FORCE CONTROL AND COLLISION AVOIDANCE STRATEGIES FOR KINEMATICALLY REDUNDANT MANIPULATORS

Farshid Shadpey

A Thesis
in
The Department
of
Electrical & Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

JULY 1997

0-612-40320-3

ABSTRACT

Compliant and Force Control of Redundant Manipulators

Shadpey Farshid, Ph.D.
Concordia University, 1997

The problem of position control of non-redundant manipulators was addressed during the initial stages of development of robotics in the 70's. In the 80's, extension of robotic applications to new non-conventional areas, such as space, underwater, hazardous environments, and micro-robotics, brought new challenges for robotic researchers. Position control strategies failed in performing tasks that needed interaction with a robot's environment. On the other hand, non-redundant manipulators were unable to perform tasks that required dexterity comparable to that provided by the human arm. Also, imprecise dynamic modeling put severe restrictions on performance of control algorithms which were based on exact knowledge of dynamic parameters. These issues have therefore attracted a lot of attention in following three areas: force and compliant motion control, redundancy resolution, and adaptive control strategies. These areas have been addressed separately. However, there exists no unique frame work for an adaptive compliant motion control scheme for redundant manipulators which enjoys all the desirable characteristics of the methods that have been proposed for each individual area, e.g., the existing compliant motion control schemes are either not applicable to redundant manipulators or cannot take full advantage of the redundant degrees of freedom.

In this thesis, the existing schemes in each of these three areas are reviewed. Based on the results of this review, a new redundancy resolution scheme at the acceleration level is proposed. The feasibility of this scheme is studied using simulations on a 3-DOF planar arm. This scheme is then extended to the 3-D workspace of a 7-DOF redundant manipulator. The performance of the extended scheme with respect to static and moving object collision avoidance and also joint limit avoidance is studied using both simulations and hardware experiments on REDIESTRO (a REdundant, Dextrous, Isotropically Enhanced, Seven Turning-pair RObot constructed in the Center for Intelligent Machines at McGill University). Based on this redundancy resolution scheme, an Augmented Hybrid Impedance Control (AHIC) scheme is proposed. The AHIC scheme provides a unified frame work for combining compliant motion control, redundancy resolution, and adaptive control in a single methodology. The feasibility of the proposed AHIC scheme is studied by computer simulations and experiments on REDIESTRO.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Professor R.V. Patel, for his invaluable support, encouragement and guidance during the course of my studies. He introduced me to this research topic and his continuous help and support in various stages of the work largely contributed to the success of this work. I consider myself to be fortunate to have been working in association with a great human being.

I would also like to thank my friend and colleague Dr. Farzam Ranjbaran for designing REDIESTRO which has proven to be a valuable research arm and also for his help in clarifying various kinematic issues presented in this work. The work regrading the cylinder-cylinder collision avoidance has also profited from his ideas and helps.

I also wish to thank all the STEAR-5 team (Dr. Cluade Tessier, Dr. Alan Robins, and Bernard Langlois) which gave me the opportunity to work as part of a great team on a project which has contributed to the success of this work. I thank Iain Bryson for his support and helpful ideas in implementation of the real-time software.

Many thanks goes to my parents Ali, and Tahmineh for encouragement and moral support during different stages of my life. I should also thank my brother Farshad who was an inspiration for working towards higher goals.

Finally, I wish to thank my wife Lida for her great patience and continuous support during the course of last fourteen years and also my lovely son Rouzbeh for his great patience during my preoccupation with this work.

Dedicated to

My wife Lida,

and my son Rouzbeh

TABLE OF CONTENT

LIST OF FIGURES xiv

LIST OF TABLES xxi

LIST OF ABBREVIATIONS.................... xxiii

CHAPTER 1 INTRODUCTION.................... 1

1.1 Introduction.................... 1

1.2 Motivation and Objectives of the Thesis 3

1.3 Thesis Outline.................... 5

1.4 Contributions And Accomplishments.................... 8

1.5 Acknowledgment 11

CHAPTER 2 REDUNDANT MANIPULATORS; KINEMATIC ANALYSIS AND REDUNDANCY RESOLUTION 13

2.1 Introduction.................... 13

2.2 Kinematic Analysis of Redundant Manipulators 14

2.3 Redundancy Resolution 16

2.3.1 Redundancy Resolution at Velocity Level.................... 16

2.3.1.1 Exact Solution.................... 17

2.3.1.2 Approximate Solution.................... 19

2.3.1.3 Configuration Control.................... 22

2.3.1.4 Configuration Control (Alternatives for Additional Tasks) . . 23

2.3.2 Redundancy Resolution at the Acceleration Level 25

2.4 Analytic Expression of Additional Tasks 27

2.4.1 Joint Limit Avoidance (JLA) 27

2.4.1.1 Definition of the Terms and Feasibility Analysis 28

2.4.1.2 Description of the Algorithms: 31

2.4.1.3 Approach I: Using Inequality Constraints 31

2.4.1.4 Approach II: Optimization Constraint 32

2.4.1.5 Performance Evaluation and Comparison 34

2.4.2 Static and Moving Obstacle Collision Avoidance 36

2.4.2.1 Algorithm Description 37

2.4.3 Posture Optimization (Task Compatibility) 38

2.5 Conclusion 40

CHAPTER 3 PRIMITIVE-BASED COLLISION AVOIDANCE FOR A 7-DOF REDUNDANT MANIPULATOR 42

3.1 Introduction 42

3.2 Primitive-Based Collision Avoidance 45

3.2.1 Cylinder-Cylinder Collision Detection 46

3.2.1.1 Review of Line Geometry and Dual Vectors 48

3.2.2 Cylinder-Sphere Collision Detection 58

3.2.3 Sphere-Sphere Collision Detection 60

3.3 Kinematic Simulation FOR a 7-DOF Redundant Manipulator 62

3.3.1 Kinematics of REDIESTRO 62

3.3.2 Main Task Tracking 63

3.3.2.1 Position Tracking 63

3.3.2.2 Orientation Tracking 63

3.3.2.3 Simulation Results 64

3.3.3 Additional Tasks 69

3.3.3.1 Joint Limit Avoidance. 69

3.3.3.2 Static and Moving Obstacle Collision Avoidance 70

3.4 Experimental Evaluation using a 7-DOF Redundant Manipulator. 76

3.4.1 Hardware And Software Configurations 77

3.4.2 Hardware Demonstration 78

3.4.3 Case 1: Collision Avoidance with Static Spherical Objects 78

3.4.4 Case 2: Collision Avoidance with a Moving Spherical Object 80

3.4.5 Case 3: Passing Through a Triangular Opening 81

3.5 Conclusion 82

CHAPTER 4 CONTACT FORCE AND COMPLIANT MOTION CONTROL . . . 87

4.1 Introduction. 87

4.2 Literature Review 89

4.2.1 Constrained Motion Approach 89

4.2.2 Compliant Motion Control 94

4.3 New Schemes For Compliant And Force Control Of Redundant Manipulators . 99

4.3.1 Configuration Control at the Acceleration Level 101

4.3.2 Augmented Hybrid Impedance Control using Computed Torque Algorithm 102

4.3.2.1 Outer-loop design 103

4.3.2.2 Inner-loop 104

4.3.2.3 Simulation Results on 3-DOF Planar Arm 105

4.3.3 Augmented Hybrid Impedance Control with Self-Motion Stabilization 112

4.3.3.1 Outer-Loop Design 112

4.3.3.2 Inner-Loop Design 113

4.3.3.3 Simulation Results on a 3-DOF Planar Arm. 114

4.3.4 Adaptive Augmented Hybrid Impedance Control 117

4.3.4.1 Outer-Loop Design 118

4.3.4.2 Inner-Loop Design 118

4.3.4.3 Simulation Results on a 3-DOF Planar Arm. 122

4.4 Conclusion 125

CHAPTER 5 AUGMENTED HYBRID IMPEDANCE CONTROL FOR A 7-DOF REDUNDANT MANIPULATOR 127

5.1 Introduction. 127

5.2 Algorithm Extension. 128

5.2.1 Task Planner and Trajectory Generator (TG) 129

5.2.2 AHIC module 129

5.2.3 Redundancy Resolution (RR) module 131

5.2.4 Forward Kinematics 133

5.2.5 Linear Decoupling (Inverse Dynamics) Controller 134

5.3 Testing and Verification 135

5.4 Stability Analysis And Trade-Off Study. 142

5.4.1 Description of the simulation environment. 142

5.4.2 Description of the sources of performance degradation 144

5.4.2.1 Kinematic instability due to resolving redundancy at the acceleration level 145

5.4.2.2 Performance degradation due to the model-based part of the controller . . . 149

5.4.3 Modified AHIC Scheme . . . 153

5.5 Conclusions . . . 159

CHAPTER 6 HARDWARE EXPERIMENTS ON CONTACT FORCE AND COMPLIANT MOTION CONTROL . . . 161

6.1 introduction . . . 161

6.2 Hardware And Software Configuration . . . 163

6.3 Preparation and Conduct of Hardware Experiments . . . 164

6.3.1 Selection of Desired Impedance . . . 164

6.3.1.1 Stability Analysis . . . 165

6.3.1.2 Impedance-controlled Axis . . . 166

6.3.1.3 Force-controlled Axis: . . . 169

6.3.2 Selection of PD Gains . . . 175

6.3.3 Selection of Force-Filter . . . 176

6.3.4 Effect of Kinematic Errors (Robustness Issue) . . . 178

6.4 Numerical Results of Strawman Tasks . . . 181

6.4.1 Data Capture And Monitoring . . . 181

6.4.2 Strawman Task I (Surface Cleaning) . . . 183

6.4.3 Strawman Task II (Peg In The Hole) . . . 187

6.5 Conclusion . . . 197

CHAPTER 7 CONCLUSION AND FUTURE WORK . . . 199

REFERENCES . . . 208

APPENDIX A SOFTWARE DEVELOPMENT FOR THE AHIC SCHEME 217

A.1 Requirement Specification 217

A.2 Software Development process. 218

A.2.1 Optimization Process 220

A.2.2 Detail Design and Coding 223

A.2.2.1 Task Planner and TG 223

A.2.2.2 AHIC Module 230

A.2.2.3 Redundancy Resolution (RR) module 233

A.2.2.4 Forward Kinematics:. 234

A.2.2.5 Inverse Dynamics: 235

APPENDIX B KINEMATIC AND DYNAMIC PARAMETERS OF REDIESTRO 236

APPENDIX C TRAJECTORY GENERATION (SPECIAL CONSIDERATION FOR ORIENTATION) 240

APPENDIX D HARDWARE AND SOFTWARE CONFIGURATION USED IN COLLISION AVOIDANCE EXPERIMENTS 244

D.1 Hardware Configuration 244

D.2 Software Configuration. 245

D.2.1 Processes residing on DSP boards 247

D.2.2 Processes residing on the Sparc 1+. 247

APPENDIX E HARDWARE AND SOFTWARE CONFIGURATION USED IN FORCE CONTROL EXPERIMENTS 248

E.1 Hardware Specification and Interfacing 248

E.2 Software Preparation and Interfacing 253

E.2.1 Controller Software. 254

E.2.1.1 Using the SGI Workstation as the Real-Time Controller. . . 256

E.2.2 Process Communication Software 262

E.2.2.1 BIT3 Software Support for the GIO-VME Bus Adaptor . . . 263

E.2.2.2 GIO-VME Bus Adaptor Driver 264

E.2.2.3 Communication Protocol Between IRIX and VME Systems 266

E.2.3 Data Acquisition And I/O Control Software 270

E.2.3.1 An Overview of the Chimera III Real-Time Operating System. 270

E.2.3.2 Generic I/O Interface 272

E.2.3.3 Subsystem Interface (SBS) 276

E.2.3.4 VME System Software Description 281

LIST OF FIGURES

Figure 2.1 Geometric representation of null space and range of 16

Figure 2.2 Damped versus undamped least-square solution 21

Figure 2.3 Kinematic control loop for a redundant manipulator 26

Figure 2.4 Reachable workspace of a 2-DOF manipulator 29

Figure 2.5 Feasibility of different trajectories for a 2-DOF manipulator. 30

Figure 2.6 Reachable workspace and desired trajectory for a 3-DOF planar arm . . . 34

Figure 2.7 Simulation results for JLA using the inequality constraint approach. . . . 35

Figure 2.8 Simulation results for JLA using the optimization approach 35

Figure 2.9 Comparison between different JLA approaches 36

Figure 2.10 Critical distance calculation: Schematic and Algorithm. 38

Figure 2.11 a) Force ellipsoid, b) Force transfer ratio in direction u 40

Figure 3.1 Perspective view of REDIESTRO 45

Figure 3.2 Cylinder representation, basic notation. 47

Figure 3.3 (A-1) Body-Body collision (non-parallel and non-intersecting axes) . . . 51

Figure 3.4 (A-2) Base-Body Collision (non-parallel and non-intersecting axes) . . . 52

Figure 3.5 Near Parallel axes 53

Figure 3.6 (A-3) Base-Base Collision (non-parallel and non-intersecting axes). . . . 54

Figure 3.7 (B-2) Base-Body Collision (intersecting axes) 56

Figure 3.8 (B-3) Base-Base Collision (intersecting axes) 56

Figure 3.9 (C-3) Base-Base Collision (parallel axes). 58

Figure 3.10 Cylinder-Sphere Collision Detection 60

Figure 3.11 Sphere-Sphere Collision Detection. 61

Figure 3.12 Simulation results for position and orientation tracking. 66

Figure 3.13 Simulation results when both main sub-tasks are active 67

Figure 3.14 Comparison between the fixed and the time-varying singularity robustness factor 68

Figure 3.15 Simulation result for JLA in the 3-D workspace with 70

Figure 3.16 REDIESTRO with simplified primitives 71

Figure 3.17 Simulation results for MOCA with fixed weighting factor 73

Figure 3.18 MOCA simulation results for the time-varying weighting factor 75

Figure 3.19 General block diagram for the hardware demonstration. 77

Figure 3.20 Case 1 79

Figure 3.21 Case 2 81

Figure 3.22 Collision avoidance with static spherical objects 84

Figure 3.23 Collision Avoidance with moving spherical object. 85

Figure 3.24 Passing through a triangular opening 86

Figure 4.1 Schematic diagram of the hybrid position and force control 94

Figure 4.2 Apparent impedance of a manipulator in each degree of freedom...... 96

Figure 4.3 Inner-outer loop control strategy 97

Figure 4.4 Surface cleaning using impedance controller 98

Figure 4.5 Block diagram of the AHIC scheme 102

Figure 4.6 3-DOF planar manipulator used in the simulation 105

Figure 4.7 General block diagram of the AHIC scheme 106

Figure 4.8 Simulation results for the AHIC scheme with Joint Limit Avoidance . . 107

Figure 4.9 Static Obstacle Collision Avoidance 108

Figure 4.10 Moving Obstacle Collision Avoidance 109

Figure 4.11 Task compatibility simulation results 110

Figure 4.12 Force-Controlled Additional Task 111

Figure 4.13 Force-Controlled Additional Task 111

Figure 4.14 General block diagram of the AHIC scheme 115

Figure 4.15 Object avoidance without self-motion stabilization 116

Figure 4.16 Moving object avoidance with Self-motion stabilization 117

Figure 4.17 Adaptive AHIC: Arm configuration and joint values. 123

Figure 4.18 Adaptive AHIC with tool orientation control 124

Figure 4.19 Adaptive Hybrid Impedance Control: Effect of imprecise force measurement. 125

Figure 5.1 Simplified block diagram of the AHIC controller 128

Figure 5.2 Different frames involved in the hybrid task specification. 131

Figure 5.3 Block diagram of a Simulink module 136

Figure 5.4 Simulink kinematic simulation used to verify the integration of the controller. 138

Figure 5.5 Desired impedance 138

Figure 5.6 Simulink one-dimensional simulation of the desired impedance 139

Figure 5.7 Step response in position-controlled directions 140

Figure 5.8 Step response in the force-controlled direction 141

Figure 5.9 Block diagram of the simulation used in the control scheme selection. . 143

Figure 5.10 Simulation model of REDIESTRO 144

Figure 5.11 Simplified block diagram of the AHIC controller simulation. 145

Figure 5.12 Simplified block diagram of the simulation used in kinematic instability analysis 146

Figure 5.13 Simulation results with non-zero initial velocity 146

Figure 5.14 Simulation result for modified RR scheme 148

Figure 5.15 Comparison between different values of damping factors 149

Figure 5.16 Block diagram of the LDPD controller. 149

Figure 5.17 Simulation results for LDPD controller 150

Figure 5.18 LDPD controller with inexact friction compensation. 152

Figure 5.19 LDPD controller experimental results 153

Figure 5.20 AHIC controller without joint friction 154

Figure 5.21 AHIC simulation 155

Figure 5.22 Simplified block diagram of the modified AHIC controller. 156

Figure 5.23 Block diagram of the complete Simulink simulation of the modified AHIC scheme. 157

Figure 5.24 Comparison between the original AHIC with modified AHIC (without friction compensation).. 158

Figure 6.1 Final hardware configuration (force control experiments) 163

Figure 6.2 Stability region of the system represented by Equation (6.3.7) with seconds. 166

Figure 6.3 Position step response in an impedance-controlled direction 168

Figure 6.4 Test scenario for selecting the desired impedance in the force-controlled direction 171

Figure 6.5 Force tracking for the test case shown in Figure 6.4 with 172

Figure 6.6 Force tracking for the test case shown in Figure 6.4 with 173

Figure 6.7 Plot of magnitude versus frequency for the second-order filter in Equation (6.3.3). 175

Figure 6.8 Effect of the force filter cutoff frequency 177

Figure 6.9 Hardware experiment to illustrate the capability of the AHIC scheme . 180

Figure 6.10 Experimental results: Exerting 20N on the surface with a rigid plate, with 5 degrees of orientation error on the surface. 181

Figure 6.11 Graphical rendering of the surface-cleaning task using MRS 182

Figure 6.12 Perspective view of the hardware setup used in the demonstration of . . 183

Figure 6.13 Different segments of Strawman Task I (surface cleaning) 184

Figure 6.14 Strawman Task I: Captured data for joint 2 185

Figure 6.15 Force data captured for Strawman Task I. 186

Figure 6.16 Position errors for the surface-cleaning hardware demonstration 187

Figure 6.17 Perspective view of the hardware setup used in demonstration of Strawman Task II. 188

Figure 6.18 Different segments of Strawman Task II 189

Figure 6.19 Strawman Task II: a) Kinematic tolerances, b) Correct initiation of the insertion 190

Figure 6.20 Strawman Task II: Hole 1 insertion/removal; interaction force/torques 195

Figure 6.21 Strawman Task II: Hole 1 insertion/removal; position information. . . . 196

Figure A.1 Directory structure of the CFC software 219

Figure A.2 Input/output block diagram of code optimization module 222

Figure A.3 Top-level block diagram of the Task planner. 224

Figure A.4 Flow-chart for the “Task Planner & TG” module 225

Figure A.5 Flow chart for the function ReadTaskFileCfig() 226

Figure A.6 Flow-chart for the function TG. 227

Figure A.7 Flow-chart for the function TgFifthinfor() 228

Figure A.8 Flow-chart for the function TgFifth() 229

Figure A.9 Input/output block diagram of AHIC 230

Figure A.10 Block diagram of the AHIC controller module 231

Figure A.11 Flow-chart for the AHIC() function 232

Figure A.12 Input-output block diagram of the RR module 233

Figure A.13 Flow-charts of a) RR module, and b) RRacc() function 234

Figure A.14 Input-output block diagram of the forward kinematics function 235

Figure C.1 Block diagram of the open-loop simulation for orientation TG. 241

Figure D.1 Block diagram of the VME bus based controller (collision avoidance experiments). 245

Figure D.2 Software Configuration (collision avoidance experiments) 246

Figure E.1 Bench Marking for the modified AHIC algorithm on the SGI workstation (Indigo 2) 250

Figure E.2 Bench Marking for the modified AHIC algorithm on the SGI workstation (Indigo 2) using IRIX real-time programming (REACT) utilities.. 252

Figure E.3 Final hardware configuration 253

Figure E.4 Software configuration block diagram of the real-time controller 254

Figure E.5 Black box representation of the AHIC residing on the SGI 255

Figure E.6 Simplified block diagram of the real-time control software running on the SGI workstation 256

Figure E.7 Simplified flow-chart of the function initTimeCounter() 259

Figure E.8 Simplified flow-chart of the function incrTimeCounter() 259

Figure E.9 Flow-chart of the real-time kinematic simulation of the controller software on the SGI workstation 261

Figure E.10 Effect of free-runner timer reset in calculating the elapsed time.. 262

Figure E.11 Block diagram of the hardware and software modules involved in the GIO-VME bus communication. 263

Figure E.12 Configuration of the shared memory segment of the dual-port RAM . . 267

Figure E.13 Block diagram of the function sendVals(n,data) 268

Figure E.14 Block diagram of the function recVals(n,data). 269

Figure E.15 Block diagram of the function sendCommand(command,arg). 269

Figure E.16 Data Flow between different modules of the real-time controller. 272

Figure E.17 Internal structure of an SBS module in Chimera III. 280

Figure E.18 Block diagram of GIOio module during the ON state 283

Figure E.19 Function block diagram of the VMEHandler() function 284

Figure E.20 Block diagram of the function GIOioCycle(). 285

Figure E.21 Block diagram of the master task on the VME system. 287

LIST OF TABLES

Table 5-1 Desired values used in the AHIC simulation (z - axis) 154

Table 5-2 Summary of equations for new/modified modules 156

Table 5-3 Desired values used in the modified AHIC simulation (z - axis) 158

Table 6-4 Control parameter used for Strawman Task I 183

Table 6-5 Desired positions, orientations, and forces for Strawman Task II 191

Table 6-6 Control parameters used in Strawman Task II 191

Table 6-7 Numerical values of the desired impedances for Strawman Task II . . 192

Table A-1 Summary of simulation environments 220

Table A-2 Functions optimized by RDM symbolic optimization module 222

Table A-3 Pre-Programed Task File (PPTF) 223

Table B-1 HD parameters of REDIESTRO 236

Table B-2 Mass (Kg) 237

Table B-3 Center of gravity in local frame {i} 237

Table B-4 Link Inertia Tensor (Kg m^2) 237

Table B-5 Motor assembly parameters 238

Table B-6 Motor assembly interface specifications 239

Table E-1 Specification of the hardware components used in the original configuration 249

Table E-2 Time bench marking for different modules of the AHIC scheme on a 68030 processor 250

Table E-3 Selection of data transfer mode for the GIO-VME adaptor 265

Table E-4 Comparison of the performance of different data transfer methods . . . 266

Table E-5 GIO/VME communication protocol - supported service request types 268

Table E-6 List of the reconfigurable modules 281

LIST OF ABBREVIATIONS

AAHIC:	Adaptive Augmented Hybrid Impedance Control
ACT:	Augmented Cartesian Target trajectory
ACTA:	Augmented Cartesian Target Acceleration trajectory
AHIC:	Augmented Hybrid Impedance Control
CFC:	Contact Force Control
CIM:	Center for Intelligent Machines at McGill University
CRA:	Cartesian Reference Acceleration
CTA:	Cartesian Target Acceleration trajectory
DMA:	Direct Memory Access
DOF:	Degrees-of-freedom
ERC:	Error Reference Controller
FwdKin:	Forward Kinematics
HIC:	Hybrid Impedance Control
JLA:	Joint Limit Avoidance
JTA:	Joint Target Acceleration
LD:	Linearized Decoupling controller
LDPD:	Linearized-Decoupled PD controller
MOCA:	Moving Object Collision Avoidance

MRS:	Multi-Robot Simulation system
ORU:	On-Orbit Replaceable Unit
PIO:	Programmed I/O
PPTF:	Pre-Programed Task File
RDM:	Robot Dynamic Modeling software
REDIESTRO:	REdundant, Dextrous, Isotropically Enhanced, Seven Turning-pair RObot
RR:	Redundancy Resolution
RTOS:	Real-Time Operating System
RTPU:	Real-Time Processing Unit
SCA:	Self Collision Avoidance
SOCA:	Static Object Collision Avoidance
SOI:	Surface of Influence
SPDM:	Special-Purpose Dextrous Manipulator
SSRMS:	Space Station Remote Manipulator System
STEAR:	Strategic Technologies in Automation and Robotics
SVD:	Singular Value Decomposition
TG:	Trajectory Generator

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION

In the Oxford English Dictionary, a robot is defined as *"a machine that looks like a human being and performs various complex acts (as walking or talking) of a human being"*. This definition expresses an ideal goal of building the perfect *companion* to serve the human being. From a scientific point of view, researchers have set intermediate goals at different stages.

The first goal was to build automated machines capable of performing repeatable tasks. This goal was achieved between the 30's and the 60's and resulted in the *industrial revolution* and automated production lines. The costly operation of changing an automated machine for each change in the operation, created the need for designing more *versatile reprogramble* machines.

The second goal was to design a *multi-functional reprogramble* robot *manipulator*. Note that the word *robot* and *manipulator* are often used together or in the place of each other in the robotics literature. This reflects the initial mandate that robotics researchers had set for the application of robots in the 70's. The main application was to *manipulate* objects (“*payloads*”) in a well-arranged and known environment. The problem of position control of manipulators was addressed in the 70’s to develop control schemes capable of controlling a manipulator’s motion in its workspace.

In the 80’s extension of robotic applications to new non-conventional areas, such as space, underwater, hazardous environments, and micro-robotics brought new challenges for robotics researchers. The goal was to develop control schemes capable of controlling a robot in performing tasks that required: (1) interaction with its environment; (2) dexterity comparable to that provided by the human arm.

Position control strategies failed in performing tasks that needed interaction with a robot’s environment. Therefore, developing control strategies capable of regulating interaction forces with the environment became necessary. On the other hand, new applications required robots to work in *cluttered* and *time-varying* environments. While most non-redundant manipulators possess enough *degrees-of-freedom* (DOF) to perform their main task(s), it is known that their limited manipulability results in a reduction in the workspace due to mechanical limits on joint articulation and presence of obstacles in the workspace. This motivated researchers to study the role of kinematic redundancy. Redundant manipulators possess extra DOFs than those required to perform the main task(s). These additional DOFs can be used to fulfill user defined additional task(s) such as joint limit avoidance and object collision avoidance. Redundancy has been recognized as a characteristic of major importance for robots in space application. This fact is reflected in the design of the SSRMS, the Space Station Remote Manipulator System, which is a 7-DOF redundant arm, and also the SPDM [62], the Special-Purpose Dextrous Manipulator, which consists of two 7-DOF arms.

Finally, imprecise kinematic and dynamic modelling of a robot manipulator and its environment puts severe restrictions on the performance of control algorithms which are based on exact knowledge of the kinematic and dynamic parameters. This has brought the challenge of developing adaptive/robust control algorithms which enable a robot to perform its tasks without exact knowledge of such parameters.

1.2 MOTIVATION AND OBJECTIVES OF THE THESIS

As mentioned in the previous section, the new applications for robot manipulators in space, underwater, and hazardous material handling have led to considerable activity in the following research areas:

- Contact Force Control (CFC) and compliant motion control
- Redundant manipulators and Redundancy Resolution (RR)
- Adaptive and robust control

Position control strategies are inadequate for tasks involving interaction with a compliant environment. Therefore, defining control schemes for tasks which demand extensive contact with the environment (such as assembly, grinding, deburring and surface cleaning) has been the subject of significant research in the last decade. Different control schemes have been proposed: Stiffness control [21], hybrid position-force control [22], impedance control [25], Hybrid Impedance Control (HIC) [26], and robust HIC [27].

Recently, free motion control of kinematically redundant manipulators has been the subject of intensive research. The extra degrees of freedom have been used to satisfy different additional tasks such as obstacle avoidance [9],[10], mechanical joint limit avoidance, optimization of user-defined objective functions, and minimization of joint velocities and acceleration [8]. Redundancy has been recognized as a major characteristic in performing tasks that require dexterity comparable to that of a human arm, e.g., in space applications such as for the Special Purpose Dexterous Manipulator (SPDM) which is intended for use in the International Space Station "Alpha". However, the compliant motion control of redundant manipulators has not attained the maturity level of their non-redundant counterparts. There is little work that addresses the problem of redundancy resolution in a compliant motion control scheme. For instance, Gertz et al. [36], Walker [35] and Lin et al. [37] have used a generalized inertia-weighted inverse of the Jacobian to resolve redundancy in order to reduce impact forces. However, these schemes are single-purpose algorithms, and they cannot be used to satisfy additional criteria. An extended impedance control method is discussed in [38] and [39]; the former also includes an HIC scheme.

Adaptive/robust compliant control has also been addressed in recent years [47], [48], and [49]. However, there exists no unique framework for an adaptive/robust compliant motion control scheme for redundant manipulators which enjoys all the desirable characteristics of the methods proposed for each individual area, e.g., existing compliant motion control schemes are either not applicable to redundant manipulators or cannot take full advantage of the redundant degrees of freedom.

The main objective of this thesis is to address the three research areas identified above in the context of redundant manipulators. In this context, the existing schemes in each of the three areas are reviewed. Based on the results of this review, a new redundancy resolution scheme at the acceleration level is proposed. The feasibility of this scheme is first studied using simulations on a 3-DOF planar arm. This scheme is then extended to the 3-D workspace of a 7-DOF redundant manipulator. The performance of the extended scheme with respect to collision avoidance for static and moving objects and avoidance of joint limits are studied using both simulations and hardware experiments on REDIESTRO (a REdundant, Dextrous, Isotropically Enhanced, Seven Turning-pair RObot constructed in the Center for Intelligent Machines at McGill University). Based on this redundancy resolution scheme, an Augmented Hybrid Impedance Control (AHIC) scheme is proposed. The AHIC scheme provides a unified framework for combining compliant motion control, redundancy resolution and object avoidance, and adaptive control in a single methodology. The feasibility of the proposed AHIC scheme is studied by computer simulations and experiments on REDIESTRO.

In order to reduce the risk of damage to REDIESTRO during implementation of new algorithms, the following steps have been followed:

- Algorithm development
- Feasibility analysis on a simple redundant 3-DOF planar arm
- Extension of algorithms to the 3D workspace of REDIESTRO
- Stability and trade-off analysis using simulations on a realistic model of the arm and its hardware accessories
- Fine tuning of the control gains in the simulation

- Performing the hardware experiments

1.3 THESIS OUTLINE

CHAPTER 2: REDUNDANT MANIPULATORS; KINEMATIC ANALYSIS AND REDUNDANCY RESOLUTION

This chapter introduces the kinematic analysis of redundant manipulators. First, different redundancy resolution schemes are introduced and a comparison between them is performed. Next, the configuration control approach at the acceleration level is described. This forms the basis of the redundancy resolution scheme used in the AHIC strategy proposed in Chapter 4. Finally analytical expressions of different additional tasks that can be used by the redundancy resolution module are given and simulation results for a 3-DOF planar arm are presented.

CHAPTER 3: PRIMITIVE-BASED COLLISION AVOIDANCE FOR A 7-DOF REDUNDANT MANIPULATOR

This chapter describes the extension of the proposed algorithm for redundancy resolution to the 3D workspace of a 7-DOF manipulator. First, a new primitive-based collision avoidance scheme in 3D space is described. The main focus is on developing the distance calculations and collision detection between the primitives (cylinder and sphere) which are used to model the arm and its environment. Next, the performance of the proposed redundancy resolution scheme is evaluated by kinematic simulation of a 7-DOF arm (REDIESTRO). At this stage, fine tuning of different control variables is performed. The performance of the proposed scheme with respect to joint limit avoidance (JLA), and static and moving object collision avoidance (SOCA, MOCA) is evaluated experimentally using REDIESTRO.

Chapter 4: *Contact Force and Compliant Motion Control*

This chapter begins with a literature review of existing contact force and compliant motion control. Based on this review, a novel compliant and force control scheme Augmented Hybrid Impedance Control (AHIC), is presented. The feasibility of using AHIC to achieve position and force tracking as well as resolving redundancy to perform additional tasks such as JLA, SOCA, MOCA is evaluated by simulation on a 3-DOF planar arm. In addition to the kinematic additional tasks described in Chapter 3, the scheme is capable of incorporating dynamic additional tasks such as multiple-point force control and minimization of joint torques to achieve a desired interaction force with the environment.

Based on the problems encountered (e.g. uncontrolled self-motion and lack of robustness with respect to model uncertainties) during simulations using the AHIC scheme, two modified versions of the original AHIC scheme are proposed. The first scheme aims to achieve self-motion stabilization and also robustness to the manipulator's model uncertainty, while the second scheme introduces an adaptive version of the AHIC controller. The stability and convergence analysis for these two schemes are given in detail. Simulations on a 3-DOF planar arm are performed to evaluate their performance.

Chapter 5: Augmented Hybrid Impedance Control for a 7-DOF Redundant Manipulator

In this chapter the extension of the AHIC scheme to the 3D workspace of REDIESTRO is given. Different modules involved in the controller are described. Considering the complexity of the control scheme and in order to reduce the risk of damage to the robot when performing the hardware experiments, the following steps were followed:

- Algorithm extension
- Software development:
- Stability analysis and trade-off study:
- Algorithm modification and fine tuning of the control gains

The first step is to extend the algorithm developed in Chapter 4 for the 2D workspace of a 3-DOF planar arm to the 3D workspace of a 7-DOF arm. New issues such as orientation and torque control are considered. Considering the huge number of operations involved in the controller and the limited processing power available, the next step is to develop the control software which is optimized both at the algorithm and code levels.

At this stage, a stability analysis and a trade-off study are performed using a realistic model of the arm and its hardware accessories. Potential sources of problems are identified. These are categorized into two different groups: Kinematic instability due to resolving redundancy at the acceleration level, and lack of robustness with respect to the manipulator's dynamic parameters. These problems are successfully resolved by modification of the AHIC scheme.

Chapter 6: Hardware Experiments on Contact Force and Compliant Motion Control

The goal of this Chapter is to demonstrate and evaluate the feasibility and performance of the proposed scheme by hardware demonstrations using REDIESTRO. The first section describes the hardware of the arm (e.g. actuators, sensors, etc.), and the control hardware (VME based controller, IO interface, etc.). The second section introduces the different software modules involved in the operation, their role, and the communication between different platforms.

Before performing the final hardware demonstrations, a detailed stability analysis is given to provide guidelines in the selection of the desired impedances. A heuristic approach is presented which enables the user to systematically select the impedance parameters based on stability and tracking requirements.

At this stage different scenarios are considered and two strawman tasks - surface cleaning and peg-in-the-hole - are selected. The selection is based on the ability to evaluate force and position tracking and also robustness with respect to knowledge of the environment and kinematic errors. These strawman tasks are also similar to the tasks that will be performed by the SPDM in space. These tasks are window cleaning and On-Orbit Replaceable Unit (ORU) insertion and removal.

CHAPTER 7: CONCLUSION AND FUTURE WORK

Based on the proposed algorithms for force and compliant motion control of redundant manipulators, general conclusions are drawn concerning the achievements of this thesis. Future avenues for research in order to extend the current work are also suggested.

1.4 CONTRIBUTIONS AND ACCOMPLISHMENTS

As indicated in the previous sections, the objectives of this thesis are to "*propose a unified framework for combining compliant motion control, redundancy resolution, and adaptive control in a single methodology*" and to demonstrate "*the feasibility of the proposed scheme by computer simulations and experiments on REDIESTRO*". The following contributions and accomplishments can be identified as the result of the work done to meet the aforementioned objectives:

1- **A novel primitive based collision detection scheme.** This scheme is general, and provides realism, efficiency of computation, and economy in preserving the amount of free space that would otherwise be wasted. All possible cases of collisions have been considered. In particular, cylinder-cylinder collision avoidance, which represents a complex case for a collision detection scheme, has been formalized using the notions of dual vectors and dual angles.

2- **Implementation of a real-time collision avoidance system for a 7-DOF redundant manipulator.** Despite the geometrical complexity of REDIESTRO, the arm is entirely modelled by decomposition of the links and attached actuators into sub-links modelled by simple volume primitives. The performance of the system has been successfully demonstrated on real hardware, i.e. the REDIESTRO manipulator.

3- **Extension of the *configuration control* approach for redundancy resolution at acceleration level.** Dynamic control of redundant manipulators in task space, such as the case of compliant control, requires the computation of joint accelerations. Hence, redundancy resolution should be performed at the acceleration level. However, most of the redundancy resolution schemes at the acceleration level suffer from uncontrolled self-motion. The sources of this problem and their solutions will be presented.

4- **A new approach, *Augmented Hybrid Impedance Control (AHIC),* for contact force and compliant motion control of redundant manipulators.** This approach is different from similar schemes proposed for redundant manipulators from following points of view:

- Different additional tasks can be easily incorporated in the AHIC scheme without modifying the scheme and the control law.
- An additional task can be included in the force-controlled subspace of the augmented task. Therefore, it is possible to have a multiple-point force control scheme.
- Task priority and singularity robustness formulations of the AHIC scheme relaxes the restrictive assumption of having a non-singular augmented Jacobian.

5- **A new adaptive compliant motion and force control scheme for redundant manipulators.** The stability and convergence of the proposed scheme guarantee asymptotic convergence in both position and force controlled subspaces assuming precise force measurements, while the scheme ensures stability of the system in the piesence of bounded force measurement errors.

6- **A new robust compliant motion and force control scheme for redundant manipulators.** The scheme is based on the AHIC approach and incorporates the following modifications:

- A new formulation for redundancy resolution which achieves self-motion stabilization I.
- Adding an error reference controller in the inner loop achieves robustness with respect to model uncertainties.

7- **Implementation and hardware demonstration of the proposed compliant motion and force control scheme on REDIESTRO**. The hardware demonstration can be considered as a major contribution from following points of view:

- There are very few cases in robotics literature where experimental results for force and compliant motion control of a 7-DOF manipulator have been reported.
- Performing tasks such as peg-in-the hole requires very accurate positioning wherein, in turn, requires a very well-calibrated arm. Considering the fact that REDIESTRO has not been kinematically calibrated and is known to have significant kinematic uncertainty, the successful operation of the peg-in-the-hole strawman task by REDIESTRO demonstrates a high level of robustness of the proposed scheme.

8- **The Robot Dynamic Modeling software (RDM) software.** Considering the complexity and large amount of computation involved in force and compliant motion control for a 7-DOF redundant manipulator, the implementation of the real-time controller, from both hardware and software points of view, by itself presents a challenge. The Robot Dynamic Modeling software developed as a side product of the research reported in this thesis provides a novel approach to modelling, simulation, and real-time controller development for general applications in robotics.

These contributions have been partially reported in the following references: [71], [82], [63], [43], [40], [42], and [64].

1.5 ACKNOWLEDGMENT

Part of the work described in this thesis was performed under the Phase II[1] and III[2] contracts awarded by the Canadian Space Agency to Bombardier Inc. (Canadair Defence System Division) for the STEAR 5 (Strategic Technologies in Automation and Robotics) project, "Trajectory Planning and Obstacle Avoidance". This work was done at Concordia University under subcontracts awarded to Concordia University (Principal Investigator: Professor R. V. Patel) by Bombardier Inc. in the context of the STEAR 5 project. The following is a description of the work that was performed under these contracts:

- The work reported in Chapter 3 was performed in Phase II of STEAR 5 (with the exception of the Cylinder-Cylinder collision avoidance scheme)
- The development of the AHIC scheme in Chapter 4 was performed in Phase III of STEAR 5. However, the robust AHIC controller with self-motion stabilization and the adaptive AHIC were developed independently of the contract.

[1] *Contract 9F006-2-0107/01-SW with Department of Public Works and Government Services Canada.*

[2] *Contract 9F006-5-0219/01-SW with Department of Public Works and Government Services Canada*

- The work reported in Chapter 5 was mostly performed in Phase III of STEAR 5. However, the development of the modified AHIC reported in this chapter was inspired by the robust AHIC control scheme with self-motion stabilization reported in Chapter 4.
- The work reported in Chapter 6 (hardware experiments) was performed under the STEAR 5 Phase III contract.

CHAPTER 2

REDUNDANT MANIPULATORS; KINEMATIC ANALYSIS AND REDUNDANCY RESOLUTION

2.1 INTRODUCTION

Particular attention has been devoted to the study of redundant manipulators in the last decade. Redundancy has been recognized as a major characteristic in performing tasks that require dexterity comparable to that of the human arm, e.g., in space applications such as in the Special Purpose Dexterous Manipulator (SPDM) which is intended for use on International Space Station "Alpha". While most non-redundant manipulators possess enough degrees-of-freedom (DOF) to perform their main task(s), i.e., position and/or orientation tracking, it is known that their limited manipulability results in a reduction in the workspace due to mechanical limits on joint articulation and presence of obstacles in the workspace. This has motivated researchers to study the role of kinematic redundancy. Redundant manipulators possess extra DOFs than those required to perform the main task(s). These additional DOFs can be used to fulfill user defined additional task(s). The additional task(s) can be represented as kinematic functions. This not only includes the kinematic functions which reflect some desirable kinematic characteristics of the manipulator such as posture control [31], joint limiting [15], and obstacle avoidance [9], [10], but can also be extended to include dynamic measures of performance by defining kinematic functions as the configuration-dependent terms in the manipulator dynamic model, e.g., impact force [37], inertia control [13], etc.

In this chapter, first, an introduction to kinematic analysis of redundant manipulators is given. In the next section, we perform an up-to-date review of existing methods proposed for redundancy resolution. We also study the performance of different redundancy resolution schemes from the following points of view:

- Robustness with respect to algorithmic and kinematic singularity
- Flexibility with respect to incorporation of different additional tasks

Based on this study, the "configuration control" approach has been selected as the basis for resolving redundancy in the force and compliant motion control schemes proposed for redundant manipulators. We also introduce the choice of the additional tasks and their analytic representation. Simulation results on a 3-DOF planar manipulator are given.

2.2 KINEMATIC ANALYSIS OF REDUNDANT MANIPULATORS

Definition: A manipulator is said to be redundant when the dimension of the task space m is less than the dimension of the joint space n. Let us denote the position and orientation of the end-effector along the axes of interest in a fixed frame by the $(m \times 1)$ vector X, and the joint positions by the $(n \times 1)$ vector q. In the case of a redundant manipulator, $r = n - m$ $(r \geq 1)$ is the *degree-of-redundancy*. The forward kinematic function is define as

$$X = f(q) \tag{2.2.1}$$

The differential kinematics are defined by

$$\dot{X} = J_e \dot{q} \tag{2.2.2}$$

and

$$\ddot{X} = J_e\ddot{q} + \dot{J}_e\dot{q} \tag{2.2.3}$$

where J_e is the $(m \times n)$ Jacobian of the end-effector. For a redundant manipulator, equations (2.2.1), (2.2.2)and (2.2.3) represent under-determined systems of equation. J_e can be viewed as linear transformation mapping R^n into R^m: The vector $\dot{q} \in R^n$ is mapped into $\dot{X} \in R^m$. Two fundamental subspaces associated with a linear transformation are its null space and its range (Figure 2.1).

The null space, denoted $\aleph(J_e)$, is the subspace of R^n defined by

$$\aleph(J_e) = \{\dot{q} \in R^n | J_e\dot{q} = 0\} \tag{2.2.4}$$

The range denoted $\Re(J_e)$, is a subspace of R^n defined by

$$\Re(J_e) = \{J_e\dot{q} | \dot{q} \in R^n\} \tag{2.2.5}$$

Equation (2.2.4) underlies the mathematical concept of using redundant manipulators. For a redundant manipulator, the dimension of $\aleph(J_e)$ is equal to $(n - m')$, where m' is the rank of the matrix J_e. If J_e has full column rank, then the dimension of $\aleph(J_e)$ is equal to the degree-of -redundancy. The joint velocities belonging to $\aleph(J_e)$, referred to as internal joint motion and denoted by $\dot{q}_\aleph$, can be specified without affecting the task space velocities. Therefore, an infinite number of solutions exists for the inverse kinematics problem. This shows the major advantage of redundant manipulators. Additional constraints can be satisfied while executing the main task specified via positions and orientations of the end-effector. The additional constraints can be incorporated using two different approaches - global and local. Global approaches ([2], [3], and [4]) achieve optimal behavior along the whole trajectory which ensure superior performance than local methods. However, the computational burden of global algorithms makes them unsuitable for real-time sensor-based robot control applications. Hence, we will focus on the local approaches.

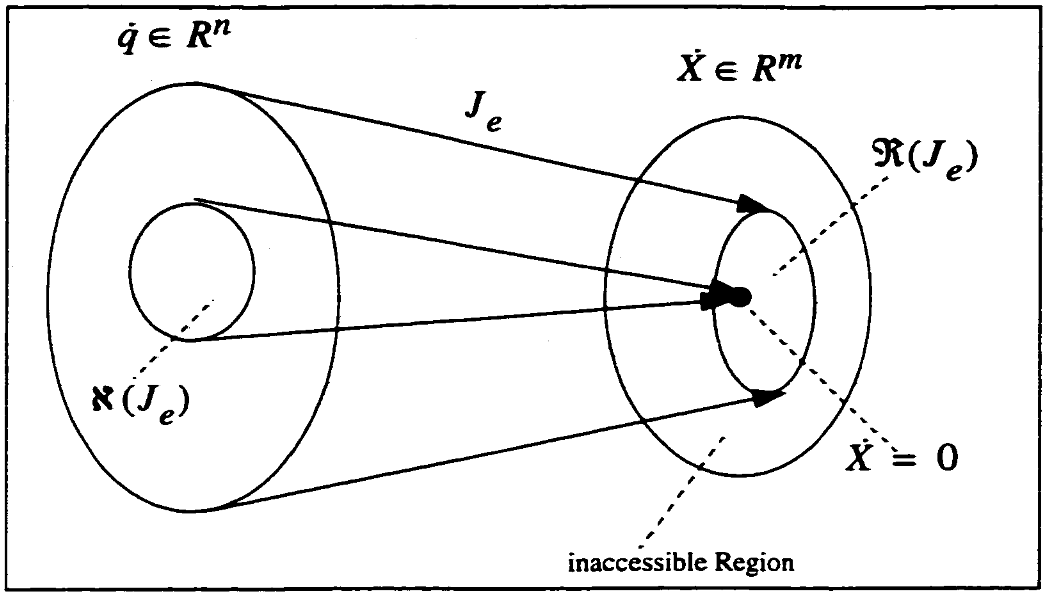


Figure 2.1 Geometric representation of null space and range of J_e

2.3 REDUNDANCY RESOLUTION

A cartesian controller generates commands expressed in Cartesian space. In the case of controlling a redundant manipulated this control inputs should be projected into joint space. Depending on the application requirements and choice of controller, redundancy can be resolved at position, velocity, and acceleration level. In most control scheme, the control input is expressed in form of a reference velocity or acceleration. Therefore, in this section we will focus on the redundancy resolution schemes proposed at velocity or acceleration levels.

2.3.1 Redundancy Resolution at Velocity Level

Solution of the inverse kinematic problem at the velocity level is of two types - exact and approximate.

2.3.1.1 Exact Solution

For a given $\dot{X}$, a solution $\dot{q}$ is selected which exactly satisfies (2.2.2). Most of the methods are based on the pseudo-inverse of the matrix J_e, denoted by $\bar{J}_e$:

$$\dot{q}_p = \bar{J}_e \dot{X} \qquad (2.3.1)$$

The pseudo inverse of J_e can be expressed as

$$\bar{J}_e = \sum_{i=1}^{m'} \frac{1}{\sigma_i} \hat{v}_i \hat{u}_i^T \qquad (2.3.2)$$

where the σ_i's, $\hat{v}_i$'s, and $\hat{u}_i$'s are obtained from the singular value decomposition of J_e [55] and the σ_i's are the non-zero singular values of J_e. Equation (2.3.1) represents the general form of a minimum 2-norm solution to the following least-squares problem:

$$min_{\dot{q}}\{\left\|J_e \dot{q} - \dot{X}\right\|\} \qquad (2.3.3)$$

If J_e has full row rank, then its pseudo inverse is given by:

$$\bar{J}_e = J_e^T (J_e J_e^T)^{-1} \qquad (2.3.4)$$

The ability of the pseudo-inverse to provide a meaningful solution in the least-squares sense regardless of whether Equation (2.2.2) is under-specified, square, or over-specified makes it the most attractive technique in redundancy resolution. However, there are major drawbacks associated with this solution. As pointed out in [1], the solution given by (2.3.1) does not guarantee generation of joint motions which avoid singular configurations - configuration in which J_e is no longer full rank. Near singular configurations, the norm of the solution obtained by (2.3.1) becomes very large. This can be seen from a mathematical point of view by (2.3.2), in which the minimum singular value approaches zero

($\sigma_{m'} \rightarrow 0$) as a singular configuration is approached, i.e., at a singular configuration, J_e becomes rank deficient. Therefore, as we can see in Figure 2.1, there are some velocities in task space which require large joint rates.

Another problem with the pseudo-inverse approach is that the joint motions generated by this approach do not preserve the repeatability and cyclicity condition, i.e., a closed path in joint space may not result in a closed path in joint space [5]. The final difficulty is that the extra degrees of freedom (when *dim(q)* > *dim(x)*) are not utilized to satisfy user-defined additional tasks. To overcome this problem, a term denoted $\dot{q}_{\aleph}$, belonging to the null space of J_e is added to the right hand side of Equation (2.3.1) [6].

$$\dot{q} = \dot{q}_p + \dot{q}_{\aleph} \tag{2.3.5}$$

Obviously $\dot{q}$ still satisfies (2.2.2). The term $\dot{q}_{\aleph}$ can be obtained by projection of an arbitrary n-dimensional vector ϑ to the null space of the Jacobian:

$$\dot{q}_{\aleph} = (I - \bar{J}_e J_e)\vartheta \tag{2.3.6}$$

where ϑ is selected as follows:

$$\vartheta = \nabla\Phi = \frac{\partial\Phi}{\partial q} = \begin{bmatrix} \frac{\partial\Phi}{\partial q_1} & \cdots & \frac{\partial\Phi}{\partial q_i} & \cdots & \frac{\partial\Phi}{\partial q_n} \end{bmatrix}^T \tag{2.3.7}$$

With this choice of the vector ϑ, the solution given by (2.3.5) acts as a gradient optimization method which converges to a local minimum of the cost function. The cost function can be selected to satisfy different objectives, such as torque and acceleration minimization [8], singularity avoidance [7], obstacle avoidance ([9], and [10]).

The other alternative is presented in the so called extended Jacobian methods [11], [12]. The Jacobian of the extended task is defined by:

$$J_E = \begin{pmatrix} J_e \\ J_c \end{pmatrix} \tag{2.3.8}$$

where J_E is the extended Jacobian matrix, J_e and J_c being the $(m \times n)$ and $(r \times n)$ Jacobian matrices of the main and additional tasks respectively. The differential kinematics of the extended task are given by:

$$\dot{Y}_{n \times 1} = \begin{pmatrix} \dot{X}_{m \times 1} \\ \dot{Z}_{r \times 1} \end{pmatrix} = J_E \dot{q} \qquad (2.3.9)$$

As a result of extending the kinematics at the velocity level, equation (2.3.9) is no longer redundant. Therefore, redundancy resolution is achieved by solving equation (2.3.9) for the joint velocities. However, there are two major drawbacks associated with this method [13]:

(i) The dimension of the additional task should be equal to the degree of redundancy which makes the approach not applicable for a wide class of additional tasks, such as those additional tasks that are not active for all time, e.g., obstacle avoidance in a cluttered environment.

(ii) The other problem is the occurrence of artificial singularities in addition to the main task kinematic singularities. The extended Jacobian J_E becomes rank deficient if either of the matrices J_e or J_c are singular, or there is a conflict between the main and additional tasks (which translates into linear dependence of the rows of J_e and J_c). In practical applications, the singularities of the end-effector are too complicated to determine a priori. Furthermore, the singularities of J_c are task dependent which makes them hard to determine analytically. Therefore, the solution of (2.3.9) based on the inverse of the extended Jacobian J_E, may result in instability near a singular configuration.

2.3.1.2 Approximate Solution

An alternative approach to dealing with the problem of artificial/kinematic singularities and large joint rates is to solve this problem for an approximate solution. The idea is to replace the exact solution of a linear equation, as in (2.2.2), with a solution which takes into account both the accuracy and the norm of the solution at the same time. This method which was originally referred to as the damped least-squares solution, has been used in different forms for redundancy resolution [14], [7]. The least-squares criterion for solving (2.2.2) is defined as follows:

$$\|J_e\dot{q}-\dot{X}\|^2+\lambda^2\|\dot{q}\|^2 \tag{2.3.10}$$

where λ, the damping or singularity robustness factor, is used to weigh the relative importance of the norms of joint rates and the tracking accuracy. This is equivalent to replacing the original equation (2.2.2) by a new augmented system of equations represented by:

$$\begin{pmatrix} J_e \\ \lambda I \end{pmatrix}\dot{q} = \begin{pmatrix} \dot{X} \\ 0 \end{pmatrix} \tag{2.3.11}$$

and finding the least-squares solution for the new system of equations (2.3.11) by solving the following consistent set of equations:

$$(J_e^TJ_e+\lambda^2I)\dot{q} = J_e^T\dot{X} \tag{2.3.12}$$

The least-squares solution is given by:

$$\dot{q}^{(\lambda)} = (J_e^TJ_e+\lambda^2I)^{-1}J_e^T\dot{X} \tag{2.3.13}$$

The practical significance of this solution is that it gives a unique solution which most closely approximates the desired task velocity among all possible joint velocities which do not exceed $\|\dot{q}^{(\lambda)}\|$.

The singular value decomposition (SVD) of the matrix in (2.3.13) is given by:

$$(J_e^TJ_e+\lambda^2)^{-1}J_e^T = \sum_{i=1}^{m'}\frac{\sigma_i}{\sigma_i^2+\lambda^2}\hat{v}_i\hat{u}_i^T \tag{2.3.14}$$

where σ_i's, $\hat{v}_i$'s, and $\hat{u}_i$'s are as in (2.3.2). By comparing the above SVD with that in (2.3.2), we notice a close relationship. Setting $\lambda = 0$, we obtain the pseudo inverse solution from (2.3.14). Moreover, if the singular values are much larger than the damping factor (which is likely to be true far from singularities), then there is little difference between the two solutions, since in this case:

$$\frac{\sigma_i}{\sigma_i^2 + \lambda^2} \approx \frac{1}{\sigma_i} \tag{2.3.15}$$

On the other hand, if the singular values are of the order of λ (or smaller), the damping factor in the denominator tends to reduce the potentially high norm joint rates. In all cases, the norm of joint rates will be bounded by:

$$\|\dot{q}_i^{(\lambda)}\| \le \frac{1}{2\lambda} \|\dot{X}_i\| \tag{2.3.16}$$

Figure 2.2 shows the comparison between solutions obtained by the two methods. As we can see the two problems associated with the pseudo inverse - discontinuity at singular configurations and large solution norms near singularities, are modified in the damped least-squares solution.

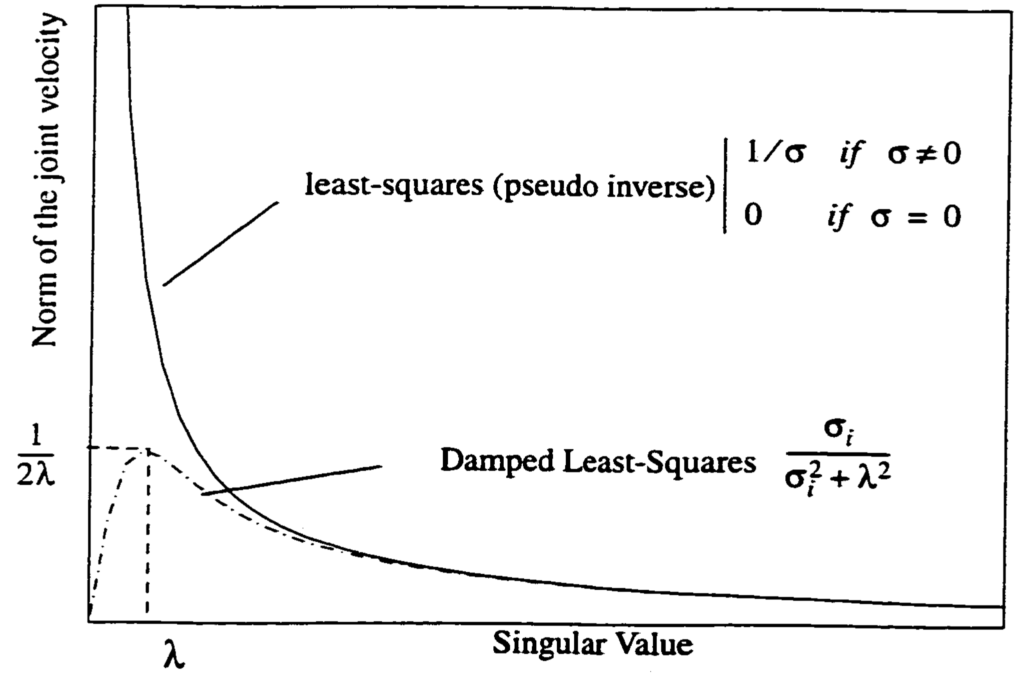


Figure 2.2 Damped versus undamped least-square solution

Based on this, Seraji [15], and Seraji and Colbaugh [16] proposed a general framework for redundancy resolution, referred to as *Configuration Control.*

2.3.1.3 Configuration Control

Under configuration control, the position vector X is augmented by the $(k \times 1)$ additional task vector Z, and the augmented $[(m+k) \times 1]$ task vector is defined by $Y^T = [X^T, Z^T]^T$. The differential augmented kinematics are defined by:

$$\dot{Y}_{(m+k)\times 1} = \begin{pmatrix} \dot{X}_{m\times 1} \\ \dot{Z}_{k\times 1} \end{pmatrix} = J_A \dot{q} \tag{2.3.17}$$

where

$$J_{A_{(m+k)\times n}} = \begin{pmatrix} J_e \\ J_c \end{pmatrix} \tag{2.3.18}$$

is the augmented Jacobian matrix, J_e and J_c being the $(m \times n)$ and $(k \times n)$ Jacobian matrices of the main and additional tasks respectively.

Seraji and Colbaugh [16] proposed a singularity robust and task prioritized formulation, denoted by configuration control, using the weighted damped least-squares method at the velocity level. The solution is given by:

$$\dot{q} = [J_e^T W_e J_e + J_c^T W_c J_c + W_v]^{-1} [J_e^T W_e \dot{X} + J_c^T W_c \dot{Z}] \tag{2.3.19}$$

which minimizes the following cost function:

$$\dot{E}_e^T W_e \dot{E}_e + \dot{E}_c^T W_c \dot{E}_c + \dot{q}^T W_v \dot{q} \tag{2.3.20}$$

where $W_e(m \times m)$, $W_c(k \times k)$ and $W_v(n \times n)$ are diagonal positive-definite weighting matrices that assign priority between the main, additional, and singularity robustness tasks. E_e and E_c are the n- and k-dimensional vectors representing the residual errors of the main and additional tasks respectively.

Note that in contrast to the extended formulation in (2.3.9), there is no restriction on the dimension(s) of the additional task(s). Therefore, the joint velocity (2.3.19) gives a special solution that minimizes the joint velocities when $k < r$, i.e., there are not as many active tasks as the degree-of-redundancy, and the best solution in the least-squares sense when $k > r$. In all cases the presence of W_v ensures the boundedness of joint velocities.

2.3.1.4 Configuration Control (Alternatives for Additional Tasks)

Configuration control can serve as a general framework for resolving redundancy. Any additional task represented as a kinematic function can be incorporated in this scheme [8]. This not only includes the kinematic functions which reflect some desirable kinematic characteristics of the manipulator such as posture control, joint limiting, and obstacle avoidance, but can also be extended to include dynamic measures of performance by defining kinematic functions as the configuration-dependent terms in the manipulator dynamic model, e.g., contact force, inertia control, etc. [13].

In this section, two general approaches for representing additional tasks are formulated:

(i) *Inequality constraints:* In many applications, the desired additional task is formulated as a set of inequality constraints $\rho(q) \geq C$, where ρ is a scalar kinematic function and C is a constant. A kinematic function is defined as:

$$Z = g(q) = \rho(q) - C \qquad and \qquad Z^d = \dot{Z}^d = \ddot{Z}^d = 0 \tag{2.3.21}$$

where the superscript d denotes the desired values. If $Z > 0$, this task is inactive.

(ii) *Kinematic optimization* of a cost function $\psi(q)$, can be incorporated in configuration control. Additional tasks can be formulated as the following constrained optimization problem: $minimum_q \ \psi(q)$ subject to $X - f(q) = 0$. The solution to this problem can be obtained using Lagrange multipliers. Let the augmented scalar objective function $\psi^{\circ}(q, \lambda)$ be defined as:

$$\psi^{\circ}(q, \lambda) = \psi(q) + \lambda^T(X - f(q)) \tag{2.3.22}$$

where λ is the $(m \times 1)$ vector of Lagrange multipliers. The necessary condition for optimiality can be written as:

$$\frac{\partial}{\partial q}\psi^{\circ} = 0 \Rightarrow \frac{\partial \psi}{\partial q} = \left(\frac{\partial f}{\partial q}\right)^T \lambda = J_e^T \lambda \tag{2.3.23}$$

$$\frac{\partial}{\partial \lambda}\psi^{\circ} = 0 \Rightarrow X = f(q) \tag{2.3.24}$$

Let N_e be full rank $(n \times r)$ matrix whose columns span the r-dimensional null space of the Jacobian J_e. The definition of the null space of J_e implies that

$$J_e N_e = 0_{m \times r} \tag{2.3.25}$$

Pre-multiplying both sides of (2.3.23) by N_e^T yields the optimality condition:

$$N_e^T \frac{\partial \psi}{\partial q} = 0 \tag{2.3.26}$$

Therefore, the additional task is represented as

$$Z = N_e^T \frac{\partial \psi}{\partial q} \qquad and \qquad Z^d = \dot{Z}^d = \ddot{Z}^d = 0 \tag{2.3.27}$$

The Jacobian of the additional task can be obtained by

$$J_c = \frac{\partial Z}{\partial q} = N_e^T \frac{\partial^2 \psi}{\partial q} + \left(\frac{\partial \psi}{\partial q}\right)^T \frac{\partial N_e}{\partial q} \tag{2.3.28}$$

2.3.2 Redundancy Resolution at the Acceleration Level

Dynamic control of redundant manipulators in task space, such as the case of compliant control, requires the computation of joint accelerations. Hence, redundancy resolution should be performed at the acceleration level. The second order differential kinematics are given in (2.2.3). We rewrite equation (2.2.3) as:

$$\ddot{X} - \dot{J}_e\dot{q} = J_e\ddot{q} \tag{2.3.29}$$

Following the procedure in Section 2.3.1, a similar formulation for $\ddot{q}$ can be obtained to yield exact and approximate solutions. The pseudo-inverse solution is given by:

$$\ddot{q}_P = \bar{J}_e(\ddot{X} - \dot{J}_e\dot{q}) \tag{2.3.30}$$

where $\bar{J}_e$ is the pseudo inverse of the Jacobian matrix. Equation (2.3.30) represents the general form of a minimum 2-norm solution to the following least-squares problem:

$$min_{\ddot{q}}\{\left\|J_e\ddot{q} - (\ddot{X} - \dot{J}_e\dot{q})\right\|\} \tag{2.3.31}$$

The solutions which are aimed at minimizing the norm of the joint acceleration vector have the shortcoming that they cannot control the joint velocities belonging to the null-space of the end-effector Jacobian or the augmented Jacobian. This may result in internal instability [17]. This problem can be attributed to the instability of the "zero dynamics" of (2.3.29) under a solution of the form (2.3.30) [18]. An example demonstrating this phenomenon is given in Section 4.3.3 .

In order to show the source of this problem more clearly, consider a simple kinematic control loop for Cartesian control of a redundant manipulator (Figure 2.3).

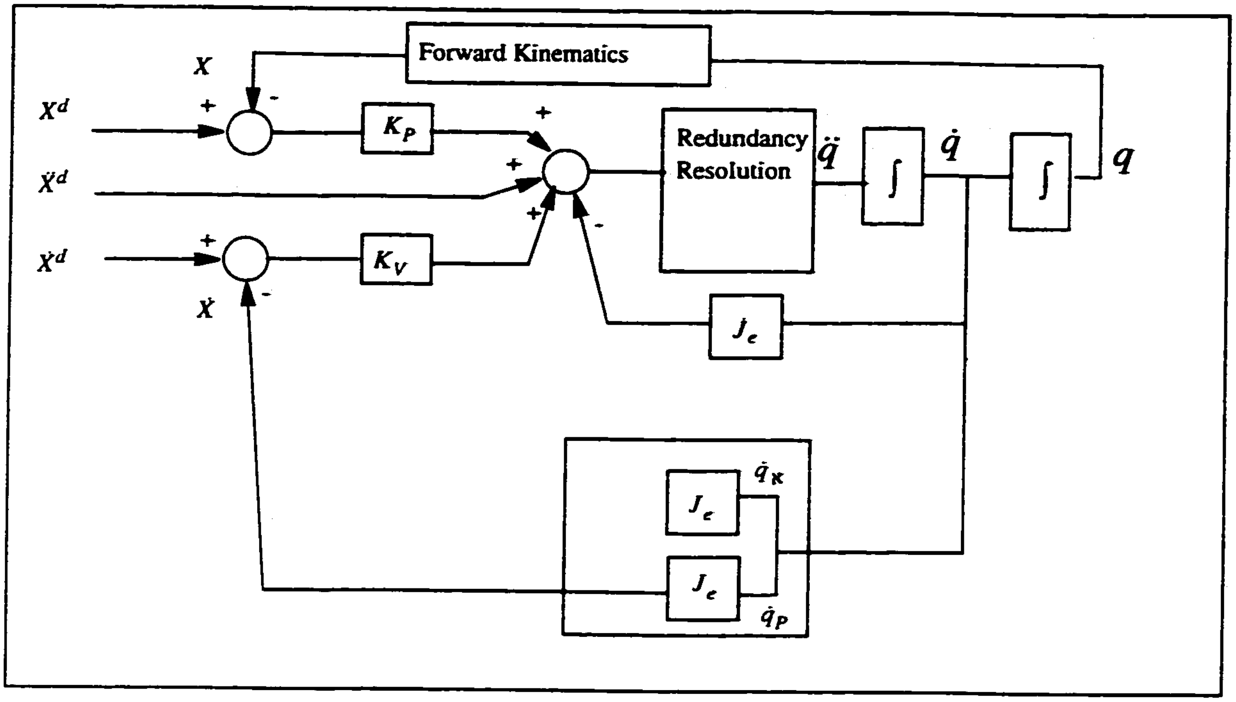


Figure 2.3 Kinematic control loop for a redundant manipulator

As we can see in Figure 2.3, the states of the system are q and $\dot{q}$. However, because of the nature of Cartesian control in which the desired trajectory is specified in task space, the feedbacks X and $\dot{X}$ are calculated by applying the nonlinear forward kinematic function to q, and the linear transformation mapping J_e to $\dot{q}$. Let us decompose $\dot{q}$ as follows:

$$\dot{q} = \dot{q}_P + \dot{q}_\aleph \qquad (2.3.32)$$

where

$$\dot{q}_\aleph \in \aleph(J_e)$$

$$\dot{q}_P \in \aleph^{\perp}(J_e) \qquad (2.3.33)$$

Using the definition of the null space, we can write:

$$\dot{X} = J_e\dot{q} = J_e\dot{q}_P + J_e\dot{q}_{\aleph} = J_e\dot{q}_P + 0 = J_e\dot{q}_P \tag{2.3.34}$$

This is equivalent to having an open-loop control for the null space component of $\dot{q}$. The question that may be asked is why the pseudo-inverse (or configuration control) at the velocity level does not exhibit this phenomenon. The reason is that, the pseudo-inverse solution at the velocity level given by (2.3.1) results in a minimum norm velocity solution. Therefore, it does not have any null space component. From a mathematical point of view, the pseudo-inverse of J_e is a projector matrix on to the $\aleph^{\perp}(J_e)$. However, the pseudo-inverse solution at the acceleration level results in a minimum norm acceleration solution which does not guarantee the elimination of the null space component of the velocity.

A solution to this problem was proposed by Hsu et al. [19]. This method requires the symbolic expression of the derivative of the pseudo-inverse of the Jacobian matrix which demands a large amount of computation. A method which combines both computational efficiency with stabilization of internal motion is proposed in Section 5.4.2.1 .

2.4 ANALYTIC EXPRESSION OF ADDITIONAL TASKS

The general strategies of defining additional tasks - inequality and optimization tasks, were explained in Section 2.3.1.4. In this section, the additional tasks most commonly encountered are formulated analytically under configuration control.

2.4.1 Joint Limit Avoidance (JLA)

Joint variables of actual mechanisms are obviously limited by mechanical constraints. In actual implementations, if some joint variables computed by the inverse kinematic module exceed their limits, these joints would be fixed at their extreme values which would restrict movement in certain directions in task space. In this section, we first introduce some relevant terminology, based on which a feasibility analysis of using kinematic redundancy resolution for joint limit avoidance will be presented. Then, we shall use two different approaches for defining algorithms which solve the problem of JLA. The performance of these algorithms will be analyzed by using computer simulations.

2.4.1.1 Definition of the Terms and Feasibility Analysis

The *reachable workspace* of a robot manipulator is defined by the geometrical locus of the position and orientation (pose) of the end-effector, $y \in \Re^m$, when the joint variables $q \in \Re^n$, $n \geq m$, range between two extreme values.

$$q_{imin} \leq q_i \leq q_{imax} \qquad\qquad i=1,2,\dots,n \tag{2.4.1}$$

The volume of the reachable workspace is finite, connected and, therefore, is entirely defined by its boundary surface. Obviously on this boundary, some loss of mobility occurs. Therefore the Jacobian matrix becomes rank deficient. The boundary of the reachable workspace can be found numerically by constrained optimization routines, or by applying an inverse kinematics algorithm [56]. As an example, in Figure 2.4, we show the reachable workspace of a two-link manipulator (using an optimization based approach).

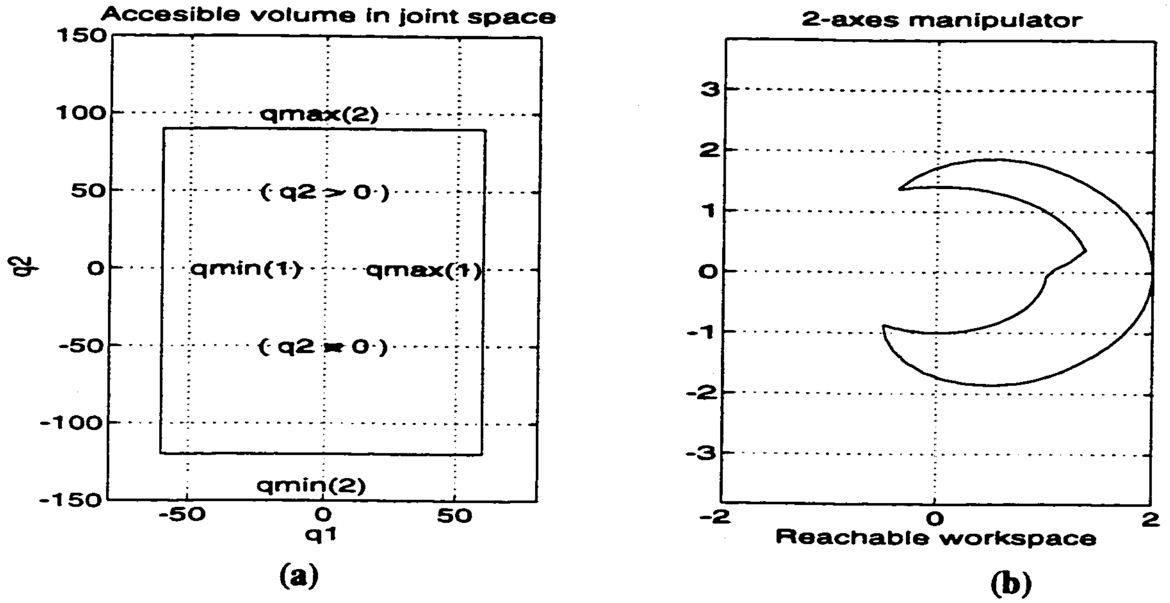


Figure 2.4 Reachable workspace of a 2-DOF manipulator in terms of a) Joint limits, b) Reachable workspace

Aspects [57] are the subspaces of the *accessible volume* in joint space in which the solution of the inverse kinematic function of Equation (2.1) is unique if $n=m$, or if n-m variables are fixed when $n>m$. The boundaries of the aspects are defined by the singularities of the Jacobian matrix J_e. Therefore, the interior of each aspect is free from singularities. Each aspect in joint space corresponds to a convex subspace of the reachable workspace. In Figure 2.4.a, we show the accessible volume in joint space and its corresponding image in task space (Figure 2.4.b).

From these plots, it is obvious that if the desired task trajectory lies inside two different aspects, the inverse kinematics of the manipulator fails to provide a continuous joint trajectory between the initial and the final points. Therefore, this trajectory is not practically realizable without re-configuration of the manipulator at the singular configuration.

In particular, it is easy to see that for the two-link planar manipulator, with joint limits indicated in Figure 2.4.a and the reachable workspace shown in Figure 2.4.b, we may encounter the following possibilities (Figure 2.5):

- The path AB (the first letter indicates the initial point) is not realizable.
- The path CE via the intermediate point D is not realizable.
- The same path CE via F is realizable.
- The path GH with initial joint position $q_2 > 0$ is not realizable.
- The same path GH by the initial configuration $q_2 < 0$ is realizable

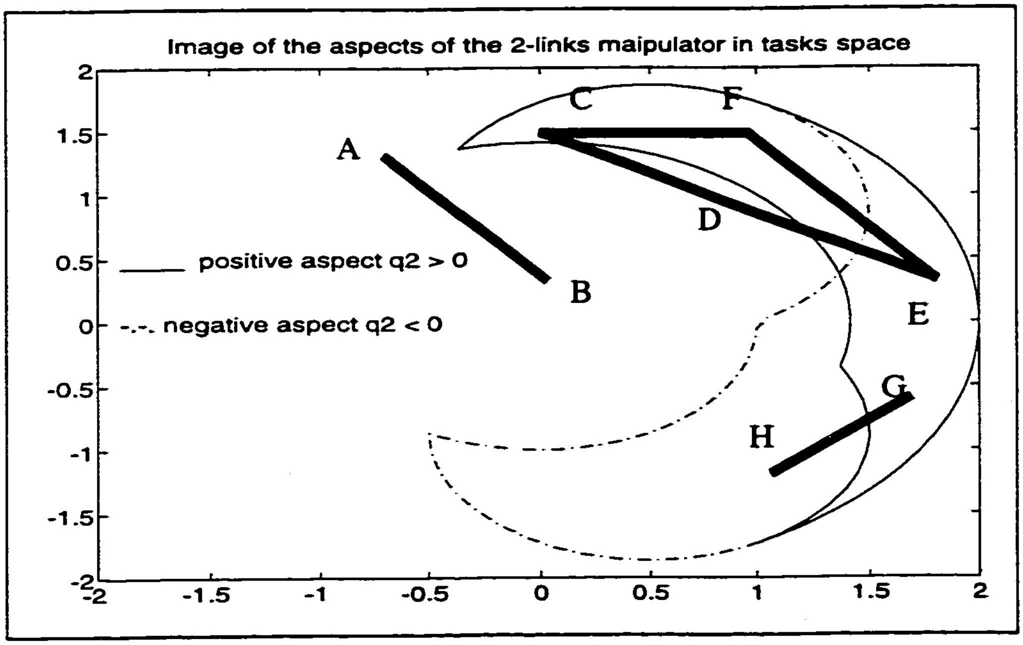


Figure 2.5 Feasibility of different trajectories for a 2-DOF manipulator

Note that by "unrealizable" we mean that there exists no continuous joint trajectory (that can be provided by the inverse kinematics) which starts from the initial configuration and satisfies the task trajectory without violating the joint limits. Thus, for realizing a task comprising motion from an initial pose to a final one, several problems may be considered, and the solutions for some of them may not be achievable by the redundancy resolution module. For instance, task AB is not realizable, but tasks CE and GH can be realized by means of a joint limit avoidance algorithm.

Although the analyzed example is concerned with a non-redundant manipulator, the main concepts are applicable to redundant manipulators under configuration control with the only difference being that in this case, the augmented task consists of the main and additional tasks which are usually not defined in the same coordinates. Therefore, the geometrical interpretation of the aspects and reachable workspace will, in general, be different in the case of redundant manipulators.

2.4.1.2 Description of the Algorithms:

Under the configuration control approach, the criterion of joint limit avoidance should be formulated as a kinematic constraint function. In the following, we present two different approaches for this formulation:

- Using inequality constraints which become active only when one or more of the limits are violated.

- Defining the secondary task as minimization of a desired cost function.

2.4.1.3 Approach I: Using Inequality Constraints

In this approach, the basic equations for the JLA algorithm are as follows. The joint limits are presented as a set of inequality constraints. If all the computed values of the joint variables satisfy the inequalities, the redundancy can be used for other tasks. However if one or more of these inequalities are violated, the JLA secondary task should be activated. This task is defined as follows:

$$Z_i = g_i(q) = q_i$$

$$Z^d{}_i = q_{mi} \qquad (2.4.2)$$

where q_m replaces either the maximum or the minimum values of the joints for i=*1,2,...,n*, and the corresponding constraint Jacobian J_c is defined by the equation:

$$J_{c_i} = \frac{\partial Z_i}{\partial q} = e_i^T \tag{2.4.3}$$

where e_i is the *ith* column of the identity matrix. For smooth incorporation of the inequality constraint into the inverse kinematics, it is desirable to define a "buffer" region where the relative importance of the JLA task progressively increases. To define this buffer, the following scheme is used [13]. When the inequality constraint is inactive, the corresponding weight W_{c_i} is zero, and on entering the "buffer" region increases gradually to its maximum value. Mathematically, we can formulate this weight selection procedure (i.e. $q_i \le q_{imax}$) as follows:

$$\begin{cases} W_{c_i} = 0 & \text{if } q_i \le q_{imax} - \tau \\ W_{c_i} = \frac{W_0}{4}\left[1 + \cos\left(\pi\left(\frac{q_{imax} - q_i}{\tau}\right)\right)\right] & \text{if } q_{imax} - \tau \le q_i \le q_{imax} \\ W_{c_i} = \frac{W_0}{2} & \text{if } q_i > q_{imax} \end{cases} \tag{2.4.4}$$

where W_0 and τ are user-defined constants representing the coefficient for the weight and width of the buffer region respectively.

2.4.1.4 Approach II: Optimization Constraint

The basic idea in the second approach is to define a kinematic objective function which is to be minimized. For joint limit avoidance the following function has been suggested:

$$\Phi(q) = \sum_{i=1}^{n}\left[\frac{q_i - q_{c_i}}{\Delta q_i}\right]^2 \tag{2.4.5}$$

where q_c is the center position around which we wish to minimize the movement and Δq is the difference between the maximum and the minimum values of the joints. Then, the redundancy resolution problem is to define a joint trajectory which optimizes equation (2.4.5) subject to the end-effector position.

Klein [58] mentioned that although the quadratic form of equation (2.4.5) is the most used function for this purpose, a better function which reflects the objective of joint limit avoidance has the form:

$$\Phi = max\frac{|q_i - q_{c_i}|}{\Delta q_i} = \left\|\frac{q - q_c}{\Delta q}\right\|_\infty \tag{2.4.6}$$

However, since the infinity norm is not a differentiable function, he proposed to use some finite order p-norm (p > 2):

$$\Phi = \left\|\frac{q - q_c}{\Delta q}\right\|_p \tag{2.4.7}$$

For most practical problems, p=6 gave good results. Note that in equation (2.4.7), the different joints have the same importance in the objective function. As an alternative to this formulation, we can introduce a diagonal weight matrix. The new objective function has the following form:

$$\Phi = \left\|K\left(\frac{q - q_c}{\Delta q}\right)\right\|_p \tag{2.4.8}$$

where K is an $n \times n$ diagonal matrix. The Jacobian and desired value for this additional task are calculated as mentioned in (2.3.27) and (2.3.28).

2.4.1.5 Performance Evaluation and Comparison

Based on these approaches, two algorithms were implemented. The simulations were carried out on a three-link planar manipulator with link lengths (0.75_m, 0.75_m, 0.5_m), *qmin*=[-90 -60 -75] degrees and *qmax*=[45 75 45]. The reachable workspace and the desired trajectory are shown in Figure 2.6.

1- Inequality constraint approach: Figure 2.7a shows the joint variables when the JLA provision was not activated. In this case, the third joint violates its minimum limit. In the second simulation, the JLA provision based on the first approach has been used with the nominal selected values W_0=100, W_v=5, W_e=10, and the buffer region τ=5 (degrees). Figure 2.7b shows that in this case, the third joint variable does not violate its limit. Note that by adjusting W_0, the discontinuity of the joint motion resulting from the nature of the inequality constraint formulation, can be controlled.

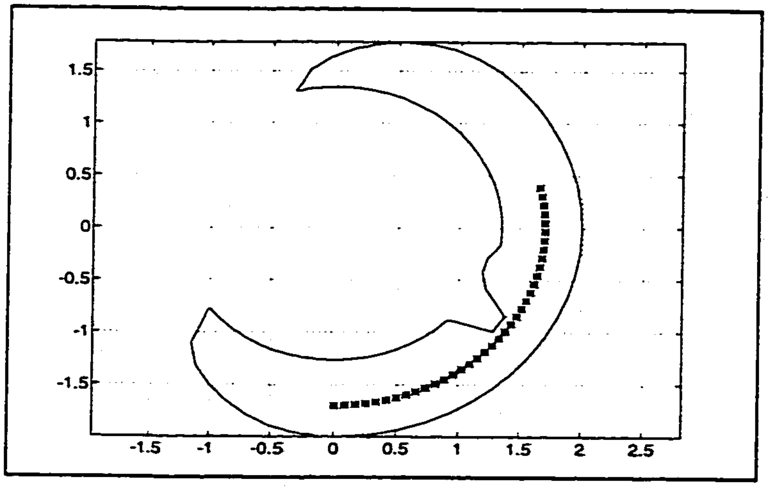
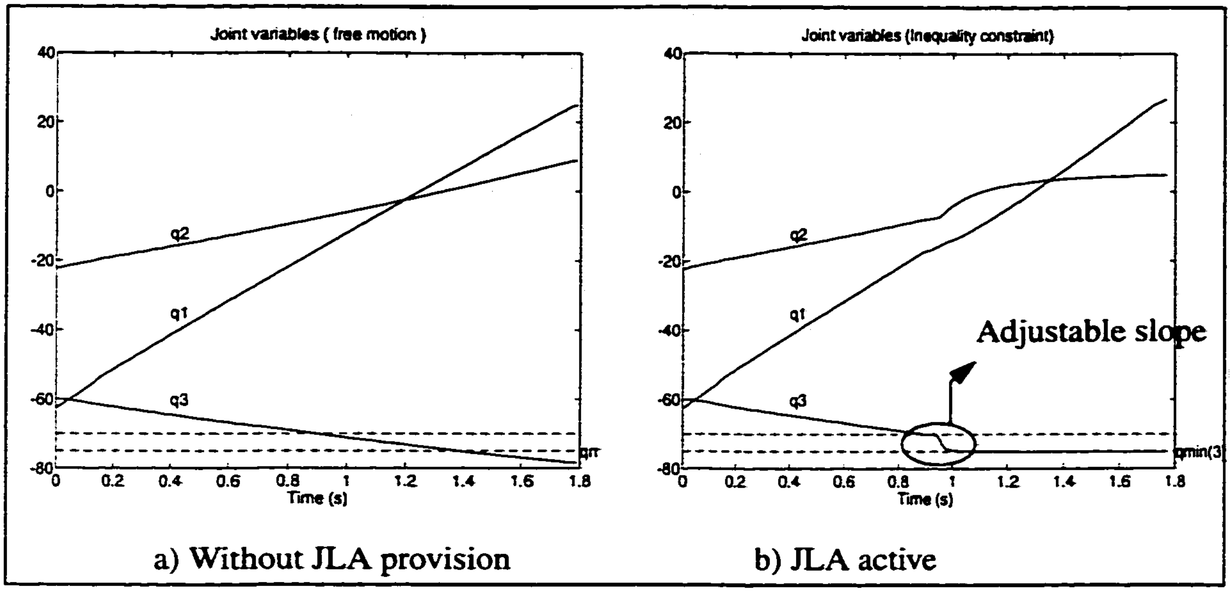


Figure 2.6 Reachable workspace and desired trajectory for a 3-DOF planar arm

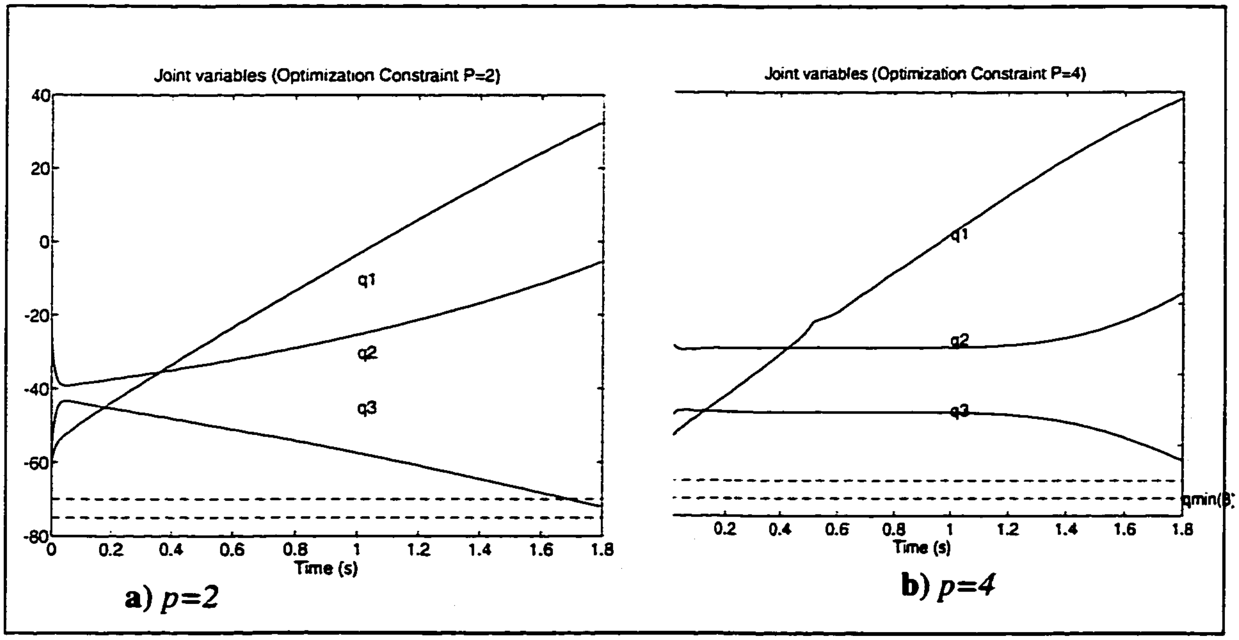


a) Without JLA provision

b) JLA active

Figure 2.7 Simulation results for JLA using the inequality constraint approach

2- Optimization approach: The following simulation used the optimization based JLA (*p*=*2*). Figure 2.8.a shows that the third joint variable enters the buffer region. Figure 2.8.b shows the results for *p*=*4*. As we can see, in this case all joints stay far from their limits.



a) *p*=*2*

b) *p*=*4*

Figure 2.8 Simulation results for JLA using the optimization approach

Figure 2.9 shows the value of the third joint variable for different approaches. As we can see, for this special case, both methods have been successful in following the desired trajectory while avoiding the joint limits. Obviously, the optimization method (p=4) has the best performance, since, the joint values are kept from approaching the limits. This in contrast to the inequality approach in which the joints move freely until coming close to the limits where the JLA becomes active and prevents from exceeding the limits. However, the optimization approach is computationally expensive (especially when the number of joints increases) compared to the simple formulation of the inequality constraint approach. Therefore, the inequality constraint approach is preferable for real-time implementations.

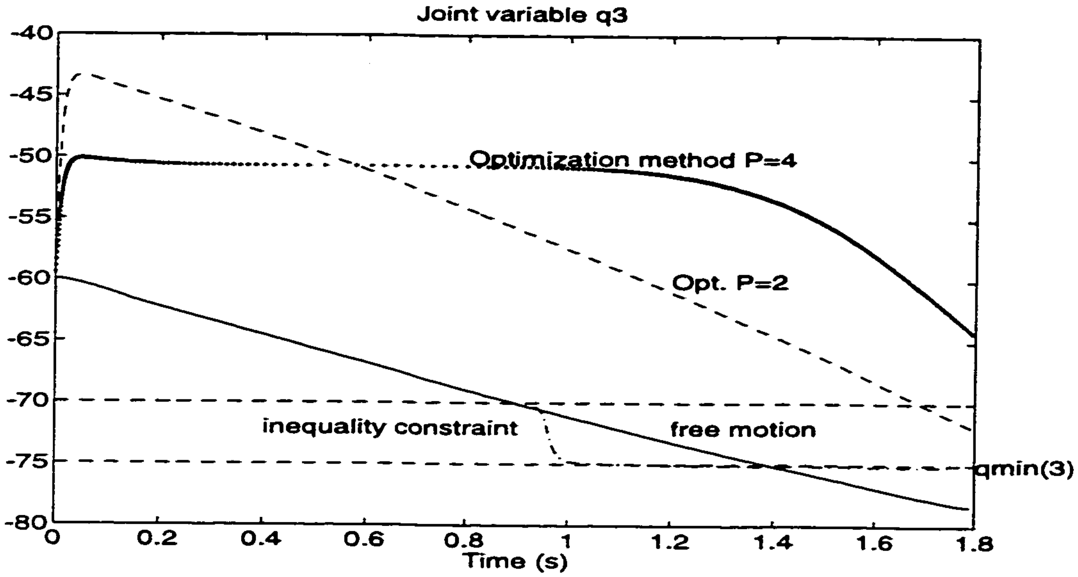


Figure 2.9 Comparison between different JLA approaches

2.4.2 Static and Moving Obstacle Collision Avoidance

In this section an outline of an algorithm for the 2-D workspace of a planar arm is given. The extension of the algorithm to 3-D workspace and simulation results are given in Chapter 3.

2.4.2.1 Algorithm Description

Similar to the JLA case, Static (and Moving) Obstacle Collision Avoidance is achieved using an inequality constraint. As in [9], the following steps are followed:

- Distance calculation
- Decision making (if there is a risk of collision for a link)
- Calculation of critical distance - the closest point on the link to the object.
- Utilizing redundancy to inhibit the motion of the critical point towards the object

For the 2-D workspace, links are modeled by straight lines and the objects are assumed to be circles. Each object is enclosed in a fictitious protection shield (represented by a circle) called the Surface of Influence (SOI). The first step involves distance calculation to find the location of the nearest point X_c (called the critical point) on each link to the obstacle by the procedure indicated in Figure 2.10. This algorithm is executed for each link and each obstacle. Then, if any of the critical distances d_{c_i} is less than the SOI, this constraint becomes active. In this case, we define the following kinematic function as the additional task:

$$z_i = g_i(q, t) = r_O - d_{c_i} \tag{2.4.9}$$

The derivative of the additional task is given below.

$$\dot{z}_i = \frac{\partial g_i}{\partial q}\dot{q} + \frac{\partial g_i}{\partial t} = -u_i^T\left(\frac{\partial X_{c_i}}{\partial q}\dot{q} - \dot{X}_o\right) \tag{2.4.10}$$

where $\dot{X}_0$ is the Cartesian velocity of the object. The desired values for the active constraints are:

$$z_i^d = \dot{z}_i^d = \ddot{z}_i^d = 0 \tag{2.4.11}$$

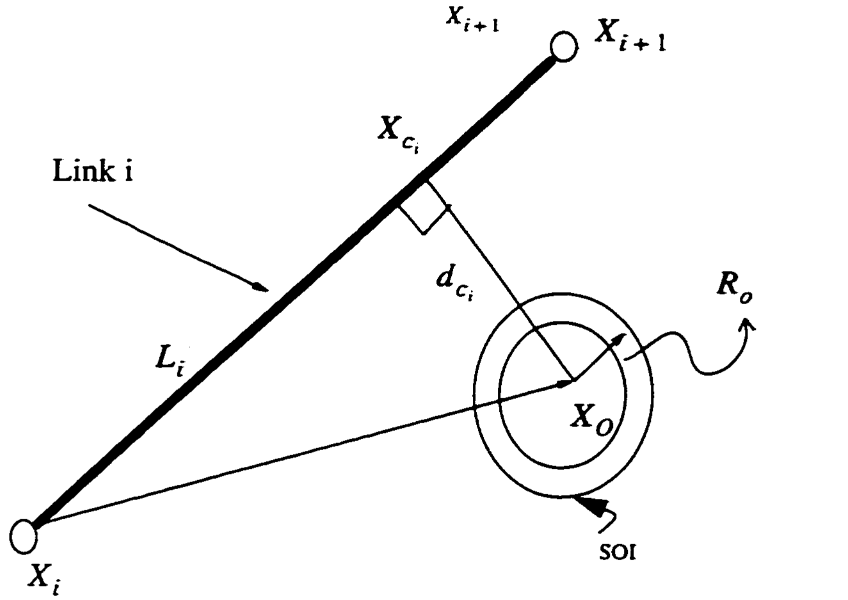
Note that we still need to calculate the Jacobian of the active constraints and its derivative. First, an intermediate term is defined as the Jacobian of the critical point, i.e.,

$$J_{X_{ci}} = \frac{\partial X_{c_i}}{\partial q} \qquad (2.4.12)$$

then, the Jacobian and its derivative are calculated as:

$$J_{c_i} = -u_i^T J_{X_{ci}} \qquad (2.4.13)$$

$$\dot{J}_{c_i} = \frac{\dot{z}_i}{d_{c_i}} u_i^T J_{X_{ci}} + \frac{1}{d_{c_i}} (\dot{X}_{c_i} - \dot{X}_o) J_{X_{ci}} + u_i^T \dot{J}_{X_{ci}} \qquad (2.4.14)$$



$$e_i = (X_{i+1} - X_i)/l_i$$

$$\alpha_i = e_i^T (X_o - X_i)$$

$$X_{c_i} = X_i + \alpha_i e_i$$

$$d_{c_i} = \|X_{c_i} - X_o\|$$

$$u_i = (X_{c_i} - X_o)/d_{c_i}$$

Figure 2.10 Critical distance calculation: Schematic and Algorithm

2.4.3 Posture Optimization (Task Compatibility)

Compliant motion control and force control are mainly needed for tasks involving heavy interaction with the environment. For this reason, an appealing additional task is to position the arm in a posture which requires minimum torque for a desired force in a certain direction. In this section, first, a kinematic index for measuring task compatibility is introduced. Then, in section 4.3.2 , it is incorporated as an additional task in the Augmented Hybrid Impedance Control (AHIC) scheme.

Similar to the manipulability ellipsoid introduced by Yoshikawa [30], a force ellipsoid can be defined by: $F_e^T(J_eJ_e^T)F_e$, where F_e is the environment reaction force. The optimal direction for exerting the force is along the major axis of the force ellipsoid which coincides with the eigen-vector of the matrix $J_eJ_e^T$ corresponding to its largest eigenvalue (Figure 2.11.a). The force transfer ratio along a certain direction is equal to the distance from the center to the surface of the force ellipsoid along this vector -- see Figure 2.11.b where u is the unit vector along the desired direction and α is the force transmission ratio along u. Since αu is a point on the surface of the ellipsoid, it should satisfy the following equation:

$$(\alpha u)^T(J_eJ_e^T)(\alpha u) = 1 \tag{2.4.15}$$

which gives $\alpha = [u^T(J_eJ_e^T)u]^{-1/2}$. Hence, Chiu [31] proposed to maximize the following kinematic function (task compatibility index)

$$\phi(q) = \alpha^2 \tag{2.4.16}$$

The desired value and the Jacobian for this additional task can be defined according to the procedure in Section 2.3.1.4 in this chapter. The simulation results are given in Section 4.3.2 .

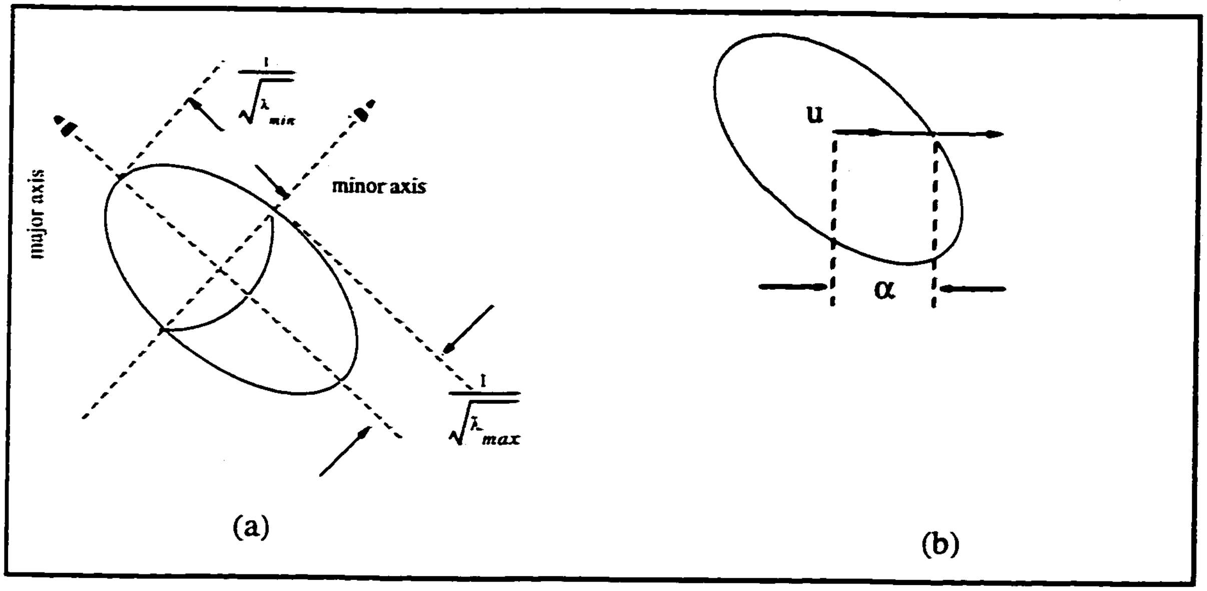

major axis
$\frac{1}{\sqrt{\lambda_{min}}}$
minor axis
$\frac{1}{\sqrt{\lambda_{max}}}$
(a)
u
α
(b)

Figure 2.11 a) Force ellipsoid, **b)** Force transfer ratio in direction *u*

2.5 CONCLUSION

In this chapter, the basic issues needed for the analysis of kinematically redundant manipulators were presented. Different redundancy resolution schemes were reviewed. Based on this review, *configuration control* at the acceleration level was found to be the most suitable approach to be used in a force and compliant motion control scheme for redundant manipulators. However, most of the redundancy resolution schemes at the acceleration level suffer from uncontrolled self-motion. In this section, the sources of this problem were presented. Their solutions will be presented in Chapters 4 and 5. The formulation of the additional tasks to be used by the redundancy resolution module were presented in this chapter. Joint limit avoidance which is one the most useful additional was studied in detail. The basic formulation of the static and moving obstacle collision avoidance task in 2D workspace was presented. We are now in the position to extend the proposed redundancy resolution scheme to the 3D workspace of REDIESTRO and evaluate the results by simulation and also experiments.

CHAPTER 3 PRIMITIVE-BASED COLLISION AVOIDANCE FOR A 7-DOF REDUNDANT MANIPULATOR

3.1 INTRODUCTION

Obstacle avoidance and collision detection are two of the main focuses of new control schemes for full or partly autonomous operation of a class of robot manipulators in cluttered environments. A compact and fast collision avoidance scheme is the major component in successful operation of robots in applications such as space, underwater, and hazardous environments. Collision avoidance can be divided into two categories: end-effector level and link level collision avoidance. Much of the work reported to date has dealt with obstacle avoidance as an off-line path planning problem, i.e., find a collision-free path for the end-effector [59], [60] or by mapping the obstacle into joint space, find a collision-free path in joint space [61]. These methods are not applicable to dynamic environments with moving objects. Moreover, for non-redundant manipulators, tracking an end-effector trajectory while avoiding collisions with the obstacles at the link level, or self-collision avoidance, is not always achievable. In recent years, kinematic redundancy has been recognized as a major characteristic for operation of a robot in a cluttered environment [62]. To implement a real-time collision avoidance scheme, three major areas: redundancy resolution, robot and environment modeling, and distance calculation should be investigated. Obviously, the accuracy level in which the arm and its environment are modeled is directly related to real-time control requirements. More detailed modeling results in more computation for calculating the critical distances between an obstacle and the manipulator. A solution to this problem is to use simple geometric primitive to represent the arm and environment. Colbaugh et al. [9] addressed this problem for a planar manipulator. The obstacles were represented by circles surrounded by a Surface of Influence (SOI), and the links were modeled by straight lines. A redundancy resolution scheme was proposed to achieve obstacle avoidance. Shadpey et al. [63][64], extended this method to a 3-D workspace of a 7-DOF manipulator. The manipulator links are represented by spheres and cylinders and the objects by spheres. Although this method is convenient for spherical or bulky objects, it results in major reduction of the workspace when dealing with long objects. Moreover, it is not capable of dealing with tasks such as passing through an opening. Glass et al. [66] proposed a new scheme for remote surface inspection. This application requires the robot to pass through circular or rectangular openings for inspection of a space structure, such as the International Space Station Alpha. However, they made the restrictive assumption of having an infinite surface with one opening which reduces the workspace of the robot. For instance, it does not permit an "elbow" to back into another opening. Moreover, the arm used in their experiment, Robotics Research Corporation 7-DOF arm (RRC), is modeled as a series of four straight lines connecting joints one, three, and five. The thickness of the links is considered via a "buffer" region in the openings. This simplified model of the arm would obviously fail when dealing with an arm with a more complex geometry such as REDIESTRO.

A simplified geometrical model for links of industrial manipulators with regard to the study of collisions either with each other or with objects in the workspace is the cylinder. Also, cylinder is a very suitable primitive for modelling many objects in the workspace such as rods, mesh structures, openings, etc., without losing much of the available workspace.

In section 2, we focus on the special cases of sphere-sphere, sphere-cylinder, and cylinder-cylinder collision detection and distance calculations. Considering the importance of cylinder-cylinder collision detection and also its complexity, a novel method of detecting collision between two cylinders using the notion of dual vectors and angles is presented.

REDIESTRO (Figure 3.1), an isotropic redundant research arm was selected to support the development of the collision avoidance system. Its special architecture, resulting from kinematic isotropic design objectives [65], represents a challenge for any collision avoidance system: There are joint offsets, bends in the links, and actuators that are large in relation to the size of the links. It is believed that a successful demonstration of the collision avoidance system on such an arm provides confidence that the system can be developed and applied to other more conventional (i.e., commercial) 7-DOF manipulator designs.

Section 3 presents the extension of the redundancy resolution module to the 3D workspace of REDIESTRO. It also describes the incorporation of different additional tasks into the redundancy resolution module. Simulation results to study the feasibility of the proposed scheme as well effects of different parameters are given. Section 4 presents the experimental evaluation of the collision avoidance schemes using REDIESTRO.

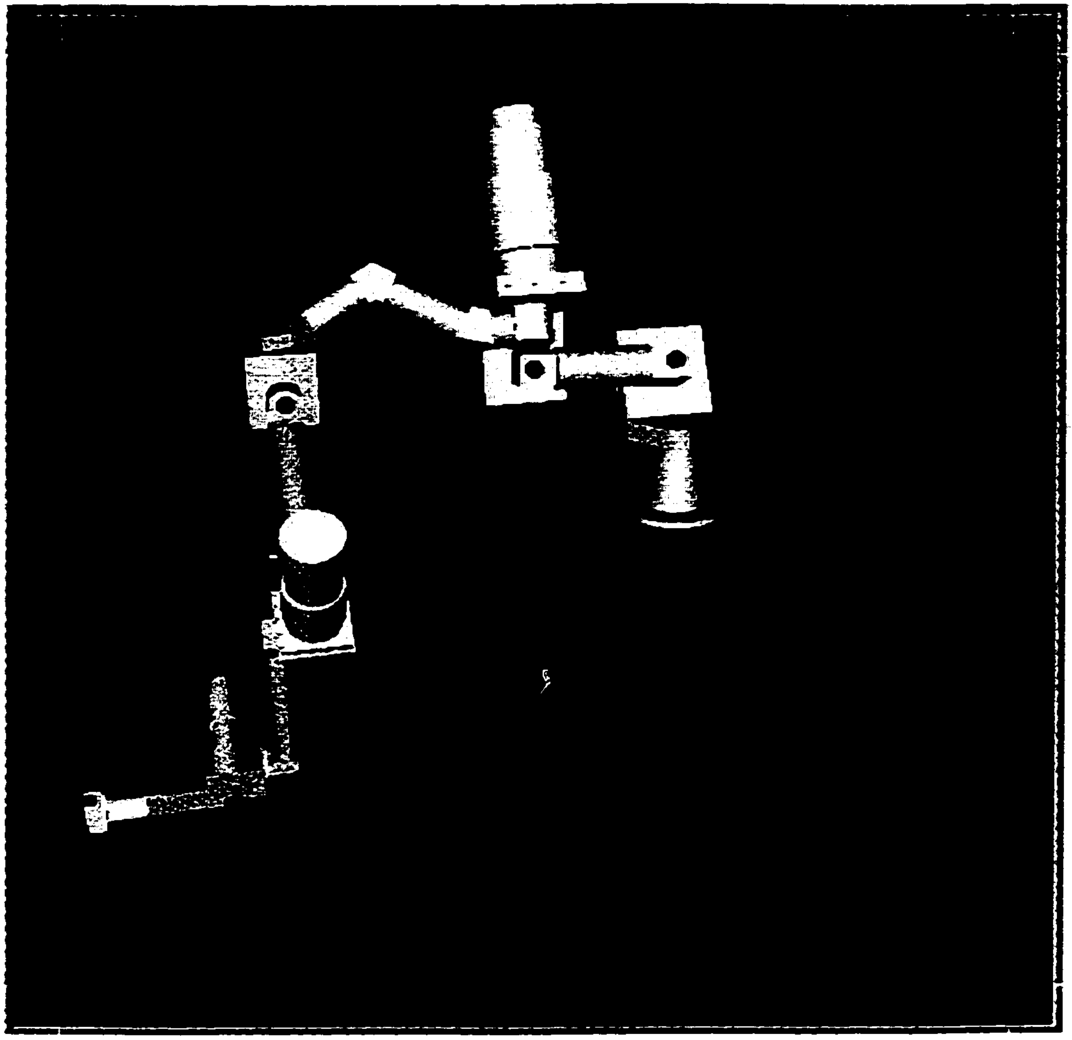


Figure 3.1 Perspective view of REDIESTRO

3.2 PRIMITIVE-BASED COLLISION AVOIDANCE

Collision avoidance for static and moving objects is achieved by introducing an inequality constraint (see Section 2.4.2) as the additional task. The idea is to model the links of the manipulator and the objects by primitives such as spheres and cylinders. The major components of the proposed scheme are outlined below:

- **Collision detection/prediction:** For those objects (sub-links) that can potentially collide, determine the critical distance h_{ij}, i.e., the distance from a critical point of the arm to that of the object. The critical points associated with the manipulator and the obstacles are denoted by P_i^c and P_j^c with position vectors p_i^c and p_j^c respectively.

- **Critical direction detection**: For any pair of critical points P_i^c and P_j^c, determine the critical direction denoted by u_{ij}, which is a unit vector along the vector joining P_i^c to P_j^c.

- Redundancy resolution: Formulate an additional task and use configuration control to inhibit the motion of the point P_i^c towards P_j^c along u_{ij}.

3.2.1 Cylinder-Cylinder Collision Detection

In order to determine the relative position of two cylinders, first the relative layout of their axes needs to be established. The axes of the cylinders being directed lines in three dimensional space, we resort to the notions of line geometry. Specifically, with the aid of dual unit vectors, (or line vectors), and the dual angles subtended by them, we will categorize the relative placement of cylinders and thus determine the possibility and the nature of collisions between the two cylinders in question.

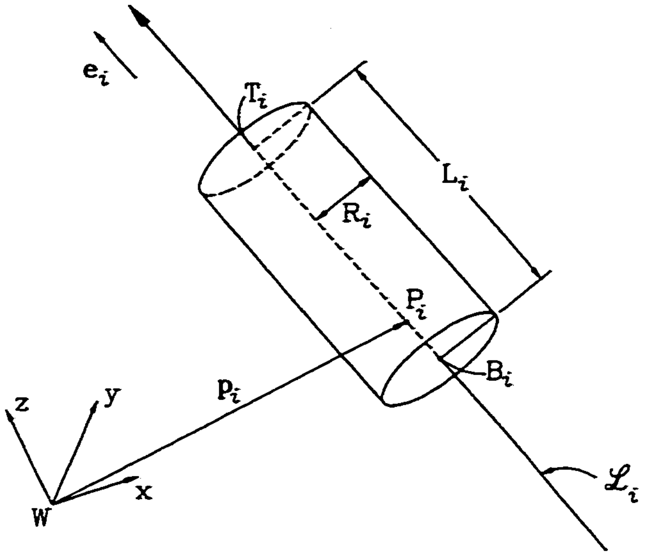


Figure 3.2 Cylinder representation, basic notation.

We consider each cylinder to be composed of three parts, the cylindrical surface plus the two circular disks as the top and the bottom of the cylinder. Four points along the axis $\mathcal{L}_i$ of each cylinder C_i are of interest, namely, P_i, B_i, T_i, and H_i. The point P_i is any point of reference along the line. The points B_i and T_i with position vectors b_i and t_i respectively, are the centers of the bottom and top of the cylinder, and H_i is the foot of the common normal of the two lines $\mathcal{L}_i$ and $\mathcal{L}_j$ on the $\mathcal{L}_i$. To avoid ambiguity for the choice of the top and bottom of the cylinder, we can always choose B_i and T_i in such away that the vector $\overrightarrow{B_iT_i}$ points along e_i, with e_i, being a unit vector defining the direction of the cylinder axis (see Figure 3.2). Each of B_i, T_i, and H_i, can alternatively be defined through their line coordinates with respect to the reference point P_i, namely,

$$b_i = p_i + \mathrm{b}_i e_i \tag{3.2.1}$$

$$t_i = p_i + \mathrm{t}_i e_i \tag{3.2.2}$$

$$h_i = p_i + \mathrm{h}_i e_i \tag{3.2.3}$$

It should be noted that for a given cylinder C_i, the scalars b_i and t_i are known and fixed values.

3.2.1.1 Review of Line Geometry and Dual Vectors

A brief review of dual numbers, vectors, and their operations, relevant to our problem is provided in this section. A more detailed discussion can be found in [67], and [68]. A line $\mathcal{L}$ can be defined via the use of a dual unit vector also called a line vector:

$$\hat{e} = e + \varepsilon m \tag{3.2.4}$$

where $e^T e = 1$, and $e^T m = 0$, and ε the dual unity has the property that $\varepsilon^2 = 0$. Here, e defines the direction of $\mathcal{L}$, while m the moment of $\mathcal{L}$ with respect to a self-understood point O, namely,

$$m = p \times e \tag{3.2.5}$$

with p being the vector directed from O to an arbitrary point P of $\mathcal{L}$. Moreover, e and m are called the primal and dual parts of $\hat{e}$.

Now, let $\mathcal{L}_i$ and $\mathcal{L}_j$, be two lines. Their *dual angle* is defined as

$$\hat{\upsilon}_{ij} = \upsilon_{ij} + \varepsilon h_{ij} \tag{3.2.6}$$

where υ_{ij} is the *projected angle* between e_i and e_j, and h_{ij} is the *distance* between $\mathcal{L}_i$ and $\mathcal{L}_j$. Furthermore,

$$\sin\hat{\upsilon}_{ij} = \sin\upsilon_{ij} + \varepsilon h_{ij}\cos\upsilon_{ij} \tag{3.2.7}$$

$$\cos\hat{\upsilon}_{ij} = \cos\upsilon_{ij} - \varepsilon h_{ij}\sin\upsilon_{ij} \tag{3.2.8}$$

Hence, the dual angle $\hat{\upsilon}_{ij}$ uniquely determines the relative layout of the two lines $\mathcal{L}_i$ and $\mathcal{L}_j$ in space. Furthermore, the following relations that are in exact analogy with real vectors can be verified,

$$\cos\hat{\upsilon}_{ij} = \hat{e}_i \cdot \hat{e}_j \tag{3.2.9}$$

$$\sin\hat{\upsilon}_{ij} = (\hat{e}_i \times \hat{e}_j) \cdot \hat{n}_{ij} \tag{3.2.10}$$

where $\hat{n}_{ij}$ is the dual vector representing the line $\mathcal{N}_{ij}$ that coincides with the common normal of $\mathcal{L}_i$ and $\mathcal{L}_j$, and with the same direction as that of the vector from H_i to H_j, namely $\hat{n}_{ij} = n_{ij} + \varepsilon\tilde{n}_{ij}$ where

$$n_{ij} = \frac{h_j - h_i}{\|h_j - h_i\|} \tag{3.2.11}$$

and $\tilde{n}_{ij} = n_{ij} \times h_i = n_{ij} \times h_j$. Hence, equations (3.2.4) through (3.2.11) uniquely determine the dual angle $\hat{\upsilon}_{ij}$ subtended by the two lines. Three different possibilities for the layout of two distinct lines $\mathcal{L}_i$ and $\mathcal{L}_j$ exist as explained below:

- (A) **Non-Parallel and Non-Intersecting Lines:** $\hat{\upsilon}_{ij}$ is a *proper dual number*, i.e., $\upsilon_{ij} \neq k\pi$, with $k = 0, 1$ and $h_{ij} \neq 0$
- (B) **Parallel Lines**: $\hat{\upsilon}_{ij}$ is a pure dual number, (its primal part is zero), i.e., $\upsilon_{ij} = k\pi$, with $k = 0, 1$ and $h_{ij} \neq 0$.
- (C) **Intersecting Lines**: $\hat{\upsilon}_{ij}$ is a real number, (its dual part is zero), i.e., $\upsilon_{ij} \neq k\pi$, with $k = 0, 1$ and $h_{ij} = 0$.

Now, for two cylinders C_i and C_j to collide, one of the three cases discussed below must occur:

- (1) **Body-Body Collision:** This situation - the most likely one - is shown in Figure 3.3, where two cylindrical bodies of an object intersect.
- (2) **Base-Body Collision:** The cylindrical body of one cylinder collides with one of the two circular disks of the other cylinder.
- (3) **Base-Base:** One of the circular disks of one cylinder collides with a circular disk of another cylinder.

(A) Cylinders with Non-Parallel and Non-Intersecting Axes

In order to characterize the types of possible collisions for two cylinders whose major axes are represented by $\mathcal{L}_i$ and $\mathcal{L}_j$, that are non-parallel and non-intersecting, the following steps are taken:

- First we need to determine the location of the points H_i along $\mathcal{L}_i$ and H_j along $\mathcal{L}_j$, i.e, the feet of the common normal on the two lines. This can be done by determining the scalars h_i and h_j, as given below:

$$\mathrm{h}_i = \frac{(p_i - p_j) \cdot (e_j \cos \upsilon_{ij} - e_i)}{\sin^2 \upsilon_{ij}} \tag{3.2.12}$$

$$\mathrm{h}_j = \frac{(p_j - p_i) \cdot (e_i \cos \upsilon_{ij} - e_j)}{\sin^2 \upsilon_{ij}} \tag{3.2.13}$$

with $h_i = p_i + \mathrm{h}_i e_i$ and $h_j = p_j + \mathrm{h}_j e_j$.

- Now, if $h_{ij} > (R_i + R_j)$ collision is not possible.

- If $h_{ij} \le (R_i + R_j)$, collision is possible, as explained below:

- (A-1) If $\mathrm{b}_i \le \mathrm{h}_i \le \mathrm{t}_i$ and $\mathrm{b}_j \le \mathrm{h}_j \le \mathrm{t}_j$, then we have a body-body collision, and the critical points P_i^c and P_j^c on the axes are H_i and H_j respectively, (Figure 3.3), with the critical direction being n_{ij}.

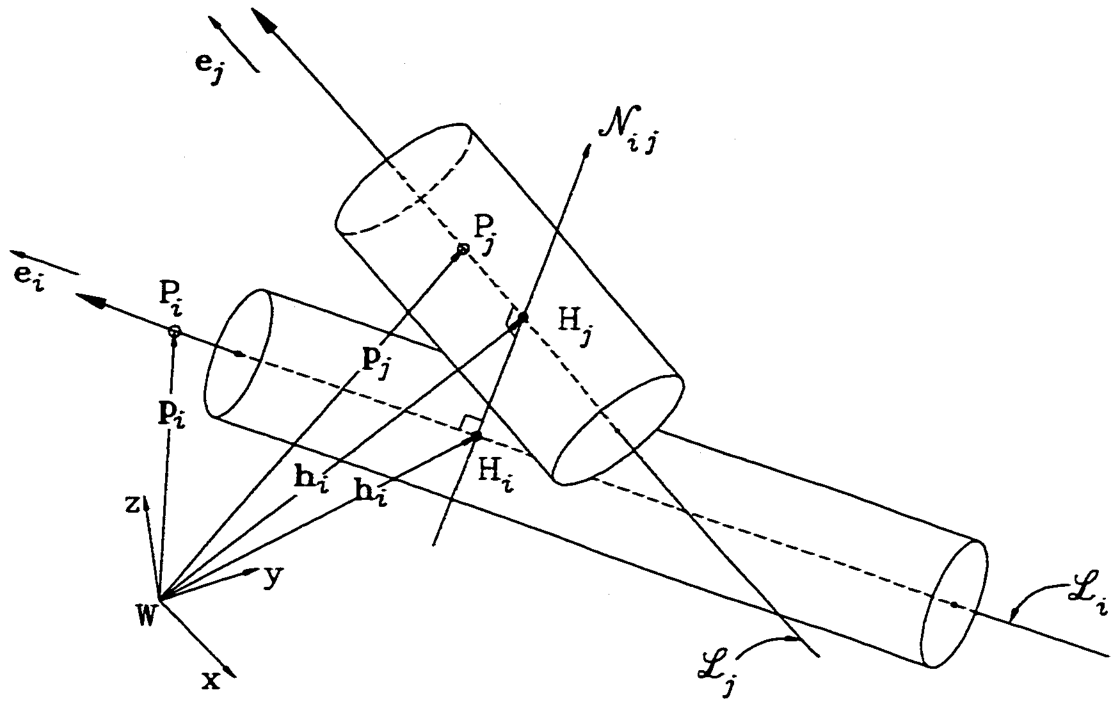


Figure 3.3 (A-1) Body-Body collision (non-parallel and non-intersecting axes)

- (A-2) If only one of the points H_i or H_j lies outside of its corresponding cylinder, then, we may or may not have a collision. However, if the two cylinders collide, it has to be in the form of a base-body collision only, (Figure 3.4). As an example, in order to determine the critical points and the critical direction, we assume that H_i lies inside C_i with H_j being outside C_j. The critical point P_j^c of C_j will thus be one of the two pints B_j or T_j, whichever lies closer to H_j. Moreover, the critical point P_i^c of the cylinder C_i is the projection of P_j^c on $\mathcal{L}_i$. If the vector p_j^c is the vector representing P_j^c, we will have

$$p_i^c = p_i + (\tilde{p}_j^c \cdot e_i)e_i \qquad (3.2.14)$$

where $\tilde{p}_j^c$ is the vector connecting P_j^c to P_i. We will thus consider that a collision occurs, whenever the following inequality is satisfied

$$\|p_i^c - p_j^c\| \le (R_i + R_j) \qquad (3.2.15)$$

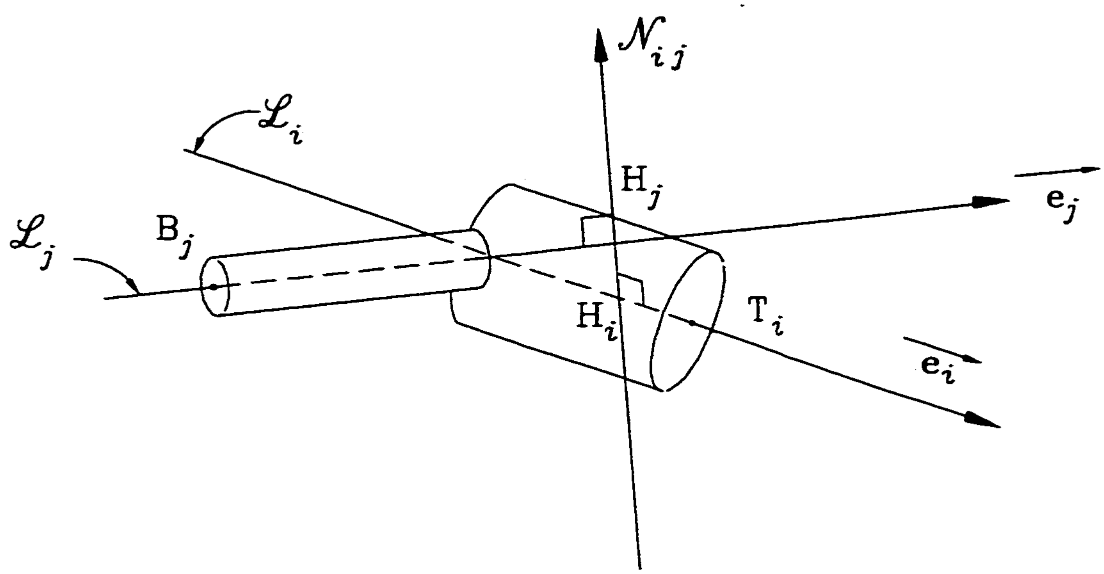


Figure 3.4 (A-2) Base-Body Collision (non-parallel and non-intersecting axes)

It has to be mentioned that the foregoing inequality gives a conservative prediction of collision between the base and the body of the two cylinders. In this manner, we are implicitly assuming that the base of the cylinder is not a simple circular disk, but, a fictitious semi-sphere of the same radius. The critical direction u_{ij} for C_i becomes,

$$u_{ij} = \frac{p_j^c - p_i^c}{\|p_i^c - p_j^c\|} \qquad (3.2.16)$$

Case (A-2) above can lead to instability in the redundancy resolution scheme if the two lines are almost parallel. In this special situation, the location of the critical points on the two lines can go through major changes with small changes in the angle υ_{ij} made by them as shown in Figure 3.5. To remedy this "ill-conditioning", we will inhibit the motion of two points of the line $\mathcal{L}_i$ towards their corresponding projections on $\mathcal{L}_j$ whenever, the two lines are almost parallel. This is achieved by identifying two critical directions - one for each end of C_i - for the redundancy resolution scheme.

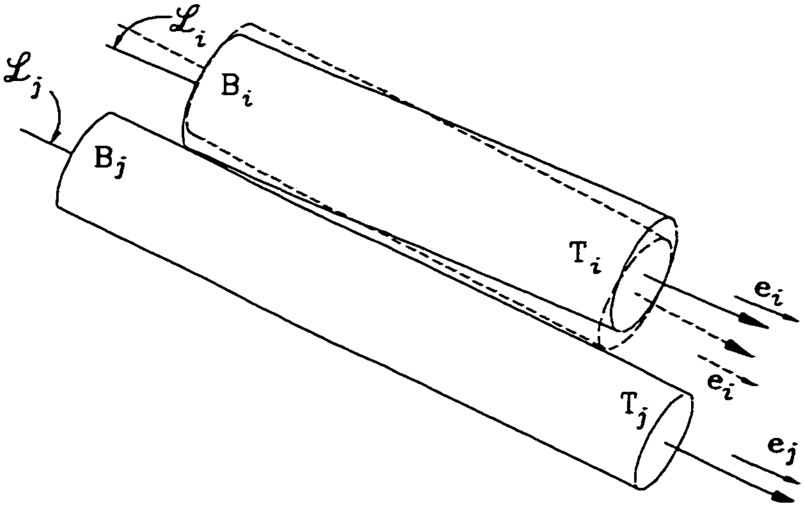


Figure 3.5 Near Parallel axes

- (A-3) If both H_i and H_j lie outside their corresponding cylinders, then we may have a base-base collision, and the critical points and direction will be determined as explained below (Figure 3.6):

Denote by $\{d_k\}$ the set of distances of B_i and T_i to B_j and T_j, i.e.

$$d_1 = \|b_i - t_j\|, \qquad d_2 = \|b_i - b_j\|$$
$$d_3 = \|t_i - t_j\|, \qquad d_4 = \|t_i - b_j\| \tag{3.2.17}$$

and $d_c \equiv min\{d_1, d_2, d_3, d_4\}$, then we have a base-base collision if, $d_c \leq (R_i + R_j)$. Once again, the foregoing prediction is conservative as it assumes two semi-spherical base bodies attached to the ends of the cylinders rather than the simple circular disks.

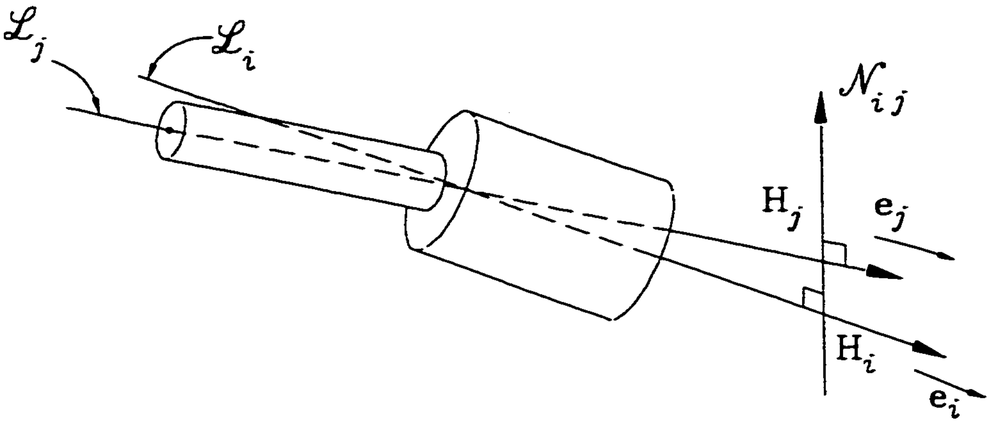


Figure 3.6 (A-3) Base-Base Collision (non-parallel and non-intersecting axes)

(B) Cylinders with Intersecting Axes

In order to characterize a collision between two cylinders with intersecting axes, we first project the end-points B_i and T_i of the cylinder C_i onto the line $\mathcal{L}_j$ and denote the projected points by B'_j and T'_j. Conversely, we project the points B_j and T_j of the cylinder C_j onto the line $\mathcal{L}_i$ and denote the projected points by B'_i and T'_i. Position vectors of the foregoing four points will take on the form:

$$b'_i = p_i + \mathrm{b}'_i e_i, \qquad t'_i = p_i + \mathrm{t}'_i e_i$$
$$b'_j = p_j + \mathrm{b}'_j e_j, \qquad t'_j = p_j + \mathrm{t}'_j e_j \tag{3.2.18}$$

with

$$\mathrm{b}'_i = -(p_i - b_j) \cdot e_i, \qquad \mathrm{t}'_i = -(p_i - t_j) \cdot e_i$$
$$\mathrm{b}'_j = -(p_j - b_i) \cdot e_j, \qquad \mathrm{t}'_j = -(p_j - t_i) \cdot e_j \tag{3.2.19}$$

- (B-2) If any one of the following four conditions holds, then we will have a base-body collision, and the critical direction will be a unit vector pointing along a vector joining the corresponding critical points, (Figure 3.7),

$$\mathrm{b}_i \le \mathrm{b}'_i \le \mathrm{t}_i, \quad and, \qquad \|b'_i - b_i\| \le (R_i + R_j)$$
$$\mathrm{b}_i \le \mathrm{t}'_i \le \mathrm{t}_i, \quad and, \qquad \|t'_i - t_i\| \le (R_i + R_j)$$
$$\mathrm{b}_j \le \mathrm{b}'_j \le \mathrm{t}_j, \quad and, \qquad \|b'_j - b_j\| \le (R_i + R_j)$$
$$\mathrm{b}_j \le \mathrm{t}'_j \le \mathrm{t}_j, \quad and, \qquad \|t'_j - t_j\| \le (R_i + R_j) \tag{3.2.20}$$

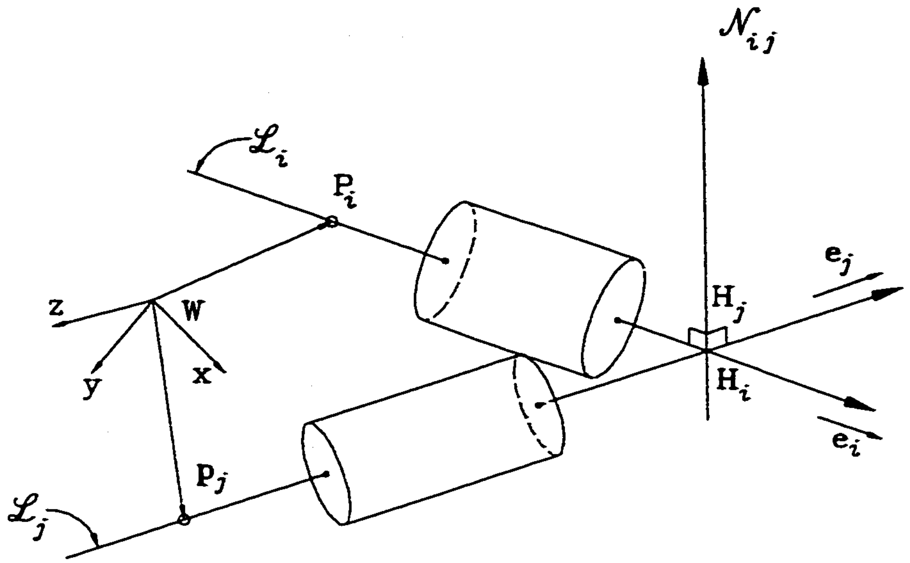


Figure 3.7 (B-2) Base-Body Collision (intersecting axes)

- (B-3) If none of the foregoing conditions is satisfied, then we will not have a base-body collision. However, we may have a base-base collision. The procedure for base-base collision detection for a pair of intersecting lines is similar to that of case (A-3) explained earlier, (Figure 3.8)

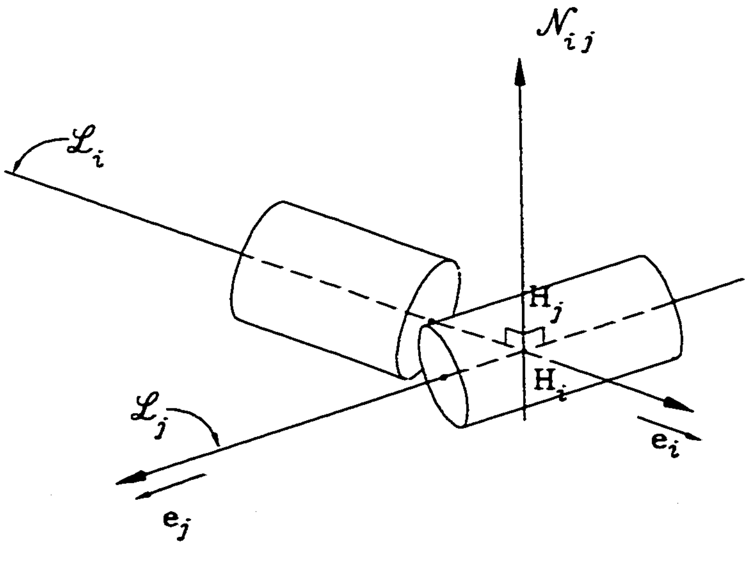


Figure 3.8 (B-3) Base-Base Collision (intersecting axes)

B) Cylinders with Parallel Axes

For the special case of two parallel lines $\mathcal{L}_i$ and $\mathcal{L}_j$ for which an infinite number of common normals exist, we resort to a unique definition for one common normal lying closest to the origin [69] - see Figure 3.9). If the line $\mathcal{N}_{ij}$ passes through the points H_i and H_j of $\mathcal{L}_i$ and $\mathcal{L}_j$ (with H_i and H_j being the closest points of the two lines to the origin), then the dual representation of $\mathcal{N}_{ij}$ is given as,

$$\hat{n}_{ij} = \frac{h_j - h_i}{\mathrm{h}_{ij}} + \varepsilon \frac{h_i \times h_j}{\mathrm{h}_{ij}} \qquad (3.2.21)$$

where, h_i and h_j are the position vectors of the points H_i and H_j respectively, and $\mathrm{h}_{ij} = \|h_j - h_i\|$ is the distance between the two lines.

If $\mathrm{h}_{ij} \geq (R_i + R_j)$, then the two cylinders do not collide. However, if $\mathrm{h}_{ij} \leq (R_i + R_j)$, then depending on the location of the cylinders along their axes relative to each other, two special cases of body-body (C-1) and base-base (C-3) collisions can occur:

- (C-1) If $\mathrm{h}_{ij} \leq (R_i + R_j)$, and the projection of either B_i or T_i on $\mathcal{L}_j$ is between B_j and T_j, then we will have a body-body collision. As in the case of near-parallel axes mentioned above in (A-3), to avoid instability, we specify two critical directions, one for each end of C_i, (Figure 3.5).

- (C-3) If (C-1) above is not satisfied, but $\mathrm{h}_{ij} \leq (R_i + R_j)$, then we will obtain the distance between the end points of the two cylinders, as in the case (A-3) above, (Figure 3.9).

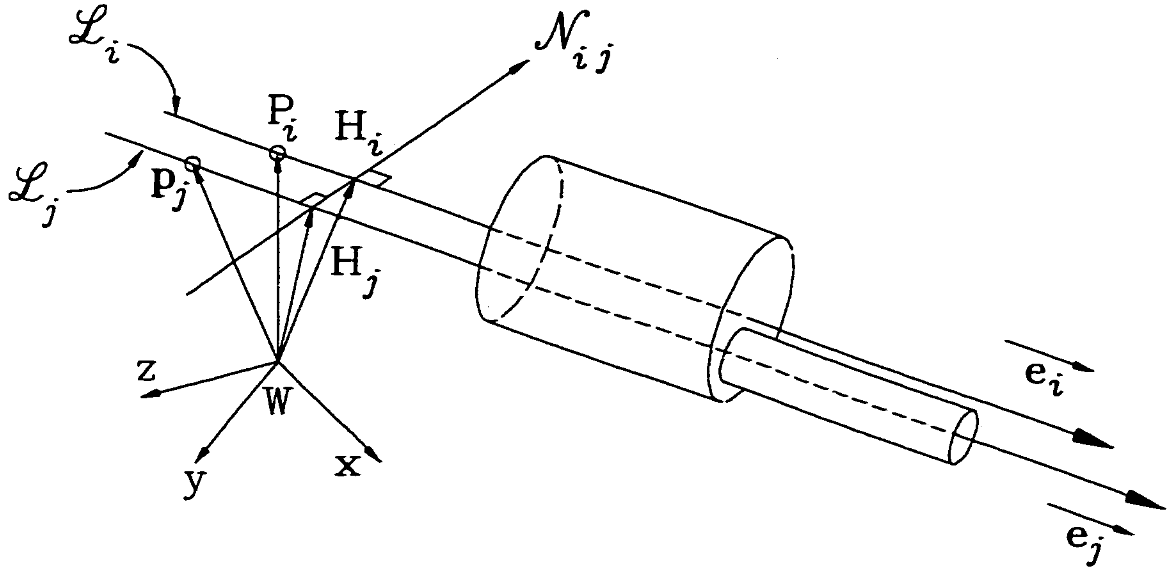


Figure 3.9 (C-3) Base-Base Collision (parallel axes)

3.2.2 Cylinder-Sphere Collision Detection

This case is simpler than that of cylinder-cylinder collision detection. Figure 3.10 shows the basic layout used for collision detection of the cylinder C_i and the sphere S_j. The notation used for the cylinder is the same as in Section 3.2.1 above. The sphere S_j is identified by the location of its center P_j and its radius. The first step is to determine if there is a risk of collision. The point H_i on line $\mathcal{L}_i$ is determined by projecting the center of the sphere on $\mathcal{L}_i$,

$$h_i = (p_{ij} \cdot e_i)e_i + p_i \tag{3.2.22}$$

where p_{ij} is the vector representing $\overrightarrow{P_iP_j}$. The critical distance h_{ij} is given by $\mathrm{h}_{ij} = \|p_j - h_i\|$

Now, if $h_{ij} > (R_i + R_j)$, there is no risk of collision. If $h_{ij} \leq (R_i + R_j)$ then the following cases can occur:

- If $h_{ij} \leq (R_i + R_j)$ and H_i lies inside the cylinder C_i, then the cylinder and the sphere are in collision and the critical points and critical direction are defined by

$$u_{ij} = \frac{p_j - h_i}{\|p_j - h_i\|} \tag{3.2.23}$$

$$p_i^c = h_i + R_i u_{ij} \tag{3.2.24}$$

$$p_j^c = p_j - R_j u_{ij} \tag{3.2.25}$$

- If $h_{ij} \leq (R_i + R_j)$ and H_i lies outside the cylinder C_i, then, we may or we may not have a collision. The critical point on the line $\mathcal{L}_i$ is either B_i or T_i depending on which is closer to H_i. Let us assume that B_i is the closer point to H_i. The critical distance h_{ij} is given by $h_{ij} = \|p_j - b_i\|$. Now, if $h_{ij} > (R_i + R_j)$, there is no risk of collision, otherwise, there is a collision and the critical points and direction are calculated by replacing h_i with b_i in equations (3.2.23) through (3.2.25). It has to be mentioned that the foregoing inequality gives a conservative prediction of collision between the sphere and the cylinder. In this manner, we are implicitly assuming that the base of the cylinder is not a simple circular disk, but, a fictitious semi-sphere of the same radius.

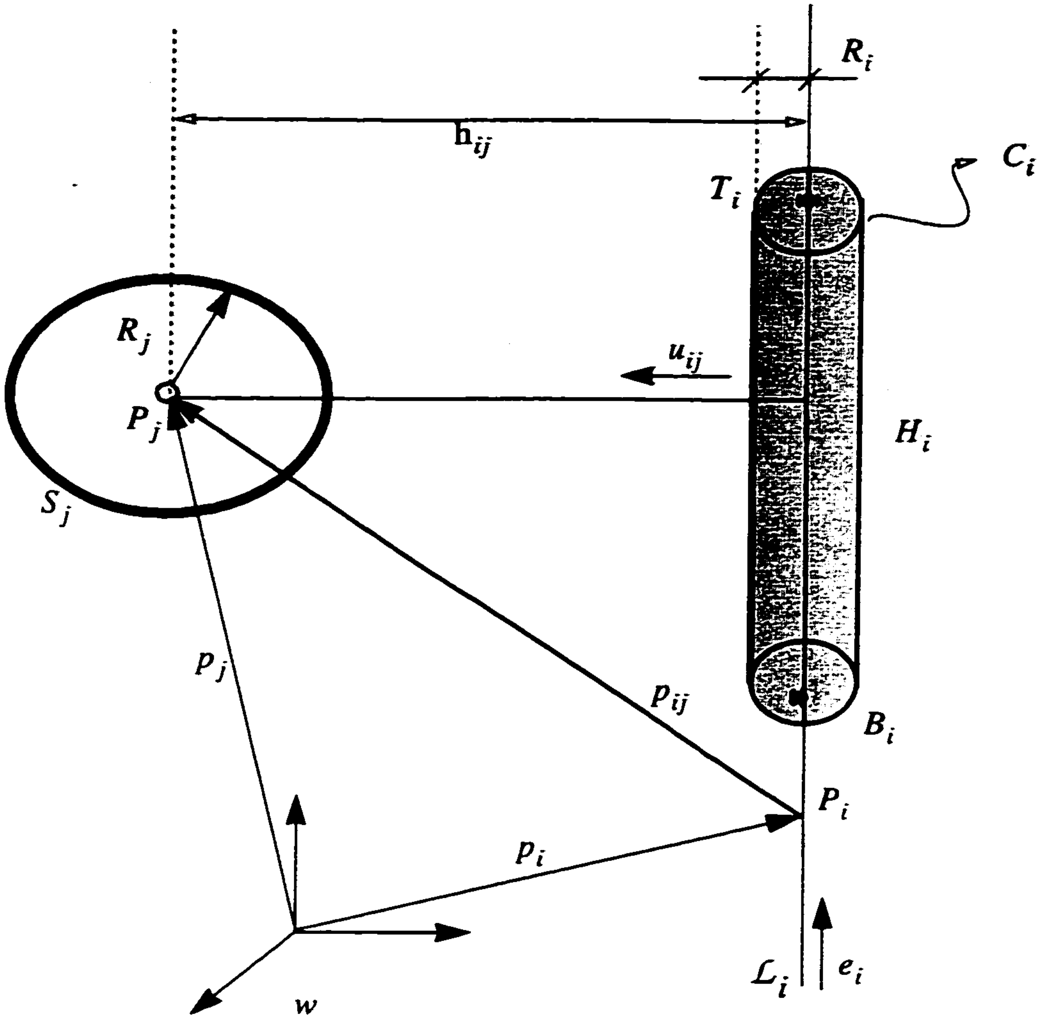


Figure 3.10 Cylinder-Sphere Collision Detection

3.2.3 Sphere-Sphere Collision Detection

This is the simplest case among the three collision detection schemes presented. The critical distance h_{ij} is the distance between the centers of the two spheres. If $\mathrm{h}_{ij} > (R_i + R_j)$, there is no risk of collision, otherwise the two spheres are in collision.

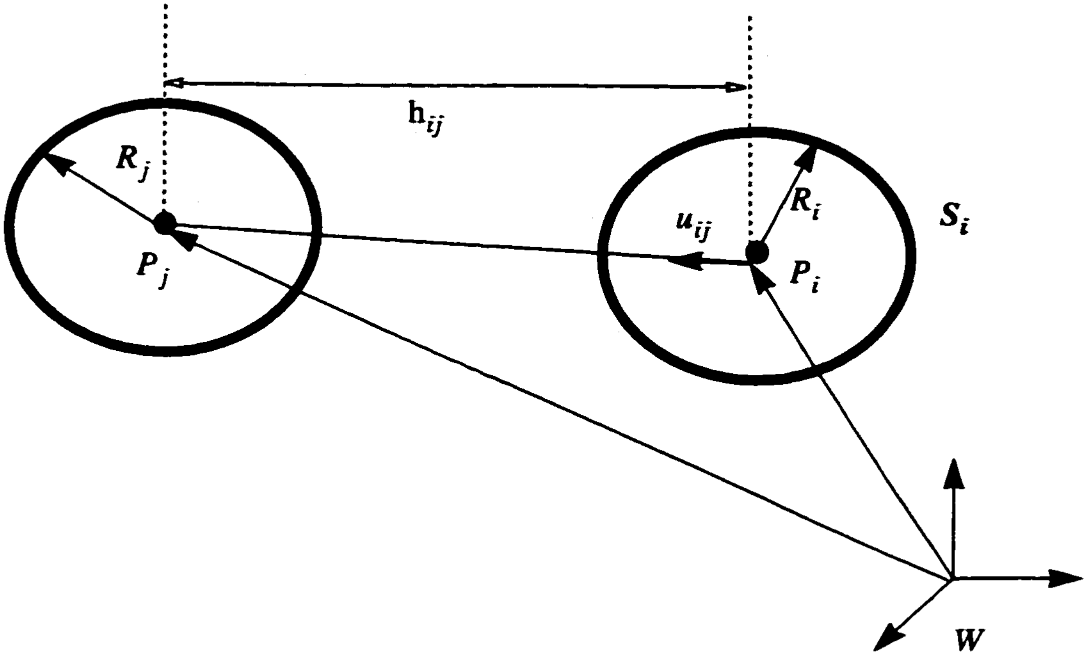


Figure 3.11 Sphere-Sphere Collision Detection

3.3 KINEMATIC SIMULATION FOR A 7-DOF REDUNDANT MANIPULATOR

In this section, the redundancy resolution scheme described in Chapter 2, will be extended for the general case of a 7-DOF redundant manipulator working in a 3-D workspace. The feasibility of the algorithms is illustrated using a kinematic simulation

3.3.1 Kinematics of REDIESTRO

The kinematic description of REDIESTRO (a perspective view of REDIESTRO is shown in Figure 3.1) is obtained by assigning a coordinate frame to each link with its z axis along the axis of rotation. Frame *{1}* is the workspace fixed frame and frame *{8}* is the end-effector frame. Two consecutive frames {i} and {i+1} are related by the 4×4 homogenous transformation matrix:

$$
{}_{i+1}^{i}T = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} ({}_{i+1}^{i}R)_{3\times3} & ({}^{1}P_8)_{3\times1} \\ 0 & 1 \end{bmatrix} \qquad (3.3.1)
$$

where $i = 1 \rightarrow 7$; α_i, θ_i, b_i, and a_i are the twist angle, joint angle, offset and link length respectively. Table B-1 (in Appendix B) gives the values of the Denavit-Hartenberg parameters for each link.The homogenous transformation relating Frame 8 (end-effector frame) to the base frame is given by:

$$
{}_{8}^{1}T = {}_{2}^{1}T{}_{3}^{2}T\ldots \qquad {}_{8}^{7}T \qquad (3.3.2)
$$

3.3.2 Main Task Tracking

The main task is described by a 6-dimensional vector consisting of the end-effector pose (position and orientation): $X^T = [P^T{}_e, O^T{}_e]$. This 6-dimensional vector is dimensionally non-homogenous and needs different treatment for the 3-dimensional vector representing the end-effector position from that representing orientation. Therefore, the main task is divided into two independent sub-tasks.

3.3.2.1 Position Tracking

The position is described in the workspace fixed reference frame. Both the desired and the actual position are described in this frame. The *ith* column of the Jacobian corresponding to the position of the end-effector in frame *{1}* is defined by

$$J_{P_e}(i)_{3\times 1} = {}^1_i\hat{Z} \times ({}^1P_{8origin} - {}^1P_{iorigin}) \qquad i = 1 \rightarrow 7 \qquad (3.3.3)$$

where ${}^1_i\hat{Z}$ is the unit vector along the *Z* axis of joint *i*, ${}^1P_8$ is the position of the end-effector, and ${}^1P_{iorigin}$ is the position of the origin of the *ith* frame with respect to frame *{1}*. The position and the velocity errors are given by

$$e_p = {}^1P_8^d - {}^1P_8, \qquad \dot{e}_p = J_{P_e}\dot{q} - {}^1\dot{P}_8^d \qquad (3.3.4)$$

where $\dot{q}$ is the vector of joint velocities, and the superscript *d* denotes the desired values.

3.3.2.2 Orientation Tracking

The orientation of the end-effector is represented by the 3×3 matrix ${}^1_8R$, called the *Direction Cosine* matrix. The *ith* column of the Jacobian matrix which relates the angular velocity of the end-effector (${}^1_e\omega$) to the joint velocity, i.e., ${}^1_e\omega = J_{O_e}\dot{q}$, can be calculated from the following equation

$$J_{O_e}(i) = {}^1_i\hat{Z} \qquad i = 1 \rightarrow 7 \tag{3.3.5}$$

The procedure for finding the orientation error and its derivative is more complicated than that for the case of position. In this case, the desired orientation is described by a 3×3 matrix whose columns are unit vectors coincident with the desired X, Y, and Z axes of the end-effector. The actual orientation of the end-effector is given by the matrix ${}^1_8R$. The orientation error is calculated as follows [70]: $e_O = {}^1K\sin\Theta$, where ${}^1K$ and Θ are the axis and angle of rotation which transform the end-effector frame to the desired orientation. The calculation of the angular velocity error is then straightforward:

$$\dot{e}_O = {}^1_e\omega^d - J_{O_e}\dot{q} \tag{3.3.6}$$

3.3.2.3 Simulation Results

The performance of redundancy resolution in tracking the main task trajectories is studied here by computer simulation. The integration step size in the following simulations is *10 ms*, and the main task consists of tracking the position and orientation trajectories, generated by linear interpolation between the initial and final configurations, specified by:

$$ {}^1P_8^{d-intial} = [61.8 \qquad 231.4 \qquad 1127.1]^T, \quad {}^1_8R^{d-initial} = \begin{bmatrix} 0.143 & -0.25 & -0.958 \\ -0.93 & 0.30 & -0.22 \\ 0.339 & 0.921 & -0.19 \end{bmatrix}$$

$$ {}^1P_8^{d-final} = [500 \qquad 500 \qquad 1102.3]^T, \quad {}^1_8R^{d-final} = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ 0 & 1 & 0 \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix}$$

The overall redundancy resolution scheme has not been changed (see Section 2.3.1.3). The only difference consists of splitting the main task into two independent sub-tasks with weighting matrices denoted by W_{P_e} and W_{O_e} corresponding to position and orientation respectively of the end-effector.

The joint velocities are calculated from

$$\dot{q} = A^{-1} b \tag{3.3.7}$$

where

$$A = J_{P_e}^T W_{P_e} J_{P_e} + J_{O_e}^T W_{O_e} J_{O_e} + J_c^T W_c J_c + W_v \tag{3.3.8}$$

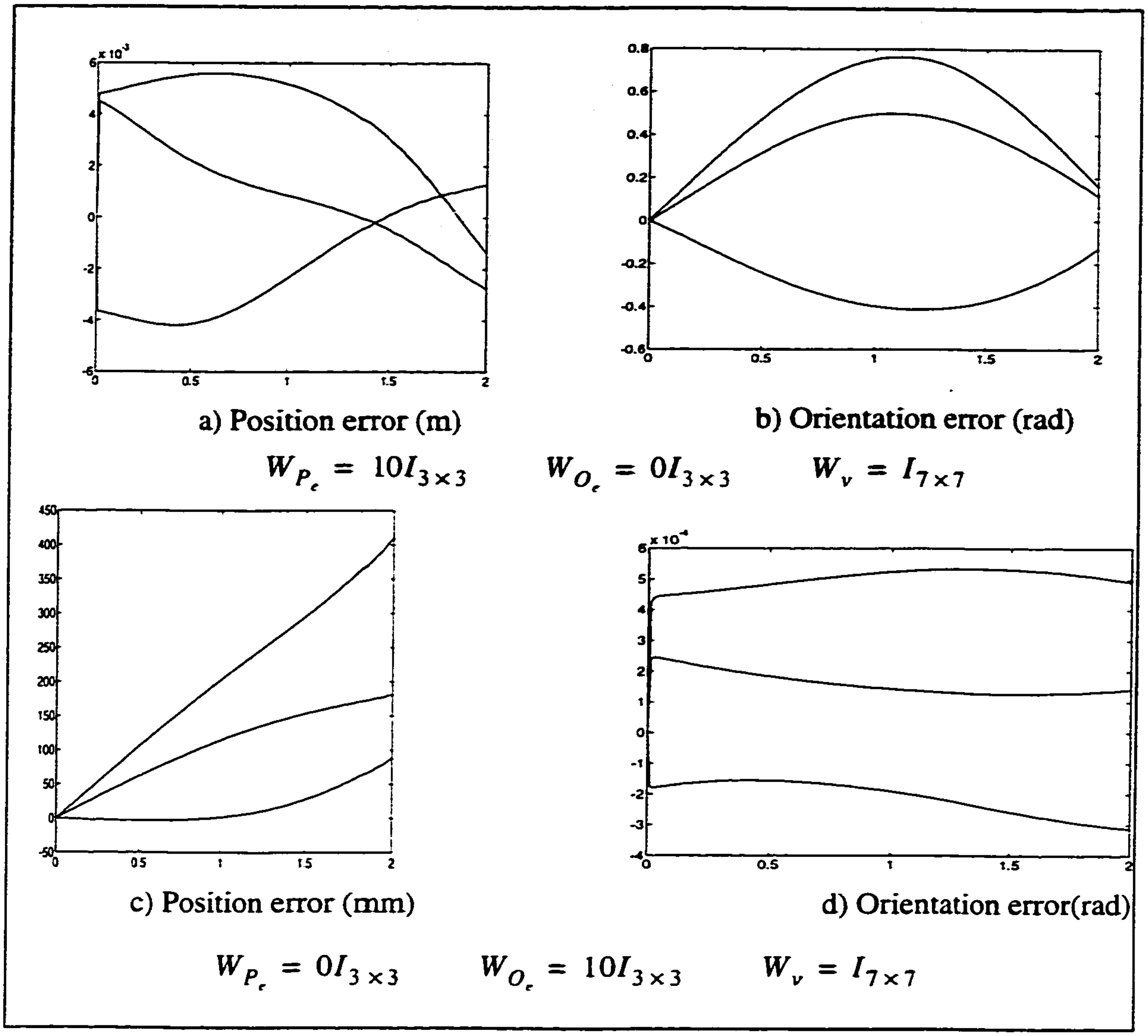
$$b = J_{P_e}^T W_{P_e} \dot{P}_r + J_{O_e}^T W_{O_e} \Omega_r + J_c^T W_c \dot{Z}_r \tag{3.3.9}$$

The subscript *c* refers to the additional task which is not active in the simulation presented in this section. It should be noted in the following simulations that redundancy resolution is implemented in closed-loop. Hence, the reference velocities are given by:

$$\dot{P}_r = {}^1\dot{P}_8{}^d + K_{p_P}({}^1P_8{}^d - {}^1P_8) \tag{3.3.10}$$

$$\Omega_r = {}^1_e\Omega^d + K_{p_O} e_O \tag{3.3.11}$$

where K_{p_P} and K_{p_O} are the position and orientation proportional gains respectively. In the first simulation, the sub-task corresponding to tracking the desired orientation is inactive. Figures 3.12a, b show the position and orientation errors. In the second simulation only the orientation sub-task is active, and the results are shown in Figures 3.12c, d. In this case, no attempt has been made to follow the position trajectory. The position and orientation errors are mainly due to the presence of W_v in the damped least-squares formulation of the redundancy resolution.



a) Position error (m)

b) Orientation error (rad)

$W_{P_e} = 10I_{3\times3}$ $W_{O_e} = 0I_{3\times3}$ $W_v = I_{7\times7}$

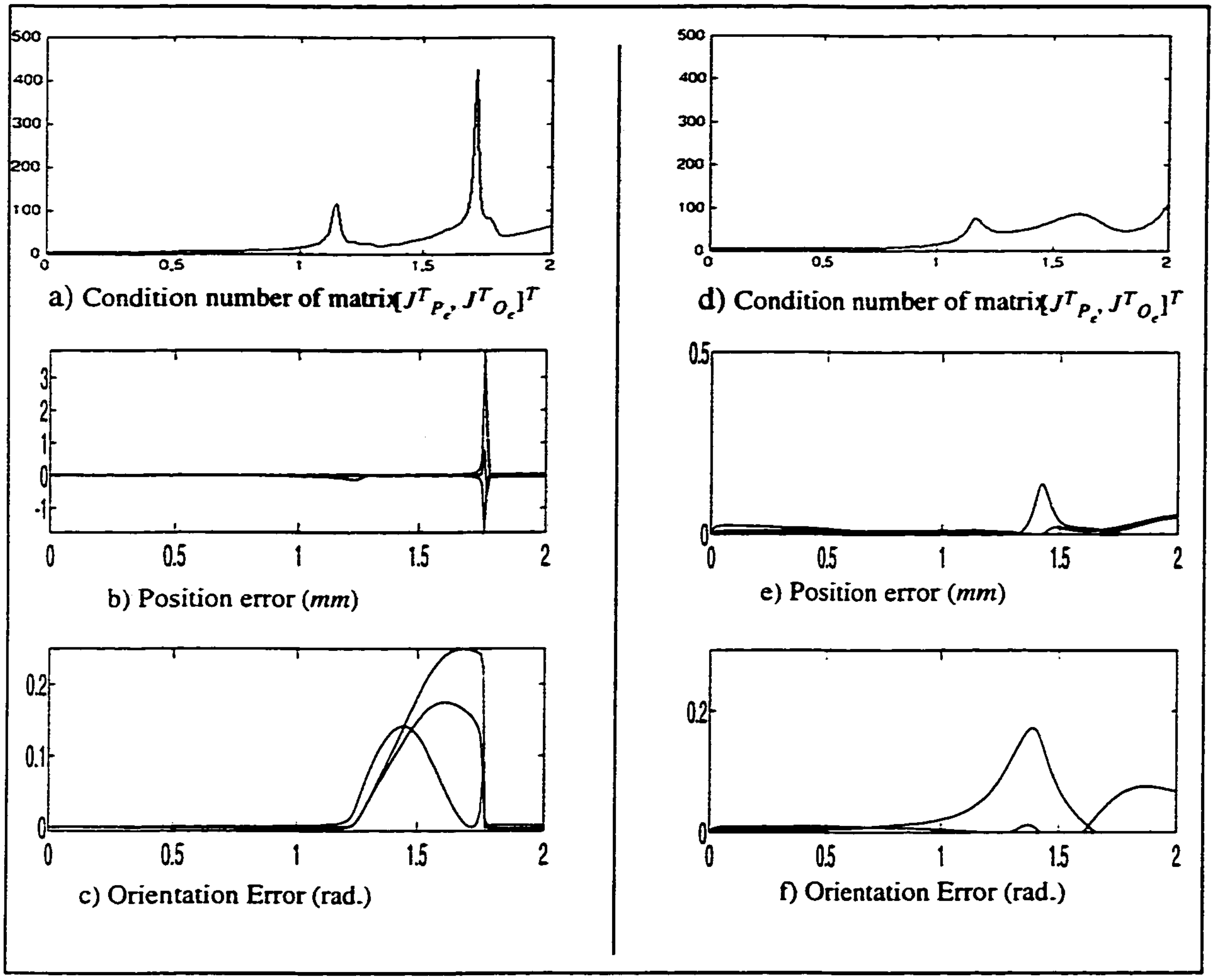
c) Position error (mm)

d) Orientation error(rad)

$W_{P_e} = 0I_{3\times3}$ $W_{O_e} = 10I_{3\times3}$ $W_v = I_{7\times7}$

Figure 3.12 Simulation results for position and orientation tracking

In the following simulations, both position and orientation sub-tasks are active. Figure 3.13a, b, and c show the results of the simulation with small W_v (the singularity robustness factor). As we can see in Figure 3.13a, at some point, the position and orientation sub-tasks are in conflict with each other. This causes the whole Jacobian of the main task to approach a singular position where the condition number of the Jacobian matrix is $Cond_{max} = 403$. Therefore, there is considerable error on both sub-tasks. Figure 3.13 d, e, and f show the simulation result with a larger value of W_v. This time, the whole Jacobian matrix remains far from singularity ($Cond_{max} = 105$), and the maximum errors are reduced significantly. However, in the case that $W_v = 20I_{7 \times 7}$, there is considerable error at the end of the trajectory. This shows that W_v should be selected as small as possible.



a) Condition number of matrix$[J^T_{P_e}, J^T_{O_e}]^T$

b) Position error (mm)

c) Orientation Error (rad.)

d) Condition number of matrix$[J^T_{P_e}, J^T_{O_e}]^T$

e) Position error (mm)

f) Orientation Error (rad.)

Figure 3.13 Simulation results when both main sub-tasks are active; left column $W_v = 1I_{3 \times 3}$, right column $W_v = 20I_{3 \times 3}$

The isotropic design of REDIESTRO reduces the risk of approaching a singular configuration over a greater part of the workspace. However, this risk cannot be eliminated completely, and the singularity robustness factor W_v should either be selected large enough, which introduces errors in the main task, or it should have a time-varying formulation whereby it is chosen with diagonal entries proportional to the inverse of the minimum singular value of the Jacobian of the main task. Figure 3.14 shows the comparison between these two approaches. As one can conclude, the variable weight formulation shows better performance because W_v has small values far from a singular configuration. Hence, it does not introduce errors on the main task, and it increases appropriately near a singular configuration. However, considering the computational complexity of the numerical implementation of the SVD algorithm for a 7-DOF arm, and depending on the available computing power, it may not be feasible for real-time control.

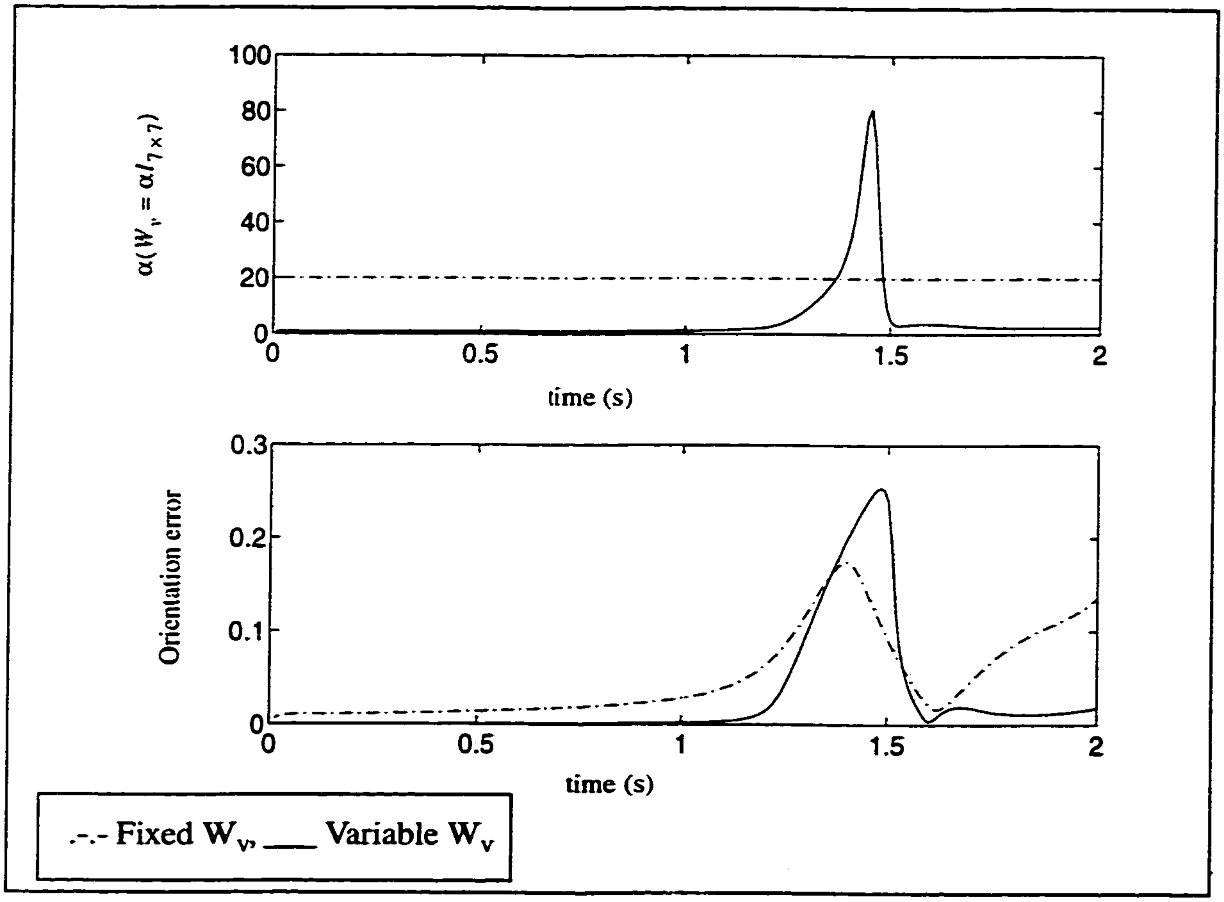


Figure 3.14 Comparison between the fixed and the time-varying singularity robustness factor

3.3.3 Additional Tasks

The additional tasks incorporated in the redundancy resolution module are as follows: Joint Limit Avoidance (JLA), Static and Moving Obstacle Collision Avoidance (SOCA, MOCA) and Self Collision Avoidance (SCA).

3.3.3.1 Joint Limit Avoidance

The JLA algorithm developed in Section 2.4.1.3 is extended here to 3-D without major modifications. In this case, the Jacobian matrix of the JLA corresponding to the *ith* joint is:$J_C = e_i^T$, where e_i is the *ith* column of the matrix $I_{7 \times 7}$. The same weight scheduling scheme is used as that implemented for JLA in Section 2.4.1.3 .

In the following simulation, the main task is the same as in Section 3.3.2 with both position tracking and orientation tracking active. Figure 3.15 shows that with JLA inactive, joint 4 has a minimum value equal to 67 degrees. When the JLA is active with minimum 80 degrees for joint 4, this joint is prevented from violating its limit while tracking the main task trajectory. The position and orientation tracking errors converge to small values except for a short transition period when the JLA task becomes active.

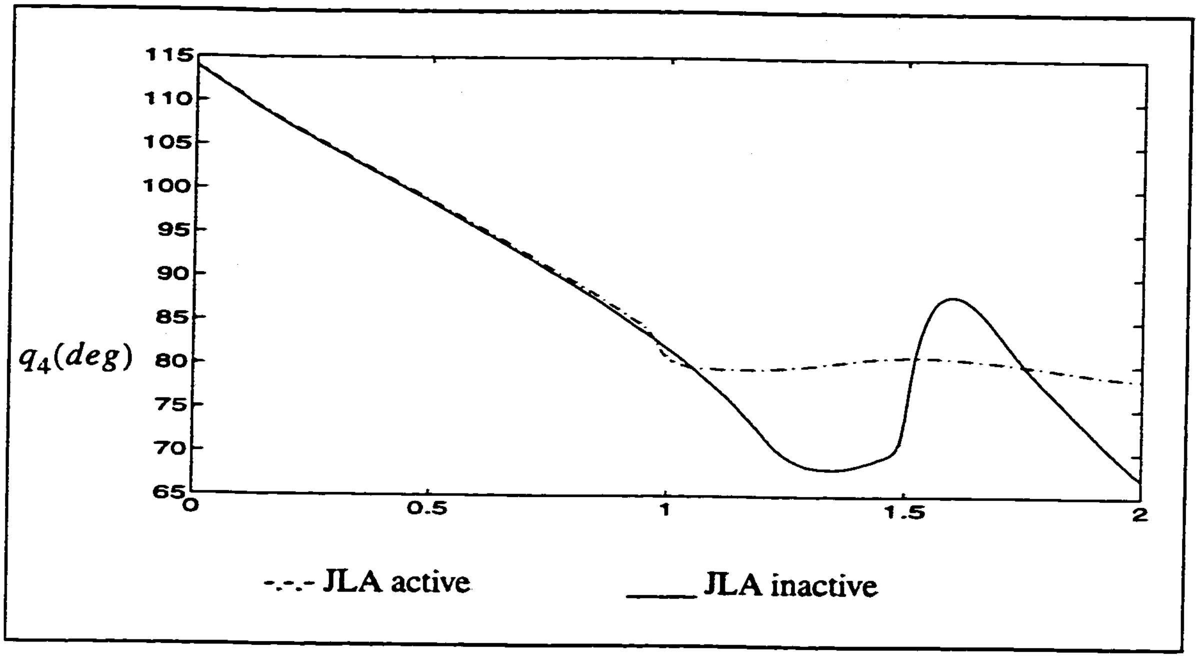


Figure 3.15 Simulation result for JLA in the 3-D workspace with $q_{4_{min}} = 80°$

3.3.3.2 Static and Moving Obstacle Collision Avoidance

A graphical rendering of REDIESTRO with its actual links and actuators is shown in Figure 3.1, while Figure 3.16 depicts the arm with each moving element of the arm enclosed in a cylindrical primitive. The links and the actuator units are modeled by 14 cylinders in total, the fourth link having the maximum number of 4 sub-links. The end-effector and the tool attached to it are enclosed in a sphere.

The environment is modeled by spherical and cylindrical objects. Each obstacle is enclosed in a cylindrical or a spherical Surface of Influence (SOI). Note that the dimensions of the SOIs are used in distance calculation, collision detection and obstacle avoidance modules rather than the actual dimensions of the obstacles.

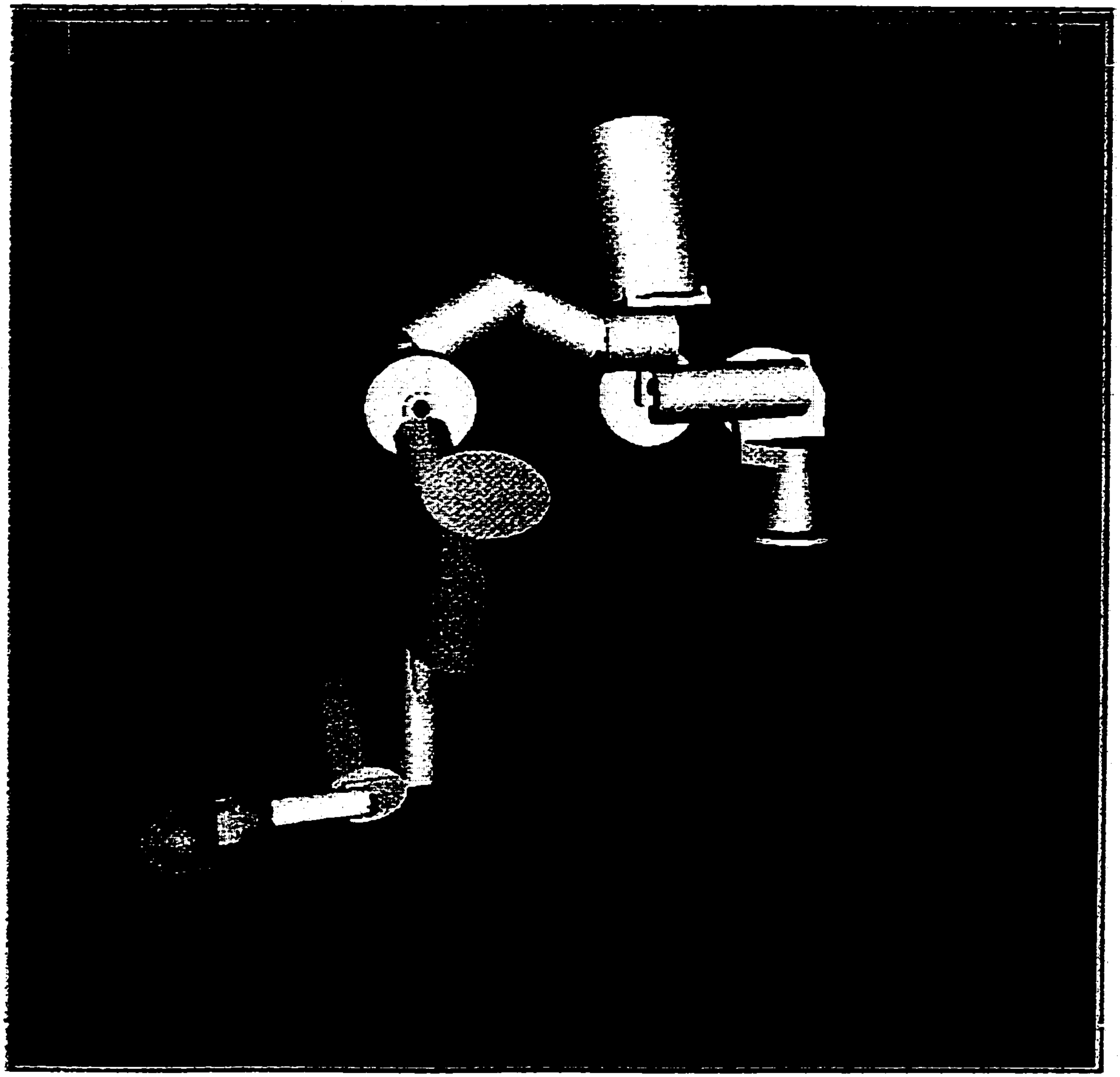


Figure 3.16 REDIESTRO with simplified primitives

Additional task formulation: Let us assume that after performing the distance calculation, the *jth* sub-link of the *ith* link of the manipulator - S_{ij} or C_{ij} depending on the primitive used for modeling - is in the risk of collision with the *kth* obstacle (S_k or C_k). The critical point on the sub-link and the obstacle (P^c_{ij} and are P^c_k) and the critical direction ($u_{ij,k}$), are determined by the collision detection algorithm described in Section 3.2. Now, the additional task z_i for the redundancy resolution module is defined by:

$$z_{i,k} = \mathrm{h}_{ij,k}, \qquad \dot{z}_{ij,k} = -u^T_{ij,k}(J^c_{ij}\dot{q} - \dot{p}^c_k) \tag{3.3.12}$$

where $\mathrm{h}_{ij,k}$ is the critical distance, $J^c_{ij}\dot{q} \equiv \partial p^c_{ij}/\partial \dot{q}$ is the Jacobian matrix mapping the joint rates $\dot{q}$ into the velocity of the critical point P^c_{ij} of the manipulator, while $\dot{p}^c_k$ is the velocity of the obstacle k. The desired values for the active constraints (additional tasks) are: $z^d_i = \dot{z}^d_i = 0$. Note that we still need to calculate the Jacobian of the active constraints. First, the Jacobian of the critical point is calculated, i.e.,

$$J^c_{ij} = [\bar{J}_{3\times i} \qquad 0_{3\times 7-i}] \tag{3.3.13}$$

The *kth* column of the matrix $\bar{J}$ is given by:

$$\bar{J}(k)_{3\times 1} = \hat{a}_k \times (p^c_{ij} - p_{korigin}) \qquad k = 1 \rightarrow i \tag{3.3.14}$$

where $\hat{a}_k$ is the unit vector in the direction of rotation of the *kth* joint, $p_{korigin}$ is the position vector of the origin of the *kth* local frame. Note, that all variables are defined in frame *{1}*.

Now, the Jacobian of the additional task to be used by the redundancy resolution module is calculated as:

$$J_c = -u^T_{ij,k}J^c_{ij} \tag{3.3.15}$$

Analysis: The performance of the obstacle avoidance scheme has been studied by various simulations for different scenarios. As an example, the simulation results for the MOCA are illustrated in Figure 3.17. In these simulations, the main task consists of keeping the position of the end-effector constant while avoiding collision with a moving object. Figure 3.17 shows the results of the simulations for different constant values of the weighting matrix corresponding to the collision avoidance task. It should be noted that, when *Wc* is too small, the object collides with the arm. When *Wc* is large enough, no collision occurs, but there is a rapid increase in the joint velocities which results in a large pulse in joint accelerations (see Figure 3.17). In a practical implementation, the maximum acceleration of each joint would be limited and this commanded joint acceleration would result in saturation of the actuators.

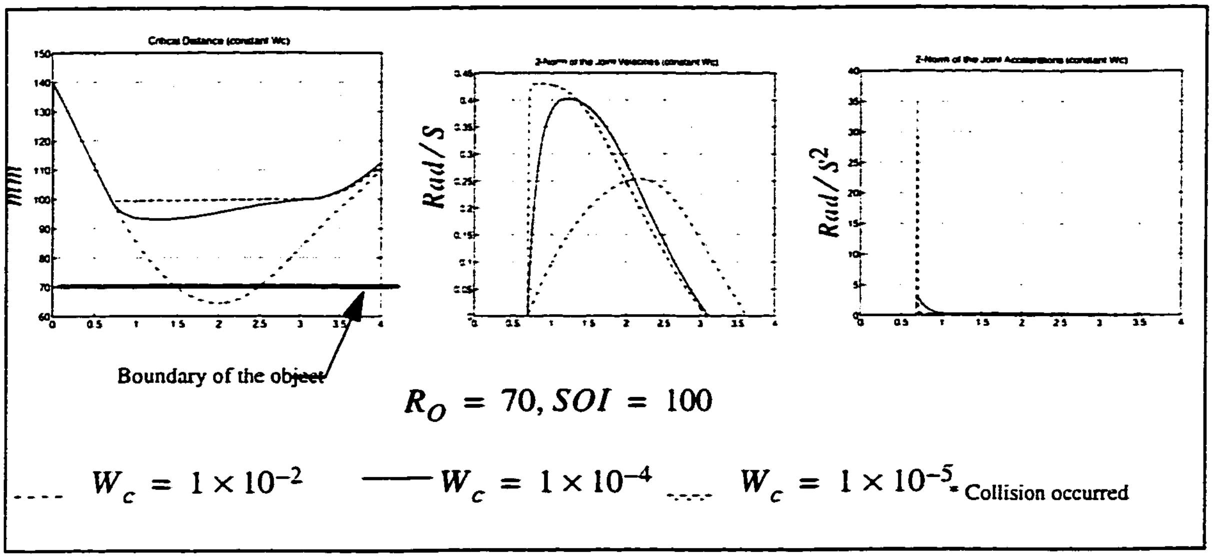


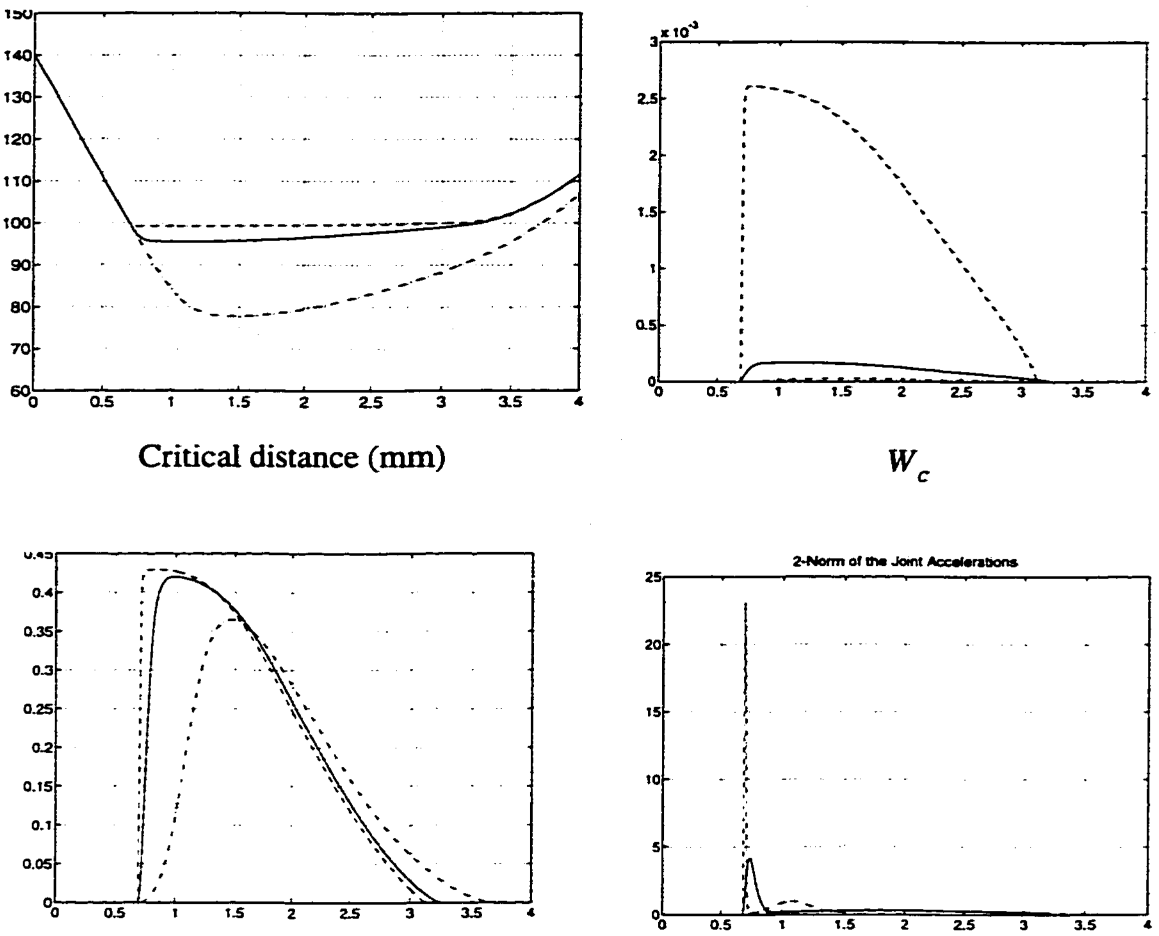
Figure 3.17 Simulation results for MOCA with fixed weighting factor

The optimal value of *Wc* depends on factors such as object velocity, end-effector velocity, and location of the critical point. Therefore, from preliminary simulations, it was observed that finding a fixed value which performs well in different situations is very difficult. To overcome this problem, a time-varying formulation [9] has been used to adjust the weighting factor automatically. In this way, the weighting factor corresponding to each active task is adjusted according to the following scheme:

$$W_c = k\left(\frac{1}{(d_c - R_O)^2} - \frac{1}{(SOI - R_O)^2}\right) \tag{3.3.16}$$

where d_c is the distance between the critical point on the link and either the center of the object for a spherical object or the projection of the critical point on the axis of the cylinder in the case of a cylindrical object. R_O and SOI are the radii and surface of the influence of the objects.

Figure 3.18 shows the results of the simulation using this formulation, which for the case of k=0.01, shows successful operation of MOCA, with minimum acceleration.



Critical distance (mm)

W_c

Norm of joint velocities (rad/s)

Norm of joint accelerations (rad/s^2)

--- k = 100, ___ k = 1, -.-.- k = 0.01 (obstacle's radius = 70 mm and SOI = 100 mm)

Figure 3.18 MOCA simulation results for the time-varying weighting factor

3.4 EXPERIMENTAL EVALUATION USING A 7-DOF REDUNDANT MANIPULATOR

The main objective of these experiments is to demonstrate the capability of the redundancy resolution module in performing the main tasks (position and orientation tracking) while using the extra degrees-of-freedom to fulfill additional tasks (obstacle and joint limit avoidance) for REDIESTRO. The general block diagram of the different modules involved in the hardware experiment is shown in Figure 3.19. The three major modules are:

- The redundancy resolution module (RR)
- The robot and its associated control hardware and software
- The robot animation software: Multi-Robot Simulation (MRS) system [71]

In order to distinguish between the performance of the robot's controller and redundancy resolution scheme, two separate control loops; one at the Cartesian space level (including the RR) and one at the low-level joint controller, have been implemented. In this way, the kinematic simulation (including RR) running on an SGI workstation, generates the desired joint trajectory and this trajectory is then transferred as the joint set points to the VME bus based controller to drive the robot's PID joint controller.

An obstacle avoidance system essentially deals with a complex environment. There are many limitations in creating (modeling) a robot's environment such as space, material, equipment and financial limitations. Creating a time-varying environment (as in the case of moving obstacles) can be even more difficult. One solution to this problem is online transmission of a robot's configuration to a workstation running a graphics visualization of the arm (MRS). MRS serves as a virtual environment; the graphics model of the robot mirrors the exact motion of the arm and the environment can be modeled in the graphics program. This has two main advantages:

- Any complex environment can be modeled with a desired precision (including a time-varying environment)
- The risk of damage to the robot is reduced.

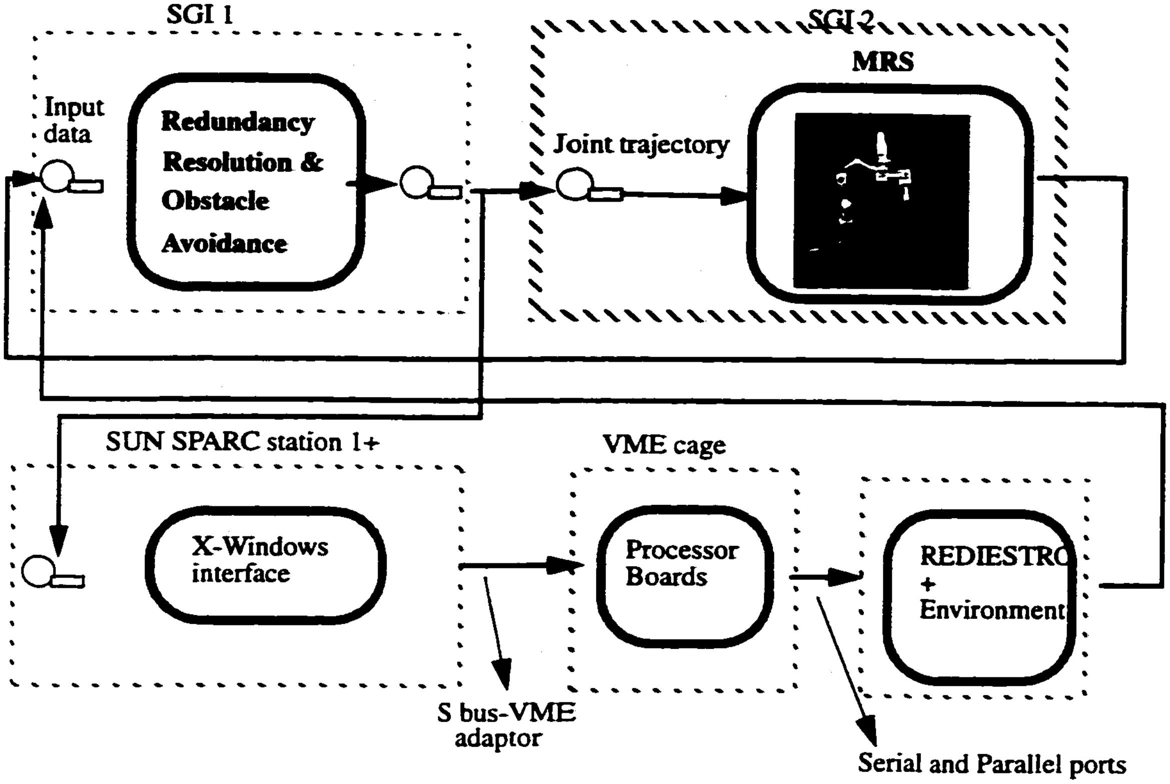


Figure 3.19 General block diagram for the hardware demonstration

3.4.1 Hardware And Software Configurations

At the time of conducting these experiments, REDIESTRO was located at Centre for Intelligence Machines (CIM) at McGill University. The independent joint PID controller was adequate to perform the experiments for collision avoidance. Appendix D describes the hardware and software configurations used in this experiments. Note that after these experiments were completed, REDIESTRO was relocated at Concordia University where a new hardware and software setup was developed to meet the additional requirements needed for force and compliant motion control.

3.4.2 Hardware Demonstration

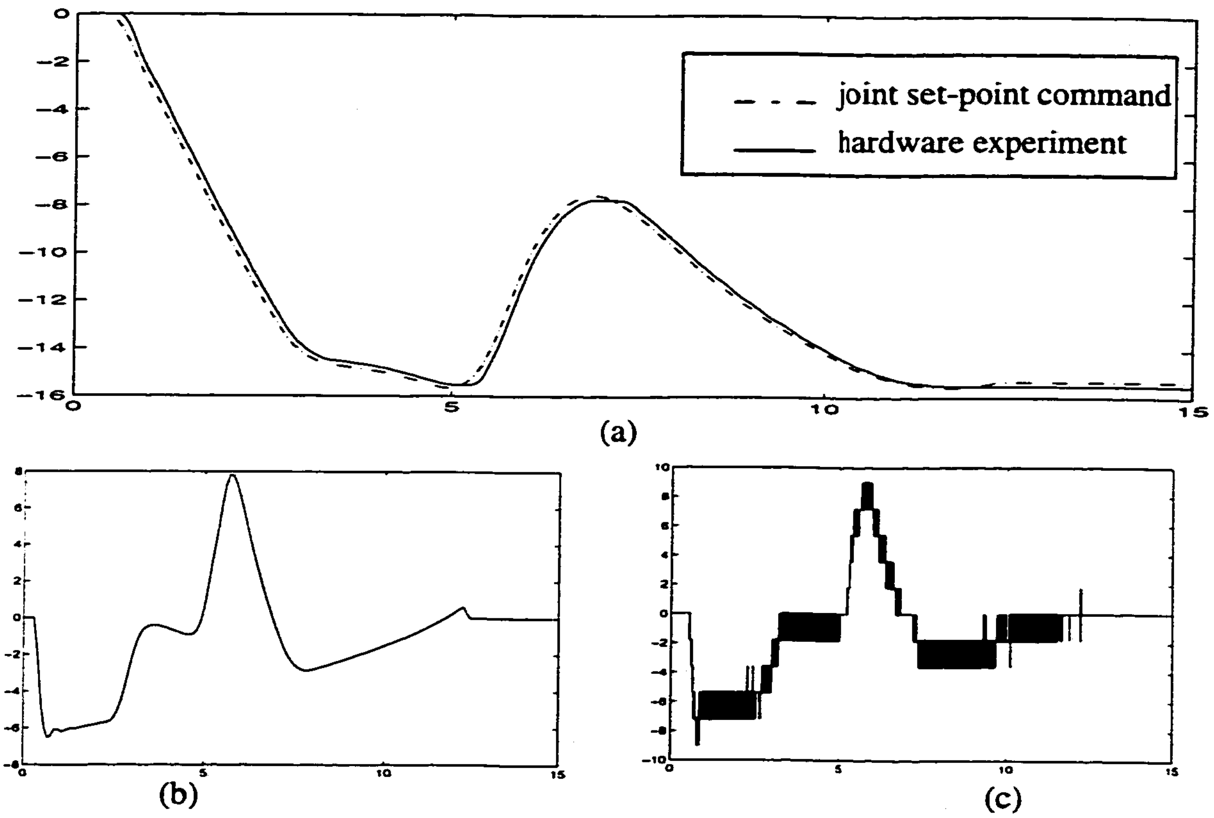
Three different scenarios were selected to verify the performance of the redundancy resolution and obstacle avoidance scheme in executing the following tasks: Position tracking, orientation tracking, static and moving obstacle collision avoidance, joint limit, and self-collision avoidance. In each of these scenarios, one or multiple features were active at different instants of execution. The sequence of steps undertaken in each case was as follows:

1. Generate the joint trajectory with the redundancy resolution and obstacle avoidance simulation.
2. Verify the result using MRS program (e.g., Are the obstacles avoided?).
3. Adjust parameters and repeat step 2 if necessary.
4. Position the static obstacles in the workspace.
5. Use the command trajectory to run the robot.
6. Record the joint history for further analysis

For demonstration purposes, the static obstacles were built using styrofoam and accurately positioned in the workspace. However, the moving object used in the second scenario was not constructed, instead, the performance of the collision avoidance algorithm was observed using the virtual models of the arm and the object in MRS.

3.4.3 Case 1: Collision Avoidance with Static Spherical Objects

In this scenario, the end-effector was commanded to move from its initial position to a final desired position: There were two static objects to be avoided in the workspace. The orientation tracking task was not activated in this scenario; the orientation of the end-effector was not controlled. Figure 3.22 shows the snapshots of the arm motion. We can see that without activating the obstacle avoidance feature (left sequence), the position trajectory is followed perfectly, but, there are several collisions with the obstacles. Figure 3.22 (right sequence) shows the successful operation of position tracking and obstacle avoidance (visualization of hardware experiment). As an example, the plot of the commanded and actual joint values and rates for joint one are given in Figure 3.20. The set-point command trajectory leads the actual joint trajectory by ~ 0.1 second which is a typical delay of a PID controller (Figure 3.20a). Figures 3.20b, c show the desired and actual rates respectively. One can see that the actual rates follow adequately the joint set-point command, except when the joint motion is dominated by stiction. The stiction effects also explain the position error at the end of the trajectory. Note that the PID controller only uses the rate information (obtained by numerically differentiating the measured joint angles) to provide damping. The oscillations shown in the PID rates are probably due to underdamped tuning of the PID parameters and noise due to numerical differentiation. This scenario demonstrates the capability of the redundancy resolution module to perform position tracking and to avoid collision with the obstacles.



(a)

(b)

(c)

Figure 3.20 Case 1: a) Joint 1 (degrees); b) derivative of the joint set-point command, c) derivative of joint trajectory in hardware experiment

3.4.4 Case 2: Collision Avoidance with a Moving Spherical Object

In the second scenario, the end-effector was commanded to keep its initial position while the orientation is changed. There was also a moving object to be avoided. In order to satisfy the main task, six degrees of freedom are required, leaving 1-DOF for other additional tasks. Figure 3.23, left sequence (simulation results), shows that without any obstacle avoidance, joint limit avoidance, and self-collision avoidance provisions, only the main task consisting of position and orientation tracking can be successfully executed. However, there are multiple collisions with objects and self-collision with the base. The right sequence of Figure 3.23 shows that by activating different modules both the main and additional tasks can be performed simultaneously (visualization of the hardware experiment). Figure 3.21 shows the actual joint angles for joints 2 and 3. The joints initially start moving to realize the commanded change of orientation, but this direction is reversed for joint 2, at 0.9 second, when the arm starts to take evasive action to prevent a collision. The joint 2 value rapidly increases to a peak value of ~ 30 degrees at 2 seconds. At 2.4 seconds, joint 2 quickly changes its direction to respect the imposed joint limit (software limit to prevent self-collision) of +35 degrees. One should note that there are more active additional tasks than the available degrees of redundancy. However, task prioritized formulation of redundancy resolution is capable of handling these difficult situations and leads only to a graceful performance degradation for the less prioritized tasks (in this case position and orientation tracking).

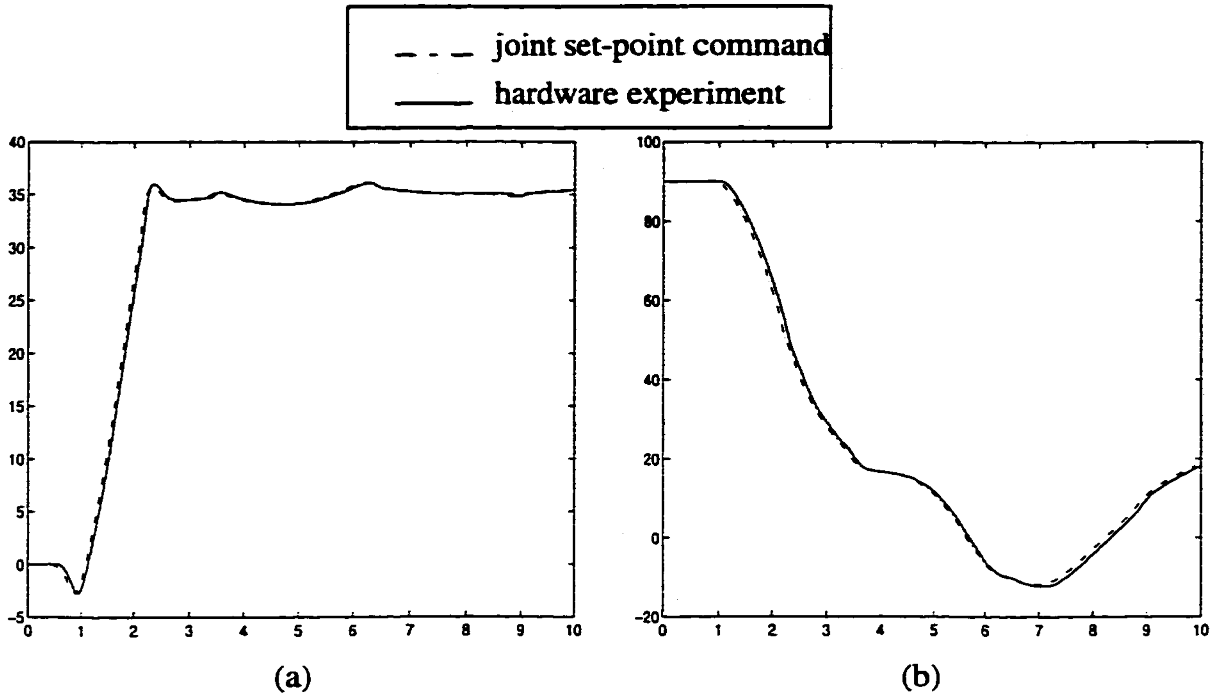


Figure 3.21 Case 2: a) joint 2, b) joint 3 (degrees)

3.4.5 Case 3: Passing Through a Triangular Opening

The environment was modeled by three cylindrical objects forming a triangular opening. The end-effector trajectory was defined as a straight line passing through this triangular opening. Each obstacle is enclosed in a cylindrical Surface of Influence (SOI). The left column in Figure 3.24 (a through g) shows the motion (simulation results) of the arm when the obstacle avoidance module is not activated. As we can see the end-effector follows the desired trajectory; however, there are multiple collisions between the links or the actuators with the obstacles. By activating the obstacle avoidance module, both the end-effector trajectory following and obstacle avoidance were achieved, as can be seen in the right column of Figure 3.24 (h through k) - visualization of the hardware experiment.

3.5 CONCLUSION

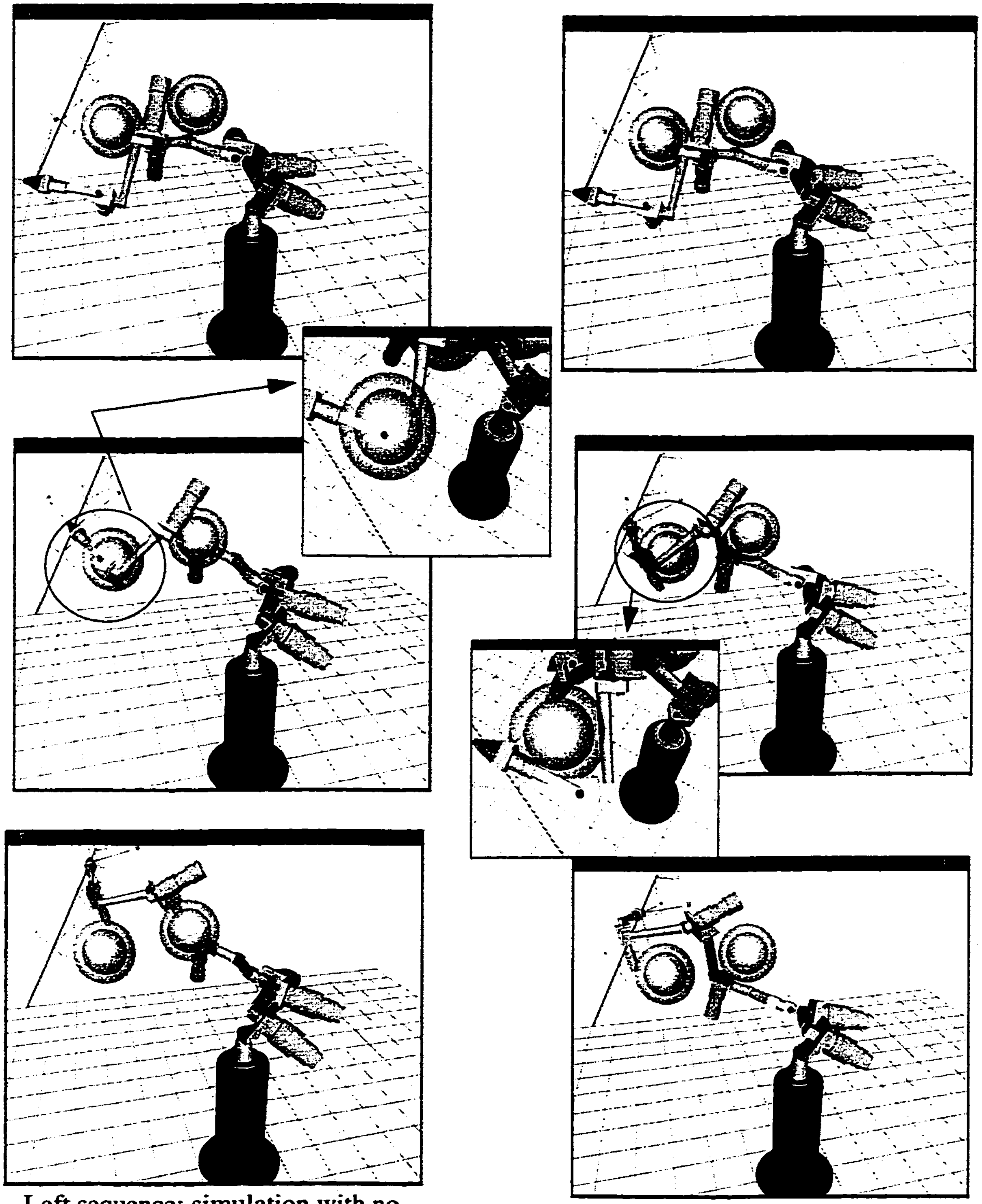
In this chapter, the extension of the redundancy resolution and obstacle avoidance module to the 3D workspace of REDIESTRO was addressed. The obstacle avoidance algorithm was modified to consider 3-D objects. A novel primitive based collision avoidance scheme was presented. This scheme is general, and provides realism, efficiency of computation, and economy in preserving the amount of free space that would otherwise be wasted. Different possible cases of collisions were considered. In particular, the cylinder-cylinder collision avoidance which represents a complex case for a collision detection scheme was formalized using the notion of dual vectors and angles.

Before performing the hardware experiments using REDIESTRO to evaluate the performance of the redundancy resolution and obstacle avoidance module, extensive simulations were performed using the kinematic model of REDIESTRO. These simulations were aimed at a study of the following issues:

- **Position and orientation tracking**: Considering the complexity of the singular regions existing in the 3D workspace of a 7-DOF manipulator, the singularity robustness formulation of redundancy was shown to be necessary in practical applications. It was shown that by a proper selection (or a time-varying formulation) of W_v, the weighting matrix of the singularity robustness task, the effect of this term on the tracking performance can be minimized.
- **Performing the additional task(s)**: Joint limit avoidance and obstacle avoidance tasks were implemented for REDIESTRO. It was shown that the formulation of additional tasks as inequality constraints, may result in rapid change in joint velocities which causing a large pulse in joint accelerations. In a practical implementation, since the maximum acceleration of each joint would be limited, such a commanded joint acceleration would result in saturation of the actuators. A time-varying formulation of the weighting matrix, W_c, was proposed which successfully overcame this problem.
- **Fine tuning of control gains and weighting matrices**

Three scenarios encompassing most of the developed redundancy resolution and obstacle avoidance system features have been successfully demonstrated on real hardware, i.e., the REDIESTRO manipulator. Despite the geometrical complexity of REDIESTRO, the arm is entirely modelled by decomposition of the links and attached actuators into sub-links modelled by simple volume primitives. Moreover, due to the complex and unusual shape of REDIESTRO, it is believed that adapting the algorithms to other manipulators can only be simpler.

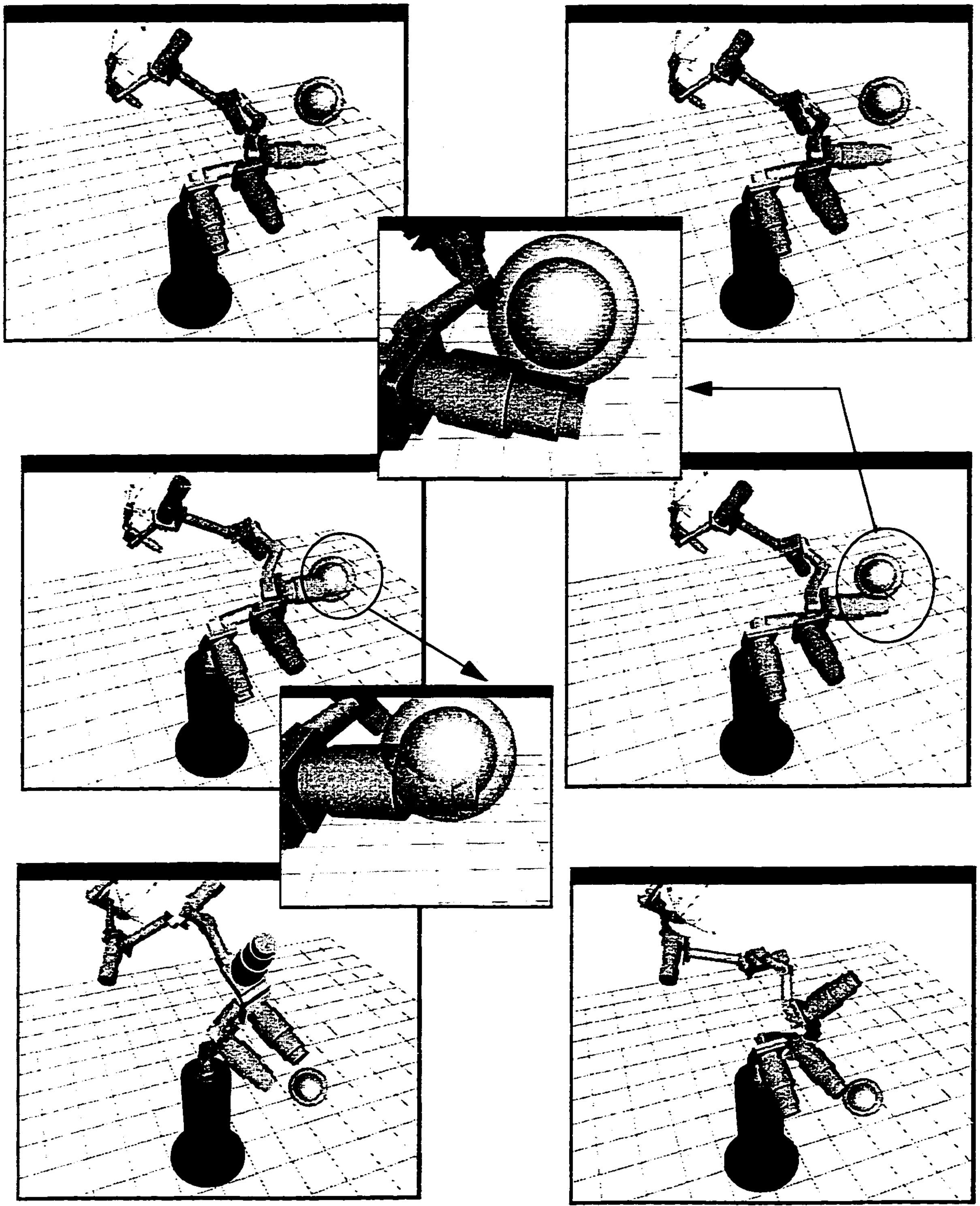
The current redundancy resolution and obstacle avoidance scheme provides an intelligently assisted tele-operation mode to the human operator in that one only needs to specify the desired location and orientation of the end-effector, and the system automatically takes care of the details of motion control, configuration selection, and generalized collision avoidance, including joint limi and self-collision avoidance, in addition to collision with objects in the workspace. However, at this stage the redundancy resolution scheme cannot handle situations where the manipulator comes in contact with environment. Further modification to the redundancy resolution scheme is needed in order for it to be used in a force or compliant control scheme. This issue will be addressed in the next chapter.



Left sequence: simulation with no obstacle avoidance provision

Right sequence: Visualization of hardware experiment

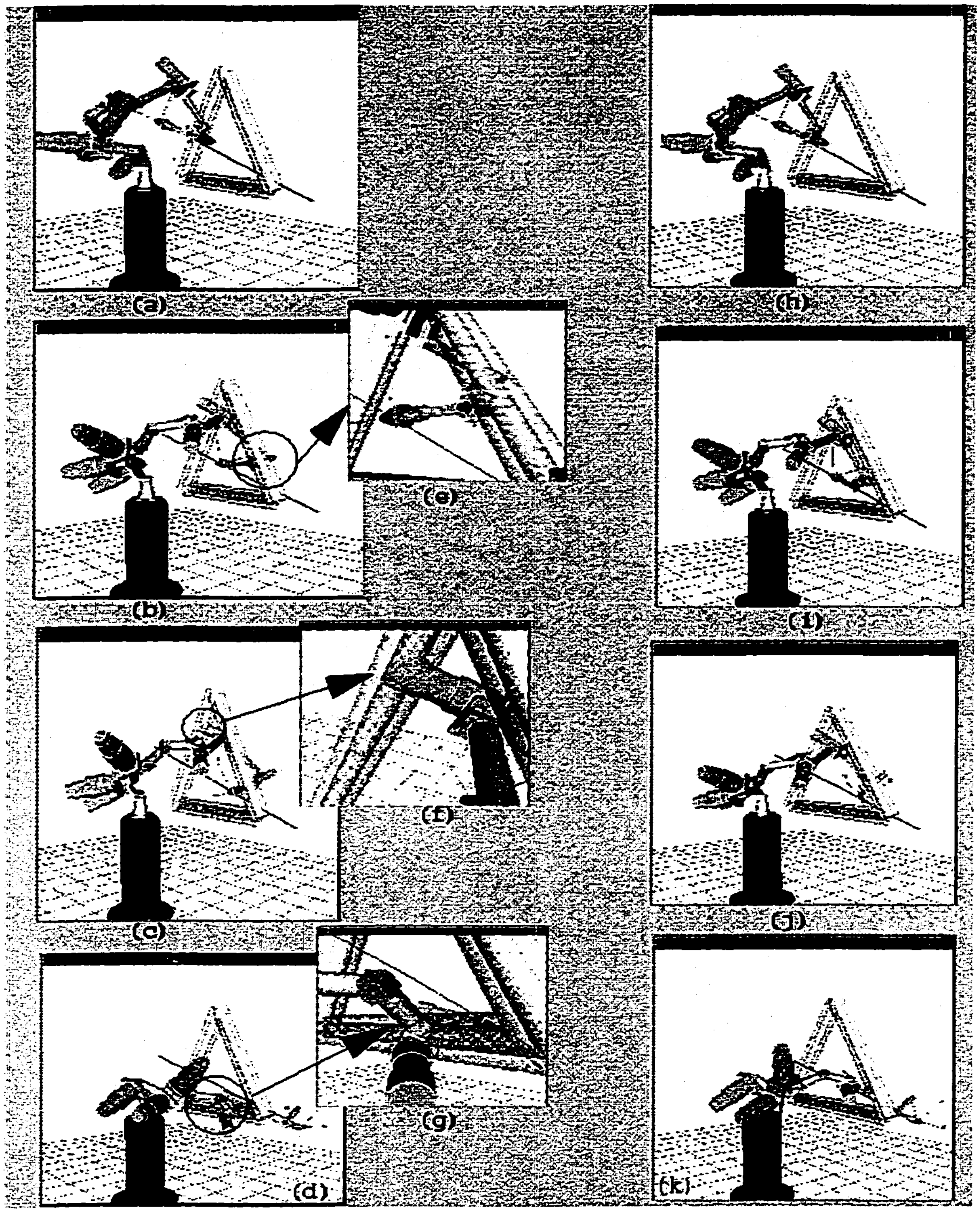
Figure 3.22 Collision avoidance with static spherical objects



Left (top to bottom): simulation with obstacle avoidance (MOCA) inactive

Right: Visualization of hardware experiment.

Figure 3.23 Collision Avoidance with moving spherical object.



Left sequence: Simulation with obstacle avoidance inactive

Right sequence: Visualization of the hardware demonstration with obstacle avoidance active

Figure 3.24 Passing through a triangular opening

CHAPTER 4

CONTACT FORCE AND COMPLIANT MOTION CONTROL

4.1 INTRODUCTION

Robotic tasks mainly fall into two categories: Constrained and unconstrained motions. During the initial stages of development in robotics, most successful applications dealt with position control of unconstrained motion of robot manipulators. The nature of these tasks does not require a robot to come in contact with its environment (work piece). Spray painting is an example of such a task in which the robot brings a spray gun near the surface to be painted and then sweep across the surface with a specified velocity. Another example is that of seam welding. In some applications where a robot comes in contact with its environment (as in the case of material handling), precise control of the interaction with the object is not required. The problem that arises when using a position control scheme in a constrained motion is that the robot-environment interaction forces are treated as disturbances. The controller tries to reject these forces, and hence, gives rise to larger interaction forces. The consequences of this are saturation, instability, or even physical failure and damage to the robot and the environment. Whitney [20] gives a historical perspective on robot force control. Force control strategies have been mainly designed to use force feedback sensory information.

Salisbury [21] proposed a stiffness control scheme. Raibert and Craig [22] proposed a hybrid position-force control scheme. Yoshikawa [23], McClamroch and Wang [24] proposed a method based on a constrained dynamic model of a manipulator. Hogan introduced the impedance control idea in a series of papers in the mid-1980's. In [25], he proposed the fundamental theory of impedance control which showed that command and control of a vector such as position or force is not enough to control the dynamic interactions between a manipulator and its environment. This emphasizes the main problem of hybrid position-force control, i.e., its failure to recognize the importance of manipulator impedance. The impedance control scheme overcomes this problem, but it ignores the distinction between position and force controlled subspaces, and no attempt is made to follow a commanded force trajectory. Therefore, Anderson and Spong [26] proposed a *Hybrid Impedance Control* (HIC) scheme, and recently, Liu and Goldenberg [27] have introduced a robust HIC method.

The aforementioned methods can be divided into two main categories, referred to as constrained motion [22], [23], [24], and compliant motion [25], [26], [28] approaches. In the next sections, an outline of these approaches is given. Note that the above mentioned algorithms are not directly applicable to redundant manipulators. However, a careful review of these algorithms gives guidelines in selection of force or compliant motion control for redundant manipulators. In the remainder of this chapter, algorithms proposed for force and compliant motion control of redundant manipulators are presented. Section 4.3.1 addresses the extension of configuration control at the acceleration level. Section 4.3.2 introduces the Augmented Hybrid Impedance Control (AHIC) scheme. The feasibility of this scheme with respect to performing both the main and additional tasks is studied using a 3-DOF planar arm. The AHIC scheme is then modified to cop with the uncontrolled self-motion. The AHIC with self-motion stabilization is presented in 4.3.3. An adaptive version of the AHIC scheme is presented in Section 4.3.4 along with a stability and convergence proof.

4.2 LITERATURE REVIEW

4.2.1 Constrained Motion Approach

This approach considers the control of a manipulator constrained by a rigid object[1] in its environment. If the environment imposes purely kinematic constraints on the end-effector motion, only a static balance of forces and torques occurs (assuming that the friction effects are neglected). This implies no energy transfer or dissipation between the manipulator and the environment. This underlies the main modeling assumption made by [24] where an algebraic vector equation restricts the feasible end-effector poses. The constrained dynamics can be formalized as:

$$H(q)\ddot{q} + h(q, \dot{q}) = \tau - J^T F \qquad \text{(a)} \qquad (4.2.1)$$

$$\Phi(p) = 0 \qquad \text{(b)}$$

where τ is the vector of applied forces (torques), $H(q)$ is the $n \times n$ symmetric positive definite inertia matrix, h is the vector of centrifugal, Coriolis, and gravitational torques. $p \in R^n$ is the generalized task coordinates, and $\Phi(p) \in R^m$ is the constraint equation, continuously differentiable with respect to p. It is assumed that the Jacobian matrix is square and of full rank. The analysis given below follows that in [24], the generalized force[2] F in (4.2.1) is given by:

$$F = \left(\frac{\partial \Phi(p)}{\partial p}\right)^T \lambda \qquad (4.2.2)$$

where $\lambda \in R^{m \times 1}$ is the vector of generalized Lagrange multipliers. Using the forward kinematic relations:

[1] *A work environment or object is said to be rigid when it does not deform as a result of application of generalized forces by the manipulator.*

[2] *In the rest of this chapter, the term "force" refers to both interaction force and torque.*

$$\dot{p} = J\dot{q}$$

$$\ddot{p} = J\ddot{q} + \dot{J}\dot{q} \tag{4.2.3}$$

and assuming that the Jacobian matrix is invertible, we can obtain the following constrained dynamics expressed with respect to generalized task coordinates directly from (4.2.1):

$$H_p(p)\ddot{p} + h_p(p, \dot{p}) = u - F \tag{4.2.4}$$

$$\Phi(p) = 0$$

where

$$\begin{aligned} H_p &= J^{-T}H(q)J^{-1} \\ h_p &= -H_p\dot{J}\dot{q} + J^{-T}h(q, \dot{q}) \\ u &= J^{-T}\tau \end{aligned} \tag{4.2.5}$$

A nonlinear transformation can then be used to transfer to a new coordinate frame. It is assumed that there is an open set $\Theta \subset R^{n-m}$ and a function Ω such that

$$\Phi(\Omega(p_2), p_2) = 0 \qquad \forall p_2 \in \Theta \tag{4.2.6}$$

where

$$p = \begin{pmatrix} p_{1_{m \times 1}} \\ p_{2_{(n-m) \times 1}} \end{pmatrix} \tag{4.2.7}$$

Now, defining another coordinate represented by the vector x, we obtain the following nonlinear transformation X:

$$x = X(p) = \begin{pmatrix} p_1 - \Omega(p_2) \\ p_2 \end{pmatrix}$$

which is differentiable and has a differentiable inverse given by:

$$p = Q(x) = \begin{pmatrix} x_1 + \Omega(x_2) \\ x_2 \end{pmatrix} \tag{4.2.8}$$

where x is partitioned conformably with (4.2.7). The Jacobian of (4.2.8) is defined by:

$$T(x) = \frac{\partial Q(x)}{\partial x} = \begin{bmatrix} I_m & \dfrac{\partial \Omega(x_2)}{x_2} \\ 0 & I_{n-m} \end{bmatrix} \tag{4.2.9}$$

Transforming the equation of motion in (4.2.4) to the generalized coordinate x, we obtain:

$$\begin{aligned} H_x(x)\ddot{x} + h_x(x, \dot{x}) &= T^T u - T^T F \\ x_1 &= 0 \end{aligned} \tag{4.2.10}$$

where

$$H_x = T^T(x)H_p(Q(x))T(x)$$

$$h_x = T^T(x)H_p(Q(x))\dot{T}(x)\dot{x} + T^T(x)h_p(Q(x), T(x)\dot{x}) \tag{4.2.11}$$

Note that in this transformed frame, the constraint equation takes the simple form $x_1 = 0$. Equations (4.2.10) can be simplified as follows:

$$E_1H_xE_2^T\ddot{x}_2 + E_1h_x = E_1T^T(u-F) \tag{4.2.12}$$

$$E_2H_xE_2^T\ddot{x}_2 + E_2h_x = E_2T^Tu$$

$$x_1 = 0$$

where E_1 and E_2 are defined by

$$I_n = [E^T{}_1, E^T{}_2] \tag{4.2.13}$$

$$E^T{}_1 = \begin{pmatrix} I_m \\ 0 \end{pmatrix} \qquad E^T{}_2 = \begin{pmatrix} 0 \\ I_{n-m} \end{pmatrix}$$

The hybrid control law is defined as

$$T^Tu = u_x + u_f \tag{4.2.14}$$

where

$$u_x = H_x[0 \quad E_2^T][\ddot{x}_d + K_v(\dot{x}_d - \dot{x}) + K_p(x_d - x)] + h_x(x, \dot{x}) \tag{4.2.15}$$

$$u_f = [E_1^T \quad 0][T^T F_d + G_F T^T(F_d - F)]$$

where K_p, K_v, and G_F are feedback gain matrices. By replacing the control law (4.2.14) in the equation of motion (4.2.12), the following closed-form system of equations is obtained

$$\begin{aligned} E_1 H_x E_2^T(\ddot{e}_2 + K_v\dot{e}_2 + K_p e_2) &= (I_m + G_F)E_1 T^T(F_d - F) \\ E_1 H_x E_2^T(\ddot{e}_2 + K_v\dot{e}_2 + K_p e_2) &= 0 \\ e_1 &= 0 \end{aligned} \tag{4.2.16}$$

where $e_1 = x_1 - x_{1d}$ and $e_2 = x_2 - x_{2d}$. The closed-loop equation of motion given by (4.2.16) implies that $e_2 \rightarrow 0$ as $t \rightarrow \infty$ through proper choice of feedback gains and also $F \rightarrow F_d$ as $t \rightarrow \infty$. Hence, it is asymptotically stable.

A hybrid position and force controller is proposed in [22] where the task space is divided into two orthogonal position controlled and force-controlled subspaces using a selection matrix S. The diagonal elements of the selection matrix S are selected as 0 or 1 depending on which degrees of freedom are force-controlled and which are position-controlled (Figure 4.1).

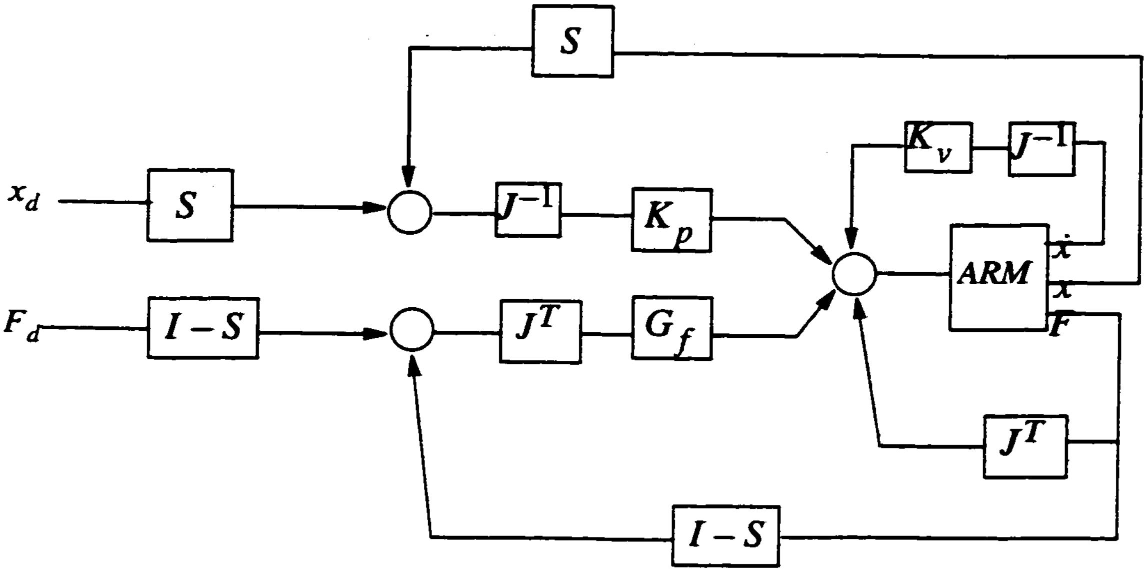


Figure 4.1 Schematic diagram of the hybrid position and force control

Mills [29] showed that the constrained motion control approach of McClamroch and Wang [24] is identical to the hybrid position and force control scheme if the selection matrix S is replaced by:

$$S = [0 \quad E_2^T]$$
$$I - S = [E_1^T \quad 0] \tag{4.2.17}$$

Note that these methods are not directly applicable to redundant manipulator.

4.2.2 Compliant Motion Control

In contrast to the constrained motion approach, compliant motion control as its name implies, deals with a compliant environment. This approach is aimed at developing a relationship between interaction forces and manipulator position instead of controlling position and force independently. This approach is limited by the assumption of small deformations of the environment, with no relative motion allowed in coupling. Salisbury [21] proposed the stiffness control method. The objective is to provide a stabilizing dynamic compensator for the system such that the relationship between the position of the closed-loop system and the interaction forces is constant over a given operating frequency range. This can be written mathematically as follows:

$$\delta F(j\omega) = K\delta X(j\omega), \qquad 0 < \omega < \omega^o \tag{4.2.18}$$

where $\delta F(j\omega)$ is the $n \times 1$ vector of deviations of the interaction forces and torques from their equilibrium values in a global Cartesian coordinate frame; $\delta X(j\omega)$ is the $n \times 1$ vector of deviations of the positions and orientations of the end-effector from their equilibrium values in a global Cartesian coordinate frame; K is the $n \times n$ real-valued nonsingular stiffness matrix; and ω^o is the bandwidth of operation. By defining K, the user governs the behavior of the system during constrained maneuvers.

Hogan [25] proposed the impedance control idea. Impedance control is closely related to stiffness control. However, stiffness is merely the static component of a robot's output impedance. Impedance control goes further and attempts to modulate the dynamics of the robot's interactive behavior. The main idea of impedance control is to make the manipulator act as a mass-spring-dashpot system in each single degree of freedom in its workspace.

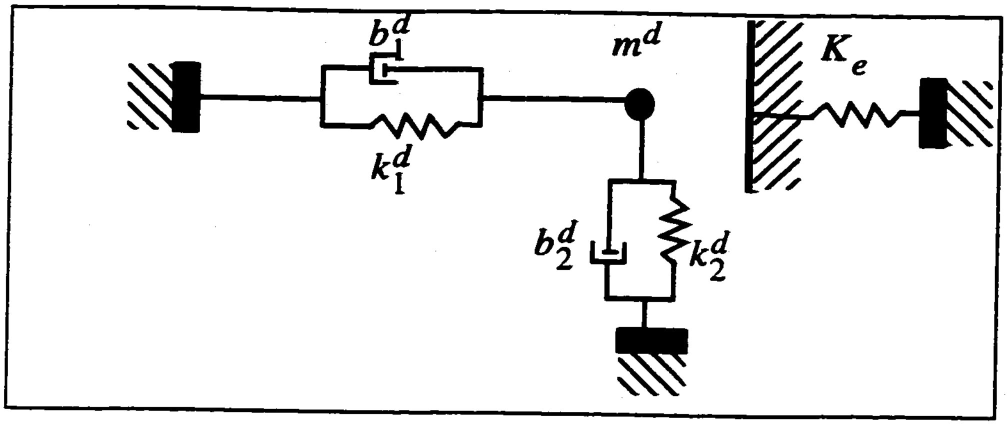


Figure 4.2 Apparent impedance of a manipulator in each degree of freedom of task space

Therefore, the manipulator is seen as an apparent impedance given by:

$$M^d(\ddot{X}-\ddot{X}^d)+B^d(\dot{X}-\dot{X}^d)+K^d(X-X^d) = -F^e \tag{4.2.19}$$

where M^d, B^d, and K^d are diagonal $m \times m$ matrices of the desired mass, damping, and stiffness. F^e is the vector of the environmental reaction forces. The superscript d refers to desired values.

First, let us define the operational-space dynamic equation of motion of the manipulator[1] as:

$$H_x(X)\ddot{X}+h_x(X,\dot{X}) = J^{-T}u+F^e \tag{4.2.20}$$

where H_x is the Cartesian inertia matrix, h_x is the vector of centrifugal, Coriolis, and gravity terms acting in operational space. Then as proposed in [26], an inner and outer loop control strategy (Figure 4.3) can be used to achieve the closed-loop dynamics specified by (4.2.19)

[1] *If we consider a non-redundant manipulator not in a singular configuration, then*

$$H_x = J^{-T}H_qJ^{-1}, \qquad h_x = J^{-T}h_q-H_x\dot{J}\dot{q}$$

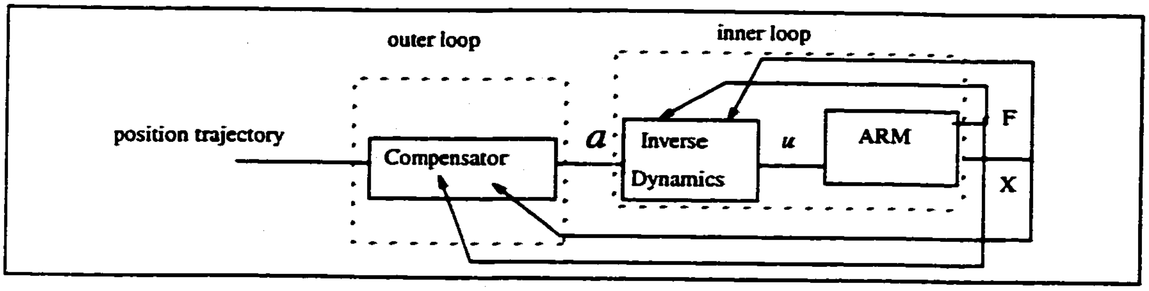


Figure 4.3 Inner-outer loop control strategy

In the absence of uncertainties on the dynamic parameters of the manipulator, the inner loop is a feedback linearization loop of the form

$$u = J^T(H_x a + h_x - F^e) \tag{4.2.21}$$

which results in the double integrator system $\ddot{X} = a$. The output of the outer loop is a target acceleration obtained by solving (4.2.19):

$$a = \ddot{X}^d - M^{d^{-1}}[B^d(\dot{X} - \dot{X}^d) + K^d(X - X^d) - F^e] \tag{4.2.22}$$

Hogan indicated that the impedance control scheme is capable of controlling the manipulator in both free space and constrained maneuvers, though eliminating the switching between free-motion and constrained motion controllers.

A typical compliant motion task is the surface cleaning scenario shown in Figure 4.4. As we can see a target trajectory is defined to be identical to the desired trajectory in free motion. However, in order to maintain contact with the environment, the target trajectory is defined to be different from the desired trajectory in constrained maneuvers. Depending on the desired impedance characteristics and the environment, the robot will follow an actual path which results in a certain contact force with the environment.

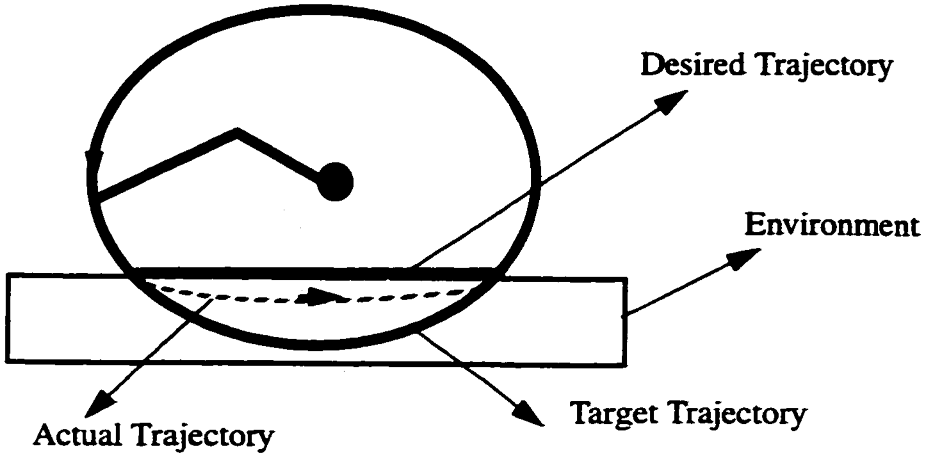


Figure 4.4 Surface cleaning using impedance controller

It should be noted that in the impedance control scheme, no attempt is made to follow a commanded force trajectory. To overcome this problem, Anderson and Spong [26] proposed a Hybrid Impedance Control (HIC) method. Again the task space is split into orthogonal position and force controlled subspaces using the selection matrix *S*. The desired equation of motion in the position-controlled subspace is identical to equation (4.2.19). However, in the force-controlled subspace, the desired impedance is defined by:

$$M^d\ddot{X} + B^d\dot{X} - F^d = -F^e \tag{4.2.23}$$

In the force-controlled subspace, a desired inertia and damping have been introduced because if only a simple proportional force feedback were applied, the response could be very under-damped for an environment with high stiffness. In the case of loss of contact with the environment or approaching the surface ($F^e = 0$), equation (4.2.23)becomes

$$M^d\ddot{X} + B^d\dot{X} = F^d \tag{4.2.24}$$

If we assume a constant desired force, positive diagonal inertia and damping matrices, and $\dot{X}(0) = 0$, then the *ith* component of the velocity vector $\dot{X}$ is given by:

$$\dot{X}_i(t) = \frac{F_i^d}{B_i^d}(1 - e^{-(B_i^d/M_i^d)t}) \qquad (4.2.25)$$

Therefore

$$|\dot{X}_i(t)| < \frac{F_i^d}{B_i^d} \qquad \textit{and} \qquad \lim_{t \to \infty} \dot{X}_i(t) = \frac{F_i^d}{B_i^d} \qquad (4.2.26)$$

This guarantees that the arm approaches the environment with a velocity that can be properly limited in order to reduce impact forces.

Again, note that these methods are not directly applicable to redundant manipulators. The main reasons are the use of the Cartesian model of manipulator dynamics, and calculation of the command input in task space. As we mentioned earlier, for a redundant manipulator, the task space requirements cannot uniquely determine joint space configurations. An augmented hybrid impedance controller which overcomes this problem will be proposed in next section.

4.3 NEW SCHEMES FOR COMPLIANT AND FORCE CONTROL OF REDUNDANT MANIPULATORS

The problem of compliant motion control of redundant manipulators has not attained the maturity level of its non-redundant counterpart. There is little work that addresses the problem of redundancy resolution in a compliant motion control scheme. There are two major issues to be addressed in extending existing compliant motion schemes to the case of redundant manipulators:

(i) The nature of compliant motion control requires expressing the manipulator's task in Cartesian space; therefore, such schemes are usually based on the Cartesian dynamic model of manipulator. However, in the presence of redundancy, there is not a unique map from Cartesian space to joint space.

(ii) Most redundancy resolution techniques are at the velocity level, and simple extensions of these techniques to the acceleration level have resulted in the self-motion phenomenon.

For instance, Gertz et al. [36], Walker [35] and Lin et al. [37] have used a generalized inertia-weighted inverse of the Jacobian to resolve redundancy in order to reduce impact force. However, these schemes are single purpose algorithms, and cannot be used to satisfy additional criteria. An extended impedance control method is discussed in [38] and [39]; the former also includes an HIC scheme. These schemes can be considered as multipurpose algorithms since different additional tasks can be incorporated in HIC without modifying the schemes and the control laws. However, there are two major drawbacks to these schemes: (i) The dimension of the additional task should be equal to the degree of redundancy, which makes the approach not applicable for a wide class of additional tasks, i.e., additional tasks that are not active for all time such as obstacle avoidance in a cluttered environment. (ii) The HIC scheme introduces the possibility of controlling tasks either by a position control or a force control scheme. The possibility of having an additional task controlled by a force controlled scheme is ignored by including the additional task in the position controlled subspace of the extended task. Shadpey et al. [40] have proposed an Augmented Hybrid Impedance Control (AHIC) scheme to overcome these problems (see Section 4.3.2). This scheme enjoys the following major advantages:

(i) Different additional tasks can be easily incorporated in the AHIC scheme without modifying the scheme and the control law.

(ii) An additional task can be included in the force-controlled subspace of the augmented task. Therefore, it is possible to have a multiple-point force control scheme.

(iii) Task priority and singularity robustness formulation of the AHIC scheme relaxes the restrictive assumption of having a non-singular augmented Jacobian.

However, the scheme in [40] exhibits the self-motion phenomenon, i.e., motion of the arm in the null space of the Jacobian. A new AHIC scheme which has the above mentioned characteristics [42] is presented in Section 4.3.3. Moreover, by modifying both the inner and outer control loops, the self-motion is damped when the dimension of the augmented task is smaller than that of the joint space. Finally, the new scheme is more amenable to an adaptive implementation. An adaptive version of the AHIC scheme [43] is presented in Section 4.3.4.

4.3.1 Configuration Control at the Acceleration Level

Similar to the pseudo-inverse solution given by (2.3.30), the following weighted damped least-squares solution can be obtained:

$$\ddot{q} = [J_e^T W_e J_e + J_c^T W_c J_c + W_v]^{-1}[J_e^T W_e(\ddot{X} - \dot{J}_e\dot{q}) + J_c^T W_c(\ddot{Z} - \dot{J}_c\dot{q})] \qquad (4.3.1)$$

This minimizes the following cost function:

$$L = \ddot{E}_e^T W_e \ddot{E}_e + \ddot{E}_c^T W_c \ddot{E}_c + \ddot{q}^T W_v \ddot{q} \qquad (4.3.2)$$

where

$$\ddot{E}_e = \ddot{X}^d - (\ddot{X} - \dot{J}_e\dot{q}) \qquad and \qquad \ddot{E}_c = \ddot{Z}^d - (\ddot{Z} - \dot{J}_c\dot{q}) \qquad (4.3.3)$$

However, this solution is incapable of controlling the null space component of joint velocities (see Section 2.3.2). A remedy for this difficulty is to differentiate the configuration control solution at the velocity level given by equation (2.3.19). This yields

$$\ddot{q} = [J_e^T W_e J_e + J_c^T W_c J_c + W_v]^{-1}[A + B] \qquad (4.3.4)$$

where

$$A = J_e^T W_e(\ddot{X} - \dot{J}_e\dot{q}) + J_c^T W_c(\ddot{Z} - \dot{J}_c\dot{q})$$

$$B = \dot{J}_e^T W_e(\dot{X} - J_e\dot{q}) + \dot{J}_c^T W_c(\dot{Z} - J_c\dot{q})$$

Therefore, following the reference joint velocity given by equation (2.3.19) and the acceleration trajectory given by (4.3.4), gives a special solution that minimizes the joint velocities when $k < r$, i.e., there are not as many active tasks as the degree-of-redundancy, and the best solution in the least-squares sense when $k > r$. In all cases the presence of W_v ensures the boundedness of the joint velocities.

4.3.2 Augmented Hybrid Impedance Control using Computed Torque Algorithm

The *AHIC* scheme, shown in Figure 4.5, can be broken down into two different control loops. The outer loop generates an *Augmented Cartesian Target Acceleration (ACTA)* trajectory reflecting the desired impedance in the position controlled subspaces, and the desired force in the force controlled subspaces of the main and additional tasks. From this point of view, the AHIC problem can be formulated as that of tracking an ACTA trajectory which is generated in real time. The inner loop control problem consists of selecting an input τ to the actuators which makes the end-effector track a desired trajectory generated by the outer loop.

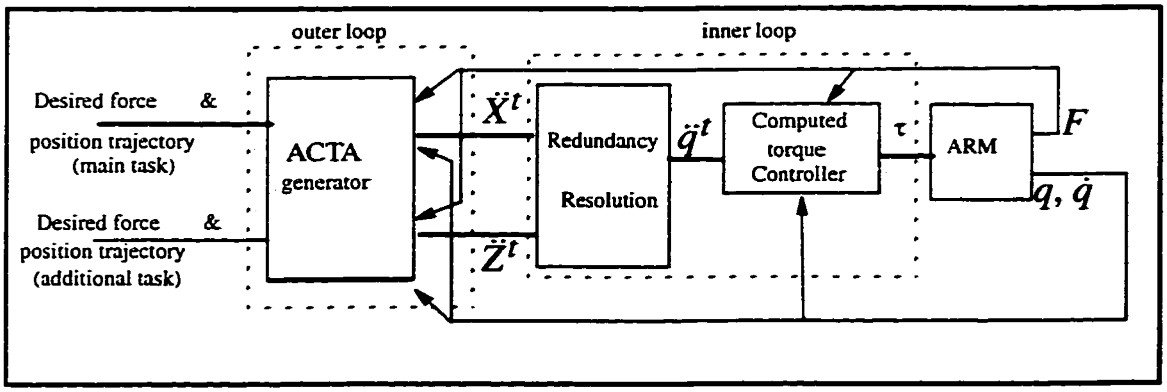


Figure 4.5 Block diagram of the AHIC scheme using the computed torque controller

4.3.2.1 Outer-loop design

The design of the outer-loop part of the AHIC scheme is described in this section. As mentioned in Section 4.2, the idea is to split the spaces corresponding to the main task X and additional task Z into position and force controlled subspaces. Impedance control is used in the position-controlled subspace. Therefore, the objective is to make the manipulator act as a mass-spring-dashpot system with desired inertia, damping, and stiffness in each dimension of the position controlled subspace of the main and additional tasks. In the force-controlled subspace, a desired inertia and damping have been introduced because, if only a simple proportional force feedback were applied, the response could be very underdamped for an environment with high stiffness.

The motion of the manipulator in both subspaces can be expressed by a single matrix equation using selection matrices S_x and S_z, as follows:

$$M_x^d(\ddot{X} - S_x\ddot{X}^d) + B_x^d(\dot{X} - S_x\dot{X}^d) + K_x^d S_x(X - X^d) - (I - S_x)F_x^d = -F_x^e \quad \text{(a)} \qquad (4.3.5)$$

$$M_z^d(\ddot{Z} - S_z\ddot{Z}^d) + B_z^d(\dot{Z} - S_z\dot{Z}^d) + K_z^d S_z(Z - Z^d) - (I - S_z)F_Z^d = -F_z^e \quad \text{(b)}$$

where the superscript d denotes the desired values; the subscripts x and z refer to the main and additional tasks respectively; the diagonal matrices M, B, and K are the desired mass, damping, and stiffness matrices; F^d and $-F^e$ are vectors of the desired and the environmental reaction forces; and S_x and S_z are the diagonal selection matrices which have 1's on the diagonal for position-controlled subspaces and 0's for the force-controlled subspaces.

Solving the motion equations (4.3.5) for the accelerations $\ddot{X}$ and $\ddot{Z}$ leads to the *Cartesian Target Acceleration* (*CTA)* trajectories of the main, $\ddot{X}^t$, and additional tasks, $\ddot{Z}^t$:

$$\ddot{X}^t = S_x\ddot{X}^d - (M_x^d)^{-1}[B_x^d(\dot{X} - S_x\dot{X}^d) + K_x^d S_x(X - X^d) - (I - S_x)F_x^d + F_x] \qquad \text{(a)} \quad (4.3.6)$$

$$\ddot{Z}^t = S_z\ddot{Z}^d - (M_z^d)^{-1}[B_z^d(\dot{Z} - S_z\dot{Z}^d) + K_z^d S_z(Z - Z^d) - (I - S_z)F_Z^d + F_z^e] \qquad \text{(b)}$$

Now the AHIC scheme can be formulated to track the *CTA* trajectories. Using the configuration control approach - equation (4.3.1), the desired *Joint Target Acceleration (JTA)* trajectory ($\ddot{q}^t$) can be found by replacing the *CTA* trajectories of the main and additional tasks in Equation (4.3.1).

$$\ddot{q}^t = [J_e^T W_e J_e + J_c^T W_c J_c + W_v]^{-1}[J_e^T W_e(\ddot{X}^t - \dot{J}_e\dot{q}) + J_c^T W_c(\ddot{Z}^t - \dot{J}_c\dot{q})] \qquad (4.3.7)$$

Remark: Duffy [44] has indicated that in the case of compliant motion of a manipulator in 3D space, the end-effector velocities (linear, angular) and forces (forces, torques) should be considered as screws represented in axis and ray coordinates. Therefore, in general the concept of orthogonality of force and position controlled subspaces is not valid. As shown in [45], the concept that is appropriate is that of "reciprocal" subspaces, i.e., the set of motion screws should be decomposed into mutually reciprocal free and constraint subspaces.

4.3.2.2 Inner-loop

The dynamics of a rigid manipulator in the absence of disturbances are described by:

$$H(q)\ddot{q} + h(q, \dot{q}) + G(q) + f(\dot{q}) = \tau + J_e^T F_x^e + J_{c1}^T F_z^e \qquad (4.3.8)$$

where τ is the vector of applied forces (torques), *H(q)* is the $n \times n$ symmetric positive definite inertia matrix, *h* is the vector of centrifugal and Coriolis forces, *f* is the vector of frictional forces, and *G* is the vector of gravitational forces. The last term on the right-hand side of the equation is only needed if another point of the manipulator (other than the end-effector) is in contact with the environment; F_z^e denotes the reaction force corresponding to a second constraint surface, and J_{cl} is the Jacobian of the contact point.

As mentioned earlier, the responsibility of the inner loop is to ensure that the manipulator tracks the JTA trajectory. Referring to the dynamic equation of the manipulator, the input torque is selected by:

$$\tau = H\ddot{q}^t + h(q, \dot{q}) + G(q) + f(\dot{q}) - J_e^T F_x^e - J_{c1}^T F_z^e \qquad (4.3.9)$$

which guarantees perfect following of the JTA trajectory in the absence of uncertainties in the manipulator's parameters.

4.3.2.3 Simulation Results on 3-DOF Planar Arm

The performance of the AHIC scheme has been studied using simulations involving a 3-revolute-joint planar manipulator (Figure 4.6).

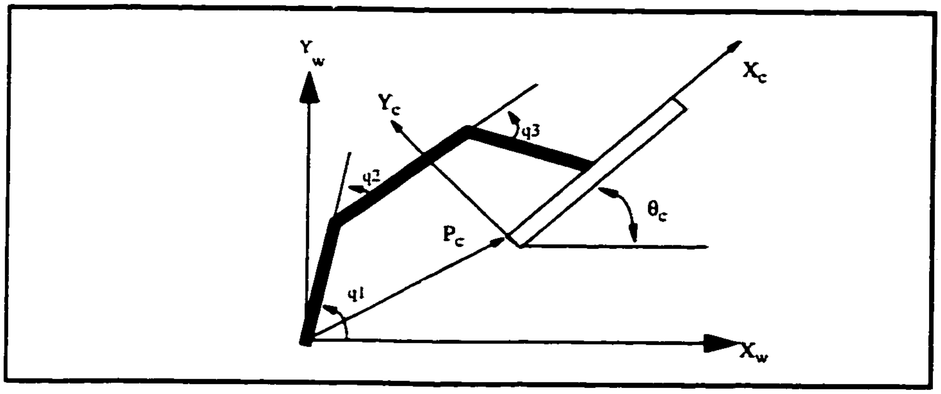


Figure 4.6 3-DOF planar manipulator used in the simulation

In all cases, a constraint surface is defined by the position P_c and orientation θ_c of a frame C attached to this surface.The main task (same for all cases), defined in the constraint frame, is specified by a desired impedance (inertia, damping, and stiffness) in tracking of the desired position trajectory in the X_c direction, and desired force trajectory in the Y_c direction. The selected values for the simulations are: m^d=*1*, b^d=*120*, and k^d=3600. The environment is modelled as a spring with stiffness K_e=*10000 N/m*. The desired position trajectory is calculated by linear interpolation between the initial and final points (constant velocity trajectory), and the force trajectory is defined by f^d= *-100 N*. For each individual case, a different additional task is defined. A general block diagram of the simulation is shown in Figure 4.7.

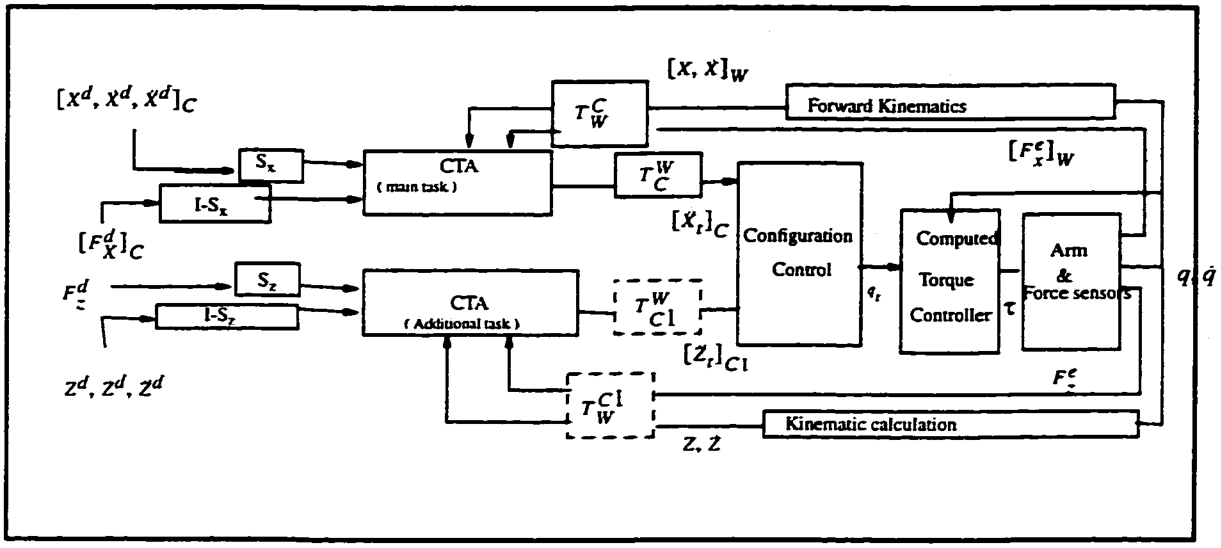
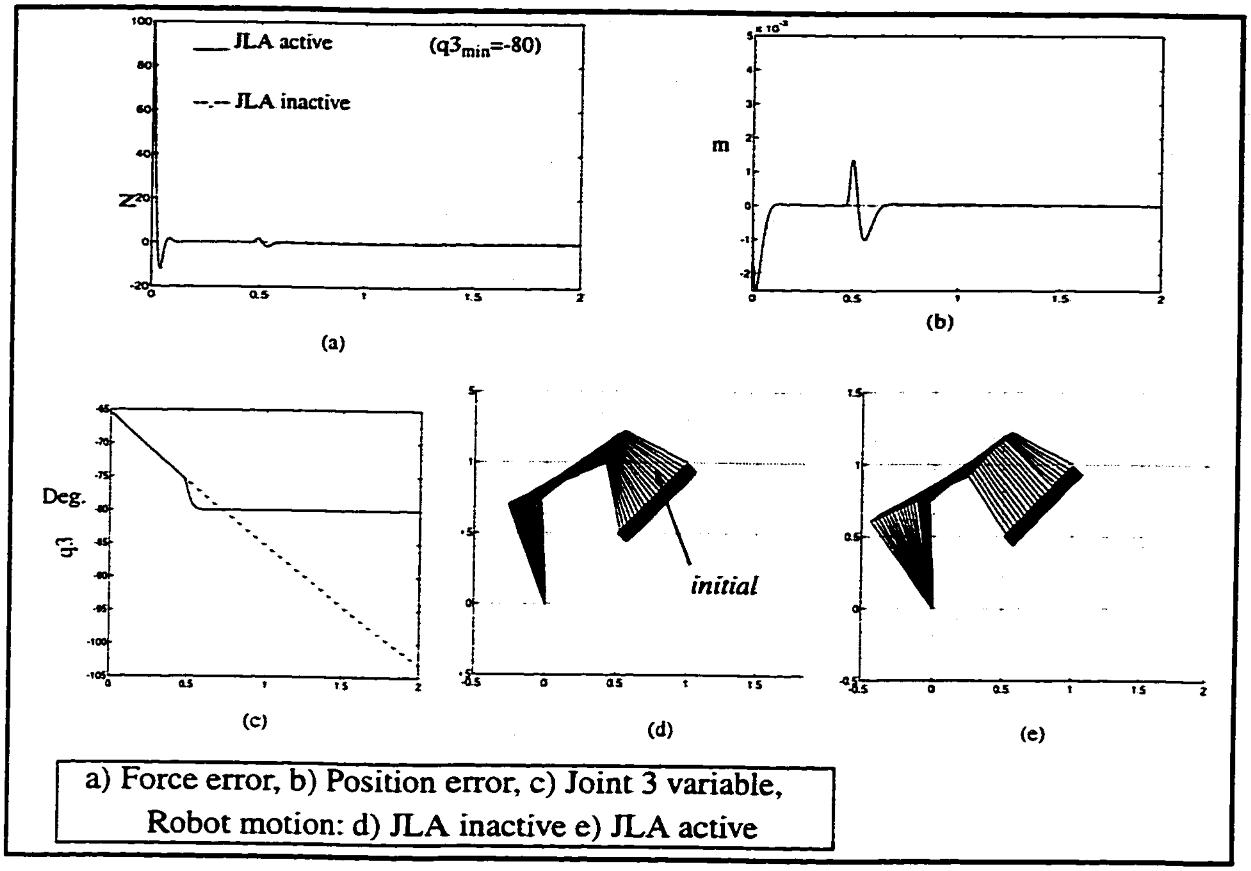


Figure 4.7 General block diagram of the AHIC scheme

In the Figure 4.7, *T* denotes the homogenous coordinate transformation between two different frames (*W* refers to the workspace, and *C* refers to the end-effector constraint surface). Note that the dashed-line blocks are needed if only the additional task is force-controlled (*C1* refers to the second constraint surface).

Joint Limit Avoidance (JLA)

The formulation of the additional task was given in Section 2.4.1 . In the first simulation, JLA is inactive, and the resulting errors in the position and force controlled subspaces (Figure 4.8) both converge to small values (practically zero). However, the joint three value goes below -100 degrees. In the second simulation, JLA is active and the minimum limit for joint three is selected as -80 degrees. The simulation results again show that the force and position trajectories are tracked correctly, and also, the limit on joint three is respected.



a) Force error, b) Position error, c) Joint 3 variable, Robot motion: d) JLA inactive e) JLA active

Figure 4.8 Simulation results for the AHIC scheme with Joint Limit Avoidance

Static and Moving Obstacle Avoidance (SOCA and MOCA)

The formulation of the additional task was given in Section 2.4.2 . The results for SOCA are indicated in Figure 4.9, When the obstacle avoidance algorithm is inactive, the main task trajectories are followed correctly. However, a collision occurs. By activating obstacle avoidance, the collision is avoided and the main task requirement is also satisfied.

In the next simulation, the position of the tip in the X_c direction is required to be fixed, while exerting a constant force equal to -100 N in the Y_c direction. Figure 4.10 shows that the main task has been accomplished within a short time, and from this time onwards, the manipulator does not move until the MOCA additional task becomes active, and successfully prevents the collision.

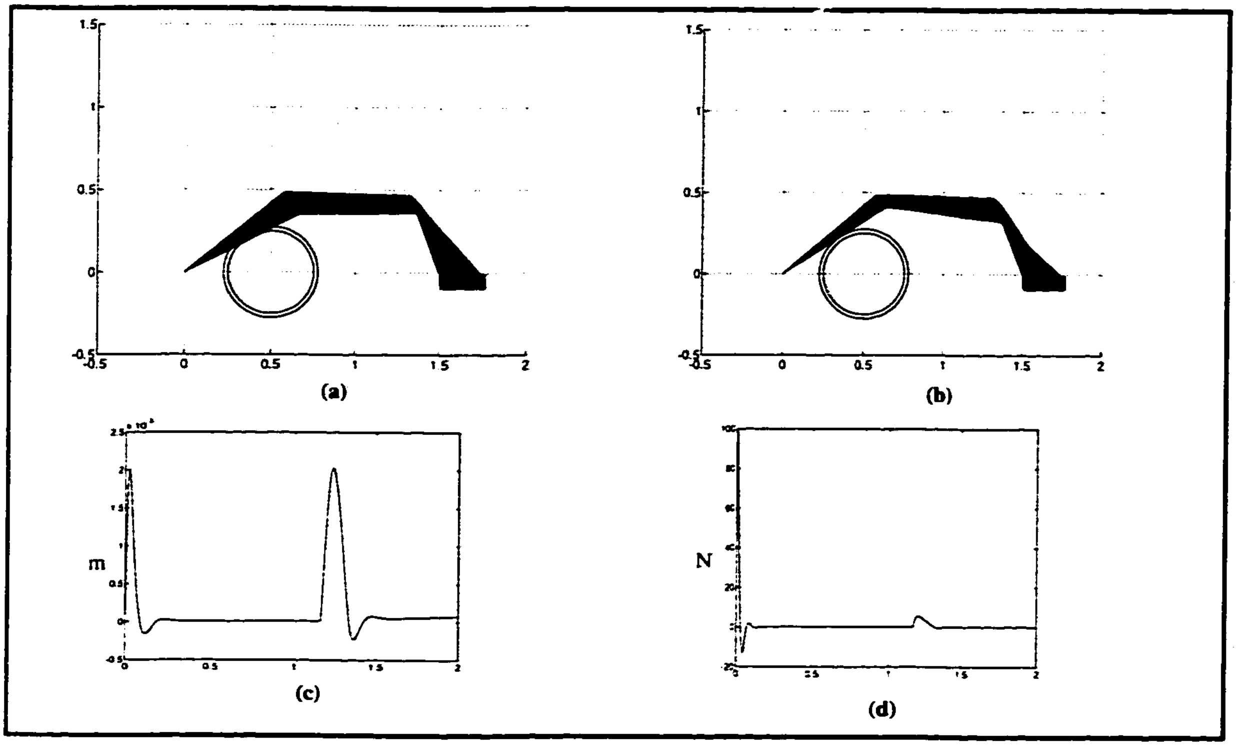


Figure 4.9 Static Obstacle Collision Avoidance: Robot motion a) SOCA off b) SOCA on, Errors c) position d) Force

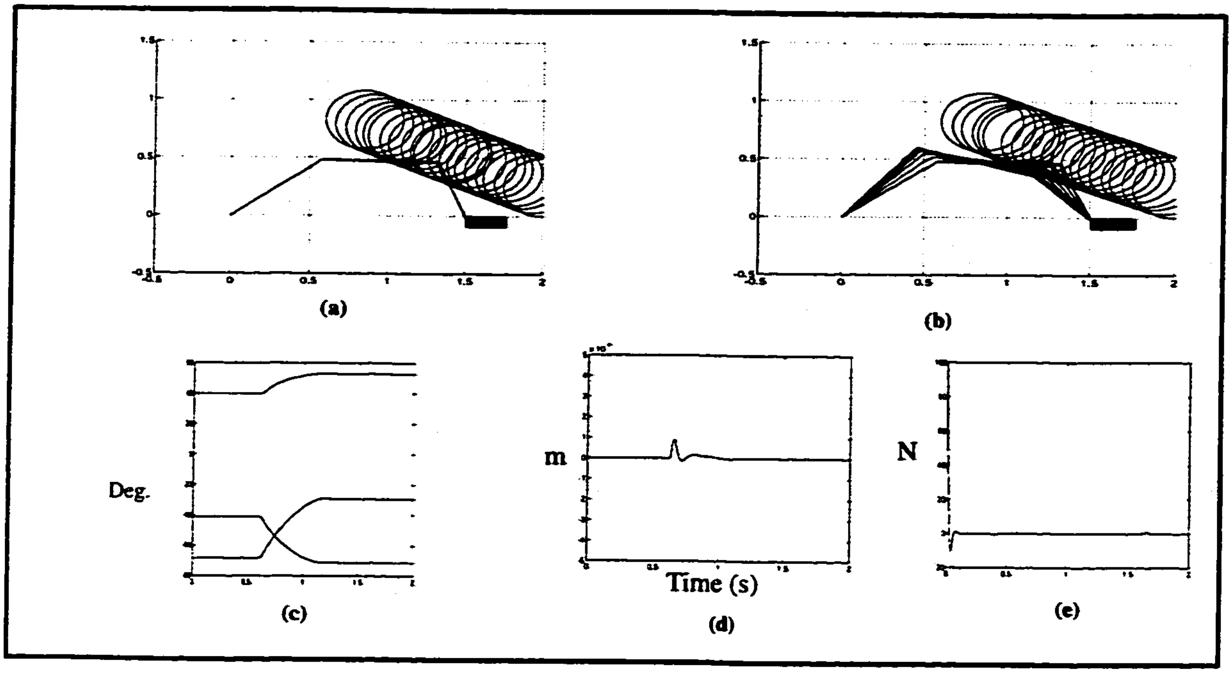


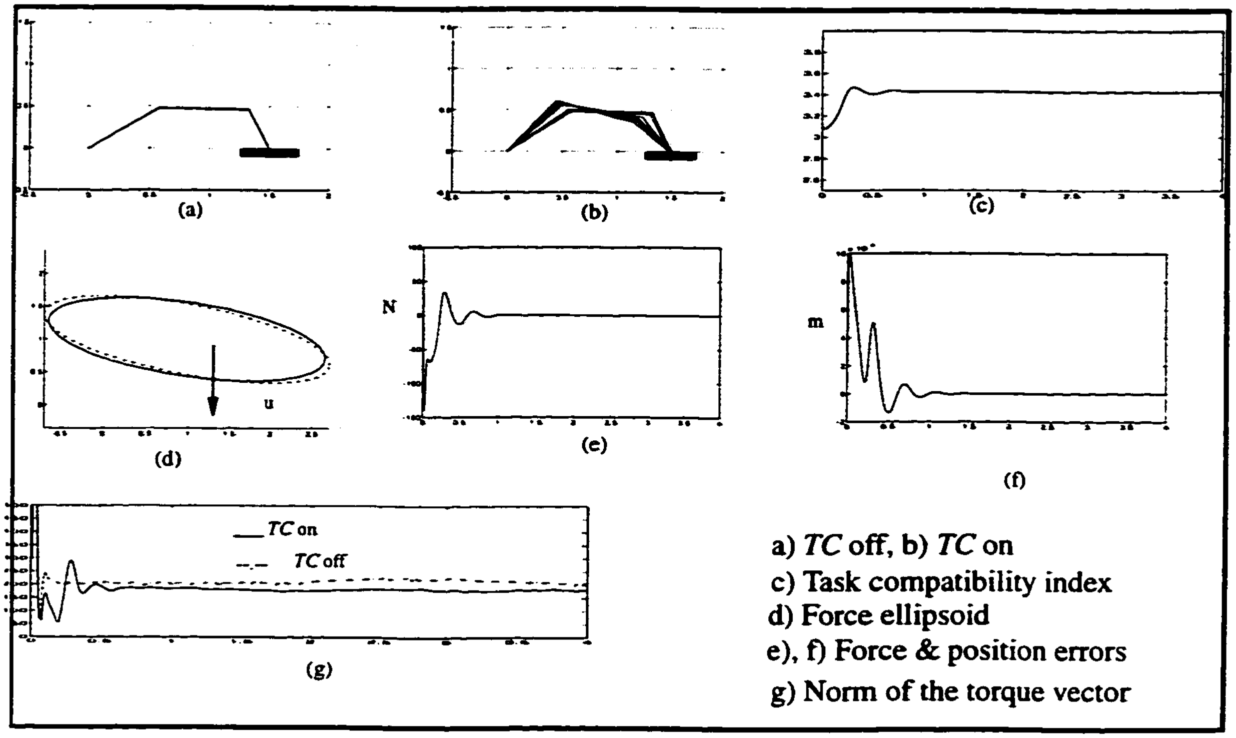
Figure 4.10 Moving Obstacle Collision Avoidance: Robot motion a) MOCA off b) MOCA on, c) Joint variables, d) Position error, e) Force error

Task Compatibility

The objective of this additional task is to position the arm in the posture which requires minimum torque for a desired force in a certain direction. The formulation of this additional task is given in Section 2.4.3 .

Figure 4.11 shows the results of the simulation for this case. The main task consists of keeping the manipulator tip at a fixed position in the X direction while exerting -100 N in the Y direction.

As we can see in Figure 4.11b, the manipulator reconfigures itself to find the posture which requires the minimum torque to exert the desired force. Figure 4.11c shows how the value of the objective function - task compatibility index given by (2.4.16) - increases to reach the optimal configuration. Figure 4.11d shows the force ellipsoid for the initial and final configurations. Note that the force transfer ratio along the Y direction has been increased. Figures 4.11e and f show that the force and position trajectories of the main task were followed correctly. Note that the required torque is reduced when the additional task is active (Figure 4.11g).



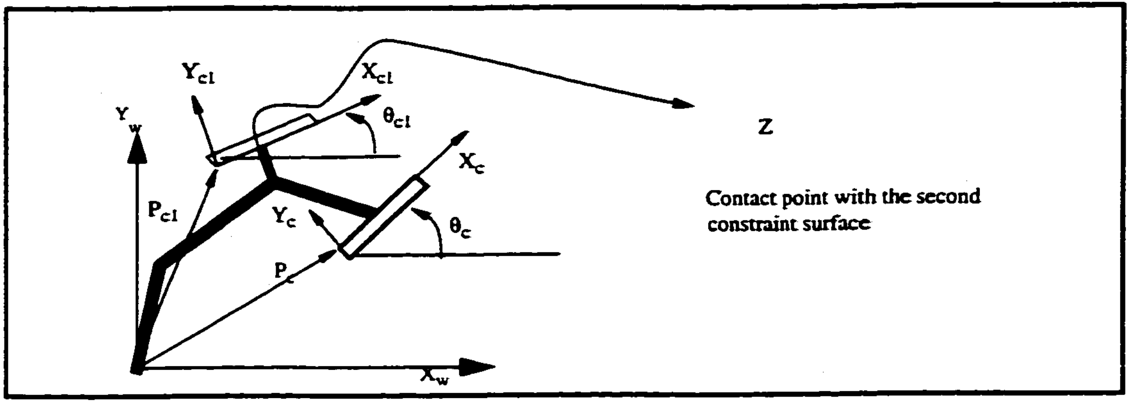
a) *TC* off, b) *TC* on
c) Task compatibility index
d) Force ellipsoid
e), f) Force & position errors
g) Norm of the torque vector

Figure 4.11 Task compatibility simulation results

Force-Controlled Additional Task

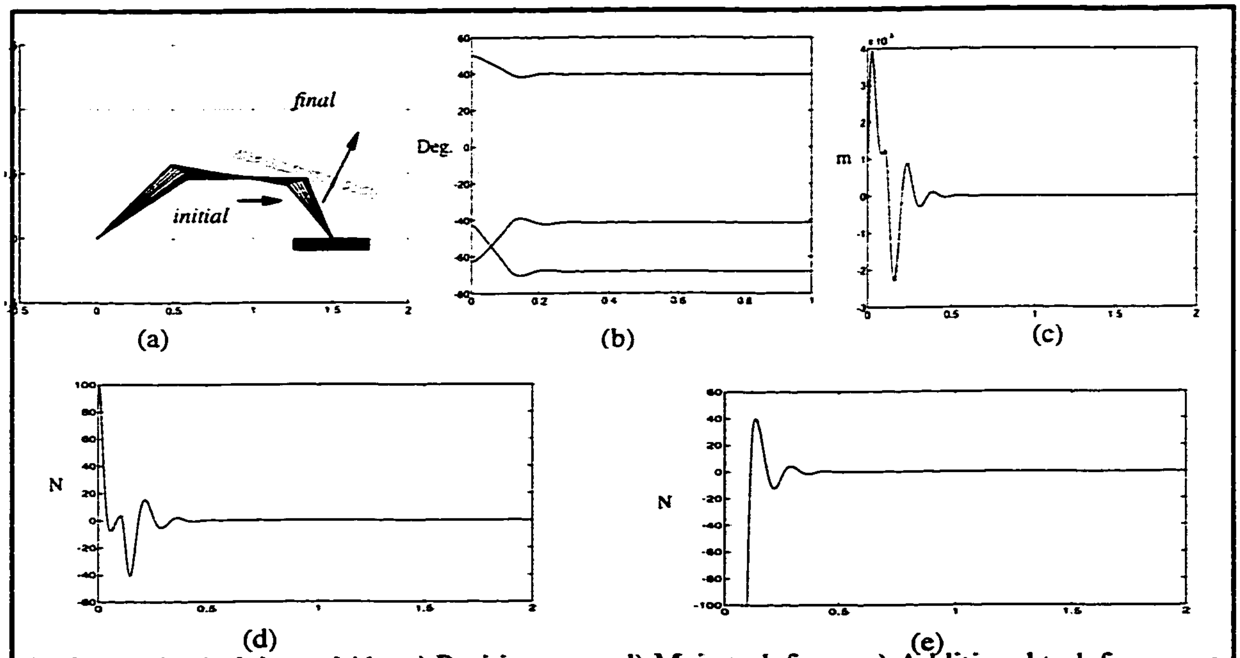
We have already noted that the additional task(s) can be included in either of the position-controlled or force-controlled subspaces. In the following simulation, the additional task consists of exerting a constant force to a second compliant surface (Figure 4.12) by an arbitrary point *Z* fixed to one of the links - in this simulation, the joint between the second and third links, joint 3.

The Jacobian of the additional task is the Jacobian of the point Z, and the desired force in the Y_{c1} direction is to be specified. The main task consists of keeping the position of the tip in the X_w direction unchanged, while exerting a constant *-100 N* force in Y_W direction on the first constraint surface. The additional task is to exert a *100 N* force (in the Y_{c1} direction) on the second constraint surface by joint three. Figure 4.13b shows the motion of the joints, Figures 4.13c, and d show that the main task is executed correctly, and finally, Figure 4.13e shows that the desired force is exerted on the second constraint surface. Note that, although initially joint three is not in contact with the second constraint surface, the AHIC scheme works correctly and makes this point move toward the surface with a bounded velocity.



Contact point with the second constraint surface

Figure 4.12 Force-Controlled Additional Task



(a) (b) (c)

(d) (e)

a) robot motion b) joint variables c) Position error d) Main task force, e) Additional task force error

Figure 4.13 Force-Controlled Additional Task

4.3.3 Augmented Hybrid Impedance Control with Self-Motion Stabilization

As we mentioned earlier, redundancy resolution at the acceleration level is aimed at minimizing joint accelerations and not controlling the self-motion of the arm. This is the major shortcoming of the AHIC scheme proposed in Section 4.3.2. In this section by modifying both the inner and outer control loops, a new AHIC control scheme is proposed which enjoys all the desirable characteristics of the previous scheme and achieves self-motion stabilization.

4.3.3.1 Outer-Loop Design

The design of the outer-loop is similar to the design in Section 4.3.2.1. The only difference is that instead of calculating an *Augmented Cartesian Target Acceleration (ACTA)* trajectory, we describe the desired motion by an *Augmented Cartesian Target (ACT)* trajectory at position, velocity, and acceleration levels.

The motion of the manipulator in both subspaces can be expressed by a single matrix equation using the selection matrices S_x and S_z, as follows:

$$M_x^d(\ddot{X}^t - S_x\ddot{X}^d) + B_x^d(\dot{X}^t - S_x\dot{X}^d) + K_x^d S_x(X^t - X^d) - (I - S_x)F_x^d = -F_x^e \quad \text{(a)} \ (4.3.10)$$

$$M_z^d(\ddot{Z}^t - S_z\ddot{Z}^d) + B_z^d(\dot{Z}^t - S_z\dot{Z}^d) + K_z^d S_z(Z^t - Z^d) - (I - S_z)F_Z^d = -F_z^e \quad \text{(b)}$$

where the same definitions as in (4.3.5) are used.

The ACT trajectory $[X^{t^T}, Z^{t^T}]^T$ is the unique solution of the differential equations (4.3.10) with initial conditions:

$$\begin{aligned} X^t(0) &= X^d(0) & \dot{X}^t(0) &= \dot{X}^d(0) \\ Z^t(0) &= Z^d(0) & \dot{Z}^t(0) &= \dot{Z}^d(0) \end{aligned} \quad (4.3.11)$$

Notice that the presence of measurement forces in these equations requires that the ACT trajectory should be generated online.

4.3.3.2 Inner-Loop Design

The dynamics of a rigid manipulator are described by equation (4.3.8). The controller should be designed to calculate the torque input to the dynamic equation (4.3.8), which ensures the tracking of the ACT trajectory. The procedure is as follows: First, a Cartesian reference trajectory is defined for both the main and additional tasks:

$$\dot{X}^r = \dot{X}^t - \Lambda_x(X - X^t) \qquad and \qquad \ddot{X}^r = \ddot{X}^t - \Lambda_x(\dot{X} - \dot{X}^t) \qquad \text{(a,b)} \qquad (4.3.12)$$

$$\dot{Z}^r = \dot{Z}^t - \Lambda_z(Z - Z^t) \qquad and \qquad \ddot{Z}^r = \ddot{Z}^t - \Lambda_z(\dot{Z} - \dot{Z}^t) \qquad \text{(c,d)}$$

where Λ_x and Λ_z are the positive-definite gain matrices. The joint reference trajectory is defined by using the task prioritized and singularity robust redundancy resolution scheme described in Section 4.3.1. This is done by replacing the Cartesian reference velocity and acceleration in equations (2.3.19) and (4.3.4) to find $\dot{q}^r$, $\ddot{q}^r$. Now a virtual velocity error is defined as:

$$s = \dot{q} - \dot{q}^r \qquad (4.3.13)$$

The control law is then given by:

$$\tau = H(q)\ddot{q}^r + C(q, \dot{q})\dot{q}^r + G(q) + f(\dot{q}) + K_D s - J_e^T F_x^e - J_{c1}^T F_z^e \qquad (4.3.14)$$

where K_D is a positive-definite matrix. This control law does not cancel the robot dynamics. However, it ensures asymptotic, or by proper choice of K_D, and Λ, exponential tracking of ACT at the same rate as that of exact cancellation (see [52] and [53]).

Remarks:

- Note that by "asymptotic tracking of ACT", we mean that the control law guarantees convergence to a solution that minimizes (2.3.20).
- The above procedure is different from the design of the controller in joint space, because in the latter, the ACT trajectory would be used to generate the desired joint trajectories $q^d, \dot{q}^d, \ddot{q}^d$. However, in the proposed algorithm, explicit calculation of the desired joint values is avoided.
- The use of the controller proposed in this section has two major advantages over the inverse dynamics (or computed torque) method which is used in Section 4.3.2:

  (i) It controls self-motion because both velocity and acceleration information are used; the computed torque method requires only a commanded acceleration trajectory.

  (ii) The formulation of this algorithm is similar to a non-adaptive version of the approach of Slotine and Li [52]. Therefore, to deal with inaccurate dynamic parameters, an adaptive implementation of this algorithm can be developed without major modifications to the inner loop which is the subject of the Section 4.3.4.

4.3.3.3 Simulation Results on a 3-DOF Planar Arm

The setup for the constrained compliant motion control is shown in Figure 4.6. A general block diagram of the simulation is shown in Figure 4.14.

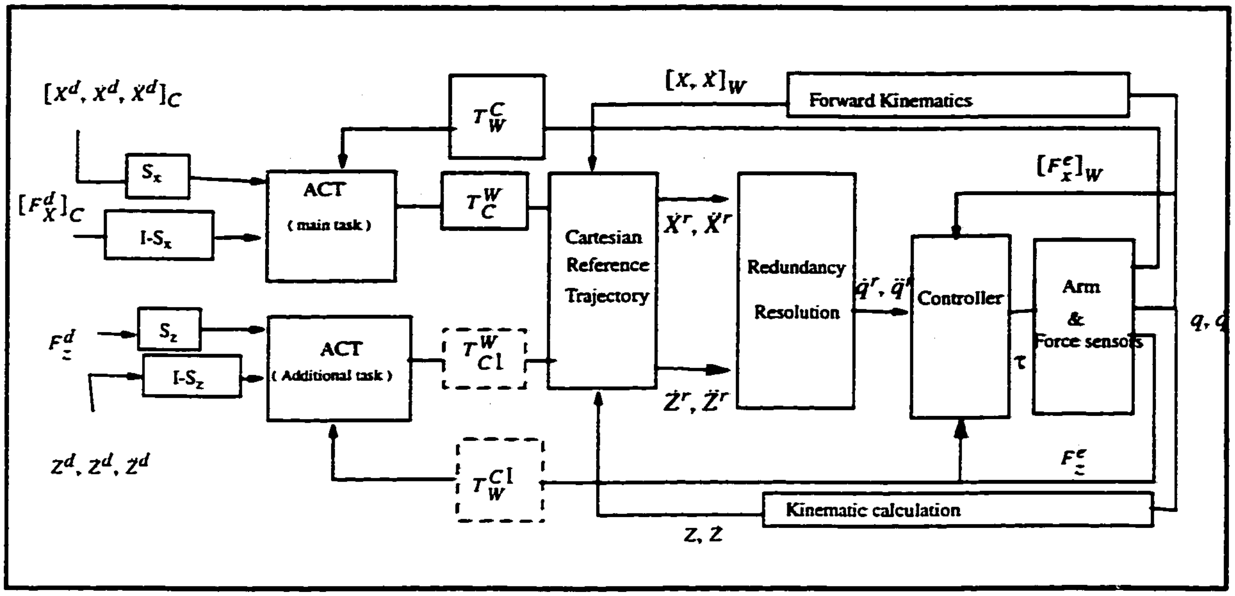
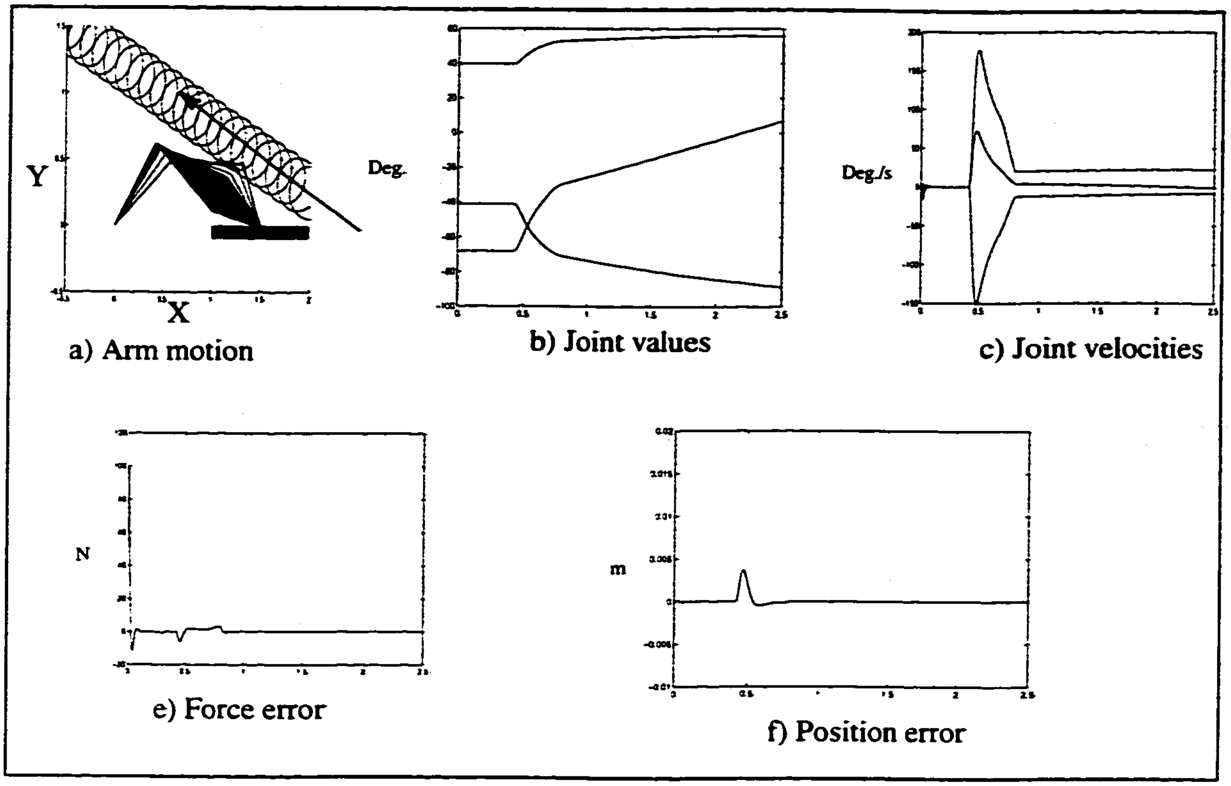


Figure 4.14 General block diagram of the AHIC scheme

Obstacle avoidance with self-motion stabilization

In this simulation, the end-effector is initially at rest and touches the constraint surface (*f=0*) at the point (1.5,0). The main task consists of keeping the position in the *X* direction constant, while exerting a desired *-100 N* in the *Y* direction. There is also a moving object enclosed in a circle in the workspace. The additional task consists of using the redundant degree of freedom to avoid this object. The simulation is carried out to compare the method proposed in Section 4.3.2 and the method proposed in this section.

As we can see in the plot of the joint velocities (Figure 4.15c, Figure 4.16c), there is a movement for a short period at the beginning to achieve the desired force - the end-effector moves in the Y direction to penetrate the surface.



a) Arm motion

b) Joint values

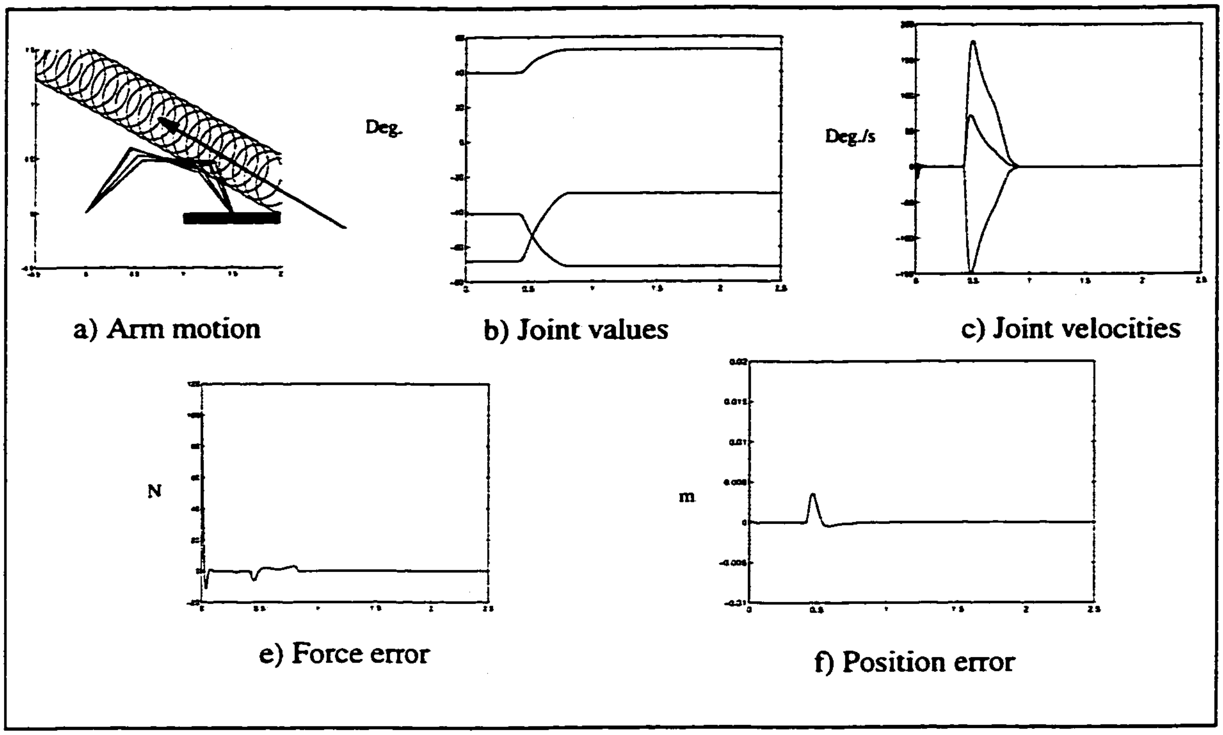
c) Joint velocities

e) Force error

f) Position error

Figure 4.15 Object avoidance without self-motion stabilization

The manipulator remains stationary until the object is close enough to the arm. The obstacle avoidance task becomes active and makes the manipulator move in the null space of the Jacobian matrix to avoid collision while satisfying the main task. The two algorithms perform in the same way up to the point that the object clears the arm. From that point onwards, the algorithm in Section 4.3.2 is unable to control the null space components of the joint velocities and causes self-motion (Figure 4.15b). However, the proposed algorithm is successful in damping out these components and preventing self-motion.



a) Arm motion

b) Joint values

c) Joint velocities

e) Force error

f) Position error

Figure 4.16 Moving object avoidance with Self-motion stabilization

4.3.4 Adaptive Augmented Hybrid Impedance Control

It has been shown that the methods do not address the uncertainty in a manipulator's dynamics may result to unstable motion in practice. This has led to considerable work on adaptive control of manipulators [51], [53]. Adaptive compliant control has also been addressed in recent years. Han et al. [47] have proposed an adaptive control scheme for constrained manipulators based on a nonlinear coordinate transformation; Lu and Meng [48] have proposed an adaptive impedance control scheme, and Niemeyer and Slotine [49] have discussed an application of the adaptive algorithm of Slotine and Li [52] to compliant motion control and redundant manipulators. However, application of the above algorithms to redundant manipulators introduces several problems. For instance, the algorithm in [47] requires definition of a nonlinear invertible transformation from joint space to a generalized task space. The algorithm in [48] is based on the Cartesian dynamic model of a manipulator and can be applied to the redundant case. However, no user defined additional tasks can be incorporated in the algorithm and redundancy is based on the generalized inertia-weighted inverse of the Jacobian. The algorithm proposed in [49] overcomes the above drawbacks. However, it is assumed that the rows of the Jacobian matrix are linearly independent. Hence, it may result in instability near singular configurations. In this section, by incorporating the adaptive algorithm of Slotine and Li in the AHIC scheme proposed in Section 4.3.3, an Adaptive Augmented Hybrid Impedance Control (AAHIC) scheme is presented which guarantees asymptotic convergence in both position and force controlled subspaces with precise force measurements. The control scheme ensures stability of the system with bounded force measurement errors. Even in the case of imprecise force measurement, the errors in the position controlled subspaces can be reduced considerably provided powerful enough actuators are available.

4.3.4.1 Outer-Loop Design

The design of the outer-loop is exactly similar to that explained in Section 4.3.3.1.

4.3.4.2 Inner-Loop Design

The dynamics of a rigid manipulator are described by equation (4.3.8). The controller should be designed to calculate the torque input to equation (4.3.8), which ensures the tracking of the ACT trajectory in the presence of uncertainties on the manipulators dynamic parameters.

It has been shown that for a suitably selected set of dynamic parameters, equation (4.3.8) can be written as:

$$H(q)\ddot{q}^r + C(q,\dot{q})\dot{q}^r + G(q) + f(\dot{q}) = Y(q,\dot{q},\dot{q}^r,\ddot{q}^r)a \qquad (4.3.15)$$

where Y is the $n \times p$ regressor matrix and a is the $p \times 1$ vector of dynamic parameters. The matrix C is defined in such a way that $\dot{H} - 2C$ is a skew-symmetric matrix [52].

Now an extension of the adaptive algorithm of Slotine and Li [52] is used to design the controller in order to ensure asymptotic tracking of the ACT trajectory. The procedure is as follows:

First, a Cartesian reference trajectory is defined for both the main and additional tasks (see equations (4.3.12)). Then, a virtual velocity error is defined (see (4.3.13)). The control law is then given by:

$$\tau = Y\hat{a} - K_D s - J_e^T \hat{F}_x^e - J_{c1}^T \hat{F}_z^e = \hat{H}(q)\ddot{q}^r + \hat{C}(q, \dot{q})\dot{q}^r + \hat{G}(q) + \hat{f}(\dot{q}) - J_e^T \hat{F}_x^e - J_{c1}^T \hat{F}_z^e \quad (4.3.16)$$

where $\hat{H}, \hat{C}, \hat{G}, \hat{f}, \hat{a}$ are calculated based on estimated values of H, C, G, f, and a respectively. $\hat{F}_x^e$ is the measured end-effector interaction force with the environment, K_D is a positive-definite matrix, and $s = \dot{q} - \dot{q}^r$. The last term on the right-hand side of the equation is only needed if another point of the manipulator (other than the end-effector) is in contact with the environment; F_z^e denotes the measured reaction force corresponding to a second constraint surface, and J_{cl} is the Jacobian of the contact point.

We use the same Lyapunov candidate function as in [48]:

$$V(t) = \frac{1}{2}[s^T H s + \tilde{a}^T \Gamma \tilde{a}] \quad (4.3.17)$$

where Γ is a constant positive definite matrix and $\tilde{a} = a - \hat{a}$. Differentiating $V(t)$ along the trajectory of the system (4.3.8) leads to

$$\dot{V}(t) = -s^T K_D s + s^T Y \tilde{a} + s^T J_e^T \tilde{F}_x^e + s^T J_{c1}^T \tilde{F}_z^e \quad (4.3.18)$$

where $\tilde{F} = F - \hat{F}$ denotes force measurement error. This suggests that the adaptation law should be selected as:

$$\dot{\hat{a}} = -\Gamma Y^T s \quad (4.3.19)$$

With this adaptation law, equation (4.3.18) leads to:

$$\dot{V}(t) = -s^T K_D s + s^T(J_e^T \tilde{F}_x^e + J_{c1}^T \tilde{F}_z^e) \le -\underline{k}_D\|s\|^2 + \|s\|(\|J_e\|\|\tilde{F}_x^e\| + \|J_{c1}\|\|\tilde{F}_z^e\|) \quad (4.3.20)$$

and

$$\dot{V}(t) \le -\underline{k}_D\|s\|^2 + \delta\|s\| \quad (4.3.21)$$

where $\underline{k}_D$ is the minimum eigenvalue value of the matrix K_D, and δ satisfies the following inequality:

$$\|J_e\|\|\tilde{F}_x^e\| + \|J_{c1}\|\|\tilde{F}_z^e\| \le \delta \quad (4.3.22)$$

We also assume that $\|J_e\| \le \alpha$ and $\|J_{c1}\| \le \beta$. Now, we consider two different cases, precise and imprecise force measurements.

Precise force measurements $\tilde{F} = 0$

In this case inequality (4.3.21) reduces to

$$\dot{V}(t) \le -\underline{k}_D\|s\|^2 \quad (4.3.23)$$

which implies $a, s \in L_\infty^n$ or boundedness of a and s. Moreover, it can be shown that

$$\|s\|^2 dt \le \frac{-1}{\underline{k}_D} dV(t) \qquad \text{(a)} \qquad (4.3.24)$$

$$\int_0^\infty \|s\|^2 dt \le \frac{-1}{\underline{k}_D}\int_0^\infty dV(t) = \frac{1}{\underline{k}_D}(V(0) - V(\infty)) \qquad \text{(b)}$$

which implies that $s \in L_2^n$ and consequently $J_e s, J_c s \in L_2^n$. In order to establish a link between S and the tracking error of ACT trajectories, we assume that the tracking error of the damped least-squares solution (2.3.19) is negligible. Therefore, multiplying both sides of equation (4.3.13) by the augmented Jacobian, leads to

$$J_e s = \dot{e}_x + \Lambda_x e_x \qquad \text{(a)} \qquad (4.3.25)$$

$$J_c s = \dot{e}_z + \Lambda_z e_z \qquad \text{(b)}$$

where

$$e_x = X - X^t \qquad e_z = Z - Z^t \qquad (4.3.26)$$

Equations (4.3.25) represent strictly proper, asymptotically stable linear time-invariant systems with inputs $J_e s, J_c s \in L_2^n$ which implies exact tracking and asymptotic convergence of the trajectories X and Z to the ACT trajectories [50], [54].

Imprecise Force Measurements $\tilde{F} \neq 0$ (Robustness Issue)

To take into account the robustness issue, we consider the effects of imprecise force measurements. It is obvious that the error in force measurements directly affects the tracking performance in the force controlled subspaces of the main and additional tasks. However, we can show boundedness of the closed-loop trajectories. Moreover, the upper-bound on the error in the position-controlled subspaces can be reduced.

In this case, the time derivative of the Lyapunov candidate function satisfies

$$\dot{V}(t) \leq -\underline{k}_D \|s\|^2 + \delta \|s\| \qquad (4.3.27)$$

As in [48], we can state that $\dot{V}(t)$ is not guaranteed to be negative semi-definite with an arbitrary value of $\underline{k}_D$ and a large δ for small values of $\|s\|$. However, positive $\dot{V}(t)$ implies increasing V and subsequently $\|s\|$, which eventually makes $\dot{V}(t)$ negative. Therefore, s remains bounded and converges to a residual set. For a fixed value of $\underline{k}_D$, the lower bound of s is determined by $\delta / \underline{k}_D$ and can be reduced by selecting a larger value of $\underline{k}_D$. Note that larger $\underline{k}_D$ increases the control effort and may saturate the actuators. Using equations (4.3.24) and boundedness of s, we can conclude boundedness of e_x and e_Z.

Remark: Dawson and Qu [46] have proposed a modification to the control law given in

(4.3.16) by adding a term $-K_{\delta}\operatorname{sgn}(s)$ to the right hand side with $K_{\delta} > \delta$. This eventually leads to the same inequality for $\dot{V}(t)$ as in (4.3.23) which implies asymptotic convergence of the errors. However, the control law proposed in [46] is discontinuous in terms of s and may excite unmodeled high-frequency dynamics.

4.3.4.3 Simulation Results on a 3-DOF Planar Arm

The setup for the constrained compliant motion control is shown in Figure 4.6. A general block diagram of the simulation is shown in Figure 4.14.

Tool Orientation Control

In this simulation the additional task is defined as the control of the orientation of a tool attached to the end-effector. In this case, the desired value is specified as $q_3 = -85°$. The end-effector is initially at the point (*X=1, Y=1*) (Figures 4.17a, c) in touch with the surface (zero interaction force).

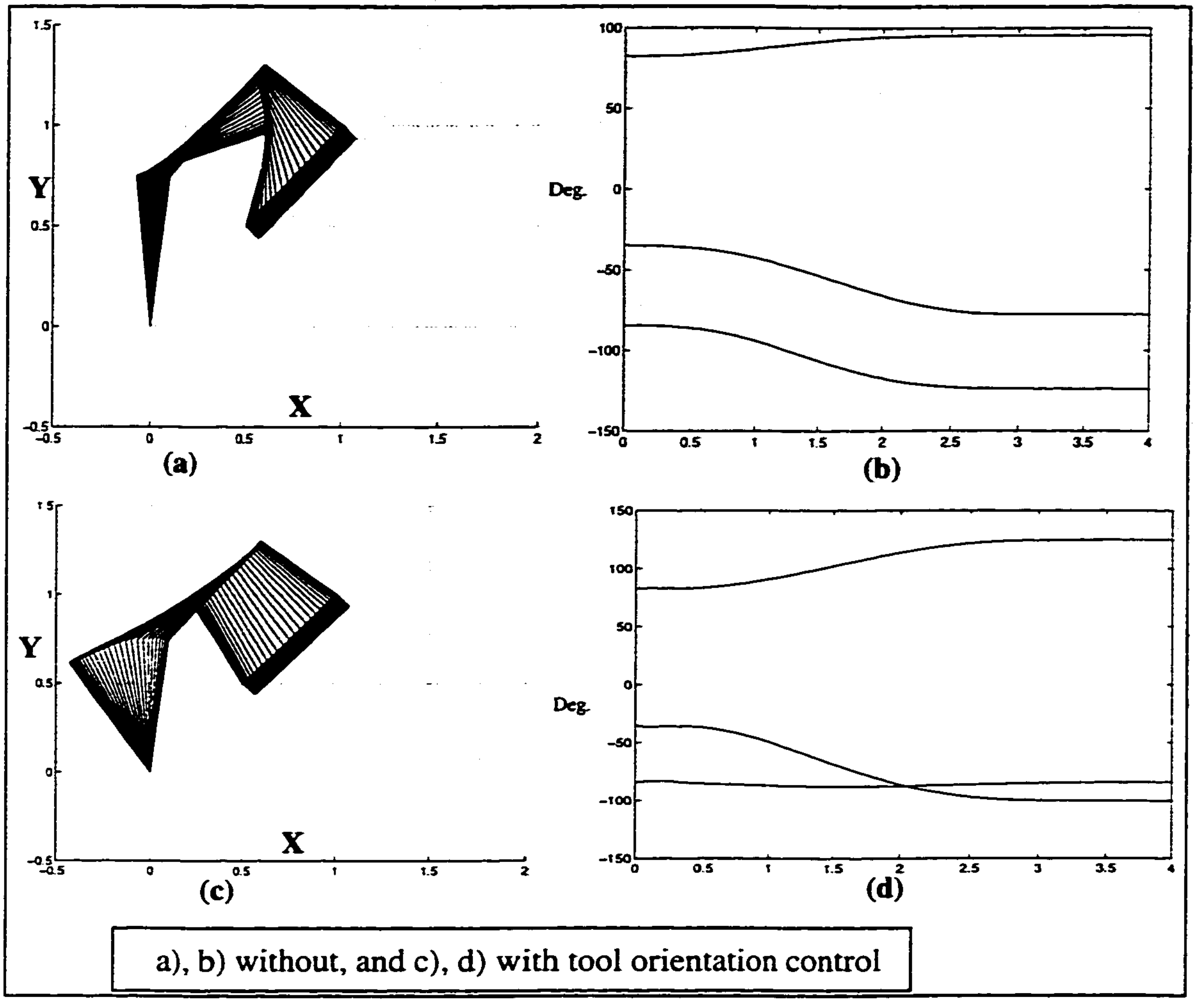
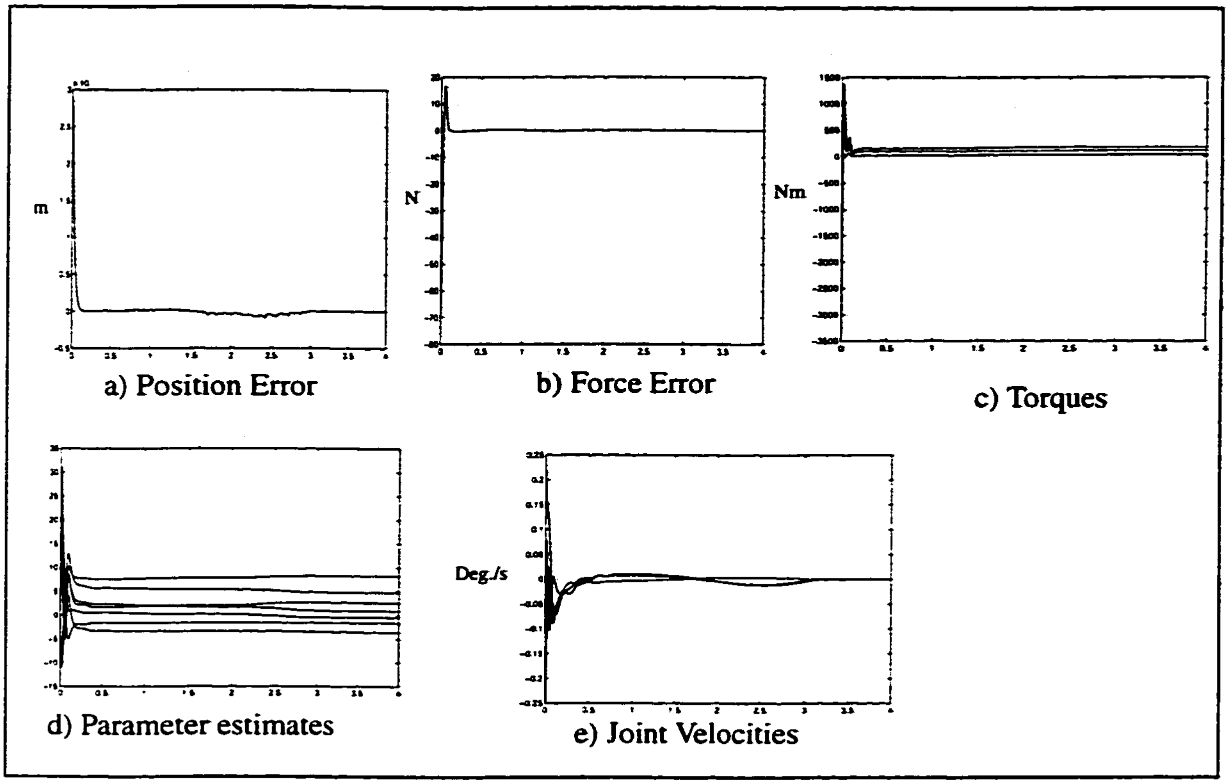


Figure 4.17 Adaptive AHIC: Arm configuration and joint values

Figures 4.17a, b show that without activating the additional task, there is no restriction on joint three. However, by activating the additional task (Figures 4.17c, d), the tool orientation is maintained at the desired value. Figures 4.18a, b show the errors in the position- and force-controlled subspaces which practically converge to zero. The dynamic parameter estimates and the velocity error are shown in Figures 4.18d, e.



a) Position Error

b) Force Error

c) Torques

d) Parameter estimates

e) Joint Velocities

Figure 4.18 Adaptive AHIC **with tool orientation control**

In order to study the effects of imprecise force measurements, the actual interaction force is augmented by a random noise uniformly distributed in the interval (-15N,15N). As we can see in Figure 4.19b, the error in the force controlled direction increases significantly as expected. The reason is that the controller in the force-controlled direction is based on force measurements and any error in this respect, directly affects the force error, e.g., the interval between 2 to 3 seconds. However, the error in the position-controlled direction (Figure 4.19a) remains practically unchanged from that of the previous simulation (Figure 4.18a), showing the robustness of the algorithm to force measurement error.

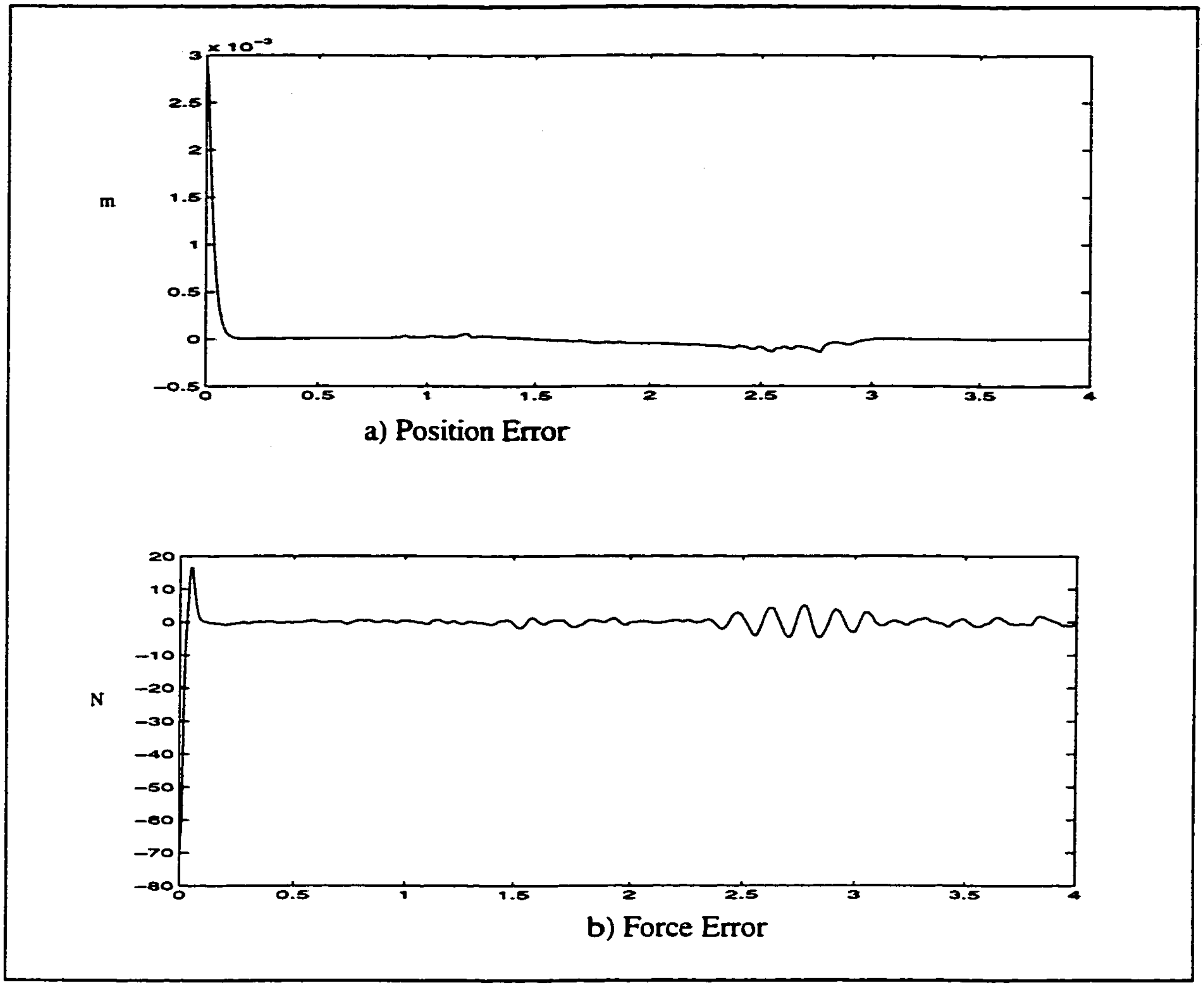


Figure 4.19 Adaptive Hybrid Impedance Control: **Effect of imprecise force measurement**

4.4 CONCLUSION

In this chapter, the problem of compliant motion and force control for redundant manipulators was addressed. A bibliographic review on existing methodologies for compliant motion and force control of robotic manipulators was performed. Based on this review, an Augmented Hybrid Impedance Control Scheme was proposed. In Section 4.3.2, An extension of configuration control at the acceleration level was proposed to perform redundancy resolution. The most useful additional tasks: Joint limit avoidance, static and moving object avoidance, and posture optimization, were incorporated into the AHIC scheme. The proposed scheme enjoys the following desirable characteristics:

- Different additional tasks can be easily incorporated into the AHIC scheme without modifying the scheme and the control law.
- The additional task(s) can be included in the force-controlled subspace of the augmented task. Therefore, it is possible to have a multiple-point force control scheme.
- Task priority and singularity robustness formulation of the AHIC scheme relax the restrictive assumption of having a non-singular augmented Jacobian.

A modified AHIC scheme was proposed in Section 4.3.3 that gives a solution to the undesirable self-motion problem which exists in most dynamic control schemes proposed for redundant manipulators.

An Adaptive Augmented Hybrid Impedance Control (AAHIC) scheme was presented which guarantees asymptotic convergence in both position- and force-controlled subspaces with precise force measurements. The control scheme also ensures stability of the system in presence of bounded force measurement errors. Even in the case of imprecise force measurements, the errors in the position controlled subspaces can be reduced considerably.

The performance of proposed AHIC schemes were studied on a 3-DOF planar arm. We are now ready to undertake the extension of the AHIC scheme to the 3-D workspace of REDIESTRO. This will be done in the next chapter.

CHAPTER 5 AUGMENTED HYBRID IMPEDANCE CONTROL FOR A 7-DOF REDUNDANT MANIPULATOR

5.1 INTRODUCTION

In Chapter 4, the AHIC scheme was developed and verified by simulation on a 3-DOF planar arm. In this chapter the extension of the AHIC scheme to the 3-D workspace of REDIESTRO is described. Figure 5.1 shows a simplified block diagram of the AHIC controller. Considering that the capabilities of the redundancy resolution scheme with respect to collision avoidance has already been fully demonstrated and in order to focus on the new issues related to Contact Force Control (CFC), the environment is assumed to be free of obstacles.

The major objectives in this chapter are as follows:

- **Software development for new modules in *C*:** Redundancy Resolution (RR), Forward Kinematics (FwdKin), Linearized Decoupling (LD) controller.
- **Preparation of different simulation environments:** To integrate and test the extended algorithms.
- **Stability and trade-off analysis.**

The complexity of the required algorithms and restrictions on the amount of computational power available have resulted in an algorithm development procedure which incorporates a high level of optimization. At the same time, the following issues which were not studied in a 2-D workspace need to be tackled in extending the schemes to a 3-D workspace:

- Extension of the AHIC scheme for orientation and torque
- control of self-motion as a result of resolving redundancy at the acceleration level for the AHIC scheme represented in Section 4.3.2
- Robustness with respect to higher-order unmodelled dynamics (joint flexibilities), uncertainties in manipulator dynamic parameters, and friction model.

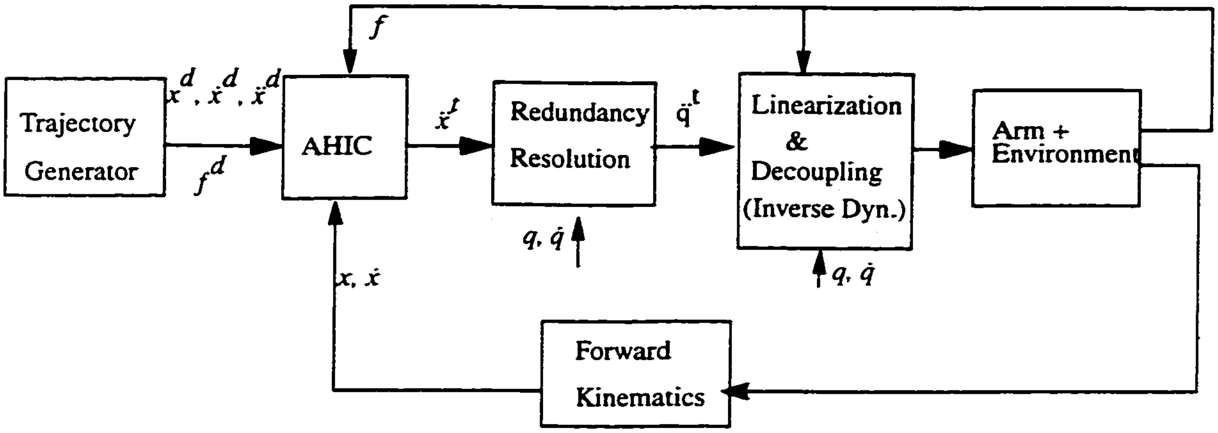


Figure 5.1 Simplified block diagram of the AHIC controller

5.2 ALGORITHM EXTENSION

In this section, different modules involved in the AHIC scheme are described. The focus is on describing the required algorithms without getting involved in the specific way in which the modules are implemented. The latter will be discussed in the software development description (see Appendix A).

5.2.1 Task Planner and Trajectory Generator (TG)

The user can specify the robot's task using a Pre-Programmed Task File (PPTF) - see Appendix A for more details. Each line indicates the desired position and orientation to be reached at the end of that segment, the hybrid task specification, and the desired impedance and force (if applicable) for each of the 6 DOFs.

In the absence of obstacles, the robot path will consist of straight lines connecting the desired position/orientation at each segment. The TG module generates a continuous path between the via points. The TG implemented to test the AHIC scheme is a fifth-order polynomial trajectory which gives continuous position, velocity, and acceleration profiles with zero jerk at the beginning and end of the motion.

5.2.2 AHIC module

Figure 5.2 shows the location of the different frames used by the AHIC module. The description of the environment is specified by the user in a configuration file. As an example, for a surface-cleaning task, the user is required to specify the location and orientation of a fixed frame $\{C\}$ with respect to the world frame. In this case, the robot's base frame $\{R_1\}$ is selected as the world frame. The tool frame $\{T\}$ is attached to the last link. Depending on the type of the tool, the user specifies the location and orientation of this frame in the last joint's local frame. The force sensor interface card also uses this information to locate the force sensor frame at $\{T\}$. The task frame $\{C_i\}$ is located at the origin of the frame $\{T\}$. However, the orientation of $\{C_i\}$ is dictated by $\{C\}$. Therefore, the frame $\{C_i\}$ moves with the tool while keeping the same orientation as the constant frame $\{C\}$.

The AHIC scheme, as implemented for the 2-D workspace, generates an Augmented Cartesian Target Acceleration *(ACTA)* for the end-effector (EE) position in real-time:

$$\ddot{X}^{t} = {M^{d}}^{-1}(-F^{e} + (I-S)F^{d} - B^{d}(\dot{X} - S\dot{X}^{d}) - K^{d}S(X - X^{d})) + S\ddot{X}^{d} \qquad (5.2.1)$$

where M^d, B^d, K^d are diagonal matrices whose diagonal elements represent the desired mass, damping, and stiffness; S is a diagonal selection matrix which specify the force - $(S_i = 0)$ or position $(S_i = 1)$ controlled axis; F^d, F^e are the desired and interaction forces.

In order to keep the concept of splitting of position and orientation control as described in Section 3.3.2 , the ACTA in the 3-D workspace will be generated separately for position/force-controlled and orientation/torque-controlled axes:

$$\ddot{P}^t(t) = M_p^{d^{-1}}(-F^e + (I - S_p)F^d - B_p^d(\dot{P} - S_p\dot{P}^d) - K_p^d S_p(P - P^d)) + S_p\ddot{P}^d \qquad (5.2.2)$$

$$\dot{\omega}^t(t) = M_o^{d^{-1}}(-N^e + (I - S_o)N^d - B_o^d(\omega - S_o\omega^d) - K_o^d S_o e_o) + S_o\dot{\omega}^d \qquad (5.2.3)$$

where the subscripts p and o indicate that the corresponding variables are specified for position/force-controlled and orientation/torque-controlled subspaces respectively. The superscript d denotes the desired values. The vector $P(3 \times 1)$ and its derivatives are the position, velocity, and acceleration of the origin of {T} expressed in frame {C}; F^d and F^e are the desired and interaction forces expressed in {C}; $S_p(3 \times 3)$ is the selection matrix used to indicate that a {C} frame axis is force- or position-controlled; $\omega, \dot{\omega}$ are the angular velocity and acceleration of the {T} frame expressed in $\{C_i\}$; e_o is the orientation error vector (see Section 3.3.2.2); N^d, N^e are the desired and interaction torques in frame $\{C_i\}$; and M^d, B^d, K^d are diagonal matrices whose diagonal elements represent the desired mass, damping, and stiffness.

Equation (5.2.2) is resolved in frame {C} while Equation (5.2.3) is resolved in frame $\{C_i\}$. The frame $\{C_i\}$ is a time-varying frame (in contrast to frame {C} which is a fixed frame) located at the origin of frame {T} and with same orientation as {C}.

All the inputs and outputs in equations (5.2.2) and (5.2.3) should be expressed in frames {C} and $\{C_i\}$ respectively.

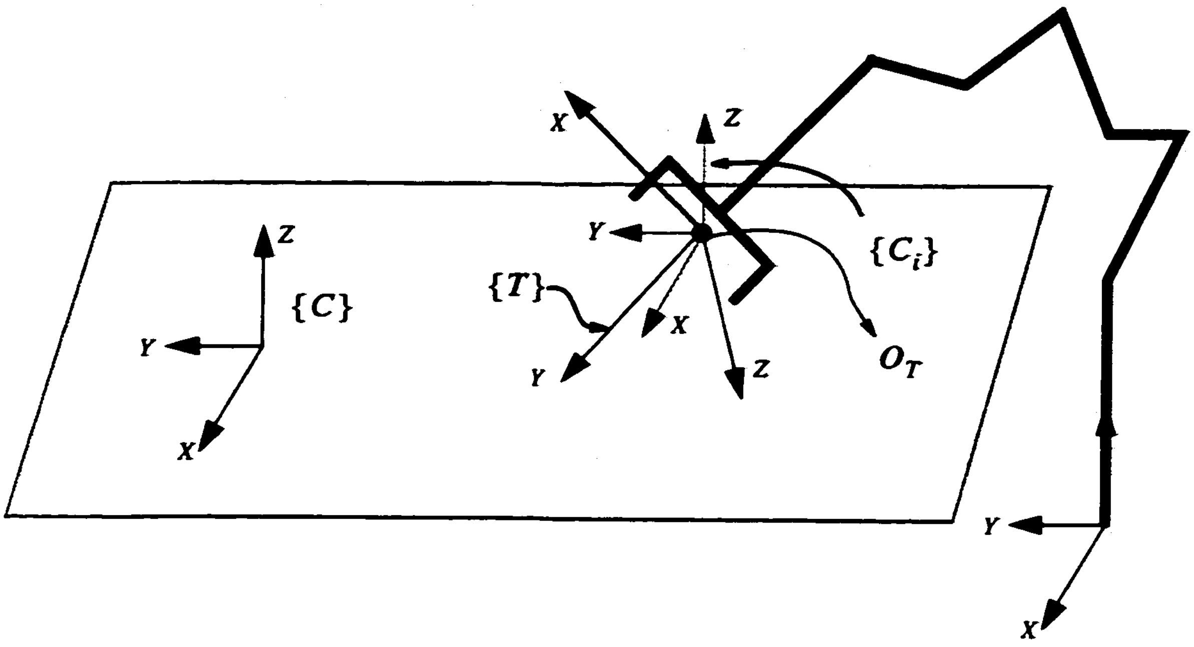


Figure 5.2 Different frames involved in the hybrid task specification

In order to make the AHIC controller module self-contained, all the necessary conversions are implemented in this module.

The location of the origin of {C} in $\{R_1\}$ $({}^{R_1}P_C)$ and the (3×3) rotation matrix ${}^{R_1}R_C$ are specified by the user in a configuration file. It should be noted that the orientations of {C} and $\{C_i\}$ in any arbitrary frame are equal.

5.2.3 Redundancy Resolution (RR) module

The RR module for the AHIC scheme should be implemented at the acceleration level. Assuming an obstacle-free workspace, the damped least-squares solution is given by:

$$\ddot{q}^t = A^{-1}b \tag{5.2.4}$$

where

$$A = J_p^T W_p J_p + J_o^T W_p J_p + J_c^T W_c J_c + W_v$$

$$b = J_p^T W_p(\ddot{P}^t - \dot{J}_p\dot{q}) + J_p^T W_p(\dot{\omega}^t - \dot{J}_o\dot{q}) + J_c^T W_c \dot{Z}^t$$

J_p and J_o are the Jacobian matrices projecting the joint rates to linear and angular velocities of frame {T}. The Jacobian matrices and the two vectors ($\dot{J}_p\dot{q}$, $\dot{J}_o\dot{q}$) are calculated by the forward kinematic module. The matrices W_p, W_o, W_v are the diagonal weighting matrices that assign priority between position/force tracking, orientation/torque tracking and singularity avoidance (in the case of conflicts between these tasks), these matrices are specified by the user in a configuration file. A complete study that demonstrates the effects of the weighting matrices is given in Section 3.3.2.3 . The vectors $\ddot{P}^t$, $\dot{\omega}^t$ are the target linear and angular accelerations of frame {T} expressed in the robot's base frame. These vectors are calculated by the AHIC module. Because the quantities are expressed in the same frame, no coordinate transformation is needed. Note that at this stage, the additional task that is incorporated into the system is joint limit avoidance. For the joint limit avoidance task, the terms $J_c^T W_c J_c$ and $J_c^T W_c$ reduce to W_c (see Section 2.4.1.3). The target acceleration for the *ith* joint in the case of violation of soft-joint limits is defined by:

$$\dot{Z}_i^t = -K_{v_i}\dot{q}_i - K_{p_i}(q_i - q_{m_i}) \qquad (5.2.5)$$

where K_p and K_v are positive-definite proportional and derivative gain matrices, and q_m is the vector of maximum or minimum joint limits.

Computational considerations:

Considering the fact that the matrix A is guaranteed to be positive definite (because of the diagonal weighting matrix W_v), a more efficient way to solve (5.2.5) is to use the Cholesky decomposition. Equation (5.2.4) can be written in the form

$$Ax = b \tag{5.2.6}$$

where $x = \ddot{q}^t$. The Cholesky decomposition of A is given [72] by: $A = LL^T$, where L is a lower-triangular matrix. This reduces to solving an upper and an lower-triangular system of linear equations:

$$Ly = b, \qquad L^T x = y \tag{5.2.7}$$

5.2.4 Forward Kinematics

This module calculates the position and orientation of frame {T}, the linear and angular velocities of {T}, and also the Jacobian matrices relating the linear and angular velocities of {T} to the joint rates. These quantities are expressed in the robot's base frame.

- Tool frame Information

The user only specifies the information to locate frame {T} in frame {7}. Therefore, $Twist(\alpha_7)$, $Length(a_7)$, $Offset(d_7)$, are specified in a configuration file which results to:

$$^7T_T = \begin{bmatrix} 1 & 0 & 0 & a_7 \\ 0 & \cos(\alpha_7) & -\sin(\alpha_7) & 0 \\ 0 & \sin(\alpha_7) & \cos(\alpha_7) & d_7\sin(\alpha_7) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Calculation of $\dot{J}_p\dot{q}$, $\dot{J}_o\dot{q}$

Calculation of two new vectors $(\dot{J}_p\dot{q}, \dot{J}_o\dot{q})$ which are required by the RR module (because of resolving redundancy at the acceleration level) are added to the forward kinematics module. The forward kinematics function at the acceleration level is defined by:

$$\ddot{X} = J\ddot{q} + \dot{J}\dot{q} \tag{5.2.8}$$

which yields

$$\ddot{X}\big|_{\ddot{q}=0} = \dot{J}\dot{q} \tag{5.2.9}$$

This suggests that the following recursive algorithm, which calculates the linear and angular accelerations of the frame {T}, can be used to calculate the vectors $(\dot{J}_p\dot{q}, \dot{J}_o\dot{q})$.

```
for $i = 1 \ldots n+1$
    ${}^{i}\omega_{i-1} = {}^{i}R_{i-1}\,{}^{i-1}\omega_{i-1}$
    ${}^{i}\omega_{i} = {}^{i}\omega_{i-1} + \dot{q}_i z_i$
    ${}^{i}\dot{\omega}_{i} = {}^{i}R_{i-1}\,{}^{i-1}\dot{\omega}_{i-1} + {}^{i}\omega_{i-1} \times \dot{q}_i z_i$
    ${}^{i}\dot{v}_{i} = {}^{i}R_{i-1}({}^{i-1}\dot{v}_{i-1} + {}^{i-1}\dot{\omega}_{i-1} \times {}^{i-1}P_i + {}^{i-1}\omega_{i-1} \times ({}^{i-1}\omega_{i-1} \times {}^{i-1}P_i))$
```

with initial values:

$${}^{0}\omega_0 = [0, 0, 0]^T, {}^{0}\dot{v}_0 = [0, 0, 0]^T \tag{5.2.10}$$

Note that the frames {8} and {T} are the same, and also, the frame {0} is located at the robot's base frame {R1}. Now, Equation (5.2.9) results in:

$$\dot{J}_p\dot{q} = {}^{0}R_T\,{}^{8}\dot{v}_8, \qquad \dot{J}_o\dot{q} = {}^{0}R_T\,{}^{8}\dot{\omega}_8 \tag{5.2.11}$$

5.2.5 Linear Decoupling (Inverse Dynamics) Controller

The equation of motion of a 7-DOF manipulator considering interaction forces/torques with its environment is given by

$$M(q)\ddot{q} + H(q, \dot{q}) + G(q) + f(q, \dot{q}) = \tau - J^T F \tag{5.2.12}$$

where $M(7 \times 7)$ is the symmetric positive-definite inertia matrix of the manipulator in joint space; $H(7 \times 1)$ is the vector of centripetal and Coriolis torques, $G(7 \times 1)$ is the gravity vector, $F(6 \times 1)$ is the interaction force/torque vector exerted by the robot on the environment at the operation point (origin of the tool frame), $J(6 \times 7)$ is the Jacobian matrix relating the linear and angular velocities of the tool frame to joint rates, $f(7 \times 1)$ is the joint friction vector, and $\tau(7 \times 1)$ is the vector of applied torques at the actuators.

The torque that is required to linearize and decouple the nonlinear equation (5.2.12) is given by:

$$t_{LD} = \tau_1 + \tau_2 \tag{5.2.13}$$

where

$$\tau_1 = \hat{M}(q)\ddot{q} + \hat{H}(q, \dot{q}) + \hat{G}(q) + J^T\hat{F} = InvDyn(q, \dot{q}, \ddot{q}^t, \hat{F}) \tag{5.2.14}$$

and

$$\tau_2 = \hat{f}(q, \dot{q}) \tag{5.2.15}$$

where ^ denotes the estimated values.

The optimized *InvDyn* function as well as closed-form representations of M, H, G are developed in C using the Robot Dynamics Modeling (RDM) software [73].

5.3 TESTING AND VERIFICATION

Testing and verification of the codes and integration of the modules has been performed in the *Simulink* environment. First, a *Simulink* module with the same input-output block diagram was built for each of the modules described in Section 5.2. Figure 5.3 shows the block diagrams for such a module.

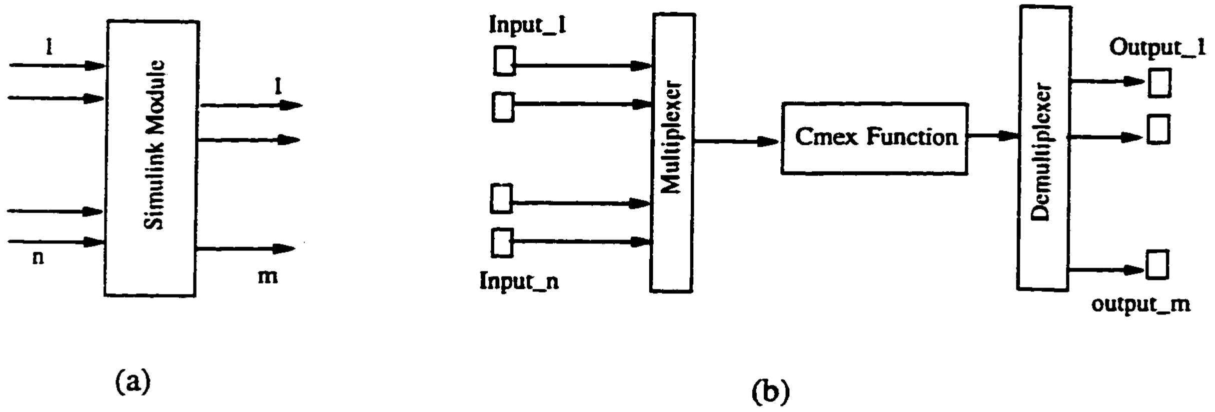


Figure 5.3 Block diagram of a *Simulink* module: a) input/output and b) internal block diagram

The Cmex function (written in *C*) is essentially a gating function which enable *Matlab* to call a *C* function. A Cmex function has the following structure:

```
{static Config_var};
void mexFunction (int nlhs, Matrix *Plhs, int nrhs, Matrix *Prhs)
{
     {input_vars and output_vars declaration};
      if (t = = 0)
               Config_var = ReadCfigFile();
      input_vars = DeMux(*Prhs);
     example(input_vars, Config_var, output_vars);
     *Plhs = Mux(output_vars);
}
```

More information about using the Cmex modules can be found in [76]. Note that using the Cmex module has the following advantages:

- Considering the fact that the simulation environment should include all the controller modules as well as the arm and the environment models, the use of the Cmex module considerably speeds up the simulation which would otherwise be too slow for practical use. Table A-1 gives the approximate execution time of the simulation of the controller without and with arm dynamics which is 5 and 30 times slower than real-time respectively. It should be noted that the kinematic simulation is fast enough for algorithm development and testing. Even the dynamic simulation can be used for short simulations to perform stability and trade-off analysis.

- It reduces considerably the testing and integration phase, because the C source function is being tested and debugged during simulation. Therefore, once the *Simulink* simulation of the controller is verified, the *C* version of the controller can be developed immediately by sequentially calling the *C* functions used in simulation from a controller *C* file.

The "kinematic" simulation developed for the purpose of verifying the integration of the controller is shown in Figure 5.4. The inverse dynamics and the model of the arm are replaced by double integrators. Note that the term "kinematic" emphasizes the fact that we assume perfect knowledge of the manipulator dynamics. However, the model of the environment is still present. The environment is modeled by a linear spring.

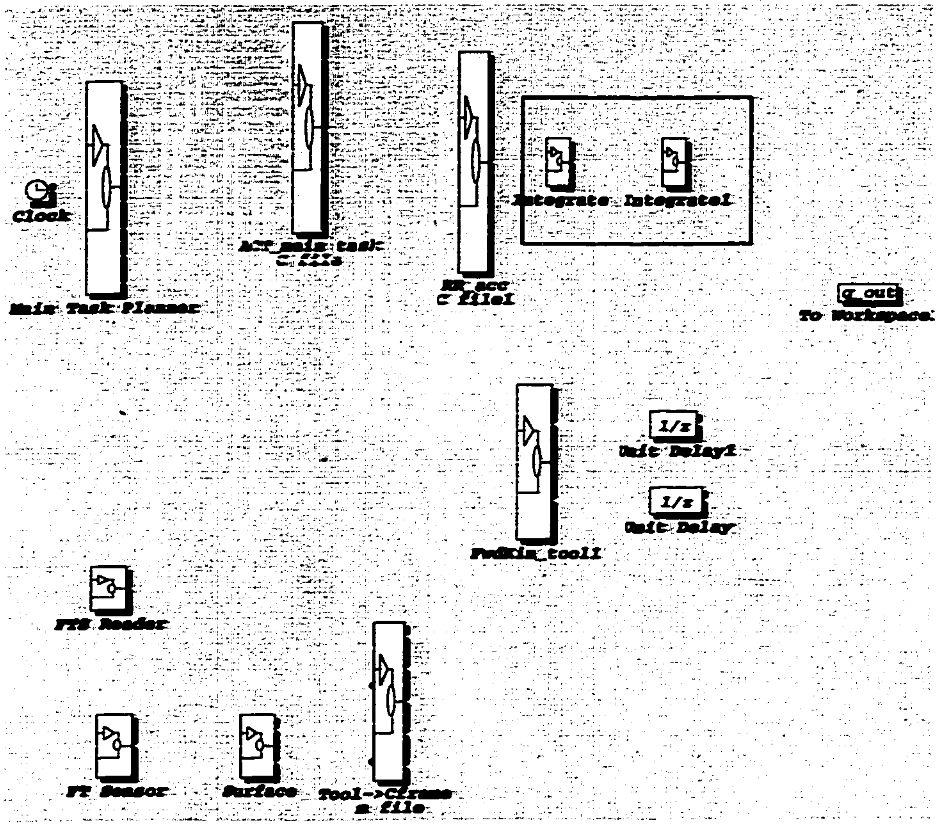


Figure 5.4 *Simulink* kinematic simulation used to verify the integration of the controller.

To verify and test the integration of the controller modules, we recall that if the AHIC scheme is successful, the manipulator acts as a desired impedance in each of the 6 DOF's of the {C} frame. Figure 5.5 shows the desired impedance in position-controlled and force-controlled axes respectively.

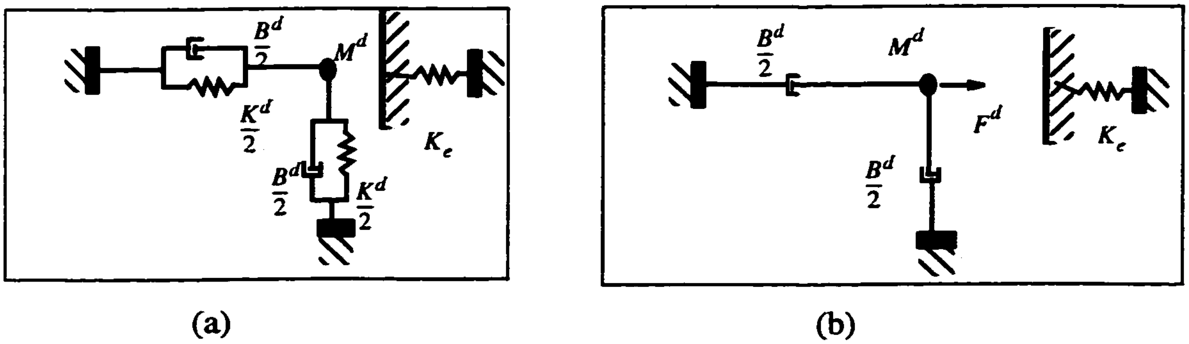


Figure 5.5 Desired impedance a) position controlled axis, and b) force-controlled axis

In order to verify the operation of the AHIC scheme, two simple one-dimensional simulations for the position and force controlled axes were used (see Figure 5.6).

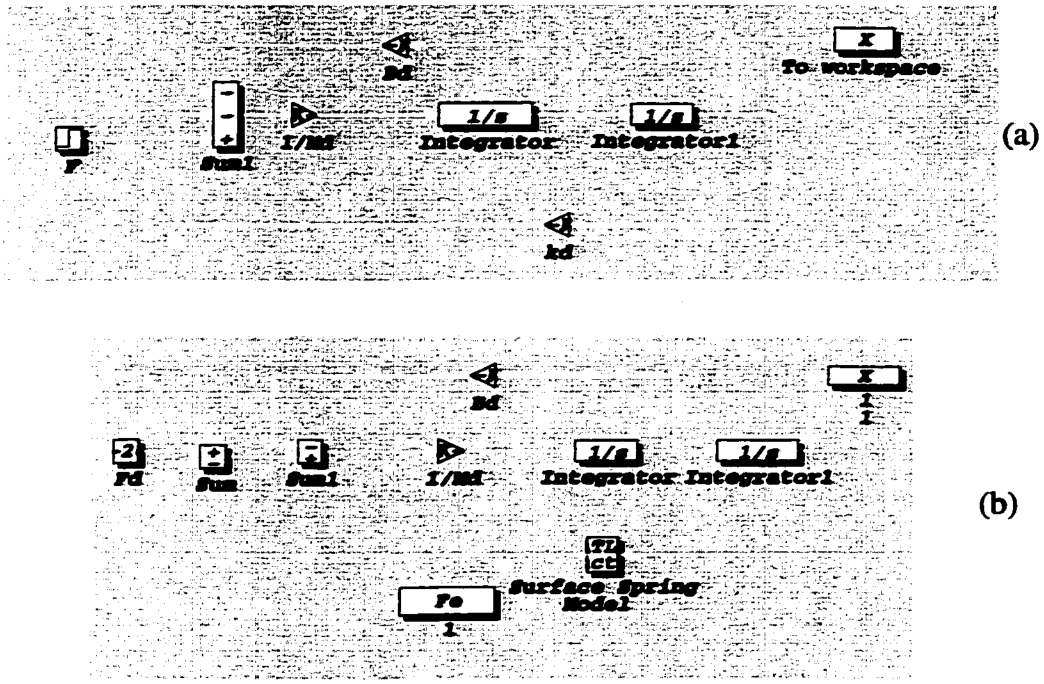


Figure 5.6 *Simulink* one-dimensional simulation of the desired impedance a) position-controlled axis, b) force controlled axis.

Now to check the correct operation of the controller in position-controlled directions, all axes were specified to be in position-control mode. A 1N symmetric step force (in all three X, Y, and Z dimensions of the {C} frame) was applied to both systems (AHIC in Figure 5.4 and impedance simulation in Figure 5.6 a). The desired impedance values can be selected arbitrarily at this stage, because we only need to compare the responses of the two systems. The impedance values used for this test in all 6 DOF's of the {C} frame were:

$$M^d = 257kg, B^d = 1100\frac{Ns}{m}, K^d = 11000\frac{N}{m} \Rightarrow \xi = 0.32, \omega_n = 6.54$$

Figure 5.7 shows the plots of the changes in the position of the origin of the frame {T} along *X* and *Y* axes of the {C} frame.

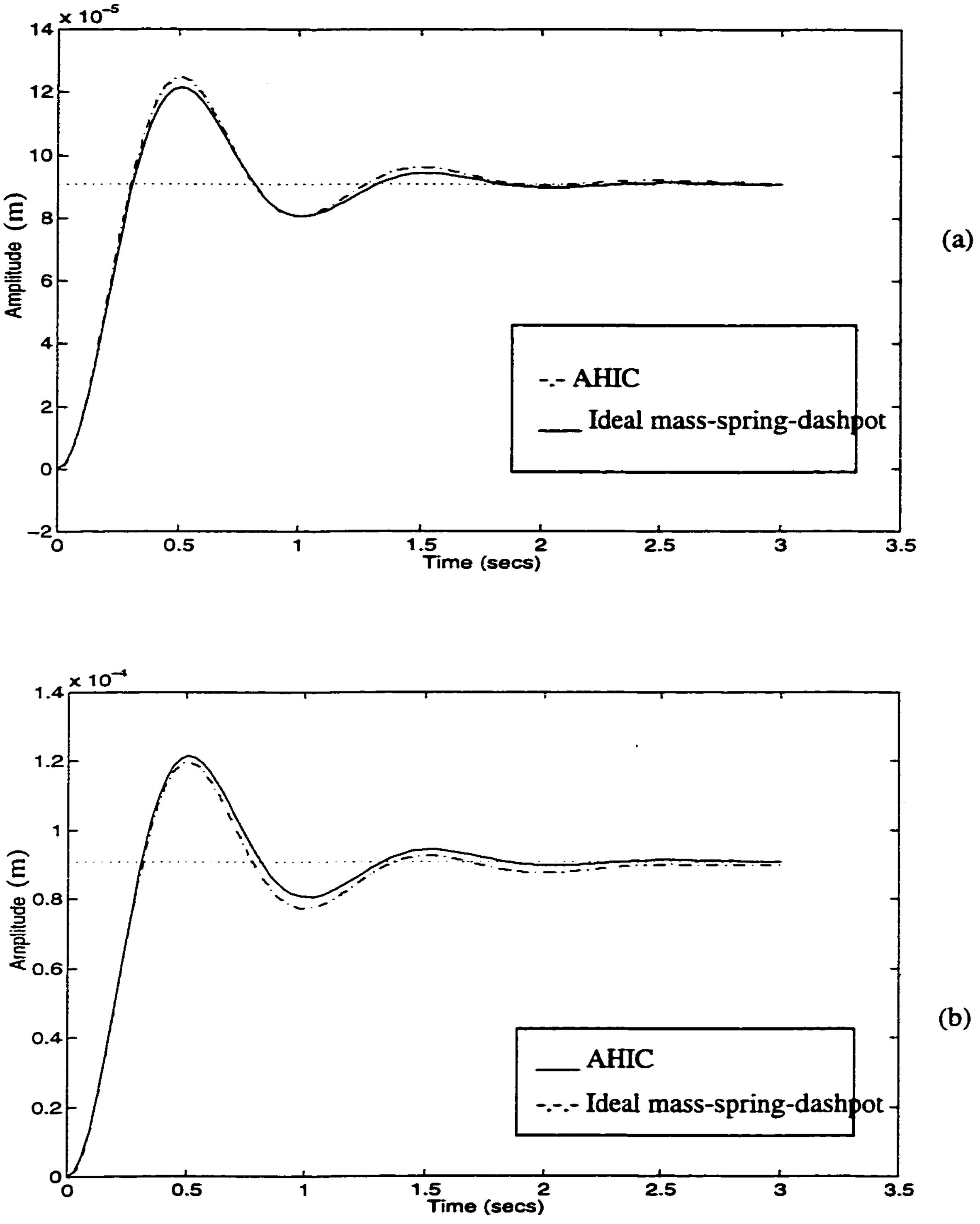


Figure 5.7 Step response in position-controlled directions - Position of the origin of {T} (expressed in {C}) in response to a step force of 1N. a) X axis, b) Y axis

The same test was performed for the force-controlled direction with the following values:

$$M^d = 257kg, B^d = 1100\frac{Ns}{m}, K_e = 11000\frac{N}{m}, F^d = 20$$

Figure 5.8 compares the force history of the AHIC after contacting the surface with that of the pure-impedance simulation in Figure 5.6b.

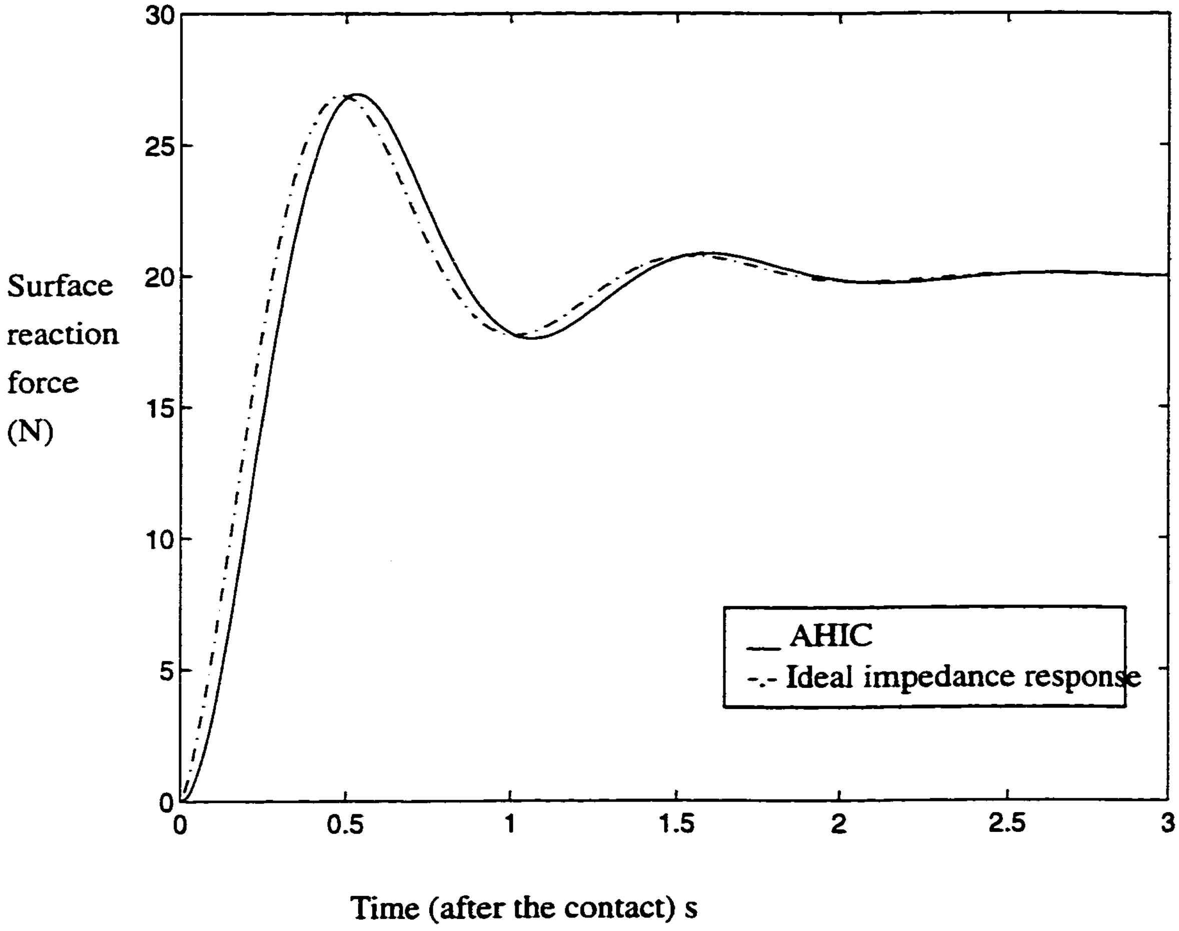


Figure 5.8 Step response in the force-controlled direction (desired force = 20N)

As one can see the response of the AHIC simulation is very close to that of the pure impedance simulation. The possible sources of the small discrepancies are as follows:

- As mentioned in Section 3.3.2.3 , the presence of the singularity robustness term (W_v) introduces some error.
- The simulation of the AHIC scheme is a discrete-time simulation with a trapezoidal integration routine written in C, in contrast to the impedance simulation which is run in continuous-time mode.
- In the AHIC simulation some delays are added to break the "algebraic-loops" (see Figure 5.4). These are not presents in the ideal impedance system simulation shown in Figure 5.6.

Note that the test results up to this point show the correct integration of different modules. Detailed study and analysis of the performance are described in the next section.

5.4 STABILITY ANALYSIS AND TRADE-OFF STUDY

In order to perform the stability analysis and trade-off study, a simulation environment using the Cmex modules has also been created. This study will allow us to identify different sources of instability and performance degradation and finalize the choice of the control scheme to be used in the hardware demonstration. Modification to the AHIC scheme to overcome these problems are presented.

5.4.1 Description of the simulation environment

The simulation was performed in the *Simulink* environment using the Cmex modules (Figure 5.9). The controller modules are described in Section 5.2.

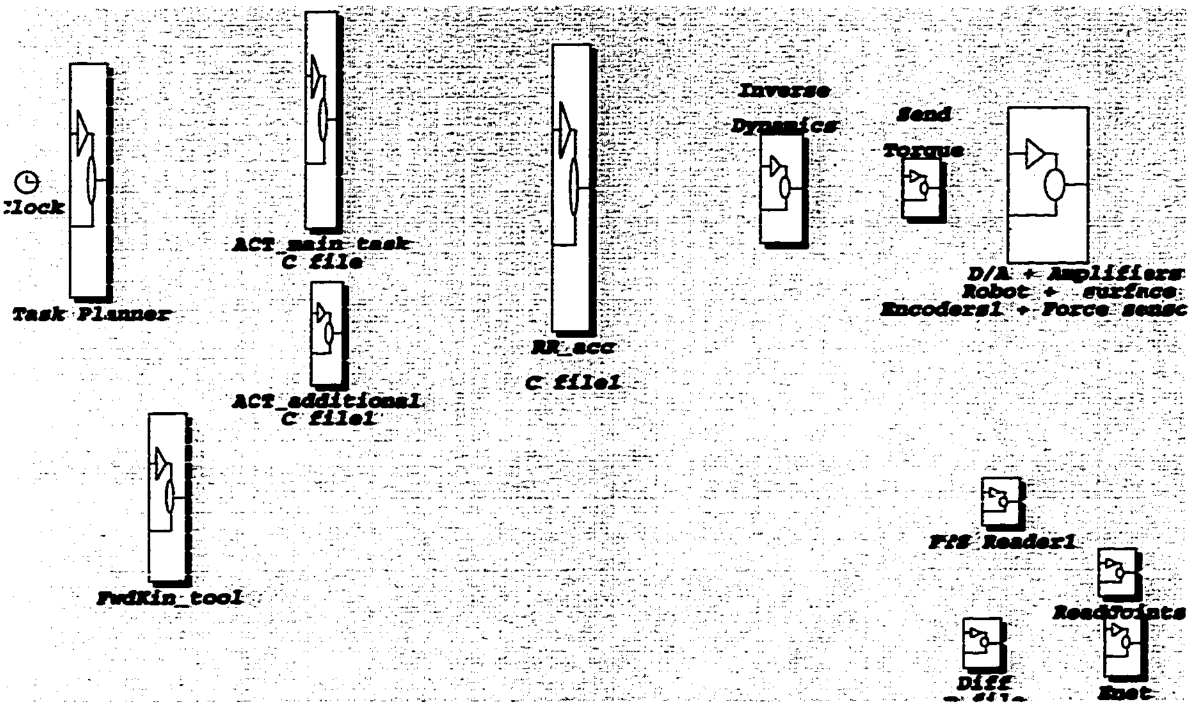


Figure 5.9 Block diagram of the simulation used in the control scheme selection.

The robot model has been developed using the RDM software [73]. It models REDIESTRO and its hardware accessories and covers the following main features:

- Optimized forward dynamic module of the arm;
- Joint friction including stiction, viscous, and Coulomb friction;
- Digitization effects of the A/D converters and encoders;
- Saturation of the actuators and current amplifiers

It also provides some additional features:

- Optimized closed-form representations of the inertia matrix, Coriolis, and Gravity vectors;
- Effect of the external forces;
- Surface and force-sensor models.

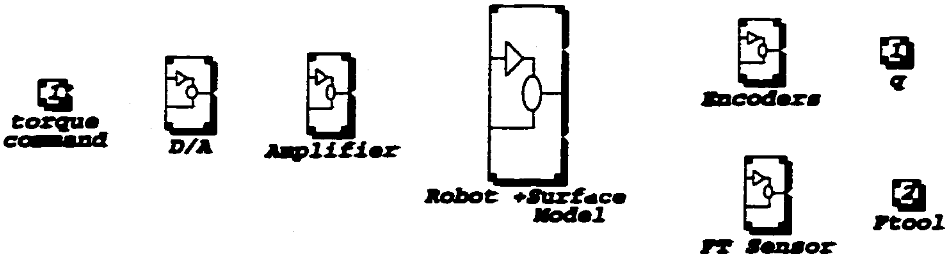


Figure 5.10 Simulation model of REDIESTRO with addition of force sensor and surface models.

In order that the simulation be as close as possible to reality, the simulation is implemented in a mixed discrete and continuous mode. The robot and the surface models use a continuous simulation (Runge-Kutta 5th-order integration), and all other modules are discrete modules with a sampling frequency of 200 Hz.

It should also be noted that the *Enet* module transfers the joint angles and the interaction forces via the network to another SGI workstation which runs the MRS graphical software for on-line 3-D graphics rendering of the movement of the arm.

5.4.2 Description of the sources of performance degradation

In this section by using different simulations and hardware experiments, we determine the sources of degradation in the performance and, in the extreme case, instability, and suggest modifications that can deal with these problems.

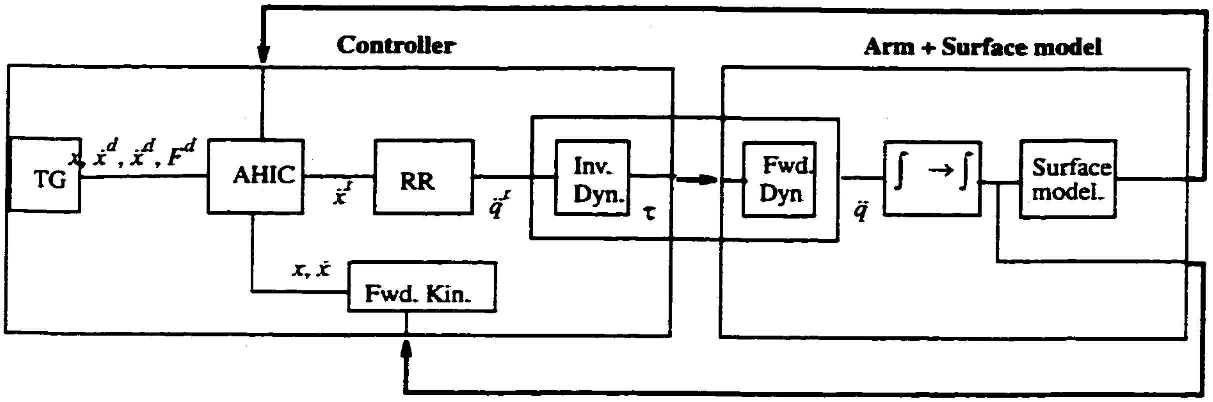


Figure 5.11 Simplified block diagram of the AHIC controller simulation.

Figure 5.11 shows a simplified block diagram of the simulation of AHIC. The major sources of performance degradation and instability are as follows:

- Kinematic instability due to resolving redundancy at the acceleration level
- Performance degradation due to the model-based part of the controller

In the following sections, these problems will first be demonstrated using simulation and/or hardware experiments. Then, the required modifications to the AHIC scheme will be described.

5.4.2.1 Kinematic instability due to resolving redundancy at the acceleration level

In order to focus on this specific problem, we assume that the inverse dynamics part of the controller perfectly decouples the manipulator's dynamics, so that, the arm model can be replaced by a double integrator. Figure 5.12 and Figure 5.4 show the simplified and *simulink* block diagrams respectively).

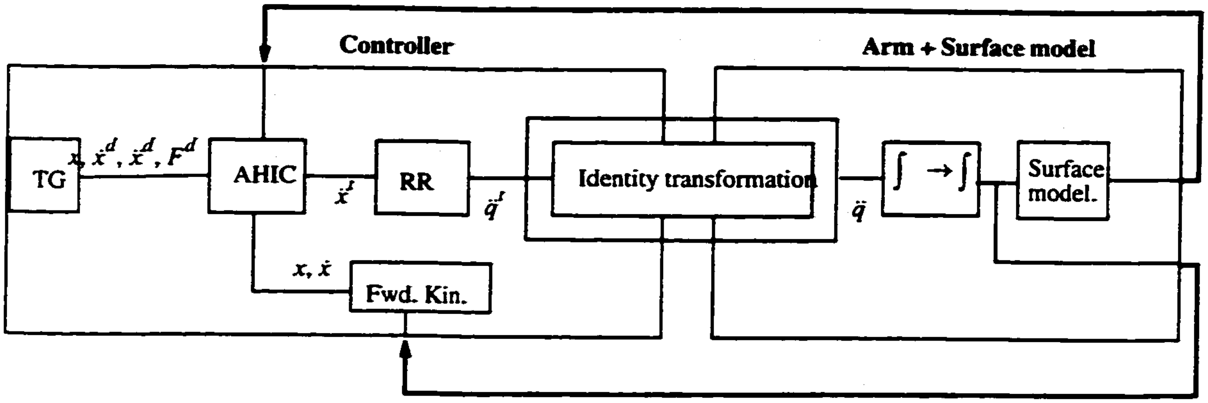
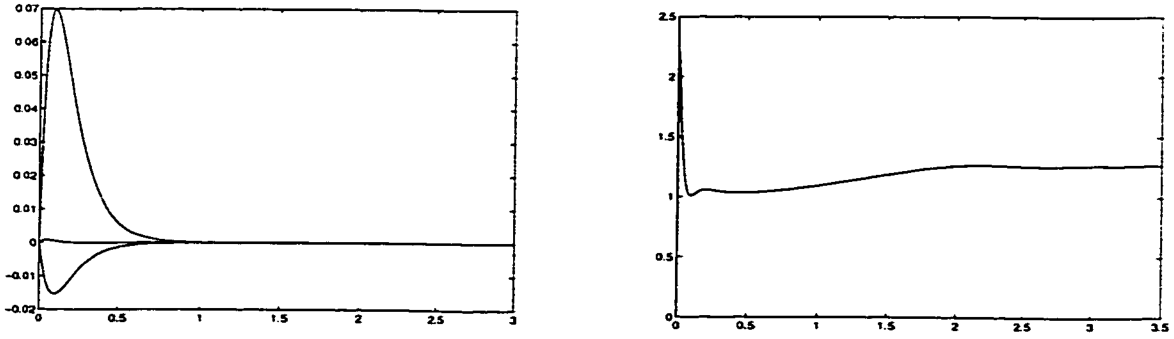


Figure 5.12 Simplified block diagram of the simulation used in kinematic instability analysis

It was previously noted (see Section 4.3.3) that resolving redundancy at the acceleration level has the drawback that the self-motion (joint motion that does not induce any movement in Cartesian space) of the arm is not controlled.

A simulation is performed with non-zero initial joint velocities. The robot is commanded to go from an initial position/orientation to a final position/orientation in 3 seconds and keep the same position/orientation thereafter (the desired velocity and acceleration are zero after 3 seconds). As we can see in Figure 5.13, the robot tracks the trajectory very well. However, the controller is not able to damp out the self-motion component of the joint velocity after reaching the final point.



a) position error (m)

b) norm of the joint velocities (Rad./s)

Figure 5.13 Simulation results with non-zero initial velocity

The following solution can be used:

- Reducing the dimension of the self-motion manifold to zero by specifying additional tasks, e.g. freezing or controlling the value of one of the joints
- Using an improved redundancy resolution scheme at the acceleration level in order to achieve self-motion stabilization [19].
- Modifying the AHIC scheme in order to be able to use redundancy resolution at the velocity level (see Section 4.3.3).

Freezing or controlling the value of one of the joints, is not a preferable option, because that eliminates a desirable redundant degree-of-freedom which otherwise could be used to fulfill additional tasks.

The solutions given for the improved redundancy resolution scheme at the acceleration level are computationally more expensive, because they require the explicit calculation of the derivative of the Jacobian matrix.

The AHIC scheme with self-motion stabilization, proposed in Section 4.3.3 , achieves this goal by modifying AHIC in order to use redundancy resolution at the velocity level. However, the model-based part of the controller (inner-loop) is much more complicated than the computed torque algorithm. The former requires tracking of a reference joint velocity - see equation (4.3.14).

The key idea to solve this problem is to control the velocity. Let us propose the following to control the velocity:

$$\ddot{q} + \lambda\dot{q} = 0 \tag{5.4.1}$$

This suggests a modification of the cost function in (4.3.2) to

$$L = \ddot{E}^T W\ddot{E} + (\ddot{q} + \lambda\dot{q})^T W_v(\ddot{q} + \lambda\dot{q}) \tag{5.4.2}$$

The damped least-squares solution for the new cost function is given by:

$$\ddot{q} = (J^T W J + W_v)^{-1}(J^T W(\ddot{X}^t - \dot{J}\dot{q}) - W_v \lambda \dot{q}) \tag{5.4.3}$$

which in fact penalizes non-zero velocities. To verify the performance of the modified redundancy resolution scheme, a simulation was performed. In order to verify the performance in the worst case, the final position/orientation was selected such that it makes robot's posture approach a singular configuration. This in fact, induces a high null-space component on the joint velocities. Again the robot was commanded to go from an initial position to a final position in 3 seconds. The robot should reach its final position in Cartesian space in 3 seconds. However, there is a large null-space component of the joint velocities that remains uncontrolled when $\lambda = 0$. Increasing the value of λ damps out these components (Figure 5.14).

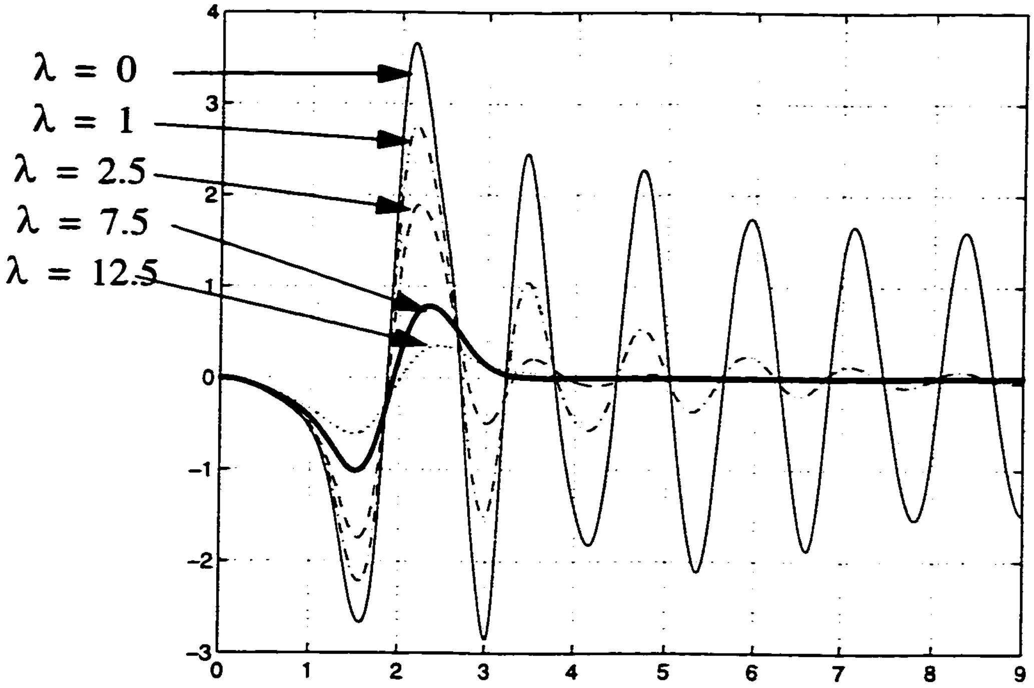


Figure 5.14 Simulation result for modified RR scheme - Joint 2 velocity (rad).

In order to study the effect of λ on tracking error, another set of simulations was performed. Figure 5.15 shows the results of these simulations. As in the previous simulations the desired position is reached after 3 seconds. For $\lambda = 5$, the velocity fades away with large oscillations. With $\lambda = 50$ the velocity fades away with no overshoot. However, there are larger tracking errors. A choice of $\lambda = 10$ gives the best result considering both tracking and velocity damping. Based on our experience a value of λ between 7.5 and 12.5 was found to be suitable for most cases.

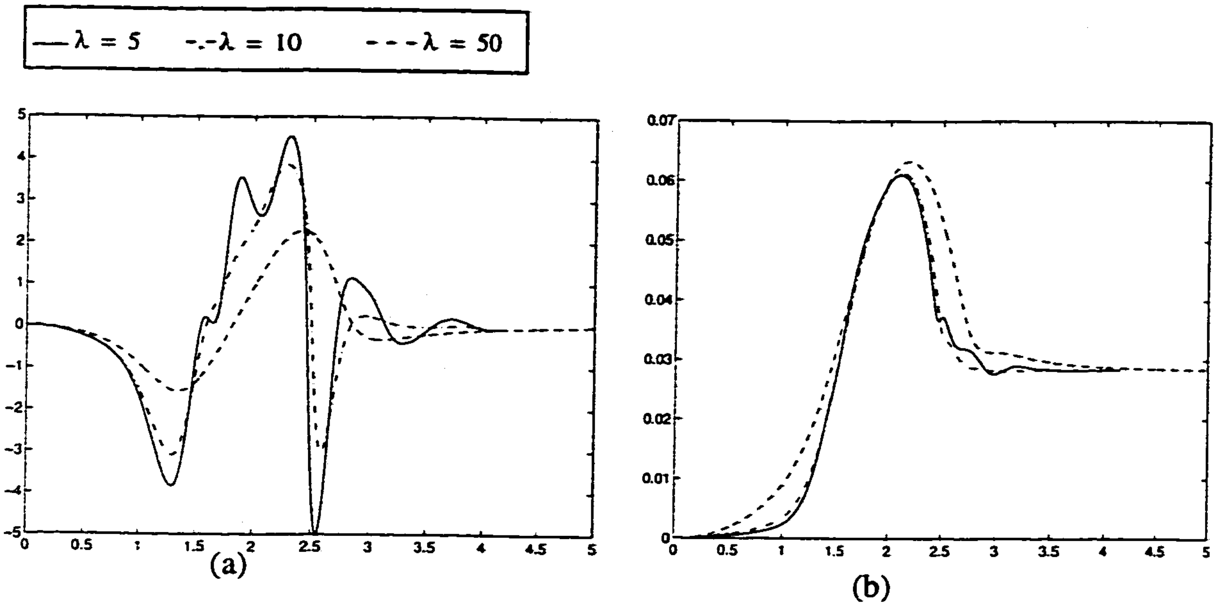


Figure 5.15 Comparison between different values of damping factors in the RR module. a) Joint 2 rate (rad), and b) norm of position error (m)

5.4.2.2 Performance degradation due to the model-based part of the controller

In order to focus on this specific problem, first let us consider the simpler case of a Linearized-Decoupled PD (LDPD) joint0-space controller as shown in Figure 5.16.

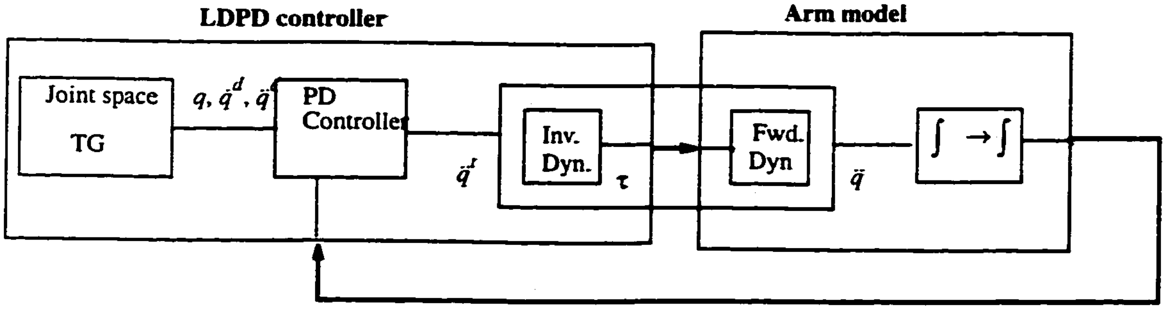


Figure 5.16 Block diagram of the LDPD controller

The model based part of the controller decouples and linearizes the manipulator's dynamics if both the model and the parameters used in the controller are perfect. However, in reality there are different sources of parameter and model mismatch. Some of the major sources of performance degradation of a model-based controller are listed below:

- Friction compensation (model & parameters)
- Unmodeled dynamics (e.g. joint flexibility)
- Imprecise dynamic and kinematic parameters
- Initial joint offsets

Simulations and hardware experiments were used to study the effects of these sources on the tracking performance. It should be noted that in order to distinguish the performance of the model-based part of the controller from the PD part, we do not select high gain values in the following simulations and experiments ($K_p = 10, K_v = 6.3$).

Figure 5.17 shows the simulation results of the LDPD controller (Joint 2) when the same friction model and parameters are used in the controller and the manipulator model. The errors essentially converge to zero.

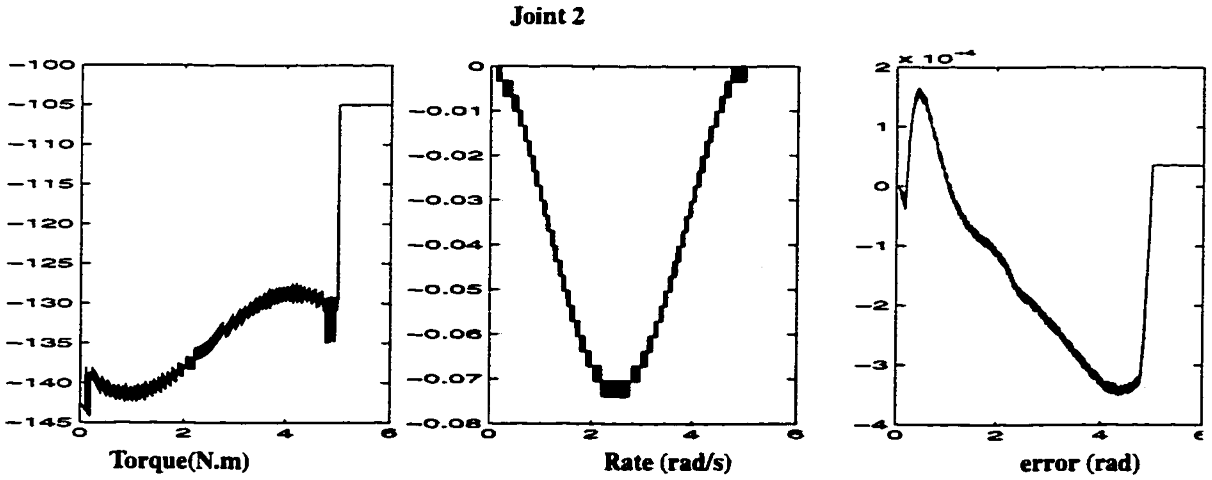


Figure 5.17 Simulation results for LDPD controller using the same parameters and model in the inverse dynamic controller and the manipulator model ($K_p = 10, K_v = 6.3$).

The simulation was repeated using an estimate of joint friction values greater that those used in the manipulator model $\tilde{f}_{fric} = 1.3 f_{fric}$. The results shown in Figure 5.18 indicate the degradation in tracking. Hardware experiment results confirm the result of the simulation (see Figure 5.19).

In order to achieve better tracking performance with the LDPD controller, two solutions are available:

- Increasing the feedback gains
- Better parameter identification

The first solution will improve tracking. However, it will decrease robustness because of the risk of exciting the higher-order unmodeled dynamics, e.g. joint flexible modes. The second option also improves tracking. However, there is a limit to the accuracy level of identification for different manipulators. Moreover, these parameters may change with time and the initially identified parameters may not be accurate after a certain time. This is practically important in space applications where, after launching the arm, periodic identification of parameters may not be feasible.

It should be noted that the AHIC scheme, shown in Figure 5.1, does not include any "control gains". The only gains in the AHIC scheme are the "impedance gains". The difference is that the control gains can be selected arbitrarily based on the accuracy level of the modelling and tracking requirements. However, the criteria for selecting the desired impedances (impedance gains) are dictated differently, e.g., by surface dynamics, stability considerations for the force control loop, etc.

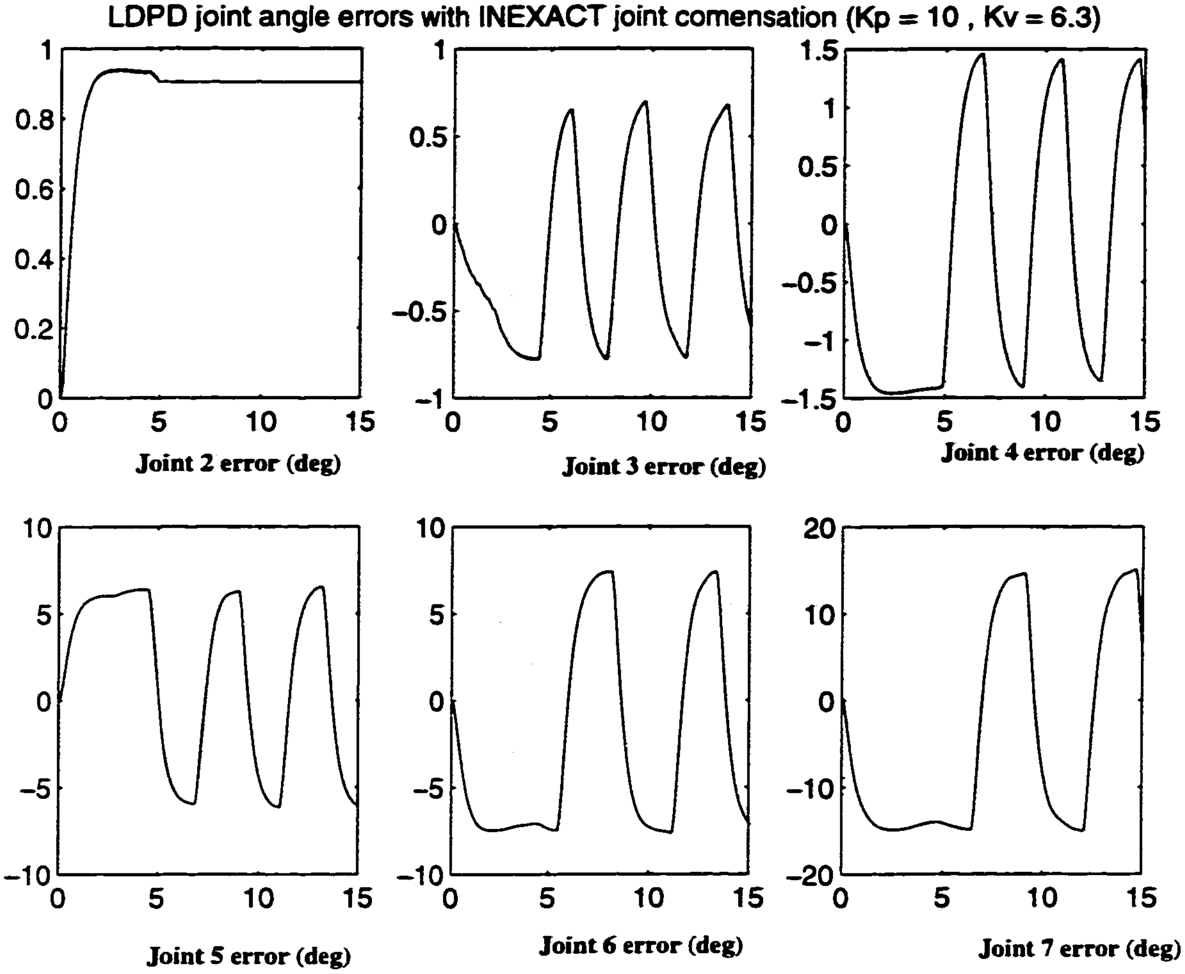
From the above statements, one can conclude that a good way of improving the performance of the AHIC scheme is a combination of the following steps:

1-Adding a PD feedback loop in the AHIC scheme

2- "Better" parameter identification

- Refinement of the friction compensation module
- Fine tuning of friction coefficients
- Accurate home positioning

The following section describes the modification to the AHIC controller.



Joint 2 error (deg)

Joint 3 error (deg)

Joint 4 error (deg)

Joint 5 error (deg)

Joint 6 error (deg)

Joint 7 error (deg)

Figure 5.18 LDPD controller with inexact friction compensation $\tilde{f}_{fric} = 1.3 f_{fric}$

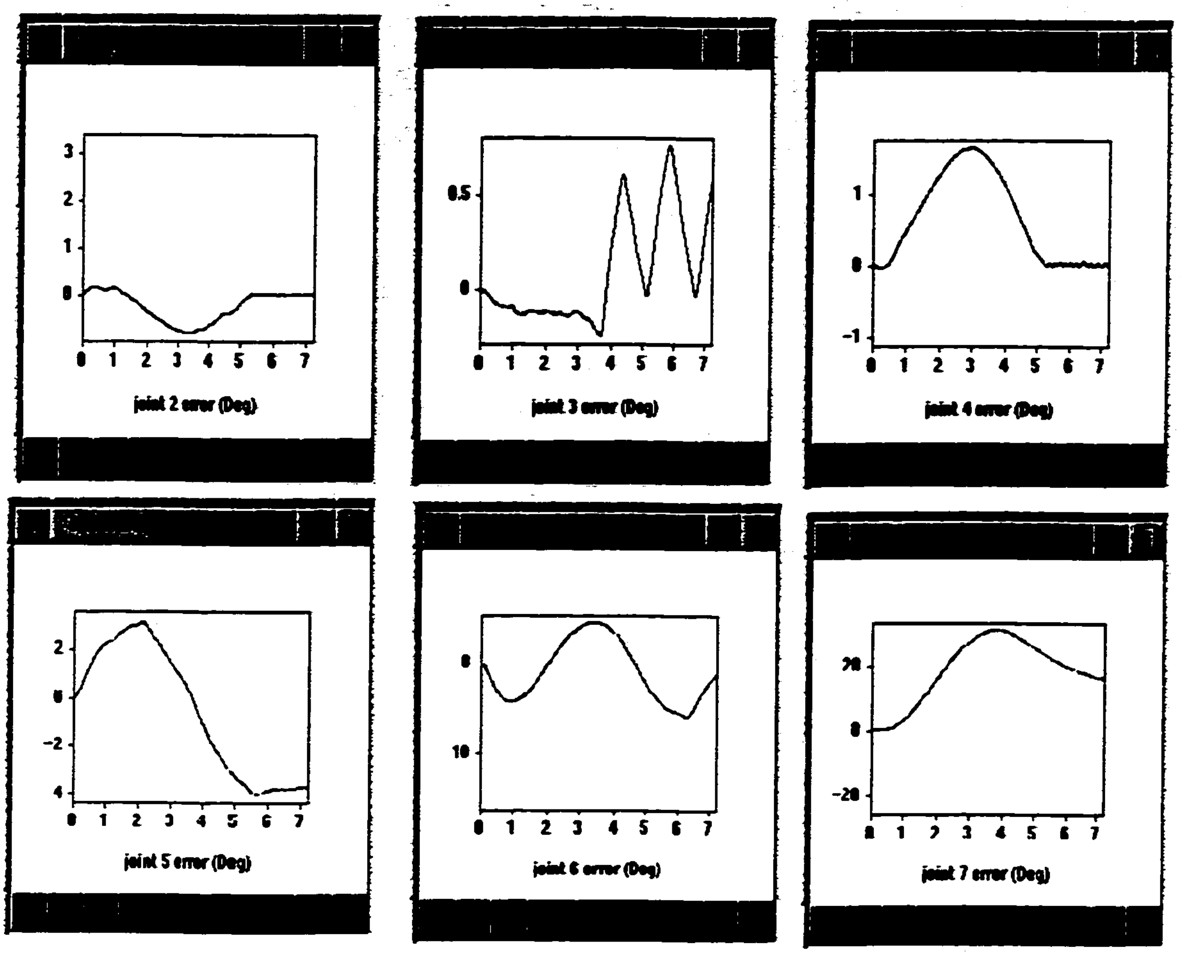


Figure 5.19 LDPD controller experimental results $K_p = 10, K_v = 6.3$.

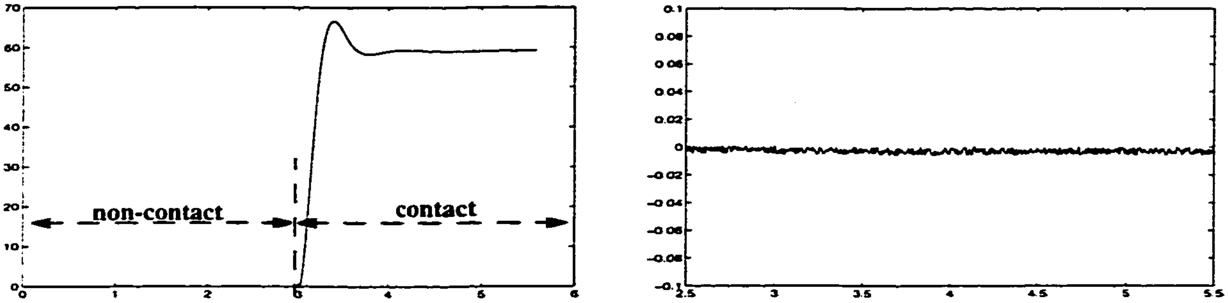
5.4.3 Modified AHIC Scheme

Section 5.4.2.2 indicated the problem associated with the model based controller using a simple example of a joint-space LDPD controller. Now, we study the same problem using the complete simulation of the AHIC scheme (see Figure 5.9). These simulations contain two segments: free motion and contact-motion. In the first segment, the tool frame is commanded to move from an initial position to a final position located on the constraint surface in 3 seconds. The second segment consists of keeping the final position (along x and y) and orientation while exerting a 60 N force on the surface. Table 5-1 summarizes the values used in the simulations.

Table 5-1 Desired values used in the AHIC simulation (z - axis)

seg.	S	M (kg)	B (Nsec/m)	K (N/m)	Fd (N)	Surface K (N/m)	Desired Eq. of motion
non contact	1	1	20	100	–	–	$M(\ddot{X}-\ddot{X}^d)+B(\dot{X}-\dot{X}^d)+K(X-X^d) = -F_e$
contact	0	100	1000	–	60	10000	$M\ddot{X}+B\dot{X}-F^d = -F_e$

Figure 5.20 shows the force tracking when joint friction is not included in the joint models. As one can see, there is a small difference between the joint target acceleration command ($\ddot{q}^t$) and the actual joint acceleration. This results in perfect tracking of the response of the desired impedance in the z direction.

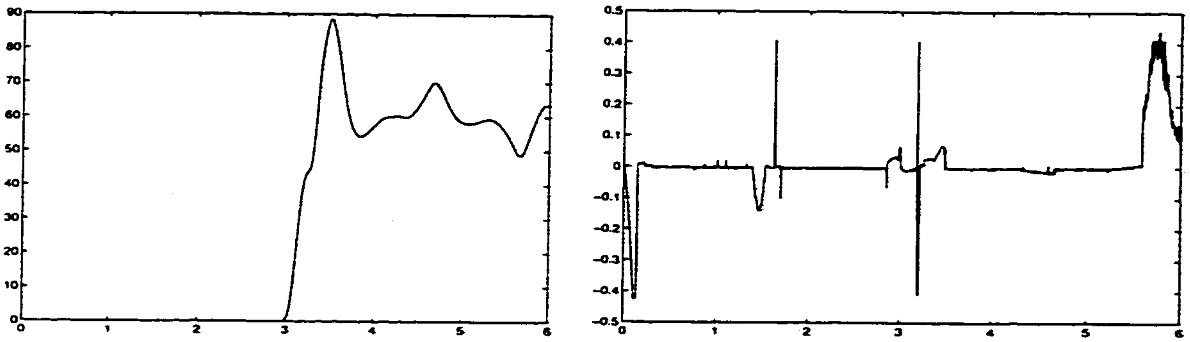


a) F_z

b) $\ddot{q}_2{}^t - \ddot{q}_2$

Figure 5.20 AHIC controller without joint friction

Now, the simulation is repeated by including friction in the manipulator model. The friction compensation uses the same model with exact parameters. Figure 5.21 b shows the error between the target and the actual joint accelerations. Although both the friction model and the friction compensation module use the same model and parameters, the friction compensation uses discretized data (e.g. joint velocities) while the friction model uses continuous values. Figure 5.21a shows the tracking degradation resulting from this slight mismatch between the model and model-based controller. As we mentioned earlier, the selection of the desired impedances are based on other criteria. Hence, they cannot be changed to deal with this problem.



a) F_z

b) $\ddot{q}_2^t - \ddot{q}_2$

Figure 5.21 AHIC simulation with friction in the model and friction compensation in the controller (using the same friction parameters as in the model)

Adopting a similar scheme to that proposed in Section 4.3.3 , a solution to this problem is to add a PD feedback loop. Figure 5.22 shows the block diagram of the modified controller. The following modifications have been performed:

- The Error Reference Controller (ERC) module which generates a Cartesian Reference Acceleration (CRA) has been added

- The position feedback which used to go to the AHIC module is now connected to ERC

- The complete target trajectory $(x^t, \dot{x}^t, \ddot{x}^t)$ is generated online using force sensor feedback

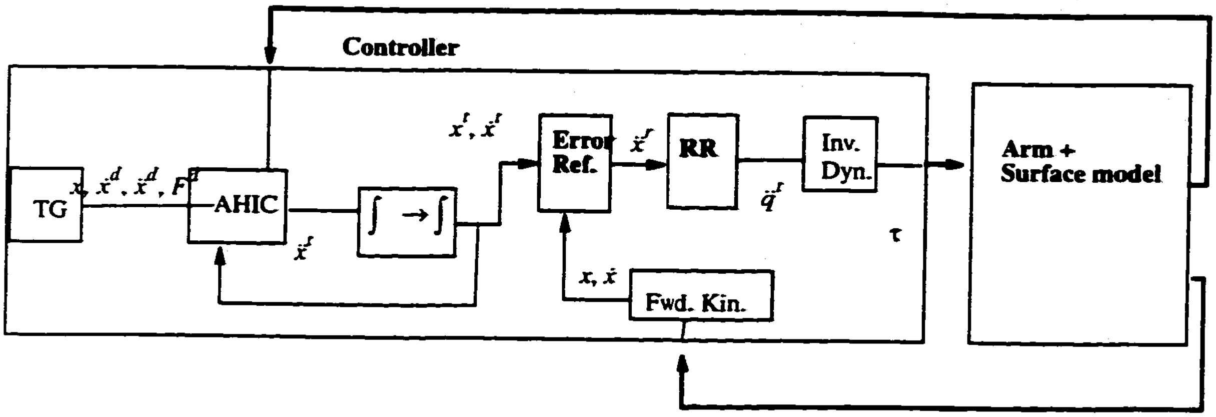


Figure 5.22 Simplified block diagram of the modified AHIC controller

Figure 5.22 shows the new/modified modules which are shaded in gray. Table 5-2 summarizes the modified equations.

Table 5-2 Summary of equations for new/modified modules

Module	Equation
AHIC	$\ddot{X}^t = {M^d}^{-1}(-F^e + (I-S)F^d - B^d(\dot{X}^t - S\dot{X}^d) - K^d S(X^t - X^d)) + S\ddot{X}^d$
REC	$\ddot{X}^r = \ddot{X}^t + K_v(\dot{X}^t - \dot{X}) + K_p(X^t - X)$
RR	$\ddot{q}^t = (J^T W J + W_v)^{-1}(J^T W(\ddot{X}^r - \dot{J}\dot{q}) - \lambda W_v \dot{q})$

At this stage another level of algorithm development was performed for the new/modified modules and functions. The complete simulation of the modified AHIC scheme was developed in the *Simulink* environment (see Figure 5.23) to study the performance of the modified scheme.

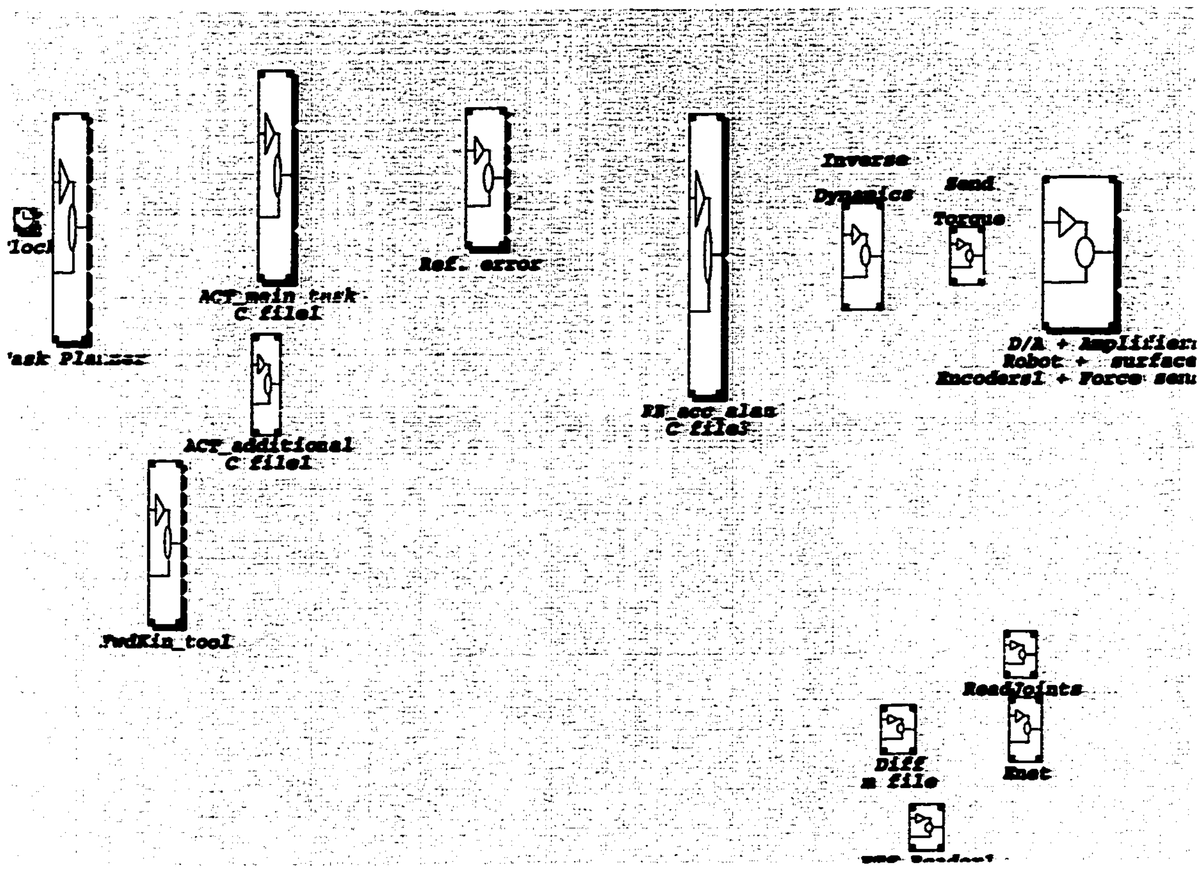


Figure 5.23 Block diagram of the complete *Simulink* simulation of the modified AHIC scheme

The simulations consist of 5 segments which are summarized in Table 5-3. The PD gains are chosen as $K_p = 100, K_v = 20$. The results of the original AHIC scheme are compared with the modified AHIC scheme. No joint friction compensation is performed to study the robustness of the algorithms.

Table 5-3 Desired values used in the modified AHIC simulation (z - axis)

seg.	S	M (kg)	B (Nsec/m)	K (N/m)	Fd (N)	Surface K (N/m)	final_time (S)	Comment
1	1	1	20	100	--	--	3	non-contact
2	0	100	1000	--	60	10000	6	contact
3	0	100	1000	--	80	10000	7	contact
4	0	100	1000	--	40	10000	10	contact
5	0	100	1000	--	0	10000	13	contact

Figure 5.24 shows the comparison between the force tracking performance of the AHIC scheme as shown in Figure 5.1, and that of the modified AHIC scheme. As one can see, even without performing friction compensation, the modified AHIC is able to regulate the interaction force (with limited error). However, the original AHIC scheme is completely incapable of regulating the force. Note that force tracking can be greatly improved by selecting the appropriate impedance values (this will be explained in Section 6.3.1).

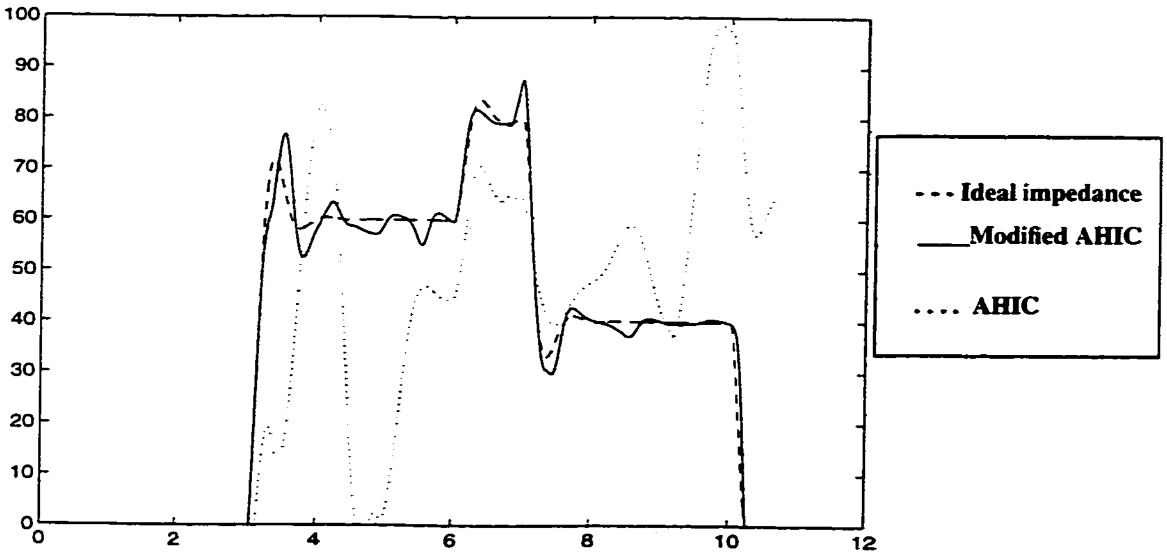


Figure 5.24 Comparison between the original AHIC with modified AHIC (without friction compensation).

5.5 CONCLUSIONS

As indicated in the introduction, the objective of this chapter was to extend the AHIC scheme to the 3D workspace of a 7-DOF manipulator (REDIESTRO); to develop and test the AHIC software; and to demonstrate by simulation the performance of the proposed scheme. From the foregoing sections, the following conclusion can be drawn:

1. The conceptual framework presented for compliant force and motion control in the 2D workspace of a 3-DOF planar manipulator, is adequate to control a 7-DOF redundant manipulator working in a 3D workspace.

2. The algorithm extension for the AHIC scheme and the required modules have been successfully developed for REDIESTRO.

3. The software development of different modules has been successfully accomplished. The code has been optimized in order to achieve real-time implementation.

4. At this stage, only joint limit avoidance has been incorporated into the redundancy resolution module. The simulation results for joint limit avoidance provide confidence that other additional tasks such as obstacle avoidance can be incorporated without major difficulty.

5. The realistic dynamic simulation environment has enabled us to study issues such as performance degradation due to imprecise dynamic modelling and uncontrolled self-motion.

6. The least-squares solution for redundancy resolution at the acceleration level was modified by adding a velocity-dependent term to the cost function. This modification successfully controlled the self-motion of the manipulator.

7. It was demonstrated by simulation that the force tracking performance of the methods based solely on inverse dynamics degrades in the presence of uncertainty in the manipulator's dynamic parameters and unmodelled dynamics. This is especially true for a manipulator equipped with harmonic drive transmissions, which introduce a high level of joint flexibility and frictional effects (As in the case of REDIESTRO).

8. The AHIC scheme has been modified by incorporating an "error reference controller". This modification successfully copes with model uncertainties in the model-based part of the controller, so that even friction compensation is not required. The modified scheme can be considered a major contribution of this thesis and increases the applicability of this type of control to a large class of industrial and research manipulators.

The above conclusions indicate that we can now proceed to the next stage to demonstrate the capabilities of the AHIC scheme by hardware demonstration using REDIESTRO. This is done in Chapter 6.

CHAPTER 6

HARDWARE EXPERIMENTS ON CONTACT FORCE AND COMPLIANT MOTION CONTROL

6.1 INTRODUCTION

In this chapter, we describe the hardware experiments performed to evaluate the performance of the proposed AHIC scheme in compliant motion and force control of REDIESTRO.

Considering the complexity and the large amount of calculations involved in a force and compliant motion control for a 7-DOF redundant manipulator, the implementation of the real-time controller, from both hardware and software points of view, by itself represents a challenge. It should be noted that there are very few cases in the literature that experimental results for force and compliant motion control of a 7-DOF manipulator have been reported. In [78], a set of experiments on contact force control carried out on a 7-DOF Robotics Research Corporation (RRC) model K1207 arm at the Jet Propulsion Laboratory is reported. It should be noted that the RRC arm is one the most advanced manipulators from both mechanical design and controller view points. On the other hand, implementation of the AHIC scheme for REDIESTRO introduces additional challenges which are listed below:

1. The REDIESTRO arm is equipped with harmonic drive transmissions which introduce a high level of joint flexibility. This makes accurate control of contact force more difficult.

2. The friction model and its parameters cannot be estimated accurately in many practical applications. The friction model that is mostly used models load independent Coulomb and viscous friction. This model is especially inadequate for a robot with harmonic drive transmissions which have high friction - experimental results show that in some configurations, the friction torques reach up to 30% of the applied torques. Also, different experimental studies [79] have shown that frictional torques in harmonic drives are very nonlinear and load dependent which makes accurate parameter identification difficult. This represents a challenge for a model-based controller.

3. Performing tasks such as peg-in-the hole requires a very accurate positioning. This needs a very well-calibrated arm. In [80], Colombina et al. described the development of an impedance controller at the External Servicing Test-bed which is a ground test-bed currently installed at the European Space Agency Research Center. The performance of the impedance controller was demonstrated for a replacement of an Orbital Replacement Unit (ORU). They reported that only misplacement of 5 mm in position and 0.5 degrees in orientation are compensated for in an ORU exchange task. Considering the fact that REDIESTRO has not been kinematically calibrated, the successful operation of the peg-in-the-hole strawman task by REDIESTRO will demonstrate a high level of robustness of the proposed scheme.

The goal of this chapter is to demonstrate the feasibility and evaluate the performance of the proposed scheme by hardware demonstrations using REDIESTRO. Before performing the final hardware demonstrations, a detailed analysis is given to provide guidelines in the selection of the desired impedances. A heuristic approach is presented which enables the user to systematically select the impedance parameters based on stability and tracking requirements.

At this stage different scenarios are considered and two strawman tasks - surface cleaning and peg-in-the-hole - are selected. The selection is based on the ability to evaluate force and position tracking and also robustness with respect to knowledge of the environment and kinematic errors. These strawman tasks are also similar to the tasks that will be performed by the SPDM in space. These tasks are window cleaning and ORU insertion and removal. Finally, numerical results for these strawman tasks are presented.

6.2 HARDWARE AND SOFTWARE CONFIGURATION

REDIESTRO was relocated from the Centre for Intelligence Machines (CIM) at McGill University, where the hardware experiments for collision avoidance were performed, to Concordia University for further development and experiments of compliant motion and force control schemes. Note that the only control mode supported by the controller at CIM was independent joint PID control (See Appendix D). The hardware and software configuration of the controller has gone through several levels of change to meet the requirements for force and compliant motion control. Appendix E describes the final hardware (see Figure 6.1) and software configurations developed at Concordia University.

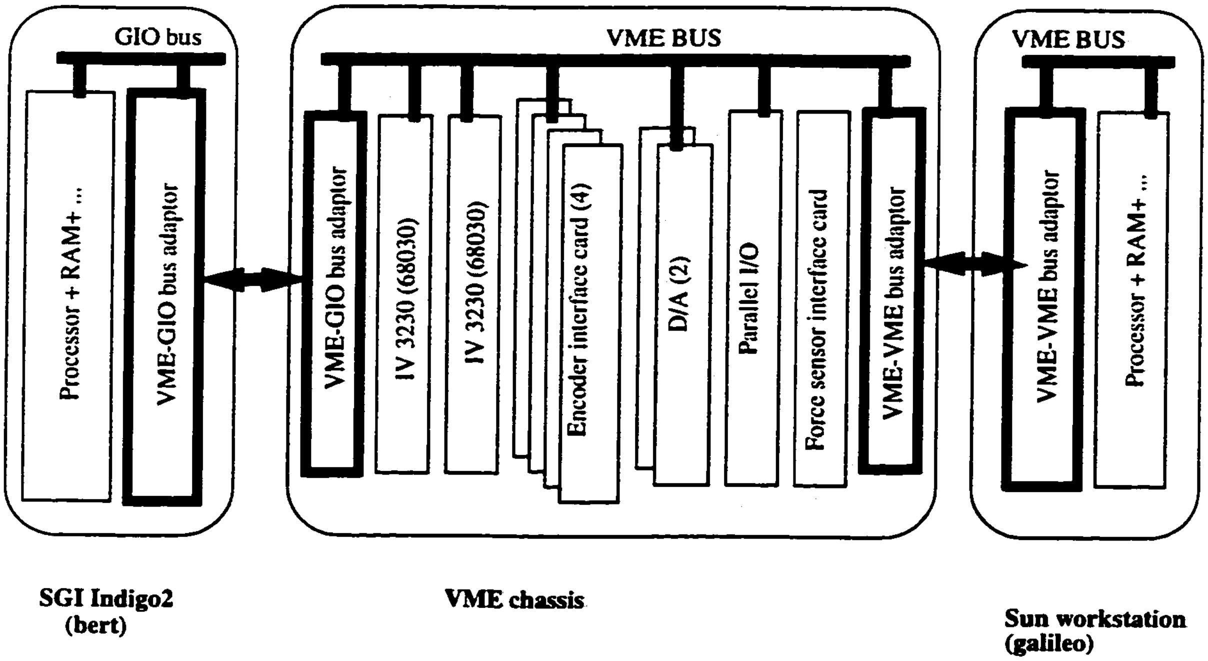


Figure 6.1 Final hardware configuration (force control experiments)

6.3 PREPARATION AND CONDUCT OF HARDWARE EXPERIMENTS

6.3.1 Selection of Desired Impedance

The desired equation of motion in a position (impedance)-controlled direction is given by:

$$m^d \ddot{e} + b^d \dot{e} + k^d e = -f_e \tag{6.3.1}$$

where $e = x - x^d$. The desired equation of motion in a force-controlled direction is given by:

$$m^d \ddot{x} + b^d \dot{x} = f^d - f_e \tag{6.3.2}$$

The environment is modeled as a linear spring. Therefore, the interaction force in (6.3.2) can be replaced by $f_e = k_e x$, which results in

$$m^d \ddot{x} + b^d \dot{x} + k_e x = f^d \tag{6.3.3}$$

Comparing the desired equation of motion in a position (impedance) controlled direction (6.3.1) with that of a force-controlled direction (6.3.3), one notices that the same guidelines for selection of impedance gains which ensure both stability and tracking performance can be used. The main difference is that in an impedance-controlled direction the stiffness is an adjustable control parameter which can be specified while in a force-control direction the stiffness is an environmental parameter which is not selectable. The complete stability analysis study and the guidelines to select the set of impedance parameters to ensure stability of the motion considering the delays in the force and position sensor loops and also stiffness of the contact are given in this section.

6.3.1.1 Stability Analysis

As we mentioned above, the same guidelines can be followed for both impedance- and force-controlled directions. Therefore, we consider the following generic system:

$$m\ddot{x}+b\dot{x}+kx = f \tag{6.3.4}$$

Equation (6.3.4) can be expressed (in Laplace transforms) as

$$s^2X(s)+2\xi\omega_n sX(s)+\omega_n^2X(s) = F(s) \tag{6.3.5}$$

where

$$\omega_n = \sqrt{\frac{k}{m}}, \qquad \xi = \frac{b}{2\sqrt{km}}, \qquad F = Laplace\left(\frac{f}{m}\right) \tag{6.3.6}$$

Now, let us introduce a delay element in the sensor (feedback) loop. Equation (6.3.5) yields

$$s^2X+2\xi\omega_n e^{-2T_s s}sx+\omega_n^2 e^{-2T_s s}X = F \tag{6.3.7}$$

The delay element $e^{-2T_s s}$ can be replaced by its approximation $e^{-2T_s s} = \frac{1-sT_s}{1+sT_s}$. Now the characteristic equation of (6.3.5) is expressed by:

$$T_s s^3+(1-2\xi\omega_n T_s)s^2+(2\xi\omega_n-\omega_n^2T_s)s+\omega_n^2 = 0 \tag{6.3.8}$$

According to the Routh stability criterion, the system expressed by (6.3.7) is stable (all roots of (6.3.8) are in the left-half of the complex plane) if and only if all coefficients of the first column of the Routh table have the same sign; This leads to

$$\omega_n<\frac{2\xi}{T_s} \qquad and \qquad \omega_n<\frac{1}{2\xi T_s} \tag{6.3.9}$$

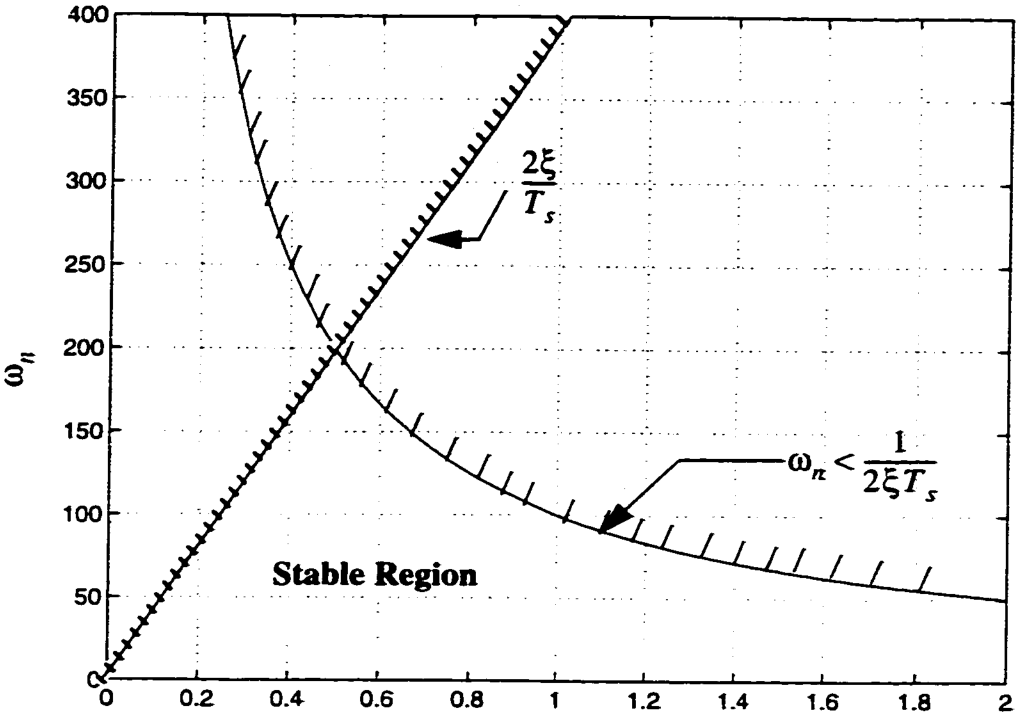


Figure 6.2 Stability region of the system represented by Equation (6.3.7) with $T_s = 0.005$ seconds.

6.3.1.2 Impedance-controlled Axis

The desired equation of motion is given by (6.3.1). In this case, the desired mass, damping, and stiffness should be specified. The following steps are required:

- Based on the sampling and sensor delays, select ξ and ω_n such that the stability condition (according to Figure 6.2) is satisfied.
- Select the desired stiffness according to the acceptable steady-state error:

$$e_{ss} = \frac{-f_e}{k^d} \tag{6.3.10}$$

where f_e is the disturbance force in a position-controlled direction such as the friction force on the surface for a surface cleaning scenario.

- Calculate the desired inertia and damping using:

$$m^d = \frac{k^d}{\omega_n^2} \tag{6.3.11}$$

$$b^d = \frac{2\xi k^d}{\omega_n} \tag{6.3.12}$$

In order to study the step response of the controller in an impedance-controlled direction, the following experiment was conducted. All axes were specified to be impedance-controlled for the segment between *t = 110s* and *t =115s*. The desired position trajectory is specified such that there was a difference of 13 cm between the initial desired position along the *z-axis* and the initial tool frame *z* position. The desired impedances for the z-axis are specified by: $m^d = 112, b^d = 700, k^d = 1100$ which correspond to $\zeta = 1, T_n = 2s$. Figure 6.3 compares the hardware experiment result with that of the ideal system of mass-spring-dashpot.

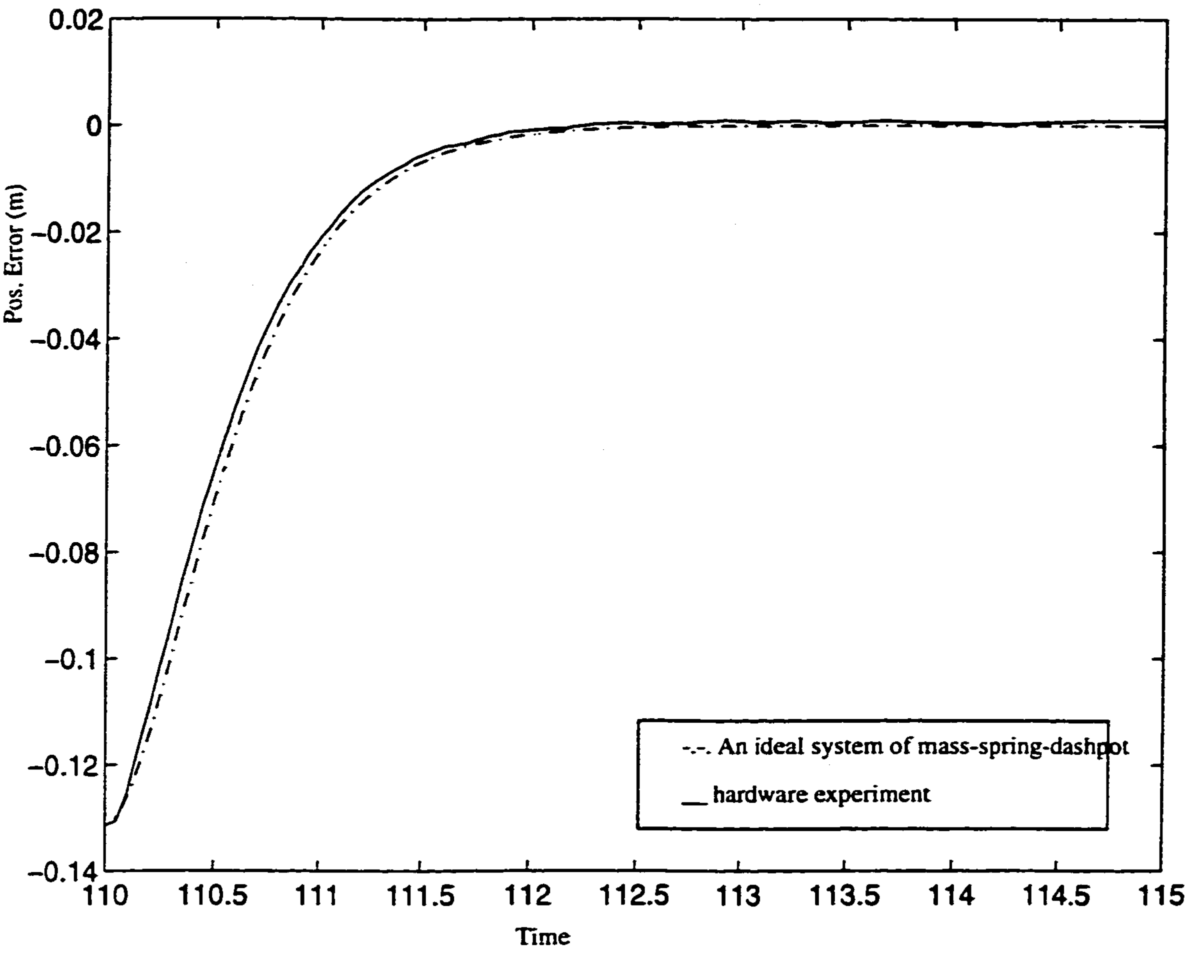


Figure 6.3 Position step response in an impedance-controlled direction (13 cm initial position error).

The desired impedances for the position (impedance)-controlled axes during the surface cleaning and the peg in hole experiments were selected as $m^d = 257, b^d = 1100, k^d = 1100$ which correspond to $\zeta = 1.03, T_n = 3.03s$.

6.3.1.3 Force-controlled Axis:

The desired equation of motion is given in (6.3.2). The desired mass and damping should be specified. In contrast to an impedance-controlled axis where the stiffness is an adjustable control parameter, in this case the stiffness k_e is the overall stiffness of contact. The contact stiffness is affected by following factors:

- Tool stiffness; the eraser pad in the case of surface cleaning and the plexi-glass peg in the case of peg in the hole.
- Environment stiffness; white-board table and its support in the case of surface cleaning and the plexi-glass hole in the case of peg in the hole.
- Transmission (joint) flexibility; the flexibility of harmonic drives.
- Structural (link) flexibility

Therefore, in order to assign ξ and ω_n for the force-controlled axis, one should know the overall stiffness of contact. Although difficult to determine, the stiffness of the tool and environment can be identified by off-line experiments; joint and link flexibilities are even more difficult to identify and characterize. Note that the force tracking steady-state error in (6.3.2) is not affected by the stiffness k_e as long as the system remains stable. However, the transient response varies with k_e. In conducting the hardware experiments, a heuristic approach has been used which allows us to achieve the desired steady-state and transient performance without an elaborate procedure to identify and characterize the overall stiffness of contact.

- Based on the estimate of the delay in the force sensor loop, select ξ and ω_n such that the stability condition according to Figure 6.2 is satisfied. The major delay in this case is due to the low-pass force sensor filter with cutoff frequency f_c equal to 7.81 Hz. The filter delay is approximately given by:

$$Delay = \frac{1}{f_c} = \frac{1}{7.81} = 0.128S \qquad (6.3.13)$$

- Based on a very conservative estimate of contact stiffness k_e, select m^d and b^d as in (6.3.11) and (6.3.12) respectively. Note that in order to have a conservative estimate of contact stiffness, one should select a higher stiffness than what is expected. This can be justified by studying the stability criterion given in this section. Equation (6.3.6) shows that ω_n increases with increasing values of k_e with the risk of violating the stability conditions given in (6.3.9).

In this case, for $\zeta = 1$, the stability margin is determined by:

$$\omega_n < \frac{1}{2\zeta T_s} \Rightarrow \omega_n < \frac{1}{2Delay} \Rightarrow \omega_n < 3.9 \qquad (6.3.14)$$

We select $\omega_n = 2.5rad/s$, Assuming $k_e = 10000N/m$ which is a conservative estimate considering the high values of joint flexibility, we calculate the desired impedances: $m^d = 1600Kg$ and $b^d = 8000Ns/m$. The first set of hardware experiments were conducted using these impedances.

The test scenario (Figure 6.4) consists of the first three segments of the surface cleaning strawman task (see Section 6.4.2). In the first segment, the eraser pad is positioned above the white-board table (all axes under position control) in 5 seconds. In the second segment, the eraser approaches the surface along the z-axis under force control, while keeping the position along the x and y axes fixed. The desired force along the z-axis is 20N. In the last segment, the eraser is commanded to move along the y axis on the surface with a desired 20N force.

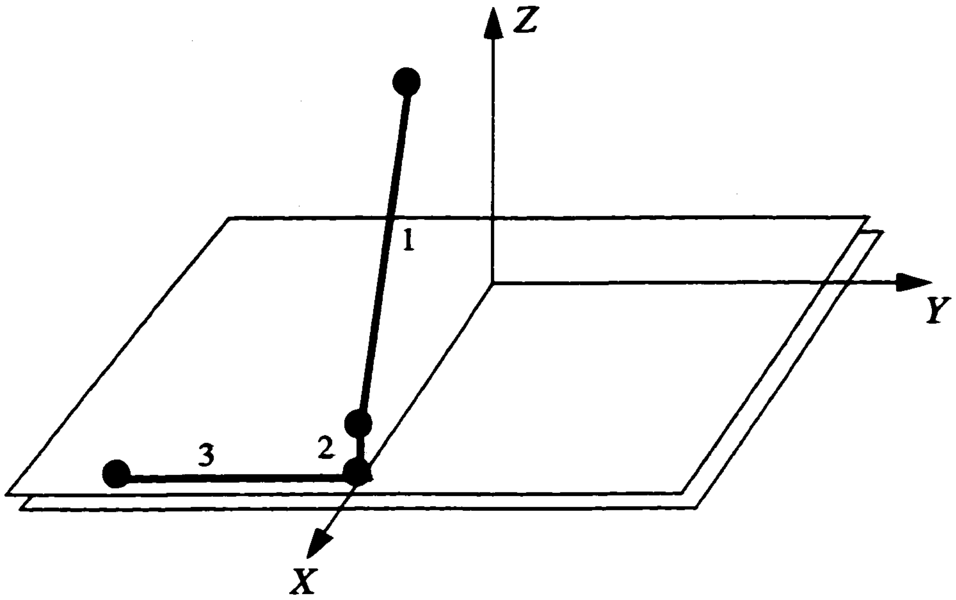


Figure 6.4 Test scenario for selecting the desired impedance in the force-controlled direction for the first strawman task

Figure 6.5 shows the plot of the interaction forces. The response time of the system (~10s) is greater than the expected value (2.53 s based on $k_e = 10000\frac{N}{m}$), which shows that the actual stiffness of the contact is much less than the estimated value.

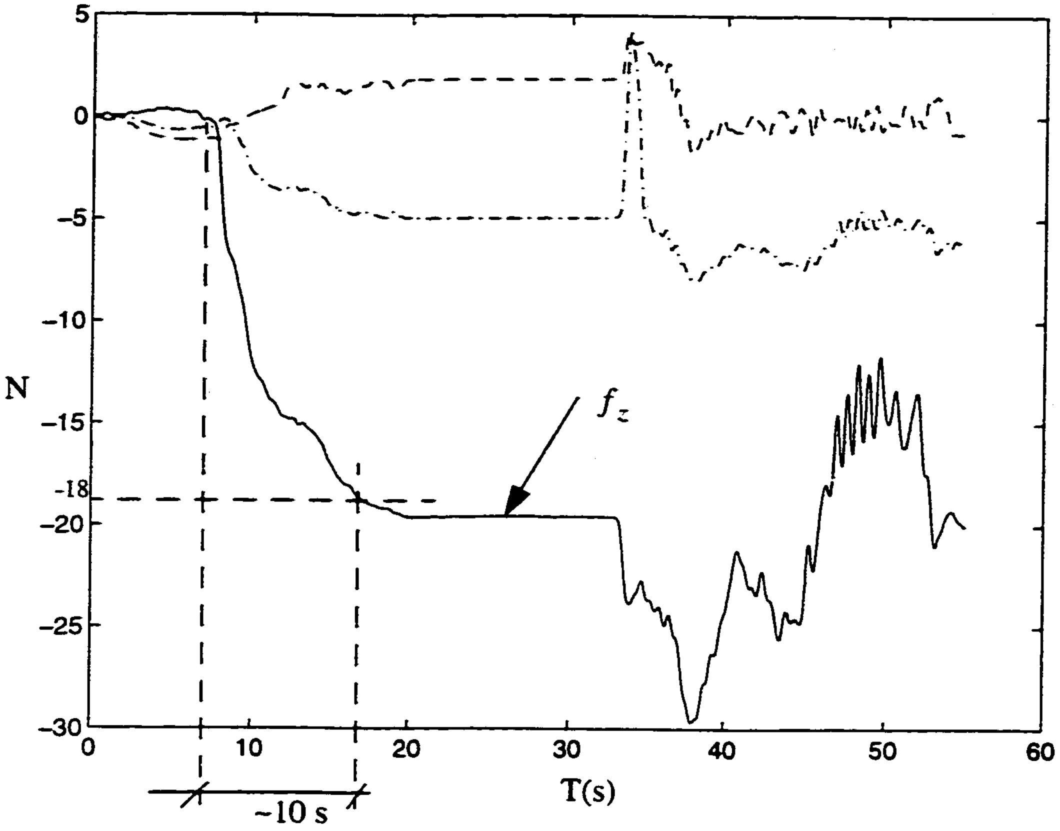


Figure 6.5 Force tracking for the test case shown in Figure 6.4 with $m^d = 1600, b^d = 8000$.

The heuristic approach of selecting the desired impedances is based on studying the actual response in hardware experiments. As an example, let us calculate a set of impedances which results in a response time that is twice as fast (5 seconds) compared to the 10 seconds in Figure 6.5. Assuming a fixed contact stiffness k_e, Equation (6.3.6) results in

$$\frac{\omega_{n_1}}{\omega_{n_2}} = \sqrt{\frac{m_2}{m_1}} = \frac{T_{n_2}}{T_{n_1}} \qquad (6.3.15)$$

Equation (6.3.15) suggests that in order to reduce T_n by a factor of 2, the desired mass should be reduced by a factor of 4. Equation (6.3.6) also results in

$$\frac{\xi_1}{\xi_2} = \sqrt{\frac{m_2}{m_1}} \cdot \frac{b_1}{b_2} \tag{6.3.16}$$

which suggests that the desired damping should be reduced by a factor of 2 in order to keep ξ constant. The next experiment was conducted using the desired inertia and damping calculated by (6.3.15) and (6.3.16) respectively ($m^d = 400, b^d = 4000$).

Figure 6.6 shows that the transient response of the system is changed. The response time is approximately two times faster than the previous case in Figure 6.5.

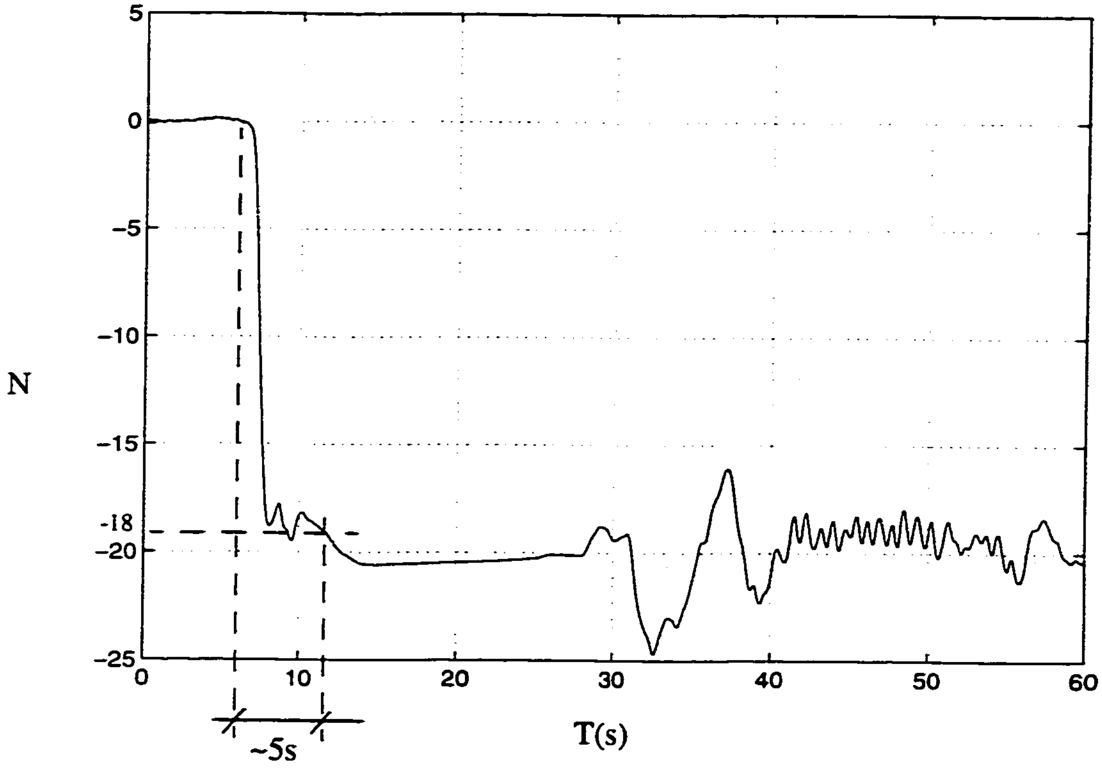


Figure 6.6 Force tracking for the test case shown in Figure 6.4 with $m^d = 400, b^d = 4000$.

Note that in both of the above experiments, the force steady-state error during segment 2 (force exertion without moving on the surface) is very small. However, the force tracking performance degrades rapidly in segment 3 where the pad starts to move on the surface.

This problem can be attributed to the unmodeled joint flexibilities. When the eraser pad is exerting a force without moving on the surface, the sole joint motion is due to the force controller along the z-axis which will eventually reach an equilibrium point when the desired force is achieved. However, when the eraser pad is commanded to move on the surface, even though the desired force is achieved along the z-axis, there are joint motions required for the movement on the surface. Without the unmodeled dynamics due to joint flexibilities, the motions along the position-controlled directions and the force-controlled directions are decoupled. Therefore, the horizontal movement on the surface should not affect the force tracking along the z-axis. However, any joint oscillation due to unmodeled joint flexibilities acts as a coupling between the position-controlled directions and the force-controlled directions which causes performance degradation in force tracking (see Figure 6.5).

The force controller in (52) can be seen as a second-order filter (see Figure 6.7) with a corner frequency of $\sqrt{k_e/m^d}$.

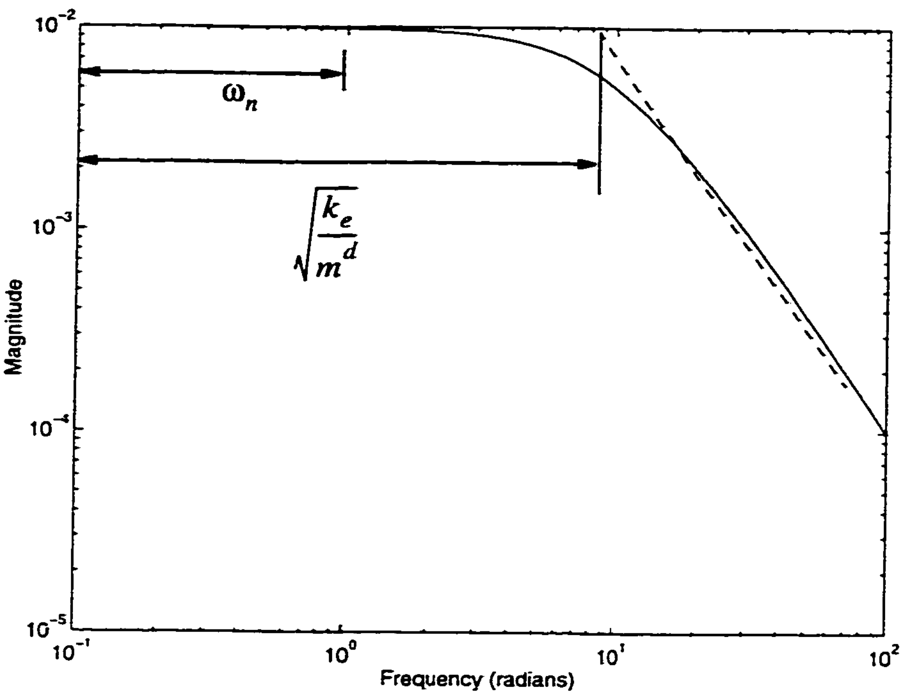


Figure 6.7 Plot of magnitude versus frequency for the second-order filter in Equation (6.3.3).

Now, in order for the force-controller in (6.3.2) to reject these disturbances (the forces due to joint oscillations), the cutoff frequency of this filter should be selected much greater than the frequency of the disturbances (in this case oscillations caused by joint flexibilities). In order to increase the cutoff frequency, one should reduce the desired inertia (m^d) to as small a value as possible while maintaining system stability. The values of the desired inertia and damping were selected experimentally as $m^d = 5.7, b^d = 477$.

6.3.2 Selection of PD Gains

In the modified AHIC scheme (see Figure 5.22) a PD controller was implemented to ensure that the reference error (error between the target trajectory generated by the AHIC controller and tool frame trajectory) converges to zero. Therefore, in order for the robot to act as closely as possible to the ideal impedance system specified by (6.3.1) and (6.3.2), the PD gains need to be selected as high as possible. Different experiments were conducted to find the best values for the PD gains. The maximum values that do not excite the unmodeled dynamics were obtained experimentally as $k_p = 400, k_v = 40$.

6.3.3 Selection of Force-Filter

The force sensor data usually contain a high level of noise which needs to be filtered out by implementation of a low-pass filter. The selection of the filter is a trade-off between noise rejection and stability requirements as the low-pass filter introduces a delay in the sensor loop which can cause instability. The JR3 force-sensor interface card provides a cascade of low-pass filters. Each succeeding filter has a cutoff frequency that is 1/4 of that of the preceding filter. For the JR3 sensor with a sample rate of 8 kHz, the cutoff frequency of the first filter is 500Hz. The subsequent filters cutoff at 125 Hz, 31.25 Hz, 7.81 Hz and so on.

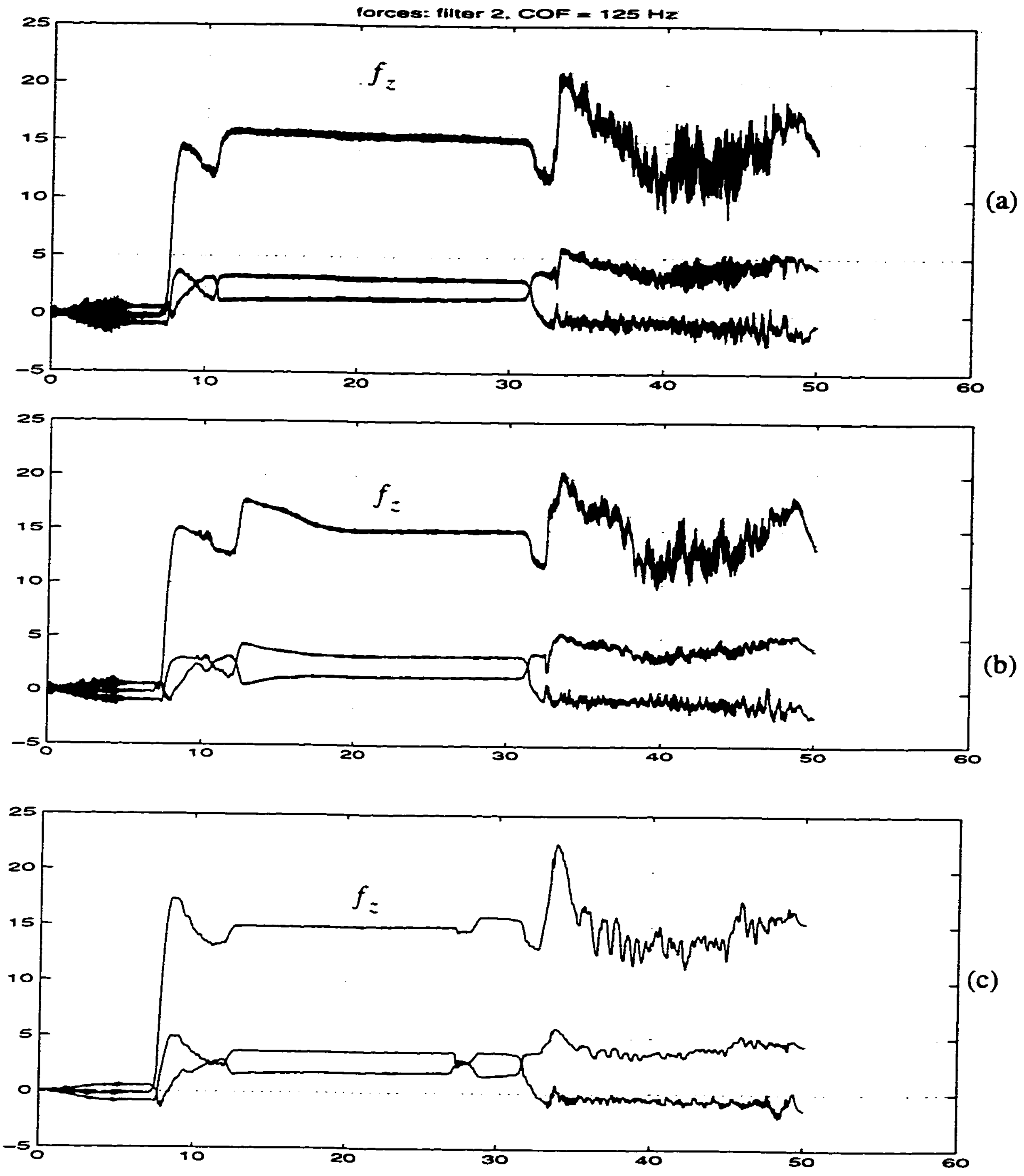


Figure 6.8 Effect of the force filter cutoff frequency; a) 125 Hz, b) 31.25 Hz, and c) 7.81 Hz

The optimal filter has been selected experimentally. Figure 6.8 shows the force measurements with different filters for the test scenario of Figure 6.4. As one may notice, the filter with 7.81 Hz cutoff frequency gives the best tracking ($f_z^d = 15N$) with maximum noise reduction.

6.3.4 Effect of Kinematic Errors (Robustness Issue)

The AHIC scheme may suffer from two major sources of kinematic errors:

- **The kinematic parameters of the arm**: In the absence of a very accurate kinematic calibration, the forward kinematics based on kinematic parameters can introduce errors in the calculated Cartesian feedback.
- **The robot's environment**: The kinematic description of the robot's environment such as position and orientation of the constraint frame introduces kinematic errors when the Cartesian feedback is transformed into the constraint frame.

Different solutions may be envisaged

1– Kinematic calibration of the arm.

2– Kinematic calibration of the environment.

3– Use of a real-time vision system.

4- Mechanical design of the tool attachment.

5– Exploitation of the capabilities of the AHIC scheme.

The Cartesian feedback (linear and angular position and rates) are calculated based on the joint angles and the forward kinematics of the manipulator. Therefore, accurate kinematic calibration can improve the performance. Kinematic calibration of the environment (robot's base coordinate and the constraint surfaces) will also improve the performance. An alternative to the kinematic calibration is to use a real-time vision system which gives the appropriate feedback expressed in a desired frame. Mechanical design of the tool attachment can also play a important role in performance improvement. For instant, using a universal joint in the surface cleaning demonstration improves the performance by rejecting interaction torques which would be present otherwise due to surface orientation errors (or arm calibration). The AHIC scheme itself can act as a tool to deal with kinematic errors at two levels:

- The impedance controller in the position-controlled directions can gracefully handle any coupling forces (disturbances) due to kinematic errors.
- The force/torque controller uses only force sensor feedback (the linear and angular position and rate feedback does not appear in the force/torque-controlled directions). This force sensor information provides error-free information about the kinematics of the environment and constraint surfaces.

In conducting the strawman tasks, we relied solely on solutions 4 and 5 (mechanical design of the tool attachment and exploiting the AHIC scheme). This emphasized the performance of the controller and its robustness with respect to kinematic errors.

As an example, the design of the peg and holes (cone-shaped peg heads and chamfered type opening at the top of the holes) can accommodate certain position errors due to imprecise kinematic parameters of the arm and its environment.

Before presenting the numerical results of the strawman tasks, let us study the performance of the AHIC scheme in identifying the correct kinematics of the environment using force sensor feedback without relying on knowledge of the kinematics of the arm and its environment.

In the following experiment, a plexi-glass rectangular plate was rigidly attached to the last link such that the plate's normal is parallel to the tool frame's x axis. The task consisted of two segments; in the first segment the manipulator was commanded to go in five seconds to a position above the surface with the tool frame's x axis perpendicular to the surface ($s = diag[1, 1, 1, 1, 1, 1]$); in the second segment, the position along the x and y axes (see Figure 6.4) was kept constant while the desired force along the z axis was specified (20 N). All three rotational axes were specified to be torque-controlled to deal with any misorientation of the surface ($s = diag[1, 1, 0, 0, 0, 0]$). Note that the position and orientation of the task frame (in this case, the surface frame) were provided by the user. Note that in the following sections, the impedance or force controlled directions are specified by $s = diag[p_x, p_y, p_z, r_x, r_y, r_z]$ where a 0 entry indicates a force/torque-controlled direction, and a 1 entry indicates an impedance-controlled direction.

In order to study the robustness of the algorithm, we introduced ~5° (along each rotational axis) of orientation error on the surface (see Figure 6.9). Figure 6.10 shows that there is considerable torque at the initial stage of contact with the surface (this is due to the orientation mismatch). This initial torque reduces very fast because the controller tries to regulate the torques to zero. Hence, the plate detects the correct orientation of the surface. We can also see the performance of the force controller in regulating the normal force to the surface to 20N.

Note that this experiment is similar to that of inserting a peg into a hole when the peg and the hole axes are not completely aligned. Therefore, the desired mass and damping for the three rotational axes, $m^d = 0.056, b^d = 48$, can also be used for the second strawman task (peg in the hole).

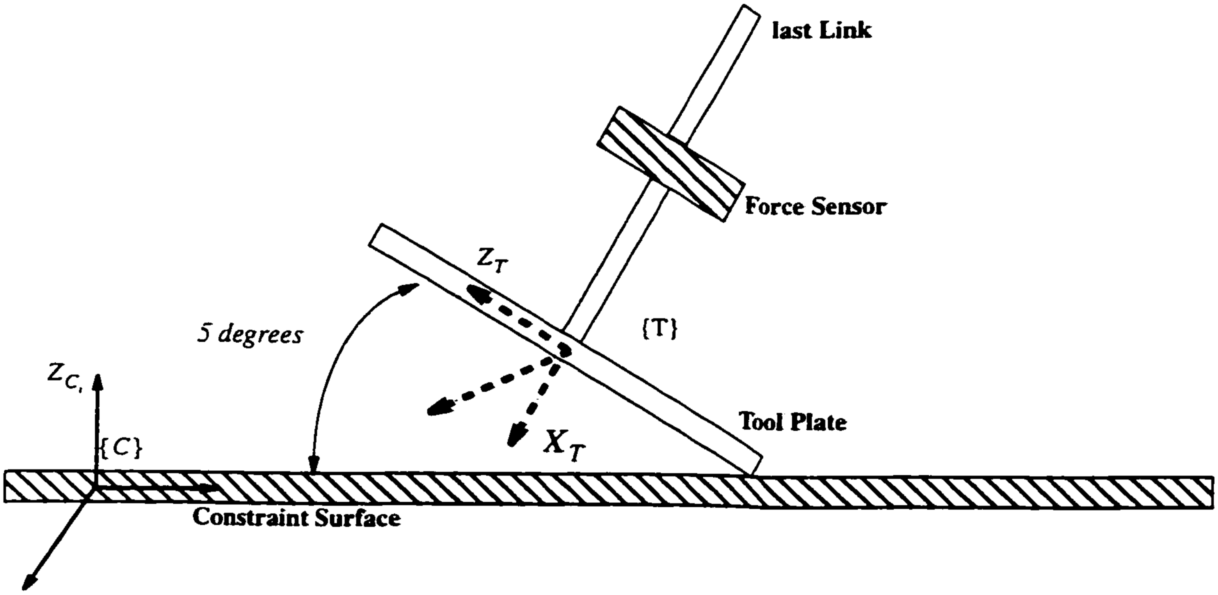


Figure 6.9 Hardware experiment to illustrate the capability of the AHIC scheme in identifying the correct kinematics of the environment using force sensor feedback

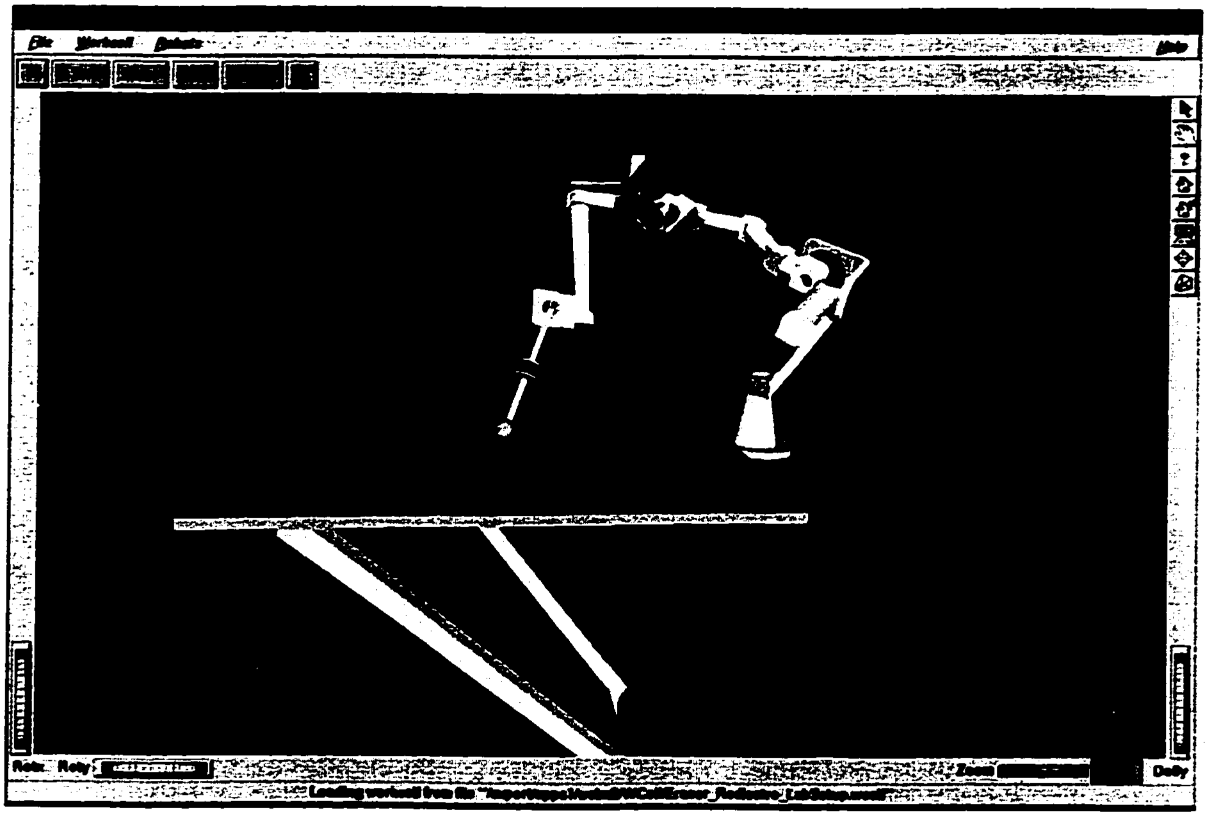
- Filtered force/torque data from the selected filter

- Commanded torques

- A time stamp is added at each sampling time.

This information is uploaded to the SGI workstation at the end of each experiment. The Cartesian feedback is then calculated using an off-line *Simulink* simulation which performs the necessary calculations and coordinate transformations automatically.

The joint angles as well as the force/torque sensor data are transferred in real time to an SGI workstation running the MRS graphical software for visualization of the motion of the arm in an graphical environment similar to that of the hardware experiments (see Figure 6.11). The communication is handled using the socket connection (TCP/IP protocol) between the SUN workstation acting as the host for the 68030 processors and the SGI workstation running MRS.



Loading workcell from file "[illegible]"

Figure 6.11 Graphical rendering of the surface-cleaning task using MRS

6.4.2 Strawman Task I (Surface Cleaning)

Figure 6.12 shows a perspective view of the setup used for the hardware demonstration of the strawman task. The five segments of the task are shown in Figure 6.13.

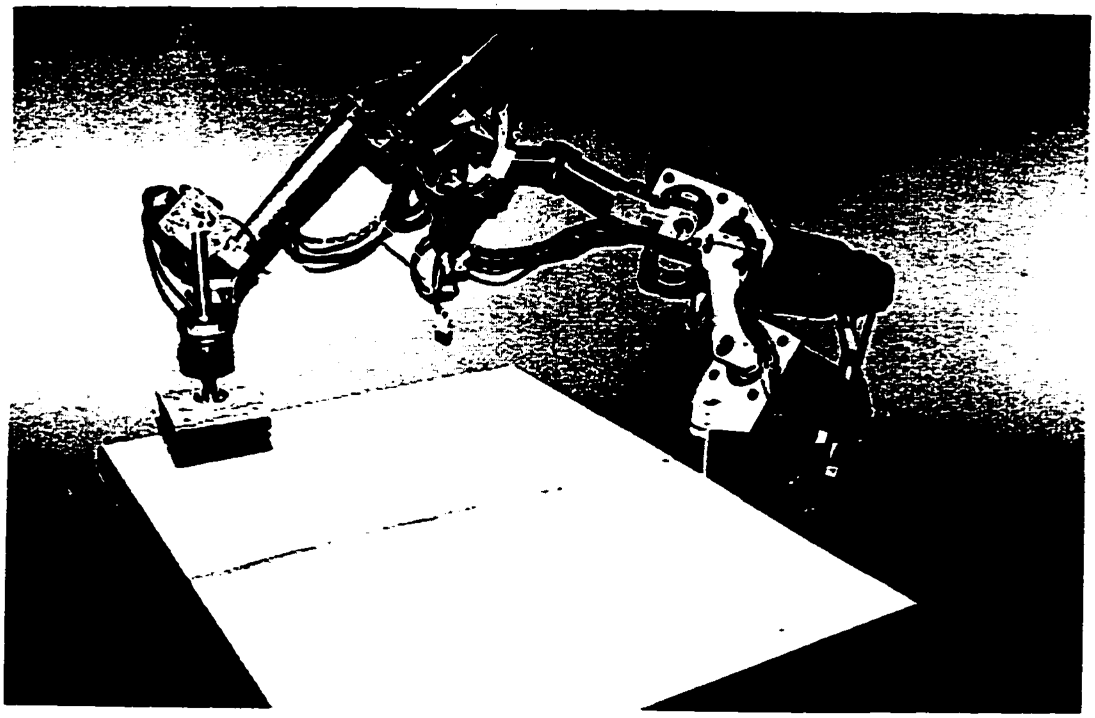


Figure 6.12 Perspective view of the hardware setup used in the demonstration of Strawman Task I (surface cleaning)

Table 6-4 Control parameter used for Strawman Task I

Seg	T (s)	Selection vector	Desired Inertia	Desired damping	Desired stiffness	Desired Force/ torques
1	5	1 1 1 1 1 1	$M_p\, M_p\, M_p\, M_p\, M_p\, M_p$	$B_p\, B_p\, B_p\, B_p\, B_p\, B_p$	$K_p\, K_p\, K_p\, K_p\, K_p\, K_p$	y[a] y y y y y
2	25	1 1 0 1 1 1	$M_p\, M_p\, M_f\, M_p\, M_p\, M_p$	$B_p\, B_p\, B_f\, B_p\, B_p\, B_p$	$K_p\, K_p$ y $K_p\, K_p\, K_p$	y y -20 y y y
3	50	1 1 0 1 1 1	$M_p\, M_p\, M_f\, M_p\, M_p\, M_p$	$B_p\, B_p\, B_f\, B_p\, B_p\, B_p$	$K_p\, K_p$ y $K_p\, K_p\, K_p$	y y -20 y y y
4	75	1 1 0 1 1 1	$M_p\, M_p\, M_f\, M_p\, M_p\, M_p$	$B_p\, B_p\, B_f\, B_p\, B_p\, B_p$	$K_p\, K_p$ y $K_p\, K_p\, K_p$	y y -20 y y y
5	100	1 1 0 1 1 1	$M_p\, M_p\, M_f\, M_p\, M_p\, M_p$	$B_p\, B_p\, B_f\, B_p\, B_p\, B_p$	$K_p\, K_p$ y $K_p\, K_p\, K_p$	y y -20 y y y
-1						

[a] "don't care"

Table 6-4 summarizes the control parameters used for this task, where y denotes information that is not needed. The desired masses for the position and the force-controlled directions are $M_p = 257$ and $M_f = 5.7$ respectively. The values of the desired damping in the position and the force-controlled directions are $B_p = 1100$ and $B_f = 477$; and the desired stiffness in the position-controlled direction is $K_p = 1100$. The PD gains are selected as $k_p = 400, k_v = 40$. Also note that in this experiment, no joint-friction compensation in the inverse dynamics module is performed.

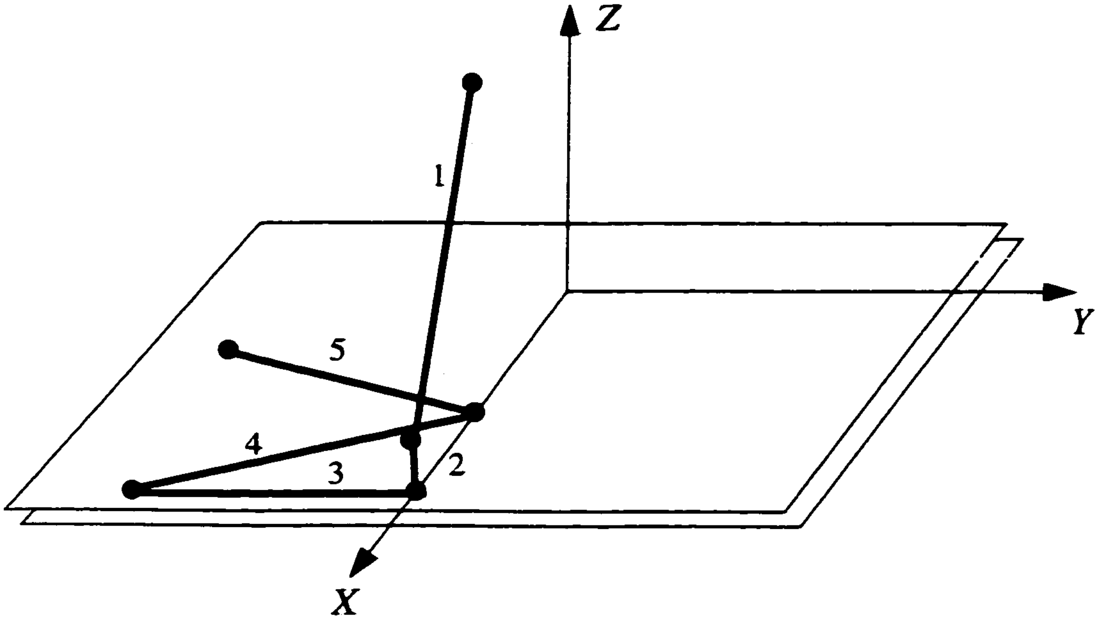
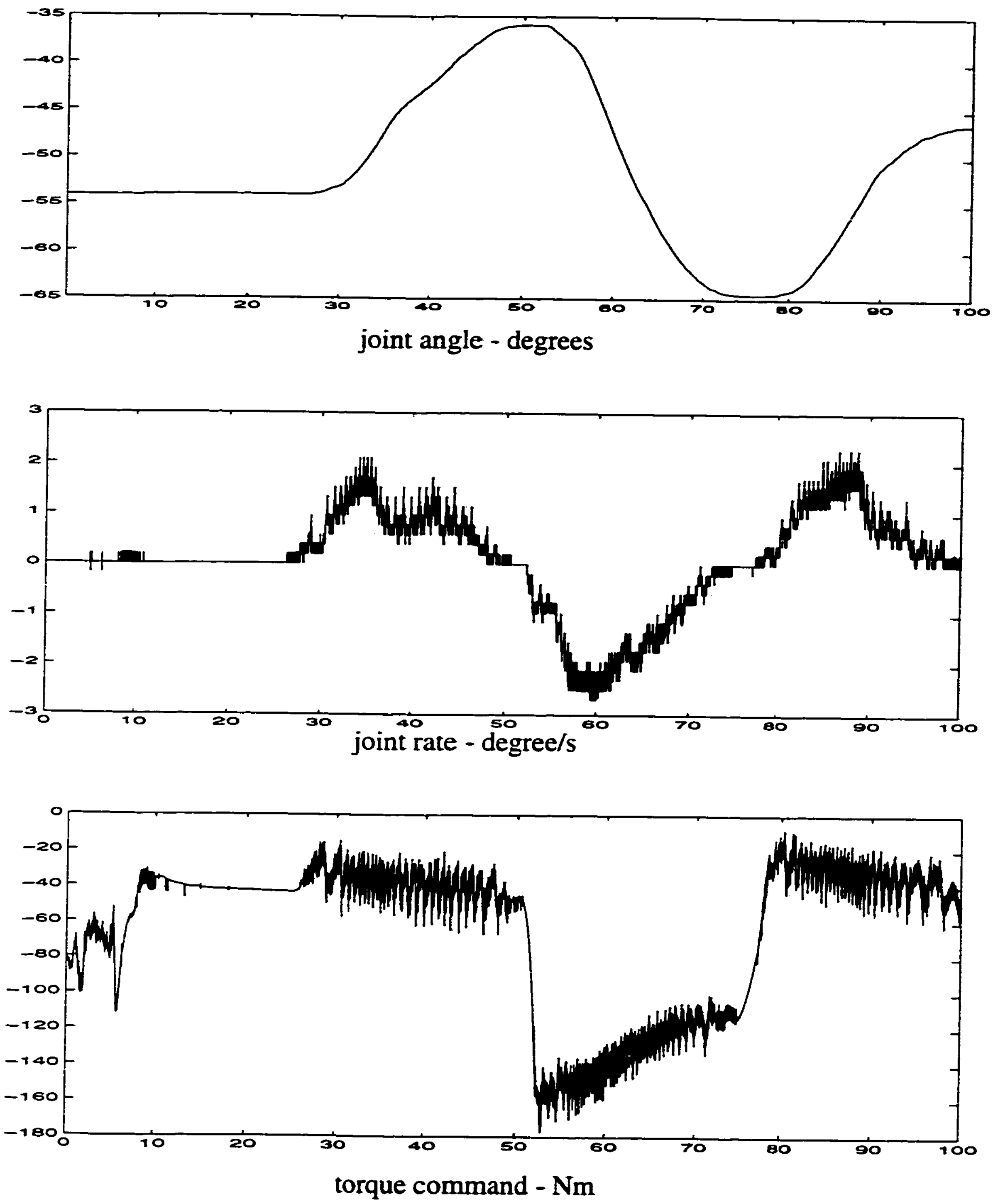


Figure 6.13 Different segments of Strawman Task I (surface cleaning)

As an example, the joint angle, rate and commanded torque for joint 2 (gathered during run-time) are shown in Figure 6.14. Note that the presence of the noise on the estimate of joint rates (due to numerical differentiation) has not affected the force and position tracking. This noise is effectively filtered out by the dynamic of the actuators and current amplifiers.



joint angle - degrees

joint rate - degree/s

torque command - Nm

Figure 6.14 Strawman Task I: Captured data for joint 2

The results of the interaction forces are given in Figure 6.15. As we can see, the force tracking in the 5 to 25 seconds segment when there is no motion in the x and y directions is almost perfect (0.04 N steady-state error).

It was noted in Section 6.3.1.3 that when the pad moves on the surface, the force tracking can degrade drastically because of unmodeled flexibility in the joints. However, by appropriate selection of the controller's cutoff frequency in a force-controlled direction (see Figure 6.8), one can achieve an acceptable level of force tracking. For the segments beyond 25 seconds, there is a low amplitude (approximately 1 N) oscillation with a frequency around 1 Hz due to unmodeled joint flexibility. However, the mean value and standard deviation ($f_{mean} = -19.60N, f_{std} = 0.6$) for the time interval between 15 to 80 seconds show the capability of the force-controller in regulation of the interaction forces even in the presence of unmodeled dynamics (joint frictions and flexibilities).

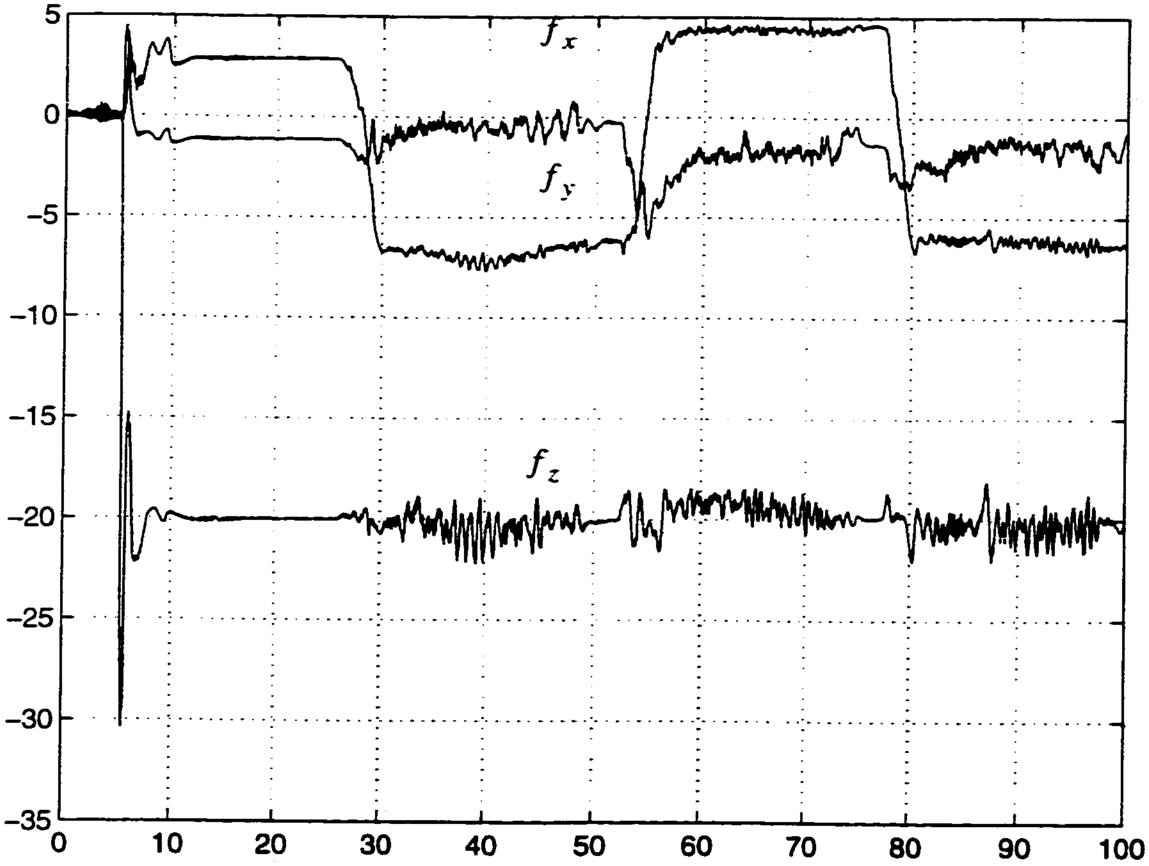


Figure 6.15 Force data captured for Strawman Task I

There is also considerable friction on the surface: approximately 5N in the y direction and 2N in the x direction. However, the impedance controller is not only stable in these directions, but is also successful in achieving acceptable tracking with 1 cm steady-state error in the y direction and 0.5 cm in the x direction (see Figure 6.16). These errors can be further reduced by assigning a larger k^d in the impedance-controlled directions

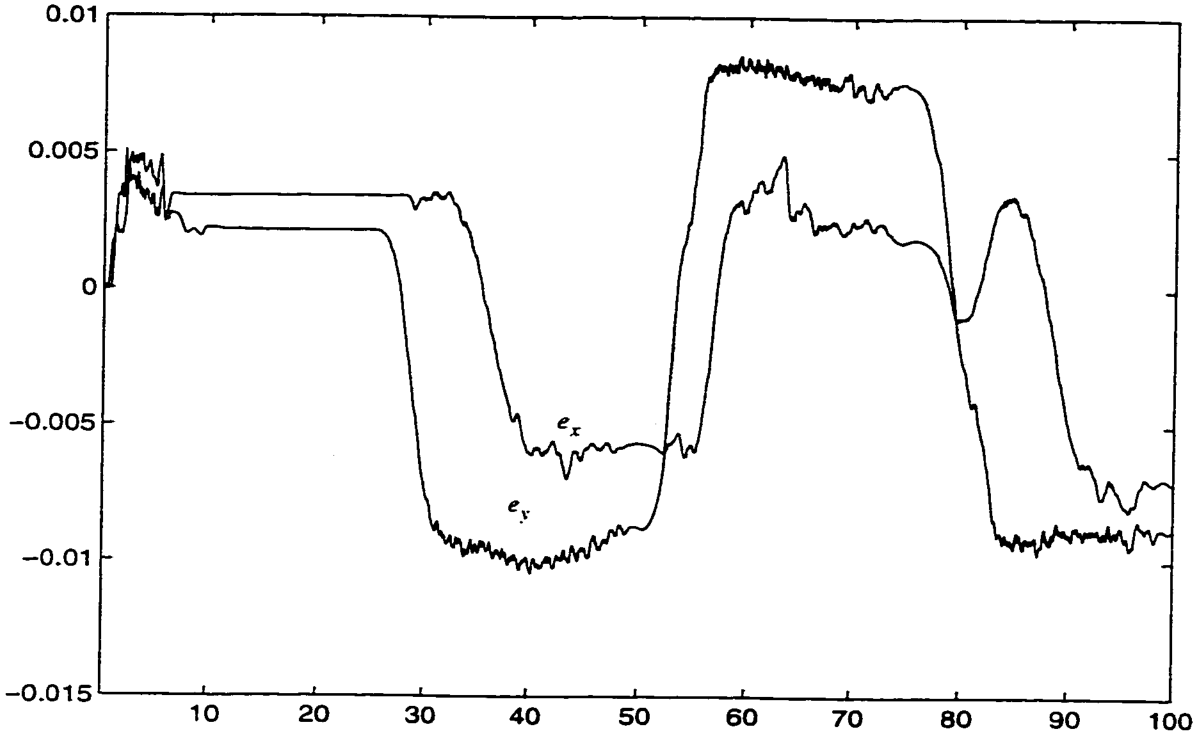
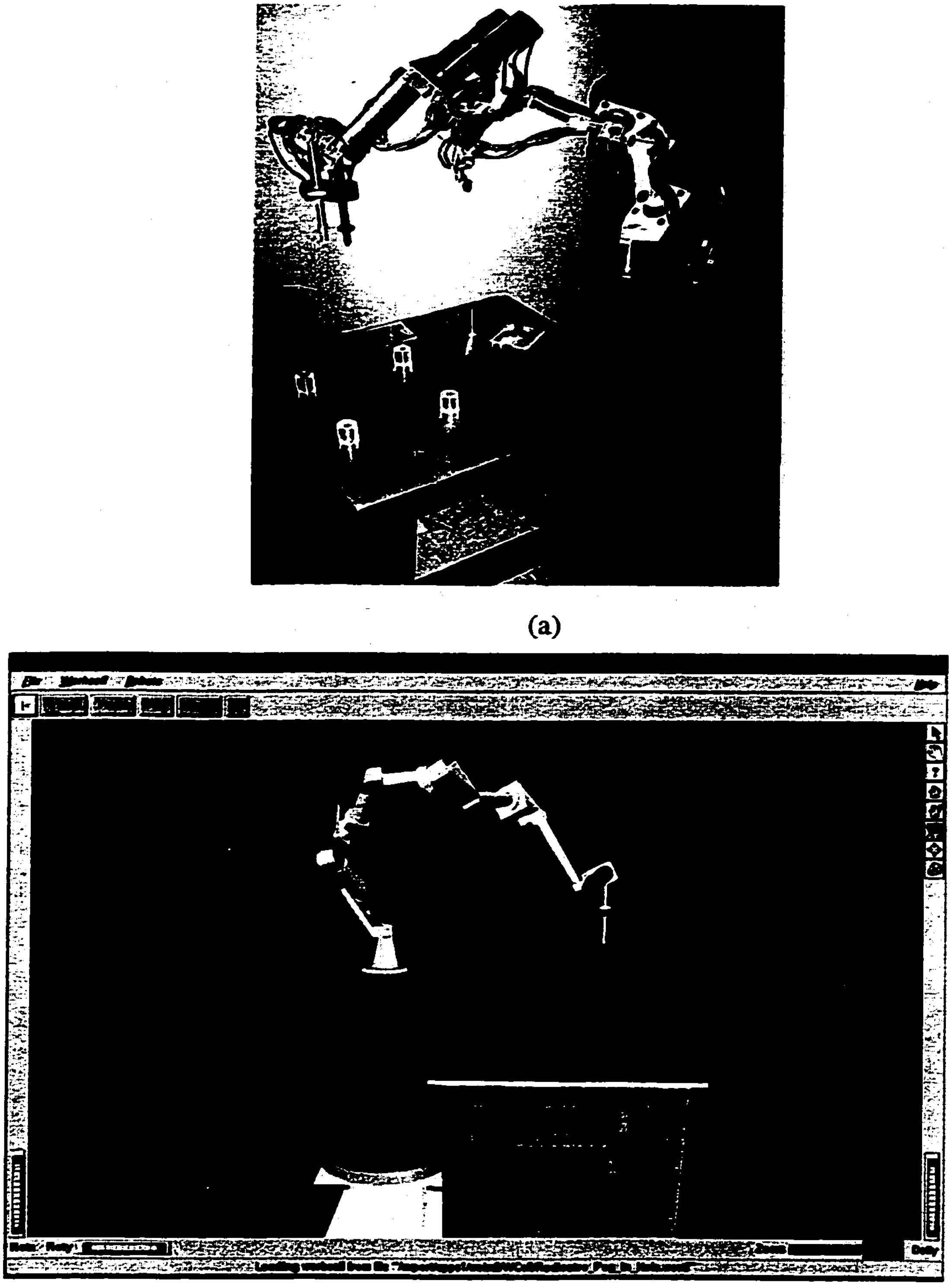


Figure 6.16 Position errors for the surface-cleaning hardware demonstration

6.4.3 Strawman Task II (Peg In The Hole)

Figure 6.17 shows a perspective view of the setup used for the hardware demonstration of the task.



(a)

(b)

Figure 6.17 Perspective view of the hardware setup used in demonstration of Strawman Task II; a) Real arm, b) MRS simulation

The complete task consists of accomplishing the insertion of a peg into, and its removal from, two different holes. In Section 6.3.4, we described the effects of kinematic errors. It was noted that kinematic errors can play a vital role in performing tasks such as the peg-in-the-hole operation. In most cases kinematic errors result in conflicts between the position and force-controlled directions which can cause oscillations and instability. Different solutions were suggested. However, in performing this strawman task, no calibration of the arm kinematics and the hole setup was performed. Instead, we relied on the kinematic design of the peg and the holes, and also on the capability of the AHIC scheme to deal with kinematic errors. The kinematic dimensions of the peg and the holes are given in Figure 6.19a. The final tolerance between the peg and the holes is 0.25 mm. However, the structural designs of the head of the peg and the top of the holes facilitate the correct initiation of the insertion process in the presence of small positioning errors. With this design, only an accuracy/repeatability of 15mm for hole 1 and 11.5 mm for hole 2 is required to initialize the insertion (see Figure 6.19b). From this point onward, the AHIC scheme is responsible for accomplishing the insertion in the presence of kinematic errors.

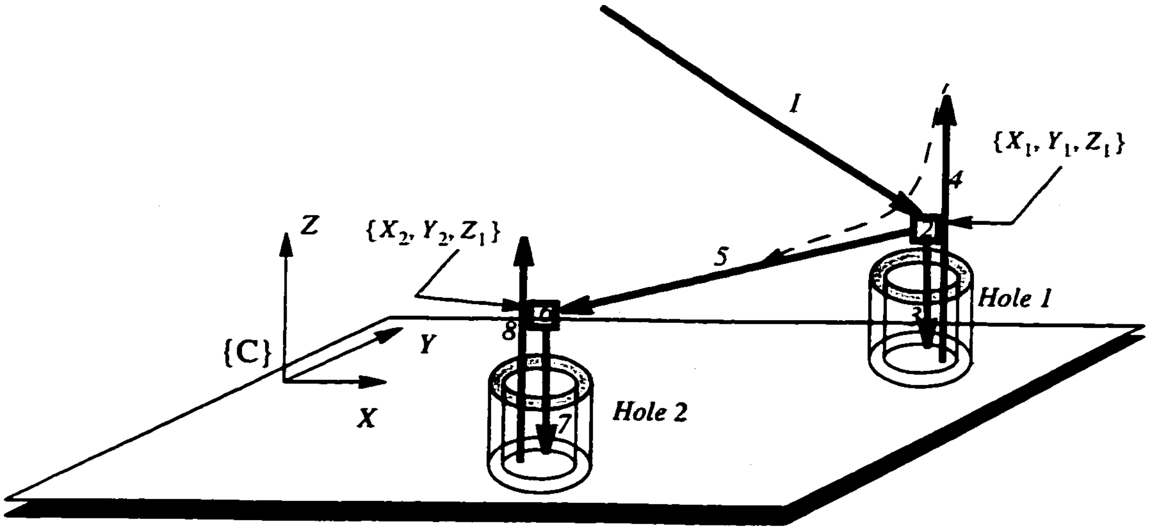


Figure 6.18 Different segments of Strawman Task II

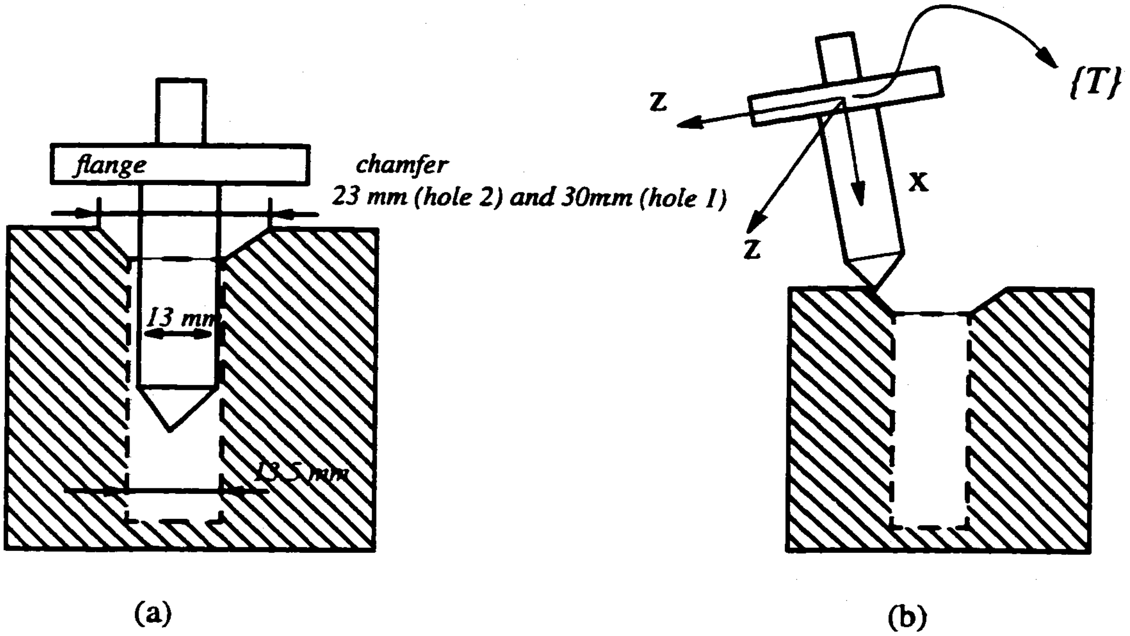


Figure 6.19 Strawman Task II: a) Kinematic tolerances, b) Correct initiation of the insertion

The strawman task consists of 8 segments. Table 6-5 summarizes the desired position, orientation, and force for the task. The control parameters are given in Table 6-6.

Table 6-5 Desired positions, orientations, and forces for Strawman Task II

Seg	T (s)	Selection vector	Desired Position (m)			Desired Orientation			Desired Force(N)			Des. Torque(Nm)		
			X	Y	Z	Rx	Ry	Rz	Fx	Fy	Fz	Tx	Ty	Tz
1	3	111111	$x1^{a}$	*y1*	*z1*	*pi/2*	*pi/2*	*0*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*
2	10	001111	NA^{b}	*NA*	*z1*	*pi/2*	*pi/2*	*0*	*0*	*0*	*NA*	*NA*	*NA*	*NA*
3	65	000000	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*0*	*0*	*-8*	*0*	*0*	*0*
4	110	000000	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*0*	*0*	*10*	*0*	*0*	*0*
5	115	111111	*x2*	*y2*	*z1*	*pi/2*	*pi/2*	*0*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*
6	125	001111	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*0*	*0*	*NA*	*NA*	*NA*	*NA*
7	190	000000	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*0*	*0*	*-8*	*0*	*0*	*0*
8	225	000000	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*0*	*0*	*10*	*0*	*0*	*0*
-1														

a. See Figure 6.18

b. NA =Not Applicable

Table 6-6 Control parameters used in Strawman Task II

Seg	T (s)	Selection vector	Desired Inertia	Desired damping	Desired stiffness	Desired Force/torques
1	3	111111	m_p^{a} m_p m_p m_p m_p m_p	b_p b_p b_p b_p b_p b_p	k_p k_p k_p k_p k_p k_p	y^{b} y y y y y
2	10	001111	m_{f1} m_{f1} m_p m_p m_p m_p	b_{f1} b_{f1} b_p b_p b_p b_p	y y k_p k_p k_p k_p	0 0 y y y y
3	65	000000	m_{f1} m_{f1} m_{f2} m_{f3} m_{f3} m_{f3}	b_{f1} b_{f1} b_{f2} b_{f3} b_{f3} b_{f3}	y y y y y y	0 0 -8 0 0 0
4	110	000000	m_{f1} m_{f1} m_{f2} m_{f3} m_{f3} m_{f3}	b_{f1} b_{f1} b_{f2} b_{f3} b_{f3} b_{f3}	y y y y y y	0 0 10 0 0 0 0
5	115	111111	m_{p1} m_{p1} m_{p1} m_{p1} m_{p1} m_{p1}	b_{p1} b_{p1} b_{p1} b_{p1} b_{p1} b_{p1}	k_p k_p k_p k_p k_p k_p	y y y y y y
6	125	001111	m_{f1} m_{f1} m_p m_p m_p m_p	b_{f1} b_{f1} b_p b_p b_p b_p	y y k_p k_p k_p k_p	0 0 y y y y
7	190	000000	m_{f1} m_{f1} m_{f2} m_{f3} m_{f3} m_{f3}	b_{f1} b_{f1} b_{f2} b_{f2} b_{f3} b_{f3}	y y y y y y	0 0 -8 0 0 0
8	225	000000	m_{f1} m_{f1} m_{f2} m_{f3} m_{f3} m_{f3}	b_{f1} b_{f1} b_{f2} b_{f3} b_{f3} b_{f3}	y y y y y y	0 0 10 0 0 0 0
-1						

a. see Table 6-7 for numerical values

b. y = not needed

Table 6-7 Numerical values of the desired impedances for Strawman Task II

Position-controlled axis					Force-controlled axis					
desired mass (kg)		desired damping Nsec/m		desired stiffness N/m	desired mass (kg)			desired stiffness N/m		
m_p	m_{p1}	b_p	b_{p1}	k_p	m_{f1}	m_{f2}	m_{f3}	b_{f1}	b_{f2}	b_{f3}
257	112	1100	700	1100	22.8	253	0.056	955	3180	48

Descriptions of the different segments are as follows:

Segment 1: In this segment the tool frame {T} is commanded to go to position $\{x_1, y_1, z_1\}$, where x_1 and y_1 are the x and y coordinate of the center of hole 1 as seen in {C}; z_1 is specified to ensure that the peg's head is above the hole's upper surface. The desired orientation is specified such that the x axis of {T} is aligned with the axis of hole 1 (see Figure 6.13). Note that because of kinematic errors, the position and orientation of the tool frame are different from their desired values.

Segment 2: It was noted that due to the presence of different sources of kinematic errors, the position and orientation of the tool frame would be different from the desired values at the end of segment 1. In segment 2, the possibility of manually correcting the tool frame position in the x and y directions is given to the operator. This is only required if the kinematic errors are such that the insertion cannot be initiated correctly (see Figure 6.19b). In this case, the operator can drag the peg to a position from where the insertion can be initialized. This is done by keeping the orientation and the tool frame's height (along the z axis) constant while the x and y axes are under force control with zero desired force. In the hardware experiment for Strawman Task II, no manual correction was needed.

Segment 3 (insertion): In this segment all axes are under force/torque control. In this way the force/torque sensor information is used to accurately align the axes of the peg and the hole. Only a negative desired force along the z axis is specified (desired force/torque for the other axes are zero). Note that no logic branching is required to detect the end of the insertion. The motion is stopped upon completion of the insertion (i.e., on achieving the desired interaction force between the peg's flange and the top surface of the hole).

Segment 4 (removal): This segment is similar to segment 3, with the difference that the desired force is in the positive z direction to accomplish the removal.

Segment 5: This is the transmission segment to locate the peg on top of hole 2. Note that in segment 4, the z-axis was under force control attempting to achieve a positive force. Because there is no constraint on the tool frame that allows the desired force to be achieved, the tool frame continues to move along the positive z direction with a bounded terminal velocity according to a time controlled schedule. By starting segment 5, all axes come under position control so as to position the peg on top of the second hole. As noted in Section 5.2.1 , the task planner module uses a pre-specified task file to calculate the coefficients of the desired trajectory for different segments before starting the task. Therefore, the initial position of the tool frame (the final position at the end of segment 4) is not known ahead of time. In this experiment, we used the desired final position for segment 1 as the initial position of segment 5. Therefore, there is an initial position error when segment 5 starts. However, as mentioned in Section 6.3.1.2, this does not cause any difficulties since the impedance controller will smoothly "attract" the tool frame to the desired trajectory (see dashed-line in Figure 6.18).

Segment 6: Similar to segment 3

Segment 7: Similar to segment 4.

Figure 6.20 and Figure 6.21 show the results of the hardware experiment for Strawman Task II. In order to get a better resolution, only the insertion and removal procedures for hole 1 are shown. The following phases can be observed in Figure 6.20:

Phase 1 (position correction): When the head of the peg touches the chamfer at the top of the hole, the interaction forces in the x and y direction modify the position mismatch (due to kinematic errors) and guide the head of the peg into the hole. This happens because x and y coordinates of the $\{T\}$ frame are force-controlled. As one can see, the interaction forces between the head of the peg and the body of the chamfer are reduced as the center of the peg enters the hole. The plot of the y coordinate for this phase shows the position modification (approximately 5mm).

Phase 2 (orientation correction): As the peg is inserted further into the hole, there is considerable force/torque build up because of the misalignment of the peg and the hole. This reduces rapidly as the torque controller for all three rotational axes reacts to modify the alignment of the peg. The interaction forces and torques reduce when correct alignment is achieved.

Phase 3 (completing the insertion). After the peg's flange touches the top surface of the hole, the force controller tries to regulate the force in the z direction to the desired value (-8N). At this point ($t = \sim 50$ s) there are minimum interaction torques (around all three rotational axes) and forces in the x and y directions. This shows correct positioning and alignment of the peg. Note that at this stage no logic (mode) branching is required. The peg remains inserted until the removal phase starts.

Phase 4 (removal): In this phase a positive desired force is specified which forces the removal process to start.

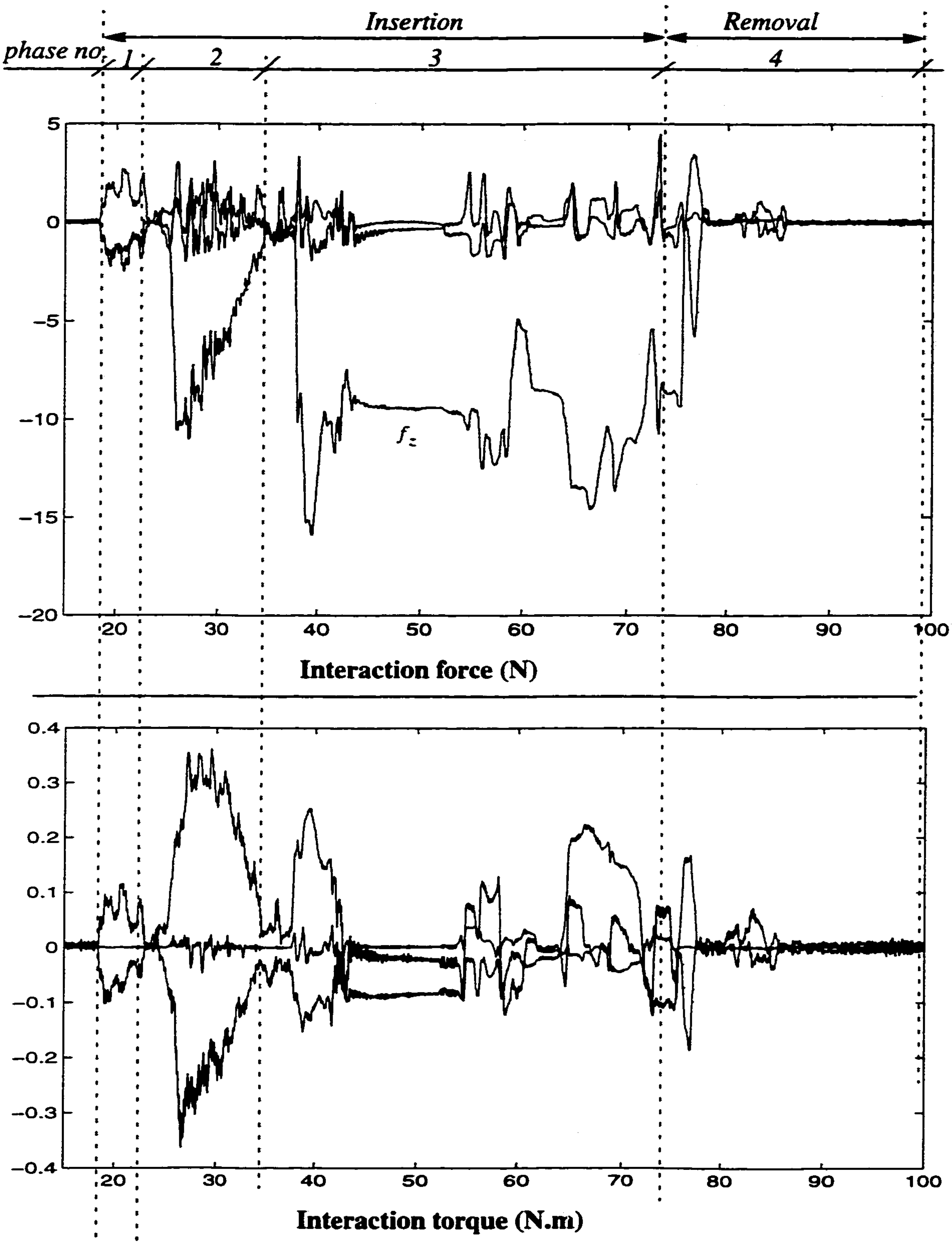
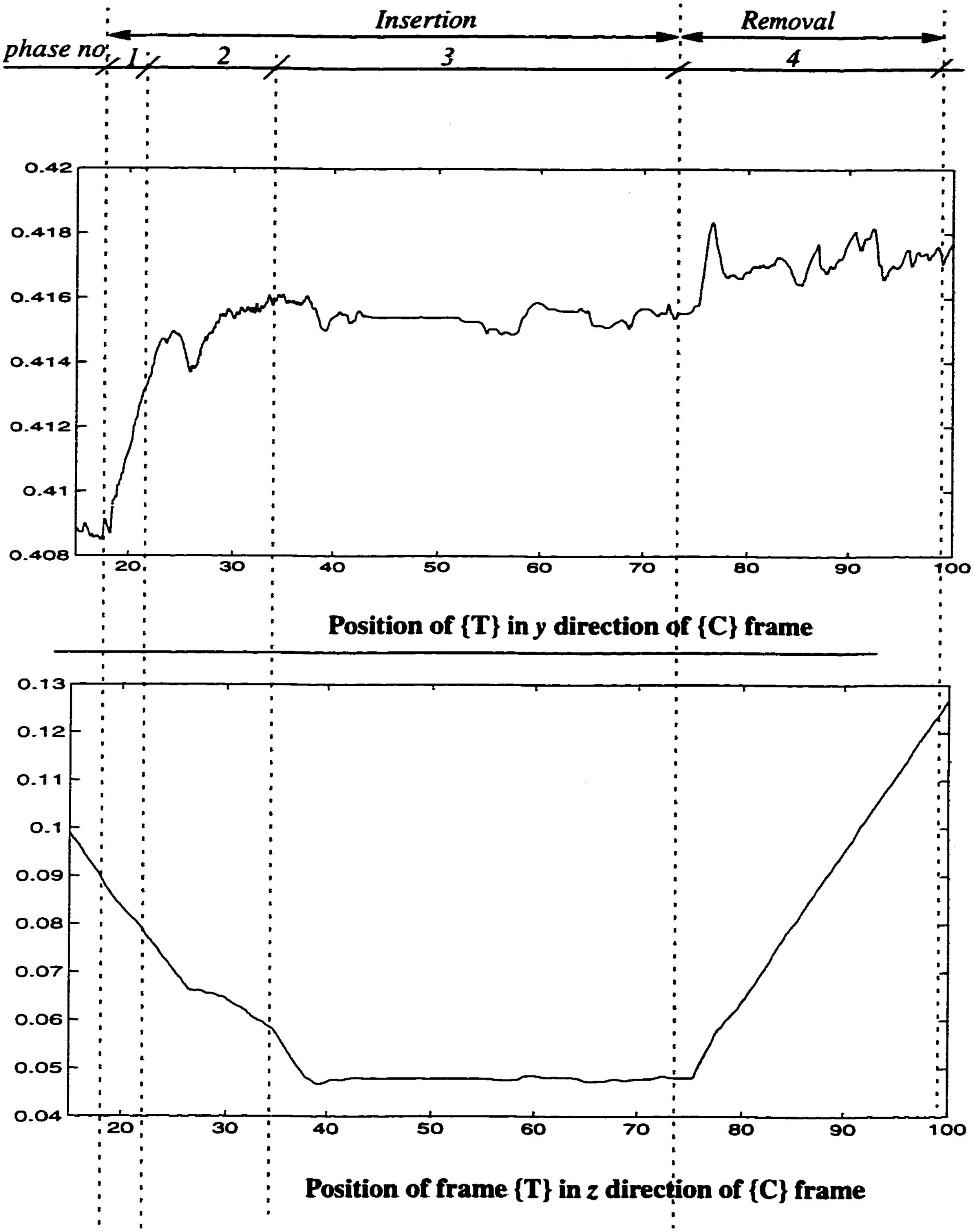


Figure 6.20 Strawman Task II: Hole 1 insertion/removal; interaction force/torques



Position of {T} in *y* direction of {C} frame

Position of frame {T} in *z* direction of {C} frame

Figure 6.21 Strawman Task II: Hole 1 insertion/removal; position information

In order to test the peg-in-the hole operation in the case of a tight fitting scenario, a layer of aluminum foil was wrapped around the peg which prevented the peg from sliding freely (under its own weight) into the hole. Strawman Task II was successfully demonstrated for this case as well. The only parameter that needed to be modified was the desired force in the z direction which was increased from 8N to 15N. This was necessary to prevent the peg from jamming.

In order to test the robustness of the scheme with respect to kinematic description of the environment, the above scenario was repeated while introducing $\sim 5°$ orientation error on the axes of the holes. Strawman Task II was successfully demonstrated for this case as well.

6.5 CONCLUSION

The goal of this chapter was to demonstrate the feasibility and to evaluate the performance of the proposed compliant motion and force control scheme via hardware demonstrations using REDIESTRO. Two strawman tasks - surface cleaning and peg-in-the-hole - were selected.

The results for the surface cleaning strawman task indicate that when there is no motion in the x and y directions, the force tracking is almost perfect (0.04 N steady-state error). When the eraser is moving on the surface, it was observed that because of unmodeled flexibility in the joints, the force tracking may degrade drastically. However, by appropriate selection of the controller's cutoff frequency in a force-controlled direction, we achieved an acceptable level of force tracking. The experiment shows that with 20N desired force, an interaction force with mean value -19.6 N and standard deviation 0.6 is achieved. This demonstrates the capability of the force-controller in regulation of the interaction forces even in the presence of unmodeled dynamics (joint frictions and flexibilities).

Strawman Task 2 was also successfully demonstrated. Considering the tolerances used for the design of the peg and the holes (0.25 mm between radius of the peg and the hole) and the fact that REDIESTRO has not been kinematically calibrated, the successful peg-in-the-hole demonstration presents the robustness of the proposed scheme with respect to poor knowledge of the environment.

CHAPTER 7 CONCLUSION AND FUTURE WORK

As indicated in the introduction, the objectives of this thesis were to *"propose a unified frame work for combining compliant motion control, redundancy resolution, and adaptive control in a single methodology"* and to demonstrate the *"the feasibility of the proposed scheme by computer simulations and experiments on REDIESTRO"*. Regarding these objectives the following concluding remarks can be drawn:

The basic issues needed for analysis of kinematically redundant manipulators were presented in Chapter 2. Different redundancy resolution schemes were reviewed. Based on this review, *configuration control* at the acceleration level was found to be the most suitable approach for a force and compliant motion control of redundant manipulators. A formulation of the additional tasks to be used by the redundancy resolution module was presented. Joint limit avoidance, one the most useful additional tasks to avoid mechanical joint limits, and self-collision avoidance, were studied in more detail. The basic formulation of static and moving obstacle collision avoidance tasks in 2D workspace was presented.

The extension of the redundancy resolution and obstacle avoidance module to the 3D workspace of REDIESTRO was addressed in Chapter 3. The obstacle avoidance algorithm was modified to consider 3-D objects. A novel primitive-based collision avoidance scheme was presented. This scheme is general, and provides realism, efficiency of computation, and economy in preserving the amount of free space that would otherwise be wasted. Possible cases of collisions were also considered. In particular, the cylinder-cylinder collision avoidance which represents a complex case for a collision detection scheme was formalized using the notions of dual vectors.

Before performing hardware experiments using REDIESTRO to evaluate the performance of the redundancy resolution and obstacle avoidance module, extensive simulations were performed using the kinematic model of REDIESTRO. The simulation results indicated that the least-squares approach for the redundancy resolution is important for practical applications in order to cop with the kinematic and artificial singularities. The latter may arise because of conflicts between the main and additional tasks. However, this introduction of singularity robustness results in tracking errors in regions away from singularity. It has been shown that by a proper selection (or time-varying formulation) of W_v, the weighting matrix in the singularity robustness task, the effect of this term on the tracking performance can be minimized. It was also shown that the formulation of the additional task as an inequality constraint, may result in considerable discontinuity in joint velocities which causes a large pulse in joint accelerations. In a practical implementation, the maximum acceleration of each joint would be limited, and this commanded joint acceleration would result in saturation of the actuators. A time-varying formulation of the weighting matrix, W_c, was proposed which successfully overcame this problem.

Three scenarios encompassing most of the developed redundancy resolution and obstacle avoidance system features were successfully demonstrated on real hardware, i.e. the REDIESTRO manipulator. These scenarios verified the performance of the redundancy resolution and obstacle avoidance scheme in executing the following tasks: position tracking, orientation tracking, static and moving obstacle collision avoidance, joint-limiting, and self-collision avoidance. In each of these scenarios one or multiple features were active at different instants of execution.

Despite the geometrical complexity of REDIESTRO, the arm is entirely modelled by decomposition of the links and the attached actuators into sublinks modeled by simple volume primitives. Moreover, due to the complex and unusual shape of REDIESTRO, it is believed that adapting the algorithms to other industrial and research manipulators can only be simpler.

In Chapter 4, we undertook a literature survey on the existing methodologies for force and compliant motion control. The comparison between different methodologies indicated that the hybrid impedance control approach is at present the most suitable scheme for compliant motion and force control. The outcome of this survey also showed that there exists no unique framework for compliant motion and force control of redundant manipulators which enjoys the following advantages:

1- Takes full advantage of redundant degrees of freedom

- Incorporate different additional tasks can be easily incorporated without modifying the scheme and the control law.
- Use redundant degrees-of-freedom to fulfill dynamic tasks such as multiple-point force control.
- Use task priority and singularity robustness formulation to cop with kinematic and artificial singularities.

2- Compatibility for execution of both force and compliant motion tasks

- Accurate force regulation
- Stable motion control in the presence of disturbance forces

3- Robustness

- with respect to higher-order unmodeled dynamics (i.e., joint flexibilities), uncertainties in manipulator dynamic parameters, and friction model and parameters.
- with respect to poor knowledge of the environment and kinematic errors

4- Adaptive implementation

- The algorithm structure should allow for easy incorporation of adaptation in the case of manipulators whose estimate of the dynamic parameters are not known.

An Augmented Hybrid Impedance (AHIC) control scheme was proposed which enjoys the aforementioned desirable characteristics. The feasibility of the scheme was evaluated by computer simulation on a 3-DOF planar arm. Most useful additional tasks: joint limit avoidance, static and moving object avoidance, self collision avoidance, and posture optimization, were incorporated into the AHIC scheme. The simulation performed for multiple point contact force control indicated one of the major characteristic of the AHIC scheme which distinguishes it from similar schemes. The additional task can not only be position-controlled, but, it can also be included into the force-control subspace. This increases the capability of the redundancy resolution module.

The simulations on the 3-DOF planar arm showed that a simple extension of the redundancy resolution module at the acceleration level using the solutions which a minimize the norm of the joint acceleration vector have the shortcoming that they cannot control the null-space components of joint velocities and may result in "internal instability". A modified AHIC was proposed that addresses this undesirable self motion problem.

An Adaptive Augmented Hybrid Impedance Control (AAHIC) scheme was presented which guarantees asymptotic convergence in both position and force controlled subspaces when precise force measurements are available. The control scheme ensures stability of the system with bounded force measurement errors. Even in the case of imprecise force measurements, the errors in the position controlled subspaces can be reduced considerably provided that powerful enough actuators are available.

The extension of the AHIC scheme to the 3D workspace of a 7-DOF manipulator (REDIESTRO) was described in Chapter 5. The complexity of the required algorithms and the restrictions with regard to the available computational power required an algorithm development procedure which incorporates a high level of optimization. At the same time, the following problems which were not studied in a 2-D workspace needed to be tackled in extending the modules to a 3-D workspace:

- An AHIC module for orientation and torque

- Uncontrolled self-motion due to resolving redundancy at the acceleration level for the AHIC scheme (the solution proposed in Chapter 4 was computationally expensive).
- Robustness issue with respect to higher-order unmodeled dynamics (joint flexibilities), uncertainties in manipulator dynamic parameters, and friction model and parameters.

A realistic dynamic simulation environment enabled us to study issues such as performance degradation due to imprecise dynamic modeling and uncontrolled self-motion. The least-square solutions for redundancy resolution at the acceleration level was modified by adding a velocity dependent term to the cost function. This modification successfully controlled the self-motion of the manipulator.

It was demonstrated by simulation that the force tracking performance of the methods based solely on inverse dynamics degrade in the presence of uncertainty in the manipulator's dynamic parameters and unmodeled dynamics. This is especially true for a manipulator equipped with harmonic drive transmissions, which introduce a high level of joint flexibility and frictional effects (as in the case with REDIESTRO). The AHIC control scheme was modified by incorporating an "error reference controller". This modification successfully cope with model uncertainties in the model-based part of the controller, and even friction compensation is not required. The modified AHIC scheme can be considered a major contribution of this thesis and will increase the applicability of this type of control to a large class of industrial and research arms.

Chapter 6 described the hardware experiments performed to evaluate the performance of the proposed AHIC scheme in compliant motion and force control of REDIESTRO. Considering the complexity and the large amount of calculation involved in force and compliant motion control of a 7-DOF redundant manipulator, the implementation of the real-time controller, from both hardware and software points of view, by itself represents a challenge. It should be noted that there are few cases to date where the experimental results for a force and compliant motion control of a 7-DOF manipulator have been reported. Moreover, implementation of the AHIC scheme for REDIESTRO introduces additional challenges which are listed below:

- The REDIESTRO arm is equipped with harmonic drive transmissions which introduce a high level of joint flexibility and make accurate control of contact force more difficult.

- The friction model and its parameters cannot be estimated accurately in many practical applications. The friction model that is most commonly used is load independent Coulomb and viscous friction. This model is especially inadequate for a robot with harmonic drive transmissions which have high friction - Experimental results on REDIESTRO show that in some configurations the friction forces reach up to 30% of the applied torque. Also, different experimental studies have shown that frictional forces in harmonic drives are very nonlinear and load dependent which makes accurate parameter identification difficult. This represents a challenge for a model-based controller.

- Performing tasks such as peg-in-the-hole requires very accurate positioning. This requires a very well-calibrated arm. Considering the fact that REDIESTRO has not been kinematically calibrated, successful operation of the peg-in-the-hole strawman task by REDIESTRO demonstrates a high level of robustness of the proposed scheme.

Before performing the final hardware demonstrations, a detailed stability analysis was given to provide guidelines in the selection of the desired impedances. A heuristic approach was presented which enables a user to systematically select the impedance parameters based on stability and tracking requirements.

Two strawman tasks: Surface cleaning and peg-in-the-hole, were selected. The selection was based on the ability to evaluate force and position tracking and also robustness with respect to knowledge of the environment and kinematic errors. These strawman tasks are also representative of the tasks that will be performed by the SPDM in space where the corresponding tasks would be window cleaning and On-Orbit Replaceable Unit (ORU) insertion and removal.

The results for the surface cleaning strawman task indicate that when there is no motion in the x and y directions, the force tracking is almost perfect (0.04 N steady-state error). When the eraser is moving on the surface, it was observed that because of unmodeled flexibility in the joints, the force tracking may degrade drastically. However, by an appropriate selection of the controller's cutoff frequency in a force-controlled direction, we achieved an acceptable level of force tracking. The experiment shows that with 20N desired force, the interaction force with mean value -19.6N and standard deviation 0.6N was achieved. This demonstrates the capability of the force-controller in regulation of the interaction forces even in the presence of unmodeled dynamics (joint frictions and flexibilities).

Strawman task 2 was also successfully demonstrated. Considering the tolerances used in the peg and the holes (0.5 mm between the peg and the hole) and the fact that REDIESTRO has not been calibrated, the successful peg-in-the-hole demonstration indicates the robustness of the proposed scheme with respect to poor knowledge of the environment.

Based on the results presented in this thesis, the main contributions can summarized as follows:

1- A novel primitive-based collision detection scheme.

2- Development and implementation of a real-time collision avoidance system for 7-DOF redundant manipulator.

4- Extension of the *configuration control* approach for redundancy resolution at the acceleration level.

3- Proposing a new approach - *Augmented Hybrid Impedance Control (AHIC)* - for contact force and compliant motion control of redundant manipulators.

4- A new adaptive compliant motion and force control scheme for redundant manipulators.

5- A new robust compliant motion and force control scheme for redundant manipulators which incorporates a new formulation for redundancy resolution and an error reference controller as the inner loop.

6- Implementation and hardware demonstration of the proposed compliant motion and force control scheme on REDIESTRO.

7- The Robot Dynamic Modeling (RDM) software developed to facilitate the research described in this thesis and to provide a novel approach in modelling, simulation, and real-time controller development for general applications in robotics.

Suggestions for Future Work

Contact force and compliant motion control is an active and vast area in robotics research. Considering the objectives of this thesis and also the time available, only a few aspects of contact force and compliant motion control for redundant manipulators were discussed. The work presented here can be enhanced by experimental evaluation of some of the algorithms and techniques proposed in this thesis that were not experimentally. It is also worthwhile investigating some new areas to which the techniques developed in this thesis can be applied. Some suggestions with respect to avenues for further work are as follows:

- The adaptive augmented hybrid impedance controller proposed in Chapter 4 was not evaluated experimentally on REDIESTRO because an accurate enough estimate of the dynamic parameters was already available. However, an extension of the adaptive algorithm to the 3D workspace of a 7-DOF manipulator would increase the applicability of the technique to manipulators with unknown/ unmodelled dynamic parameters.
- Joint flexibility effects were treated as disturbances in the algorithms developed in this thesis. A more detail study to investigate the effects of joint flexibility on force tracking performance should be conducted. This would allow one to design a controller based on a flexible-joint model of the manipulator.

- The obstacle avoidance scheme proposed in Chapter 3 requires exact kinematic knowledge of the environment (provided in a data base or by a real-time vision system). Having developed a contact force control scheme, an alternative is to formulate the obstacle avoidance problem as that of controlling the contact force between one point of the manipulator and its environment. In this way, an obstacle avoidance scheme based on a "sensitive skin" (an array of force sensors) can be developed which relaxes the requirement of the exact kinematic knowledge of the environment.

- The effect of time-varying formulation of the weighting matrices (in redundancy resolution module) on the stability and convergence properties of the closed-loop system can be further studied.

- One interesting area is to study the possibility of extending the algorithms developed in this thesis to cooperative dual-arm control. The fact that the scheme proposed in this thesis can regulate the stable interaction of one arm with its environment, will help to control two arms in cooperative dual-arm operation.

- The REDIESTRO arm used in the hardware experiments is an *isotropic* arm which is designed to provide maximum positioning accuracy and least sensitivity to variation of kinematic parameters of the arm in the neighborhood of isotropic configurations. One interesting area is to investigate the effect of the isotropic design in force control.

REFERENCES

[1] A. A. Maciejewski and C. A. Klein, "The singular value decomposition: Computation and application to robotics," *Int. Journal of Robotics Research*, Vol. 8, No. 6, Dec. 1989.

[2] Y. Nakamura and H. Hanafusa, "Optimal redundancy control of robot manipulators," *Int. Journal of Robotics Research*, Vol. 6, No. 1, pp. 32-42, 1987.

[3] K. Kazerounian and Z. Wang, "Global versus local optimization in redundancy resolution of robotics manipulators," *Int. Journal of Robotics Research*, Vol. 7. No. 5, pp.3-12, 1988.

[4] K. C. Suh and J. M. Hollerbach, "Local versus global torque optimization of redundant manipulators," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 619-624, 1987.

[5] C. A. Kelin and C. H. Hung, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 13, pp. 245-250, 1983.

[6] R. V. Dubey, J. A. Euler, and S. M. Babock, " An efficient gradient projection optimization scheme for a seven-degree-of-freedom redundant robot with spherical wrist," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 28-36, Philadelphia, PA, 1988.

[7] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for manipulator control," *ASME Journal of Dynamic Systems, Measurment, and Control*, Vol. 108, pp.163-171, 1986.

[8] H. Seraji, "Task options for redundancy resolution using configuration control," *30th IEEE Conf. on Decision and Contr*ol, pp. 2793-2798, 1991.

[9] R.Colbaugh, H.Seraji, and K. Glass, "Obstacle avoidance of redundant robots using configuration control," *Int. Journal of Robotics Research*, Vol. 6, pp. 721-744, 1989.

[10] J. Bailieul, "Avoiding Obstacles and resolving kinematic redundancy," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1698-1704, 1986.

[11] O. Egeland, "Task-space tracking with redundant manipulators," *IEEE Journal of Robotics and Automation*, Vol. 3, pp. 471-475, 1987.

[12] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem of redundant manipulators," *IEEE Journal of Robotics and Automation*, Vol. 4, pp. 403-410, 1988.

[13] H. Seraji and R. Colbaugh, "Improved Configuration Control for redundant robots," *Journal of Robotic Systems*, Vol. 7, No. 6, pp. 897-928, 1990.

[14] C. W. Wampler, "Manipulator inverse kinematic solution based on vector formulation and damped least-squares methods," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 16, No. 1. pp. 93-101, 1986.

[15] H. Seraji, "Task options for redundancy resolution using configuration control," *30th IEEE Conf. on Decision and Control*, pp. 2793-2798, 1991.

[16] H. Seraji, R. Colbaugh, "Singularity-robustness and task prioritization in configuration control of redundant robots," *29th IEEE Conf. on Decision and Control*, pp. 3089-3095,1990.

[17] J. M. Hollerbach, and K. C. Suh, "Redundancy resolution for manipulators through torque optimization," *Int. Journal of Robotics Research*, vol. 3, pp. 308-316, 1987.

[18] A. De Luca, "Zero Dynamics in Robotic Systems," in *Nonlinear Synthesis*, C. I. Byrnes and A. Kurzhanski (Eds.), Progress in Systems and Control Series, Birkhauser, Boston, MA, 1991.

[19] P. Hsu, J. Hauser, and S. Sastry, "Dynamic Control of Redundant Manipulators," *Journal of Robotic Systems*, vol. 6, pp. 133-148,1989.

[20] D. E. Whitney, "Historical prespective and state of art in robot force control," *Int. Journal of Robotics Research*, Vol. 6, No. 1, Dec. 1987.

[21] K.J. Salisbury, "Active stiffness control of manipulators in Cartesian coordinates," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 95-100, 1980.

[22] M. Raibert, J. Craig, "Hybrid position-force control of manipulators," *ASME Journal of Dynamic Systems, Measurment, and Control*, Vol. 102, pp. 126-133, 1981.

[23] Y. Yashikawa, "Dynamic hybrid position/force control of robot manipulators," *IEEE Journal of Robotics and Automation*, Vol. 3, No. 5, pp. 386-392, 1987.

[24] N. H. McClamroch and D. Wang, "Feedback stablization and tracking in constrained robots," *IEEE Trans. on Automatic Control*, Vol. 33, No. 5, pp. 419-426, 1988.

[25] N. Hogan, "Impedance control: An approach to manipulation," *ASME Journal of Dynamic Systems, Measurment, and Control*, Vol. 107, pp. 8-15, 1985.

[26] R. Anderson, and M.W. Spong, "Hybrid impedance control of robotic manipulators," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1073-1080, 1987.

[27] G.J. Liu, A.A. Goldenberg, "Robust hybrid impedance control of robot manipulators," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 287-292, 1991.

[28] H. Kazerooni., T. B. Sheridan, and P. K. Houpt, "Robust compliant motion for manipulators: Part I: The fundemental concepts of compliant motion" *IEEE Trans. on Robotics and Automation*, Vol. 2, No. 2, pp. 83-92, 1986.

[29] J. K. Mills, "Hybrid Control: A constrained motion perspective," *Journal of Robotic Systems*, Vol. 8, NO. 2, pp. 135-158, 1991.

[30] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," *Rob. Res., 1st Int. Symp.*, MIT Press, pp. 735-747, 1984.

[31] S. L. Chiu, "Task compatibility of manipulator postures," *Int. Journal of Robotics Research*, Vol. 7, No. 5, pp 13-21,Oct. 1988.

[32] J. Angeles, F. Ranjbaran, and R. V. Patel, "On the design of the kinematic structure of seven-axes redundant manipulators for maximum conditioning", *Proc. IEEE International Conf. Robotics and Automation*, pp.494-499, 1992.

[33] R. P. Paul, *"Robot Manipulators,"* MIT Press, Cambridge, MA, pp. 28-35, 1981.

[34] J. Y. S. Luh, M.W. Walker, and R. P. C. Paul, "Resolved-Acceleration of Mechanical Manipulators, " *IEEE Transaction on Automatic Control*, vol. AC-25, no. 3, pp. 468-474, June 1980.

[35] I.D. Walker, "The use of kinematic redundancy in reducing impact and contact effects in manipulation," *Proc. IEEE International Conf. Robotics and Automation*, pp. 434-439, 1990.

[36] M.W. Gertz, J. Kim, and P. Khosla, "Exploiting redundancy to reduce impact force," *IEEE/RSJ Workshop on Intell. Rob. Sys*, pp. 179-184, 1991.

[37] Z. Lin, R.V. Patel, and C.A. Balafoutis, "Augmented impedance control: An approach to impact reduction for kinematically redundant manipulators," to appear *Journal of Robotic Systems*, 1995.

[38] N. Adachi, Z. X. Peng, S. Nakajima, "Compliant motion control of redundant manipulators," *IEEE/RSJ Workshop on Intell. Rob. Sys.*, pp. 137-141, 1991.

[39] W.S. Newman, M.E. Dohring, "Augmented impedance control: An approach to compliant control of kinematically redundant manipulators," *Proc. IEEE International Conf. Robotics and Automation*, pp. 30-35, 1991.

[40] F. Shadpey, R.V. Patel, C. Balafoutis, and C. Tessier, " Compliant Motion Control and Redundancy Resolution for Kinematically Redundant Manipulators," *American Control Conference*, Seattle, WA, June 1995.

[41] C. Tessier et al., "Trajectory Planning and Object Avoidance (STEAR 5) - Phase II," Final Report, Vol. 1, DSS Canada, Contract No. 9F006-2-0107/01-SW , 1995.

[42] F. Shadpey and R.V. Patel, "Compliant motion control with self-motion Stabilization for kinematically redundant manipulators," *Third IASTED Int. Conf. on Robotics and Manufacturing*, June 1995, Cancun, Mexico.

[43] F. Shadpey and R.V. Patel, "Adaptive Compliant Motion Control of Kinematically Redundant Manipulators," *submitted to IEEE conf. on Decision and Contro*l, Dec. 1995.

[44] J. Duffy, "The fallacy of modern hybrid control theory that is based on "orthogonal complements" of twist and wrench spaces," *Journal of Robotic Systems*, Vol. 7, No. 2, pp. 139-144, 1990.

[45] P. R. Sinha and A.A. Goldenberg, "A unified theory for hybrid control of manipulators," *Proc. IFAC 12th World congress*, Sydney, Australia,1993.

[46] D. Dawson and Z. Qu, "Comments on impedance control with adaptation for robotic manipulators," *IEEE Trans. on Robotics and Au*tomation, Vol 7, No. 6, Dec. 1991.

[47] Y. Han, L. Liu, R. Lingarkar, N. Sinha, and M. Elbestawi, "Adaptive Control of Constrained Robotic manipulators," *Int. Journal of Robotics Research*, vol. 7, no. 2, pp.50-56, 1992.

[48] W. S. Lu, and Q. H. Meng, "Impedance control with adaptation for robotic manipulators," *IEEE Trans. on Robotics and Au*tomation, Vol 7, No. 3, June 1991.

[49] G. Niemeyer, and J.J. Slotine, "Performance in adaptive manipulator control," *Int. Journal of Robotics Research*, Vol. 10, No. 2, April 1991.

[50] R. Ortega and M. Spong, "Adaptive motion control of rigid robots: A tutorial." In *Proc. IEEE conf. on Decision and Control.*, Austin, Texas, 1988.

[51] N . Sadegh, and R. Horowitz, "Stability analysis of an adaptive controller for robotic manipulator," *in Proc. IEEE Int. Conf. Robotics and Automation.*

[52] J. J. Slotine, W. Li, "On the Adaptive Control of Robot Manipulators," *Int. Journal of Robotics Research*, vol. 6, no. 3, pp. 49-59, 1987.

[53] J. J. Slotine, and W. Li, *"Applied Nonlinear Control,"* Prentice Hall Inc., Englewood cliffs, NJ, 1991.

[54] K. S. Narendra and A. M. Annaswamy, *"Stable Adaptive Systems,"* Prentice Hall Inc., Englewood cliffs, NJ, 1989

[55] G. H. Golub and C. F. Van Loan, *"Matrix Computations,"* 2nd ed., John Hopkins Univ. Press, Baltimore, 1989

[56] L. Sciavicco , B. Siciliano, "An algorithm for reachable workspace for 2R and 3R planar pair mechanical arms", *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 1, pp 628-629, Philadelphia, PA,1988.

[57] P. Borrel , "Contribution a la modelisation geometrique des robots manipulateurs : Application a la conception assistee par ordinateur", *These d'Etat*, USTL, Monpollier, France, July 1986.

[58] C. A. Klien, "Use of redundancy in design of robotic systems," *Proc. 2nd Int. Symp. Robotic Res.*, Kyoto, Japan, 1984.

[59] S. Borner, and R. B. Kelley,"A novel representation for planning 3-D collision free paths," *IEEE Transaction on Syst., Man, and Cybernetics*, vol. 20, no. 6, pp. 1337-1351, 1990.

[60] T. Hasegawa, H. Terasaki,"Collision avoidance: "Divide-and-conquer approach by space characterization and intermediate goals," *IEEE Transaction on Syst., Man, and Cybernetics,* vol. 18, no. 3, pp. 337-347, 1988.

[61] P. Khosla and R. V. Volpe, "Superquadric artificial potentials for obstacle avoidance and approach," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1778-1784, 1988.

[62] D. G. Hunter, "An overview of the Space Station Special Purpose Dexterous Manipulator," *National Research Council Canada*, NRCC no. 28817, Issue A, 7 April 1988

[63] F. Shadpey, C. Tessier, R.V. Patel, B. Langlois, and A. Robins, "A trajectory planning and object avoidance system for kinematically redundant manipulators: An experimental evaluation," *AAS/AIAA American Astrodynamics Conference*, Aug. 1995, Halifax, Canada.

[64] F. Shadpey, C. Tessier, R.V. Patel, and A. Robins, "A trajectory planning and obstacle avoidance system for kinematically redundant manipulators," *CASI Conference on Astronautics*, Ottawa, Nov. 1994.

[65] F. Ranjbaran, J. Angeles, M. A. Gonzalez-Palacios, and R. V. Patel , "The mechanical design of a seven-axes manipulator with kinematic isotropy," *Journal of Robotics and Intelligent Systems*, Vol.~13, pp. 1-21, 1995. .

[66] K. Glass, R. Colbaugh, D. Lim, and H. Seraji, “Real-time Collision avoidance-for redundant manipulators,” *IEEE Transaction on Robotics and Automation*, pp. 448-457, vol 11. no. 10, 1995.

[67] G. R., Veldkamp, ``On the use of dual numbers, vectors and matrices in instantaneous, spatial kinematics,” *Mechanism and Machine Theory*, vol. 11, pp. 141-156, 1976.

[68] A. T., Yang, and F. Freudenstein “Application of dual-number quaternion algebra to the analysis of spatial mechanisms,” *Trans. ASME J. Appl. Mech.*, pp. 300-308, 1964.

[69] M. A. Gonzalez Palacios, J. Angeles and F. Ranjbaran, “The kinematic synthesis of serial manipulators with a prescribed Jacobian”, *Proc. IEEE Int. Conf. Robotics Automat.*, Atlanta, Georgia, 1993, vol. 1, pp 450-455.

[70] J. Y. S. Luh, M.W. Walker, and R. P. C. Paul, “Resolved-Acceleration of mechanical manipulators, “ *IEEE Transaction on Automatic Contro*l, vol. AC-25, no. 3, pp. 468-474, June 1980.

[71] F. Shadpey, M. Noorhosseini, I. Bryson, R. V. Patel, “An integrated robotic development environment for task planning and obstacle Avoidance”, *Third European Conf. on fEng. System Design & Analysis (ASME)*, Montpellier, France, July 1996.

[72] D. S. Watkins, “Fundamentals of Matrix Computations,” John Willey and Sons.

[73] F. Shadpey, R, V. Patel, “Robot dynamic modelling software (RDM): User's guide ”, Concordia University, Feb. 1997.

[74] B. W. Char, et al., “ Maple V Language Reference Manual,”, Watt. Springer-Verlag, New York, 1991

[75] D. B. Stewart, R. A. Volpe, and P. K. Khosla, 1992, "Integration of Real-Time Software Module for Reconfigurable Sensor-Based Control Systems", *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '92)*, Raleigh, North Carolina, pp. 325-332.

[76] "Matlab External Interface Guide for UNIX Workstation," The MathWorks Inc., 1992.

[77] J. J. Criag, "Introduction to Robotics: Mechanics and Control," Addison-Wasely, 2nd Ed., 1995.

[78] H. Seraji, D. Lim, and R. Steele, "Experimens in contact control," *J. Rob. Syst.*, Vol. 13, No.2, pp. 53 - 73, 1996.

[79] T.D. Tuttle and W.P Seering," A nonlinear model of a harmonic drive gear transmission," *IEEE Trans. Rob. and Aut.*, Vol. 12, No., 3, June 1996.

[80] Colombina, F. Didot, G. Magnani, and A. Rusconi, "External servicing testbed for automation and robotics," *IEEE Robotics & Automation Magazine*, Mar. 1996, pp. 13-23

[81] REACT in IRIX 5.3 - Technical Report," Silicon Graphics Inc., Revised Dec. 1994.

[82] F. Shadpey, F. Ranjbaran, R.V. Patel, and J. Angeles, "A compact cylinder-cylinder collision avoidance scheme for redundant manipulators," *Sixth Int. Symp. on Robotics and Manufacturing (ISRAM)*, Montpellier, France, May, 1996.

APPENDIX

A SOFTWARE DEVELOPMENT FOR THE AHIC SCHEME

A.1 REQUIREMENT SPECIFICATION

The block diagram of the AHIC scheme is shown in Figure 5.1. As one can see, the presence of interaction forces in the AHIC scheme, requires closed-loop implementation of all software modules. This introduces additional requirements for the software development phase from the following perspectives:

- Real-time implementation requires high level of code and algorithm optimization.

- In order to test the integrated system, either both the controller and the simulation of the arm and the environment should be implemented on the same platform, or an appropriate communication scheme should be developed to allow real-time (in the case of hardware demonstration) or scaled-time data passing between different platforms.

- The directory design should ease the management of the source codes and configuration files. This is necessary because one source code can be used (linked) by different modules on different platforms. The *Makefiles* should be designed in such a way as to reflect this fact. For instant, the derivative function *"Diff.c"* can be used by the real-time controller running on the SGI workstation, and also in the real-time code running on the processors located on the VME based controller. It should be noted that although both of these platforms use the ANSI-C standard, they are not compatible at the object code level. Therefore, the *Makefile* on each platform should use the same source code, but, each generates its own object code.

A.2 SOFTWARE DEVELOPMENT PROCESS

The steps that are followed at different levels of the algorithm extension, integration, testing, and verification are listed below:

1- Design of the directory structure for the CFC software: Figure A.1 shows this structure. It was noted in the previous sections that the CFC controller and simulation is a multi-platform software. In order to ease the software development and also testing and debugging processes, a file-server has been used to create a common disk space for different platforms at Concordia University. The *Common* directory contains the source and header files of the utility function, as well as the functions used on different simulation and controller modules. It also contains the common user-specified configuration files. The *Matlab* directory contains the *Cmex* [76] (C-files with gating functions for *Matlab*) used in the *Simulink* simulation environment.The *SGI* directory contains the files used in the real-time simulation and control on the SGI workstation. *Chimera* contains the files used by the VME-based processors during the hardware experiments- the development environment is a SUN 4/370 workstation.

2- Algorithm extension: Based on the block-diagram of the AHIC (see Figure 5.1), the input-output block diagram of each module are identified, and the required *C* function is generated. At this stage, the algorithmic optimization as well as symbolic code optimization are performed.

3- **Testing and verification of single modules:** In most cases an the operation of module is verified by comparing the results with an equivalent *m-file* in Matlab.

4- **Test and verification of the controller** in the *Simulink* environment using the *Cmex* modules (without arm dynamics).

5- **Trade-off study and stability/robustness analysis** to finalize the choice of control scheme. This is performed on the *Simulink* environment. The rigid model of the arm and its hardware accessories are generated by the RDM software [73].

6- **A second level of algorithm development** is performed for the new (or modified) modules based on the trade-off study and the stability/ robustness analysis.

7- Based on the controller simulation module (step 4) a scaled-time version of the controller is developed on the SGI workstation. The kinematic simulation of step 4 is used for verification/debugging.

8- **A real-time version of the kinematic simulation** of the controller is built to verify the real-time operation.

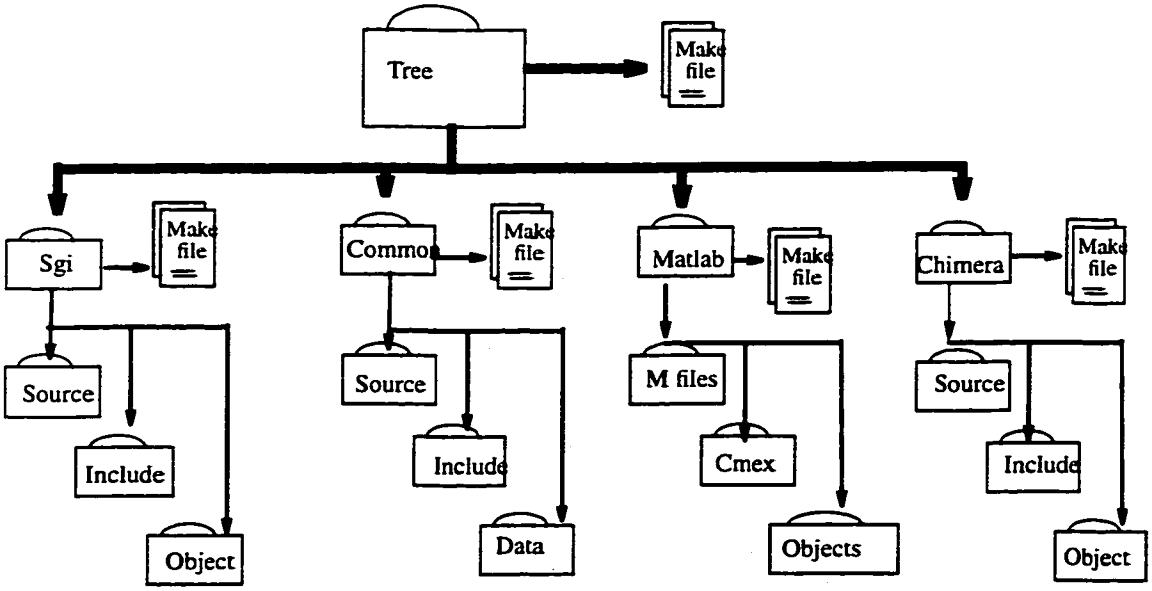


Figure A.1 Directory structure of the CFC software

Table A-1 summarizes the different simulation environments used during the software development

Table A-1 Summary of simulation environments

Type of simulation	platform	Purpose	Software	Timing
Kinematic simulation (Controller)	SGI	Algorithm development & verification	Simulink + Cmex module	Scaled-time ~(5:1)
Dynamic simulation controller + ARM model (rigid model)	SGI	Selection of control scheme	Simulink + Cmex module	Scaled-time ~(30:1)
Kinematic simulation (Controller)	SGI	Controller verification	C	scaled-time ~(1:4)
Kinematic simulation	SGI	Controller Verification	C	Real-time
Hardware controller	SGI+ VME controller + Sun 4	Hard-ware demo.	C + Chimera	Real-time

A.2.1 Optimization Process

During algorithm development in Section 3.3, the main focus was the generality and clarity of the codes (though a certain level of code optimization was performed), while the main criterion in extending the AHIC modules is the real-time implementation. This section describes the optimization process which is performed at the algorithm and code levels

- Optimization at the Algorithm Level

At this stage, different methods are exploited to find the most efficient numerical or algorithmic solution. As an example, the damped least-squares solution of redundancy resolution may be found using matrix inversion according to the following method:

```
Step 1   LU decomposition of A
Step 2   for($i = 1 \rightarrow 7$)
             solve   $(LU)(A^{-1})_i = I_i$
```

where the subscript *i* denotes the *ith* column of a matrix. Step 1 requires $\frac{1}{3}N^3$ flops, while solving each equation in step two using forward and back-substitution requires $\left(\frac{1}{6}+\frac{1}{2}\right)N^3$ flops [72]. Therefore, the total operations count for this procedure is $\left(\frac{1}{3}N^3+\frac{2}{3}N^4\right)$. A more efficient algorithm takes advantage of the positive-definite property of the matrix A to use the Cholesky decomposition which only requires $\frac{1}{6}N^3$ flops, the total count for the algorithm would be $\frac{5}{6}N^3$. For $N = 7$ we have a reduction of order 6.

- Symbolic optimization of the code

The main purposes of this stage of optimization is to combine the generality of algorithm development with the highest degree of optimization resulting from introducing the parameters of the special manipulator (target manipulator) for which the algorithm is being developed. Figure A.2 shows the block diagram for the symbolic optimization stage. As we can see the two main inputs to this module are a *Maple* script file containing the algorithm (expressed in recursive or closed form), and a data file for the target manipulator. The data file may contain numerical and/or symbolic values for the target manipulator. As an example the data file to calculate the forward kinematic function of REDIESTRO contains the following lines.

$$link_{twist} = array([0, ALP2, ALP3, ALP4, ALP5, ALP6, ALP7, ALP8])$$

$$link_{length} = array([0, 0, L3, 0.L5, 0, L7, L8])$$

$$link_{offset} = array([OFF1, OFF2, OFF3, OFF4, OFF5, OFF6, OFF7, OFF8]$$

In this way to apply the algorithm to a new manipulator it is only necessary to run this stage for the new target manipulator. The output file is the C code tailored and optimized for the target manipulator.

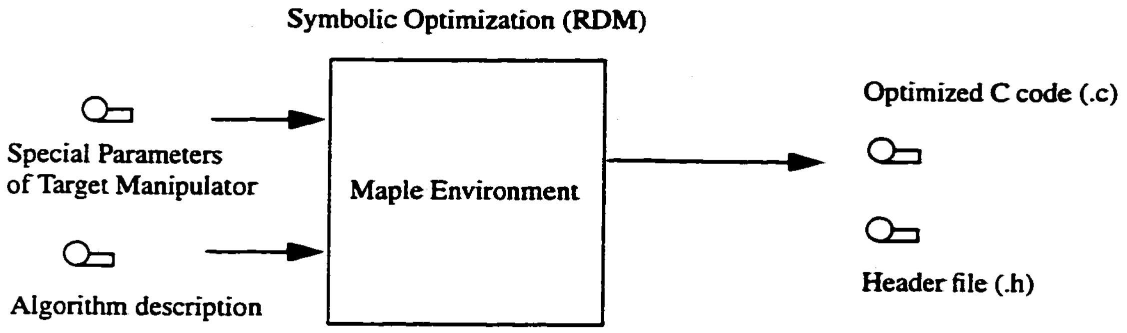


Figure A.2 Input/output block diagram of code optimization module

The other aspect of code optimization is removing all redundant calculations and assignments. Symbolic manipulation software (such as *Mathematica* or *Maple* [74]) provides optimization tools. However, in our first stages of development, we found out that this tool is not useful when the amount of calculation and number of variables become large. The reason is that the optimization is being performed only on the final output variables which, for a 7 DOF manipulator, need a huge number of intermediate calculations.

The Robot Dynamic Modeling Software (RDM) [73] was used to perform the symbolic optimization of the codes. Table A-2 gives the complete list of the functions that are symbolically optimized by RDM and their computational costs.

Table A-2 Functions optimized by RDM symbolic optimization module

Algorithm	**add.**	**multiplication**
Forward Kinematics	322	420
Inverse dynamics	1150(add+mult)	-
Angle-Axis derivative to Angular velocity & acceleration	144	206
Inertia matrix	962	1185
Coriolis + gravity + external force	505	536
RR	434(add+mult)	-
Gravity vector	148	226
Coriolis	495	531

A.2.2 Detail Design and Coding

The major modules involved in the CFC are described in Section 5.2 of Chapter 5. In this appendix the detailed design and coding of each module is given.

A.2.2.1 Task Planner and TG

As described in Section 5.2.1 of Chapter 5, the user task is specified by splitting it into several segments. Each segment is described by the "nominal trajectories" for the position and orientation of the tool frame {T} in frame {C} and $\{C_i\}$ respectively, the hybrid task specification, and the desired forces and torques (see Figure 5.2). An example of a PPTF is given in Table A-3.

Table A-3 Pre-Programed Task File (PPTF)

Seg	T s	Pos. Flag	Des. Pos.	orien Flag	Des. Orient.	Selection vector	Desired Inertia	Desired damping	Desired stiffness	Desired Force/ torques
1	3	1	xxx	1	xxx	111111	xxxxxx	xxxxxx	xxxxxx	yyyyyy
2	6	1	xxy	1	xxx	110111	xxxxxx	xxxxxx	xxyxxx	yyxyyy
3	10	1	xxy	0	yyy	110yyy	xxxyyy	xxxyyy	xxyyyy	yyxyyy
-1										

Description of the PPTF entries are given below:

1- Segment number: this entry- seg(i), specifies a segment no. for each segment of a complete task. (-1) entry indicates the end of the task.

2- T[i]: Final time *(s)* at which the desired position and orientation of *seg(i)* are to be reached (absolute time with respect to start of the task)

3- Position Flag (0/1): Enables/disables the position tracking during execution of *seg(i)*.

4- Desired Position: Desired position *(m)* of the tool frame origin expressed in frame {C} at t = T(i).

5- Orientation Flag(0/1): Similar to item 3.

6- Desired Orientation: Similar to item 4. The desired orientation is given in the form of X-Y-Z fixed angles.

7- Selection Vector *S(i)*: This vector $[S_p(3), S_o(3)]$ specifies the controller mode (position/force) for the 3 positional DOF's of the {C} frame and (orientation/torque) for the 3 orientational DOF's of the $\{C_i\}$. A 0 entry indicates that the axis is force/torque controlled and a 1 entry indicates a position/orientation controlled axis.

8- Desired inertia M(i)

9- Desired damping B(i):

10- Desired stiffness K(i)

11- Desired force/torque F(i): This vector $[f(3), n(3))]$ specifies the desired interaction forces/torques exerted by the robot on the environment expressed in frame {C} and $\{C_i\}$ respectively.

Note that, depending on the nature of the hybrid task for each segment (specified by *S(i)*), some entries for that segment will not be used by the controller, and will be discarded ("don't care" entries). For instance, if an axis is specified to be force-controlled, the corresponding value for the desired position will be discarded, or if this axis is position-controlled, then the desired force along this axis is a "don't care".

Figure A.3 shows the input/output block diagram for this module

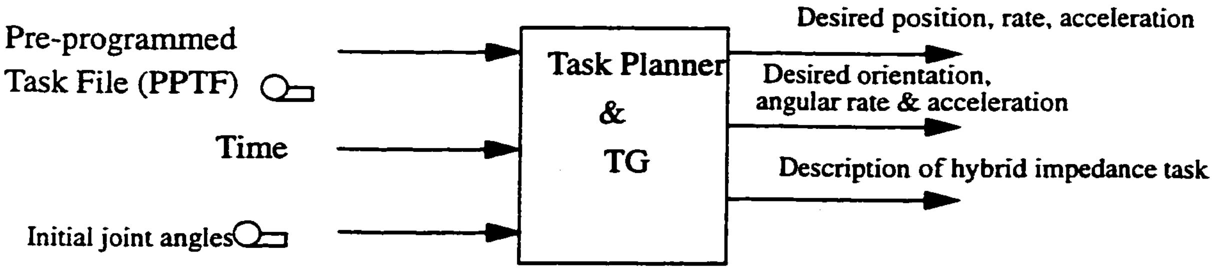

Pre-programmed
Task File (PPTF)
Time
Initial joint angles
Task Planner
&
TG
Desired position, rate, acceleration
Desired orientation,
angular rate & acceleration
Description of hybrid impedance task

Figure A.3 Top-level block diagram of the Task planner

Figure A.4 through Figure A.8 show the detailed flow-charts of the main functions that are required for this module. The function *ddAA2alpha()* which calculates the angular velocity and acceleration based on the angle-axis representation of the orientation and its first and second order derivatives is described in Appendix C. The Optimized C code for this function is produced by the symbolic optimization routine provided by the RDM software.

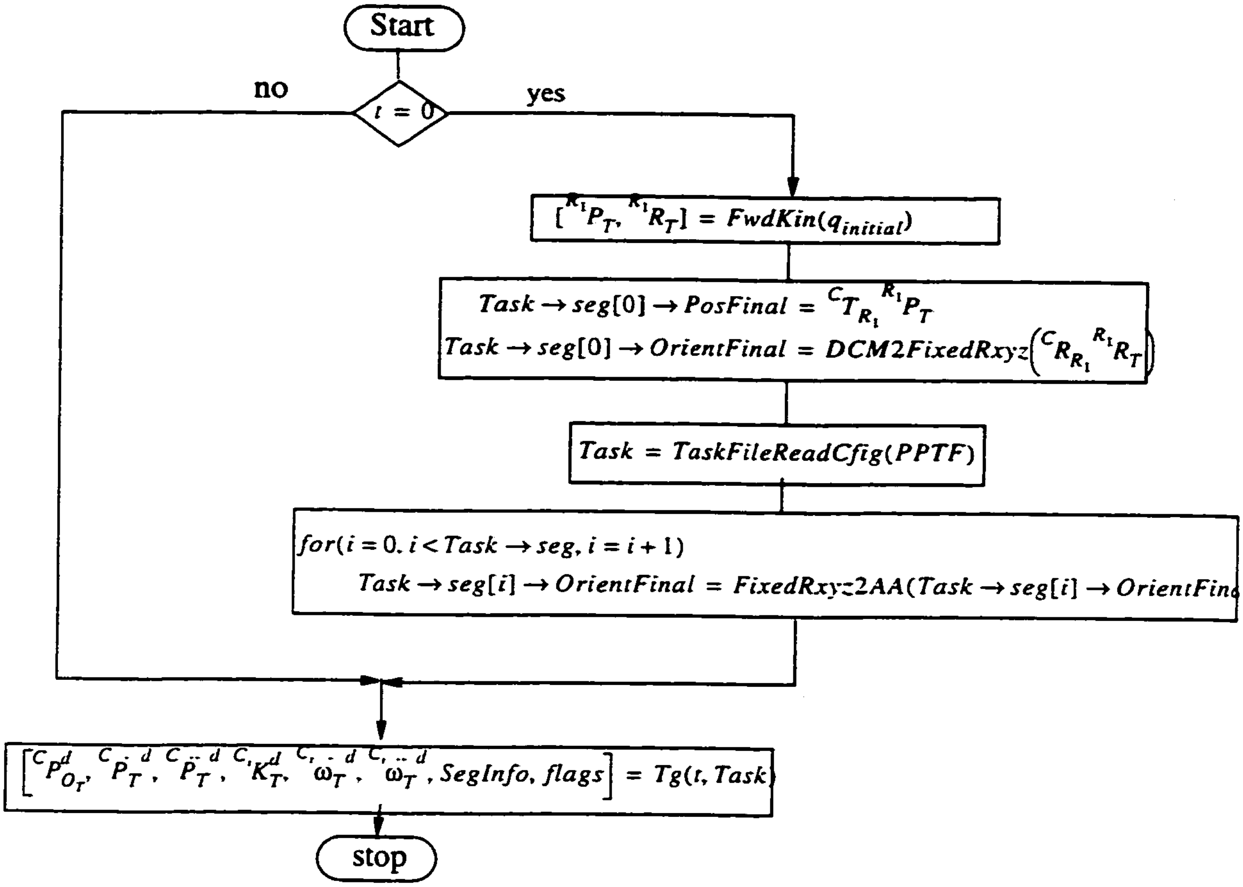

Start
no
yes
$t = 0$
$[{}^{R_1}P_T, {}^{R_1}R_T] = FwdKin(q_{initial})$
$Task \rightarrow seg[0] \rightarrow PosFinal = {}^{C}T_{R_1}{}^{R_1}P_T$
$Task \rightarrow seg[0] \rightarrow OrientFinal = DCM2FixedRxyz({}^{C}R_{R_1}{}^{R_1}R_T)$
$Task = TaskFileReadCfig(PPTF)$
$for(i = 0. i < Task \rightarrow seg, i = i + 1)$
$Task \rightarrow seg[i] \rightarrow OrientFinal = FixedRxyz2AA(Task \rightarrow seg[i] \rightarrow OrientFin$
$\left[{}^{C}P^{d}_{O_T}, {}^{C}\dot{P}^{d}_{T}, {}^{C}\ddot{P}^{d}_{T}, {}^{C,}K^{d}_{T}, {}^{C,}\dot{\omega}^{d}_{T}, {}^{C,}\ddot{\omega}^{d}_{T}, SegInfo, flags\right] = Tg(t, Task)$
stop

Figure A.4 Flow-chart for the "Task Planner & TG" module

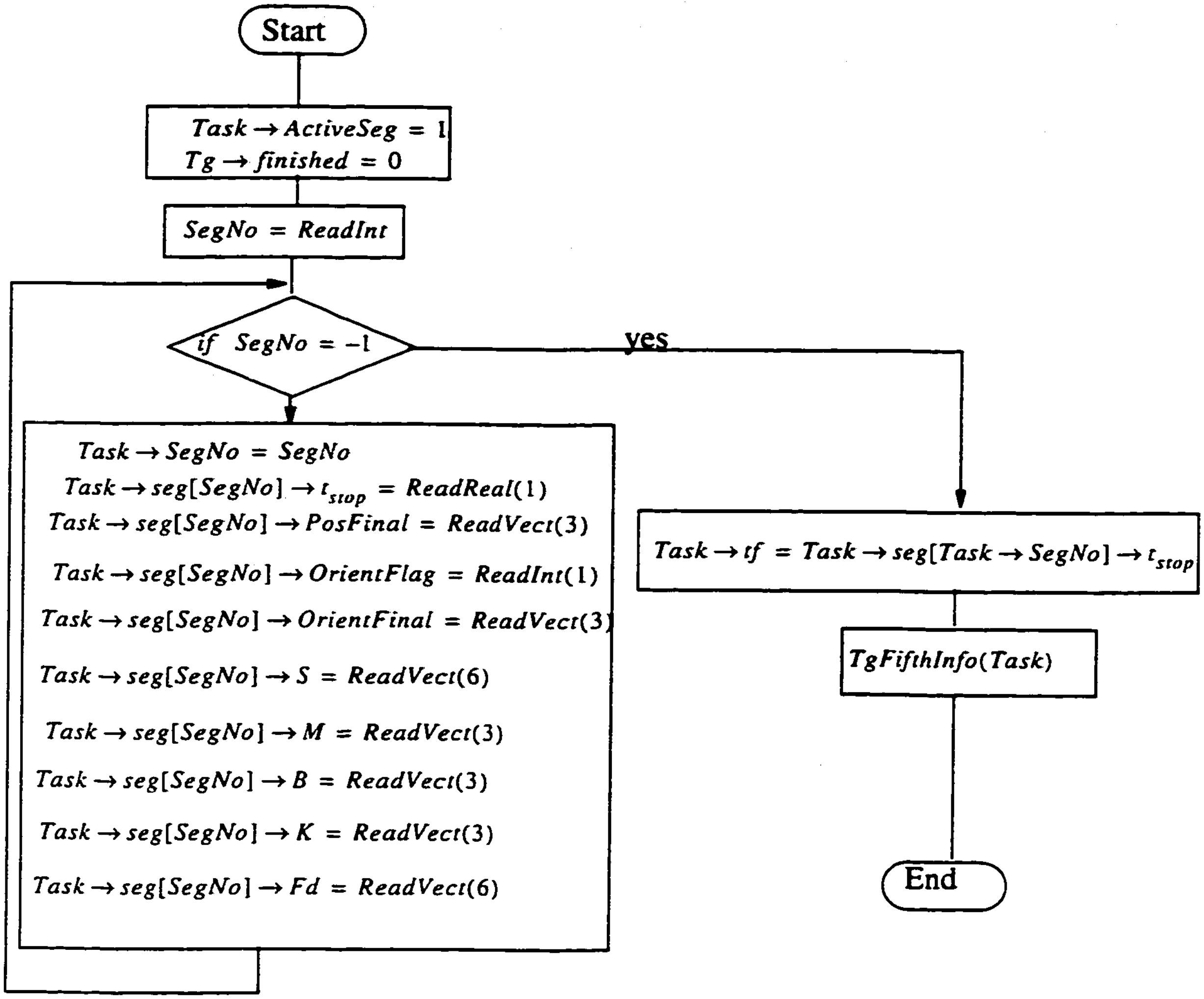

Start
Task → ActiveSeg = 1
Tg → finished = 0
SegNo = ReadInt
if SegNo = −1
yes
Task → SegNo = SegNo
Task → seg[SegNo] → t_stop = ReadReal(1)
Task → seg[SegNo] → PosFinal = ReadVect(3)
Task → seg[SegNo] → OrientFlag = ReadInt(1)
Task → seg[SegNo] → OrientFinal = ReadVect(3)
Task → seg[SegNo] → S = ReadVect(6)
Task → seg[SegNo] → M = ReadVect(3)
Task → seg[SegNo] → B = ReadVect(3)
Task → seg[SegNo] → K = ReadVect(3)
Task → seg[SegNo] → Fd = ReadVect(6)
Task → tf = Task → seg[Task → SegNo] → t_stop
TgFifthInfo(Task)
End

Figure A.5 Flow chart for the function *ReadTaskFileCfig*() which reads the PPTF

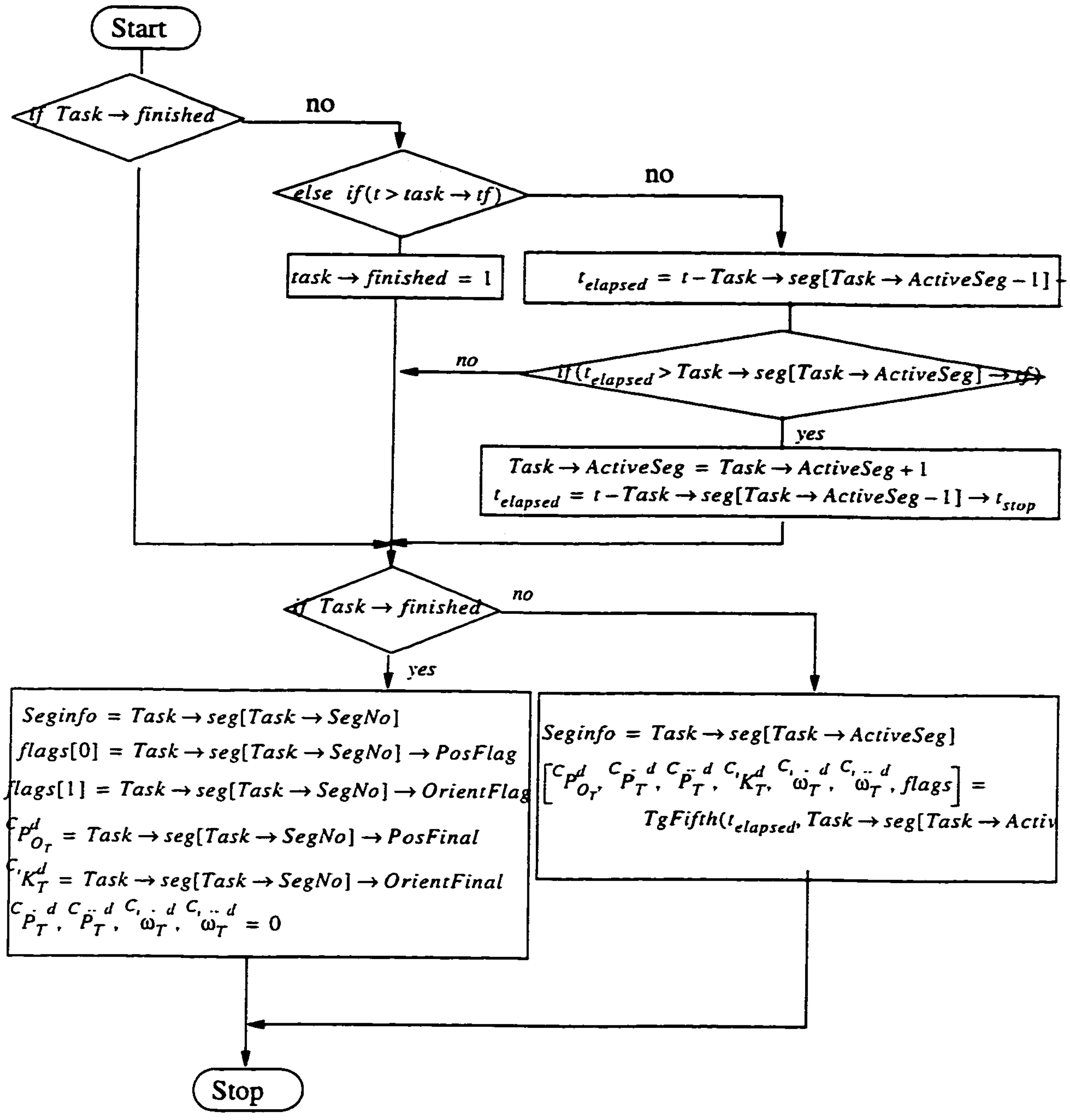
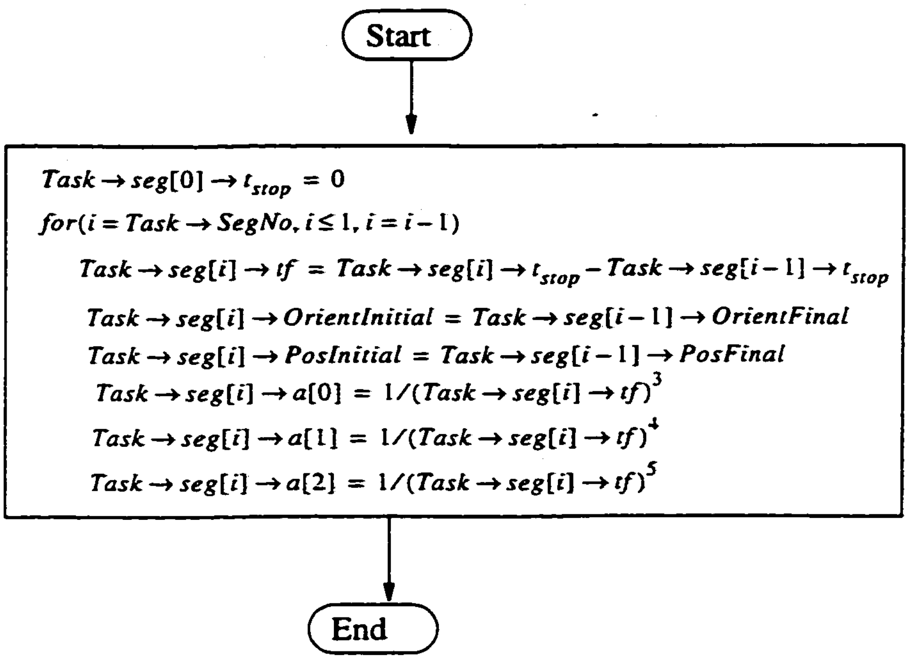

Start
if Task → finished
no
else if(t > task → tf)
no
task → finished = 1
$t_{elapsed}$ = t − Task → seg[Task → ActiveSeg − 1]
no
if($t_{elapsed}$ > Task → seg[Task → ActiveSeg] → tf)
yes
Task → ActiveSeg = Task → ActiveSeg + 1
$t_{elapsed}$ = t − Task → seg[Task → ActiveSeg − 1] → t_{stop}
if Task → finished
no
yes
Seginfo = Task → seg[Task → SegNo]
flags[0] = Task → seg[Task → SegNo] → PosFlag
flags[1] = Task → seg[Task → SegNo] → OrientFlag
${}^{C}P^{d}_{O_T}$ = Task → seg[Task → SegNo] → PosFinal
${}^{C'}K^{d}_{T}$ = Task → seg[Task → SegNo] → OrientFinal
${}^{C}\dot{P}^{d}_{T}, {}^{C}\ddot{P}^{d}_{T}, {}^{C'}\dot{\omega}^{d}_{T}, {}^{C'}\ddot{\omega}^{d}_{T} = 0$
Seginfo = Task → seg[Task → ActiveSeg]
$\left[{}^{C}P^{d}_{O_T}, {}^{C}\dot{P}^{d}_{T}, {}^{C}\ddot{P}^{d}_{T}, {}^{C'}K^{d}_{T}, {}^{C'}\dot{\omega}^{d}_{T}, {}^{C'}\ddot{\omega}^{d}_{T}, flags\right]$ =
TgFifth($t_{elapsed}$, Task → seg[Task → Activ
Stop

Figure A.6 Flow-chart for the function TG, which detects the active segment of the task, and calls the fifth-order trajectory-generator with appropriate parameters for the segment.



```
$Task \rightarrow seg[0] \rightarrow t_{stop} = 0$
$for(i = Task \rightarrow SegNo, i \leq 1, i = i - 1)$
    $Task \rightarrow seg[i] \rightarrow tf = Task \rightarrow seg[i] \rightarrow t_{stop} - Task \rightarrow seg[i-1] \rightarrow t_{stop}$
    $Task \rightarrow seg[i] \rightarrow OrientInitial = Task \rightarrow seg[i-1] \rightarrow OrientFinal$
    $Task \rightarrow seg[i] \rightarrow PosInitial = Task \rightarrow seg[i-1] \rightarrow PosFinal$
    $Task \rightarrow seg[i] \rightarrow a[0] = 1/(Task \rightarrow seg[i] \rightarrow tf)^3$
    $Task \rightarrow seg[i] \rightarrow a[1] = 1/(Task \rightarrow seg[i] \rightarrow tf)^4$
    $Task \rightarrow seg[i] \rightarrow a[2] = 1/(Task \rightarrow seg[i] \rightarrow tf)^5$
```

Figure A.7 Flow-chart for the function *TgFifthinfor()*, which pre-calculates the required parameters for the fifth-order TG module

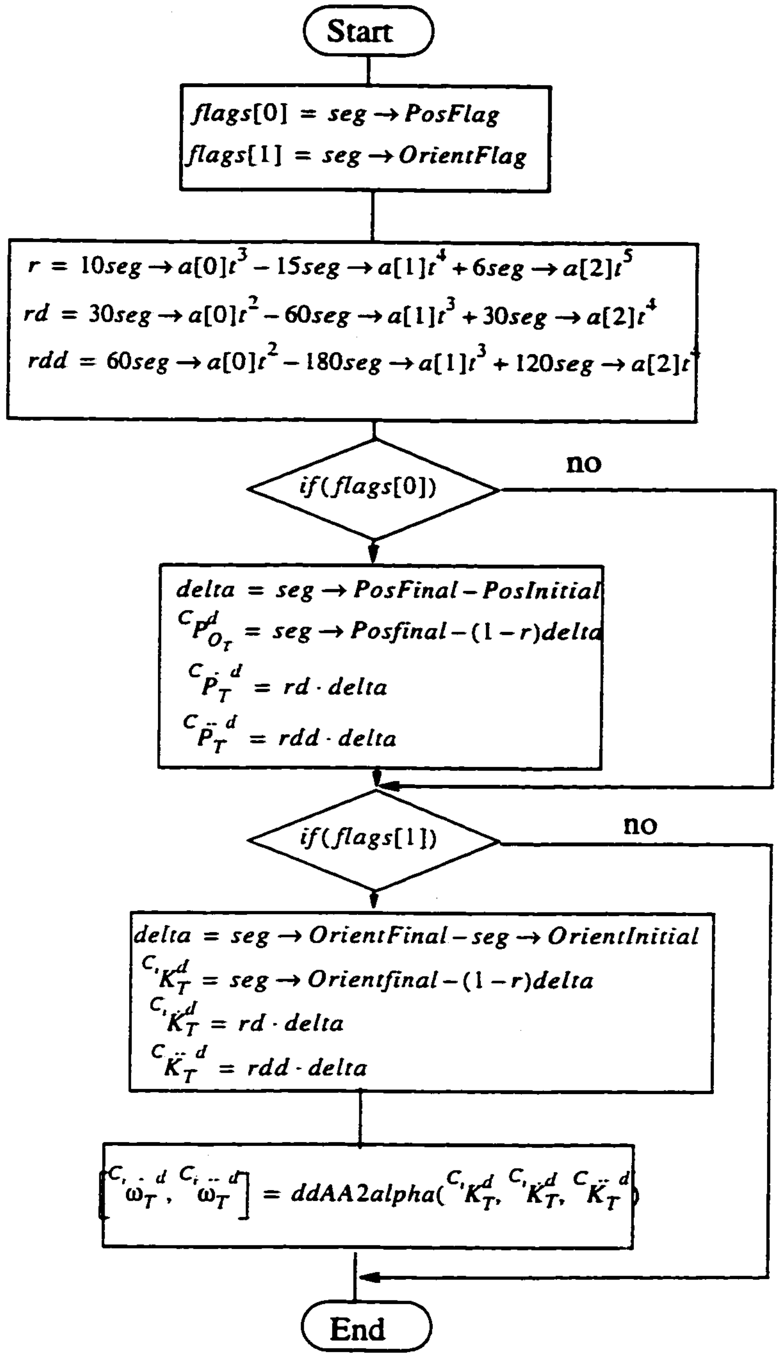

Start
$flags[0] = seg \rightarrow PosFlag$
$flags[1] = seg \rightarrow OrientFlag$
$r = 10seg \rightarrow a[0]t^3 - 15seg \rightarrow a[1]t^4 + 6seg \rightarrow a[2]t^5$
$rd = 30seg \rightarrow a[0]t^2 - 60seg \rightarrow a[1]t^3 + 30seg \rightarrow a[2]t^4$
$rdd = 60seg \rightarrow a[0]t^2 - 180seg \rightarrow a[1]t^3 + 120seg \rightarrow a[2]t$
$if(flags[0])$
no
$delta = seg \rightarrow PosFinal - PosInitial$
${}^{C}P^{d}_{O_T} = seg \rightarrow Posfinal - (1-r)delta$
${}^{C}\dot{P}^{d}_{T} = rd \cdot delta$
${}^{C}\ddot{P}^{d}_{T} = rdd \cdot delta$
$if(flags[1])$
no
$delta = seg \rightarrow OrientFinal - seg \rightarrow OrientInitial$
${}^{C,}K^{d}_{T} = seg \rightarrow Orientfinal - (1-r)delta$
${}^{C,}\dot{K}^{d}_{T} = rd \cdot delta$
${}^{C}\ddot{K}^{d}_{T} = rdd \cdot delta$
$\left[{}^{C,}\dot{\omega}^{d}_{T}, {}^{C,}\ddot{\omega}^{d}_{T}\right] = ddAA2alpha({}^{C,}K^{d}_{T}, {}^{C,}\dot{K}^{d}_{T}, {}^{C}\ddot{K}^{d}_{T})$
End

Figure A.8 Flow-chart for the function *TgFifth()*, the fifth-order trajectory generator

A.2.2.2 AHIC Module

A detailed description of this module is given in Section 5.2.2 of Chapter 5.

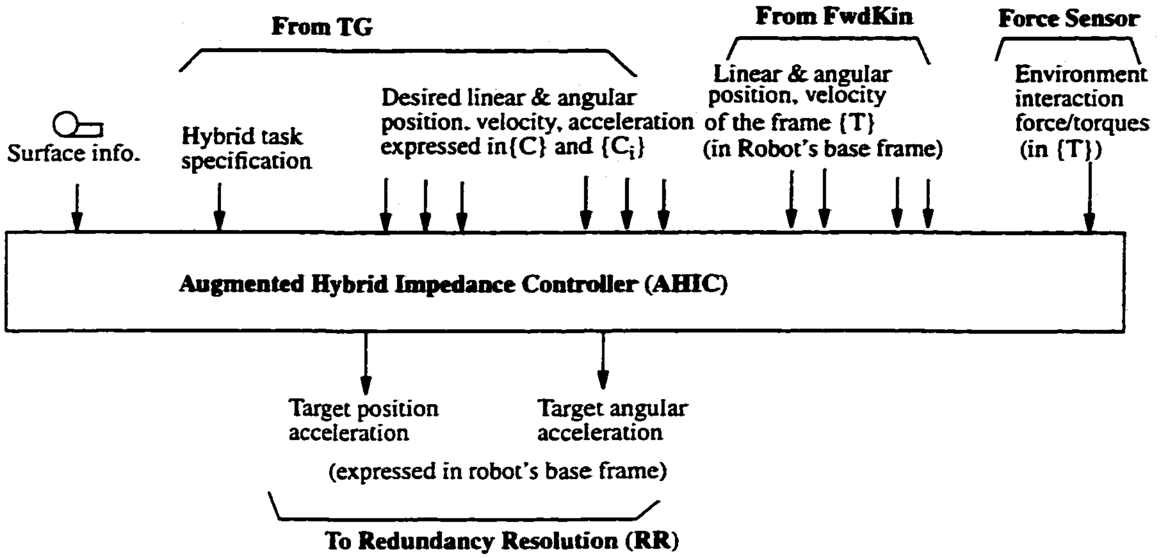

From TG
From FwdKin
Force Sensor
Surface info.
Hybrid task specification
Desired linear & angular position, velocity, acceleration expressed in{C} and {C$_i$}
Linear & angular position, velocity of the frame {T} (in Robot's base frame)
Environment interaction force/torques (in {T})
Augmented Hybrid Impedance Controller (AHIC)
Target position acceleration
Target angular acceleration
(expressed in robot's base frame)
To Redundancy Resolution (RR)

Figure A.9 Input/output block diagram of AHIC

The AHIC module (Figure A.9) consists of the AHIC function and the function *CframeReadCfig()* which reads the position and orientation of the frame {C} in the robot's base frame {R_1}(see Figure 5.2). This information is read from the configuration file *Cframe.cfig*.

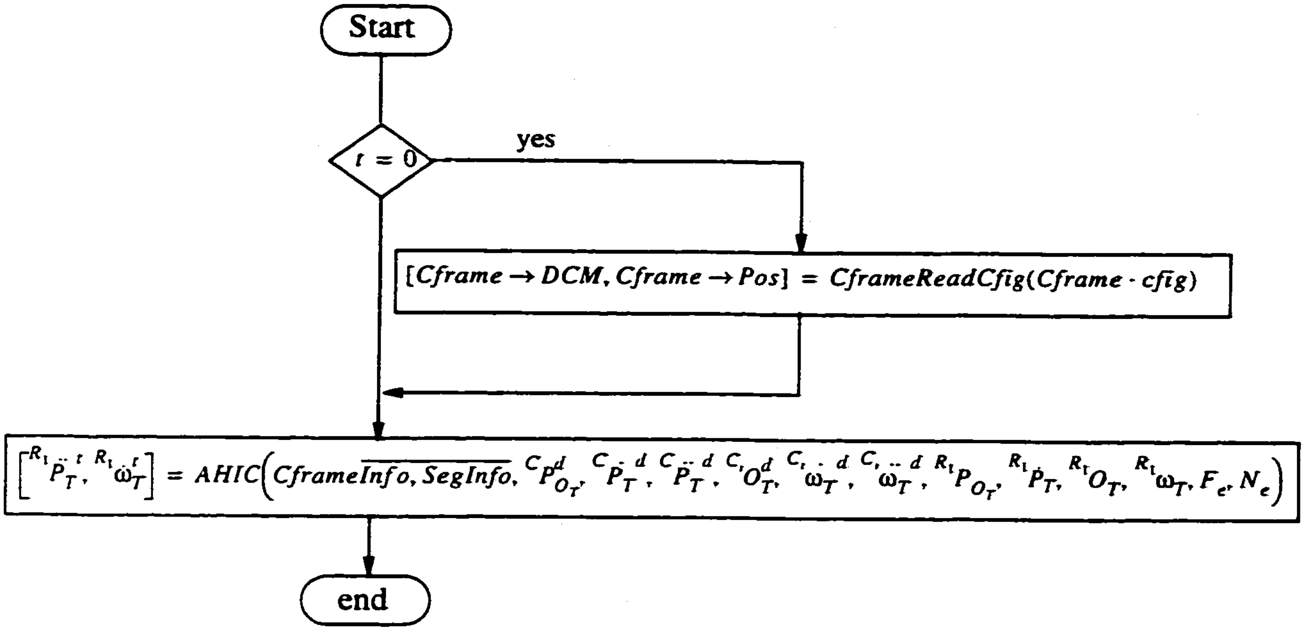

Start
$t = 0$
yes
$[Cframe \rightarrow DCM, Cframe \rightarrow Pos] = CframeReadCfig(Cframe \cdot cfig)$
$\left[{}^{R_t}\ddot{P}^{t}_{T}, {}^{R_t}\ddot{\omega}^{t}_{T}\right] = AHIC\left(Cframe\overline{Info, SegInfo}, {}^{C}P^{d}_{O_T}, {}^{C}\dot{P}^{d}_{T}, {}^{C}\ddot{P}^{d}_{T}, {}^{C_t}O^{d}_{T}, {}^{C_t}\dot{\omega}^{d}_{T}, {}^{C_t}\ddot{\omega}^{d}_{T}, {}^{R_t}P_{O_T}, {}^{R_t}\dot{P}_{T}, {}^{R_t}O_{T}, {}^{R_t}\omega_{T}, F_e, N_e\right)$
end

Figure A.10 Block diagram of the AHIC controller module

Three other functions, *iHTj()*, *jHTi()*, and *OrientTrack()* are called in this module. The first two functions perform homogenous transformations from frame *i* to *j* and vice-versa. The last function calculates the orientation error (see Section 3.3.2.2).

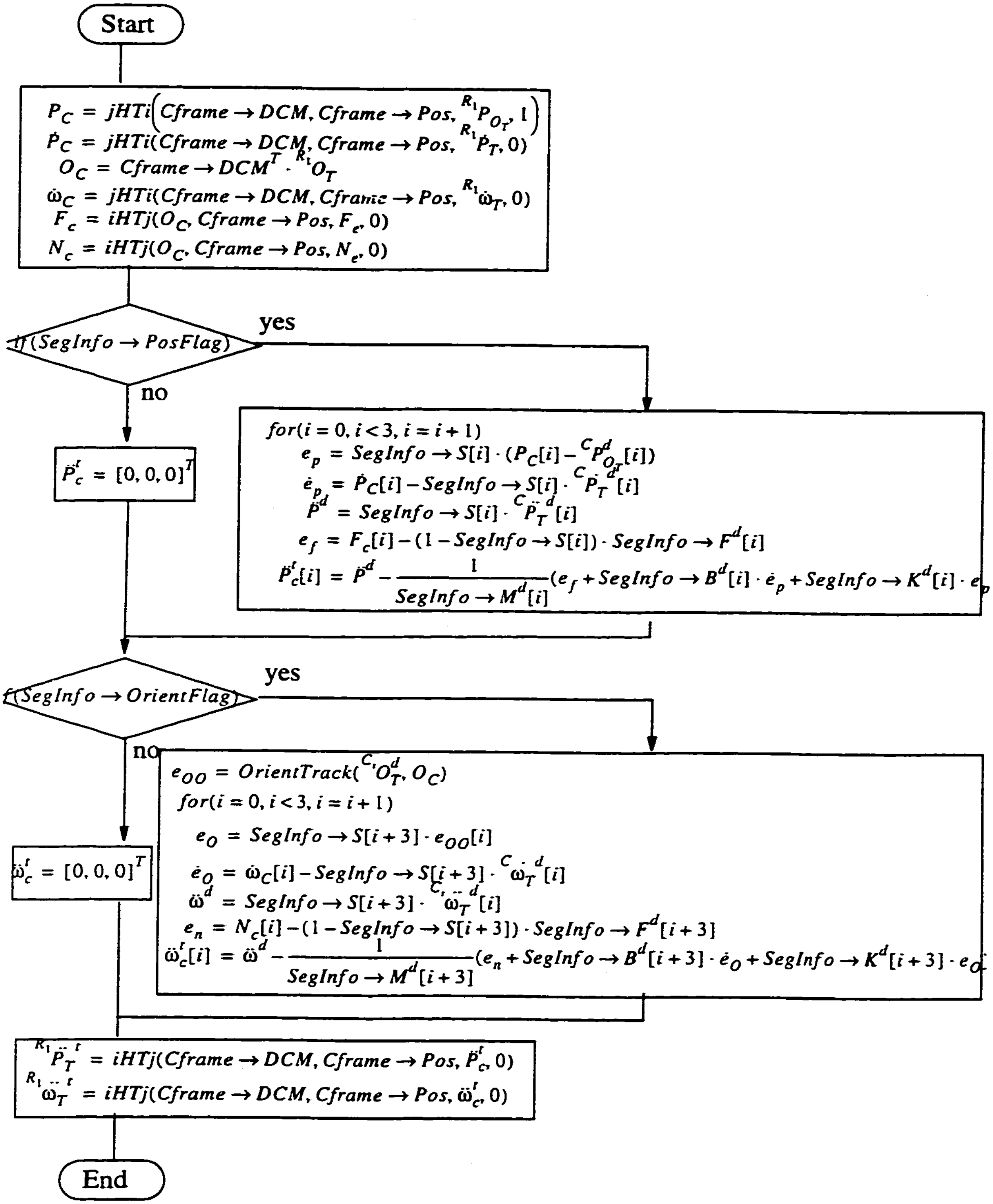

Start
$P_C = jHTi\left(Cframe \rightarrow DCM, Cframe \rightarrow Pos, {}^{R_1}P_{O_T}, 1\right)$
$\dot{P}_C = jHTi(Cframe \rightarrow DCM, Cframe \rightarrow Pos, {}^{R_1}\dot{P}_T, 0)$
$O_C = Cframe \rightarrow DCM^T \cdot {}^{R_1}O_T$
$\dot{\omega}_C = jHTi(Cframe \rightarrow DCM, Cframe \rightarrow Pos, {}^{R_1}\dot{\omega}_T, 0)$
$F_c = iHTj(O_C, Cframe \rightarrow Pos, F_e, 0)$
$N_c = iHTj(O_C, Cframe \rightarrow Pos, N_e, 0)$
$if(SegInfo \rightarrow PosFlag)$
yes
no
$\ddot{P}_c^t = [0, 0, 0]^T$
$for(i = 0, i < 3, i = i + 1)$
$e_p = SegInfo \rightarrow S[i] \cdot (P_C[i] - {}^{C}P_{O_T}^d[i])$
$\dot{e}_p = \dot{P}_C[i] - SegInfo \rightarrow S[i] \cdot {}^{C}\dot{P}_T^d[i]$
$\ddot{P}^d = SegInfo \rightarrow S[i] \cdot {}^{C}\ddot{P}_T^d[i]$
$e_f = F_c[i] - (1 - SegInfo \rightarrow S[i]) \cdot SegInfo \rightarrow F^d[i]$
$\ddot{P}_c^t[i] = \ddot{P}^d - \frac{1}{SegInfo \rightarrow M^d[i]}(e_f + SegInfo \rightarrow B^d[i] \cdot \dot{e}_p + SegInfo \rightarrow K^d[i] \cdot e_p$
$f(SegInfo \rightarrow OrientFlag)$
yes
no
$\ddot{\omega}_c^t = [0, 0, 0]^T$
$e_{OO} = OrientTrack({}^{C}O_T^d, O_C)$
$for(i = 0, i < 3, i = i + 1)$
$e_O = SegInfo \rightarrow S[i + 3] \cdot e_{OO}[i]$
$\dot{e}_O = \dot{\omega}_C[i] - SegInfo \rightarrow S[i + 3] \cdot {}^{C}\dot{\omega}_T^d[i]$
$\ddot{\omega}^d = SegInfo \rightarrow S[i + 3] \cdot {}^{C}\ddot{\omega}_T^d[i]$
$e_n = N_c[i] - (1 - SegInfo \rightarrow S[i + 3]) \cdot SegInfo \rightarrow F^d[i + 3]$
$\ddot{\omega}_c^t[i] = \ddot{\omega}^d - \frac{1}{SegInfo \rightarrow M^d[i + 3]}(e_n + SegInfo \rightarrow B^d[i + 3] \cdot \dot{e}_O + SegInfo \rightarrow K^d[i + 3] \cdot e_O$
${}^{R_1}\ddot{P}_T^t = iHTj(Cframe \rightarrow DCM, Cframe \rightarrow Pos, \ddot{P}_c^t, 0)$
${}^{R_1}\ddot{\omega}_T^t = iHTj(Cframe \rightarrow DCM, Cframe \rightarrow Pos, \ddot{\omega}_c^t, 0)$
End

Figure A.11 Flow-chart for the *AHIC()* function

A.2.2.3 Redundancy Resolution (RR) module

A detailed description of this module is given in Section 5.2.3 of Chapter 5. The input-output block diagram is given in Figure A.12.

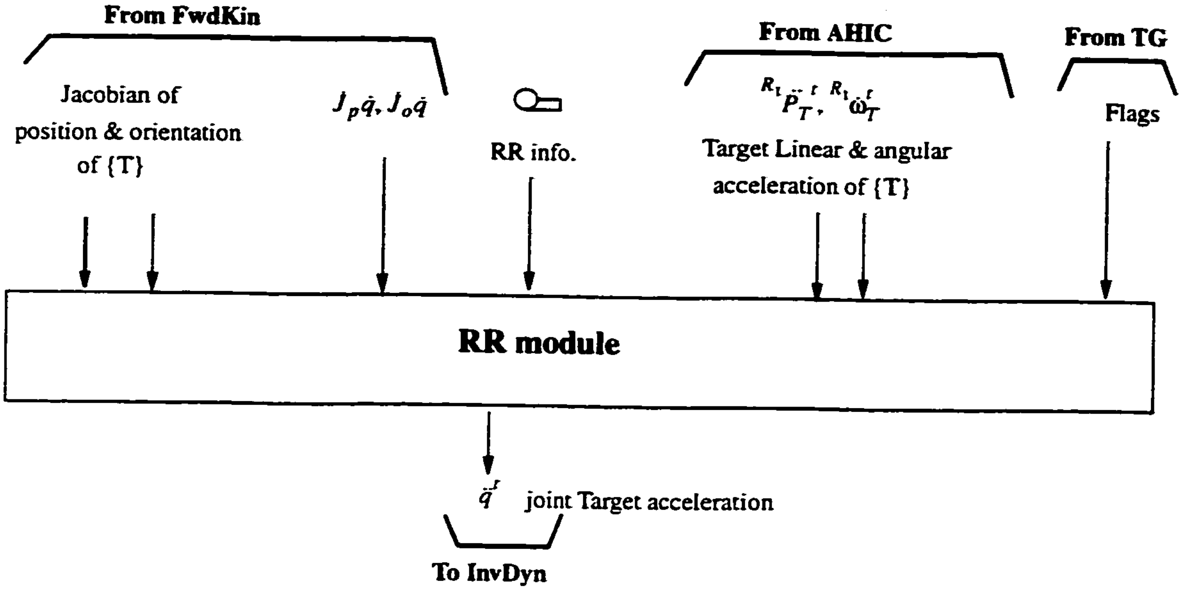
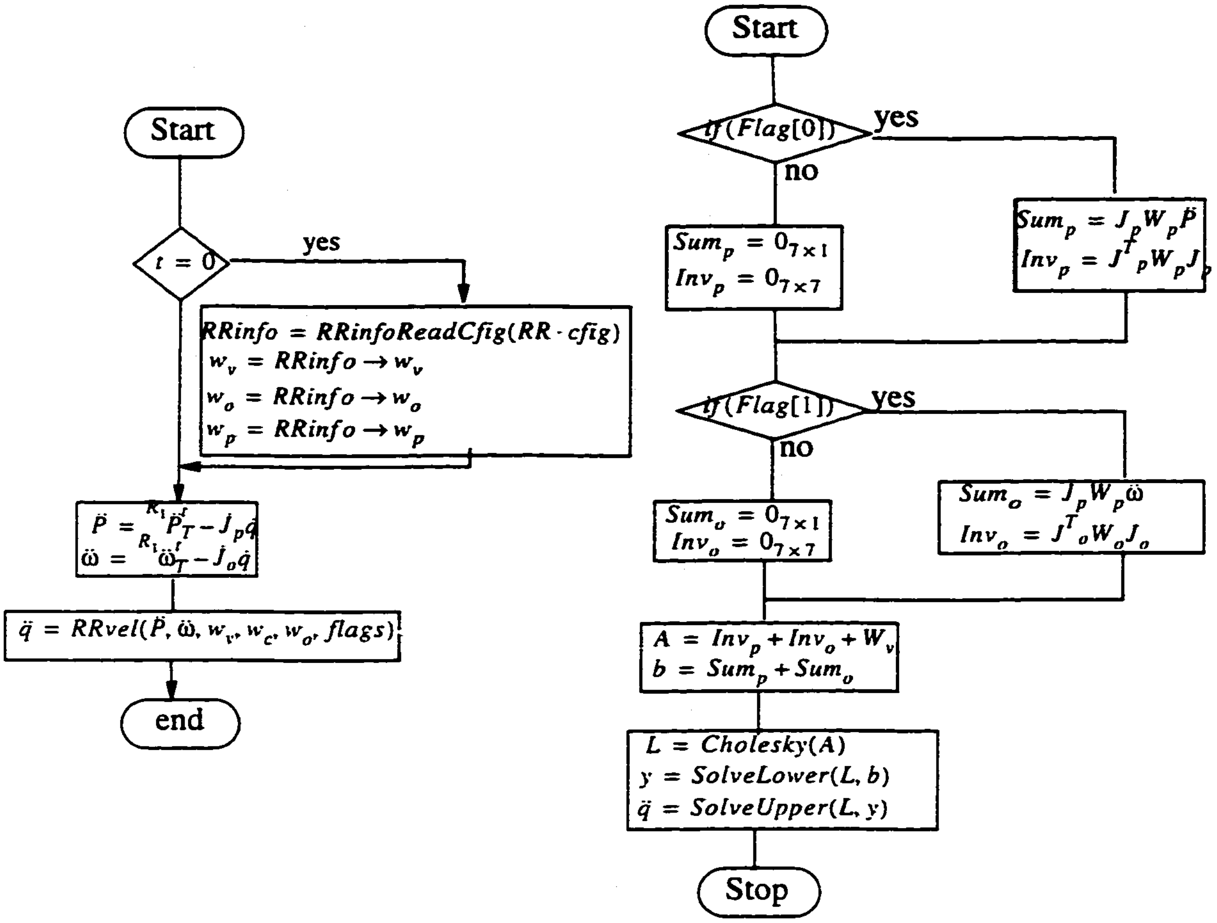

From FwdKin
Jacobian of position & orientation of {T}
$\dot{J}_p\dot{q}, \dot{J}_o\dot{q}$
RR info.
From AHIC
${}^{R_1}\ddot{P}_T^t, {}^{R_1}\dot{\omega}_T^t$
Target Linear & angular acceleration of {T}
From TG
Flags
RR module
$\ddot{q}^t$ joint Target acceleration
To InvDyn

Figure A.12 Input-output block diagram of the RR module

Figure A.13 (a,b) show the flow charts for the RR module and its function. The latter shows the calculation using matrix operations. However, because of the presence of too many redundant calculations in the formulation of the two terms, $J^T{}_pW_pJ_p$ and $J_pW_p\ddot{P}$, and the corresponding terms for the orientation, the optimized formulations of these terms are calculated by the symbolic optimization module of RDM.



(a) (b)

Figure A.13 Flow-charts of a) RR module, and b) *RRacc()* function

A.2.2.4 Forward Kinematics:

A detailed description of this module is given in Section 5.2.4 in Chapter 5. This function has been optimized by RDM's symbolic optimization module for REDIESTRO. Therefore, the DH parameters of REDIESTRO (see Table B-1) are not used as an input to the function (they are introduced in a header file).

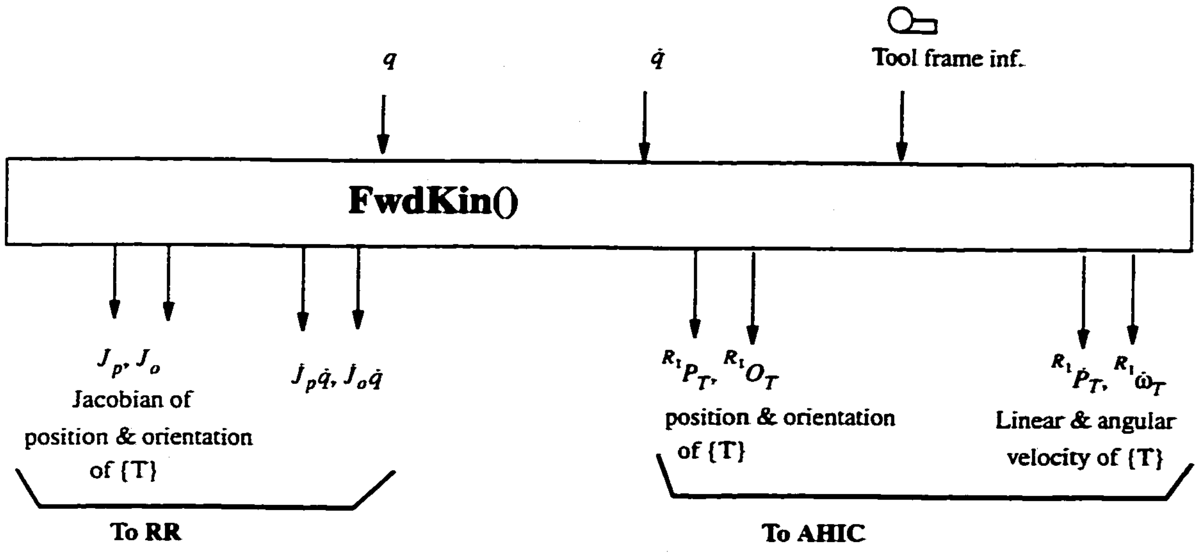
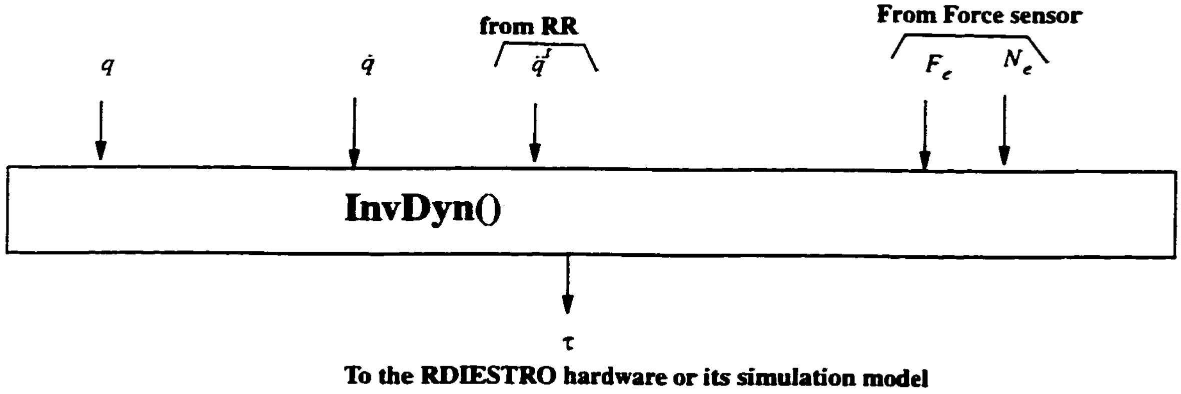


Figure A.14 Input-output block diagram of the function *RED_PosJacDjac_tool()*

A.2.2.5 Inverse Dynamics:

A detailed description of this module is given in Section 5.2.5 in Chapter 5. This module is symbolically optimized by RDM for REDIESTRO. Therefore, the dynamic parameters of REDIESTRO (see Appendix B) are not used as input to this function.



APPENDIX

B KINEMATIC AND DYNAMIC PARAMETERS OF REDIESTRO

This appendix summarizes the kinematic and dynamic parameters of REDIESTRO. It also provides the mechanical specification of the actuators and related hardware.

Table B-1 HD parameters of REDIESTRO

i	α_{i-1} **(deg)**	**a(i-1) mm**	**b(i) mm**	**q(i)[a]**
1	0.	0.	952.29	q(1)
2	-58.31	0.	-22.91	q(2)
3	-20.0289	231.13	36.93	q(3)
4	105.26	0.	0.	q(4)
5	60.91	398.84	-471.59	q(5)
6	59.88	0.	578.21	q(6)
7	-75.47	135.59	-145.05	q(7)
Tool	0	234.44	0	0

[a] Isotropic Configuration: q = [q1, -11.01, 91.94, 113.93, -2.26, 150.25, 63.76]

Table B-2 Mass (Kg)

Link1	**Link2**	**Link3**	**Link4**	**Link5**	**Link6**	**Link7**
17.313	5.580	28.586	7.390	5.987	2.557	0.200

Table B-3 Center of gravity in local frame {i}

.	Link1	Link2	Link3	Link4	Link5	Link6	Link7
X	0.00048	0.1155	-0.0011	0.3071	0	0.0919	0.06345
Y	-0.1607	-0.0036	-0.1176	-0.0408	-0.1326	-0.0343	0
Z	-0.1186	-0.0389	-0.1539	0.0699	0.1507	-0.0882	-0.0034

Table B-4 Link Inertia Tensor (Kg m^2)[a]

.	Link1	Link2	Link3	Link4	Link5	Link6	Link7
Ixx	0.89926	0.02573	1.6620	0.09297	0.8284	0.67522	0.004435
Iyy	0.31342	0.13223	0.7860	0.8881	0.7019	0.69288	0.005547
Izz	0.62745	0.11099	0.9387	0.8753	0.1317	0.03904	0.001136
Ixy	-2.7e-5	-0.0045	0.0001	-0.1203	0.00009	-0.00914	0.0
Iyz	0.3689	0.0012	0.1221	-0.0204	0.26852	-0.04921	0.0
Izx	-1.2e-5	-0.0404	0.0003	0.1411	0.00016	0.130282	-0.00189

a. The inertia tensor (in the frame located at center of gravity with the same orientation as the local frame {i}) is defined by:

$$I_{C^{A}_{i}} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{zx} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{yz} & -I_{zz} \end{bmatrix}$$

Table B-5 Motor assembly parameters

	1	**2**	**3**	**4**	**5**	**6**	**7**
Encoder resolution[a] (pulse/rev.)	200	360	360	360	1000	1000	1000
Gear ratio	200	260	260	260	160	160	110
Torque constant[b](Nm/A)	40	55	55	55	32	32	5.76
Maximum input current (A)	4.9	8.1	8.1	8.1	3.1	3.1	4.1
Actuator moment of inertia[c] (Kg m^2)	10.1	57.4	57.4	57.4	2.43	2.43	0.11
Columb friction(N.m)	19.2	47.3	47.3	47.3	10.24	10.24	0.92
Stiction (N.m)	15.36	25.84	25.84	25.84	8.2	8.2	0.74
Viscous coefficient (Nm.s/Rad)	0.14	0.34	0.34	0.34	0.09	0.09	0.02

a. The encoder resolutions will be four times greater (4*Encoder resolution) if the quadrature feature is used

b. Specified at the output shaft

c. Specified at the output shaft

Table B-6 Motor assembly interface specifications

		1	2	3	4	5	6	7
Encoders	Interface card resolution (bit)	32	32	32	32	32	32	32
Motors	Current amplifier gain (A/V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2
	Current amplifier Max. Current[a] (A)	12	12	12	12	12	12	12
	DAC (bits)	12	12	12	12	12	12	12
	DAC: Max. Output (V)	10	10	10	10	10	10	10
Force sensor (JR3)	Receiver card (bits)	14	14	14	14	14	14	14
	Max. Force	fx, fy = 200N, fz = 400 N; mx,my, mz = 12.5 Nm,						

[a] This is adjustable by changing a resistor in the hardware. At the moment it is set to maximum. allowable current for each motor.

APPENDIX C TRAJECTORY GENERATION (SPECIAL CONSIDERATION FOR ORIENTATION)

Special consideration for orientation

The desired orientation at the end of each segment is specified by the user in the preprogramed task file, PPTF (see Appendix A). This orientation is specified in the form of X-Y-Z Fixed Angles [77]. In this representation, the orientation is specified by a 3 dimensional vector $[\gamma, \beta, \alpha]$ which can be converted to a Direction Cosine Matrix (DCM) representation as follows:

$$R_{XYZ}(\gamma, \beta, \alpha) = R_Z(\alpha)R_Y(\beta)R_X(\gamma)$$

$$R_{XYZ}(\gamma, \beta, \alpha) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} \quad \text{(C.1)}$$

Let assume that the initial orientation $[\gamma_i, \beta_i, \alpha_i]$ and final orientation $[\gamma_f, \beta_f, \alpha_f]$ are specified in PPTF. Then the equivalent angle-axis representation is calculated based on the routine indicated in [77]. Having calculated the initial vector $[K_{x_i}, K_{Y_i}, K_{Z_i}]$ and the final orientation vector $[K_{x_f}, K_{Y_f}, K_{Z_f}]$ in the angle-axis form, the fifth order trajectory generator can be used to find the desired orientation vector $K(t)$.

It should be noted that the first and second derivatives ($\dot{K}(t)$, $\ddot{K}(t)$) of the desired orientation vector are not the angular velocity ω and acceleration Ω respectively.

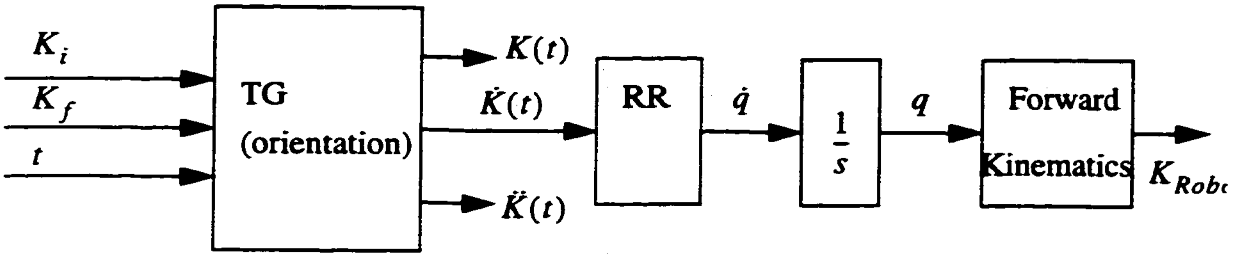


Figure C.1 Block diagram of the open-loop simulation for orientation TG.

The open-loop simulation (see Figure C.1) shows the robot's orientation $K_{Robot}(t)$. It does not follow the desired orientation $K(t)$. The desired angular velocity and acceleration can be calculated as follows:

$$[\omega, \dot{\omega}] = f(K(t), \dot{K}(t), \ddot{K}(t)) \qquad \text{(C.2)}$$

Derivation of the above function is explained below. The calculation of the angle-axis formulation from the DCM representation is as follows:

$$K(t) = [K_x, K_y, K_z]^T = k(t)\theta(t) \;\; where \;\; \theta(t) = \|K(t)\| \;\; and \; k(t) = \frac{K(t)}{\theta(t)} \qquad \text{(C.3)}$$

$$R = \begin{bmatrix} k_x^2 \upsilon\theta + c\theta & k_x k_y \upsilon\theta - k_z s\theta & k_x k_z \upsilon\theta + k_y s\theta \\ k_x k_y \upsilon\theta + k_z s\theta & k_y^2 \upsilon\theta + c\theta & k_y k_z \upsilon\theta - k_x s\theta \\ k_x k_z \upsilon\theta - k_y s\theta & k_y k_z \upsilon\theta + k_x s\theta & k_z^2 \upsilon\theta + c\theta \end{bmatrix} = \begin{bmatrix} a_x & n_x & s_x \\ a_y & n_y & s_y \\ a_z & n_z & s_z \end{bmatrix} \qquad \text{(C.4)}$$

where $\upsilon\theta = 1 - c\theta$. which yields:

$$tr(R) = 2c\theta + 1 \quad where \quad tr(R) = a_x + n_y + s_z \qquad \text{(C.5)}$$

$$k = \frac{vect(R)}{s\theta} \quad where \quad vect(R) = \frac{1}{2}\begin{bmatrix} n_z - s_y \\ s_x - a_z \\ a_y - n_x \end{bmatrix} \tag{C.6}$$

Now, we differentiate (C.3) with respect to time

$$\dot{K}(t) = \dot{k}(t)\theta(t) + k(t)\dot{\theta}(t) \tag{C.7}$$

We need to find $\dot{k}$, $\dot{\theta}$ as a linear function of K, ω.

$$\frac{d}{dt}(vect(R)) = vect(\dot{R}) \tag{C.8}$$

$$\dot{R}R^{-1} = \Omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{C.9}$$

$$vect(\dot{R}) = vect(\Omega R) = \frac{1}{2}X\omega \quad where \quad X = tr(R)I - R \tag{C.10}$$

$$tr(\dot{R}) = Tr(\Omega R) = -2s\theta k^T\omega \tag{C.11}$$

Now (C.6) yields

$$\dot{k} = \frac{(tr(R)I - R)\omega}{2s\theta} - \frac{c\theta k\dot{\theta}}{s\theta} \tag{C.12}$$

Differentiating (C.5) with respect to time results in

$$\dot{\theta} = \frac{tr(\dot{R})}{-2s\theta} \tag{C.13}$$

Substituting (C.11) into (C.13) yields

$$\dot{\theta} = k^T\omega \tag{C.14}$$

Equations (C.12) and (C.14) yields

$$\omega = 2s\theta N^{-1}\dot{k} \tag{C.15}$$

where

$$N = \theta M + 2s\theta kk^T \qquad and \qquad M = tr(R)I - R - 2c\theta kk^T \tag{C.16}$$

Substituting in (C.7) from Equations (C.12) and (C.14) results in

$$2s\theta\dot{K} = M\omega\theta + 2s\theta kk^T\omega = F\omega \tag{C.17}$$

where

$$F = M\theta + 2s\theta kk^T \tag{C.18}$$

Differentiating (C.17) yields

$$2c\theta\dot{K} + 2s\theta\ddot{K} = \dot{F}\omega + F\dot{\omega} \tag{C.19}$$

$$\dot{\omega} = F^{-1}(2c\theta\dot{K} + 2s\theta\ddot{K} - \dot{F}\omega) \tag{C.20}$$

Now, we need to find $\dot{F}$

$$\dot{F} = \dot{M}\theta + M\dot{\theta} + 2c\theta\dot{\theta}kk^T + 2s\theta(\dot{k}k^T + k\dot{k}^T) \tag{C.21}$$

where

$$\dot{M} = -2s\theta k^T\omega I - \Omega R + 2s\theta k^T\omega kk^T - 2c\theta(\dot{k}k^T + k\dot{k}^T) \tag{C.22}$$

The optimized C code for this function, *ddAA2alpha()*, is produced by the symbolic optimization routine provided by the RDM [73] software.

APPENDIX D

HARDWARE AND SOFTWARE CONFIGURATION USED IN COLLISION AVOIDANCE EXPERIMENTS

This appendix summarizes the hardware and software configuration used for the collision avoidance experiments described in Section 3.4.2 . Note at this time REDIESTRO was located at the Center for Intelligence Machines (CIM) at McGill University. It was then relocated at Concordia University to perform the force control experiments.

D.1 HARDWARE CONFIGURATION

A block diagram of the VME bus based controller of REDIESTRO is provided in Figure D.1. The processing power is supplied by a Challenger processor board which contains two Texas Instruments TMS320C30 DSP's rated at 33 MFLOPS with onboard memory, inter-chip communication via shared memory, RS232, RS422, and parallel ports. The Digital to Analog (D/A) converter board sends low-level control signals to the power amplifiers. The Pulse Width Modulated (PWM) amplifiers supply current to the motors. The Parallel Input/Output board handles digital signals from limit switches, brakes, Interface (I/F) circuitry, amplifiers, and control signals.

The robot is actuated by Direct Current (DC) servo motors driving harmonic drive transmissions. Each joint is also equipped with a brake and a joint encoder. The encoder boards receive signals from encoders and incremental/decremental on-chip counters and uses hardware implemented digital filters for noise reduction.

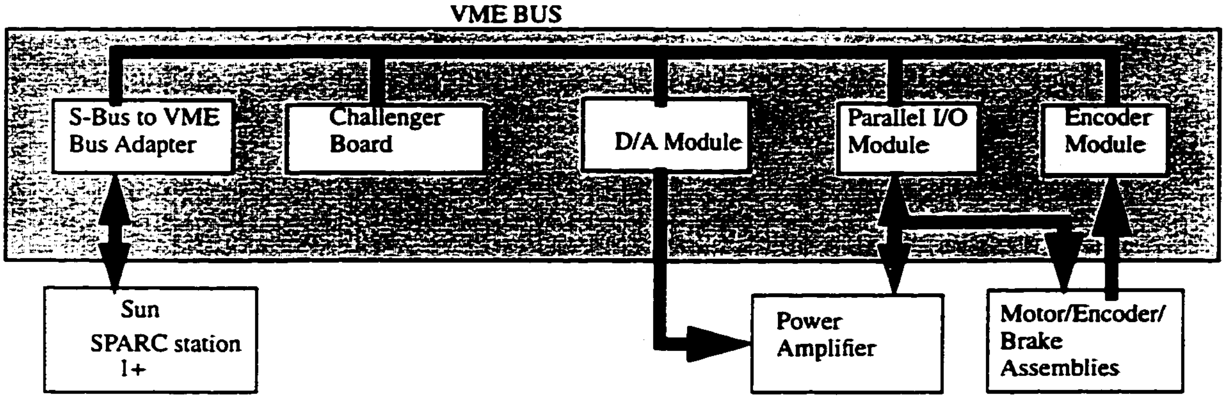


Figure D.1 Block diagram of the VME bus based controller (collision avoidance experiments)

D.2 SOFTWARE CONFIGURATION

The software configuration for REDIESTRO is shown in Figure D.2. Different elements reside either on a Sparc 1+ workstation or the DSP boards in the VME cage.

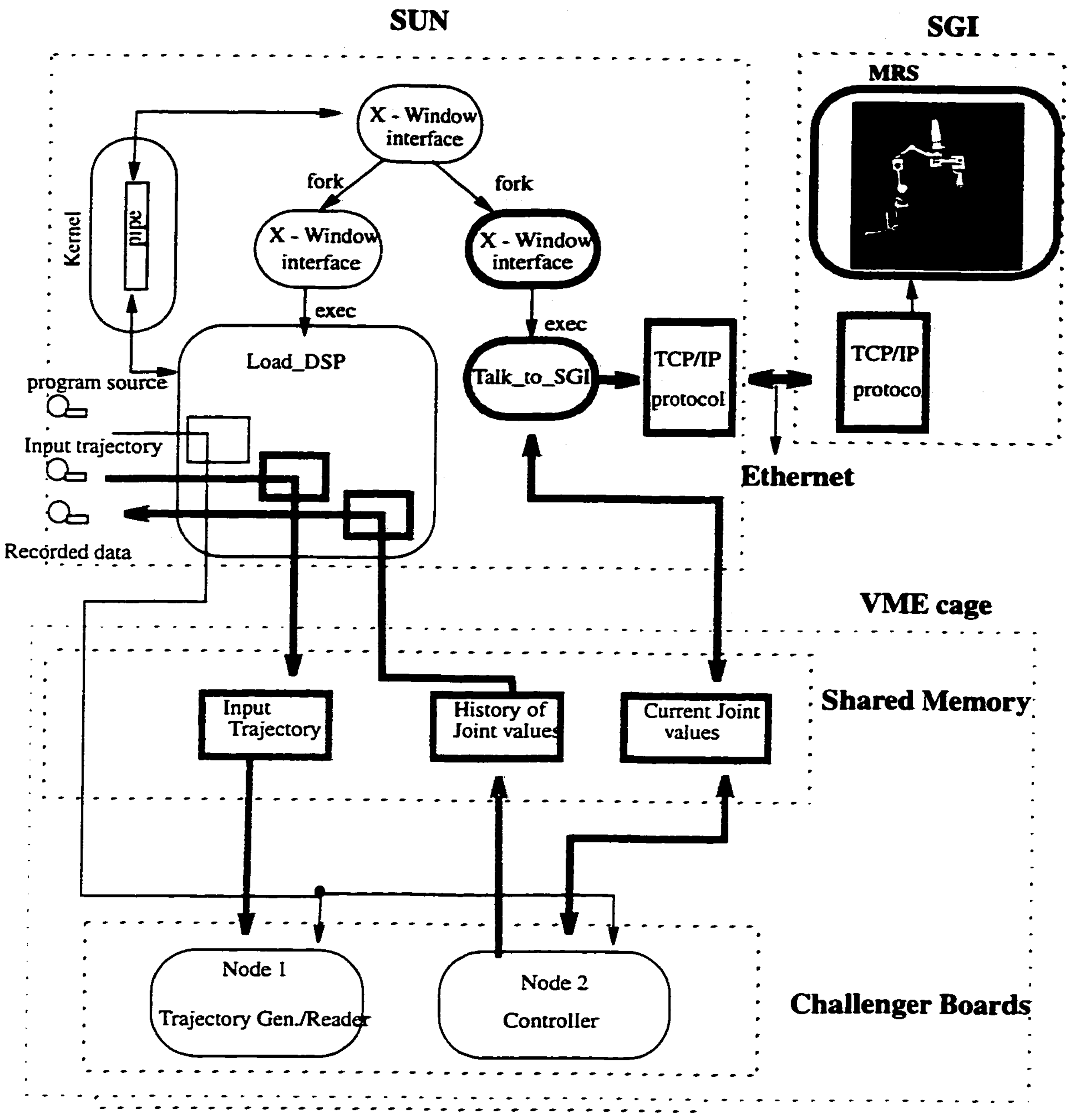


Figure D.2 Software Configuration (collision avoidance experiments)

D.2.1 Processes residing on DSP boards

The processes that actually control the robot and interact with the motors, brakes, and power amplifiers, and encoders reside on the DSP boards. There are two processor boards (referred to as nodes 1, 2) which run the set point generation and low level PID controller respectively. The controller frequency is 200 HZ and needs the joint values (from the encoders) and the joint set points (from node 1) as input. Estimates of the joint rates are calculated by numerical differentiation.

D.2.2 Processes residing on the Sparc 1+

The Sparc 1+ workstation acts as the front-end for running the control software for REDIESTRO. The first process that is invoked by the user, is the X-Windows interface. This is the visual interface between the user and the control process running on the DSP boards. In order to simplify the X-Windows routine and restrict its responsibility to direct interaction with the user, it does not directly establish the connection with the processor boards on the VME cage, instead, another process called Load_DSP is invoked - refer to Figure D.2. The latter establishes connection with the processor cards and the shared memory upon execution and goes into an infinite loop waiting for commands from the X-Windows process. Considering the wide range of requests coming from the user via the X-Windows, a pipe has been selected as the inter-process communication tool between the X-Windows and Load_DSP processes. The first thing that a user is required to do is to download the appropriate process onto the DSP cards., i.e. trajectory generation in node 1, and controller in node 2. Then, the user can give high level commands like move_to_point, move_joint_no, etc.

In the hardware demonstration, the SGI workstation running the MRS program is connected on-line to the SUN workstation to provide computer generated 3-D visualization of the actual motion. The server routine periodically reads the joint angles from the shared memory and uses the Ethernet link and TCP/IP protocol to send the actual joint angles to the MRS program running on the SGI workstation.

APPENDIX E HARDWARE AND SOFTWARE CONFIGURATION USED IN FORCE CONTROL EXPERIMENTS

REDIESTRO was relocated from Center for Intelligence Machines (CIM) at McGill University, where the hardware experiments for collision avoidance were performed, to Concordia University for further development and experimentation for compliant motion and force control. Note that the only control mode supported by the controller at CIM was independent joint PID control (See Appendix D). The hardware and software configuration of the controller has gone through several levels of change to meet the requirements for force and compliant motion control. in this appendix the final hardware and software configurations are described.

E.1 HARDWARE SPECIFICATION AND INTERFACING

The system used in the hardware experiments consists of the following components:

1- REDIESTRO a 7-DOF redundant arm

2- A 6-DOF wrist force/torque sensor (JR3)

3- Controller hardware

4- Environment

Appendix B gives the specification of the arm as well as its hardware accessories. A JR3 6-DOF force/torque sensor is attached to the end-effector of the REDIESTRO. The environment consists of the tool attachment and the setup used for the two strawman tasks which will be described in Sections 6.4.2 and 6.4.3 .

Table E-1 Specification of the hardware components used in the original configuration

Type	model	Manufactur e	Description	No.
Processors	IV 3230	IRONICS	68030 processor card	2
I/O	XVME-240	XYCOM	64 bit Digital Input/Output card (DIO)	1
	XVME-505/2	XYCOM	4 Channel, 12 bit Digital to Analog Converter (DAC)	2
	Whedco	Whedco	32 bit, dual channel Incremental Encoder Interface	4
	VF1	JR3	Force sensor receiver card, includes a DSP processor	1
Bus Adaptor	Bit3-412	Bit3	VME/VME bus adaptor	1
Bus Adaptor	Bit3-607	Bit3	GIO/VME bus adaptor	1

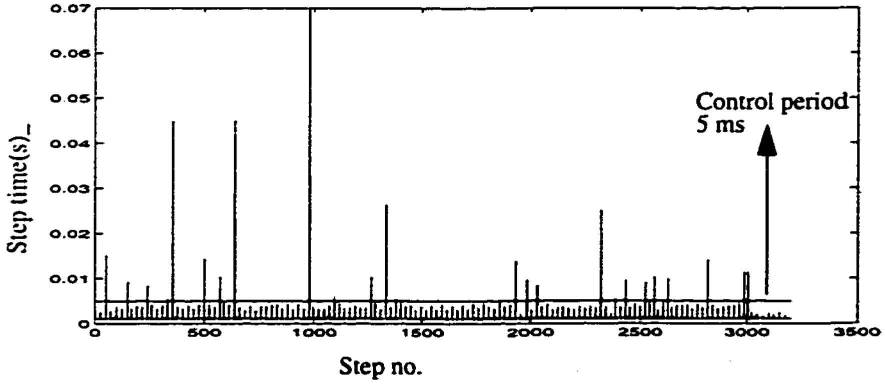
Table E-1 summarizes the specification of different components of the VME-based controller. The VME-based controller in conjunction with the Real-Time Operating System (RTOS) Chimera III [75], was used to implement the PID and the joint level LDPD controller in the initial stages of software development (see Section 5.4.2). This also provided a check on the correct operation of the driver software developed for the IO cards. The operation of the robot using this system also verified the correct connection (wiring) and integration of different modules. However, initial bench marking of the different modules of the AHIC scheme (see Table E-2) indicated that the two 68030 processor boards cannot provide enough computational power to run the AHIC scheme with 200 Hz control frequency.

Table E-2 Time bench marking for different modules of the AHIC scheme on a 68030 processor

Module	No. of flops	Execution time on a 68030 board (ms)
Redundancy resolution (non-optimized)	446 + 7x7 matrix inversion	4.4
Forward kinematics	742	4.2
Inverse dynamics	1050	4.4
AHIC	--	~2.5
Force + other I/O	--	~3.5
Total		~19

This led us to consider the alternative of using the SGI workstation (INDIGO 2) to run the controller and keep the VME-based environment with Chimera III for data acquisition. A more detailed study has indicated that two major problems should be solved in order to use this alternative for real-time control:

- Making the SGI's operating system (IRIX 5.3) perform each controller step within 5ms.
- Fast communication link between the SGI and VME bus (at least 40 kilobytes/s)



Task Time = 16 s
Cont. Freq. = 200 HZ
No. of contr. steps = 3200
Total execut. time = 4.58 S
Max. step time = 409 mS
Min. step time = 1 mS
Average step time = 1.4 mS
Standard Deviation = 7.4e-3

Figure E.1 Bench Marking for the modified AHIC algorithm on the SGI workstation (Indigo 2)

Figure E.1 indicates the first problem. As we can see, the average execution time is 1.4 ms which confirms the result that the algorithm can be run 4 times faster than real-time. However, because of the time-sharing nature of the operating system, individual steps can take up to 70 ms which is enough to cause instability.

Further investigation revealed that the IRIX operating system provides a rich set of real-time programming features that are collectively referred to as the REACT extensions. These features can be used to accurately time events, use signals in interrupt routines, control allocation of real memory to the process and provide for priority scheduling [81]. A real-time system provides bounded and usually fast response to specific external events and thus a programmer can schedule a particular process to run within a specified time limit after the occurrence of an event. While deterministic responses can be achieved on a uniprocessor, the IRIX real-time strategy requires at least two processors for optimal response. Typically, one processor services interrupts and other jobs, while the other services high-priority real-time jobs. Effectively, unpredictable loads such as interrupts are serviced on a processor other than the one running the real-time application. UNIX implements priority aging for processes where a CPU bound process gradually has its priority lowered as it runs. This ensures that lower priority processes are not starved of the CPU. While this is appropriate for most timesharing environments, it is not so for real-time environments. Thus IRIX has introduced the notion of a fixed or non-degrading priority. Thus a programmer can maintain the priority order of a particular set of processes in the system and control as much of the CPU as is appropriate. This even allows us to use an SGI workstation with only one processor for real-time applications.

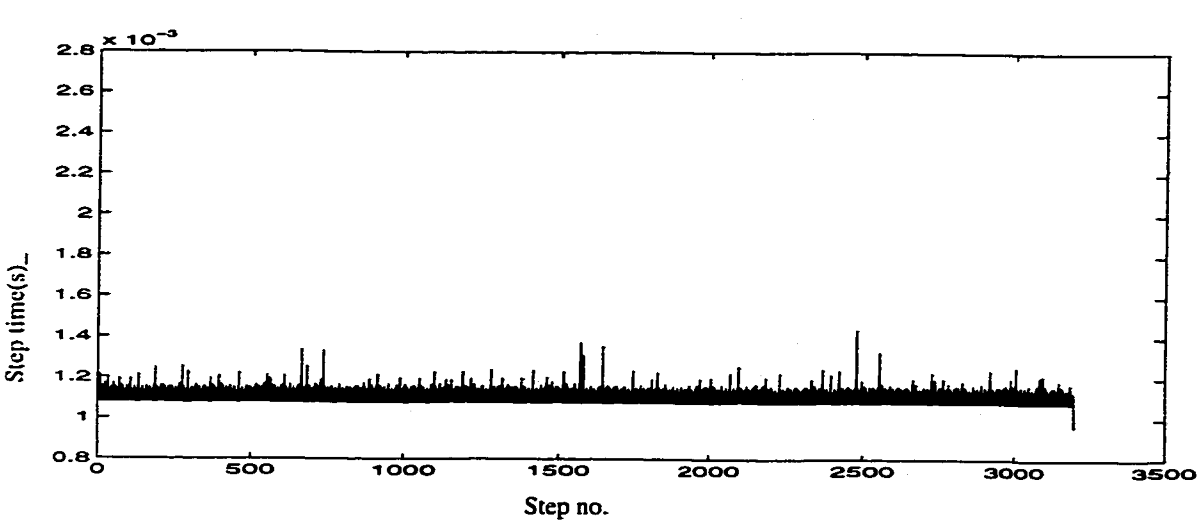


Figure E.2 Bench Marking for the modified AHIC algorithm on the SGI workstation (Indigo 2) using IRIX real-time programming (REACT) utilities.

Figure E.2 shows the result of bench marking of the AHIC algorithm on the SGI workstation when this process is assigned with the highest non-degrading priority (also called "real-time" priority).

The second problem in using this alternative for the real-time controller is to find a fast communication link between the SGI workstation and the VME bus (at least 40 kilobytes/s) which receives the data (joint values, numerically differentiated joint rates, and force sensor readings) from the VME-based data acquisition and sends back the calculated torque command. The BIT3 model 607 adaptor was used for connecting the GIO to the VME bus. The adaptor card on the VME bus is equipped with a 128 kilobyte dual-port RAM.

Figure E.3 shows the final hardware configuration used for the hardware demonstration.

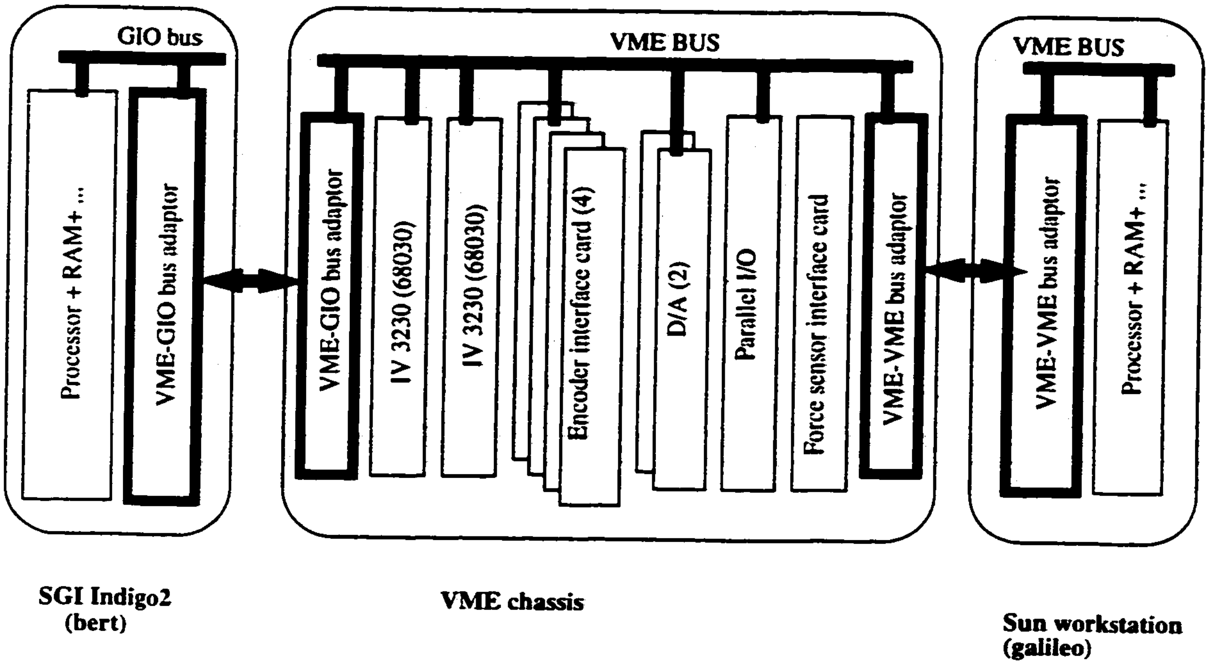


Figure E.3 Final hardware configuration

E.2 SOFTWARE PREPARATION AND INTERFACING

The software configuration of the real-time controller is shown in Figure E.4. Based on the platform and functionality, they can be categorized into the following groups:

- Controller software: Running on the SGI workstation.
- Process communication software: Communication between processes residing on the SGI and processor cards on the VME bus.
- Data acquisition and I/O control software: Running on the VME-based processors.

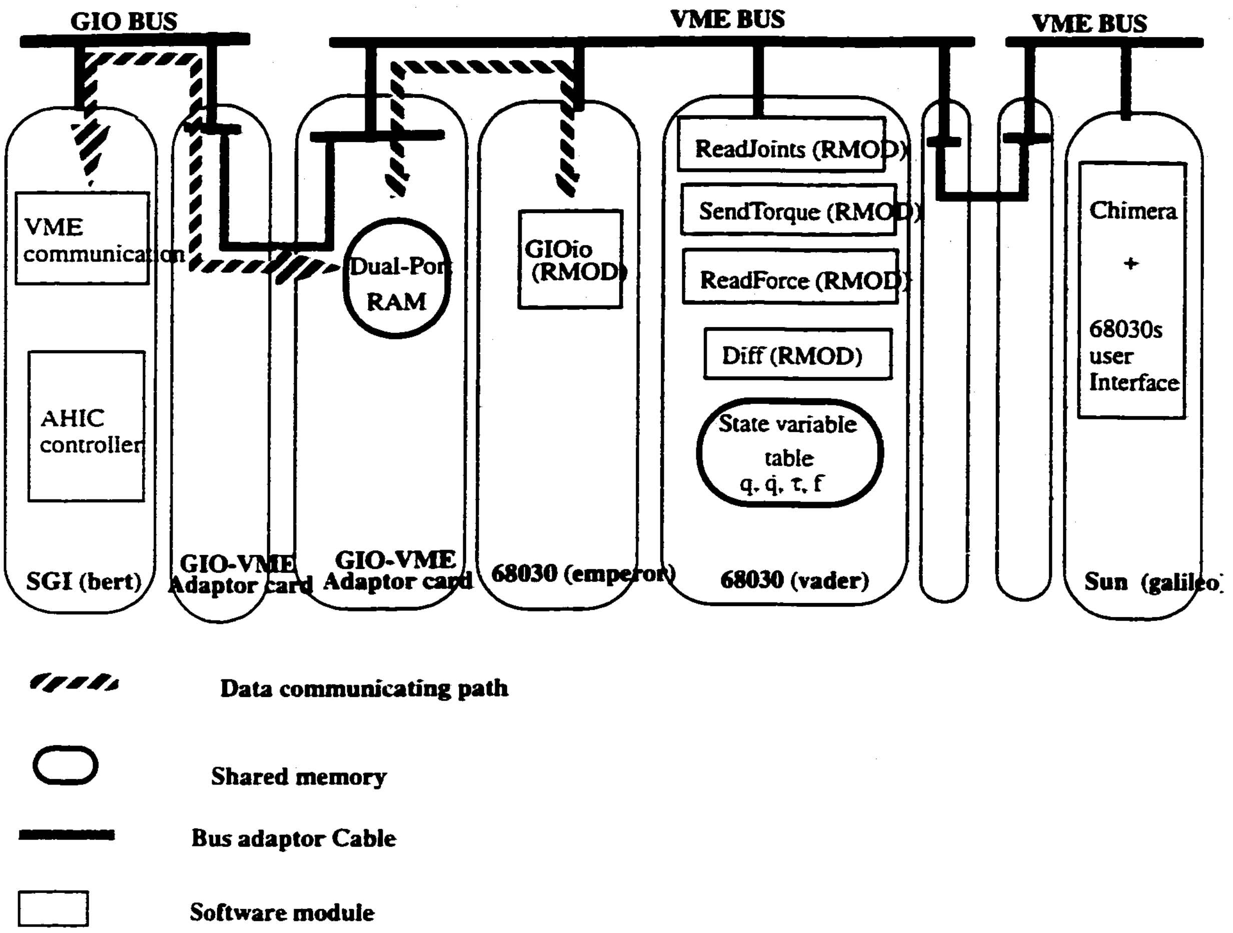


Figure E.4 Software configuration block diagram of the real-time controller

E.2.1 Controller Software

The Controller software runs on an SGI Indigo2 workstation (see Figure E.5). It should be noted that the modified AHIC is not a static function which calculates its output only based on the inputs. The presence of the integrator (see Figure 5.22) in this scheme implies that the controller must have knowledge of the current time (and also the elapsed time since the previous step).

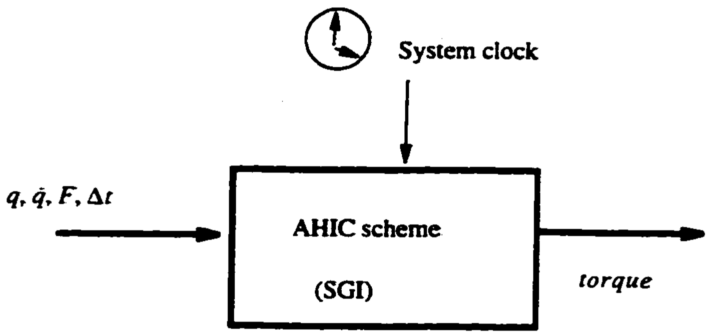


Figure E.5 Black box representation of the AHIC residing on the SGI

The real-time controller software that is used to control REDIESTRO, also includes a communication module with the VME-based processors used to run data acquisition software. The general block diagram of the real-time control software is shown in Figure E.6. The block *gettime()* should be replaced by the time utilities described in the next section. The modules *ReceiveSensorData* and *SendtroqueCommand* will be described in the next section. It should be noted that in order to achieve maximum control frequency, no attempt has been made to execute the control loop with a fixed period. This means that the main loop is performed according to the maximum availability of the cpu. Therefore, the control period will be different at each step. This does not create any problems for the controller as long as each control period is measured accurately and the dynamic part of the controller, e.g. the integrator, takes this into account.

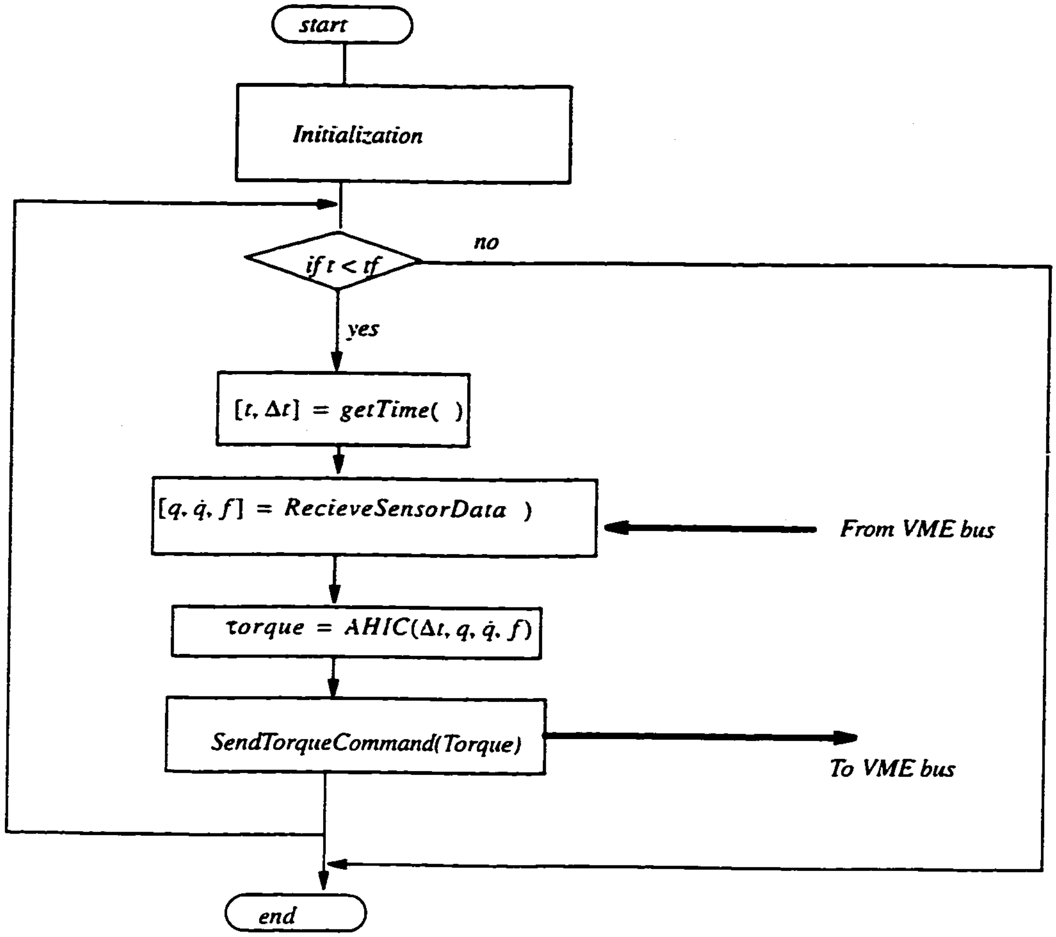

start
Initialization
no
if t < tf
yes
[t, Δt] = getTime( )
[q, q̇, f] = RecieveSensorData)
From VME bus
torque = AHIC(Δt, q, q̇, f)
SendTorqueCommand(Torque)
To VME bus
end

Figure E.6 Simplified block diagram of the real-time control software running on the SGI workstation

E.2.1.1 Using the SGI Workstation as the Real-Time Controller

This section describes the use of the features included in the IRIX operating system (on the SGI workstations) with REACT that are useful in measuring time. As mentioned earlier, the integration routines inside the controller require the measurement of the elapsed time from the previous control step. By getting the time at each control step, the controller can calculate the elapsed time. The IRIX operating system with REACT supports two ways of accessing time information:

- The Unix-compatible function *gettimeofday()*
- Direct user code access to the free-running hardware timer.

The resolution of the time returned by *gettimeofday()* is at least 10 milliseconds. Therefore, it is not recommended for applications with a very fast control loop. The free-running hardware timer provides the highest resolution and accuracy.

The SGI processor has a free-running timer which consists of a 32-bit counter clocked continuously by the system bus-clock. The SGI_CYCLECNTR of *syssgi()* command returns the address of this counter and the period of its clock signal. On the Indigo2 workstation, which is being used for the controller, the clock period is 40 ns. After querying for the counter's address, a user can subsequently use *mmap()* to map the counter into the virtual address space. The process can subsequently read this counter directly without the overhead of a system call. The time required to read this counter is approximately 100ns.

Also, note that because this is a 32 bit counter, it is reset after 2^{32} clock pulses. For the Indigo2 workstation, the counter resets every 171.8 s $(2^{32} \times 40e-9)$. Therefore, the counter resets should be detected by the timer program.

A *timeCounter* structure is defined which is used to store the required information to evaluate the time. This structure is shown below:

```
typedef struct timeCounter {
                unsigned                phys_addr;
                unsigned                *iotimer_addr;
                unsigned                cycleval;
                unsigned long            prevTime;
                unsigned long            elapsedTime;
                unsigned long            curTime;
                unsigned long           startTime;
                unsigned long            endTime;
                int                     reset_no;
                .
                .
} timeCounter;
```

where *phys_addr* is the physical address of the counter; *iotimer_addr* is a pointer for the address of the counter memory mapped into the process virtual address space; *cycleeval* is the resolution of the clock used to update the counter; *prevTime* is the value of the counter at the previous call; *curTime* is the current value of the counter; *startTime* is the counter value when the timer is initialized; and *endTime* is the counter value when the timer is closed.

Three functions *initTimeCounter()*, *incrTimeCounter()*, and *freeTimeCounter()* have been developed. The flowchart of the first two functions are shown in Figure E.7 and Figure E.8.

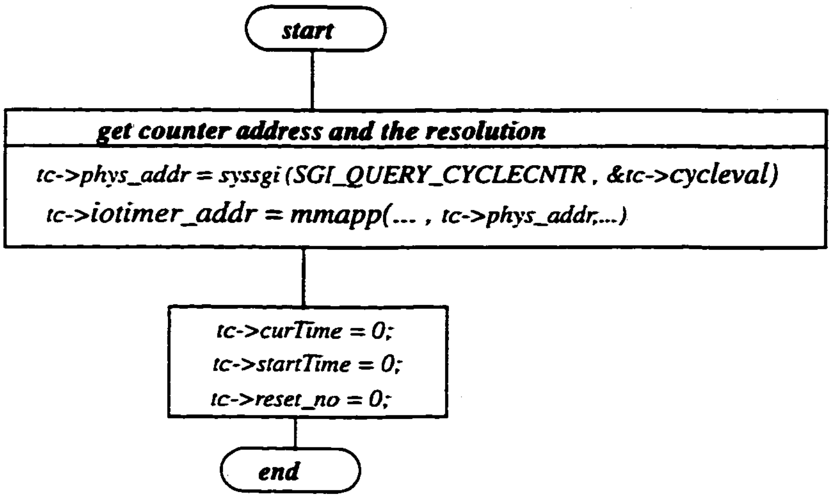

start
get counter address and the resolution
tc->phys_addr = syssgi (SGI_QUERY_CYCLECNTR , &tc->cycleval)
tc->iotimer_addr = mmapp(... , tc->phys_addr,...)
tc->curTime = 0;
tc->startTime = 0;
tc->reset_no = 0;
end

Figure E.7 Simplified flow-chart of the function *initTimeCounter(timeCounter *tc, ...)*

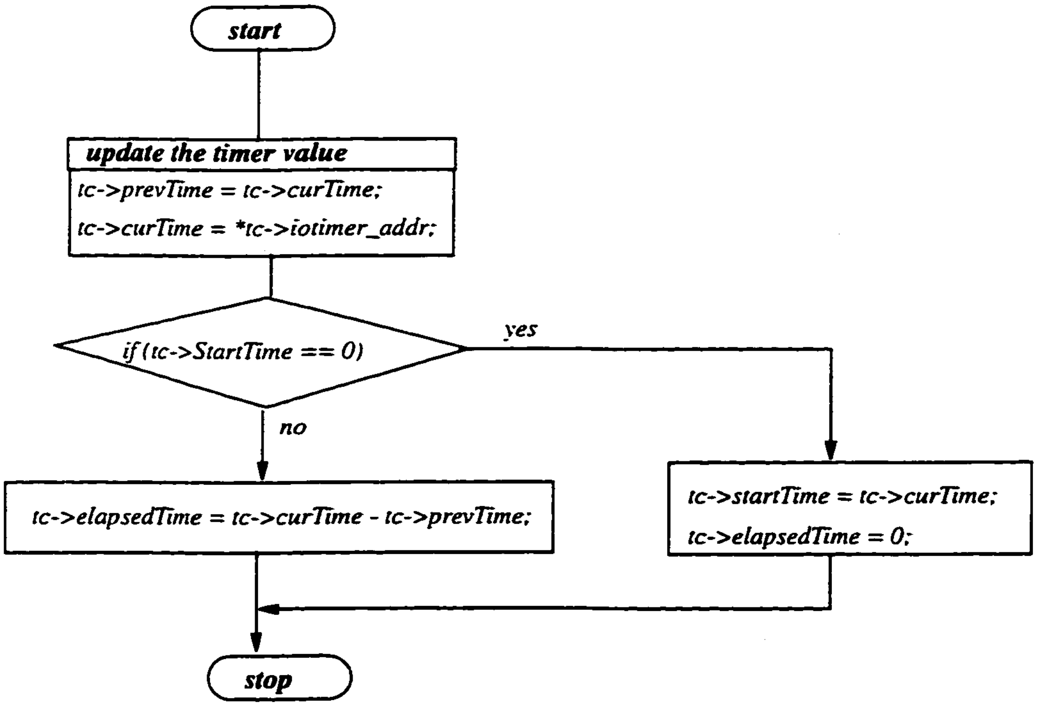

start
update the timer value
tc->prevTime = tc->curTime;
tc->curTime = *tc->iotimer_addr;
if (tc->StartTime == 0)
yes
no
tc->elapsedTime = tc->curTime - tc->prevTime;
tc->startTime = tc->curTime;
tc->elapsedTime = 0;
stop

Figure E.8 Simplified flow-chart of the function *incrTimeCounter(timeCounter *tc)*

Implementation and test of the real-time controller on the SGI

In order to implement and test the operation of the real-time version of the controller, the time evaluation functions are incorporated into a real-time version of the controller) and the integration is tested by a kinematic simulation (see Figure E.9).

Note that in Figure E.9, the time difference with respect to the first call to the *incrTimeCounter()* is calculated in terms of counts;

$$deltaTime = tc \rightarrow curTime - tc \rightarrow startTime \tag{E.1}$$

where all variables are declared as "unsigned long". The terms *tc->curTime* and *tc->startTime* are the current and initial number of the counts respectively. As mentioned earlier, the SGI workstation uses a 32 bit counter. Therefore, one would expect that when the counter is reset after counting to 2^{32}, *deltaTime* would not express the correct time difference. However, because the left hand-side of (E.1) is calculated using the 2's complement notation, *deltaTime* expresses the correct count difference even though the counter is reset. The problem occurs not when the counter is reset, but, after a counter reset, where the count number comes back to its initial value. In this situation, *deltaTime* would show a zero count difference. In other words, *deltaTime* would be seen as a shifted version of the free-running timer (shifted by *tc->startTime*) which resets to zero after 2^{32} counts to zero (see Figure E.10). Therefore, one should detect the decreasing edge on *deltaTime* and keep track of the number of resets in order to calculate the absolute elapsed time from the initial reference time.

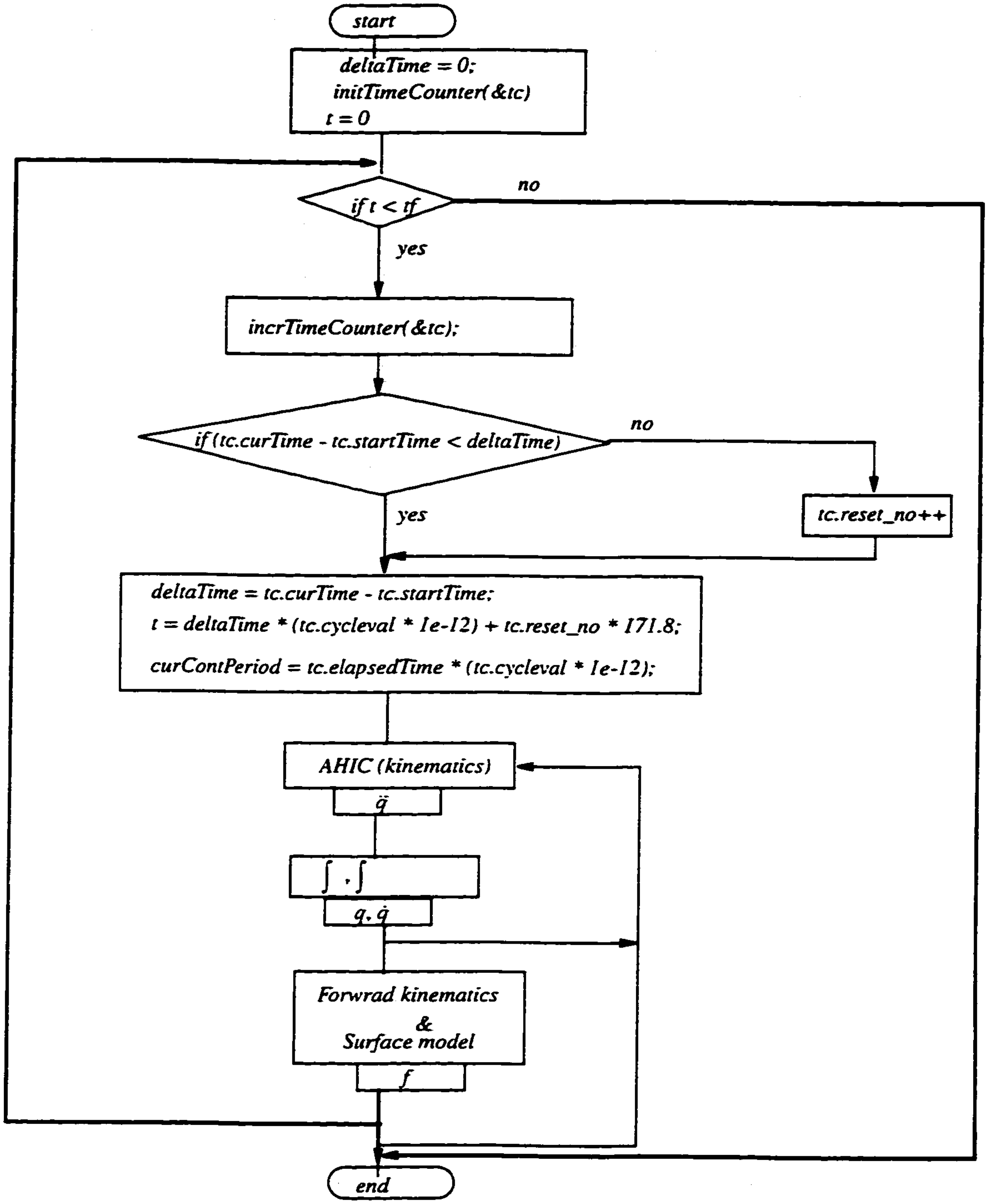

start
deltaTime = 0;
initTimeCounter(&tc)
t = 0
no
if t < tf
yes
incrTimeCounter(&tc);
if (tc.curTime - tc.startTime < deltaTime)
no
tc.reset_no++
yes
deltaTime = tc.curTime - tc.startTime;
t = deltaTime * (tc.cycleval * 1e-12) + tc.reset_no * 171.8;
curContPeriod = tc.elapsedTime * (tc.cycleval * 1e-12);
AHIC (kinematics)
$\ddot{q}$
$\int , \int$
$q, \dot{q}$
Forwrad kinematics
&
Surface model
f
end

Figure E.9 Flow-chart of the real-time kinematic simulation of the controller software on the SGI workstation

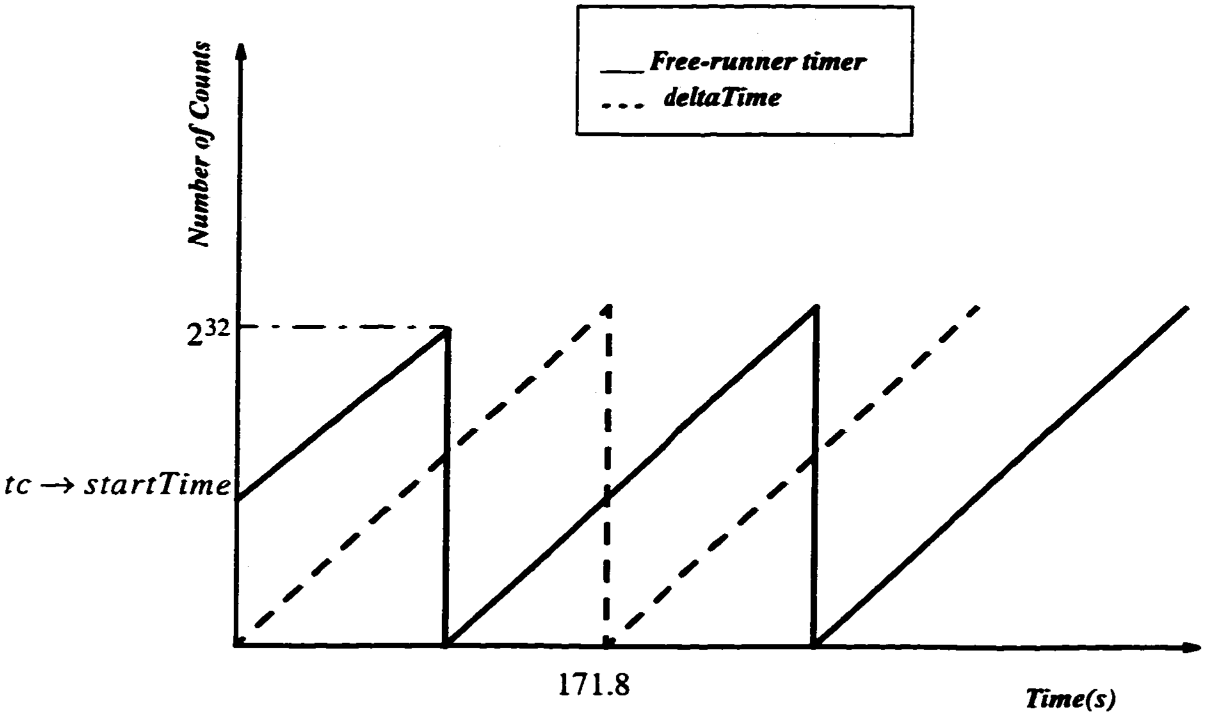


Figure E.10 Effect of free-runner timer reset in calculating the elapsed time.

E.2.2 Process Communication Software

Figure E.3 shows the final hardware configuration of the real-time controller. It was noted that the control software is run on the SGI workstation, while the data acquisition and I/O control are handled by the 68030 processors on the VME chassis. A BIT3 bus adaptor which connects the IRIX workstation's GIO bus to the VME bus is used for communication between the processes on the SGI workstation and the VME processes (Figure E.11). The adaptor card on the VME bus is equipped with 128 kilobytes of dual-port RAM. In this section the communication software developed for this purpose will be described.

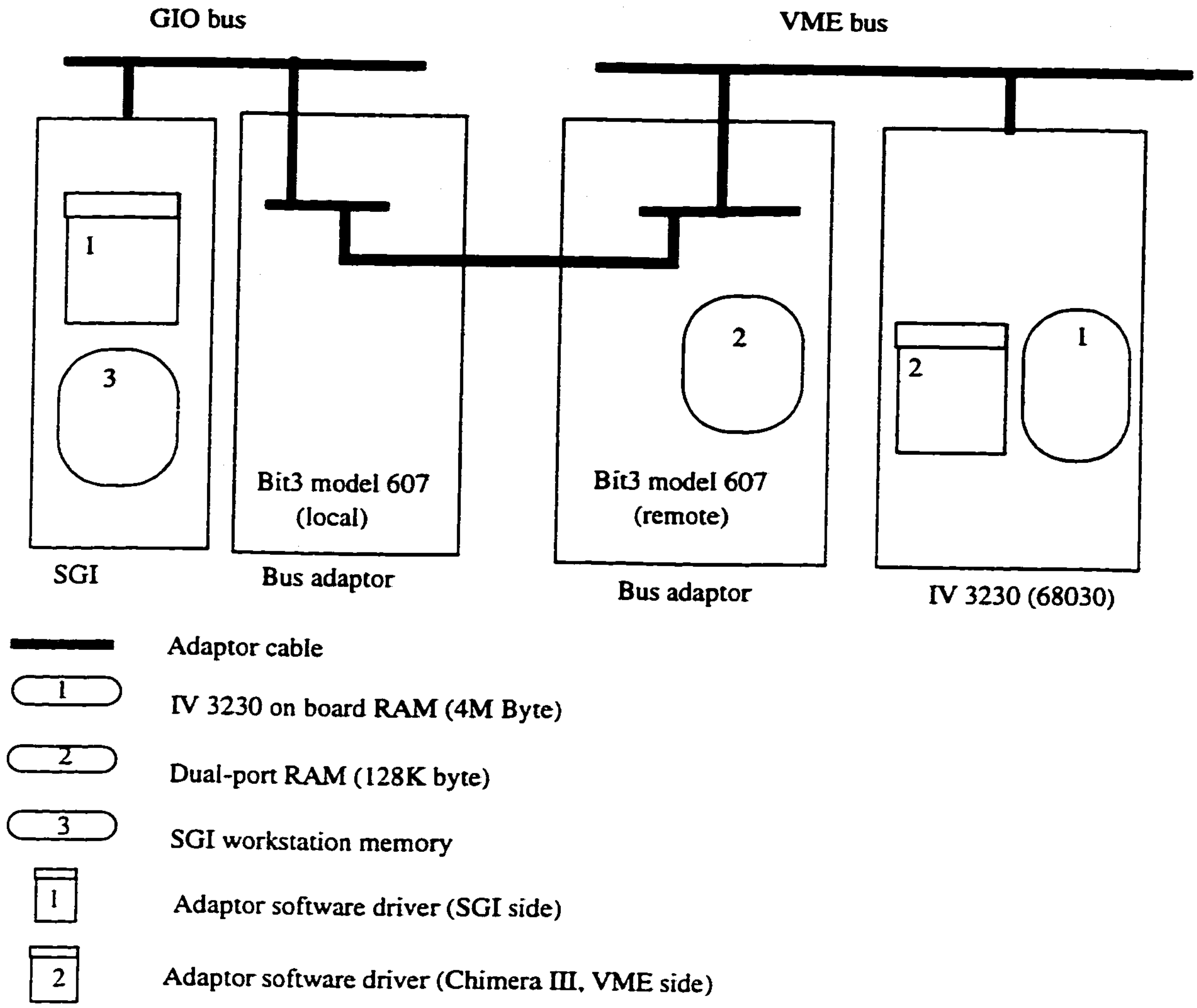


Figure E.11 Block diagram of the hardware and software modules involved in the GIO-VME bus communication.

E.2.2.1 BIT3 Software Support for the GIO-VME Bus Adaptor

BIT 3 support software (Model 963) for the GIO bus provides a configurable device driver for IRIX and provides support routines to access all adaptor resources. Therefore, the VME bus memory, dual-port RAM, and GIO local system memory can be shared by two systems. The three major mechanisms to transfer (or share) data on the two systems are as follows

- Direct Memory Access (DMA)

- Programmed I/O (PIO)
- Memory mapping

Direct Memory Access, DMA, is the automatic transfer of data from one memory location to another without intervention from a processor once the transfer is started.

The PIO access from GIO to VME bus uses window-mapping in combination with the system calls *read()*, *write()*, and *lseek()*. Each of the seven window mapping registers on the adaptor card controls access to 256K bytes of VME bus address space.

The memory mapping mechanism uses a combination of hardware and system-level software to translate the virtual addresses used by the application to the actual physical address. If the page of the virtual memory is already loaded into physical memory, the virtual to physical address translation is completely handled by address translation hardware. An mmap() call to a device driver uses the same virtual memory mechanism to create a section of application address space that accesses a region of the memory corresponding to the VME (or dual-port RAM) physical address space.

The software support also provides interrupt handling routines which can be used as an efficient way of synchronization between different processes.

In the remainder of this section, we will skip the detailed description of the hardware and software mechanisms used by the adaptor to perform the communication; instead, we will focus on the functional description of the data transfer, the message passing and synchronization scheme developed as part of the integrated real-time controller used in the hardware demonstration. A detailed description of the low-level hardware and software operations can be found in the BIT3 Model 607 Adaptor Hardware Manual and also in the Model 963 Support Software Manual.

E.2.2.2 GIO-VME Bus Adaptor Driver

The file *gioio.c* is a C file developed to implement higher-level functions to ease the use of the low-level functions provided by BIT3 software support. It includes the following functions:

- ***initGIO():*** Performs initialization of the adaptor hardware
- ***sendVals(int n, double *val):*** Writes *n* doubles (stored from address *val*) into a specified location of the memory on the VME bus.
- ***recvVals(int n, double *val):*** Reads *n* doubles from a specified location of the memory on the VME bus.
- ***sendCommand(long command, double *args):*** Sends a command along its arguments to processes located on the VME system
- ***closeGIO():*** Performs the necessary action before closing the device driver

As we mentioned earlier, the adaptor provides different hardware mechanisms for data transfer between the two systems. In the above functions, the hardware transfer mode can be selected by the user using *#define* directive in the header file as summarized in Table E-3.

Table E-3 Selection of data transfer mode for the GIO-VME adaptor

#define	Mode	Note
USR_DMA	DMA	
RWMETHOD	PIO	
MMAPLOCALMETHOD	Memory Mapping	Physical memory: GIO virtual Memory: VME
MMAPREMOTEMETHOD*	Memory Mapping	Physical memory: VME virtual Memory: GIO
*Selected method in final implementation		

In order to select the best mode of data transfer for our application, several tests were performed. The results indicate that DMA is not suitable. DMA logic is usually employed when large amounts of data need to be moved. However, in our application we need to transfer only a few variables within a high-frequency control loop.

In order to get the maximum cpu time on the IRIX workstation, the controller process uses the highest priority which prevents other processes from taking up cpu time. However, REACT suggests that in order to get the highest deterministic behavior, the real-time process should not itself include system calls that are handled by the operating system kernel. This is the main reason for not using the second mode of data transfer (PIO) which requires using system calls, e.g. *read()* and *write()*. Memory mapping was found to be the most efficient mode for our application, because it has almost no system overhead.

Table E-4 Comparison of the performance of different data transfer methods

Method	Size of Data	Average communication time (ms)	Percentage of steps over 5ms
DMA	512 Byte	3.5 ms	3
PIO	200 Byte	3	0.8
Memory Mapping	200 Byte	1.1	0

Also note that, the data transfer function can be used to read from and write to any memory space located on the VME bus. The shared memory segment can be either on the 68030 processor card or the dual-port RAM located on the adaptor card connected to the VME bus (see Figure E.11). In our case, the shared memory is selected to be the dual-port RAM for the following reasons:

- The read and write operation from the IRIX system can be accomplished independently of the traffic on the VME bus.
- Both systems can access the dual-port RAM at the same time with the adaptor arbitrating simultaneous accesses.

Now let us consider the timing and synchronization issues in implementing the data transfer functions. In order to prevent data corruption, simultaneous read and write, e.g. GIO read /VME write and GIO write/VME read, should be prevented. On the other hand, in our application, one system should be able to request a service from the other system. This requires implementation of a simple communication protocol.

E.2.2.3 Communication Protocol Between IRIX and VME Systems

In the implementation of the real-time controller, the controller software residing on the SGI acts as the master task. Figure E.6 shows a simplified block-diagram and the data communication in each control loop. At the beginning of each control period, the IRIX control process needs to read the sensory data gathered by the data acquisition software on the VME system. It performs the control step and returns the commanded torque to the VME system which performs the I/O control. This indicates that the communication protocol should support the following features:

- Prevent data corruption
- Let the master task (IRIX) notify the VME system of its service request
- The VME system should notify the master task upon termination of servicing of the request

The configuration for the shared memory between the two systems is shown in Figure E.12. The data structure is defined by:

```
typedef stuct gioio_t {
            long                    reqType;
            double                  data[20];
}
```

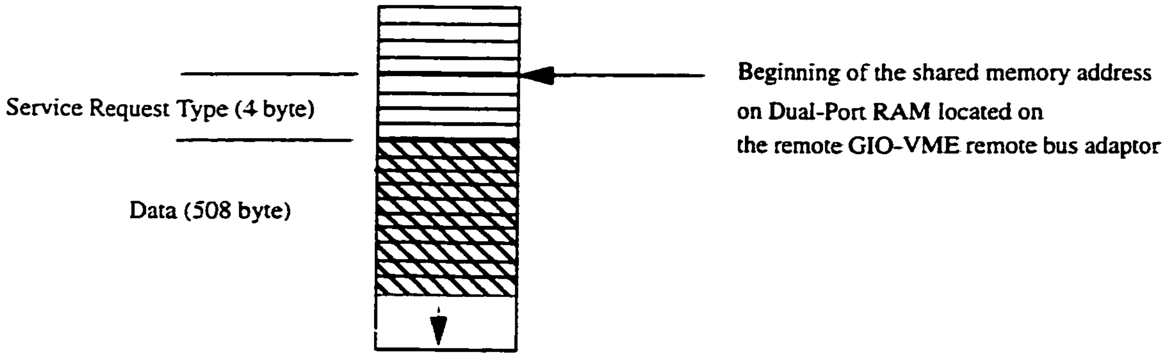

Service Request Type (4 byte)
Data (508 byte)
Beginning of the shared memory address
on Dual-Port RAM located on
the remote GIO-VME remote bus adaptor

Figure E.12 Configuration of the shared memory segment of the dual-port RAM

Table E-5 summarizes the different request types supported by the communication protocol.

Table E-5 GIO/VME communication protocol - supported service request types

Service Request Type (IRIX)	Macro	Numerical Value	Action (VME)
Requesting to send a command torque to the robot	RW_WRITE	0xFFFFFFFF	Reading the torque command
Requesting to read sensory data	RW_READ	0x00000000	Writing the sensory data into the shared memory
Requesting to release the brakes and activate the I/O control to send the command torque to the robot	START_CONTROL	0x00000001	Sending a start signal to the VME bus controller task
Requesting to engage the brakes and stop the I/O control to send the command torque to the robot	STOP_CONTROL	0x00000010	Send a stop signal to the VME bus master task
Requesting the force receiver to reset the offsets at the current configuration	RESET_FORCE_SENSOR	0x00000100	Send a signal to the VME bus master task

Now let us consider the operation sequence in each of the following functions; *recvVals()*, *sendVals()*, *send Command()*. In the following Flowcharts, we assume that the *gioInit()* function has memory mapped the shared memory segment into the variable *shared_mem* address. The *shared_mem* is a pointer to the *gioio_t* structure.

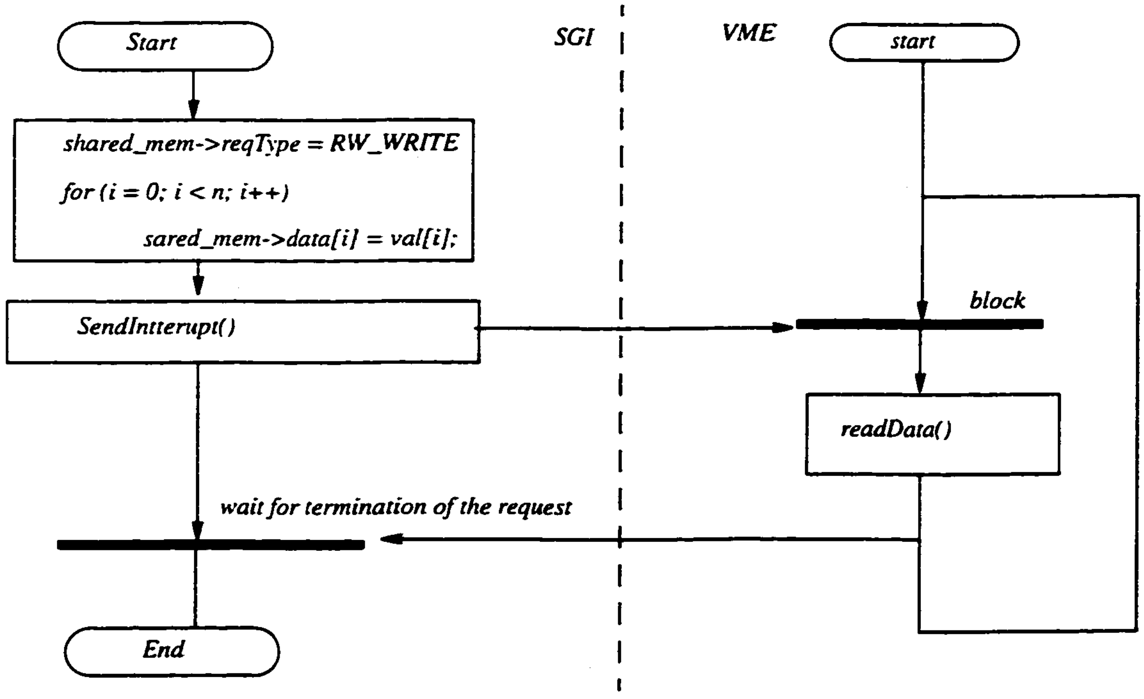

SGI
VME
Start
start
shared_mem->reqType = RW_WRITE
for (i = 0; i < n; i++)
sared_mem->data[i] = val[i];
SendInttertupt()
block
readData()
wait for termination of the request
End

Figure E.13 Block diagram of the function *sendVals(n,data)*

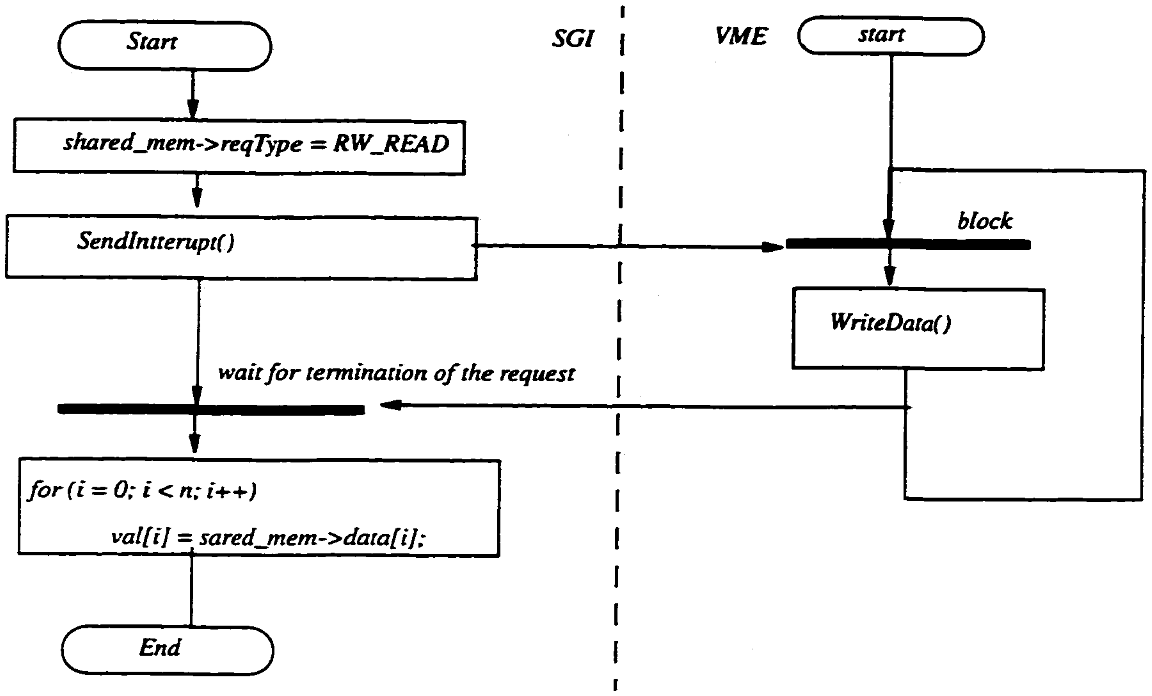

Start
SGI
VME
start
shared_mem->reqType = RW_READ
SendIntterupt()
block
WriteData()
wait for termination of the request
for (i = 0; i < n; i++)
val[i] = sared_mem->data[i];
End

Figure E.14 Block diagram of the function *recVals(n,data)*

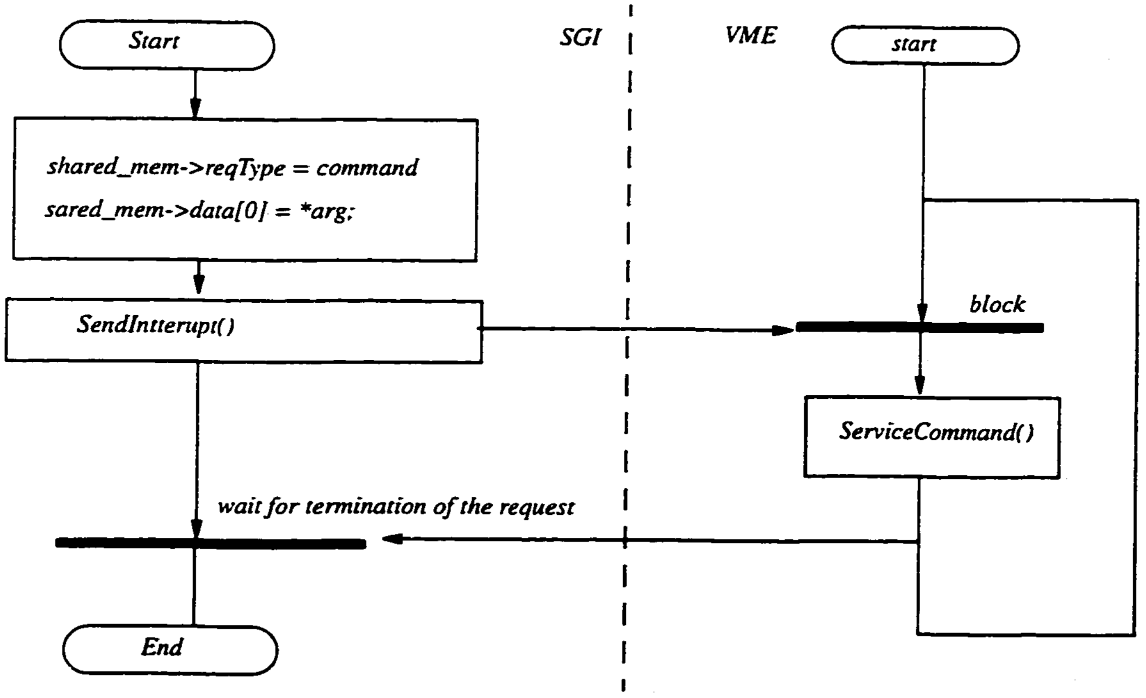

Start
SGI
VME
start
shared_mem->reqType = command
sared_mem->data[0] = *arg;
SendIntterupt()
block
ServiceCommand()
wait for termination of the request
End

Figure E.15 Block diagram of the function *sendCommand(command,arg)*

In Figure E.12 through Figure E.15, the main emphasis is on the modules on the SGI side. Therefore, the VME modules are shown as pseudo functions. A description of the communication module on the VME side is given in next section.

E.2.3 Data Acquisition And I/O Control Software

The VME-based processors essentially act as "intermediaries" between the controller process running on the SGI workstation and the I/O cards connected to the robot (see Figure E.16).

In this section, first a general overview of the Chimera III RTOS is presented. The two major features of Chimera III which are used in the implementation of the VME-based data acquisition software, the I/O interface and the subsystem support (SBS), are described in more detail. The next part gives the software description of the subsystem implemented to perform data acquisition and communication between the IRIX and VME processes.

E.2.3.1 An Overview of the Chimera III Real-Time Operating System

Chimera III was developed in the Robotics Institute of Carnegie Mellon University. It provides a convenient programming environment for real-time applications. The main features that are available in Chimera III are as follows:

- **Task control primitives**: These primitives include creating a task; terminating a task; accessing the physical time; task timing; and assigning task priorities.
- **Local communication and synchronization**: These primitives include support for shared memory and semaphores for the processes running on the same Real-Time Processing Unit (RTPU).

- **Interprocessor communication and synchronization:** These primitives include support for interprocessor shared memory, remote semaphores, interprocess message passing, one-to-one triple buffer communication, global state variable table, extended file system, and ethernet interface passing.
- **Generic interface**: This provides implementation of the reconfigurable I/O device drivers
- **Reconfigurable subsytems**: These provide a modular approach for designing real-time systems.
- **Libraries:** These consist of the *C* libraries generally available for programming in a UNIX environment

As one may notice, most of the features provided by Chimera III are not needed for the downgraded role of the 68030 processors in the final hardware configuration. However, Chimera III gives us the advantage of using a modular software design with less programming overhead which would otherwise be too time consuming. The major primitives that are used for data-acquisition and I/O control are the generic I/O interface and reconfigurable subsytems which will be explained in more detail.

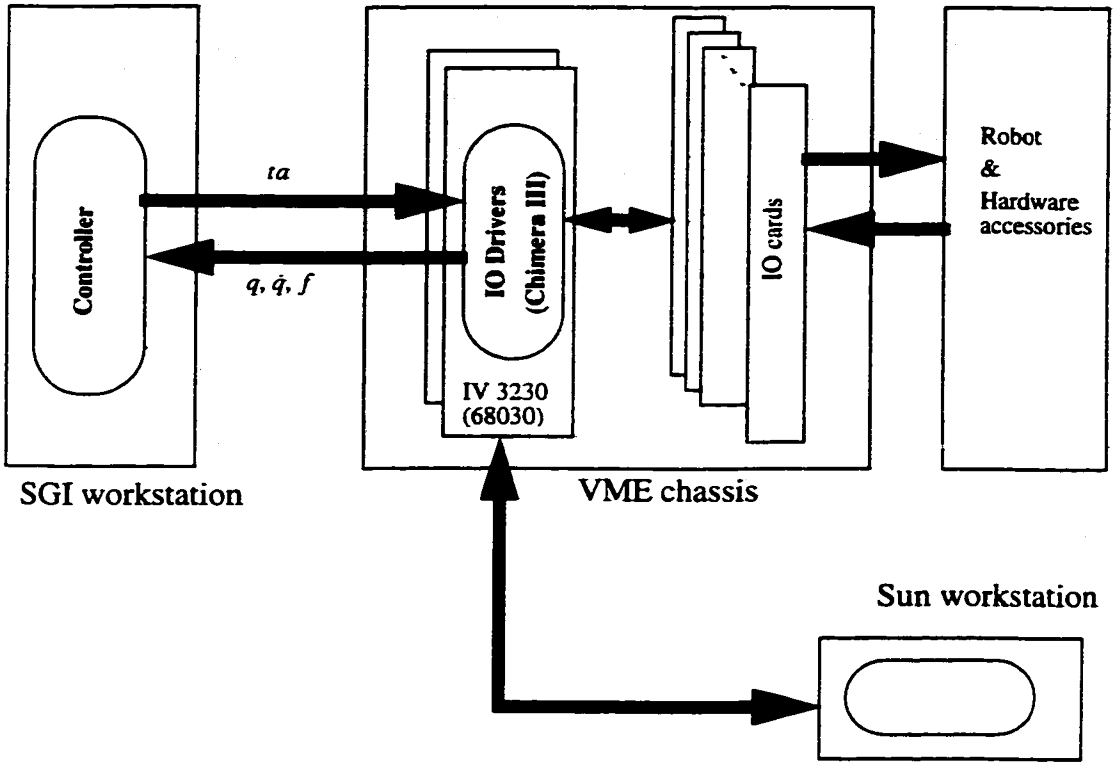


Figure E.16 Data Flow between different modules of the real-time controller

E.2.3.2 Generic I/O Interface

One of the fundamental concepts of the modular software design is that the modules are independent of the target hardware. Chimera III implements an IOD interface which acts as a level of abstraction between the user and the I/O hardware. The following routines provided by Chimera III can be used to access all I/O cards.

```
IOD *iodOpen ( iodname, ports, flags)
        char            *iodname;
        unsigned        ports;
        unsigned        flags;
```

The call to iodOpen() requests that the I/O device named *iodname* be opened. This device should be already defined for Chimera III and have a software driver. The *ports* argument specifies which ports of the device are to be opened at this time. Various options can be selected through the *iodOpen()* flags argument. First the direction of the port should be specified using one the following flags; *IOD_READ, IOD_WRITE,* and *IOD_RW.*

Whenever the data is read from or written to a port, it is passed between the user's program and driver through a buffer. The size of each element in that buffer can be either byte (1 byte), word (2 bytes) or long (4 bytes). The size of each element is specified through one of the following flags; *IOD_BYTE, IOD_WORD,* and *IOD_LONG.*

```
void iodClose(iod)
        IOD                 *iod;
```

When the task finishes with its port, it should do an *iodClose()* which frees up the ports for use by another task. The argument *iod* is the IOD pointer returned by *iodOpen().*

I/O ports that are opened for reading or writing can be read using the *iodRead()* command or written using *iodWrite()* command, which have the following syntax:

```
int iodRead ( iod, buffer, ports)
        IOD                 *iod;
        pointer             *buffer;
        unsigned            port;


int iodWrite ( iod, buffer, ports)
        IOD                 *iod;
        pointer             *buffer;
        unsigned            port;
```

Most I/O devices have various options which can be modified dynamically. To change these options, the iodControl() routine is used. It has the following syntax:

```
void iodControl ( iod, command, arg)
            IOD                 *iod;
            unsigned            command;
            pointer             arg;
```

- Device Drivers Prototype

Let us now consider a device driver prototype. For each driver, two structures should be defined;

```
typedef struct _xxxlocal_t {
            unsigned                        ports;
            unsigned                        flags;
            int                             elsize;
} xxxLocal_t;

typedef struct _xxxglobal_t {
            sysIODEntry         *sysi;
            xxxReg_t            *reg;
            int                 mutex;
} xxxGlobal_t;
```

where xxx refers to the name of the device. The *xxxReg_t* structure is a memory map of the register of the I/O device. It is important that this structure be defined precisely for the I/O device.

Each driver has an internal state, which includes a global and a local component. The global component is the state information shared by each instance of the driver, while the local component is unique for each instance of the driver. Each time that a task calls *iodOpen()* on a specific device, we consider that a new instance of the driver has been created.

The local and global state of the driver are stored in structures of the type *xxxLocal_t* and *xxxGlobal_t* respectively. These structures are completely definable by the programmer writing the device drivers.

The driver should include the following functions:

```
int xxxProbe(sysIODEntry *sysi);
```

where xxx refers to the name of the device. This function is called through Chimera's IOD interface if the device has not previously been opened by *iodOpen()*. First, it checks if the device memory/register is present at the location expected. Then, it allocates the memory required by the global structure of the device. It also performs all the necessary software initialization required by the device.

*int xxxOpen(xxxGlobal_t *xg, unsigned ports, unsigned flags);*

This function is called each time that iodOpen() is called. It allocates and initializes the local structure for the current instance of the device.

*int xxxRead(xxxGlobal_t *xg, xxxLocal_t *xl, pointer buffer, unsigned ports);*
*int xxxWrite(xxxGlobal_t *xg, xxxLocal_t *xl, pointer buffer, unsigned ports);*

These functions are called when a task calls *iodRead()* and *iodWrite()*. The port argument specifies which port should be read from or written to. The driver is responsible for ensuring that the proper port(s) are accessed and placed in subsequent memory locations in the *buffer*.

*int xxxStatus(xxxGlobal_t *xg, xxxLocal_t *xl, unsigned cmd, unsigned *arg);*
*int xxxControl(xxxGlobal_t *xg, xxxLocal_t *xl, unsigned cmd, unsigned *arg);*

These functions are called when a task calls *iodStatus() or iodControl()*. The function *xxxStatus()* evaluates the desired status of the device specified by *cmd*, and returns that by *arg*. The *xxxControl()* performs the specified command on the device.

*int xxxClose(xxxGlobal_t *xg, xxxLocal_t *xl);*
*int xxxCleanup(xxxGlobal_t *xg);*

The *xxxClose()* function is called whenever a task calls iodClose(). It frees up the local structure used by that instance of the device. The function *xxxCleanup()* is called when the last instance of the device is closed and it frees up the global structure of the device.

E.2.3.3 Subsystem Interface (SBS)

Chimera III has been designed to support reconfigurable software design. This section describes Chimera's support for reconfigurable systems, including the module interface specifications, the subsystem (SBS) interface, and the user input/output utilities for a reconfigurable module.

A reconfigurable module (RMOD) consists of several components. Each component is defined as a C subroutine, which can be called by the underlying Chimera III subsystem (SBS) utilities. The components are called *init, off, on, cycle, kill, error, clear,* and *sync*. For each RMOD, a local state structure which stores the variables which are shared by different components of that module is defined. Therefore, for the module *xxx* a local state structure *xxxLocal_t* and functions (xxx*Init()*, *xxxOff()*, *xxxOn()*, *xxxCycle()*, *xxxKill()*, *xxxError()*, *xxxClear()*, and *xxxSync()*) should be defined. Figure E.17 shows different states and a timing block diagram of an RMOD module. The user specifies different information for the module in a *.rmod* file which has the following format:

```
MODULE                         ReadDiffJoint
DESC                           Reads the joint values in Rad
INCONST                        none
OUTCONST                       q dq
INVAR                          none
OUTVAR                         q dq
TASKTYPE                       periodic
PERIOD                         0.005
EOF
```

Once all the modules are specified, a subsystem file *(.sbs)* can be used to specify general information, including the name, the state variable table (a shared memory segment which can be accessed by different modules), the name of the RTPU on which to execute the SBS interface. It contains a list of one or more RTPUs, which can be used by the subsystem to run the modules. A *.sbs* file has the following format:

```
SUBSYSTEM                      test
SVARFILE                       test_gio.svar
MASTER                         vader
RTPU                           vader
RTPU                           emperor
EOF
```

where *vader* and *emperor* are the names of the processors.

The *xxInit()* routine is called whenever a module is spawned to create a new task. It is used to initialize the task. Generally speaking, the following operations should be performed by *xxxInit()*:

- Allocate memory for the dynamically-sized variable within the *xxxLocal_t* structure.
- Create and attach to any resources required by the task, such as shared memory, semaphores.
- Read the configuration files (*.cfig*) that are necessary to initialize the variables.
- Initialize - *iodOpen()*, any I/O card used by the task.
- Get pointer to the state variable table (SVAR) used by the task; (SVAR will be described later).

Once the *xxInit()* routine completes its execution, the *OUTCONSTs* are copied to the global table, and the task enters the *OFF* state. The task remains in the *OFF* state until an *on* signal is received. When the *on* signal is received, the *xxxOn()* routine is called. The purpose of this routine is to bring the task up to date with the current state of the system. Before the routine is called, both the *OUTVARs* and *INVARs* are read in from the global table. Once the task enters the *ON* state, it can enter any of Chimera's kernel states, which are *running, ready,* or *blocked* on either a time or resource signal. If the task is periodic, it enters a *time-blocked* state, otherwise it enters a *resource-blocked* state. It remains in that state until a wakeup signal arrives, which places the task in the *ready* state, The wakeup signal is either a timer interrupt for periodic tasks, or a resource signal (unblocking from a semaphor by a *SemV()* call). At this time, it first reads the *INVAR*s from the global SVAR table, then it calls the *xxxcycle()* routine, and writes its *OUTVAR*s back to global table. After that it goes back into its *blocked* state until its next cycle.

The task remains in an *ON* state as long as it does not receive the *off* or *kill* signals. The *off* and *kill* signals both cause the routine *xxxOff()* to be called. The purpose of this routine is to turn off the task in a predictable manner, by ensuring that *OUTVAR*s are written to a global table. When *xxxOff()* returns, the task goes into an *OFF* state if the *off* signal was received or the routine *xxxKill()* is called if the *kill* signal was received. The *xxxKill()* routine must free up all resources used by the task and the task goes back into a *NOT-CREATED* state.

The SBS interface is designed to manage the creation and execution of reconfigurable modules. The management of these modules can be controlled by a user program. A subsystem consists of a master task, which executes on the master RTPU and oversees the creation and management of the reconfigurable task, and a server on each RTPU which is a part of the subsystem. Each RTPU on which the subsystem is to execute must start by executing the SBS server: *sbsServer()*. This call usually placed as the first instruction in *main()*, spawns a background server which handles incoming requests from the SBS master task. On all but the master RTPU, this is the only instruction that is required. The *main()* can exit after making this call. Therefore, the complete program for the secondary RTPUs is the following:

```
#include                        <chimera.h>
#include                        <sbs.h>
main()
{
            sbsServer();
            exite(0);
}
```

The subsystem is defined through a .sbs file. It must be initialized by a program executing on the master RTPU, using the *sbsInit()* routine.

The following routines can be used to program the operation of the subsystem:

- *sbstask_t *sbsSpawn (sbs, rtpuname, rmodfile):* is used to spawn the module specified by *rmodfile* on the RTPU *rtpuname*. It returns a pointer to an *sbstask_t* structure.
- *sbson (stask,argptr):* sends an *on* signal to the task specified by the *stask* pointer.
- *sbsoff (stask, argptr)*: sends an off signal.
- *sbskill (stask, argptr):* sends a kill signal.

The SBS support in Chimera allows modules to send a signal back to the master task controlling the execution of the subsystem. The master can wait for a signal from the subsystem using the following routine:

- *sbsSigWait (sbs, staskptr)*: blocks the master task until it receives a signal from one of the modules. It returns a pointer to the stask_t structure of the module that sends the signal and also an unsigned variable representing the type of the signal.

User signals can be generated using the macro *SBS_SIG(x)*, where x is a value in the range *SBS_SIG_END* $< x <$ *31*. The following function is used to send a signal to the master task by a module:

- *sbsSigSend(stask, SBS_SIG(x))*

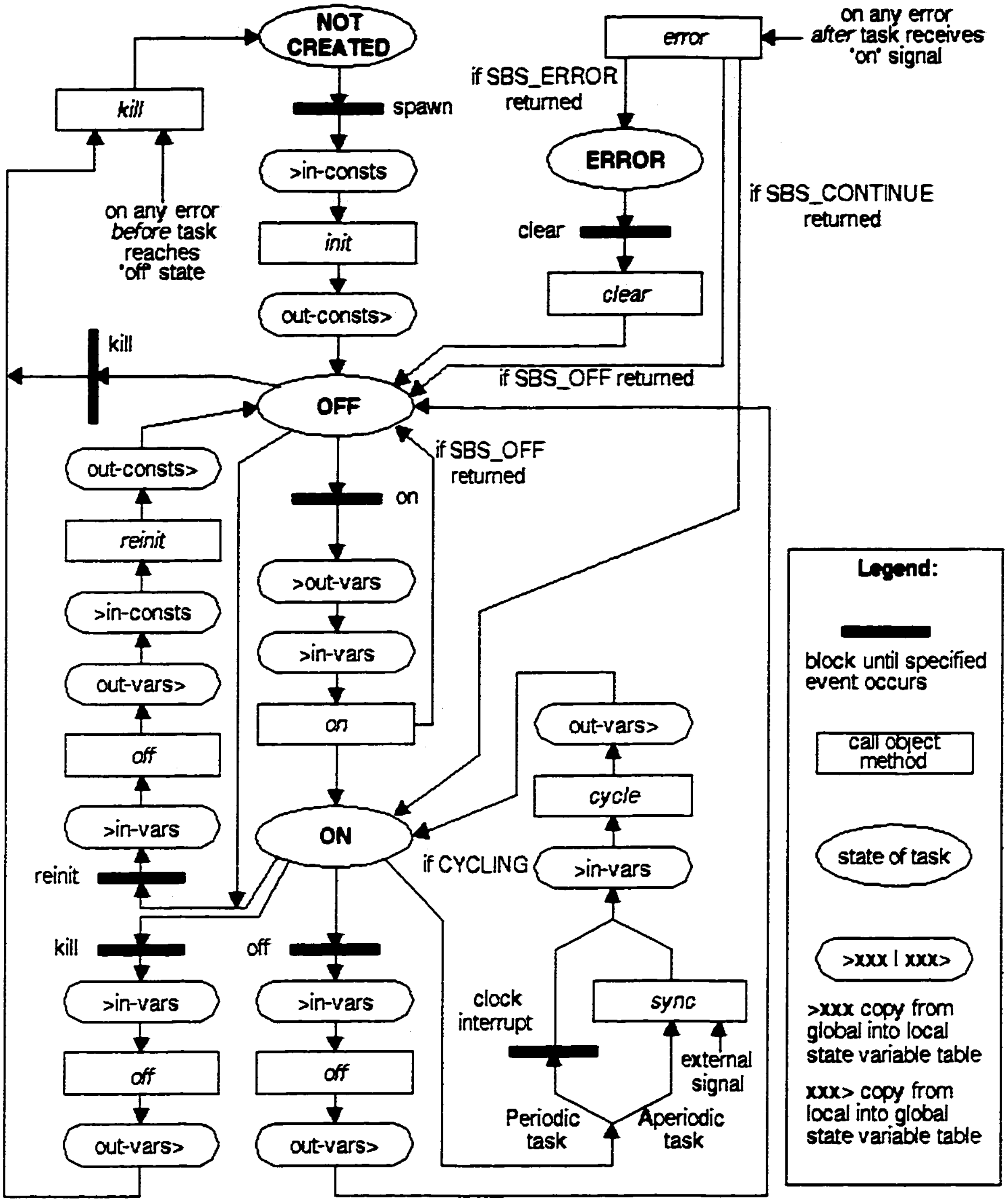


Figure E.17 Internal structure of an SBS module in Chimera III.

E.2.3.4 VME System Software Description

Table E-6 summarizes the different modules that are used in the subsystem that performs data acquisition and I/O control using two 68030 processors.

Table E-6 List of the reconfigurable modules

No	1	2	3	5
.rmod file	read_diffjoints.rmod	sendtorque.rmod	readforce.rmod	gioio.rmod
Source file	ReadDiffJoint.c	SendTorque.c	ReadForce.c	GIOio.c
Configuration file	home.dat acutator.cfig	actuator.cfig	JR3.cfig	none
type	periodic	periodic	periodic	synchronous
INVAR	None	τ	None	$q, \dot{q}, f$
OUTVAR	$q,$	none	f	τ
Freq. (Hz)	200	200->500	200	NA
IO device	Encoder interface boards	DAC DIO	JR3 interface card	bit3 adapter card
drivers	xcode	xdac xpio	xfor	bert0
signals to master task	None	None	None	START_CONTROL STOP_CONTROL

The *read_diffjoints.rmod* reads the encoder card interfaces, performs the necessary calculations and puts back the joint values in the SVAR table. This module also calculates the joint rates by differentiating the joint values using the *finite difference method.* The *readforce.rmod* module communicates with the force-sensor interface card. The *sendtorque.rmod* module reads the command torque from the SVAR table and performs the required calculations and send the results to the D/A converter. The *gioio.rmod* module performs the communication with the real-time controller running on the SGI workstation. This module and the master task of the subsystem will be described in more detail:

- GIOio Module

This module is responsible for communication between the real-time controller process running on the SGI workstation and the modules running on the VME processors. Figure E.18 shows the block diagram of the data communication between different modules. The reconfigurable GIOio module communicates with other modules on the VME processors via the state variable table. At the same time, it can send/receive data to/from the real-time controller on the SGI workstation via the shared memory segment (dual-port RAM) on the BIT3 bus adaptor. The local structure of this module is defined by:

```
typedef struct {
        double                    *f;
        double                    *dq;
        double                    *q;
        double                    *qd;
        double                    *ta;
        IOD                       *biod;
        int                       sem;
        gioio_t                   *gioio;
        long                      command;
}GIOioLocal_t;
```

where *f* represents the vector of the interaction forces and torques, *dq* is the vector of the joint rates obtained by numerical differentiation, *q* is the vector of joint angles, *ta* is the vector of commanded torques, *biod* is the IOD pointer which is required to use the driver of the BIT3 bus adaptor, *sem* is the variable used for the binary semaphore and task synchronization, *gioio* is a pointer to the memory structure shared by the GIOio module and the real-time controller on the SGI workstation (see Section E.2.2), and *command* is a 4 byte long variable.

A call to *GIOioInit()* performs the following steps:

- Opens connection to the BIT3 driver using: *iodOpen(bert0,...)*
- Gets pointer to shared memory location on the dual-port RAM (*gioio = shared_mem).*
- Initializes the *sem* and *command* fields to 0
- Installs the interrupt handler routine for the VME interrupt level (2) used by the real-time controller to send interrupts

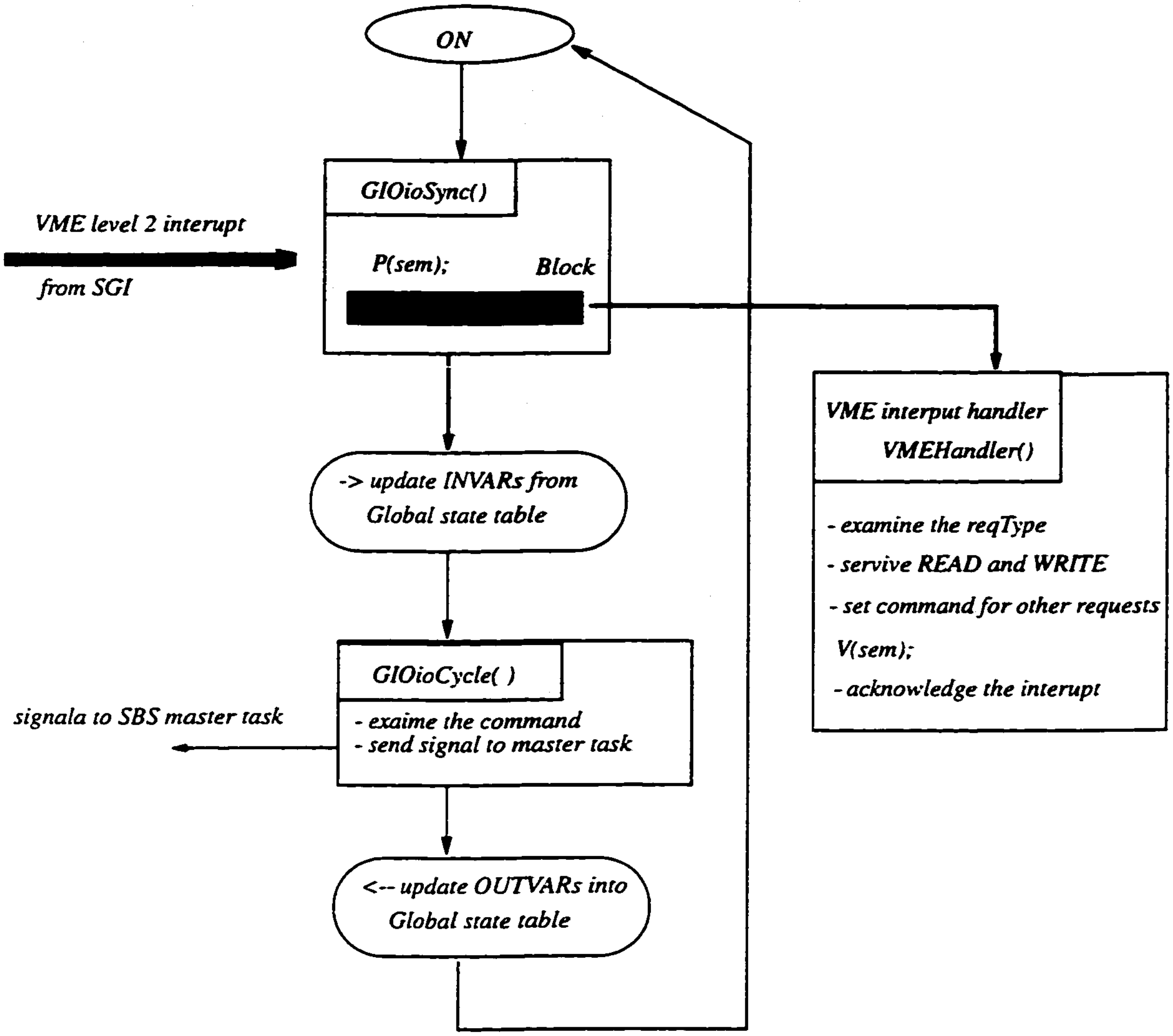


Figure E.18 Block diagram of GIOio module during the ON state

Figure E.18 shows the block diagram of the GIOio module during the ON state. This module is a synchronous module. Therefore, after receiving an on signal, it executes the *GIOioSync()* which blocks the task on the semaphore represented by the *sem* variable. A semaphore is an integer variable. The *P()* and *V()* operations (also known as *wait* and *signal* or *up* and *down* operations respectively) can be used to perform synchronization between different tasks using a common resource. If the *sem* value is less than or equal to zero, a task would block when performing a *P()* operation until another task increases the *sem* value to positive (greater than zero) by a *V()* operation. Therefore, the GIO module blocks by calling *GIOioSync()* which performs a *P(sem)* operation which blocks the task. Upon receiving an interrupt from the real-time controller on the SGI workstation, it executes the interrupt handler routine.

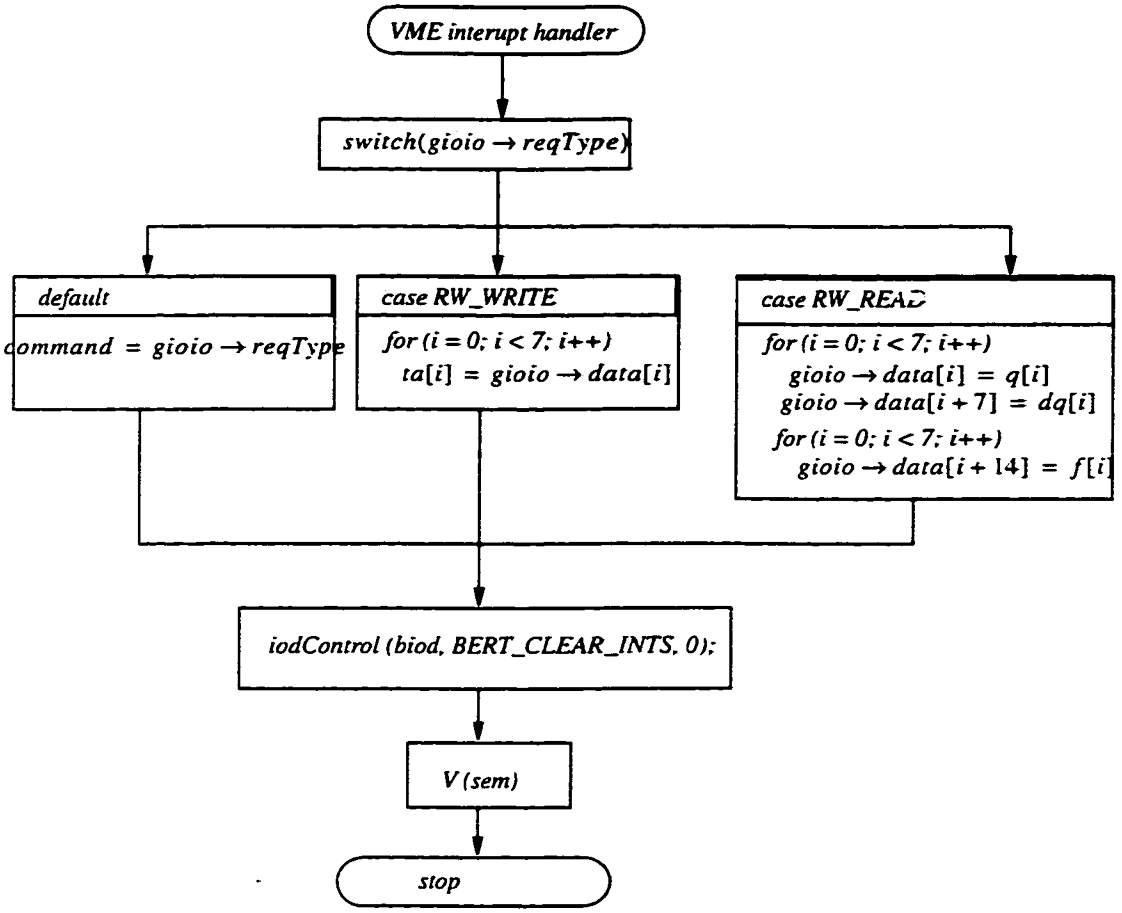


Figure E.19 Function block diagram of the *VMEHandler()* function

Figure E.19 shows different steps undertaken within the interrupt handler routine. It first examines the *reqType* field of the *gioio* pointer and performs appropriate actions; for a RW_READ request from the real-time control process on the SGI, joint angles, rates, and interaction forces (available in the global state variable table) are written into shared memory locations pointed by $gioio \rightarrow Data$; for an RW_WRITE request, the commanded torques which have already been written into the data field of the shared memory segment by the real-time control process are read (and are available to the *SendTorque* module via the global state table of the subsystem); for other types of requests, the request type is stored in the *command* field of the local structure of the module. At this point, the interrupt handler routine acknowledges the interrupt which lets the real-time controller get unblocked and continue. The last action consists of performing a *V(sem)* operation which unblocks the task and lets the function *GIOioSync()* to return. At this point the function *GIOioCycle()* is called. The block diagram of this function is shown in Figure E.20. It examines the *command* variable local structure and sends an appropriate signal to the master task. The functions *GIOioOff()* is used to put this module into the OFF state.

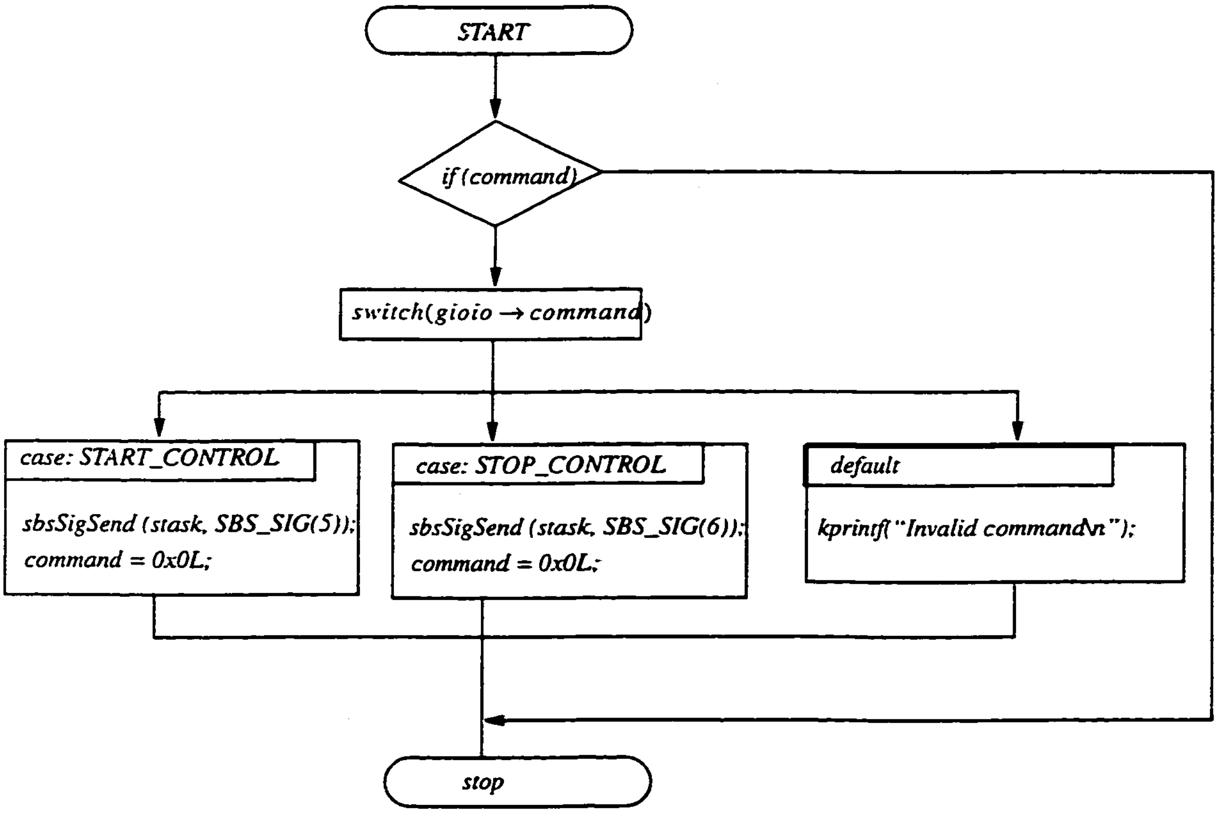


Figure E.20 Block diagram of the function GIOioCycle()

- The SBS master task.

The operation of a reconfigurable subsystem under Chimera III has been described in Section E.2.3.3 of this appendix. It was mentioned that a user program, which runs on the master RTPU, can be used to specify the operation of a subsystem and its modules. Figure E.21 shows the block diagram of the master task controlling the I/O modules and the *GIOio* module. The first step is to initialize the subsystem using the *sbsInit("fc.sbs")* command. The file *fc.sbs* specifies the name of the RTPU available for the subsystem and the name of the state variable table (*.svar*). The SVAR table file contains the definition of all INVARs and OUTVARs of all the modules defined in the subsystem (see Table E-6). The master task spawns all the modules. Then, it sends an *on* signal for the modules *ReadDiffJoints* and *ReadForce* in order to update the global state table. It also turns on the *GIOio* module which "echoes" back the command generated by the real-time controller on the SGI workstation to it. At this point, the master task pools the signal coming from the subsystem, e.g., *GIOio* module. Hence it blocks until it receives an *SBS_SIG(5)* from the *GIOio* module which shows that the real-time controller on the SGI workstation has been started. At this point, it turns on the *SendTorque* module which initially releases the brakes and periodically sends the torque command to the current amplifiers. From this point on, the master task again blocks until it receive a new signal. However, the subsystem modules perform their tasks which result in a closed-loop control of the robot. After receiving a *STOP_CONTROL* command from the SGI, the *GIOio* module notifies the master task by sending an *SBS_SIG(6)*. Upon receiving this signal the master task sends an *Off* signal to all the modules.

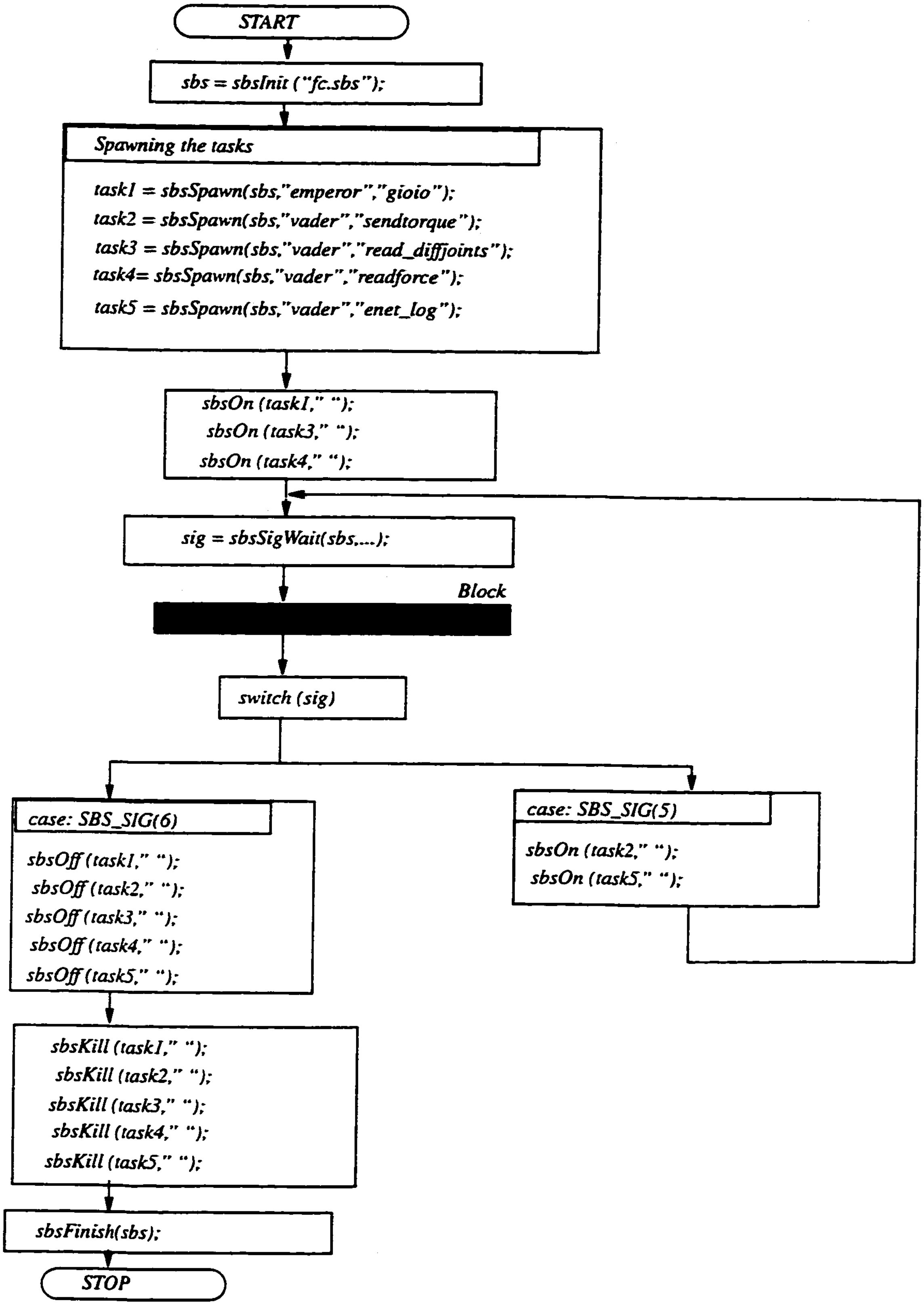


Figure E.21 Block diagram of the master task on the VME system