note that the y-axis is logarithmic, while the x-axis is linear.

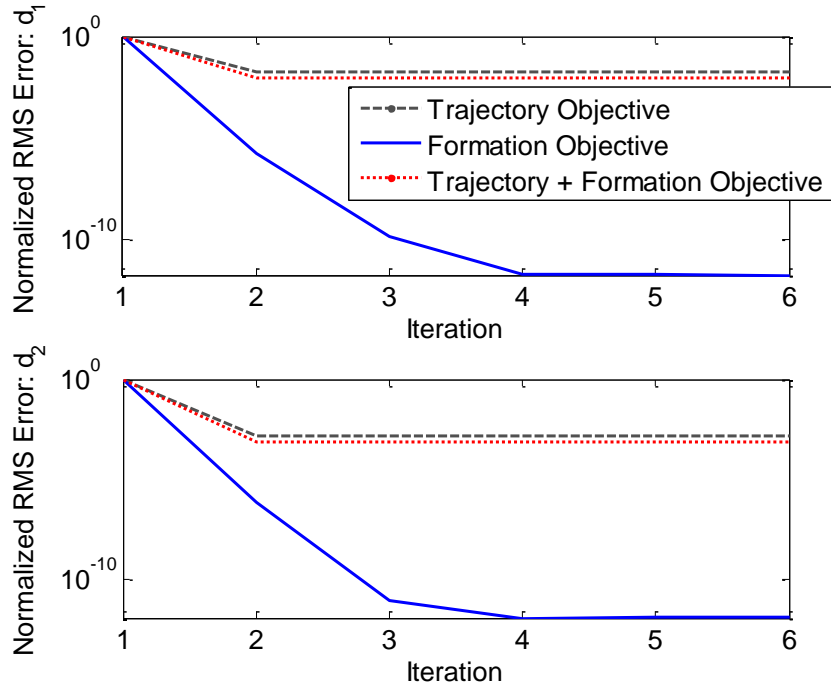


Figure I.7: Normalized RMS E_{d1} and E_{d2} formation errors. Note that the y-axis is logarithmic, while the x-axis is linear.

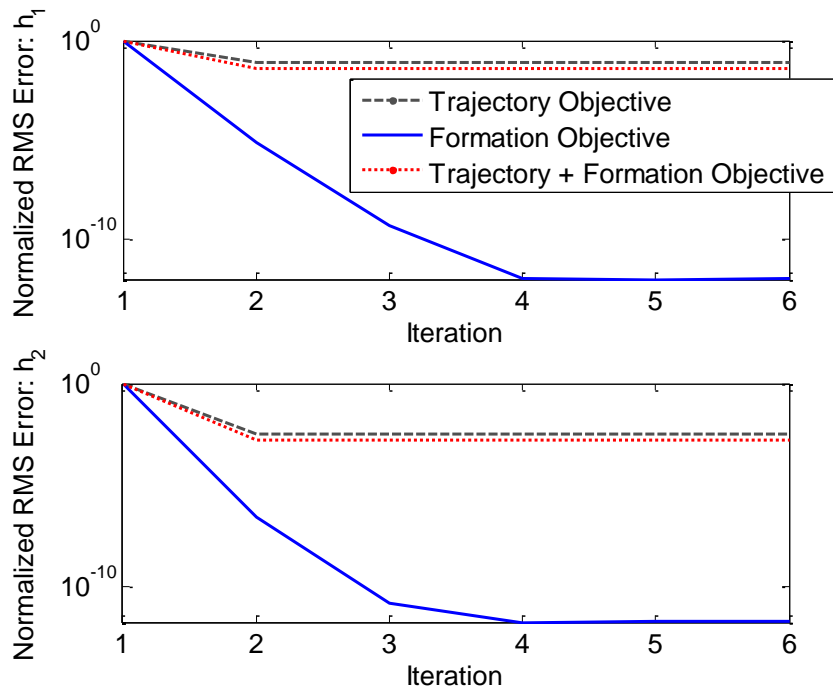


Figure I.8: Normalized RMS E_{h1} and E_{h2} formation errors. Note that the y-axis is logarithmic, while the x-axis is linear.

Appendix J

2-D ILC Design for Dissimilar Systems

J.1 Introduction

Cross Coupled Control (CCC) has been applied to multi-axis systems in which there is a primary objective that defines manufacturing process performance. Individual axis performance is deemphasized in favor of a coupled axis, appropriately defined to measure the primary performance objective [34, 112]. The classic example of the CCC approach is a computer numerically controlled (CNC) robot where the primary objective is the dimensional accuracy of a manufactured part, not individual axis objectives. Performance is defined by a coupled axis, termed contour error, which is the normal distance from the prescribed trajectory and is a metric of the primary objective, i.e. dimensional accuracy. The redefinition of performance objectives developed in CCC has been integrated into the framework of Iterative Learning Control (ILC) by [2] to form Cross Coupled Iterative Learning Control (CCILC). ILC is a control algorithm that can be applied to systems that track a repeated trajectory [37]. The algorithm exploits trajectory repetition to improve reference tracking based off input and output information learned in previous iterations. By directly considering the primary objective and exploiting trajectory repetition, CCILC has been shown to achieve superior performance in comparison to CCC and individual axis ILC alone in contoured trajectory tracking problems [2, 78].

CCC and CCILC have been traditionally applied to planar manufacturing robots in which the X and Y axes have similar yet individual dynamics and are actuated

and sensed by identical hardware. CCILC is a special form of a Multi-Input Multi-Output (MIMO) approach in which two Single-Input Single-Output (SISO) systems are coupled together through the output. This paper considers CCILC applied to a general set of systems, where the individual dynamics, as well as the actuation and sensing hardware, need not be common among the different systems. Previous CCC publications have alluded to potential problems when dissimilar systems are coupled [34, 112, 113]. The work in this paper builds off of the previous work of the authors [114], exploiting performance disparities between two subsystems that have a coupled objective in manufacturing robots. Given a system containing a fast subsystem and a slow subsystem, we apply a weighting filter that penalizes fast subsystem performance in the frequency ranges that are un-trackable by the slow subsystem. This filter shows up in the derivation of the contour error. The proposed controller framework enforces dynamics in the fast subsystem that compensate for inadequacies in the slow subsystem.

The main motivation for this work is manufacturing systems. Besides the example shown in this paper, performance limitations due to dynamical dissimilarities between axes arises in other manufacturing systems where the individual control objectives can be easily handled in one subsystem, while the other subsystem is only capable of achieving poor performance results. One example would be robotic manipulators tooling parts on a conveyor line. Here, the agile robotic manipulator can easily compensate for the positioning of the low-bandwidth conveyor system; if the robotic manipulator knows the conveyor positioning error [115]. Outside of manufacturing, some other examples include chemical mixing [116], hybrid system applications [117], and multi-phase system applications such as heating and air conditioning systems [118].

The CCILC method presented here is applied to a micro-Robotic Deposition (μ -RD) manufacturing system, a rapid prototyping process in which a colloidal ink is

extruded through a micro-sized nozzle while being positioned in space to fabricate three-dimensional structures [119]. The extrusion and positioning systems are drastically different, with extrusion system performance measured in volume and positioning system performance measured in distance. Additionally, the positioning system has a bandwidth that is over 100 times faster than the extrusion system.

The following sections establish the control problem and outline the solution and μ RD implementation. The class of systems valid for this modification of CCILC is defined in Section J.2. Coupling of multiple systems is defined in general and for dissimilar systems in Section J.3. Section J.4 presents CCILC in the Norm Optimal framework. The μ RD systems, particularly the two dissimilar axes of interest, and learning controller design are described in Section J.5. Experimental results are presented and discussed in Section J.6. Section J.7 summarizes the paper and provides concluding statements.

J.2 Class of Systems

In this paper we consider stable, linear time-invariant (LTI), causal, discrete-time MIMO systems, P , which perform the same task repetitively. P is given as

$$P \triangleq \begin{cases} x_j(k+1) = Ax_j(k) + Bu_j(k) \\ \delta y_j(k) = Cx_j(k) + Du_j(k), \end{cases} \quad (\text{J.1})$$

$$y_j(k) = \delta y_j(k) + y_o(k) + d_j(k) \quad (\text{J.2})$$

where $k = 0, 1, \dots, N-1$ is the discrete time index, $j = 0, 1, \dots$ is the iteration index, $u_j(k) \in \mathbb{R}^{q_i}$ is the control, $y_j(k) \in \mathbb{R}^{q_o}$ is the output, $y_o(k) \in \mathbb{R}^{q_o}$ is iteration-invariant, $d_j(k) \in \mathbb{R}^{q_o}$ corresponds to stochastic (iteration-varying) external disturbances, $x_j(k) \in \mathbb{R}^n$ are system states, and (A, B, C, D) are appropriately sized real-valued matrices. It is assumed that $x_j(0) = x_o$ for all j , and note that $y_o(k)$ can be

used to capture iteration-invariant initial conditions, feedback control, and external disturbances. In the lifted-domain [63, 64], the discrete-time behavior of the system is represented by its convolution matrix \mathbf{P} using impulse response data $H_{m,n}(k)$, (J.3), where $\{m, n\}$ identify the indices for the impulse response data and range from $0, \dots, N-1$. Note that a bold variable is used to denote a lifted system description.

$$\mathbf{P} = \begin{bmatrix} H_{0,0} & & 0 \\ \vdots & \ddots & \\ H_{N-1,0} & \cdots & H_{N-1,N-1} \end{bmatrix}. \quad (\text{J.3})$$

For MIMO LTI systems, $H_{m,n}(k)$ contains the impulse response from each of the q_i inputs to each of the q_o outputs and can be derived using the matrices in (J.1),

$$H_{m,n} : \begin{cases} D, & m = n \\ CA^{m-n-1}B, & m > n. \end{cases} \quad (\text{J.4})$$

Given $H_{m,n}(k) \in \mathbb{R}^{q_o \times q_i}$, system $\mathbf{P} \in \mathbb{R}^{Nq_o \times Nq_i}$ is a lower triangular matrix with a block Toeplitz structure. While the results presented in this paper are for an LTI system, the same design process can be applied to LTV systems. In the case of LTV systems, $H_{m,n}$ is of the form,

$$H_{m,n} : \begin{cases} D(m), & m = n \\ C(m)A(m-1)A(m-2) \dots A(n+1)B(n), & m > n. \end{cases} \quad (\text{J.5})$$

During trial j , system \mathbf{P} maps the input signal u_j to the measured output signal y_j , i.e., $\mathbf{y}_j = \mathbf{P}\mathbf{u}_j$, with \mathbf{u}_j and \mathbf{y}_j defined in (J.6) and (J.7), respectively.

$$\mathbf{u}_j = \begin{bmatrix} u_j^T(0) & u_j^T(1) & \cdots & u_j^T(N-1) \end{bmatrix}^T \quad (\text{J.6})$$

$$\mathbf{y}_j = \begin{bmatrix} y_j^T(0) & y_j^T(1) & \cdots & y_j^T(N-1) \end{bmatrix}^T \quad (\text{J.7})$$

$$\text{with} \quad u_j^T(k) = \begin{bmatrix} u_j^1(k) & \cdots & u_j^{q_i}(k) \end{bmatrix}$$

$$\text{and} \quad y_j^T(k) = \begin{bmatrix} y_j^1(k) & \cdots & y_j^{q_o}(k) \end{bmatrix}$$

In this paper we adopt a widely used norm optimal ILC update law [64,65]

$$\mathbf{u}_{j+1} = \mathbf{L}_u \mathbf{u}_j + \mathbf{L}_e \mathbf{e}_j \quad (\text{J.8})$$

where the error signal is comprised of the individual error signals from each independent axis as shown in (J.9).

$$\begin{aligned} \mathbf{e}_j &= \begin{bmatrix} e_j^T(0) & e_j^T(1) & \cdots & e_j^T(N-1) \end{bmatrix}^T \\ e_j^T(k) &= \begin{bmatrix} e_j^1(k) & \cdots & e_j^{q_o}(k) \end{bmatrix} \end{aligned} \quad (\text{J.9})$$

The error signal in (J.9) is defined as $\mathbf{e}_j = \mathbf{y}_r - \mathbf{y}_j$, where \mathbf{y}_r is the reference signal and is assumed iteration invariant. In (J.8), \mathbf{L}_u and \mathbf{L}_e are solutions to a quadratic optimization problem detailed in Section J.4. These lifted matrices are generally non-causal, time-invariant linear operators on the control and error signals, respectively.

Previous work in [120] introduced time-varying designs for these filters to address particular challenges at specific time intervals. The objective of this work is to implement a time-varying ILC design which couples the output performance of two dissimilar systems in the norm optimal framework. The coupling of multiple dissimilar systems through the output performance is presented in the following section.

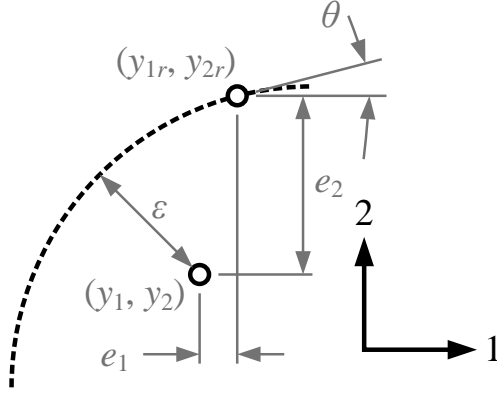


Figure J.1: 2D trajectory illustrating contour (ε) and individual errors (e_1, e_2) for two individual axes. These errors are defined with respect to the desired position (y_{1r}, y_{2r}) and the actual position (y_1, y_2) of a system defined with respect to the (**Axis – 1**, **Axis – 2**) coordinate frame. Linearized coupling gains ($c_1(k, \theta), c_2(k, \theta)$) at point in time (k) with respect to the tangent angle (θ) can be used to simplify the derivation of the contour error.

J.3 Coupling of Multiple Dissimilar Systems

When combining multiple individual systems or axes, one may couple these axes through a common desired output. For MIMO systems which consist of two or more individual axes, an additional error signal known as the contour error can be defined, as illustrated by the 2D example in Fig. J.1. Contour errors, ε , for a general class of MIMO systems can be defined with respect to the individual error signals, e_1, e_2, \dots, e_{q_o} , and trajectory dependent gains known as coupling gains [1, 36], $c_1(k, \theta), c_2(k, \theta), \dots, c_{q_o}(k, \theta)$, where k is the time interval from $k = 0, 1, \dots, N - 1$, θ is defined as the instantaneous angle of the reference trajectory with respect to the horizontal axis of the coordinate system [1], and $1, 2, \dots, q_o$ are the individual outputs.

When the class of MIMO systems described in Section J.2 is comprised of dissimilar axes, an additional weighting component should be added to the definition of the contour error to account for variations between the individual systems such as time-constants, system resonances, and system bandwidths. Previous work in [114] presented a coupled learning controller which incorporated an additional weighting

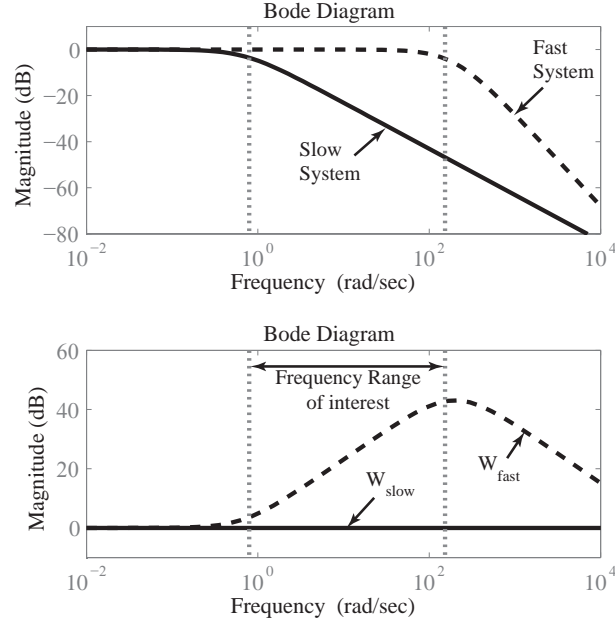


Figure J.2: Subplot 1: Example complementary sensitivity plot for a dynamically slow and fast system. Note the frequencies of interest lie between the cutoff frequencies of the two systems, respectively. In this frequency range the fast system can compensate for performance limitations from the slower system. Subplot 2: Weighting filters designed using the fast and slow system from subplot 1. Note that W_{fast} is calculated by dividing the complementary sensitivity of the fast system by the complementary sensitivity of the slow system. W_{slow} is set to one to reflect no additional weighting on the slow system.

gain into the derivation of the contour error in order to compensate for dominant time constant dissimilarities between two systems. This gain was applied across all frequencies, thereby indiscriminately increasing the weighting applied to the error signals at all frequencies. In this paper, we extend the idea of additional weighting through the introduction of a weighting filter. The weighting filter is used to compensate for dynamic inconsistencies across a range of frequencies when combining dynamically diverse systems. The weighting filters are derived from the relationship between the fast and slow systems in order to compensate for specific dynamic differences between the two systems.

Consider the complementary sensitivity plots of two dynamically diverse stable systems, subplot 1 of Fig. J.2. Both systems can easily handle low frequency signals,

while the high frequency response following the cutoff frequency of the faster system is unimportant since it can be categorized as either unattainable reference trajectories or noise. The frequency range of interest lies *between* the cutoff frequencies of the two systems. In this range, the low bandwidth of the slow system can be compensated for by the additional tracking capabilities of the fast system by coupling the two systems through the contour error. An important criteria for coupling dynamically diverse systems is signal equivalence. This requires that the signals are modified in order to balance the dynamics between the two systems. While a bandpass filter may target the frequency range of interest, the amplitude and shape of the filter should compensate for dynamic differences between the two systems. A more direct method for designing an appropriate weighting filter comes from comparing the dynamics of the two systems directly.

Dividing the complementary sensitivity of the fast system by the complementary sensitivity of the slow system results in a filter that maintains low frequency performance, amplifies signals in the frequency range of interest, and minimizes high frequency signals, as illustrated in subplot 2 of Fig. J.2. Amplifying the errors in this frequency range forces the faster system to respond, thereby relinquishing some of the performance strain from the slower system. Note that care must be taken to ensure minimal amplification of the high frequency signals. A low-pass filter may be added to the ratio of complementary sensitivities to ensure the filter attenuates high frequency noise. A general definition of the weighting filters for two SISO systems is provided in (J.10), where T represents the complimentary sensitivity of a system. A typical low-pass filter is provided in (J.11).

$$W_{fast} = \frac{T_{fast}}{T_{slow}} \cdot F_{lowpass}, W_{slow} = 1. \quad (\text{J.10})$$

$$F_{lowpass} = \frac{k}{z - \alpha_1}. \quad (\text{J.11})$$

Mathematically, for the two axes, represented as individual systems in Fig. J.3, the modified contour error can be defined as,

$$\varepsilon(k) = W_{fast}(\mathbf{q}) \cdot c_1(k, \theta) \cdot e_1(k) + W_{slow}(\mathbf{q}) \cdot c_2(k, \theta) \cdot e_2(k) \quad (\text{J.12})$$

$$\varepsilon(k) = C_Q(\mathbf{q}, k, \theta) \cdot e(k), \quad (\text{J.13})$$

where \mathbf{q} is the backwards time shift operator defined as $\mathbf{q}y(k) \equiv y(k-1)$ and $W_{fast}(\mathbf{q})$ and $W_{slow}(\mathbf{q})$ are the filters given in (J.10). Note that Axis-1 is assumed to be the fast system, while Axis-2 is defined as the slow system, respectively. Equation (J.13) illustrates that the weighting filters and coupling gains are combined into a single variable, $C_Q(\mathbf{q}, k, \theta)$. Linearized coupling gains $c_1(k, \theta)$ and $c_2(k, \theta)$ have the following format

$$c_1(k, \theta) = -\sin \theta(k); c_2(k, \theta) = \cos \theta(k), \quad (\text{J.14})$$

Note that the use of trajectory-dependent coupling gains leads to a time-varying controller. Figure J.3 provides a block diagram representation of two individual axes coupled together through CCILC.

The generalized structure for the norm optimal controller is given in the following section.

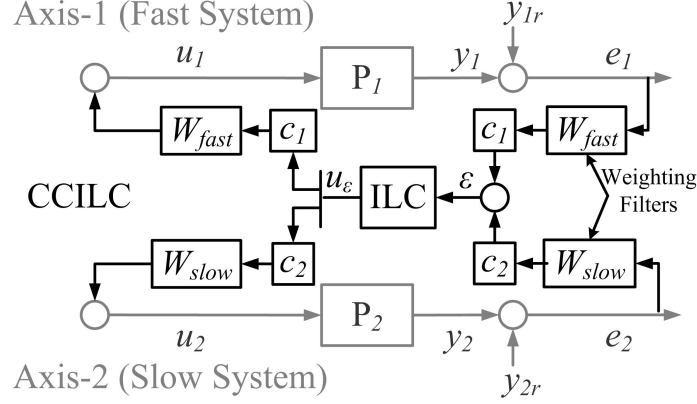


Figure J.3: Block diagram of a two-axis MIMO system in which the two independent SISO axes are coupled together via CCILC. $P_{(\cdot)}$ represents the plant sensitivity function defined as $\frac{G_{(\cdot)}}{1+G_{(\cdot)}k_{(\cdot)}}$, where $G_{(\cdot)}$ is the open-loop axis model and $k_{(\cdot)}$ is a feedback controller used to stabilize the open-loop axis. Note that $P_{(\cdot)}$ is different from the complementary sensitivity function, $T_{(\cdot)}$, defined as $\frac{G_{(\cdot)}k_{(\cdot)}}{1+G_{(\cdot)}k_{(\cdot)}}$. In this example, the fast and slow system descriptions are associated with Axis-1 and Axis-2, respectively.

J.4 Norm Optimal ILC

The norm optimal algorithm is designed to minimize a quadratic optimization problem [71, 75, 76],

$$\mathcal{J} = \mathbf{e}_{j+1}^T \mathbf{Q} \mathbf{e}_{j+1} + \mathbf{u}_{j+1}^T \mathbf{S} \mathbf{u}_{j+1} + (\mathbf{u}_{j+1} - \mathbf{u}_j)^T \mathbf{R} (\mathbf{u}_{j+1} - \mathbf{u}_j). \quad (\text{J.15})$$

where $(\mathbf{Q}, \mathbf{R}, \mathbf{S})$ are symmetric, positive definite real-valued matrices of appropriate dimension and $\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R}$ is positive definite. Applying the substitution $\mathbf{e}_{j+1} = \mathbf{e}_j - \mathbf{P}(\mathbf{u}_{j+1} - \mathbf{u}_j)$, differentiating \mathcal{J} with respect to \mathbf{u}_{j+1} , setting the result to zero, and rearranging the solution, yields the general norm optimal controller,

$$\mathbf{u}_{j+1} = \mathbf{L}_u \mathbf{u}_j + \mathbf{L}_e \mathbf{e}_j \quad (\text{J.16})$$

$$\mathbf{L}_u = (\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R})^{-1} (\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{R})$$

$$\mathbf{L}_e = (\mathbf{P}^T \mathbf{Q} \mathbf{P} + \mathbf{S} + \mathbf{R})^{-1} \mathbf{P}^T \mathbf{Q}.$$

For many designs, $(\mathbf{Q}, \mathbf{R}, \mathbf{S}) \triangleq (q\mathbf{I}, s\mathbf{I}, r\mathbf{I})$, with q, s, r real-valued positive scalars. In [120], a novel time-varying design for the \mathbf{Q} weighting matrix was introduced,

$$\mathbf{Q}^{tv} = \Sigma_Q \cdot [\mathbf{\Gamma}\mathbf{1}_Q + \mathbf{\Gamma}\mathbf{2}_Q \cdot \mathbf{C}_Q^T \mathbf{C}_Q] \quad (\text{J.17})$$

where the \mathbf{C}_Q matrix contains the terms used to define coupling between the individual error signals of the MIMO system, $\mathbf{\Gamma}\mathbf{1}_Q$ and $\mathbf{\Gamma}\mathbf{2}_Q$ refer to the weighting matrices applied to the coupled or individual error signals, and Σ_Q determines the overall weighting on the error signal compared to the control and change in control signals.

The coupling matrix \mathbf{C}_Q is derived from the definition of the contour error given in Eq. (J.13). Applying the lifted approach to Eq. (J.13) and writing the term \mathbf{C}_Q as the lifted form of $C_Q(\mathbf{q}, k, \theta)$, the coupling of the error terms is represented by the convolution matrix \mathbf{C}_Q using the combined impulse response data of the weighting filters, $\{W_{fast}, W_{slow}\}$, and the coupling gains, $\{c_1, c_2\}$ for the two SISO system example in Fig. J.3, respectively.

$$\mathbf{C}_Q = \begin{bmatrix} C_{0,0} & & 0 \\ \vdots & \ddots & \\ C_{N-1,0} & \cdots & C_{N-1,N-1} \end{bmatrix}. \quad (\text{J.18})$$

For MIMO systems comprised of two dynamically different SISO systems, $C_{m,n}$ contains the impulse response data of the weighting filters combined with the coupling gains in a two element vector format. Define W_{fast} and W_{slow} with the real-valued matrices, $\{A_{W_{fast}}, B_{W_{fast}}, C_{W_{fast}}, D_{W_{fast}}\}$ and $\{A_{W_{slow}}, B_{W_{slow}}, C_{W_{slow}}, D_{W_{slow}}\}$, respectively. Using these matrices, along with the vector descriptions of the coupling gains, $\mathbf{c}_1 = [c_1(0, \theta) \cdots c_1(N-1, \theta)]$ and $\mathbf{c}_2 = [c_2(0, \theta) \cdots c_2(N-1, \theta)]$, $C_{m,n}$ can be defined as,

$$C_{m,n} : \begin{cases} \begin{bmatrix} D_{W_{fast}} \mathbf{c}_{1m} & D_{W_{slow}} \mathbf{c}_{2m} \end{bmatrix}, & m = n \\ \begin{bmatrix} C_{W_{fast}} A_{W_{fast}}^{m-n-1} B_{W_{fast}} \mathbf{c}_{1n} & C_{W_{slow}} A_{W_{slow}}^{m-n-1} B_{W_{slow}} \mathbf{c}_{2n} \end{bmatrix}, & m > n. \end{cases} \quad (\text{J.19})$$

The matrices $\mathbf{\Gamma 1}_Q$ and $\mathbf{\Gamma 2}_Q$ refer to the amount of weighting applied to the coupled or individual signals, respectively. These matrices are of the form provided in (J.20) and (J.21), where the inner block diagonal matrices are shown for a 2 DOF system.

$$\mathbf{\Gamma 1}_Q = \begin{bmatrix} \begin{bmatrix} \gamma(1) & 0 \\ 0 & \gamma(1) \end{bmatrix} & & 0 \\ & \ddots & \\ 0 & & \begin{bmatrix} \gamma(N) & 0 \\ 0 & \gamma(N) \end{bmatrix} \end{bmatrix}, \quad (\text{J.20})$$

$$\mathbf{\Gamma 2}_Q = \begin{bmatrix} \begin{bmatrix} 1 - \gamma(1) & 0 \\ 0 & 1 - \gamma(1) \end{bmatrix} & & 0 \\ & \ddots & \\ 0 & & \begin{bmatrix} 1 - \gamma(N) & 0 \\ 0 & 1 - \gamma(N) \end{bmatrix} \end{bmatrix}. \quad (\text{J.21})$$

As can be seen from (J.20) and (J.21), the individual elements in $\mathbf{\Gamma 1}_Q$ and $\mathbf{\Gamma 2}_Q$ are related. Selecting $(\gamma(k) = 1)$ refers to all of the weighting being applied to the individual signals (nominal ILC design), while $(\gamma(k) = 0)$ results in only the coupled signals being weighted (CCILC design).

The gain matrix $\mathbf{\Sigma}_Q$ determines the overall weighting on the error signals with

respect to the control signals and change in control signals and is of the form shown in (J.22). Note that the inner diagonal matrix is illustrated for a 2 DOF system.

$$\Sigma_Q = \begin{bmatrix} \begin{bmatrix} \sigma_Q(1) & 0 \\ 0 & \sigma_Q(1) \end{bmatrix} & & & \\ & & 0 & \\ & & & \ddots \\ & 0 & & \begin{bmatrix} \sigma_Q(N) & 0 \\ 0 & \sigma_Q(N) \end{bmatrix} \end{bmatrix}. \quad (\text{J.22})$$

Recall from Section J.3 that the coupling gains are derived from the desired output trajectory, while the weighting filters are designed to compensate for dynamic differences between the axes. Using (J.17) and the more traditional format for \mathbf{S} and \mathbf{R} , $(\mathbf{S}, \mathbf{R}) \triangleq (s\mathbf{I}, r\mathbf{I})$, a modified cost function can be determined.

$$\mathcal{J} = \mathbf{e}_{j+1}^T \mathbf{Q}^{tv} \mathbf{e}_{j+1} + \mathbf{u}_{j+1}^T \mathbf{S} \mathbf{u}_{j+1} + (\mathbf{u}_{j+1} - \mathbf{u}_j)^T \mathbf{R} (\mathbf{u}_{j+1} - \mathbf{u}_j) \quad (\text{J.23})$$

An essential part of the design process involves determining weighting matrices for the cost function in (J.23). References [78, 120] present some guidelines for designing and tuning the matrices based on performance and robustness requirements. The work in this paper focuses on time-variation in the \mathbf{Q} matrix primarily due to a coupled output objective defined as the contour error (J.13). For further examples of systems which implement time-varying weighting matrices see [120].

J.5 System Setup

In the next two sections we institute a change of variables where Axis-1 is a y-axis positioning system and Axis-2 is an extrusion system.

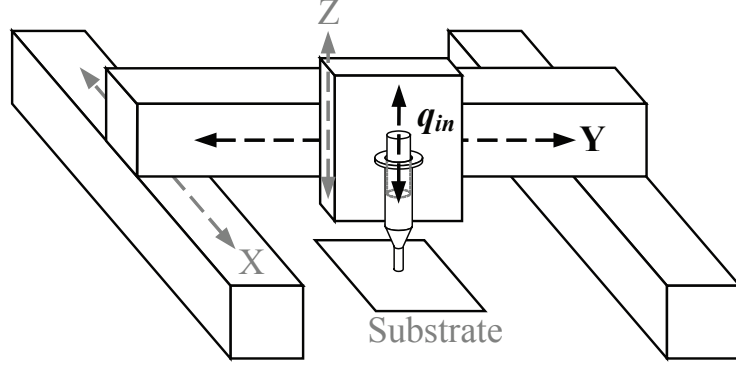


Figure J.4: Multi-axis robotic testbed with extrusion system included. Note that the design example used in this paper only couples together the extrusion system and the y-axis positioning system.

In order to explore the performance benefits of combining two dissimilar SISO systems or axes into a MIMO format, time-varying (CCILC) and time-invariant (Nominal ILC) norm optimal learning controllers are implemented on the y-axis positioning and extrusion systems of a μ -RD system, Fig. J.4 and Fig. J.5 respectively. The primary objective in μ -RD is dimensional accuracy of the extruded build material. The extrusion and positioning systems are drastically different, with extrusion system performance measured in volume and positioning system performance measured in distance. The positioning system has a bandwidth that is more than 100 times faster than the extrusion system. Here, we show that the proposed control method exploits the disparity in axes performance, incongruently penalizing the fast positioning axis error in certain frequencies.

The input for the y-axis is amplifier current and the output is axis position, y_{out} . The input to the extrusion system is plunger displacement rate, q_{in} , and the output is build material volumetric flowrate, q_{out} .

Dynamic models of the two axes were developed in [121] and [20]. Numerical values for the y-axis plant model, G_y in Eq. (J.24), along with a stabilizing feedback controller, k_{Gy} in Eq. (J.25), can be found in the Appendix A. The extrusion system, P_q presented in Eq. (J.26), is open-loop stable and therefore only requires open-loop

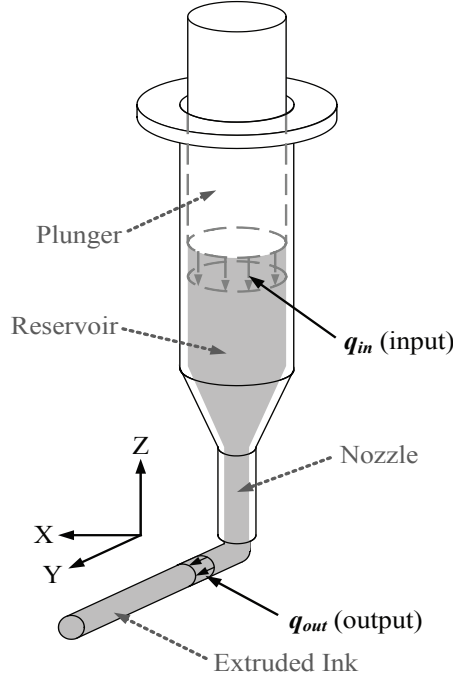


Figure J.5: Extrusion system for material deposition

input signals.

$$G_y(z) = \frac{K(z + \alpha_1)(z^2 - \alpha_2 z + \alpha_3)(z^2 - \alpha_4 z + \alpha_5)}{(z - \beta_1)(z - 1)(z^2 - \beta_2 z + \beta_3)(z^2 - \beta_4 z + \beta_5)}. \quad (\text{J.24})$$

$$k_{Gy}(z) = \frac{K(z - \alpha_1)(z - \alpha_2)(z - \alpha_3)}{(z - \beta_1)(z - \beta_2)(z - \beta_3)}. \quad (\text{J.25})$$

$$P_q(z) = \frac{0.00019766}{z - 0.9998} \quad (\text{J.26})$$

The MIMO system is subject to a combined trajectory which integrates material extrusion with linear stage positioning. Explicitly stated, the y-axis proceeds at a constant velocity while the extrusion system has a pulsed trajectory. The combination of these reference trajectories corresponds to the extrusion of a long cylinder of material deposited on a flat substrate, Fig. J.6. The primary objective is to achieve sharp and accurately placed transitions from no flowrate to a nominal flowrate with

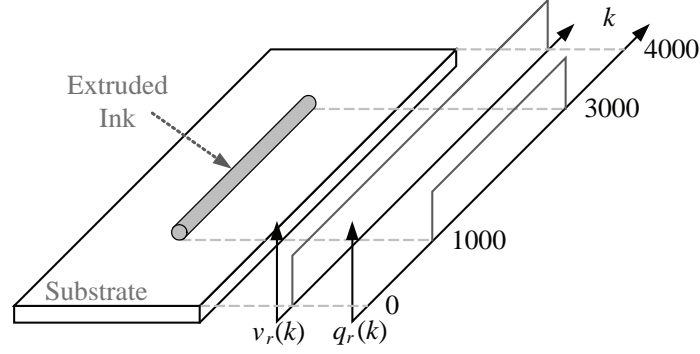


Figure J.6: Diagram of the desired fabricated structure and the corresponding reference trajectories. Reference trajectories for the two axes are the desired flowrate, q_r , and desired y-axis position, y_r . Position reference is shown in terms of axis velocity, $v_r(k) = (y_r(k) - y_r(k-1))/0.001$, where 0.001 is the sample time.

consistent nominal flowrate regulation.

Controller Design

The objective of this work is to pursue a primary performance objective by coupling two dissimilar axes through the desired output. The coupling of the output signals translates to a coupling of the error signals, as illustrated in the cost function of Eq. (J.23). The coupling between the signals results from the combination of coupling gains, $(c_y(k, \theta), c_q(k, \theta))$, and weighting filters, (W_{fast}, W_{slow}) in Fig. J.3. The coupling gains are derived from the reference trajectory (Fig. J.6) using the definition provided in Eq. (J.14). The trajectories in Fig. J.6 present an interesting challenge that we have addressed. Here, $c_q(k, \theta) = 1$ and $c_y(k, \theta) = 0$ for all $k = 0, 1, \dots, N-1$ with the exception of the start ($k = 1000$) and stop ($k = 3000$) locations of the q_r desired flowrate; this correlates to a single sample number for each location. In order to force additional compensatory movement from the y-axis, c_y must have a nonzero value for longer than one sample point. Therefore, the lifted time vectors of the coupling gains are filtered using a Gaussian filter with a bandwidth of 3 Hz. The resulting vectors are illustrated in Fig. J.7. This unique problem will be seen with

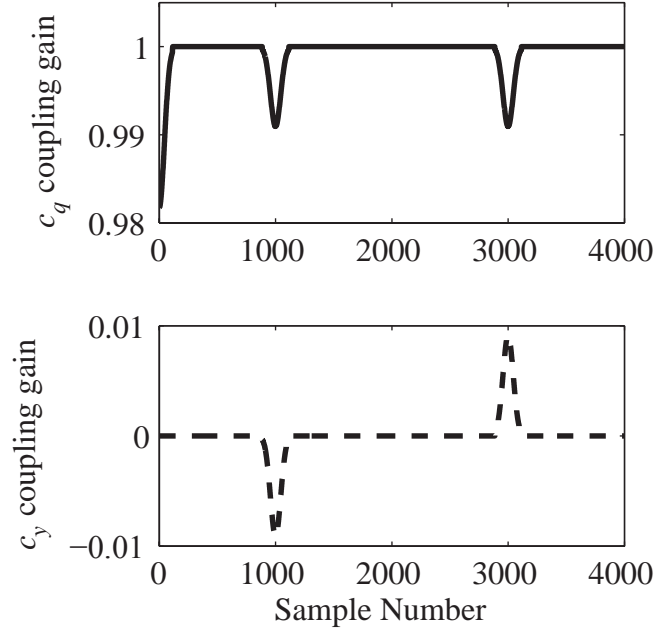


Figure J.7: Coupling gains used in the derivation of the contour error. Note that the vectors have been filtered using a Gaussian filter with a 3 Hz bandwidth in order to ensure the ability to force compensatory action from the y-axis.

any reference trajectory that contains discontinuities.

For simplicity, the weighting filters have been redefined as $W_y \triangleq W_{fast}$ and $W_q \triangleq W_{slow}$, respectively. The weighting filters, W_y and W_q (Fig. J.8), for this example are of the form,

$$W_y = \frac{T_y}{P_q} \cdot F_{lowpass}; W_q = 1, \quad (\text{J.27})$$

$$\text{with } F_{lowpass} = \frac{0.3297}{z - 0.6703}, \quad (\text{J.28})$$

where P_q is the complimentary sensitivity (open-loop stable model) of the slower extrusion axis and $T_y = \frac{G_y k_{Gy}}{1 + G_y k_{Gy}}$ is the complimentary sensitivity for the faster y-positioning axis, respectively. Note that for this example, open-loop system stability results in P_q replacing T_{slow} in the calculation of W_y , (J.10). The ratio of the two models is used as a filter which weights the faster axis more heavily during the

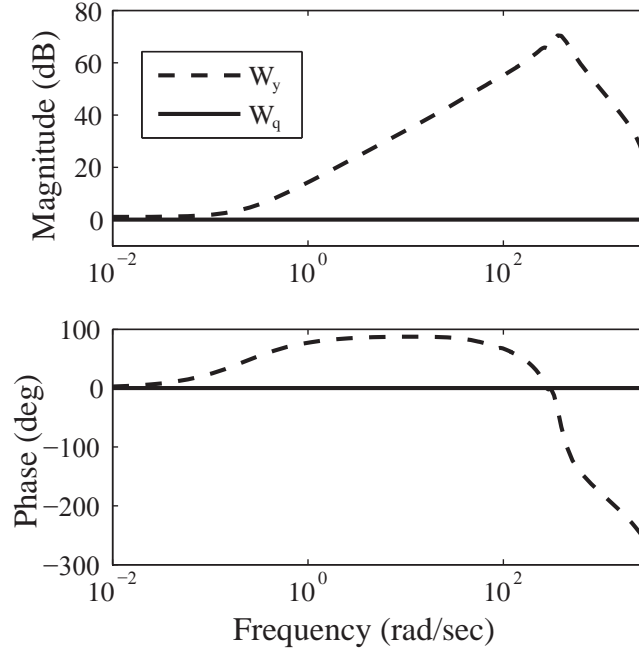


Figure J.8: Weighting filters used to compensate for the dissimilar dynamics between the positioning stage and the extrusion system. Note that only the fast system requires a weighting filter, W_y , as demonstrated by a weighting filter of 1 for the slow system, W_q .

frequency range between the cutoff frequencies of the slow and fast axes. A lowpass filter combined with this ratio minimizes the amplification of any high frequency signals. The shape of this weighting filter forces the faster system to assume some of the performance load for the slower system, while maintaining robustness in the presence of high frequency noise or disturbances.

Learning filters of the form described in (J.16), with \mathbf{Q} replaced by the time-varying weighting matrix of the form in (J.17), were designed using the methodology detailed in [120]. Heuristic tuning of the \mathbf{S} and \mathbf{R} weighting matrix gains resulted in the constant gain values ($s_y = 0.01, s_q = 0.001, r_y = 0.02, r_q = 0.01$) for the nominal ILC controllers and ($s_y = 0.01, s_q = 0.0005, r_y = 0.02, r_q = 0.01$) for the coupled CCILC controllers. The weighting gains for \mathbf{Q}^{tv} are ($\gamma(k) = 1, 1 - \gamma(k) = 0, \sigma_Q(k) = 2$) for nominal ILC (individual axis) control and ($\gamma(k) = 0, 1 - \gamma(k) = 1, \sigma_Q(k) = 2$) for CCILC (coupled axis) control, for all $k = 0, 1, \dots, N - 1$. The values for $\sigma_Q(k)$

were chosen to maximize the system performance, while ensuring convergence and robustness.

Extrusion Trials

As a complement to the experimental data, each controller is evaluated visually in a manufacturing motivated example. Advanced architecture structures built by μ RD require two materials to be directly opposed to each other to create distinct material, and hence physical property, domains. Here we evaluate the potential of Nominal ILC and CCILC to create these features by testing a butt-weld of two line segments five times for each controller; essentially two segments of the shape in Fig. J.6 oriented end-to-end. Ideally, each material transition should have minimal overlap so that the material property change is abrupt and should maintain a constant line width. This simple test uses one material and the evaluation metric is the regulation of line width. A multi-material example would simply require changing materials and identifying material specific Nominal ILC and CCILC signals for the y-axis and extrusion axis; displayed previously in the extrusion system axis in [122].

J.6 Results

The generalized CCILC controller introduced in Section J.4 is applied to the μ RD system. Figures J.9, J.10 and J.11 display signals for iteration 15. The performance of the combined system is hindered by the extrusion system performance, in which plunger displacement rate is limited to $\pm 20 \text{ mm}^3/\text{sec}$ to minimize actuator wear. When applying ILC to the extrusion axis and y-axis independently, the coupled output of the MIMO system poorly approximates the reference signal, Fig. J.11. Figure J.11 also shows the performance for the y-axis controlled by feedback and the extrusion axis in open-loop. Labeled as 'standard' in the figure, this control methodology

is the common practice in μ RD and yields poor flowrate modulation.

The CCILC approach penalizes the contribution of the y-axis to the contour error, yielding a coupled control signal that modifies the y-axis trajectory, Fig. J.9, to compensate for poor extrusion system performance. The feedforward input to the extrusion system remains relatively unchanged, Fig. J.10, thereby maintaining sub-threshold actuation inputs. Qualitatively, the y-axis briefly slows down and then accelerates into the desired position of the flowrate pulse and dwells momentarily to accumulate material volume, Fig. J.9. Then the y-axis, driven by the feedback controller, accelerates out of the dwell to minimize its individual axis tracking. This coupled axis behavior is intuitive in that it spatially positions the extrusion system in the correct location for the flowrate profile that is achievable by the extrusion system. Similar axes behavior has been designed on a similar system via *ad hoc* reference shaping [123]; however the method presented here achieves axes coordination automatically. The coordination of axes leads to a 14% average reduction in the root mean squared (RMS) tracking of the converged contour error, as compared to ILC applied to each system independently.

The primary objectives of the μ RD process are the sharpness of the flowrate pulse and constant flowrate, which is analogous to accurate material starting and stopping, and constant diameter of the extruded material. Figure J.11 shows a contour tracking plot. Here CCILC yields the quickest transition from zero flowrate to a nominal flowrate and the longest duration of constant flowrate. The transition sharpness and increased duration of constant flowrate of the CCILC system are illustrated in the experimental images provided in the right-hand-side of Fig. J.11.

Lastly, CCILC and Nominal ILC are evaluated through a series of extrusion trials, detailed in Subsection J.5. The improved material flowrate transition sharpness seen in Fig. J.11 is realized in the ability to adjoin materials with minimal material overlap, Fig. J.12. For the ILC case, the inability to precisely transition from zero flowrate

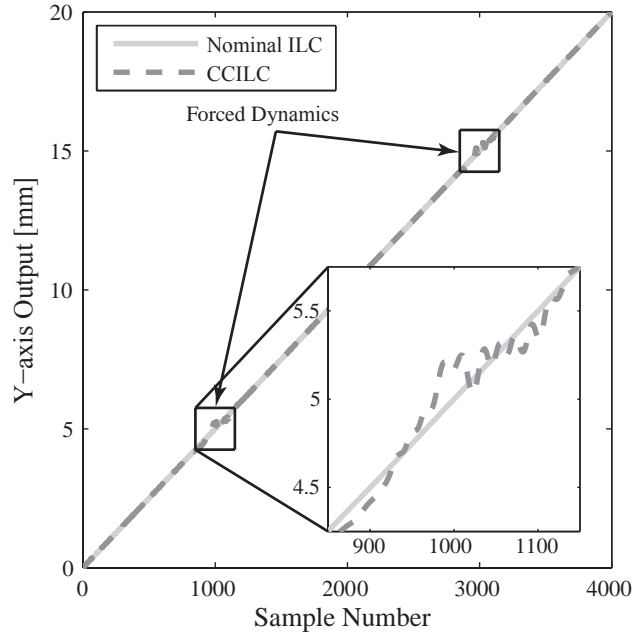


Figure J.9: Y-axis output for the Nominal ILC and CCILC cases. Axes coupling forces additional dynamics in the response to compensate for extrusion system inadequacies.

to the nominal flowrate leads to material overlap at the line segment abutment and therefore swelling and poor line width regulation. CCILC improves the flowrate performance and the two line segments can be placed neatly opposed to each other without material overlap. The implications of this improved CCILC performance are the ability to more closely achieve the ideal structure discussed in Section J.5; namely discrete divisions of materials in advanced architecture structures without a significant overlap section.

J.7 Concluding Remarks and Future Direction

In this paper, we investigate the coupling of dynamically dissimilar axes in manufacturing systems with a coupled primary objective. The key contributions in this work include 1) the introduction of a coupled output objective for dissimilar systems that incorporates the dynamic differences of the systems into the derivation of the desired

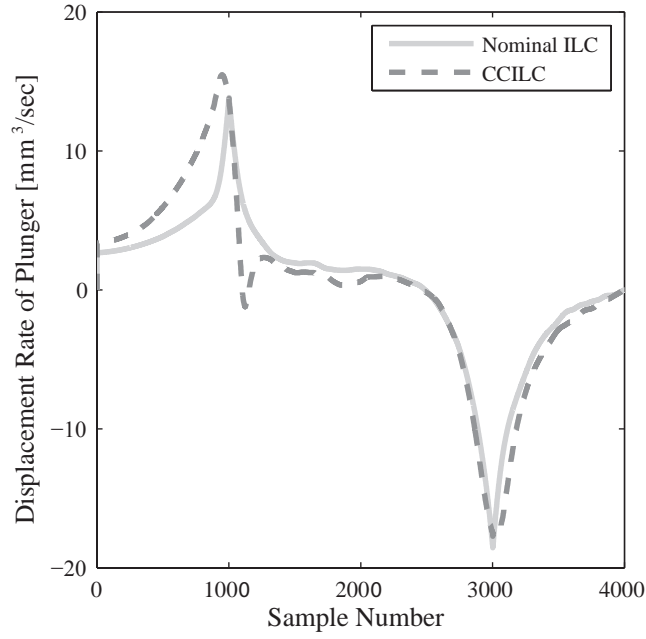


Figure J.10: Input signals for the extrusion system for the Nominal ILC and CCILC cases. Despite fairly consistent input signals that are within the input signal constraints, the CCILC controller results in much sharper material start and stop features as illustrated in Fig. J.11

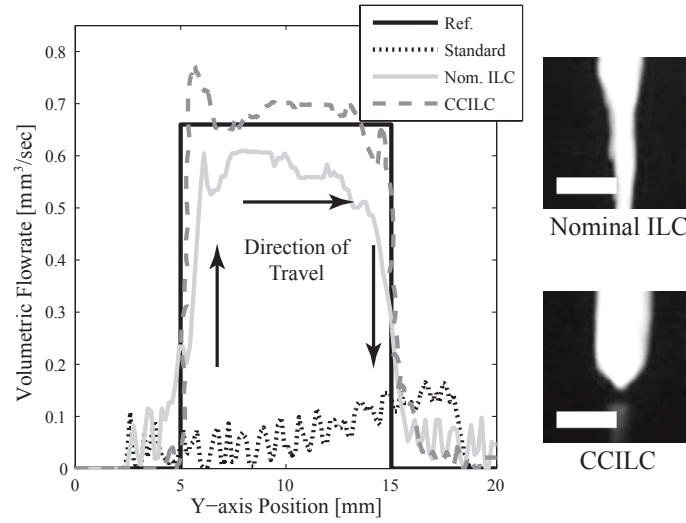


Figure J.11: Contour plot of the output signals for the μ RD system, along with experimental images of the start positions ($k=1000$) for the Nominal ILC and CCILC cases. Note the improved sharpness of the deposited material for the CCILC case. The scale bar is 0.5 mm.

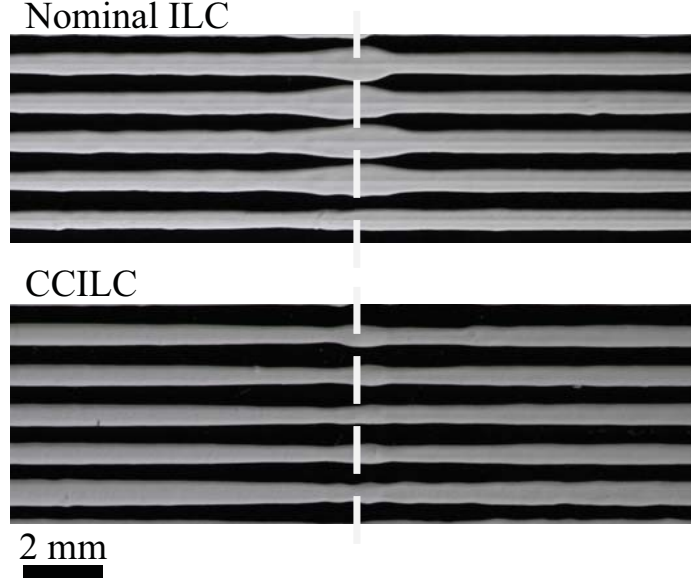


Figure J.12: Extrusion trials of the Nominal ILC and CCILC depositing two adjoined lines of material. Adjoining location denoted by white dashed line. CCILC minimizes material overlap by depositing a structure with sharp material starts and stops.

objective, 2) the development of a novel framework for designing learning controllers which minimize a coupled objective for dissimilar systems, and 3) validation of this controller through experimental testing.

The traditional CCILC structure was adapted to include weighting filters that penalize contributions to a coupled objective, defined as the contour error, within a certain frequency range. This framework engages the underutilized high performance axis to assist low performance axes. In order to demonstrate the potential performance improvements obtained by coupling the output of the two dissimilar axes, a CCILC controller was applied experimentally to a μ RD system. This MIMO system consists of a positioning system and an extrusion system that is constrained by actuation limits. The generalized CCILC approach transfers actuation load from the extrusion system to the underutilized positioning system, thereby modifying the trajectory of the positioning system to compensate for extrusion system inadequacies. The experimental results display an average reduction of 14% in the RMS of the contour error using CCILC as compared to ILC designs. Extrusion trials illus-

trate that the additional material extrusion accuracy realized using CCILC leads to improved performance when joining multiple segments of material. Future work will investigate how well this improved extrusion capability translates to more advanced multi-material structures, as well as explore additional design approaches for optimizing the extrusion performance of the combined system.

References

- [1] Y. Koren and C. Lo, “Variable-gain cross-coupling controller for contouring,” *CIRP Annals*, vol. 40, pp. 371–374, 1991.
- [2] K. Barton and A. Alleyne, “A cross-coupled Iterative Learning Control design for precision motion control,” *IEEE Transactions on Control Systems Technology*, vol. 16, pp. 1218–1231, 2008.
- [3] T. Anderson, N. Golden, J. Smith, and F. Conti, “Novint: 3d haptics technology software,” R&D 100 Award Entry, 2007.
- [4] D. Sumpter and S. Pratt, “A modelling framework for understanding social insect foraging,” *Behav Ecol Sociobiol*, vol. 53, pp. 131–144, 2003.
- [5] S. Gross, S. Aron, J. L. Deneubourg, and J. M. Pasteels, “Self-organized shortcuts in the argentine ant,” *Naturwissenschaften*, vol. 76, pp. 579–581, 1989.
- [6] D. Sumpter, “The principles of collective animal behaviour,” *Phil. Trans. R. Soc. B*, vol. 361, pp. 5–22, 2006.
- [7] J. Toner and Y. Tu, “Flock, hers, and school: A quantitative theory of flocking,” *Physical Review E*, vol. 58, pp. 4828–4858, 1998.
- [8] D. Cruz, J. McClintock, B. Perteet, O. Orqueda, Y. Cao, and R. Fierro, “Decentralized cooperative control: A multivehicle platform for research in networked embedded systems,” *IEEE Control Systems Magazine*, pp. 58–78, 2007.
- [9] C. Kube and E. Bonabeau, “Cooperative transport by ants and robots,” *Robotics and Autonomous Systems*, vol. 30, pp. 85–101, 2000.
- [10] K. Sugawara and M. Sano, “Cooperative acceleration of task performance: Foraging behavior of interacting multi-robots system,” *Physica D*, vol. 100, pp. 343–354, 1997.
- [11] T. Balch and R. Arkin, “Behavior-based formation control for mulit-robot teams,” *IEEE Transactions on Robotics and Automation*, 1999.
- [12] R. D’Andrea and G. Dullerud, “Distributed control design for spatially interconnected systems,” *IEEE Transactions on automatic control*, vol. 48, pp. 1478–1495, 2003.

- [13] S. Hirata and K. Fuwa, "Chimpanzees (pan troglodytes) learn to act with other individuals in a cooperative task," *Primates*, vol. 48, pp. 13–21, 2007.
- [14] T. Clutton-Brock and G. Parker, "Punishment in animal societies," *Nature*, pp. 209–216, 1995.
- [15] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, pp. 123–140, 1984.
- [16] D. Kim and S. Kim, "An Iterative Learning Control method with application for CNC machine tools," *IEEE Transactions on Industry Applications*, vol. 32, pp. 66–72, 1996.
- [17] M. Mezghani, G. Roux, M. Cabassud, M. Le Lann, B. Dahhou, and G. Casamatta, "Application of iterative learning control to an exothermic semi-batch chemical reactor," *IEEE Trans. Control Systems Technology*, vol. 10, pp. 822–834, 2002.
- [18] B. Wu, E. Poh, and G. Xu, "Satellite formation keeping via real-time optimal control and Iterative Learning Control," in *Proc. of IEEE Aerospace Conference*, 2009.
- [19] H.-S. Ahn and Y. Chen, "Iterative Learning Control for multi-agent formation," in *ICROS-SICE Conference*, 2009.
- [20] D. Bristow and A. Alleyne, "A high precision motion control system with application to microscale robotic deposition," *IEEE Transactions on Control Systems Technology*, vol. 16, pp. 1008–1020, 2006.
- [21] M. Norrlöf, "An adaptive Iterative Learning Control algorithm with experiments on an industrial robot," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 245–251, 2002.
- [22] Z. Liu, M. Ang Jr, and W. Seah, "Reinforcement learning of cooperative behavior for multi-robot tracking of multiple moving targets," in *International Conference on Intelligent Robots and Systems*, 2005, pp. 1220–1225.
- [23] H. Fang, R. Fan, B. Thuilot, and P. Martinet, "Trajectory tracking control of farm vehicles in presence of sliding," *Robotics and Autonomous Systems*, vol. 54, pp. 828–839, 2006.
- [24] J.-U. Park, M. Hardy, S. J. Kang, K. Barton, K. Adair, D. Mukhopadhyay, C. Y. Lee, M. S. Strano, A. G. Alleyne, J. G. Georgiadis, P. M. Ferreira, and J. A. Rogers, "High-resolution Electrohydrodynamic jet printing," *Nature Materials*, vol. 6, pp. 782 – 789, August 2007.

- [25] D.-Y. Chen, J.-C. Lee, Y.-S. Shin, S.-E. Park, T.-U. Yu, Y.-J. Kim, and J. Hwang, "Structuring of conductive silver line by electrohydrodynamic jet printing and its electrical characterization," *Journal of Physics: Conference Series*, vol. 142, pp. 012039(1)–(4), 2008.
- [26] J. Park, J. Lee, U. Paik, Y. Lu, and J. Rogers, "Nanoscale patterns of oligonucleotides formed by electrohydrodynamic jet printing with applications in biosensing and nanomaterials assembly," *Nano Letters*, vol. 8, no. 12, pp. 4210–4216, 2008.
- [27] D. Wang, M. Edirisinghe, and S. Jayasinghe, "Solid freeform fabrication of thin-walled ceramic structures using an electrohydrodynamic jet," *Journal of the American Ceramic Society*, vol. 89, no. 5, pp. 1727–1729, 2006.
- [28] D. Lee, Y. Shin, S. Park, T. Yu, and J. Hwang, "Electrohydrodynamic printing of silver nanoparticles by using focused nanocolloid jet," *Applied Physics Letters*, vol. 90, pp. 0819051–0819053, 2007.
- [29] A. Sullivan and S. Jayasinghe, "Development of a direct three-dimensional biomicrofabrication concept based on electrospraying a custom made siloxane sol," *Biomicrofluidics*, vol. 1, pp. 0341031–03410310, 2007.
- [30] H. K. Choi, J.-U. Park, O. O. Park, P. M. Ferreira, J. G. Georgiadis, and J. A. Rogers, "Scaling laws for jet pulsations associated with high-resolution electrohydrodynamic printing," *Applied Physics Letters*, vol. 92, no. 12, p. 123109, 2008. [Online]. Available: <http://link.aip.org/link/?APL/92/123109/1>
- [31] K. Barton, S. Mishra, A. Alleyne, J. Rogers, and P. Ferreira, "A desktop electrohydrodynamic jet printing system," *IFAC Mechatronics*, vol. 20, pp. 611–616, 2010.
- [32] —, "High resolution sensing and control of electrohydrodynamic jet printing," *submitted to Control Engineering Practice*, 2010.
- [33] S. Mishra, K. Barton, A. Alleyne, J. Rogers, and P. Ferreira, "High speed drop-on-demand printing with a pulsed electrohydrodynamic jet," *Journal of Micromechanics and Microengineering*, vol. 20, pp. 095026:1–8, 2010.
- [34] Y. Koren, "Cross-coupled biaxial computer control for manufacturing systems," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 102, pp. 265–272, 1980.
- [35] P. Kulkarni and K. Srinivasan, "Optimal contouring control of multi-axis feed drive servomechanisms," *ASME J. of Engineering for Industry*, vol. 111, pp. 140–148, 1989.
- [36] K. Srinivasan and P. Kulkarni, "Cross-coupled control of biaxial feed drive servomechanisms," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 112, pp. 225–232, 1990.

- [37] D. Bristow, M. Tharayil, and A. Alleyne, “A survey of Iterative Learning Control,” *Control Systems Magazine*, vol. 26, pp. 96–114, 2006.
- [38] H.-S. Ayn, Y. Chen, and K. Moore, “Iterative Learning Control: Brief survey and categorization,” *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 37, pp. 1099–1121, 2007.
- [39] Y. Chen and C. Wen, *Iterative Learning Control: Convergence, Robustness, and Applications*. London: Springer, 1999.
- [40] S. Kawamura and N. Sakagami, “Analysis on dynamics of underwater robot manipulators basing on iterative learning control and time-scale transformation,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2002, pp. 1088–1094.
- [41] C. Giessen, Q. Zou, and S. Devasia, “Inversion-based precision-positioning of inertial reaction devices,” in *Proc. of the American Control Conference*, 2004, pp. 3788–3793.
- [42] Y. Chen, C. Wen, J.-X. Xu, and M. Sun, “High-order iterative learning identification of projectile’s aerodynamic drag coefficient curve from radar measured velocity data,” *IEEE Trans. Control Systems Technology*, 1998.
- [43] R. Horowitz, “Learning control of robot manipulators,” *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, vol. 115, pp. 402–411, 1993.
- [44] K. L. Moore, *Iterative Learning Control for Deterministic Systems*. Springer-Verlag, 1993.
- [45] K. Moore, M. Dahley, and S. Bhattacharyya, “Iterative Learning Control: a survey and new results,” *Journal of Robotic Systems*, vol. 9, pp. 563–594, 1992.
- [46] J.-X. Xu and Y. Tan, *Linear and Nonlinear Iterative Learning Control*. Berlin: Springer, 2003.
- [47] D. Hoelzle, A. Alleyne, and A. Wagoner Johnson, “Micro-robotic deposition guidelines by a design of experiments approach to maximize fabrication reliability for the bone scaffold application,” *Acta Biomater*, vol. 4, pp. 897–912, 2008.
- [48] J. Cesarano III, J. Dellinger, M. Saavedra, and D. Gill, “Customization of load-bearing hydroxyapatite lattice,” *International Journal of Applied Ceramic Technology*, 2005.
- [49] Y. Tang, R. Landers, and S. Balakrishnan, “Hierarchical optimal force-position-contour control of machining processes: Part I - controller methodology,” in *Proc. of the American Control Conference*, 2005, pp. 4506–4511.

- [50] —, “Hierarchical optimal force-position-contour control of machining processes: Part II - illustrative example,” in *Proc. of the American Control Conference*, 2005, pp. 4512–4517.
- [51] K. Barton and A. Alleyne, “A norm optimal approach to time-varying ILC with application to a multi-axis robotic testbed,” *IEEE Transactions on Control Systems Technology*, pp. 1–15, 2009.
- [52] —, “Precision coordination and motion control of multiple systems via iterative learning control,” in *Proc. of IEEE American Control Conference*, 2010.
- [53] S. Hubbard, P. Babak, S. Sigurdsson, and K. Magnusson, “A model of the formation of fish schools and migrations of fish,” *Ecological Modelling*, vol. 174, pp. 359–374, 2004.
- [54] M. Porfiri, G. Roberson, and D. Stilwell, “Tracking and formation control of multiple autonomous agents: A two-level consensus approach,” *Automatica*, vol. 43, pp. 1318–1328, 2007.
- [55] D. Klein, C. Matlack, and K. Morgansen, “Cooperative target tracking using oscillator models in three dimensions,” in *Proc. of American Control Conference*, 2007, pp. 2569–2575.
- [56] D. Lee, “Semi-autonomous teleoperation of multiple wheeled robots over the internet,” in *Proc. of ASME Dynamic Systems and Control Conference*, 2008, pp. 147–154.
- [57] E. Lalish, K. Morgansen, and T. Tsukamaki, “Decentralized reactive collision avoidance for multiple unicycle-type vehicles,” in *Proc. of American Control Conference*, 2008, pp. 5055–5061.
- [58] A. Huth and C. Wissel, “The simulation of the movement of fish schools,” *Journal of Theoretical Biology*, vol. 156, pp. 365–385, 1992.
- [59] D. Morgan and I. Schwartz, “Dynamic coordinated control laws in multiple agent models,” *Physics Letters A*, vol. 340, pp. 121–131, 2005.
- [60] Y. Chen and Z. Wang, “Formation control: a review and new consideration,” in *Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, 2005, pp. 3181–3186.
- [61] R. Longman, “Iterative Learning Control and repetitive control for engineering practice,” *International Journal of Control*, vol. 73, pp. 930–954, 2000.
- [62] M. Norrlöf and S. Gunnarsson, “Time and frequency domain convergence properties in Iterative Learning Control,” *International Journal of Control*, vol. 75, pp. 1114–1126, 2002.

- [63] M. Phan and R. Longman, “A mathematical theory of learning control for linear discrete multivariable systems,” in *Prod. of the AIAA/AAS Astrodynamics Specialist Conference*, 1988, pp. 740–746.
- [64] J. van de Wijdeven and O. Bosgra, “Residual vibration suppression using hankel Iterative Learning Control,” *Int. J. of Robust and Nonlinear Control*, vol. 18, pp. 1034–1051, 2008.
- [65] D. Bristow and B. Hencsey, “A q, l factorization of norm-optimal Iterative Learning Control,” in *Proc. of IEEE Conference on Decision and Control*, 2008, pp. 2380–2384.
- [66] P. Goldsmith, “On the equivalence of causal LTI iterative learning control and feedback control,” *Automatica*, vol. 38, pp. 703–708, 2002.
- [67] K. Ogata, *Modern Control Engineering Fourth Edition*. Prentice Hall, 2002.
- [68] D. Bristow, K. Barton, and A. Alleyne, *The Control Handbook, Second Edition: Iterative Learning Control*. CRC Press, Inc., 2010.
- [69] M. Norrlöf and S. Gunnarsson, “Experimental comparison of some classical Iterative Learning Control algorithms,” *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 636–641, 2002.
- [70] D. Gorinevsky, “Loop shaping for iterative control of batch processes,” *IEEE Control System Magazine*, vol. 22, pp. 55–65, 2002.
- [71] J. Lee, K. Lee, and W. Kim, “Model-based Iterative Learning Control with a quadratic criterion for time-varying linear systems,” *Automatica*, vol. 36, pp. 641–657, 2000.
- [72] T. Donkers, J. van de Wijdeven, and O. Bosgra, “Robustness against model uncertainties of norm optimal Iterative Learning Control,” in *Proc. of the American Control Conference*, Seattle, WA, USA, 2008, pp. 4561–4566.
- [73] D. Bristow, “Weighting matrix design for robust monotonic convergence in norm optimal Iterative Learning Control,” in *Proc. of IEEE American Control Conference*, 2008, pp. 4554–4560.
- [74] M. Steinbuch, O. Bosgra, and Group, “Personal communication,” Technical University of Eindhoven, Eindhoven, The Netherlands, 2007.
- [75] M. Norrlöf and S. Gunnarsson, “On the design of ILC algorithms using optimization,” *Automatica*, vol. 37, pp. 2011–2016, 2001.
- [76] N. Amann, D. Owens, and E. Rogers, “Iterative Learning Control for discrete-time systems with exponential rate of convergence,” *IEE Proceedings: Control Theory and Applications*, vol. 143, pp. 217–224, 1996.

- [77] J. Frueh and M. Phan, “Linear quadratic optimal learning control LQL,” *Int. Journal of Control*, 2000.
- [78] K. Barton, J. van de Wijdeven, A. Alleyne, M. Steinbuch, and O. Bosgra, “Norm optimal cross-coupled Iterative Learning Control,” in *Proc. of IEEE Conference on Decision and Control*, 2008, pp. 3020–3025.
- [79] G. Oriolo, S. Panzieri, and G. Ulivi, “Learning optimal trajectories for non-holonomic systems,” *International Journal of Control*, vol. 73, pp. 980–991, 2000.
- [80] D. Bristow, A. Alleyne, and M. Tharayil, “Optimizing learning convergence speed and converged error for precision motion control,” *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 130, p. 054501, 2008.
- [81] B. Hennen, I. Rotariu, and M. Steinbuch, “Time-frequency analysis of position-dependent dynamics in a motion system with ILC,” in *Proc. of the American Control Conference*, Seattle, WA, USA, July 11-13 2007.
- [82] M. Wassink, M. van de Wal, C. Scherer, and O. Bosgra, “LPV control for a wafer stage: beyond the theoretical solution,” *Control Engineering Practice*, vol. 13, pp. 231–245, 2005.
- [83] M. van de Wal, G. van Baars, F. Sperling, and O. Bosgra, “Multivariable H_∞/μ feedback control design for high-precision wafer stage motion,” *Control Engineering Practice*, vol. 10, pp. 739–755, 2002.
- [84] J. Dhupia, A. Powalka, A. Ulsoy, and R. Katz, “Effect of a nonlinear joint on the dynamic performance of a machine tool,” *ASME Journal of Manufacturing Science and Engineering*, vol. 129, pp. 943–950, 2007.
- [85] A. Sebastian and S. Salapaka, “Design methodologies for robust nanopositioning,” *IEEE Transactions on Control Systems Technology*, vol. 13, pp. 868–876, 2005.
- [86] J. Prezelj and M. Čudina, “Noise as a signal for on-line estimation and monitoring of welding process,” *ACTA Acustica United with Acustica*, vol. 89, pp. 280–286, 2003.
- [87] K. Barton, D. Hoelzle, A. Alleyne, and A. Wagoner Johnson, “Cross coupled Iterative Learning Control of systems with dissimilar dynamics: Design and implementation for manufacturing applications,” *International Journal of Control*, 2010.
- [88] M. Liggins, C. Chong, I. Kadar, M. Alford, V. Vannicola, and S. Thmopoulos, “Distributed fusion architectures and algorithms for target tracking,” *Proceedings of the IEEE*, vol. 85, pp. 95–107, 1997.

- [89] R. Beard, R. McLain, D. Nelson, D. Kingston, and D. Johanson, "Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs," *Proceedings of the IEEE*, vol. 94, pp. 1306–1324, 2006.
- [90] Y. Chen and Z. Wang, "Formation control: A review and a new consideration," *Intelligent Robots and Systems*, pp. 3181–3186, 2005.
- [91] V. Gough, "Contribution to discussion of papers on research in automobile stability, control and tyre performance," in *Proc. Auto Div. Inst. Mech. Eng.*, 1956, pp. 392–394.
- [92] D. Stewart, "A platform with six degrees of freedom: a new form of mechanical linkage which enables a platform to move simultaneously in all six degrees of freedom developed by elliott-automation," *Aircraft Engineering and Aerospace Technology*, vol. 38, pp. 30–35, 1966.
- [93] S. Salcudean, L. Stocco, and I. Chau, "Three-degree-of-freedom parallel planar manipulator," US Patent:6339969, 2002.
- [94] D. Bristow, J. Dong, A. Alleyne, P. Ferriera, and S. Salapaka, "High bandwidth control of precision motion instrumentation," *Review of Scientific Instruments*, vol. 79, pp. 10 370 401–10 370 414, 2008.
- [95] Q. Liu, S. Lorenc, and L. Bernold, "Adaptive control for an crane-mounted inverted stewart platform," in *ASCE Proc. of Robotics 2000*, 2000, pp. 126–132.
- [96] S. Yamaguchi, W. Koterayama, and M. Inada, "Development of a motion simulator for underwater vehicles using a parallel mechanism," in *Proc. 13th International Offshore and Polar Engineering Conference*, 2003, pp. 186–190.
- [97] N. Pouliot, C. Gosselin, and M. Nahon, "Motion simulation capabilities of three-degree-of-freedom flight simulators," *Journal of Aircraft*, vol. 35, pp. 9–17, 1998.
- [98] G. Dunlop, P. Ellis, and N. Afzulpurkar, "The satellite tracking keyhole problem: a parallel mechanism mount solution," *IPENZ Trans*, vol. 20, pp. 1–7, 1993.
- [99] T.-C. Tsai and Y.-L. Hsu, "Development of a parallel surgical robot with automatic bone drilling carriage for stereotactic neurosurgery," *Biomedical Engineering: Applications, Basis and Communications*, vol. 19, pp. 269–277, 2007.
- [100] J.-P. Merlet, "Direct kinematics and assembly modes of parallel manipulators," *The International Journal of Robotics Research*, vol. 11, pp. 150–161, 1992.
- [101] R. Nair and J. Maddocks, "On the forward kinematics of parallel manipulators," *The International Journal of Robotics Research*, vol. 13, pp. 171–187, 1994.

- [102] J. Carretero, R. Podhorodeski, M. Nahon, and C. Gosselin, "Kinematics analysis and optimization of a new three degree-of-freedom spatial parallel manipulator," *ASME Journal of Mechanical Design*, vol. 122, pp. 17–24, 2000.
- [103] L.-W. Tsai and S. Joshi, "Kinematics and optimization of a spatial 3-upu parallel manipulator," *ASME Journal of Mechanical Design*, vol. 122, pp. 439–446, 2000.
- [104] R. Stamper, *A Three Degree of Freedom Parallel Manipulator with only translational degrees of freedom*. University of Maryland, Doctor of Philosophy, 1997.
- [105] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. John Wiley and Sons, Inc., 2006.
- [106] W. H. I. Reed, A. W. Hall, and L. E. J. Barker, "Analog techniques for measuring the frequency response of linear physical systems excited by frequency-sweep inputs," *NASA-TN-D-508*, 1960.
- [107] A. Stubbs, V. Vladimerou, A. Fulford, D. King, J. Strick, and G. Dullerud, "Multivehicle systems control over networks," *IEEE Control Systems Magazine*, pp. 56–69, 2006.
- [108] D. Hoelzle, A. Alleyne, and A. Wagoner Johnson, "Basis task approach to iterative learning control with applications to micro-robotic deposition," *to appear IEEE Transactions on Control Systems Technology*, 2010.
- [109] K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation," in *In Proc. of the Int. Conf. on Intelligent Robots and Systems*, 1996, pp. 546–553.
- [110] J. Herbst, "A machine learning approach to workflow management," *Lecture Notes in Computer Science*, vol. 1810, pp. 183–194, 2000.
- [111] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. John Wiley and Sons, Inc., 1996.
- [112] G. Chiu and M. Tomizuka, "Contouring control of machine tool feed drive system: A task coordinate frame approach," *Transactions on Control Systems Technology*, pp. 130–139, 2001.
- [113] A. Poo, J. Bollinger, and G. Younkin, "Dynamic errors in type 1 contouring systems," *IEEE Transactions on Industry Applications*, vol. 1A-8, pp. 477–484, 1972.
- [114] K. Barton, D. Hoelzle, A. Alleyne, and A. Wagoner Johnson, "Cross coupled iterative learning control of dissimilar systems," in *Proc. of ASME Dynamic Systems and Control Conference*, 2009, pp. DSCC2009–2727.

- [115] T. Sugar and V. Kumar, *Lecture Notes in Control and Information Sciences: Control and coordination of multiple mobile robots in manipulation and material handling tasks*. Springer Berlin/Heidelberg, 1999.
- [116] E. Haseltine and J. Rawlings, “Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics,” *Journal of Chemical Physics*, vol. 117, pp. 6959–6969, 2002.
- [117] A. Julius and G. Pappas, “Probabilistic testing for stochastic hybrid systems,” in *Proc. of 47th IEEE Conference on Decision and Control*, Cancun, Mexico, Dec 9-11 2008, pp. 4030–4035.
- [118] R. Shat, B. Rasmussen, and A. Alleyne, “Application of a multivariable adaptive control strategy to automotive air conditioning systems,” *International Journal of Adaptive Control and Signal Processing*, vol. 18, pp. 199–221, 2004.
- [119] J. Cesarano, R. Segalman, and P. Calvert, “Robocasting provides moldless fabrication from slurry deposition,” *Ceramic Industry*, vol. 148, pp. 94–102, 1998.
- [120] K. Barton and A. Alleyne, “A norm optimal approach to time-varying ILC with application to a multi-axis robotic testbed,” *Accepted to IEEE Transactions on Control Systems Technology*, 2009.
- [121] D. Hoelzle, A. Alleyne, and A. Wagoner Johnson, “Learning control for robotic deposition using machine vision,” in *Proc. of IEEE American Control Conference*, 2008, pp. 4541–4547.
- [122] —, “Iterative Learning Control using a basis signal library,” in *Proc. of IEEE American Control Conference*, St. Louis, MO, USA, June 10-12 2009.
- [123] J. Comb, P. Leavitt, and E. Rapoport, “Velocity profiling in an extrusion apparatus,” Patent, 2000.

Vita

Kira Barton was born in Colorado Springs, Colorado on September 8th, 1978 to Alan and Mary Barton. She is one of four siblings that grew up in the Colorado mountain air. Kira graduated as class salutatorian from Summit High School in Breckenridge, Colorado and moved to Boulder, Colorado in Fall 1997 to attend the University of Colorado at Boulder. In Spring 2000, Kira was awarded the University of Colorado Semester at Sea and the Ian Gini Scholarships to participate in the Semester at Sea study abroad program during which she visited ten countries around the world. She received her Bachelor of Science Degree with Distinction in Mechanical Engineering from the University of Colorado in 2001. Following her graduation, she worked for British Petroleum in Alaska and then moved to Woods Hole Massachusetts to work as a researcher at the Woods Hole Oceanographic Institute through the Summer of 2004.

Kira continued her education in mechanical engineering at the University of Illinois at Urbana-Champaign and completed her Master of Science Degree in 2006 under the guidance of Prof. Andrew Alleyne. She continued her doctoral research focusing on precision coordination and motion control for emerging applications, with a specialization in iterative learning control. In the Fall of 2007, Kira was awarded an NSF International Research and Education in Engineering (IREE) travel grant to study abroad in The Netherlands. Her work has been experimentally validated on several manufacturing platforms and has resulted, to date, in eight peer reviewed journal publications and numerous international conference papers.

In addition to research, Kira is very interested in classroom teaching and mentoring. She was awarded the ASME Graduate Teaching Fellowship for the 2009-2010 academic year and was the course instructor for an undergraduate senior / first-year graduate student level controls course in the Fall of 2009. She has mentored several undergraduate students as part of the Research Experience for Undergrads (REU) program through the NSF Center for Nanoscale Chemical-Electrical-Mechanical-Manufacturing Systems (Nano-CEMMS).

After completing her doctorate, Kira will begin working as a postdoctoral researcher at the University of Illinois and in September 2011 she will join the Mechanical Engineering Department at the University of Michigan at Ann Arbor as an Assistant Professor.