

Air Force Institute of Technology

**AFIT Scholar**

---

Theses and Dissertations

Student Graduate Works

---

12-1996

## Development of Object-Based Teleoperator Control for Unstructured Applications

Hyunki Cho

Follow this and additional works at: <https://scholar.afit.edu/etd>



Part of the [Artificial Intelligence and Robotics Commons](#), and the [Controls and Control Theory Commons](#)

---

### Recommended Citation

Cho, Hyunki, "Development of Object-Based Teleoperator Control for Unstructured Applications" (1996). *Theses and Dissertations*. 5918.  
<https://scholar.afit.edu/etd/5918>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact [AFIT.ENWL.Repository@us.af.mil](mailto:AFIT.ENWL.Repository@us.af.mil).



Development of Object Based Teleoperator Control  
for Unstructured Applications

THESIS  
Hyunki Cho  
Captain, Republic of Korea Army

AFTT/GE/ENG/96D-01

19970110 019

**DISTRIBUTION STATEMENT A**  
Approved for public release;  
Distribution Unlimited

DTIC QUALITY INSPECTED 5

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY  
**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/96D-01

Development of Object-Based Teleoperator Control  
for Unstructured Applications

THESIS  
Hyunki Cho  
Captain, Republic of Korea Army

AFIT/GE/ENG/96D-01

Approved for public release; distribution unlimited

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GE/ENG/96D-01

Development of Object-Based Teleoperator Control  
for Unstructured Applications

THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology  
Air University  
In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science (Electrical Engineering)

Hyunki Cho, B.S.  
Captain, Republic of Korea Army

December, 1996

Approved for public release; distribution unlimited

## *Acknowledgements*

This thesis could not have been done without those people who provided their endless guidance and assistance. First of all, I would like to thank my thesis advisor, Maj. Dean L. Schneider, who provided his insight comments and guidances. I also want to thank Dr. Curtis H. Spenny and Dr. John J. D'Azzo for their assistances as member of my thesis committee. All of them responded with enthusiasm whenever I needed their supports. Thanks to Steven Parmley who looked my *C*-codes and helped to understand the Chimera Real Time Operating systems, and Capt. Mark W. Hunter who provided his intellectual comments for the grasp stability and MATLAB *m*-files of the Contact Force Allocation.

I wish to acknowledge the continuouse support of staffs in the International Military Students Office who helped me and other international military students and their family both in and out of the school since we arrived here. I want to offer my congratulations and my gratitude to all the Korean officers for their patients and assistances.

Finally, my deepest gratitude goes to my lovely wife, Youjung, and my beautiful daughter, Geonlan, who suffered through the long hours that I couldn't be there when they needed me.

Hyunki Cho

## *Table of Contents*

	Page
Acknowledgements . . . . .	ii
List of Figures . . . . .	v
List of Tables . . . . .	vii
List of Abbreviations and Symbols . . . . .	viii
Abstract . . . . .	xii
I. Introduction . . . . .	1-1
1.1 Motivation . . . . .	1-1
1.2 Problem Statement . . . . .	1-2
1.3 Method of Approach . . . . .	1-3
1.4 Contributions . . . . .	1-5
1.5 Overview of Thesis . . . . .	1-5
II. Literature Review . . . . .	2-1
2.1 Introduction . . . . .	2-1
2.2 Review of Object Manipulation Methods and Stability Concerns . . . . .	2-2
2.2.1 Coordinative Manipulability Approach . . . . .	2-2
2.2.2 Impedance Control Approach . . . . .	2-4
2.2.3 Hybrid Control Approach . . . . .	2-6
2.3 Macro / Micro Manipulation Approach . . . . .	2-8
2.4 Selection of Method and Summary . . . . .	2-9

	Page
III. Development of Mathematical Model and Hierarchical Control Structure . . . . .	3-1
3.1 Overview . . . . .	3-1
3.2 Coordinate Frame Assignment . . . . .	3-2
3.3 Force Evaluation and Derivation of Dynamic Equations	3-4
3.3.1 Force Sensor . . . . .	3-4
3.3.2 SO Frame . . . . .	3-5
3.3.3 Object Frame . . . . .	3-6
3.4 Fingertip Actuation System(FAS) . . . . .	3-11
3.5 Gross and Fine Motion Control Structures . . . . .	3-11
3.5.1 Gross Motion Control . . . . .	3-11
3.5.2 Fine Motion Control . . . . .	3-13
3.5.3 FAS Centering Algorithm . . . . .	3-15
3.6 Inverse Kinematics . . . . .	3-16
3.7 Building Hierarchical Control Structures . . . . .	3-19
3.8 Summary . . . . .	3-20
IV. Simulation Experiment . . . . .	4-1
4.1 Overview . . . . .	4-1
4.2 Desired Robot Modeling . . . . .	4-1
4.2.1 Forward Kinematics . . . . .	4-1
4.2.2 Jacobian Matrix . . . . .	4-3
4.2.3 Dynamic Model . . . . .	4-4
4.2.4 Joint Range Limitation . . . . .	4-4
4.3 Gross Motion Control Simulation . . . . .	4-5
4.3.1 Free Space Motion Tracking without SO . . . . .	4-8
4.3.2 Free Motion Tracking with SO . . . . .	4-8
4.3.3 Contact with Stiff Environment without SO . . . . .	4-10



	Page
4.3.4 Contact with Stiff Environment with SO . . . . .	4-12
4.4 Fine Motion Control Simulation . . . . .	4-15
4.4.1 Step 1: One-Finger Motion Control Simulation . . . . .	4-17
4.4.1.1 Performance using Nominal Gain Set . . . . .	4-17
4.4.1.2 Compliance . . . . .	4-18
4.4.2 Step 2: Two-Finger Motion Control Simulation . . . . .	4-23
4.4.2.1 Determination of Grasping Matrix . . . . .	4-28
4.4.2.2 Step Disturbance Input . . . . .	4-30
4.4.2.3 Normally Distributed Random Distur-	
bance Input . . . . .	4-32
4.5 Summary . . . . .	4-38
V. Implementation . . . . .	5-1
5.1 Overview of Experiment . . . . .	5-1
5.2 Gross Motion Results . . . . .	5-1
5.3 Gross + Fine Motion Control Results and Stability . . . . .	5-3
5.3.1 Compliance . . . . .	5-7
5.3.2 Stability . . . . .	5-9
5.3.3 Instability . . . . .	5-9
5.4 Summary . . . . .	5-14
VI. Conclusion and Recommendation of Future Work . . . . .	6-1
6.1 Research Conclusion . . . . .	6-1
6.2 Recommendation of Future Work . . . . .	6-3
6.3 Summary . . . . .	6-4
Bibliography . . . . .	BIB-1
Appendix A. Forward Kinematics Development: 3 DOF Planar Case of PUMA 560 . . . . .	A-1

	Page
Appendix B.    MATLAB Files and SIMULINK Diagrams . . . . .	B-1
B.1 Files for Gross Motion Control . . . . .	B-1
B.2 SIMULINK Block Diagrams of Gross Motion Control .	B-7
B.3 Files for Fine Motion Control . . . . .	B-10
B.3.1 Finger # 1 . . . . .	B-10
B.3.2 Finger # 2 . . . . .	B-14
B.3.3 Object and Disturbance . . . . .	B-18
B.3.4 CFA Algorithm . . . . .	B-20
B.4 SIMULINK Block Diagrams of Fine Motion Control .	B-26
Appendix C.    Selection of Suitable Module Sampling Rate . . . . .	C-1
Appendix D.    C-codes for Implementation . . . . .	D-1
D.1 <i>imped.c</i> and <i>imped.rmod</i> . . . . .	D-1
D.2 Function files of <i>imped.c</i> . . . . .	D-13
D.3 <i>force.c</i> and <i>force.rmod</i> . . . . .	D-21
D.4 <i>fine.c</i> and <i>fine.rmod</i> . . . . .	D-29
D.5 <i>gross.c</i> and <i>gross.rmod</i> . . . . .	D-36
Vita . . . . .	VITA-1

## *List of Figures*

Figure	Page
2.1. (a) Stiffness Control, (b) Damping Control [30] . . . . .	2-5
2.2. Impedance Control [30] . . . . .	2-6
3.1. Assumed Robotic Structure with Multi-fingered Hand . . . . .	3-2
3.2. Coordinate reference frames . . . . .	3-3
3.3. Forces acting on SO frame . . . . .	3-5
3.4. Forces acting on a rigid object . . . . .	3-7
3.5. Mechanical model of Fingertip Actuation System (FAS) and Contact Force Relationship . . . . .	3-10
3.6. Gross Motion Control System . . . . .	3-13
3.7. Fine Motion Control System . . . . .	3-15
3.8. Mathematical Derivation of FAS Centering Algorithm . . . . .	3-16
3.9. Diagram of FAS Centering Algorithm . . . . .	3-17
3.10. Data Flow of Desired Control Structure . . . . .	3-19
4.1. PUMA 560 Articulated Robot [25] . . . . .	4-1
4.2. Desired Dynamic Model for Simulation . . . . .	4-5
4.3. Diagram of Gross Motion Control Simulation: (a) Free space motion with and without SO, (b) Contact with environment . . . . .	4-6
4.4. Gross Motion Control in Free Space Motion Tracking without SO : (a) Measured position (manipulator moves left along $x$ -axis while maintaining the distance on $z$ -axis), (b) Error between desired and measured position, (c) End-effector orientation ( $\theta_{total} = 0^\circ$ ) . . . . .	4-9
4.5. Gross Motion Control in Free Space Motion Tracking with SO : (a) Measured position (manipulator moves left along $x$ -axis while maintaining the distance on $z$ -axis), (b) Error between desired and measured position, (c) End-effector orientation ( $\theta_{total} = 0^\circ$ ), (d) Force measurement . . . . .	4-11

Figure	Page
4.6. Gross Motion Control in Contact with Stiff Environment without SO : (a) Measured position (manipulator tries to move into the wall which lies along $x$ -axis and maintains the same position unless the stiff environment allows the manipulator to move), (b) Error between desired and measured position, (c) End-effector orientation ( $\theta_{total} = 0^\circ$ ), (d) Force measurement . . . . .	4-13
4.7. Gross Motion Control in Contact with Stiff Environment with SO : (a) Measured position (manipulator tries to move into the wall which lies along $x$ -axis and maintains the same position unless the stiff environment allows the manipulator to move), (b) Error between desired and measured position, (c) End-effector orientation ( $\theta_{total} = 0^\circ$ ), (d) Force measurement . . . . .	4-14
4.8. Diagram of Fine Motion Control ( $-y_p$ is the direction of gravitational effect) . . . . .	4-16
4.9. One-Finger Motion Control Response under Disturbance Force Input (Step Input with maximum 0.5 (cm)) . . . . .	4-19
4.10. Joint Stability using Phase Plot for Disturbance Force Input (Step Input with maximum 0.5 (cm)) . . . . .	4-20
4.11. One-Finger Motion Control Response under Disturbance Input (Random Input (Normal Distribution) with Mean zero and $3\sigma = 0.25$ (cm)) . . . . .	4-21
4.12. Joint Stability using Phase Plot for Disturbance Input (Random Input (Normal Distribution) with Mean zero and $3\sigma = 0.25$ (cm))	4-22
4.13. Finger Response using $\mathbf{K}_{fp} = 0.001$ and $\mathbf{K}_e = 50$ . . . . .	4-24
4.14. Finger Response using $\mathbf{K}_{fp} = 0.001$ and $\mathbf{K}_e = 150$ . . . . .	4-25
4.15. Finger Response using $\mathbf{K}_{fp} = 0.006$ and $\mathbf{K}_e = 150$ . . . . .	4-26
4.16. Finger Response using $\mathbf{K}_{fp} = 0.006$ and $\mathbf{K}_e = 50$ . . . . .	4-27
4.17. Block Diagram of Simulation for Two-Finger Motion Control . .	4-28
4.18. Coordinate Assignments of each Fingertip Contact, Object and Palm Frames for Two-Finger Motion Control Simulation . . . . .	4-29
4.19. The Distance of Two Fingertips under Step Disturbance Input (Magnitude 0.5 (cm)) . . . . .	4-31

Figure	Page
4.20. Two-Finger Motion Control - Finger 1 Performance with Step Disturbance Input (Magnitude 0.5 (cm)) . . . . .	4-33
4.21. Two-Finger Motion Control - Finger 1 Joint Stability with Step Disturbance Input (Magnitude 0.5 (cm)) . . . . .	4-34
4.22. Two-Finger Motion Control - Finger 2 Performance with Step Disturbance Input (Magnitude 0.5 (cm)) . . . . .	4-35
4.23. Two-Finger Motion Control - Finger 2 Joint Stability with Step Disturbance Input (Magnitude 0.5 (cm)) . . . . .	4-36
4.24. The Distance of Two Fingertips under Normally Distributed Random Disturbance Input (Mean Zero and $3\sigma = 0.25$ (cm)) . . . . .	4-37
4.25. Fingertips Distance Error after adding Internal Force (-0.018 (N))	4-38
4.26. Two-Finger Motion Control - Finger 1 Performance with Normally Distributed Random Disturbance Input (Mean Zero and $3\sigma = 0.25$ (cm)) . . . . .	4-39
4.27. Two-Finger Motion Control - Finger 1 Joint Stability with Normally Distributed Random Disturbance Input (Mean Zero and $3\sigma = 0.25$ (cm)) . . . . .	4-40
4.28. Two-Finger Motion Control - Finger 2 Performance with Normally Distributed Random Disturbance Input (Mean Zero and $3\sigma = 0.25$ (cm)) . . . . .	4-41
4.29. Two-Finger Motion Control - Finger 2 Joint Stability with Normally Distributed Random Disturbance Input (Mean Zero and $3\sigma = 0.25$ (cm)) . . . . .	4-42
5.1. Module Block Diagram of Gross Motion Control . . . . .	5-2
5.2. Gross Motion Control Demonstration Results for Free Space Motion with Super Object (0.45 Kg) . . . . .	5-4
5.3. Gross Motion Control Demonstration Results for Contact with Environment with Super Object (0.45 Kg) . . . . .	5-5
5.4. Gross + Fine Motion Planning using PUMA 560 . . . . .	5-6
5.5. Gross + Fine Motion Control Block Diagram . . . . .	5-8

Figure	Page
5.6. Speed : Force - 500 Hz, Fine - 50 Hz, Gross - 5 Hz, Gain : 0.001, Metal case . . . . .	5-10
5.7. Speed : Force - 500 Hz, Fine - 50 Hz, Gross - 5 Hz, Gain : 0.001, Soft chair case . . . . .	5-11
5.8. Speed : Force - 500 Hz, Fine - 50 Hz, Gross - 5 Hz, Gain : 0.003, Metal case . . . . .	5-12
5.9. Speed : Force - 500 Hz, Fine - 50 Hz, Gross - 5 Hz, Gain : 0.003, Soft chair case . . . . .	5-13
A.1. Link coordinate systems for a PUMA 560 . . . . .	A-1
B.1. SIMULINK Block Diagram of Gross Motion Control . . . . .	B-8
B.2. Subsystem Block Diagrams of Gross Motion Control . . . . .	B-9
B.3. SIMULINK Block Diagram of Fine Motion Control for One-Finger Motion Control . . . . .	B-27
B.4. Subsystem Block Diagram of Fine Motion Control . . . . .	B-28
B.5. SIMULINK Block Diagram of Fine Motion Control for Two-Finger Motion Control (Main Block Diagram) . . . . .	B-29
B.6. Finger Subsystem Block Diagram of Fine Motion Control for Two- Finger Motion Control (FAS and Finger Dynamic Subsystem Block Diagrams are the same of One-Finger Simulation) . . . . .	B-30
C.1. Stable Cases of Test Sets for Overall Time History . . . . .	C-3
C.2. Unstable Cases of Test Sets When Disturbance Input is applied .	C-4
C.3. Unstable Cases of Test Sets When Disturbance Input is applied. (cont'd) . . . . .	C-5
C.4. Unstable cases of Test Sets When Disturbance Input is applied. (cont'd) . . . . .	C-6
C.5. Unstable cases of Test Sets for Overall Time History . . . . .	C-7
C.6. Unstable cases of Test Sets for Overall Time History (cont'd) . .	C-8
C.7. Unstable cases of Test Sets for Overall Time History (cont'd) . .	C-9

Figure	Page
C.8. Unstable cases of Test Sets for Overall Time History (cont'd) . .	C-10

*List of Tables*

Table	Page
4.1. Joint Range Limitation of PUMA 560 Planar Case . . . . .	4-5
4.2. Set of Applied Gains to Gross Motion Control Simulation . . . . .	4-7
4.3. Nominal Gain set of Applied to the Fine Motion Control Simulation	4-16
4.4. Experiment Sets of Fingertip Compliance . . . . .	4-23
5.1. Module Sampling Rates of Test Set #5 in Appendix C . . . . .	5-7
A.1. PUMA 560 D-H parameters . . . . .	A-2
A.2. ROBOTICA Input Data File for a PUMA 560 6 DOF Robot . . . . .	A-6
C.1. The Sets of Tests for the Gross + Fine Motion Control and Their Performances . . . . .	C-2



## *List of Abbreviations and Symbols*

<i>Abbreviations</i>	<i>Description</i>
ACC .....	air combat command
CFA .....	contact force allocation
D-H convention .....	Denavit-Hartenberg convention
DOF .....	degree of freedom
FAS .....	fingertip actuation system
OBC .....	object-based control
SO .....	super object including multifingered hand and object
SR inverse .....	singular robust inverse

<i>Symbols</i>	<i>Description</i>
$\mathbf{A}_i^j$ .....	$j$ -th link coordinate transformation matrix with respect to $i$ -th link
$d_{ifp}$ .....	length of $i$ -th fingertip link
$\mathcal{F}$ .....	resultant force and moment at the wrist
$\mathcal{F}_{obj}$ .....	force represented on the object frame
$\mathbf{f}_c$ .....	contact force vector
$\mathbf{F}_{cmd}$ .....	force applied by the operator
$\mathbf{F}_e$ .....	environmental force
$\mathbf{F}_{SO}$ .....	resultant force of the external forces applied to the SO
$\mathbf{f}_c$ .....	vector form of all fingertip contact forces
$\delta \mathbf{f}_i$ .....	force error between the optimal and measured contact force
$\mathbf{f}_{id}$ .....	$i$ -th fingertip optimal contact force provided by CFA algorithm

$\mathbf{f}_e$ .....	environmental force applied to the object
$\mathbf{f}_{iFAS}$ .....	force applied by $i$ -th fingertip actuator
$\mathbf{f}_{ext}^i$ .....	$i$ -th contact external force in contact frame
$\mathbf{f}_g$ .....	grasping force vector
$\mathbf{f}_i$ .....	force applied to the object at the $i$ -th contact
$\mathbf{f}_{int}^i$ .....	$i$ -th contact internal force in contact frame
$\mathbf{f}_m$ .....	manipulating force vector
$\mathbf{f}_o$ .....	resultant force of the external forces applied to the object
$\mathbf{f}_{ip}$ .....	force applied by $i$ -th finger actuator
$\mathbf{f}_{ip}^c$ .....	force applied by $i$ -th finger actuators at the FAS centered position
$\mathbf{f}_{ip}^e$ .....	$i$ -th finger force error to be eliminated to provide the FAS centering action
$\mathbf{g}_i(\mathbf{q}_i)$ .....	gravitational vector of $i$ -th finger
$\mathbf{g}(\Theta)$ .....	gravitational vector of robot arm
$\mathbf{H}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)$ .....	Coriolis and centripetal vector of $i$ -th finger
$\mathbf{H}(\Theta, \dot{\Theta})$ .....	Coriolis and centripetal vector of robot arm
$\mathbf{J}^*$ .....	solution of SR inverse of Jacobian
$\mathbf{J}_i(\mathbf{q}_i)$ .....	Jacobian matrix of $i$ -th finger
$\mathbf{J}(\Theta)$ .....	Jacobian matrix of robot arm
$\mathbf{K}_e$ .....	environmental stiffness gain
$\mathbf{K}_{fp}$ .....	force to position conversion gain
$\mathbf{K}_{fv}$ .....	force to velocity conversion gain
$\mathbf{K}_i$ .....	integral gain of PID controller
$\mathbf{K}_p$ .....	proportional gain of PID controller
$\mathbf{K}_v$ .....	derivative gain of PID controller
$l$ .....	number of contacts
$\mathbf{M}_i(\mathbf{q}_i)$ .....	inertia matrix of $i$ -th finger
$\mathbf{M}(\Theta)$ .....	inertia matrix of robot arm
$m_{obj}\mathbf{g}$ .....	force due to the gravitational load of object

$m_{SOg}$ .....	force due to the gravitational load of SO
$n$ .....	number of contacts
$\mathbf{n}, \mathbf{s}, \mathbf{a}$ .....	normal, sliding and approach vector of the forward kinematic
$\mathbf{o}_o, \mathbf{x}_o, \mathbf{y}_o, \mathbf{z}_o$ .....	absolute coordinates fixed at the base frame
$\mathbf{o}_{obj}, \mathbf{x}_{obj}, \mathbf{y}_{obj}, \mathbf{z}_{obj}$ .....	object coordinates fixed at the center of object mass
$\mathbf{o}_p, \mathbf{x}_p, \mathbf{y}_p, \mathbf{z}_p$ .....	palm coordinates fixed at the center of palm frame
$\mathbf{p}$ .....	position of SO with respect to the base frame
$\mathbf{p}_e$ .....	position of environment with respect to the base frame
$\dot{\mathbf{q}}_i$ .....	derivative of $i$ -th finger joints
$\ddot{\mathbf{q}}_i$ .....	acceleration of $i$ -th finger joints
$\mathbf{q}_i = [q_{i1} \ q_{i2} \ \dots \ q_{ik}]^T$ .....	relative displacements of finger joint angles
$\mathbf{q}_{ifp}$ .....	$i$ -th fingertip actuator joint displacement
$\mathbf{r}$ .....	position vector of object with respect to base frame
$\mathbf{r}_{obj}^i$ .....	$i$ -th contact position with respect to the object frame
$\hat{\mathbf{r}}_{obj}^i$ .....	skew symmetric matrix of $i$ -th contact position with respect to the object frame
$\delta \mathbf{r}_i$ .....	deviation of position required to reduce the force error of $i$ -th fingertip in contact frame
$\tau$ .....	torque applied by robot arm actuators
$\tau_e$ .....	torque reaction due to the environmental force
$\tau_{ip}$ .....	torque applied by $i$ -th finger actuators
$\tau_m$ .....	torque required by gravitational load and dynamic forces of SO due to the operator's input
$\Theta = [\theta_1 \ \theta_2 \ \dots \ \theta_n]^T$ .....	relative displacements of robot arm joint angles
$\dot{\Theta}$ .....	derivative of robot arm joints
$\ddot{\Theta}$ .....	acceleration of robot arm joints
$\Theta_d$ .....	desired robot arm joints
$\mathbf{T}$ .....	homogeneous transformation matrix

$W$  ..... grasping matrix  
 $W^\#$  ..... pseudo inverse of grasping matrix

*Abstract*

For multi-fingered end-effectors in unstructured applications, the main issues are control in the presence of uncertainties and providing grasp stability and object manipulability. The suggested concept in this thesis is object-based teleoperator control which provides an intuitive way to control the robot in terms of the grasped object and reduces the operator's conceptual constraints. The general control law is developed using a hierarchical control structure, i.e., human interface / gross motion control level in teleoperation control and fine motion control / object grasp stability in autonomous control. The gross motion control is required to provide the position / orientation of the Super Object (SO), and the sufficient grasping force to the fine motion control. Impedance control is applied to the gross motion control to respond to the environmental forces. The fine motion control consists of serially connecting the finger in position control and the Fingertip Actuation System (FAS) in force control. The FAS has a higher bandwidth response than does the finger actuation system and operates near the center of its joint range. The finger motion controller attempts not only to track the displacement of the FAS but also to provide an FAS centering action. Simulation experiments in both gross and fine motion control are performed. The integrated gross / fine motion control is implemented using the planar configuration of PUMA 560. The results show that the desired contact force can be maintained in the direction of FAS motion. The mathematical proof of system stability and the extension to spatial systems are required to complete the research.

# Development of Object-Based Teleoperator Control for Unstructured Applications

## *I. Introduction*

### *1.1 Motivation*

Robots have been used as standard tools in industrial systems, mainly where a variety of repetitive tasks are carried out. In fact, robots are very useful in certain environments when hazardous or laborious tasks are required. The techniques that control the robot autonomously or remotely have been addressed by many researchers. For the autonomous or teleoperation control of robots, the parameters to describe between the robot and the workspace environment should be predictable or preprogrammed so that the developed techniques can be used for the advanced applications such as object handling or interface with other robots. These are called *Structured Applications*. If we think about more general types of robotic applications, the techniques should be developed for *Unstructured Applications*. The definition of *Unstructured Applications* is that the relationship between the robot and environment is highly variable and unpredictable so that the tasks should be non-repetitive and can not be pre-programmed.

In general, the tasks performed in the military are in a dangerous environment, require human intensity, and are laboriously repetitive. One such task of interest to the Air Force is teleoperation assisted bomb loading or munitions handling. This application consists of maneuvering a munition into the proper position and orientation to be attached to a bomb or missile rack on an aircraft. Current procedures require three technicians to perform this task which is extremely laborious under the best of conditions. Under combat conditions, this task can be extremely dangerous,

and a goal of the Air Combat Command (ACC) is to reduce the number of personnel required for this operation. The key to workload reduction is the presentation of the maximum amount of information to the operator about the munition (object) being manipulated while reducing the operator decision-making workload.

Generally, the object handling systems consist of more than two robotic structures such as dextrous manipulators and require coordinated control schemes. The important issues of these systems are (1) how to coordinate the manipulation of a set of robotic structures and (2) how to handle an object without it breaking or falling. Both of these issues require the attention of the operator even during preplanned and repetitive structured applications. The object handling control scheme is a more difficult problem while carrying out non-repetitive tasks in complicated or unstructured environments. The teleoperated control method based on the Object-Based Control (OBC) presented here can be a feasible solution to provide a control architecture for general teleoperation systems with dextrous grasping.

## *1.2 Problem Statement*

Towards the goal of a general dextrous manipulation system, the main objective of this thesis is to develop an object based teleoperation control architecture for general telerobotic applications which provides a general grasping end-effector which is more robust and stable in the presence of perturbations of the object due to disturbance forces. This in effect will achieve a high fidelity object propulsion system with the manipulator being transparent to the user. The tasks necessary to accomplish this research are: to develop the mathematical model of teleoperated multirobotic system handling an object, and the general control law development for 4 cases of object manipulability: free space motion tracking of the end-effector, end-effector in contact with an object, free space motion tracking with a grasped object, and grasped object manipulability in contact with the environment.

The main obstacles to be overcome by an object based control method are to determine the force components involved in object manipulation and to provide grasp stability for a general motion command given by an operator. Generally, the grasping stability can be accomplished by applying as much internal forces as an object can withstand, and by providing special fixtures that constraint the grasp. However, these techniques can only be applied to structured tasks with specific grasps. Current techniques and formulations for the control of OBC systems are based on structured environments. This application is centered about teleoperator controlled systems and requires different inputs than autonomous systems. The information required by the operator must be object based to minimize the requirements of data transfer.

One way of doing this is to present an operator with a way of controlling the object itself (for example, a munition or missile) without having to impose the conceptual constraints of controlling a robot to get the desired motion of the end-effector which is attached to the object of interest. It is most intuitive for the operator to think in terms of controlling the object itself, i.e., the object having its own independent propulsion system commanded by forces applied by the operator. The main goal of the OBC Architecture is to provide such an intuitive interface for an operator with an arbitrary user input device. Additionally, precise force and position control is also required by this application.

### *1.3 Method of Approach*

A multi-fingered hand system for the coordinative manipulation can be modeled as a set of robots which are coupled to each other, and an object manipulated by this system can be modeled as a set of control constraints. One of the possible ways of achieving this model is to decompose the control tasks into appropriate levels. Each of the levels is represented as a hierarchical control structure. The desired control system can be divided into 4 levels, i.e., object for the grasp stability at the lowest level, fine motion (position and force) control for a multi-fingered hand with



an object, gross motion (position and force) control for the robot arm, and human interface as a highest level.

The primary emphasis of this thesis is to review the current mathematical model of control architectures and to select appropriate models which can be applied to each control level. The selected models will be redefined at each control hierarchy by determining the interacting force components between the control levels. The gross motion control should asymptotically follow the desired operator's motion input and consider the uncertainties such as the object and unexpected environments. The fine motion control is required to track the exact force and position of fingertips and to achieve the object grasp stability. The force applied by an operator must be seen in the object frame.

Once the individual control structures are defined, the general control law will be developed considering four cases of manipulability: (1) free space motion tracking without object, (2) end-effector contact with object, (3) free space motion tracking with grasping object and (4) grasping object manipulability in contact with the environment. For the hierarchical control structure, system variables must be carefully specified at each level of hierarchy so that the overall control structure can be easily obtained. The main control objective is to follow the commanded trajectory at the gross motion control level while maintaining the object grasp stability at the fine motion control level. For the object grasp stability, an additional actuation device called Fingertip Actuation System (FAS) would be used between the end of the finger and the contact force sensor to apply an appropriate force to the object in any required direction.

A simulation is used for the verification of the general control law and is an intermediate step to implementation. A 3-Degree of Freedom (DOF) planar robot is used for the simulation. The simulated robot kinematics are derived from the PUMA 560, links 2,3 and 5. The gross and fine motion controllers are simulated separately to establish the characteristics of stable operation. The control concepts

are modeled in the MATLAB Simulink environment [1, 2]. Due to the limitations of the experimental environment, the implementation would be accomplished by demonstrating a 3-DOF planar robot that validates the concept of coupled position and force control scheme, i.e., the last link is controlled by force control in the direction of end-effector movement and the end-effector position is accompanied by the result of force control. The concept used in simulation is transferred to the CHIMERA Real Time Operating System [35] for implementation on the PUMA 560. First, experiments on the gross motion control with the SO are based on 4 cases of manipulabilities. Then, the integrated gross / fine motion control which are simulated separately is experimented to validate the grasp stability and the requirements of both gross and fine motion controllers.

#### *1.4 Contributions*

The primary contributions of this thesis effort are listed below:

1. Developed a hierarchical control architecture based on the concept of Object-Based Control.
2. Implemented and tested a controller based on this architecture.
3. Provided the basis for intrinsic human-machine interface allowing the accomplishment of complex unstructured tasks such as munition handling.

#### *1.5 Overview of Thesis*

This thesis is organized into five chapters. Chapter Two is a review of current literature regarding coordinative control techniques and corresponding theories as well as two different methods of compliant motion control. Chapter Three presents the theoretical framework of the object based mathematical model, development of teleoperated motion control concepts for the coordinative manipulation, and building hierarchical control structure. Chapters Four and Five contain the simulation

and implementation results respectively and discuss the experimental results and analysis. Finally, Chapter Six concludes with a summary and provides some recommendations for future work.

## *II. Literature Review*

### *2.1 Introduction*

Unlike a human being who has a highly developed manipulation system, it is difficult to design a mechanical system which can perform versatile manipulation. Even though some mechanical structures may perform certain tasks with high accuracy and efficiency, these systems only have the capability of performing tasks in specific areas and lack flexibility to carry out other tasks. For flexible manipulation, the robotic manipulator with a dexterous hand is one possible approach. Robotic manipulators contribute to increase productivity and safety developments for many tasks which are dangerous and repetitive. However, many of those same tasks, including operation of the robot, still require the involvement of a human operator. These manipulative requirements need complicated techniques to attain human capabilities in artificial dexterous hands.

In most previous work, the main points addressed for object handling systems are how to design the coordinative control system and the grasping stability of an object handled by a multirobotic system. In fact, the grasping stability can be obtained by coordinating each manipulator carefully and distributing appropriate loads to the manipulators in order to avoid slippage and breakage. In order to reduce the burden on an operator for intelligent decision making, the general solution is to control forces in the object frame rather than in the base frame. If a system is designated this way, the operator needs only to command an object position. Considerable research effort has been performed to develop stable and implementable solutions for this problem.

The purpose of this literature review is to evaluate current control schemes used in object handling systems and provide possible solutions to use in this thesis problem. Many control schemes have been introduced over a wide variety of applications and it is difficult to separate them. Thus, this chapter is presented the

object manipulation approaches consisted of three parts, i.e., coordinative manipulability, impedance and hybrid control, and the review of macro / micro manipulation approaches.

## *2.2 Review of Object Manipulation Methods and Stability Concerns*

*2.2.1 Coordinative Manipulability Approach.* The general meaning of coordinative motion control in robotic systems is to control the manipulation of a desired object with multirobotic systems in order to achieve a given task. This includes topics such as materiel transfer, assembly operation, object grasping to be able to carry out other tasks, and so on. The basic concept of this control is to apply an operator's command to the multiple manipulators using geometric techniques to handle an object.

Arimoto et al. [6] proposed a bilateral master-slave control scheme of multiple robotic mechanisms. By assigning a group of some manipulators as the part of master and others as the part of slave, the master group is position controlled to follow a given trajectory while the slave group is servoed to follow the master arm with fixed relative position and orientation to accommodate an object. This approach does not consider very complicated geometrical and dynamical relations between manipulators and object. Thus, it is quite difficult to achieve the high accuracy of manipulating the object but, in practical use, it is easy to implement and reduces the operator's control effort.

Tarn, Bejczy and Yun [13] derived the closed chain formulation which was formed by two robot arms and the object through the ground. They assumed that robot arms grasped the object so tightly that only rotation between the fingers and object is possible. In this case, the contact between end-effector and object is considered as a joint of one degree of freedom. This forms a closed kinematic chain. They linearized and decoupled the dynamic equations of two robot arms including the object held by the two arms. The coordinated control of two robot arms as well

as the single arm can be derived from the linearized and decoupled system. This work showed that the structure of the controller doesn't need to be changed from one task to another task, however, the input command should be in task space so that the both robot arms can accomplish their common tasks. This method can be implemented for structured applications because accurate kinematic and dynamic control requires the knowledge of robot arms and task descriptions.

For the practical applications of coordinative control systems, errors such as a robot arm geometric error or errors due to task uncertainties may cause either critical system degradations or instability of object handling. Koga et al. [12] developed a good coordinated motion control architecture of robot arms based on a virtual internal model in order to lessen the effect of those errors. The virtual internal model is a reference model driven by sensory information, provided by a force sensor in this case, implemented in the controller. In this control architecture, the virtual internal model generates a virtual reference signal for a robot with specified dynamics, and the servo compensator controls the robot to achieve the same behavior of a virtual internal model. This research was performed by mainly considering the system state bounded against the breakage of the manipulated object and the geometric errors caused by robots arms. Since the virtual model adjusted each manipulator's trajectory, the conflicting action between the two arms can be minimized. Another similar approach is used by Soloway and Alberts [14]. Using the knowledge of the forces and torques acting on the jointly manipulated object, they present the extension of kinematic resolved rate control method to accommodate multi robotic manipulators. Since the geometric errors generate unwanted forces on the manipulated object and lead to slippage of the end-effectors or damage the object, the sensor based force control is required to provide the capability to apply a controlled force on the environment through the jointly manipulated object.

Another way to provide coordinative manipulation is to establish a complete dynamic model of a multirobotic manipulator and a desired object, and attempts

to obtain the object trajectory and its grasping stability. Murray, Li and Sastry [3] develop a combined control concept via the grasp constraint, which is required to connect the parameters of manipulators and the object. The main requirement of this concept is that the geometric model of the manipulators and object must be completely known and, in order to obtain the contact stability, the object and manipulators are assumed to be rigid. The drawback with these assumptions is that the system's performance would be degraded if any of these assumptions are failed. Hsu [4] propose the same control concept for the preplanned application such as screwing a nut onto a bolt.

*2.2.2 Impedance Control Approach.* Under the uncertainty of the environment, the interaction between the robot and its environment is a problem in minimizing deviations from desired motions while simultaneously minimizing interaction force which can cause the instability of a robot [15]. For this reason, the environment forces, including the gravitational force, should be accommodated rather than resisted so that an improved result, for instance, contact stability, can be obtained. Impedance control is able to provide such a compliant response unless there is no environmental contact. Whitney [30] provides the impedance control scheme in Fig (2.2) that is a combination of two different types of controllers: stiffness and damping controllers shown in Fig (2.1).

If the environment is an admittance, the manipulator should be an impedance [18]. This is true in Whitney's force control scheme in which the stiffness control which converts force into position is modeled as a spring mechanism and the damping control which converts force into velocity is modeled as a damper. Different from the hybrid position and force control, the impedance control is focused on the dynamic relationship between the force and position rather than tracking the desired position and force. Therefore, the impedance control is useful not only for the uncertainty on the manipulator but also for insufficient knowledge of the environment.

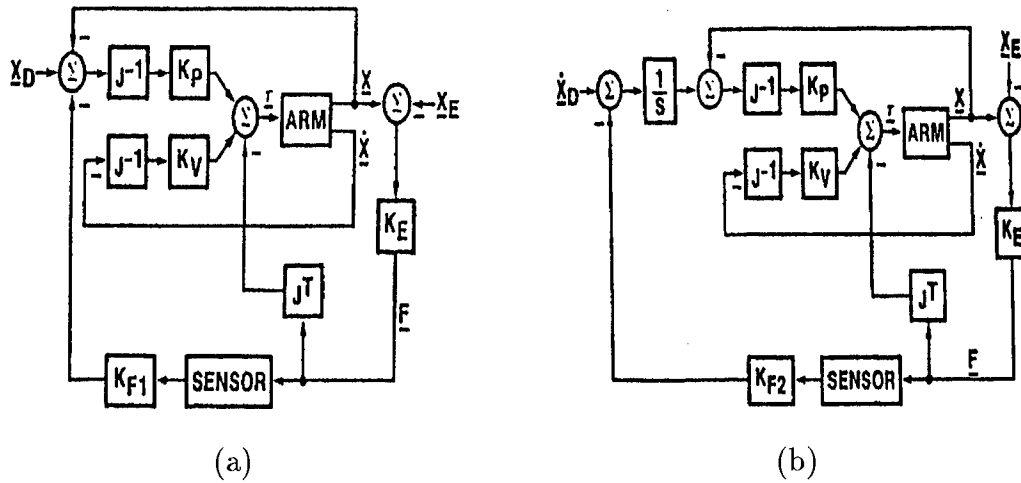


Figure 2.1 (a) Stiffness Control, (b) Damping Control [30]

For extended applications, the above statement can be developed in multiple robotic systems. If all of the manipulators in this system are considered as impedances, the rigid object grasped by multiple robot arms can be considered as an admittance. Under the contact with the stiff object, impedance control for the each manipulator provides the open-loop force control capability while the desired position gives the constant contact force. Cutkosky [15] applies the concept of impedance control in his static grasping model, i.e. the impedance control is used to feedback the resolved force of manipulators so that they can be stiff in the unconstrained directions and compliant in the constrained directions. One effort using the impedance control concept has been proposed by Schneider and Cannon [16]. In their concept, the controlled impedance is applied to the manipulated object directly, thus, the intuitive object behavior can be easily specified. For the general cases of teleoperation, the virtual object dynamics fixed in its apparent center of mass is used rather than using the actual object dynamics to obtain the desired object motion. Additionally, when using this concept, the internal force needs to be controlled explicitly.

Even though the impedance control is extremely simple and robust in the presence of parameter uncertainties, the achieved manipulation speed may be slow.



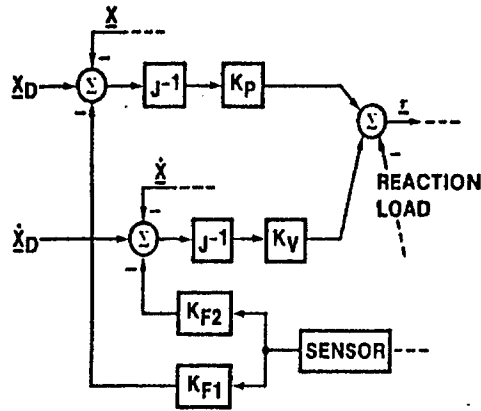


Figure 2.2 Impedance Control [30]

Another limitation of impedance control is that it does not perform well in tracking the desired force and position compared with the dynamic hybrid position and force control [17].

*2.2.3 Hybrid Control Approach.* A common method in coordinative motion control is a hybrid position and force control scheme in which the end-effector is position controlled in the unconstrained directions while controlling force in the constrained directions. Hayati [7] used this method to develop the cooperative control architecture of multiple manipulators. Based on certain point in the object fixed by an absolute coordinate frame, two subspaces of position and force are defined for the control of the object. Compared with conventional position and force control techniques [8, 9] that define the control directions, in this approach the object itself can have natural and artificial constraints and the manipulators can exert the force at the end-effector without contact with the environment. The distribution of force to each manipulator requires a precise knowledge of the mass property of all the manipulators and the object. For the unstructured applications, the problem that appeared in this paper is the stability of the system. To stabilize the system, the authors assumed that either the mass of object should be much less than that of

the manipulators or the object mass should be known when it has a large amount of mass. If stability concerns appear when large scale objects need to be handled, this approach may not work well. Another drawback of the hybrid approach is that its control structure has to be changed for a given task. This means that the control law can be valid only for the perfectly structured model so that the planning mode can perform with mathematical precision. Rather than selecting appropriate matrices, Chiaverini and Sciavicco [10] provides the parallel approach to force and position control which has no selection mechanism and provides the robustness for the conflicting situations between the position and force tasks. This approach is based on Hogan's impedance control concept [18], i.e. the manipulator and environment are defined by two complementary components: admittance (accept motion input and yield force output) and impedance (accept force input and yield motion output). The limitation of parallel approach is that the sensor measurements must always be available in order to provide sufficient knowledge of the environment.

Among the coordinative control concepts introduced in recent years, Nakamura developed a concept based on grasping stability. Using his concept, grasping stability can be defined in two way: object stability and contact stability. The former is the ability to return to the static equilibrium position and the latter is the ability to maintain contact with the object when it is disturbed by external forces. The grasping stability can be easily implemented in the static case model. However, both static and dynamic problems can not be separately handled in the object manipulation of a multirobotic system because of their dependency. Excessive contact force to provide the static equilibrium can cause object instability and inappropriate dynamic force also cause the object to be dropped. To achieve the purposes of both properties properly, two separated controllers, i.e. a coupled nature of position control corresponding to dynamic equilibrium and force control corresponding to static equilibrium, are required. This is a quite different concept of conventional hybrid position and force control approach. The main idea is that there is no constraint

direction and both position and force control can be performed in the direction of end-effector work. If the grasped object is sufficiently rigid, the displacement of one robotic mechanism could cause the motions of other robotic mechanisms. This may be valid with the following assumptions used by Nakamura: (1) A robotic mechanism makes a frictional point contact with the object, and (2) a contact point does not move on the object surface by the motion command. Also, these assumptions allow us to consider contacts which can produce only 3-axis force on the grasped object. As Nakamura mentioned in his work, it is required to develop the feedback control systems that guarantee the stability. The modified control concept is presented in Chapter 3.

### *2.3 Macro / Micro Manipulation Approach*

In general, the macro manipulation is characterized by a slow and large scale motion control using the macro robotic structure, and the micro manipulation is characterized by a fast and precise motion control using the relatively smaller robotic structure than the macro motion manipulators. In most cases, the micro motion manipulator is attached to the end of the macro motion manipulator, and the micro motion controller compensates for the error, or responds to the imperfectness of the macro motion controller. Lew and Trudnowski [31] applied the micro manipulator (rigid link) to damp out the vibration in the macro manipulator (flexible link). They used two types of controllers, i.e., the PD controller is designed to provide the joint angle motion of micro manipulator, and the flexible motion compensator is designed to control the damping of the macro manipulator. The dynamic connection between micro and macro motion controller is the flexible motion compensator which compensates between the micro and macro motion controller. The dynamics of both manipulators are not coupled each other and the PD feedback loop and flexible motion compensator are independent. In fact, they can be considered as damping control.

Stevens and How [32] developed the combined micro and macro control architecture as a coupled system. Basically, they designed the controllers so that the macro motion controller is to reach the large scale of work volume in the world frame, and the micro motion controller is to zero out the error between the desired and the current end-effector positions. Two controllers are combined by two coupling terms, i.e., the output of micro motion controller added to the macro manipulator, and the position of macro manipulator fed back to the micro motion controller to compute the position difference. These two coupling terms make the overall system closed.

For the object handling systems, the precise force and position control schemes are required to prevent the object from falling or breaking by the fingers. Both introduced control schemes have the complementary control that results in force and position, i.e., Lew and Trudnowski [31] focused on controlling the inertial force and Stevens and How [32] developed the position control of the manipulator tip. However, both control schemes used the high bandwidth response of micro motion controller to respond faster than the macro motion controller to compensate the position or force errors. This is the main idea of the Fingertip Actuation System (FAS) discussed and developed in this thesis.

#### *2.4 Selection of Method and Summary*

This chapter has reviewed the motivation for developing coordinative control concept for handling the object under the unstructured environment. Some works have been introduced and evaluated to provide a possible approach. Based on the hierarchical control structure, two ways have been suggested in this thesis: one is the teleoperated control for the robot arm (gross motion) and another is the autonomous control for the SO (fine motion). For the teleoperated control under the unstructured environment, the robot controller is not required to track exactly the desired operator's input because the controller itself admits the presence of workspace uncertainties. Since it is teleoperated control, we assume the operator can handle

this error. For this reason, the impedance control concept can be used for the gross motion control algorithm. However, the fine motion requires exact motion tracking to provide the grasp stability. Nakamura [5] presents a force domain control concept which provides the dynamic and static equilibriums in order to obtain the object grasp stability. The main objective for the coordinative manipulation with the object is to provide an autonomous control algorithm that can handle the grasp stability without being considered by the operator. This should be applied to the unstructured applications. The hybrid position and force control concept is used for fine motion control. The next chapter develops the overall control architecture of an object handling system based on hierarchical control structures.

### *III. Development of Mathematical Model and Hierarchical Control Structure*

#### *3.1 Overview*

The purpose of this chapter is to derive the basic equations used to resolve the motion (position and force) of the manipulator and multi-fingered hand with grasped object and develop the hierarchical control structures to provide an overall concept of manipulations. The assumed robotic structure of planar configuration is shown in Fig (3.1) and is referred in this thesis to develop the general object-based teleoperator control architecture. A two-fingered hand is attached to the end of robot arm and is used to grasp and manipulate an object. The Super Object (SO) presented in Chapter 1 consists of a two-fingered hand and an object and is used for the payload of the robot arm in gross motion control. The additional actuation device, called Fingertip Actuation System (FAS) located between the finger and the contact force sensor, is used to increase the grasp stability in fine motion control.

There are two steps to define the components of forces occurred in gross or fine motion control level. As mentioned in Chapter 1, the assumed robotic structure can be separated into two robots, i.e., one is the robot arm and the other is a two-fingered hand. The robot arm simply handles the gross motion of the SO and resists the external forces (gravitational load of two-fingered hand with or without an object, environmental compliance forces or teleoperator's command). The two-fingered hand is required to provide grasp stability after picking up an object. This is accomplished by the fine motion control. We assign a division of the motion control to a hierarchy based upon the type of motions [3, 28, 29]. At the upper level of the hierarchy, the operator's intelligence is used to direct overall strategy and task being pursued and the lower levels become more autonomous. Thus, the human interface and the gross motion control levels are included by the concept of teleoperation control, and the fine motion control and the object grasp stability levels are included

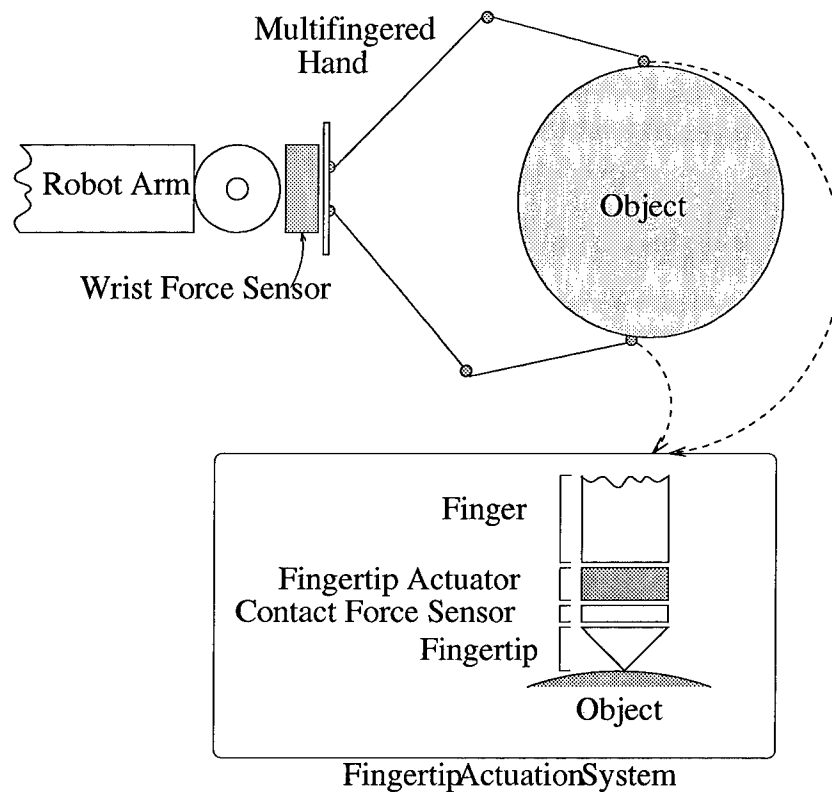


Figure 3.1 Assumed Robotic Structure with Multi-fingered Hand

by the concept of autonomous control. The critical issue in developing hierarchical control structures is to define the signals which are incoming to and outgoing from each level of hierarchy. One such problem is to determine the forces acting on either the SO frame or the object frame and establish the relationship between them. The following sections show the framework to develop the general control structure based on appropriate coordinate frames.

### 3.2 Coordinate Frame Assignment

In the control of multiple robotic systems such as dextrous hands, one of the important issues is to control the each robot simultaneously to achieve the desired task. The assignment of coordinate reference frames is significant to incorporate the kinematic issues and recognize the interacting forces between each control level.

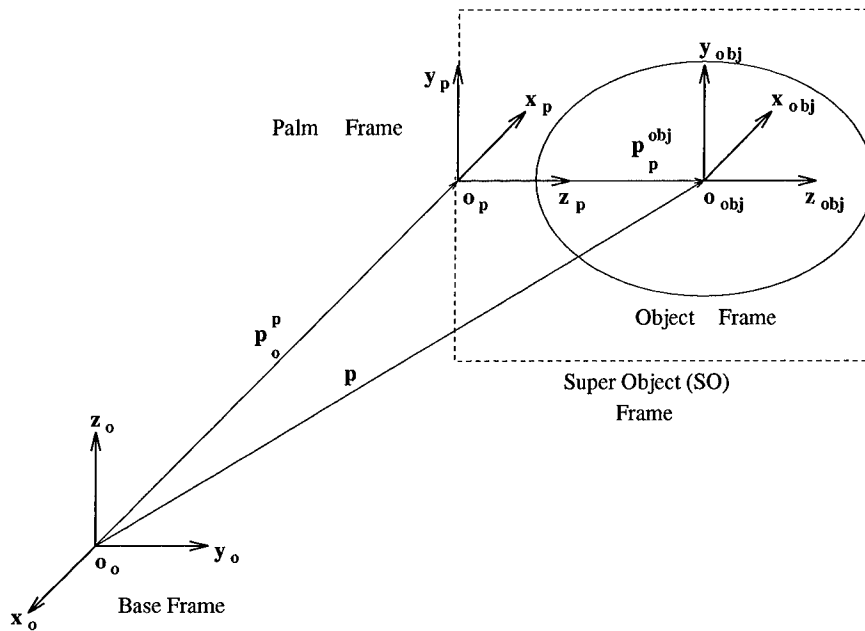


Figure 3.2 Coordinate reference frames

There are five reference frames in Fig (3.2): base frame, end-effector or SO frame, palm frame, object frame, and contact frame.

- **Base Frame** : Absolute coordinate frame (World Frame). The position and orientation of the SO is expressed in this frame.
- **End-Effector or SO Frame** : End-effector or SO coordinate frame is fixed to the center of mass of a multifingered hand or the center of mass of the end-effector with an object in its grasp (including the mass of object). We assume that (1) the movement of the object with respect to the end-effector is small compared to the overall gross motion of the object caused by the gross motion joints of the manipulator, and (2) the mass of the object will dominate the mass characteristics of the multifingered hand / object (SO) and the center of mass of SO can be assumed to be located at the center of the object mass. Thus, the origin,  $\mathbf{o}_{obj}$ , can be placed at the center of the object mass and the position vector,  $\mathbf{r}$ , is the position of the object represented in the base frame.



- **Palm Frame** : Multifingered hand base frame. This is an intermediate frame between the robot arm and fingers to express the position of all fingers with respect to the base frame. The position vector,  $\mathbf{p}$ , is the position of the palm frame,  $\mathbf{o}_p$ , represented in the base frame. This frame is used as the fine motion control base frame.
- **Contact Frame** : Contact coordinate frames are to represent the finger contact positions and forces applied by fingers to an object. For simplification, contact frames are chosen so that the  $z$ -axis of these frames are pointing inward of the normal direction on the object surface.
- **Object Frame** : Object coordinate frame fixed its center of mass. By the assumption that the mass of the object will dominate the mass characteristics of the multifingered hand / object (SO), the position and orientation of this frame should be the same as that of the end-effector frame so that the total force on the object can be evaluated.

### 3.3 Force Evaluation and Derivation of Dynamic Equations

*3.3.1 Force Sensor.* There are two kinds of force sensors needed to develop the control concept in this thesis, i.e. a wrist force/torque sensor and fingertip contact force sensors. The wrist force/torque sensor is mounted between the wrist and the hand to measure the 6-axis force information in the palm frame i.e.,  $\mathcal{F} = [F_x \ F_y \ F_z \ M_x \ M_y \ M_z]^T$ , applied to the super object. The sensed forces by this device are used for the gross motion control and calculation of optimal contact force for each finger. The fingertip contact force sensor is a device in line between the fingertip and the fingertip actuation system that measures 3-axis force information,  $\mathbf{f} = [f_x, f_y, f_z]^T$ , applied to the object shown in Fig (3.1). For simplicity, we assume that the contact between the fingertip and the object is only a point contact so that the moments are not required. In contrast to the wrist force sensor, the force sensed

by fingertip contact force sensor is fed back only to the fingertip actuation system to provide robust grasp stability.

*3.3.2 SO Frame.* The forces acting on this frame are forces due to the operator's command input as applied by the wrist, gravitational load and environmental forces. Fig (3.3) shows those forces acting on the SO frame. The total mass of multi-fingered hand and object is called the mass of SO. To simplify the problem, we assume that the mass of the multi-fingered hand is much smaller than that of the grasped object. Thus, the mass of multi-fingered hand can be ignored and the center of mass of SO is the center of mass of the object located at the center of its gravity.

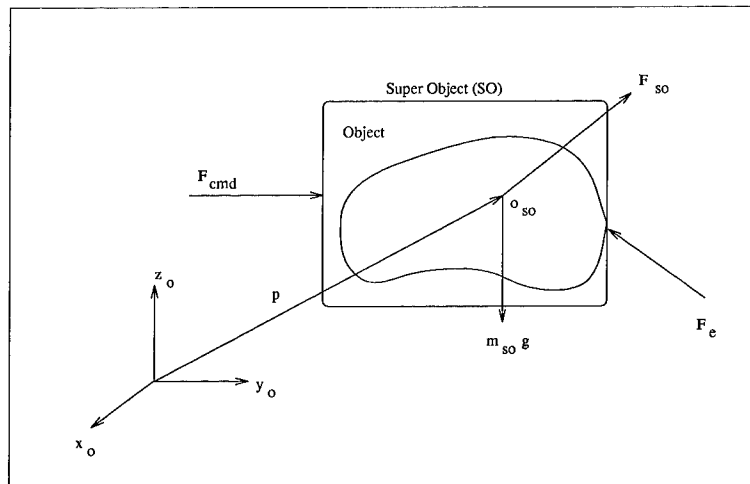


Figure 3.3 Forces acting on SO frame

- $\mathbf{F}_{cmd}$  : Force due to operator's input.
- $\mathbf{F}_{SO}$  : Resultant force of the external forces applied to the SO.
- $\mathbf{F}_e$  : Environmental interaction force.
- $m_{SO}g$  : Force due to the gravitational load of SO.

The equation of resultant force,  $\mathbf{F}_{SO}$ , is represented by

$$\mathbf{F}_{SO} = m_{SO}\mathbf{g} + \mathbf{F}_{cmd} + \mathbf{F}_e \quad (3.1)$$

where  $m_{SO}$  is the total mass of SO and  $\mathbf{g}$  is the gravitational acceleration. If the generalized coordinate vector of a manipulator,  $\Theta = [\theta_1 \ \theta_2 \ \dots \ \theta_n]^T$  denotes the relative displacements,  $\theta_i$ , between two links at  $i$ -th joint, the manipulator equations of motion [22, 25] can be represented in the form

$$\mathbf{M}(\Theta)\ddot{\Theta} + \mathbf{H}(\Theta, \dot{\Theta})\dot{\Theta} + \mathbf{g}(\Theta) = \tau - \mathbf{J}^T(\Theta)\mathbf{F}_{SO} \quad (3.2)$$

where  $\mathbf{M}(\Theta)$  is the symmetric and positive definite joint space inertia matrix,  $\mathbf{H}(\Theta, \dot{\Theta})$  the Coriolis and centripetal vector,  $\mathbf{g}(\Theta)$  the gravitational vector,  $\tau$  the torque applied by actuators, and  $\mathbf{J}^T(\Theta)$  the transpose of Jacobian matrix. The last term of right hand side in Eq (3.2) is the torque required to apply the force,  $\mathbf{F}_{SO}$ , statically to the end-effector.

*3.3.3 Object Frame.* The object frame includes a manipulated object and multi-fingered hand. The forces acting on this frame could be a gravitational force due to the mass of the object, total sum of contact forces applied by each finger and an environmental force in case of contact with environment. We make some assumptions to simplify the problem, i.e.

1. The grasped object is rigid.
2. There is a frictional point contact with the object. Thus, only 3-axis force needs to be considered at the contact points.
3. Each contact point does not move on the object surface by the change of contact force and object orientation. Thus, there is no slip of finger on contact point.

The first and third assumptions provide a simplified kinematic problem. Without these assumptions, we need to consider the specific kinematics dealing with the change of object motion. The second assumption means that the forces can be exerted in any direction within the friction cone for the contact according to the friction between the finger and an object. The forces in Fig (3.4) are defined as

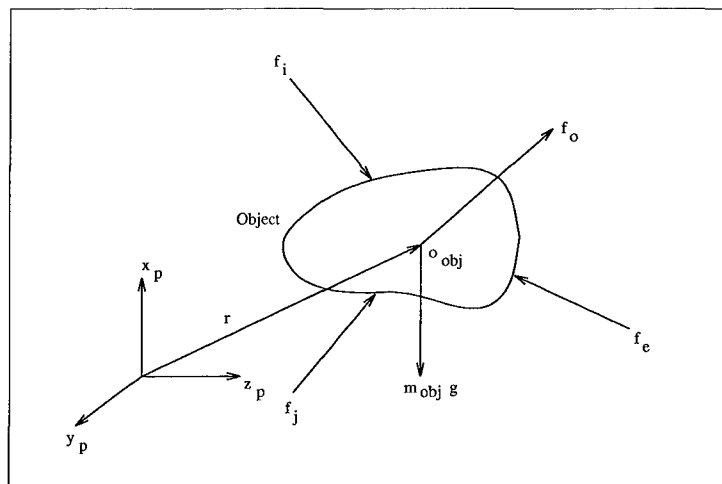


Figure 3.4 Forces acting on a rigid object

- $\mathbf{f}_i$  : Contact forces applied to the object by the  $i$ -th finger.
- $\mathbf{f}_o$  : Resultant force of the forces applied to the object.
- $\mathbf{f}_e$  : Environmental force.
- $m_{obj}\mathbf{g}$  : Force due to the gravitational load of object.

The resultant force at the object center of mass,  $\mathbf{f}_o$ , is represented by

$$\mathbf{f}_o = \sum_{i=1}^l \mathbf{f}_i + m_{obj}\mathbf{g} + \mathbf{f}_e \quad (3.3)$$

where  $m_o$  is the mass of the object,  $l$  is the number of contacts, and  $\mathbf{g}$  is the gravitational acceleration. For the restricted workspace in contact with the environment, most position and force control schemes separate the control directions: position

control accomplished in the unconstrained direction (tangential to a constraint surface) and force control in the constrained direction (normal to a constraint surface). The requirement of force control in this method is to keep the static equilibrium in the constrained direction while the position controller tracks the desired trajectory. In contrast, dynamic coordinative manipulation such as handling an object using multi-robotic structure requires the control of force and in the direction of motion. In Nakamura's concept [5], the force controller is required to satisfy static and dynamic equilibrium while the position controller is a result of the force control. The static equilibrium generates the desired contact force at the contact point so that grasp stability can be obtained. The dynamic equilibrium provides the acceleration to each finger necessary to follow the desired motion at the contact point while maintaining contact with the object. This concept implies that the motion control in the object frame should be autonomous rather than teleoperated.

We assume that the mass of object dominates the mass characteristics of the multifingered hand. The measured force on the SO should be the sum of fingertip contact forces in Eq (3.3), that is,  $\mathbf{F}_{SO}$  is the desired input force of object frame. From Eq (3.1), let  $\mathcal{F}_{obj} = [\mathbf{F}_{SO} \ \mathbf{M}_{SO}]$ , then

$$\mathcal{F}_{obj} = \mathbf{W} \mathbf{f}_c \quad (3.4)$$

where  $\mathbf{f}_c$  is defined by

$$\mathbf{f}_c = [\mathbf{f}_1 \ \mathbf{f}_2 \ \cdots \ \mathbf{f}_l]^T \quad (3.5)$$

where  $l$  denotes the number of contacts, and  $\mathbf{W}$  is defined by

$$\mathbf{W} = \begin{bmatrix} \mathbf{E}_1 & \mathbf{E}_2 & \cdots & \mathbf{E}_l \\ \hat{\mathbf{r}}_{obj}^1 & \hat{\mathbf{r}}_{obj}^2 & \cdots & \hat{\mathbf{r}}_{obj}^l \end{bmatrix} \quad (3.6)$$

where  $\hat{\mathbf{r}}_{obj}^i$  is the skew symmetric of  $i$ -th contact position vector represented by the object frame. Using the pseudoinverse of Eq (3.4), the vector form of all contact

forces is represented by

$$\mathbf{f}_c = \mathbf{W}^\# \mathcal{F}_{obj} + \mathbf{f}_{int} \quad (3.7)$$

where  $\mathbf{f}_{int}$  is the desired internal force to grasp the object. As mentioned by [3, 26, 27], the first and second terms of right hand side in Eq (3.7) represent the manipulating force and the grasping force respectively. We assume that the result of a Contact Force Allocation (CFA) algorithm developed by Hunter [23] can provide a near optimum contact force to satisfy the friction cone constraints and the stability of object grasp as well as generating the commanded force.

Additionally, we need to consider the static force applied to the object that is controlled explicitly. The purpose of finger controller task is to make the force such as

$$\mathbf{f}_i = \mathbf{f}_{ip} \quad (3.8)$$

where  $\mathbf{f}_i$  is the fingertip contact force and  $\mathbf{f}_{ip}$  is the force applied by  $i$ -th finger actuator. This ignores the dynamics of FAS. The purpose of FAS task is to eliminate of force error between  $\mathbf{f}_i$  and  $\mathbf{f}_{ip}$  so that  $\mathbf{f}_{ip}$  can follow  $\mathbf{f}_i$ . The FAS can be represented by

$$\mathbf{f}_{iFAS} = \mathbf{f}_i - \mathbf{f}_{ip} \quad (3.9)$$

where  $\mathbf{f}_{iFAS}$  is defined as the error function. Thus, the FAS shown in Fig (3.5) can maintain the contact position and improve the feasibility of grasp stability. This is discussed more in the following section.

In addition to following the desired contact force, the  $i$ -th finger actuators is also driven to provide the FAS centering action once there exists the difference between the FAS current and centered positions. The dynamic model of finger is

$$\mathbf{f}_{ip} = \mathbf{f}_{ip}^c + \mathbf{f}_{ip}^e \quad (3.10)$$

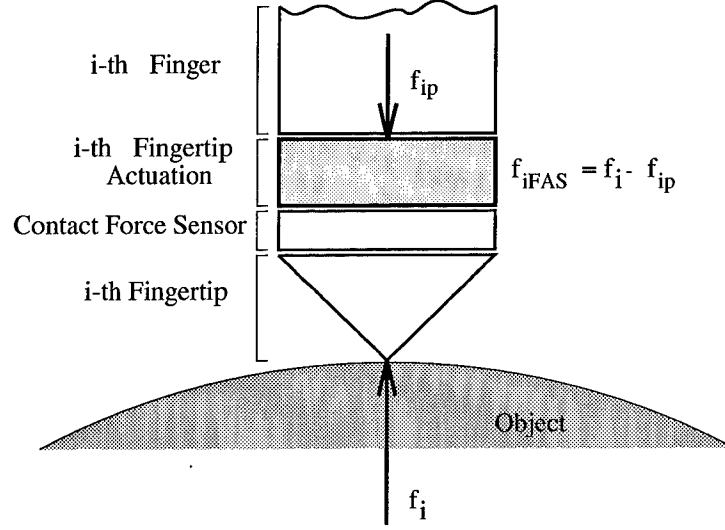


Figure 3.5 Mechanical model of Fingertip Actuation System (FAS) and Contact Force Relationship

where  $\mathbf{f}_{ip}^c$  denotes the force applied by the  $i$ -th finger actuators at the FAS centered position, and  $\mathbf{f}_{ip}^e$  denotes the force error due to the FAS response and input command of the  $i$ -th finger position controller.

A generalized coordinate for each finger is represented at the palm frame, i.e.,

$$\mathbf{q}_i = [q_{i1} \ q_{i2} \ \cdots \ q_{ik}]^T$$

where  $\mathbf{q}_i$  is the joint position of  $i$ -th finger and  $k$  denotes the number of joints of each finger. The equation of motion of  $i$ -th finger is

$$\mathbf{M}_i(\mathbf{q}_i)\ddot{\mathbf{q}}_i + \mathbf{H}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)\dot{\mathbf{q}}_i + \mathbf{g}_i(\mathbf{q}_i) = \tau_{ip} \quad (3.11)$$

where  $\mathbf{M}_i(\mathbf{q}_i)$ ,  $\mathbf{H}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)$  and  $\mathbf{g}_i(\mathbf{q}_i)$  are  $i$ -th finger inertia matrix, Coriolis and Centrifugal vector and gravity vector respectively, and  $\tau_{ip}$  is the joint torque required to achieve the desired input of  $i$ -th finger.

### 3.4 *Fingertip Actuation System(FAS)*

Most control models of coordinative manipulation [3, 5, 26, 27] have decomposed the contact force into two components: manipulating and grasping forces. The manipulating component tracks a given object trajectory and the grasping component maintains a desired internal force to prevent slipping or breakage. Under structured models, those tasks could be achieved by using the existing control models. However, for unstructured models, we need to consider the uncertainties that occur while the fingers are both manipulating and grasping the object. They are generated from imperfect fingertip control, uncertainties in the external forces applied to the object, and uncertainties in the estimations of object properties, and they cause degradation of the grasping stability.

In Fig (3.5), the FAS is attached to the end of each finger and it is required to perform force control explicitly. The FAS is required to have a higher bandwidth than the finger actuators in order to increase the grasping stability by responding to external disturbances. It can apply a force in any direction and its displacement range is much less than the range of finger actuators. The FAS reduces the force error,  $\delta \mathbf{f}_i$ , between the desired contact force,  $\mathbf{f}_{id}$ , provided by the CFA algorithm and the measured contact force,  $\mathbf{f}_i$ .

### 3.5 *Gross and Fine Motion Control Structures*

*3.5.1 Gross Motion Control.* The main objective of gross motion control is to track the desired position and orientation of the SO asymptotically. In addition, the force between the controlled SO and the environment should be controlled so that the necessary total force for the fine motion control can be provided. With this purpose, the subtasks of gross motion control can be specified by 4 cases of manipulability: (1) free space motion tracking without an object, (2) contact with environment without an object, (3) free space motion tracking with a grasped object, and (4) contact with environment with a grasped object. To accomplish these control



tasks, we assign the impedance to account for the uncertainty in the SO such as object mass. In an ordinary robotic control system, the motion of a robot arm can be controlled so as to follow a desired command accurately. With the requirements of manipulation such as interactions between a robot and its working environment, compliant motion control is required once the robot contacts the environment. The gravity effect on the SO must also be included. As mentioned in previous sections, the forces resolved by the gross motion control will effect the fine motion control in any cases of manipulability since this force is used to calculate the optimal contact force for each finger via the CFA algorithm. Thus, whenever the SO contacts the environment until the operator selects an option to release the object, the object should not slip or fall down under unexpected environmental forces.

The control technique used in this level is an impedance control strategy which implies that force does not need to be controlled explicitly and the desired position and orientation of the SO can be obtained either in contact or non-contact with the environment. If the position of the SO is  $\mathbf{p}$  and the position of environment is  $\mathbf{p}_e$ , the force exerted on the environment is given by

$$\mathbf{F}_e = \begin{cases} \mathbf{K}_e(\mathbf{p} - \mathbf{p}_e) & \text{if } \mathbf{p} > \mathbf{p}_e \\ 0 & \text{otherwise} \end{cases} \quad (3.12)$$

where  $\mathbf{K}_e$  is the environmental stiffness. From Eq (3.2), the equation of motion in joint space can be written as

$$\mathbf{M}(\Theta)\ddot{\Theta} + \mathbf{H}(\Theta, \dot{\Theta})\dot{\Theta} + \mathbf{g}(\Theta) = \tau - \tau_m - \tau_e \quad (3.13)$$

where  $\tau_m$  is the torque required by the gravitational load and dynamic forces of the SO due to the operator's input and  $\tau_e$  is the torque reaction due to the environmental force,  $\mathbf{F}_e$ . By applying the Proportional Integral Derivative(PID) control law [22, 33]

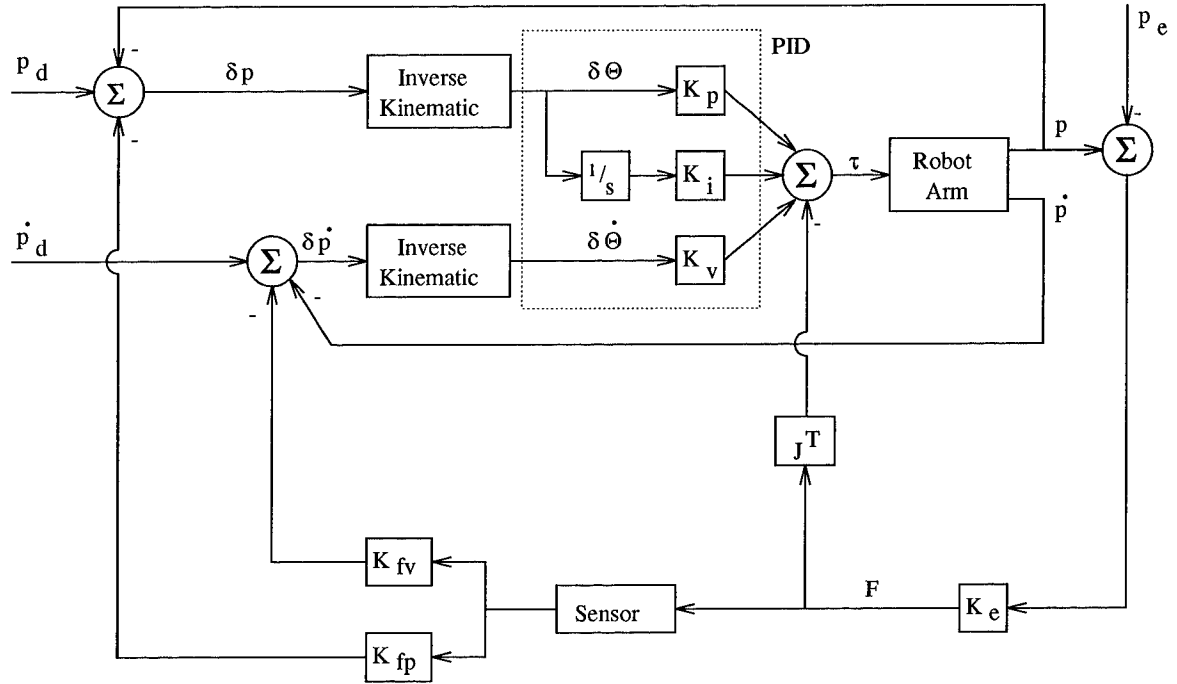


Figure 3.6 Gross Motion Control System

that provides the zero steady state tracking error response

$$\tau = \mathbf{K}_p(\Theta_d - \Theta) + \mathbf{K}_i \int (\Theta_d - \Theta) + \mathbf{K}_v(\dot{\Theta}_d - \dot{\Theta}) \quad (3.14)$$

where  $\mathbf{K}_p$ ,  $\mathbf{K}_i$  and  $\mathbf{K}_v$  are the proportional, integral, and derivative gains respectively, the desired control system based on Whitney's force feedback architectures mentioned in Chapter 2 can be developed and is shown in Fig (3.6).

**3.5.2 Fine Motion Control.** While the gross motion control is performed to achieve the desired position and orientation of SO asymptotically, the fine motion control is required to perform the force control in the object frame that produces the stable grasping force and the small displacement of contact position but not slipping. The purpose of fine motion control is to autonomously provide a stable grasping force to the finger and resist unexpected external forces. The rationale at this level is to control the desired force rather than position. The proposed input in

this level is the desired contact force which needs to be controlled explicitly. Thus, the position control can be considered as a result of force control.

The force controller must produce both static and dynamic equilibrium. Thus, the force controller can contribute both the object grasping stability and the movement. It is assumed that there is only a small deviation of contact position due to the result of force control or the disturbance produced by the gross motion control, and also small amount of force error existing between the desired and measured contact force at each fingertip because we already assume that the CFA algorithm provides sufficient conditions for the grasping force. If  $\mathbf{f}_{id}$  is the desired  $i$ -th fingertip contact force from the CFA and  $\mathbf{f}_i$  is the measured  $i$ -th fingertip contact force, then the force error is given by

$$\delta \mathbf{f}_i = \mathbf{f}_{id} - \mathbf{f}_i \quad (3.15)$$

and the deviation of position required to reduce the force error of the  $i$ -th fingertip in the contact frame is given by

$$\delta \mathbf{x}_i = \mathbf{K}_{fp} \delta \mathbf{f}_i \quad (3.16)$$

where  $\mathbf{K}_{fp}$  is the gain to convert force to position.

The FAS can be considered as a force controller and its response must be faster than the finger position controller because it needs to respond to the force change rapidly. The FAS also requires to have a sufficient range of motion in order to be able to respond to the force change adequately between the finger controller and the object. The finger controller operates based on the result of FAS response. This controller provides the FAS centering action so that the FAS is able to operate in the near center of its joint range. For this reason, both FAS and finger have their own independent controllers but the motion must be coupled between the position and force control loops, i.e., one is to control the  $i$ -th fingertip to keep the desired

contact force (explicit force control) and the other is to control the  $i$ -th finger to follow the result of force control (pure position control).

Fig (3.7) shows the fine motion control block diagram developed in this section. In force control, the fingertip force error between the desired ( $\mathbf{f}_{id}$ ) and the measured and disturbance forces ( $\mathbf{f}_i + \mathbf{f}_{idist}$ ) is converted into the position difference ( $\delta \mathbf{x}_i$ ) which is the desired motion input of the FAS. The response of FAS ( $\delta \mathbf{r}_{fi}$ ) is restricted by applying the maximum allowable joint range called the dead-band. This allows the FAS to operate within its reasonable joint range to be able to grasp the object, even though there is a large disturbance force presented on the object instantaneously. In finger position control, the difference ( $\delta \mathbf{r}_i$ ) between current ( $\mathbf{r}_i$ ) and joint centered ( $\mathbf{r}_{fi} + \mathbf{r}_{id}$ ) fingertip positions is provided to make the FAS centering action. The result of finger position control causes the response of the fingertip force control.

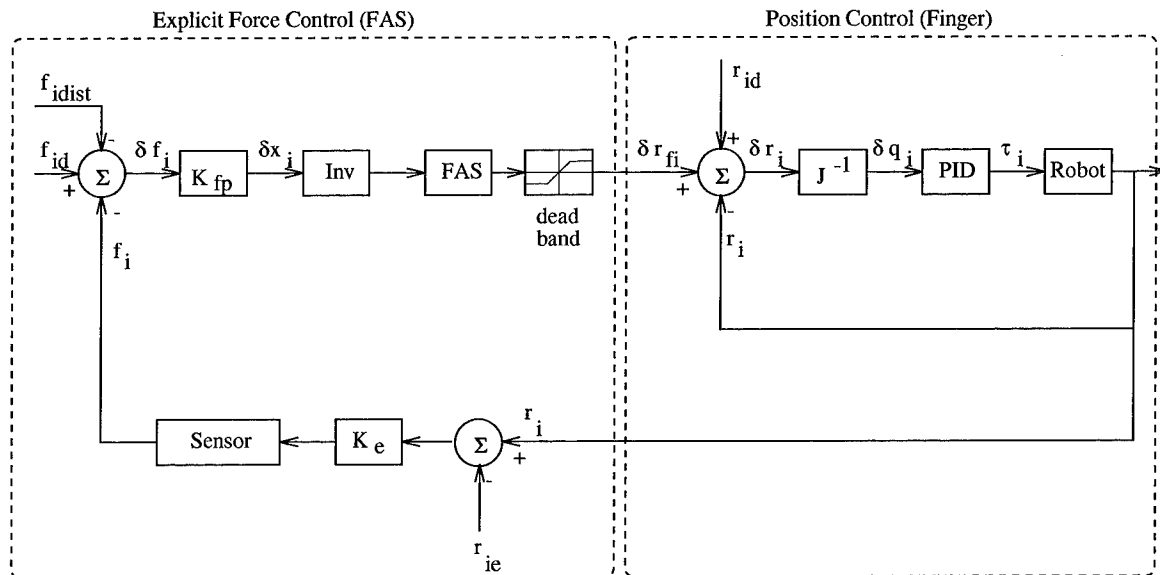


Figure 3.7 Fine Motion Control System

**3.5.3 FAS Centering Algorithm.** The motion of FAS is restricted to be centered in its joint displacement range to be able to maintain contact with the object. As stated in previous section, the FAS operates to apply the desired contact

force to the object autonomously. The response of FAS due to the force error,  $\Delta \mathbf{x}_i = \mathbf{K}_{fp}(\mathbf{f}_{id} - \mathbf{f}_i)$ , is superimposed on the commands of finger actuators. Eq (3.17) describes the relationship between position displacement and joint displacement in the planar case.

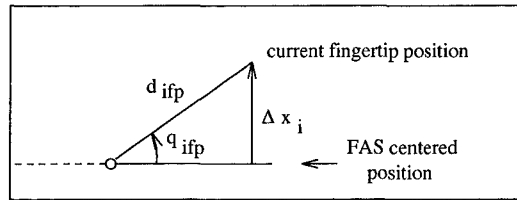


Figure 3.8 Mathematical Derivation of FAS Centering Algorithm

$$\mathbf{q}_{ifp} = \text{asin}\left(\frac{\Delta \mathbf{x}_i}{d_{ifp}}\right) \quad (3.17)$$

where  $\mathbf{q}_{ifp}$  is the  $i$ -th fingertip actuator joint displacement, and  $d_{ifp}$  the length of  $i$ -th fingertip link.

By the movement of fingertip actuator, the next higher level joint actuators follow into the direction of fingertip actuator movement which implies the pure position control, and the overall FAS centering algorithms can be represented as a fine motion control. Since the FAS centering algorithm was restricted in the SO, it can be also considered as an object propulsion system. The diagram of this algorithm is shown in Fig (3.9).

### 3.6 Inverse Kinematics

In both gross and fine motion control, the desired commands are in object space rather than in joint space. An inverse kinematic algorithm is required to determine the joint space displacement that doesn't perturb the grasp stability at singular points or their neighborhoods. In the singular points or their neighborhoods, a small change in task space requires an enormous change in joint space and can cause

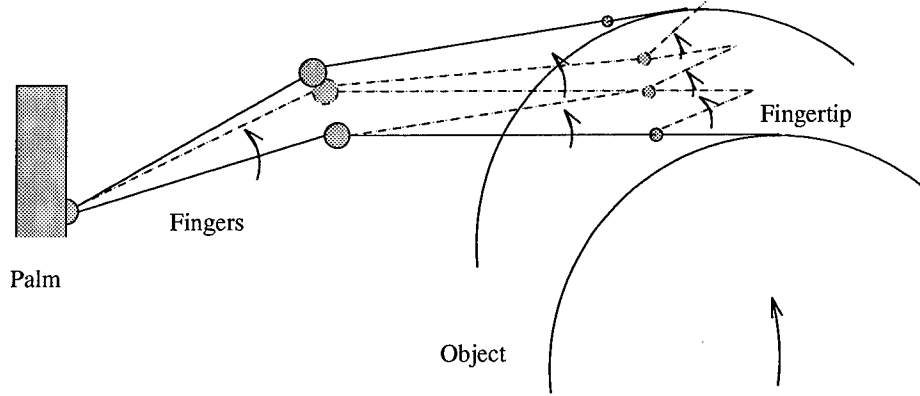


Figure 3.9 Diagram of FAS Centering Algorithm

unstable results of either the robot arm or fingers. Another requirement is to provide joint space solution in the limitation of joint range resulting from the physical design of the robot arm and fingers. In general, using the following equation

$$\delta \mathbf{x} = \mathbf{J}(\Theta) \delta \Theta \quad (3.18)$$

where  $\delta \mathbf{x}$  is the increments of position and orientation,  $\mathbf{J}(\Theta)$  is the Jacobian matrix, and  $\delta \Theta$  is the joint space increments, the inverse kinematics are solved as

$$\delta \Theta = \mathbf{J}^{-1}(\Theta) \delta \mathbf{x} \quad (3.19)$$

However, for kinematically redundant robotic structures, Eq (3.19) cannot provide the inverse solution at the singular points that make  $\mathbf{J}(\Theta)$  not full rank, i.e.,  $\det \mathbf{J}(\Theta) = 0$ . To achieve the inverse solution of  $\mathbf{J}(\Theta)$  even at the singular points, the pseudoinverse is a proposed way which is defined as

$$\delta \Theta = \mathbf{J}^{\#}(\Theta) \delta \mathbf{x} \quad (3.20)$$

where  $\mathbf{J}^{\#}(\Theta)$  is defined as the pseudoinverse of  $\mathbf{J}(\Theta)$ . At the singular points or their neighborhoods, Eq (3.20) satisfies the exactness of  $\mathbf{x} - \mathbf{J}(\Theta) \delta \Theta$  but it fails the

feasibility of  $\delta\Theta$  due to the nature of discontinuity of the pseudoinverse at singular points or the enormous change of  $\delta\Theta$  in the neighborhood of singularity. For the control of a finger grasped object, this result can cause the object to drop or break.

To satisfy both exactness and feasibility, Nakamura's Singular Robust Inverse (SR Inverse) [5] is used because it provides not only the most feasible solution but also the minimized error. Suppose that  $\delta\mathbf{e}$  consists of  $\delta\mathbf{x} - \mathbf{J}(\Theta)\delta\Theta$  which represents the exactness and  $\delta\Theta$  which represents the feasibility. Then

$$\delta\mathbf{e} = \begin{pmatrix} \min\|\delta\mathbf{e}\|_{\mathbf{W}} \\ \delta\mathbf{x} - \mathbf{J}(\Theta)\delta\Theta \\ \delta\Theta \end{pmatrix} \quad (3.21)$$

where  $\|*\|_{\mathbf{W}}$  is the weighted norm and  $\mathbf{W} \in R^{(m+n) \times (m+n)}$  the weighting matrix, symmetric and positive definite. For the simple computation, we assume that  $\mathbf{W}$  is the identity matrix. The following equation is used;

$$\min\|\delta\mathbf{e}\|_{\mathbf{W}} = \sqrt{\delta\mathbf{e}^T \mathbf{W} \delta\mathbf{e}} \quad (3.22)$$

Let  $\mathbf{J}(\Theta) \in R^{m \times n}$  where  $m$  is the number of constraints and  $n$  the number of degrees of freedom. Then, the general solution of the SR-inverse using Eq (3.22) is obtained as following equations.

$$\mathbf{J}^*(\Theta) = [\mathbf{J}^T(\Theta)\mathbf{J}(\Theta) + k\mathbf{E}]^{-1}\mathbf{J}^T(\Theta) \quad (3.23)$$

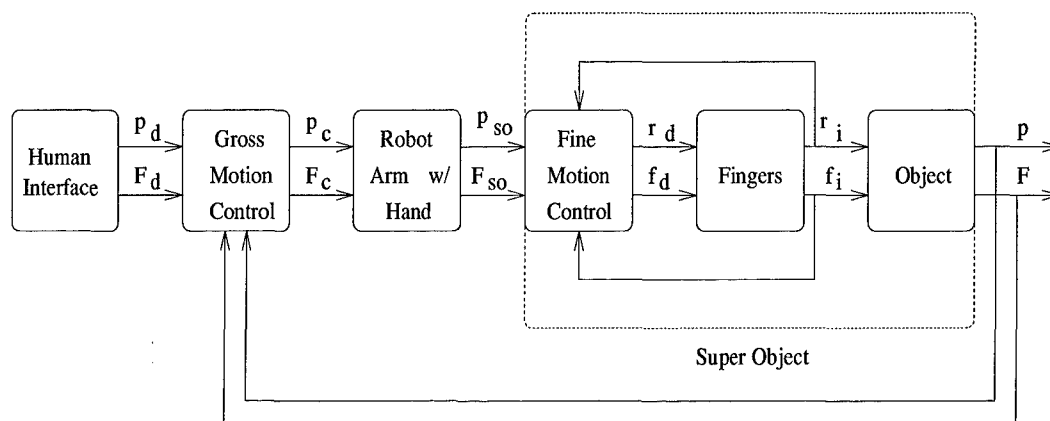
or

$$\mathbf{J}^*(\theta) = \mathbf{J}^T(\Theta)[\mathbf{J}(\Theta)\mathbf{J}^T(\Theta) + k\mathbf{E}]^{-1} \quad (3.24)$$

where  $\mathbf{J}^*(\Theta)$  is the SR Inverse,  $k$  is a scale factor allowing the computation to avoid the singularity, and  $\mathbf{E}$  is the identity matrix. Two equations produce the same results of  $\mathbf{J}^*(\Theta) \in R^{n \times m}$ .

### 3.7 Building Hierarchical Control Structures

As stated in previous sections, we model overall robot structure as a connected set of robots: the arm, a multi-fingered hand(finger) and object. Normally, the hierarchical control structures are divided by their control purposes based on the task description. The data flow of the proposed control structure is shown graphically in Fig (3.10).



- $P_d$  : Desired Super Object Position
  - $F_d$  : Desired Environment Force
  - $P_c$  : Position Command of Super Object
  - $F_c$  : Environment Force Command
  - $r_d$  : Desired i-th Finger Position
  - $f_d$  : Desired i-th Finger Contact Force
  - $r_i$  : Resolved i-th Finger Position
  - $f_i$  : Resolved i-th Finger Contact Force
  - $P_{so}$   $P$  : Super Object (Object) Position
  - $F_{so}$   $F$  : Super Object (Object) Resultant Force
- In Contact with Environment

Figure 3.10 Data Flow of Desired Control Structure

At the lowest level, the interaction between the fingers and object should operate without the operator's intelligent decision making. The resolved forces are fed back to the fine motion control level rather than feeding back to a higher level.



At the fine motion control level, the position and force control are highly coupled and, furthermore, the finger motions are also dependent on the others. This means that the movement of one finger will affect the other's resolved position and force under the assumption stated previously. The next higher level connects both gross and fine motion control. By the assumption that the displacement of fine motion control is small and doesn't affect the overall motion of the SO, the output of gross motion control, i.e., sensed force due to the payload of SO and its resolved position based on the estimation of the object's mass, is input to the fine motion control. The fine motion control returns the output of its resolved object position. Thus, the gross motion control must provide enough force to the fine motion control to resist movement when the robot touches the environment or is disturbed by other external forces, i.e., it provides a stable base for fine motion operations. The highest level is human supervision through the sensor system and input devices.

### *3.8 Summary*

This chapter has developed the general control structure of the robot including a multi-fingered hand. To accomplish the 4 cases of manipulability, the control scheme was separated into two main parts, i.e. the gross motion control which is required to control the position and orientation of SO and the force when the SO is in contact with the environment, and the fine motion control which is required to control the finger contact force to stably grasp an object and the position due to the result of force control. The significant issue is to define the input and output signals of each control level and the connections to yield consistent results.

## IV. Simulation Experiment

### 4.1 Overview

This chapter evaluates the concept of a control system proposed in the previous chapter. The MATLAB SIMULINK environment [1, 2] is used for the simulation. For the simple case and implementation in Chapter 5, a 3 DOF planar robot is used as the desired robot model for gross and fine motion controls with kinematic data from a PUMA 560 articulated robot [25]. Fig (4.1) shows the structure of PUMA 560 and its kinematic data. Joint 2, 3 and 5 are selected to provide planar motion.

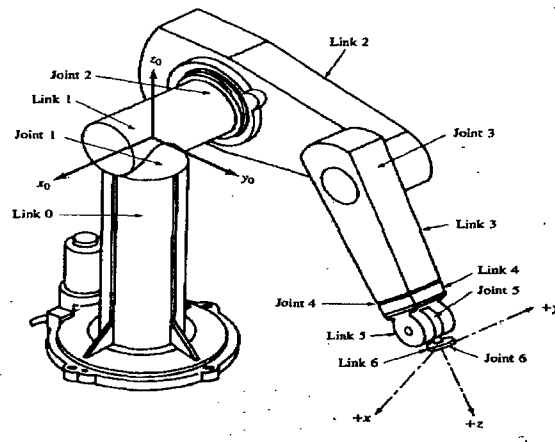


Figure 4.1 PUMA 560 Articulated Robot [25]

### 4.2 Desired Robot Modeling

**4.2.1 Forward Kinematics.** The forward kinematics is used to transform joint space position to Cartesian space position and orientation. In this thesis, the forward kinematics is limited to the  $x-z$  plane in accordance with the Denavit-Hartenberg(D-H) convention [22] of PUMA 560 in Fig (4.1), i.e.,  $\theta_1 = \theta_4 = \theta_6 = 0$ .

The homogeneous transformation matrix is given by

$$\mathbf{T} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where  $\mathbf{n}$ ,  $\mathbf{s}$  and  $\mathbf{a}$  are the normal, sliding and approach vectors respectively. The terms,  $\mathbf{n}$ ,  $\mathbf{s}$  and  $\mathbf{a}$ , represents the rotational matrix and  $\mathbf{p}$  represents the position vector of the end-effector. The complete forward kinematics computation is found in Appendix A. The computed forward kinematics equations of the 3 DOF planar robot are:

$$\begin{aligned} n_x &= \sin(\theta_2 + \theta_3 + \theta_5) \\ n_y &= 0 \\ n_z &= \cos(\theta_2 + \theta_3 + \theta_5) \\ s_x &= -\cos(\theta_2 + \theta_3 + \theta_5) \\ s_y &= 0 \\ s_z &= \sin(\theta_2 + \theta_3 + \theta_5) \\ a_x &= 0 \\ a_y &= 1 \\ a_z &= 0 \end{aligned} \tag{4.1}$$

$$\begin{aligned} p_x &= a_2 \cos(\theta_2) + d_4 \cos(\theta_2 + \theta_3) + (d_6 + d_t) \cos(\theta_2 + \theta_3 + \theta_5) \\ p_y &= 0 \\ p_z &= a_2 \sin(\theta_2) + d_4 \sin(\theta_2 + \theta_3) + (d_6 + d_t) \sin(\theta_2 + \theta_3 + \theta_5) \end{aligned} \tag{4.2}$$

where  $a_2$ ,  $d_4$  and  $d_6$  are the lengths of link 2, 3 and 5 respectively and  $d_t$  is the distance from the end of the last link to the center of SO frame(object frame).

*4.2.2 Jacobian Matrix.* The Jacobian matrix for the end effector or finger is used to translate the force in Cartesian space into torque in joint space,  $\tau = \mathbf{J}^T(\Theta)\mathbf{F}$ , and the end-effector displacement at a sample time or velocity into joint displacements or rates,  $\delta\Theta = \mathbf{J}(\Theta)\delta\mathbf{r}$ .

In motion tracking problems, the position and orientation are the most important concern. The gross motion control counts on the position and orientation problems while the fine motion control concerns only the position problem. If the desired tasks,  $\mathbf{r}(\Theta)$ , are represented by Eq (4.3)

$$\mathbf{r}(\Theta) = \begin{pmatrix} \mathbf{r}_1(\Theta) \\ \mathbf{r}_2(\Theta) \end{pmatrix} \quad (4.3)$$

where  $\mathbf{r}_1(\Theta)$  and  $\mathbf{r}_2(\Theta)$  represent the task 1 (SO position) and task 2 (SO orientation) vectors which are functions of the joint angles,  $\Theta$ , the differentiation of Eq (4.3) with respect to  $\Theta$  is given by

$$\mathbf{J}(\Theta) = \frac{\partial \mathbf{r}}{\partial \Theta} \in R^{m \times n} \quad (4.4)$$

where  $\mathbf{J}(\Theta)$  is the Jacobian matrix,  $m$  the number of task constraints and  $n$  the number of DOF. From Eq (4.2), the desired task 1 and task 2 are

$$\begin{aligned} \mathbf{r}_1(\Theta) &= \begin{pmatrix} p_x \\ p_z \end{pmatrix} \\ \mathbf{r}_2(\Theta) &= \sin(\theta_2 + \theta_3 + \theta_5) \end{aligned} \quad (4.5)$$

By differentiating Eq (4.5), the Jacobian matrix can be obtained by the following equations.

$$\mathbf{J}(\Theta) = \begin{pmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{pmatrix} \quad (4.6)$$

$$\begin{aligned}
J_{11} &= -a_2 \sin(\theta_2) - d_4 \sin(\theta_2 + \theta_3) - (d_6 + d_t) \sin(\theta_2 + \theta_3 + \theta_5) \\
J_{12} &= -d_4 \sin(\theta_2 + \theta_3) - (d_6 + d_t) \sin(\theta_2 + \theta_3 + \theta_5) \\
J_{13} &= -(d_6 + d_t) \sin(\theta_2 + \theta_3 + \theta_5) \\
J_{21} &= a_2 \cos(\theta_2) + d_4 \cos(\theta_2 + \theta_3) + (d_6 + d_t) \cos(\theta_2 + \theta_3 + \theta_5) \\
J_{22} &= d_4 \cos(\theta_2 + \theta_3) + (d_6 + d_t) \cos(\theta_2 + \theta_3 + \theta_5) \\
J_{23} &= (d_6 + d_t) \cos(\theta_2 + \theta_3 + \theta_5) \\
J_{31} &= \cos(\theta_2 + \theta_3 + \theta_5) \\
J_{32} &= J_{33} = J_{31}
\end{aligned} \tag{4.7}$$

*4.2.3 Dynamic Model.* The purpose of simulations is to show the developed control concept using the currently available simulation tool. The actual robot takes the torque input and produces the joint space output. The equation of motion is

$$\tau = \mathbf{M}(\Theta)\ddot{\Theta} + \mathbf{H}(\Theta, \dot{\Theta})\dot{\Theta} + \mathbf{g}(\Theta) \tag{4.8}$$

where  $\tau$  is the torque applied by actuators,  $\mathbf{M}(\Theta)$  the symmetric and positive definite joint space inertia matrix,  $\mathbf{H}(\Theta, \dot{\Theta})$  the Coriolis and centripetal vector, and  $\mathbf{g}(\Theta)$  the gravitational vector. If we assume that  $\mathbf{M}$  is a unity matrix and  $\mathbf{g}$  is ignored, then the inverse dynamic consists of amplified double integrator terms with joint velocity feedback. The inverse dynamic equation is

$$\ddot{\Theta} = \mathbf{M}^{-1}(\Theta)[\tau - \mathbf{H}(\Theta, \dot{\Theta})\dot{\Theta}] \tag{4.9}$$

This eliminates the nonlinear terms in the manipulator's equation of motion and results in a second order linear system shown in Fig (4.2).

*4.2.4 Joint Range Limitation.* One of the significant problem in inverse kinematics is to overcome the joint range limitation which is a physical constraint

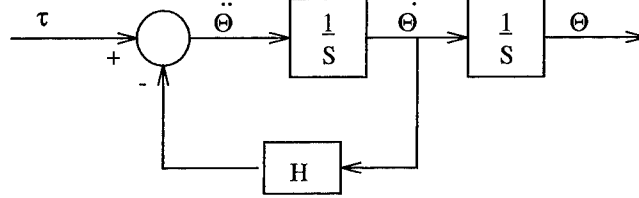


Figure 4.2 Desired Dynamic Model for Simulation

of robotic manipulators. For most robots, the solution of inverse kinematics is not unique. This means that one of the joint values determined by the inverse kinematics could be over its limit. Thus, the simulation and demonstration algorithm should consider explicitly that a joint at the end of its motion keeps either its minimum or maximum value while the other joints adjust their values so that the desired end-effector motion can be achieved. The set of joint range limits of the planar case PUMA 560 is shown in Table 4.1. The  $z$ -axis is according to the D-H parameter table determined in the forward kinematics.

Joint $i$	Joint range(Degree)
2	$-45^\circ$ to $225^\circ$
3	$-225^\circ$ to $45^\circ$
5	$-100^\circ$ to $100^\circ$

Table 4.1 Joint Range Limitation of PUMA 560 Planar Case

### 4.3 Gross Motion Control Simulation

As mentioned in Chapter 3 Fig (3.6), the gross motion controls the position and orientation of SO and the force for the contact with environment including the gravity effect of the SO. The trajectory planning for the simulations is shown in Fig (4.3). The free space motion tracking with and without the SO is simulated using the trajectory planning in Fig (4.3) (a) which is the straight line movement in the negative  $x$  direction. The robot performance in contact with the environment with and without the SO is simulated using the trajectory planning in Fig (4.3) (b)

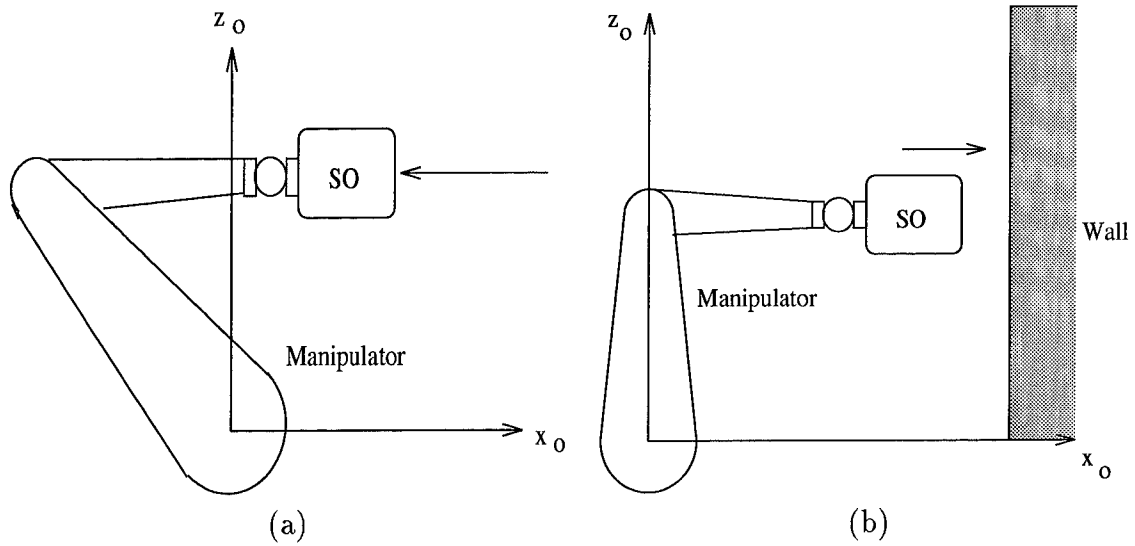


Figure 4.3 Diagram of Gross Motion Control Simulation: (a) Free space motion with and without SO, (b) Contact with environment

which has a wall on the  $x$ -axis and moves in a straight line in the positive  $x$  direction. All of trajectory plans keep a constant input on the  $z$ -axis. The initial robot position of both trajectory plans is  $\Theta = [90^\circ \ -90^\circ \ 0^\circ]^T$ .

For this simulation, the appropriate gains,  $\mathbf{K}_e$ ,  $\mathbf{K}_{fp}$  and  $\mathbf{K}_{fv}$ , need to be obtained. In general,  $\mathbf{K}_{fp}$  and  $\mathbf{K}_{fv}$  depend on the characteristics of environment,  $\mathbf{K}_e$ , i.e., if  $\mathbf{K}_e$  is large,  $\mathbf{K}_{fp}$  needs to be decreased, and if  $\mathbf{K}_e$  is small,  $\mathbf{K}_{fp}$  needs to be increased. However, the initial  $\mathbf{K}_e$  is selected heuristically. The relationship between  $\mathbf{K}_{fp}$  and  $\mathbf{K}_{fv}$  is derived by the following equations.

$$\Delta \mathbf{f}(t) = m \Delta \ddot{\mathbf{p}}(t) \quad (4.10)$$

where  $\Delta \mathbf{f}$  is the force error between the desired and the measured contact forces,  $m$  is the total mass of the systems, and  $\Delta \mathbf{p}$  is the position deviation due to force error. The Laplace transform of Eq (4.10) yields

$$\Delta \mathbf{F}(s) = ms^2 \Delta \mathbf{P}(s) \quad (4.11)$$

$$\Delta \mathbf{P}(s) = \frac{1}{ms^2} \Delta \mathbf{F}(s) \quad (4.12)$$

$$\mathcal{L}^{-1}\{\Delta \mathbf{P}(s)\} = \frac{t}{m} \Delta \mathbf{f}(t) \quad (4.13)$$

Thus, the terms,  $t/m$  can be represented as  $\mathbf{K}_{fp}$  and  $1/m$  can be represented as  $\mathbf{K}_{fv}$  based on Eq (4.12) and (4.13) respectively. In the gross motion control simulation, the system is running at the sampling rate  $t = 0.01$  (sec). Therefore,  $\mathbf{K}_{fv}$  is 100 times larger than  $\mathbf{K}_{fp}$ . However, the initial selection of  $\mathbf{K}_{fp}$  depends on the selection of  $\mathbf{K}_e$  and they are heuristically chosen at the initial step. The actual hardware PID gains are the same as used in the PUMA 560 Joint 2, 3 and 5 because the same kinematic structure is used in this simulation and the dynamic structure is the ideal case for robot performance. The SR-Inverse gain,  $k$ , is selected by the measure of manipulability of the manipulator. For the simple computation, a constant number is used in this simulation. The gains used in this simulation are listed in Table (4.2).

Joint $i$		2	3	5
PID gain set	$\mathbf{K}_p$	5000	1500	160
	$\mathbf{K}_I$	5	5	5
	$\mathbf{K}_v$	114	25	12
environmental stiffness gain $\mathbf{K}_e$			15100	
force to position gain $\mathbf{K}_{fp}$			0.000254	
force to velocity gain $\mathbf{K}_{fv}$			0.0254	
SR-Inverse gain $k$			0.01	

Table 4.2 Set of Applied Gains to Gross Motion Control Simulation

The specific SIMULINK block diagram and MATLAB function files are listed in Appendix B.



*4.3.1 Free Space Motion Tracking without SO.* The free motion tracking without SO involves the only position and orientation control in free space. Based on Fig (4.3) (a), the position command is

$$\begin{aligned} p_x &= p_{x0} - 0.12t \\ p_z &= p_{z0} \end{aligned} \tag{4.14}$$

where  $p_{x0} = 48.93$  (cm) and  $p_{z0} = 43.18$  (cm) are the initial positions of the SO. The desired orientation of the SO is  $\theta_{total} = \theta_2 + \theta_3 + \theta_5 = 0^\circ$ . Fig (4.4) (a), (b) and (c) show the desired position orientation tracking and their errors. In Fig (4.4) (b), the errors at the initial points are relatively larger than the other points, but the controller track the desired input quickly within 0.5 (sec). The  $z$ -axis error is larger than the  $x$ -axis error for the entire time history because of the gravity effect of the link masses. For the orientation,  $\theta_{total} = \theta_2 + \theta_3 + \theta_5$ , Fig (4.4) (c) shows the orientation of end-effector. As a consistent result of position control, the orientation is disturbed at the initial points, but it overcomes the error eventually as the time goes to  $t=5$  (sec). If  $t > 5$  (sec), the orientation will be more disturbed to achieve the desired position due to workspace constraints. In both position and orientation, the errors are not perfectly eliminated because the constant SR-Inverse gain was used for overall trajectories. This means that the error was added intentionally to the inverse solution even though there is no singular point. Another reason is that the concept of impedance control was used for the gross motion control. From the teleoperation point of view, this is good enough for the manipulator feasibility rather than the exactness of trajectory following.

*4.3.2 Free Motion Tracking with SO.* Fig (4.5) shows the free space motion of the manipulator with the SO. The same trajectory used in the previous simulation is applied. The difference is that a 20 (N) SO (the maximum load of PUMA 560 is 2.04 (kg) which is 20 (N)) is attached to the tool frame. Thus, the end-effector

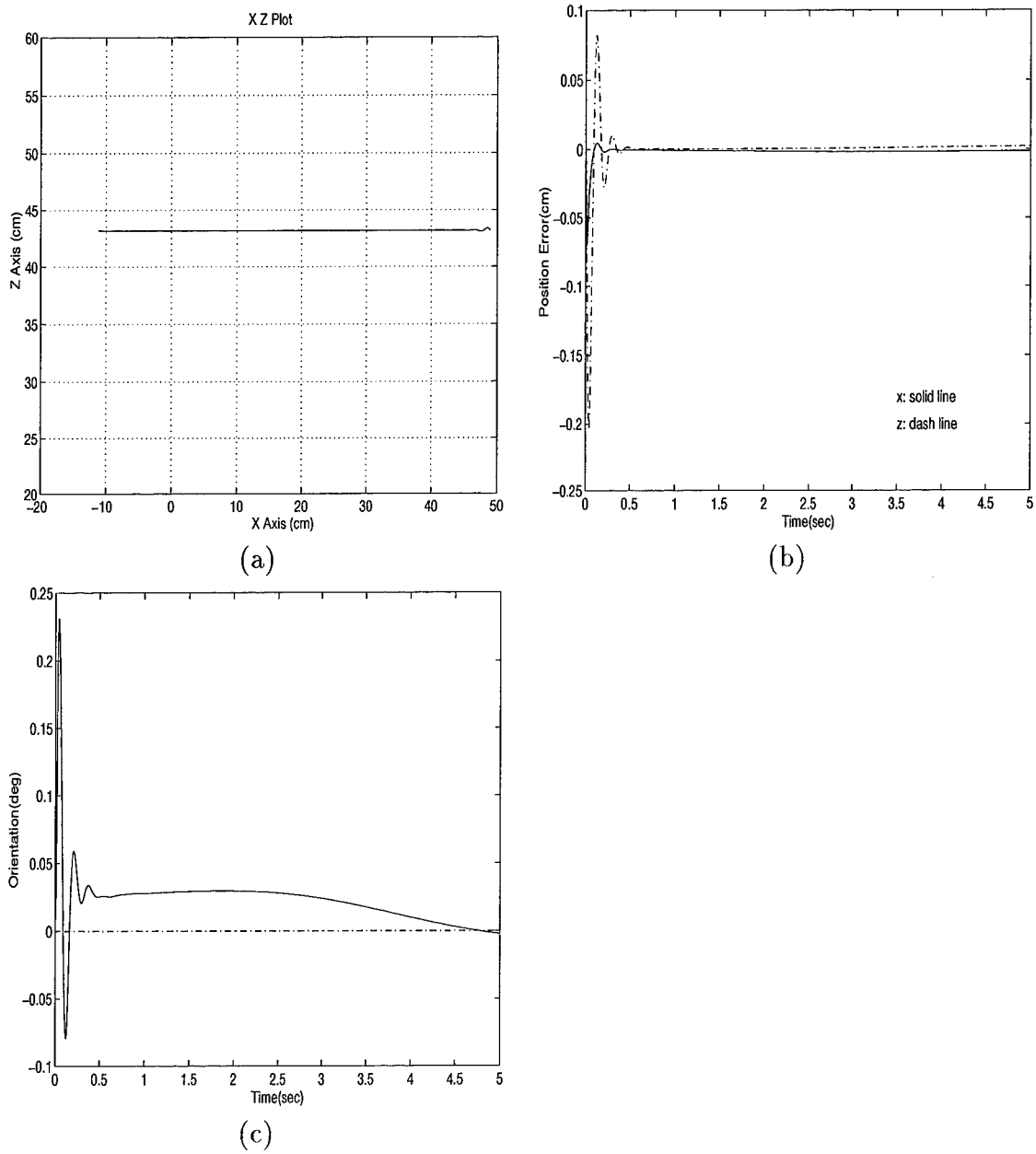


Figure 4.4 Gross Motion Control in Free Space Motion Tracking without SO : (a) Measured position (manipulator moves left along  $x$ -axis while maintaining the distance on  $z$ -axis), (b) Error between desired and measured position, (c) End-effector orientation ( $\theta_{total} = 0^\circ$ )

motion should be disturbed mainly by the gravitational force due to the SO. The main error occurs in the motion along the  $z$ -axis because the controller compensates for the effect of gravity on the SO. This error is shown in Fig (4.5) (a) and (b) that the largest error occurs at the initial time and then tries to track the desired trajectory. This allows the system to pick up an undetermined object because the impedance control can regulate unexpected force occurring at the SO rather than tracking. Since we are operating as a teleoperator, the human will be zeroing out those errors which the controller can not track. The position deviation mainly occurs on the  $z$ -axis since this is the direction of the gravity vector. If the mass of SO is quite large, other dynamic disturbance forces may have an effect on the SO. However, at low speeds, they can be ignored except for the gravitational force. Fig (4.5) (b) shows the position error on the  $z$ -axis that exists as a small value at all times. The orientation is also bounded with the range  $-1^\circ \sim 3.7^\circ$  which is 15 times greater than the free motion tracking simulation without the SO. Since there are two other types of feedbacks from the force measurement (gravity of SO), the exact trajectory tracking is not possible. However, for the stable performance of robot, these are acceptable results due to the purpose of gross motion control.

*4.3.3 Contact with Stiff Environment without SO.* In this simulation, Fig (4.3) (b) is applied to see the robot performance under the contact with the environment. Fig (4.6) shows the simulation results of the manipulator without an SO in contact with stiff environment. The main objective is to simulate the stable robot performance under the contact with environment in the presence of uncertainties such as unexpected workspace limits or external forces. Let the initial joint position be the same used in free space tracking, i.e.,  $\Theta = [90^\circ \ -90^\circ \ 0^\circ]^T$ , and the wall located at  $p_{xe} = 55$  (cm). When the manipulator hits the wall, the impedance control attempts to pull back the manipulator while the position errors generate the torque commands to approach the desired position. By these conflicting commands, the manipulator will stay at the surface of wall until the human recognizes the existence

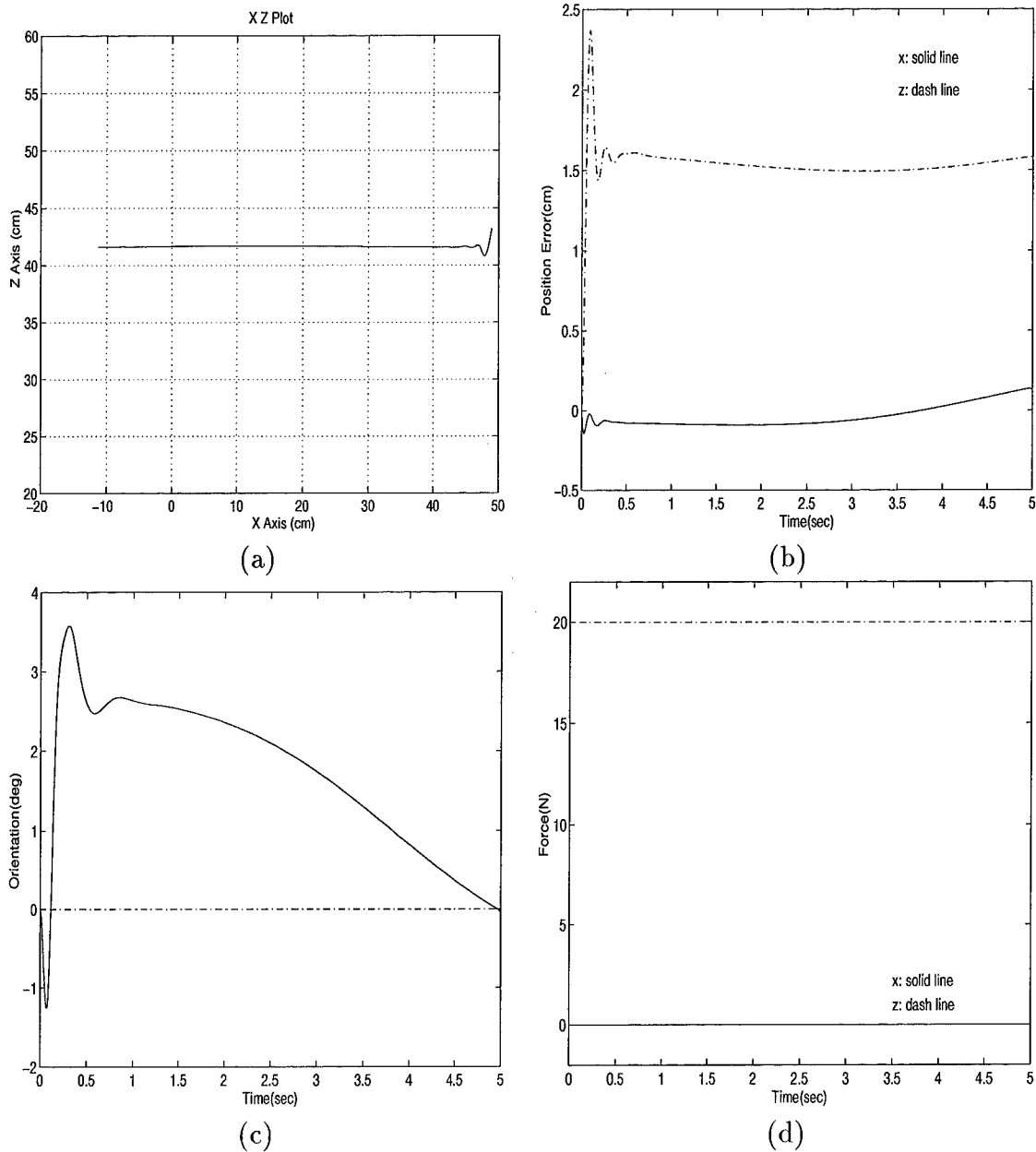


Figure 4.5 Gross Motion Control in Free Space Motion Tracking with SO : (a) Measured position (manipulator moves left along  $x$ -axis while maintaining the distance on  $z$ -axis), (b) Error between desired and measured position, (c) End-effector orientation ( $\theta_{total} = 0^\circ$ ), (d) Force measurement

of the wall and provides no position command to the teleoperator. Fig (4.6) (c) shows the manipulator orientation within their joint ranges. Due to the conflicting commands, the end-effector orients in an upward direction as a result of impedance control as the measured force in Fig (4.6) (d) is increased. The measured force is obtained by the following relationship.

$$F_x = \mathbf{K}_e(p_x - p_{xe}) \quad (4.15)$$

where  $\mathbf{K}_e$  is the environmental stiffness gain listed in Table (4.2) and  $p_x - p_{xe}$  is the difference between the measured position of end effector and environment in  $x$ -axis.

*4.3.4 Contact with Stiff Environment with SO.* This section simulates the overall external force effects that we can account in the simulation. Including previous planning, a 20 (N) SO is added to the end of robot as a load. The force in both  $x$  and  $z$  directions can be seen during the motion and how the impedance controller overcomes these effects over the time history. As shown in Fig (4.7) (a), the motion is relatively coarse compared with the other cases. However, the performance is still acceptable to show the controller being stable. The orientation of end-effector shows the combined performance of the free space and contact simulations. This simulates the worst case of robot operation by the teleoperator.

For both contact cases (with and without SO), a single environmental stiffness,  $\mathbf{K}_e$ , was applied to the controller. In true applications, this  $\mathbf{K}_e$  is a variable of the environment over which we have no control. Unfortunately,  $\mathbf{K}_e$  has an effect on system performance. The advantage of a teleoperator is that the operator will in all likelihood know what he is contacting and can select the gain accordingly.

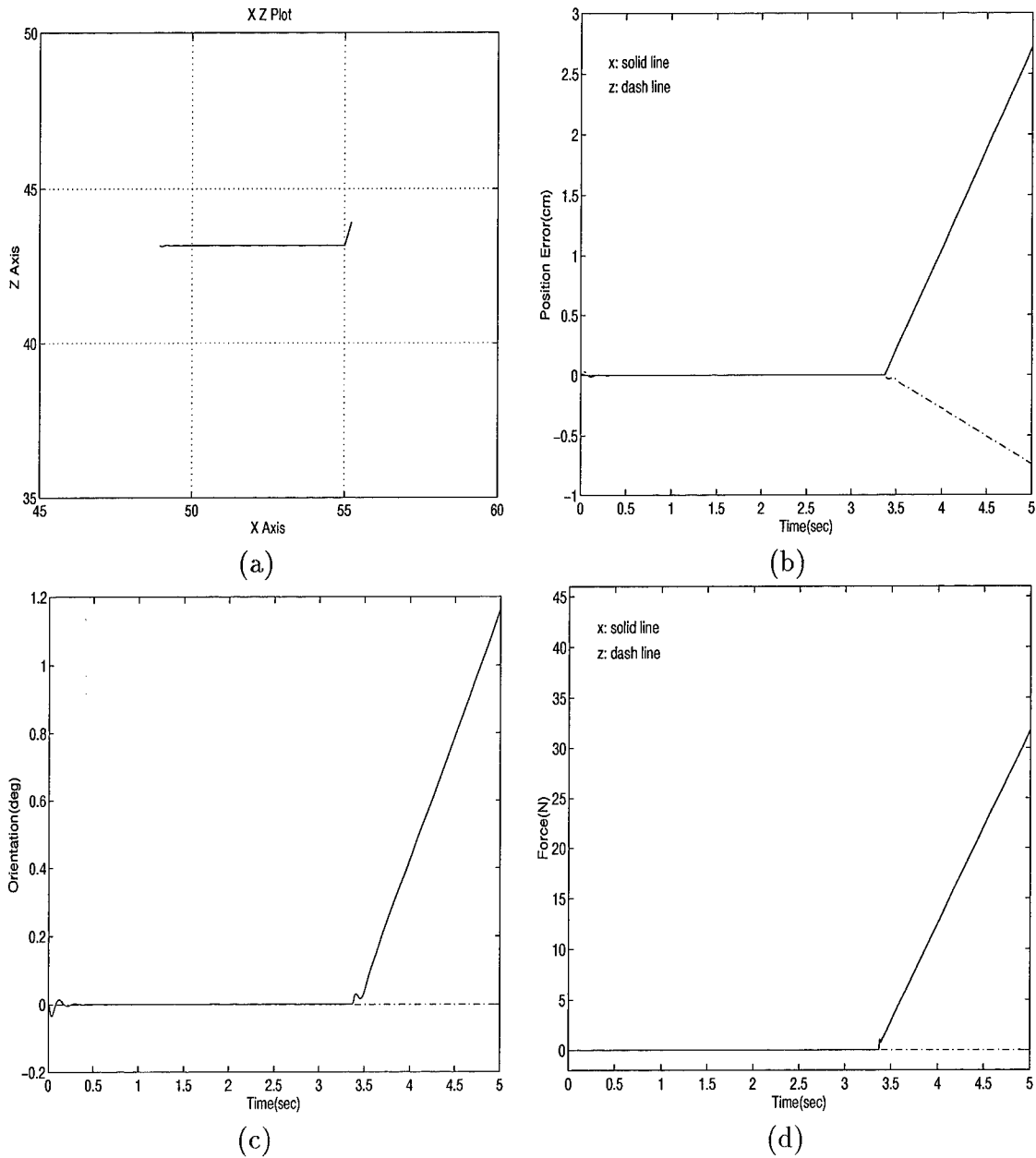


Figure 4.6 Gross Motion Control in Contact with Stiff Environment without SO : (a) Measured position (manipulator tries to move into the wall which lies along  $x$ -axis and maintains the same position unless the stiff environment allows the manipulator to move), (b) Error between desired and measured position, (c) End-effector orientation ( $\theta_{total} = 0^\circ$ ), (d) Force measurement

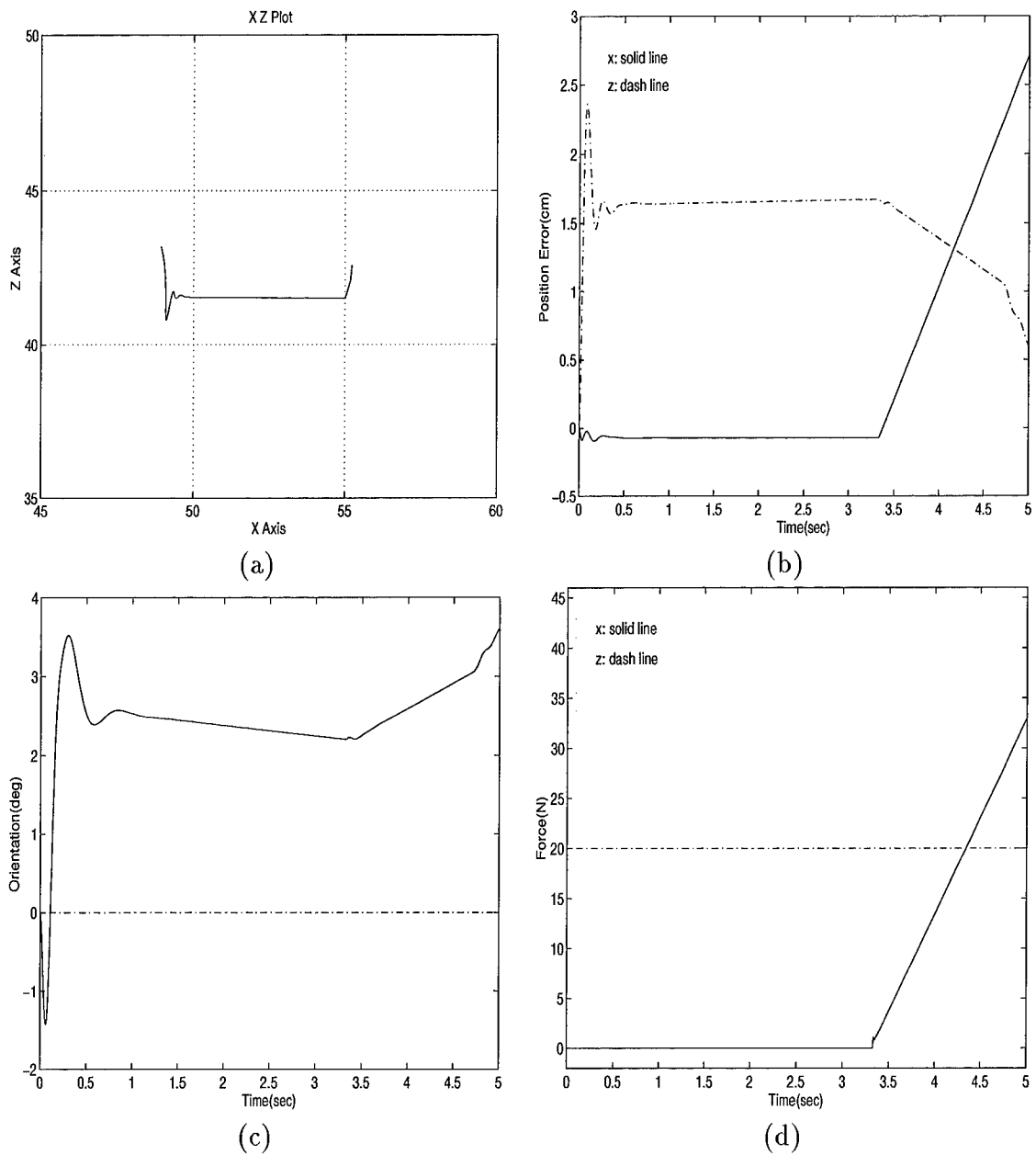


Figure 4.7 Gross Motion Control in Contact with Stiff Environment with SO : (a) Measured position (manipulator tries to move into the wall which lies along  $x$ -axis and maintains the same position unless the stiff environment allows the manipulator to move), (b) Error between desired and measured position, (c) End-effector orientation ( $\theta_{total} = 0^\circ$ ), (d) Force measurement

#### 4.4 *Fine Motion Control Simulation*

To simulate the fine motion control scheme described in the previous chapter for the simplest case, the same structure described in Section (4.2) is applied in this section. Since the fine motion control is restricted in the palm frame, the coordinate is changed from  $z$ -axis to  $y$ -axis. Based on the hierarchical control structure, the fine motion control inputs are the forces fed by the wrist force sensor attached to the robot arm. In addition to these forces, the fine motion control attempts to respond due to the object disturbance forces to achieve the desired contact force. We assume that the optimal contact force for each finger can be obtained using Mark Hunter's Contact Force Allocation algorithm [23] and the contact force can be explicitly fed back to the fine motion controller. Thus, each fingertip actuator operates under force control and the finger actuators operate under position control due to the contact position deviation resulting from the force control. The fine motion control is divided into two steps: the first step is to simulate one-finger motion control as an intermediate goal, and the second step is to simulate a coordinated two-finger motion control as the integration goal. The diagrams of both simulations are shown in Fig (4.8). Both diagrams show only the SO because the fine motion control operates in the SO with palm as a base frame.

The derivation of the environmental stiffness and force to position gains are the same procedure described in Section (4.3). Since the same kinematic and dynamic structure are used in this simulation, the PID gains for the finger joints are also equal to the ones of gross motion control. This assumes that each finger is a planar configured PUMA robot. Table (4.3) shows the gains used in the fine motion control simulations.

There are some assumptions required in this simulation due to the limitations of MATLAB SIMULINK environment. The assumptions are:



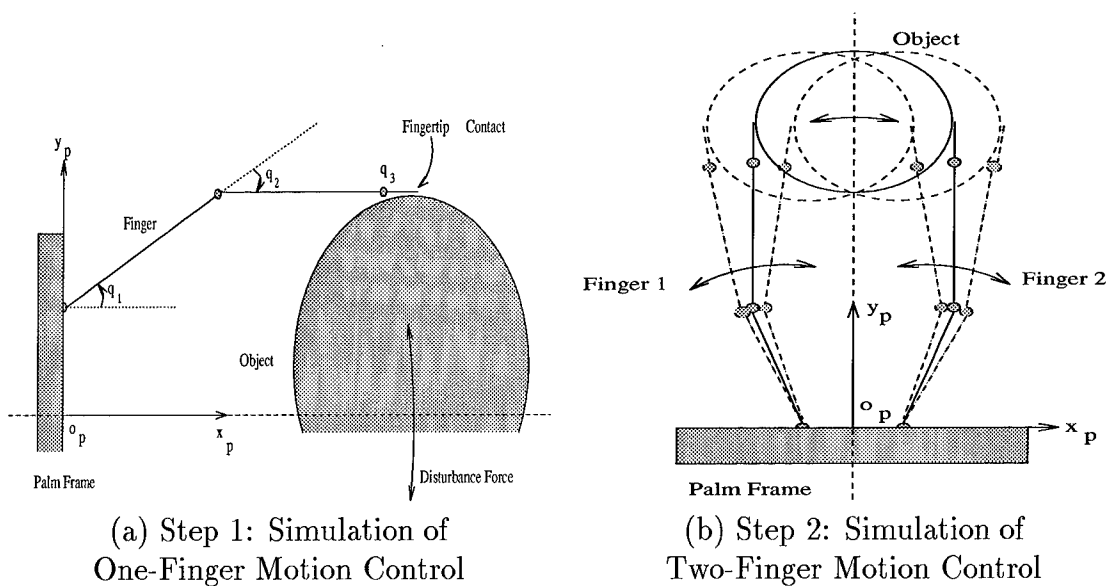


Figure 4.8 Diagram of Fine Motion Control ( $-y_p$  is the direction of gravitational effect)

	Joint $i$	2	3
PID gain set	$K_p$	5000	1500
	$K_I$	5	5
	$K_v$	116	25
environmental stiffness gain $K_e$		$x$ -axis	$y$ -axis
		100	100
force to position gain $K_{fp}$		0.003	

Table 4.3 Nominal Gain set of Applied to the Fine Motion Control Simulation

1. The FAS responds perfectly when the finger joints operate to keep the FAS joint centered in its range of actuator motion. When the finger joints operate to provide the FAS centering action due to the response of FAS, the FAS also responds simultaneously due to the movements of finger joints. This assumption allows the above physical phenomenon which can not be obtained through the simulation.
2. The FAS operates faster than the finger joints. As mentioned in Section (3.4) and Section (3.5.2), the FAS should response fast enough to zero

out the force error by each sample period. This is shown in the results of simulations of this section.

3. The initial contact is already obtained and initial contact force is also provided.
4. CFA algorithm provides an optimal contact force to each fingertip.

The MATLAB M-files and SIMULINK block diagram used are listed in Appendix B.

#### 4.4.1 Step 1: One-Finger Motion Control Simulation.

4.4.1.1 *Performance using Nominal Gain Set.* One-finger simulation is the intermediate step for the two-finger simulation in next section and the demonstration of the control division scheme in Chapter 5. The desired force input is  $\mathbf{f}_{id} = [0 \ -10]^T$  (N) and the object disturbances in this simulation are the step input with magnitude 0.5 (cm) at  $t = 0$  (sec) to verify the stability of finger performance and the normally distributed random disturbance input with mean zero and  $3\sigma = 0.25$  (cm) for simulating the general finger performance under varying disturbance. The object displacement as an object disturbance is applied here because certain disturbance forces can cause to the object to move so that it can represent the hardware implementation. The disturbance inputs are applied only in the  $y$ -axis shown in Fig (4.8) (a). Let the initial joint positions of finger and fingertip be,  $\mathbf{q}_i = [30^\circ \ -30^\circ \ 0^\circ]^T$ , and be the contact position at  $t = 0$ . Thus, the initial contact position is  $\mathbf{r}_i = [0.8633 \ 0.5199]^T$  (m).

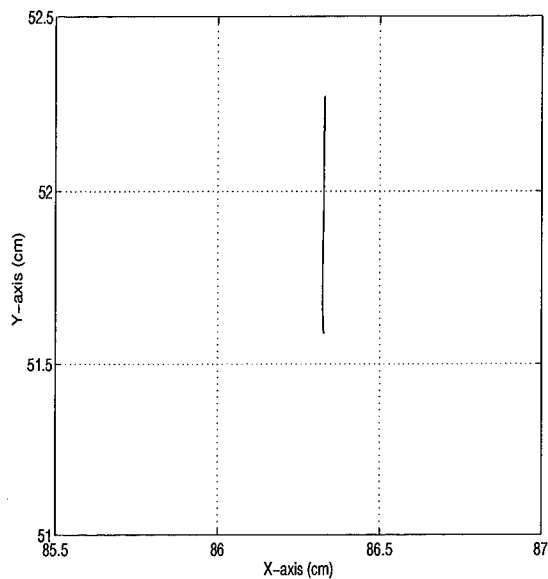
Fig (4.9) and (4.10) show the results of finger performance using step input. By applying the constant disturbance input on the object for entire time history, the FAS responds adequately to track the desired contact force as shown in Fig (4.9). The contact position deviation due to this result is added to the current contact position as a new desired position input for the movement of finger to provide an

FAS centering action. This result is shown in Fig (4.9) (a). Once the FAS is centered of its joint range, in this case  $t = 15$  (sec), the finger does not move further from its initial position because the finger joints respond to provide the FAS centering action whenever disturbance forces exist. In addition to this, the FAS has the limit of its joint range  $\pm 10^\circ$ , called the dead-band, to prevent instantaneous reaction due to the abrupt disturbance input. In Fig (4.10) (a) and (b), Joint 1 and 2 converge asymptotically to their equilibrium points while responding to the disturbance input. Fig (4.10) (c) shows the convergence of FAS to asymptotic equilibrium and also the FAS centering result.

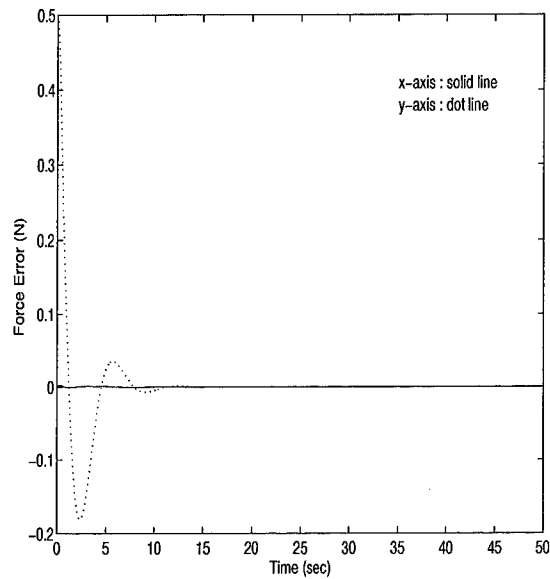
Fig (4.11) and (4.12) show the results of finger performance using the random disturbance input applied only in the  $y$ -axis. The movement of finger is quite close to the step disturbance input. As shown in Fig (4.11) (c), even though the random disturbance input is applied to the object, the FAS is centered in its range. The range of contact position is within  $50.3 \sim 52.8$  (cm) in the  $y$ -axis and  $86.2 \sim 86.5$  (cm) in the  $x$ -axis direction (Fig (4.11) (a)). Different from the step disturbance input, the force error in Fig (4.11) (b) is much larger than the case of step disturbance input because the varying disturbance input causes larger joint rate and FAS response. Fig (4.12) (a) and (b) show the results of Joint 1 and 2 stabilities converged to their equilibrium point, and Fig (4.12) (c) shows the results of FAS stability and centering action under the random disturbance input.

*4.4.1.2 Compliance.* This section simulates fingertip compliance motion control by varying the  $\mathbf{K}_{fp}$  and  $\mathbf{K}_e$  defined by Fig (3.7). The sets of  $\mathbf{K}_{fp}$  and  $\mathbf{K}_e$  are shown in Table (4.4).

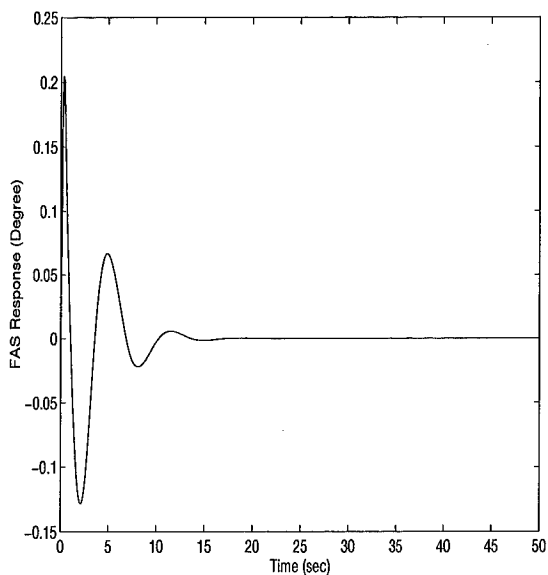
The gains in Table (4.4) are heuristically selected to show the comparison of fine motion control between the relative stiff and compliant environment. To get the desired fingertip contact force under the stiff or compliant environment (object), the stiffness gain,  $\mathbf{K}_{fp}$ , needs to be changed interactively by the operator. Fig (4.13)



(a) Measured contact position

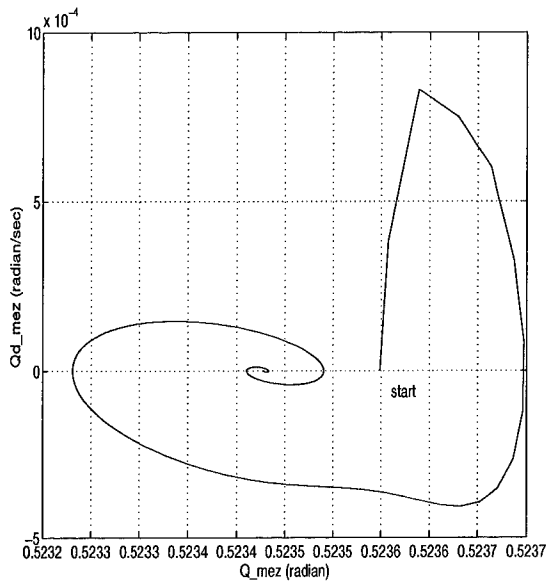


(b) Force error

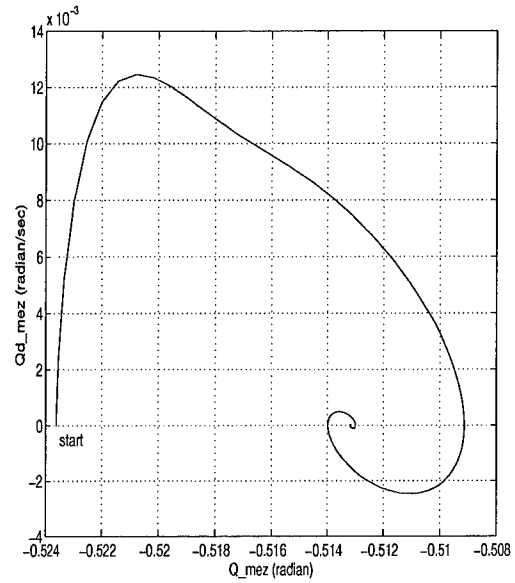


(c) FAS measurement

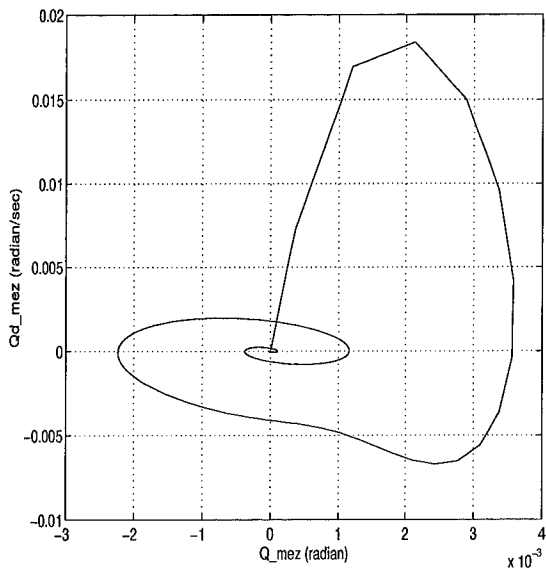
Figure 4.9 One-Finger Motion Control Response under Disturbance Force Input (Step Input with maximum 0.5 (cm))



(a) Joint 1

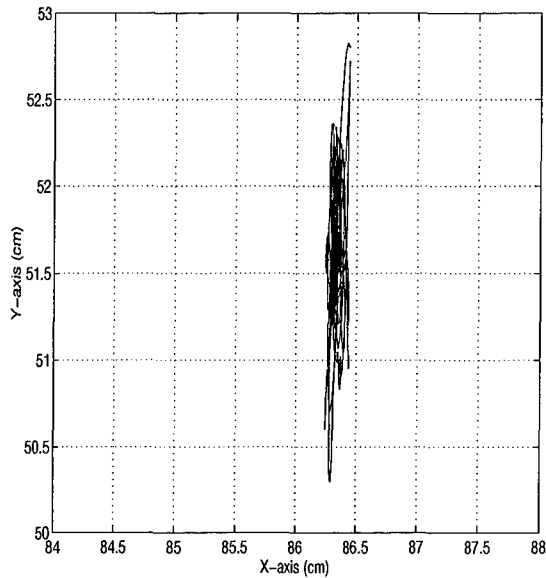


(b) Joint 2

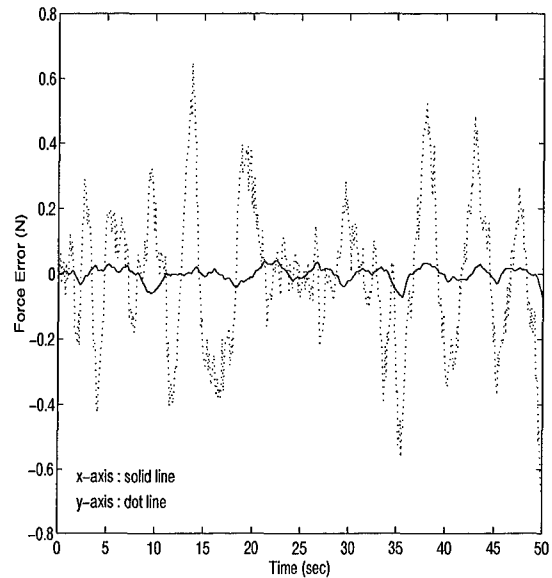


(c) FAS

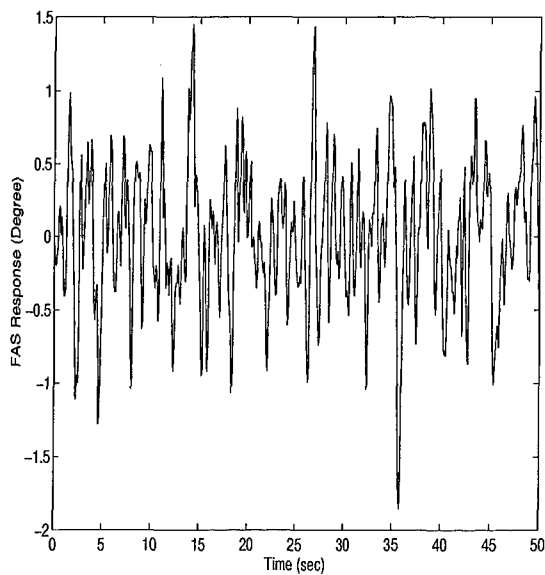
Figure 4.10 Joint Stability using Phase Plot for Disturbance Force Input (Step Input with maximum 0.5 (cm))



(a) Measured contact position

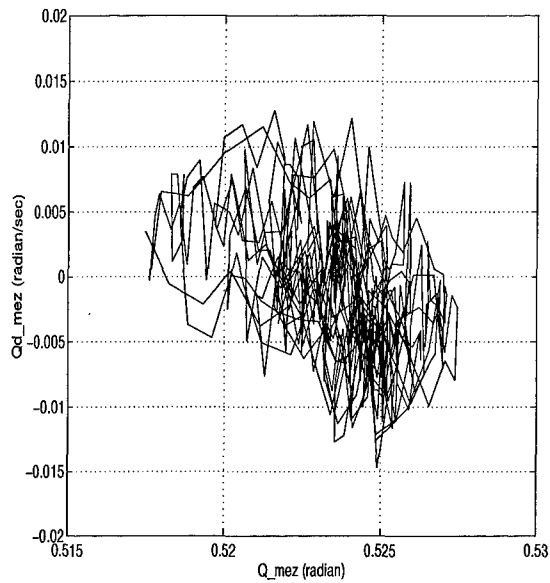


(b) Force error

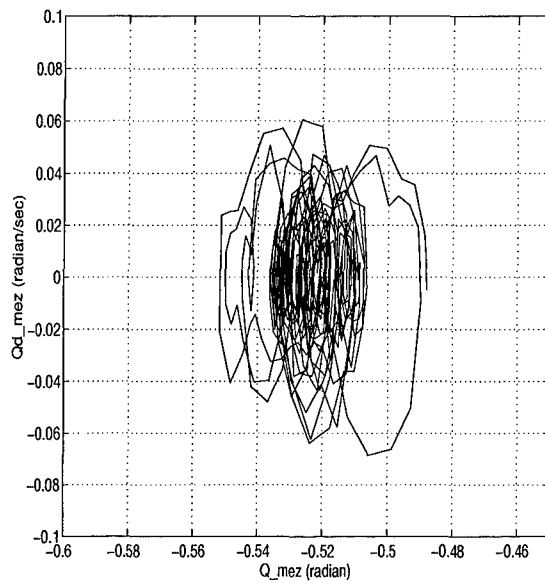


(c) FAS measurement

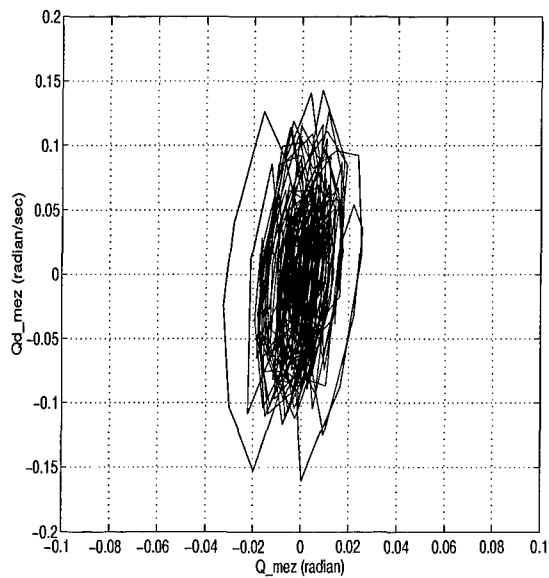
Figure 4.11 One-Finger Motion Control Response under Disturbance Input (Random Input (Normal Distribution) with Mean zero and  $3\sigma = 0.25$  (cm))



(a) Joint 1



(b) Joint 2



(c) FAS

Figure 4.12 Joint Stability using Phase Plot for Disturbance Input (Random Input (Normal Distribution) with Mean zero and  $3\sigma = 0.25$  (cm))

Experiment set	$\mathbf{K}_{fp}$	$\mathbf{K}_e$
Set # 1	0.001	50
Set # 2	0.001	150
Set # 3	0.006	150
Set # 4	0.006	50

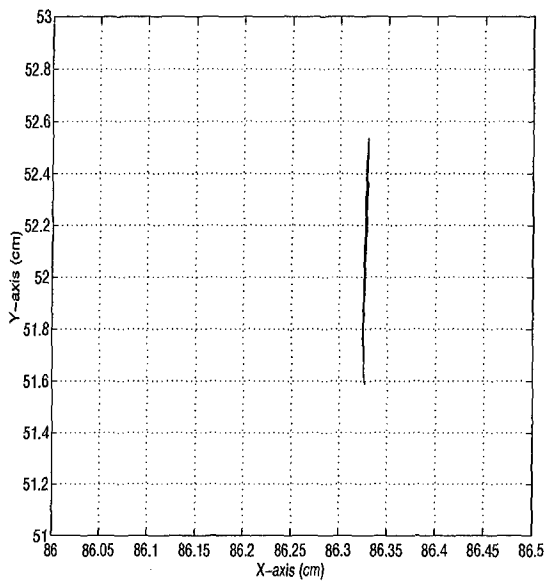
Table 4.4 Experiment Sets of Fingertip Compliance

through (4.16) show the response of fingertip using the sets of these gains and show how to combine between  $\mathbf{K}_{fp}$  and  $\mathbf{K}_e$ . For the stiff environment (object), the stiffness gain needs to be decreased as shown in Fig (4.14) and (4.15). For the compliant environment (object), the stiffness gain should be increased as shown in Fig (4.13) and (4.16). In the implementation phase, the compliant motion control will use metal and soft surfaces.

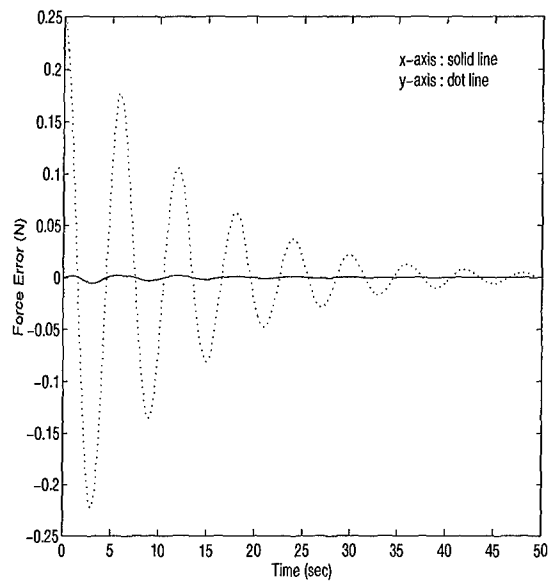
*4.4.2 Step 2: Two-Finger Motion Control Simulation.* A two-finger motion control is the integrating step using the previous section's motion control to simulate the object grasp stability and finger performance under unexpected object disturbance inputs. The CFA algorithm is used for the contact force distribution using the result of object wrench due to the object gravitational load, i.e.,  $\mathcal{F} = [0 \ 10 \ 0 \ 0 \ 0 \ 0]^T$ . The step disturbance input with magnitude 0.5 (cm) is used for the first experiment. The normally distributed random disturbance input with mean zero and  $3\sigma = 0.25$  (cm) is used for the second experiment to verify the finger performances under general disturbance inputs. These disturbances are applied only in the  $x$ -axis shown in Fig (4.8) (b). The overall block diagram applied in these experiments is shown in Fig (4.17).

As shown in Fig (4.8) (b), two fingers grasped an object are straight upward to show the object gravity effect of both fingers. Fig (4.18) shows the coordinate frames of each fingertip contact, object and palm. All of the selected frames are based on the planar case of robot. We choose the  $y$ -axis of both contact frames pointing in the direction of the inward surface normal at the points of contacts. For

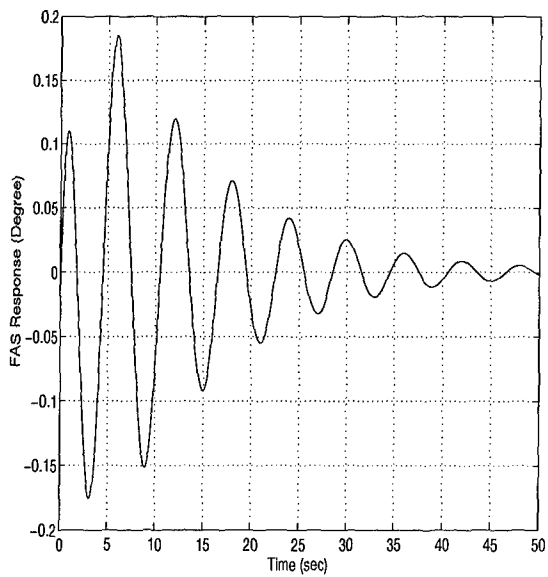




(a) Measured contact position

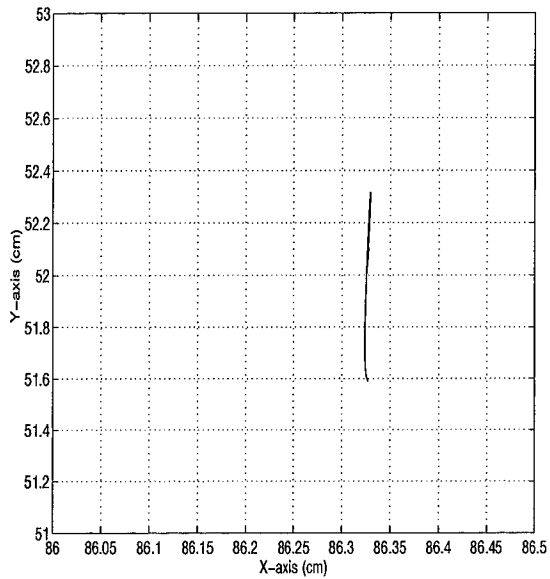


(b) Force error

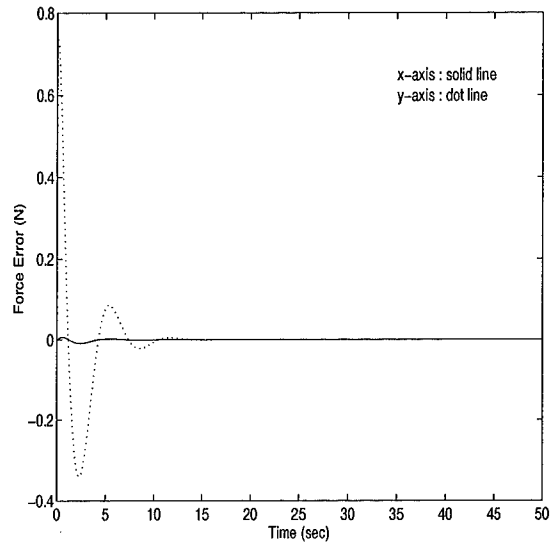


(c) FAS measurement

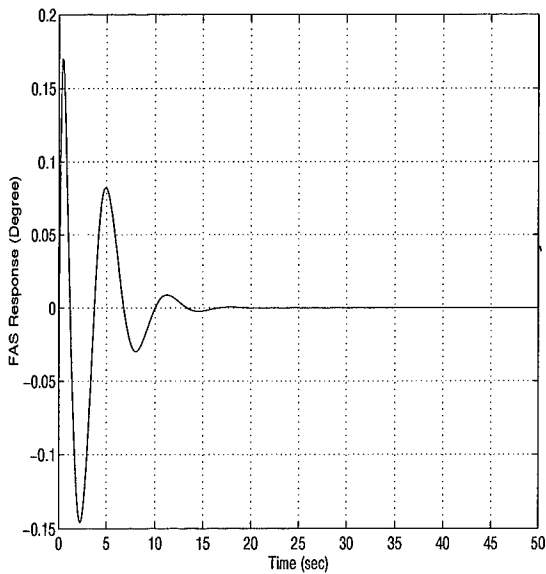
Figure 4.13 Finger Response using  $K_{fp} = 0.001$  and  $K_e = 50$



(a) Measured contact position

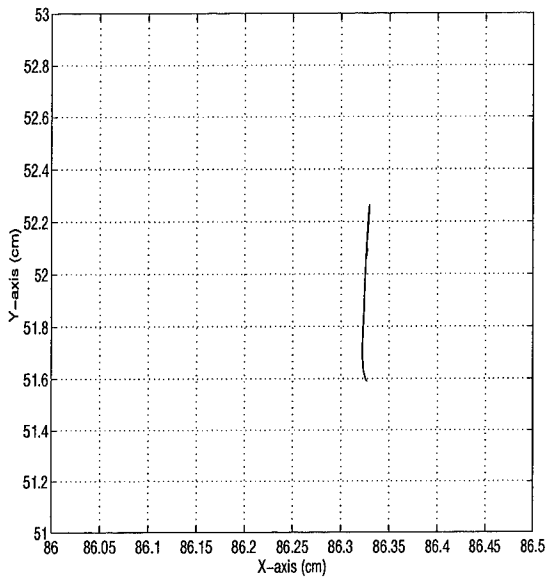


(b) Force error

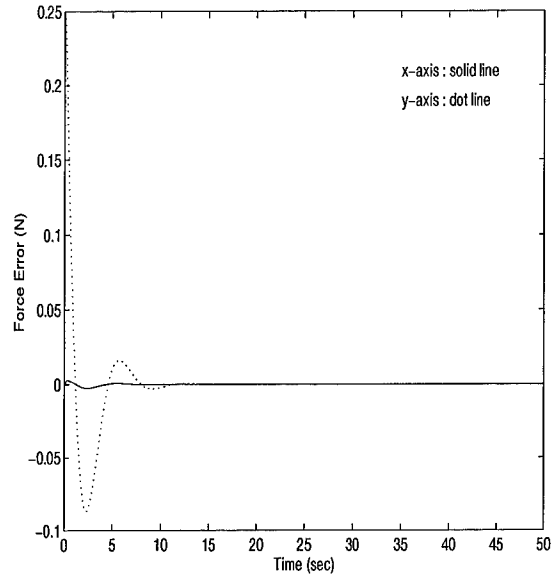


(c) FAS measurement

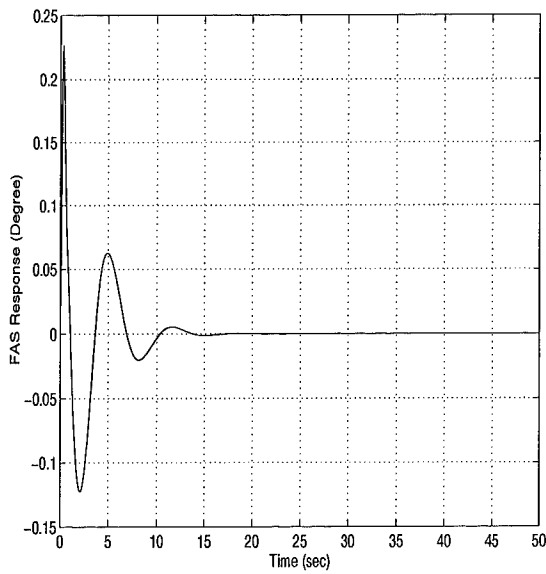
Figure 4.14 Finger Response using  $K_{fp} = 0.001$  and  $K_e = 150$



(a) Measured contact position

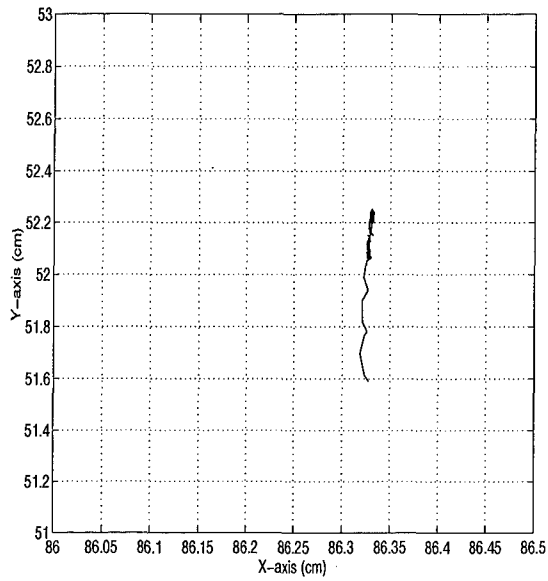


(b) Force error

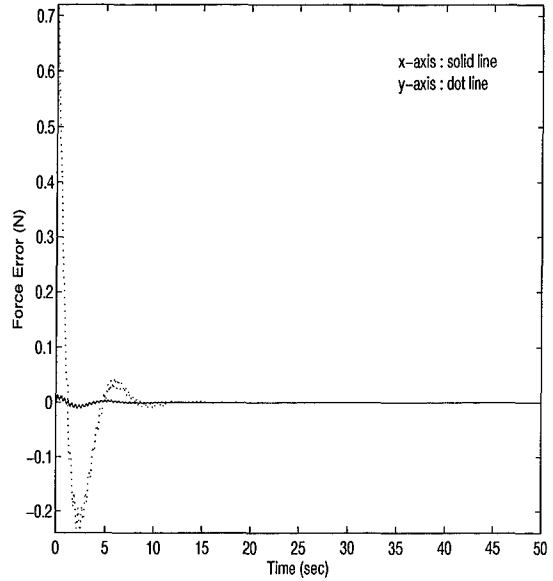


(c) FAS measurement

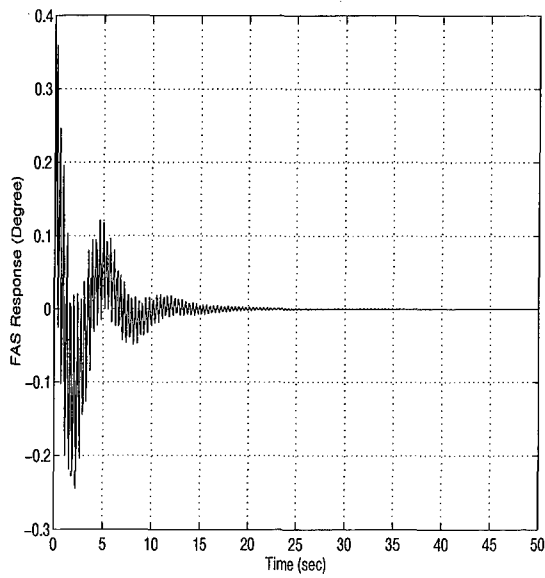
Figure 4.15 Finger Response using  $K_{fp} = 0.006$  and  $K_e = 150$



(a) Measured contact position



(b) Force error



(c) FAS measurement

Figure 4.16 Finger Response using  $K_{fp} = 0.006$  and  $K_e = 50$

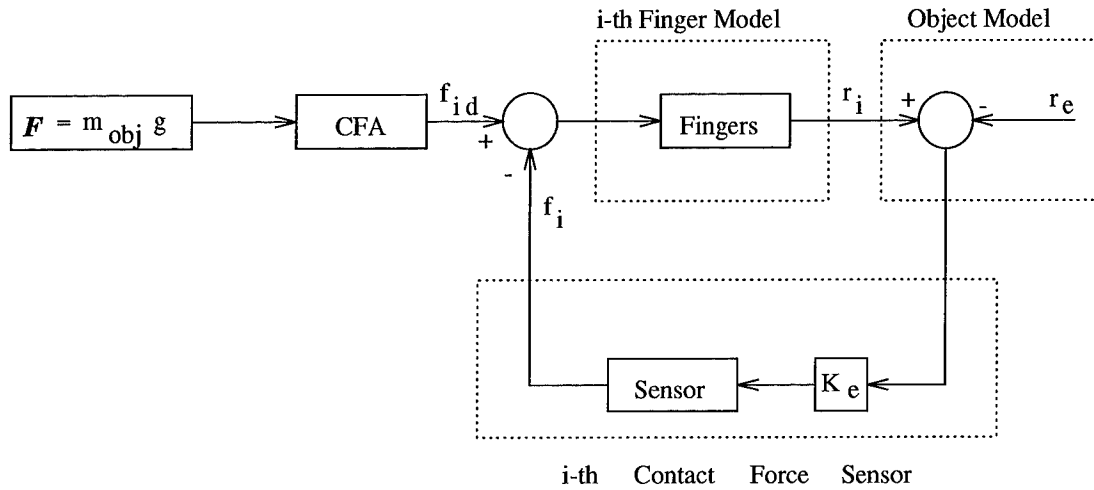


Figure 4.17 Block Diagram of Simulation for Two-Finger Motion Control

simplification, the directions of Finger # 2 contact and object frames are same and Finger # 1 contact frame is rotated by  $-90^\circ$ . By the assumption stated in Section (3.3), the directions of both contact and object frames are not changed. The initial joint positions of fingertip contacts are  $\mathbf{q}_1 = [30^\circ \ -30^\circ \ 0^\circ]^T$  and  $\mathbf{q}_2 = [-30^\circ \ 30^\circ \ 0^\circ]^T$  and the initial contact forces in the contact frames due to the object gravity are  $\mathbf{f}_1 = [12.5 \ 5]^T$  (N) for finger one and  $\mathbf{f}_2 = [-12.5 \ 5]^T$  (N) for finger two. The initial contact positions with respect to the palm frame are  $\mathbf{r}_1 = [-0.5159 \ 0.8633]^T$  (m) and  $\mathbf{r}_2 = [0.5159 \ 0.8633]^T$  (m). All of disturbance inputs are applied only in the  $x$ -axis with respect to the palm frame. The assumptions used in one-finger motion control simulation are also used in this simulation. Since the same kinematic and dynamic structures of one-finger motion control simulation are used here, the same gain sets are also applied.

*4.4.2.1 Determination of Grasping Matrix.* The determination of grasping matrix follows the method described by Nakamura [5]. The force on the object frame is required in the contact frame to do the force control in the FAS. If

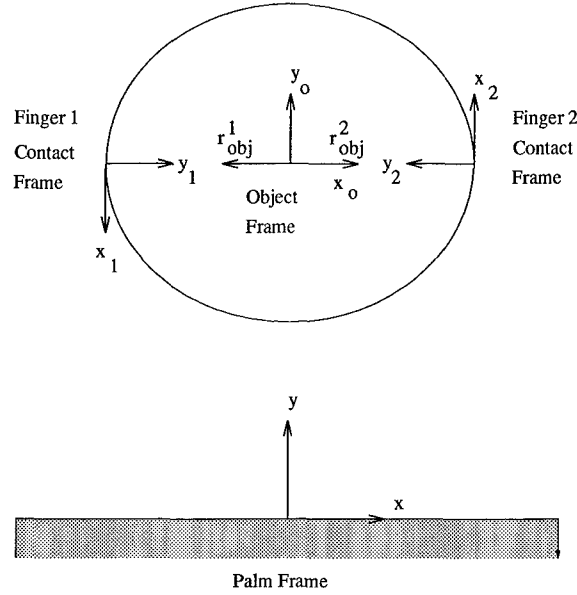


Figure 4.18 Coordinate Assignments of each Fingertip Contact, Object and Palm Frames for Two-Finger Motion Control Simulation

the force on the object frame is obtained as

$$\mathcal{F}_{obj} = \mathbf{W}\mathbf{f}_c \quad (4.16)$$

where  $\mathcal{F}_{obj}$  is the force represented on the object frame,  $\mathbf{W}$  is the grasping matrix, and  $\mathbf{f}_c = [\mathbf{f}_1 \ \mathbf{f}_2]^T$  where  $\mathbf{f}_1$  and  $\mathbf{f}_2$  are the fingertip contact forces of contact 1 and 2. From the Fig (4.18), the  $i$ -th contact position with respect to the object frame is

$$\mathbf{r}_{obj}^1 = [r_{objx}^1 \ 0 \ 0] \quad (4.17)$$

$$\mathbf{r}_{obj}^2 = [r_{objx}^2 \ 0 \ 0] \quad (4.18)$$

Thus, the grasping matrix for each fingertip is obtained by following equations

$$\mathbf{W} = \begin{bmatrix} \mathbf{E}_1 & \mathbf{E}_2 \\ \hat{\mathbf{r}}_{obj}^1 & \hat{\mathbf{r}}_{obj}^2 \end{bmatrix} \quad (4.19)$$

$$\hat{\mathbf{r}}_{obj}^i = \begin{bmatrix} 0 & -r_{objz}^i & r_{objy}^i \\ r_{objz}^i & 0 & -r_{objx}^i \\ -r_{objy}^i & r_{objx}^i & 0 \end{bmatrix} \quad (4.20)$$

where  $\hat{\mathbf{r}}_{obj}^i$  is the skew symmetric matrix of  $i$ -th contact position vector with respect to the object frame. The calculated grasping matrix is

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -r_{objx}^1 & 0 & 0 & -r_{objx}^2 \\ 0 & r_{objx}^1 & 0 & 0 & r_{objx}^2 & 0 \end{bmatrix} \quad (4.21)$$

Using the result of Eq (4.21), the fingertip contact forces can be determined as

$$\mathbf{f}_c = \mathbf{W}^\# \mathcal{F}_{obj} + \mathbf{f}_{int} \quad (4.22)$$

where  $\mathbf{f}_{int}$  is the nullspace solution of pseudoinverse.

*4.4.2.2 Step Disturbance Input.* The nominal gains listed in Table (4.3) are applied for simulating the stability of two-finger motion under the step disturbance input applied to the object. The results of this simulation are similar to the results of the one-finger motion simulation. As stated in Chapter 3, the movement of one finger should cause the other finger movements. Thus, whenever the disturbance input is applied to the object, both fingers should respond simultaneously. The important thing is that both fingers should follow the direction of disturbance force input to achieve the desired contact force and the FAS centering action, and, in addition to these, both finger should grasp the object without losing contact. The constraint is that the distance between both finger contact positions should be constant. The distance of two fingertips for all time history is shown in

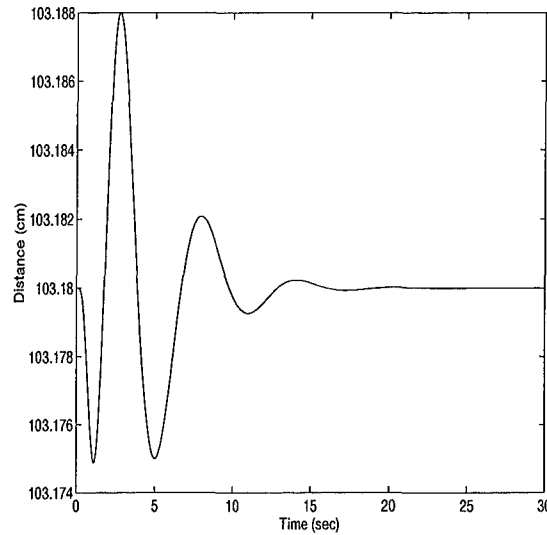


Figure 4.19 The Distance of Two Fingertips under Step Disturbance Input (Magnitude 0.5 (cm))

Fig (4.19). Compared with the original distance of two fingertips (103.18 (cm)), the maximum distance error range is about 0.013 (cm) which would cause the object to slip or be dropped. The solution to reduce this error will be discussed in context of the normally distributed random disturbance experiment in the next section.

Fig (4.20) and (4.22) show the results of finger 1 and finger 2 performances respectively. Different from the one-finger motion simulation, the disturbance input is applied to  $x$ -axis in the palm frame. The control purpose is that the positions on  $y$ -axis of both fingers are constant while the positions on  $x$ -axis are varied due to the responses of both FAS's. Fig (4.21) and (4.23) show the stabilities of both finger 1 and finger 2 joints respectively. In Fig (4.21) and (4.23), we show that the joint velocity of FAS (Fig (4.21) (c) and (4.23) (c)) is larger than the joint velocities of finger (Fig (4.21) (a) and (b) and (4.23) (a) and (b)). Fig (4.21) and (4.23) also show the stabilities of every joint which converge their asymptotic equilibrium points while the FAS joint operates in its near centering location.

Fig (4.21) (a) and (4.23) (a) have different responses that the movement of Joint 1 in Fig (4.21) (a) has about  $1 \times 10^{-4}$  (radian) while the movement of Joint 1 in



Fig (4.23) (a) has about  $1.2 \times 10^{-4}$  (radian). Since the direction of step disturbance input was applied to  $+x$ -axis, the measured contact force in  $y$ -axis with respect to contact frame was decreased in Finger 1 (Fig (4.20) (b)) and increased in Finger 2 (Fig (4.22) (b)), but the contact force errors in  $x$ -axis for both fingers were shown differently. This is the reason that Joint 1 and 2 of both fingers do not have similar results.

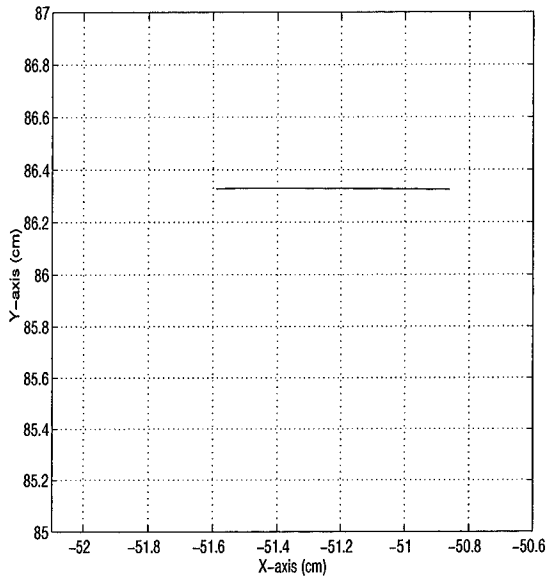
*4.4.2.3 Normally Distributed Random Disturbance Input.* This experiment shows the responses of each finger under the normally distributed random disturbance input applied on the object. The applied disturbance has the mean zero and  $3\sigma = 0.25$  (cm). Therefore, the disturbance is applied on the object in the  $\pm x$ -axis with respect to the palm frame. Since the direction of disturbance input is instantaneously changed, the magnitudes of every joint velocities should be larger than the results of previous experiment. This causes the larger FAS and finger responses. The distance error of two fingertips in Fig (4.24) (a) is about total 0.15 (cm) which is about 10 times larger than previous experiment. The original distance between two fingertips grasped an object is 103.18 (cm). For the non-slip constraint, we desire the distance of two fingertips during the operation should be less than or equal to 103.18 (cm). If  $i$ -th contact force,  $\mathbf{f}_i$ , is specified as

$$\mathbf{f}_i = \mathbf{f}_{ext}^i + \mathbf{f}_{int}^i \quad (4.23)$$

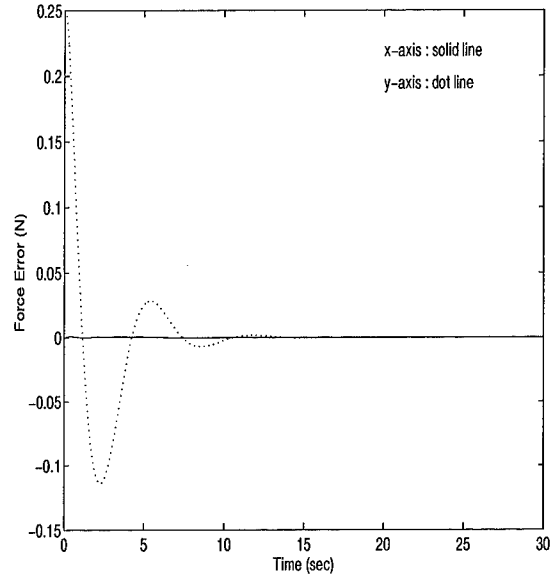
where  $\mathbf{f}_{ext}^i$  is the  $i$ -th contact external force applied by the external forces (in this case, the gravitational force due to the object load and disturbance force are only applied) and  $\mathbf{f}_{int}^i$  is the  $i$ -th contact internal force to grasp the object. By the following equation,

$$\mathcal{F}_{obj} = \mathbf{W}\mathbf{f}_c \quad (4.24)$$

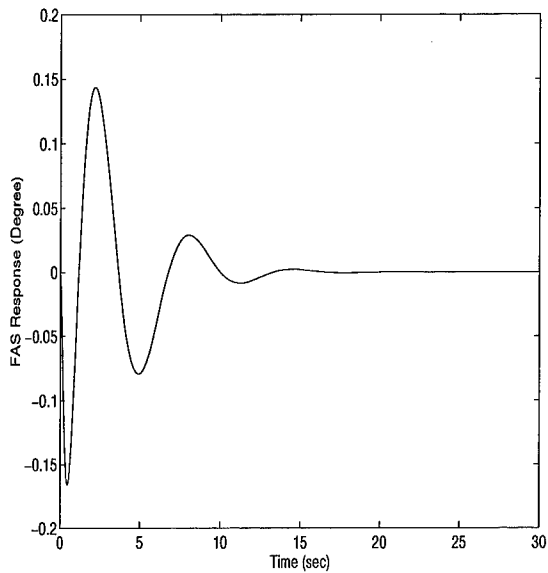
the internal force,  $\mathbf{f}_{int}^i$ , is mapped into the null space of  $\mathbf{W}$ . Since the grasp stability can be obtained by applying as much internal force as the object can withstand, we



(a) Measured contact position

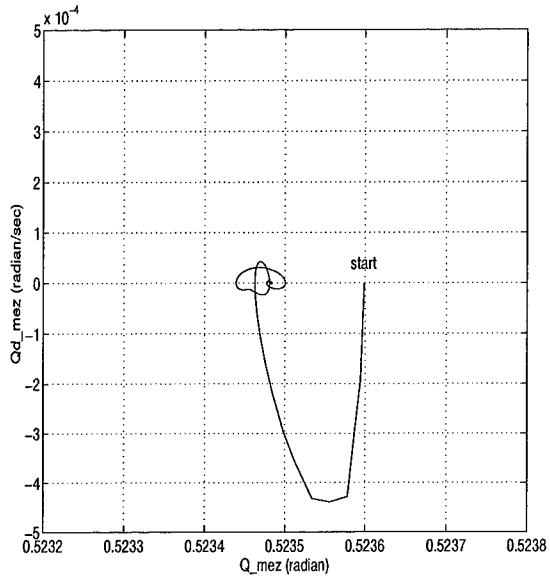


(b) Contact force error

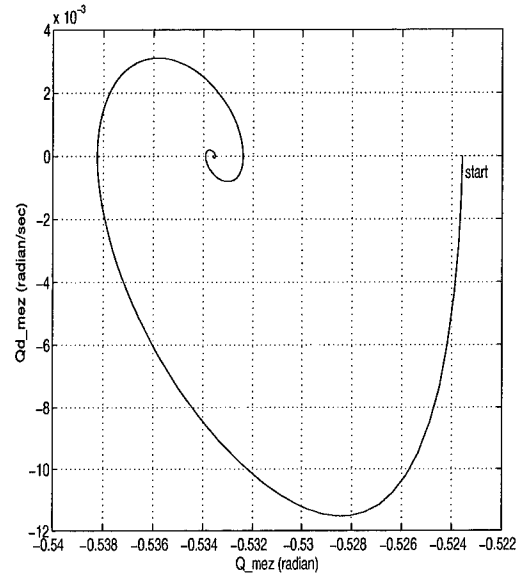


(c) FAS measurement

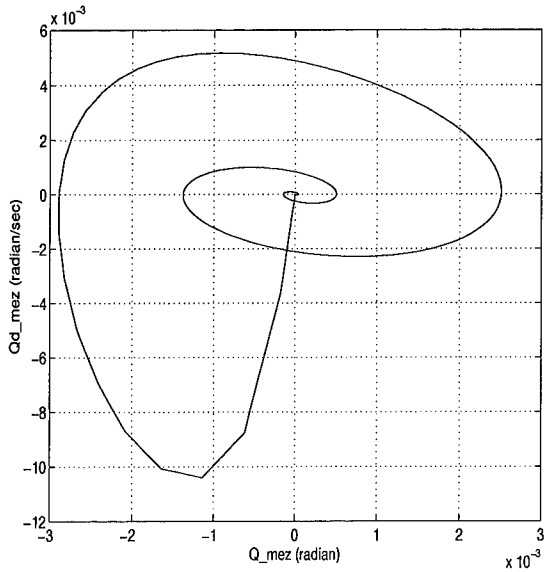
Figure 4.20 Two-Finger Motion Control - Finger 1 Performance with Step Disturbance Input (Magnitude 0.5 (cm))



(a) Joint 1

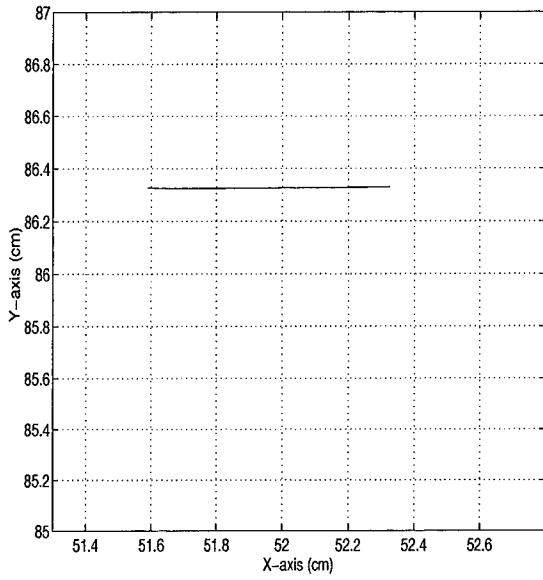


(b) Joint 2

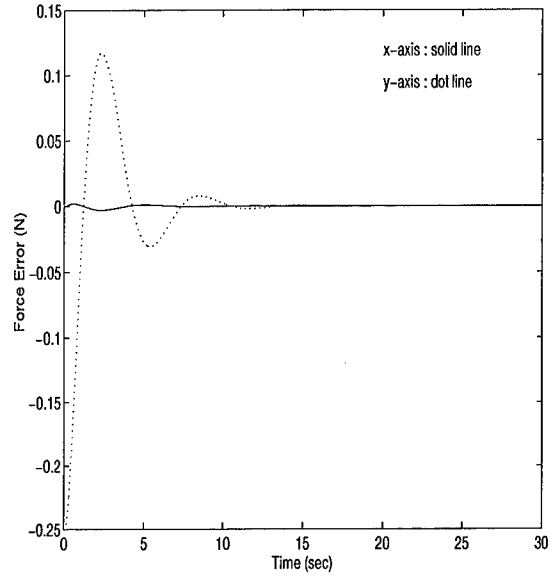


(c) FAS

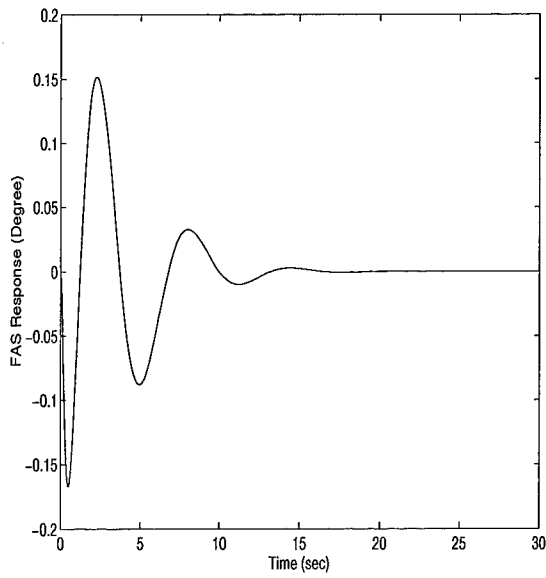
Figure 4.21 Two-Finger Motion Control - Finger 1 Joint Stability with Step Disturbance Input (Magnitude 0.5 (cm))



(a) Measured contact position

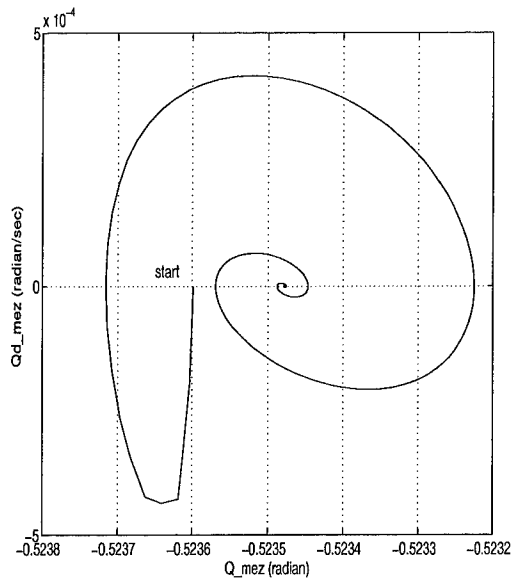


(b) Contact force error

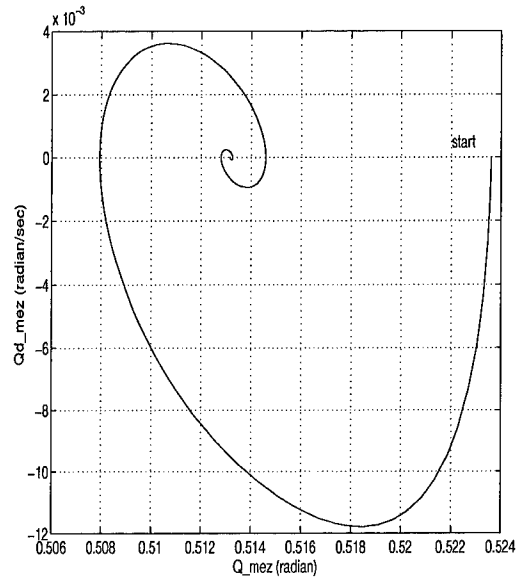


(c) FAS measurement

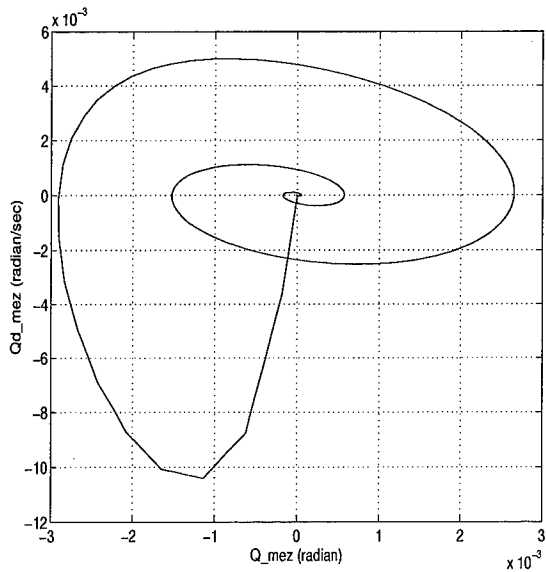
Figure 4.22 Two-Finger Motion Control - Finger 2 Performance with Step Disturbance Input (Magnitude 0.5 (cm))



(a) Joint 1



(b) Joint 2



(c) FAS

Figure 4.23 Two-Finger Motion Control - Finger 2 Joint Stability with Step Disturbance Input (Magnitude 0.5 (cm))

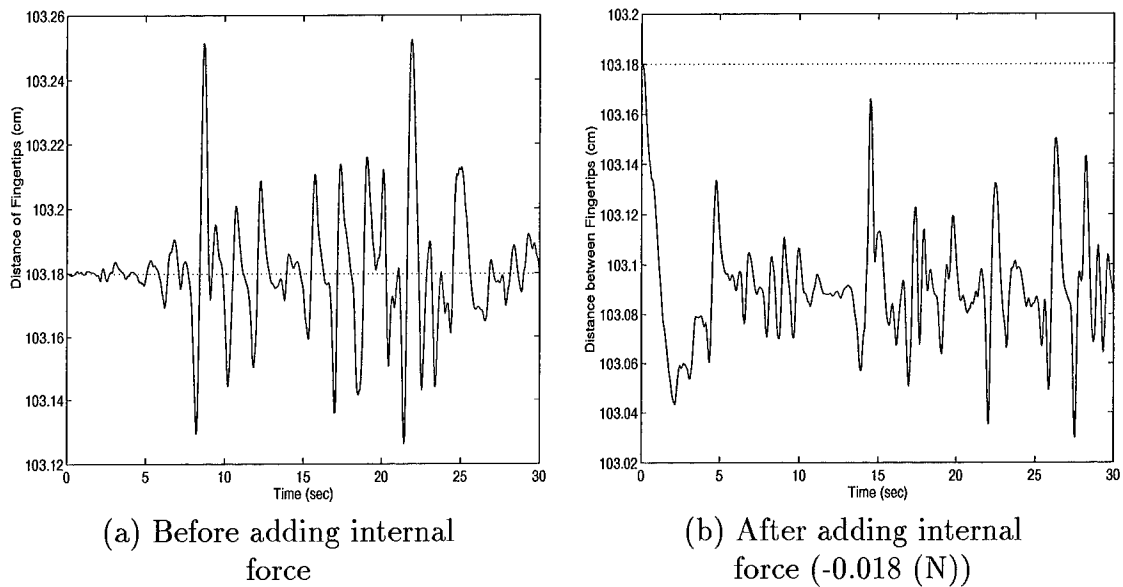


Figure 4.24 The Distance of Two Fingertips under Normally Distributed Random Disturbance Input (Mean Zero and  $3\sigma = 0.25$  (cm))

increase the internal force to make a non-slip grasping. For this experiment, we added  $-0.018$  (N) in the  $y$ -axis of both contact frames shown in Fig (4.18). To get better performance in force control, the stiffness gain is also increased to  $\mathbf{K}_{fp} = 0.0043$ . The distance of two fingertips after adding the additional internal force is shown in Fig (4.24) (b), and the fingertips distance error, measured distance - nominal distance (103.18 (cm)), is shown in Fig (4.25).

Fig (4.26) and (4.28) show the results of both finger performances for the random disturbance input. The position deviations of both fingertips are relatively larger than the case of step disturbance input compared both maximum magnitudes of disturbance inputs. The main cause of the larger position deviation of both fingertips is that we use the normally distributed random disturbance with zero mean and  $3\sigma = 0.25$  (cm) rather than the step disturbance with 0.25 (cm) at  $t = 0$  (sec). Thus, there are some disturbances larger than 0.25 (cm), and the direction of disturbance is continuously changing while the step disturbance is not changed at

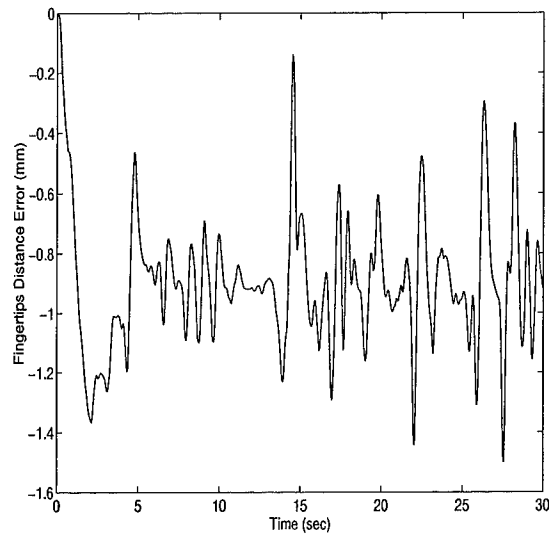
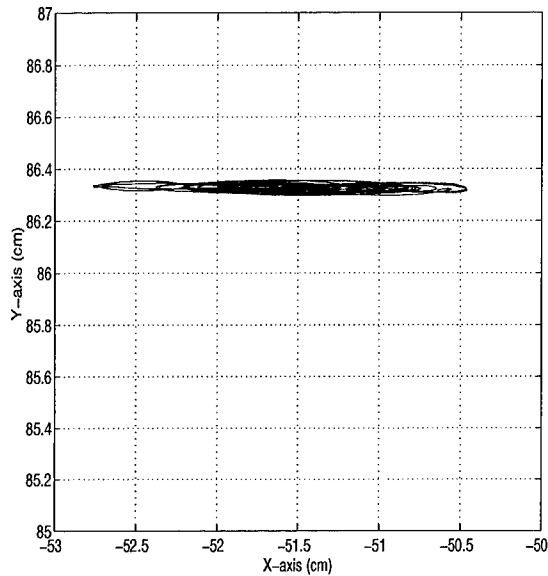


Figure 4.25 Fingertips Distance Error after adding Internal Force (-0.018 (N))

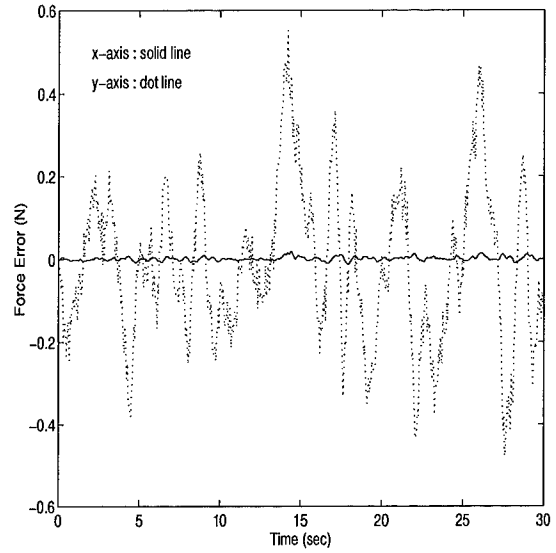
all after  $t = 0$  (sec). The stability is shown in Fig (4.27) and (4.29) for both fingers with larger joint rates. The FAS centering actions are also achieved even though there exists the unexpected disturbance input like the random input.

#### 4.5 Summary

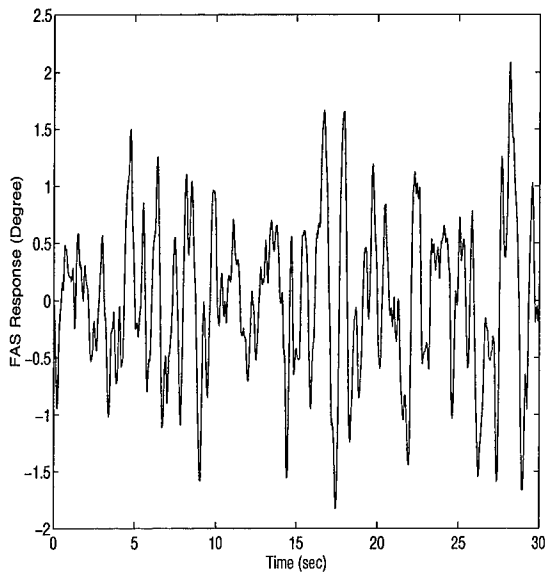
The gross and fine motion control schemes are simulated using the MATLAB SIMULINK environment. The concepts described in Chapter 3 are used in two separate ways. Based on the 4 cases of manipulability, the gross motion control is simulated in the view of teleoperation control. In the fine motion control simulation, the control purposes are imposed to the robot as autonomous control. In particular, a two-finger motion control is simulated to integrate the gross and fine motion control concepts. In next chapter, the implementation of gross motion and integration of both gross and fine motion control concepts are experimented using the AFIT PUMA 560.



(a) Measured contact position



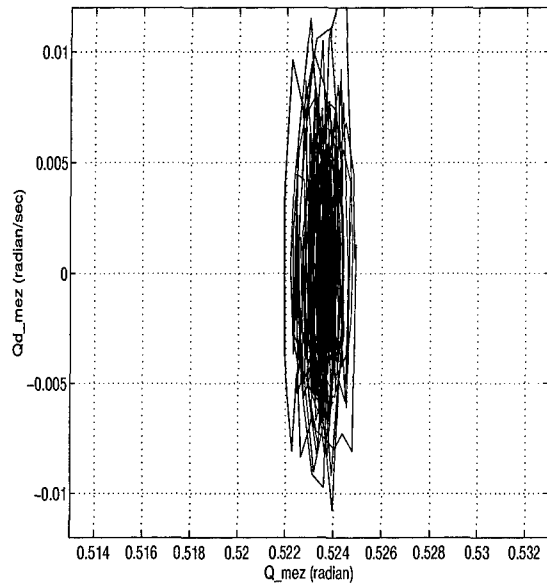
(b) Contact force error



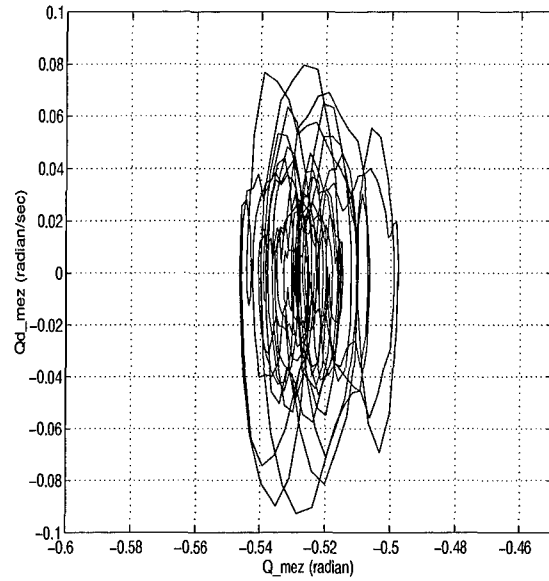
(c) FAS measurement

Figure 4.26 Two-Finger Motion Control - Finger 1 Performance with Normally Distributed Random Disturbance Input (Mean Zero and  $3\sigma = 0.25$  (cm))

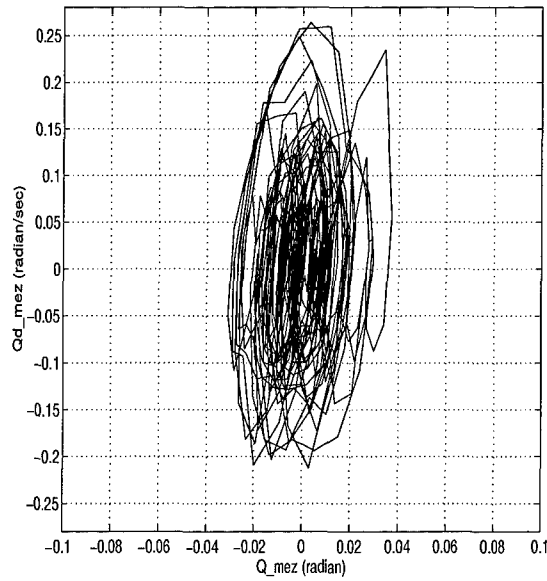




(a) Joint 1

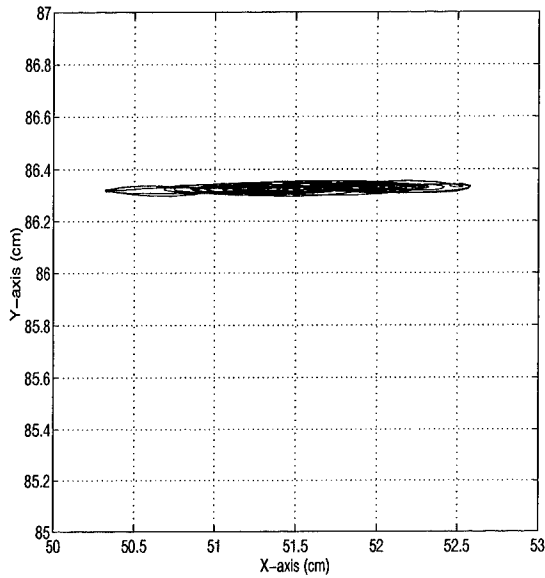


(b) Joint 2

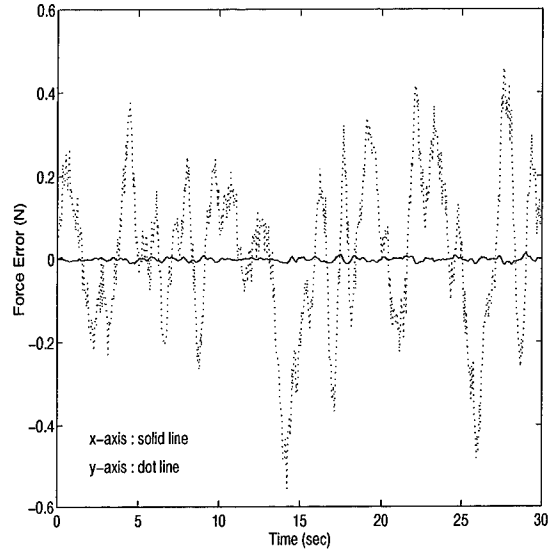


(c) FAS

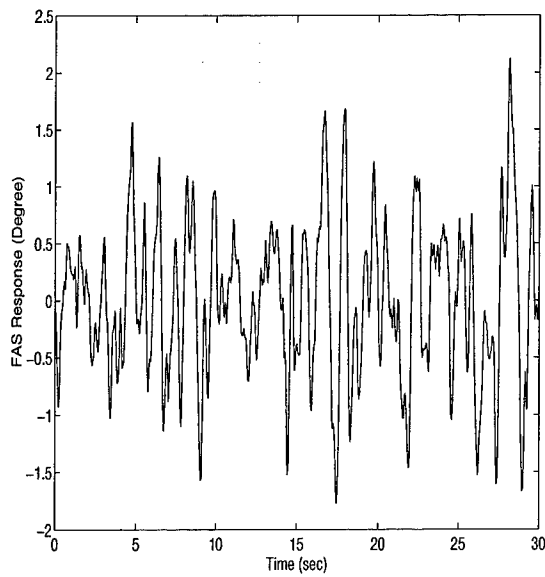
Figure 4.27 Two-Finger Motion Control - Finger 1 Joint Stability with Normally Distributed Random Disturbance Input (Mean Zero and  $3\sigma = 0.25$  (cm))



(a) Measured contact position

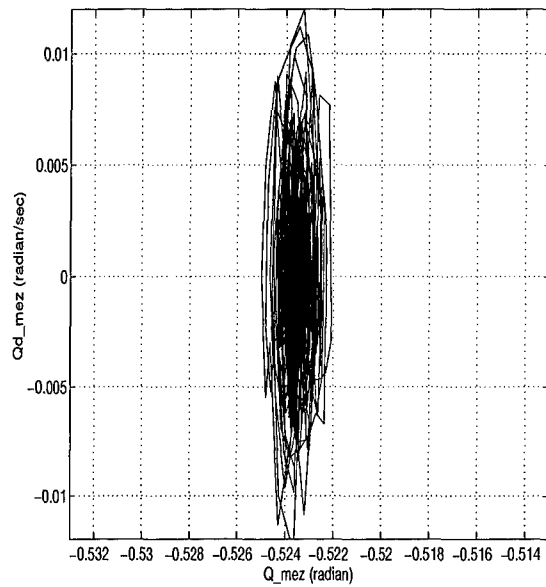


(b) Contact force error

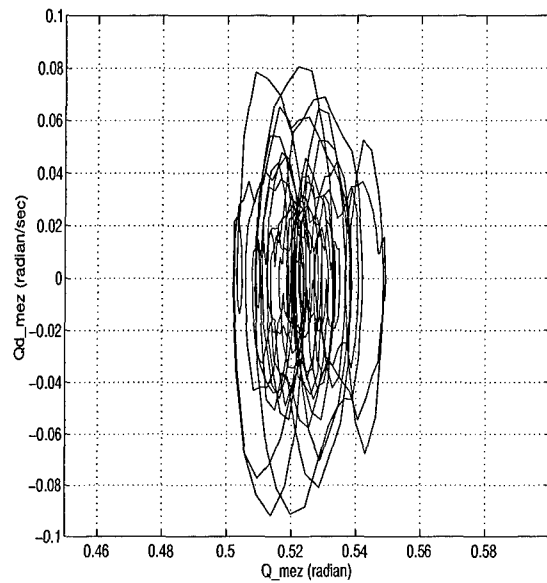


(c) FAS measurement

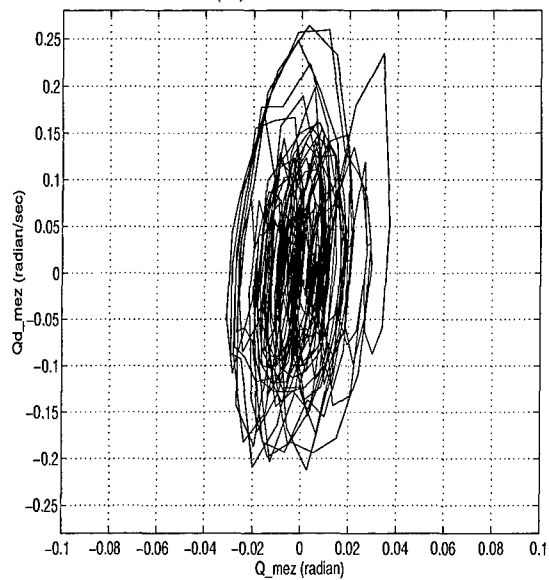
Figure 4.28 Two-Finger Motion Control - Finger 2 Performance with Normally Distributed Random Disturbance Input (Mean Zero and  $3\sigma = 0.25$  (cm))



(a) Joint 1



(b) Joint 2



(c) FAS

Figure 4.29 Two-Finger Motion Control - Finger 2 Joint Stability with Normally Distributed Random Disturbance Input (Mean Zero and  $3\sigma = 0.25$  (cm))

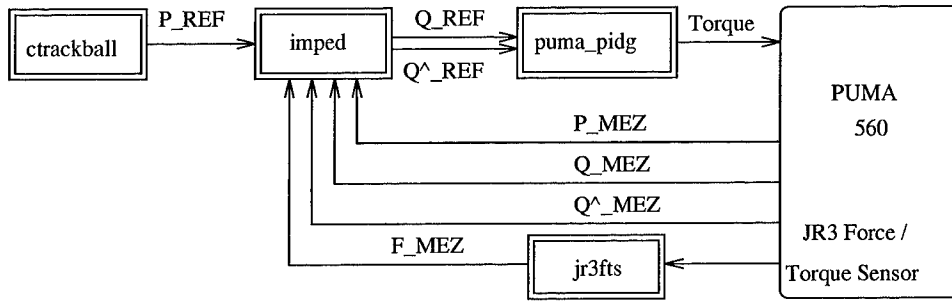
## V. Implementation

### 5.1 Overview of Experiment

This chapter validates the simulation results obtained in the previous chapter. There are two experimental tasks to be performed. The first experiment shows the gross motion control for the robot arm motion with and without payload (SO). The gross motion plans are the same as in Chapter 4. The second experiment demonstrates a special case of fine and gross motion control concepts. This is a partial validation of overall control architecture using the AFIT PUMA 560 Joint 2, 3 and 5 as a one finger motion attached at the end of robot arm. The last joint (Joint 5) is used for the fingertip motion control, and the other two joints are used for fine and gross motion control respectively. By responding to the fingertip actuator, the fine motion control provides the fingertip actuator centering action to provide the desired contact force, and the gross motion control provides the sufficient actuation range for the finger motion. There are three independent controllers operated with different speed in this experiment. The Chimera Real Time Operating System [35] is used to demonstrate both control concepts. The *c*-code used in these experiments is listed in Appendix D.

### 5.2 Gross Motion Results

The gross motion control simulation in Chapter 4 is implemented on a 3 DOF PUMA 560 robot which is equipped with a JR3 Force/Torque Sensor. The modules used in this experiment are *imped.c* (kinematic module) (listed in Appendix D) running at a rate of 100 Hz, *ctrackball.c* (input module) running at a rate of 60 Hz, and *jr3fts.c* (force feedback module) running at a rate of 50 Hz. The joint position and velocity are measured at a rate of 500 Hz by *puma\_pidg.c* module. The module block diagram is shown in Fig (5.1) to describe the inputs and outputs of each module. The motion planning used in Fig (4.3) is applied again in this experiment





- P\_REF : Reference Position
- P\_MEZ : Measured Position
- Q\_REF : Reference Joint Value
- Q\_MEZ : Measured Joint Value
- Q^\_REF : Reference Joint Velocity Value
- Q^\_MEZ : Measured Joint Velocity Value
- F\_MEZ : Measured Force Value
-  : Module Block
-  : Physical Description of Robot and Device

Figure 5.1 Module Block Diagram of Gross Motion Control

to validate the concept using the physical robot. There are 2 cases of experiments based on the 4 cases of manipulabilities.

Fig (5.2) shows the results of free space motion control with the SO (0.45 Kg). Using the concept of impedance control, the controller attempts to regulate the force measurement in  $z$ -axis while keeping the desired position input. Fig (5.2) (b) and (d) show the results of position and force with respect to the base frame. Since the gravity of SO affects the performance of the end-effector mainly in the  $z$ -axis, the deviation of position output in  $z$ -axis is also larger than the response of  $x$ -axis. By the same reason, the result of second task in Fig (5.2) (c), orientation, has an error range about  $1^\circ \sim -2^\circ$  mainly caused by the gravity effect of SO.

Fig (5.3) shows the results of contact with environment. In this experiment, the same object is used. The wall is located around  $x = 60$  (cm) which means that

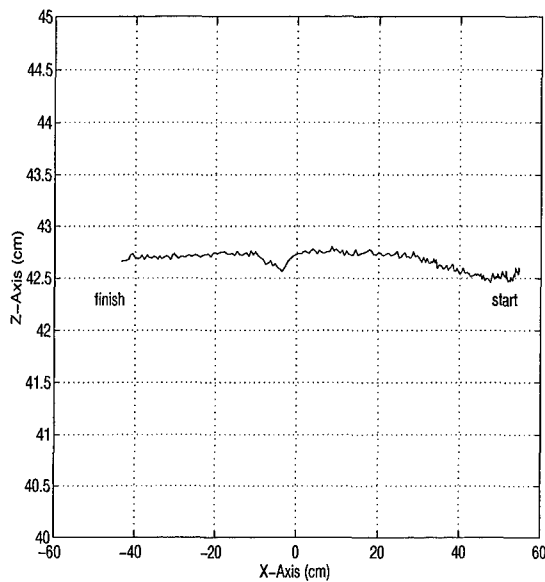
the wall is a little inclined to the  $-x$  direction. As shown in Fig (5.3) (b) and (d), the responses of position and force are correlated to each other. For example, if the error in  $x$ -axis is increased, the measurement of force in  $x$ -axis is also increased and this result affects the position controller by the concept of impedance control. This correlated performance makes contact with the environment while regulating the force in both  $x$  and  $z$ -axis. The orientation of end-effector has the error about  $2^\circ$  mainly caused by the gravity effect of SO.

The results of both experiments are quite acceptable in the view of gross (coarse) motion operated by the teleoperator. From these experiments, we see that the controller does not control position and/or orientation or force exactly. This is a characteristic of an impedance controller. We remind the reader that this is the portion of the system (gross motion controller) under teleoperator control.

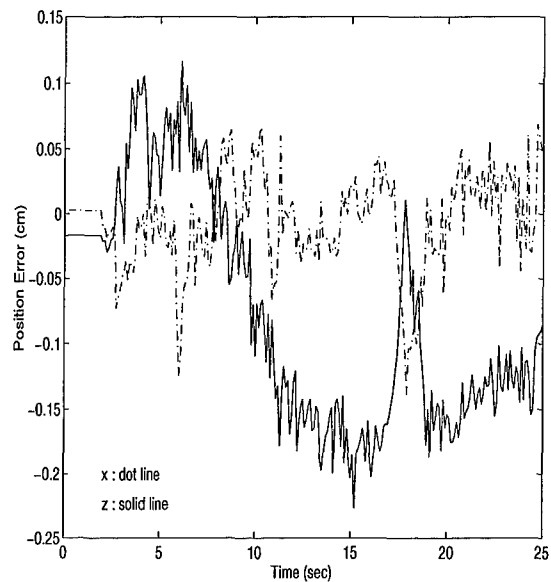
### 5.3 Gross + Fine Motion Control Results and Stability

This experiment is the integration of fine and gross motion control to show a partial validation of grasp stability of a multifingered hand attached to the end of robot arm. As shown in Chapter 4, the higher speed of low level actuation system provides more robust grasp stability. For this experiment, the Joint 2, 3 and 5 of PUMA 560 are selected to make a three-link serial planar revolute joint robot, i.e., the Joint 5 represents the motion of fingertip actuation system, and the Joint 2 and 3 represent the motions of gross and fine motion control system. All three different systems are correlated by the performance of each other. Fig (5.4) shows the motion planning of this experiment.

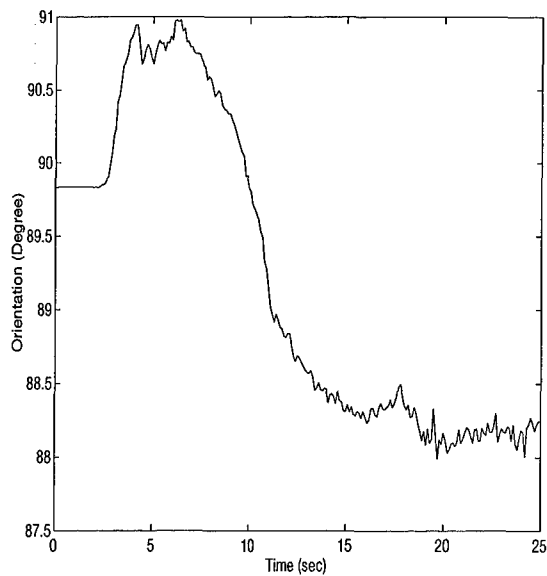
There are three modules experimented with different sampling rates: *force.c* module, *fine.c* module, and *gross.c* module. In Appendix C, the sets of tests are listed to demonstrate the performance of fingertip under different sampling rates. The purpose of these tests is to show that the higher sampling rate in Real Time Operating System corresponds to the higher speed of system response. For this



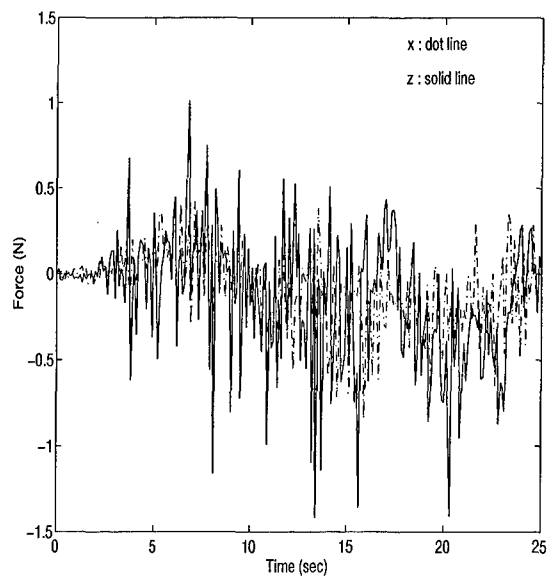
(a) Measured end-effector position



(b) Position error

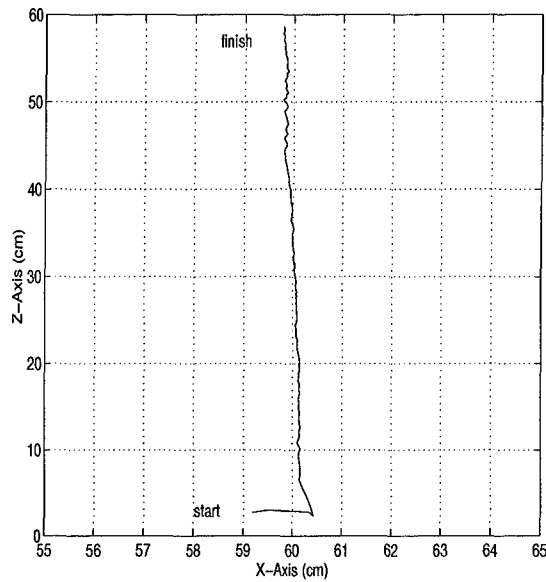


(c) Orientation of end-effector  
( $\Theta_2 + \Theta_3 + \Theta_5 = 90^\circ$ )

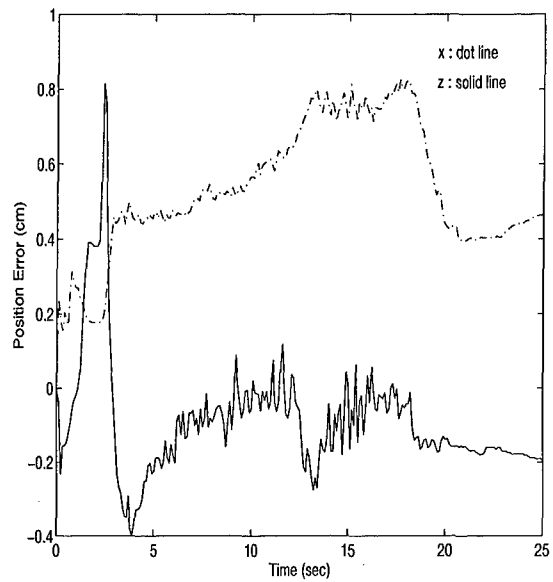


(d) Measured force with respect to Base Frame

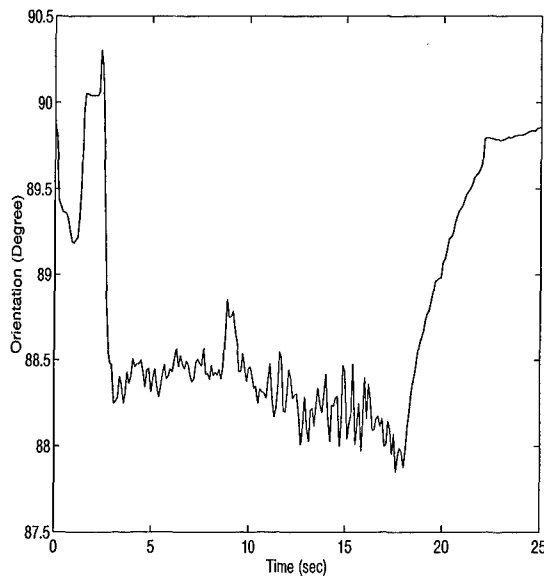
Figure 5.2 Gross Motion Control Demonstration Results for Free Space Motion with Super Object (0.45 Kg)



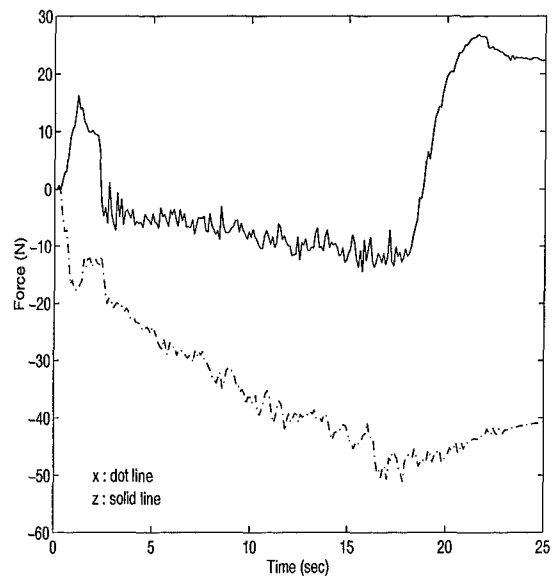
(a) Measured end-effector position



(b) Position error



(c) Orientation of end-effector  
( $\Theta_2 + \Theta_3 + \Theta_5 = 90^\circ$ )



(b) Measured force with respect to Base Frame

Figure 5.3 Gross Motion Control Demonstration Results for Contact with Environment with Super Object (0.45 Kg)



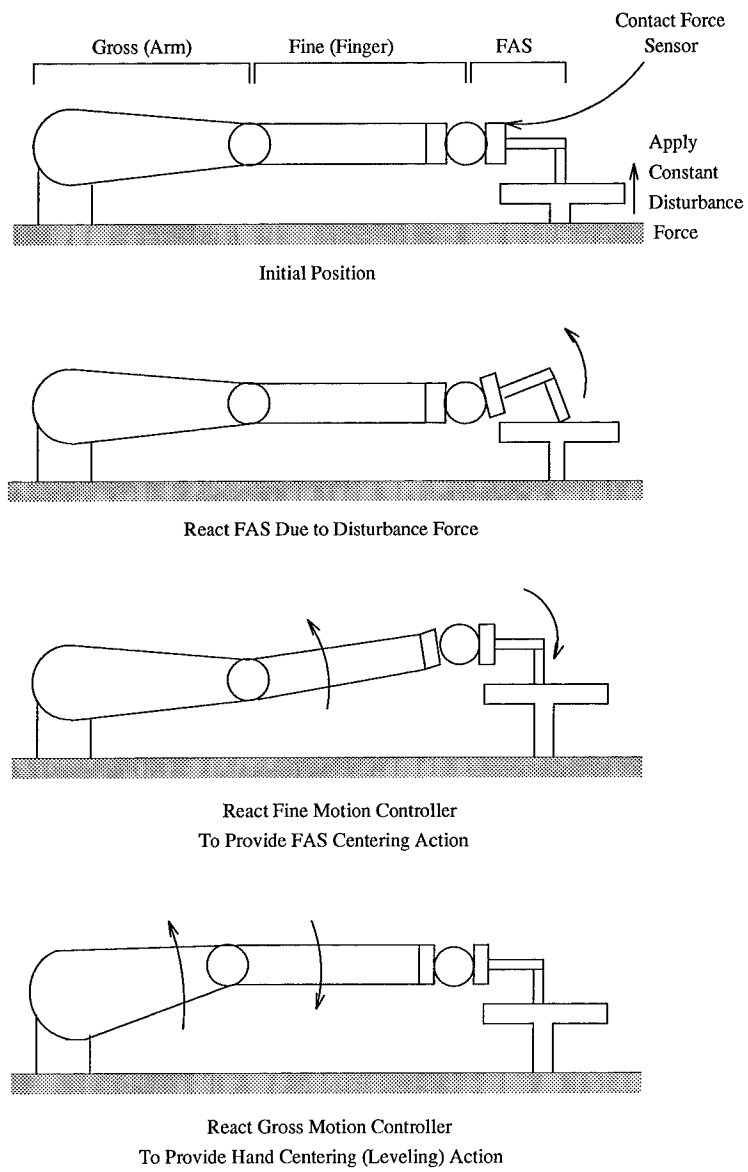


Figure 5.4 Gross + Fine Motion Planning using PUMA 560

integrated control system, it is necessary to provide the lower level control hierarchy with higher bandwidth response so that the lower level control system can respond for the unexpected disturbance or imperfect control performance. The test set #5 in Table (5.1) is chosen for further experiment because it provides the best performance for any cases of experiment.

Module Sampling Rate			Force
<i>gross</i> (Hz)	<i>fine</i> (Hz)	<i>force</i> (Hz)	Error Range (N)
5	50	500	-1.9695 ~ 1.5882

Table 5.1 Module Sampling Rates of Test Set #5 in Appendix C

The control block diagram used is shown in Fig (5.5) which is extended from the fine motion control block diagram developed in Fig (3.7) in Chapter 3. Thus, in Fig (5.5), the Joint 5 (FAS) responds due to the force error and the position difference between the FAS centering location and current FAS location affects the movement of the Joint 3 (Fine). When the Joint 3 operates due to the movement of the Joint 5, the Joint 5 also operates by the FAS centering algorithm. Finally, the Joint 2 which is the slowest control level operates due to the motion of the Joint 3 and provides the Hand leveling action. The force to position gain ( $\mathbf{K}_{fp}$ ) which is the stiffness gain in force control is obtained by experimenting with the materials of fingertip and object. There are two types of object used, i.e., soft surface and metal surface. For this special case of experiment, we assume that the objects are known and not arbitrary.

*5.3.1 Compliance.* The control concept developed combines position and force control which operates in the same direction and causes coupled motions. In general, the high accuracy of position control requires the rigidity of finger. However, it is difficult to accomplish stable operation under force control with such a rigid body. A simple way to overcome this problem is to design the finger to be a spring

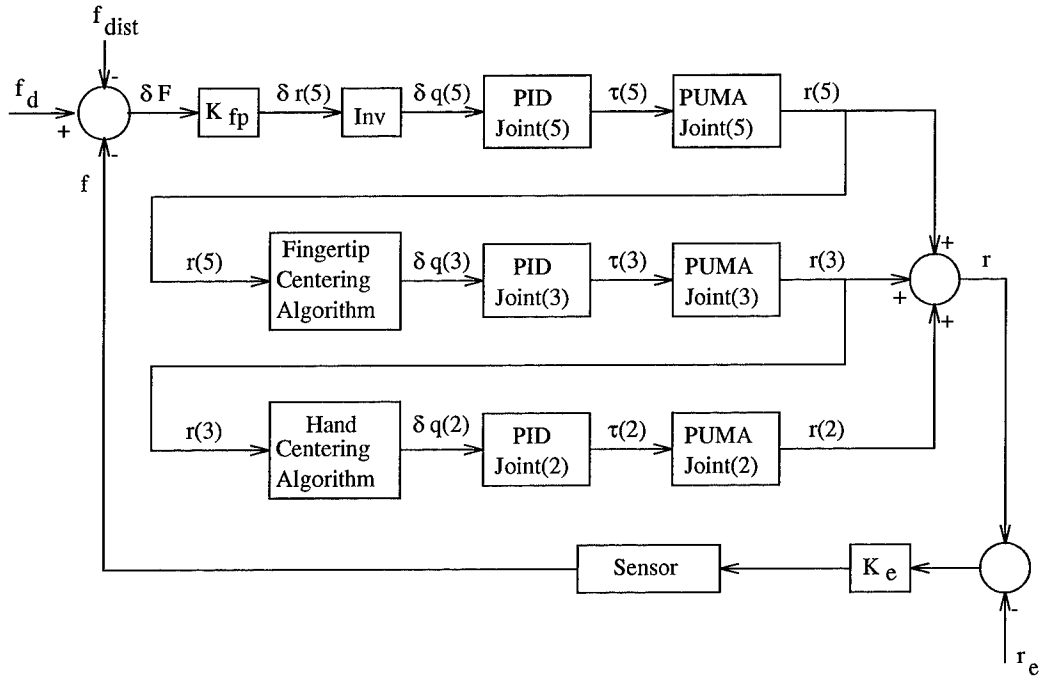


Figure 5.5 Gross + Fine Motion Control Block Diagram

model. Thus,

$$\mathbf{f} = \mathbf{K}_e(\mathbf{r} - \mathbf{r}_e) \quad (5.1)$$

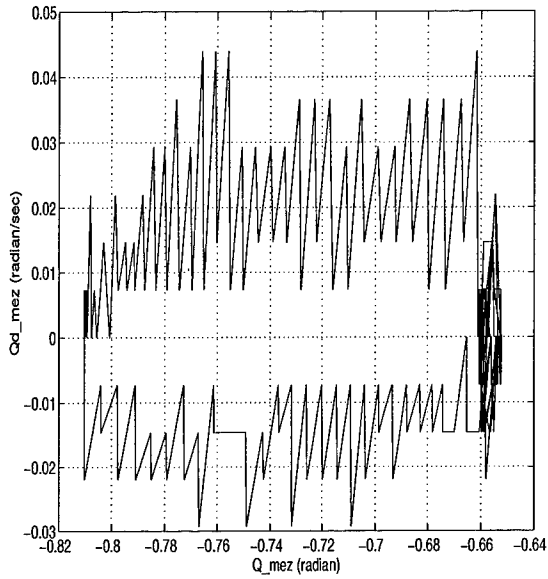
where  $\mathbf{f}$  is the measurement of contact force,  $\mathbf{K}_e$  is the environmental stiffness gain and  $\mathbf{r}$  and  $\mathbf{r}_e$  are the measured fingertip position and the location of environment respectively. In this experiment, the environmental stiffness gain,  $\mathbf{K}_e$ , is not known. Therefore, as shown in Fig (5.5), the force to position conversion gain,  $\mathbf{K}_{fp}$ , is used as a compliance gain. The initial selection of  $\mathbf{K}_{fp}$  is performed heuristically. However, this gain should be changed by the characteristics of object surfaces, i.e.,  $\mathbf{K}_{fp}$  is increased when the object surface is more compliant, and is decreased when the object surface is more stiff. In this experiment,  $\mathbf{K}_{fp}$  is chosen as

$$\begin{aligned} \mathbf{K}_{fp} &= 0.001 \text{ for metal case} \\ \mathbf{K}_{fp} &= 0.003 \text{ for soft surface case} \end{aligned} \quad (5.2)$$

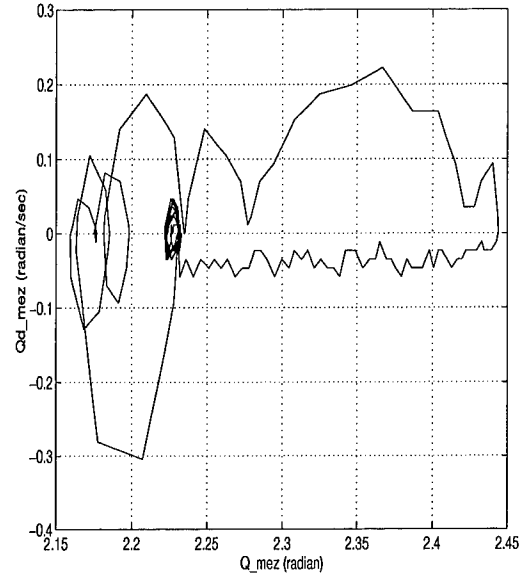
Fig (5.6) (d), (5.7) (d) and (5.9) show the results of force controls using combinations between  $\mathbf{K}_{fp}$  and object surfaces. As stated previously, we can obtain better performances for both object cases by changing the stiffness gain shown in Fig (5.6) (d) and (5.9) (d). However, if we apply the inappropriate gain to the controller, worse performance of force control occurred as shown in Fig (5.7) (d), or even the controller failed to make the contact (in this case, the force error was not plotted as shown in Fig (5.8)). Furthermore, the initial selection of  $\mathbf{K}_{fp}$  without known the object can be made by the operator's general knowledge for the object before operating the controller. One interesting result from these experiments was that the finger did not stay at one position sometimes (it had hiccup at  $t = 23$  (sec) shown in Fig (5.6)) even though there were no disturbance inputs. This happened because we only controlled the stiffness.

*5.3.2 Stability.* The same method [36] is used to validate the system stability. Using the same way done in Chapter 4, the phase variables such as joint position and velocity of every joints are plotted in the phase plane based on the combinations of  $\mathbf{K}_{fp}$  and the object surfaces. As shown in Fig (5.4), the fingertip at the equilibrium point is perturbed by the disturbance force occurred on the object, and it will return asymptotically to the equilibrium point. Fig (5.6) (a), (b) and (c) through (5.9) (a), (b) and (c) show these results. In addition to these results, the FAS centering actions are also shown in Fig (5.6) (c) through (5.9) (c) that respond in the center of its joint range even though there is an unexpected disturbance applied to the object.

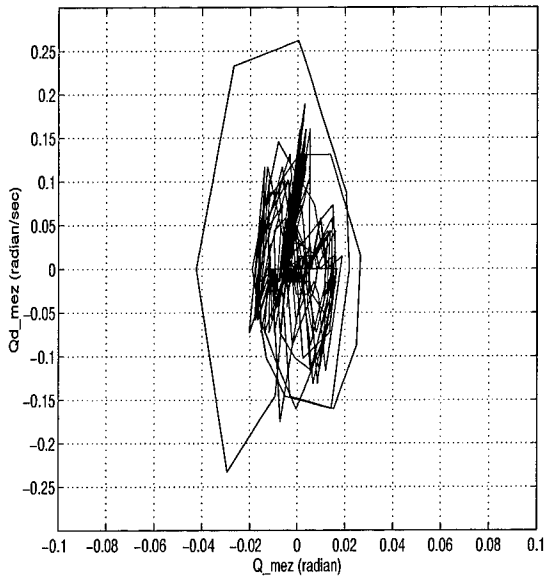
*5.3.3 Instability.* One of requirements of FAS is that it must be faster than the finger actuators so that it can achieve the stable operation under disturbance inputs. The previous section shows only positive results of this assumption. As a contradiction, this section discusses the unstable results of fine motion control resulting from improper selection of FAS and finger actuator speeds, i.e., the speed



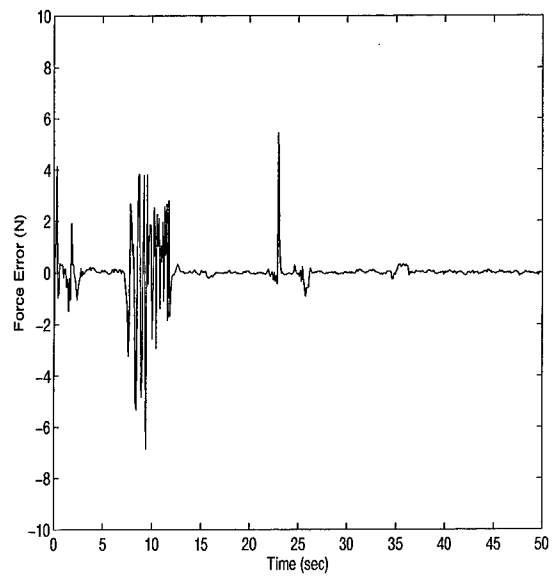
(a) Joint 2



(b) Joint 3

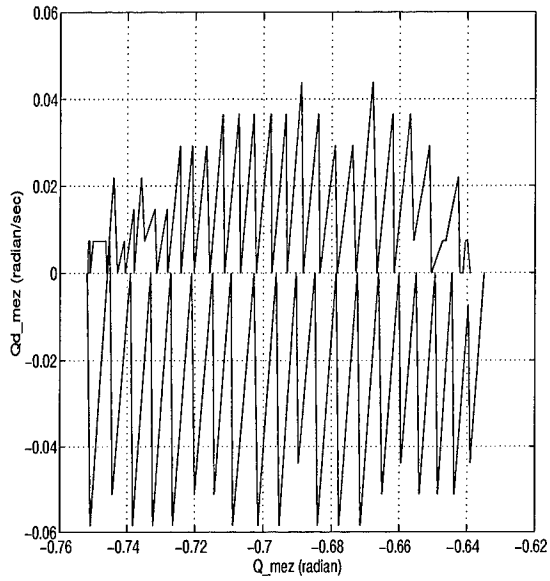


(c) Joint 5

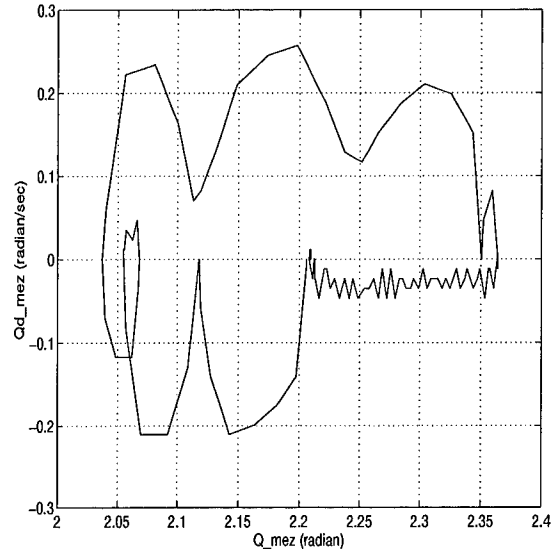


(d) Force Error

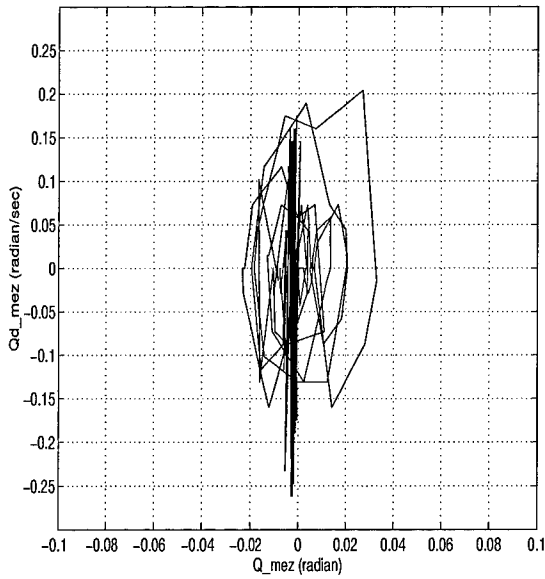
Figure 5.6 Speed : Force - 500 Hz, Fine - 50 Hz, Gross - 5 Hz, Gain : 0.001, Metal case



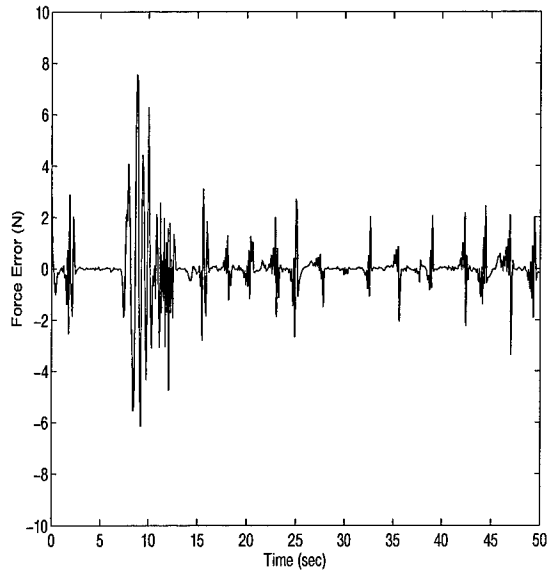
(a) Joint 2



(b) Joint 3

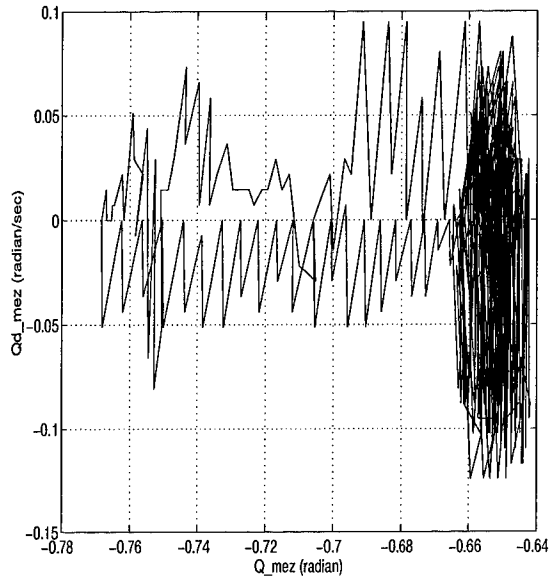


(c) Joint 5

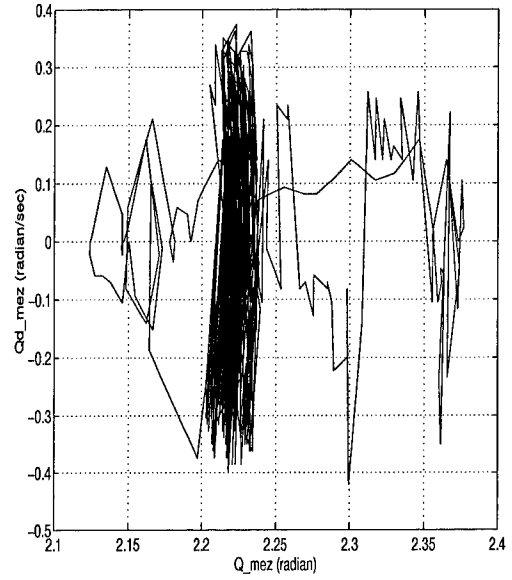


(d) Force error

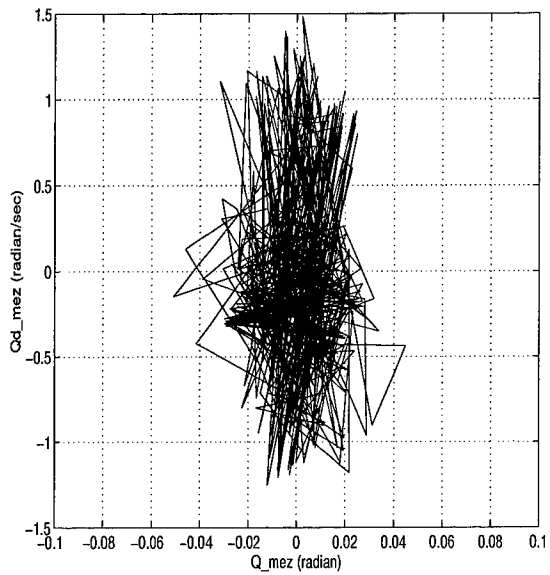
Figure 5.7 Speed : Force - 500 Hz, Fine - 50 Hz, Gross - 5 Hz, Gain : 0.001, Soft chair case



(a) Joint 2

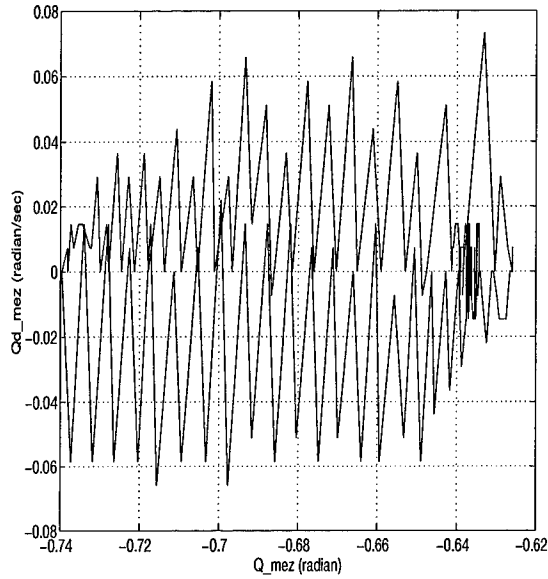


(b) Joint 3

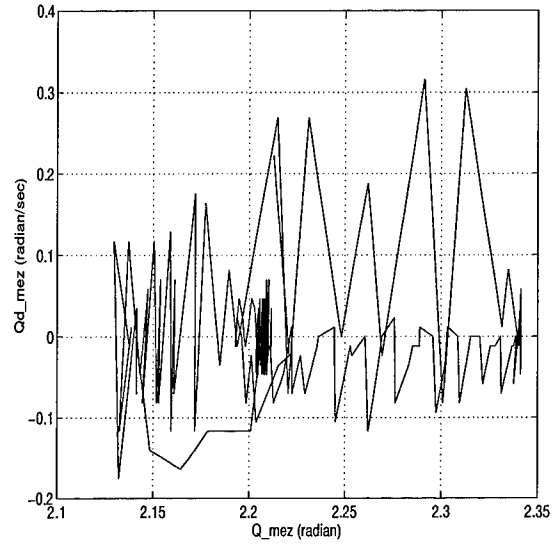


(c) Joint 5

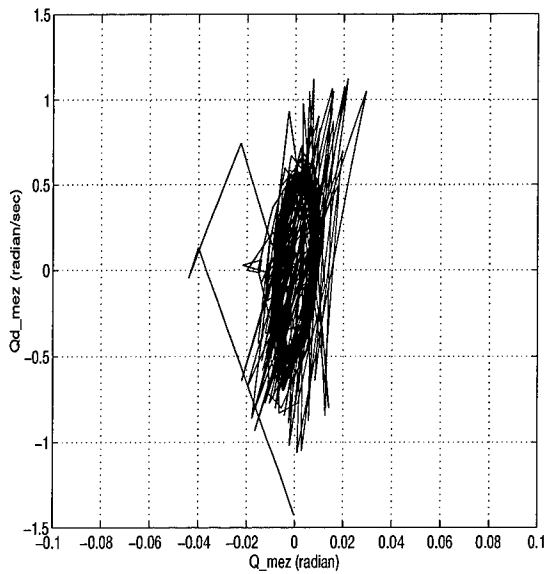
Figure 5.8 Speed : Force - 500 Hz, Fine - 50 Hz, Gross - 5 Hz, Gain : 0.003, Metal case



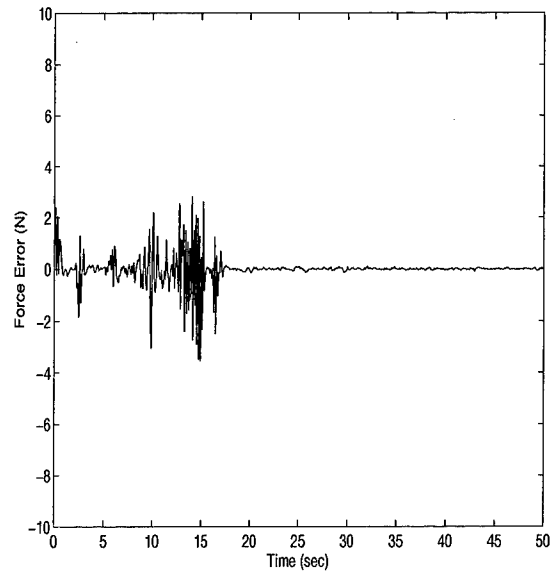
(a) Joint 2



(b) Joint 3



(c) Joint 5



(d) Force error

Figure 5.9 Speed : Force - 500 Hz, Fine - 50 Hz, Gross - 5 Hz, Gain : 0.003, Soft chair case



of *gross* module is faster than the speed of *fine* module and the speed of *fine* module is faster than the speed of *force* module. The sets of tests are shown in Appendix C. The results can be interpreted that the fine motion control level actuators should be a higher speed than the gross motion control actuators. In other words, the FAS should operate faster than the fine motion controller which operates faster than the gross motion controller. This implies that the order of controller speed must be used to achieve the stable operation.

#### 5.4 Summary

The concepts of gross and fine motion control are implemented using the AFIT PUMA 560. The capability of impedance control used in the gross motion control is proved under the contact with the physical environment. The integration of gross and fine motion control is demonstrated for the special cases of control purpose. The concept of FAS centering algorithm is well achieved and provides the robustness of object grasping problem under unexpected disturbance inputs.

## *VI. Conclusion and Recommendation of Future Work*

### *6.1 Research Conclusion*

This thesis has presented a general control architecture of teleoperation control including object manipulation in unstructured applications. The main problems in unstructured applications are to determine the force components between the robot and environment and to provide grasp stability of the object manipulation for an operator's motion command. The hierarchical control structure was suggested as a way to solve the given problems. The teleoperation systems involving object handling requires the operator's attention in most cases. The method developed in this thesis attempts to reduce the operator's conceptual constraints and provides an intuitive way to control the robot in terms of Object Based Control (OBC). There were two main control levels based on the control hierarchy. The gross motion control which includes the robot arm and Super Object (SO) is to achieve the position and orientation of SO. Since this is a teleoperation system, the concept of impedance control was used for the gross motion control to control the contact with environment. In experiments of both simulation and demonstration, the gross motion control shows satisfactory results for the 4 cases of manipulability. In the contact with environment, the dynamic relationship between the robot and environment was well controlled by the impedance controller.

The fine motion control involves the finger motions and object grasp stability in the Super Object (SO) of gross motion control. The connection between the gross and fine motion control is the forces provided by the gross motion wrist force / torque sensor to determine the fingertip contact forces. To accomplish successful object manipulation, stable grasping is the main issue to be addressed. Providing the optimal fingertip contact force [24] to each finger is the main way to ensure grasp stability. However, for unexpected disturbance inputs occurring on the object, the finger controllers in position control must react appropriately to avoid dropping or

breaking the object. The Fingertip Actuation System (FAS) has higher bandwidth response than does the finger motion system and operates in force control to maintain the desired optimal contact force of each fingertip. The Fingertip Actuation System (FAS) centering algorithm using the finger and FAS was developed to make the grasp stability robust. The FAS centering algorithm causes the FAS to be near the center of its displacement range by using the finger in position control. The command to the position control is the FAS displacement from its centered location. The FAS centering action was designed to be able to reduce the error in the finger contact positions when the object model is uncertain. Contrary to the gross motion control, the fine motion control should be autonomous so that the workload requirements of the operator are reduced.

The simulation experiment in fine motion control used the FAS centering algorithm for a two-finger object grasp. The main goal of this level is to show the successful grasp stability for two types of disturbances (step and normally distributed random inputs) applied to the grasped object. In this experiment, the FAS responds in the direction of disturbance input on the object and the finger attempts to reduce the offset between the FAS centered and current positions by moving the finger links. This provides adequate FAS centering action and accomplishes the object grasp stability. The implementation step is an integration of gross and fine motion control to partially validate the developed control architecture using the 3 Degree Of Freedom (DOF) planar robot that each DOF represents a different robot with its own controller (FAS, Fine, or Gross) and each robot depends on the performances of other robots (controllers). This special case of experiment develops the force and position controlled actuator centering / leveling algorithm as a general case of serial actuation system and can be applied to the separately controlled robot applications.

## 6.2 Recommendation of Future Work

The control architecture developed in this thesis is only applied to the case of planar robot for the gross and fine motion control. In general, robots require the control architecture to be able to operate the spatial cases. Thus, it is necessary to extend the control architecture developed in this thesis to the spatial case.

In fine motion control, the grasp stability can be achieved by applying as much the internal force as the object can withstand. However, for unexpected disturbance forces occurring on the object or imperfect grasp due to the object uncertainty, finger force overload must be prevented. The concept of explicit force control [30] was used to track the desired fingertip contact force. As mentioned in Section (5.3.1), the problem was that the performances of fingertips in the real applications do not stay at one position (they had hiccups) even though there are no disturbance inputs. This problem was solved by applying the order of controller speeds (force > fine > gross) in this thesis. However, this is not a feasible solution in general cases. Another problem was that the performance of fine motion controller using the explicit force control did not provide the minimum force requirement to grasp an object. If there is an instantaneously large amount of disturbance input applied to the object, the deviation of FAS can be increased and the joint rates of the fingers can also be increased. This instantaneous large velocity can cause the object to drop or break due to the over-response of the fine motion controller. To reduce this effect, we need to add the damping control mode in the finger position controller so that the abrupt joint rates of the fingers can be controlled. It is recommended that impedance control is used for finger control to include the damping mode.

One concern of this thesis is to show the contact stability of the fingertip. The proof of contact stability was only shown by using the results of experiments rather than proving through mathematical procedures. Therefore, the proof of contact stability using mathematical procedures is also recommended as further research.

### 6.3 Summary

1. The summary of research conclusion is listed below:
  - Teleoperation control including object handling system
    - Force components need to be determined
    - Grasp stability should be achieved for an operator's motion command
    - How to reduce the operator's control attention
    - Hierarchical control structure is suggested way
  - Gross Motion Control
    - Robot arm and SO - achieve the position and orientation of SO
    - Impedance control - address environment uncertainties
    - Teleoperation control
  - Fine motion control / grasp stability
    - Serial force and position control
    - Force is controlled in the direction of motion
    - FAS centering algorithm - provide the sufficient joint range to the FAS
    - Experimented for different environments
    - Well achieved grasp stability
  - Implementation - integrated gross and fine motion control
    - PUMA 560 planar configuration
    - FAS centering and hand leveling motion
    - Partial validation of developed control architecture
2. The summary of recommendations is listed below:
  - Extend to spatial applications
  - Need to add damping mode for force control - impedance control
    - Control the finger to response the abrupt disturbance input
    - Need to damp the unforeseen disturbance input
  - Prove and analysis architecture stability through mathematical method

## Bibliography

1. *MATLAB User's Guide (High-Performance Numeric Computation and Visualization Software)*, The Math Work Inc., 1992.
2. *MATLAB SIMULINK (Dynamic System Simulation Software)*, The Math Work Inc., Mar. 1992.
3. Murray, R. M., Li, Z. and Sastry, S. S., *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994.
4. Hsu, Ping, "Coordinated Control of Multiple Manipulator System," *IEEE Transactions on Robotics and Automation*, Vol. 9, No. 4, Aug 1993.
5. Nakamura, Y., *Advanced Robotics Redundancy and Optimization*, Addison-Wesley Publishing Company, 1991.
6. Arimoto, S., Miyazaki, F. and Kawamura, S., "Cooperative Motion Control of Multiple Robot Arms or Fingers," *IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 1407-1412, 1987.
7. Hayati, S., "Hybrid Position/Force Control Of Multi-Arm Cooperative Robots," *IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 82-89, 1986.
8. Mason, M. T., "Compliance and Force Control for Computer Controlled Manipulators," *IEEE Transactions on System, Man, and Cybernetics*, Vol. SMC-11, pp. 418-432, 1981.
9. Craig, John J., *Introduction to Robotics: Mechanics and Control*, Addison-Wesley Publishing Company, Inc. 1986.
10. Chiaverini, Stefano and Sciavcco, Lorenzo, "The Parallel Approach to Force / Position Control of Robotic Manipulators," *IEEE Transactions on Robotics and Automation*, Vol. 9, No. 4, Aug 1993.
11. Yun, X., "Nonlinear Feedback Control of Two Manipulators in Presence of Environmental Constraints," *IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1252-1257, 1989.
12. Koga, M., Kosuge, K., Furuta, K. and Nosaki, K., "Coordinated Motion Control of Robot Arms Based on the Virtual Internal Model," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, pp. 77-85, Feb. 1992.
13. Tarn, T. J., Bejczy, A. K. and Yun, X., "Design of Dynamic Control of Two Cooperating Robot Arms: Closed Chain Formulation," *IEEE International Conference on Robotics and Automation*, pp. 7-13, 1987.
14. Soloway, D. I. and Alberts, T. E., "Force Control of A Multi-Arm Robot System," *IEEE International Conference on Robotics and Automation*, pp. 1490-1496, 1988.

15. Cutkosky, Mark R., *Robotic Grasping and Fine Manipulation*, Boston: Kluwer Academic Publishers, 1985.
16. Schneider, S. A. and Cannon, R. H., "Object Impedance Control for Cooperative Manipulation: Theory and Experimental Results," *IEEE International Conference on Robotics and Automation*, pp. 1076-1083, 1989.
17. Asada, H. and Slotine, J. J. E., *Robot Analysis and Control*, John Wiley and Sons, 1986.
18. Hogan, N., "Impedance Control: An approach to manipulation, Part I," *Transaction of ASME, Journal of Dynamic System, Measurement, and Control*, Vol. 107, Mar. 1985.
19. Hogan, N., "Impedance Control: An approach to manipulation, Part II," *Transaction of ASME, Journal of Dynamic System, Measurement, and Control*, Vol. 107, Mar. 1985.
20. Hogan, N., "Impedance Control: An approach to manipulation, Part III," *Transaction of ASME, Journal of Dynamic System, Measurement, and Control*, Vol. 107, Mar. 198.
21. Tarn, T. J., Xi, N., Guo, C. and Berjczy, A. K., "Function-Based Control Sharing for Robotic Systems," *IEEE International Conference on Intelligent Robotics*, pp. 1-6, 1995.
22. Spong, M. W. and Vidyasagar, M., *Robot Dynamics and Control*, John Wiley and Sons, 1989.
23. Hunter, M. W., Spenny, C. H., *Contact Force Assignment Using Fuzzy Logic*, Air Force Institute of Technology, 1995.
24. Hunter, M. W., *Contact Force Assignment Using Fuzzy Logic Based Reactions*, PhD Prospectus, Air Force Institute of Technology, 1996.
25. Fu, K. S., Gonzalez, R. C. and Lee, C. S. G., *Robotics (Controls, Sensing, Vision, and Intelligence)*, McGraw-Hill Book Company, 1987.
26. Nagai, K. and Yoshikawa, T., "Manipulating and Grasping Forces in Manipulation by Multifingered Robot Hands," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 1, pp. 67-77, Feb. 1991.
27. Nagai, Kiyoshi and Yoshikawa, Tsuneo, "Dynamic Manipulation / Grasping Control of Multifingered Robot Hands," *Proceedings of the IEEE Conference on Robotics and Automation*. pp. 624-629 1993.
28. Snyder, W. E., *Introduction Robots(Computer Interfacing and Control)*, Prentice-Hall Inc., 1985.
29. Stangaard, A. C. Jr., *Robotics and AI(An Introduction to Applied Machine Intelligence)*, Prentice-Hall Inc., 1987.

30. Whitney, D. E., "Historical Perspective and State of the Art in Robot Force Control," *IEEE Proceedings of the International Conference of Robotics and Automation*, pp. 262-268, 1986.
31. Lew, Jae Y. and Trudnowski, Dan J., "Vibration Control of a Micro / Macro-Manipulator System," *IEEE Control Systems*, pp. 26-31, Feb. 1996.
32. Stevens, H. D. and How, Jonathan, "The Limitations of Independent Controller Design for a Multiple-link Flexible Macro-manipulator Carrying a Rigid Mini-manipulator," *Robotics for Challenging Environments, Proceedings of RCE II, the Second Conference*, pp. 93-99, June, 1996.
33. Schneider, D. L., *EENG 540 Class Notes*, 1994.
34. Nethery, John, *Robotica : User's guide and reference manual*, University of Illinois, September 1993.
35. Stewart, David B. and Khosla, Pradeep K., *Chimera 3.1, The Real-Time Operating System for Reconfigurable Sensor-Based Control Systems*, Advanced Manipulators Laboratory, The Robotics Institute and Department of Electrical and Computer Engineering, Carnegie Mellon University, 1993.
36. D'Azzo, J. J. and Houpis, C. H., *Linear Control System Analysis and Design, Conventional and Modern*, 4th Edition, McGraw-Hill, Inc., 1995.



*Appendix A. Forward Kinematics Development: 3 DOF Planar Case  
of PUMA 560*

Fig (4.1) shows the overall structure of 6 DOF PUMA 560 and the following procedures generate the appropriate 3 DOF planar robot derived from PUMA 560. The purpose is to provide the consistent equations to fit a special case of PUMA robot. First, the overall forward kinematics were computed using the ROBOTICA mathematic software [34], then some of joints are set to be zero. The result forms the planar equations of motion of the manipulator. The link coordinate systems is shown in Fig (A.1) and the D-H parameters of 6 DOF PUMA 560 are listed in Table A.1.

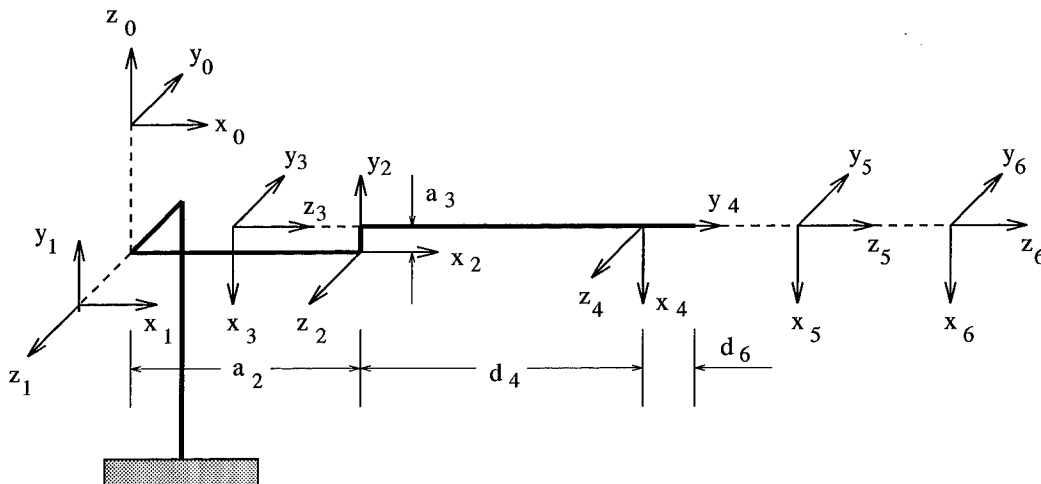


Figure A.1 Link coordinate systems for a PUMA 560

The homogeneous matrix,  $\mathbf{T}_0^i$ , is to specify the  $i$ -th link coordinate frame with respect to the base coordinate frame and it can be expressed by the following

Joint <i>i</i>	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	0	0	$90^\circ$
2	$\theta_2$	0	$a_2$	0
3	$\theta_3 - 90^\circ$	0	$-a_3$	$-90^\circ$
4	$\theta_4$	$d_4$	0	$90^\circ$
5	$\theta_5$	0	0	$-90^\circ$
6	$\theta_6$	$d_6$	0	0

Table A.1 PUMA 560 D-H parameters

relationship.

$$\begin{aligned}
 \mathbf{T}_0^i &= \mathbf{A}_0^1 \mathbf{A}_1^2 \cdots \mathbf{A}_{i-1}^i \quad \text{for } i = 1, 2, \dots, n \\
 &= \begin{bmatrix} \mathbf{n}_i & \mathbf{s}_i & \mathbf{a}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A.1})
 \end{aligned}$$

where  $\mathbf{A}_{i-1}^i$  is the  $i$ -th link coordinate transformation matrix with respect to  $i - 1$  link,  $[\mathbf{n}_i, \mathbf{s}_i, \mathbf{a}_i] \in R^{3 \times 3}$  is the orientation matrix of  $i$ -th coordinate frame, and  $\mathbf{p}_i \in R^3$  is the position vector expressing the origine of  $i$ -th coordinate frame with respect to the origin of base coordinate frame.

Using the ROBOTICA mathematic software, the  $i$ -th link coordinate transformation is obtained. The results are:

$$\begin{aligned}
 \mathbf{A}[1] = & \begin{array}{cccc|c}
 & | & \text{Cos}[q1] & 0 & & \text{Sin}[q1] & 0 & & | \\
 & | & \text{Sin}[q1] & 0 & & -\text{Cos}[q1] & 0 & & | \\
 & | & 0 & 1 & & 0 & 0 & & | \\
 & | & 0 & 0 & & 0 & 1 & & |
 \end{array}
 \end{aligned}$$

$$\begin{aligned}
 \mathbf{A}[2] = & \begin{array}{cccc|c}
 & | & \text{Cos}[q2] & -\text{Sin}[q2] & 0 & & a2 \text{ Cos}[q2] & & | \\
 & | & \text{Sin}[q2] & \text{Cos}[q2] & 0 & & a2 \text{ Sin}[q2] & & | \\
 & | & 0 & 0 & 1 & & 0 & & | \\
 & | & 0 & 0 & 0 & & 1 & & |
 \end{array}
 \end{aligned}$$

$$\begin{aligned}
 \mathbf{A}[3] = & \begin{array}{cccc|c}
 & | & \text{Sin}[q3] & 0 & & \text{Cos}[q3] & a3 \text{ Sin}[q3] & & | \\
 & | & -\text{Cos}[q3] & 0 & & \text{Sin}[q3] & -(a3 \text{ Cos}[q3]) & & | \\
 & | & 0 & -1 & & 0 & 0 & & | \\
 & | & 0 & 0 & & 0 & 1 & & |
 \end{array}
 \end{aligned}$$

$$A[4] = \begin{vmatrix} \cos[q_4] & 0 & \sin[q_4] & 0 \\ \sin[q_4] & 0 & -\cos[q_4] & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$A[5] = \begin{vmatrix} \cos[q_5] & 0 & -\sin[q_5] & 0 \\ \sin[q_5] & 0 & \cos[q_5] & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$A[6] = \begin{vmatrix} \cos[q_6] & -\sin[q_6] & 0 & 0 \\ \sin[q_6] & \cos[q_6] & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$T[1,1] = \cos[q_1] \cos[q_4] \cos[q_5] \cos[q_6] \sin[q_2 + q_3] - \cos[q_5] \cos[q_6] \sin[q_1] \sin[q_4] + \cos[q_1] \cos[q_2 + q_3] \cos[q_6] \sin[q_5] - \cos[q_4] \sin[q_1] \sin[q_6] - \cos[q_1] \sin[q_2 + q_3] \sin[q_4] \sin[q_6]$$

$$T[1,2] = \cos[q_1] \cos[q_4] \cos[q_5] \sin[q_2 + q_3] \sin[q_6] - \cos[q_4] \cos[q_6] \sin[q_1] - \cos[q_1] \cos[q_6] \sin[q_2 + q_3] \sin[q_4] - \cos[q_5] \sin[q_1] \sin[q_4] \sin[q_6] - \cos[q_1] \cos[q_2 - q_3] \sin[q_5] \sin[q_6]$$

$$T[1,3] = \cos[q_1] \cos[q_2 + q_3] \cos[q_5] - \cos[q_1] \cos[q_4] \sin[q_2 + q_3] \sin[q_5] + \sin[q_1] \sin[q_4] \sin[q_5]$$

$$T[1,4] = a_2 \cos[q_1] \cos[q_2] + a_3 \cos[q_1] \sin[q_2 + q_3] + d_4 \cos[q_1] \cos[q_2 + q_3] + d_6 \cos[q_1] \cos[q_2 + q_3] \cos[q_5] - d_6 \cos[q_1] \cos[q_4] \sin[q_2 + q_3] \sin[q_5] + d_6 \sin[q_1] \sin[q_4] \sin[q_5]$$

$$T[2,1] = \cos[q_4] \cos[q_5] \cos[q_6] \sin[q_1] \sin[q_2 + q_3] +$$

$$\begin{aligned} & \text{Cos}[q1] \text{Cos}[q5] \text{Cos}[q6] \text{Sin}[q4] + \\ & \text{Cos}[q2 + q3] \text{Cos}[q6] \text{Sin}[q1] \text{Sin}[q5] + \\ & \text{Cos}[q1] \text{Cos}[q4] \text{Sin}[q6] - \\ & \text{Sin}[q1] \text{Sin}[q2 + q3] \text{Sin}[q4] \text{Sin}[q6] \end{aligned}$$

$$\begin{aligned} T[2,2] = & \text{Cos}[q1] \text{Cos}[q4] \text{Cos}[q6] - \\ & \text{Cos}[q6] \text{Sin}[q1] \text{Sin}[q2 + q3] \text{Sin}[q4] - \\ & \text{Cos}[q4] \text{Cos}[q5] \text{Sin}[q1] \text{Sin}[q2 + q3] \text{Sin}[q6] - \\ & \text{Cos}[q1] \text{Cos}[q5] \text{Sin}[q4] \text{Sin}[q6] - \\ & \text{Cos}[q2 - q3] \text{Sin}[q1] \text{Sin}[q5] \text{Sin}[q6] \end{aligned}$$

$$\begin{aligned} T[2,3] = & \text{Cos}[q2 + q3] \text{Cos}[q5] \text{Sin}[q1] - \\ & \text{Cos}[q4] \text{Sin}[q1] \text{Sin}[q2 + q3] \text{Sin}[q5] - \\ & \text{Cos}[q1] \text{Sin}[q4] \text{Sin}[q5] \end{aligned}$$

$$\begin{aligned} T[2,4] = & a2 \text{Cos}[q2] \text{Sin}[q1] + a3 \text{Sin}[q1] \text{Sin}[q2 + q3] - \\ & d4 \text{Cos}[q2 + q3] \text{Sin}[q1] + \\ & d6 \text{Cos}[q2 + q3] \text{Cos}[q5] \text{Sin}[q1] + \\ & d6 \text{Cos}[q4] \text{Sin}[q1] \text{Sin}[q2 + q3] \text{Sin}[q5] - \\ & d6 \text{Cos}[q1] \text{Sin}[q4] \text{Sin}[q5] \end{aligned}$$

$$\begin{aligned} T[3,1] = & -(\text{Cos}[q2 + q3] \text{Cos}[q4] \text{Cos}[q5] \text{Cos}[q6]) + \\ & \text{Cos}[q6] \text{Sin}[q2 + q3] \text{Sin}[q5] + \\ & \text{Cos}[q2 + q3] \text{Sin}[q4] \text{Sin}[q6] \end{aligned}$$

$$\begin{aligned} T[3,2] = & \text{Cos}[q2 + q3] \text{Cos}[q6] \text{Sin}[q4] + \\ & \text{Cos}[q2 + q3] \text{Cos}[q4] \text{Cos}[q5] \text{Sin}[q6] - \\ & \text{Sin}[q2 + q3] \text{Sin}[q5] \text{Sin}[q6] \end{aligned}$$

$$T[3,3] = \text{Cos}[q5] \text{Sin}[q2 + q3] + \text{Cos}[q2 + q3] \text{Cos}[q4] \text{Sin}[q5]$$

$$\begin{aligned} T[3,4] = & a2 \text{Sin}[q2] - a3 \text{Cos}[q2 + q3] + \\ & d4 \text{Sin}[q2 + q3] + d6 \text{Cos}[q5] \text{Sin}[q2 + q3] + \\ & d6 \text{Cos}[q2 + q3] \text{Cos}[q4] \text{Sin}[q5] \end{aligned}$$

$$T[4,1] = 0$$

$$T[4,2] = 0$$

$$T[4,3] = 0$$

$$T[4,4] = 1$$

The ROBOTICA input file used for these calculations are listed in Table A.2 based on the D-H table.

To make the 3 DOF planar motion, the joints rotated along  $y$ -axis are taken and other joints are set to zero. For the simple case, the link length between joint 3 and 4,  $a_3$ , is also set to zero. The final forms of 3 DOF planar robot corresponding to 6 DOF PUMA 560 are generated as following equations.

$$\mathbf{T} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} T[1,1] & T[1,2] & T[1,3] & T[1,4] \\ T[2,1] & T[2,2] & T[2,3] & T[2,4] \\ T[3,1] & T[3,2] & T[3,3] & T[3,4] \\ T[4,1] & T[4,2] & T[4,3] & T[4,4] \end{bmatrix} \quad (\text{A.2})$$

$$\begin{aligned} n_x &= \sin(q_2 + q_3 + q_5) \\ n_y &= 0 \\ n_z &= \cos(q_2 + q_3 + q_5) \\ s_x &= -\cos(q_2 + q_3 + q_5) \\ s_y &= 0 \\ s_z &= \sin(q_2 + q_3 + q_5) \\ a_x &= 0 \\ a_y &= 1 \\ a_z &= 0 \end{aligned} \quad (\text{A.3})$$

$$\begin{aligned} p_x &= a_2 \cos(q_2) + d_4 \cos(q_2 + q_3) + (d_6 + d_t) \cos(q_2 + q_3 + q_5) \\ p_y &= 0 \\ p_z &= a_2 \sin(q_2) + d_4 \sin(q_2 + q_3) + (d_6 + d_t) \sin(q_2 + q_3 + q_5) \end{aligned} \quad (\text{A.4})$$

A Robotica input data file for a PUMA 6-dof robot  
 {Kinematics}

---

DOF = 6  
 The Denavit-Hartenburg parameters  
 joint1 = revolute  
 a1 = 0  
 alpha1 = Pi/2  
 d1 = 0  
 theta1 = q1  
 joint2 = revolute  
 a2 = a2  
 alpha2 = 0  
 d2 = 0  
 theta2 = q2  
 joint3 = revolute  
 a3 = a3  
 alpha3 = -Pi/2  
 d3 = 0  
 theta3 = q3-Pi/2  
 joint4 = revolute  
 a4 = 0  
 alpha4 = Pi/2  
 d4 = d4  
 theta4 = q4  
 joint5 = revolute  
 a5 = 0  
 alpha5 = -Pi/2  
 d5 = 0  
 theta5 = q5  
 joint6 = revolute  
 a6 = 0  
 alpha6 = 0  
 d6 = d6  
 theta6 = q6

Table A.2 ROBOTICA Input Data File for a PUMA 560 6 DOF Robot

## *Appendix B. MATLAB Files and SIMULINK Diagrams*

The files and specific simulation block diagrams used in this thesis are listed in this Appendix. The gross and fine motion simulations are performed separately but the kinematics and dynamics model of each motion control have the same data and structures except for gains (These gains are listed in Chapter 4).

### *B.1 Files for Gross Motion Control*

```
% fwdkin.m
%
% This function file is to compute the PUMA 3 DOF(Planar) forward
% kinematics using qmez. The equations are derived from Robotica.

function p=fwdkin(q)

% PUMA link parameters

a1 =0.4318;   a2 = 0.43307;   a3 = 0.05625;

% Using the D-H convention, get the position vector

pos(1)=a1*cos(q(1))+a2*cos(q(1)+q(2))+a3*cos(q(1)+q(2)+q(3));

pos(2)=a1*sin(q(1))+a2*sin(q(1)+q(2))+a3*sin(q(1)+q(2)+q(3));

p=[pos(1); pos(2)];

% End of function file
```

```

% jac3dof.m
%
% This file is to calculate the planar 3 DOF PUMA Jacobian using
% qmez.
% Task 1 is positioning the end-effector.
% Task 2 is orientation.

function J=jac3dof(u)

a1 = 0.4318;   a2 = 0.43307;   a3 = 0.05625;

j11=-a1*sin(u(1))-a2*sin(u(1)+u(2))-a3*sin(u(1)+u(2)+u(3));
j12=-a2*sin(u(1)+u(2))-a3*sin(u(1)+u(2)+u(3));
j13=-a3*sin(u(1)+u(2)+u(3));
j21=a1*cos(u(1))+a2*cos(u(1)+u(2))+a3*cos(u(1)+u(2)+u(3));
j22=a2*cos(u(1)+u(2))+a3*cos(u(1)+u(2)+u(3));
j23=a3*cos(u(1)+u(2)+u(3));
j31=cos(u(1)+u(2)+u(3));
j32=j31;   j33=j31;

J1=[j11 j12 j13;
    j21 j22 j23];

J2=[j31 j32 j33];

J=[J1;J2];

% End of function file.

```



```

% sinvkin.m
%
% This function file is to compute the joint increment
% associate with position increment.
% The algorithm is developed by using the Nakamura's
% singular robust inverse.

function dq=sinvkin(u)

qmez=[u(1); u(2); u(3)]; dr=[u(4); u(5); u(6)];

% define PUMA parameters

J1HI = 3.926990; % 225 degrees in rads
J1LO = -0.785398; % -45 degrees in rads
J2HI = 0.785398; % 45 degrees in rads
J2LO = -3.926990; % -225 degrees in rads
J3LO = -1.745329; % -100 degrees in rads
J3HI = 1.745329; % 100 degrees in rads

k = 0.01;
a1 = 0.4318; a2 = 0.43307; a3 = 0.05625;

pmez=fwdkin(qmez);
pref=dr(1:2,1)+[pmez(1); pmez(2)];

% check the workspace limitation. if the desired position is
% out bound of workspace, then the controller should command
% to stay in previous joint position until the position input
% is within its workspace.

if sqrt(pref(1)^2+pref(2)^2) > (a1+a2+a3)

dq=[0;0;0];

else
J=jac3dof(qmez); % getting the jacobian
J_sr=inv(J'*J+k*eye(3))*J'; % take singular inverse
delta_q=J_sr*dr;
q=qmez+delta_q; % update joint values

% check for limitation of updated joint angles. if the joint

```

```

% angles are out of limitation, then their maximum or minimum
% angle is next joint command.

    if (q(1) < J1L0)
q(1) = J1L0;
        elseif (q(1) > J1HI)
q(1) = J1HI;
        else
q(1) = q(1);
        end
    if (q(2) < J2L0)
q(2) = J2L0;
        elseif (q(2) > J2HI)
q(2) = J2HI;
        else
q(2) = q(2);
        end
    if (q(3) < J3L0)
q(3) = J3L0;
        elseif (q(3) > J3HI)
q(3) = J3HI;
        else
q(3) = q(3);
        end

    Qmez=[q(1);q(2);q(3)];
    dq=Qmez-qmez;
end      % end of if-else loop.

% End of sinvkin.m function file.

```

```

% envir.m
%
% This file is to set the position of environment and its stiffness gain.
% The output is the environmental force components.

function Fe=envir(u)

Ke = 15100;   % environment stiffness gain

x=[u(1); u(2)];
xe = 0.55;

if x(1) > xe
    xnet=[x(1)-xe;0];
else
    xnet=[0;0];
end

Fe = Ke*[xnet(1);xnet(2)];   % environmental interacting
                             % force

% End of function file

```

```

% input1.m
%
% This function file is to compute the desired position vector
% and initial joint command based on each sample time and initial
% joint angle.

function p=input1(u)

% PUMA link parameters

a1 =0.4318;   a2 = 0.43307;   a3 = 0.05625;

% initial joint values

q = [u(3); u(4); u(5)];

% Using the D-H convention, get the position vector

pos(1)=a1*cos(q(1))+a2*cos(q(1)+q(2))+a3*cos(q(1)+q(2)+q(3));
pos(2)=a1*sin(q(1))+a2*sin(q(1)+q(2))+a3*sin(q(1)+q(2)+q(3));

p=[pos(1)+0.018*u(1); pos(2)];   % contact with environment
% p=[pos(1)-0.12*u(1); pos(2)];   % free space motion tracking

% End of function file

% input2.m
%
% This function file is to compute the desired task2 vector.
% The desired input is set to the sum of initial joint
% angles.

function p=input2(u)

p = sin(u(1)+u(2)+u(3));

% End of function file

```

*B.2 SIMULINK Block Diagrams of Gross Motion Control*

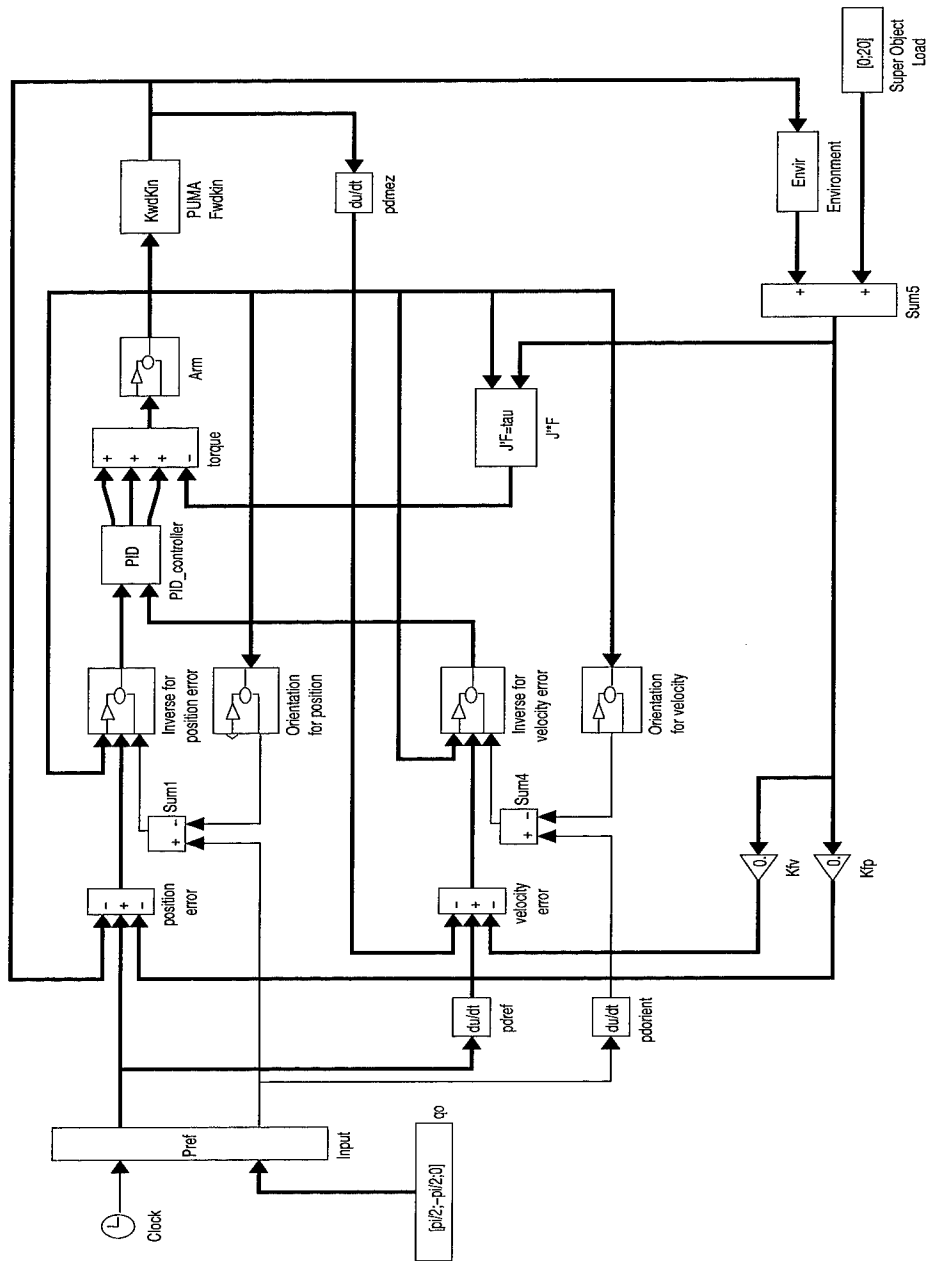
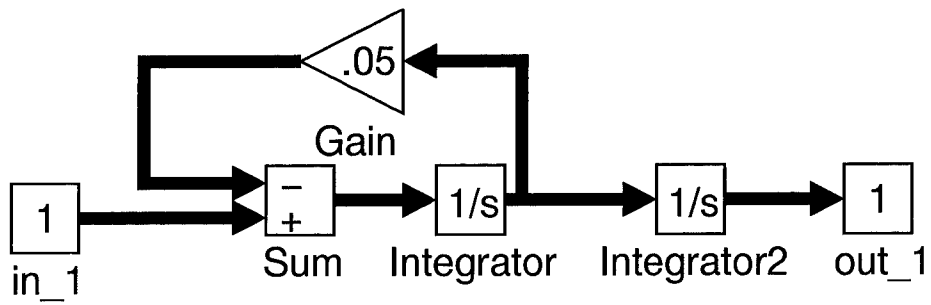
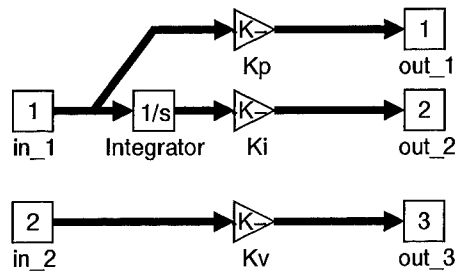


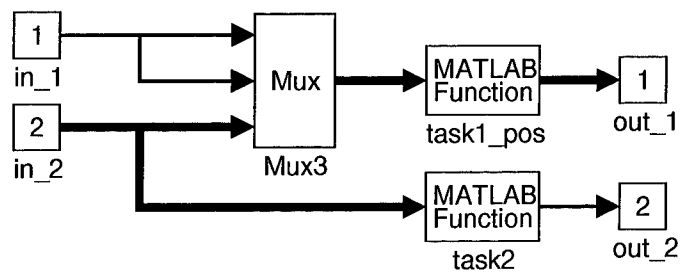
Figure B.1 SIMULINK Block Diagram of Gross Motion Control



(a) Robot Arm Dynamic Model



(b) Model of PID Controller



(c) Model of Input Generator

Figure B.2 Subsystem Block Diagrams of Gross Motion Control

### B.3 Files for Fine Motion Control

#### B.3.1 Finger # 1.

```
% fwdkin1.m
%
% This function file is to compute the 3 DOF(Planar) finger 1 forward
% kinematics using qmez. The equations are derived from Robotica.

function p=fwdkin1(q)

% PUMA link parameters

a1 =0.4318;   a2 = 0.43307;   a3 = 0.05625;

% Using the D-H convention, get the position vector

pos(1)=-a1*sin(q(1))-a2*sin(q(1)+q(2))-a3*sin(q(1)+q(2)+q(3)) - 0.3;

pos(2)= a1*cos(q(1))+a2*cos(q(1)+q(2))+a3*cos(q(1)+q(2)+q(3));

p=[pos(1); pos(2)];

% End of function file
```



```

% jac3dof1.m
%
% This file is to calculate the planar 3 DOF PUMA Jacobian using
% qmez.
% Task 1 is positioning the end-effector.
% Task 2 is orientation.

function J=jac3dof1(u)

a1 = 0.4318;   a2 = 0.43307;   a3 = 0.05625;

j11=-a1*cos(u(1))-a2*cos(u(1)+u(2))-a3*cos(u(1)+u(2)+u(3));
j12=-a2*cos(u(1)+u(2))-a3*cos(u(1)+u(2)+u(3));
j21=-a1*sin(u(1))-a2*sin(u(1)+u(2))-a3*sin(u(1)+u(2)+u(3));
j22=-a2*sin(u(1)+u(2))-a3*sin(u(1)+u(2)+u(3));

J = [j11 j12;
      j21 j22];

% End of function file.

```

```

% sinvkin1.m
%
% This function file is to compute the joint increment
% associate with position increment for the finger 1.
% The algorithm is developed by using the Nakamura's
% singular robust inverse.

function dq=sinvkin1(u)

qmez=[u(1); u(2); 0]; dr=[u(3); u(4)];

% define PUMA parameters

J1HI = 3.926990;           % 225 degrees in rads
J1LO = -0.785398;         % -45 degrees in rads
J2HI = 0.785398;         % 45 degrees in rads
J2LO = -3.926990;        % -225 degrees in rads

k = 0.01;
a1 = 0.4318;   a2 = 0.43307;   a3 = 0.05625;

J=jac3dof1(qmez);           % getting the jacobian
J_sr=inv(J'*J+k*eye(2))*J'; % take singular inverse
delta_q=J_sr*dr;
q(1) = qmez(1)+delta_q(1);   % update joint values
q(2) = qmez(2)+delta_q(2);

% check for limitation of updated joint angles. if the joint
% angles are out of limitation, then their maximum or minimum
% angle is next joint command.

    if (q(1) < J1LO)
q(1) = J1LO;
        elseif (q(1) > J1HI)
q(1) = J1HI;
        else
q(1) = q(1);
        end
        if (q(2) < J2LO)
q(2) = J2LO;
        elseif (q(2) > J2HI)
q(2) = J2HI;

```

```
        else
q(2) = q(2);
        end

        Qmez=[q(1);q(2)];
        dq=[Qmez-qmez(1:2,1)];

% End of sinvkin.m function file.
```

### B.3.2 Finger # 2.

```
% fwdkin2.m
%
% This function file is to compute the 3 DOF(Planar) finger 2 forward
% kinematics using qmez. The equations are derived from Robotica.

function p=fwdkin2(q)

% PUMA link parameters

a1 =0.4318;   a2 = 0.43307;   a3 = 0.05625;

% Using the D-H convention, get the position vector

pos(1)=-a1*sin(q(1))-a2*sin(q(1)+q(2))-a3*sin(q(1)+q(2)+q(3)) + 0.3;

pos(2)=a1*cos(q(1))+a2*cos(q(1)+q(2))+a3*cos(q(1)+q(2)+q(3));

p=[pos(1); pos(2)];

% End of function file
```

```

% jac3dof2.m
%
% This file is to calculate the planar 3 DOF Jacobian of finger 2 using
% qmez.

function J=jac3dof2(u)

a1 = 0.4318;   a2 = 0.43307;   a3 = 0.05625;

j11=-a1*cos(u(1))-a2*cos(u(1)+u(2))-a3*cos(u(1)+u(2)+u(3));
j12=-a2*cos(u(1)+u(2))-a3*cos(u(1)+u(2)+u(3));
j21=-a1*sin(u(1))-a2*sin(u(1)+u(2))-a3*sin(u(1)+u(2)+u(3));
j22=-a2*sin(u(1)+u(2))-a3*sin(u(1)+u(2)+u(3));

J = [j11 j12;
     j21 j22];

% End of function file.

```

```

% sinvkin2.m
%
% This function file is to compute the joint increment
% associate with position increment for the finger 2.
% The algorithm is developed by using the Nakamura's
% singular robust inverse.

function dq=sinvkin2(u)

qmez=[u(1); u(2); 0]; dr=[u(3); u(4)];

% define PUMA parameters

J1LO = -3.926990;           % -225 degrees in rads
J1HI =  0.785398;         %  45 degrees in rads
J2LO = -0.785398;         % -45 degrees in rads
J2HI =  3.926990;         % 225 degrees in rads

k = 0.01;
a1 = 0.4318;   a2 = 0.43307;   a3 = 0.05625;

J=jac3dof2(qmez);           % getting the jacobian
J_sr=inv(J'*J+k*eye(2))*J'; % take singular inverse
delta_q=J_sr*dr;
q(1) = qmez(1)+delta_q(1);   % update joint values
q(2) = qmez(2)+delta_q(2);

% check for limitation of updated joint angles. if the joint
% angles are out of limitation, then their maximum or minimum
% angle is next joint command.

    if (q(1) < J1LO)
q(1) = J1LO;
        elseif (q(1) > J1HI)
q(1) = J1HI;
        else
q(1) = q(1);
        end
    if (q(2) < J2LO)
q(2) = J2LO;
        elseif (q(2) > J2HI)
q(2) = J2HI;

```

```
        else
q(2) = q(2);
        end

Qmez=[q(1);q(2)];
dq=[Qmez-qmez(1:2,1)];

% End of sinvkin.m function file.
```

### B.3.3 Object and Disturbance.

```
%
% obj_dist.m
%
% This file is to set the position of object and
% determine the object disturbance force components.

function out = obj_dist(u)

% assign the inputs (finger positions, joint values
% and desired contact forces)

q1 = [u(6); u(7); u(8)]; % joint position of finger 1
q2 = [u(1); u(2); u(3)]; % joint position of finger 2
p1 = [u(9); u(10)]; % position of finger 1
p2 = [u(4); u(5)]; % position of finger 2
step = u(11);
f1 = [u(15); u(16)]; % initial contact force of finger 1
f2 = [u(12); u(13)]; % initial contact force of finger 2
t = u(18); % time

% assign the value of parameters

Ke = 50; % environmental gain

% initial contact position

contact1 = fwdkin1([pi/6;-pi/6;0]);
contact2 = fwdkin2([-pi/6;pi/6;0]);

% object disturbance

xe_step = 0.005*step;
% xe_sine = 0.005*sin(t*pi/5);
sigma = 0.0025/3;
xe_rand = sigma*randn(1,1);
disturb = xe_step;

% contact positions after disturbed

dist_con1 = contact1 + [disturb;0];
dist_con2 = contact2 + [disturb;0];
```



```

% compute object disturbance and measured force

% finger1

R1 = [ cos(q1(1)+q1(2)+q1(3)-pi/2)  sin(q1(1)+q1(2)+q1(3)-pi/2)  0;
      -sin(q1(1)+q1(2)+q1(3)-pi/2)  cos(q1(1)+q1(2)+q1(3)-pi/2)  0;
      0                                0                                1];

F_net1 = Ke*R1*([p1;0] - [dist_con1;0]);
Fe1 = F_net1(1:2,1) + f1;

% finger2

R2 = [ cos(q2(1)+q2(2)+q2(3)+pi/2)  sin(q2(1)+q2(2)+q2(3)+pi/2)  0;
      -sin(q2(1)+q2(2)+q2(3)+pi/2)  cos(q2(1)+q2(2)+q2(3)+pi/2)  0;
      0                                0                                1];

F_net2 = Ke*R2*([p2;0] - [dist_con2;0]);
Fe2 = F_net2(1:2,1) + f2;

Fe = [Fe1; Fe2];
out = [Fe; disturb];

% End of function file

```

B.3.4 CFA Algorithm. [23]

```
% tc_ini.m
%this is an initialization script for the two contact problem
%of contact force assignment, user must specify contact positions
%and normal directions, in object coordinates, before using
%also specify mu, the contact coefficient of friction
% Mark Hunter, Capt, Air Force Institute of Technology
% 29 Mar 96
%
% Modified by Capt. Hyunki Cho, 10 May 96
%

clear all
global T_i n in mu mu2 eta e_I e_Ic num_I e_n_oi r Wp

%assume coeffecient of friction
mu=.4;
mu2=mu^2;
eta=1/sqrt(1+mu^2);

% degrees to radians
deg2rad=pi/180;

% establish object base coordinates
x=[1 0 0]';
y=[0 1 0]';
z=[0 0 1]';

% enter number of contacts
n=2;

% initialize variables

in=1:n;

num_I=(n*n-n)/2; %number of internal forces

% contact positions in object x-y-z coordinates of r

r1 = fwdkin1([pi/6;-pi/6;0]);
r1_x = r1(1,1);
r2 = fwdkin2([-pi/6;pi/6;0]);
```

```

r2_x = r2(1,1);

r(:,1)=[r1_x 0 0]'; %you specify contact positions
r(:,2)=[r2_x 0 0]'; %r(:,1) and r(:,2)

% contact normal directions, in object x-y-z coordinates

e_n_oi(:,1)=[-r1_x 0 0]'; %you specify contact normals
e_n_oi(:,2)=[-r2_x 0 0]'; %e_n_oi(:,1) and e_n_oi(:,2)

for i=1:n

% make sure e_n_oi is converted to unit vector
%
e_n_oi(:,i)=e_n_oi(:,i)/norm(e_n_oi(:,i));

% determine appropriate tangent vectors to contact surface

e_t(:,1)=cross(e_n_oi(:,i),x);
e_t(:,2)=cross(e_n_oi(:,i),y);
e_t(:,3)=cross(e_n_oi(:,i),z);

for j=1:3
norm_e_t(j)=norm(e_t(:,j));
end

i_et=max(find(norm_e_t==max(norm_e_t)));
e_t_1(:,i)=e_t(:,i_et)/norm(e_t(:,i_et));

e_t_2(:,i)=cross(e_t_1(:,i),e_n_oi(:,i));
e_t_2(:,i)=e_t_2(:,i)/norm(e_t_2(:,i));

T_i(3*i-2:3*i,1:3)=...
[e_n_oi(:,i) e_t_1(:,i) e_t_2(:,i)];

end

clear W

```

```

d2r=pi/180; %degree to radian conversion

e_y=[0 1 0]'; %establish global basis vectors
e_x=[1 0 0]';
e_z=[0 0 1]';

% establish internal force unit vectors and their weights associated
% with the fuzzy membership functions

for i=1:n
for j=1:n
    jj=(i-1)*n+j;
    if i==j
e_I(:,jj)=[0 0 0]';
    else
        r_ij=r(:,i)-r(:,j);
        e_I(:,jj)=-r_ij/norm(r_ij); %internal force unit vectors
    end
e_Ic(:,jj)=T_i(3*i-2:3*i,1:3)*e_I(:,jj); %e_I in
                                                %appropriate contact frame
end
end

% begin initialization for pseudo-inverse solution/external force solution
I3=eye(3,3);

for i=1:n
eval(['P' int2str(i) ']=[0 -r(3,' int2str(i) ') r(2,' int2str(i) ');
        r(3,' int2str(i) ') 0 -r(1,' int2str(i) ');
        -r(2,' int2str(i) ') r(1,' int2str(i) ') 0];'])
end

for i=1:n
eval(['Wi=[I3;P',int2str(i),'];']);
W=[W Wi];
end

Wp=pinv(W);

%ensure this grasp configuration will allow force closure

```

```
if( (e_Ic(2,2)^2+e_Ic(3,2)^2-e_Ic(1,2)^2*mu2)>0 | ...  
    (e_Ic(2,3)^2+e_Ic(3,3)^2-e_Ic(1,3)^2*mu2)>0)  
error('Contact configuration is not consistent with force  
      closure')  
  
end  
  
%end initialization routine
```

```

function F=two_c(u)

%this function determines the min norm solution of the contact forces
%F=[f1x f1y f1z f2x f2y f2z]'
%for the two contact problem. this function is called with the
%object wrench, [fx, fy, fz, Mx, My, Mz]' to be applied to the object,
%and returns the stable contact forces required to so. Run tc_ini before
%using, to initialize the contact configuration variables. Only run tc_ini
%once for any contact configuration. Run this function as many times
%as required for contact force solution.
% Mark Hunter, Capt, Air Force Institute of Technology
% 29 Mar 96
%
% Modified by Capt. Hyunki Cho, 10 May 96
%

global T_i n in mu mu2 eta e_I e_Ic num_I e_n_oi r Wp

Q=[u(1);u(2);u(3);u(4);u(5);u(6)];

Fo=Wp*Q;

jj=2; %for contact one
i=1;
foc1=T_i(3*i-2:3*i,1:3)*Fo(1:3); %convert external contact force to contact frame

a1=e_Ic(2,jj)^2+e_Ic(3,jj)^2-e_Ic(1,jj)^2*mu2;
b1=2*(foc1(2)*e_Ic(2,jj)+foc1(3)*e_Ic(3,jj)-foc1(1)*e_Ic(1,jj)*mu2);
c1=foc1(2)^2+foc1(3)^2-foc1(1)^2*mu2;

k11=(-b1+sqrt(b1^2-4*a1*c1))/(2*a1);
k12=(-b1-sqrt(b1^2-4*a1*c1))/(2*a1);

jj=3; %for contact two
i=2;
foc2=T_i(3*i-2:3*i,1:3)*Fo(4:6); %convert external contact force to contact frame

a2=e_Ic(2,jj)^2+e_Ic(3,jj)^2-e_Ic(1,jj)^2*mu2;
b2=2*(foc2(2)*e_Ic(2,jj)+foc2(3)*e_Ic(3,jj)-foc2(1)*e_Ic(1,jj)*mu2);
c2=foc2(2)^2+foc2(3)^2-foc2(1)^2*mu2;

k21=(-b2+sqrt(b2^2-4*a2*c2))/(2*a2);

```

```
k22=(-b2-sqrt(b2^2-4*a2*c2))/(2*a2);
```

```
m1=max(real(k11),real(k12));
```

```
m2=max(real(k21),real(k22));
```

```
k=max(m1,m2);
```

```
F1=Fo(1:3)+k*e_I(1:3,2);
```

```
F2=Fo(4:6)+k*e_I(1:3,3);
```

```
F=[F1; F2];
```

#### *B.4 SIMULINK Block Diagrams of Fine Motion Control*



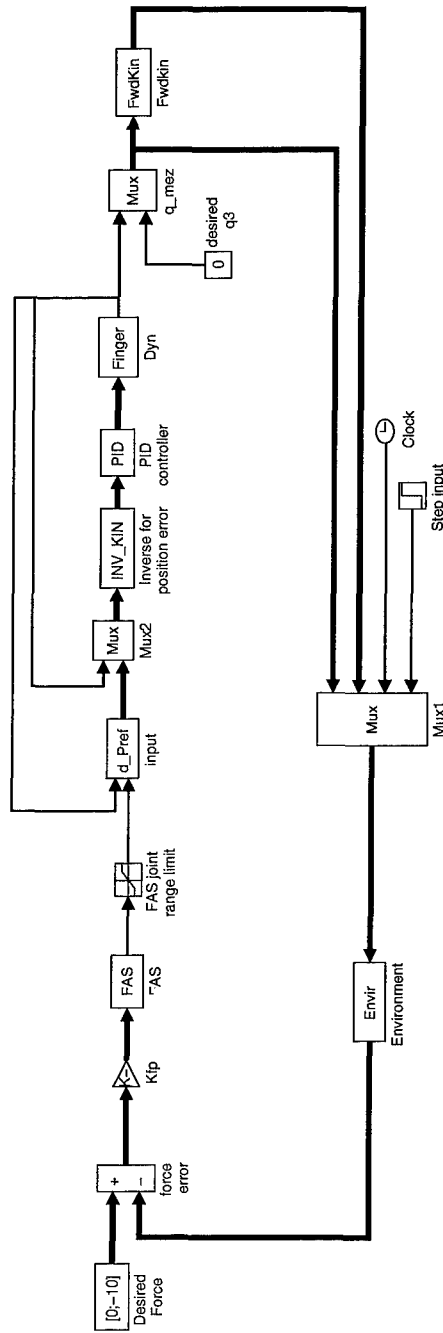
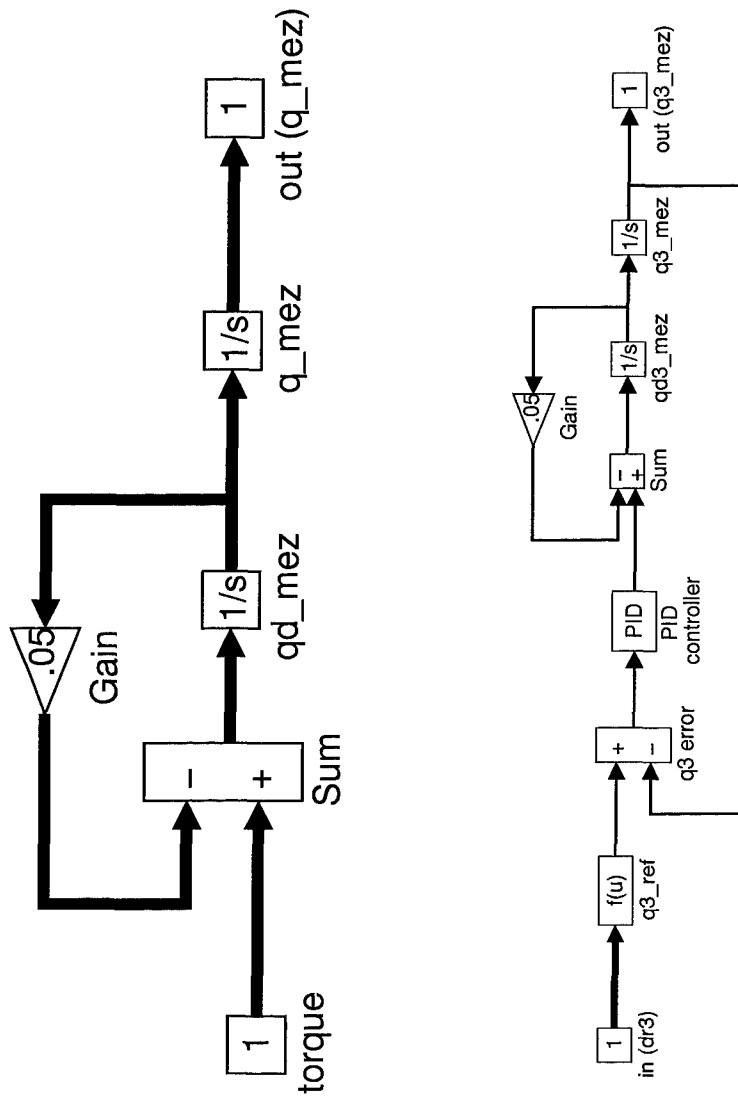


Figure B.3 SIMULINK Block Diagram of Fine Motion Control for One-Finger Motion Control



(a) Finger Dynamic Model      (b) Fingertip Dynamic Model

Figure B.4 Subsystem Block Diagram of Fine Motion Control

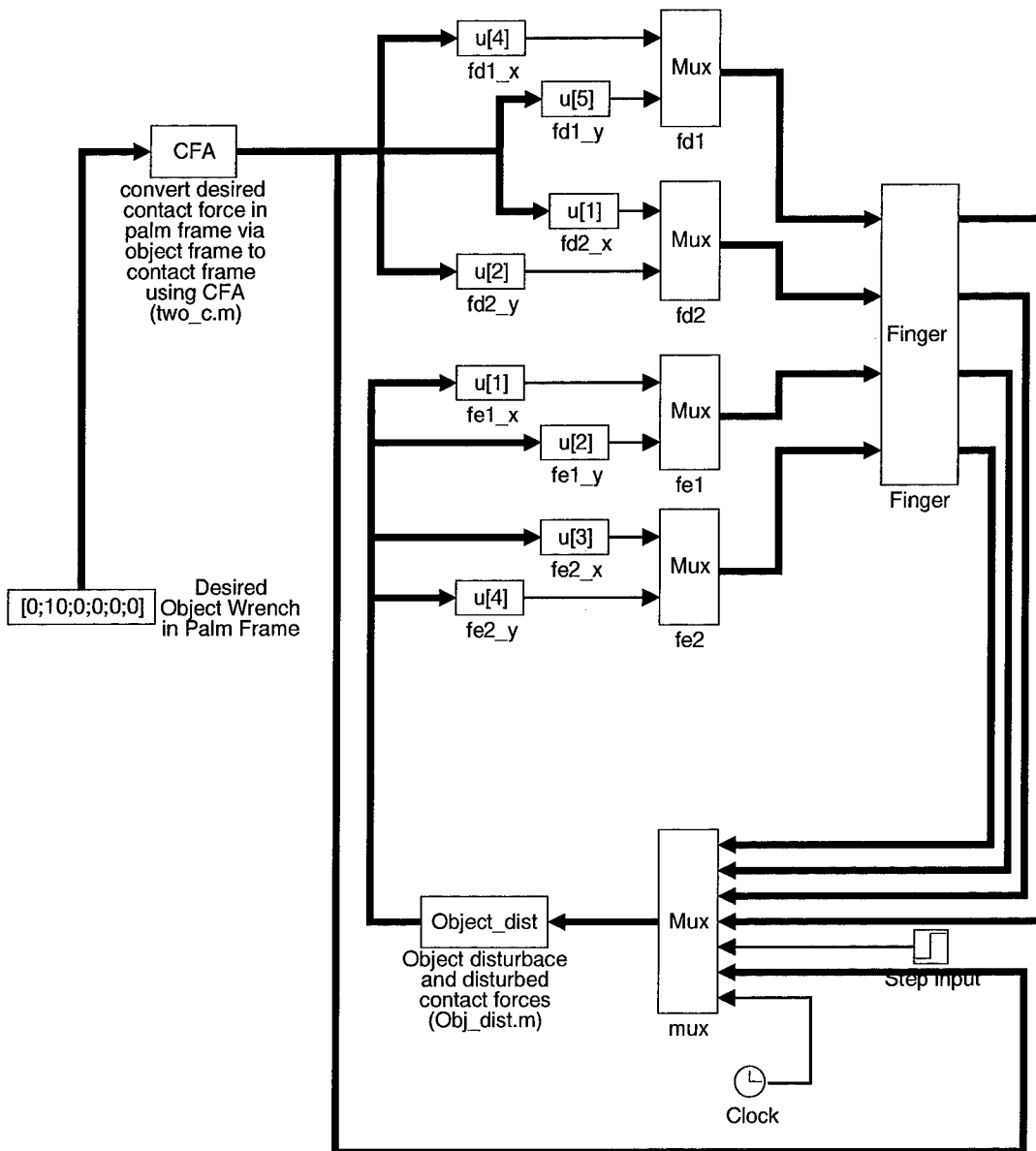


Figure B.5 SIMULINK Block Diagram of Fine Motion Control for Two-Finger Motion Control (Main Block Diagram)

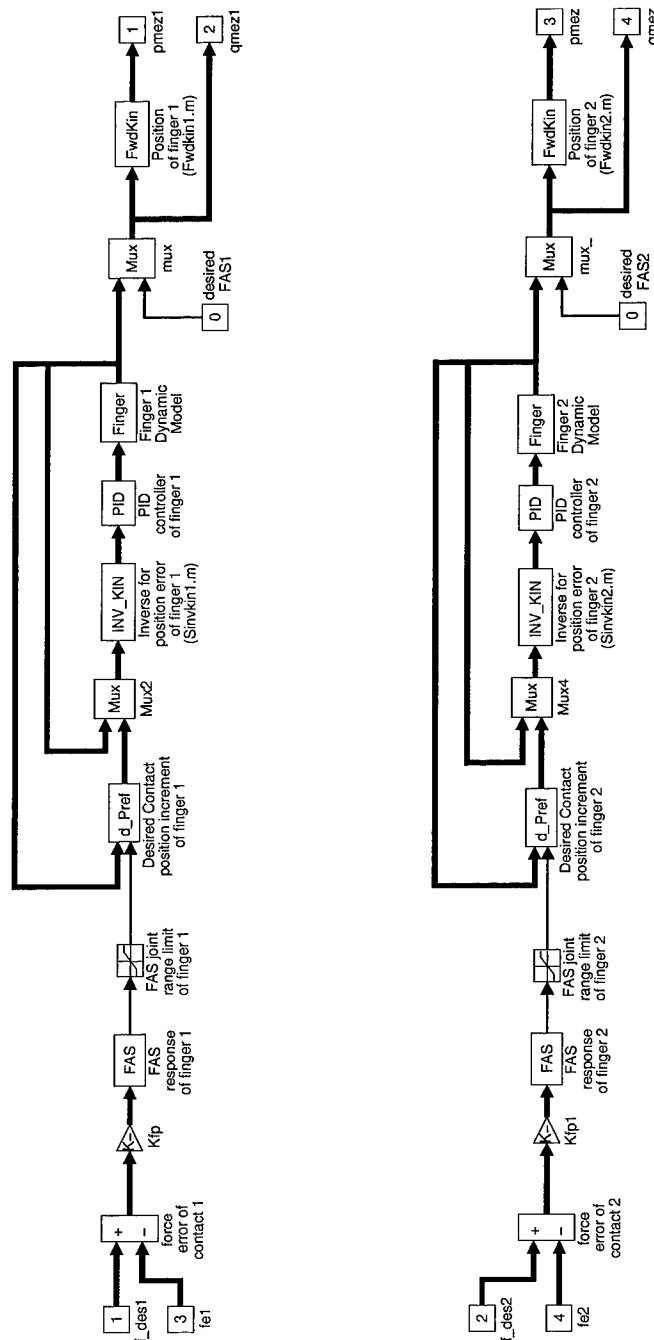


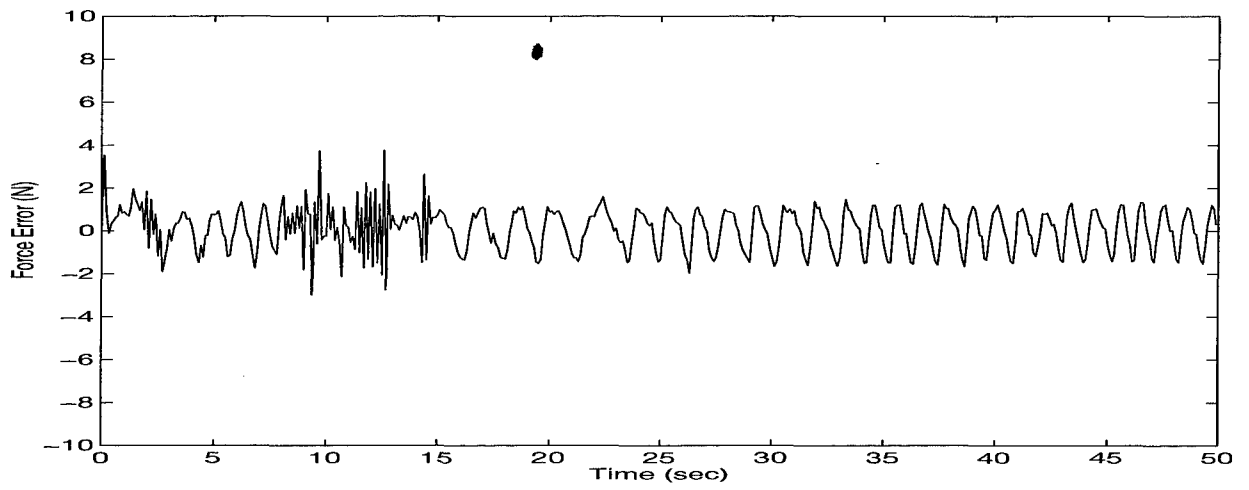
Figure B.6 Finger Subsystem Block Diagram of Fine Motion Control for Two-Finger Motion Control (FAS and Finger Dynamic Subsystem Block Diagrams are the same of One-Finger Simulation)

### *Appendix C. Selection of Suitable Module Sampling Rate*

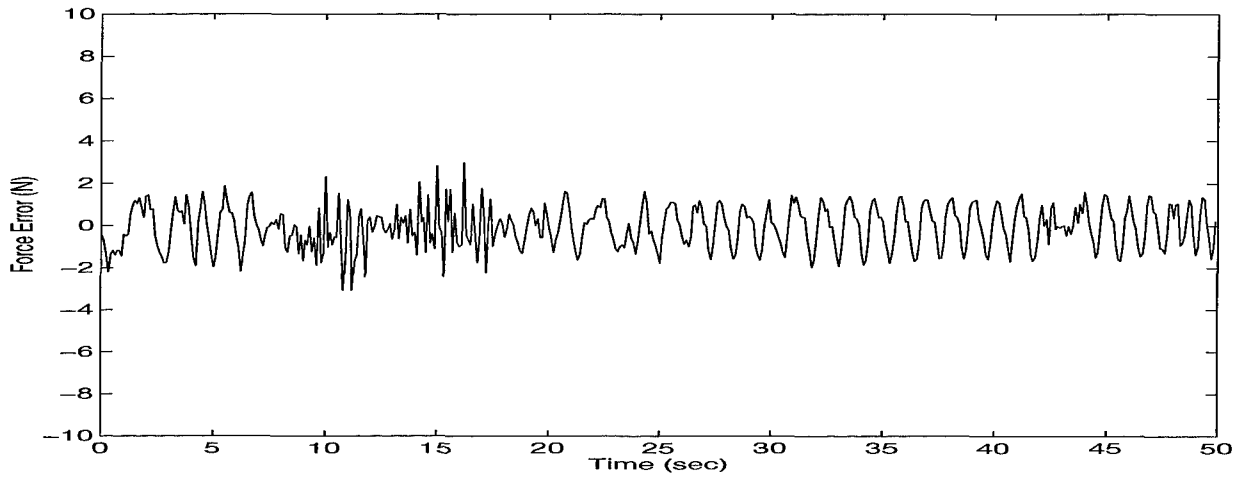
The combinations of module sampling rates and their performances are listed here. The AFIT PUMA 560 was used to show the validation of assumption that the response of lower level control system must be faster than the higher level control system. The consequent results compared with the Chapter 4 and 5 are obtained here and provide the selection of appropriate module sampling rate for the integrated gross and fine motion control experiment. Table (C.1) shows the sets of module sampling rates and corresponding force error ranges. In this thesis, the test set # 5 is selected for the integrated gross and fine motion control experiment.

Test set	Module Sampling Rate			Force Error Range (N)
	<i>gross</i> (Hz)	<i>fine</i> (Hz)	<i>force</i> (Hz)	
<i>test # 1</i>	5	5	5	Unstable
<i>test # 2</i>	5	5	50	-1.561 ~ 3.021
<i>test # 3</i>	5	5	500	-3.8754 ~ 2.9153
<i>test # 4</i>	5	50	50	-1.9554 ~ 1.4682
<i>test # 5</i>	5	50	500	-1.9695 ~ 1.5882
<i>test # 6</i>	5	500	500	-2.0683 ~ 2.3929
<i>test # 7</i>	50	50	50	-8.2449 ~ 6.2894
<i>test # 8</i>	50	50	500	-4.7790 ~ 4.3200
<i>test # 9</i>	50	500	500	-2.1248 ~ 1.7294
<i>test # 10</i>	500	500	500	Unstable
<i>test # 11</i>	500	500	50	-4.2566 ~ 4.3765
<i>test # 12</i>	500	500	5	Unstable
<i>test # 13</i>	500	50	5	Unstable
<i>test # 14</i>	500	50	50	Unstable
<i>test # 15</i>	50	50	250	-1.0660 ~ 3.4588
<i>test # 16</i>	50	250	50	-4.8848 ~ 1.4682
<i>test # 17</i>	250	250	250	-0.1895 ~ 0.1740
<i>test # 18</i>	250	50	50	Unstable
<i>test # 19</i>	250	250	50	Unstable
<i>test # 20</i>	50	250	250	-1.8636 ~ 1.3270
<i>test # 21</i>	250	50	250	-6.9602 ~ 9.7271

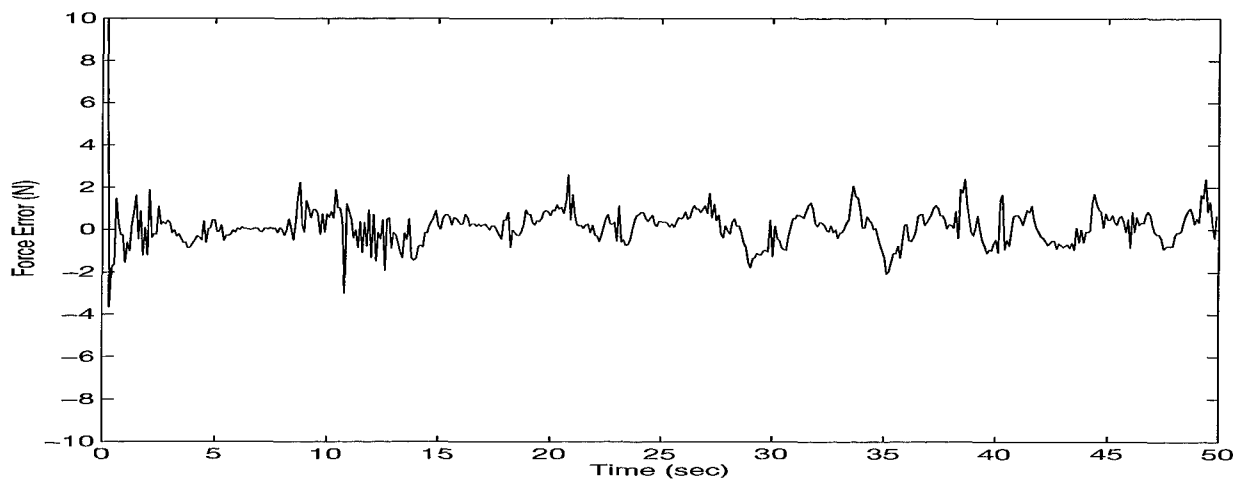
Table C.1 The Sets of Tests for the Gross + Fine Motion Control and Their Performances



(a) test # 4

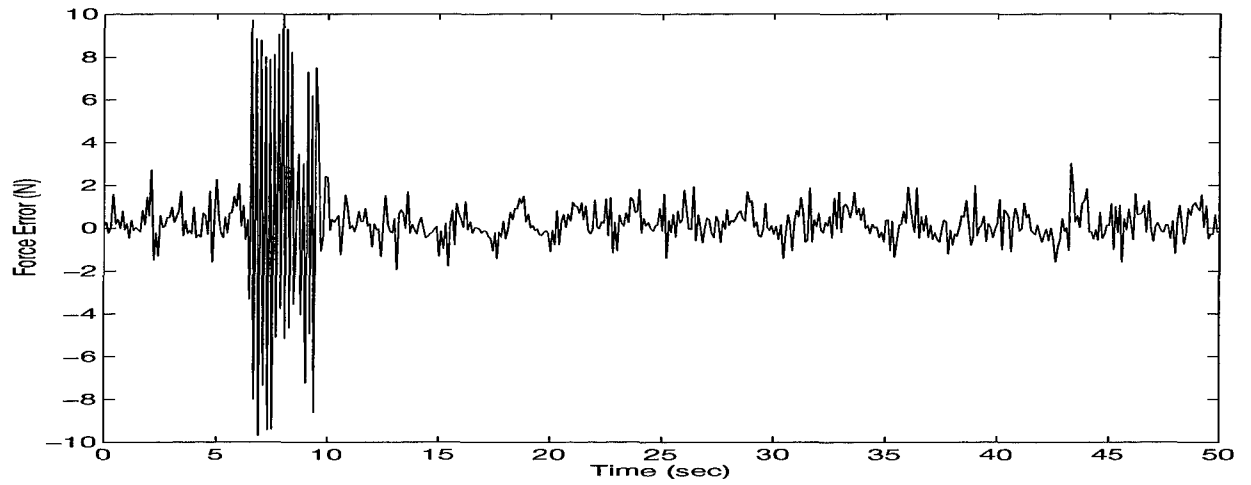


(b) test # 5

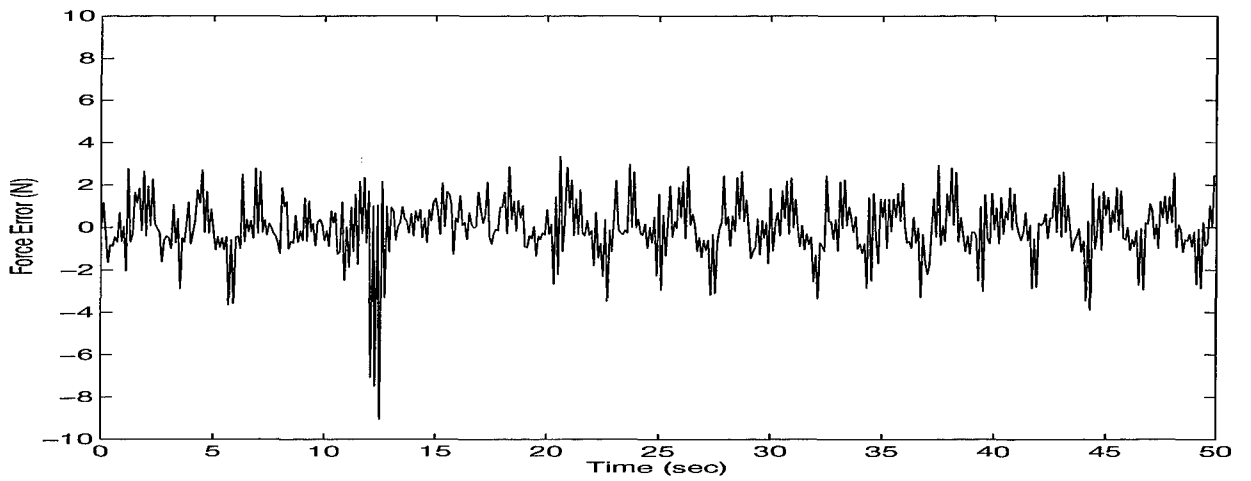


(c) test # 6

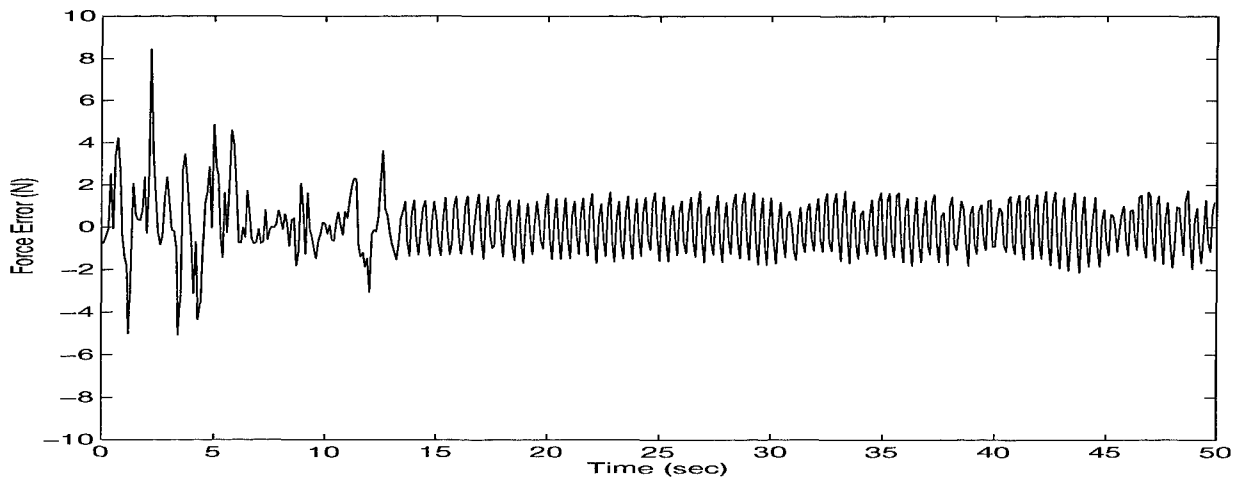
Figure C.1 Stable Cases of Test Sets for Overall Time History



(a) test # 2



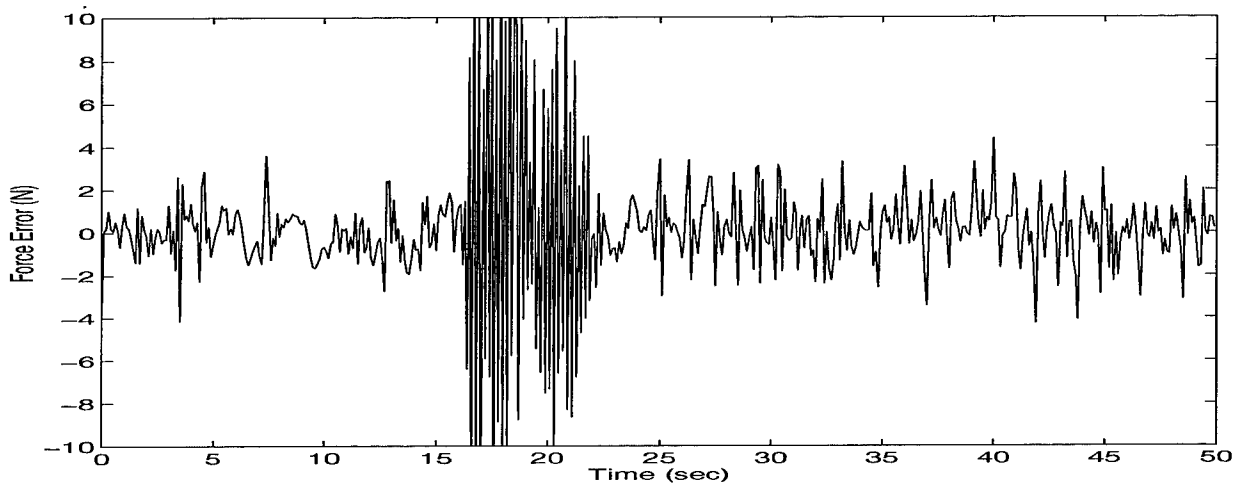
(b) test # 3



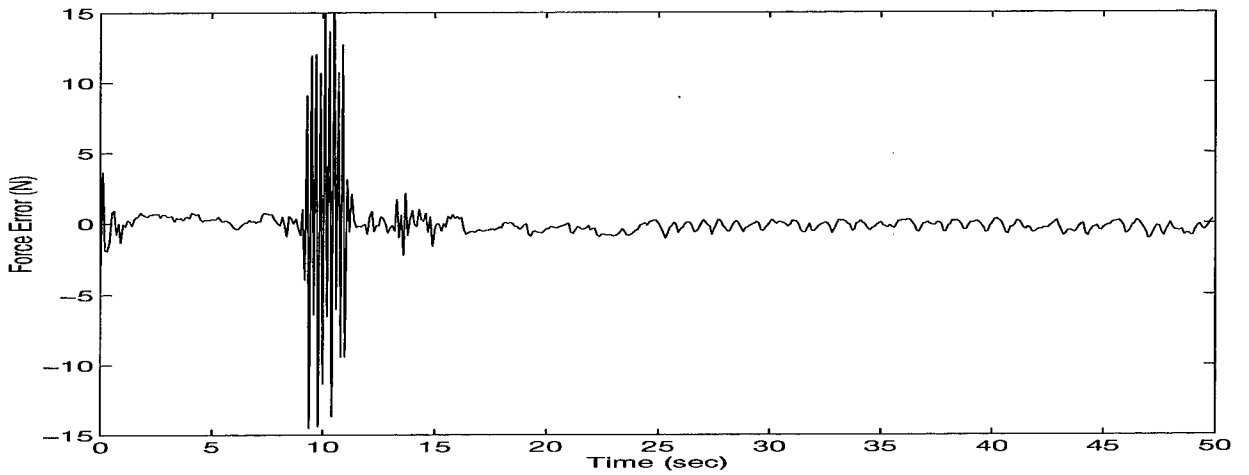
(c) test # 9

Figure C.2 Unstable Cases of Test Sets When Disturbance Input is applied

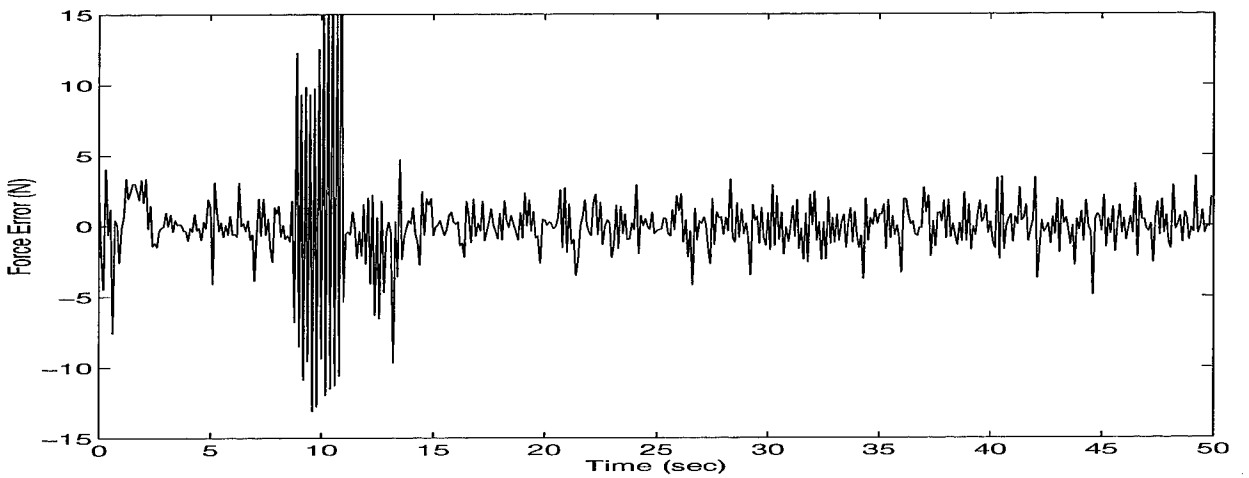




(d) test # 11

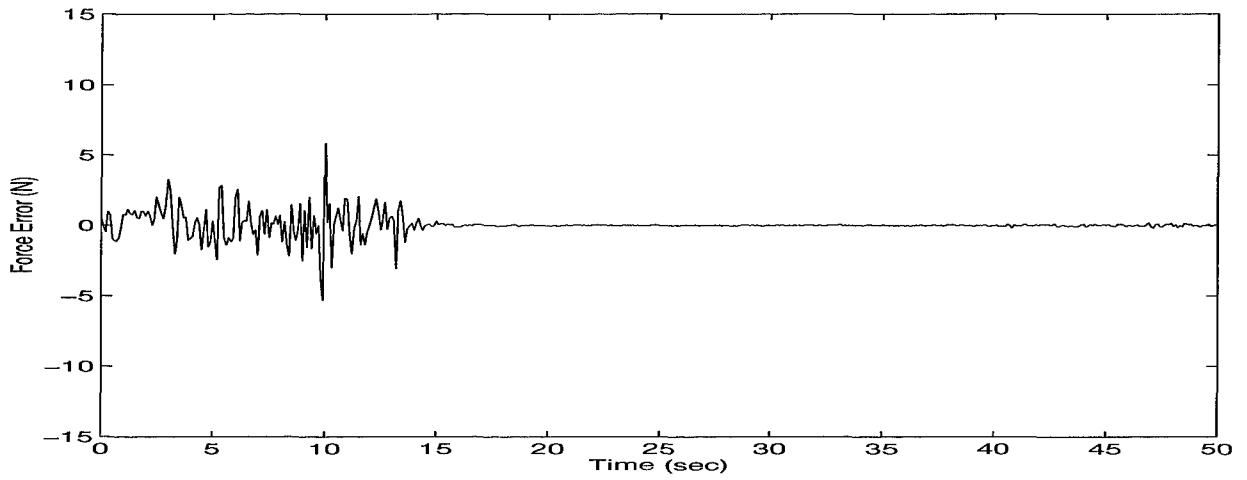


(e) test # 15

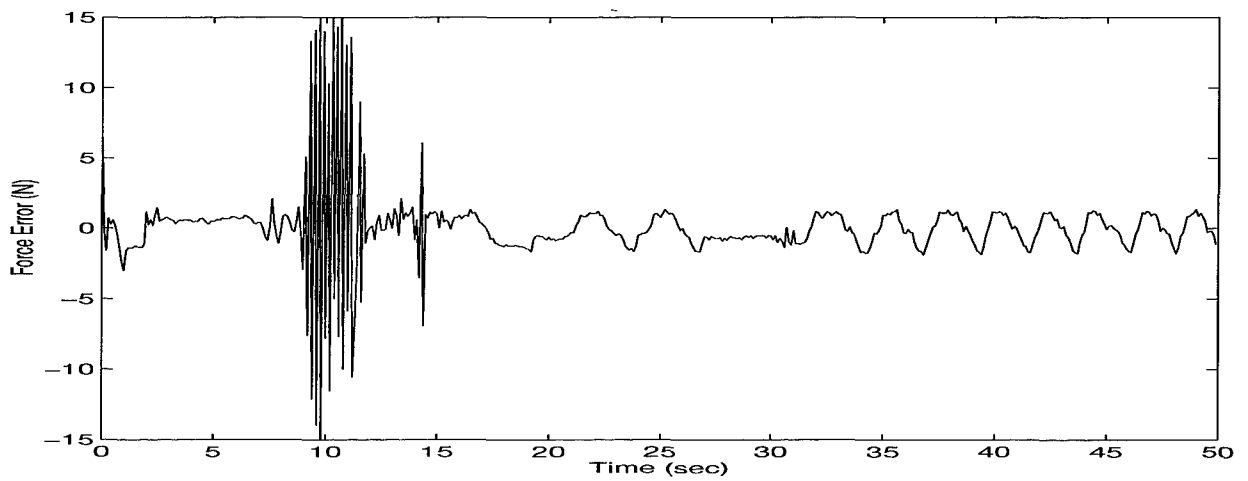


(f) test # 16

Figure C.3 Unstable Cases of Test Sets When Disturbance Input is applied.  
(cont'd)

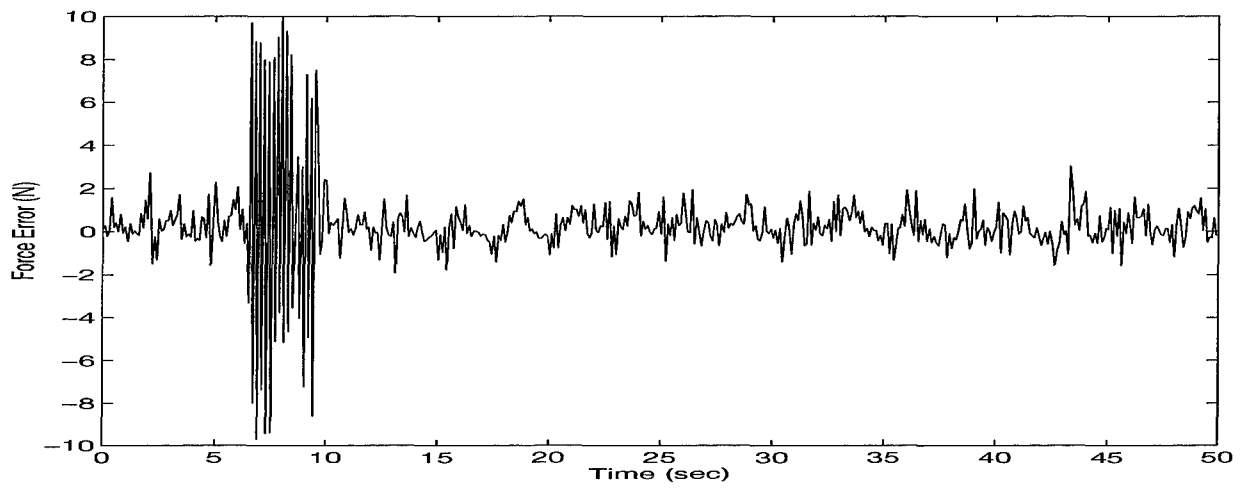


(g) test # 17

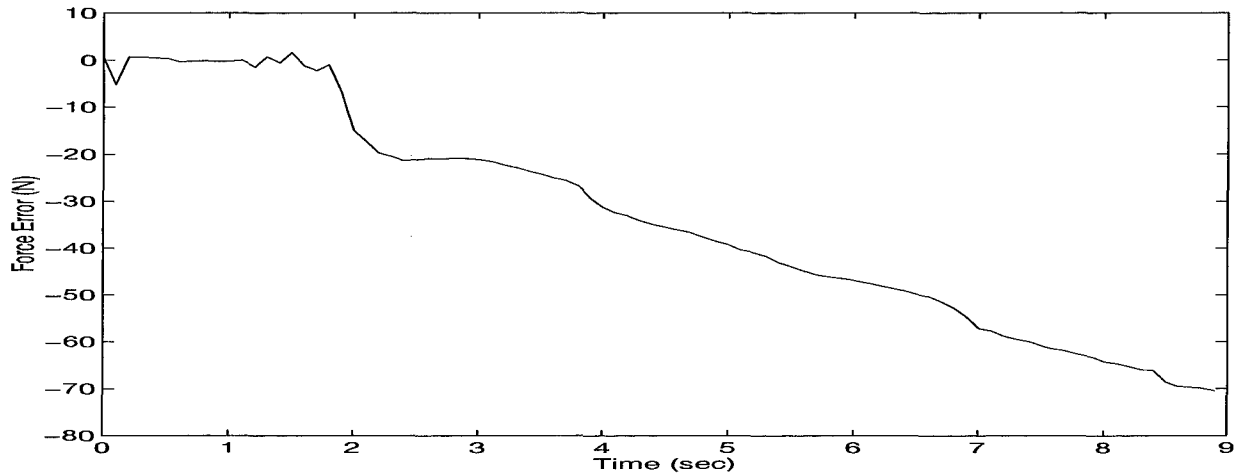


(h) test # 20

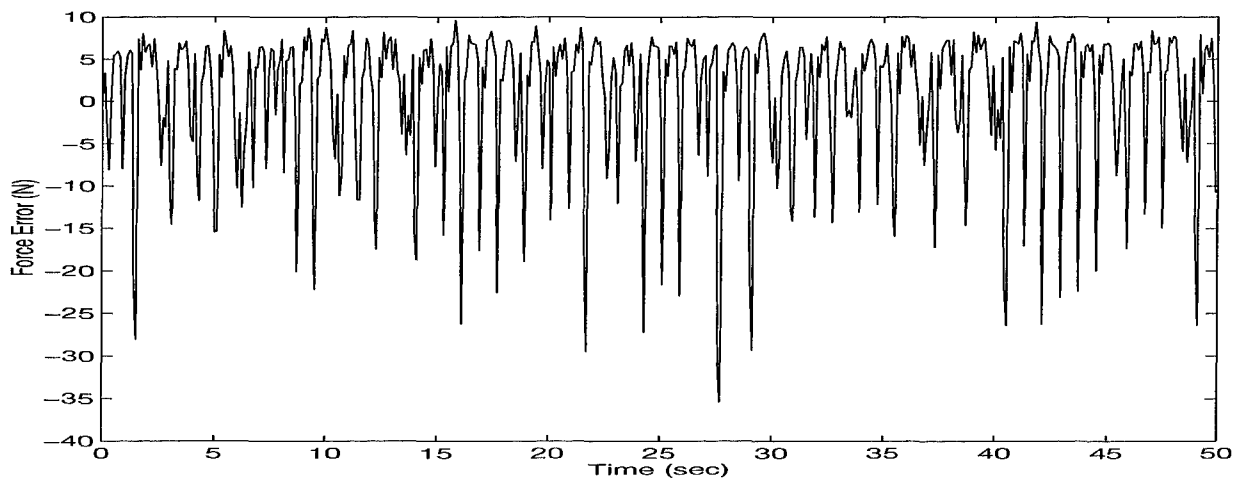
Figure C.4 Unstable cases of Test Sets When Disturbance Input is applied. (cont'd)



(a) test # 7

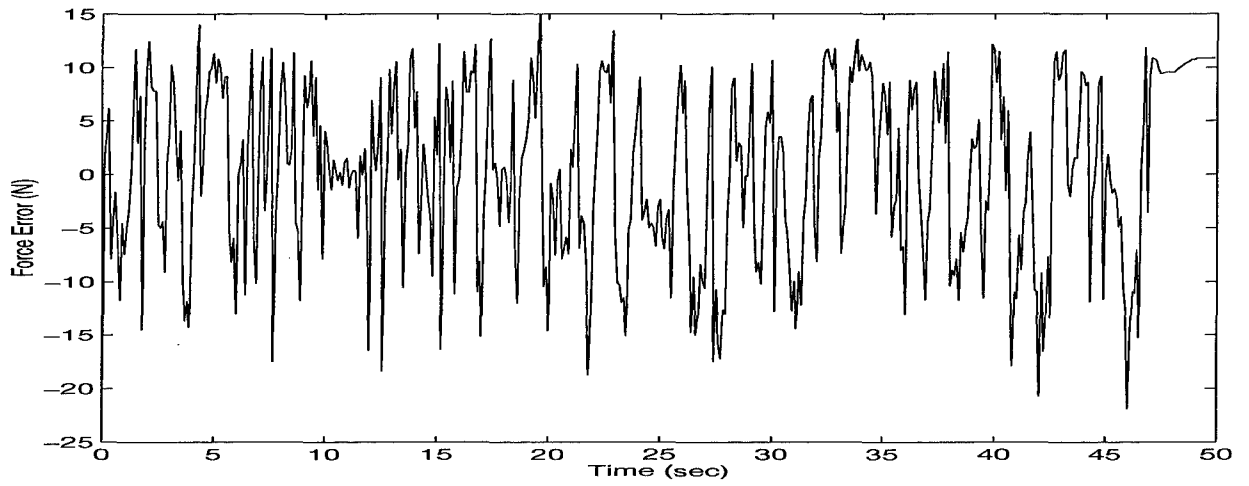


(b) test # 10

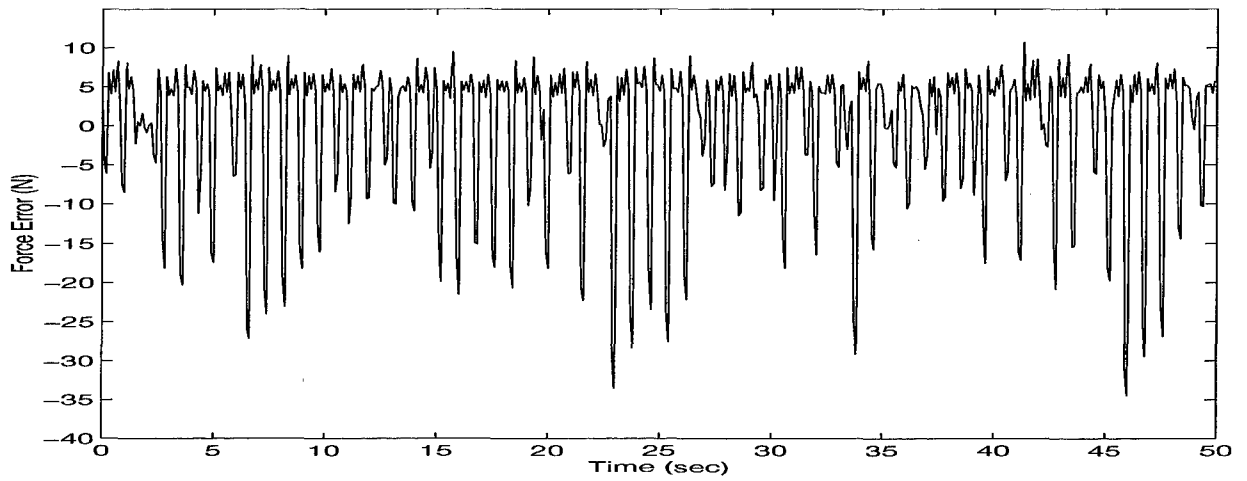


(c) test # 12

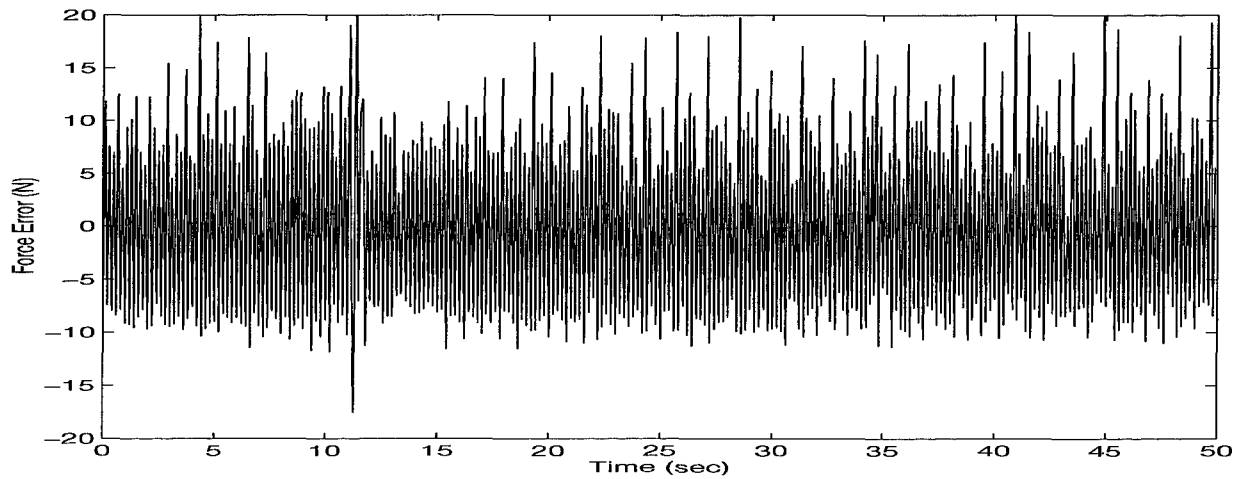
Figure C.5 Unstable cases of Test Sets for Overall Time History



(d) test # 13

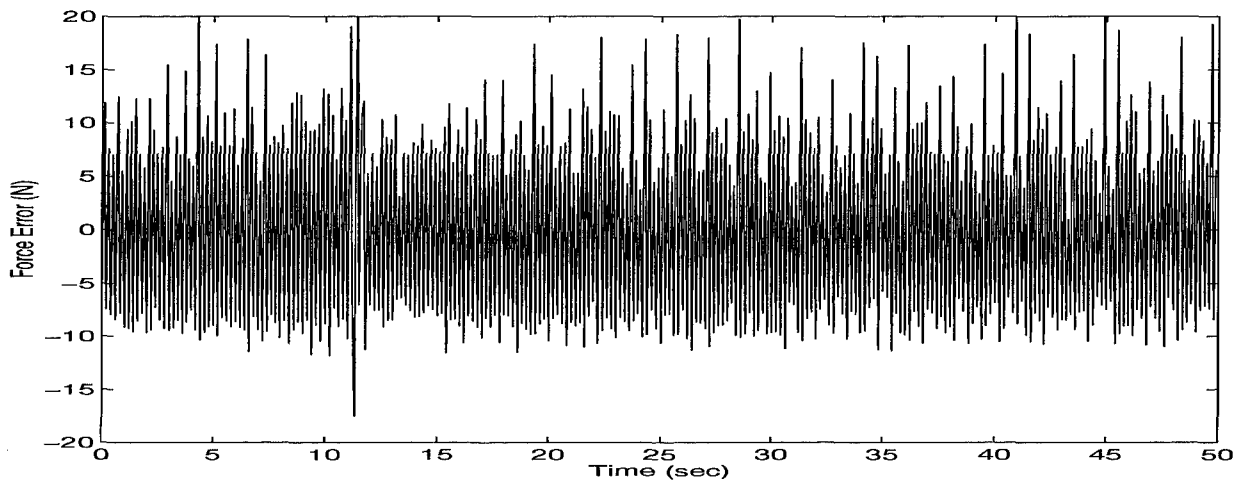


(e) test # 14

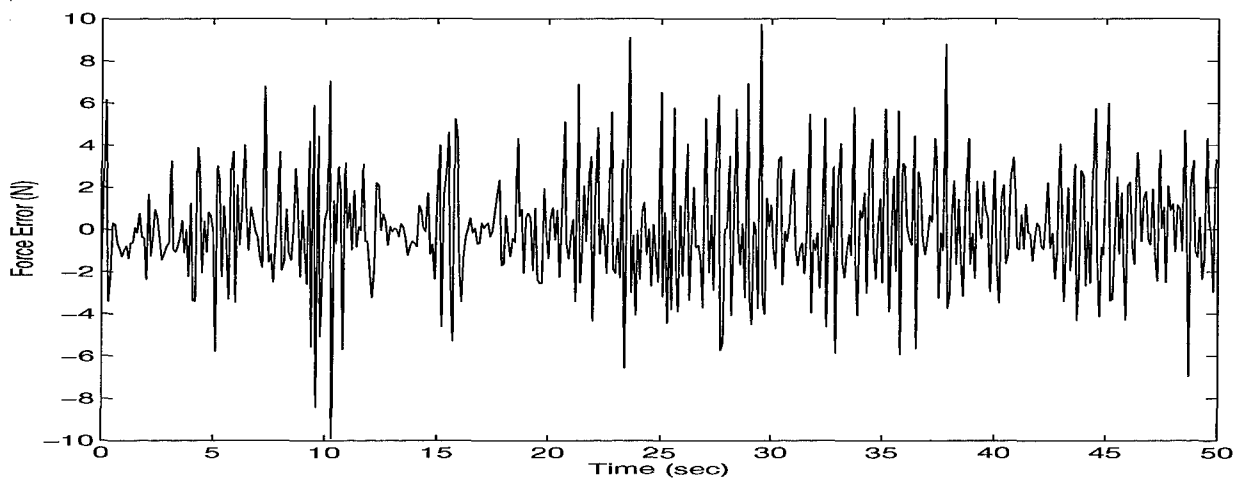


(f) test # 18

Figure C.6 Unstable cases of Test Sets for Overall Time History (cont'd)

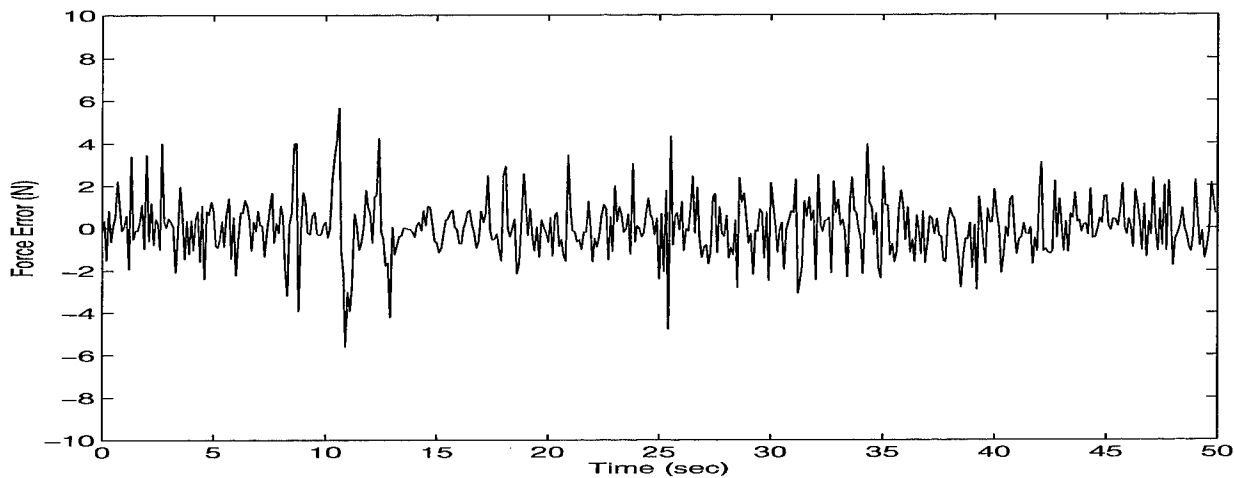


(g) test # 19

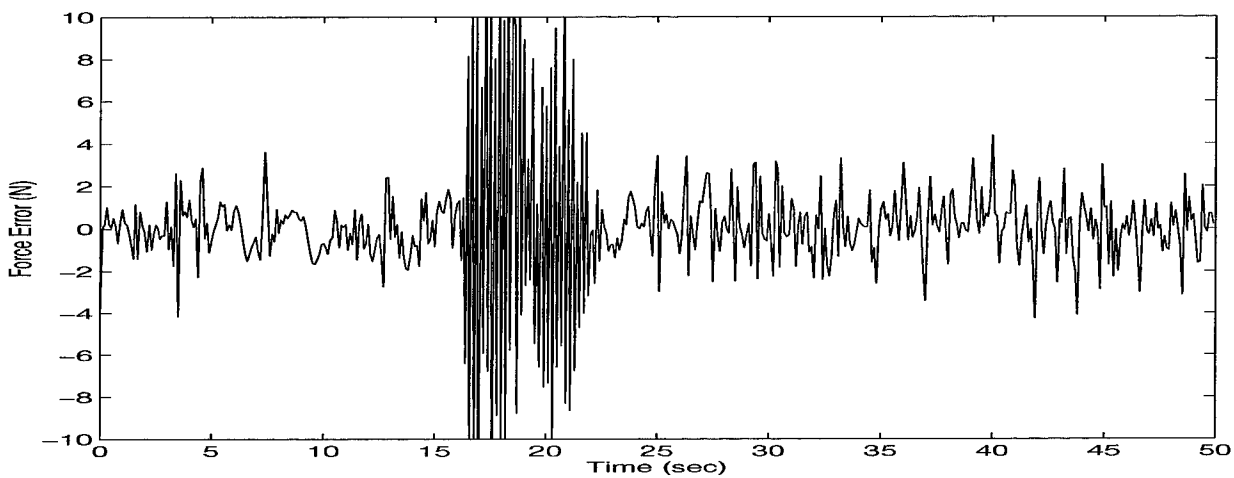


(h) test # 21

Figure C.7 Unstable cases of Test Sets for Overall Time History (cont'd)



(i) test # 8



(j) test # 11

Figure C.8 Unstable cases of Test Sets for Overall Time History (cont'd)

## Appendix D. C-codes for Implementation

The following C-codes are consisted of two parts based on the purpose of the implementation. *imped.c* is for the impedance control and *force.c*, *fine.c* and *gross.c* are for the integration of gross and fine motion control. All of C-codes have their own reconfigurable files.

### D.1 *imped.c* and *imped.rmod*

```
/* ***** */
/* */
/*  imped.c */
/* */
/*  created by Hyunki Cho      11-08-95      */
/*      ( Republic of Korea, Army ) */
/*      Air Force Institute of Technology */
/* */
/*  modified: */
/* */
/*      $ date $ $ initials $ $ comments      $ */
/* */
/* */
/*  reviewed by      $ Some name here $      $ date $ */
/* */
/* ----- */
/* */
/*  Control the PLANAR PUMA3DOF Motion and Force (Joint 2,3
/*  and 5) */
/* */
/*  State variable table: */
/*      INCONST:      none */
/*      OUTCONST:     none */
/*      INVAR:        F_MEZ - measured force */
/*                   N_MEZ - measured normal vector */
/*                   O_MEZ - measured orient vector */
/*                   A_MEZ - measured approach vector */
/*                   P_MEZ - measured position vector */
/*                   P_REF - desired next position */
/*                   Q_MEZ - measured current joint */
/*                   Q^_MEZ - measured joint velocity */
```

```

/*      OUTVAR:          Q_REF - reference joint          */
/*                      Q^_REF - reference joint velocity */
/*                                                              */
/*      Special notes:                                     */
/*          This is an Impedance control module using the   */
/*          Nakamura's Singular Robust Inverse Kinematic   */
/*          solution.                                       */
/*                                                              */
/* ***** */

/* ***** */
/* include files */
/* ***** */

#include <chimera.h>
#include <sbs.h>
#include <math.h>

/* ***** */
/* external function prototypes */
/* ***** */

extern void psrInv();
extern void choFwdKin();
extern int pumaConfig();

/* ***** */
/* module 'Local_t' definition as required by Chimera */
/* ***** */

typedef struct {
    svarVar_t *svarNmez, *svarOmez, *svarAmez, *svarPmez,
        *svarPref;
    float *Qref, *Qdref, *Qmez, *Qdmez;
    float *Fmez, *Pref, *Nmez, *Omez, *Amez, *Pmez;
    float *lastPmez;
    float srKgain, f2pgain, f2vgain, herz;
    float lastq[6], lastqd[6];
    float task2Q[6], task2dQ[6];
    int config;
} impedLocal_t;

```



```

/* ***** */
/* module initialization as required by Chimera */
/* ***** */

SBS_MODULE(imped);

/* ***** */
/* functions */
/* ***** */

/* ***** */
/* impedInit Initialize the module. */
/* ***** */

int impedInit(cinfo, local, stask)
cfigInfo_t *cinfo;
impedLocal_t *local;
sbsTask_t *stask;
{
    sbsSvar_t *svar = &stask->svar;

    /* Get pointers to state variables. */

    local->Fmez = svarTranslateValue(svar->vartable, "F_MEZ",float);
    local->Qref = svarTranslateValue(svar->vartable, "Q_REF",float);
    local->Qdref = svarTranslateValue(svar->vartable, "Q^_REF",float);
    local->Qmez = svarTranslateValue(svar->vartable, "Q_MEZ",float);
    local->Qdmez = svarTranslateValue(svar->vartable, "Q^_MEZ",float);
    local->svarPref = svarTranslate(svar->vartable, "P_REF");
    local->svarPmez = svarTranslate(svar->vartable, "P_MEZ");
    local->svarNmez = svarTranslate(svar->vartable, "N_MEZ");
    local->svarOmez = svarTranslate(svar->vartable, "O_MEZ");
    local->svarAmez = svarTranslate(svar->vartable, "A_MEZ");
    local->Pref = svarValue(local->svarPref, float);
    local->Pmez = svarValue(local->svarPmez, float);
    local->Nmez = svarValue(local->svarNmez, float);
    local->Omez = svarValue(local->svarOmez, float);
    local->Amez = svarValue(local->svarAmez, float);

    /* One time initialization. */

    cfigCompulsory(cinfo, "F2P_GAIN", &local->f2pgain, VT_FLOAT, 1);

```

```

    cfigCompulsory(cinfo, "F2V_GAIN", &local->f2vgain, VT_FLOAT, 1);
    cfigCompulsory(cinfo, "SR_K_GAIN", &local->srKgain, VT_FLOAT, 1);
    cfigCompulsory(cinfo, "HERZ", &local->herz, VT_FLOAT, 1);

    return (int) local;
}

/* ***** */
/* impedReinit Re-Initialize the module. */
/* ***** */

int impedReinit(local, stask)
impedLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */
/* impedOn Start up the module. */
/* ***** */

int impedOn(local, stask)
impedLocal_t *local;
sbsTask_t *stask;
{
    char str_cfg[16];
    int i;
    float joint[6], noap[12];
    sbsSvar_t *svar = &stask->svar;

    for (i = 0; i < 6; ++i)
    {
        joint[i] = local->Qmez[i];
        local->lastq[i] = joint[i];
        local->lastqd[i] = local->Qdmez[i];
        local->task2Q[i] = joint[i];
        local->task2dQ[i] = local->Qdmez[i];
    }

    /* Compute initial configuration. */

```

```

local->config = pumaConfig(joint, str_cfg);

/* Compute initial position and normal vector to prevent large */
/* jump in the first cycle. */

choFwdKin(joint, noap);

for (i=0; i<3; i++)
{
local->Nmez[i] = noap[i];
local->Omez[i] = noap[i+3];
local->Amez[i] = noap[i+6];
local->Pmez[i] = noap[i+9];
local->Pref[i] = noap[i+9];
local->lastPmez[i] = noap[i+9];
}

/* Write these values to the state variable table directly. */

svarWrite(local->svarNmez);
svarWrite(local->svarOmez);
svarWrite(local->svarAmez);
svarWrite(local->svarPmez);
svarWrite(local->svarPref);

/* Display initial configuration information. */

printf("PUMA configuration: %s\n", str_cfg);

/* Return from start up. */
return I_OK;
}

/* ***** */
/* impedCycle Process module information. */
/* ***** */

int impedCycle(local, stask)
impedLocal_t *local;
sbsTask_t *stask;
{
int i;

```

```

float *nmez = local->Nmez, *omez = local->Omez;
float *amez = local->Amez, *pmez = local->Pmez;
float *pref = local->Pref, *fmez = local->Fmez;
float *qref = local->Qref, *qdref = local->Qdref;
float *qmez = local->Qmez, *qdmez = local->Qdmez;
float *joint = local->lastq, *jointd = local->lastqd;
float *task2q = local->task2Q, *task2dq = local->task2dQ;
float q[3], qd[3], Q[3], Qd[3];
float pdes[2], pos[2], lastpmez[2], noap[12];
float pdref[2], pdmez[2], dp[3], ddp[3];
float force[2];
float T = local->herz, K = local->srKgain;
float f2p = local->f2pgain, f2v = local->f2vgain;
float pmez_f[2], pdmez_f[2];
float task2ref, task2dref;

/* Assign joint values to Planar PUMA 3 DOF ***** */

q[0] = joint[1];
q[1] = joint[2];
q[2] = joint[4];

qd[0] = jointd[1];
qd[1] = jointd[2];
qd[2] = jointd[4];

/* Task 1 Calculation ***** */

/* bring the last Pmez from local variables table and store */
/* to lastpmez. */

lastpmez[0] = local->lastPmez[0];
lastpmez[1] = local->lastPmez[2];

/* take the desired position input (pref). */

pdes[0] = pref[0];
pdes[1] = pref[2];

/* take the current position (pos) from local variable table. */

pos[0] = pmez[0];

```

```

pos[1] = pmez[2];

/* re-store the pos to lastPmez for next step. */

local->lastPmez[0] = pos[0];
local->lastPmez[2] = pos[1];

/* compute the desired velocity using pdes and pmez. (pdref) */

for (i=0; i<2; i++)
    pdref[i] = (pdes[i] - pos[i])/T;

/* compute the current velocity using pmez and lastpmez.      */
/* (pdmez). */

for (i=0; i<2; i++)
    pdmez[i] = (pos[i] - lastpmez[i])/T;

/* transform the contact force to Cartesian coordinate frame. */

force[0] = nmez[0]*fmez[1]+omez[0]*fmez[0]-amez[0]*fmez[2];
force[1] = nmez[2]*fmez[1]+omez[2]*fmez[0]-amez[2]*fmez[2];

/* compute the position due to the Sensed Feedback Force. */
/* (pmez_f) --> wrist force sensor. */

for (i=0; i<2; i++)
    pmez_f[i] = force[i]*f2p;

/* compute the velocity due to the sensed feedback force. */
/* (pdmez_f). */

for (i=0; i<2; i++)
    pdmez_f[i] = force[i]*f2v;

/* compute the position / velocity error and store to dp[2] */
/* /ddp[2]. */

for (i=0; i<2; i++)
    {
    dp[i] = pdes[i] - pos[i] - pmez_f[i];
    ddp[i] = pdref[i] - pdmez[i] - pdmez_f[i];
    }

```

```

    }

/* Task 2 Calculation ***** */

/* bring the task2 referenrce input stored in task2ref /      */
/* task2dref. */

task2ref = sin(task2q[1] + task2q[2] + task2q[4]);
task2dref = sin(task2dq[1] + task2dq[2] + task2dq[4]);

/* compute the error and store to dp[2]/ddp[2] */

dp[2] = task2ref - sin(q[0] + q[1] + q[2]);
ddp[2] = task2dref - sin(qd[0] + qd[1] + qd[2]);

/* Compute SR-Inverse ***** */

psrInv(Q, dp, q, pos, K);
psrInv(Qd, ddp, qd, pdmez, K);

/* store new joint values (Qref) to the variable joint and      */
/* jointd. */

joint[1] = Q[0]; /* They were the 3 DOF PLANAR */
joint[2] = Q[1]; /* Thus, re-assign joint values */
joint[4] = Q[2]; /* to spacial case 6 DOF. */

jointd[1] = Qd[0];
jointd[2] = Qd[1];
jointd[4] = Qd[2];

for (i=0; i<6; i++)
{
    qref[i] = joint[i]; /* Store to local variable table*/
    qdref[i] = jointd[i]; /* for 6 DOF calculation. */
}

/* Compute measured forward kinematics using the result Qref.** */

choFwdKin(joint, noap);

for (i=0; i<3; i++)

```

```

    {
        local->Nmez[i] = noap[i];
        local->Omez[i] = noap[i+3];
        local->Amez[i] = noap[i+6];
        local->Pmez[i] = noap[i+9];
    }

    svarWrite(local->svarPmez); /* write the NOAP vector to the */
    svarWrite(local->svarNmez); /* state variable table to use */
    svarWrite(local->svarOmez); /* next step and check the */
    svarWrite(local->svarAmez); /* difference to the P_REF. */

    return I_OK;
}

/* ***** */
/* impedOff Stop the module. */
/* ***** */

int impedOff(local, stask)
impedLocal_t *local;
sbsTask_t *stask;
{
    kprintf("imped: OFF\n");
    return I_OK;
}

/* ***** */
/* impedKill Clean up after the module. */
/* ***** */

int impedKill(local, stask)
impedLocal_t *local;
sbsTask_t *stask;
{
    kprintf("imped: FINISHED\n");
    return I_OK;
}

/* ***** */
/* impedError Attempt automatic error recovery. */
/* ***** */

```

```

int impedError(local, stask, mptr, errmsg, errcode)
impedLocal_t *local;
sbsTask_t *stask;
errModule_t *mptr;
char *errmsg;
int errcode;
{
    /* Return after not correcting error. */

    return SBS_ERROR;
}

/* ***** */
/* impedClear Clear error state of the module. */
/* ***** */

int impedClear(local, stask, mptr, errmsg, errcode)
impedLocal_t *local;
sbsTask_t *stask;
errModule_t *mptr;
char *errmsg;
int errcode;
{
    /* Return after not clearing error. */

    sbsNewError(stask, "Clear not defined, still in error state",
                errcode);

    return SBS_ERROR;
}

/* ***** */
/* impedSet Set module parameters. */
/* ***** */

int impedSet(local, stask)
impedLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

```



```

/* ***** */
/* impedGet Get module parameters. */
/* ***** */

int impedGet(local, stask)
impedLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */
/* impedSync For modules which synchronize to */
/* something other than the clock. */
/* ***** */

int impedSync(local, stask)
impedLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

```

```

#
# imped.rmod
#
# Reconfigurable module file for the impedance control of
# the puma 3 dof planar robot.
#

MODULE      imped
DESC        controls puma in 6 DOF
INCONST     NDOF DH
OUTCONST    none
INVAR       F_MEZ P_REF N_MEZ O_MEZ A_MEZ P_MEZ Q_MEZ Q^_MEZ
OUTVAR      Q_REF Q^_REF
TASKTYPE    periodic
FREQ        100

LOCAL
#F2P_GAIN 0.00041
#F2V_GAIN 0.041
F2P_GAIN 0.00015
F2V_GAIN 0.0015
SR_K_GAIN 0.01
HERZ 100

EOF

```

## D.2 Function files of *imped.c*

```
/* ***** */
/*
/*      psrinvkin.c
/*
/*      modified by      Hyunki Cho      11-10-95
/*
/*      refer to Nakamura's singular robust
/*      inverse
/*
/* -----
/*
/*      Computes SR-inverse kinematics given the error between
/*      current position and desired position for PUMA3DOF PLANAR
/*      case (using joint 2,3 and 5).
/*
/* ***** */

/* ***** */
/*      include files
/* ***** */

#include <chimera.h>
#include <sbs.h>
#include <math.h>
#include "puma.h"

/* ***** */
/*      macro definitions
/* ***** */

#define J2L0    -3.926990      /* -225 degrees in rads */
#define J2HI    0.785398      /* 45 degrees in rads */
#define J3L0    -0.785398      /* -45 degrees in rads */
#define J3HI    3.926990      /* 225 degrees in rads */
#define J5L0    -1.745329      /* -100 degrees in rads */
#define J5HI    1.745329      /* 100 degrees in rads */
#define a2    0.4318      /* length of link 2 */
#define a3    -0.02032
#define d4    0.43307      /* length of link 4 */
#define d6    0.05588      /* length of tool */
```

```

/* ***** */
/* psrInv Compute singular robust inverse kinematics.*/
/* ***** */

void psrInv(q,dp,qmez,pmez,k)
float      dp[3], pmez[2];
float      q[3], qmez[3];
float k;
{
    int i,j;
    float pjac(), inverse();
    float tjac(), multi();
    float p[2], dq[3];
    float      J[3][3], Jt[3][3], Jm[3][3], Jm_inv[3][3], J_sri[3][3];

    for (i=0; i<2; i++)
        p[i] = pmez[i] + dp[i];

/* check limit of workspace whether it is singular position or not. */
/* if singular position, then stay at the current position,          */
/* if not singular position, then proceed SR-Inverse.                */

    if (sqrt(p[0]*p[0]+p[1]*p[1]) >= sqrt((a2+d4+d6)*(a2+d4+d6)+a3*a3))
    {
        for (i = 0; i < 3; i++)
            q[i]=qmez[i];
    }
    else
    {
        /* compute singular robust inverse. */

        pjac(J, qmez); /* get jacobian matrix. */
        tjac(Jt, J); /* compute transpose of jacobian. */
        multi(Jm, Jt,J); /* multiply jacobian and transpose J.*/

        for (i = 0; i < 3; i++)
            Jm[i][i] = Jm[i][i] + k; /* add k term in diagonal of Jm. */

        inverse(Jm_inv, Jm); /* compute inverse of Jm added by k. */
        multi(J_sri, Jm_inv, Jt); /* compute J SR-inverse. */

        /* compute joint increments. */

```

```

dq[0] = J_sri[0][0]*dp[0] + J_sri[0][1]*dp[1] + J_sri[0][2]*dp[2];
dq[1] = J_sri[1][0]*dp[0] + J_sri[1][1]*dp[1] + J_sri[1][2]*dp[2];
dq[2] = J_sri[2][0]*dp[0] + J_sri[2][1]*dp[1] + J_sri[2][2]*dp[2];

/* add the joint increment to qmez. */

for (i = 0; i < 3; i++)
    q[i] = qmez[i] + dq[i];

/* check for the defined joint limit. */

if (q[0] < J2L0) q[0] = J2L0;
if (q[0] > J2HI) q[0] = J2HI;

if (q[1] < J3L0) q[1] = J3L0;
if (q[1] > J3HI) q[1] = J3HI;

if (q[2] < J5L0) q[2] = J5L0;
if (q[2] > J5HI) q[2] = J5HI;

} /* end of if-else loop. */
} /* end of singular robust inverse function. */

/* ***** */
/* Compute PUMA3DOF PLANAR case Jacobian */
/* ***** */
float pjac(j,angle)
float angle[3];
float j[3][3];
{
    float c2, c23, c235, s2, s23, s235;

    s2 = sin(angle[0]);          c2 = cos(angle[0]);
    s23 = sin(angle[0]+angle[1]); c23 = cos(angle[0]+angle[1]);
    s235 = sin(angle[0]+angle[1]+angle[2]);
    c235 = cos(angle[0]+angle[1]+angle[2]);

    j[0][0] = -a2*s2 - a3*s23 + d4*c23 + d6*c235;
    j[0][1] = -a3*s23 + d4*c23 + d6*c235;
    j[0][2] = d6*c235;
    j[1][0] = -a2*c2 - a3*c23 - d4*s23 - d6*s235;

```

```

j[1][1] = -a3*c23 - d4*s23 - d6*s235;
j[1][2] = -d6*s235;
j[2][0] = c235; j[2][1]= c235; j[2][2]= c235;

} /* end of PUMA3DOF PLANAR jacobian function. */

/* ***** */
/* Compute Transpose matrix of PUMA3DOF PLANAR case Jacobian */
/* ***** */

float tjac(jt,jac)
float jt[3][3];
float jac[3][3];
{

    int i,j;

    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            if ( i == j )
                jt[i][i]=jac[i][i];
            else
                jt[j][i]=jac[i][j];

} /* end of matrix transpose function. */

/* ***** */
/* Compute Inverse matrix Using Determinant and Adjoint matrix */
/* ***** */

float inverse(inv_A, A)
float inv_A[3][3];
float A[3][3];
{

float    det, a, b, c, d, e, f, g, h, m;

    a = A[0][0];    b = A[0][1];    c = A[0][2];
    d = A[1][0];    e = A[1][1];    f = A[1][2];
    g = A[2][0];    h = A[2][1];    m = A[2][2];

    det = a*e*m + b*f*g + c*d*h - c*e*g - b*d*m - a*f*h;

```

```

    inv_A[0][0] = (e*m-f*h)/det;
    inv_A[1][0] = (f*g-d*m)/det;
    inv_A[2][0] = (d*h-e*g)/det;
    inv_A[0][1] = (c*h-b*m)/det;
    inv_A[1][1] = (a*m-c*g)/det;
    inv_A[2][1] = (b*g-a*h)/det;
    inv_A[0][2] = (b*f-c*e)/det;
    inv_A[1][2] = (c*d-a*f)/det;
    inv_A[2][2] = (a*e-b*d)/det;

} /* end of matrix inversion function. */

/* ***** */
/* Matrix multiplication for 3 by 3 */
/* ***** */

float multi(D,B,C)
float D[3][3], B[3][3], C[3][3];
{

    int i,j,k;
    float temp;

    D[0][0] = 0; D[0][1] = 0; D[0][2] = 0;
    D[1][0] = 0; D[1][1] = 0; D[1][2] = 0;
    D[2][0] = 0; D[2][1] = 0; D[2][2] = 0;

    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            for (k = 0; k < 3; k++)
                {
                    temp=B[i][k]*C[k][j];
                    D[i][j]=D[i][j]+temp;
                }

} /* end of matrix multiplication function. */

```

```

/* ***** */
/*                                             */
/*   chofwdkin.c                               */
/*                                             */
/*   created by      Wayne F. Carriker      11-24-92   */
/*   (taken from code obtained from Fred Seiler)      */
/*                                             */
/*   modified by      Hyunki Cho      12-16-95   */
/*   Air Force Institute of Technologe      */
/*                                             */
/*   modified for Chimera 3.0 release      */
/*   changed all double precision to float      */
/*                                             */
/* ----- */
/*                                             */
/*   Compute forward kinematics given current joint positions */
/*                                             */
/* ***** */

/* ***** */
/* include files */
/* ***** */

#include <chimera.h>
#include <sbs.h>
#include <math.h>
#include "puma.h"

/* ***** */
/* choFwdKin Compute forward kinematics. */
/* ***** */

void choFwdKin(jtang, noap)
float jtang[6];
float noap[12];
{
    float a2, a3, d3, d4, d6;
    float c1, c2, c23, c3, c4, c5, c6, s1, s2, s23, s3, s4, s5, s6;

```



```

a2 = PUMA560_A2;
a3 = PUMA560_A3;
d3 = PUMA560_D3;
d4 = PUMA560_D4;
d6 = PUMA560_D6;

```

```

s1 = sin(jtang[0]);    c1 = cos(jtang[0]);
s2 = sin(jtang[1]);    c2 = cos(jtang[1]);
s3 = sin(jtang[2]);    c3 = cos(jtang[2]);
s4 = sin(jtang[3]);    c4 = cos(jtang[3]);
s5 = sin(jtang[4]);    c5 = cos(jtang[4]);
s6 = sin(jtang[5]);    c6 = cos(jtang[5]);
s23 = s2*c3 + c2*s3;   c23 = c2*c3 - s2*s3;

```

```

/* The following solution was obtained using macsyma. */

```

```

/* n vector */

```

```

noap[0] = (-c1 * c23 * s4 - s1 * c4) * s6 +
  ((c1 * c23 * c4 - s1 * s4) * c5 - c1 * s23 * s5) * c6;
noap[1] = (c1 * c4 - s1 * c23 * s4) * s6 +
  ((c1 * s4 + s1 * c23 * c4) * c5 - s1 * s23 * s5) * c6;
noap[2] = s23 * s4 * s6 + (-c23 * s5 - s23 * c4 * c5) * c6;

```

```

/* o vector */

```

```

noap[3] = (c1 * s23 * s5 + (s1 * s4 - c1 * c23 * c4) * c5) * s6 +
  (-c1 * c23 * s4 - s1 * c4) * c6;
noap[4] = (s1 * s23 * s5 + (-c1 * s4 - s1 * c23 * c4) * c5) * s6 +
  (c1 * c4 - s1 * c23 * s4) * c6;
noap[5] = (c23 * s5 + s23 * c4 * c5) * s6 + s23 * s4 * c6;

```

```

/* a vector */

```

```

noap[6] = (c1 * c23 * c4 - s1 * s4) * s5 + c1 * s23 * c5;
noap[7] = (c1 * s4 + s1 * c23 * c4) * s5 + s1 * s23 * c5;
noap[8] = c23 * c5 - s23 * c4 * s5;

```

```
/* p vector */

noap[9] = (noap[6]) * d6 + c1 * s23 * d4 - s1 * d3 +
  c1 * c23 * a3 + c1 * c2 * a2;
noap[10] = (noap[7]) * d6 + s1 * s23 * d4 + c1 * d3 +
  s1 * c23 * a3 + s1 * c2 * a2;
noap[11] = (noap[8]) * d6 + c23 * d4 - s23 * a3 - s2 * a2;

return;
}
```

### D.3 force.c and force.rmod

```
/* ***** */
/* */
/* force.c */
/* */
/* created by Hyunki Cho          02-23-96          */
/*          ( Republic of Korea, Army )          */
/* */
/* ----- */
/* */
/* This module specify the force control on fifth link */
/* */
/* State variable table: */
/*          INCONST:          none */
/*          OUTCONST:         none */
/* */
/*          INVAR:            F_MEZ - measured force */
/*                               Q_MEZ - measured joint values */
/*                               Q_TEMP2 - temporary reference */
/*                                   joint 2 */
/*                               Q_TEMP3 - temporary reference */
/*                                   joint 3 */
/*          OUTVAR:           Q_REF - refernce joint values */
/* */
/* Special notes: */
/*          This module is to compute the joint 5 increment */
/*          due to the force error between the desired and */
/*          measured forces. */
/* */
/* ***** */

/* ***** */
/* include files */
/* ***** */

#include <chimera.h>
#include <sbs.h>
#include <math.h>

/* ***** */
/* module 'Local_t' definition as required by Chimera */
```

```

/* ***** */

typedef struct {
    float *Qtemp2, *Qtemp3;
    float *Fref;
    float *Fmez;
    float *Qmez, *Qref;
    float f2pgain, tool;
    float Deadband;

} forceLocal_t;

/* ***** */
/* module initialization as required by Chimera */
/* ***** */

SBS_MODULE(force);

/* ***** */
/* forceInit Initialize the module. */
/* ***** */

int forceInit(cinfo, local, stask)
cfigInfo_t *cinfo;
forceLocal_t *local;
sbsTask_t *stask;
{
    sbsSvar_t *svar = &stask->svar;

    /* Get pointers to state variables. */

    local->Fmez = svarTranslateValue(svar->vartable, "F_MEZ",float);
    local->Qmez = svarTranslateValue(svar->vartable, "Q_MEZ",float);
    local->Qref = svarTranslateValue(svar->vartable, "Q_REF",float);
    local->Qtemp2 = svarTranslateValue(svar->vartable, "Q_TEMP2",float);
    local->Qtemp3 = svarTranslateValue(svar->vartable, "Q_TEMP3",float);

    /* One time initialization. */

    cfigCompulsory(cinfo, "F_REF", local->Fref, VT_FLOAT, 3);
    cfigCompulsory(cinfo, "F2P_GAIN", &local->f2pgain, VT_FLOAT, 1);
    cfigCompulsory(cinfo, "TOOL", &local->tool, VT_FLOAT, 1);
}

```

```

    cfigCompulsory(cinfo, "DEADBAND", &local->Deadband, VT_FLOAT, 1);

    return (int) local;
}

/* ***** */
/* forceReinit Re-Initialize the module. */
/* ***** */

int forceReinit(local, stask)
forceLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */
/* forceOn Start up the module.          */
/* ***** */

int forceOn(local, stask)
forceLocal_t *local;
sbsTask_t *stask;
{
    int i;
    float *fdes = local->Fref;

    printf("Explicite Force Control: ON\n");
    printf("Desired Contact Force: Fx = %f Fy = %f Fz = %f\n",
           fdes[0], fdes[1], fdes[2]);

    return I_OK;
}

/* ***** */
/* forceCycle Process module information. */
/* ***** */

int forceCycle(local, stask)
forceLocal_t *local;
sbsTask_t *stask;
{
    int i;

```

```

float *fref = local->Fref;
float *fmez = local->Fmez;
float *qmez = local->Qmez, *qref = local->Qref;
float *qtemp2 = local->Qtemp2;
float *qtemp3 = local->Qtemp3;
float force;
float joint, dq;
float static d6 = 0.05588;
float pos_f;
float deadband = local->Deadband;

/* assign current Qmez[4] to joint. */

joint = qmez[4];

/* determine the force error. */

force = fmez[1] - fref[1];

/* compute the last link movement due to the force error. */
/* using the force to position gain, the displacement of */
/* link 5 can be obtained easily. */

pos_f = force*local->f2pgain;

/* compute fifth joint increment. */

dq = asin(pos_f/(d6 + local->tool));

/* add dq to joint 5 to provide appropriate motor torque of */
/* joint 5. */

joint = joint + dq;

/* check the joint limit. in this case, the displacement of */
/* joint 5 is restricted intentionally to prevent lossing */
/* contact or unallowable motion of link 5. */

if (joint < -deadband) joint = -deadband;
if (joint > deadband) joint = deadband;

/* copy joint values to Qtemp. */

```

```

    qref[1] = qtemp2[0];
    qref[2] = qtemp3[0];
    qref[4] = joint;

    return I_OK;
}

/* ***** */
/* forceOff Stop the module. */
/* ***** */

int forceOff(local, stask)
forceLocal_t *local;
sbsTask_t *stask;
{
    kprintf("force: OFF\n");
    return I_OK;
}

/* ***** */
/* forceKill Clean up after the module. */
/* ***** */

int forceKill(local, stask)
forceLocal_t *local;
sbsTask_t *stask;
{
    kprintf("force: FINISHED\n");
    return I_OK;
}

/* ***** */
/* forceError Attempt automatic error recovery. */
/* ***** */

int forceError(local, stask, mptr, errmsg, errcode)
forceLocal_t *local;
sbsTask_t *stask;
errModule_t *mptr;
char *errmsg;
int errcode;

```

```

{
    /* Return after not correcting error. */

    return SBS_ERROR;
}

/* ***** */
/* forceClear Clear error state of the module. */
/* ***** */

int forceClear(local, stask, mptr, errmsg, errcode)
forceLocal_t *local;
sbsTask_t *stask;
errModule_t *mptr;
char *errmsg;
int errcode;
{
    /* Return after not clearing error. */

    sbsNewError(stask, "Clear not defined, still in error state",
                errcode);

    return SBS_ERROR;
}

/* ***** */
/* forceSet Set module parameters. */
/* ***** */

int forceSet(local, stask)
forceLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */
/* forceGet Get module parameters. */
/* ***** */

int forceGet(local, stask)
forceLocal_t *local;
sbsTask_t *stask;

```



```
{
    return I_OK;
}

/* ***** */
/* forceSync For modules which synchronize to */
/* something other than the clock. */
/* ***** */

int forceSync(local, stask)
forceLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}
```

```
#
# force.rmod
#
# Reconfigurable module file for the explicite force
# control of the puma 3 DOF planar robot at last link.
#
```

```
MODULE      force
DESC        controls puma in 3 DOF planar
INCONST     none
OUTCONST    none
INVAR       F_MEZ Q_MEZ Q_TEMP2 Q_TEMP3
OUTVAR      Q_REF
TASKTYPE    periodic
FREQ        500
```

```
LOCAL
F_REF 0 6 0
F2P_GAIN -0.001
```

```
TOOL 0.1905
DEADBAND 0.1745
```

```
EOF
```

#### D.4 *fine.c* and *fine.rmod*

```
/* ***** */
/*
/*   fine.c
/*
/*   created by      Hyunki Cho           03-08-96
/*                   Air Force Institute of Technology
/*
/* -----
/*
/*   State variable table:
/*       INCONST:      none
/*       OUTCONST:     none
/*       INVAR:        Q_MEZ - measured current joint
/*       OUTVAR:       Q_TEMP3 - temporary reference
/*                       joint 3
/*
/*   Special Note:
/*       This module is to compute the joint 3 increment
/*       to make the joint 5 centering action in its
/*       center of joint range. The controller takes the
/*       difference of position between joint 5 current
/*       position and its centering position.
/*
/* ***** */

/* ***** */
/* include files */
/* ***** */

#include <chimera.h>
#include <sbs.h>
#include <math.h>

/* ***** */
/* module 'Local_t' definition as required by Chimera */
/* ***** */

typedef struct {
    float *Qtemp3;
    float *Qmez;
    float Tool;
```

```

float Dq3_limit;

} fineLocal_t;

/* ***** */
/* module initialization as required by Chimera */
/* ***** */

SBS_MODULE(fine);

/* ***** */
/* fineInit Initialize the module. */
/* ***** */

int fineInit(cinfo, local, stask)
cfigInfo_t *cinfo;
fineLocal_t *local;
sbsTask_t *stask;
{
    sbsSvar_t *svar = &stask->svar;

    /* Get pointers to state variables. */

    local->Qtemp3 = svarTranslateValue(svar->vartable, "Q_TEMP3",float);
    local->Qmez = svarTranslateValue(svar->vartable, "Q_MEZ",float);

    /* One time initialization. */

    cfigCompulsory(cinfo, "TOOL", &local->Tool, VT_FLOAT, 1);
    cfigCompulsory(cinfo, "DQ3_LIMIT", &local->Dq3_limit, VT_FLOAT, 1);

    return (int) local;
}

/* ***** */
/* fineReinit Re-Initialize the module. */
/* ***** */

int fineReinit(local, stask)
fineLocal_t *local;
sbsTask_t *stask;
{

```

```

    return I_OK;
}

/* ***** */
/* fineOn Start up the module.          */
/* ***** */

int fineOn(local, stask)
fineLocal_t *local;
sbsTask_t *stask;
{
    kprintf("Last joint Actuator centered algorithm : On\n");
    return I_OK;
}

/* ***** */
/* fineCycle Process module information. */
/* ***** */

int fineCycle(local, stask)
fineLocal_t *local;
sbsTask_t *stask;
{
    int i;
    float *qtemp3 = local->Qtemp3;
    float *qmez = local->Qmez;
    float q3, q5, dq;
    float static d4 = 0.43307;
    float static d6 = 0.05588;
    float tool = local->Tool;
    float dq_limit = local->Dq3_limit;

    /* copy current joints values. */

    q3 = qmez[2];
    q5 = qmez[4];

    dq = asin(sin(q5)*(d6 + tool)/(d6+tool+d4));

    /* restrict the joint displacement at one sample period */
    /* to prevent unallowable motion of link 3 or make sure */
    /* to maintain contact with object. */

```

```

if (dq < -dq_limit) dq = -dq_limit;
if (dq > dq_limit) dq = dq_limit;

/* add to current joint 3 value to provide appropriate */
/* joint 3 motor torque. */

q3 = q3 + dq;

/* check the limit of joint 3 and joint 5. */

if (q3 < -0.785398) q3 = -0.785398;
if (q3 > 3.926990) q3 = 3.926990;

/* copy joint 3 value to Qtemp3 as a temporary value. */

qtemp3[0] = q3;

return I_OK;
}

/* ***** */
/* fineOff Stop the module. */
/* ***** */

int fineOff(local, stask)
fineLocal_t *local;
sbsTask_t *stask;
{
    kprintf("fine: OFF\n");
    return I_OK;
}

/* ***** */
/* fineKill Clean up after the module. */
/* ***** */

int fineKill(local, stask)
fineLocal_t *local;
sbsTask_t *stask;
{
    kprintf("fine: FINISHED\n");

```

```

    return I_OK;
}

/* ***** */
/* fineError Attempt automatic error recovery. */
/* ***** */

int fineError(local, stask, mptr, errmsg, errcode)
fineLocal_t *local;
sbsTask_t *stask;
errModule_t *mptr;
char *errmsg;
int errcode;
{
    /* Return after not correcting error. */

    return SBS_ERROR;
}

/* ***** */
/* fineClear Clear error state of the module. */
/* ***** */

int fineClear(local, stask, mptr, errmsg, errcode)
fineLocal_t *local;
sbsTask_t *stask;
errModule_t *mptr;
char *errmsg;
int errcode;
{
    /* Return after not clearing error. */

    sbsNewError(stask, "Clear not defined, still in error
                                state", errcode);

    return SBS_ERROR;
}

/* ***** */
/* fineSet Set module parameters.          */
/* ***** */

int fineSet(local, stask)

```

```

fineLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */
/* fineGet Get module parameters.          */
/* ***** */

int fineGet(local, stask)
fineLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */
/* fineSync For modules which synchronize to */
/* something other than the clock.          */
/* ***** */

int fineSync(local, stask)
fineLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

```



```
#
# fine.rmod
#
# Reconfigurable module file for the finger fine control of
# the puma 3 dof planar robot.
#

MODULE      fine
DESC        controls puma in 6 DOF
INCONST     NDOF DH
OUTCONST    none
INVAR       Q_MEZ
OUTVAR      Q_TEMP3
TASKTYPE    periodic
FREQ        50

LOCAL
TOOL 0.1905
#DQ3_LIMIT 0.00872
DQ3_LIMIT 0.01745

EOF
```

## D.5 gross.c and gross.rmod

```
/* ***** */
/*
/* gross.c
/*
/* created by Hyunki Cho 03-13-96
/* (Republic of Korea, Army)
/*
/* -----
/*
/* State variable table:
/* INCONST: none
/* OUTCONST: none
/* INVAR: Q_MEZ - measured current joint
/* OUTVAR: Q_TEMP2 - temporary reference
/* joint 2
/*
/* Special Note:
/* This module is to compute the joint 2 increment
/* to make the joint 3 leveling action to make the
/* proper performance of joint 5.
/*
/* ***** */

/* ***** */
/* include files */
/* ***** */

#include <chimera.h>
#include <sbs.h>
#include <math.h>

/* ***** */
/* module 'Local_t' definition as required by Chimera */
/* ***** */

typedef struct {
    float *Qtemp2;
    float *Qmez;
    float Dq2_limit;
} grossLocal_t;
```

```

/* ***** */
/* module initialization as required by Chimera */
/* ***** */

SBS_MODULE(gross);

/* ***** */
/* grossInit Initialize the module. */
/* ***** */

int grossInit(cinfo, local, stask)
cfigInfo_t *cinfo;
grossLocal_t *local;
sbsTask_t *stask;
{
    sbsSvar_t *svar = &stask->svar;

    /* Get pointers to state variables. */

    local->Qtemp2 = svarTranslateValue(svar->vartable, "Q_TEMP2",float);
    local->Qmez = svarTranslateValue(svar->vartable, "Q_MEZ",float);

    /* One time initialization. */

    cfigCompulsory(cinfo, "DQ2_LIMIT", &local->Dq2_limit, VT_FLOAT, 1);

    return (int) local;
}

/* ***** */
/* grossReinit Re-Initialize the module. */
/* ***** */

int grossReinit(local, stask)
grossLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */

```

```

/* gross0n Start up the module.          */
/* ***** */

int gross0n(local, stask)
grossLocal_t *local;
sbsTask_t *stask;
{

    kprintf("Gross Motion Control : On\n");

    return I_OK;
}

/* ***** */
/* grossCycle Process module information. */
/* ***** */

int grossCycle(local, stask)
grossLocal_t *local;
sbsTask_t *stask;
{
    int i;
    float *qtemp2 = local->Qtemp2;
    float *qmez = local->Qmez;
    float q2, q3, q_t, dq;
    float static a2 = 0.4317;
    float static d4 = 0.43307;
    float dq_limit = local->Dq2_limit;

    /* copy current joints values. */

    q2 = qmez[1];
    q3 = qmez[2];

    /* determine the displacement of joint 2 that joint 3 is      */
    /* centered its displacement range with respect to joint 2. */

    q_t = (q2+q3) - 1.5708;

    dq = d4*q_t/a2; /* assume that the joint increments between */
                   /* joint 2 and 3 are inverse relationship    */
                   /* between their link length. */

```

```

/* restrict the joint displacement at one sample period */
/* to prevent unallowable motion of link 2 or make sure */
/* to maintain contact with object. */

if (dq < -dq_limit) dq = -dq_limit;
if (dq > dq_limit) dq = dq_limit;

/* add to current joint 2 value to provide appropriate */
/* joint 2 motor torque. */

q2 = q2 + dq;

/* check the limit of joint 2. range of joint 2 is restricted */
/* intentionally to make desired motion (prevent unallowable */
/* or loosing contact. */

if (q2 > 0.1705) q2 = 0.1705;
if (q2 < -1.5708) q2 = -1.5708;

/* copy joint 2 value to Qtemp2 as a temporary value. */

qtemp2[0] = q2;

return I_OK;
}

/* ***** */
/* grossOff Stop the module.          */
/* ***** */

int grossOff(local, stask)
grossLocal_t *local;
sbsTask_t *stask;
{
    kprintf("gross: OFF\n");
    return I_OK;
}

/* ***** */
/* grossKill Clean up after the module.    */
/* ***** */

```

```

int grossKill(local, stask)
grossLocal_t *local;
sbsTask_t *stask;
{
    kprintf("gross: FINISHED\n");
    return I_OK;
}

/* ***** */
/* grossError Attempt automatic error recovery. */
/* ***** */

int grossError(local, stask, mptr, errmsg, errcode)
grossLocal_t *local;
sbsTask_t *stask;
errModule_t *mptr;
char *errmsg;
int errcode;
{
    /* Return after not correcting error. */

    return SBS_ERROR;
}
/* ***** */
/* grossClear Clear error state of the module. */
/* ***** */

int grossClear(local, stask, mptr, errmsg, errcode)
grossLocal_t *local;
sbsTask_t *stask;
errModule_t *mptr;
char *errmsg;
int errcode;
{
    /* Return after not clearing error. */

    sbsNewError(stask, "Clear not degrossd, still in error
                                                                    state", errcode);

    return SBS_ERROR;
}

```

```

/* ***** */
/* grossSet Set module parameters. */
/* ***** */

int grossSet(local, stask)
grossLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */
/* grossGet Get module parameters. */
/* ***** */

int grossGet(local, stask)
grossLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

/* ***** */
/* grossSync For modules which synchronize to */
/* something other than the clock. */
/* ***** */

int grossSync(local, stask)
grossLocal_t *local;
sbsTask_t *stask;
{
    return I_OK;
}

```

```
#
# gross.rmod
#
# Reconfigurable module file for the finger gross control of
# the puma 3 dof planar robot.
#
```

```
MODULE      gross
DESC        controls puma in 1 DOF
INCONST     none
OUTCONST    none
INVAR       Q_MEZ
OUTVAR      Q_TEMP2
TASKTYPE    periodic
FREQ        5
```

```
LOCAL
#DQ2_LIMIT 0.00436
DQ2_LIMIT 0.00436
```

```
EOF
```



*Vita*

Capt. Hyunki Cho was [REDACTED] on-  
[REDACTED]. He grew up in Chungjoo-Si, Chungchungbook-Do,  
and graduated from Chungjoo High School in February 1986. He entered the Korea  
Military Academy (KMA) in March 1986. He learned a lot of things required for  
the great ROK Army officer. In March 1990, he graduated from the KMA with a  
Bachelor of Science degree in Electrical Engineering, and was commissioned in the  
Republic of Korea Army. After the Officer's Basic Course in the Army Infantry  
School, he was assigned to 1st ROK Army Infantry Division as a platoon leader. In  
January 1991, he took the Officer's English Course for six months, and was assigned  
to United Nations Command Security Force, Joint Security Area (Panmoonjum)  
around Demilitarized zone as a platoon leader of Joint Security Force. In July 1992,  
he was assigned to Joint Security Force as a Deputy Commander for one year. In  
June 1994, he entered AFIT enrolled in the School of Engineering, Department of  
Electrical and Computer Engineering. In March 1992, he married Youjung Han.  
They now have one child, Geonlan, and his wife is pregnancy.

[REDACTED]  
[REDACTED]  
[REDACTED]

VITA-1

# REPORT DOCUMENTATION PAGE

Form Approved

OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE December 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE Development of Object-Based Teleoperator Control for Unstructured Applications		5. FUNDING NUMBERS	
6. AUTHOR(S) Hyunki Cho Captain, Republic of Korea Army		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GE/ENG/96D-01	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583		10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Captain Thomas Deeter Director, AFMC Robotics and Automation Center of Excellence (RACE), Robotics and Automation Branch SA-ALC/TIER Bldg 324 505 Perrin Rd. Kelly AFB, TX 78241-6435		11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public release; Distribution Unlimited		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) For multi-fingered end-effectors in unstructured applications, the main issues are control in the presence of uncertainties and providing grasp stability and object manipulability. The suggested concept in this thesis is object-based teleoperator control which provides an intuitive way to control the robot in terms of the grasped object and reduces the operator's conceptual constraints. The general control law is developed using a hierarchical control structure, i.e., human interface / gross motion control level in teleoperation control and fine motion control / object grasp stability in autonomous control. The gross motion control is required to provide the position / orientation of the Super Object (SO), and the sufficient grasping force to the fine motion control. Impedance control is applied to the gross motion control to respond to the environmental forces. The fine motion control consists of serially connecting the finger in position control and the Fingertip Actuation System (FAS) in force control. The FAS has a higher bandwidth response than does the finger actuation system and operates near the center of its joint range. The finger motion controller attempts not only to track the displacement of the FAS but also to provide an FAS centering action. Simulation experiments in both gross and fine motion control are performed. The integrated gross / fine motion control is implemented using the planar configuration of PUMA 560. The results show that the desired contact force can be maintained in the direction of FAS motion. The mathematical proof of system stability and the extension to spatial systems are required to complete the research.			
14. SUBJECT TERMS Object-Based Teleoperator Control, Structured / Unstructured Applications, Impedance Control, Serial Actuation System, Gross (Macro) Motion Control, Fine (Micro) Motion Control, Fingertip Actuation System (FAS)			15. NUMBER OF PAGES 208
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED			16. PRICE CODE
18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	