Optimization of Force Distribution In Redundantly-Actuated Robotic Systems

by

Meyer Nahon

B. App. Sc. (Queen's University), 1978
M. App. Sc. (University of Toronto), 1982

Department of Mechanical Engineering
McGill University
Montreal, Canada

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

February 15, 1991

Abstract

This thesis presents an analysis of redundantly-actuated robotic systems with emphasis on systems which have a time-varying kinematic structure such as mechanical hands, walking machines and multiple manipulators grasping a common object.

Firstly, graph theory is used to characterize the kinematic structure of these systems and show that they can be decomposed into two subsystems, each with different properties. The contacts which occur between the constituent bodies in the system are then analyzed in order to determine the system's mobility (or number of degrees of freedom). It is found that this mobility varies during the task and that, at any given time, there will be more actuators active than are necessary.

The kinematic and dynamic equations governing the motion of these systems are then studied and compared to those of more conventional robotic systems. Although the inverse dynamics equations can be formulated in a number of ways, they always constitute an underdetermined system of linear equations. This allows their treatment as equality constraints in an optimization problem. In order to account for the limitations of passive contacts and actuator capabilities, inequality constraints are also considered.

The formulation of the optimization problem is then studied with emphasis on problems which are solvable in real-time and which produce time-continous solutions. Quadratic programming is found to be a good choice of problem formulation. A quadratic-programming algorithm which efficiently includes both equality and inequality constraints is presented. A number of linear and quadratic objective functions which could be optimized are reviewed and the limitations of linear programming are made apparent through the use of numerical examples. Quadratic objective functions which minimize internal force, power consumption and solution discontinuities are examined. Finally, other applications of redundant actuation are briefly touched upon—the full dynamic balancing of linkages and the reduction of impact shocks in robotic systems.

Résumé

Cette thèse présente une analyse mécanique des systèmes robotiques à motorisation redondante avec un accent particulier sur les organes de préhension, les machines à pattes et les ensembles de robots manipulant un objet commun.

En premier lieu, la théorie des graphes est appliquée à la classification de leur structure cinématique et il est démontré que celle-ci change durant la tâche, entraînant ainsi un changement du nombre de degrés de liberté (ddl) du système. En comparant le nombre de ddl avec le nombre de moteurs installés dans le système, il devient évident que ces systèmes disposent de plus de moteurs actifs qu'il n'est nécessaire.

Les équations cinématiques et dynamiques de ces systèmes font l'objet d'une étude approfondie. Il est démontré que l'excès de moteurs permet d'optimiser la distribution des forces dans le système. De ce point de vue, les équations de la dynamique du système deviennent des contraintes d'égalité dans un problème d'optimisation. En raison des types de contacts qui peuvent exister dans les systèmes que nous considérons, il devient essentiel d'inclure des contraintes d'inégalite dans l'optimisation.

Les méthodes d'optimisation sont considérées du point de vue analytique et numérique, en insistant sur celles qui peuvent être réalisées en temps réel. La programmation linéaire, qui est souvent utilisée pour ce genre de problèmes, n'est pas appropriée dû à la possibilité de solutions multiples. La programmation quadratique devient alors la méthode de choix. Un algorithme capable de résoudre le problème de programmation quadratique avec contraintes linéaires de manière rapide et efficace est donc présenté.

Puis, le sujet de la fonction objectif qui devrait être optimisée est abordé. Plusieurs fonctions sont sugérées telles que la minimisation des forces internes, de la consommation d'énergie et des discontinuités des solutions obtenues. Finalement, d'autres applications de la motorisation redondante sont considérées, telles que le balancement dynamique complet de mécanismes à quatre barres et la réduction des chocs.

Acknowledgements

I would like to extend my sincere thanks to my thesis advisor, Professor Jorge Angeles, not only for his guidance and support, but for showing me other dimensions in solving problems, and for his patience in the face of my impetuousness. His depth and breadth of knowledge were an inspiration throughout this work.

As well, I would like to thank Professor Vincent Hayward who, through early dicussions, helped to shape my perception of redundant actuation which is central to this work. I am also grateful to Professor Lloyd Reid, my master's thesis supervisor and previous employer, who was the first to instill in me an interest in research and an aspiration to produce work of the highest quality. I would like to acknowledge the financial support of the Natural Sciences and Engineering Research Council throughout the course of my graduate studies.

I am indebted to the graduate students in my group who have made my stay at McRCIM a warm and enriching experience. Although all of them have contributed to this work in some way, I would like to single out Clément Gosselin and Ma Ou for numerous insightful discussions which helped me in the understanding of my work.

I am also beholden to my parents who, even though they never thought that pursuing doctoral studies was a particularly practical move, spurred me on to finish the degree and 'get serious about my life'.

Finally, and most importantly, I would like to thank my wife, Inna Sharf, for her unflagging love and support through the ups and downs of this thesis. I am especially grateful for all those walks to and from school, her toleration of my sometimes-obsession with work and for being foolhardy enough to marry me. For her technical help, I am also extremely grateful.

À mes parents

Contents

Abstract i

Résumé ii

List of Figures x

List of Tables xiv

Claim of Originality xv

1 Introduction 1

1.1 A Review of Cooperating Robotic Devices 2

1.1.1 Legged Vehicles 2

1.1.2 Mechanical Hands 3

1.1.3 Cooperating Manipulators 5

1.2 An Introduction to Some Problems 6

1.3 A Brief Overview of Previous Work 12

1.4 Organization of the Present Work 14

2 The Kinematic Structure **16**

2.1 Graph Representation 17

2.2 Classification of the Kinematic Structure 19

2.3 Classification of Joints and Contacts 22

2.3.1 Rolling Contact 27

2.4 Mobility 28

2.5 Connectivity 32

2.6 Actuation 35

2.6.1 Redundant Actuation 36

3 Kinematics and Statics **39**

3.1 Position Analysis 40

3.1.1 Forward Kinematics 41

3.1.2 Inverse Kinematics 45

3.2 Velocity Analysis 47

3.3 Acceleration Analysis 52

3.4 Mobility and Connectivity Revisited 53

3.5 Controllability of a Kinematic Chain 54

3.6 Kinematic/Static Duality 58

4 Dynamics **61**

4.1 Dynamics of Open Kinematic Chains 62

4.2 Dynamics of Closed Kinematic Chains 63

4.2.1 The Complete Formulation 65
4.2.2 A Formulation Using Superposition 70
4.2.3 The Most Compact Formulation 73
4.2.4 Calculating Actuator and Constraint Wrenches 75
4.2.5 Example 76
4.3 The Effect of Changes in Topology 79
4.3.1 Reduction of Solution Discontinuities Upon Changes in Topology . 81
4.4 Inequality Constraints 83
4.4.1 Unilateral Contacts 84
4.4.2 The Friction Cone and Pyramids 84
4.4.3 Actuator Limitations 86
4.4.4 Limits on the Constraint Wrenches 87
4.4.5 Limiting Solution Discontinuities Upon Changes in Topology 88

5 Optimization Techniques **92**

5.1 Optimality—General Objective Functions 93
5.1.1 Unconstrained Optimization 95
5.1.2 Constrained Optimization 96
5.2 Optimality—Particular Objective Functions 100
5.2.1 Convex Functions 100
5.2.2 Linear Programming 101
5.2.3 Quadratic Programming 104

5.3 Duality 107

5.3.1 Removing the Equality Constraints 108

5.3.2 Linear Programming 110

5.3.3 Quadratic Programming 110

5.4 Methods of Solution 111

5.4.1 Linear Programming 112

5.4.2 Quadratic Programming 114

6 Objective Functions **129**

6.1 Linear Objective Functions 129

6.1.1 Orin and Oh's Objective Function 130

6.1.2 Kerr and Roth's Objective Function 132

6.1.3 Cheng and Orin's Objective Functions 137

6.2 Quadratic Objective Functions 143

6.2.1 Minimum Internal Force 145

6.2.2 Distributing the Load 152

6.2.3 Minimizing a Norm of the Actuator Torques 153

6.2.4 Minimizing a Norm of the Joint Constraint Wrenches 155

6.2.5 Nakamura's Strain Energy Objective Function 156

6.2.6 Danowski's and Pfeiffer et al.'s Objective Functions 157

6.2.7 Minimum Power Consumption 159

6.2.8 Smoothing of Jump Discontinuities Upon Changes in Topology . . . 170

7 Further Applications of Redundant Actuation 179

7.1 Dynamic Balancing of Linkages 180

7.1.1 Dynamics 182

7.1.2 Forces Acting on the Frame 183

7.1.3 Moments Acting on the Frame 185

7.1.4 Numerical Example 186

7.2 Smoothing Impact Shocks 190

7.2.1 Reducing the Effects of Shocks 191

7.2.2 Numerical Example 192

8 Conclusions 201

8.1 Recommendations for Future Work 203

References 206

Appendices 219

A Inverse Dynamics of Serial Manipulators 219

A.1 Vector Form of the Equations 220

A.2 Component Form of the Equations 221

B The Quadratic-Programming Algorithm 224

List of Figures

1.1 Robotic Systems, Machines and Mechanisms 7

1.2 Force Indeterminacy, Passive and Active Compliance 10

2.1 Graph Representation of a Mechanical Hand 19

2.2 Classification of Kinematic Chains 20

2.3 Graph Decomposition of Cooperating Robotic Devices 21

2.4 Graph of a Planar Three-Legged Walking Machine with Wave Gait 23

2.5 The Six Possible Types of Lower Kinematic Pairs 24

2.6 A Prismatic Pair with Unidirectional Vertical Force 25

2.7 Some Possible Types of Upper Kinematic Pairs 26

2.8 Reduced Degree of Freedom Due to Friction 27

2.9 A Planar Three-Legged Walking Machine 28

2.10 Mobility of Cooperating Robotic Devices 31

2.11 Two Planar Three-Fingered Hands with $M = 3$ 33

2.12 Finding the Connectivity of a Hybrid Kinematic Chain 34

2.13 Vertical Forces on the Legs of a Planar Walking Machine 38

3.1 A Serial and a Parallel Manipulator 40

3.2 A Parallel Manipulator with Coincident Pairs of Spherical Joints 42

3.3 Joint Variables of Some Planar Cooperating Robotic Devices 44

3.4 Planar Three-Legged Walking Machine 55

4.1 Decomposition of the Parallel Subchain 66

4.2 Another Decomposition of the Parallel Subchain 71

4.3 A Planar Two-Fingered Hand Grasping an Object 76

4.4 Geometric Interpretation of Scheme for Reduction of Discontinuities 83

4.5 Graphical Representation of the Friction Cone 86

5.1 A Unidimensional Convex Function 101

5.2 Non-unique Minimum in Linear Programming 102

5.3 Discontinuity in the Linear-Programming Solution 103

5.4 Unique Minimum in Quadratic Programming 107

6.1 Kerr and Roth's Example 133

6.2 The Solutions to Kerr and Roth's Example in y-space 135

6.3 The Ohio State DIGITS System 138

6.4 Diagrammatic Representation of Two-Fingered Grasp 140

6.5 Contact Forces and Joint Torques for Finger #1 141

6.6 Diagrammatic Representation of Four-Fingered Grasp 143

6.7 A Three-Fingered Hand and its Task 147

6.8 The Pick-and-Pour Task and Resulting Forces and Torques 150

6.9 Representation of an Electric Motor and its Gear Train 163

6.10 Two Puma 560 Robots Rotating a Payload 168

6.11 Grasping Forces and Moments for Manipulator #1 171

6.12 Actuator Torques for Manipulator #1 172

6.13 Winding Resistive Losses and Local Performance Indices 173

6.14 Four-Legged Walking Machine 176

6.15 X-position of the Centre of Mass 177

6.16 Foot/Ground Contact Forces 178

7.1 Four-Bar Linkage 181

7.2 Prescribed Motion of the Input Link 187

7.3 Unbalanced Linkage with One Actuator 187

7.4 Unbalanced Linkage with Four Actuators 187

7.5 Balanced Linkage with One Actuator 189

7.6 Balanced Linkage with Three Actuators 189

7.7 Balanced Linkage with Four Actuators 189

7.8 Two Puma 560 Manipulators Rotating a Payload 193

7.9 Actuator Torques for Manipulator # 1 ($\rho = 0,\ \Delta = \infty$) 195

7.10 Actuator Torques for Manipulator # 2 ($\rho = 0,\ \Delta = \infty$) 196

7.11 Actuator Torques for Manipulator # 1 ($\rho = 0,\ \Delta = 11$) 197

7.12 Actuator Torques for Manipulator # 2 ($\rho = 0,\ \Delta = 11$) 198

7.13 Actuator Torques for Manipulator # 1 ($\rho = 10,\ \Delta = \infty$) 199

7.14 Actuator Torques for Manipulator # 2 ($\rho = 10,\ \Delta = \infty$) 200

List of Tables

5.1 CPU Times for an Equality-Constrained Problem 120

6.1 CPU Times for Kerr and Roth's Example 136

6.2 CPU Times for Cheng & Orin's Two-Finger Example 142

6.3 CPU Times for Cheng's Four-Finger Example 142

6.4 Hartenberg-Denavit Parameters and Inertia Properties of Link i of Each Finger 148

6.5 CPU Times for the Three-Finger Pick-and-Pour Example 149

6.6 Puma 560 Motor Parameters 169

6.7 Global Performance 170

7.1 Four-Bar Linkage Parameters 188

Claim of Originality

The author claims the originality of certain ideas advanced in this thesis, the most significant of these being listed below:

(i) an explanation of why systems with time-varying topology are redundantly actuated and the impact this has on the methods required to control them;

(ii) a proof that a system is more controllable with a given set of actuators than with any subset of those;

(iii) a general framework to provide time-continuous force setpoints in real-time for redundantly-actuated robotic devices. This includes an algorithm based on the symbolic preprocessing of equations for equality-constrained optimization problems, as well as the extension of an existing inequality-constrained optimization technique to efficiently include equality constraints;

(iv) a number of objective functions which may be usefully optimized in the context of redundantly-actuated robotic devices. These include: the scale-independent minimization of internal forces, the optimization and limiting the non-working constraint wrenches, and the minimization of power losses in systems powered by dc servomotors;

(v) a proof that the power imparted to a redundantly-actuated system of cooperating robotic devices cannot be optimized;

(vi) an analysis of cooperating robotic devices at changes in topology and the proposal of policies to smooth the solution discontinuities occurring at those times; and

(vii) the application of redundant actuation to the full dynamic balancing of linkages and to the reduction of impact shocks effects.

These contributions have been partly reported in a preliminary form in Nahon and Angeles (1989a), (1989b), (1990a), (1990b), (1991a) and (1991b).

Chapter 1

Introduction

Early work in robotics focused on the development of single-arm anthropomorphic computer-controlled devices which functioned in isolation from one other. As the complexity of tasks evolved, so did the demands on robotic systems. Multiple robot workcells are now becoming commonplace where once isolated robots worked alone. As these workcells develop, there is an increasing desire to have the robot arms collaborate on a single task—e.g., lifting a heavy object. Similarly, just as man outdistanced the 'lower' animals by developing a dextrous hand, he wishes to bestow this same ability on his robots. Thus, an increasing amount of robotics research is aimed at the development of multi-fingered dextrous hands. Furthermore, mobility is becoming an increasingly desirable attribute for robots, allowing them to go to their task rather than have their task come to them. Legged locomotion has long been recognized as superior to wheeled or tracked locomotion for mobility in rough and unstructured environments (Bekker, 1960) causing legged robots to become the object of considerable attention for their potential utility in military and forestry applications.

The above is not meant as a whimsical reflection but rather to bring to light the importance and relevance of certain seemingly unrelated robotic systems: cooperating manipulators, dextrous hands and legged robotic vehicles, which are the focus of the present work. Although they may appear unrelated at first glance, these systems can be

perceived as an assemblage of robotic devices (i.e., arms, fingers or legs) which must work cooperatively to achieve the same end—whether that be to lift a load, manipulate a part, or walk. The purpose of this thesis is to study certain aspects which these systems have in common. More specifically, its motivation is to determine the commands which should be used to control these systems so that the individual robotic devices work cooperatively rather than antagonistically.

1.1 A Review of Cooperating Robotic Devices

Mechanical hands, legged vehicles and cooperating manipulators belong to a class of systems which can be called *cooperating robotic devices.* Although these systems are different in many respects, they also exhibit certain marked similarities. However, advances in their design and control has taken place with relatively little cross-fertilization. A brief overview of the history and present state of these systems is now presented.

1.1.1 Legged Vehicles

Legged vehicles have existed for at least 25 years, though the early ones could not really be considered *robotic* since they either had fixed leg motion or were under human rather than computer control. For example, the 4-legged 'walking truck' developed at General Electric (Mosher, 1969) had 3 joints per leg, each of which was directly controlled by the human operator seated in the vehicle. The task of simultaneously controlling all 12 joints was so onerous that even an experienced operator could only control the vehicle for a few minutes at a time (Todd, 1985). It became apparent, from this early experience, that a viable legged vehicle would have to make extensive use of computer control to relieve the operator of the low-level control tasks and use him (or her) principally as a supervisory controller (Orin, 1982).

Concerted development and design of modern robotic legged vehicles has been

centered at the Ohio State University, starting with the early OSU Hexapod (McGhee and Iswandhi, 1979; Klein and Briggs, 1980) through to the recently-completed six-legged Adaptive Suspension Vehicle (Waldron *et al.*, 1984; 1987). Many of the analytical advances in the area of legged vehicles have also come from that institution—examples being the works of McGhee and Pai (1974), Orin and Oh (1981), Waldron (1986) and Kumar and Waldron (1988). A number of legged vehicles have been built by other researchers including: a six-legged axisymmetric multi-function platform (Russell, 1983), a six-legged single-passenger vehicle (Sutherland and Ullner, 1984), and a series of small terrain-adaptable quadrupeds (Hirose, 1984). Furthermore, a considerable amount of research on legged locomotion has been performed in the Soviet Union (Umnov and Pogrebniac, 1975; Gurfinkel *et al.*, 1981; Popov, 1982; Bessonov and Umnov, 1983), though the results of this work are less readily accessible.

The above machines have the common trait of being *statically stable*—i.e., the vertical projection of their centre of mass position falls within the support polygon (the polygon formed by the connecting the foot/ground contact points). This allows the vehicle to stop at any time during its gait without falling over. Recently, *dynamically stable* legged vehicles have been developed by Raibert *et al.* (1984, 1986), Miura and Shimoyama (1984), Takanishi *et al.* (1990). The principal objective of these one-, two- or four-legged research vehicles is the development of control techniques which allow controllable dynamic stability. Interestingly enough, the dynamically stable machines built until now will fall over if they stop running because they are not statically stable. No machine has yet been built which can make the transition between statically and dynamically stable walks. A more detailed overview of the history and development of legged robots is given by Todd (1985).

1.1.2 Mechanical Hands

Until recently, the only end effectors available for robotic manipulators were simple jaw-grippers and specialized tool-holders designed to hold particular tools. Jaw

grippers have the advantage of being simple and versatile. However, because of their simplicity, they are unable to perform any dextrous manipulation—that is, maneuvering of an object relative to the end-effector. As a result, robot arms equipped with jaw grippers are constrained to do all manipulation at the arm level and the accuracy of the manipulation possible with this arrangement is limited by the usually coarse accuracy of the robot arm.

Dextrous mechanical hands, which allow manipulation to be performed by the hand rather than by the arm, have appeared relatively recently in the history of robotics and are of particular interest in tasks requiring fine manipulation. However, the development of dextrous hands has been hindered by their considerable mechanical complexity and the amount of computing power required to control them. As computing power has come down in cost and experience has been gained with the high-density mechanical drive and actuation systems required, some mechanical hands have begun to appear—though principally as research tools.

Early work by Salisbury and Craig (1982), Salisbury and Roth (1983), Kerr and Roth (1986), Mason and Salisbury (1985) and Cutkosky (1985) laid the groundwork in this field and identified problems which would have to be surmounted for mechanical hands to become viable. Dextrous hands, as opposed to simple grasping, possibly anthropomorphic, hands, were designed by Skinner (1975), Tomović and Stojiljković (1975), Crossley and Umholtz (1977), Okada (1979), Salisbury (Mason and Salisbury, 1985), Jacobsen *et al.* (1984; 1986) and Santoso (1987). The Utah/M.I.T. hand (Jacobsen *et al.*, 1984; 1986), which represents the state of the art, is an anthropomorphic 4-fingered hand with 4 joints per finger, an opposed thumb and tactile sensors. Each joint is actuated by two opposing tendons, each of which in turn, is driven by an actuator. There are therefore 32 actuators which are remotely located.

1.1.3 Cooperating Manipulators

Although industrial robotic manipulators have existed since the early 1970's they are almost invariably used to perform tasks individually. Recently, it has become clear that improvements in productivity would be attained with workcells in which multiple robots would collaborate to perform a single task and therefore would, at times, manipulate the same object. Hayward and Hayati (1988), for example, point out that multiple manipulators are well suited for tasks such as the transport of inertial loads in the absence of a gravitational field and the transport of flexible payloads.

To date, most of the work dealing with multiple manipulators has focused on their control rather than their mechanical design, on the tacit assumption that these systems would simply be composed of two separate existing manipulators. The control problems introduced by the interaction of multiple manipulators include a consideration of

1. The controller architecture and control laws needed to consistently control multiple manipulators (Hayati, 1986; Tarn *et al.*, 1988; Hu and Goldenberg, 1990),

2. The more complex software environment required to synchronize multiple processes (Hayward and Hayati, 1988),

3. The necessity of moving-obstacle avoidance (Tournassoud, 1988), and

4. The sequencing of tasks in the workcell (Hussaini and Jakopac, 1986).

Among existing control philosophies, some (e.g., Zheng and Luh, 1986, 1989) treat the various manipulators in the workcell unequally by appointing one of these as the 'leader' and the other(s) as 'follower(s)', thereby simplifying the task requirements for the individual robots. Other researchers prefer not to impose this artificial constraint and treat all robots as equals, but must deal with more complex sequencing problems.

Very recently, some research has emerged which also deals with the mechanical design of multi-armed robotic systems. Thus, the field is well enough advanced that such

systems are now being designed, particularly for space-based applications, on the premise that their arms will work together at all times (Borduas *et al.*, 1989; Iwata *et al.*, 1989; McCain, 1990).

1.2 An Introduction to Some Problems

From the mechanical engineer's perspective, robotics fits into the broader and more classical field of the theory of machines and mechanisms (TMM). Figure 1.1 shows a cross section of robotic devices and machines:

(a) A serial robotic manipulator *not* in contact with its environment,

(b) A parallel robotic manipulator—often used as a flight simulator platform,

(c) An oil pumping rig,

(d) A mechanical hand,

(e) A walking machine, and

(f) Two manipulators of the type shown in (a) handling a common payload.

These systems are all examples of *kinematic chains*—or couplings of rigid bodies by means of mechanical constraints (Angeles, 1989). Except for the serial robotic manipulator, all these systems bear the similarity that they incorporate kinematic loops—that is, a path can be traced along successive bodies and joints which starts and ends at the same point. As well, the first three systems are different from the last three in one important respect: their kinematic structure does not change with time. By *kinematic structure*, or *topology*, we mean the connections between the bodies which make up the kinematic chain—a topic which will be discussed in detail in Chapter 2.

Most mechanisms and machines have a fixed structure—even though the position and orientation of their members may change, the connections between them do

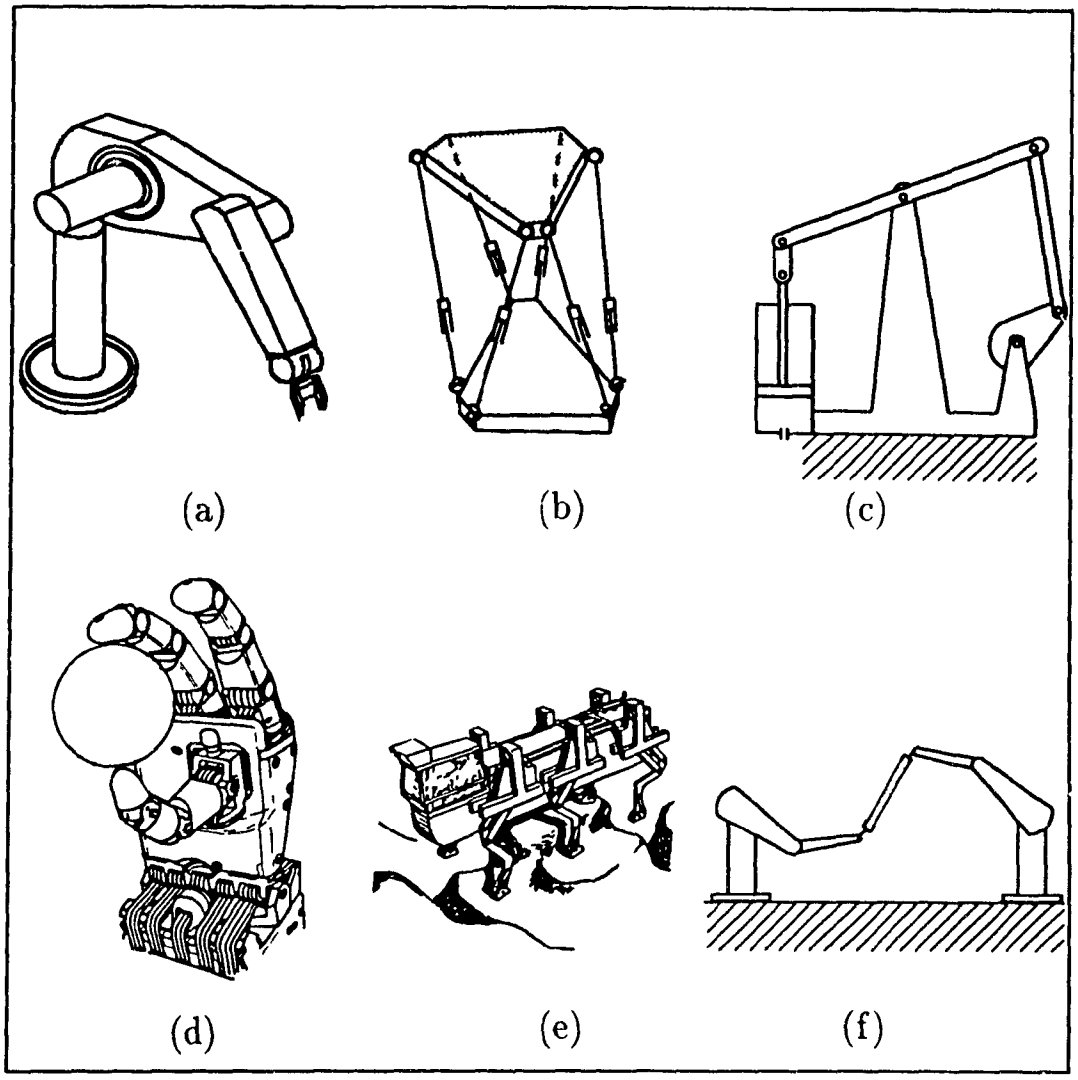

Figure 1.1 *Robotic Systems, Machines and Mechanisms*

not. As a result, the *mobility* of the system (Hunt, 1978) (a.k.a., its *number of degrees of freedom*)—i.e., the number of independent variables needed to fully specify the position and orientation of all its links—remains constant throughout its operation. At each joint, an actuator may be installed in order to control the joint variable at that joint. Therefore, a machine normally needs a number of actuators equal to its mobility in order to be fully controlled; in fact, it is common practice to install *as many actuators as dictated by its mobility*. For example, the platform shown in Figure 1.1(b) has a mobility of six and six actuators. The possibility of installing more actuators than necessary in a device of fixed topology is rarely suggested, primarily because of the negative effect that

redundant actuators can have if they are not properly coordinated. Other causes for the neglect of redundant actuation in fixed-topology devices are their cost, weight and the extra complexity they require in their control. An example of the relatively simple control requirements of a non-redundantly actuated system is that we can always uniquely find the actuator forces or torques required to effect a *prescribed* motion of that system.

In contrast to fixed-topology systems, the mobility of systems with time-varying topology varies during their task, and obvious questions which come to mind are therefore: How many actuators should be installed in these systems ? and, how many actuators should be driven at any given time ?

To answer the first question, it must be recalled that the individual robotic devices which make up the systems of interest may act independently, at times. For example, when the fingers of a mechanical hand are not in contact with each other or with a common object, they can move independently. During this part of their task, they can be considered as independent systems and each device must have installed at least as many actuators as required by its mobility. Thus, the number of actuators in the system is chosen according to the mobility of its constituent kinematic subchains, when these act independently. In the case of the two six-axis manipulators shown in Figure 1.1(f), there would be a total of twelve actuators installed in the system, and all of them would be driven when the manipulators are not grasping the common payload.

The answer to the second question is not as straightforward. When individual subchains come into contact with each other directly or through a commonly-grasped object, closed kinematic chains are formed and *the mobility of the system is decreased.* In order to keep the system non-redundantly actuated, a control policy could be adopted to turn off the same number of actuators as the reduction in mobility, while keeping the corresponding joints free to move. Alternatively, all actuators could be kept active, while ensuring that the control commands to the actuators are not antagonistic. If the commands conflict with each other, large forces may be generated both in the robotic

devices and on the common object—e.g., in the case of a mechanical hand, the grasped object might be crushed. We can therefore expect to encounter more control problems if we persist in driving too many actuators, and, should we choose this strategy, we must ensure that there will be compensating benefits.

The tradeoff between added control complexity and the potential benefits of redundant actuation touches on a number of related topics in robotics—most notably, that of dynamics. As previously mentioned, the problem of finding the actuator commands required to effect a *prescribed* motion of a non-redundantly actuated system—a problem also known as *inverse dynamics*—is relatively straightforward. This is not the case for multiple cooperating robotic devices which are redundantly actuated, as their corresponding inverse dynamics equations are underdeterminate—there are fewer force/moment balance equations than unknown forces and moments. This implies that there are an infinite number of solutions to these equations. Failure to choose a 'good' solution to these equations can result in the generation of excessive forces mentioned previously. In order to choose the 'best' solution, optimization techniques have been proposed to minimize an objective function while satisfying the equations. In this light, the force/moment balance equations can be viewed as constraints in an optimization problem. Once a set of optimum forces is found, it can be used as a setpoint in a force controller, such as that suggested by Hayati (1986) to ensure coordinated use of all the installed actuators.

This approach to the control problem can be regarded as *active compliance* in the same sense that this technique is applied to the control of a serial robotic manipulator in contact with its environment (see e.g., Asada and Slotine, 1986). It can also be contrasted to *passive compliance*—an approach which has also been proposed in the context of walking machines and mechanical hands by a small minority or researchers (Umnov and Pogrebniac, 1975; Lallemand, 1988; Gao and Song, 1990). In the latter approach, the force distribution in the system is not *actively* controlled, but rather assumed to take place passively due to structural compliance in the system. However, just as passive compliance has limited application in the force control of serial manipulators in contact

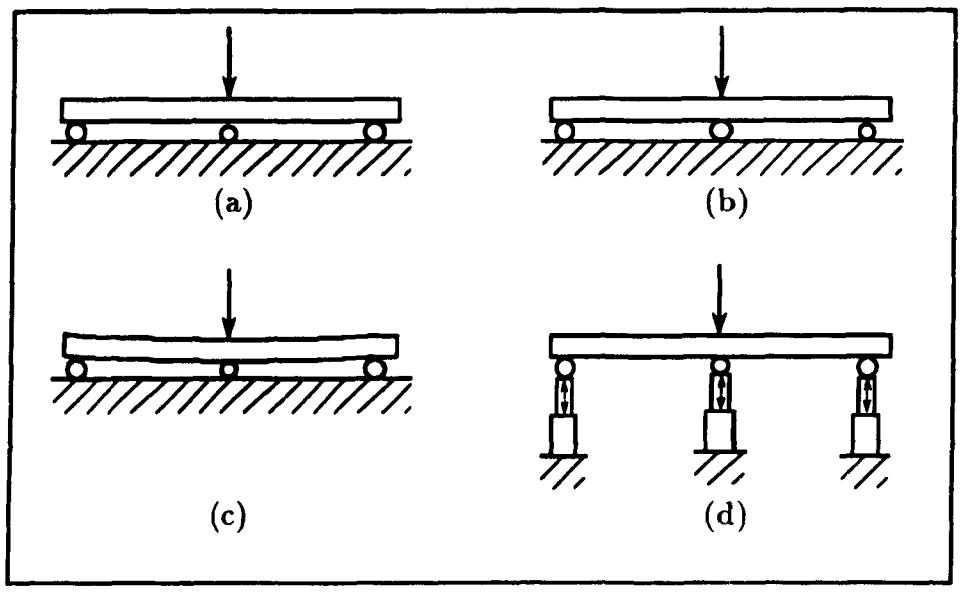

Figure 1.2 *Force Indeterminacy, Passive and Active Compliance*

with their environment (Asada and Slotine, 1986), its applicability to multiple cooperating robotic devices is also restricted. It requires that the structural design of the system be compliant enough to obtain desirable force distribution properties. This can, in turn, preclude accurate motion control which can often prove to be a serious drawback. For example, a mechanical hand, whose express purpose is to perform fine manipulation, may become limited in this very respect by its structural flexibility. Conversely, if the system is relatively rigid, the passive compliance approach can become extremely sensitive to configuration errors. In fact, active compliance can be regarded as an approach in which the compliance of the system can be adjusted according to the task, thereby making the system less sensitive to (uncontrolled) structural flexibility.

These concepts are exemplified by the simple system shown in Figure 1.2—a beam resting on three roller supports. The relevant equations for the beam are the vertical force balance and the moment balance equations, since lateral motion is unconstrained. These two equations are underdetermined because there are three unknowns: the vertical forces acting at each of the three supports. The result of this underdeterminacy is that it is impossible to determine the forces acting on each bearing. If the beam is relatively rigid, then a small difference in the size of one of the supporting bearings will cause a large

difference in the resulting force distribution—in Figure 1.2(a), the two outer bearings will each support half the load, while in Figure 1.2(b) they will support none of it. On the other hand, if the beam is very flexible, as in Figure 1.2(c), differences in the bearings should not cause much variation in the force distribution, but the system will be difficult to control accurately. The approach proposed here and by most other researchers, illustrated in Figure 1.2(d), is to assume that the mechanical system is rigid, but introduce compliance through the control strategy. This is done by choosing an optimal set of forces—in this case, perhaps, such that each bearing supports one third of the load—and slightly move the supports vertically to achieve this balance.

It was previously mentioned that the force/moment balance equations can be viewed as constraints in an optimization problem. They are not the *only* constraints which need be considered, however. Certain systems—most notably mechanical hands and walking machines—have unilateral physical constraints due to the nature of the contacts involved. These inequality constraints generally arise as a result of passive frictional contacts in systems which depend on these for force transmission between different parts of the system. The contact conditions between the feet and ground in walking machines or between the fingertips and manipulated object in a mechanical hand are good examples of these. For example, Figure 1.1(d) depicts a mechanical hand holding an object where, in addition to the force/moment balance equations, there exist further inequality constraints at the fingertip/object contacts. Normal contact forces cannot be negative and the magnitude of the tangential force at each fingertip cannot exceed the maximum force due to static friction. Thus, any solution method to the optimization problem must be able to consider inequality constraints. As well, since it is intended that the result of the optimization will be used as a controller setpoint, solutions must be obtained in real time. Summarizing, the real-time control of cooperating robotic devices involves the solution of an optimization problem subject to both equality and inequality constraints.

Another question which must be answered before the optimization problem is solved is quite basic: what should we optimize? In fact, when a system is redundantly

actuated, it becomes possible to optimize the force distribution it exerts on its environment and/or its internal force distribution. Thus, in the case of a mechanical hand, the finger-object contact forces can be reduced and homogenized, reducing chances of slippage and crushing, and allowing a lighter structure to be built to carry the same loads. Redundant actuation can also allow greater safety in case of breakdown of individual actuators because a redundantly-actuated mechanism can still be controlled if one or more actuators breaks down. Yet another advantage of redundant actuation is that it permits us to choose a solution to the inverse dynamics problem which satisfies the inequality constraints previously mentioned—which might not be possible in a determinate system.

Finally, it is interesting to note that the human body makes ample use of redundant actuation *and* of control strategies where certain actuators are turned off during part of their stroke. For example, the human arm has three joints with a total of 7 to 9 degrees of freedom (Shipman *et al.*, 1985) to position and orient the hand, but uses 29 muscle groups (Gray, 1985) for actuation of these degrees of freedom. Not all muscle groups are active at all times, but rather only the ones which have a good mechanical advantage. At any given time, there will always be more muscles active than are absolutely necessary but they will always act cooperatively.

1.3 A Brief Overview of Previous Work

To the author's knowledge, the first observation that the inverse dynamics problem of cooperating robotic devices admitted many solutions was made by McGhee and Orin (1976) in the context of walking machines. The work of Williams and Siereg (1979) was also notable by being the first to use the term 'redundant actuation'. Orin and Oh (1981) adopted a linear-programming approach to solve the inverse dynamics problem while developing a technique to determine forces in the design of walking machines. Klein *et al.* (1983) also noted the problem while obtaining force setpoints for the controller of the OSU Hexapod, and used a pseudoinverse solution. Kerr and Roth (1986)

noted the underdeterminacy of the force problem in mechanical hands and proposed a more conservative linear-programming approach which stays as far away as possible from the inequality constraints. Waldron (1986) and Kumar and Waldron (1988) proposed a technique based on 'interaction forces' applicable to systems with zero contact torques to solve the problem but neglected to demonstrate the computational efficiency of this scheme. Nakamura *et al.* (1987) proposed their own inequality-constrained nonlinear programming algorithm to minimize internal forces, but, again, did not consider its potential for real-time implementation. Finally, Zheng and Luh (1989) proposed another inequality-constrained nonlinear optimization formulation, but found that computational complexity and frequent switching of joint torques led to problematic solutions.

Until about 1988, the only approaches which seemed to be implementable in real-time were those based on pseudoinverse solutions which could not consider inequality constraints. Cheng and Orin (1989) were able to greatly speed the linear-programming solution by formulating a 'compact dual' problem, thereby allowing real-time implementation of linear programming. However, this approach yields discontinuous solutions due to the peculiarities of linear programming. Klein and Kittivatcharapong (1990) applied a nonlinear-programming algorithm to minimize a linear objective function in walking machines but were forced to accept suboptimal solutions to avoid discontinuities.

One aspect which has been neglected by previous works is a consideration of the behavior of the systems in question upon changes in topology. Very few authors even consider examples in which changes in the system's topology occur, even though the force optimization problem is of interest principally for systems with time-varying topology. The few works which present examples including changes in topology report results which exhibit severe discontinuities in their force time histories.

1.4 Organization of the Present Work

The present work presents a unified approach to the problem of finding force setpoints for the control of cooperating robotic devices. Chapter 2 presents an analysis of the kinematic structure, or topology, of cooperating robotic devices, where graph theory is used to classify their structure. The nature of the contacts and joints occurring in these systems is then studied with emphasis on the freedom they allow and the constraints they impose on the bodies which they couple. These concepts are then brought together to determine the mobility and connectivity of the systems of interest. Chapter 3 expands the kinematic analysis to include not only the effects of kinematic structure, but also the geometry and motion of the system. The position, velocity and acceleration kinematics of open and closed kinematic chains are reviewed. More comprehensive evaluations of the mobility and connectivity of the systems of interest are introduced and the duality between kinematics and statics is reviewed.

The dynamics of cooperating robotic devices is important when trying to determine the forces and torques acting on the system during a particular task. Chapter 4 investigates the formulation of the system's governing equations and shows that the inverse dynamics equations are underdetermined when the system is redundantly actuated. This allows us to optimize an objective function while searching for a solution to the inverse dynamics equations. The effect of time-varying topology on the dynamics equations is investigated to show that the coefficients of the motion equations vary discontinuously, causing jump discontinuities in the force time histories. Since the actuators are not expected to be able to respond to discontinuous torque commands, a smoothing of these discontinuities is presented. Inequality constraints on the solution to the dynamics equations are also introduced in this chapter to represent the limitations of passive frictional contacts, actuator and joint limitations and smoothness constraints on the solution.

The necessity to optimize having been established, Chapter 5 investigates techniques which can be used to perform the optimization with an emphasis on real-time im-

plementation. Criteria for the existence and uniqueness of minima are reviewed because these are useful in comparing linear and quadratic programming. It is demonstrated that the latter has superior performance in terms of both the smoothness of its solution, and its computational speed. Algorithms are then presented for the solution of the optimization problem and an existing technique for inequality-constrained quadratic optimization is extended to include equality constraints. Chapter 6 then presents some linear-quadratic objective functions which can be minimized while finding a solution. These include 'internal force', power losses and solution discontinuities.

Alternative uses for redundant actuation are presented in Chapter 7, for systems with time-varying or constant topology. These include the full dynamic balancing of linkages and the reduction of the effects of shocks in linkages. Finally, Chapter 8 concludes with recommendations for future work.

Chapter 2

The Kinematic Structure

A kinematic chain may be defined as a set of bodies, each of which is called a link, coupled by joints between adjacent bodies. Thus, cooperating robotic devices can be classified as kinematic chains with particular characteristics in their kinematic structure which set them apart from other types of robotic systems. This chapter presents a classification of the structure, or *topology*, of the kinematic chains of interest—i.e., a description of the number of links and joints in the system and their interconnections, disregarding geometric details such as link lengths and shapes. The *graph representation* of a kinematic chain is introduced in order to provide a systematic framework for classifying and analyzing its topology. Graph theory will then allow us to classify the chains of interest as a subclass of more general kinematic chains. Since the topology of a kinematic chain is affected by the constraints imposed by the contacts between its constituent rigid bodies, the contacts which occur within the robotic systems of interest will be characterized.

Kinematic chains may be subdivided into *structures*, the purpose of which is to transmit forces, and *mechanisms*,[1] the purpose of which is to transmit motion. The distinguishing difference between the two is the *mobility* of the chain—the mobility of a structure is non-positive, while that of a mechanism is positive. Robotic systems may be analyzed as mechanisms or structures, depending on the intent of the analysis. In general,

[1] In the present context, the term mechanism includes open kinematic chains

their purpose is that of a mechanism, but they can also be instantaneously considered as structures by assuming their actuators to be locked in a certain position. One of the important tasks when designing a complex mechanism such as a mechanical hand will be to ensure that it has the capability to move an object in as many degrees-of-freedom as desired and, conversely, that the object will not move when the actuators are locked. The concepts discussed in the earlier sections will therefore be brought together to determine the *mobility* of kinematic chains and the *connectivity* of any two links in the chain.

2.1 Graph Representation

Graph theory is a field of applied mathematics (Harary, 1969) which provides a useful abstraction for the analysis and classification of the topology of kinematic chains. The graph representation of kinematic chains has been used by, among others, Dobrjanskyj and Freudenstein (1967), Baker (1981), Davies (1981), Angeles and Gosselin (1988), Gosselin (1988) and Tsai and Lee (1989). It consists of a diagram where each link is represented by a point and each joint by a line. Thus, the graph representation of a kinematic chain will take the form of a collection of points connected by lines. Since the terminology of graph theory is not standardized, Harary (1969) advises a clear definition of terms before embarking on an analysis making use of graph-theoretical concepts. The pertinent definitions for the present purposes are as follows:

Definition 2.1 A *graph* $\mathcal{G}$ consists of a finite nonempty set $\mathcal{V} = \mathcal{V}(\mathcal{G})$ of P *points* together with a prescribed set $\mathcal{X}$ of Q unordered pairs of distinct points of $\mathcal{V}$. Each pair $x = \{u, v\}$ of points in $\mathcal{X}$ is a *line* of $\mathcal{G}$; x is said to *join* u and v, and u and v are said to be *adjacent*.

Definition 2.2 The *degree* of a point u in a graph $\mathcal{G}$ is the number of lines incident with u.

Definition 2.3 A *subgraph* of $\mathcal{G}$ is a graph having all its points and lines in $\mathcal{G}$.

Definition 2.4 A *walk* is an alternating sequence of points and lines, beginning and ending with points, in which each line is incident with the two points immediately preceding and following it.

Definition 2.5 A *path* is a walk with all its points and, necessarily, all its lines distinct.

Definition 2.6 A *cycle* is a walk beginning and ending at the same point, including at least three points, and with all but its first and last points distinct.

Definition 2.7 A graph is said to be *connected* if every pair of points is joined by a path.

Definition 2.8 A *tree* is a connected graph which has no cycles.

Definition 2.9 A *spanning tree* $\mathcal{T}$ of a graph $\mathcal{G}$ is a connected subgraph of $\mathcal{G}$ which has no cycles and contains all the points in $\mathcal{G}$.

Definition 2.10 A *chord* of $\mathcal{T}$ is a line of $\mathcal{G}$ which is not in $\mathcal{T}$.

Definition 2.11 The *cycle basis* corresponding to $\mathcal{T}$, and denoted by $\mathcal{Z}(\mathcal{T})$ is the set of subgraphs of $\mathcal{G}$ obtained by successively combining $\mathcal{T}$ with one chord of $\mathcal{T}$, for all chords of $\mathcal{T}$.

Definition 2.12 The *cycle rank* of $\mathcal{G}$ (or number of independent cycles in $\mathcal{G}$), denoted by c, is the cardinality of (or number of elements in) $\mathcal{Z}(\mathcal{T})$. The cycle rank is an invariant of $\mathcal{G}$ and is not affected by the chosen $\mathcal{T}$.

Theorem 2.1 Known as Euler's Theorem—If $\mathcal{G}$ is a connected graph, then

$$c = Q - P + 1 \tag{2.1}$$

In subsequent sections, eq.(2.1) will be useful to determine the number of independent loops in a kinematic chain, allowing the derivation of simple mobility equations. To illustrate the above concepts, consider the mechanical hand shown in Figure 2.1(a). Its associated graph can be drawn as shown in (b), and a spanning tree can be drawn as shown in (c). The spanning tree is not unique and thus, other spanning trees could have

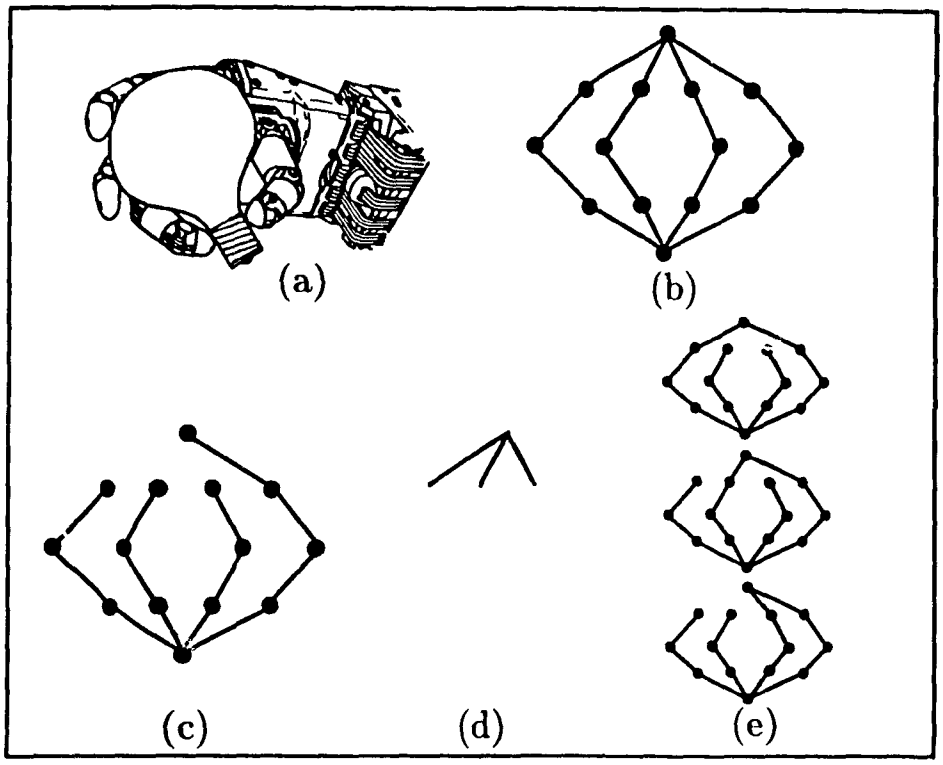

Figure 2.1 *Graph Representation of a Mechanical Hand*

been chosen. The chords corresponding to this spanning tree are shown in (d) and finally, the resulting cycle basis, which has a cardinality of 3, is shown in (e).

2.2 Classification of the Kinematic Structure

The variety of ways in which links and joints can be coupled results in a wide diversity of kinematic chains which can be classified into various groups. This classification allows us to see where cooperating robotic devices fit in relation to other kinematic chains and provides a means to begin their analysis. The concepts of graph theory introduced in §2.1 can be used to systematically describe the interconnections between the elements which constitute a kinematic chain. We need only consider systems whose graphs are connected since other cases represent the trivial addition of uncoupled kinematic chains to the system. This is implicit in the following definitions:

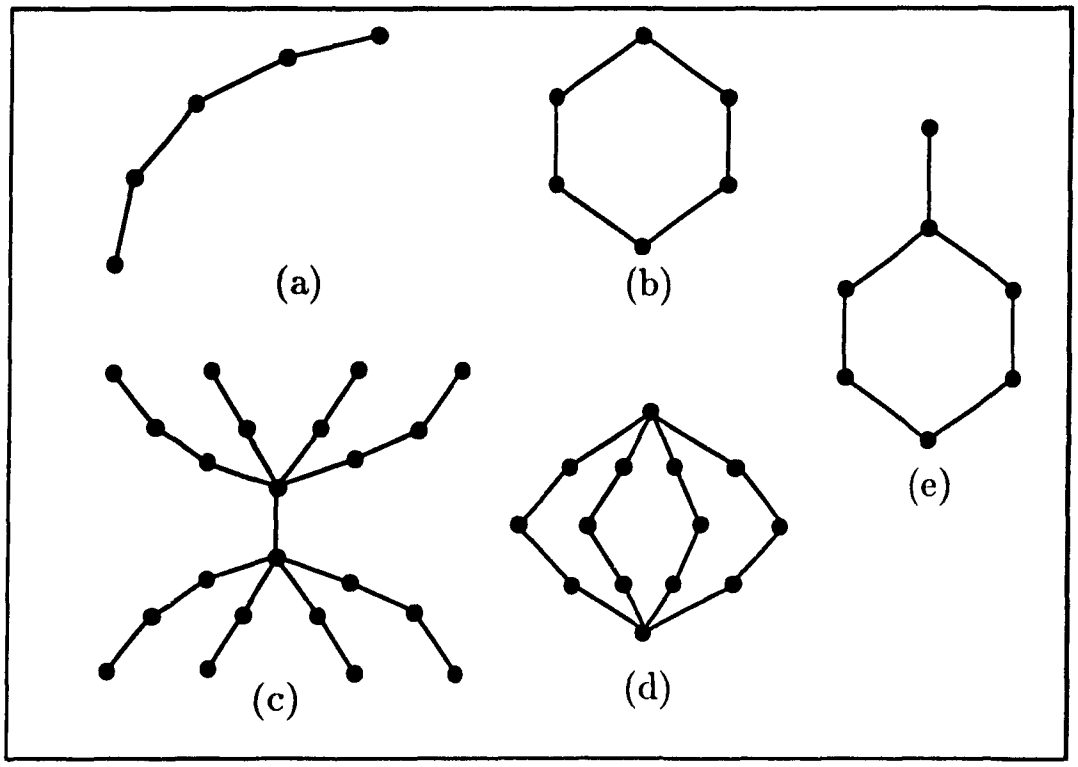

Figure 2.2 *Classification of Kinematic Chains*

Definition 2.13 A *simple* kinematic chain is one whose graph has all points of degree less than or equal to two.

Definition 2.14 A *complex* kinematic chain is one whose graph has at least one point of degree greater than two.

Definition 2.15 An *open* kinematic chain is one whose graph has no cycles.

Definition 2.16 A *closed* kinematic chain is one whose graph has at least one cycle and no points of degree one. Alternatively, it can be defined as one whose graph has no points of degree less than two.

Definition 2.17 A *hybrid* kinematic chain is one whose graph has at least one cycle *and* at least one point of degree one. A hybrid kinematic chain is always complex.

We can use the above definitions to classify the topology of any kinematic chain into the five broad categories shown in Figure 2.2: (a) simple open chain (e.g., a

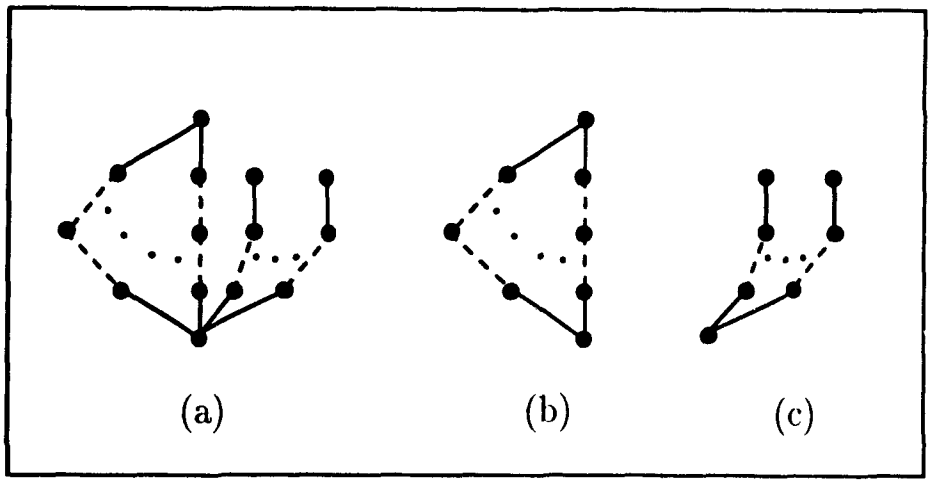

Figure 2.3 *Graph Decomposition of Cooperating Robotic Devices*

serial manipulator), (b) simple closed chain (e.g., a four-bar linkage), (c) complex open chain (e.g., a mechanical hand not grasping an object), (d) complex closed chain (e.g., a Stewart platform), and (e) complex hybrid chain (e.g., a walking machine with some of its feet not in ground contact).

Cooperating robotic systems have graphs of the form shown in Figures 2.3(a): two poles joined by an arbitrary number of parallel paths and an arbitrary number of open paths emanating from one of the poles. To illustrate this, in a mechanical hand, the two poles represent the palm of the hand and the grasped object, the parallel paths represent the fingers in contact with the object, while the open paths represent the fingers not in contact with the object. Thus, cooperating robotic systems can be classified as *complex hybrid kinematic chains* with particular features which become more apparent when we decompose the graph shown in Figure 2.3(a) into two subgraphs,

The first of these subgraphs, shown in Figure 2.3(b), represents a *parallel* kinematic chain—a particular instance of complex closed kinematic chain defined as follows:

Definition 2.18 A *parallel* kinematic chain is a closed kinematic chain whose graph has two points, called 'poles', of degree p where $p \geq 2$, and the remaining points of degree two.

The second subgraph, shown in Figure 2.3(c), represents a *star*—a particular instance of a tree structure defined as follows:

Definition 2.19 A kinematic *star* is an open kinematic chain whose graph has one point, called the 'pole', of degree q where $q \geq 1$, q points of degree one and the remaining points of degree two.

The above decomposition simplifies the analysis of the hybrid chain because, for most purposes, the two kinematic subchains can be analyzed separately. Thus, as will become apparent in the following sections, the mobility of the hybrid chain is the sum of the mobilities of the parallel and the star subchains. As well, in Chapter 3, the kinematics of the parallel and the star subchains will be treated independently of each other. Finally, Chapter 4 will show that the dynamics of the two subchains can also be treated separately.

The topology of a kinematic chain is an instantaneous property, which remains constant for most mechanisms. A characteristic of cooperating robotic devices is that their topological graph changes discretely during their task. That is, as the task progresses, some kinematic subchains which were closed are opened, and others which were open are closed—e.g., when the leg of a walking machine contacts the ground. This is exemplified by Figure 2.4 which shows the topological graph of a three-legged walking machine walking with a wave gait (Todd, 1985). In fact, the topology will change from parallel to hybrid parallel/star, and vice versa, as time progresses.

2.3 Classification of Joints and Contacts

The elements which couple the links of a kinematic chain are called joints or, more formally, kinematic pairs (Angeles, 1982). These can be classified according to the dimensionality of contact, and by the number of degrees of freedom which they allow or the number of constraints they impose. A *lower kinematic pair* is one in which

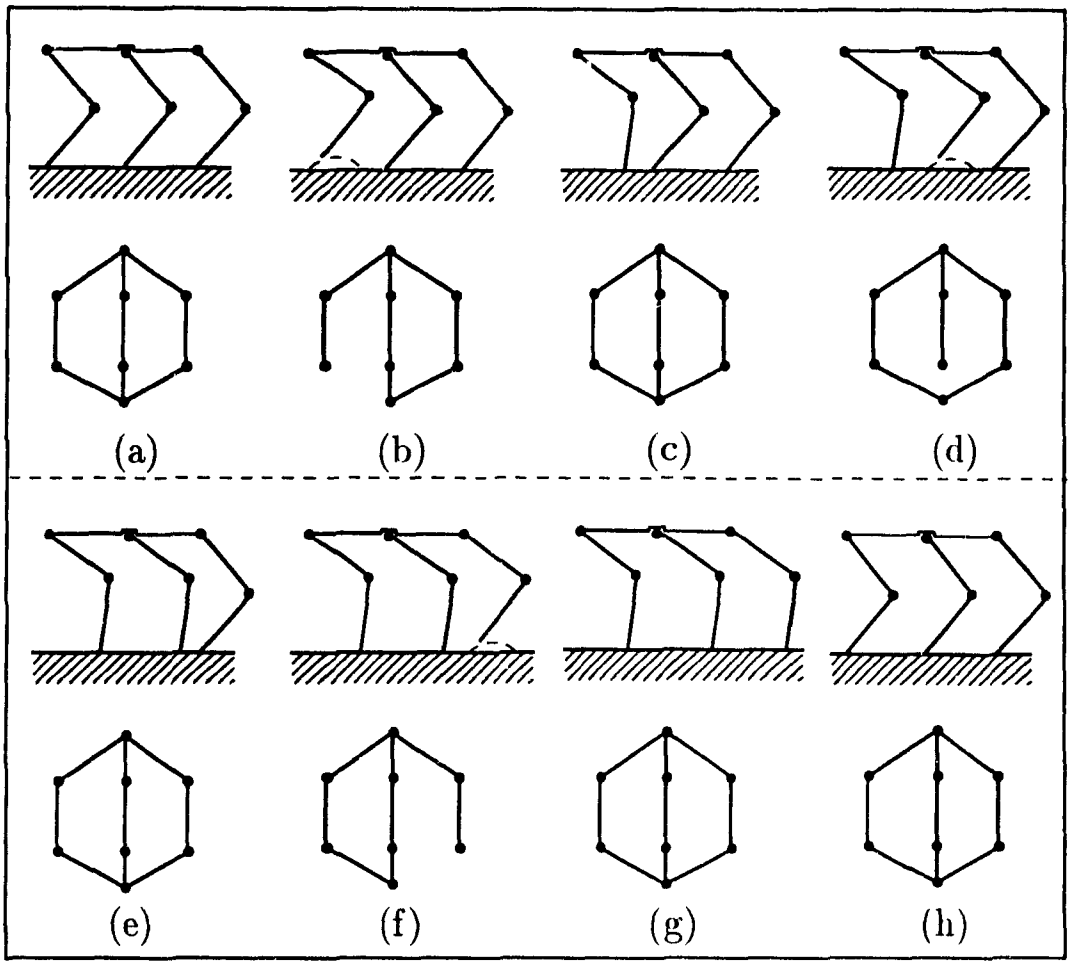

Figure 2.4 *Graph of a Planar Three-Legged Walking Machine with Wave Gait*

contact between the two links which it couples takes place along a *surface*, while a *higher kinematic pair* is one in which contact takes place along a *line* or a *point*. The six possible types of lower pairs are shown in Figure 2.5 (from Angeles, 1982): (a) revolute, (b) prismatic, (c) screw, (d) cylindrical, (e) spherical and (f) planar. Robotic mechanisms almost invariably make exclusive use of these pairs—usually revolute or prismatic– since they are fully controllable with a single motor and provide desirable qualities such as a stable contact, and a large contact area to reduce wear. It should also be noted that the first two lower pairs—revolute and prismatic—can be used in various combinations to form the last four lower pairs—e.g., a cylindrical pair is a revolute pair and a prismatic pair, the axes of which are collinear.

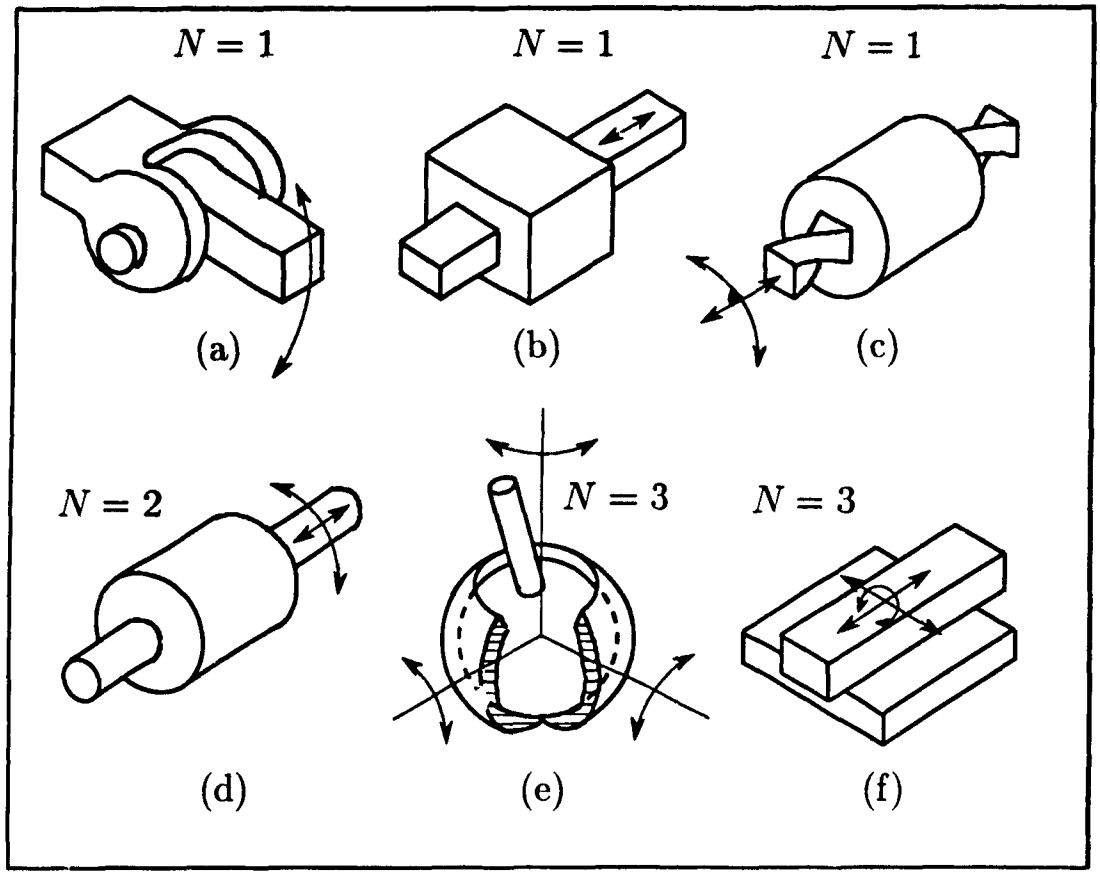

Figure 2.5 *The Six Possible Types of Lower Kinematic Pairs*

If the contact which takes place in the kinematic pair is assumed to be frictionless, each of the above pairs has a unique *degree of freedom*, N, associated with it as shown in Figure 2.5. Correspondingly, the contact surfaces will exert $6 - N$ forces in the *constrained* directions—i.e., the directions *not* corresponding to a degree of freedom. Kinematic pairs can also be characterized according to whether the forces exerted in the constrained directions can be bidirectional or unidirectional. Of the kinematic pairs shown in Figure 2.5, the first five can exert bidirectional forces in all their constrained directions, while the last one can exert bidirectional forces in two directions but only a unidirectional force in the vertical direction. It should be noted that the unidirectionality or bidirectionality of the constraint forces in a particular kinematic pair depends upon its fabrication—e.g., a prismatic pair which can only sustain a unidirectional vertical force is shown in Figure 2.6. Since kinematic pairs which do not allow bidirectional forces to

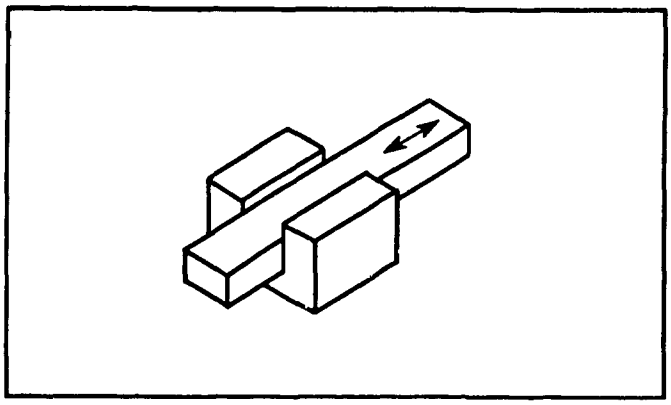

Figure 2.6 *A Prismatic Pair with Unidirectional Vertical Force*

be exerted occur frequently in the kinematic chains of interest—usually at the interface between the individual robotic devices and one of the poles—, considerable effort in later chapters will be directed toward ensuring that the constraints on the unidirectionality of the contact forces are met.

Esroy (1976) provides a comprehensive catalogue of 40 possible higher kinematic pairs. Some of these are shown in Figure 2.7, with emphasis on the ones that are likely to appear in cooperating robotic devices: (a) non-rolling point contact, (b) & (c) rolling point contact, (d) non-rolling line contact, and (e) & (f) rolling-line contact. Once again, the degree of freedom, N, associated with each frictionless upper kinematic pair is shown in Figure 2.7. This type of coupling will almost invariably occur at the interface between the individual robotic devices and their payload or the ground. Often, the treatment of higher pairs tends to be more complex than that of lower kinematic pairs because: *a)* they usually cannot sustain bidirectional forces, *b)* they are inherently less stable than lower pairs since contact takes place along a point or line, and *c)* they often involve nonholonomic constraints—i.e., the constraints which they impose cannot be written in terms of generalized coordinates, but rather in terms of generalized velocities. The coupling shown in Figure 2.7(c) is a good example of a nonholonomic coupling.

The number of degrees of freedom allowed by each kinematic pair may be reduced when frictional contact is present. If friction is assumed to be present in a

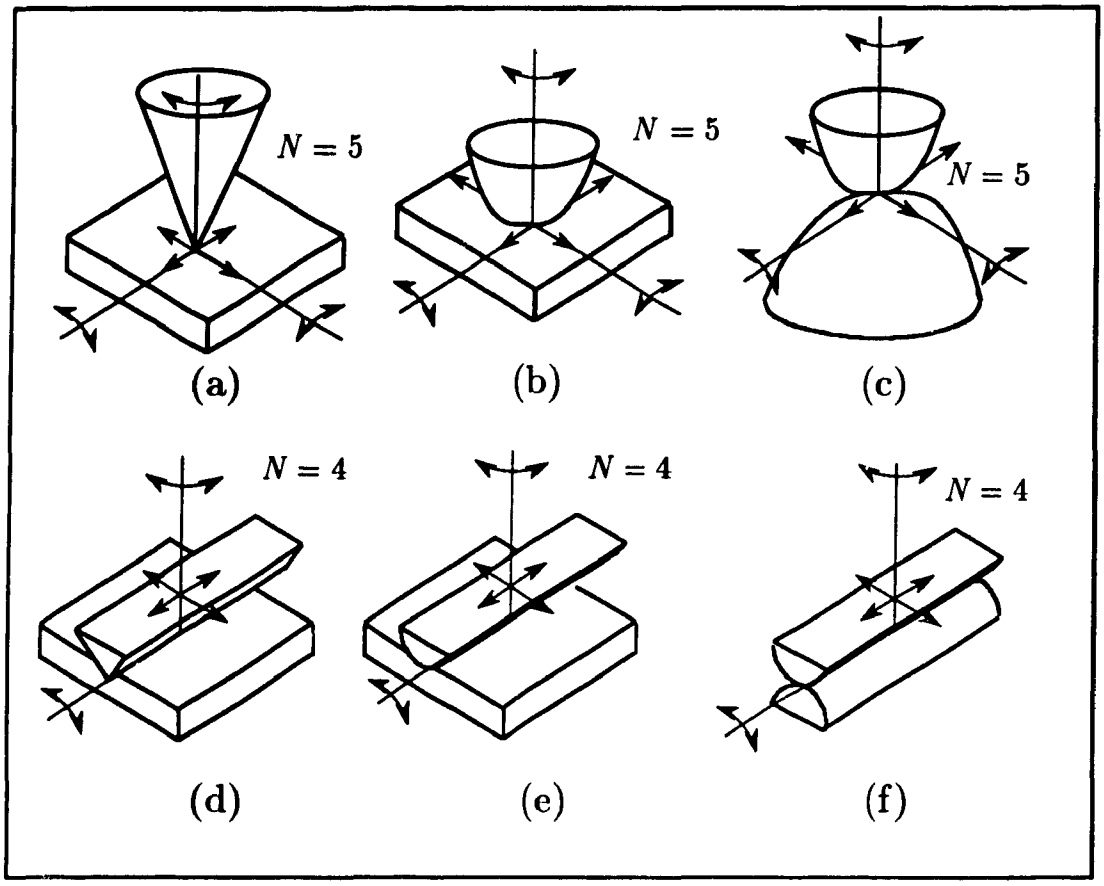

Figure 2.7 *Some Possible Types of Upper Kinematic Pairs*

particular direction of a kinematic pair, then that will be a constrained direction whenever magnitude of the force or torque along it is smaller than a certain critical value called the *breakaway* force or torque. Once the breakaway force or torque is exceeded, the ensuing motion adds to the degree of freedom of the kinematic pair. Figure 2.8(a) shows that the effect of the presence of friction on a fixed-point contact is a reduction in N of 2, as compared to the same contact without friction in Figure 2.7(a). Similarly, Figure 2.8(b) shows a 'soft-finger' contact (Mason and Salisbury, 1985) which has its degree of freedom further reduced to $N = 2$. Once again, because frictional contacts are widespread in the systems of interest, some effort will be directed toward ensuring that the constraints inherent in these contacts are satisfied.

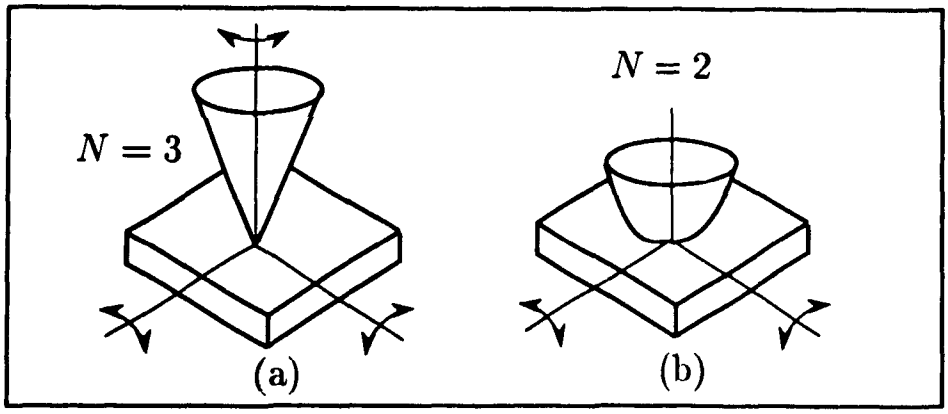

Figure 2.8 *Reduced Degree of Freedom Due to Friction*

2.3.1 Rolling Contact

Some of the kinematic pairs shown in Figure 2.7 involve a rolling contact point or line, and the constraints which they impose are nonholonomic (see e.g., Angeles, 1989). The analysis of these contacts tends to be more involved than that of non-rolling contacts since the constraint equations must be formulated in terms of contact velocities rather than positions. Furthermore, a nonholonomic constraint reduces the number of degrees of freedom of motion allowed at the contact but does not affect the dimension of the *configuration space* of the contact. This is in contrast to a holonomic constraint, which reduces both the number of degrees of freedom of motion allowed at the contact *and* the dimension of the configuration space. Cai (1988) presents a study of these contacts in which the motion of the contact point or line is analyzed in considerable detail.

In the present work, the position of the contact point at the fingertip/object or foot/ground contact is assumed to be known at all times and the distinction of whether the contact is rolling or not is therefore irrelevant. However, it is emphasized that if rolling contact exists, the techniques required to determine the position of the contact point will be substantially more complex than for non-rolling contacts.

We can now proceed to find how the coupling of links and joints affects the capacity of motion of the system as a whole.

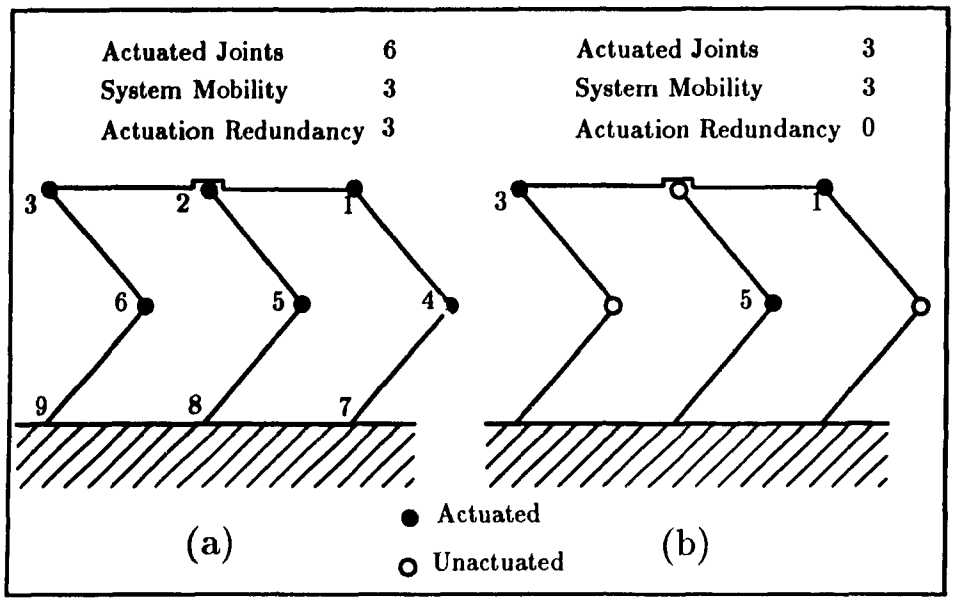

Figure 2.9 *A Planar Three-Legged Walking Machine*

2.4 Mobility

The mobility of a kinematic chain (Hunt, 1978) can be defined as the minimum number of independent variables necessary to specify the location of all links in the chain relative to a reference link. The choice of reference link does not affect the resulting mobility. A preliminary evaluation of the mobility of a kinematic chain can be found from the general mobility formula, generally attributed to Chebyshev, Grübler and/or Kutzbach (see e.g., Hunt, 1978):

$$M = d(b - g - 1) + \sum_{i=1}^{g} f_i \tag{2.2}$$

where M is the mobility of the kinematic chain, d is the degree-of-freedom of each unconstrained individual body (6 in 3-D; 3 in 2-D), b is the number of rigid bodies in the chain, g is the number of joints, and f_i is the number of degrees of freedom allowed by the i-th joint. For the machine in Figure 2.9(a), $d = 3$, $b = 8$, $g = 9$, since revolute joints have one degree of freedom $f_1 = \ldots = f_6 = 1$, and if the foot/ground contacts are modeled as revolute joints $f_7 = f_8 = f_9 = 1$—yielding $M = 3$.

The results of §2.1 to 2.3 are now applied to find particular forms of eq.(2.2)

for systems with a hybrid parallel/star topology. This is done by first decomposing the system into its constituent parallel and star subchains and finding the mobility of each of these. Firstly, we know that there are no cycles in the graph of the star subchain. Thus, writing eq.(2.1) in terms of b_s and g_s, the number of bodies and the number of joints in the star subchain, we have:

$$g_s - b_s + 1 = 0 \tag{2.3}$$

If we let $g'_s = \sum_{i=1}^{g_s} f_{si}$ where f_{si} is the number of degrees of freedom allowed by the i-th joint in the star subchain, and substitute eq.(2.3) into eq.(2.2), we obtain

$$M_s = g'_s \tag{2.4}$$

which clearly shows that the mobility of the star subchain is nothing but the total number of degrees of freedom allowed by its joints. Turning our attention now to the parallel subchain, we can rewrite eq.(2.1) in terms of b_p and g_p, the number of bodies and joints in the parallel subchain, as:

$$c = g_p - b_p + 1 \tag{2.5}$$

where c is now the number of *independent loops* in the subchain.

Once again, if we let $g'_p = \sum_{i=1}^{g_p} f_{pi}$ where f_{pi} is defined analogously to f_{si}, and substitute eq.(2.5) into (2.2), we obtain:

$$M_p = -dc + g'_p \tag{2.6}$$

For kinematic chains with topological graphs of the form shown in Figure 2.3(a), we can relate the number of independent loops in the chain to p, the number of paths between the two poles in the parallel subchain:

$$c = p - 1 \tag{2.7}$$

which, when substituted into Equation (2.6) yields:

$$M_p = d(1 - p) + g'_p \tag{2.8}$$

The total mobility of the hybrid kinematic chain can now simply be found as the sum of the mobilities of its constituent subchains, i.e.,

$$\begin{aligned} M &= M_s + M_p \\ &= g'_s + d(1-p) + g'_p \\ &= d(1-p) + g' \end{aligned} \tag{2.9}$$

where $g' = g'_s + g'_p$, the total number of degrees of freedom allowed by all the joints in the system.

Equation (2.9) has some interesting implications regarding how the system mobility is affected by the number of joint degrees of freedom of the individual robotic devices involved. Consider the 3-D example of a system of three manipulators with mobilities of 4, 5 and 6, respectively, all grasping the same object (i.e., $d = 6$, $p = 3$, $g' = 15$). Equation (2.9) shows that this system has a mobility of 3. That is, for each degree of freedom of less than d in each manipulator, the system's mobility is decreased by one, and for each degree of freedom greater than d in each manipulator, the system's mobility is increased by one. Thus, if $d = 6$, the system must have $g' = 6p$ in order to have a mobility of six. Figure 2.10 shows a graphical representation of eq.(2.9) for $d = 6$. However, as pointed out by Waldron (1966), the *relative* mobility between two particular links—called their *connectivity* in the present work—is often of greater importance than the mobility of the kinematic chain as a whole, and this will be treated in §2.5.

In a system with time-varying topology, the number of constituent bodies remains constant while the number of joints varies as contact is made or broken between the various bodies in the chain. Referring to eq.(2.9), it is apparent that if the number of paths between the two poles is increased by one—e.g., a leg, which was previously lifted, comes into contact with the ground—, the only way for the system's mobility to remain unchanged is for g' to be increased by d as a result of the new contact (i.e., the foot/ground contact must have d degrees of freedom). Since any non-trivial contact must have at most 5 degrees of freedom, the system's mobility will be reduced whenever a

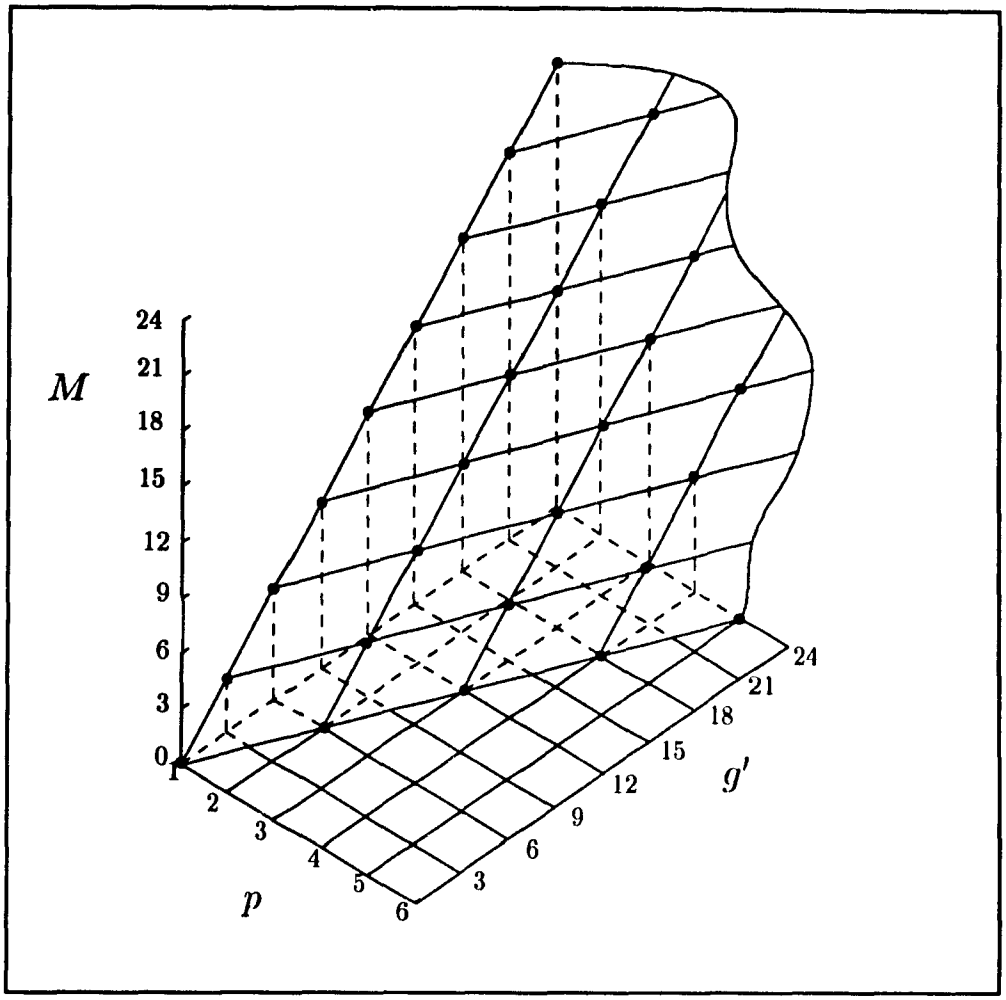

Figure 2.10 *Mobility of Cooperating Robotic Devices*

new path is formed in its parallel subchain. Conversely, whenever an existing path is broken, the system's mobility will be increased. Thus, we can conclude that *a system with time-varying topology will also have a time-varying mobility.*

It should be noted that the presence of frictional contacts can also cause a time-varying mobility of the system, this time *without* a change in the system's topology. More specifically, in §2.3 we showed that the number of degrees of freedom of a contact can vary depending on whether or not the breakaway force or torque is exceeded. Referring to eq.(2.9), this would correspond to a situation where p stays constant while g' varies.

The preceding treatment of a kinematic chain's structure and mobility has been

based solely on its topology. In fact, the kinematic chain's *geometry* can also have an effect on its mobility. More specifically the mobility of kinematic chain with a particular geometry (i.e., link lengths and joint axes orientations) may be greater than that predicted by the equations previously given. Using Hervé's (1978) classification, these kinematic chains are termed *paradoxical*—Bennett's mechanism (Hunt, 1978) probably being the most common example. Angeles and Gosselin (1988) have proposed a more comprehensive technique based on the kinematic chain's Jacobian matrix to determine its mobility. This approach will be applied to cooperating robotic devices in the next chapter when we deal with geometry and motion.

2.5 Connectivity

The connectivity of two bodies in a kinematic chain (Hunt, 1978) can be defined as the minimum number of independent variables necessary to specify the location of one body relative to the other. In practical terms, it may be of greater importance than the kinematic chain's mobility since it allows us to determine whether a given combination of robotic devices will be able to move their payload as desired. To illustrate this, consider the planar three-fingered hand shown in Figure 2.11(a) which has $p = 3$ and $g' = 9$ (the 6 planar revolute joints have one degree of freedom each, as do the 3 revolute contacts). Using eq.(2.9), we find that $M = 3$, which would appear to be sufficient to position and orient the object relative to the palm. However, due to the small number of joints in the left finger, the object only has a connectivity of 2 relative to the palm—since, if the angles of the joint and contact of the the left finger are specified, the object is fixed relative to the palm. Only the compensatingly large number of joints in the right finger allows the mobility of the system to be 3. By contrast, the planar hand shown in Figure 2.11(b) also has $p = 3$ and $g' = 9$ resulting in $M = 3$, but the homogeneous distribution of joints among the fingers in this case allows the connectivity of the object relative to the palm to be 3. Since, in general, we are interested in designing a system of cooperating robotic devices with a prescribed connectivity between the two poles, the determination of this

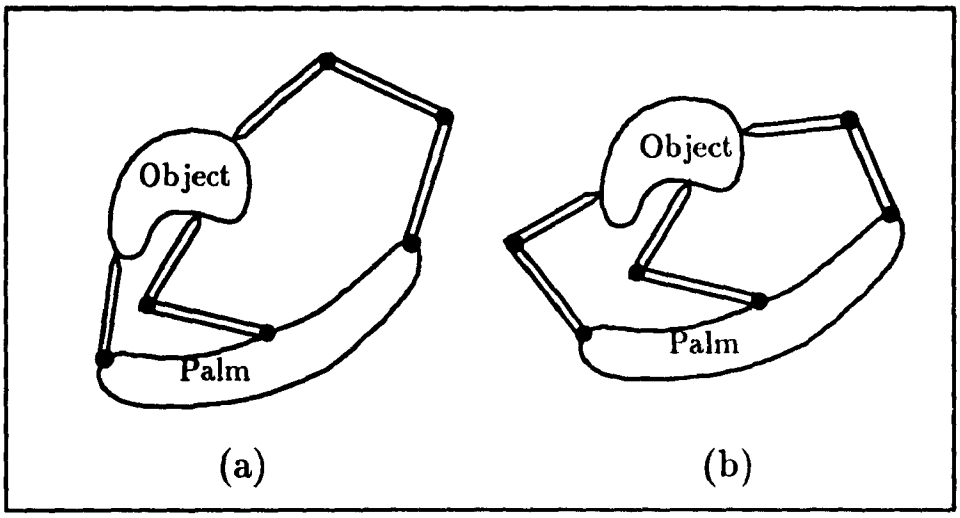

Figure 2.11 *Two Planar Three-Fingered Hands with $M = 3$*

quantity will be at least as important as that of the system's mobility

The relationship between mobility and connectivity is intuitive: the M independent variables necessary to specify the location of all links in the chain relative to a reference link can be partitioned into *a)* the independent variables necessary to specify the location of a second link relative to the reference link, and *b)* the independent variables necessary to specify the location of all other links relative to the reference link, once the second link has been fixed. The number of independent variables in *(a)* is the connectivity, while the number of variables in *(b)* is the mobility of the new kinematic chain formed when the reference link and the second link are considered as a single link. This relationship was noted in the context of mechanical hands by Salisbury and Craig (1982), but extends to more general chains. A suitable equation to determine the connectivity of two links would therefore be

$$C = M - M' \tag{2.10}$$

where M' is the mobility of the kinematic chain formed when the two links in question are treated as one.

Figure 2.12 shows the application of this procedure to the poles of the hybrid kinematic chain whose graph is shown in (a) to obtain the graph shown in (b). Considering

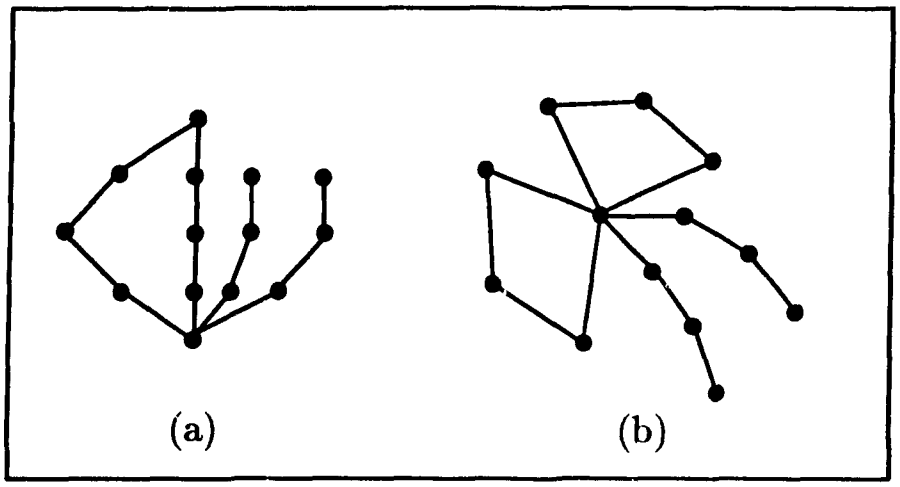

Figure 2.12 *Finding the Connectivity of a Hybrid Kinematic Chain*

the poles as one link results in a set of p uncoupled simple closed chains and q uncoupled simple open chains—where p and q are the number of paths between the two poles and the number of uncoupled serial chains in the original system, respectively. For each of the simple closed chains, the mobility can be found from eq.(2.6) with $c = 1$. Therefore we can write the following equation for the connectivity of the poles in a hybrid chain

$$C = M - \sum_{i=1}^{p} \max(0, -d + g'_{pi}) - \sum_{j=1}^{q} g'_{sj} \tag{2.11}$$

where g'_{pi} is the number of degrees of freedom allowed by all the joints of path i in the parallel subchain, while g'_{sj} is the number of degrees of freedom allowed by all the joints in serial subchain j. The function $\max(0, \ldots)$ is required to ensure that the 'negative degrees of freedom' which can be obtained from Chebyshev-type formulae are not included. The connectivity has two upper limits:

$$C \leq M, \qquad C \leq d \tag{2.12}$$

Equation (2.11) is useful in determining the dimensionality of relative motion possible between two links, and is of particular importance when considering the two poles of cooperating robotic systems. Thus, if we want to design a mechanical hand which can manipulate an object with six degrees of freedom relative to the palm, we must design it

such that $C = 6$. Another important use of eq.(2.11) is to determine the connectivity of the two poles when the actuators are driven. This will be addressed in §2.6.

Finally, it must be emphasized that eq.(2.11) is subject to the same limitations mentioned in the §2.4 for Chebyshev-type mobility equations since it considers only the topology of the chain and not its geometry. The next chapter will introduce other, more reliable, connectivity equations which consider the kinematic chain's geometry as well as its topology.

2.6 Actuation

The preceding discussion of mobility and connectivity implicitly assumed that all kinematic pairs in the kinematic chain were free to move according to their degree of freedom discussed in §2.3. In practice, some or all of the joints will be actuated in order to control the motion of the mechanism. The effect of an actuator at a joint, when it is powered, is to instantly specify that joint's position and remove the corresponding degree of freedom. If g_a actuators are installed, the mobility of the kinematic chain when these actuators are locked is denoted by M_a and found as

$$M_a = M - g_a \tag{2.13}$$

Analogously, a minimum of C actuators is necessary to fix the position and orientation of one pole relative to the other, and the connectivity of the two poles when the actuators are locked is denoted by C_a and found from

$$C_a = M - \sum_{i=1}^{p} \max(0, -m + g'_{pi} - g'_{pai}) - \sum_{j=1}^{q} (g'_{sj} - g'_{saj}) \tag{2.14}$$

where g'_{pai} and g'_{saj} are the number of actuated joint degrees of freedom in the i-th path between the two poles and in the j-th serial subchain, respectively.

The purpose of installing actuators in the kinematic chain is to control its configuration; we therefore want M_a, the mobility of the *actuated* chain, to be reduced

to zero when the actuators are powered. From eq.(2.13), this implies that a minimum of $g_a = M$ actuators are needed to fully fix the position and orientation of all links in a kinematic chain. Furthermore, since $M \geq C$, it is sufficient to reduce M_a to zero to also reduce C_a to zero.

It is therefore conventionally accepted to install M actuators in a kinematic chain since this will reduce its mobility to zero when the actuators are powered. In §2.4, we showed that kinematic chains with time-varying topology also have time-varying mobility. Thus, the approach taken for these systems is to install a number of actuators equal to the *maximum* mobility of the system. This maximum mobility occurs when all the subchains act as simple open chains—e.g., when the fingers of a hand are not grasping an object. As some of these subchains are closed, the mobility of the system will be reduced (c.f., §2.4) and, if all actuators remain powered, the system becomes *redundantly actuated* with a redundancy r of

$$r = g_a - M \tag{2.15}$$

2.6.1 Redundant Actuation

If we substitute eq.(2.9) into eq.(2.15), we find that

$$r = g_a - d(1-p) - g' \tag{2.16}$$

Since $g' = g_a + g_u$, where g_u is the number of unactuated joint degrees of freedom in the kinematic chain, we can write

$$r = d(p-1) - g_u \tag{2.17}$$

which shows that the actuation redundancy increases when there are fewer unactuated joint degrees of freedom. For example, if three manipulators rigidly grasp an object, we will have $d = 6, p = 3, g_u = 0$ yielding $r = 12$. By contrast, if three fingers hold an object, and we assume that the fingertip/object contacts are non-rolling contacts with friction, we find that $d = 6, p = 3, g_u = 9$, yielding $r = 3$.

As was just noted, redundant actuation will occur when the number of actuators in the system is chosen to satisfy requirements such as the degree of freedom required by the kinematic subchains within the larger chain—e.g., the fingers of a hand or the legs of a walking machine. Because the system has a time-varying topology, the individual subchains can form closed loops which have the potential to become redundantly actuated. As will be seen in greater detail in Chapter 4, one of the results of redundant actuation is that it causes the equations of motion of the system to become underdeterminate, and this is usually cited as the reason for needing to optimize the force distribution. In fact, the system could more easily be made determinate by 'feathering' the redundant actuators—i.e, not powering the redundant actuators while leaving the corresponding joint free to move—thereby avoiding the need to perform force optimization (such a technique is proposed in §3.5).

However, there are a number of benefits to be drawn from persisting in driving all available actuators. A glimpse of these advantages can be found in the field of structural engineering where the concept of static indeterminacy is closely related to redundant actuation. The principal difference between the two is that structural analysis usually deals with kinematic chains whose mobility is nonpositive--i.e., structures as opposed to mechanisms. One of the advantages of a statically indeterminate structure over a statically determinate one is that its internal forces and moments are reduced (see e.g., West, 1980). This allows a lighter structure to be built to carry the same loads since the stresses in the structure tend to be lower.

In fact, these benefits can be readily translated to redundantly actuated robotic systems. A brief example of this is given by the three-legged planar walking machine with a mobility of three shown in Figure 2.9. The particular geometry shown allows a decoupling of the vertical and lateral force systems. In Figure 2.9(a), where six actuators are driven, the machine has an actuation redundancy of three ($g_a = 6$; $M = 3$). In Figure 2.9(b), the machine has three driven actuators and is therefore not redundantly actuated ($g_a = 3$; $M = 3$). There is only one set of actuator torques and foot-ground contact forces which

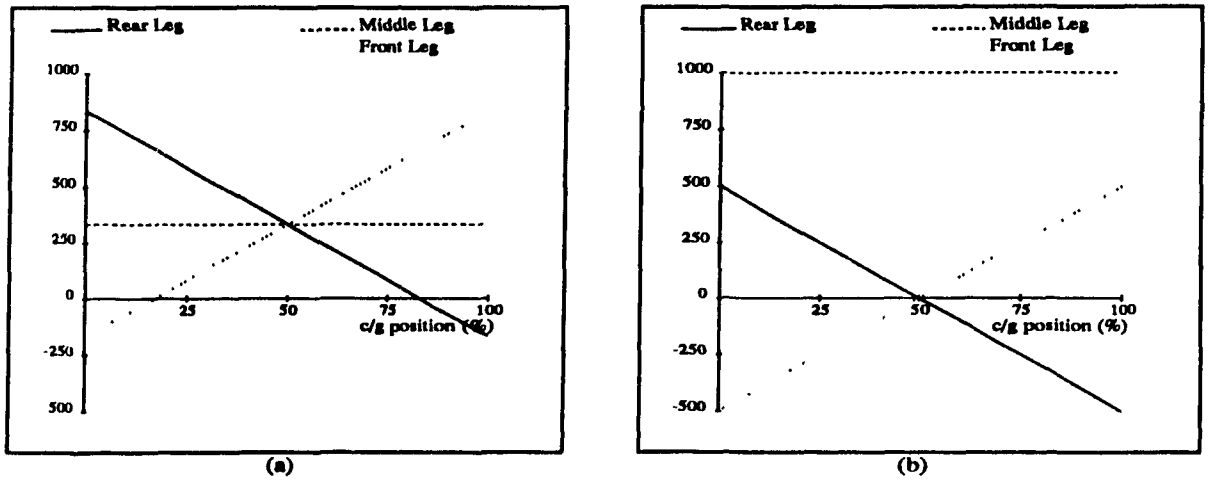

Figure 2.13 *Vertical Forces on the Legs of a Planar Walking Machine*

can satisfy the force balance equations of the non-redundantly-actuated walking machine. By contrast, the foot-ground contact forces of the redundantly-actuated walking machine can be reduced and homogenized, thus reducing chances of foot slippage. Figure 2.13(a) shows the variation of the vertical forces on the three legs using a solution which minimizes the norm of the vector of vertical contact forces as a load of 1000 N is moved along the body. Figure 2.13(b) shows the same forces when only actuators 1, 3 and 5 are driven. In both cases, negative foot/ground contact forces are generated because their non-negativity has not been accounted for. Other combinations of three driven actuators yield similar results. Furthermore, with the geometry shown, the machine is not controllable with certain combinations of only three actuators (e.g., 1, 2 and 3) because the actuated system will then be in a singular pose in which the geometry causes an increase in its mobility—a situation which will be analyzed in greater detail in Chapter 3.

Redundant actuation also allows greater safety in case of breakdown of individual actuators. If the mechanism is redundantly actuated, it can still be controlled if one or more actuators breaks down—up to the degree of actuation redundancy. Redundant actuation may also be applied to fixed-topology parallel-architecture robots in order to make them lighter and faster, though the advantage obtained through redundant actuation might be offset by the weight of the extra actuators.

Chapter 3

Kinematics and Statics

While the previous chapter dealt primarily with the kinematic *structure* of cooperating robotic devices, this chapter deals with their *motion*. The motion analysis serves as the underlying foundation for the dynamic analysis which follows in the next chapter. A large body of research exists relating to the kinematics of more conventional robotic systems such as the serial and parallel manipulators shown in Figure 3.1, and analogies may be drawn between the kinematics of those systems and the ones of present interest. This is particularly true of parallel manipulators, the kinematic structure of which bears a strong resemblance to that of the parallel subchain of cooperating robotic devices, with the important difference that it is not time-varying. Emphasis in this chapter is placed on the analysis of the parallel subchain since the star subchain can be simply analyzed using conventional techniques for serial manipulators.

The motion analysis which follows is decomposed into three parts. In §3.1, the position kinematics of serial and parallel manipulators is reviewed and that of cooperating robotic devices is analyzed. Sections 3.2 and 3.3 do the same for velocity and acceleration kinematics. Section 3.4 introduces mobility and connectivity criteria which consider not only the structure (as in Chapter 2), but also the geometry of the system. Finally, §3.5 extends the discussion begun in Chapter 2 regarding the number of actuators which need to be driven in order to control a system of cooperating robotic devices.

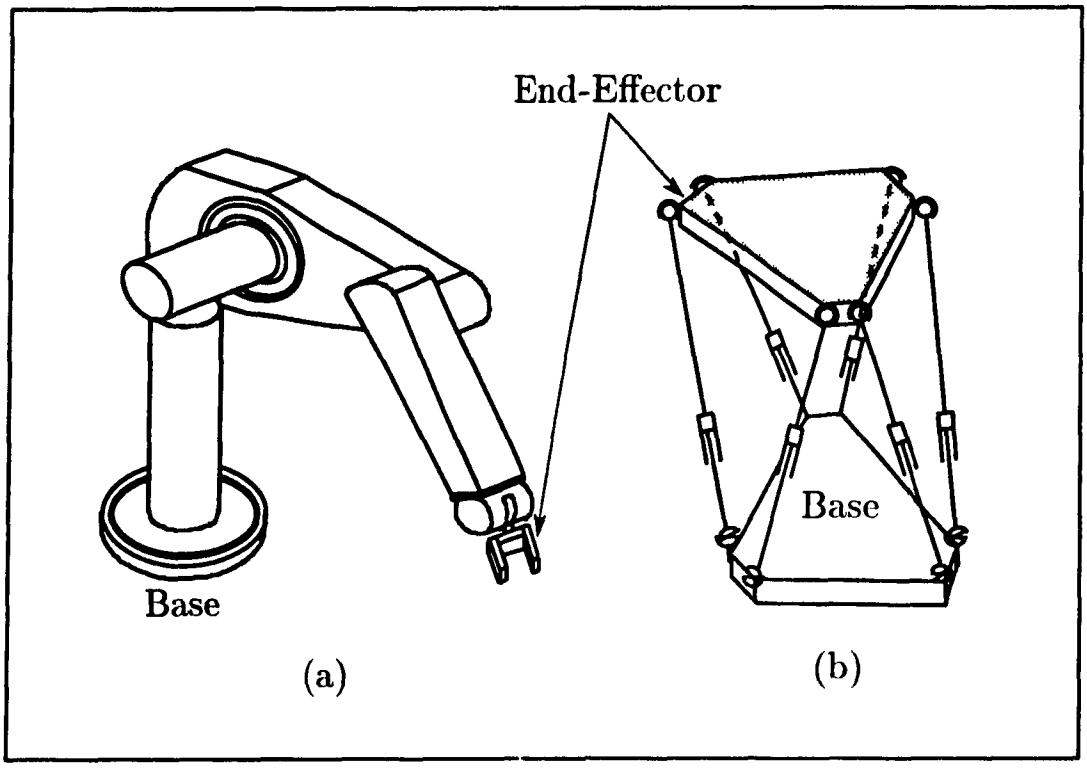

Figure 3.1 *A Serial and a Parallel Manipulator*

3.1 Position Analysis

A robotic manipulator composed of rigid links which are coupled by joints is characterized by two features: its *architecture* and its *configuration*. The architecture, normally quantified by the manipulator's Hartenberg-Denavit parameters (Hartenberg and Denavit, 1964), describes the fixed characteristics of the links' dimensions and geometry. The configuration, normally quantified by a set of *independent* joint coordinates denoted by the vector $\mathbf{q}$, describes the time-varying joint angles or lengths. Another commonly-used description of a manipulator's configuration is the *pose* (position and orientation) of a reference frame fixed to its end-effector relative to an inertial reference frame, denoted by the vector $\mathbf{x}$. The dimension of $\mathbf{x}$, which is always greater than or equal to the dimension of the task space—i.e., 6 for general 3-D motion—, depends on the method used to describe rotations (Spring, 1986). For example, the use of Euler parameters or linear invariants results in a vector $\mathbf{x}$ of dimension 7. However, irrespective

of the method chosen, $\mathbf{x}$ will only have as many *independent* components as the dimension of the task space. The mapping between the two descriptions of the manipulator's configuration, which is not necessarily one-to-one, is the focus of attention in position kinematics. Thus, the following topics will be addressed:

1. **Forward Kinematics:** Given the joint coordinates, $\mathbf{q}$, find the Cartesian coordinates of the end-effector, $\mathbf{x}$,
2. **Inverse Kinematics:** Given the Cartesian coordinates of the end-effector $\mathbf{x}$, find the joint coordinates, $\mathbf{q}$, and
3. **Number of Solutions:** The number of solutions which exist for the forward and inverse kinematics problems.

3.1.1 Forward Kinematics

The joint coordinates of a serial manipulator, denoted by the vector $\mathbf{q}_s$ of dimension M, are the joint variables of its M joints, all of which are actuated (recall that the mobility of a serial manipulator is equal to its total number of joint degrees of freedom). If these are specified, the corresponding *unique* set of Cartesian coordinates for the end-effector, denoted by the vector $\mathbf{x}_s$ which has six independent components, can be found from a system of equations of the form:

$$\mathbf{x}_s = \mathbf{f}_s(\mathbf{q}_s) \tag{3.1}$$

This problem is straightforward and requires little computation.

In the case of parallel manipulators, the joint coordinates, denoted by the vector $\mathbf{q}_p$ of dimension M, normally refer to the M *actuated* joint variables—e.g., the six leg lengths of the platform shown in Figure 3.1(b). This forward kinematics problem is considerably more complex than that for a serial manipulator since only one joint variable on each path between the two poles is usually actuated. For a parallel manipulator of

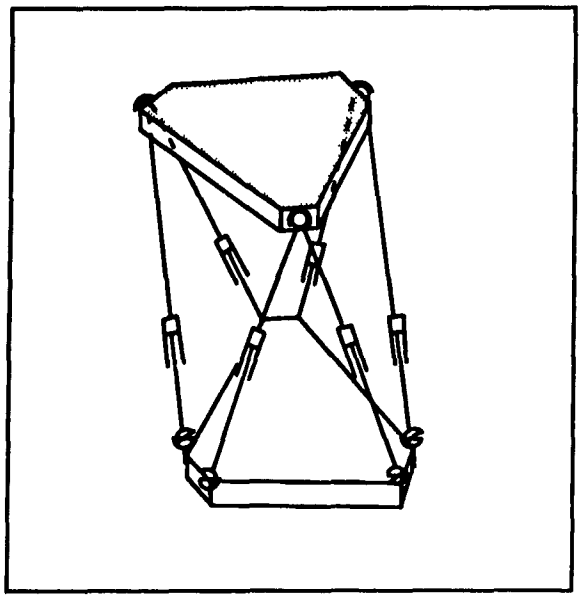

Figure 3.2 *A Parallel Manipulator with Coincident Pairs of Spherical Joints*

general architecture, this amounts to solving a system of nonlinear equations of the form:

$$\mathbf{f}_p(\mathbf{q}_p, \mathbf{x}_p) = \mathbf{0} \tag{3.2}$$

for $\mathbf{x}_p$, the vector of Cartesian coordinates of the parallel manipulator's end-effector. In general, this problem can only be solved using iterative methods, such as a Newton-Raphson technique (Press *et al.*, 1986), but for some particular architectures, closed-form solutions may be found. The number of solutions to this problem has only been determined for these particular architectures. For example, the parallel manipulator with coincident pairs of spherical joints shown in Figure 3.2 is known to have 16 solutions to its direct kinematics problem (Merlet, 1990).

It is emphasized that the joint coordinates of both serial and parallel manipulators are a set of *M independent, actuated* joint variables. By contrast, cooperating robotic devices contain more than M actuated joints, and so, the actuated joint variables are *not* independent. For clarity, the precise statement of the forward kinematics problem in cooperating robotic devices can be stated as follows:

> *Given* $\mathbf{q}_c$, *the vector of actuated joint variables, find the pose of a reference frame fixed in the moving pole, denoted by its vector of Cartesian coordinates* $\mathbf{x}_c$.

The constraints which the dependent actuated joint variables must satisfy in order to be a valid set are called the *loop closure equations*. These equations can be written in terms of the forward kinematics problem for each of the individual robotic devices as

$$\mathbf{f}_{s1}(\mathbf{q}_{s1}) = \ldots = \mathbf{f}_{sp}(\mathbf{q}_{sp}) \tag{3.3}$$

where $\mathbf{f}_{si}$ and $\mathbf{q}_{si}$ represent the forward kinematics functions and the joint coordinates of the i-th robotic device, respectively. The closure equations specify that the pose of an arbitrary reference frame in the moving pole is the same, irrespective of the path used to obtain it. It should be apparent that *all* the joint variables in cooperating robotic devices are actuated except for those at the contact between the individual robotic devices and a grasped object or the ground, *when that contact is passive*. For clarification, Figure 3.3(a) shows a three-legged planar walking machine with passive foot-ground contacts. The joint angles $\theta_1, \ldots, \theta_6$ are known, while θ_7, θ_8 and θ_9 are not. A similar statement holds for the joint angles of the three-fingered planar mechanical hand shown in Figure 3.3(b). However, the three planar cooperating manipulators shown in Figure 3.3(c) have no passive contacts—*all* the joints are actuated, and all the joint variables can be assumed known. Thus, the forward kinematics problem of cooperating robotic devices is considerably easier than that for a parallel manipulator since most, if not all the joint variables along each path, are prescribed.

In the case of cooperating manipulators where all the joints are actuated, the Cartesian coordinates of the reference frame in the moving pole can be found by simply solving the forward kinematics of any one of the p paths between the two poles, i.e.,

$$\mathbf{x}_c = \mathbf{f}_{si}(\mathbf{q}_{si}), \qquad i = 1, \ldots, p \tag{3.4}$$

Since eq.(3.4) yields a unique $\mathbf{x}_c$, the solution to the forward kinematics problem of cooperating manipulators is unique.

In the case of mechanical hands and walking machines, where some of the

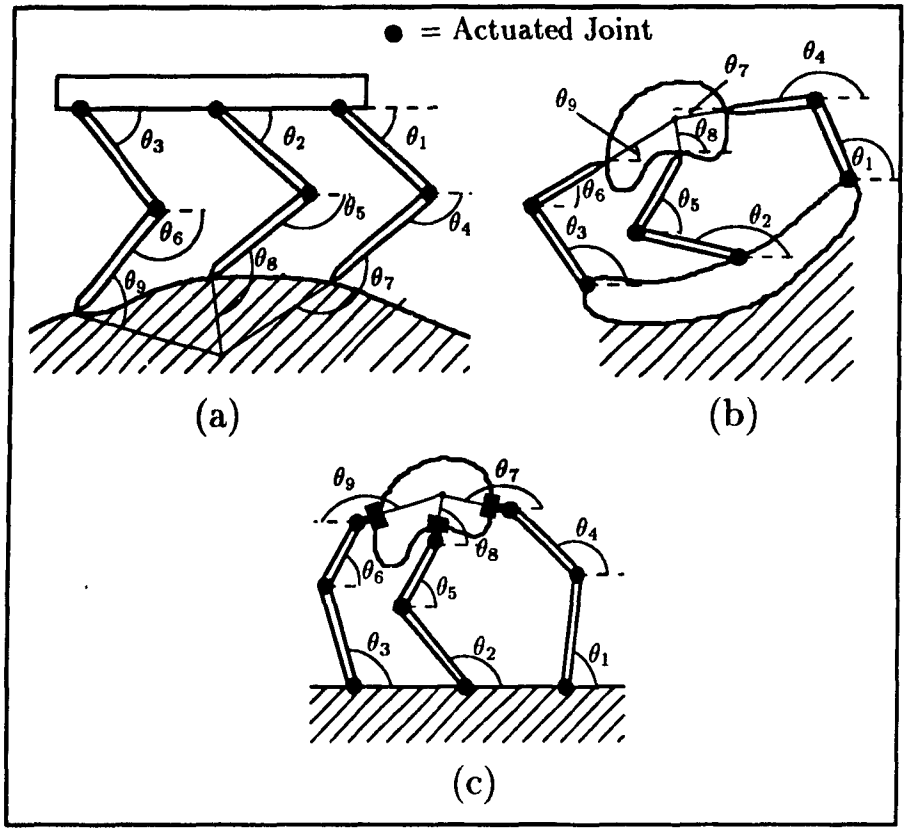

Figure 3.3 *Joint Variables of Some Planar Cooperating Robotic Devices*

contacts are passive, the *position* of the finger/object contact points or the foot/ground contact points can be found using the forward kinematics equations of the individual robotic devices of the form

$$\mathbf{p}_i = \mathbf{f}_{si}(\mathbf{q}_{si}), \qquad i = 1, \ldots, p \tag{3.5}$$

where $\mathbf{q}_{si}$ denotes the vector of actuated joint variables in the i-th path. Equation (3.5) will yield a *unique* solution for the positions of the contact points. Once the positions of three *noncollinear* points are known, the pose of the reference frame in the moving pole can be calculated *uniquely* using, for example, the method outlined by Angeles (1986). Thus, we can say that *the solution to the direct kinematics problem of cooperating robotic manipulators is unique.*

3.1.2 Inverse Kinematics

The inverse kinematics problem for a serial manipulator consists of rewriting eq.(3.1) as

$$\mathbf{f}_s(\mathbf{q}_s) - \mathbf{x}_s = \mathbf{0} \tag{3.6}$$

and solving this system of nonlinear equations for $\mathbf{q}_s$ with $\mathbf{x}_s$ prescribed.

For serial manipulators of general architecture, eq.(3.6) cannot be solved in closed form—iterative routines must be used—and there may be up to 16 different solutions (Tsai and Morgan, 1985). When the manipulator has particular features in its architecture, solution of the problem in closed form may be possible--e.g., a manipulator with three consecutive joint axes intersecting at a point has, at most, 8 solutions to its inverse kinematics problem which can be found in closed form (Pieper, 1968). Finally, the simpler the architecture of a robot, the fewer distinct solutions to its inverse kinematics problem will exist.

In the case of parallel manipulators, when $\mathbf{x}_p$, the Cartesian coordinates of the end-effector reference frame are known, the Cartesian coordinates of reference frames at the *p attachment points* at which the legs are attached to the end-effector can be easily calculated since all frames are in the same rigid body. If we assume that the Cartesian coordinates of the end-effector are composed of the vector $\mathbf{p}$ which describes the position of the origin of the end-effector reference frame relative to an inertial frame, and the elements of the rotation matrix $\mathbf{Q}$ which represents the orientation of the end-effector frame relative to the inertial frame, we can write

$$\mathbf{p}_i = \mathbf{p} + \mathbf{s}_i, \qquad i = 1, \ldots, p \tag{3.7a}$$

$$\mathbf{Q}_i = \mathbf{Q}\mathbf{Q}_i^r, \qquad i = 1, \ldots, p \tag{3.7b}$$

where $\mathbf{s}_i$ denotes a vector from the end-effector frame to the i-th attachment point and $\mathbf{Q}_i^r$ is the rotation matrix which represents the orientation of the attachment point frame relative to the end-effector frame. The Cartesian coordinates of the i-th attachment

point, $\mathbf{x}_i$, consist of the vector $\mathbf{p}_i$ and the elements of the rotation matrix $\mathbf{Q}_i$, where the former represents a vector from the inertial frame to the attachment point while the latter represents the orientation of the attachment point frame relative to the inertial frame. Once these are known, the inverse kinematics problem for a parallel manipulator is nothing but p independent serial inverse kinematics problems—one for each path between the two poles. As mentioned above, this solution is straightforward when the legs have particular architectural features, and complex when the legs have a general architecture. Thus, the usual claim that the inverse kinematics problem for a parallel manipulator is straightforward (Merlet, 1990) implicitly assumes that the leg geometry is simple. For all common parallel manipulators, this assumption is verified and the inverse kinematics problem reduces to

$$q_{pi} = g_i(\mathbf{x}_i), \qquad i = 1, \ldots, p \tag{3.8a}$$

where q_{pi} is the i-th component of $\mathbf{q}_p$. Equation (3.8a) can be written vectorially as

$$\mathbf{q}_p = \mathbf{g}(\mathbf{x}_i) = \mathbf{g}_p(\mathbf{x}_p) \tag{3.8b}$$

and has a unique solution. For manipulators which have a solution to their inverse kinematics problem of this form, eq.(3.2) for the direct kinematics problem can therefore be rewritten as

$$\mathbf{f}_p(\mathbf{q}_p, \mathbf{x}_p) = \mathbf{g}_p(\mathbf{x}_p) - \mathbf{q}_p = \mathbf{0} \tag{3.9}$$

Since, when the leg architecture is general, there may be up to 16 inverse kinematic solutions for each parallel path between the two poles and the solution for each path is independent of that for the other paths, there may be up to 16^p solutions (Gosselin, 1988) to the inverse kinematic problem of a parallel manipulator with a general leg architecture.

The inverse kinematics problem of the parallel subchain of cooperating robotic devices is similar to that for parallel manipulators, and can be stated as follows:

Given the Cartesian coordinates, $\mathbf{x}_c$, of a reference frame fixed in the moving pole find the joint coordinates of the p parallel paths.

In the case of cooperating manipulators, the inverse kinematics problem entails rewriting eq.(3.4) as

$$\mathbf{f}_{si}(\mathbf{q}_{si}) - \mathbf{x}_c = \mathbf{0}, \qquad i = 1, \ldots, p \tag{3.10}$$

and solving for $\mathbf{q}_{si}$ with $\mathbf{x}_c$ prescribed. As such, this problem is nothing but p times the inverse kinematics problem of a serial manipulator, and its complexity will be dependent on the architecture of the manipulators, as discussed above. For a system of manipulators with fully general architecture, there will again be, at most, 16^p possible solutions.

In the case of mechanical hands, the position vector $\mathbf{p}_i$ of the i-th finger/object contact point can be found from

$$\mathbf{p}_i = \mathbf{p} + \mathbf{s}_i, \qquad i = 1, \ldots, p \tag{3.11}$$

where $\mathbf{p}$ is the position vector of the origin of the frame in the moving pole, while $\mathbf{s}_i$ is a vector from the origin of that frame to the i-th finger/object contact point. From $\mathbf{p}_i$, the inverse kinematics problem is solved for each parallel path by solving

$$\mathbf{f}_{si}(\mathbf{q}_{si}) - \mathbf{p}_i = \mathbf{0}, \qquad i = 1, \ldots, p \tag{3.12}$$

for $\mathbf{q}_{si}$. For walking machines, the analysis is identical except that the finger/object contact points become the foot/ground contact points and the moving pole becomes the fixed pole. Since eq.(3.12) represents only a *positioning* inverse kinematics problem, it can be solved in closed form as the solution of, at most, a quartic polynomial having 4 distinct roots. The inverse kinematics problem of a mechanical hand or walking machine with fingers or legs of a general architecture therefore has, at most, 4^p possible solutions.

3.2 Velocity Analysis

The velocity analysis of robotic devices entails a determination of the mapping between the joint rates, $\dot{\mathbf{q}}$, and the *twist* of the end effector, $\mathbf{t}$, which is defined as

$$\mathbf{t} \equiv \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix} \tag{3.13}$$

where $\boldsymbol{\omega}$ is the angular velocity vector and $\mathbf{v}$ is the translational velocity vector; both of dimension 3. These mappings are considerably simpler than the functions used in position kinematics since they are linear. Once again, we are concerned with a forward and an inverse problem, depending on which of the two vectors ($\dot{\mathbf{q}}$ or $\mathbf{t}$) is prescribed and which is unknown.

The forward velocity kinematics problem for serial manipulators entails finding the manipulator's Jacobian which maps actuated joint rates into end-effector twist, i.e.,

$$\mathbf{t}_s = \mathbf{J}_s \dot{\mathbf{q}}_s \tag{3.14}$$

where

$$\mathbf{J}_s = \mathbf{J}_s(\mathbf{q}_s) = \frac{\partial \mathbf{t}_s}{\partial \dot{\mathbf{q}}_s} \tag{3.15}$$

When the i-th joint of the manipulator is revolute, the corresponding column of the Jacobian is (Whitney, 1972)

$$\mathbf{j}_{si} = \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_i \times \mathbf{r}_i \end{bmatrix} \tag{3.16a}$$

where $\mathbf{e}_i$ represents a unit vector aligned with the i-th joint axis, while $\mathbf{r}_i$ represents a vector from the origin of a reference frame fixed to the i-th link to the origin of a reference frame fixed to the end-effector. It should be noted that $\mathbf{j}_{si}$ is nothing but the vector of Plücker coordinates of the i-th joint axis and origin as seen from the origin of the end-effector reference frame. If the i-th joint is prismatic, the corresponding column of the Jacobian is

$$\mathbf{j}_{si} = \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_i \end{bmatrix} \tag{3.16b}$$

The manipulator's Jacobian can be useful when solving the inverse position kinematics problem iteratively. The latter problem consists of solving eq.(3.6) which can be rewritten as follows:

$$\mathbf{g}_s(\mathbf{q}_s, \mathbf{x}_s) = \mathbf{f}_s(\mathbf{q}_s) - \mathbf{x}_s = \mathbf{0} \tag{3.17}$$

A Newton-Raphson technique solves eq.(3.17) using the iterative formula

$$\mathbf{q}_s|_{n+1} = \mathbf{q}_s|_n - \left(\frac{\partial \mathbf{g}_s}{\partial \mathbf{q}_s}\right)^{-1}\Bigg|_n \mathbf{g}_s(\mathbf{q}_s|_n, \mathbf{x}_s) \tag{3.18}$$

where $\mathbf{q}_s|_i$ is the i-th iterate for the solution. Thus, in order to use eq.(3.18), we require

$$\mathbf{G}_s|_n \equiv \left(\frac{\partial \mathbf{g}_s}{\partial \mathbf{q}_s}\right)\Bigg|_n = \left(\frac{\partial \mathbf{f}_s}{\partial \mathbf{q}_s}\right)\Bigg|_n \tag{3.19}$$

where the matrix $\mathbf{G}_s$ is nothing but a linear transformation of $\mathbf{J}_s$. Anderson and Angeles (1989) have shown that, when $\mathbf{x}_s$ is the vector of dimension 7 consisting of the three components of the translational position of the end-effector and the four components of the linear invariants of $\mathbf{Q}$, the rotation matrix of the end-effector, this transformation can be written as

$$\mathbf{G}_s = \mathbf{H}\mathbf{J}_s \tag{3.20a}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{1}_3 \mathrm{tr}(\mathbf{Q}) - \mathbf{Q} & \mathbf{0}_3 \\ \mathrm{vect}(\mathbf{Q}) & \mathbf{0}^T \\ \mathbf{0}_3 & \mathbf{1}_3 \end{bmatrix} \tag{3.20b}$$

where $\mathbf{0}_3$ and $\mathbf{1}_3$ are the 3×3 zero and identity matrices, respectively, and $\mathbf{0}$ is a zero vector of dimension 3.

There is much less consistency in the literature regarding the definition of the Jacobian of parallel manipulators. One approach is to proceed analogously to the convention used for serial manipulators. That is, the vector $\dot{\mathbf{q}}_p$ represents the vector of *actuated* joint velocities, while the matrix $\mathbf{J}_p$ maps $\dot{\mathbf{q}}_p$ into $\mathbf{t}_p$ as follows

$$\mathbf{t}_p = \mathbf{J}_p \dot{\mathbf{q}}_p \tag{3.21}$$

A second convention often encountered (Gosselin, 1988) defines the manipulator's Jacobian, $\mathbf{J}'_p$ to satisfy

$$\dot{\mathbf{q}}_p = \mathbf{J}'_p \mathbf{t}_p \tag{3.22a}$$

where,

$$\mathbf{J}'_p = \frac{\partial \dot{\mathbf{q}}_p}{\partial \mathbf{t}_p} = \mathbf{J}_p^{-1} \tag{3.22b}$$

This second definition of the Jacobian is useful when solving the direct position kinematics problem of parallel manipulators using the Newton Raphson technique. If we follow a procedure analogous to that for the inverse position kinematics of serial manipulators, the iterative solution of eq.(3.9) will require the use of a matrix

$$\mathbf{G}_p|_n \equiv \left(\frac{\partial \mathbf{g}_p}{\partial \mathbf{x}_p}\right)\Bigg|_n \tag{3.23}$$

which is a linear mapping of the Jacobian matrix $\mathbf{J}'_p$. That mapping will of course change depending on the convention used to describe the rotations in the vector $\mathbf{x}_p$.

Finally, a third convention has been found useful when determining the mobility of kinematic chains with closed loops, such as parallel manipulators (Angeles, 1989). This Jacobian, denoted by $\mathbf{J}''_p$, satisfies

$$\mathbf{J}''_p \dot{\mathbf{q}}'_p = \mathbf{0} \tag{3.24}$$

where $\dot{\mathbf{q}}'_p$ is a vector of *all* the joint rates rather than just the actuated ones.

We can now proceed to define certain conventions for cooperating robotic devices. In these systems, neither the forward nor the inverse position kinematics problem requires iterative solution, and so, there is no reason to prefer a Jacobian of the form given by eq.(3.22a) over one of the form given by eq.(3.21). We therefore define the Jacobian of a system of cooperating robotic devices to satisfy

$$\mathbf{t}_c = \mathbf{J}_c \dot{\mathbf{q}}_c \tag{3.25}$$

where $\dot{\mathbf{q}}_c$ is the vector of the *actuated* joint rates, while $\mathbf{t}_c$ is the twist of a reference frame in the moving pole. Equation (3.25) allows us to solve the forward velocity kinematics problem—i.e., given the actuated joint rates, find the end-effector twist. Just as the actuated joint coordinates were dependent, so are the actuated joint rates, which must satisfy

$$\mathbf{J}_{s1} \dot{\mathbf{q}}_{s1} = \ldots = \mathbf{J}_{sp} \dot{\mathbf{q}}_{sp} \tag{3.26}$$

where $\dot{\mathbf{q}}_{si}$ represents *all* the joint rates in path i and $\mathbf{J}_{si}$ is the serial Jacobian of the i-th path. In the case of cooperating manipulators, it is straightforward to relate $\mathbf{J}_c$ to the

serial Jacobians of the individual paths between the two poles. If we consider each path as a serial manipulator, we can write eq.(3.14) for each path as

$$\mathbf{t}_c = \mathbf{J}_{si}\dot{\mathbf{q}}_{si}, \qquad i = 1, \ldots, p \tag{3.27}$$

If we add all these equations, we find

$$\sum_{i=1}^{p} \mathbf{J}_{si}\dot{\mathbf{q}}_{si} = p\mathbf{t}_c \tag{3.28}$$

In matrix notation, this can be written in the form of eq. (3.25) with

$$\mathbf{J}_c = \frac{1}{p}\begin{bmatrix}\mathbf{J}_{s1} & \mathbf{J}_{s2} & \ldots & \mathbf{J}_{sp}\end{bmatrix}, \qquad \dot{\mathbf{q}}_c = \begin{bmatrix}\dot{\mathbf{q}}_{s1} \\ \vdots \\ \dot{\mathbf{q}}_{sp}\end{bmatrix} \tag{3.29}$$

When solving the inverse velocity kinematics problem (i.e., given the twist of the moving pole, $\mathbf{t}_c$, find the vector of actuated joint rates, $\dot{\mathbf{q}}_c$), it is important *not* to attempt to solve eq.(3.25) for $\dot{\mathbf{q}}_c$ since $\mathbf{J}_c$ is not square. The inverse velocity kinematics problem is better solved using

$$\dot{\mathbf{q}}_{si} = \mathbf{J}_{si}^{-1}\mathbf{t}_c \qquad i = 1, \ldots, p \tag{3.30}$$

where the inversion shown is used strictly for notational simplicity and need not be performed explicitly. The vector of actuated joint rates can then be obtained as

$$\dot{\mathbf{q}}_c = \mathbf{P}\begin{bmatrix}\dot{\mathbf{q}}_{s1} \\ \vdots \\ \dot{\mathbf{q}}_{sp}\end{bmatrix} = \mathbf{P}\begin{bmatrix}\mathbf{J}_{s1}^{-1} \\ \vdots \\ \mathbf{J}_{sp}^{-1}\end{bmatrix}\mathbf{t}_c \tag{3.31}$$

where $\mathbf{P}$ is a matrix which projects the vector of all joint rates into the vector of actuated joint rates. It consists of columns whose entries are either all zeros (corresponding to the unactuated joint rates) or all zeros except for one unity entry.

The second definition of the manipulator's Jacobian, which will be useful to determine the mobility of cooperating robotic devices, is one which is identical in form to the third definition for parallel manipulators outlined above, namely

$$\mathbf{J}_c'\dot{\mathbf{q}}_c' = \mathbf{0} \tag{3.32}$$

where $\dot{\mathbf{q}}'_c$ is the vector of *all* the joint rates in the parallel subchain. It is straightforward to relate $\mathbf{J}'_c$ to the serial Jacobians of the individual paths between the two poles. Taking the difference of eq.(3.27) for i and $i+1$:

$$\mathbf{J}_{si}\dot{\mathbf{q}}_{si} - \mathbf{J}_{s,i+1}\dot{\mathbf{q}}_{s,i+1} = \mathbf{0}, \qquad i = 1, \ldots, p-1 \tag{3.33}$$

If we assemble these pairs in matrix form, we obtain eq.(3.32) with

$$\mathbf{J}'_c = \begin{bmatrix} \mathbf{J}_{s1} & -\mathbf{J}_{s2} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{s2} & -\mathbf{J}_{s3} & \ldots & \mathbf{0} & \mathbf{0} \\ \vdots & & & \ddots & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{J}_{s,p-1} & -\mathbf{J}_{sp} \end{bmatrix}, \qquad \dot{\mathbf{q}}'_c = \begin{bmatrix} \dot{\mathbf{q}}_{s1} \\ \vdots \\ \dot{\mathbf{q}}_{sp} \end{bmatrix} \tag{3.34}$$

3.3 Acceleration Analysis

The acceleration kinematics of cooperating robotic devices is obtained from the mapping between joint rates and end-effector twist in a straightforward manner. Taking the time derivative of eq.(3.25), we obtain

$$\dot{\mathbf{t}}_c = \begin{bmatrix} \dot{\boldsymbol{\omega}}_c \\ \dot{\mathbf{v}}_c \end{bmatrix} = \dot{\mathbf{J}}_c\dot{\mathbf{q}}_c + \mathbf{J}_c\ddot{\mathbf{q}}_c \tag{3.35}$$

where $\dot{\boldsymbol{\omega}}_c$ and $\dot{\mathbf{v}}_c$ are vectors of dimension 3 denoting the angular and translational accelerations of the reference frame in the moving pole. Equation (3.35) can be used to solve the forward acceleration kinematics problem—i.e., given the actuated joint accelerations $\ddot{\mathbf{q}}_c$, find the twist rate $\dot{\mathbf{t}}_c$ of the end-effector. Once again, the actuated joint accelerations are not independent since they must satisfy

$$\dot{\mathbf{J}}_{s1}\dot{\mathbf{q}}_{s1} + \mathbf{J}_{s1}\ddot{\mathbf{q}}_{s1} = \ldots = \dot{\mathbf{J}}_{sp}\dot{\mathbf{q}}_{sp} + \mathbf{J}_{sp}\ddot{\mathbf{q}}_{sp} \tag{3.36}$$

In the case of cooperating manipulators, eq.(3.29) can be used to obtain a relationship between $\dot{\mathbf{J}}_c$ and the time derivative of the constituent serial path Jacobians:

$$\dot{\mathbf{J}}_c = \frac{1}{p}\begin{bmatrix} \dot{\mathbf{J}}_{s1} & \ldots & \dot{\mathbf{J}}_{sp} \end{bmatrix} \tag{3.37}$$

where $\dot{\mathbf{J}}_{si}$ can be found by differentiating $\mathbf{J}_{si}$ as given by eqs.(3.16a) and (3.16b).

Once again, since $\mathbf{J}_c$ is not square, no attempt should be made to solve eq.(3.35) for $\ddot{\mathbf{q}}_c$ when solving the inverse acceleration kinematics problem—i.e., given the end-effector twist rate, $\dot{\mathbf{t}}_c$, find the vector of actuated joint accelerations, $\ddot{\mathbf{q}}_c$. This problem is better solved using

$$\ddot{\mathbf{q}}_{si} = \mathbf{J}_{si}^{-1}(\dot{\mathbf{t}}_c - \dot{\mathbf{J}}_{si}\dot{\mathbf{q}}_{si}) \qquad i = 1, \ldots, p \tag{3.38}$$

where, once again, the inversion shown should not be performed explicitly. The vector of actuated joint accelerations can then be obtained as

$$\ddot{\mathbf{q}}_c = \mathbf{P}\begin{bmatrix} \ddot{\mathbf{q}}_{s1} \\ \vdots \\ \ddot{\mathbf{q}}_{sp} \end{bmatrix} = \mathbf{P}\begin{bmatrix} \mathbf{J}_{s1}^{-1}(\dot{\mathbf{t}}_c - \dot{\mathbf{J}}_{s1}\dot{\mathbf{q}}_{s1}) \\ \vdots \\ \mathbf{J}_{sp}^{-1}(\dot{\mathbf{t}}_c - \dot{\mathbf{J}}_{sp}\dot{\mathbf{q}}_{sp}) \end{bmatrix} \tag{3.39}$$

3.4 Mobility and Connectivity Revisited

In §2.4, where we dealt with the mobility of cooperating robotic devices, it was noted that eq.(2.2) only treats the topology of the kinematic chain with no consideration of its geometry. However, geometrical considerations may affect a system's mobility and must be considered. Angeles (1989) describes methods to determine the mobility of kinematic chains which consider both the chain's topology and its geometry. For example, the mobility of a simple open kinematic chain such as one of the q serial chains which make up the star subchain would be found from:

$$\overline{M}_{si} = \dim[\text{range}(\mathbf{J}_{si})] \tag{3.40}$$

The mobility of the complete star subchain would then be found from

$$\overline{M}_s = \sum_{i=1}^{q} \overline{M}_{si} \tag{3.41}$$

The mobility of complex closed kinematic chains such as the parallel subchain of a system of cooperating robotic devices is then given as (Angeles, 1989)

$$\overline{M}_p = \dim[\mathcal{N}(\mathbf{J}_c')] \tag{3.42}$$

where $\mathbf{J}'_c$ was defined by eq.(3.32) and $\mathcal{N}(\star)$ represents the nullspace of the matrix in the brackets. In most cases, the mobility found from eq.(3.42) will be the same as that found from eq.(2.8) (i.e., $\overline{M}_p = M_p$). However, when $\mathbf{J}'_c$ becomes rank deficient, the parallel subchain *gains* a degree of freedom, and we find that $\overline{M}_p > M_p$. Angeles and Gosselin (1988) show a number of examples in which this this technique is applied to complex closed kinematic chains. In the next section, this method will be useful in determining whether or not a certain set of actuators can control a given kinematic chain in a prescribed orientation.

The mobility of the complete hybrid kinematic chain can be found by adding the mobilities of its constituent star and parallel subchains—i.e.,

$$\overline{M} = \overline{M}_s + \overline{M}_p \tag{3.43}$$

This concept of a mobility which includes geometric effects can be extended to formulate an equation for the connectivity of the two poles of a system of cooperating robotic devices which also includes geometric effects—i.e., modified to read

$$\overline{C} = \overline{M} - \overline{M}' \tag{3.44}$$

where $\overline{M}'$ is the mobility of the kinematic chain formed when the two poles are treated as a single link.

3.5 Controllability of a Kinematic Chain

In §2.6, we found that a kinematic chain would be controllable if it had at least as many driven actuators as its mobility. This implied that when the actuators were driven, the mobility of the actuated system, M_a, was reduced to zero. In order to find the mobility of the actuated system including geometric effects, we consider the Jacobian given by eqs.(3.32) and (3.34) to be partitioned as

$$\mathbf{J}'_c = [\mathbf{J}_a \quad \mathbf{J}_u] \tag{3.45}$$

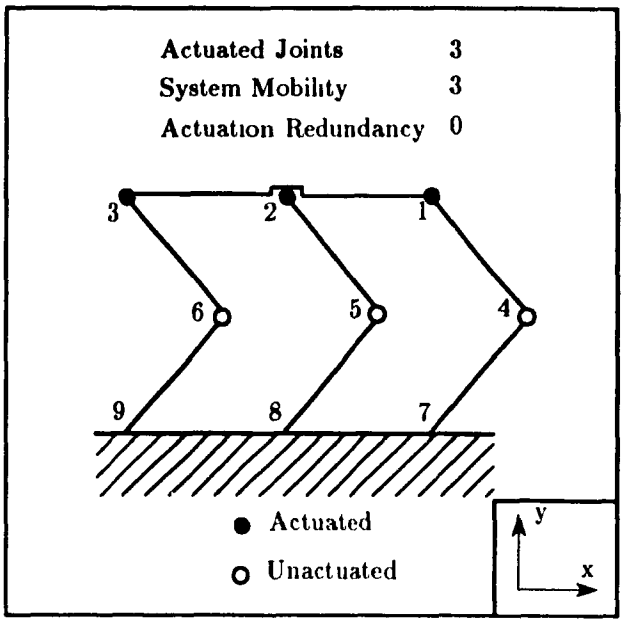

Figure 3.4 *Planar Three-Legged Walking Machine*

where $\mathbf{J}_a$ is composed of the columns of $\mathbf{J}'_c$ corresponding to the actuated joint degrees of freedom, and is of dimension $cd \times g'_{pa}$, where c is the number of cycles in the parallel subchain, d is the degree of freedom of an unconstrained body and g'_{pa} is the number of actuated joint degrees of freedom in the parallel subchain. Similarly, $\mathbf{J}_u$ is composed of the columns of $\mathbf{J}'_c$ corresponding to the unactuated joint degrees of freedom, and is of dimension $cd \times g'_{pu}$, where g'_{pu} is defined analogously to g'_{pa}. The mobility of the actuated parallel subchain is given by

$$\overline{M}_a = \dim[\mathcal{N}(\mathbf{J}_u)] \tag{3.46}$$

This method is now applied to the planar three-legged walking machine shown in Figure 3.4 with actuators 1, 2 and 3 active to show that certain singular conditions cannot be controlled with this set of actuators.

The Jacobian $\mathbf{J}'_c$ is a 6×9 matrix since there are 9 joints, 2 cycles and $d = 3$. According to eq.(3.34) $\mathbf{J}'_c$ can be formed from the Jacobians of the individual legs as

follows

$$\mathbf{J}'_c = \begin{bmatrix} \mathbf{J}_{s1} & -\mathbf{J}_{s2} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{s2} & -\mathbf{J}_{s3} \end{bmatrix}, \qquad \dot{\mathbf{q}}'_c = \begin{bmatrix} \dot{\mathbf{q}}_{s1} \\ \dot{\mathbf{q}}_{s2} \\ \dot{\mathbf{q}}_{s3} \end{bmatrix} \tag{3.47}$$

For motion in a plane, the 3-dimensional cross product, denoted by $\mathbf{e} \times \mathbf{r}_i$, can be written in 2-dimensional vector form as a linear mapping of the 2-dimensional vector $\mathbf{r}_i$. This mapping is denoted as $\mathbf{E}\mathbf{r}_i$, with the 2×2 matrix $\mathbf{E}$ defined as

$$\mathbf{E} \equiv \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{3.48}$$

so that the block elements of $\mathbf{J}'_c$ can be written as

$$\mathbf{J}_{s1} = \begin{bmatrix} 1 & 1 & 1 \\ \mathbf{E}\mathbf{r}_7 & \mathbf{E}\mathbf{r}_4 & \mathbf{E}\mathbf{r}_1 \end{bmatrix}, \qquad \mathbf{J}_{s2} = \begin{bmatrix} 1 & 1 & 1 \\ \mathbf{E}\mathbf{r}_8 & \mathbf{E}\mathbf{r}_5 & \mathbf{E}\mathbf{r}_2 \end{bmatrix}, \qquad \mathbf{J}_{s3} = \begin{bmatrix} 1 & 1 & 1 \\ \mathbf{E}\mathbf{r}_9 & \mathbf{E}\mathbf{r}_6 & \mathbf{E}\mathbf{r}_3 \end{bmatrix} \tag{3.49}$$

where vector $\mathbf{r}_i$ is directed from the centre of joint i to a reference point on the body of the walking machine.

Thus, if we assume that the leg links are each 1 unit long, the body is 2 units long, the body is parallel to the ground and $\sqrt{2}$ units above it and the reference point on the body is at the centre of joint 2, we obtain

$$\mathbf{J}'_c = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 & -1 & 0 & 0 & 0 \\ -\sqrt{2} & -\sqrt{2}/2 & 0 & \sqrt{2} & \sqrt{2}/2 & 0 & 0 & 0 & 0 \\ -1 & -(\frac{2+\sqrt{2}}{2}) & -1 & 0 & \sqrt{2}/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & -1 & -1 & -1 \\ 0 & 0 & 0 & -\sqrt{2} & -\sqrt{2}/2 & 0 & \sqrt{2} & \sqrt{2}/2 & 0 \\ 0 & 0 & 0 & 0 & -\sqrt{2}/2 & 0 & -1 & -(\frac{2-\sqrt{2}}{2}) & -1 \end{bmatrix} \tag{3.50a}$$

$$\dot{\mathbf{q}}'_c = [\dot{\theta}_7 \quad \dot{\theta}_4 \quad \dot{\theta}_1 \quad \dot{\theta}_8 \quad \dot{\theta}_5 \quad \dot{\theta}_2 \quad \dot{\theta}_9 \quad \dot{\theta}_6 \quad \dot{\theta}_3]^T \tag{3.50b}$$

If joints 1, 2 and 3 are actuated, we remove the corresponding columns from $\mathbf{J}'_c$ to obtain

$$\mathbf{J}_u = \begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 \\ -\sqrt{2} & -\sqrt{2}/2 & \sqrt{2} & \sqrt{2}/2 & 0 & 0 \\ -1 & -(\frac{2+\sqrt{2}}{2}) & 0 & \sqrt{2}/2 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & -\sqrt{2} & -\sqrt{2}/2 & \sqrt{2} & \sqrt{2}/2 \\ 0 & 0 & 0 & -\sqrt{2}/2 & -1 & -(\frac{2-\sqrt{2}}{2}) \end{bmatrix} \tag{3.51}$$

which is rank deficient and has a nullspace with a dimension of one. Thus, it is apparent that if only three actuators are used to drive this system, there exist configurations in which it cannot be controlled.

The dimension of the nullspace, or *nullity*, of $\mathbf{J}_u$ is the difference between its number of columns and its rank. Hence, whenever the number of columns in $\mathbf{J}_u$ exceeds its rank, the actuated system will become mobile. Obviously, if $\mathbf{J}_u$ has no columns—i.e., all joints are actuated—, the actuated system is bound to be controllable. When the system *does* have unactuated joints, we may want to compare its controllability with different sets of actuators active. This can be done by comparing the condition number of the corresponding $\mathbf{J}_u$'s—the higher the condition number, the closer $\mathbf{J}_u$ is to becoming rank deficient. The condition number of a rectangular matrix can be found as (Golub and Van Loan, 1983):

$$\kappa(\mathbf{J}_u) = \frac{\sigma_{max}}{\sigma_{min}} \tag{3.52}$$

where σ_{max} and σ_{min} are the maximum and minimum singular values of $\mathbf{J}_u$.

As was noted in §2.6, a kinematic chain with a mobility of M nominally needs only M actuators to be active. There are K combinations of M actuators which could be driven without yielding a redundantly actuated system, where K is given by

$$K = \frac{g_a!}{M!(g_a - M)!} \tag{3.53}$$

The condition numbers of the $\mathbf{J}_u$'s corresponding to each candidate set of driven actuators could be compared to determine which set would best control the system. However, for a 6-legged walking machine with three actuators per leg and all six legs in ground contact ($g_a = 18$ and $M = 6$), this would entail the evaluation of the condition number of 18564 30×30 matrices—clearly not feasible in a real-time context.

But more importantly, Golub and Van Loan (1983) show that *by adding a column to a matrix, the largest singular value increases and the smallest singular value is diminished.* This implies that the condition number of $\mathbf{J}_u$ will be increased whenever

a column is added to it—i.e., we deactivate an actuator—, thereby bringing $\mathbf{J}_u$ closer to rank deficiency. We can therefore conclude that *a redundantly-actuated system will be more controllable with a given set of actuators active than with any subset of those* actuators active. Finally, it becomes evident that, from this perspective, it is desirable to keep as many actuators as possible active.

3.6 Kinematic/Static Duality

Because there exists a duality between the kinematics and statics of robotic manipulators, it is appropriate to discuss the statics of the cooperating robotic devices at this point, as a transition between their kinematics and dynamics. Kinematic/static duality can be derived by considering the power input to and output from a system which can neither store nor dissipate energy, namely, a system in which kinetic energy, strain energy, friction and damping are all absent and where gravitational forces are considered as external forces applied to the system. In this case, the principle of conservation of energy allows us to conclude that the power input to the system is equal to the power output from the system.

The principle of conservation of energy is first applied to serial manipulators in order to demonstrate the analogy between the kinematic and static relations—a result usually obtained through the principle of virtual work (see e.g., Asada and Slotine, 1986). The power supplied to the actuators can be written as

$$\pi_i = \boldsymbol{\tau}_s^T \dot{\mathbf{q}}_s \tag{3.54}$$

where $\boldsymbol{\tau}_s$ is a vector of the actuator torques applied at each joint. If we assume that no gravitational forces act on any of the intermediate links, the power output to a load at the end-effector is

$$\pi_o = \mathbf{w}_s^T \mathbf{t}_s \tag{3.55}$$

where $\mathbf{w}_s$ is a vector composed of forces and moments (hereafter called wrench) applied

by the end-effector. Equating these two powers, we obtain

$$\boldsymbol{\tau}_s^T \dot{\mathbf{q}}_s = \mathbf{w}_s^T \mathbf{t}_s \tag{3.56}$$

Substituting eq.(3.14) into eq.(3.56) yields

$$(\boldsymbol{\tau}_s^T - \mathbf{w}_s^T \mathbf{J}_s)\dot{\mathbf{q}}_s = 0 \tag{3.57}$$

Since this equation must be satisfied for *all* $\dot{\mathbf{q}}_s$, we can say

$$\boldsymbol{\tau}_s = \mathbf{J}_s^T \mathbf{w}_s \tag{3.58}$$

This equation tells us how a general wrench $\mathbf{w}_s$ will be mapped into a set of actuator torques $\boldsymbol{\tau}_s$. On the other hand, if $\boldsymbol{\tau}_s$ is known, we can solve the system given by eq.(3.58) to obtain $\mathbf{w}_s$ since $\mathbf{J}_s$ is usually square. The similarity of eq.(3.58) and eq.(3.14) leads us to say that the velocity kinematics and the statics problem are *dual* to each other.

A similar procedure is now applied to systems composed of cooperating robotic devices. In exactly the same way, we equate the power input to the system and the power output from the system:

$$\boldsymbol{\tau}_c^T \dot{\mathbf{q}}_c = \mathbf{w}_c^T \mathbf{t}_c \tag{3.59}$$

where $\dot{\mathbf{q}}_c$ is the vector of actuated joint rates, $\boldsymbol{\tau}_c$ is a vector of the actuator torques in the system, $\mathbf{w}_c$ is the wrench applied by the moving pole on its environment, while $\mathbf{t}_c$ is the twist of the moving pole. Equation (3.31) can be substituted into eq.(3.59) to obtain

$$(\boldsymbol{\tau}_c^T \mathbf{P} \begin{bmatrix} \mathbf{J}_{s1}^{-1} \\ \vdots \\ \mathbf{J}_{sp}^{-1} \end{bmatrix} - \mathbf{w}_c^T)\mathbf{t}_c = 0 \tag{3.60}$$

Equation (3.60) must be satisfied for *all* $\mathbf{t}_c$, thereby yielding

$$\mathbf{w}_c = \mathbf{A}\boldsymbol{\tau}_c \tag{3.61a}$$

with matrix $\mathbf{A}$ defined as

$$\mathbf{A} = [\, \mathbf{J}_{s1}^{-T} \quad \dots \quad \mathbf{J}_{sp}^{-T} \,]\, \mathbf{P}^T \tag{3.61b}$$

Equation (3.61a) specifies how a general set of actuator torques $\boldsymbol{\tau}_c$ will be mapped into a wrench at the moving pole $\mathbf{w}_c$. On the other hand, we might be interested in determining the actuator torques $\boldsymbol{\tau}_c$ required to exert a prescribed wrench at the moving pole $\mathbf{w}_c$, as was done previously for serial manipulators. This would be straightforward if the system given by eq.(3.61a) were determinate, i.e., if matrix $\mathbf{A}$ were square. However, when the system is redundantly-actuated, the system is underdetermined, i.e., matrix $\mathbf{A}$ has fewer rows than columns, and therefore does not have a unique solution. A general procedure for resolving this underdeterminacy will be detailed in Chapters 5 and 6. For the time being, it is only important to note that this non-uniqueness of the solution exists. The next chapter will show that even when the effect of dynamic forces are considered, the resulting system of equations will remain underdetermined.

Chapter 4

Dynamics

A number of software packages exist to find the dynamic forces acting in mechanisms in motion (e.g., IMP, ADAMS, DADS). However, these packages are primarily aimed at systems with fixed topology and are not able to analyze systems which are redundantly actuated. This chapter is therefore intended to deal with the formulation of the dynamics equations of redundantly actuated robotic systems. This will allow us to determine the forces which accompany the motions discussed in the previous chapter. Broadly speaking, there exist two possible problems in the dynamics of robot manipulators:

1. **Forward Dynamics**—given the initial condition of the system, the external wrenches acting on the system and the actuator torques, find the corresponding time histories of the joint and/or Cartesian coordinates of the manipulator. This entails solving the motion equations for the relevant accelerations and integrating these to obtain velocities and positions. Solution of the forward dynamics problem is particularly useful when it is desired to simulate a system for design or animation purposes.

2. **Inverse Dynamics**—given the motion of the system, find the actuator and constraint wrenches acting in the system. This problem may be formulated as a set of simultaneous linear equations which must be solved for actuator and constraint wrenches. Solution of the problem is useful for design and control purposes. When used in control applications, such as a 'computed torque' scheme (see e.g., Asada and Slotine, 1986), it is imperative that the problem be solvable in real-time.

Since the present work focuses on the control of redundantly-actuated robotic systems, we will deal primarily with the real-time solution of the inverse dynamics problem. The desired motion of the system is therefore assumed to be prescribed, while the actuator torques required to achieve this motion must be found. It will be shown that, as was found when we dealt with statics in §3.6, the inverse dynamics problem is underdetermined. This will lead us to treat the dynamics equations as constraints in an optimization problem. However, these are not the only constraints which must be imposed on the solution to the optimization problem—other ones being necessary to account for limitations of passive contacts and actuator capabilities. This chapter will therefore also deal with these latter unilateral constraints on an allowable solution.

4.1 Dynamics of Open Kinematic Chains

The dynamics equations of open kinematic chains consisting of rigid bodies are well understood. The most recently-developed solutions to the forward dynamics problem lead to efficient $O(n)$ algorithms, where n is the number of links in the kinematic chain. A good example of these is the method recently developed by Rodriguez *et al.* (1989)—applicable to both open and closed kinematic chains—which applies optimal filtering techniques to the solution of the forward dynamics problem. The inverse dynamics problem of these systems was the subject of intensive investigation in the early 1980's and solution techniques were developed which are computationally efficient. For example, the work of Luh *et al.* (1980) presents a recursive Newton-Euler technique to calculate actuator torques from prescribed joint positions, velocities and accelerations with roughly 1500 floating point operations[1] for a six-link robot. Further work in this area has yielded techniques which consume as little as 950 floating point operations (Angeles *et al.*, 1989). Since modern workstations can easily perform 1.5 Mflops (i.e., 1.5 million floating point operations per second), one evaluation of the inverse dynamics model can be performed in less than 1 ms—a rate which will easily accommodate real-time operation.

[1] Any *single* multiplication, division, addition or subtraction between two real numbers

Appendix A gives a recursive inverse dynamics algorithm for serial manipulators which is essentially that of Luh *et al.* (1980). This algorithm can be used to calculate the actuator and constraint wrenches acting in any serial subchain within the system of cooperating robotic devices which is uncoupled from the remainder of the system—e.g., a finger of a mechanical hand which is not in contact with the grasped object. It will also be used in certain of the inverse dynamics formulations of multiple-loop linkages which follow. The algorithm is made up of two phases which can be described as follows:

1. Outward recursion from the manipulator's base to its tip to calculate the motion of each link in Cartesian space. That is, given the motion of the base link and all the joint angles, joint rates and joint accelerations, the Cartesian motion of the other links is calculated by starting at the base and working outward to the last link.

2. Inward recursion from the manipulator's tip to the base to calculate the wrenches acting on each link, including both constraint and actuator wrenches.

4.2 Dynamics of Closed Kinematic Chains

It is useful to partition the wrenches in the system into:

1. **Contact Wrenches**, which act at the interface between the individual robotic devices in the system and one of the poles.

2. **Actuator Wrenches**, which are the forces and/or moments supplied by the actuators at the actuated joints in the system;

3. **Constraint Wrenches**, which are the non-working wrenches acting at the joints in the constrained directions; and

4. **External Wrenches**, which may act at any point in the system due to external loads.

In the context of the inverse dynamics problem, wrenches of the first three types are *unknown*, while wrenches of the last type are assumed to be prescribed.

The inward recursion of the inverse dynamics algorithm developed by Luh *et al.* (1980) relies on having a single unknown wrench acting on the last link in the chain (that being the wrench exerted by the second-to-last link) and is therefore not applicable to closed kinematic chains since they have no 'last link' with a single unknown wrench. Researchers in robotics have only recently approached the inverse dynamics problem of manipulators containing kinematic loops. Luh and Zheng (1985) proposed the 'virtual cut' method which involves 'cutting' each kinematic loop in the system at an unactuated joint to produce a kinematic chain with a tree structure. Using a serial-chain algorithm and explicit calculation of a set of Lagrange multipliers, the torques required at the actuated joints can be found. Nakamura and Ghodoussi (1989) later improved the virtual cut method by avoiding the explicit calculation of the Lagrange multipliers. The virtual cut technique was primarily intended for serial manipulators with local planar closed loops for which the explicit formulation of the kinematic constraint equations is tractable. However, these closed-chain constraint equations can be tedious to construct in the case of multiple 3-dimensional loops.

In the present analysis, it is presumed that all the joint coordinates, velocities and accclerations are known. If only the motion of one of the poles is known, the inverse kinematics problem described in Chapter 3 must first be solved. Given these inputs, there are a number of ways in which the dynamics equations of the system can be written, each of which reduces to a system of linear equalities of the form

$$\mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \tag{4.1}$$

where $\mathbf{A}_1$ is a matrix of dimension $m \times n$, $\mathbf{x}$ is a vector of dimension n which contains the wrenches acting in the system, while $\mathbf{b}_1$ is a vector of dimension m which represents the motion of the system. The numerical values of m and n depend on the method used to formulate the dynamics equations. The following sections will discuss three ways in

which this can be done—each differing in complexity, computational requirements, and amount of information generated. It will be shown that no matter how the equations are formulated, if the system is redundantly actuated, n will always be greater than m by an amount r, the degree of actuation redundancy. In this case, the fact that the linear system given by eq.(4.1) contains more unknowns than equations implies that there exist many solutions which will satisfy the equations This allows us to formulate an optimization problem in order to find the 'best' solution in a pre-established sense. In this light, we say that eq.(4.1) constitutes a set of *constraints* to an optimization problem in which we minimize a certain *objective function*, $f(\mathbf{x})$, of vector $\mathbf{x}$ of *design variables* (a.k.a. *decision variables*).

4.2.1 The Complete Formulation

The first and most comprehensive method for writing the dynamics equations of the system explicitly considers all wrenches acting in the system. The force and moment balance equations are written for each link in the chain resulting in a large system of equations, where vector $\mathbf{x}$ contains the contact, actuator *and* constraint wrenches.

The system is first broken down into its individual rigid bodies as shown in Figure 4.1(b). The prescribed motion of the base of path i is given by $\boldsymbol{\omega}_0^i$, $\dot{\boldsymbol{\omega}}_0^i$ and $\mathbf{a}_0^i$—its angular velocity and acceleration and its translational acceleration, respectively—*n.b.* the translational velocity of the base link is not explicitly required. A forward recursion is performed from the base link to each of the p end links—fingertips, end-effectors or feet—by recursively solving the following vector equations for $j = 1$ to b_{p_i}, where b_{p_i} is the number of links in path i:

When joint j is a revolute:

$$\boldsymbol{\omega}_j^i = \boldsymbol{\omega}_{j-1}^i + \dot{q}_j^i \mathbf{e}_{j-1}^i \tag{4.2a}$$

$$\dot{\boldsymbol{\omega}}_j^i = \dot{\boldsymbol{\omega}}_{j-1}^i + \ddot{q}_j^i \mathbf{e}_{j-1}^i + \boldsymbol{\omega}_{j-1}^i \times \dot{q}_j^i \mathbf{e}_{j-1}^i \tag{4.2b}$$

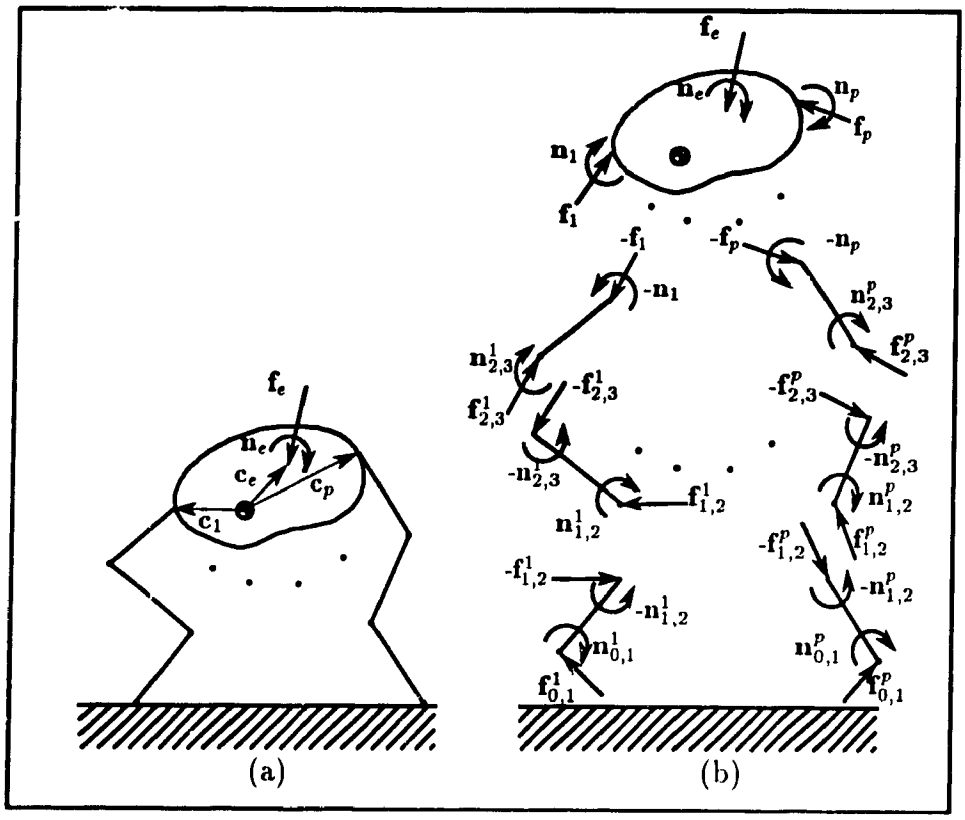

Figure 4.1 *Decomposition of the Parallel Subchain*

$$\mathbf{a}_j^i = \mathbf{a}_{j-1}^i + \dot{\boldsymbol{\omega}}_j^i \times \mathbf{r}_j^i + \boldsymbol{\omega}_j^i \times (\boldsymbol{\omega}_j^i \times \mathbf{r}_j^i) \tag{4.2c}$$

When joint j is prismatic:

$$\boldsymbol{\omega}_j^i = \boldsymbol{\omega}_{j-1}^i \tag{4.3a}$$

$$\dot{\boldsymbol{\omega}}_j^i = \dot{\boldsymbol{\omega}}_{j-1}^i \tag{4.3b}$$

$$\mathbf{a}_j^i = \mathbf{a}_{j-1}^i + \dot{\boldsymbol{\omega}}_j^i \times \mathbf{r}_j^i + \boldsymbol{\omega}_j^i \times (\boldsymbol{\omega}_j^i \times \mathbf{r}_j^i) + \ddot{q}_j^i \mathbf{e}_{j-1}^i + 2\boldsymbol{\omega}_{j-1}^i \times \dot{q}_j^i \mathbf{e}_{j-1}^i \tag{4.3c}$$

where, for each path i, $\boldsymbol{\omega}_j^i$ denotes the angular velocity of link j, $\mathbf{a}_j^i$ denotes its translational acceleration at the origin of frame j which is attached to link j, q_j^i is the j-th joint variable, $\mathbf{e}_j^i$ is a vector aligned with the j-th joint axis and $\mathbf{r}_j^i$ represents the vector from the origin of frame $(j-1)$ to the origin of frame j.

For both types of joints:

$$\mathbf{a}^i_{cj} = \mathbf{a}^i_j + \dot{\boldsymbol{\omega}}^i_j \times \mathbf{c}^i_j + \boldsymbol{\omega}^i_j \times (\boldsymbol{\omega}^i_j \times \mathbf{c}^i_j) \tag{4.4}$$

where $\mathbf{a}^i_{cj}$ is the translational acceleration of link j at its centroid, and $\mathbf{c}^i_j$ is a vector from the origin of frame j to the centroid of the link j.

We can now write the Newton-Euler equations for link j in parallel path i as:

$$\mathbf{f}^i_{j-1,j} - \mathbf{f}^i_{j,j+1} = m^i_j(\mathbf{a}^i_{cj} + \mathbf{g}) - \mathbf{f}^i_{ej} \tag{4.5a}$$

$$\mathbf{n}^i_{j-1,j} - \mathbf{n}^i_{j,j+1} + \hat{\mathbf{c}}^i_{j,j-1} \times \mathbf{f}^i_{j-1,j} - \hat{\mathbf{c}}^i_{j,j} \times \mathbf{f}^i_{j,j+1} = \mathbf{I}^i_j \dot{\boldsymbol{\omega}}^i_j + \boldsymbol{\omega}^i_j \times \mathbf{I}^i_j \boldsymbol{\omega}^i_j - (\mathbf{n}^i_{ej} + \mathbf{c}^i_{ej} \times \mathbf{f}^i_{ej}) \tag{4.5b}$$

where $\mathbf{f}^i_{k,k+1}$ and $\mathbf{n}^i_{k,k+1}$ are, respectively, the force and moment applied by link k of path i to link $k+1$ in the same path, and $\hat{\mathbf{c}}^i_{k,l}$ represents a vector from the centroid of link k to the origin of frame l. The centroid moment of inertia and mass of link j are denoted by $\mathbf{I}^i_j$ and m^i_j, respectively. Finally, $\mathbf{f}^i_{ej}$ and $\mathbf{n}^i_{ej}$ are the external force and moment applied to link j, while $\mathbf{c}^i_{ej}$ represents a vector from the centroid of link j to the point of application of $\mathbf{f}^i_{ej}$.

For notational convenience, we assign a special symbol to the wrench acting at the distal end of the last link of each path.

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix}, \qquad \mathbf{f}_i \equiv \mathbf{f}^i_{b_{pi},b_{pi}+1}, \qquad \mathbf{n}_i \equiv \mathbf{n}^i_{b_{pi},b_{pi}+1}, \qquad i = 1, \ldots, p \tag{4.6}$$

where p is the number of paths between the two poles.

Using this notation, the equations of motion of the moving pole are written as

$$\sum_{i=1}^{p} \mathbf{f}_i = m_o(\mathbf{a}_o + \mathbf{g}) - \mathbf{f}_e \tag{4.7a}$$

$$\sum_{i=1}^{p} (\mathbf{n}_i + \mathbf{c}_i \times \mathbf{f}_i) = \mathbf{I}_o \dot{\boldsymbol{\omega}}_o + \boldsymbol{\omega}_o \times \mathbf{I}_o \boldsymbol{\omega}_o - (\mathbf{n}_e + \mathbf{c}_e \times \mathbf{f}_e) \tag{4.7b}$$

where m_o and $\mathbf{I}_o$ represent the moving pole's mass and centroid moment of inertia, $\boldsymbol{\omega}_o$ and $\mathbf{a}_o$ are its angular velocity and translational acceleration, and $\mathbf{c}_i$ is the vector from

the moving pole's centroid to contact point i. Furthermore, $\mathbf{f}_e$ is the vector of the net external applied force on the moving pole, of dimension 3, $\mathbf{n}_e$ is the vector of net external applied torques, also of dimension 3 and $\mathbf{c}_e$ is the vector from the object's centroid to the point of application of the external force.

Equations (4.5a) and (4.5b) are written for each link in each path of the system, while eqs.(4.7a) and (4.7b) are written for the moving pole. Once these vector equations are expressed in a single frame, they can be assembled into a single system of simultaneous equations of the form given by eq.(4.1) with

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \dots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{C}_1 & \dots & \mathbf{1} & \mathbf{C}_p & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & -\mathbf{1} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{1} & \hat{\mathbf{C}}^1_{1,0} & -\mathbf{1} & -\hat{\mathbf{C}}^1_{1,1} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & & \ddots & & \vdots & \vdots & & & & \ddots & & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \dots & -\mathbf{1} & -\hat{\mathbf{C}}^p_{b_{pp},b_{pp}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{1} & \hat{\mathbf{C}}^p_{b_{pp},b_{pp}-1} \end{bmatrix} \tag{4.8a}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{f}_1 \\ \vdots \\ \mathbf{n}_p \\ \mathbf{f}_p \\ \mathbf{n}^1_{0,1} \\ \mathbf{f}^1_{0,1} \\ \mathbf{n}^1_{1,2} \\ \mathbf{f}^1_{1,2} \\ \vdots \\ \mathbf{n}^p_{b_{pp}-1,b_{pp}} \\ \mathbf{f}^p_{b_{pp}-1,b_{pp}} \end{bmatrix}, \qquad \mathbf{b}_1 = \begin{bmatrix} m_o(\mathbf{a}_o + \mathbf{g}) - \mathbf{f}_e \\ \mathbf{I}_o\dot{\boldsymbol{\omega}}_o + \boldsymbol{\omega}_o \times \mathbf{I}_o\boldsymbol{\omega}_o - (\mathbf{n}_e + \mathbf{c}_e \times \mathbf{f}_e) \\ m^1_1(\mathbf{a}^1_{c1} + \mathbf{g}) - \mathbf{f}^1_{\epsilon 1} \\ \mathbf{I}^1_1\dot{\boldsymbol{\omega}}^1_1 + \boldsymbol{\omega}^1_1 \times \mathbf{I}^1_1\boldsymbol{\omega}^1_1 - (\mathbf{n}^1_{\epsilon 1} + \mathbf{c}^1_{\epsilon 1} \times \mathbf{f}^1_{e1}) \\ \vdots \\ m^p_{b_{pp}}(\mathbf{a}^p_{cb_{pp}} + \mathbf{g}) - \mathbf{f}^p_{\epsilon b_{pp}} \\ \mathbf{I}^p_{b_{pp}}\dot{\boldsymbol{\omega}}^p_{b_{pp}} + \boldsymbol{\omega}^p_{b_{pp}} \times \mathbf{I}^p_{b_{pp}}\boldsymbol{\omega}^p_{b_{pp}} - (\mathbf{n}^p_{\epsilon b_{pp}} + \mathbf{c}^p_{eb_{pp}} \times \mathbf{f}^p_{eb_{pp}}) \end{bmatrix} \tag{4.8b}$$

where $\mathbf{C}_k$ is the 3×3 matrix defined by

$$\mathbf{C}_k \equiv \frac{\partial(\mathbf{c}_k \times \mathbf{v})}{\partial \mathbf{v}} \tag{4.8c}$$

while $\hat{\mathbf{C}}^i_{k,l}$ is the 3×3 matrix defined by

$$\hat{\mathbf{C}}^i_{k,l} \equiv \frac{\partial(\hat{\mathbf{c}}^i_{k,l} \times \mathbf{v})}{\partial \mathbf{v}} \tag{4.8d}$$

for any 3-dimensional vector **v**.

This system of equations consists of $m = 6(1 + \sum_{i=1}^{p} b_{pi})$ equations with, at most, $n = 6g_p$ unknowns, where g_p is the number of joints in the parallel subchain. The number of unknowns, n, will be reduced by g'_{pu}, the number of unactuated joint degrees of freedom in the parallel subchain since we know the force or moment corresponding to an unactuated degree of freedom is zero. We also know that there are $b_p = 2 + \sum_{i=1}^{p} b_{pi}$ bodies in the system—the 2 poles and all the links in each path between the two poles. The degree of underdeterminacy of this system—i.e., the number of columns minus the number of rows in $\mathbf{A}_1$—is therefore

$$\begin{aligned} n - m &= 6g_p - g'_{pu} - 6(b_p - 1) \\ &= 6(g_p - b_p + 1) - g'_{pu} \end{aligned} \tag{4.9}$$

Substituting eq.(2.5) into eq.(4.9), we obtain

$$n - m = 6c - g'_{pu} \tag{4.10}$$

into which we can substitute eq.(2.6) (with $d = 6$) to find

$$n - m = -M_p + g'_p - g'_{pu} \tag{4.11}$$

but since $g'_p = g'_{pa} + g'_{pu}$, we can write

$$n - m = g'_{pa} - M_p \tag{4.12}$$

Since the number of actuated joint degrees of freedom in the star subchain is always equal to its mobility, we have

$$g'_{sa} = M_s \tag{4.13}$$

which allows eq.(4.12) to be rewritten as

$$\begin{aligned} n - m &= g'_{pa} + g'_{sa} - M_p - M_s \\ &= g_a - M \end{aligned} \tag{4.14}$$

This difference in turn, is nothing but the actuation redundancy r as defined by eq.(2.15). Thus, it becomes clear that the system given by eqs.(4.1), (4.8a) and (4.8b) has r more columns than rows. That is, *the degree of underdeterminacy of the system of inverse dynamics equations of a redundantly-actuated system is equal to its actuation redundancy.*

4.2.2 A Formulation Using Superposition

The system of equations obtained in §4.2.1 gives access to *all* the wrenches acting in the system and is therefore well suited to design applications. However, the large dimensionality of the formulation makes it unsuitable for implementation in a real-time controller for most realistic 3-D systems. We can assume that, while the computation of the constraint wrenches is useful for design purposes, it is less essential for control purposes. Use of the principle of superposition allows us to avoid computing the constraint wrenches and concentrate exclusively on the variables of interest: the actuator and contact wrenches.

The parallel kinematic subchain has the structure shown in Figure 4.2(a)—two poles joined by p paths. An easy way to obtain the *equivalent tree structure* is to remove one of the poles—for example, in the case of a hand-object system, we separate the grasped object from the hand, as shown in Figure 4.2(b). For each of the p resulting serial subchains, we can write

$$\mathbf{t}_i = \begin{bmatrix} \boldsymbol{\omega}_i \\ \mathbf{v}_i \end{bmatrix} = \mathbf{J}_{si}\dot{\mathbf{q}}_i \tag{4.15}$$

where $\boldsymbol{\omega}_i$ and $\mathbf{v}_i$ are vectors of dimension 3 representing the angular and translational velocities of contact point i; $\mathbf{J}_{si}$ is the $6 \times g'_{pi}$ Jacobian of the arising i-th serial manipulator; $\dot{\mathbf{q}}_i$ is the vector of dimension g'_{pi} of joint rates for path i; and g'_{pi} is the number of joint degrees of freedom in path i.

The torques acting in each path of the closed chain, $\boldsymbol{\tau}_i$, can be separated into two components: 1) $\boldsymbol{\tau}'_i$, the component associated with the inertial forces in that subchain

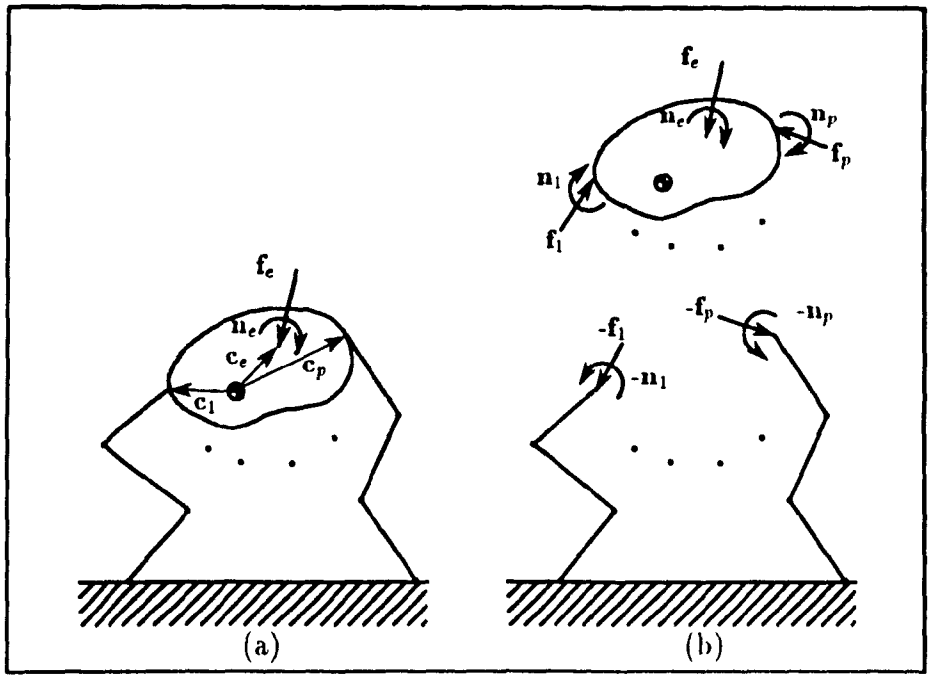

Figure 4.2 *Another Decomposition of the Parallel Subchain*

acting as a serial chain, and 2) $\boldsymbol{\tau}_i''$, the component associated with $\mathbf{w}_i$, the wrench exerted by the tip of subchain i due to contact with the removed pole. Therefore, we can write

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_i' + \boldsymbol{\tau}_i'' \tag{4.16}$$

where $\boldsymbol{\tau}_i$ is a vector of dimension g_{pi}'.

The recursive algorithm given in Appendix A can be used to calculate $\boldsymbol{\tau}_i'$—the torques acting in the i-th serial subchain in the absence of a contact wrench. In order to find $\boldsymbol{\tau}_i''$, we formulate the following expressions for the power produced by $\boldsymbol{\tau}_i''$ and by $\mathbf{w}_i$

$$\pi_i'' = \dot{\mathbf{q}}_i^T \boldsymbol{\tau}_i'' \tag{4.17a}$$

$$\pi_i = \mathbf{t}_i^T \mathbf{w}_i \tag{4.17b}$$

These two powers are equal in the absence of dissipation. If we substitute eq.(4.15) into eq.(4.17b) and equate eqs.(4.17a) and (4.17b), we obtain

$$\dot{\mathbf{q}}_i^T \boldsymbol{\tau}_i'' = \dot{\mathbf{q}}_i^T \mathbf{J}_{si}^T \mathbf{w}_i \tag{4.18}$$

Since eq.(4.18) must apply for any $\dot{\mathbf{q}}_i$, we can write

$$\boldsymbol{\tau}_i'' = \mathbf{J}_{si}^T \mathbf{w}_i \tag{4.19}$$

and eq.(4.16) becomes

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_i' + \mathbf{J}_{si}^T \mathbf{w}_i \tag{4.20}$$

It is interesting to note that this is nothing but an extension of the result obtained in eq.(3.58) to include the effect of the inertia of the manipulator. If the joints in the serial paths are not all actuated eq.(4.20) can be partitioned as

$$\begin{bmatrix} \boldsymbol{\tau}_{ai} \\ \boldsymbol{\tau}_{ui} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau}_{ai}' \\ \boldsymbol{\tau}_{ui}' \end{bmatrix} + \begin{bmatrix} \mathbf{J}_{sai}^T \\ \mathbf{J}_{sui}^T \end{bmatrix} \mathbf{w}_i \tag{4.21}$$

where $\boldsymbol{\tau}_{ai}$ is a vector of dimension g'_{pai}, the number of actuated joint degrees of freedom in path i; and $\boldsymbol{\tau}_{ui}$ is a vector of dimension g'_{pui}, the number of unactuated joint degrees of freedom in path i. Since the joint degrees of freedom corresponding to $\boldsymbol{\tau}_{ui}$ are unactuated, $\boldsymbol{\tau}_{ui} = \mathbf{0}$ and we can expand eq.(4.21) as

$$\mathbf{J}_{sai}^T \mathbf{w}_i - \boldsymbol{\tau}_{ai} = -\boldsymbol{\tau}_{ai}' \tag{4.22a}$$

$$\mathbf{J}_{sui}^T \mathbf{w}_i = -\boldsymbol{\tau}_{ui}' \tag{4.22b}$$

Equations (4.22a) and 4.22b) can be compactly written as

$$\widetilde{\mathbf{A}}_i \mathbf{x}_i = \tilde{\mathbf{b}}_i \tag{4.23a}$$

where

$$\widetilde{\mathbf{A}}_i = \begin{bmatrix} \mathbf{J}_{sai}^T & -\mathbf{1}_{g'_{pai}} \\ \mathbf{J}_{sui}^T & \mathbf{0}_{g'_{pui} \times g'_{pai}} \end{bmatrix}, \quad \mathbf{x}_i = \begin{bmatrix} \mathbf{w}_i \\ \boldsymbol{\tau}_{ai} \end{bmatrix}, \quad \mathbf{w}_i = \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix}, \quad \tilde{\mathbf{b}}_i = \begin{bmatrix} -\boldsymbol{\tau}_{ai}' \\ -\boldsymbol{\tau}_{ui}' \end{bmatrix} \tag{4.23b}$$

where $\mathbf{f}_i$ is the vector of contact forces of dimension 3, $\mathbf{n}_i$ is the vector of contact torques, also of dimension 3, $\mathbf{1}_k$ denotes the $k \times k$ identity matrix, and $\mathbf{0}_{k \times l}$ denotes the $k \times l$ zero matrix.

The complete equations governing the motion of the system can now be formed by combining eqs.(4.7a) and (4.7b) for the object, with eq.(4.23a) for each serial subchain

in the equivalent tree structure. This yields an augmented system of the form given by eq.(4.1) with

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{B}_1 & \cdots & \mathbf{B}_p \\ \mathbf{D}_1 & \cdots & \mathbf{D}_p \\ \widetilde{\mathbf{A}}_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \widetilde{\mathbf{A}}_p \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_p \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} m_o(\mathbf{a}_o + \mathbf{g}) - \mathbf{f}_e \\ \mathbf{I}_o\dot{\omega}_o + \omega_o \times \mathbf{I}_o\omega_o - (\mathbf{n}_e + \mathbf{c}_e \times \mathbf{f}_e) \\ \widetilde{\mathbf{b}}_1 \\ \vdots \\ \widetilde{\mathbf{b}}_p \end{bmatrix} \tag{4.24a}$$

$$\mathbf{B}_i = \begin{bmatrix} \mathbf{0}_{3\times 3} & \mathbf{1}_3 & \mathbf{0}_{3\times g'_{pai}} \end{bmatrix}, \qquad \mathbf{D}_i = \begin{bmatrix} \mathbf{1}_3 & \mathbf{C}_i & \mathbf{0}_{3\times g'_{pai}} \end{bmatrix} \tag{4.24b}$$

The linear system represented by eqs.(4.1), (4.24a) and (4.24b) is composed of $m = (6 + \sum_{i=1}^{p}[g'_{pai} + g'_{pui}])$ equations in, at most, $n = (6p + \sum_{i=1}^{p} g'_{pai})$ unknowns- -i.e., p contact wrenches of 6 elements each, plus g'_{pai} actuator torques in each path. When a general wrench cannot be exerted at the i-th contact, the number of unknowns is reduced by c_i—the number of contact degrees of freedom. Thus, the degree of underdeterminacy can be calculated as

$$\begin{aligned} n - m &= 6p + \sum_{i=1}^{p} g'_{pai} - \sum_{i=1}^{p} c_i - 6 - \sum_{i=1}^{p}(g'_{pai} + g'_{pui}) \\ &= 6(p-1) - \sum_{i=1}^{p}(c_i + g'_{pui}) \end{aligned} \tag{4.25}$$

Using eq.(2.7), and the fact $\sum_{i=1}^{p}(c_i + g'_{pui}) = g'_{pu}$, where g'_{pu} is the total number of unactuated joint degrees of freedom in the parallel subchain, we find that eq.(4.25) can be rewritten as

$$n - m = 6c - g'_{pu} \tag{4.26}$$

which is identical to eq.(4.10). Thus, the degree of underdeterminacy using this formulation is once again equal to r, the degree of actuation redundancy in the system.

4.2.3 The Most Compact Formulation

The third, most compact and most commonly used method of formulating the dynamics equations—see e.g., Nakamura *et al.* (1987), Hsu *et al.* (1988), Alberts and

Soloway (1988), Cheng and Orin (1989)—considers only a single body in the system. This has the advantage that the size of the resulting set of linear equations is greatly reduced and permits a fast solution. However, it only includes the contact wrenches in the vector of design variables, **x**. In the case of mechanical hands and cooperating manipulators, the single body for which the equations of motion are written is the grasped object, while in the case of walking machines, that body is the complete machine considered as a single rigid body—i.e., neglecting the internal joints of the machine. Because the vector of design variables includes only the contact wrenches, there is an attendant loss in generality which may be circumvented by calculating the actuator and constraint wrenches as functions of the contact wrenches. This will be shown in §4.2.4.

Referring to Figure 4.2(b), the vector equations of motion of the moving pole are given by eqs.(4.7a) and (4.7b). These can be written more compactly in the form of eq.(4.1) with

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \cdots & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{C}_1 & \mathbf{1} & \mathbf{C}_2 & \cdots & \mathbf{1} & \mathbf{C}_p \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{f}_1 \\ \vdots \\ \mathbf{n}_p \\ \mathbf{f}_p \end{bmatrix} \tag{4.27a}$$

$$\mathbf{b}_1 = \begin{bmatrix} m_o(\mathbf{a}_o + \mathbf{g}) - \mathbf{f}_e \\ \mathbf{I}_o\dot{\boldsymbol{\omega}}_o + \boldsymbol{\omega}_o \times \mathbf{I}_o\boldsymbol{\omega}_o - (\mathbf{n}_e + \mathbf{c}_e \times \mathbf{f}_e) \end{bmatrix} \tag{4.27b}$$

with $\mathbf{C}_k$ defined by eq.(4.8c). Using eqs.(4.27a) and (4.27b), eq.(4.1) now represents 6 scalar equations in, at most, $6p$ unknowns—i.e, p contact wrenches of 6 elements each. Once again, when a general wrench cannot be exerted at the i-th contact, the number of unknowns is reduced by c_i—the number of contact degrees of freedom. As well, if the i-th path contains g'_{pui} unactuated joints, it loses the ability to independently exert that number of contact forces or torques. Thus, the degree of underdeterminacy can be calculated as

$$\begin{aligned} n - m &= 6p - \sum_{i=1}^{p}(c_i + g'_{pui}) - 6 \\ &= 6(p-1) - \sum_{i=1}^{p}(c_i + g'_{pui}) \end{aligned} \tag{4.28}$$

which is identical to eq.(4.25), and, once again, the underdeterminacy of the system of equations is r. Matrix $\mathbf{A}_1$ is of dimension $6 \times 6p$ when a general grasping wrench of dimension 6 can be exerted, rendering the system underdetermined whenever $p > 1$. This will usually be the case for multiple manipulators handling a common payload. If contact torques cannot be exerted—i.e., the contact wrench is of dimension 3—, the matrix $\mathbf{A}_1$ is of $6 \times 3p$, and the system is underdetermined whenever $p > 2$. This situation normally arises in walking machines and mechanical hands since they cannot usually exert torques at the foot/ground or fingertip/object contact.

4.2.4 Calculating Actuator and Constraint Wrenches

It is possible to use the most compact formulation of the dynamics equations which includes only the contact wrenches as unknowns, and yet still calculate the corresponding actuator and constraint wrenches. For example, once the contact wrenches are solved for in eqs.(4.1), (4.27a) and (4.27b), the actuator torques can be found using eq.(4.20). Since there are many possible solutions to these equations, we can conclude that there also exist many sets of actuator torques which will yield the same payload motion.

Another relationship which is of particular interest is one which allows us to find the constraint wrenches from a given set of contact wrenches. The vector $\hat{\mathbf{w}}_i$ containing the constraint *and* actuator wrenches in the i-th parallel path can be found in a manner analogous to that used for the actuator torques only. That is,

$$\hat{\mathbf{w}}_i = \hat{\mathbf{w}}'_i + \mathbf{F}_i^T \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \tag{4.29a}$$

$$\hat{\mathbf{w}}_i = \begin{bmatrix} \mathbf{n}_{c1} \\ \mathbf{f}_{c1} \\ \vdots \\ \mathbf{n}_{cg_{pi}} \\ \mathbf{f}_{cg_{pi}} \end{bmatrix}, \qquad \mathbf{F}_i^T = \begin{bmatrix} \mathbf{1} & \mathbf{R}_1 \\ \mathbf{0} & \mathbf{1} \\ \vdots & \vdots \\ \mathbf{1} & \mathbf{R}_{g_{pi}} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{4.29b}$$

where $\hat{\mathbf{w}}'_i$ contains the wrenches due only to the weight and inertia of subchain i. This term is calculated by the inverse dynamics algorithm of serial chains found in Appendix A. The

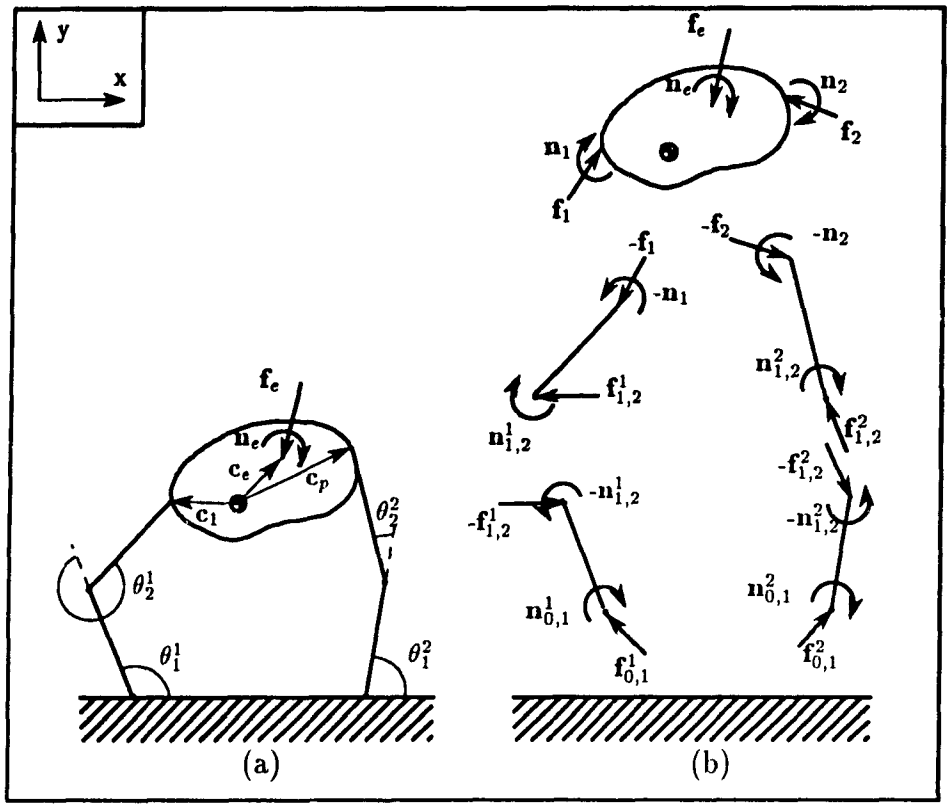

Figure 4.3 *A Planar Two-Fingered Hand Grasping an Object*

second term on the right-hand side of eq.(4.29a) is the contribution due to the contact wrench acting at the tip of the i-th path. The components $\mathbf{n}_{cj}$ and $\mathbf{f}_{cj}$ are vectors of dimension 3 which represent the moments and forces acting at the j-th joint of the i-th path, respectively. Matrix $\mathbf{F}_i^T$ maps the contact wrench into joint wrenches, while matrix $\mathbf{R}_k$ is defined analogously to C_k in eq.(4.8c) but with $\mathbf{c}_k$ replaced by $\mathbf{r}_k$, the vector from the k-th joint of the i-th path to the tip of that path. Finally, g_{pi} is the number of joints in the i-th parallel path.

4.2.5 Example

The three methods of formulating the dynamics equations are now applied to the two-fingered planar mechanical hand shown in Figure 4.3 in order to highlight their differences. The first method involves writing the three force and moment balance

equations for each of the 5 moving bodies in the system. Using the notation shown in Figure 4.3, we have

$$f_1^x + f_2^x = m_o a_o^x \tag{4.30a}$$

$$f_1^y + f_2^y = m_o(a_o^y + g) \tag{4.30b}$$

$$-f_1^x c_1^y + f_1^y c_1^x - f_2^x c_2^y + f_2^y c_2^x = I_o \dot{\omega}_o \tag{4.30c}$$

$$f_{0,1}^{1x} - f_{1,2}^{1x} = m_1^1 a_1^{1x} \tag{4.31a}$$

$$f_{0,1}^{1y} - f_{1,2}^{1y} = m_1^1(a_1^{1y} + g) \tag{4.31b}$$

$$n_{0,1}^1 - n_{1,2}^1 - f_{0,1}^{1x}\hat{c}_{1,0}^{1y} + f_{0,1}^{1y}\hat{c}_{1,0}^{1x} + f_{1,2}^{1x}\hat{c}_{1,2}^{1y} - f_{1,2}^{1y}\hat{c}_{1,2}^{1x} = I_1^1 \dot{\omega}_1^1 \tag{4.31c}$$

$$f_{1,2}^{1x} - f_1^x = m_2^1 a_2^{1x} \tag{4.32a}$$

$$f_{1,2}^{1y} - f_1^y = m_2^1(a_2^{1y} + g) \tag{4.32b}$$

$$n_{1,2}^1 - f_{1,2}^{1x}\hat{c}_{2,1}^{1y} + f_{1,2}^{1y}\hat{c}_{2,1}^{1x} + f_1^x\hat{c}_{2,3}^{1y} - f_1^y\hat{c}_{2,3}^{1x} = I_2^1 \dot{\omega}_2^1 \tag{4.32c}$$

$$f_{0,1}^{2x} - f_{1,2}^{2x} = m_1^2 a_1^{2x} \tag{4.33a}$$

$$f_{0,1}^{2y} - f_{1,2}^{2y} = m_1^2(a_1^{2y} + g) \tag{4.33b}$$

$$n_{0,1}^2 - n_{1,2}^2 - f_{0,1}^{2x}\hat{c}_{1,0}^{2y} + f_{0,1}^{2y}\hat{c}_{1,0}^{2x} + f_{1,2}^{2x}\hat{c}_{1,2}^{2y} - f_{1,2}^{2y}\hat{c}_{1,2}^{2x} = I_1^2 \dot{\omega}_1^2 \tag{4.33c}$$

$$f_{1,2}^{2x} - f_2^x = m_2^2 a_2^{2x} \tag{4.34a}$$

$$f_{1,2}^{2y} - f_2^y = m_2^2(a_2^{2y} + g) \tag{4.34b}$$

$$n_{1,2}^2 - f_{1,2}^{2x}\hat{c}_{2,1}^{2y} + f_{1,2}^{2y}\hat{c}_{2,1}^{2x} + f_2^x\hat{c}_{2,3}^{2y} - f_2^y\hat{c}_{2,3}^{2x} = I_2^2 \dot{\omega}_2^2 \tag{4.34c}$$

These equations must be solved for 16 unknowns: 4 components of the contact forces, f_1^x, f_1^y, f_2^x and f_2^y; 4 actuator torques, $n_{0,1}^1$, $n_{1,2}^1$, $n_{0,1}^2$ and $n_{1,2}^2$; and 8 internal reaction force components, $f_{0,1}^{1x}$, $f_{0,1}^{1y}$, $f_{1,2}^{1x}$, $f_{1,2}^{1x}$, $f_{0,1}^{2x}$, $f_{0,1}^{2x}$, $f_{1,2}^{2x}$ and $f_{1,2}^{2x}$. Thus, the system has one more unknown than equations, which concurs with the fact that it has a mobility of three, controlled by 4 actuators—i.e., an actuation redundancy of one.

The second method entails writing the three force and moment balance equations for the grasped object and the equations governing the motion of each finger acting independently. The Jacobian of each two-link finger is required and can be written as

$$\mathbf{J}_{si} = \begin{bmatrix} j^i_{11} & j^i_{12} \\ j^i_{21} & j^i_{22} \end{bmatrix} \tag{4.35a}$$

where

$$j^i_{11} = l^i_1 \sin\theta^i_1 - l^i_2 \sin(\theta^i_1 + \theta^i_2) \tag{4.35b}$$

$$j^i_{12} = -l^i_2 \sin(\theta^i_1 + \theta^i_2) \tag{4.35c}$$

$$j^i_{21} = l^i_1 \cos\theta^i_1 + l^i_2 \cos(\theta^i_1 + \theta^i_2) \tag{4.35d}$$

$$j^i_{22} = l^i_2 \cos(\theta^i_1 + \theta^i_2) \tag{4.35e}$$

and l^i_j is the length of link j in finger i, while θ^i_j is the joint angle of joint j in finger i, as shown in Figure 4.3(a). Thus, we obtain the following set of equations

$$f^x_1 + f^x_2 = m_o a^x_o \tag{4.36a}$$

$$f^y_1 + f^y_2 = m_o(a^y_o + g) \tag{4.36b}$$

$$-f^x_1 c^y_1 + f^y_1 c^x_1 - f^x_2 c^y_2 + f^y_2 c^x_2 = I_o \dot{\omega}_o \tag{4.36c}$$

$$j^1_{11} f^x_1 + j^1_{21} f^y_1 - n^1_{0,1} = -n^{1\prime}_{0,1} \tag{4.37a}$$

$$j^1_{12} f^x_1 + j^1_{22} f^y_1 - n^1_{1,2} = -n^{1\prime}_{1,2} \tag{4.37b}$$

$$j^2_{11} f^x_2 + j^2_{21} f^y_2 - n^2_{0,1} = -n^{2\prime}_{0,1} \tag{4.38a}$$

$$j^2_{12} f^x_2 + j^2_{22} f^y_2 - n^2_{1,2} = -n^{2\prime}_{1,2} \tag{4.38b}$$

where $n^{\prime}_{j,k}$ is the actuator torque due only to inertial effects applied by link j on link k in finger i. These equations must be solved for 8 unknowns: 4 components of the contact forces, f^x_1, f^y_1, f^x_2 and f^y_2; and the 4 actuator torques, $n^1_{0,1}$, $n^1_{1,2}$, $n^2_{0,1}$ and $n^2_{1,2}$.

Finally, using the third method, we need only write the three force and moment balance equations for the grasped object:

$$f_1^x + f_2^x = m_o a_o^x \tag{4.39a}$$

$$f_1^y + f_2^y = m_o(a_o^y + g) \tag{4.39b}$$

$$-f_1^x c_1^y + f_1^y c_1^x - f_2^x c_2^y + f_2^y c_2^x = I_o \dot{\omega}_o \tag{4.39c}$$

and solving for the 4 unknown contact force components: f_1^x, f_1^y, f_2^x and f_2^y.

It should be apparent that no matter which formulation is used for the dynamics equations of the system, the resulting $m \times n$ matrix $\mathbf{A}_1$ will be rectangular with $n - m = r$. Thus, in all three of the above cases, the number of unknowns exceeds the number of equations by one—the actuation redundancy in the system.

It should also be apparent that the matrix $\mathbf{A}_1$ is a function of the geometry of the system. Therefore, in general, this matrix is time-varying, although, with certain formulations of particular problems, it is constant for each constant topology of the system. In the case of a walking machine for which the equations of motion have been formulated using the method of §4.2.3, the entries of matrix $\mathbf{A}_1$ will vary whenever motion is present since the location of the foot/ground contact points will vary in relation to the machine's mass centre. On the other hand, if the same formulation of the dynamics equations is used for multiple arms manipulating a common object, matrix $\mathbf{A}_1$ will stay constant as long as the contact points do not move on the grasped object. Thus, in the latter case, if the desired solution is one which minimizes a norm of $\mathbf{x}$, it need only be found once each time a new set of contact points is established.

4.3 The Effect of Changes in Topology

It is useful to investigate in more detail the changes which occur in the system, e.g., as given by eqs.(4.1) and (4.21a) to (4.24b), upon changes in topology. In the previous

sections, we dealt solely with the dynamics of the parallel kinematic subchain. In fact, the systems under study are made up, at any given instant, of the parallel *and* the star subchains; where the latter represent the fingers, legs or manipulators which are not in contact with the common object or ground. In the discussion which follows, it is assumed that any given path can be open, in which case that path acts as a serial chain, or closed, in which case the path becomes part of the parallel chain. It is also assumed that the system of dynamical equations describes *all* kinematic chains in the system, including open and closed ones. Thus, the vector of forces is written as follows

$$\mathbf{x} = [\mathbf{x}_1 \quad \dots \quad \mathbf{x}_l \quad \dots \quad \mathbf{x}_p]^T \tag{4.40}$$

where $\mathbf{x}_i$ contains $\mathbf{w}_i$, the contact wrench at the tip of subchain i, and $\boldsymbol{\tau}_{ai}$, the vector of actuator torques in subchain i. When a subchain is open, e.g., a finger is not in contact with the object, the corresponding $\mathbf{w}_i$ is forced to be zero by zero entries in the corresponding columns of $\mathbf{A}_1$. Furthermore, for each open subchain there will be a corresponding subsystem of equations within the system given by eq.(4.24a) which will be decoupled from the remaining equations in the system. When one of the open subchains becomes part of the closed chain, there is an increase in the number of non-zero contact wrenches in the system—i.e., the contact wrench now acting at the tip of the previously-open subchain; the corresponding columns of $\mathbf{A}_1$ will no longer be zero; and the corresponding subsystem of equations become coupled to the rest of the system. After the change in topology, the system of inverse dynamics equations becomes

$$\mathbf{A}_1'\mathbf{x} = \mathbf{b}_1 \tag{4.41}$$

If we assume that the change in topology is one where the previously open l-th subchain comes into contact with the object, the form of the $\mathbf{A}_1$ and $\mathbf{A}_1'$ matrices will be

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{B}_1 & \cdots & \mathbf{0} & \cdots & \mathbf{B}_p \\ \mathbf{D}_1 & \cdots & \mathbf{0} & \cdots & \mathbf{D}_p \\ \widetilde{\mathbf{A}}_1 & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & & & \vdots \\ \mathbf{0} & \cdots & \mathbf{G} & \cdots & \mathbf{0} \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & \widetilde{\mathbf{A}}_p \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \mathbf{0}_{g'_{pal} \times 6} & -\mathbf{1}_{g'_{pal}} \\ \mathbf{0}_{g'_{pul} \times 6} & \mathbf{0}_{g'_{pul} \times g'_{pal}} \end{bmatrix} \tag{4.42a}$$

$$\Downarrow$$

$$\mathbf{A}'_1 = \begin{bmatrix} \mathbf{B}_1 & \cdots & \mathbf{B}_l & \cdots & \mathbf{B}_p \\ \mathbf{D}_1 & \cdots & \mathbf{D}_l & \cdots & \mathbf{D}_p \\ \widetilde{\mathbf{A}}_1 & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & & & \vdots \\ \mathbf{0} & \cdots & \widetilde{\mathbf{A}}_l & \cdots & \mathbf{0} \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & \widetilde{\mathbf{A}}_p \end{bmatrix}, \quad \widetilde{\mathbf{A}}_l = \begin{bmatrix} \mathbf{J}^T_{sal} & -\mathbf{1}_{g'_{pal}} \\ \mathbf{J}^T_{sul} & \mathbf{0}_{g'_{pul} \times g'_{pal}} \end{bmatrix} \tag{4.42b}$$

It becomes clear that, when a change in topology occurs, the system of inverse dynamics equations changes discontinuously, thereby causing a discontinuous change in its solution. Furthermore, since this is a characteristic of the dynamics of the system, and not of the optimization technique, *it will occur irrespective of the type of optimization being performed on the system.* It should be noted that the system given by eq.(4.42a) has been written as such to point out the transition which occurs upon changes in topology. In fact, the system corresponding to the open subchain l in $\mathbf{A}_1$ can be solved independently, while this is not the case with $\mathbf{A}'_1$. Practically speaking, when the larger system decouples into two smaller systems as in this case, it is always computationally more efficient to solve the two systems independently.

4.3.1 Reduction of Solution Discontinuities Upon Changes in Topology

The preceding section showed that the solution to the system of dynamics equations will tend to change discontinuously upon changes in the topology of the system. This will, in turn, result in discontinuous commands being sent to the actuators in the

system. The actuators are not expected to be able to respond to these discontinuous commands and a method of reducing the solution discontinuities is required. The following method for doing this makes use of the fact that any solution to the system given by eq.(4.42a) is also a solution to eq.(4.42b)—that is, a solution in which no wrench is exerted at the tip of the l-th subchain. If we denote the optimal solutions to $\mathbf{A}_1\mathbf{x} = \mathbf{b}_1$ and $\mathbf{A}_1'\mathbf{x} = \mathbf{b}_1$ as $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively, we can write:

$$\mathbf{A}_1'\mathbf{x}_1 = \mathbf{b}_1, \qquad \mathbf{A}_1'\mathbf{x}_2 = \mathbf{b}_1 \tag{4.43}$$

Taking a convex combination (Roberts and Varberg, 1973) of $\mathbf{x}_1$ and $\mathbf{x}_2$ and premultiplying it by $\mathbf{A}_1'$, we find

$$\mathbf{A}_1'[\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2] = \alpha\mathbf{A}_1'\mathbf{x}_1 + (1-\alpha)\mathbf{A}_1'\mathbf{x}_2 = \alpha\mathbf{b}_1 + (1-\alpha)\mathbf{b}_1 = \mathbf{b}_1 \tag{4.44a}$$

where

$$0 \leq \alpha \leq 1 \tag{4.44b}$$

Thus, any convex combination of two solutions is also a solution. A sequence of suboptimal convex combinations of $\mathbf{x}_1$ and $\mathbf{x}_2$ which never exceeds a maximum Euclidean norm of the rate of change of $\mathbf{x}$ can be found as follows:

$$\Delta\mathbf{x}^k = (\mathbf{x}_2 - \mathbf{x}_{out}^{k-1})/\Delta t \tag{4.45a}$$

$$||\Delta\mathbf{x}^k||_{max} = \mathrm{LIM}(||\Delta\mathbf{x}^k||) \tag{4.45b}$$

$$\mathbf{x}_{out}^k = \mathbf{x}_{out}^{k-1} + \frac{||\Delta\mathbf{x}^k||_{max}}{||\Delta\mathbf{x}^k||}\Delta\mathbf{x}^k \tag{4.45c}$$

where the superscript k denotes a value at the present time step, while $k-1$ denotes a value at the previous time step. The function LIM($\star$) serves to limit the value of the scalar in the brackets to a predetermined value. The solution vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ and the series of intermediate solutions are shown graphically in Figure 4.4.

The principal problem with this method of smoothing the solution is that it is only applicable as long as the entries of $\mathbf{A}_1$ and $\mathbf{b}_1$ do not change while the smoothing

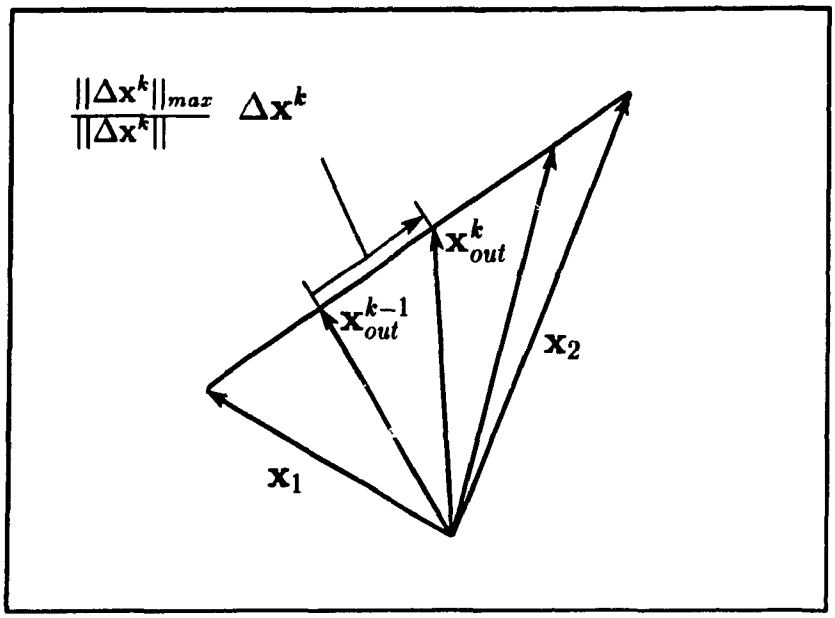

Figure 4.4 *Geometric Interpretation of Scheme for Reduction of Discontinuities*

operation is occurring other than the change represented by eqs.(4.42a) and (4.42b). Thus, the system would have to be brought to a stop and held *immobile* whenever a change in topology occurred.

4.4 Inequality Constraints

Any solution to eq.(4.1) must obviously satisfy that system of equations. However, in many systems composed of redundantly-actuated robotic devices, it is important that the solution also satisfy certain *additional* constraints. The following sections discuss a number of important considerations which include:

1. The limitations of kinematic pairs which cannot sustain bidirectional forces (see §2.3);

2. The maximum lateral forces which can be resisted by passive frictional contact;

3. The maximum torque output limitations of the actuators;

4. The maximum force and moment capabilities of the joints; and

5. Limits on the time-discontinuity which is permitted of the solution.

All these constraints can be formulated mathematically as linear *inequalities*, rather than as equalities. These inequalities can then be assembled and written compactly as

$$\mathbf{A}_2\mathbf{x} \geq \mathbf{b}_2 \tag{4.46}$$

where this equation represents a short-hand notation for a component-wise system of inequalities. Namely, each component of the vector on the left-hand side must be greater than the corresponding component on the right-hand-side.

4.4.1 Unilateral Contacts

When multiple manipulators handle a common payload, their end-effectors are assumed to *grasp* the payload, and be capable of exerting a fully general wrench on it. There are no constraints implicit in this type of contact. By contrast, certain kinematic pairs discussed in §2.3 require some components of the wrench exerted across these pairs to be unidirectional. A characteristic of mechanical hands and walking machines is that these types of pairs frequently exist at the interface between the individual robotic systems (i.e., fingers or legs). Thus, whereas an active, *grasping* end effector can push or pull an object, a fingertip can only push. By including this limitation in the optimization as a simple bound on f_{N_i}, the normal force at the i-th contact point:

$$f_{N_i} \geq 0 \tag{4.47}$$

we can guarantee that, if a solution is found, it will satisfy *force closure* on the system (Salisbury and Roth, 1983).

4.4.2 The Friction Cone and Pyramids

A consideration of the limitations of passive frictional contact is related to the above discussion. Whereas the active grip in multiple-robot systems is assumed to

be strong enough to generate any desired tangential force through the contact and resist slippage, this assumption cannot be made with the passive contact found in mechanical hands and walking machines. In these systems, we must ensure that the magnitude of the tangential force does not exceed the product of the normal force by the coefficient of static friction, μ. This limitation has been graphically described as the requirement to stay within a *friction cone* (Orin and Oh, 1981; Kerr and Roth, 1986) with its apex at the contact point, and its axis normal to the contact surface. Since the cone is a quadratic surface, including this constraint exactly would require the use of a quadratic inequality. However, since the optimization problem is eased considerably by considering only linear constraints, we follow the suggestion of Orin and Oh (1981) and Kerr and Roth (1986) to replace the friction cone by a piecewise linear k-sided friction pyramid which can be described by linear inequalities. Two obvious choices for this pyramid are: one which circumscribes the friction cone or one which is inscribed by the friction cone, shown graphically in Figure 4.5. Thus for the case of $k = 4$, this constraint can be included as:

$$|f_{xi}| \leq \mu' f_{Ni} \tag{4.48a}$$

$$|f_{yi}| \leq \mu' f_{Ni} \tag{4.48b}$$

which can be expanded into the following four inequality constraints:

$$\mu' f_{Ni} + f_{xi} \geq 0 \tag{4.49a}$$

$$\mu' f_{Ni} - f_{xi} \geq 0 \tag{4.49b}$$

$$\mu' f_{Ni} + f_{yi} \geq 0 \tag{4.49c}$$

$$\mu' f_{Ni} - f_{yi} \geq 0 \tag{4.49d}$$

where f_{xi} and f_{yi} are the tangential forces acting at the i-th contact point, $\mu' = \mu$ for the circumscribed pyramid, $\mu' = \sqrt{2}\mu/2$ for the more conservative inscribed pyramid, and μ is the radius of the friction cone. The friction cone may also be more accurately represented by the k-sided pyramid, with $k > 4$.

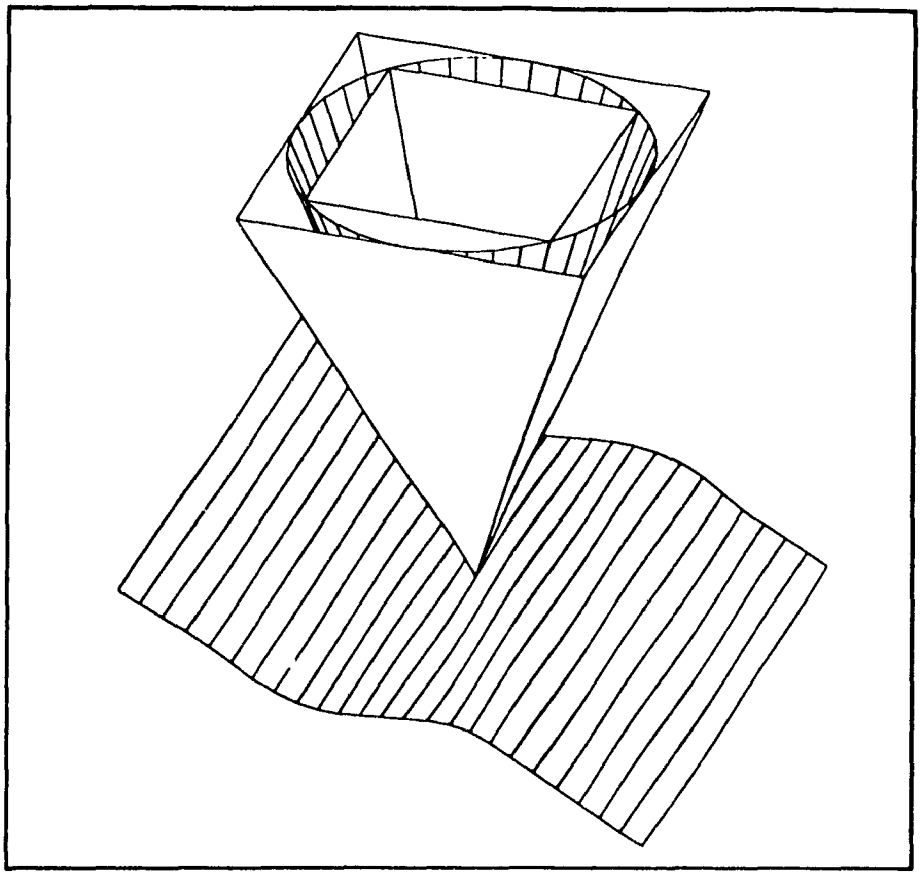

Figure 4.5 *Graphical Representation of the Friction Cone*

4.4.3 Actuator Limitations

A further source of inequality constraints which applies equally to multiple-robot workcells, mechanical hands and walking machines, is the consideration of limits on the torque that can be exerted by an actuator. The constraint on the actuator torques can be included as:

$$|\tau_i| \leq \tau_{lim} \tag{4.50}$$

where τ_{lim} is the vector of maximum torques which the actuators can produce. Equation (4.50) can be expanded into two linear inequalities as follows:

$$\tau_i \geq -\tau_{lim} \tag{4.51a}$$

$$-\tau_i \geq -\tau_{lim} \tag{4.51b}$$

The form given by eqs.(4.51a) and (4.51b) is sufficient if the actuator torques are included in the vector of design variables, as in the first two methods of formulating the dynamics equations. If the actuator torques are not included in this vector, eqs.(4.51a) and (4.51b) must be transformed into limits on the contact wrench by substituting in eq.(4.20) to obtain

$$\mathbf{J}_{si}^T \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \geq -\boldsymbol{\tau}_i' - \boldsymbol{\tau}_{lim} \tag{4.52a}$$

$$-\mathbf{J}_{si}^T \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \geq \boldsymbol{\tau}_i' - \boldsymbol{\tau}_{lim} \tag{4.52b}$$

Thus, the method of formulating the dynamics equations given in §4.2.3 would require the use of the inequalities given by eqs.(4.52a) and (4.52b) rather than those given by eqs.(4.51a) and (4.51b).

4.4.4 Limits on the Constraint Wrenches

In addition to limiting the actuator torques, we may wish to limit the constraint wrenches acting at all the joints. This can be done by limiting the vector of actuator and constraint wrenches $\hat{\mathbf{w}}_i$ as follows:

$$|\hat{\mathbf{w}}_i| \leq \hat{\mathbf{w}}_{lim} \tag{4.53}$$

where $\hat{\mathbf{w}}_{lim}$ is the vector of maximum wrenches which the joints can sustain. Equation (4.53) can be expanded into two linear inequalities as follows:

$$\hat{\mathbf{w}}_i \geq -\hat{\mathbf{w}}_{lim} \tag{4.54a}$$

$$-\hat{\mathbf{w}}_i \geq -\hat{\mathbf{w}}_{lim} \tag{4.54b}$$

The form given by eqs.(4.54a) and (4.54b) is sufficient if the constraint wrenches are included in the vector of design variables, as in the first method of formulating the dynamics equations. If the constraint wrenches are not included in this vector, eqs.(4.54a)

and (4.54b) must also be transformed into limits on the contact wrench, this time by substituting in eq.(4.29a) to obtain

$$\mathbf{F}_i^T \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \geq -\hat{\mathbf{w}}_i' - \hat{\mathbf{w}}_{lim} \tag{4.55a}$$

$$-\mathbf{F}_i^T \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \geq \hat{\mathbf{w}}_i' - \hat{\mathbf{w}}_{lim} \tag{4.55b}$$

Thus, the last two methods of formulating the dynamics equations would require the use of the inequalities given by eqs.(4.55a) and (4.55b) rather than those given by eqs.(4.54a) and (4.54b).

4.4.5 Limiting Solution Discontinuities Upon Changes in Topology

It was found in §4.3 that changes in the topology are discrete and cause discontinuous changes in the coefficients of the dynamics equations, as represented by matrix $\mathbf{A}_i$. These invariably lead to discontinuous changes in the solution to the force distribution problem found with any optimization technique, unless the optimization problem is modified to prevent them. The actuators in the system are not expected to be able to respond to these discontinuous commands and a method of reducing the solution discontinuities is required. Section 4.3.1 proposed a technique for smoothing the force solution which requires the system to remain immobile while the smoothing takes place – an excessively restrictive requirement. The following method makes use of inequality constraints to limit the rate of change of the actuator torques and is much less restrictive on the allowable motion of the system. Since the optimization problem must be solved at each instant in time, at any given instant the solution from the previous instant is available and can be denoted as $\mathbf{x}^-$, where the superscript (-) indicates values at the previous instant. Let us denote one element of $\mathbf{x}$ by x_i and the corresponding element of $\mathbf{x}^-$ by x_i^-. The difference between x_i and x_i^- can be limited by imposing the following limit on the magnitude of the change in this component of the solution:

$$|x_i - x_i^-| \leq \Delta_i \tag{4.56}$$

If the actuator torques are components of the vector $\mathbf{x}$, the rate of change of each component of $\mathbf{x}$ which represents an actuator torque can be limited by expanding eq.(4.56) into the following linear inequality constraints

$$-x_i \geq -\Delta_i - x_i^- \tag{4.57a}$$

$$x_i \geq -\Delta_i + x_i^- \tag{4.57b}$$

Since these inequalities are of the form given by eq.(4.46), they can be simply appended to that system as follows:

$$\mathbf{A}_2'\mathbf{x} \geq \mathbf{b}_2' \tag{4.58a}$$

where

$$\mathbf{A}_2' = \begin{bmatrix} \mathbf{A}_2 \\ \widehat{\mathbf{A}}_2 \end{bmatrix} \qquad \mathbf{b}_2' = \begin{bmatrix} \mathbf{b}_2 \\ \widehat{\mathbf{b}}_2 \end{bmatrix} \tag{4.58b}$$

$$\widehat{\mathbf{A}}_2 = \begin{bmatrix} -1 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \dots & 0 & -1 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} \qquad \widehat{\mathbf{b}}_2 = \begin{bmatrix} -\Delta_1 - x_1^- \\ -\Delta_1 + x_1^- \\ \vdots \\ -\Delta_q - x_q^- \\ -\Delta_q + x_q^- \end{bmatrix} \tag{4.58c}$$

where q is the total number of elements in $\mathbf{x}$ on which limits in discontinuity are imposed.

If the formulation of the inverse dynamics equations presented in §4.2.3 is used, vector $\mathbf{x}$ does not include the actuator torques. In this case, if it is desired to impose limits on the discontinuity allowed in the torque commanded from a given actuator, we must proceed as follows:

$$|\tau_{ij} - \tau_{ij}^-| \leq \Delta_{ij} \tag{4.59}$$

where τ_{ij} represents actuator torque j in path i, while τ_{ij}^- represents that same actuator torque at the previous time step. If we expand this inequality as was done in eq.(4.57a) and (4.57b), and substitute in eq.(4.20), we obtain

$$-\mathbf{j}_{ij}^T \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \geq -\Delta_{ij} - \tau_{ij}^- + \tau_{ij}' \tag{4.60a}$$

$$\mathbf{j}_{ij}^T \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \geq -\Delta_{ij} + \tau_{ij}^- - \tau_{ij}' \tag{4.60b}$$

where $\mathbf{j}_{ij}$ is the j-th column of $\mathbf{J}_{pi}$, the Jacobian of the path which contains the actuator in question, and τ'_{ij} is the portion of τ_{ij} due only to inertial effects. These inequalities are again in the form given by eq.(4.46), and they can be simply appended to that system to obtain eqs.(4.58a) and (4.58b) with

$$\widehat{\mathbf{A}}_2 = \begin{bmatrix} -\mathbf{j}_{11}^T & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{j}_{11}^T & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & \ddots & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{j}_{pr}^T \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{j}_{pr}^T \end{bmatrix}, \qquad \hat{\mathbf{b}}_2 = \begin{bmatrix} -\Delta_{11} - \tau_{11}^- + \tau'_{11} \\ -\Delta_{11} + \tau_{11}^- - \tau'_{11} \\ \vdots \\ -\Delta_{pr} - \tau_{pr}^- + \tau'_{pr} \\ -\Delta_{pr} + \tau_{pr}^- - \tau'_{pr} \end{bmatrix} \tag{4.61}$$

where r is the number of actuators in each path.

This method for smoothing discontinuities in the force solution is more general than the one proposed in §4.3.1 since it does not require the system to remain immobile upon changes in topology. It also fits well into the existing framework of optimization since it simply involves appending further inequalities to the existing ones. Since the CPU time of the solution method for inequality-constrained quadratic optimization which will be presented in §5.4.2 depends only on the the number of constraints which are *active* at the solution—i.e., satisfied as equalities—, the addition of new inequalities will increase the CPU time only when these are active—i.e., at discontinuities. At discontinuities, only one of these will be active at any given instant—the one corresponding to the element of $\mathbf{x}$ which undergoes the largest change at the discontinuity , and the CPU time will therefore only be slightly increased.

The primary limitation of this technique is that Δ_i and Δ_{ij} cannot be chosen arbitrarily small—if they are too restrictive, the new inequality constraints will cause the optimization problem to become infeasible. That is, no solution will exist which can satisfy the dynamics equations while having the prescribed degree of smoothness. Furthermore, the values of Δ_i and Δ_{ij} which cause the optimization problem to become infeasible cannot be known *a priori*. This is particularly true when impacts occur. Actual use of this technique might involve an iterative process of imposing a certain limit; seeing if the problem is feasible; relaxing the limit if it is not; and repeating the process. Since

this would be highly inefficient in a real-time implementation, yet another method was developed to smooth the discontinuities. This last method will be presented in §6.2.8, along with a numerical example applying both smoothing techniques to smooth the solution discontinuity occurring when the foot of a walking machine comes into contact with the ground.

Chapter 5

Optimization Techniques

In the preceding chapter, the equations of motion for redundantly-actuated robotic systems were formulated. It was found that, if the motion of the system is prescribed and the wrenches required to achieve this motion must be calculated, the system of linear equations which must be solved is underdetermined, thereby admitting a multiplicity of solutions. In any problem which allows a choice of solutions, it is only natural to want to choose the 'best' one—opt mization techniques provide a mathematical framework for doing this.

The most general optimization problem is one in which we wish to optimize an arbitrary objective function while respecting certain specified constraints. The constraints are usually easily formulated mathematically because they are imposed by quantifiable physical phenomena. For example, the equations of motion discussed in the preceding chapter constitute equality constraints to the optimization problem which will now be formulated. By contrast, the choice of the objective function is more arbitrary simply because many different criteria are desirable. The choice of both objective and constraint functions must also be made in view of the techniques which will be required to solve the resulting problem. For example, if the exact mathematical statement of a constraint or objective function is complex, it will necessitate the use of cumbersome optimization techniques which, in turn, are less likely to converge and require more computing power.

When this latter resource is limited, as in on-board control units in walking machines, it is desirable to simplify the function in question by choosing a simpler but more restrictive expression which, if satisfied, will guarantee that the original expression is also satisfied.

In the context of optimization, the goal of the designer is to formulate a reasonably accurate mathematical representation of a physically significant problem which can be solved using the available resources. It will be shown in this chapter that quadratic programming is well adapted to this task as it allows the optimization of physically meaningful objective functions. Furthermore, when the constraints are linear, numerical techniques exist to solve the optimization problem in real-time using present-day hardware.

Section 5.1 will describe the mathematical framework and terminology of optimization, with emphasis on the characterization of optimality conditions. These concepts will be particularly useful when we later compare the behavior of the solutions provided by various optimization techniques. Section 5.2 will investigate the features of particular optimization problems which we can expect to solve in real-time—linear and quadratic programs. In §5.3, the concept of duality will be briefly reviewed since it plays a significant role in the most efficient numerical techniques which exist both in linear and quadratic programming. Finally, §5.4 will introduce the various numerical techniques which were implemented in the present work.

5.1 Optimality—General Objective Functions

The mathematical statement of a general constrained optimization problem is

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{5.1a}$$

$$\text{subject to} \quad h_i(\mathbf{x}) = 0 \qquad i = 1, \ldots, m_{eq} \tag{5.1b}$$

$$g_j(\mathbf{x}) \geq 0 \qquad j = m_{eq} + 1, \ldots, m \tag{5.1c}$$

where $\mathbf{x}$ is an n-dimensional vector of *design variables*. Function $f(\mathbf{x})$ is normally called

the *objective function*, while eq.(5.1b) represents a set of equality constraints and eq.(5.1c) represents a set of inequality constraints. Maximization of a function $F(\mathbf{x})$ can be performed in the above framework by minimizing $-F(\mathbf{x})$. Similarly, an inequality of the form $G_i(\mathbf{x}) \leq 0$ can be included as $-G_i(\mathbf{x}) \geq 0$. For convenience, we will limit our discussion to objective functions which are twice-continuously differentiable throughout the region of interest.

Our goal is to find a minimum point, denoted by $\mathbf{x}^*$, in the space of all values of the vector $\mathbf{x}$ (called the *design space*) which is the solution to the problem stated in eqs.(5.1a) to (5.1c). It is therefore useful to review the *optimality conditions* which this point must satisfy in order to be a minimum. This will be done by starting with the optimality conditions for an unconstrained objective function—i.e., eq.(5.1a) alone—and then introducing the equality and inequality constraints. This sequence closely follows that used by Gill *et al.* (1981).

Since conditions for the existence and uniqueness of optima are most conveniently written in terms of derivatives of functions, it is useful to define the *gradient* of a function $f(\mathbf{x})$ as

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \partial f(\mathbf{x})/\partial x_1 \\ \vdots \\ \partial f(\mathbf{x})/\partial x_n \end{bmatrix} \tag{5.2}$$

and the *Hessian* matrix of that function as

$$\mathbf{H}(f(\mathbf{x})) = \begin{bmatrix} \partial^2 f(\mathbf{x})/\partial x_1^2 & \partial^2 f(\mathbf{x})/\partial x_1 \partial x_2 & \dots & \partial^2 f(\mathbf{x})/\partial x_1 \partial x_n \\ \partial^2 f(\mathbf{x})/\partial x_2 \partial x_1 & \partial^2 f(\mathbf{x})/\partial x_2^2 & \dots & \partial^2 f(\mathbf{x})/\partial x_2 \partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 f(\mathbf{x})/\partial x_n \partial x_1 & \partial^2 f(\mathbf{x})/\partial x_n \partial x_2 & \dots & \partial^2 f(\mathbf{x})/\partial x_n^2 \end{bmatrix} \tag{5.3}$$

We can categorize minima into *global* and *local* ones as follows:

1. A global minimum is one where the value of the objective function at $\mathbf{x}^*$ is less than or equal to that at every point in its δ-neighborhood of the design space, for *all* positive values of δ.

2. A local minimum is one where the value of the objective function at $\mathbf{x}^*$ is less than or equal to that at every point in its δ-neighborhood of the design space, for *some* positive values of δ.

We can also categorize minima as *strong* and *weak* ones:

1. A strong minimum is one where the value of the objective function at $\mathbf{x}^*$ is strictly less than that at every other point in its δ-neighborhood.

2. A weak minimum is a local or global minimum which does not qualify as a strong minimum.

It can be inferred from these definitions that a strong global minimum will have a unique minimum value of the objective function at a *unique* point in the design space. By contrast, a weak global minimum will have a unique value of the objective at many points in the design space. The optimality conditions which follow all relate to local minima but it will be shown in §5.2 that, for certain problems, a local minimum is guaranteed to be the global minimum.

5.1.1 Unconstrained Optimization

Sufficient conditions for a point $\mathbf{x}^*$ to be a strong local minimum of an unconstrained objective function are:

1. $\nabla f(\mathbf{x}^*) = \mathbf{0}$

2. $\mathbf{H}(f(\mathbf{x}^*))$ is positive-definite

Item (1) above is a first-order condition, known as the *normality condition*, which specifies that $\mathbf{x}^*$ must be a *stationary* point in the design space. Item (2) is a second-order condition, known as the *convexity condition*, which ensures that the stationary point is neither a maximum nor a saddle point.

5.1.2 Constrained Optimization

The constraints given by eqs.(5.1b) and (5.1c) are nonlinear in general. However, in Chapter 4, we found that almost all the constraints of interest in the present physical problem are linear—the only exceptions being the friction cone constraints which are quadratic. It was also shown that these latter constraints can be approximated by piecewise-linear friction pyramids, and that, as the number of sides used for these pyramids is increased, they form a closer approximation to the cone. Furthermore, the numerical techniques required to solve linearly-constrained optimization problems are substantially less computationally intensive than those required to solve non-linearly-constrained problems. It was therefore decided to concentrate the present investigation on linearly-constrained optimization problems since *a)* the necessary approximation used is minor, and *b)* the numerical methods available allow real-time execution.

Assuming that we limit ourselves to linear constraints, the general optimization problem given by eqs.(5.1a) to (5.1c) reduces to

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{5.4a}$$

$$\text{subject to} \quad \mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \tag{5.4b}$$

$$\mathbf{A}_2\mathbf{x} \geq \mathbf{b}_2 \tag{5.4c}$$

where $\mathbf{A}_1$ is a matrix of dimension $m_{eq} \times n$, while $\mathbf{A}_2$ is a matrix of dimension $m_{in} \times n$ and $m_{eq} + m_{in} = m$.

5.1.2.1 Linear Equality Constraints

The equality-constrained optimization problem is that given by eqs.(5.4a) and (5.4b). The presence of the equality constraints partitions the design space into two orthogonal subspaces—one *feasible*, the other *infeasible*. The former consists of all the points in the design space which satisfy eq.(5.4b)—denoted as *feasible points*—, while

the latter consists of all points which violate the equation—denoted as *infeasible points.* Similarly, a *feasible direction* in the design space is one which, if taken from a feasible point will necessarily lead to another feasible point. Algebraically, it can be defined as a vector which lies in the nullspace of the constraint matrix $\mathbf{A}_1$. If we let $\mathbf{Z}$ denote a matrix whose columns form a basis for the nullspace of $\mathbf{A}_1$, i.e., $\mathbf{Z}$ is an *orthogonal complement* of $\mathbf{A}_1$, we can write every feasible direction as a linear combination of the columns of $\mathbf{Z}$.

Under linear equality constraints, the sufficient conditions for a point $\mathbf{x}^\star$ in the design space to be a strong local minimum are (Gill *et al.*, 1981):

1. It must satisfy $\mathbf{A}_1\mathbf{x}^\star = \mathbf{b}_1$,
2. $\mathbf{Z}^T\nabla f(\mathbf{x}^\star) = \mathbf{0}$, or equivalently, $\nabla f(\mathbf{x}^\star)$ must lie in the range of $\mathbf{A}_1^T$—i.e., a vector $\boldsymbol{\lambda}$ of *Lagrange multipliers* of dimension m_{eq} must exist so that $\nabla f(\mathbf{x}^\star) = \mathbf{A}_1^T\boldsymbol{\lambda}^\star$,
3. $\mathbf{Z}^T\mathbf{H}(f(\mathbf{x}^\star))\mathbf{Z}$ is positive-definite.

The first item simply states that $\mathbf{x}^\star$ must satisfy the constraints. The second item is directly analogous to the stationarity condition in an unconstrained optimization problem: $\mathbf{Z}^T\nabla f(\mathbf{x}^\star)$, known as the *projected gradient*, is nothing but the projection of the gradient of the objective function onto the feasible subspace and it must vanish for $\mathbf{x}^\star$ to be a *constrained stationary point.* We are not concerned with the gradient in the infeasible directions, since we cannot move in those directions without violating the first item.

The third item is analogous to the second-order condition in the unconstrained optimization problem: $\mathbf{Z}^T\mathbf{H}(f(\mathbf{x}^\star))\mathbf{Z}$, known as the *projected Hessian matrix*, is nothing but the projection of the Hessian matrix of the objective function onto the feasible subspace and it must be positive-definite in order to guarantee that the constrained stationary point is not a maximum or a saddle point. Once again, we are unconcerned with the sign definition of the Hessian along the infeasible directions, since we cannot move in those directions without violating the first item.

It is also important to gain a good understanding of the significance of the Lagrange multipliers $\lambda_i, i = 1, \ldots, m_{eq}$ in the optimization problem and its solution. Essentially, if we think of each equality constraint as a 'bi-directional wall'—a wall on which we can push *and* pull—in the design space, the corresponding Lagrange multiplier tells us how much we are pushing or pulling on that wall. An alternate interpretation, is that the magnitude of the Lagrange multiplier tells us how much the solution would change if its corresponding constraint were removed (Haftka and Kamat, 1985).

5.1.2.2 Linear Inequality Constraints

The inequality-constrained optimization problem is that given by eqs.(5.4a) and (5.4c). The presence of the inequality constraints partitions the design space into *feasible* and *infeasible* regions. The former consists of all the points in the design space which satisfy eq.(5.4c)—denoted as *feasible points*—, while the latter consists of all points which violate the inequality—denoted as *infeasible points*. Each inequality constraint can be thought of as defining a hyperplane in the the design space which separates the feasible from infeasible regions.

If we denote the i-th row of $\mathbf{A}_2$ by $\mathbf{a}_i$ and the i-th component of $\mathbf{b}_2$ by b_i, then at each feasible point, the i-th constraint is said to be *active* if $\mathbf{a}_i^T\mathbf{x} = b_i$; *inactive* if $\mathbf{a}_i^T\mathbf{x} > b_i$. This distinction is important because, if an inequality constraint is active, we can treat it as an equality constraint. The geometrical interpretation of an active constraint is that, if a feasible point lies on the i-th bounding hyperplane, the i-th constraint is active.

Any direction is a feasible direction with respect to an inactive constraint, since it is possible to move a small distance in any direction without violating the constraint. In the case of active constraints, there are two types of feasible directions, denoted by $\mathbf{p}$: the first being a *binding* direction which satisfies $\mathbf{a}_i^T\mathbf{p} = 0$, i.e., the i-th constraint remains active; the second being a *non-binding* direction which satisfies $\mathbf{a}_i^T\mathbf{p} > 0$, i.e., the i-th constraint becomes inactive.

The q active constraints at a given feasible point are denoted by $\widehat{\mathbf{A}}_2\mathbf{x} = \widehat{\mathbf{b}}_2$, where $\widehat{\mathbf{A}}_2$ and $\widehat{\mathbf{b}}_2$ are subsets of $\mathbf{A}_2$ and $\mathbf{b}_2$, respectively. As in the preceding section, we let $\mathbf{Z}$ denote a matrix whose columns form a basis for the nullspace of $\widehat{\mathbf{A}}_2$. Any binding direction can therefore be written as a linear combination of the columns of $\mathbf{Z}$.

Under linear inequality constraints, the sufficient conditions for a point $\mathbf{x}^*$ in the design space to be a strong local minimum are (Gill *et al.*, 1981):

1. It must satisfy $\mathbf{A}_2\mathbf{x}^* \geq \mathbf{b}_2$, with $\widehat{\mathbf{A}}_2\mathbf{x}^* = \widehat{\mathbf{b}}_2$,
2. $\mathbf{Z}^T\nabla f(\mathbf{x}^*) = \mathbf{0}$, or equivalently, $\nabla f(\mathbf{x}^*)$ must lie in the range of $\widehat{\mathbf{A}}_2^T$—i.e., a vector $\boldsymbol{\lambda}$ of Lagrange multipliers of dimension q must exist so that $\nabla f(\mathbf{x}^*) = \widehat{\mathbf{A}}_2^T\boldsymbol{\lambda}^*$,
3. $\lambda_i^* > 0, \qquad i = 1, \ldots, q$,
4. $\mathbf{Z}^T\mathbf{H}(f(\mathbf{x}^*))\mathbf{Z}$ is positive-definite.

Once again, the first item states that $\mathbf{x}^*$ must satisfy the constraints. The second item has the same interpretation as the second optimality condition under equality constraints—that the projected gradient of the objective function vanish at the solution. As was the case under equality constraints, no importance is placed on the gradient of the objective function in the constrained directions, even though we can now move off the constraint, i.e., in a non-binding direction. The reason for this is that, if the constraint is active and its corresponding Lagrange multiplier is positive, then the gradient of the objective function in the non-binding direction is guaranteed to be positive, a proof of which is given in Gill *et al.* (1981). A move in this direction could therefore only lead to an increase in the objective function.

The third item, which is new, specifies that all Lagrange multipliers corresponding to active constraints must be positive, the ones corresponding to inactive constraints being zero or non-existent, depending on the convention. This is a reflection of the fact that the inequality constraints represent 'uni-directional walls' in the design space, i.e., a wall we can push, but not pull, on. When dealing with inequality-constrained

optimization problem, the first-order optimality conditions, i.e., items (2) and (3) above, are often referred to as the *Kuhn-Tucker conditions* (Kuhn and Tucker, 1951).

Finally, the fourth item is analogous to the third optimality condition under equality constraints—a requirement that the projected Hessian of the objective function be positive definite at the solution.

5.2 Optimality—Particular Objective Functions

In general, we are not only interested in finding a local minimum, but a global one. For general objective functions, we must usually content ourselves with a local minimum because a global minimum is so difficult to find (Gill *et al.*, 1981). However, the form of certain objective functions can guarantee that if a minimum is found, it will be global *and* unique—i.e., it will be a strong global minimum. As will be seen in the following sections and the next chapter, these properties are highly desirable in the physical problem under study. In this section, optimality conditions for two particular forms of the objective function, linear and quadratic, will be discussed. Furthermore, these forms will also allow us to use numerical solution techniques which can be expected to be executed in real-time.

5.2.1 Convex Functions

A *convex* function $f(\mathbf{x})$ is one which satisfies the condition that (Roberts and Varberg, 1973):

$$f(\alpha\mathbf{x}_1 + \beta\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + \beta f(\mathbf{x}_2), \qquad 0 < \alpha, \beta < 1, \qquad \alpha + \beta = 1 \tag{5.5}$$

for any two values $\mathbf{x}_1$ and $\mathbf{x}_2$ of the independent variable. The one-dimensional interpretation of this is shown in Figure 5.1—the straight line joining any two points on the curve always lies on or above the curve. Another feature of a convex, twice-continuously differentiable function is that its Hessian matrix is positive semi-definite.

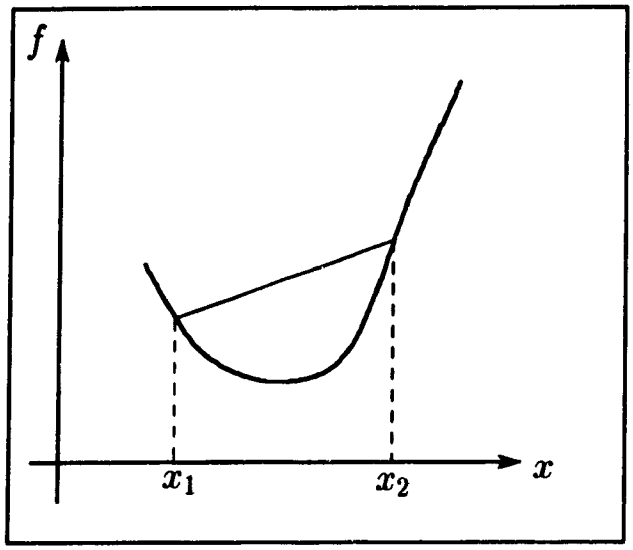

Figure 5.1 *A Unidimensional Convex Function*

A *strictly convex* function satisfies the condition that:

$$f(\alpha \mathbf{x}_1 + \beta \mathbf{x}_2) < \alpha f(\mathbf{x}_1) + \beta f(\mathbf{x}_2), \qquad 0 < \alpha, \beta < 1, \qquad \alpha + \beta = 1 \tag{5.6}$$

Finally, the Hessian matrix of a strictly convex, twice-continuously differentiable function is positive-definite.

5.2.2 Linear Programming

A linear objective function is an example of a particular function which is convex (though not *strictly* convex):

$$f(x) = \mathbf{c}^T \mathbf{x} \tag{5.7}$$

The gradient of this function is $\nabla f = \mathbf{c}$, i.e., a constant vector, and its Hessian matrix is identically the zero matrix As well, the unconstrained function is unbounded and so we must impose constraints in order to find a finite minimum. Because the Hessian matrix of this function is not positive-definite—it is positive-semidefinite—, the second order optimality condition, which requires the projected Hessian matrix to be positive-definite, cannot be satisfied. A point which satisfies the first order optimality conditions

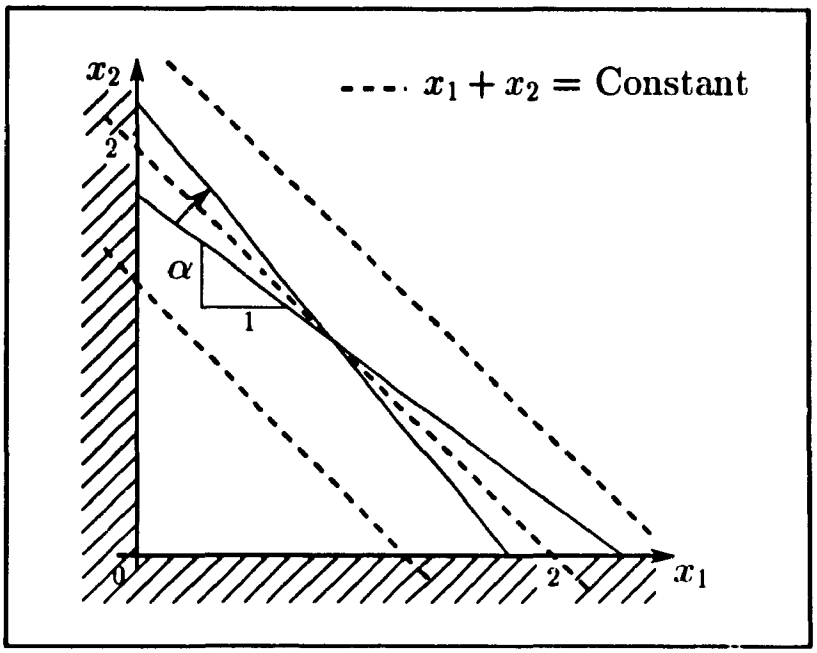

Figure 5.2 *Non-unique Minimum in Linear Programming*

and has a positive-semidefinite Hessian matrix qualifies as a weak minimum. This implies that the minimum value of the objective function may be reached at many contiguous points in the design space—a situation which occurs when the rank of the matrix of active constraints is less than n, the dimension of the vector of design variables (Gill *et al.*, 1981).

Geometrically speaking, this situation corresponds to a case where an active constraint becomes parallel to the objective function. This is exemplified in Figure 5.2 where the following two-dimensional minimization of a linear objective function is performed subject to one linear equality and two linear inequality constraints:

$$\min_{x_1, x_2} \quad x_1 + x_2 \tag{5.8a}$$

$$\text{subject to} \quad \alpha x_1 + x_2 = \alpha + 1 \tag{5.8b}$$

$$x_1 \geq 0, \qquad x_2 \geq 0 \tag{5.8c}$$

The slope of the equality constraint $(-\alpha)$ is varied; when it becomes equal to that of the objective function ($\alpha = 1$), neither of the non-negativity constraints given by eq.(5.8c) is active and the rank of the matrix of active constraints becomes less than

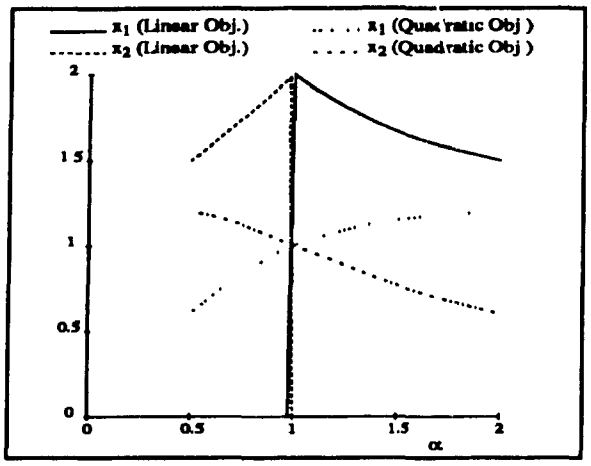

Figure 5.3 *Discontinuity in the Linear-Programming Solution*

two. At this value of α, a unique minimum value of the objective function is reached ($x_1 + x_2 = 2$) at a non-unique set of design variables; namely, any non-negative x_1 and x_2 which satisfy $x_1 + x_2 = 2$. As shown in Figure 5.3, when the components of the solution are plotted as a function of α, they exhibit severe discontinuities. This is known as the phenomenon of 'alternate optima' in linear programming whereby the solution to an n-dimensional linear-programming problem can touch up to n vertices of the feasible region simultaneously (Wilde and Beightler, 1967). Furthermore, in higher-dimensional problems, alternate optima, and hence, non-uniqueness of the solution, are the rule rather than the exception (Wilde and Beightler, 1967; Freund, 1985). Although in the context of force optimization in redundantly-actuated robotic systems this problem has only very recently been discovered (Cheng, 1989; Klein and Kittivatcharapong, 1990), the study of the effect of changes in a programming problem's coefficients is known as 'sensitivity analysis' and is a well-researched area. Freund (1985) presents a thorough analysis of the behavior of linear-programming problems during changes in the coefficients of the constraint matrix.

This has serious implications in the physical problem under study because the presence of a continuously-varying constraint matrix is particularly relevant. As was shown in Chapter 4, matrix $\mathbf{A}_1$ represents the configuration, or geometry, of the

system, and hence, its entries will vary smoothly as the geometry of a given grasp or walk changes, even if the topology of the system does not change. Recall that the solution to the optimization problem is meant to be used as a setpoint for a force controller. It would be highly undesirable for the controller to receive discontinuous setpoints—particularly if these discontinuities occur strictly due to the optimization technique used, and not as a result of a discontinuity inherent in the physical system.

In a less general formulation of the dynamics equations such as that considered in §4.2.3, Alberts and Soloway (1988) have noted that the equality constraints remain constant for a given topology of the system. It might then be concluded that the question of continuity of the solution under continuous changes in the matrix $\mathbf{A}_1$ would be less critical in this situation. However, even in this case, linear programming yields discontinuous solutions due to its non-unique solution. The results obtained by Cheng (1989), which will be reviewed in the next chapter, offer good evidence of this.

It is therefore important to investigate other methods which will yield better-behaved solutions under the conditions of present interest.

5.2.3 Quadratic Programming

The quadratic objective function

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{5.9}$$

with positive-definite $\mathbf{W}$ is an example of a strictly convex function. Its gradient is $\nabla f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{c}$, while its Hessian matrix is the constant matrix $\mathbf{W}$.

An important feature of this objective function, whether constrained or not by linear functions, is that it is guaranteed to have a *unique* global minimum. For the unconstrained problem, it is trivial to show that the stationarity condition is satisfied at $\mathbf{x}^* = -\mathbf{W}^{-1}\mathbf{c}$. Under equality constraints, the location of that minimum can be found in

closed form by using the stationarity condition found in §5.1.2.1 as follows:

$$\nabla f(\mathbf{x}^*) = \mathbf{W}\mathbf{x}^* + \mathbf{c} = \mathbf{A}_1^T \boldsymbol{\lambda}^* \tag{5.10}$$

or, equivalently,

$$\mathbf{x}^* = \mathbf{W}^{-1}\mathbf{A}_1^T \boldsymbol{\lambda}^* - \mathbf{W}^{-1}\mathbf{c} \tag{5.11}$$

If we multiply both sides of eq.(5.11) by $\mathbf{A}_1$, we obtain

$$\mathbf{A}_1\mathbf{x}^* = \mathbf{A}_1\mathbf{W}^{-1}\mathbf{A}_1^T \boldsymbol{\lambda}^* - \mathbf{A}_1\mathbf{W}^{-1}\mathbf{c} \tag{5.12}$$

Since eq.(5.4b) must be satisfied, the left-hand-side of eq.(5.12) can be replaced by $\mathbf{b}_1$ to yield:

$$\mathbf{b}_1 = \mathbf{A}_1\mathbf{W}^{-1}\mathbf{A}_1^T \boldsymbol{\lambda}^* - \mathbf{A}_1\mathbf{W}^{-1}\mathbf{c} \tag{5.13}$$

which can, in turn, be rewritten as

$$\boldsymbol{\lambda}^* = (\mathbf{A}_1\mathbf{W}^{-1}\mathbf{A}_1^T)^{-1}(\mathbf{b}_1 + \mathbf{A}_1\mathbf{W}^{-1}\mathbf{c}) \tag{5.14}$$

Finally, substituting eq.(5.14) into eq.(5.11), we obtain:

$$\mathbf{x}^* = \mathbf{W}^{-1}\mathbf{A}_1^T(\mathbf{A}_1\mathbf{W}^{-1}\mathbf{A}_1^T)^{-1}(\mathbf{b}_1 + \mathbf{A}_1\mathbf{W}^{-1}\mathbf{c}) - \mathbf{W}^{-1}\mathbf{c} \tag{5.15}$$

which can also be written as:

$$\mathbf{x}^* = \mathbf{x}_1 - \mathbf{x}_2 \tag{5.16a}$$

$$\mathbf{x}_1 = \mathbf{W}^{-1}\mathbf{A}_1^T(\mathbf{A}_1\mathbf{W}^{-1}\mathbf{A}_1^T)^{-1}\mathbf{b}' \tag{5.16b}$$

$$\mathbf{b}' = \mathbf{b}_1 + \mathbf{A}_1\mathbf{x}_2 \tag{5.16c}$$

$$\mathbf{x}_2 = \mathbf{W}^{-1}\mathbf{c} \tag{5.16d}$$

The matrix operations shown in eqs.(5.15) to (5.16d) should not be performed explicitly because they are slow and lead to ill-conditioning of the problem. Section 5.4 will present quick and efficient techniques to solve this problem numerically.

The solution to the inequality-constrained quadratic optimization problem can be found in the same way using $\widehat{\mathbf{A}}_2$ instead of $\mathbf{A}_1$ and $\hat{\mathbf{b}}_2$ instead of $\mathbf{b}_1$. Of course, the

difficult part of this problem is to first *find* $\widehat{\mathbf{A}}_2$ and $\hat{\mathbf{b}}_2$—i.e., to find which of the inequality constraints are active at the solution, while taking into account that all Lagrange multipliers must be non-negative.

Finally, it is now shown that the second-order conditions specified in §5.1.1 and 5.1.2 are satisfied in the case of the quadratic objective function specified by eq.(5.9). Clearly, when the objective function is unconstrained, the Hessian matrix of $f(\mathbf{x})$ is positive-definite by definition and the second-order optimality condition given in §5.1.1 is satisfied. In the case of linearly-constrained optimization, the positive-definiteness of the projected Hessian is demonstrated as follows:

$$\mathbf{y}^T\mathbf{Z}^T\mathbf{H}(f(\mathbf{x}^*))\mathbf{Z}\mathbf{y} = \mathbf{y}^T\mathbf{Z}^T\mathbf{W}\mathbf{Z}\mathbf{y} \tag{5.17a}$$

$$= z^T\mathbf{W}\mathbf{z} \tag{5.17b}$$

where

$$\mathbf{z} = \mathbf{Z}\mathbf{y} \tag{5.17c}$$

Since the expression given by eq.(5.17b) must be positive for any vector $\mathbf{z}$, so must the expression given by eq.(5.17a) for any vector $\mathbf{y}$. Therefore, the projected Hessian, $\mathbf{Z}^T\mathbf{W}\mathbf{Z}$, is positive-definite.

From the above, we can conclude that *the strictly convex quadratic program has a unique global optimum.* To show the geometric significance of this, consider the strictly convex quadratic programming problem illustrated in Figure 5.4 and described as:

$$\min_{x_1,x_2} \quad x_1^2 + x_2^2 \tag{5.18a}$$

$$\text{subject to} \quad \alpha x_1 + x_2 = \alpha + 1 \tag{5.18b}$$

$$x_1 \geq 0, \qquad x_2 \geq 0 \tag{5.18c}$$

As was done for the linear-programming problem, the slope of the equality

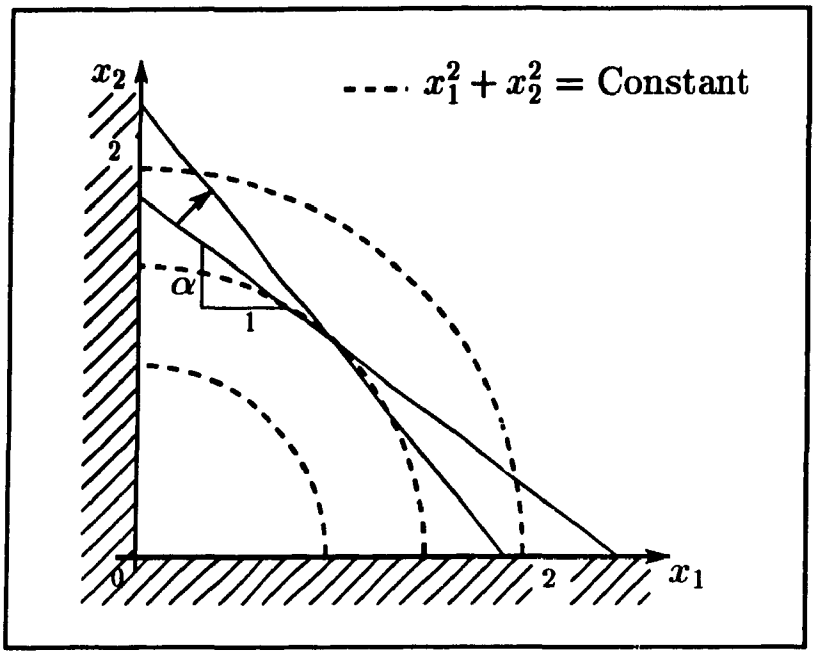

Figure 5.4 *Unique Minimum in Quadratic Programming*

constraint varied. Since the constraint can no longer become parallel to the objective function, the solution can no longer jump discontinuously when the constraints are varied smoothly. Figure 5.3 shows the variation in the solution to this problem as α is varied. The sensitivity analysis performed by Boot (1964) shows that the design variable solution is continuous with continuous changes in the constraint coefficients for a strictly convex quadratic programming problem with linear constraints.

Of course, when the topology of the kinematic chain changes, changes in matrix $\mathbf{A}_1$ are no longer continuous (c.f., §4.3). At those instants, a discontinuous change in the optimal solution will occur, even with quadratic programming, unless specific measures are taken to avoid it (c.f., §4.3.1, 4.4.5, 6.2.8).

5.3 Duality

In certain cases, the optimization problem, or *primal* problem, has a corresponding *dual* problem which is formulated in terms of the Lagrange multipliers of the

primal problem. The space formed by the Lagrange multipliers is dual to the space of the design variables. If the dual problem is solved, it can provide the solution to the primal through simple transformations. For certain forms of the primal problem, it may be more efficient to find the solution to the dual problem and perform these transformations to obtain the minimum value of the design variables. Thus, the dual problems to the linear and quadratic problems of present interest will be reviewed as they play a role in some of the numerical techniques to be studied in §5.4.

However, before presenting the dual problems, it is useful to show that an optimization problem with equality and inequality constraints can be transformed into one with only inequality constraints.

5.3.1 Removing the Equality Constraints

Any solution to the underdetermined system of linear equations given by eq.(5.4b) can be written as (Lawson and Hanson, 1974)

$$\mathbf{x} = \mathbf{x}^+ + \mathbf{N}\mathbf{y} \tag{5.19}$$

where the first and second terms on the right-hand side are the particular and a homogeneous solutions to the system given by eq.(5.4b). In most works, $\mathbf{x}^+$ is taken to be the minimum-norm solution of eq.(5.4b)—i.e.,

$$\mathbf{x}^+ = \mathbf{A}_1^+ \mathbf{b}_1 \tag{5.20a}$$

where

$$\mathbf{A}_1^+ = \mathbf{A}_1^T (\mathbf{A}_1 \mathbf{A}_1^T)^{-1} \tag{5.20b}$$

Matrix $\mathbf{A}_1^+$ is the right-generalized inverse, or pseudoinverse, of $\mathbf{A}_1$ (Rao and Mitra, 1971), while the columns of matrix $\mathbf{N}$ span the nullspace of $\mathbf{A}_1$ and $\mathbf{y}$ is an arbitrary vector whose dimension depends upon the number of columns in $\mathbf{N}$. As shown by Angeles *et al.* (1987), Householder reflections can be used to obtain a matrix $\mathbf{N}$ whose

columns form an orthogonal basis for the nullspace of $\mathbf{A}_1$. The elimination of the equality constraints proceeds by substituting eq.(5.19) into eqs.(5.4a) to (5.4c) to obtain

$$\min_{\mathbf{y}} \quad g(\mathbf{y}) \tag{5.21a}$$

$$\text{subject to} \quad \mathbf{A}_1(\mathbf{x}^+ + \mathbf{N}\mathbf{y}) = \mathbf{b}_1 \tag{5.21b}$$

$$\mathbf{A}_2(\mathbf{x}^+ + \mathbf{N}\mathbf{y}) \geq \mathbf{b}_2 \tag{5.21c}$$

where

$$g(\mathbf{y}) = \mathbf{c}^T(\mathbf{x}^+ + \mathbf{N}\mathbf{y}) \tag{5.21d}$$

if $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$, while

$$g(\mathbf{y}) = \mathbf{c}^T(\mathbf{x}^+ + \mathbf{N}\mathbf{y}) + \frac{1}{2}(\mathbf{x}^+ + \mathbf{N}\mathbf{y})^T\mathbf{W}(\mathbf{x}^+ + \mathbf{N}\mathbf{y}) \tag{5.21e}$$

if $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x}$.

Rearranging these equations and recalling that *a)* since $\mathbf{N}$ is an orthogonal complement of $\mathbf{A}_1$, $\mathbf{A}_1\mathbf{N} = \mathbf{0}$, *b)* $\mathbf{A}_1\mathbf{x}^+ = \mathbf{b}_1$, and *c)* a constant term in the objective does not affect the solution to the optimization problem and can therefore be dropped; the following equivalent minimization problem is obtained:

$$\min_{\mathbf{y}} \quad h(\mathbf{y}) \tag{5.22a}$$

$$\text{subject to} \quad \mathbf{A}_2\mathbf{N}\mathbf{y} \geq (\mathbf{b}_2 - \mathbf{A}_2\mathbf{x}^+) \tag{5.22b}$$

where

$$h(\mathbf{y}) = \mathbf{c}^T\mathbf{N}\mathbf{y} \tag{5.22c}$$

if $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$, and

$$h(\mathbf{y}) = (\mathbf{c}^T + \mathbf{x}^{+T}\mathbf{W})\mathbf{N}\mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{N}^T\mathbf{W}\mathbf{N}\mathbf{y} \tag{5.22d}$$

if $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x}$.

The reduced optimization problem given by eqs.(5.22a) to (5.22c) contains no equality constraints. An important feature of these equations is that they are of reduced

dimension when compared to eqs.(5.4a) to (5.4c) if $\mathbf{N}$ is composed of linearly independent columns. In this case, matrix $\mathbf{N}$ is of dimension $n \times r$, and vector $\mathbf{y}$ is of dimension r, where r is the nullity of $\mathbf{A}_1$ (and $r = n - m_{eq}$ if $\mathbf{A}_1$ is of full rank). As pointed out by Cheng and Orin (1989), this reduction in dimensionality helps to speed the numerical solution of the problem.

5.3.2 Linear Programming

The dual problem corresponding to the primal problem given by eqs.(5.22a) to (5.22c) is given by Cheng and Orin (1989) as

$$\max_{\boldsymbol{\lambda}} \quad (\mathbf{b}_2 - \mathbf{A}_2\mathbf{x}^+)^T\boldsymbol{\lambda} \tag{5.23a}$$

$$\text{subject to} \quad \mathbf{N}^T\mathbf{A}_2^T\boldsymbol{\lambda} = \mathbf{N}^T\mathbf{c} \tag{5.23b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{5.23c}$$

The primal problem had r unknowns and m_{in} inequality constraints, while the dual problem has m_{in} unknowns and r equality constraints, not including the non-negativity constraints on $\boldsymbol{\lambda}$. As will be discussed in more detail in §5.4.1, Cheng and Orin (1989) used this latter feature to obtain substantially faster solutions to the underdetermined force distribution problem than had previously been possible with linear programming.

5.3.3 Quadratic Programming

The dual problem corresponding to the primal given by eqs.(5.22a), (5.22b) and (5.22d) is given by Goldfarb and Idnani (1983) as

$$\max_{\boldsymbol{\lambda}} \quad (\tilde{\mathbf{b}}_2 + \widetilde{\mathbf{A}}_2\widetilde{\mathbf{W}}^{-1}\tilde{\mathbf{c}})^T\boldsymbol{\lambda} - \frac{1}{2}\boldsymbol{\lambda}^T\widetilde{\mathbf{A}}_2\widetilde{\mathbf{W}}^{-1}\widetilde{\mathbf{A}}_2^T\boldsymbol{\lambda} \tag{5.24a}$$

$$\text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0} \tag{5.24b}$$

where

$$\tilde{\mathbf{c}} = \mathbf{N}^T(\mathbf{c} + \mathbf{W}\mathbf{x}^+), \qquad \widetilde{\mathbf{W}} = \mathbf{N}^T\mathbf{W}\mathbf{N} \tag{5.24c}$$

$$\widetilde{\mathbf{A}}_2 = \mathbf{A}_2\mathbf{N}, \qquad \tilde{\mathbf{b}}_2 = \mathbf{b}_2 - \mathbf{A}_2\mathbf{x}^+ \tag{5.24d}$$

As in the linear-programming case, the primal problem had r unknowns and m_{in} inequality constraints. By contrast, however, the dual of the quadratic programming problem has m_{in} unknowns and no constraints other than the non-negativity constraints on the Lagrange multipliers. The method of Goldfarb and Idnani, based on the solution of this dual problem, provides solutions even more quickly than the linear-programming approach proposed by Cheng and Orin (1989).

5.4 Methods of Solution

Methods for solving the linear and quadratic programming problems outlined in §5.2 are numerous, but few of them are amenable to real-time implementation. For example, the IMSL (1987) library includes routines called DLPRS for linear programming and QPROG for quadratic programming. Although these were investigated, they were found to be excessively slow when compared to the in-house-written routines. Furthermore, the QPROG routine, being relatively new in IMSL's arsenal, is not fully debugged and failed to find solutions to certain problems where a solution existed, as verified by the in-house-written routine. The long execution times of the IMSL routines are at least partly due to extensive error checking and verification of the input data which would not be justifiable in a real-time application. A comparison of the execution speeds of these algorithms will be given in the numerical examples of Chapter 6. The following sections present a summary of the numerical techniques which were implemented in the present work.

5.4.1 Linear Programming

The general linear-programming problem of interest in the present work can be stated as follows:

$$\min_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x} \tag{5.25a}$$

$$\text{subject to} \quad \mathbf{A_1 x} = \mathbf{b_1} \tag{5.25b}$$

$$\mathbf{A_2 x} \geq \mathbf{b_2} \tag{5.25c}$$

where $\mathbf{x}$ is a vector of design variables of dimension n, $\mathbf{c}$ is a weighting vector of dimension n, $\mathbf{A_1}$ is an $m_{eq} \times n$ matrix of coefficients for the equality constraints, $\mathbf{b_1}$ is a vector of dimension m_{eq}, $\mathbf{A_2}$ is an $m_{in} \times n$ matrix of coefficients for the inequality constraints and $\mathbf{b_2}$ is a vector of dimension m_{in}. Finally, it is assumed that $m_{eq} < n$ and $m_{eq} + m_{in} = m$.

This problem is most commonly solved using the simplex method (Strang, 1976). A number of variations of this method exist but the one chosen for implementation was that found in Press *et al.* (1986). It consists of the following three steps:

1. Introducing 'slack variables' to convert the problem into *normal form* (a form which has no inequality constraints other than non-negativity constraints),

2. Finding an initial feasible solution, and

3. Stepping from one vertex to another of the feasible region to find the optimal feasible solution.

There are two relevant points which must be considered in this or any other implementation of the simplex method:

1. The 'standard' linear-programming problem includes non-negativity constraints on all the design variables. As proposed by Orin and Oh (1981), these can be circumvented by splitting all variables on which we do not want the constraint into two

parts, one positive and the other negative, i.e.,

$$x_i = x_i^+ - x_i^- \tag{5.26}$$

Thus, the problem is reformulated in terms of a new set of variables which is up to twice as large as the original set. This unfortunately leads to an increase in the computation time required to find a solution because of the increased dimensionality of the problem.

2. The number of steps taken in item (3) above determines how long the execution of the simplex method will take. A rule of thumb is that this number tends to be equal to $m_{eq} + m_{in}$ (Strang, 1976; Press *et al.*, 1986). Therefore, the larger the dimension of the dynamics equations we formulate and the more inequality constraints we include on the solution, the slower the execution will be.

Cheng and Orin (1989) recently proposed a 'compact-dual' linear programming (LP) formulation as a way of overcoming some of these obstacles and obtaining a real-time optimal solution to the force optimization problem in closed kinematic chains. The technique preconditions the problem by

1. Removing the equality constraints from the 'original' formulation to obtain the 'compact-primal' formulation. This technique, detailed in §5.3.1, is commonly used in optimization circles and was proposed by Kerr and Roth (1986) in the context of redundantly-actuated robotic systems. Rather than using the method outlined in §5.3.1 to find $\mathbf{x}^+$, the particular solution to eq.(5.25b) and $\mathbf{N}$, the matrix whose columns span the nullspace of $\mathbf{A}_1$, Cheng and Orin (1989) partitioned the design variables into 'free' and 'basic' variables and used Gaussian elimination to obtain $\mathbf{x}^+$ and $\mathbf{N}$. This technique is not recommended, as it treats the variables unequally and tends to introduce ill-conditioning into the problem. Rather, Householder reflections should be used to efficiently find the particular solution given by eqs.(5.20a) and eqs.(5.20b) and an orthogonal basis for the nullspace of $\mathbf{A}_1$, as shown by Angeles *et al.* (1987).

2. Formulating the dual problem which corresponds to the primal problem given by eqs.(5.22a) to (5.22c) to obtain the 'compact-dual' formulation as given by eqs.(5.23a) to (5.23c). The significant features of these latter equations is that *a)* the problem is already in normal form, thereby eliminating the need for item (1) of the simplex algorithm, *b)* the equations include non-negativity constraints even though the original problem did not, thereby circumventing the problem of having to double the dimension of the vector of design variables, and *c)* assuming the primal problem had more constraints than variables, the number of constraints is reduced.

Using these features of the 'compact-dual' problem to good advantage, Cheng and Orin (1989) showed that the linear-programming problem could be solved in real-time. However, this ingenious formulation does not overcome the problem of discontinuous solutions, as will be evident in the numerical examples shown in Chapter 6.

5.4.2 Quadratic Programming

The general linear-quadratic programming problem of present interest can be stated as:

$$\min_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{5.27a}$$

$$\text{subject to} \quad \mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \tag{5.27b}$$

$$\mathbf{A}_2\mathbf{x} \geq \mathbf{b}_2 \tag{5.27c}$$

where $\mathbf{W}$ is an $n \times n$ weighting matrix and the remaining vectors and matrices are as defined in §5.4.1. In order to solve this problem, a number of methods were investigated. Some of these methods are intended to solve the quadratic optimization problem in the absence of inequality constraints and are presented in §5.4.2.1 and compared in §5.4.2.2. The method suggested by Goldfarb and Idnani (1983) for quadratic optimization in the presence of inequality constraints only is then presented in §5.4.2.3. These methods are then combined in §5.4.2.4 to obtain a technique which will solve the quadratic optimization problem with linear equality *and* inequality constraints.

5.4.2.1 Solving the Equality-Constrained Problem

When the inequality constraints given by eq.(5.27c) are absent, the problem is considerably easier to solve. From the discussion of inequality constraints given in §4.4, it should be apparent that an important case in which inequality constraints are absent is that of cooperating manipulators where there are no constraints on the contact wrench and we are not interested in imposing limits on the actuator torques. For this simplified problem, solutions can be obtained in substantially less time than when inequality constraints are present. As well, certain techniques are available to efficiently solve this problem which are not practical when inequality constraints are present. In the present work, the following techniques were investigated to solve the equality-constrained linear-quadratic optimization problem:

1. Closed-form solution using *a)* explicit inversion, or *b)* orthogonal decomposition,
2. Explicit Lagrange multipliers,
3. Direct substitution.

The closed-form solution is conceptually the simplest of the techniques considered. It entails finding the solution to eqs.(5.27a) and (5.27b) using eq.(5.15). When $\mathbf{c} = \mathbf{0}$, this is nothing but the weighted pseudoinverse solution to eq.(5.27b). When $\mathbf{c} = \mathbf{0}$ and $\mathbf{W} = \mathbf{1}$, it corresponds to the unweighted pseudoinverse solution. Kumar and Waldron (1988) have shown that, in the absence of contact moments, the unweighted pseudoinverse solution corresponds to one which has zero 'interaction force' components, i.e., a solution which has no force components along the line joining any two contact points. Thus, the contact force field is a helicoidal vector field which is homologous to the velocity field for points in a single rigid body.

The $\mathbf{W}^{-1}$ term in eq.(5.15) often does not need to be calculated in real-time since it usually remains constant. This is not the case when the method of calculating the dynamics equations given in §4.2.3 is used and it is desired to minimize the actuator

wrenches. It will be shown in §6.2.4 that, when this is the case, the weighting matrix is modified by the time-varying Jacobian matrices of the individual robotic devices.

There exist many ways in which the solution given by eq.(5.15) can be found. One of these is to explicitly perform the matrix multiplications and inversions shown in that equation. This has two disadvantages: it is computationally slow and prone to ill-conditioning (Golub and Van Loan, 1983). The computational complexity and stability of the other inversion in eq.(5.15) can be greatly improved by using an orthogonal decomposition with Householder transformations (Lawson and Hanson, 1974). A further advantage of the orthogonal decomposition technique is that it can be extended to include inequality constraints using the method of Goldfarb and Idnani (1983). This will be treated in §5.4.2.3 and 5.4.2.4.

As shown by eq.(5.16a), the solution $\mathbf{x}^*$ is composed of two parts, namely, $\mathbf{x}_1$ and $\mathbf{x}_2$. It is noted that $\mathbf{x}_1$, as given by eq.(5.16b), is the solution to $\mathbf{A}_1\mathbf{x} = \mathbf{b}'$ which minimizes $\mathbf{x}^T\mathbf{W}\mathbf{x}$. The procedure starts by finding this solution:

1. Use Cholesky decomposition (Golub and Van Loan, 1983) to obtain a lower-triangular matrix $\mathbf{L}$ where $\mathbf{W} = \mathbf{L}\mathbf{L}^T$.

2. Use a sequence of n Householder reflections to bring $\mathbf{L}^{-1}\mathbf{A}_1^T$ into upper-triangular form, i.e.,

$$\mathbf{L}^{-1}\mathbf{A}_1^T = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}, \tag{5.28}$$

   where $\mathbf{R}$ is a $m_{eq} \times m_{eq}$ upper-triangular matrix, $\mathbf{0}$ is a zero matrix of dimension $(n - m_{eq}) \times m_{eq}$ and $\mathbf{Q}$ is the orthogonal $n \times n$ product of n Householder reflections.

3. Now let

$$\mathbf{y} = \begin{bmatrix}\mathbf{y}_a\\ \mathbf{y}_b\end{bmatrix} = \mathbf{Q}^T\mathbf{L}^T\mathbf{x} \tag{5.29}$$

   where $\mathbf{y}_a$ and $\mathbf{y}_b$ are m_{eq}- and $(n - m_{eq})$-dimensional vectors, respectively.

4. $\mathbf{A}_1\mathbf{x} = \mathbf{b}'$ can be rewritten as:

$$\mathbf{A}_1\mathbf{L}^{-T}\mathbf{Q}\mathbf{Q}^T\mathbf{L}^T\mathbf{x} = \mathbf{b}' \tag{5.30}$$

which, after substituting in eqs.(5.28) and (5.29), can be rewritten as

$$[\mathbf{R}^T \quad \mathbf{0}^T]\begin{bmatrix}\mathbf{y}_a\\ \mathbf{y}_b\end{bmatrix} = \mathbf{b}' \tag{5.31}$$

5. The minimum-norm solution of the system given by eq.(5.31) is:

$$\mathbf{y}_1 = \begin{bmatrix}\mathbf{y}_a\\ \mathbf{y}_b\end{bmatrix}_1 = \begin{bmatrix}\mathbf{R}^{-T}\mathbf{b}'\\ \mathbf{0}\end{bmatrix} \tag{5.32}$$

This minimizes $\mathbf{y}^T\mathbf{y} = \mathbf{x}^T\mathbf{L}\mathbf{Q}\mathbf{Q}^T\mathbf{L}^T\mathbf{x} = \mathbf{x}^T\mathbf{W}\mathbf{x}$.

6. Since the solution $\mathbf{y}_1$ is unique, eq.(5.29) can be used to obtain the corresponding $\mathbf{x}_1$ which minimizes $\mathbf{x}^T\mathbf{W}\mathbf{x}$ as

$$\mathbf{x}_1 = \mathbf{L}^{-T}\mathbf{Q}\mathbf{y}_1 \tag{5.33}$$

7. Now that $\mathbf{x}_1$ has been found, we complete the solution by finding $\mathbf{x}_2$ as

$$\mathbf{x}_2 = \mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{c} \tag{5.34}$$

and then finding $\mathbf{x}^*$ as given by eq.(5.16a).

A second method to find the solution of the equality-constrained quadratic optimization problem involves the explicit use of Lagrange multipliers (Wilde and Beightler, 1967). These are added as variables in the optimization problem to obtain a determinate system of equations as follows:

1. Write the Lagrangian of the equality-constrained optimization problem given by eqs.(5.27a) and (5.27b) as

$$L = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} - \boldsymbol{\lambda}^T(\mathbf{A}_1\mathbf{x} - \mathbf{b}_1) \tag{5.35}$$

2. Set the derivatives of the Lagrangian with respect to the design variables and with respect to the Lagrange multipliers to zero, i.e.,

$$\left.\frac{\partial L}{\partial \mathbf{x}}\right|_{\mathbf{x}^*} = \mathbf{0}, \qquad \left.\frac{\partial L}{\partial \boldsymbol{\lambda}}\right|_{\boldsymbol{\lambda}^*} = \mathbf{0} \tag{5.36}$$

thereby enforcing the stationarity of the Lagrangian at the solution.

3. Equation (5.36) can be rewritten as a determinate system of $n+m_{eq}$ linear equations in $n + m_{eq}$ unknowns. This system is then solved for $\mathbf{x}^*$ and $\boldsymbol{\lambda}^*$.

Whereas the closed-form solution technique deals with matrices of order m_{eq} or n, the use of explicit Lagrange multipliers transforms the problem into one of order $(m_{eq}+n)$. As will be shown in the next section, this leads to substantially longer execution times than for the closed-form solution. Therefore, although this method can be extended to handle inequality constraints through the use of slack variables (Wilde and Beightler, 1967), it was not pursued further.

The last technique is based on symbolic pre-processing of eqs.(5.27a) and (5.27b) in order to reduce the real-time computational load. In some sense, this approach is similar to that proposed by Klein *et al.* (1983), who solved eq.(5.15) symbolically, with $\mathbf{c} = \mathbf{0}$ and $\mathbf{W} = \mathbf{1}$, off-line and stored the solution as a compact algebraic equation to be solved in real-time. Although their approach was possible in the simplified case they investigated, it is not feasible for a more general system—i.e, with $\mathbf{c} \neq \mathbf{0}$, $\mathbf{W} \neq \mathbf{1}$ or coupled vertical and horizontal force/moment equations. More specifically, finding the inverse in eq.(5.15) symbolically is impossible except for the simplest of systems, even if a powerful symbolic manipulation package such as MACSYMA (1983) is used. However, direct substitution (Beveridge and Schechter, 1970) *can* be used to good advantage to reduce the amount of numerical computations required in real-time.

Direct substitution entails performing some of the optimization off-line symbolically and storing the results in the form of simple algebraic relations. This method is applicable to more complex systems than Klein *et al.*'s, but also has certain disadvantages. The following steps are performed *off-line* with MACSYMA (1983):

1a) Partition the system of equation (5.27b) into

$$[\mathbf{A}_a \quad \mathbf{A}_b]\begin{bmatrix}\mathbf{x}_a \\ \mathbf{x}_b\end{bmatrix} = \mathbf{b}_1 \tag{5.37}$$

such that $\mathbf{A}_a\mathbf{x}_a = \mathbf{b}_1 - \mathbf{A}_b\mathbf{x}_b$ is a square system (i.e., $\mathbf{A}_a$ is $m_{eq} \times m_{eq}$),

1b) Symbolically perform $\mathbf{x}_a = \mathbf{A}_a^{-1}(\mathbf{b}_1 - \mathbf{A}_b\mathbf{x}_b)$ thus yielding $\mathbf{x}_a(\mathbf{x}_b)$,

1c) Substitute $\mathbf{x}_a(\mathbf{x}_b)$ into the objective function, differentiate the objective function symbolically with respect to $\mathbf{x}_b$ and equate the result to $\mathbf{0}$. Since the objective function is quadratic, this will yield the coefficients of an $(n - m_{eq}) \times (n - m_{eq})$ system of linear equations in $\mathbf{x}_b$.

The above need only be performed once for each possible topology of the system and the following steps must be implemented numerically in real-time:

2a) Calculate the coefficients from the algebraic equations found in step (1c),

2b) Solve the $(n - m_{eq}) \times (n - m_{eq})$ linear system for $\mathbf{x}_b$,

2c) Use the relations found in step (1b) to obtain $\mathbf{x}_a$.

The principal disadvantage of this method is that by partitioning $\mathbf{x}$ into $\mathbf{x}_a$ and $\mathbf{x}_b$, it treats the various components unequally and can lead to ill-conditioning. Thus $\mathbf{A}_a$ can become rank-deficient even when $\mathbf{A}_1$ is not. Physical considerations must therefore be taken into account when choosing the partition of $\mathbf{x}$. Another precautionary measure is to store more than one solution, with each solution solving a different partition, and use the best-conditioned one. Finally, it is noted that the direct substitution method is unable to handle inequality constraints. In situations where these are unimportant, this method is recommended as long as it is applied with care.

5.4.2.2 An Equality-Constrained Example

The various techniques outlined in the preceding section were used to solve the underdetermined force distribution in the three-legged planar walking machine shown in Figure 2.9(a). The machine was assumed to be static and its equations of motion were formulated using the method proposed in §4.2.2. This resulted in a system of 9 equations in 12 unknowns—i.e., the system had three redundant actuators.

The inequality constraints which should normally be satisfied at the foot/ground

Method	CPU Tiine per Iteration with Diagonal Weighing (ms)[a]	CPU Time per Iteration with General Positive-Definite Weighing (ms)[a]
Closed Form Using Explicit Inversion	46.6	58.6
Closed Form Using Orthogonal Decomposition	15.3	29.2
Explicit Lagrange Multipliers	64.6	65.9
Direct Substitution Using MACSYMA	2.5	7.5

Table 5.1 *CPU Times for an Equality-Constrained Problem*

contact were ignored, thereby resulting in an equality-constrained problem. The problem was solved with $\mathbf{c} = \mathbf{0}$ and two cases for $\mathbf{W}$: one in which $\mathbf{W}$ was diagonal, and the other in which $\mathbf{W}$ was a more general positive-definite matrix.

Table 5.1, shows the execution times required for one solution of the under-determined force distribution problem. The explicit use of Lagrange multipliers is substantially slower than the other methods because it results in a system of order $n + m_{eq}$, and is therefore not recommended for real-time applications. The closed-form solution with explicit inversion deals with matrices of order n and m_{eq}, and is therefore faster than the use of Lagrange multipliers. Orthogonal decomposition allowed a two- to three-fold increase in computation speed over explicit inversion. Finally, direct substitution, which deals in real-time only with matrices of order $n - m_{eq}$, was between 4 and 26 times faster than the other techniques. For the present example the partition chosen causes $\mathbf{A}_a$ to become rank-deficient, even though $\mathbf{A}_1$ is not, when the ground contact points of the first

[a] Using double-precision on a Sun 3/60 Workstation with 68881 floating-point co-processor

and last legs are coincident—an unlikely occurrence. Thus, this technique is worthy of further investigation for systems where inequality constraints are unimportant.

5.4.2.3 Solving the Inequality-Constrained Problem

A number of methods were investigated in order to solve the problem with inequality constraints but the one that proved most promising was that proposed by Goldfarb and Idnani (1983). They and Powell (1983, 1985) have demonstrated the speed and numerical stability of this algorithm on some benchmark optimization problems, and compared its performance to that of existing techniques. This method can be seen as an extension of the orthogonal-decomposition algorithm introduced in the previous section. In fact, as will be seen in the next section, the latter algorithm can be used as the initialization step when both equality and inequality constraints are considered. The method also has the added advantage that no initial guess is required to start the search for a solution.

The method of Goldfarb and Idnani (1983) deals with the solution of the primal problem given by eq.(5.27a) subject to the inequality constraints of eq.(5.27c). In accordance with eqs.(5.24a) and (5.24b), the primal has a corresponding dual problem which is given by

$$\max_{\boldsymbol{\lambda}} \quad (\mathbf{b}_2 + \mathbf{A}_2\mathbf{W}^{-1}\mathbf{c})^T\boldsymbol{\lambda} - \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{A}_2\mathbf{W}^{-1}\mathbf{A}_2^T\boldsymbol{\lambda} \tag{5.38a}$$

$$\text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0} \tag{5.38b}$$

An algorithm which steps through the n-dimensional space of $\mathbf{x}$ is called a *primal method*, while one which steps through the q-dimensional space of $\boldsymbol{\lambda}$ is called a *dual method*. The method of Goldfarb and Idnani is a dual *active set* method. At each step, it minimizes $f(\mathbf{x})$ given by eq.(5.27a) subject to an *active set*, A, which is a subset of q *active* constraints in eq.(5.27c). The objective is to find the $\mathbf{x}$ which minimizes $f(\mathbf{x})$, while *satisfying* eq.(5.27c). The algorithm's stability is enhanced by dealing with

inherently stable decompositions of the matrices involved: the Cholesky decomposition of $\mathbf{W}$ ($\mathbf{W} = \mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ is lower-triangular), and the $\mathbf{QR}$ decomposition of $\mathbf{L}^{-1}\mathbf{A}^T$, i.e.,

$$\mathbf{L}^{-1}\mathbf{A}^T = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} \tag{5.39}$$

where $\mathbf{A}$ is comprised of the rows of $\mathbf{A}_2$ corresponding to the active constraints. Since matrix $\mathbf{Q}$ always appears in conjunction with $\mathbf{L}^{-T}$ in the algorithm, $\mathbf{J}^T$ is stored instead of $\mathbf{Q}$, where $\mathbf{J}^T = \mathbf{Q}^T\mathbf{L}^{-1}$. The speed of the algorithm is enhanced by making use of matrix updating techniques to modify the above $\mathbf{QR}$ decomposition as $\mathbf{A}$ changes when a constraint is added or dropped, rather than recalculating it *ab initio*. The method presented by Goldfarb and Idnani (1983) for quadratic optimization with inequality constraints is made up of the following steps (minor changes have been included to speed execution or rectify omissions in the original reference):

0a) Find $\mathbf{L}$ and $\mathbf{L}^{-1}$, where $\mathbf{W} = \mathbf{L}\mathbf{L}^T$ and $\mathbf{L}$ is lower-triangular—Cholesky decomposition of the weighting matrix.

0b) Set $\mathbf{x} = -\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{c}$ (the unconstrained minimum); $A = \emptyset$ (null active set); $\boldsymbol{\lambda} = [\,]$ (no Lagrange multipliers); $q = 0$ (no active inequalities); $\mathbf{J}^T = \mathbf{L}^{-1}$. Note that, since equality constraints are not considered, $m_{eq} = 0$, $\mathbf{A}_1 = [\,]$.

1) Evaluate $s_i = \mathbf{a}_i^T\mathbf{x} - b_i$, the residual for all *inactive* constraints, where $\mathbf{a}_i^T$ is the i-th row of $\mathbf{A}_2$ and b_i is the i-th component of $\mathbf{b}_2$. If all residuals are ≥ 0, the optimum has been found. Otherwise, choose the 'most violated constraint', p, to be added to the active set. Set $\mathbf{n}^+ = \mathbf{a}_p$ (the normal of the constraint to be added); $\boldsymbol{\lambda}^+ = \begin{bmatrix}\boldsymbol{\lambda}\\ 0\end{bmatrix}$.

2) If $q = 0$ (no active constraints), set $\boldsymbol{\lambda} = [\,]$ (no Lagrange multipliers).

3) Compute $\mathbf{d} = \mathbf{J}^T\mathbf{n}^+$; $\mathbf{z} = \mathbf{J}_2\mathbf{d}_2$ (the step direction in primal space); If $q > 0$, compute $\mathbf{r} = \mathbf{R}^{-1}\mathbf{d}_1$ (the negative of the step direction in dual space); Where $\mathbf{d} = \begin{bmatrix}\mathbf{d}_1\\ \mathbf{d}_2\end{bmatrix}$, $\mathbf{J}^T = \begin{bmatrix}\mathbf{J}_1^T\\ \mathbf{J}_2^T\end{bmatrix}$, $\mathbf{d}_1$ and $\mathbf{d}_2$ are q- and $(n-q)$-dimensional vectors respectively, while $\mathbf{J}_1^T$ and $\mathbf{J}_2^T$ are $q \times n$ and $(n-q) \times n$ matrices, respectively.

4) Set $\sigma_1 = L$ (σ_1 is the maximum step length in dual space without violating dual feasibility, and L is the largest number which can be represented by the computer).

5) Set

$$\sigma_1 = \min_{j=1,\ldots,q} \left\{ \frac{\lambda_j^+(\mathbf{x})}{r_j} \right\} = \frac{\lambda_l^+(\mathbf{x})}{r_l}$$
$$\text{subject to} \quad r_j > 0$$

6) If $||\mathbf{z}|| = 0$, set $\sigma_2 = L$; Otherwise, set $\sigma_2 = -s_p(\mathbf{x})/\mathbf{z}^T\mathbf{n}^+$ (σ_2 is the step length in primal space necessary to satisfy constraint p).

7) $\sigma = \min(\sigma_1, \sigma_2)$ (the step to be taken).

8) If $\sigma = L$, the problem is not feasible, STOP.

9) If $\sigma_2 = L$, take a step in dual space only: Set $\boldsymbol{\lambda}^+ = \boldsymbol{\lambda}^+ + \sigma \begin{bmatrix} -\mathbf{r} \\ 1 \end{bmatrix}$; Drop element l from $\boldsymbol{\lambda}^+$; Update $\mathbf{J}^T$ and $\mathbf{R}$ (see §5.4.2.5); Drop constraint l from A; $q = q - 1$; Evaluate the residual, s_i, for each inactive constraint; Go to step 2.

10) If $\sigma = \sigma_2$, take a full step in both primal and dual spaces to satisfy constraint p: Set $\mathbf{x} = \mathbf{x} + \sigma\mathbf{z}$; $\boldsymbol{\lambda}^+ = \boldsymbol{\lambda}^+ + \sigma \begin{bmatrix} -\mathbf{r} \\ 1 \end{bmatrix}$; $\boldsymbol{\lambda} = \boldsymbol{\lambda}^+$; Update $\mathbf{J}^T$ and $\mathbf{R}$ (see §5.4.2.5); Add constraint p to A; $q = q + 1$; Go to Step 1.

11) If $\sigma = \sigma_1$, take a partial step in both primal and dual spaces to a point on the boundary of the dual feasibility region: Set $\mathbf{x} = \mathbf{x} + \sigma\mathbf{z}$; $\boldsymbol{\lambda}^+ = \boldsymbol{\lambda}^+ + \sigma \begin{bmatrix} -\mathbf{r} \\ 1 \end{bmatrix}$; Drop element l from $\boldsymbol{\lambda}^+$; Update $\mathbf{J}^T$ and $\mathbf{R}$ (see §5.4.2.5); Drop constraint l from A; $q = q - 1$; Evaluate the residual, s_i, for each inactive constraint; Go to step 2.

Since Goldfarb and Idnani (1983) provide a detailed description of their technique, the remainder of this discussion will concentrate on highlighting the changes made in the present implementation, namely, its modification to include equality constraints.

5.4.2.4 Combining Equality and Inequality Constraints

The combined presence of equality and inequality constraints is now addressed by combining the techniques presented in the preceding sections. Equality constraints were not considered in the optimization technique presented by Goldfarb and Idnani (1983); therefore, their method had to be extended to include these efficiently. One possible solution would have been to write the equivalent minimization problem with inequality constraints only as shown in §5.3.1. Instead, it was found more efficient to alter the algorithm's initialization procedure to include the equality constraints.

Goldfab and Idnani (1983) initialize the algorithm at the unconstrained minimum because the equality constraints of eq.(5.27b) are not considered. In order to modify the method to include equality constraints, Powell (1983) proposed adding them one at a time. Although this approach works, it was found computationally intensive. The conceptual problem lies in treating the equality similarly to the inequality constraints. Since the equalities are known, by definition, to be active at the solution, they should be included from the outset, i.e., they should be included directly at initialization rather than iteratively in the body of the algorithm.

The approach taken here is to initialize the algorithm at the equality-constrained minimum—i.e., the solution of eqs.(5.27a) and (5.27b). It was previously shown that this minimum can be found from eqs.(5.16a) to (5.16d) using the technique outlined in §5.4.2.1. This technique proved to be compatible with the original algorithm for subsequent processing of the inequality constraints.

The search is then allowed to proceed in a space of reduced dimensions $n - m_{eq}$. Once the equalities are part of the active set, they should not be dropped. This constrains the search for a solution in the $(n - m_{eq})$-dimensional subspace rather than letting it proceed in the full n-dimensional space—an inherently more efficient procedure.

In order to implement the above technique, changes were made to Steps (0)

and (5) of the algorithm presented in §5.4.2.3. The modifications to Step (0) cause the equality constraints to be included directly at initialization, while the modification to Step (5) prevents removal of the equality constraints in the body of the algorithm.

0a) As before.

0b) Evaluate $\mathbf{x}_2 = \mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{c}$ and $\mathbf{b}' = \mathbf{b}_1 + \mathbf{A}_1\mathbf{x}_2$.

0c) Using Householder reflections, find $\mathbf{Q}$ and $\mathbf{R}$ to satisfy eq.(5.28).

0d) Find $\mathbf{y}_1$ using eq.(5.32).

0e) Find $\mathbf{x}_1$ using eq.(5.33).

0f) Set $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$ (the equality-constrained minimum); Evaluate $\mathbf{J}^T = \mathbf{Q}^T\mathbf{L}^{-1}$; Set $q = m_{eq}$; Evaluate $||\mathbf{a}_i||, i = m_{eq}+1, \ldots, m$ for use in evaluating the 'most violated constraint' in Step (1); The contents of A and $\boldsymbol{\lambda}$ need not be set for the equality constraints.

1) to 4) As before.

5) Set

$$\sigma_1 = \min_{j=m_{eq}+1,\ldots,q} \left\{ \frac{\lambda_j^+(\mathbf{x})}{r_j} \right\} = \frac{\lambda_l^+(\mathbf{x})}{r_l}$$
$$\text{subject to} \quad r_j > 0$$

6) to 11) As before.

The measure for choosing which of the violated constraints to add at Step (1) is not clearly specified by Goldfarb and Idnani (1983). They suggest choosing the 'most violated constraint' without specifying how to measure constraint violation—presumably using the value of the residual:

$$\min_{i=m_{eq}+1,\ldots,m} \quad \mathbf{a}_i^T\mathbf{x} - b_i \tag{5.40a}$$

Powell (1983) suggests using the value of the residual normalized by the norm of $\mathbf{a}_i$:

$$\min_{i=m_{eq}+1,\ldots,m} \quad \frac{\mathbf{a}_i^T\mathbf{x} - b_i}{||\mathbf{a}_i||} \tag{5.40b}$$

Both approaches were tried and Powell's was found to be more efficient and therefore adopted. Using this technique, it was found that inequality constraints were rarely added to the active set if they were not also in the final active set. Added constraints therefore rarely had to be dropped, and the time to find a solution was increased only proportionally to the number of inequalities in the final active set. This is advantageous in our application as it implies that the friction cone can be approximated by a pyramid with a large number of sides without affecting the speed of the solution, since only one of these constraints can be active at a given contact point. This is in contrast to linear programming, where the addition of further inequality constraints leads to slower execution.

The quadratic-programming algorithm described above was implemented as a Fortran subroutine whose listing is given in Appendix B. This routine was used to solve a number of numerical examples which will be shown in Chapters 6 and 7.

5.4.2.5 Updating $\mathbf{J}^T$ and $\mathbf{R}$

The technique used to update $\mathbf{J}^T$ and $\mathbf{R}$ outlined by Goldfarb and Idnani (1983) was not modified in principle. However, instead of Givens rotations, Householder reflections were used to perform all orthogonalizations, due to their superior computational efficiency (Golub and Van Loan, 1983).

When adding a constraint, the updated matrices, denoted here by $\mathbf{J}^{T+}$ and $\mathbf{R}^+$, are found as follows:

$$\mathbf{R}^+ = \begin{bmatrix} \mathbf{R} & \mathbf{d}_1 \\ \mathbf{0} & \pm||\mathbf{d}_2|| \end{bmatrix}, \qquad \mathbf{J}^{T+} = \begin{bmatrix} \mathbf{J}_1^T \\ \mathbf{Q}'\mathbf{J}_2^T \end{bmatrix} \tag{5.41}$$

where $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{J}_1^T$ and $\mathbf{J}_2^T$ are defined in Step (3) of §5.4.2.3, and $\mathbf{Q}'$ is the $(n-q)\times(n-q)$ product of $(n-q)$ Householder reflections, which satisfies

$$\mathbf{Q}'\mathbf{d}_2 = \pm||\mathbf{d}_2||\mathbf{e}_1 \tag{5.42}$$

where $\mathbf{e}_1$ is the unit vector $\mathbf{e}_1 = [1 \quad 0 \quad 0 \quad \ldots]^T$ of dimension $(n-q)$. In eqs.(5.41) and (5.42), the positive sign is chosen when the first component of $\mathbf{d}_2$ is negative, otherwise the negative sign is used.

When removing the l-th constraint, the updated matrices, denoted here by $\mathbf{J}^{T-}$ and $\mathbf{R}^-$, are found as follows. Remove the l-th column from $\mathbf{R}$ and partition it as:

$$\mathbf{R}_l = \begin{bmatrix} \mathbf{R}_1 & \mathbf{S} \\ \mathbf{0} & \mathbf{T} \end{bmatrix} \tag{5.43}$$

where $\mathbf{R}_1$ is an $(l-1)\times(l-1)$ upper-triangular matrix, $\mathbf{S}$ is an $(l-1)\times(q-l)$ matrix and $\mathbf{T}$ is a $(q-l+1)\times(q-l)$ upper-Hessenberg matrix (Strang, 1976). Find the $(q-l+1)\times(q-l+1)$ matrix product of Householder reflections $\widehat{\mathbf{Q}}$ such that

$$\widehat{\mathbf{Q}}\mathbf{T} = \begin{bmatrix} \mathbf{R}_2 \\ 0 \end{bmatrix} \tag{5.44}$$

where $\mathbf{R}_2$ is $(q-l)\times(q-l)$ upper-triangular. Finally, the updated matrices are

$$\mathbf{R}^- = \begin{bmatrix} \mathbf{R}_1 & \mathbf{S} \\ \mathbf{0} & \mathbf{R}_2 \end{bmatrix}, \qquad \mathbf{J}^{T-} = \begin{bmatrix} \mathbf{J}_{1a}^T \\ \widehat{\mathbf{Q}}\mathbf{J}_{1b}^T \\ \mathbf{J}_2^T \end{bmatrix} \tag{5.45}$$

where $\mathbf{J}_{1a}^T$ and $\mathbf{J}_{1b}^T$ are matrices of dimension $(l-1)\times n$ and $(q-l+1)\times n$, respectively, which make up a row partition of $\mathbf{J}_1^T$. Note that, when the constraint to be dropped is the last one added ($l=q$), $\mathbf{R}^-$ is obtained by simply dropping the last row and column of $\mathbf{R}$, and $\mathbf{J}^T$ is unchanged. As suggested by Golub and Van Loan (1983) for economy of operations, the Householder matrices $\mathbf{Q}$, $\mathbf{Q}'$ and $\widehat{\mathbf{Q}}$ are never explicitly calculated, but rather stored in a factored form and applied as needed.

5.4.2.6 Projected Gradient Methods

Klein and Kittivatcharapong (1990) recently suggested that gradient-projection algorithms could be used to solve the force optimization problem in the context of walk-

ing machines. They applied Rosen's (1960) gradient projection method to solve two formulations of the force optimization problem—an 'interior formulation' and an 'exterior formulation'—and obtained quasi-real-time solutions for a six-legged walking machine.

Although the gradient projection method can handle a general objective function, Klein and Kittivatcharapong (1990) chose to use a linear objective function, thereby resulting in solution discontinuity problems. In order to resolve this, the 'interior formulation' stops its search for a solution at a sub-optimal point in the space of design variables, thereby avoiding jumping from one vertex of the constraint polygon to another. By contrast, the 'exterior formulation' continues to a true optimum but includes a term in the solution which tends to smooth the solution. Given that the desired objective function is linear, both methods are sub-optimal. As will be shown by the numerical results in the next chapter, solution discontinuities could be avoided altogether by choosing a quadratic objective function.

One inherent disadvantage of gradient search techniques is that they require a good initial guess in order to converge quickly to a solution. The dependence of the method on a good initial guess is not problematic when the solution is continuous in time because the solution at the previous instant can be used as a guess for that at the present instant (a strategy used by Klein and Kittivatcharapong). However, as previously mentioned, redundantly-actuated robotic systems tend to suffer from discontinuous changes in their topology. As shown in §4.3, this leads to discontinuous changes in the constraint matrix $\mathbf{A}_1$ which, in turn, causes discontinuous changes in the solution to the optimization problem. Therefore, the solution at the previous time may not be a good initial guess for the solution when changes in topology occur. The example used by Klein and Kittivatcharapong (1990) does not consider this situation.

Chapter 6

Objective Functions

In the preceding chapter, the particular cases of linear and quadratic programming were investigated in considerable detail, since they allow a computationally more economical solution to the problem at hand than more general objective functions. This chapter will study some of the objective functions which have been suggested and which the present work proposes to use. Section 6.1 will review the linear objective functions put forward by Orin and Oh (1981), Kerr and Roth (1986) and Cheng and Orin (1989). The drawbacks of linear programming outlined in the previous chapter will become apparent when the numerical examples are presented. Section 6.2 will then review some quadratic objective functions which have been proposed—most notably those of Klein *et al.* (1983), Nakamura *et al.* (1987), Nakamura (1988a, 1988b), Kopf (1988a , 1988b) and Pfeiffer *et al.* (1990). Some new objective functions will also be presented and the solutions produced by quadratic optimization will be shown to be superior to those produced by linear programming.

6.1 Linear Objective Functions

When considering optimization with linear inequality constraints, linear programming comes naturally to mind. Thus, it is not surprising that most of the previous works found dealing with optimization of underdeterminate systems in the presence of

inequality constraints adopted a linear-programming approach (Orin and Oh, 1981; Kerr and Roth, 1986; Cheng and Orin, 1989; Cheng, 1989). In the preceding chapter, we found that the general linear-programming problem could be stated as

$$\min_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x} \tag{6.1a}$$

$$\text{subject to} \quad \mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \tag{6.1b}$$

$$\mathbf{A}_2\mathbf{x} \geq \mathbf{b}_2 \tag{6.1c}$$

where $\mathbf{x}$ is an n-dimensional vector of design variables and $\mathbf{c}$ is a vector of weights of dimension n.

It was also shown that there are certain disadvantages inherent to the use of linear objective functions. Among these, the question of whether this problem can be solved in real-time has been put to rest by the 'compact-dual' formulation of Cheng and Orin (1989). However, even this method has not resolved concerns regarding continuity of the solution. The work of Klein and Kittivatcharapong (1990) was primarily concerned with developing formulations with better continuity of the solution, but persisted in minimizing a linear objective function, and hence, had to settle for sub-optimal solutions to avoid discontinuities.

6.1.1 Orin and Oh's Objective Function

The earliest proposed solution to the inverse dynamics of redundantly actuated systems seems to be that found in the work of Orin at Ohio State University in the context of walking machines (McGhee and Orin, 1976; Orin and Oh, 1981). In those works, linear-programming was used as an off-line design tool to obtain an optimum solution. This formulation was quite general, as it included a consideration of inequality constraints on a number of the design variables. However, the inefficiency of the 'original' linear-programming problem, coupled with the slowness of computers of that time resulted in this technique being dismissed out-of-hand by most authors interested in real-time control.

The objective function proposed by Orin and Oh minimizes a weighted sum of the walking machine's energy consumption and the maximum vertical load on its legs. When there are p legs in ground contact, this objective function can be written as

$$f(\mathbf{x}) = \rho f_{N,max} + \sum_{i=1}^{p} (\mathbf{c}_i + \boldsymbol{\Theta}_i \mathbf{d}_i)^T \boldsymbol{\tau}_i \tag{6.2}$$

where ρ is the weight placed on minimizing the maximum normal force relative to that placed on reducing the energy consumption. Vectors $\mathbf{c}_i$ and $\mathbf{d}_i$ represent the characteristics of the drive motors and transmission system, as detailed by McGhee and Orin (1976), while $\boldsymbol{\Theta}_i$ is a diagonal matrix whose entries are the joint rates in the i-th leg. Finally, $f_{N,max}$ is the maximum normal force on any of the legs. This last variable is appended to the vector of design variables and the following constraints are added to the original system of inequality constraints:

$$f_{N_i} - f_{N,max} \leq 0, \qquad i = 1, \ldots, p \tag{6.3}$$

thereby ensuring that the normal forces on all the legs, which are constrained to be positive, will not exceed $f_{N,max}$.

This objective function represents one of the more detailed approaches which has been suggested to date. Its drawback, however, is that the second term in eq.(6.2)—the expression for the energy consumption of the system—is specific to the series-wound motors and non-backdrivable worm-gear drive system used in the OSU Hexapod. Although this objective function applies well to that type of system, many present-day robotic systems tend to be driven by dc servomotors through backdrivable reduction gearing. It will be shown in §6.2.7.2 that the power consumption of the latter type of system is better minimized using a quadratic objective function. Thus, this linear objective function was not implemented in the present work.

It should be noted, however, that the suggestion of introducing $f_{N,max}$ as an additional design variable to be minimized is significant, and can also be performed using a quadratic objective function.

6.1.2 Kerr and Roth's Objective Function

While considering the force distribution problem in the context of mechanical hands, Kerr and Roth (1986) proposed a more conservative linear programming approach which maximized the 'distance' from the solution to the inequality constraints. They first reduced the problem to one with only inequality constraints, as shown in §5.3.1, resulting in a minimization problem with $\mathbf{y}$ as the vector of design variable (c.f., eqs.(5.22a) to (5.22c)). The inequality constraints of this problem are

$$\mathbf{A_2 N y} - (\mathbf{b_2} - \mathbf{A_2 x^+}) \geq \mathbf{0} \tag{6.4}$$

If we define the *residual* of an inequality constraint to be

$$s_i = \hat{\mathbf{a}}_i^T \mathbf{y} - \hat{b}_i \tag{6.5}$$

where $\hat{\mathbf{a}}_i^T$ is the i-th row of $\mathbf{A_2 N}$ and $\hat{b}_i$ is the i-th component of $(\mathbf{b_2} - \mathbf{A_2 x^+})$, then the method of Kerr and Roth maximizes the minimum residual of the inequality constraints. This is done by appending a new element, d, to the vector of design variables and modifying the inequality constraints of eq.(6.4) as follows:

$$\mathbf{A_2 N y} - (\mathbf{b_2} - \mathbf{A_2 x^+}) \geq \mathbf{d} \tag{6.6}$$

where $\mathbf{d}$ is a vector of dimension $n - m_{eq}$ whose elements are all d.

As explained in greater detail by Kerr and Roth (1986), maximizing d will result in a solution which is 'as far away as possible' from all the constraints. The only item overlooked by this technique is that the constraints must be normalized before any meaningful 'distance' can be formulated. Thus Kerr and Roth's formulation suffers the drawback that the solution can be changed simply by multiplying one of the inequality constraints by a scaling factor. This problem is easily rectified by redefining the residual as

$$\overline{s_i} = \frac{\hat{\mathbf{a}}_i^T}{||\hat{\mathbf{a}}_i||}\mathbf{y} - \frac{\hat{b}_i}{||\hat{\mathbf{a}}_i||} \tag{6.7}$$

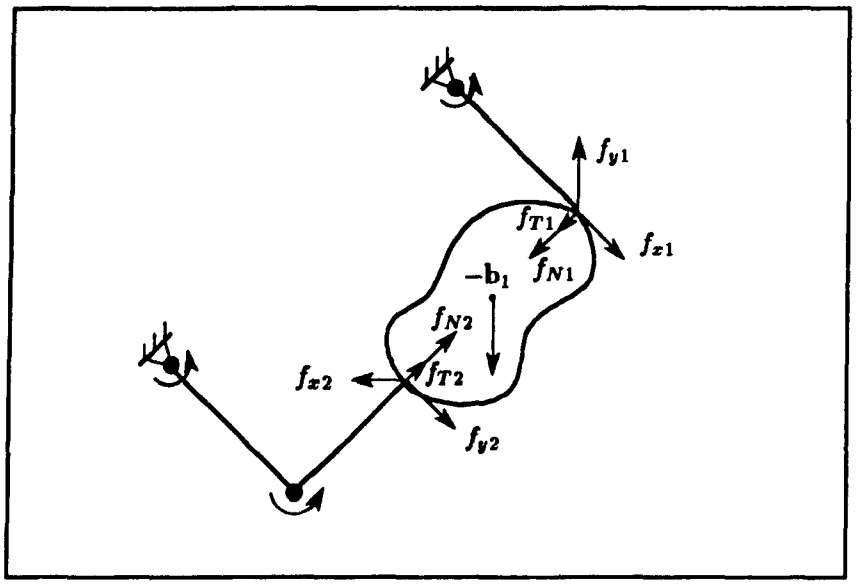

Figure 6.1 *Kerr and Roth's Example*

and rewriting eq.(6.6) as

$$(\overline{\mathbf{A}_2\mathbf{N}})\mathbf{y} - (\overline{\mathbf{b}_2 - \mathbf{A}_2\mathbf{x}^+}) \geq \overline{\mathbf{d}} \tag{6.8}$$

where the i-th row of $(\overline{\mathbf{A}_2\mathbf{N}})$ is given by $\hat{\mathbf{a}}_i^T/||\hat{\mathbf{a}}_i||$, the i-th element of $(\overline{\mathbf{b}_2 - \mathbf{A}_2\mathbf{x}^+})$ is $\hat{b}_i/||\hat{\mathbf{a}}_i||$ and $\overline{\mathbf{d}}$ is a vector of dimension $n - m_{eq}$ whose elements are all $\overline{d}$.

The numerical example chosen to illustrate the optimization technique presented above is the one used by Kerr and Roth (1986), as shown in Figure 6.1. The grasped body is free to move in three dimensions, but the fingers are constrained to move in the plane of the paper. The fingers are considered to make 'soft finger contact' with the body, so that three forces and one torque about the surface normal can be generated, and the vector of contact wrenches can be written as

$$\mathbf{x} = [f_{x1} \quad f_{y1} \quad f_{N1} \quad f_{T1} \quad f_{x2} \quad f_{y2} \quad f_{N2} \quad f_{T2}]^T \tag{6.9}$$

where f_{T1} and f_{T2} are the torques exerted by the fingertips on the object. The force/moment

balance equations are written in the form of eq.(6.1b) with

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} 0 \\ -1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.10}$$

The inequality constraints originate from limits on tangential and torsional friction forces which can be generated at the contacts, as well as limits on the actuator torques which can be produced. These are written in the form of eq.(6.1c) with

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & \mu & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & \mu & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu_t & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & \mu & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \mu & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu_t & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mu_t & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mu_t & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\tau_m \\ -\tau_m \\ -\tau_m \\ \tau_m \\ \tau_m \\ \tau_m \end{bmatrix} \tag{6.11a}$$

where $\mu = 0.25$, $\mu_t = 0.5$ and $\tau_m = 5.0$ N-m.

Since $\mathbf{A}_1$ is of dimension 6×8 and of full rank, its nullity is $(n - m_{eq}) = 2$, and vector $\mathbf{y}$ is of dimension 2. Kerr and Roth's linear formulation, which maximizes the minimum component in the residual vector $(\mathbf{A}_2\mathbf{N}\mathbf{y} - [\mathbf{b}_2 - \mathbf{A}_2\mathbf{x}^+])$, was implemented. The solution to this linear-programming problem happens to touch two vertices of the feasible set and the solution is therefore not unique. Any convex combination of the following two

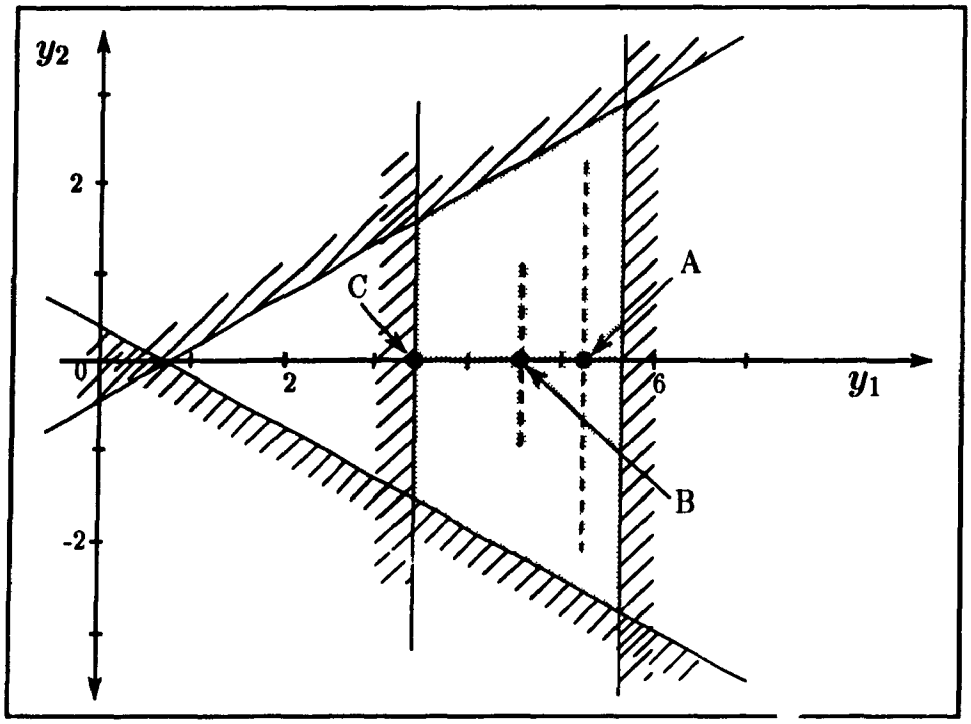

Figure 6.2 *The Solutions to Kerr and Roth's Example in* **y***-space*

vectors

$$\mathbf{x}^* = [-0.5 \quad 0 \quad 3.2 \quad 1.3 \quad 0 \quad -0.5 \quad 4.2 \quad 1.3]^T \tag{6.12a}$$

$$\mathbf{x}^* = [-0.5 \quad 0 \quad 3.2 \quad -1.3 \quad 0 \quad -0.5 \quad 4.2 \quad -1.3]^T \tag{6.12b}$$

which optimize the objective function, yield the same optimum value of the objective function.

These solutions are represented in **y**-space by the dotted line passing through point A in Figure 6.2. The relevant inequality constraints which bound the feasible **y**-space are shown, where the infeasible side of each constraint is cross-hatched. The feasible region is shown by the shaded area. As should be apparent from this figure, the solution found does not truly maximize the minimum 'distance' to the constraints since it is closer than it should be to the inequality constraint on the right side of the figure. If the constraints are normalized as suggested above, any point on the dotted line passing through point B optimizes the objective function. This solution *does* maximize the minimum distance to the constraints, but, unfortunately, is neither unique.

Algorithm	Kerr & Roth LP	Quadratic Programming
IMSL Routine CPU Time (ms)[a]	586.7	55.0
In-House Routine CPU Time (ms)[a]	38.0	19.5
Number of Unknowns	5	8
Number of Equality Constraints	0	6
Number of Inequality Constraints	20	20

Table 6.1 *CPU Times for Kerr and Roth's Example*

Quadratic programming was then used to minimize an objective function of the form $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{x}$. As will be discussed later in this chapter, this objective function yields the minimum contribution of the nullspace of $\mathbf{A}_1$, and hence, minimizes the 'internal forces'. Since vector $\mathbf{x}$ contains both forces and torques, it is important to notice that this optimization will weigh 1 N-m of torque and 1 N of force equally. The unique solution to this problem is then found to be

$$\mathbf{x}^\star = [-0.5 \quad 0 \quad 2.0 \quad 0 \quad 0 \quad -0.5 \quad 3.0 \quad 0]^T \tag{6.13}$$

Since Kerr and Roth's (1986) formulation stays 'as far away' as possible from the boundaries of the inequality constraints, it yields a more conservative solution with larger 'internal force'. The CPU times required to obtain a solution for linear and quadratic programming using both IMSL routines and the routines developed in the present work are shown in Table 6.1.

Two conclusions can be drawn from this table:

1. Quadratic programming is appreciably faster than linear programming for this example[1];

2. The routines developed in the present work are substantially more efficient than

[a] Using double-precision on a Sun 3/80 Workstation with 68881 floating-point co-processor

[1] *n.b.*, the formulation used by Kerr and Roth corresponds to the 'compact-primal' formulation of Cheng and Orin (1989)

those provided by IMSL.

Finally, it is emphasized that quadratic programming delivers a *unique* solution, whereas linear-programming does not.

6.1.3 Cheng and Orin's Objective Functions

The linear-programming formulation developed by Cheng and Orin (1989) was described in §5.4.1. In their terminology, the original linear-programming formulation is referred to as the 'original LP', the problem obtained after the equality constraints are removed is called the 'compact-primal LP' and finally, the problem obtained after reformulation in the dual space is called the 'compact-dual LP'.

Cheng and Orin (1989) and Cheng (1989) proposed a number of objective functions which could be minimized in a task involving a hand grasping an object:

1. 'Minimum effort'—minimize the sum of the normal forces applied to the grasped object;

2. 'Load balance'—minimize the maximum normal force exerted on the object (i.e., as discussed in §6.1.1); possibly combined with item 1;

3. 'Safety margin on friction constraints'—minimize the distance to the sides of the friction pyramid using the same technique as proposed by Kerr and Roth (1986). Note that this method suffers from the same drawback of non-invariance noted in §6.1.2 and should be modified accordingly; and

4. 'Temporal continuity'—in an effort to circumvent the discontinuities obtained by Cheng (1989), the sign of the lateral contact forces is constrained to remain the same throughout the task while, at the same time, minimizing the 'load balance' objective function.

There are several problems inherent in the approach proposed in item 4: *a)* as

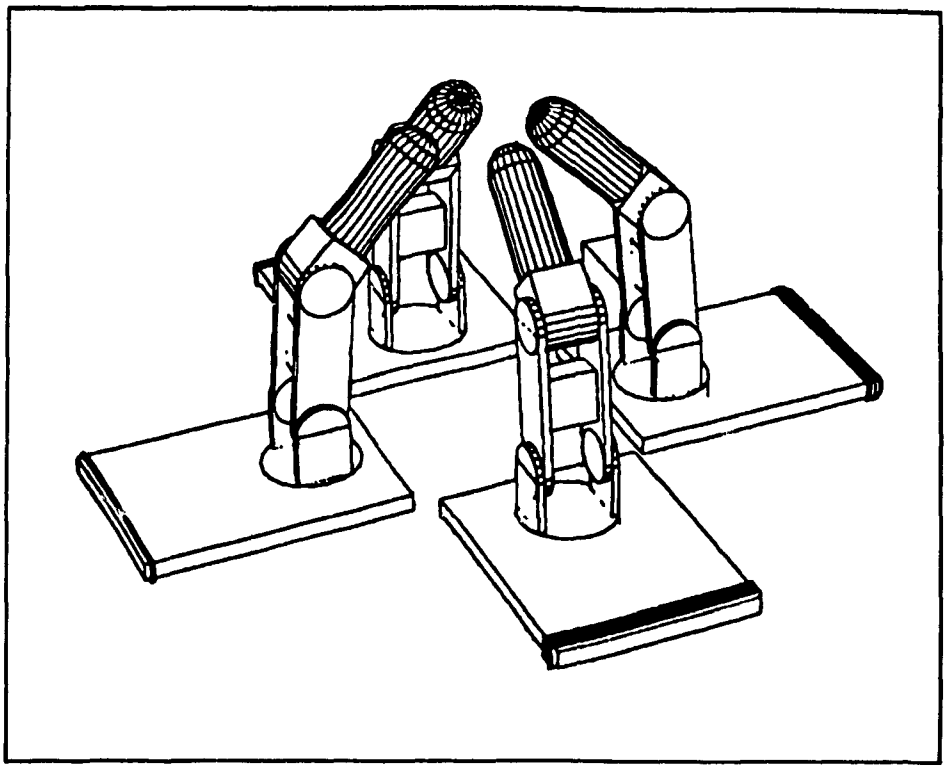

Figure 6.3 *The Ohio State DIGITS System*

acknowledged by Cheng (1989), it is arbitrary and may not work in a more general task, *b)* constraining the sign of the lateral force only works when optimizing the 'load balance' objective function, i.e., it does not work when minimizing the sum of the contact forces, thereby limiting the generality of the objective function which can be used with this formulation, and *c)* even in the case where this approach does work, closer examination of the results obtained by Cheng (1989) shows that the fingers are fighting against each other in their efforts to rotate the ball. In fact, the technique proposed in §4.4.5 might well be a better remedy to the problem of discontinuous solutions obtained with linear programming. This was not investigated in the present work since quadratic programming was found to be superior to linear programming in all respects. Thus, it was felt preferable to focus on a technique which is inherently continuous rather than try to find 'patches' for one which is not.

Cheng and Orin (1989) and Cheng (1989) evaluated their proposed techniques on a simulation of the Ohio State DIGITS system shown in Figure 6.3 using two- and four-

fingered grasps. The two-fingered grasp is shown diagrammatically in Figure 6.4. Hard finger point contact is assumed so that there are three unknown forces at each contact point. The prescribed task is one where a ball of mass 0.91 kg and 51 mm diameter is rotated sinusoidally about its vertical axis with an amplitude of 30° and a frequency of 2 Hz. Further details of the apparatus and the task can be found in Cheng and Orin (1989) and Cheng (1989). The linear objective function to be minimized is the sum of the normal contact forces on the object (called 'minimum effort' by Cheng and Orin). Friction constraints and maximum actuator torque constraints, neglecting the inertial torques τ', are imposed. Since the object cannot be rotated about an axis passing through the two contact points, the problem can be formulated with five scalar equations of motion and six unknown contact forces, yielding an underdeterminacy of one. Throughout the task, one of the friction constraints is active (i.e., satisfied as an equality), thereby consuming the redundancy in the problem, and making the solution unique. The resulting contact forces at the tip of Finger #1 are shown in Figure 6.5(a), while the corresponding actuator torques are shown in Figure 6.5(b). The contact forces are shown as components in a 'contact coordinate frame' defined in Cheng (1989). In this frame, the z-component is the negative of the normal contact force defined in the present work, the x-component is upward, and the y-component completes a right-handed coordinate frame.

Quadratic programming was also implemented to minimize the 'internal force' in this system, as was done for the Kerr and Roth example. Again, the same friction constraint was active throughout the task, thereby yielding the *same* solution as linear programming, shown in Figures 6.5(a) and 6.5(b).

The execution times taken to obtain the optimal solution using the various formulations and algorithms are shown in Table 6.2. These figures confirm the results of Cheng and Orin (1989) and Cheng (1989), who cite speedups of over 30 times when solving the 'compact-dual LP' formulation as compared to the 'original LP' formulation, using IMSL routines. As shown in that table, quadratic programming was marginally slower than linear programming due to the low dimensionality of this example. It was found that

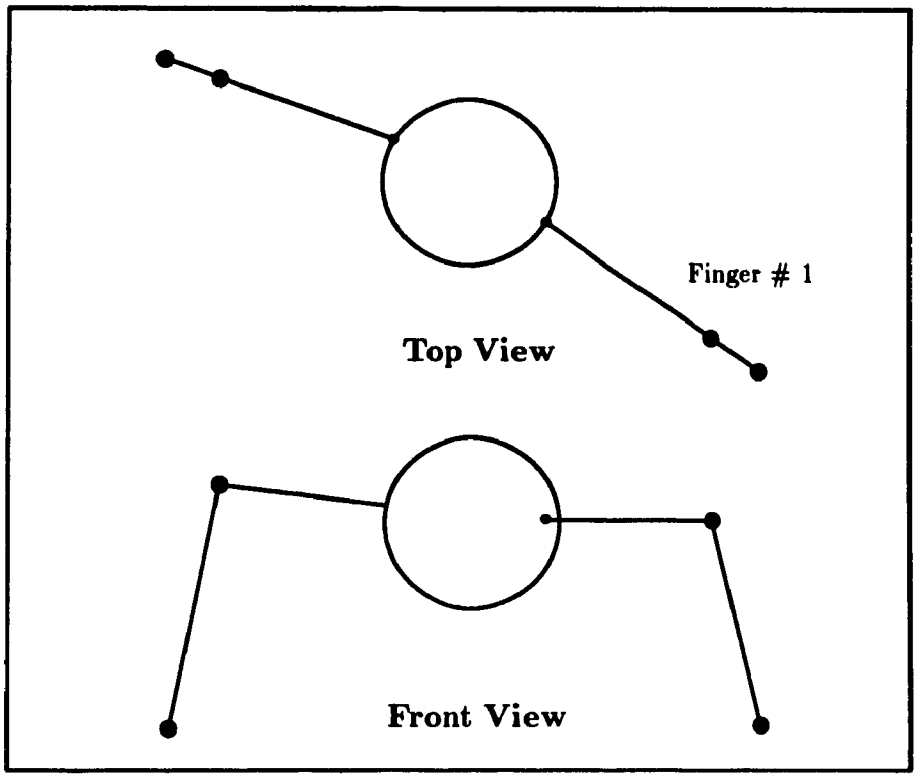

Figure 6.4 *Diagrammatic Representation of Two-Fingered Grasp*

the higher the dimensionality of the problem, $n - m_{eq}$, the greater the speed advantage of the quadratic-programming algorithm over the linear-programming algorithm.

The substantial differences between linear and quadratic programming become apparent in the four-fingered grasp example. A diagram of this arrangement is shown in Figure 6.6. The task is the same as previously described for the two-fingered grasp. This example has six equations of motion and 12 unknown contact forces, resulting in an underdeterminacy of six. For a linear objective function defined as the sum of the normal contact forces (called 'minimum effort' by Cheng and Orin), the results are shown in Figures 6.5(c) and 6.5(d). The solution exhibits severe discontinuities, due to its non-uniqueness. In fact, all the various combinations of linear programming formulation and algorithm gave different, discontinuous results for this example. It is therefore not apparent that linear programming can yield acceptable results for problems with a higher dimensionality.

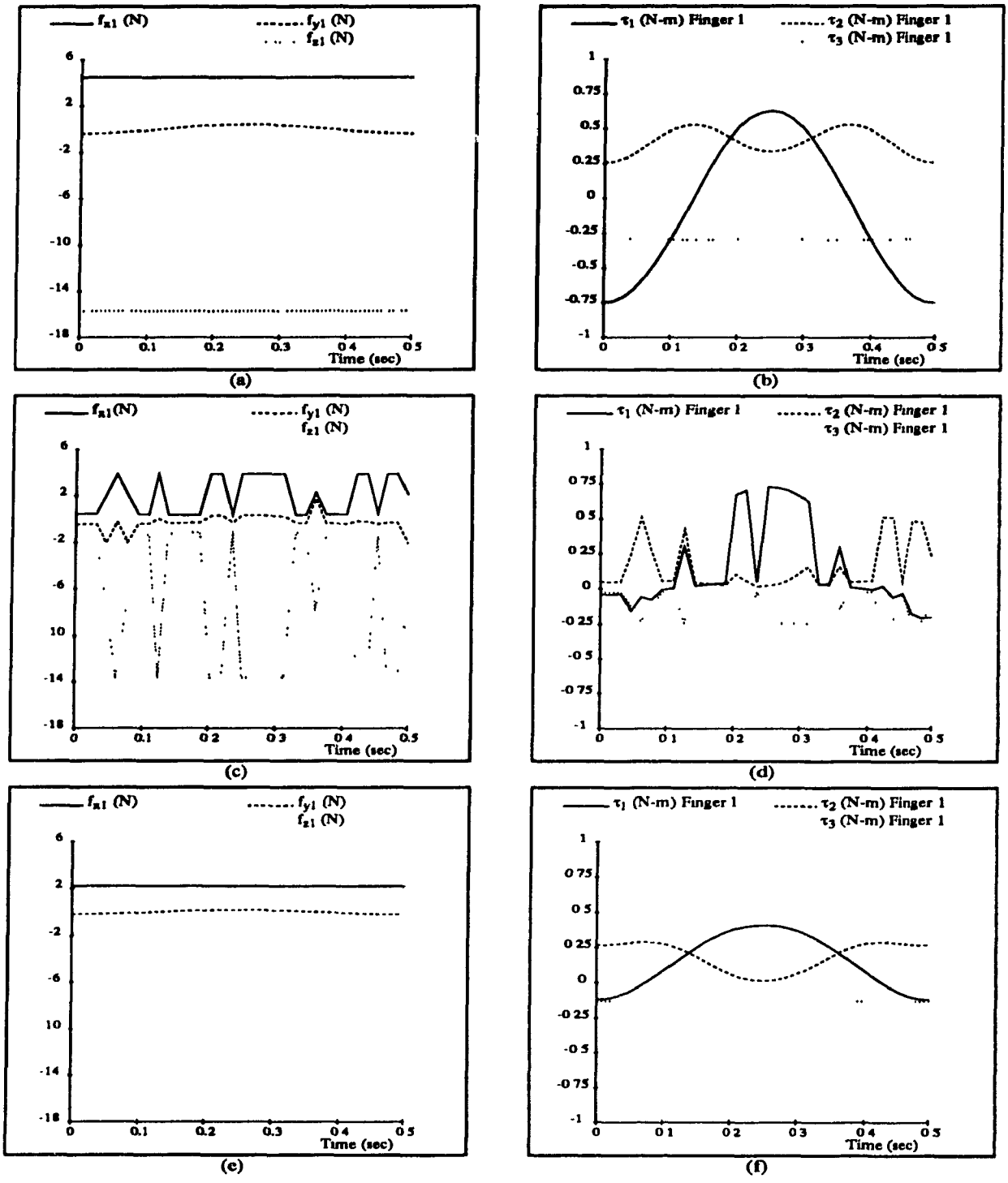

Figure 6.5 *Contact Forces and Joint Torques for Finger #1*

Algorithm	Original LP	Compact Primal LP	Compact Dual LP	Quadratic Programming
IMSL Routine CPU Time (ms)[a]	1232.1	443.1	32.4	58.3
In-House Routine CPU Time (ms)[a]	102.5	15.7	11.3	13.1
Number of Unknowns	12	2	20	6
Number of Equality Constraints	5	0	1	5
Number of Inequality Constraints	20	20	0	20

Table 6.2 *CPU Times for Cheng & Orin's Two-Finger Example*

Algorithm	Original LP	Compact Primal LP	Compact Dual LP	Quadratic Programming
IMSL Routine CPU Time (ms)[a]	7200	2932	361	177
In-House Routine CPU Time (ms)[a]	600	193	128	88
Number of Unknowns	24	12	40	12
Number of Equality Constraints	6	0	6	6
Number of Inequality Constraints	40	40	0	40

Table 6.3 *CPU Times for Cheng's Four-Finger Example*

Once again, quadratic programming was used to minimize the 'internal force' in this system. Three inequality constraints were active throughout the task. The smooth results shown in Figures 6.5(e) and 6.5(f) were obtained for the contact forces and joint torques at Finger #1. The timing results for the 4-finger grasp example are shown in Table 6.3. For this problem of higher dimensionality, quadratic programming is substantially faster than linear programming. It should be noted, as well, that the speed of the Sun 3/80 used for these simulations can already be greatly and cheaply surpassed by newer RISC-based workstations. This example makes the advantages of quadratic programming over linear programming most apparent.

[a]Using double-precision on a Sun 3/80 Workstation with 68881 floating-point co-processor

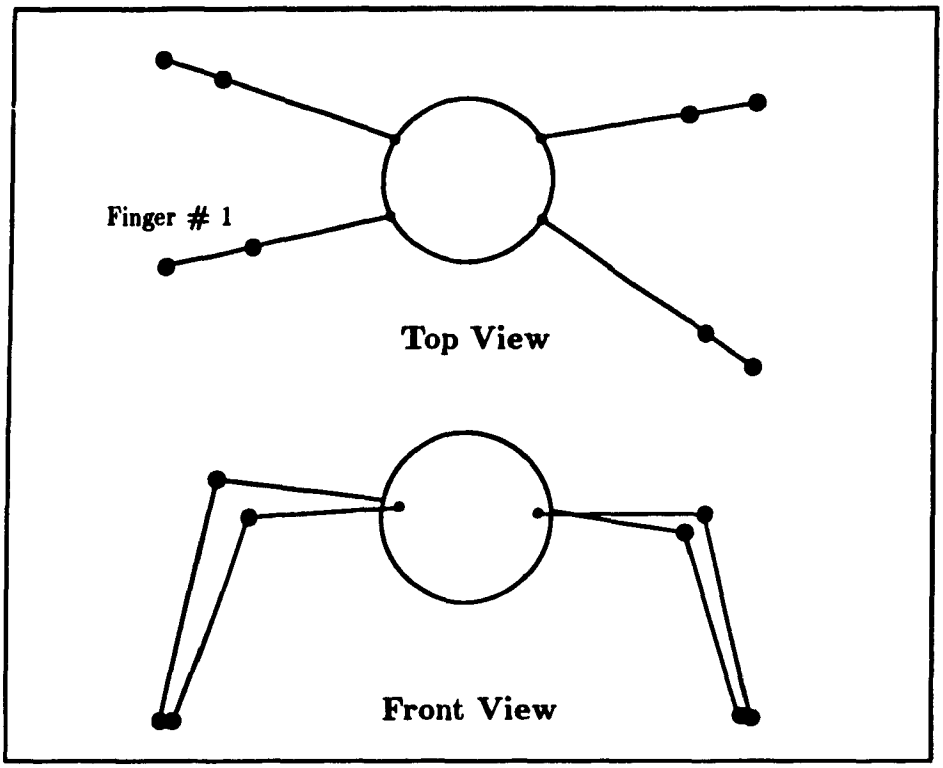

Figure 6.6 *Diagrammatic Representation of Four-Fingered Grasp*

6.2 Quadratic Objective Functions

In the preceding chapter, it was found that the general convex linear-quadratic objective function can be stated as:

$$\min_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{6.14a}$$

$$\text{subject to} \quad \mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \tag{6.14b}$$

$$\mathbf{A}_2\mathbf{x} \geq \mathbf{b}_2 \tag{6.14c}$$

where $\mathbf{x}$ is an n-dimensional vector of design variables, $\mathbf{W}$ is an $n \times n$ positive-definite weighting matrix, and $\mathbf{c}$ is a weighting vector of dimension n. The objective function given by eq.(6.14a) allows the optimization of the wrenches in $\mathbf{x}$ or any other quantities which can be reduced to a linear-quadratic function of $\mathbf{x}$. As was shown in Chapter 5, the condition that $\mathbf{W}$ be positive-definite ensures that the solution to the problem will be unique and continuous in time, in the absence of changes in the topology of the system.

Quadratic objective functions have been frequently proposed to solve the underdeterminate force distribution problem, but almost always in the form of the minimum-norm solution to eq.(6.14b), i.e., neglecting the inequality constraints. Klein *et al.* (1983) were amongst the earliest proponents of this solution. They were able to find a *symbolic* expression for the pseudoinverse solution to generate vertical force setpoints for the controller of the OSU Hexapod. As discussed in §5.4.2.1, this symbolic solution is computationally very efficient but has severe limitations, namely, *a)* it can only be performed for extremely simple systems, e.g., when the vertical and lateral force systems are decoupled, and *b)* it cannot handle inequality constraints. The first objection can be overcome by performing a numerical pseudoinverse solution, as outlined in §5.4.2.1—whether weighted or unweighted and whether using explicit inversion or, preferably, Householder reflections (Golub and Van Loan, 1983). This method appears to be the one most commonly proposed by other researchers, e.g., Hayati (1986), Zheng and Luh (1986), Alberts and Soloway (1988), Park and Starr (1989) and Kumar and Waldron (1989). The second objection is more difficult to overcome, particularly if real-time solutions are required. In fact, the method outlined in §5.4.2.4 can be viewed as an extension of the pseudoinverse solution to include inequality constraints, and it is a natural method to use for systems where inequality constraints are important.

A number of other techniques have been suggested to minimize a quadratic objective function with linear equality and inequality constraints, though numerical results are almost never shown and the feasibility of implementing these techniques in real-time is never discussed. Waldron (1986) proposed a technique, applicable to systems with zero contact torques, using 'interaction forces', which reduces to the pseudoinverse solution when $\mathbf{A}_1$ is of full rank (Kumar and Waldron, 1988). The importance of inequalities was noted, and a solution was proposed whereby the problem would be solved successively up to 15 times, for the case when six legs are in ground contact, and the best solution selected from among these. No timing results were given to indicate that these computations would be manageable in real time. Nakamura *et al.* (1987) proposed

an inequality-constrained nonlinear-programming approach to minimize internal forces in multiple robotic mechanisms subject to quadratic inequality constraints. They presented two algorithms to solve this problem, but again, with no indication that they would be feasible for real-time control. Finally, Luh and Zheng (1988) suggested the method of 'direct approximate programming' to find the force distribution in multiple manipulators handling a common payload but did not provide any numerical results.

6.2.1 Minimum Internal Force

The minimization of 'internal force' in redundantly-actuated systems has been proposed by numerous authors, the earliest and most consistent proponent being Nakamura (Nakamura *et al.*, 1987). This approach minimizes $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{x}$, i.e., the norm of the vector of contact wrenches, assuming that the dynamics equations are written in the form outlined in §4.2.3. If the dynamics equations are formulated using the methods of §4.2.1 or 4.2.2, the solution obtained when $\frac{1}{2}\mathbf{x}^T\mathbf{x}$ is minimized will be different, since in each case, the vector $\mathbf{x}$ contains different elements. This serves to highlight the fact that there may be more than one interpretation to the 'minimum internal force solution'.

The geometrical interpretation of this technique is that it minimizes the solution contribution lying in the nullspace of $\mathbf{A}_1$, i.e., that which satisfies $\mathbf{A}_1\mathbf{x} = \mathbf{0}$. Since $\mathbf{b}_1$ represents the desired motion of the system, as well as the external applied loads—see e.g., eq.(4.8b)—and the component in the nullspace of $\mathbf{A}_1$ contributes nothing to it, the internal force is often interpreted as the component of the solution which tends to *crush* or *tear* apart a grasped object or *grip* the ground.

A solution with *zero* internal force is obtained by minimizing eq.(6.14a) with $\mathbf{c} = \mathbf{0}$ and $\mathbf{W} = \mathbf{1}$ subject to the equality constraints of eq.(6.14b). This is nothing but the pseudoinverse solution commonly used to solve the underdetermined force problem. It will now be shown that the addition of the inequality constraints, eq.(6.14c), to the problem is equivalent to minimizing the solution component in the nullspace of $\mathbf{A}_1$. Equation (6.14a)

becomes:

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{x} \tag{6.15}$$

Any solution to eqs.(6.14a) to (6.14c) can be written as

$$\mathbf{x} = \mathbf{x}^+ + \mathbf{x}^- \tag{6.16a}$$

$$\mathbf{x}^+ = \mathbf{A}_1^+\mathbf{b}_1 \tag{6.16b}$$

$$\mathbf{A}_1^+ = \mathbf{A}_1^T(\mathbf{A}_1\mathbf{A}_1^T)^{-1} \tag{6.16c}$$

where $\mathbf{x}^+$ is the minimum-norm solution of eq.(6.14b), $\mathbf{A}_1^+$ is the right generalized inverse of $\mathbf{A}_1$ (Rao and Mitra, 1971), and $\mathbf{x}^-$ is the component of the solution lying in the nullspace of $\mathbf{A}_1$. Substituting eqs.(6.15) and (6.16a) to (6.16c) into eqs.(6.14a) to (6.14c) yields the following equivalent minimization problem:

$$\min_{\mathbf{x}^-} \quad \frac{1}{2}(\mathbf{x}^+ + \mathbf{x}^-)^T(\mathbf{x}^+ + \mathbf{x}^-) \tag{6.17a}$$

$$\text{subject to} \quad \mathbf{A}_1\mathbf{x}^- = \mathbf{0} \tag{6.17b}$$

$$\mathbf{A}_2\mathbf{x}^- \geq (\mathbf{b}_2 - \mathbf{A}_2\mathbf{x}^+) \tag{6.17c}$$

Expanding the objective function of eq.(6.17a), we obtain:

$$f(\mathbf{x}^-) = \frac{1}{2}(\mathbf{x}^{+T}\mathbf{x}^+ + 2\mathbf{x}^{+T}\mathbf{x}^- + \mathbf{x}^{-T}\mathbf{x}^-) \tag{6.18}$$

Since $\mathbf{x}^{+T}\mathbf{x}^+$ is constant, it can be dropped from the objective function. Furthermore, $\mathbf{x}^{+T}\mathbf{x}^- = 0$ since the two solution components are orthogonal. Thus, eqs. (6.17a) to (6.17c) reduce to

$$\min_{\mathbf{x}^-} \quad \frac{1}{2}\mathbf{x}^{-T}\mathbf{x}^- \tag{6.19a}$$

$$\text{subject to} \quad \mathbf{A}_1\mathbf{x}^- = \mathbf{0} \tag{6.19b}$$

$$\mathbf{A}_2\mathbf{x}^- \geq (\mathbf{b}_2 - \mathbf{A}_2\mathbf{x}^+) \tag{6.19c}$$

From the equivalent minimization problem given by eqs.(6.19a) to (6.19c), it becomes apparent that solving the system given by eqs.(6.15), (6.14b) and (6.14c) minimizes the norm of the solution component in the nullspace of $\mathbf{A}_1$.

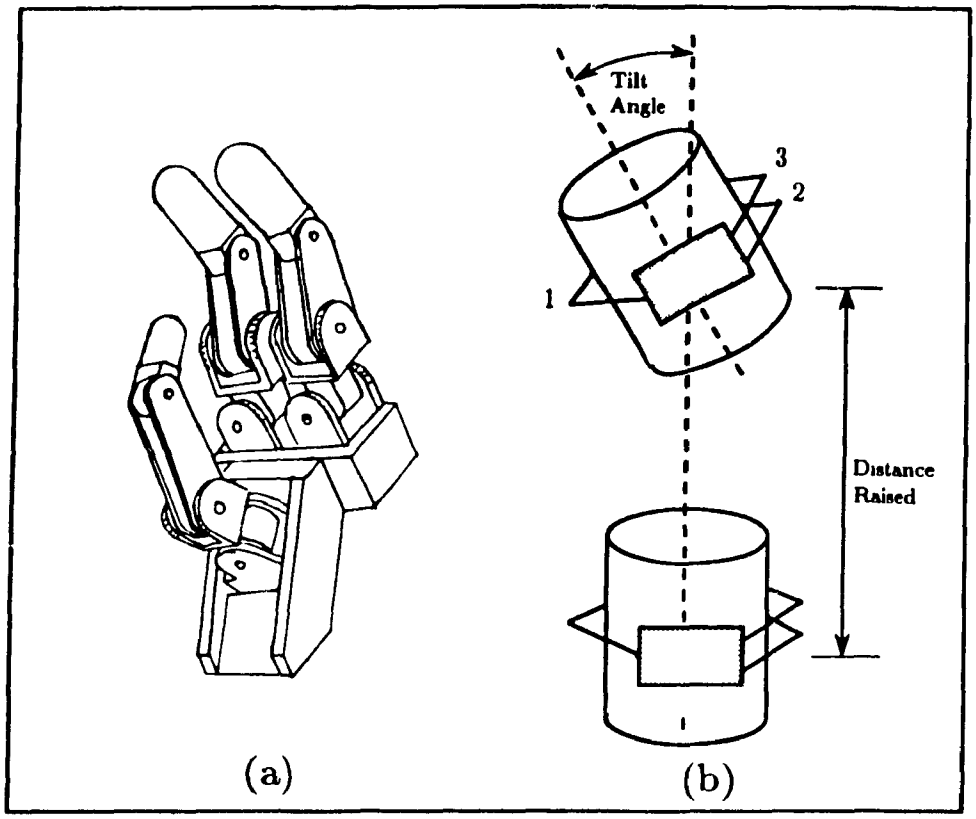

Figure 6.7 *A Three-Fingered Hand and its Task*

Equation (6.19b) specifies that $\mathbf{x}^-$ lies in the nullspace of $\mathbf{A}_1$. Kerr and Roth (1983) have shown that this constraint can be included implicitly by letting

$$\mathbf{x}^- = \mathbf{N}\mathbf{y}$$

where the columns of $\mathbf{N}$ form a basis for the nullspace of $\mathbf{A}_1$ and $\mathbf{y}$ is an arbitrary vector of dimension $n - m_{eq}$. This approach then becomes identical to that used in §5.3.1 to remove the equality constraints from the optimization problem, and the minimization problem of eqs.(6.19a) to (6.19c) then reduces to an inequality-constrained problem of the form given by eqs.(5.22a), (5.22b) and (5.22d).

The technique of minimizing internal forces has already been applied to a number of examples in §6.1.2 and 6.1.3 in order to compare its results to those of linear-programming. A further example is now shown which considers the three-fingered hand depicted in Figure 6.7(a) raising and pouring a glass of water. The three fingers are

Parameter	$i = 1$	$i = 2$	$i = 3$
a_i (mm)	20	35	25
b_i (mm)	0	0	0
α_i (deg)	-90	0	0
m_i (g)	21.6	37.8	27.0
I_i^{xx} (g-mm^2)	1440	2520	1800
I_i^{yy} (g-mm^2)	1440	5120	2310
I_i^{zz} (g-mm^2)	1440	5120	2310

Table 6.4 Hartenberg-Denavit Parameters and Inertia Properties of Link i of Each Finger

identical and have the kinematic parameters and inertia properties given in Table 6.4 The task geometry is shown in Figures 6.7(b), 6.8(a) and 6.8(b). From $t = 0$ to 1 s, the glass, of 60 mm diameter and 120 mm height, is raised vertically upward over a distance of 200 mm, and from $t = 1$ to 2 s, the glass is tilted through 90 degrees, thereby emptying its contents.

The combined mass of the glass and water is assumed to decrease from 0.4 kg before the water is poured to 0.15 kg when it is fully tipped. The corresponding moments of inertia are calculated assuming that the glass and water are a uniform cylindrical solid before the water is poured, and a cylindrical shell when it is fully poured. All transitions between the extreme values are done as linear functions of the tilt angle. Hard point contact is assumed so that each contact wrench consists of three forces Thus, there are nine unknown contact forces and six scalar equations of motion, resulting in a redundancy of three. Friction constraints are imposed on the contact forces with a coefficient of friction $\mu' = 0.5$, while maximum actuator torques of 0.25 N-m are allowed. Quadratic programming was implemented for this problem to minimize the 'internal force', as defined by eq.(6.15). Three inequality constraints were active throughout the task. The contact forces and actuator torques obtained for Finger #1 are shown in Figures 6.8(c) and 6.8(d), and for Finger #3 in Figures 6.8(e) and 6.8(f). The results for Finger #2 are

Algorithm	Quadratic Programming
IMSL Routine CPU Time (ms)[a]	88.3
In-House Routine CPU Time (ms)[a]	38.0
Number of Unknowns	9
Number of Equality Constraints	6
Number of Inequality Constraints	33

Table 6.5 *CPU Times for the Three-Finger Pick-and-Pour Example*

not shown, but are very similar to those for Finger #3. The CPU times required to solve the optimization problem at one instant are shown in Table 6.5.

This simulation shows that quadratic programming yields results which are continuous in time and can be obtained fast enough to be supplied as setpoints for a force controller. Once again, it should be noted that the speed of the Sun 3/80 can already be greatly and cheaply surpassed by more recent RISC-based workstations.

6.2.1.1 Achieving Invariance

Although the concept of 'internal force' is an appealing physical interpretation of the nullspace component of the solution, it has certain flaws. For example, the solution to a given problem will depend on the units used to describe it because the vector $\mathbf{x}$ contains elements which are not dimensionally homogeneous, i.e., some have units of torque, while others have units of force. Since the minimization of $\frac{1}{2}\mathbf{x}^T\mathbf{x}$ depends on the relative numerical magnitude of its component forces and moments, and this relative magnitude can be changed simply depending on the units used, the solution will not be invariant with respect to units.

[a] Using double-precision on a Sun 3/80 Workstation with 68881 floating-point co-processor

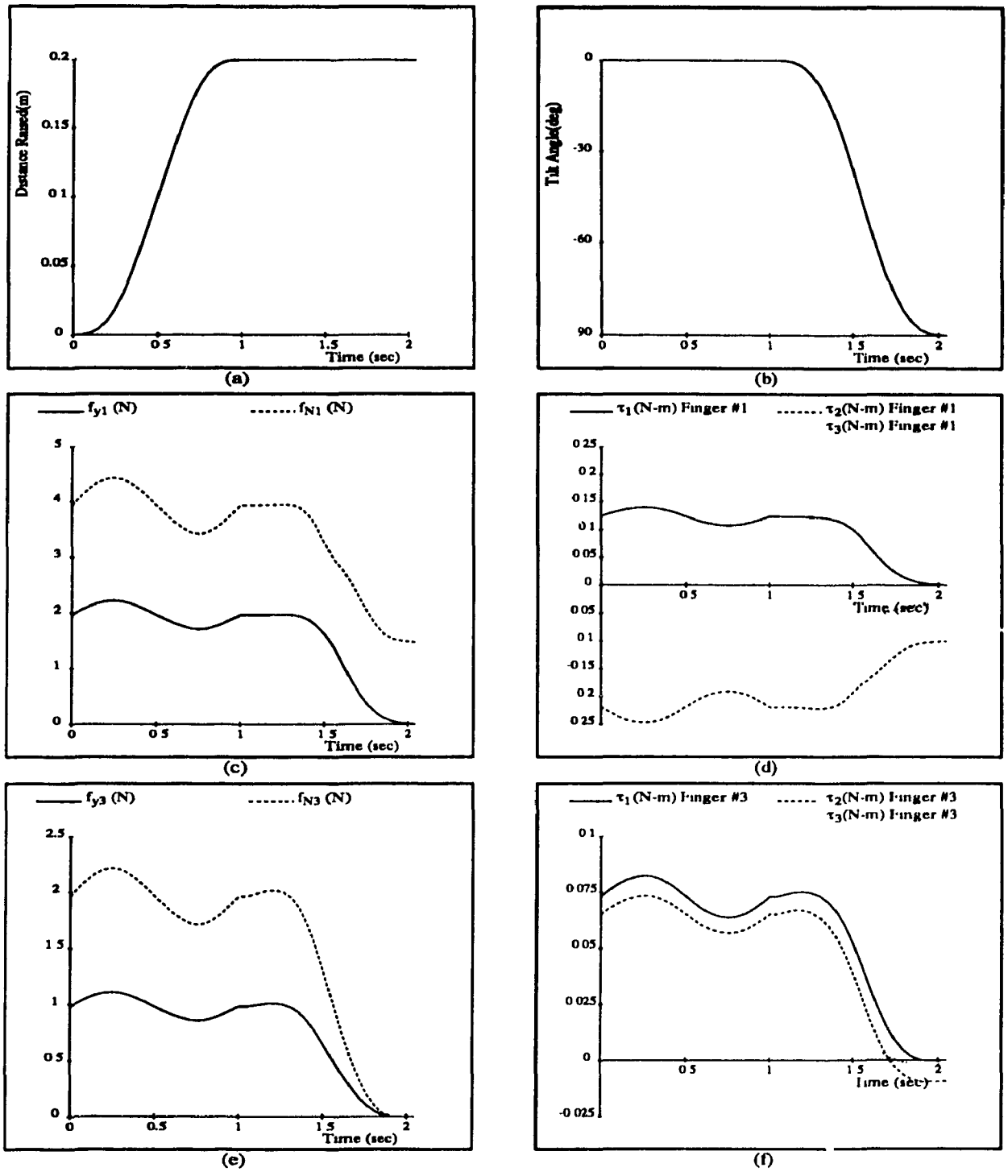

Figure 6.8 *The Pick-and-Pour Task and Resulting Forces and Torques*

It is therefore important to homogenize the units of the components of $\mathbf{x}$. This can be done by multiplying all forces by a characteristic length of the system denoted by l. Obviously, the solution obtained with the foregoing formulation will depend on the choice of l. Since $\mathbf{x}$ contains contact forces and torques, a logical characteristic length might be the root-mean-square length of the vectors from the object's centroid to the configuration-dependent contact points. Thus, we would minimize the function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x}$ where

$$\mathbf{W} = \begin{bmatrix} \mathbf{1} & & & & \mathbf{0} \\ & l^2\mathbf{1} & & & \\ & & \ddots & & \\ & & & \mathbf{1} & \\ \mathbf{0} & & & & l^2\mathbf{1} \end{bmatrix}, \qquad l = \sqrt{\frac{1}{p}\sum_{i=1}^{p} \|\mathbf{c}_i\|^2} \tag{6.20}$$

where $\mathbf{1}$ is the 3×3 identity matrix and $\mathbf{c}_i$ is a vector from the object's centroid to contact point i. This method will yield a solution invariant with changes in units which may still be interpreted as one which does not contribute to the motion of the object. If we let $\hat{\mathbf{x}} = \mathbf{L}^T\mathbf{x}$, where $\mathbf{L}$ is the lower-triangular factor of the Cholesky decomposition of $\mathbf{W}$ ($\mathbf{W} = \mathbf{L}\mathbf{L}^T$), then minimizing $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x}$ subject to eqs. (6.14b) and (6.14c) can be rewritten as

$$\min_{\hat{\mathbf{x}}} \quad \frac{1}{2}\hat{\mathbf{x}}^T\hat{\mathbf{x}} \tag{6.21a}$$

$$\text{subject to} \quad \widehat{\mathbf{A}}_1\hat{\mathbf{x}} = \mathbf{b}_1 \tag{6.21b}$$

$$\widehat{\mathbf{A}}_2\hat{\mathbf{x}} \geq \mathbf{b}_2 \tag{6.21c}$$

where $\widehat{\mathbf{A}}_1 = \mathbf{A}_1\mathbf{L}^{-T}$ and $\widehat{\mathbf{A}}_2 = \mathbf{A}_2\mathbf{L}^{-T}$. The solution of this problem will minimize the solution contribution in the nullspace of $\widehat{\mathbf{A}}_1$ which can also be interpreted as that component of the solution contributing nothing to the payload's motion, included in $\mathbf{b}_1$.

Including a non-identity weighting matrix $\mathbf{W}$ will cause an increase in the computational load due to the Cholesky decomposition that must now be performed. However, since $\mathbf{W}$ is only a function of the grasp points, its Cholesky decomposition need only be performed when these change. Furthermore, since $\mathbf{W}$ is diagonal, the cost incurred will be minimal.

It should be apparent from the above manipulations that 'the solution which contributes nothing to the motion of the object' is not unique, and some care must be exercised when attempting to attribute a physical interpretation to the internal force. Finally, an example in which the *invariant* internal force is minimized is given in §6.2.7.4 where it is compared to the minimization of power losses in the system.

6.2.2 Distributing the Load

In the early stages of comparing various cost functions for walking machines, it appeared that it would be detrimental to minimize the vertical and lateral foot contact forces together. It was felt that minimizing an objective function $f(\mathbf{x})$ which contained the sum of squares of all these forces would result in a system where lower vertical forces would be generated at the expense of large lateral forces. Intuitively, the ideal distribution would have the vertical force on each leg equal to W/p, where W is the weight of the machine and p is the number of legs in ground contact, and the lateral force equal to zero. A second objective function $g(\mathbf{x})$ which minimized the square of the departure of the vertical force on each leg from W/p was therefore formulated. Yet, when the two solutions were compared, they were found to be identical. In the following explanation to this phenomenon, f_{N_i} is the component of $\mathbf{x}$ representing the vertical force on the i-th leg, $\sum_{i=1}^{p} f_{N_i} = W$ and $g(\mathbf{x})$ is that part of the objective which is not dependent on the vertical contact forces.

$$f(\mathbf{x}) = \sum_{i=1}^{p} f_{N_i}^2 + g(\mathbf{x}) \tag{6.22}$$

$$\begin{aligned} g(\mathbf{x}) &= \sum_{i=1}^{p} (f_{N_i} - W/p)^2 + g(\mathbf{x}) && (6.23a)\\ &= \sum_{i=1}^{p} (f_{N_i}^2 - 2W f_{N_i}/p + W^2/p^2) + g(\mathbf{x}) && (6.23b)\\ &= f(\mathbf{x}) - 2W/p \sum_{i=1}^{p} f_{N_i} + W^2/p && (6.23c)\\ &= f(\mathbf{x}) - W^2/p && (6.23d) \end{aligned}$$

Since the two objective functions differ only by the constant second term in eq.(6.23d), the minimization of either of these two objective functions will yield the same result.

The approach considered above can be generalized to the case where it is desired to minimize the departure of the vector of design variables $\mathbf{x}$ from a desired value $\mathbf{x}_o$. For example, minimizing the normal forces exerted on the object could result in a weak grasp which would still satisfy the constraint equations. It may be more appropriate, in this case, to minimize the departure from a *desired* normal force which would depend on the particular task at hand. When grasping an egg, for example, the desired normal force would be lower than when grasping a steel object of the same size, shape and weight. Under these conditions, the objective function might be

$$f(\mathbf{x}) = \mathbf{c}^T(\mathbf{x} - \mathbf{x}_o) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_o)^T\mathbf{W}(\mathbf{x} - \mathbf{x}_o) \tag{6.24}$$

which, after expansion and dropping constant terms, yields the equivalent objective function below:

$$f(\mathbf{x}) = \tilde{\mathbf{c}}^T\mathbf{x} + \frac{1}{2}\mathbf{x}\mathbf{W}\mathbf{x} \tag{6.25a}$$

where

$$\tilde{\mathbf{c}}^T = \mathbf{c}^T + \mathbf{x}_o^T\mathbf{W} \tag{6.25b}$$

6.2.3 Minimizing a Norm of the Actuator Torques

Among others, Zheng and Luh (1989) and Danowski (1989) have proposed minimizing the norm of the actuator torques by using the objective function

$$f(\boldsymbol{\tau}) = \frac{1}{2}\boldsymbol{\tau}^T\boldsymbol{\tau} \tag{6.26a}$$

where $\boldsymbol{\tau}$ denotes a vector composed of *all* the actuator torques in the system, i.e.,

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_p \end{bmatrix} \tag{6.26b}$$

Equation (4.20) can be written for each path between the two poles, which yields, upon assembling of the whole system of equations,

$$\boldsymbol{\tau} = \boldsymbol{\tau}' + \mathbf{J}^T\mathbf{x} \tag{6.27a}$$

where

$$\boldsymbol{\tau}' = \begin{bmatrix} \boldsymbol{\tau}'_1 \\ \vdots \\ \boldsymbol{\tau}'_p \end{bmatrix}, \qquad \mathbf{J} = \mathrm{diag}\,[\,\mathbf{J}_{s1} \quad \dots \quad \mathbf{J}_{sp}\,] \tag{6.27b}$$

Substituting eq.(6.27a) into eq.(6.26a) and dropping the constant term $\boldsymbol{\tau}'^T\boldsymbol{\tau}'$, we obtain the equivalent objective function given by

$$g(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{6.28a}$$

where

$$\mathbf{c} = \begin{bmatrix} \mathbf{J}_{s1}\boldsymbol{\tau}'_1 \\ \vdots \\ \mathbf{J}_{sp}\boldsymbol{\tau}'_p \end{bmatrix}, \qquad \mathbf{W} = \mathrm{diag}\,[\,\mathbf{J}_{s1}\mathbf{J}_{s1}^T \quad \dots \quad \mathbf{J}_{sp}\mathbf{J}_{sp}^T\,] \tag{6.28b}$$

where each block element $\mathbf{J}_{si}\mathbf{J}_{si}^T$ is of dimension 6×6.

Since eq.(6.28a) is formally identical to eq.(6.14a), the optimization problem is left unchanged in principle and can be solved using the linear-quadratic programming techniques detailed in Chapter 5. As well, since all matrices in the new problem are of the same dimension as the original problem, we can expect that the increase in the CPU time required to solve this optimization problem will be that required to perform *a)* the matrix multiplications in eq.(6.28b), and *b)* the Cholesky decomposition of $\mathbf{W}$ *at each time step*. Furthermore, minimization of a combination of the actuator torques and the contact wrenches can be easily performed by adding a term $\frac{1}{2}\mathbf{x}^T\mathbf{x}$ to eq.(6.28a).

Alberts and Soloway (1988) claim that minimizing the norm of $\mathbf{x}$, the contact wrenches at the interface between multiple manipulators and their common payload, is equivalent to minimizing the actuator torques acting in the system. However, eqs.(6.28a) and (6.28b) clearly show that the two are not equivalent since the relative weighting

between the elements of $\mathbf{x}$ will be altered by the configuration of the manipulator, as given by its Jacobian.

6.2.4 Minimizing a Norm of the Joint Constraint Wrenches

The joint constraint wrenches can also be minimized by adopting a similar approach to that used in §6.2.3. If we let $\hat{\mathbf{w}}$ denote the vector composed of the actuator and constraint wrenches at *all* the joints in the system, i.e.,

$$\hat{\mathbf{w}} = \begin{bmatrix} \hat{\mathbf{w}}_1 \\ \vdots \\ \hat{\mathbf{w}}_p \end{bmatrix} \tag{6.29a}$$

then we are interested in minimizing

$$f(\hat{\mathbf{w}}) = \frac{1}{2}\hat{\mathbf{w}}^T\hat{\mathbf{w}} \tag{6.29b}$$

Equation (4.29a) can be written for each path between the two poles. Upon assembling of all the arising equations, one obtains:

$$\hat{\mathbf{w}} = \hat{\mathbf{w}}' + \mathbf{F}^T\mathbf{x} \tag{6.30a}$$

where

$$\hat{\mathbf{w}}' = \begin{bmatrix} \hat{\mathbf{w}}'_1 \\ \vdots \\ \hat{\mathbf{w}}'_p \end{bmatrix}, \qquad \mathbf{F} = \text{diag}\,[\,\mathbf{F}_1 \quad \dots \quad \mathbf{F}_p\,] \tag{6.30b}$$

Once again, eq.(6.30a) can be substituted into eq.(6.29b). If the constant term $\hat{\mathbf{w}}'^T\hat{\mathbf{w}}'$ is dropped, we obtain the equivalent objective function given by

$$g(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{6.31a}$$

where

$$\mathbf{c} = \begin{bmatrix} \mathbf{F}_1\hat{\mathbf{w}}'_1 \\ \vdots \\ \mathbf{F}_p\hat{\mathbf{w}}'_p \end{bmatrix} \qquad \mathbf{W} = \text{diag}\,[\,\mathbf{F}_1\mathbf{F}_1^T \quad \dots \quad \mathbf{F}_p\mathbf{F}_p^T\,] \tag{6.31b}$$

where each block element $\mathbf{F}_i\mathbf{F}_i^T$ is of dimension $6g_{p_i} \times 6g_{p_i}$ and g_{p_i} is the number of joints

in the i-th path between the two poles.

Equation (6.31a) is formally identical to eq.(6.14a), and the optimization problem is again left unchanged in principle. The same comments apply regarding the increased computational load as were made in §6.2.3. Minimization of a combination of the constraint *and* contact wrenches would be performed by adding a term $\frac{1}{2}\mathbf{x}^T\mathbf{x}$ to eq.(6.31a).

6.2.5 Nakamura's Strain Energy Objective Function

Nakamura (1988b) has proposed a technique to minimize the strain energy stored in an object which is grasped by multiple robotic devices. The elastic strain energy is written for the object as

$$U = \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{6.32a}$$

where $\mathbf{x}$ represents the vector of contact wrenches applied to the object and

$$\mathbf{W} = \text{diag}\left[\,\mathbf{W}_1 \quad \ldots \quad \mathbf{W}_p\,\right] \tag{6.32b}$$

$$\mathbf{W}_i = \text{diag}\left[\,\frac{1}{k^x_{ni}} \quad \frac{1}{k^y_{ni}} \quad \frac{1}{k^z_{ni}} \quad \frac{1}{k^x_{fi}} \quad \frac{1}{k^y_{fi}} \quad \frac{1}{k^z_{fi}}\,\right] \tag{6.32c}$$

The scalars k^x_{ni}, k^y_{ni} and k^z_{ni} represent the object's torsional stiffness in the x, y and z directions, respectively, while k^x_{fi}, k^y_{fi} and k^z_{fi} represent its translational stiffness in those same directions. Nakamura evaluated these stiffnesses by *a)* assuming that the object behaved as a cylindrical solid from all directions, and *b)* assuming that the contact force and torque vectors were both aligned with the principal axis of the cylinder. No numerical results were shown by Nakamura.

Since the effect of this method is simply to add a weighting matrix to the more usual internal force minimization and, as was shown in §6.2.1.1, a weighting matrix should be added anyways to achieve invariance, this method is not expected to significantly increase the computational load to solve the optimization problem. This, of course, neglects

the additional time required to calculate the object stiffnesses. Depending on the geometrical complexity of the grasped object and the technique used to estimate its stiffness, this may or may not be a reasonable assumption.

6.2.6 Danowski's and Pfeiffer et al.'s Objective Functions

Danowski (1989) and Pfeiffer *et al.* (1990) analyze the dynamics of *carausius morosus*, an insect popularly known as the walking stick, and present a mathematical model to simulate its motion. They model the insect as consisting of a single central body and six legs, each of which is made up of three links, the model thus consisting of 19 bodies in all. The insect is modeled to move with an alternating tripod gait. The form of the resulting equations is very similar to that given in §4.2.2—it is composed of six equations for the main body and three equations for each 3-link leg. This results in a system of 24 scalar equations with 36 unknowns, the latter being the contact force vector $\mathbf{f}$ with 3 components at each of six legs, and the 18 components of the actuator torque vector $\boldsymbol{\tau}$. The degree of actuation redundancy is therefore 12 when all six legs are on the ground.

In order to resolve this underdeterminacy, Danowski (1989) proposed to minimize a quadratic objective function of the form given by eq.(6.14a) with $\mathbf{c} = \mathbf{0}$. The need to satisfy the inequality constraints usually considered important in walking machines was neglected on the premise that the insect can *grasp* the ground. The numerical solution method adopted was that of explicit Lagrange multipliers outlined in §5.4.2.1. This formulation results in a determinate system of equations of order 60 ($= n + m_{eq}$). When only 3 legs are in ground contact, the order of the system solved is *increased* to 69 since further equalities were used to constrain the contact forces at the non-contacting legs to be zero. As was shown in §4.3, when certain legs are no longer in ground contact, their dynamics are decoupled from the rest of the system and should be solved independently to improve computing efficiency.

The particular objective functions implemented by Danowski (1989) are:

1. Minimum norm of $\mathbf{f}$, the vector of foot contact forces. It is claimed that this solution "yields high active torques in the outer leg joints" and evokes human experience to imply that this is not realistic—that it is more natural for a biological system to resist external loads using forces than torques.

2. Minimum norm of $\boldsymbol{\tau}$, the vector of all actuator torques in the system, as given in §6.2.3.

3. Minimization of the elastic bending energy in the legs. It is contended that this yields a more 'natural' solution since it takes the bending load of the legs into account, thereby resulting in low actuator torques and bending loads especially in the outer, thin leg segments. Each leg link is modeled as a beam of uniform circular cross-section. The bending energy is therefore

$$U = \int_0^l \frac{M(x)^2}{EI} dx \tag{6.33}$$

where $M(x)$ is the bending moment at any point x along the link. Since this objective function places no penalty on the lateral contact forces, it is bound to come closer to foot slippage.

4. A combination of (1) and (3) which minimizes the bending load in all the legs, as well as the contact forces on the two front legs, on the premise that these front legs are used as 'feelers' and should not be heavily loaded.

The results shown by Danowski (1989) lead to the following conclusions:

1. Although the foot contact-force time histories are significantly different for the different objective functions, the actuator torques are not. The technique with the lowest contact forces is that which minimizes the norm of the contact-force vector. The techniques with the highest contact forces are those with the bending energy term in the objective function.

2. All techniques exhibit greater or lesser discontinuities when the feet cc me into contact or break contact with the ground (i.e., at changes in topology). The technique which shows the least severe discontinuities is that which minimizes the contact forces. The ones with the worst discontinuities are the ones with the bending energy term in the objective function.

Since the benefits of the objective function proposed by Danowski (1989) and Pfeiffer *et al.* (1990) were not apparent, their technique was not pursued further.

6.2.7 Minimum Power Consumption

The minimization of power can be important, particularly in situations where the installed power is restricted such as in space-based robotic applications. Techniques which claim to minimize power have been proposed by Carignan and Akin (1989), Kopf (1988a, 1988b), Zheng and Luh (1989) and Orin and Oh (1981). With the exception of Orin and Oh (1981), these works claim the power to be proportional to the square of actuator torques, though no justification is given for this assumption. The objective function proposed by Orin and Oh (1981), which is justified in that work, was reviewed in §6.1.1 and was concluded to be suitable for the particular mechanical construction of the OSU Hexapod.

The previous chapters have clearly shown that a system composed of multiple cooperating robotic devices admits an infinity of solutions to its inverse dynamics equations. In the following section, it will be shown that the power imparted to the system cannot be optimized since this is strictly a function of the prescribed motion. However, assuming certain loss characteristics for the dc servomotors commonly used in robotic manipulators (Armstrong *et al.*, 1986; Leahy and Saridis, 1989), it is shown that the minimization of power losses can be cast as a linear-quadratic optimization problem. A model of the dc servomotors is derived and the losses which can be minimized are identified. These are then written in the form of eq.(6.14a). Local and global perfor-

mance indices are then proposed to allow comparison of the minimum power loss and the minimum internal force approaches. An example of two Puma 560 robotic manipulators handling a payload is shown to demonstrate the proposed technique. Various objective functions are compared for the same trajectory: minimum internal force, minimum power using the present formulation and minimum power using the formulation proposed by Kopf (1988a, 1988b).

6.2.7.1 On the Constancy of Power Imparted to The System

The power supplied to the actuators is consumed in a number of ways, the most useful being that component which is imparted to the system, i.e., the manipulators and the grasped object.

The power supplied by the actuators to the system is

$$\pi = \sum_{i=1}^{p} \boldsymbol{\tau}_i^T \dot{\boldsymbol{\theta}}_i \tag{6.34}$$

where $\boldsymbol{\tau}_i$ is the vector of actuator torques in the i-th manipulator, and $\boldsymbol{\theta}_i$ is the corresponding vector of joint rates. It was shown in §4.2 that there exists many sets of actuator torques which will result in the same *prescribed* motion of the manipulator/payload system. Thus, it would appear that we should be able to minimize π. However, it will now be shown that π cannot be optimized by varying the $\boldsymbol{\tau}_i$ while also satisfying the inverse dynamics equations given by eq. (6.14b).

Substituting eq.(4.20) into eq.(6.34), we obtain

$$\begin{aligned} \pi &= \sum_{i=1}^{p} (\boldsymbol{\tau}_i' + \mathbf{J}_{si}^T \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix})^T \dot{\boldsymbol{\theta}}_i \\ &= \sum_{i=1}^{p} (\boldsymbol{\tau}_i'^T \dot{\boldsymbol{\theta}}_i + [\mathbf{n}_i^T \quad \mathbf{f}_i^T] \mathbf{J}_{si} \dot{\boldsymbol{\theta}}_i) \end{aligned} \tag{6.35}$$

where $\boldsymbol{\tau}_i'$ is uniquely determined by the payload motion for manipulators which are not kinematically redundant. We now recall that the expression $\mathbf{J}_{si}\dot{\boldsymbol{\theta}}_i$ is nothing but the *twist*,

$\mathbf{t}_i$, at the tip of subchain i, which is defined as

$$\mathbf{t}_i = \begin{bmatrix} \boldsymbol{\omega}_i \\ \mathbf{v}_i \end{bmatrix} \tag{6.36}$$

where $\boldsymbol{\omega}_i$ and $\mathbf{v}_i$ are the angular velocity of the i-th manipulator's end link and the translational velocity of a point P_i of this link, respectively. Substituting this expression into eq.(6.35), we obtain

$$\begin{aligned} \pi &= \sum_{i=1}^{p} (\boldsymbol{\tau}_i'^T \dot{\boldsymbol{\theta}}_i + \mathbf{n}_i^T \boldsymbol{\omega}_i + \mathbf{f}_i^T \mathbf{v}_i) \\ &= \sum_{i=1}^{p} \boldsymbol{\tau}_i'^T \dot{\boldsymbol{\theta}}_i + \sum_{i=1}^{p} \mathbf{n}_i^T \boldsymbol{\omega}_i + \sum_{i=1}^{p} \mathbf{f}_i^T \mathbf{v}_i \end{aligned} \tag{6.37}$$

Since the end links of all the subchains are rigidly attached to the same rigid object, we can write

$$\boldsymbol{\omega}_1 = \boldsymbol{\omega}_2 = \ldots = \boldsymbol{\omega}_p = \boldsymbol{\omega} \tag{6.38}$$

Furthermore, we can write all the translational tip velocities, $\mathbf{v}_i$, in terms of the velocity $\mathbf{v}_r$ of a reference point on the payload as

$$\mathbf{v}_i = \mathbf{v}_r + \boldsymbol{\omega} \times \mathbf{c}_i \tag{6.39}$$

where $\mathbf{c}_i$ is a vector from the reference point to the i-th grasping point. Substituting eqs.(6.38) and (6.39) into eq.(6.37) yields

$$\pi = \sum_{i=1}^{p} \boldsymbol{\tau}_i'^T \dot{\boldsymbol{\theta}}_i + (\sum_{i=1}^{p} \mathbf{n}_i^T)\boldsymbol{\omega} + (\sum_{i=1}^{p} \mathbf{f}_i)^T \mathbf{v}_r + \sum_{i=1}^{p} \mathbf{f}_i^T (\boldsymbol{\omega} \times \mathbf{c}_i) \tag{6.40}$$

Using

$$\mathbf{f}_i^T (\boldsymbol{\omega} \times \mathbf{c}_i) = (\mathbf{c}_i \times \mathbf{f}_i)^T \boldsymbol{\omega} \tag{6.41}$$

we can obtain

$$\begin{aligned} \pi &= \sum_{i=1}^{p} \boldsymbol{\tau}_i'^T \dot{\boldsymbol{\theta}}_i + (\sum_{i=1}^{p} \mathbf{n}_i^T)\boldsymbol{\omega} + (\sum_{i=1}^{p} \mathbf{f}_i)^T \mathbf{v}_r + \sum_{i=1}^{p} (\mathbf{c}_i \times \mathbf{f}_i)^T \boldsymbol{\omega} \\ &= \sum_{i=1}^{p} \boldsymbol{\tau}_i'^T \dot{\boldsymbol{\theta}}_i + (\sum_{i=1}^{p} \mathbf{f}_i)^T \mathbf{v}_r + (\sum_{i=1}^{p} [\mathbf{n}_i + \mathbf{c}_i \times \mathbf{f}_i])^T \boldsymbol{\omega} \end{aligned} \tag{6.42}$$

Comparing the last two terms of eq.(6.42) to the right-hand-sides of eqs.(4.7a) and (4.7b), we find that eq.(6.42) can be rewritten as

$$\pi = \sum_{i=1}^{p} \boldsymbol{\tau}_i'^T \dot{\boldsymbol{\theta}}_i + [m_o(\mathbf{a}_o + \mathbf{g}) - \mathbf{f}_e]^T \mathbf{v}_r + (\mathbf{I}_o \dot{\boldsymbol{\omega}}_o + \boldsymbol{\omega}_o \times \mathbf{I}_o \boldsymbol{\omega}_o - \mathbf{n}_e - \mathbf{c}_e \times \mathbf{f}_e)^T \boldsymbol{\omega} \qquad (6.43)$$

It is apparent that all the variables in eq.(6.43) are strictly functions of the *prescribed* motion, and so, cannot be modified. Thus, since none of the terms in π can be altered by changing the grasping wrench, the power imparted to the system cannot be optimized by choosing $\mathbf{f}_i$ and $\mathbf{n}_i$ for $i = 1, \ldots, p$.

In fact, this result is intuitively obvious. If we consider a manipulator/payload system and assume the payload motion to be completely known, we can uniquely find the motion of the manipulators through inverse position, velocity and acceleration kinematics. Thus, the motion of the complete system is known and the energy contained in that system is also known at every instant. If the energy is known at every instant, the power being imparted to the system is uniquely known as well. Therefore, if we do not take into account any losses in the system and the motion of the system is prescribed, then the power which must be supplied to the system by the actuators is unique and cannot be optimized.

6.2.7.2 Minimization of Power Losses

Although the power imparted to the system cannot be optimized by changing the grasping wrench, the power input to the actuators may be reduced by minimizing the losses in the drivetrain. These losses can be substantial, particularly in geared systems. In order to do this, certain actuators characteristics must be assumed. For example, if the actuators are electric motors, we can model the i-th motor as shown in Figure 6.9, where the symbols are defined as follows:

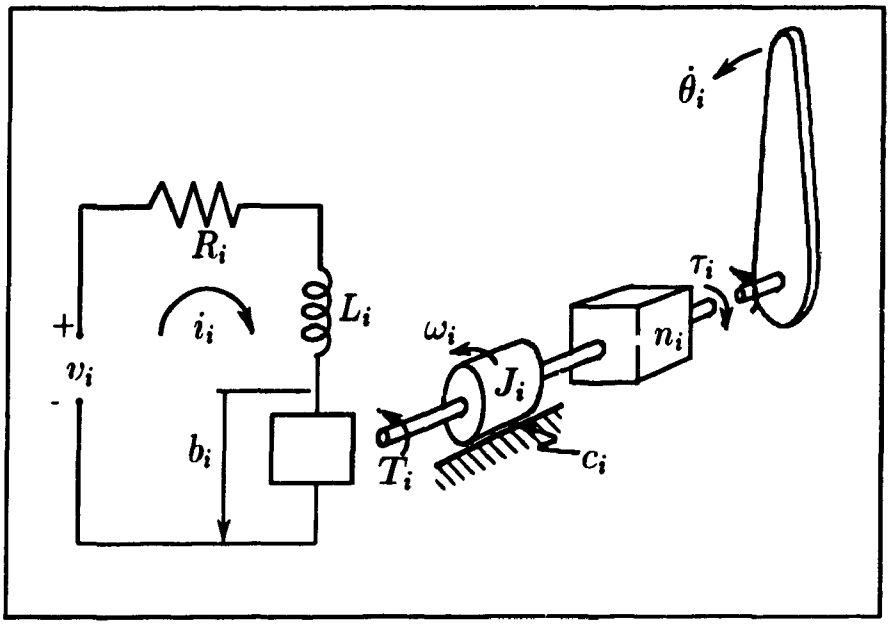

Figure 6.9 *Representation of an Electric Motor and its Gear Train*

$i_i \equiv$ armature current	$J_i \equiv$ rotor inertia
$v_i \equiv$ armature voltage	$c_i \equiv$ rotor viscous damping constant
$R_i \equiv$ armature resistance	$\omega_i \equiv$ rotor angular velocity
$L_i \equiv$ armature inductance	$n_i \equiv$ gear ratio
$b_i \equiv$ back emf	$\dot{\theta}_i \equiv$ link angular velocity about joint axis
$T_i \equiv$ motor torque	$f_i \equiv$ break-away torque
	$\tau_i \equiv$ load torque as found from eq.(4.20)

The constitutive relations for the electromechanical coupling are

$$T_i = K_i i_i \qquad b_i = B_i \omega_i \tag{6.44}$$

where K_i and B_i are the motor's torque constant and back-emf constant, respectively, and are equal if consistent units are chosen for T_i, i_i, b_i and ω_i. The governing equations for this system are

$$v_i = i_i R_i + L_i \frac{di_i}{dt} + B_i \omega_i \tag{6.45a}$$

$$K_i i_i = J_i \dot{\omega}_i + c_i \omega_i + \text{sgn}(\omega_i) \frac{f_i}{n_i} + \frac{\tau_i}{n_i} \tag{6.45b}$$

Since we are more interested in writing the relevant equations in terms of the

rates on the link side of the gear train, we can use the relations

$$\omega_i = \qquad \dot{\omega}_i = n_i\ddot{\theta}_i \tag{6.46}$$

and substitute them into eq.(6.45b) to obtain

$$n_iK_ii_i = n_i^2J_i\ddot{\theta}_i + n_i^2c_i\dot{\theta}_i + \mathrm{sgn}(\dot{\theta}_i)f_i + \tau_i \tag{6.47a}$$

or

$$i_i = \frac{1}{n_iK_i}(n_i^2J_i\ddot{\theta}_i + n_i^2c_i\dot{\theta}_i + \mathrm{sgn}(\dot{\theta}_i)f_i + \tau_i) \tag{6.47b}$$

For the dc servomotors used in robotic applications, the armature inductance can be neglected (Leahy and Saridis, 1989; Electro-Craft Corporation, 1980), so that eq.(6.45a) can be rewritten as

$$v_i = i_iR_i + B_i\omega_i = i_iR_i + n_iB_i\dot{\theta}_i \tag{6.48}$$

On the other hand, the power consumed by the motor can be written as

$$\begin{aligned}\pi_i &= i_iv_i \\ &= i_i^2R_i + i_in_iB_i\dot{\theta}_i\end{aligned} \tag{6.49}$$

Substituting eq.(6.47b) into the second term of eq.(6.49), we obtain

$$\pi_i = i_i^2R_i + [n_i^2J_i\ddot{\theta}_i\dot{\theta}_i + n_i^2c_i\dot{\theta}_i^2 + \mathrm{sgn}(\dot{\theta}_i)f_i\dot{\theta}_i + \tau_i\dot{\theta}_i] \tag{6.50}$$

where B_i/K_i is equal to 1 and hence, was omitted. The first term in eq.(6.50) represents the winding resistive loss; the second term is the rate of change of the kinetic energy of the rotor; the third and fourth terms are the velocity-dependent losses; while the last term is the power imparted to the system. It was previously shown that the last term summed over all the actuators cannot be optimized, since it is strictly a function of the motion of the system. The rate of change of the kinetic energy of the rotor cannot be minimized either because it is also strictly a function of the system motion. Similarly, the

velocity-dependent losses cannot be minimized once the system motion and the actuator characteristics (c_i and f_i) are known. However, we *can* minimize the component of the winding resistive losses which is dependent on the load torques. If we substitute eq.(6.47b) into this term, we obtain

$$\begin{aligned} i_i^2 R_i &= \frac{R_i}{n_i^2 K_i^2}[n_i^2 J_i \ddot{\theta}_i + n_i^2 c_i \dot{\theta}_i + \mathrm{sgn}(\dot{\theta}_i) f_i + \tau_i]^2 \\ &= \frac{R_i}{n_i^2 K_i^2}[n_i^4 J_i^2 \ddot{\theta}_i^2 + n_i^4 c_i^2 \dot{\theta}_i^2 + f_i^2 + \tau_i^2 \\ &\quad + 2n_i^4 J_i c_i \ddot{\theta}_i \dot{\theta}_i + 2\mathrm{sgn}(\dot{\theta}_i) n_i^2 J_i \ddot{\theta}_i f_i + 2n_i^2 J_i \ddot{\theta}_i \tau_i \\ &\quad + 2\mathrm{sgn}(\dot{\theta}_i) n_i^2 c_i \dot{\theta}_i f_i + 2n_i^2 c_i \dot{\theta}_i \tau_i + 2\mathrm{sgn}(\dot{\theta}_i) f_i \tau_i] \end{aligned} \tag{6.51}$$

Of the above terms, only the ones which are functions of τ_i can be minimized, the rest being strictly functions of the prescribed motion. Collecting the terms which are functions of the load torque and summing them over all the actuators, we obtain the objective function which must be minimized as

$$f(\boldsymbol{\tau}) = \sum_{i=1}^{n} [\frac{R_i}{n_i^2 K_i^2}(n_i^2 J_i \ddot{\theta}_i + n_i^2 c_i \dot{\theta}_i + \mathrm{sgn}(\dot{\theta}_i) f_i \tau_i + \frac{1}{2}\frac{R_i}{n_i^2 K_i^2}\tau_i^2] \tag{6.52}$$

where n is the total number of actuators in the system. This objective function can be rewritten as

$$f(\boldsymbol{\tau}) = \mathbf{c}^T \boldsymbol{\tau} + \frac{1}{2}\boldsymbol{\tau}^T \mathbf{W} \boldsymbol{\tau} \tag{6.53a}$$

where

$$\mathbf{c} = \begin{bmatrix} R_1(n_1^2 J_1 \ddot{\theta}_1 + n_1^2 c_1 \dot{\theta}_1 + \mathrm{sgn}(\dot{\theta}_1) f_1)/n_1^2 K_1^2 \\ \vdots \\ R_n(n_n^2 J_n \ddot{\theta}_n + n_n^2 c_n \dot{\theta}_n + \mathrm{sgn}(\dot{\theta}_n) f_n)/n_n^2 K_n^2 \end{bmatrix} \tag{6.53b}$$

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau}_1 \\ \vdots \\ \boldsymbol{\tau}_p \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} R_1/n_1^2 K_1^2 & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & R_n/n_n^2 K_n^2 \end{bmatrix} \tag{6.53c}$$

This optimization problem differs from that proposed by Kopf (1988a, 1988b) due to the presence of the linear term, $\mathbf{c}^T\boldsymbol{\tau}$ in the objective function which includes the effects of motor inertia, viscous damping and dry friction losses. Because of this omission,

that approach will not yield a solution with minimum power losses. Upon substituting eq.(4.20) into eq.(6.53a) and dropping terms which are not dependent on the grasping wrenches, the objective function can be rewritten as

$$g(\mathbf{x}) = \tilde{\mathbf{c}}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\widetilde{\mathbf{W}}\mathbf{x} \tag{6.54a}$$

$$\tilde{\mathbf{c}} = \mathbf{J}(\mathbf{c} + \mathbf{W}\boldsymbol{\tau}') \qquad \widetilde{\mathbf{W}} = \mathbf{J}\mathbf{W}\mathbf{J}^T \tag{6.54b}$$

where $\boldsymbol{\tau}'$ and $\mathbf{J}$ were previously defined in eq.(6.27b).

This objective function can be minimized subject to the constraints given by eqs.(6.14b) and (6.14c), in order to obtain the solution which results in the minimum power losses in the system.

6.2.7.3 Comparison to Internal Force Minimization

The minimization of power losses is important in certain circumstances, such as space-based robotics. Proposals for multiple-arm space robotic systems are becoming increasingly common, and the installed power in these situations is very limited. A reduction of the power requirements for these systems would be beneficial, but because the payloads to be considered in that context (e.g., satellites) are often fragile and expensive, it is also undesirable to exert large unnecessary forces on them.

As discussed in §6.2.1, the 'internal force' can be viewed as the component of the solution which tends to crush or tear the payload apart, and the present approach of minimizing power losses makes no attempt to reduce these forces, we can expect that they will be generated. Furthermore, since in general it is not desirable to exert unnecessary loads on the payload, we must determine whether the benefit obtained by imposing these forces, i.e., the power saved, outweighs their detrimental effect.

For the purposes of this section, the minimization of internal force implies the minimization of $\frac{1}{2}\hat{\mathbf{x}}^T\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ was defined in §6.2.1.1. The solution obtained when

minimizing power losses can be compared to the minimum internal force solution by evaluating the following 'internal force index':

$$\eta_f = \frac{(\hat{\mathbf{x}}^T\hat{\mathbf{x}})_{int.force}}{(\hat{\mathbf{x}}^T\hat{\mathbf{x}})_{power}} \tag{6.55}$$

where the subscripts indicate what is being minimized. Since the internal force is always positive and $(\hat{\mathbf{x}}^T\hat{\mathbf{x}})_{int.force} < (\hat{\mathbf{x}}^T\hat{\mathbf{x}})_{power}$, this ratio is bound to lie between 0 and 1. A value of this index close to 1 will indicate that power minimization has not increased the internal force by much from the minimum possible.

The reduction of power losses obtained can be determined from the 'power index':

$$\eta_p = \frac{(\pi)_{power}}{(\pi)_{int.force}} \tag{6.56}$$

This index is bound to be less than 1 when $(\pi)_{int\ force}$ is positive. It may exceed 1 when both $(\pi)_{int.force}$ and $(\pi)_{power}$ are negative, i.e., when the actuators are used to *extract* energy from the system. If $(\pi)_{int.force}$ remains positive, we would like η_p to remain as low as possible. By comparing η_f and η_p, we can determine which of the two optimization approaches is more desirable. If η_f stays close to 1 while η_p stays small throughout a given trajectory, we could safely say that the power minimization approach has yielded better results. On the other hand, if the reverse is true, we would tend to favor the internal force minimization approach.

The indices described above have a drawback: they are local—that is, they provide information on the solution at one point in the trajectory, while it may be desired to compare the performance over the *complete* trajectory with global indices. In order to form these, we can integrate the quantities used to form the local indices—the power consumed by the motors and the internal force applied to the object—over the duration of the task. Thus, we define the following global indices:

$$\eta'_f = \frac{\int_{t_0}^{t_f}(\hat{\mathbf{x}}^T\hat{\mathbf{x}})_{int.force}dt}{\int_{t_0}^{t_f}(\hat{\mathbf{x}}^T\hat{\mathbf{x}})_{power}dt}, \qquad \eta'_p = \frac{\int_{t_0}^{t_f}(\pi)_{power}dt}{\int_{t_0}^{t_f}(\pi)_{int\ force}dt} \tag{6.57}$$

where t_0 and t_f are the times at which the task starts and ends, respectively. If we assume that the manipulators and payload are at rest at $t = t_0$, these indices will always

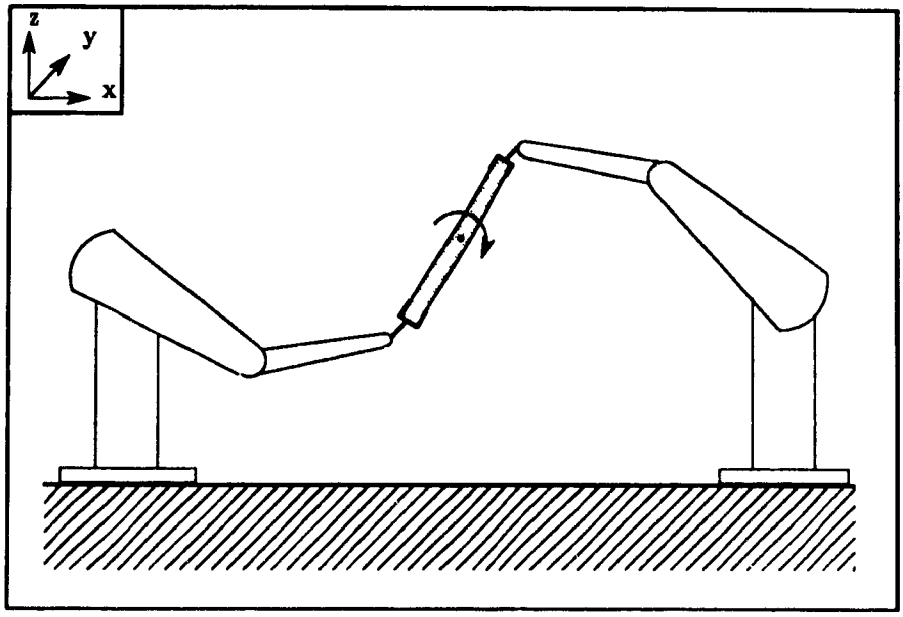

Figure 6.10 *Two Puma 560 Robots Rotating a Payload*

lie between 0 and 1. Similar to the local indices, if η'_f were close to 1 while η'_p were close to zero for a prescribed maneuver, this would indicate that a power minimization strategy should be adopted; while if the reverse were true, an internal force minimization strategy should be adopted.

6.2.7.4 Numerical Example

An example is now considered to compare the minimization of power loses and the minimization of internal force. Two Puma 560 robots, shown in Figure 6.10, rotate a common payload about a horizontal axis in the absence of a gravitational field. Puma manipulators were chosen as they have been studied extensively by other researchers, so that all the values needed in the present exercise are known. The Hartenberg-Denavit parameters were taken from (Armstrong *et al.*, 1986), as were the link masses, moments of inertia and centroid locations used in the dynamic equations of motion. The remaining parameters, shown in Table 6.6, quantify the dc servomotors used in the Puma.

The payload was a solid aluminum bar 1 m long, with a rectangular cross-

Parameter	Joint #					
	1	2	3	4	5	6
n^a	62.61	107.36	53.69	76.01	71.91	76.73
$n_i^2 J_i$ (kg-m^2)^a	1.14	4.71	0.83	0.2	0.179	0.193
$n_i^2 c_i$ (kg-m^2/s)^b	4.00	3.5	3.5	0.48	0.55	0.65
f_i (N-m)^b	5.95	6.82	3.91	1.07	0.89	0.94
τ_{lim} (N-m)^a	97.6	186.4	89.4	24.2	20.1	21.3
R_i (Ω)^c	1.6	1.6	1.6	3.9	3.9	3.9
K_i (N-m/amp)^c	0.26	0.26	0.26	0.09	0.09	0.09

Table 6.6 *Puma 560 Motor Parameters*

section 50 mm high and 100 mm deep. Its mass was therefore 13.5 kg and its moment of inertia about the relevant horizontal axis through its centroid was 1.128 kg-m^2. The payload, initially at an angle of 45° counterclockwise from the horizontal, was rotated through 90° clockwise in two seconds. Periodic splines (Angeles *et al.*, 1988) were used to ensure that the payload followed a smooth trajectory that began and ended with zero velocity and acceleration.

The underdetermined system of equations given by eq.(6.14b) was solved for this system with the following objectives:

1. Minimum internal force, i.e., minimum $\frac{1}{2}\hat{\mathbf{x}}^T\hat{\mathbf{x}}$;

2. Minimum power losses, as derived in the present work; and

3. Minimum power losses, as derived by Kopf (1988a, 1988b)

The resulting components of the grasping wrench and actuator torques for manipulator #1, on the left in Figure 6.10, are shown in Figures 6.11 and 6.12 for the three objective functions. The results are not shown for the second manipulator due to their similarity to those shown. The total winding resistive power loss for all the motors is

[a] Armstrong *et al.*, 1986
[b] Leahy and Saridis, 1989
[c] Daneshmend, 1990

Objective Function Being Minimized	$\int_{t_0}^{t_f}(\pi)dt$ (W-s)	$\int_{t_0}^{t_f}(\mathbf{x}'^T\mathbf{x}')dt$ (N^2m^2s)	η'_f	η'_p
Internal force	77.51	0.972	—	—
Power	75.90	117.2	0.00825	0.979
Power (Kopf)	76.88	18.16	0.05300	0.992

Table 6.7 *Global Performance*

shown in Figure 6.13(a), while η_f and η_p, the internal force and power indices, are shown in Figure 6.13(b). It should be noted that η_p actually exceeds 1 for the objective function proposed by Kopf during part of the trajectory due to the approximations inherent in that approach. Table 6.7 shows the total energy consumed by the motors, the integral of the internal force over the complete trajectory, as well as the global performance indices η'_f and η'_p.

It is apparent from Table 6.7 and Figures 6.11, 6.12 and 6.13 that, for the example considered, the minimum internal force approach yields the best compromise. The winding resistive loss is about 20-25% higher for this approach than for the minimum power loss approach, but since these are only *part* of the losses – c.f., eq.(6.50)– , the effect on the total energy used during the maneuver is only about 2%. By contrast, the internal force is two orders of magnitude less with the minimum internal force approach than with the minimum power losses approach. The method proposed by Kopf (1988a, 1988b) to minimize power does not minimize power for the reasons stated earlier, but yields a solution somewhere between the two approaches proposed here.

6.2.8 Smoothing of Jump Discontinuities Upon Changes in Topology

Solution of the optimization problem given by eqs.(6.14a) to (6.14c) yields a setpoint for the controller *at a given instant, for a given topology.* As was shown in

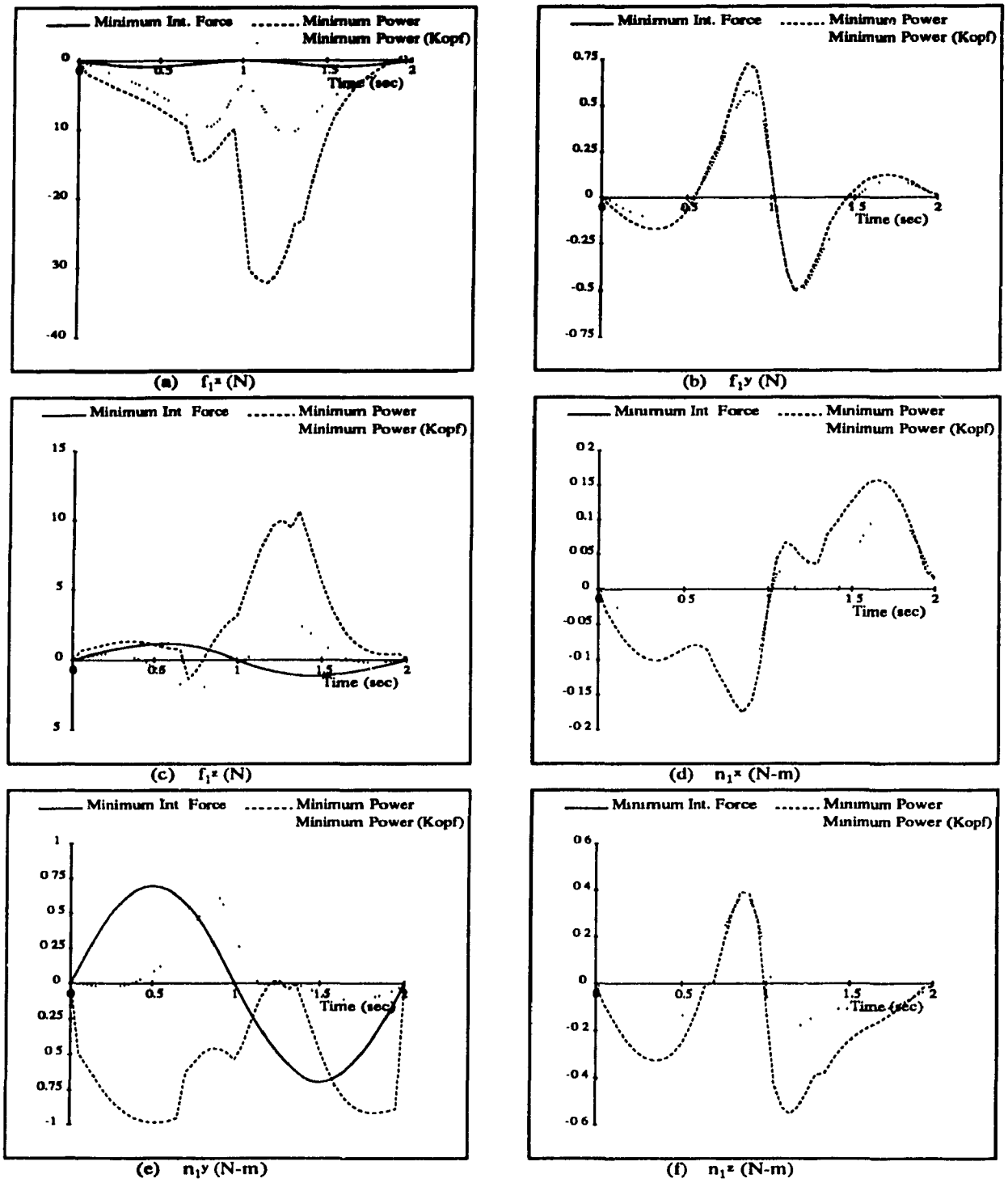

Figure 6.11 *Grasping Forces and Moments for Manipulator #1*

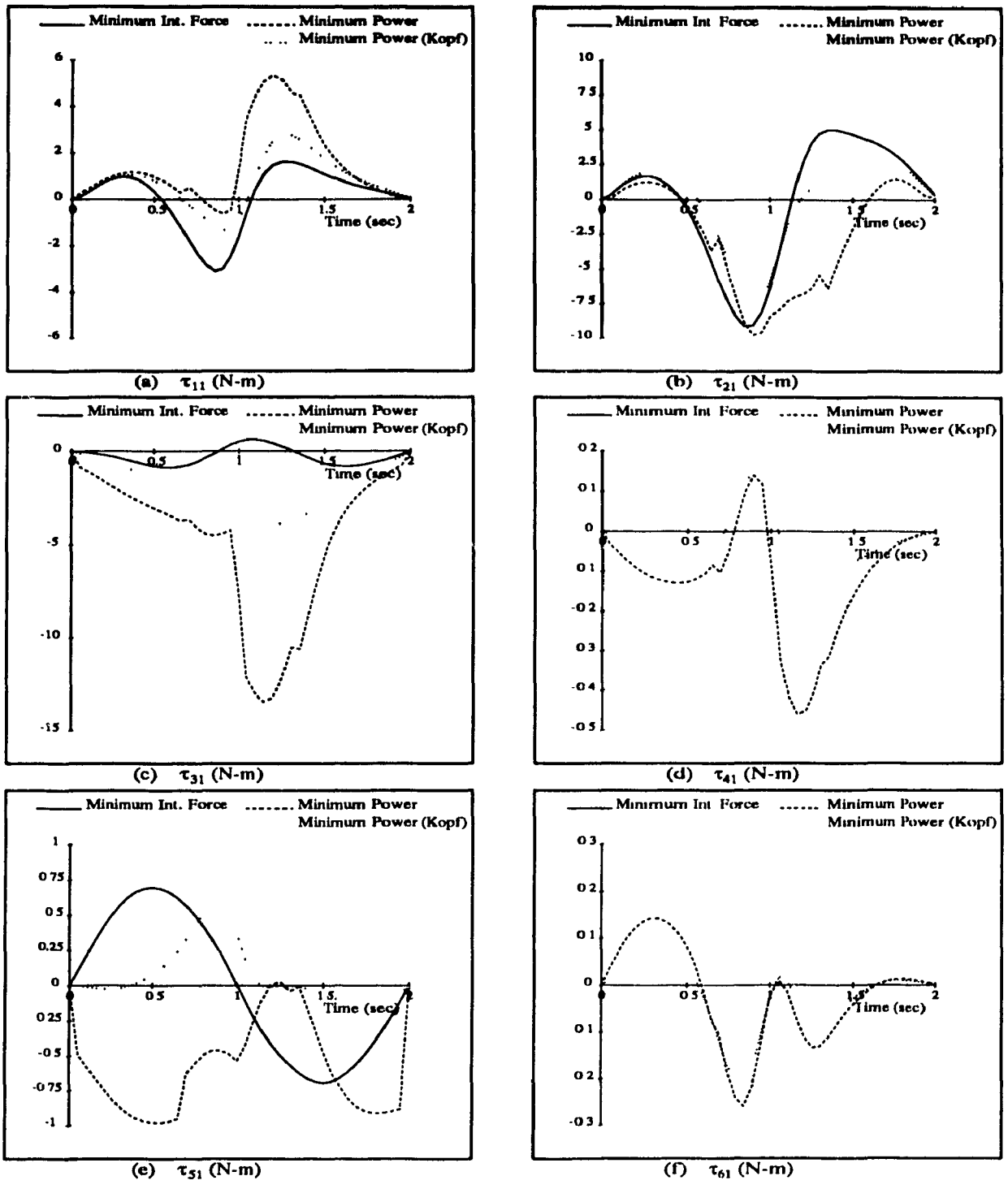

Figure 6.12 *Actuator Torques for Manipulator #1*

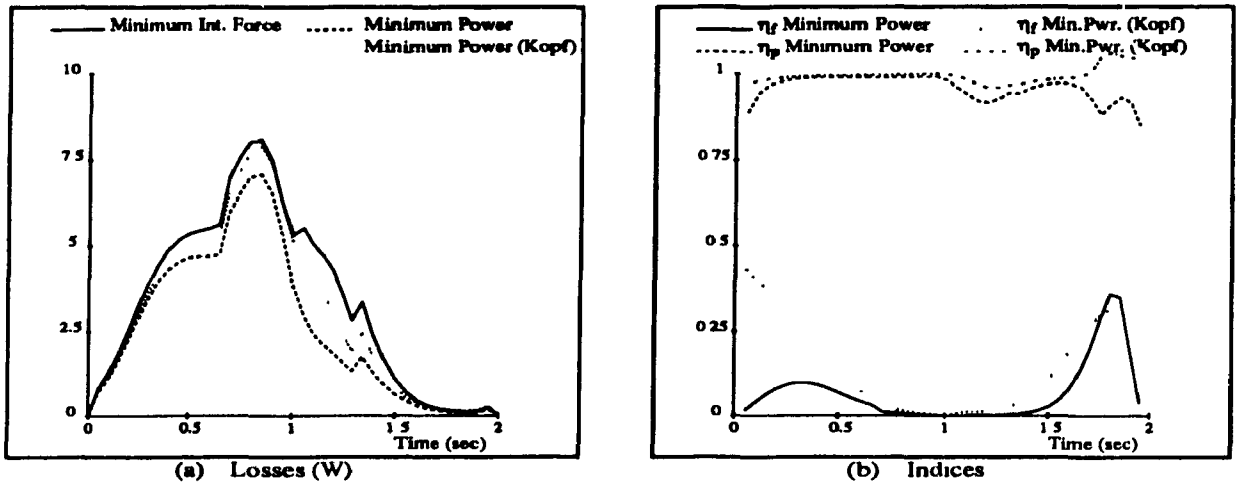

Figure 6.13 *Winding Resistive Losses and Local Performance Indices*

Chapter 4, if the topology does not change, continuous changes in configuration of the system, i.e., changes of the joint variables, result in continuous changes of the coefficients of $\mathbf{A}_1$ and $\mathbf{b}_1$, which, in turn, result in continuous changes in the optimal solution vector $\mathbf{x}$. Thus, as long as there are no changes in topology, we can expect the optimal force solution to be continuous.

As was also shown in Chapter 4, when the topology of the system changes, the entries of $\mathbf{A}_1$ change discontinuously. When this occurs, the solution to equations (6.14a) to (6.14c) will also change discontinuously, thereby causing an abrupt change in the commanded actuator torques. The actuators will be unable to respond to these commands and the system will not follow the prescribed motion. Two techniques were proposed in Chapter 4 for alleviating this problem. This section presents a third technique to do this, which places less restrictions on the allowable motions.

Rather than imposing a hard limit on the amount by which the solution can change from one instant to the next, it is more desirable to impose a penalty on large changes in the solution. This can be accomplished by modifying the objective function given in eq.(6.14a) as follows:

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + \frac{1}{2}(\mathbf{x}-\mathbf{x}^-)^T\widehat{\mathbf{W}}(\mathbf{x}-\mathbf{x}^-) \tag{6.58}$$

where $\widehat{\mathbf{W}}$ is a positive-semidefinite weighting matrix, and $\mathbf{x}^-$ was previously defined as the contact wrench solution vector at the previous instant. The new term in the objective function will tend to minimize the difference between the solution at the present time step and that at the previous time step, thus smoothing the time history of the contact wrench. This new objective function can be expanded as follows

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\widehat{\mathbf{W}}\mathbf{x} - (\mathbf{x}^-)^T\widehat{\mathbf{W}}\mathbf{x} + \frac{1}{2}(\mathbf{x}^-)^T\widehat{\mathbf{W}}\mathbf{x}^- \tag{6.59}$$

At the present instant, $\mathbf{x}^-$ is known and fixed. Therefore, the last term in eq.(6.59) is constant, and can be dropped from the objective function without affecting the solution. The objective function can be rewritten in the same form as eq.(6.14a), namely,

$$g(\mathbf{x}) = \tilde{\mathbf{c}}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\widetilde{\mathbf{W}}\mathbf{x} \tag{6.60a}$$

where

$$\widetilde{\mathbf{W}} = \mathbf{W} + \widehat{\mathbf{W}}, \qquad \tilde{\mathbf{c}} = \mathbf{c} - \widehat{\mathbf{W}}\mathbf{x}^- \tag{6.60b}$$

The formulation given by eqs.(6.60a) and (6.60b) allows us to smooth discontinuities in the variables contained in vector $\mathbf{x}$. If this vector does not include the actuator torques, e.g., if the method of §4.2.3 was used to formulate the dynamics equations, we can specify the following objective function to smooth these:

$$h(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + \frac{1}{2}(\boldsymbol{\tau} - \boldsymbol{\tau}^-)^T\widehat{\mathbf{W}}(\boldsymbol{\tau} - \boldsymbol{\tau}^-) \tag{6.61}$$

where $\boldsymbol{\tau}$ denotes a vector composed of *all* the actuator torques in the system, as defined in eq.(6.26b), while $\boldsymbol{\tau}^-$ is the same vector at the previous instant.

Substituting eq.(6.27*a*) into eq.(6.61) and dropping all terms which are constant, we obtain an equivalent objective function of the form of eq.(6.60a) with

$$\widetilde{\mathbf{W}} = \mathbf{W} + \mathbf{J}\widehat{\mathbf{W}}\mathbf{J}^T, \qquad \tilde{\mathbf{c}} = \mathbf{c} + \mathbf{J}\widehat{\mathbf{W}}(\boldsymbol{\tau}' - \boldsymbol{\tau}^-) \tag{6.62}$$

Since eq.(6.60a) is formally identical to eq.(6.14a), the optimization problem is left unchanged in principle and the same solution technique can be used to solve the

problem. As well, since all matrices in the new problem are of the same dimension as the original problem, we can expect that there will be no increase in the CPU time required to find a solution. Therefore, this technique also fits well into the existing framework of the optimization problem as it only requires the alteration of certain weighting parameters in the objective function. Furthermore, this technique does not have the disadvantage of the one proposed in §4.4.5, i.e., having to choose Δ_i or $\Delta_{i,j}$, perhaps iteratively.

Since the intent of the proposed technique is to smooth the force time histories *upon changes in topology*, it is desirable to ensure that the force solution is not affected when the topology does not change. This can be accomplished by setting $\widehat{\mathbf{W}} \neq \mathbf{0}$ only when changes in topology are expected, and ramping it linearly down to $\widehat{\mathbf{W}} = \mathbf{0}$ within some time T following the change in topology.

6.2.8.1 Numerical Example—A Four-Legged Walking Machine

This numerical example is intended to show that *a)* discontinuities will occur in the solution produced by eqs.(6.14a) to (6.14c), upon changes in the topology of the system, even when the motion is continuous, and *b)* the techniques presented in §4.4.5 and 6.2.8 can be used to smooth these discontinuities. The four-legged walking machine shown in Figure 6.14(a) is supported by three of its legs while its centre of mass moves forward as shown in Figure 6.15; its fourth leg is lowered to contact the ground at $t = 0.5$ s *with zero contact velocity*. The inertia of the legs is assumed to be negligible, allowing us to write the dynamical equations for the body of the machine as follows:

$$\mathbf{x} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \\ \mathbf{f}_4 \end{bmatrix}, \qquad \mathbf{b}_1 = \begin{bmatrix} m_o(\mathbf{a}_o + \mathbf{g}) \\ \mathbf{I}_o\dot{\boldsymbol{\omega}}_o + \boldsymbol{\omega}_o \times \mathbf{I}_o\boldsymbol{\omega}_o \end{bmatrix} \tag{6.63}$$

where $\mathbf{f}_i$ is the 3-dimensional vector of foot contact forces, if we assume that moments cannot be generated at the foot/ground contact. Furthermore, vectors $\mathbf{a}_o$ and $\boldsymbol{\omega}_o$ represent the acceleration of the centre of mass and the rotational velocity of the body, respectively.

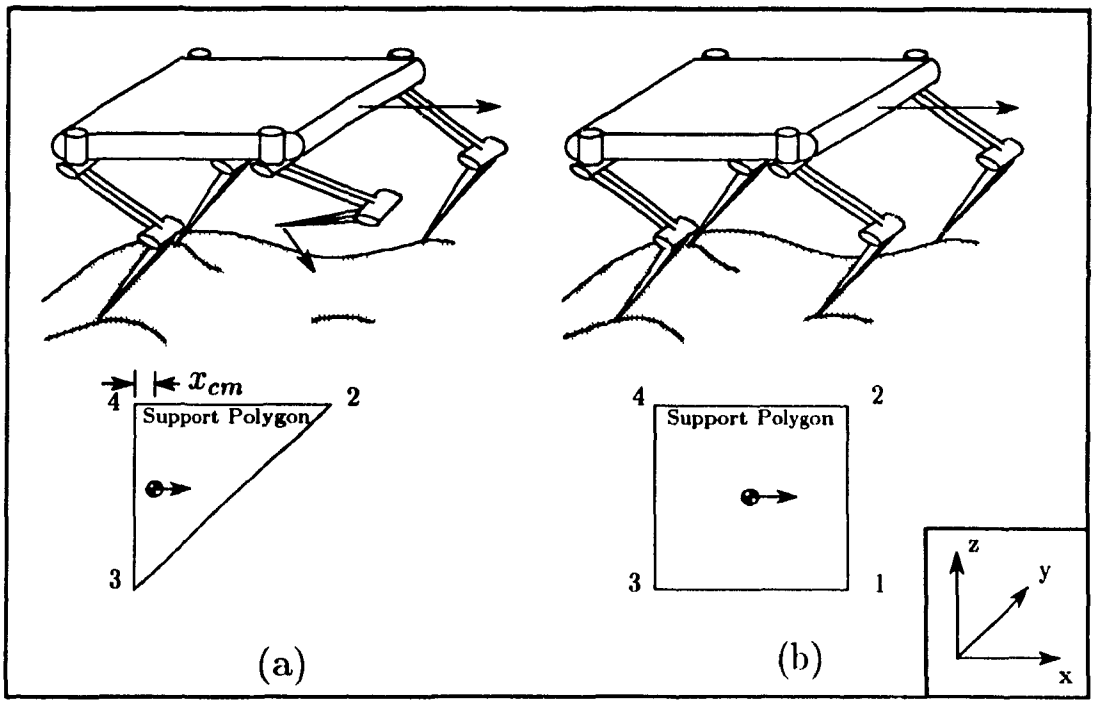

Figure 6.14 *Four-Legged Walking Machine*

Therefore, matrix $\mathbf{A}_1$ will undergo a discontinuous change at $t = 0.5$ s as follows

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{C}_2 & \mathbf{C}_3 & \mathbf{C}_4 \end{bmatrix}$$

$$\Downarrow \tag{6.64}$$

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{C}_1 & \mathbf{C}_2 & \mathbf{C}_3 & \mathbf{C}_4 \end{bmatrix}$$

where $\mathbf{0}$ denotes the 3×3 zero matrix, $\mathbf{1}$ denotes the 3×3 identity matrix, matrices $\mathbf{C}_i$ are defined by eq.(4.8c) and the $\mathbf{c}_i$ vectors are defined as follows:

$$\mathbf{c}_1 = \begin{bmatrix} 1.5 - x_{cm} \\ -0.5 \\ -0.7 \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} 1.5 - x_{cm} \\ 0.5 \\ -1.0 \end{bmatrix}, \quad \mathbf{c}_3 = \begin{bmatrix} -x_{cm} \\ -0.5 \\ -0.6 \end{bmatrix}, \quad \mathbf{c}_4 = \begin{bmatrix} -x_{cm} \\ 0.5 \\ -0.7 \end{bmatrix} \tag{6.65}$$

The objective function appearing in eqs.(6.60a) and (6.60b) was minimized with $\mathbf{c} = \mathbf{0}$, $\mathbf{W} = \mathbf{1}$ and $\widehat{\mathbf{W}} = \rho\mathbf{1}$, i.e., placing a weight of ρ on the continuity of the contact forces. Contact constraints are imposed on the system to ensure non-negativity of the normal contact forces between the feet and the ground, and foot friction forces within the friction pyramid with a coefficient of friction $\mu = 0.5$. Furthermore, the inequality

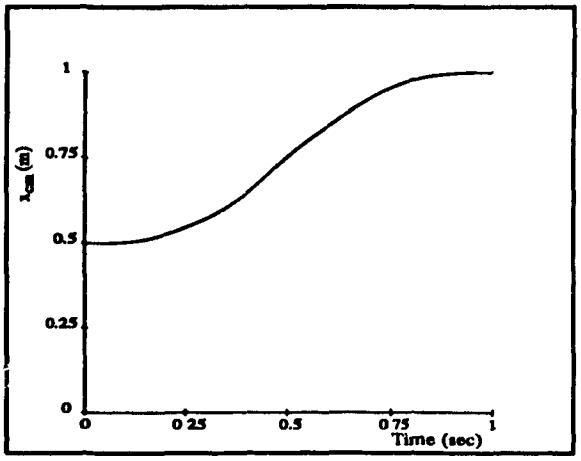

Figure 6.15 *X-position of the Centre of Mass*

constraints given by eq.(4.58c) are included to limit all contact force discontinuities to $\Delta_1 = \ldots = \Delta_q = \Delta$ Newtons per time step of 10 ms.

Three cases were investigated: *a)* no effort is made to smooth the solution to the optimization problem ($\rho = 0, \Delta = \infty$); *b)* inequality constraints are used to limit the rate of change of each contact force to 20 N/step ($\rho = 0$, $\Delta = 20$); and *c)* a penalty is imposed on discontinuities in the contact forces through the objective function ($\rho = 10$, $T = 0.5$ s, $\Delta = \infty$). Figure 6.16 shows the resulting norm of the contact forces at each of the four feet (all forces at foot #1 are zero until $t = 0.5$ s). Both of the proposed techniques result in considerably smoother force time histories of the foot contact forces.

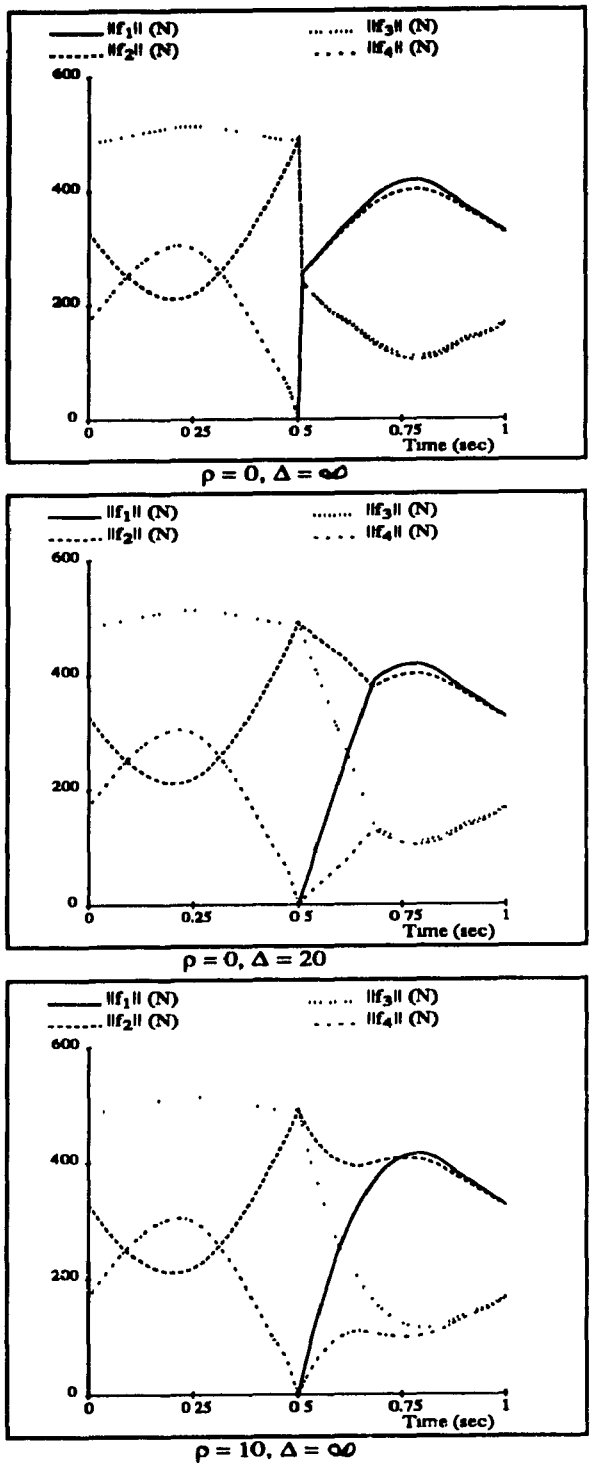

Figure 6.16 *Foot/Ground Contact Forces*

Chapter 7

Further Applications of Redundant Actuation

Until now, redundant actuation has been treated as being intrinsically coupled to mechanical systems with time-varying topology. The implicit justification for this has been that systems whose topology varies are always redundantly actuated during some part of their task, while systems whose topology does not change are very unlikely to be redundantly actuated. However, redundant actuation can also be applied to fixed-topology systems in order to smooth or homogenize the force distribution within them, as will be shown in this chapter.

Perhaps the earliest work dealing with redundantly actuated fixed-topology systems was that of Williams and Seireg (1979) who observed that the human musculoskeletal structure was redundantly actuated and allowed many inverse dynamics solutions. Nakamura (1988a) was among the first to suggest redundant actuation in the context of mechanical linkages with fixed topology when he proposed a two-degree-of-freedom planar five-bar linkage driven by three actuators as a subassembly of the finger of a mechanical hand. Hayward (1988), in designing a three-degree-of-freedom parallel wrist, found that the use of one redundant actuator would allow the wrist to avoid kinematic singularities. Soon thereafter, Nakamura and Ropponen (1989) proposed a three-degree-of-freedom spatial linkage driven by four actuators, while Gardner *et al.* (1989) suggested

a two-degree-of-freedom parallel robotic manipulator driven by four actuators. Other than these few works, no proponents of redundant actuation of fixed-topology linkages are apparent.

In most existing works, redundant actuation of fixed-topology linkages is put forward as a way of reducing the actuator and constraint wrenches acting in the system—just as this was done in the case of systems with time-varying topology. However, once the force distribution problem of a redundantly-actuated system is formulated as an optimization problem, the objective functions which may be minimized are virtually unlimited. This chapter presents two less conventional applications of redundant actuation in an attempt to show the range of possibilities which exist. In §7.1, redundant actuation is used to achieve the full dynamic balancing of linkages—i.e., linkages which exert no dynamical forces at their supports. In §7.2, it is shown that redundant actuation can be used to reduce the effects of shocks in mechanisms.

7.1 Dynamic Balancing of Linkages

The dynamic balancing of linkages is a classical problem in the theory of machines and mechanisms, and is of practical importance wherever linkages must run at high speeds (Berkof and Lowen, 1969; Berkof, 1973; Bagci, 1979; Kochev, 1988; Feng, 1989). For example, the four-bar linkage shown in Figure 7.1(a), when in motion, will exert forces and moments on the frame on which it is mounted. The free-body diagram of each link of this linkage is drawn as shown in Figure 7.1(b), where we denote the net force which acts on the frame as $\mathbf{f}_1 - \mathbf{f}_4$. Berkof and Lowen (1969) have shown that a four-bar linkage can be constructed with $\mathbf{f}_1 - \mathbf{f}_4 = \mathbf{0}$ by keeping the centre of mass of the linkage motionless. They accomplished this by writing an equation for the position of the centre of mass and setting all the coefficients of the time-dependent terms to zero. The conditions which must be satisfied in order to achieve this are:

$$m_1||\mathbf{c}_{11}|| = m_2||\mathbf{c}_{22}||\frac{l_1}{l_2} \tag{7.1a}$$

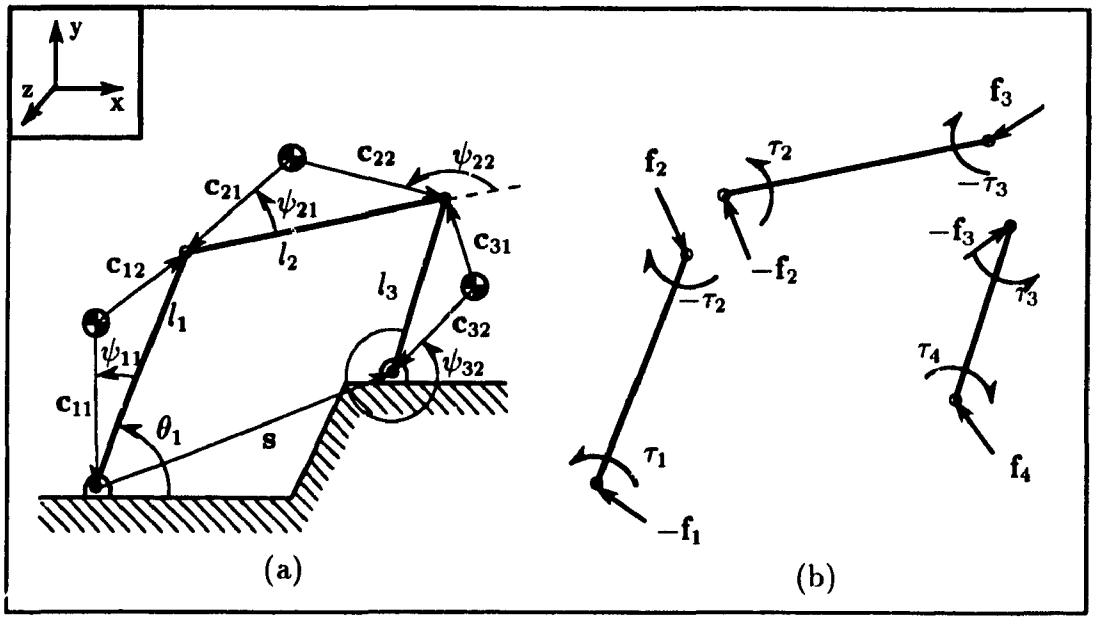

Figure 7.1 *Four-Bar Linkage*

$$m_3||\mathbf{c}_{32}|| = m_2||\mathbf{c}_{21}||\frac{l_3}{l_2} \tag{7.1b}$$

$$\psi_{11} = \psi_{22}, \qquad \psi_{32} = \psi_{21} + \pi \tag{7.1c}$$

When these conditions are met, the force $\mathbf{f}_1$ is equal and opposite to $\mathbf{f}_4$ at all times, so that, when added, the two forces cancel each other. However, as noted by Ku in the commentary on the work of Berkof and Lowen (1969), certain problems remain unresolved with this approach:

- Although the net force $\mathbf{f}_1 - \mathbf{f}_4$ is zero for the balanced linkage, the individual cyclic forces $\mathbf{f}_1$ and $\mathbf{f}_4$ still exist and may be large. Thus, each of the linkage supports at the frame is subjected to a cyclic shaking force.

- A cyclic rocking moment acting on the frame due to the equal but opposite forces acting at the two linkage supports (i.e., $\mathbf{s} \times \mathbf{f}_1$) still exists and may be large.

The latter of these two criticisms was addressed by Berkof (1973) for the particular case of 'in-line linkages'—i.e., where the centre of mass of each link lies along

the line joining the link's two pivots. In that work, other members such as flywheels and pendulums were added to the linkage to cancel the inertia of the primary links. The purpose of this section is to show that redundant actuation can be used to correct the two above-mentioned problems without limiting the analysis to particular linkages.

7.1.1 Dynamics

In order to treat the four-bar linkage problem, we first write the dynamics equations which must be satisfied by any set of actuator torques which is used to control the system. We write the equations assuming that there is an actuator installed at each joint. If a given joint is not actuated, its corresponding actuator torque need only be set to zero. These equations take on the form

$$-\mathbf{f}_1 + \mathbf{f}_2 = m_1\mathbf{a}_1 \tag{7.2a}$$

$$-\mathbf{f}_2 + \mathbf{f}_3 = m_2\mathbf{a}_2 \tag{7.2b}$$

$$-\mathbf{f}_3 + \mathbf{f}_4 = m_3\mathbf{a}_3 \tag{7.2c}$$

$$\tau_1 - \tau_2 + \mathbf{c}_{11}^T\mathbf{E}\mathbf{f}_1 - \mathbf{c}_{12}^T\mathbf{E}\mathbf{f}_2 = I_1\dot{\omega}_1 \tag{7.2d}$$

$$\tau_2 - \tau_3 + \mathbf{c}_{21}^T\mathbf{E}\mathbf{f}_2 - \mathbf{c}_{22}^T\mathbf{E}\mathbf{f}_3 = I_2\dot{\omega}_2 \tag{7.2e}$$

$$\tau_3 - \tau_4 + \mathbf{c}_{31}^T\mathbf{E}\mathbf{f}_3 - \mathbf{c}_{32}^T\mathbf{E}\mathbf{f}_4 = I_3\dot{\omega}_3 \tag{7.2f}$$

where the moment equations are written about the respective centre of mass of each link, and I_i is the moment of inertia of link i about its centre of mass. Matrix $\mathbf{E}$ is defined as

$$\mathbf{E} \equiv \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{7.2g}$$

Equations (7.2a) to (7.2f) can be written more compactly in the form

$$\mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \tag{7.3a}$$

where the 9×12 matrix $\mathbf{A}_1$ is defined as

$$\mathbf{A}_1 = \begin{bmatrix} -\mathbf{1} & \mathbf{1} & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_2 & -\mathbf{1} & \mathbf{1} & \mathbf{0}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_2 & \mathbf{0}_2 & -\mathbf{1} & \mathbf{1} & \mathbf{0}_{2\times 4} \\ \mathbf{c}_{11}^T\mathbf{E} & -\mathbf{c}_{12}^T\mathbf{E} & \mathbf{0}^T & \mathbf{0}^T & \mathbf{d}_1^T \\ \mathbf{0}^T & \mathbf{c}_{21}^T\mathbf{E} & -\mathbf{c}_{22}^T\mathbf{E} & \mathbf{0}^T & \mathbf{d}_2^T \\ \mathbf{0}^T & \mathbf{0}^T & \mathbf{c}_{31}^T\mathbf{E} & -\mathbf{c}_{32}^T\mathbf{E} & \mathbf{d}_3^T \end{bmatrix} \tag{7.3b}$$

$\mathbf{1}$ represents the 2×2 identity matrix, $\mathbf{0}_2$ is the 2×2 zero matrix, $\mathbf{0}_{2\times 4}$ is the 2×4 zero matrix, $\mathbf{0}$ is the zero vector of dimension 2 and the other arrays are

$$\mathbf{d}_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{d}_2 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}, \qquad \mathbf{d}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} \tag{7.3c}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \\ \mathbf{f}_4 \\ \boldsymbol{\tau} \end{bmatrix}, \qquad \boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix}, \qquad \mathbf{b}_1 = \begin{bmatrix} m_1\mathbf{a}_1 \\ m_2\mathbf{a}_2 \\ m_3\mathbf{a}_3 \\ I_1\dot{\omega}_1 \\ I_2\dot{\omega}_2 \\ I_3\dot{\omega}_3 \end{bmatrix} \tag{7.3d}$$

If only one actuator is active (e.g., $\tau_1 \neq 0$ and $\tau_2 = \tau_3 = \tau_4 = 0$), three columns can be removed from matrix $\mathbf{A}_1$, and the system of equations has a unique solution once the motion is prescribed. However, if all four actuators are active, the system of equations is underdetermined. As in the case of systems with time-varying topology, optimization techniques may be applied to find the optimum solution to the underdetermined case.

7.1.2 Forces Acting on the Frame

In this section, we start by showing that the net force on the frame $\mathbf{f}_1 - \mathbf{f}_4$ can only be changed by linkage design and *not* through redundant actuation. Adding eqs.(7.2a), (7.2b) and (7.2c), we obtain:

$$-\mathbf{f}_1 + \mathbf{f}_4 = m_1\mathbf{a}_1 + m_2\mathbf{a}_2 + m_3\mathbf{a}_3 \tag{7.4}$$

It is apparent that the left-hand side of eq.(7.4) is the net external force acting

on the three moving links of the four-bar linkage and so we can write

$$-\mathbf{f}_1 + \mathbf{f}_4 = M\mathbf{a}_C \tag{7.5}$$

where $\mathbf{a}_C$ is the acceleration of the centre of mass of the three moving links, while $M = m_1 + m_2 + m_3$. It is noted that the right-hand-side of eq.(7.5) is determined strictly by the motion of the linkage and the masses of the moving links and is independent of the actuator torques τ_1 to τ_4. Thus, the net force acting on the frame cannot be altered by redundant actuation. This implies that redundant actuation alone cannot be used to make $\mathbf{f}_1 = \mathbf{f}_4 = \mathbf{0}$.

However, if we design a linkage to satisfy the conditions given by eqs.(7.1a) through (7.1c), we find that $\mathbf{a}_C = \mathbf{0}$ and therefore $\mathbf{f}_1 - \mathbf{f}_4 = \mathbf{0}$. We can now use redundant actuation to cancel both $\mathbf{f}_1$ and $\mathbf{f}_4$. This can be done by minimizing the objective function f defined as

$$f = ||\mathbf{f}_1||^2 + ||\mathbf{f}_4||^2 \tag{7.6}$$

under the equality constraints given by eq.(7.3a), and checking whether a minimum exists which renders f zero, which would obviously mean that $\mathbf{f}_1 = \mathbf{f}_4 = \mathbf{0}$.

We can verify the existence of this minimum by setting $\mathbf{f}_1 = \mathbf{0}$ in eqs.(7.2a) to (7.2c). Once this is done, the other forces in the system can be found uniquely, from which $\mathbf{f}_4 = \mathbf{0}$, and we are left with a smaller underdetermined system of three equations, eqs.(7.2d) to (7.2f), in four unknowns, τ_1 to τ_4. This underdeterminacy could be resolved by deactivating one of the actuators, thereby setting the corresponding torque to zero, or by finding an optimum solution to the reduced system.

It is therefore apparent that we require at least three actuators, two of which would be redundant, to ensure that the forces acting on the frame, $\mathbf{f}_1$ and -$\mathbf{f}_4$, will be zero. As well, since these forces have been reduced to zero, the shaking moment due to the reaction forces has also been eliminated, thereby solving the two criticisms leveled at Berkof and Lowen's balancing method.

7.1.3 Moments Acting on the Frame

Berkof (1973) showed that an in-line four-bar linkage, which was defined in 7.1, could be fully force- and moment-balanced by adding flywheels to it. However, it is physically impossible to cancel the reaction moments of more general four-bar linkages. The net moment exerted by a force-balanced linkage on its frame, which is the same irrespective of the reference point used to calculate it, is $-\tau_1 + \tau_4 + \mathbf{s}^T\mathbf{E}\mathbf{f}_1$, where $\mathbf{s}$ is indicated in Figure 7.1(a). Just as it was impossible to alter the net force acting on the frame, the net moment acting on the frame cannot be altered through redundant actuation This is again shown by manipulating the dynamics equations. We start by adding eqs.(7.2d), (7.2e) and (7.2f), to obtain:

$$\tau_1 - \tau_4 + \mathbf{c}_{11}^T\mathbf{E}\mathbf{f}_1 + (\mathbf{c}_{21} - \mathbf{c}_{12})^T\mathbf{E}\mathbf{f}_2 + (\mathbf{c}_{31} - \mathbf{c}_{22})^T\mathbf{E}\mathbf{f}_3 - \mathbf{c}_{32}^T\mathbf{E}\mathbf{f}_4 = I_1\dot{\omega}_1 + I_2\dot{\omega}_2 + I_3\dot{\omega}_3 \quad (7.7)$$

For a balanced linkage, we substitute $\mathbf{f}_1 = \mathbf{f}_4$ into eqs.(7.2a) and (7.2c) to obtain:

$$\mathbf{f}_2 = \mathbf{f}_1 + m_1\mathbf{a}_1, \qquad \mathbf{f}_3 = \mathbf{f}_1 - m_3\mathbf{a}_3 \quad (7.8)$$

Substituting eq.(7.8) into eq.(7.7) and simplifying yields

$$\tau_1 - \tau_4 + (\mathbf{c}_{11} - \mathbf{c}_{12} + \mathbf{c}_{21} - \mathbf{c}_{22} + \mathbf{c}_{31} - \mathbf{c}_{32})^T\mathbf{E}\mathbf{f}_1 = I_1\dot{\omega}_1 + I_2\dot{\omega}_2 + I_3\dot{\omega}_3 + m_1(\mathbf{c}_{12} - \mathbf{c}_{21})^T\mathbf{E}\mathbf{a}_1 + m_3(\mathbf{c}_{31} - \mathbf{c}_{22})^T\mathbf{E}\mathbf{a}_3 \quad (7.9)$$

Noting that the term in brackets on the left-hand-side of the equation is just the vector directed from the right-hand support to the left-hand support, we can write

$$\tau_1 - \tau_4 - \mathbf{s}^T\mathbf{E}\mathbf{f}_1 = I_1\dot{\omega}_1 + I_2\dot{\omega}_2 + I_3\dot{\omega}_3 + m_1(\mathbf{c}_{12} - \mathbf{c}_{21})^T\mathbf{E}\mathbf{a}_1 + m_3(\mathbf{c}_{31} - \mathbf{c}_{22})^T\mathbf{E}\mathbf{a}_3 \quad (7.10)$$

The left-hand-side of the equation is nothing but the net moment acting on the moving links while the right-hand-side is, once again, purely a function of the motion of those links and their inertial properties. Thus, it is apparent that the net moment on the frame is determined strictly by the prescribed motion and the inertial properties of the linkage.

We have already shown that through proper design *and* redundant actuation, we can produce a linkage for which $\mathbf{f}_1 = \mathbf{f}_4 = \mathbf{0}$. Furthermore, we have shown that this

can be accomplished with only two redundant actuators. If we choose the fourth joint to be unactuated, i.e., $\tau_4 = 0$, we are left with

$$\tau_1 = I_1\dot{\omega}_1 + I_2\dot{\omega}_2 + I_3\dot{\omega}_3 + m_1(\mathbf{c}_{12} - \mathbf{c}_{21})^T\mathbf{E}\mathbf{a}_1 + m_3(\mathbf{c}_{31} - \mathbf{c}_{22})^T\mathbf{E}\mathbf{a}_3 \qquad (7.11)$$

and $-\tau_1$ will be the only reaction acting on the frame.

7.1.4 Numerical Example

A sample linkage will now be introduced in order to clarify the techniques discussed above. The linkage chosen is that treated by Angeles and Lee (1989), the dimensions of which are shown in Table 7.1. This is an unbalanced linkage which therefore exerts a non-zero net force and moment on its support frame. The input link is rotated through the *cycloidal* maneuver given by

$$\theta_1(t)[\text{rad}] = 5\pi t - \frac{1}{4}\sin 20\pi t, \qquad 0 \leq t \leq 0.1\text{s} \qquad (7.12)$$

This maneuver, which has zero initial and final velocity and acceleration, was taken from Angeles and Lee (1989) and is shown in Figure 7.2. When only one actuator is used to drive the linkage, the resulting inverse dynamics equations are determinate. Figure 7.3(a) shows the magnitude of the forces acting in the linkage when only τ_1 is used to drive the linkage. The corresponding actuator torque required is shown in Figure 7.3(b).

When all four actuators are active, the system becomes redundantly actuated and the objective function $f = ||\mathbf{f}_1||^2 + ||\mathbf{f}_4||^2$ can be minimized. The resulting magnitude of the forces in the linkage are shown in Figure 7.4(a), while Figure 7.4(b) shows the required actuator torques. The magnitudes of $\mathbf{f}_1$ and $\mathbf{f}_4$ are equal, but they act in differing directions, thereby resulting in a net force on the support frame which is identical to that when only one actuator was driven. It is apparent that the forces $\mathbf{f}_1$ to $\mathbf{f}_4$ have been reduced and homogenized by the redundant actuation but they have not been reduced to zero.

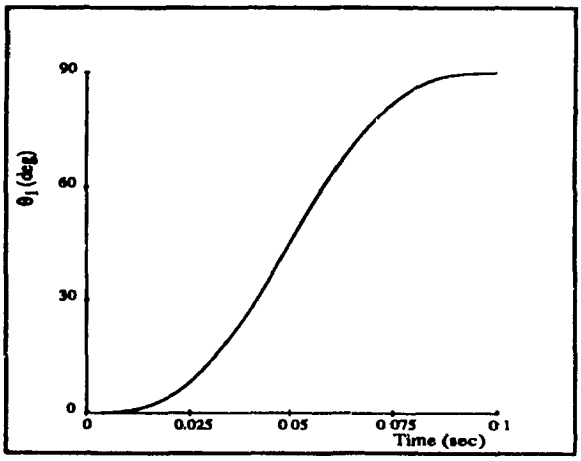

Figure 7.2 *Prescribed Motion of the Input Link*

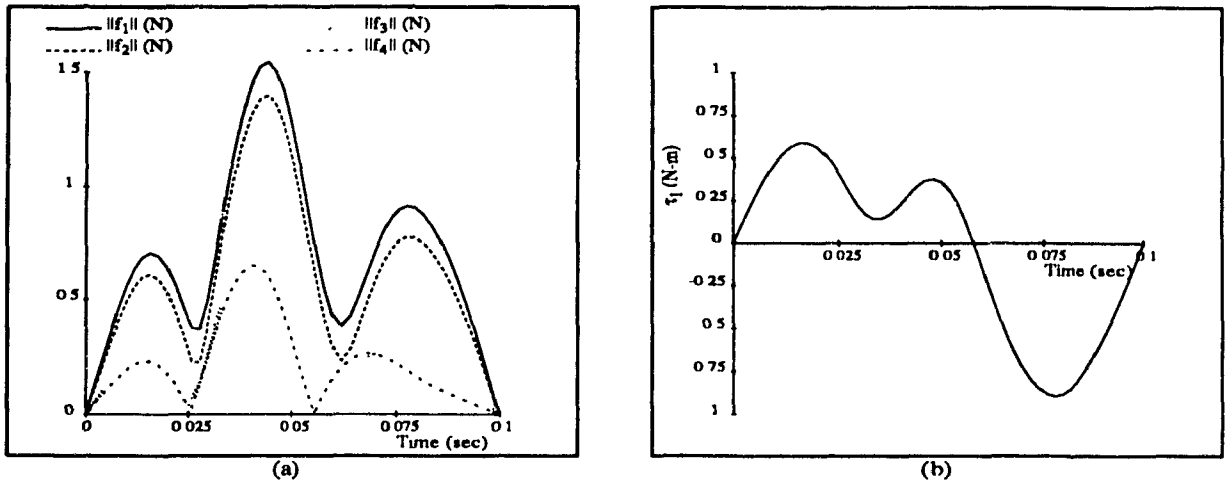

Figure 7.3 *Unbalanced Linkage with One Actuator*

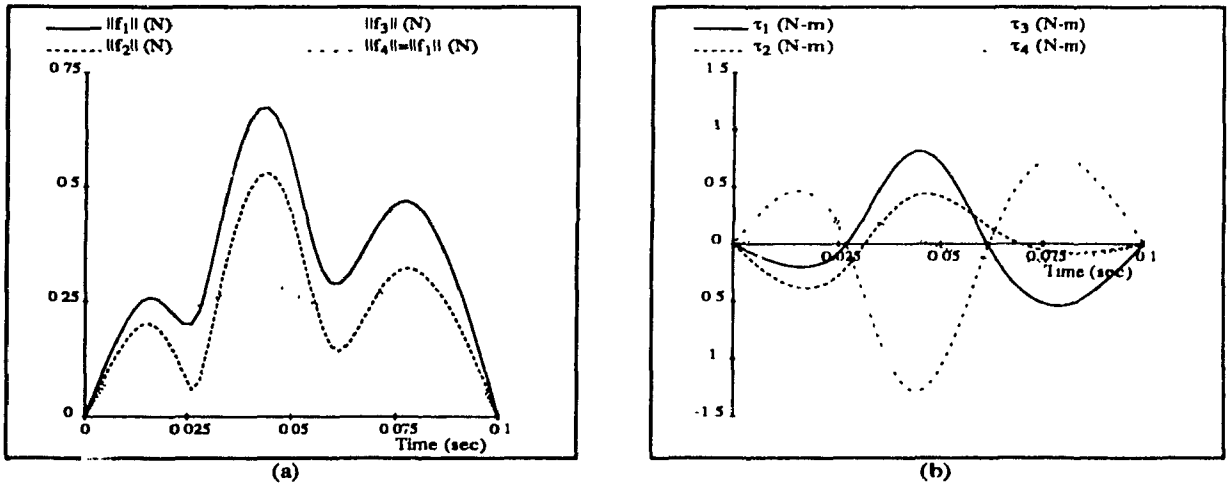

Figure 7.4 *Unbalanced Linkage with Four Actuators*

Parameter	Unbalanced Linkage	Balanced Linkage	Parameter	Unbalanced Linkage	Balanced Linkage
l_1 (m)	1	1	ψ_{11} (deg)	0	45
l_2 (m)	$2\sqrt{2}$	$2\sqrt{2}$	ψ_{21} (deg)	0	10
l_3 (m)	2	2	ψ_{22} (deg)	180	45
s^x (m)	3	3	ψ_{31} (deg)	0	190
s^y (m)	0	0	$\|\|\mathbf{c}_{11}\|\|$ (m)	0.5	0.656
m_1 (kg)	3×10^{-4}	3×10^{-4}	$\|\|\mathbf{c}_{12}\|\|$ (m)	0.5	0.709
m_2 (kg)	6.5×10^{-4}	6.5×10^{-4}	$\|\|\mathbf{c}_{21}\|\|$ (m)	1.414	3.487
m_3 (kg)	5×10^{-4}	5×10^{-4}	$\|\|\mathbf{c}_{22}\|\|$ (m)	1.414	0.856
I_1 (kg-m^2)	0.75×10^{-4}	0.75×10^{-4}	$\|\|\mathbf{c}_{31}\|\|$ (m)	1	5.186
I_2 (kg-m^2)	4.16×10^{-4}	4.16×10^{-4}	$\|\|\mathbf{c}_{32}\|\|$ (m)	1	3.205
I_3 (kg-m^2)	5×10^{-4}	5×10^{-4}			

Table 7.1 *Four-Bar Linkage Parameters*

A balanced linkage with the same dimensions but with the location of its link centres of mass altered as shown in Table 7.1 is now introduced. The same maneuver is used and the results obtained with only τ_1 active are shown in Figure 7.5. In this case, the magnitudes of $\mathbf{f}_1$ and $-\mathbf{f}_4$ are equal but they act in opposite directions so that $\mathbf{f}_1 - \mathbf{f}_4 = \mathbf{0}$. Figure 7.6 shows the forces and actuator torques acting in the linkage when τ_1, τ_2 and τ_3 are active and $f = ||\mathbf{f}_1||^2 + ||\mathbf{f}_4||^2$ is minimized. We find that $\mathbf{f}_1 = \mathbf{f}_4 = \mathbf{0}$, and the only reaction acting on the frame is $-\tau_1$. With all four actuators active and the same objective function being minimized, the results shown in Figure 7.7 are obtained. Once again, $\mathbf{f}_1$ and $\mathbf{f}_4$ have been reduced to zero, this time with the net reaction on the frame being $\tau_4 - \tau_1$. Obviously, with all four joints actuated, the individual torques are smaller than with only three actuated joints.

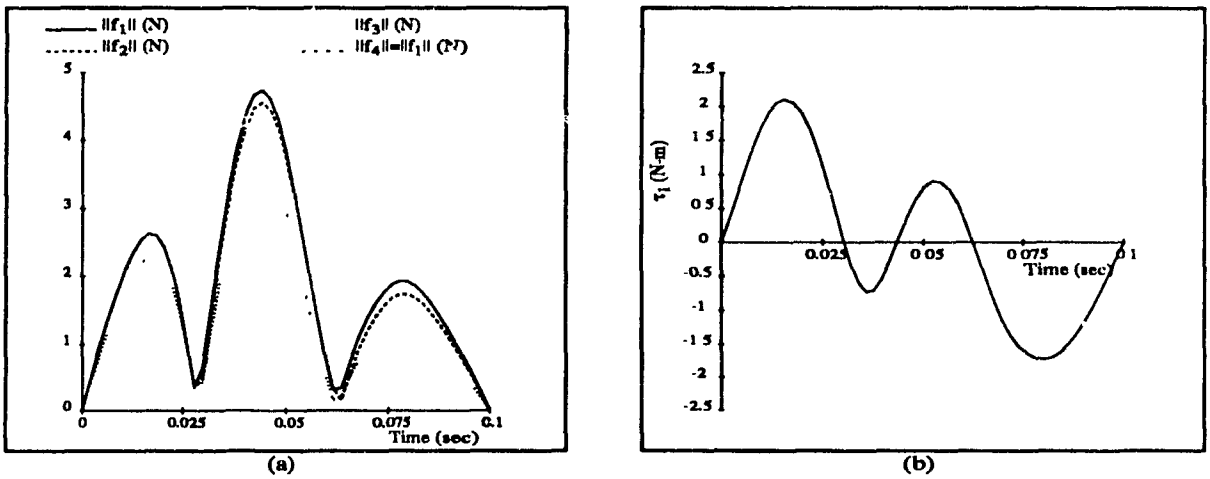

Figure 7.5 *Balanced Linkage with One Actuator*

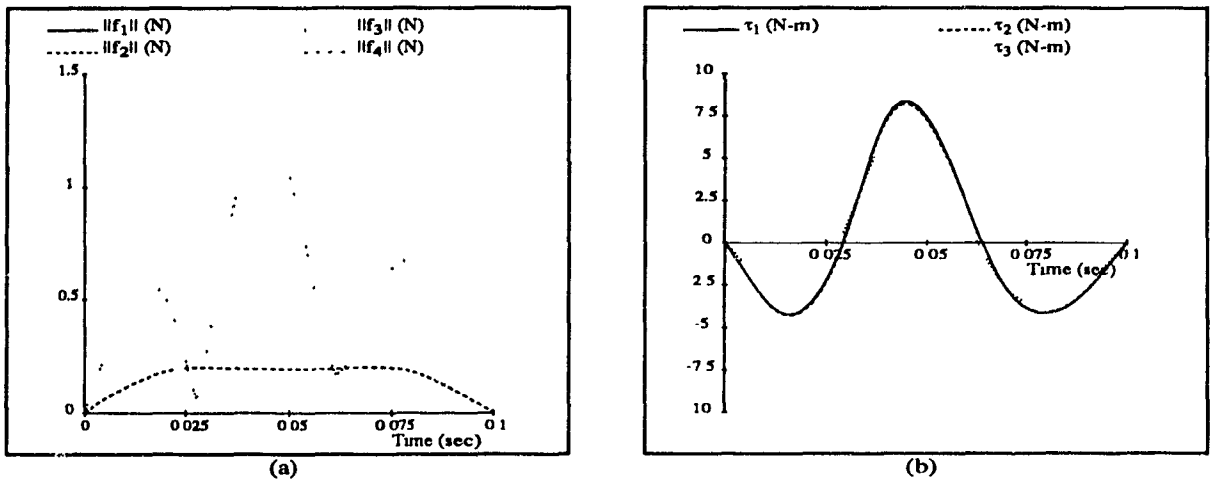

Figure 7.6 *Balanced Linkage with Three Actuators*

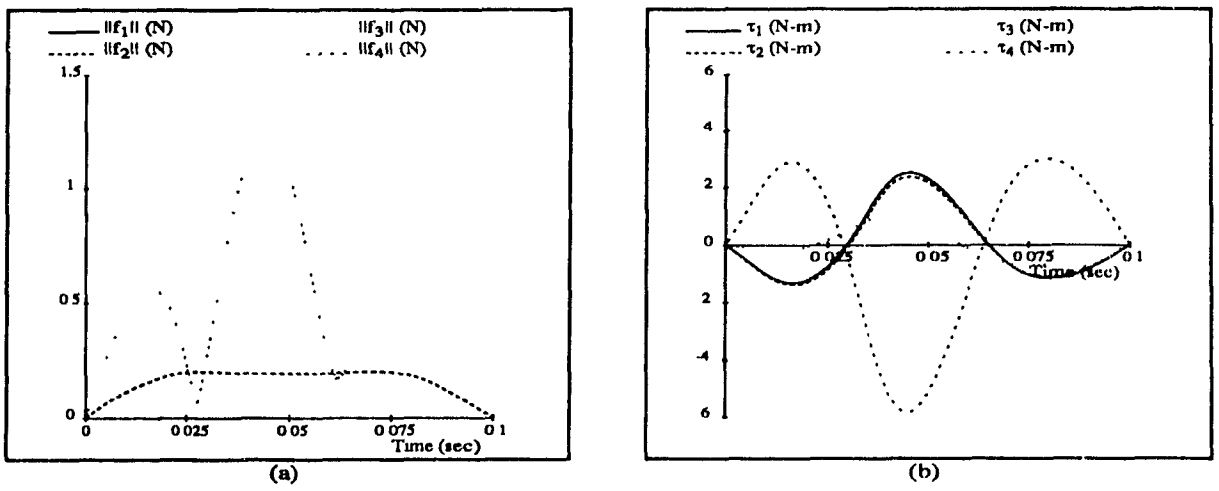

Figure 7.7 *Balanced Linkage with Four Actuators*

7.2 Smoothing Impact Shocks

In Chapter 4, the dynamics equations of redundantly actuated robotic systems were studied with some emphasis on the behavior of these equations upon changes in the topology of the system. It was found that the equations of motion could be written as

$$\mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \tag{7.13}$$

where $\mathbf{b}_1$ could be interpreted as representing the *motion* of the system, while $\mathbf{A}_1$ represented its kinematic structure and configuration. It was shown that, when the topology of the system changed, the entries of matrix $\mathbf{A}_1$ would change discontinuously, and a number of techniques were proposed in Chapters 4 and 6 to smooth the ensuing discontinuities in the solution to the optimization problem. Thus, the work of those chapters was concerned primarily with the reduction of discontinuities *due to changes in the topology of the system* and it was implicitly assumed that vector $\mathbf{b}_1$ would not change discontinuously during the task. Physically, this corresponds to the situation where, for example, the foot of a walking machine contacts the ground *with zero relative velocity*, so that the motion is continuous and there is no shock. It is important to realize that shocks are not *intrinsically* tied to changes in topology, although they often occur together. Here, it is emphasized that the two should be viewed as distinct phenomena.

But now, what of the second phenomenon which has not yet been addressed and may be of interest—the smoothing of solution discontinuities due to shocks? A shock, or mild impact, consists of finite forces applied abruptly to the system, either between two of its constituent bodies or to a single body by an external influence. A shock results in a discontinuous change in vector $\mathbf{b}_1$ of eq.(7.13), which *may* be accompanied by a discontinuous change in $\mathbf{A}_1$ if the topology of the system changes at that instant. In this section, we are interested in determining whether the smoothing techniques presented in §4.4.5 and 6.2.8 could be used to reduce discontinuities due to shocks, just as they were used to smooth discontinuities due to changes in topology.

7.2.1 Reducing the Effects of Shocks

The underlying idea is as follows: given that there are many solutions to the force distribution problem in a redundantly actuated system, it is desired to find the solution which minimizes solution discontinuities. A shock is reflected in vector $\mathbf{b}_1$ in one of two ways: as a jump discontinuity in the external wrench $\mathbf{w}_e = [\mathbf{n}_e^T \quad \mathbf{f}_e^T]^T$, or as a jump discontinuity in the motion variables in $\mathbf{b}_1$. The present work assumes that these jumps can be modeled, and that the time history of $\mathbf{w}_e$ and the motion of the system can be calculated and measured, even *during* the shock. The smoothing techniques developed previously can therefore be applied to cancel the shock forces as much as possible.

Shocks are similar to impacts in that they represent abrupt changes in the motion and forces applied to the system. However, whereas impacts normally involve the generation of infinitely large forces over infinitesimally short time intervals, the forces and time intervals of shocks are assumed to be finite. A large body of work exists regarding the modeling of impacts in general mechanical systems (e.g., Kane, 1968; Wittenburg, 1977; Lötstedt, 1984; Haug *et al.*, 1986, Pfeiffer, 1991). Some of this work has been adapted to robotic systems in the works of Zheng and Hemami (1985), Zheng (1987) and Wang and Mason (1987). Finally, the works of Johnson (1958), Parker and Paul (1985), Kahng and Amirouche (1987) and Youcef-Toumi and Gutz (1989) are better classified as dealing with shocks since make an effort to model and measure the finite forces and time duration of the shock.

Most of the above works apply energy methods to a model of two colliding bodies. They assume the motion, and hence, the kinetic energies of the two bodies, to be known immediately before the collision. At one instant during the collision, the two bodies have zero relative velocity and their kinetic energy has been completely transformed into strain energy. Finally, as the bodies recover their initial shapes, part of the strain energy is recovered as kinetic energy, while part is lost as heat, the relative proportion being referred to as the *coefficient of restitution*. Using certain assumptions about the

elastic properties of the colliding materials, i.e., the force produced as a function of the material's deformation, the maximum force between the two bodies can be estimated (Johnson, 1958).

In the present work, it is assumed that we are provided with an external force $\mathbf{f}_0$, which represents the shock force applied to the common pole. This shock force may either be due to a collision between the pole and the j-th serial chain, or between the pole and an external object. The shock force can be introduced into the dynamic analysis presented in Chapter 4 by modifying the equations of motion of the moving pole given by eqs.(4.7a) and (4.7b) to reflect the shock as follows

$$\sum_{i=1}^{p} \mathbf{f}_i = m_o(\mathbf{a}_o + \mathbf{g}) - \mathbf{f}_e - \mathbf{f}_0 \tag{7.14a}$$

$$\sum_{i=1}^{p} \mathbf{n}_i + \sum_{i=1}^{p} (\mathbf{c}_i \times \mathbf{f}_i) = \mathbf{I}_o\dot{\boldsymbol{\omega}}_o + \boldsymbol{\omega}_o \times \mathbf{I}_o\boldsymbol{\omega}_o - \mathbf{n}_e - \mathbf{c}_e \times \mathbf{f}_e - \mathbf{c}_0 \times \mathbf{f}_0 \tag{7.14b}$$

where $\mathbf{c}_0$ denotes a vector directed from the centroid of the pole to the point of collision.

As was shown in Chapter 4, these equations are underdetermined and the contact wrenches, $\mathbf{f}_i$ and $\mathbf{n}_i$, for $i = 1, \ldots, p$, can be chosen according to an optimality criterion. The goal is therefore to choose these contact wrenches to offset the effect of $\mathbf{f}_0$ as much as possible. The smoothing techniques presented in §4.4.5 and 6.2.8 are used to find this solution. The first of these imposes inequality constraints on the solution to ensure that the actuator torques at a given instant do not differ from those at the previous instant by more than a prescribed amount, while the second method alters the objective funtion to penalize large changes in the actuator torques. Once again, the latter approach is favored since it cannot cause the optimization problem to become infeasible.

7.2.2 Numerical Example

This example shows the behavior of the smoothing techniques presented in §4.4.5 and 6.2.8 for a shock during which *there is no change in topology*. This is done to ensure that the shock effects observed in the example are not confused by the effects of a

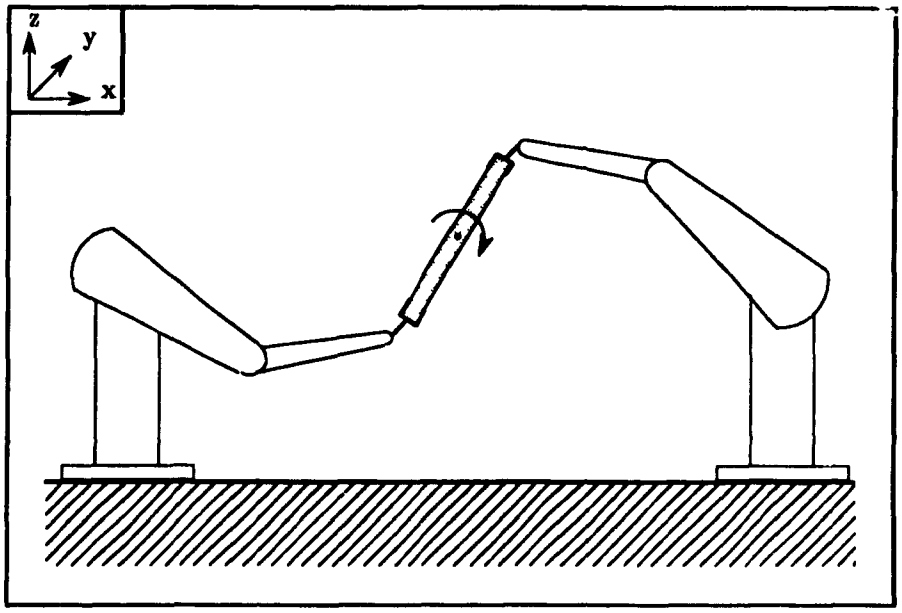

Figure 7.8 *Two Puma 560 Manipulators Rotating a Payload*

change in topology. The two Puma 560 robots shown in Figure 7.8 are used to perform the same maneuver as in §6.2.7.4. All relevant data regarding the manipulators and their task were given in that section.

However, at $t = 1.0$ s in the present example, an external force is imposed in the x-direction of $f_0^x = 100N$ at the centre of mass of the payload (i.e., $\mathbf{c}_0 = \mathbf{0}$), thereby representing a collision between it and the environment.

The objective function given by eqs.(6.60a) and (6.62) was minimized with $\mathbf{c} = \mathbf{0}$, $\mathbf{W} = \mathbf{1}$ and $\widehat{\mathbf{W}} = \rho\mathbf{1}$, i.e., placing a weight of ρ on the continuity of the actuator torques. The only inequality constraints included in this problem are those given by eq.(4.61) to limit all actuator torque discontinuities to $\Delta_{11} = \ldots = \Delta_{pr} = \Delta$ Nm per time step of 10 ms. Three cases were investigated: *a)* no effort is made to smooth the solution to the optimization problem ($\rho = 0$, $\Delta = \infty$); *b)* inequality constraints are used to limit the rate of change of each actuator torque to 11 Nm/step ($\rho = 0$, $\Delta = 11$); and *c)* a penalty is imposed on discontinuities in the actuator torques through the objective function ($\rho = 10$, $T = 1.0$ s, $\Delta = \infty$).

For case (*a*), the actuator torques for manipulators #1 and #2 are shown in Figures 7.9 and 7.10, respectively. Since the maneuver takes place in a plane, two of the wrist actuators in each manipulator are not active, as is apparent from the figures.

The results for case (*b*), where the discontinuities are smoothed using inequality constraints, are shown in Figures 7.11 and 7.12. The same two actuators which were inactive in case (*a*) remain inactive. However, the discontinuity in actuator # 3 is reduced, while that in actuators # 2 and 5 is increased.[1] In effect, the inequality constraints have distributed the discontinuity more evenly over the active actuators, thereby decreasing the largest discontinuity. This is desirable, assuming that all the actuators are equally able to withstand discontinuities. If they are not, the $\Delta_{i,j}$'s can be made more restrictive for the more sensitive actuators. It should be noted that, for the present example, with equal $\Delta_{i,j}$'s on all the actuators, a value of Δ lower than 11 will cause the problem to become infeasible.

The results for case (*c*), where the actuator torques are smoothed using a new objective function, are shown in Figures 7.13 and 7.14. This time, the two previously-inactive actuators become active. As well, the discontinuities in actuators # 1 and 3 are reduced, while those in actuators # 2, 4, 5 and 6 are increased. Once again, the largest discontinuity has been decreased by distributing the shock more evenly, this time over *all* the actuators. For the present example, a further increase in ρ, the weight on the actuator torque discontinuities, will not smooth the results any more, but will not cause the problem to become infeasible. The second method for reducing the discontinuities thus has two advantages over the first: it cannot cause the optimization problem to become infeasible and it distributes the discontinuities evenly over *all* the actuators, rather than just the active ones. Finally, if the actuators were not all equally able to withstand discontinuities, the weighting matrix $\widehat{\mathbf{W}}$ could be modified to penalize discontinuities in the more sensitive actuators more heavily.

[1] n.b., the plot scales are different for the various actuators

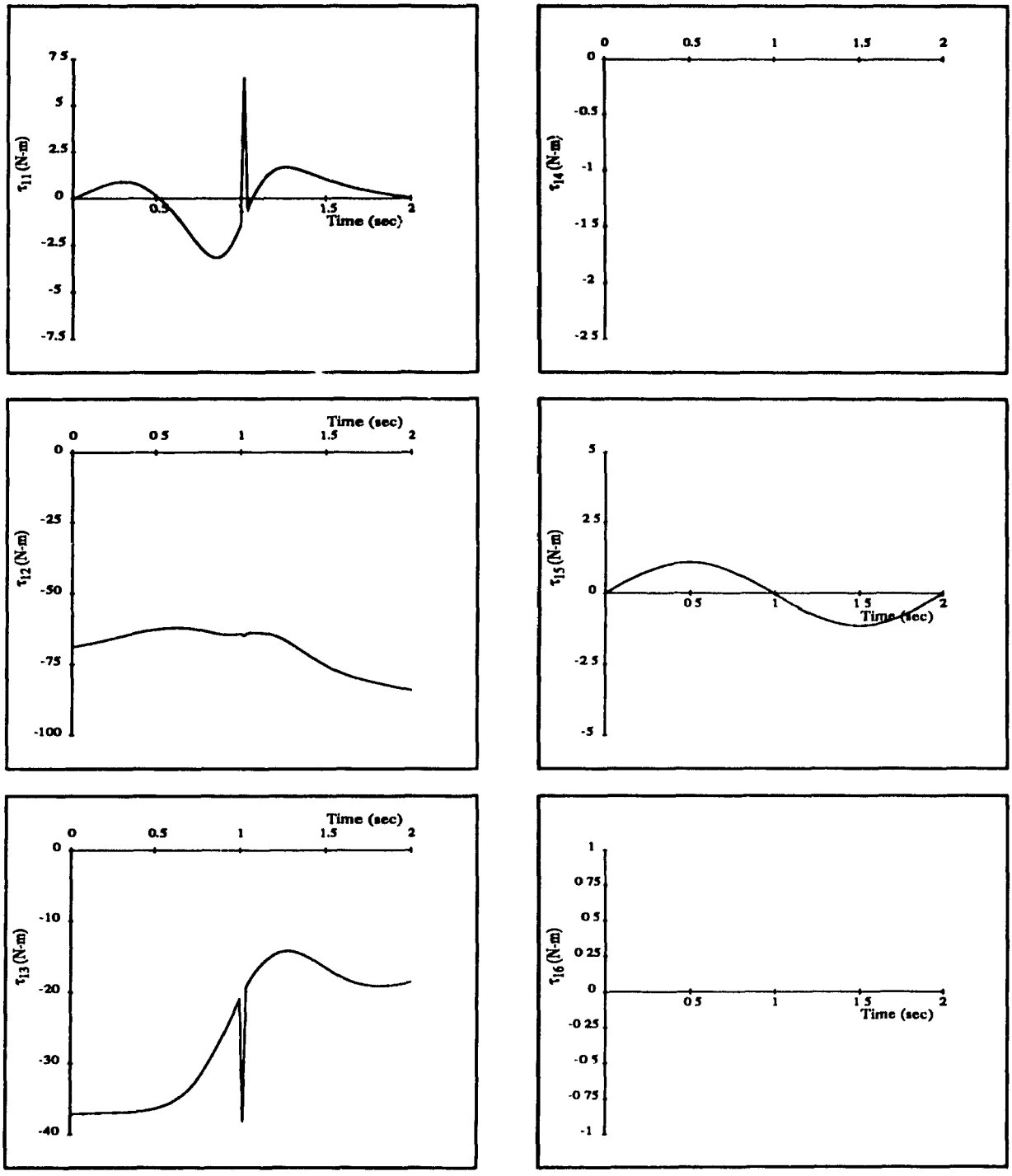

Figure 7.9 *Actuator Torques for Manipulator # 1 ($\rho = 0,\ \Delta = \infty$)*

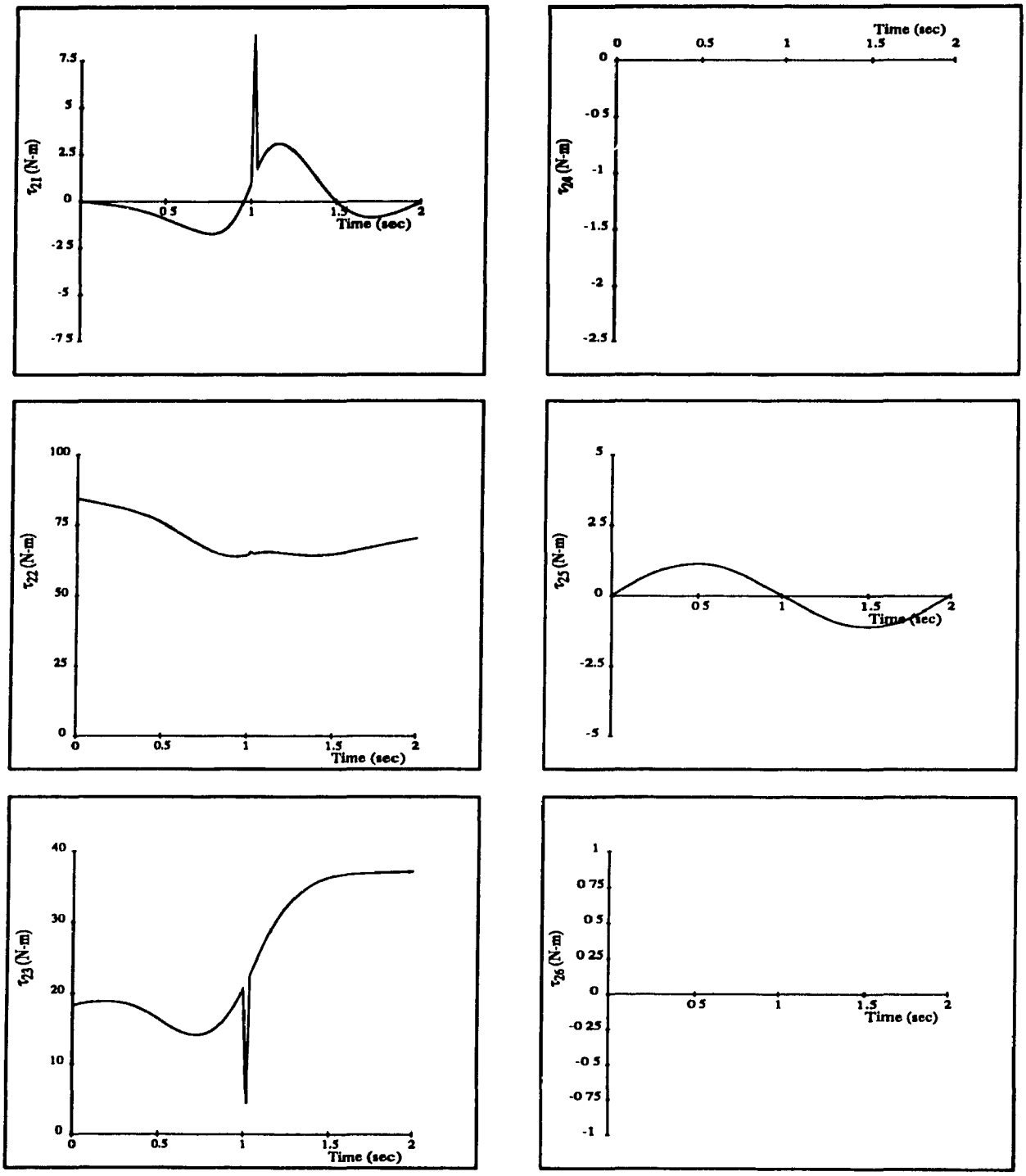

Figure 7.10 *Actuator Torques for Manipulator # 2 ($\rho = 0$, $\Delta = \infty$)*

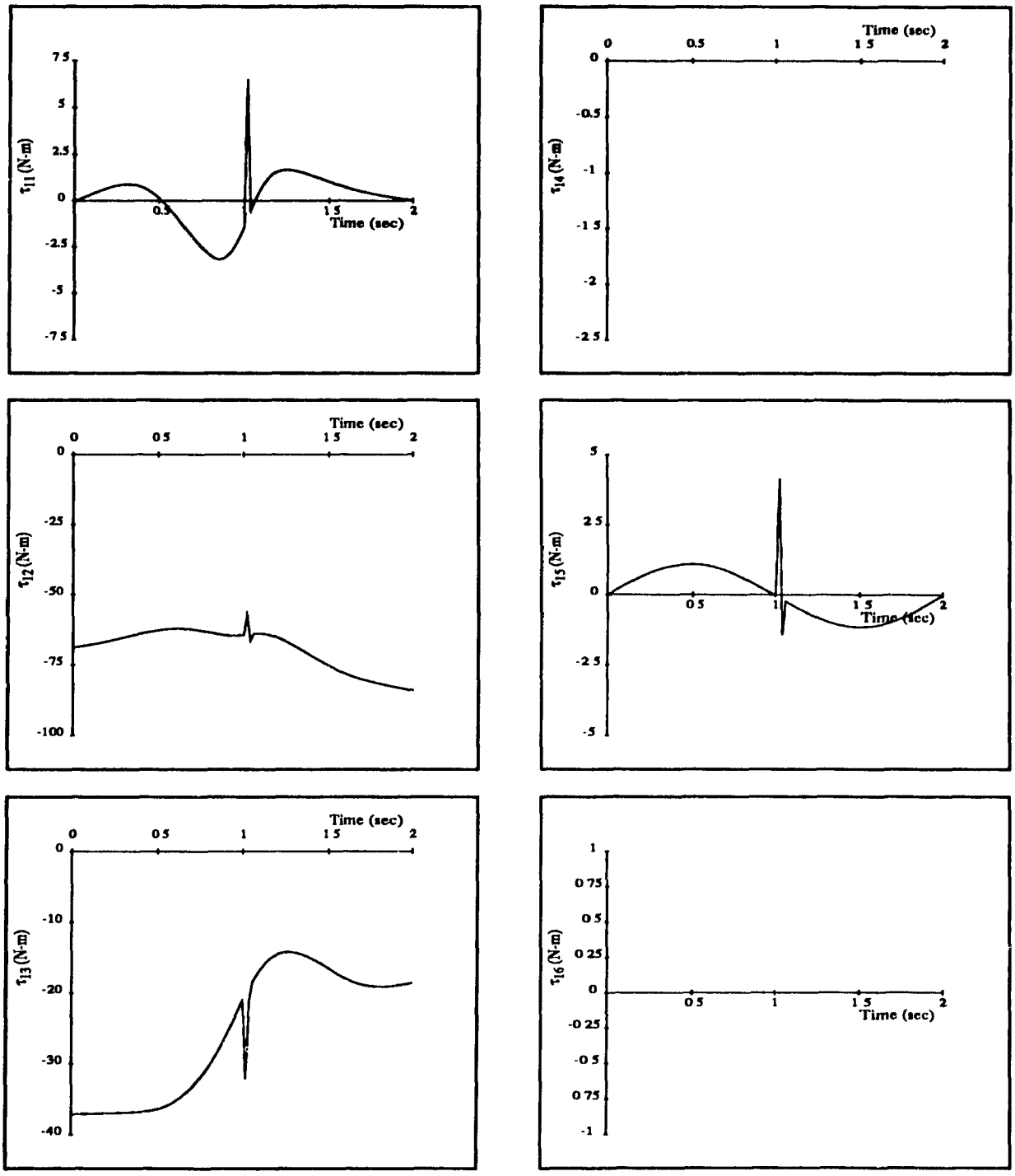

Figure 7.11 *Actuator Torques for Manipulator # 1 ($\rho = 0,\ \Delta = 11$)*

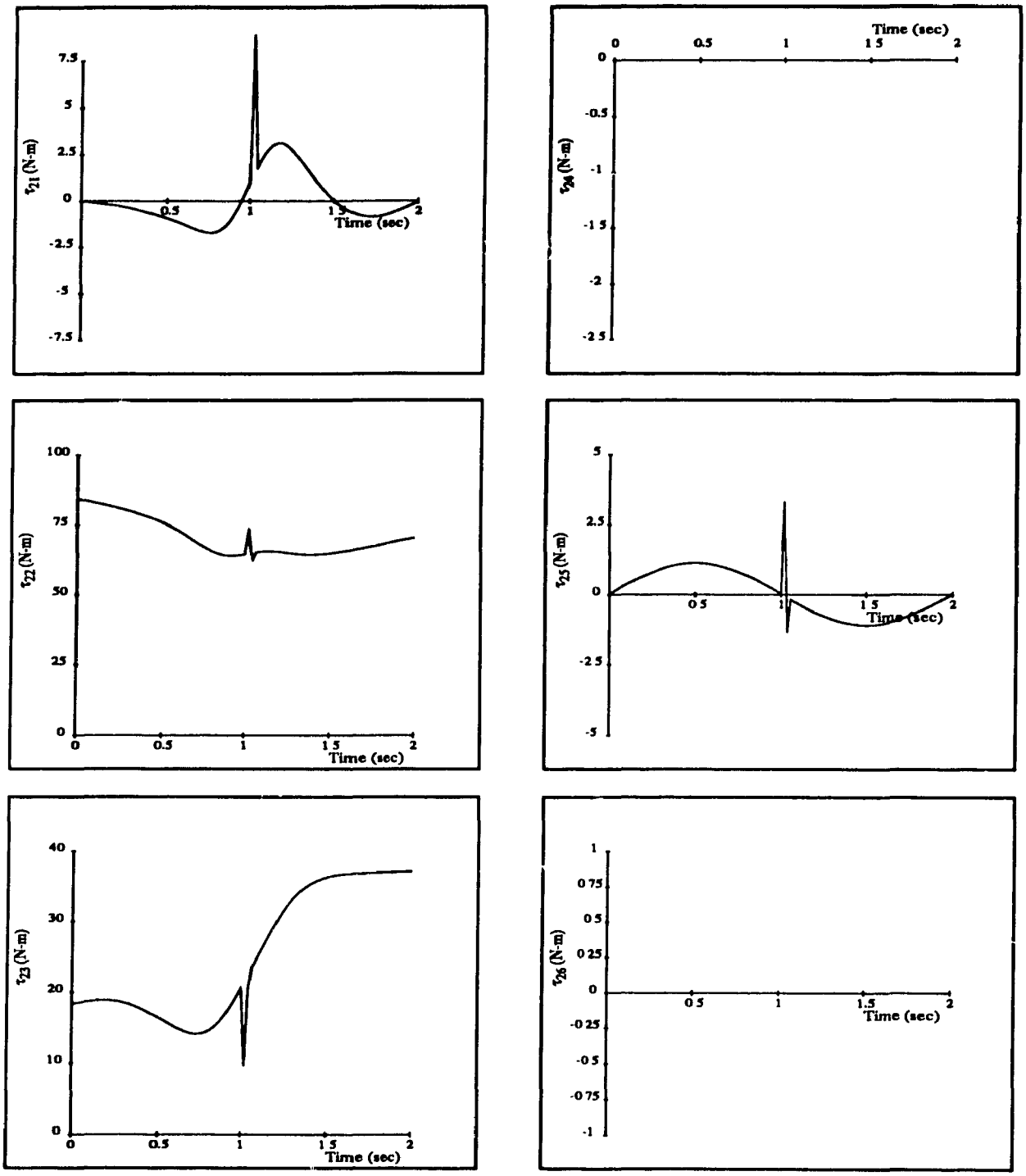

Figure 7.12 *Actuator Torques for Manipulator # 2 ($\rho = 0,\ \Delta = 11$)*

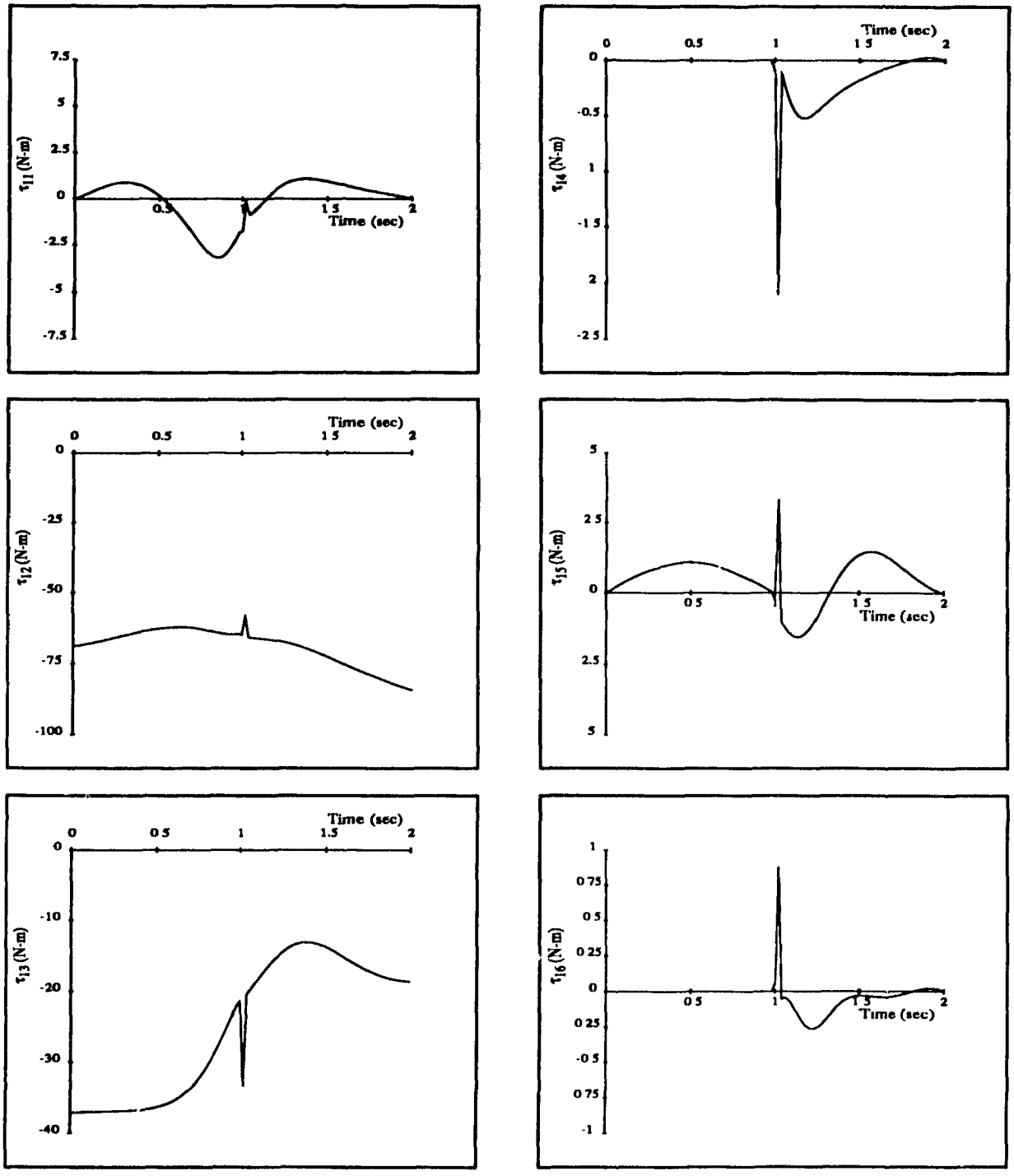

Figure 7.13 *Actuator Torques for Manipulator # 1 ($\rho = 10,\ \Delta = \infty$)*

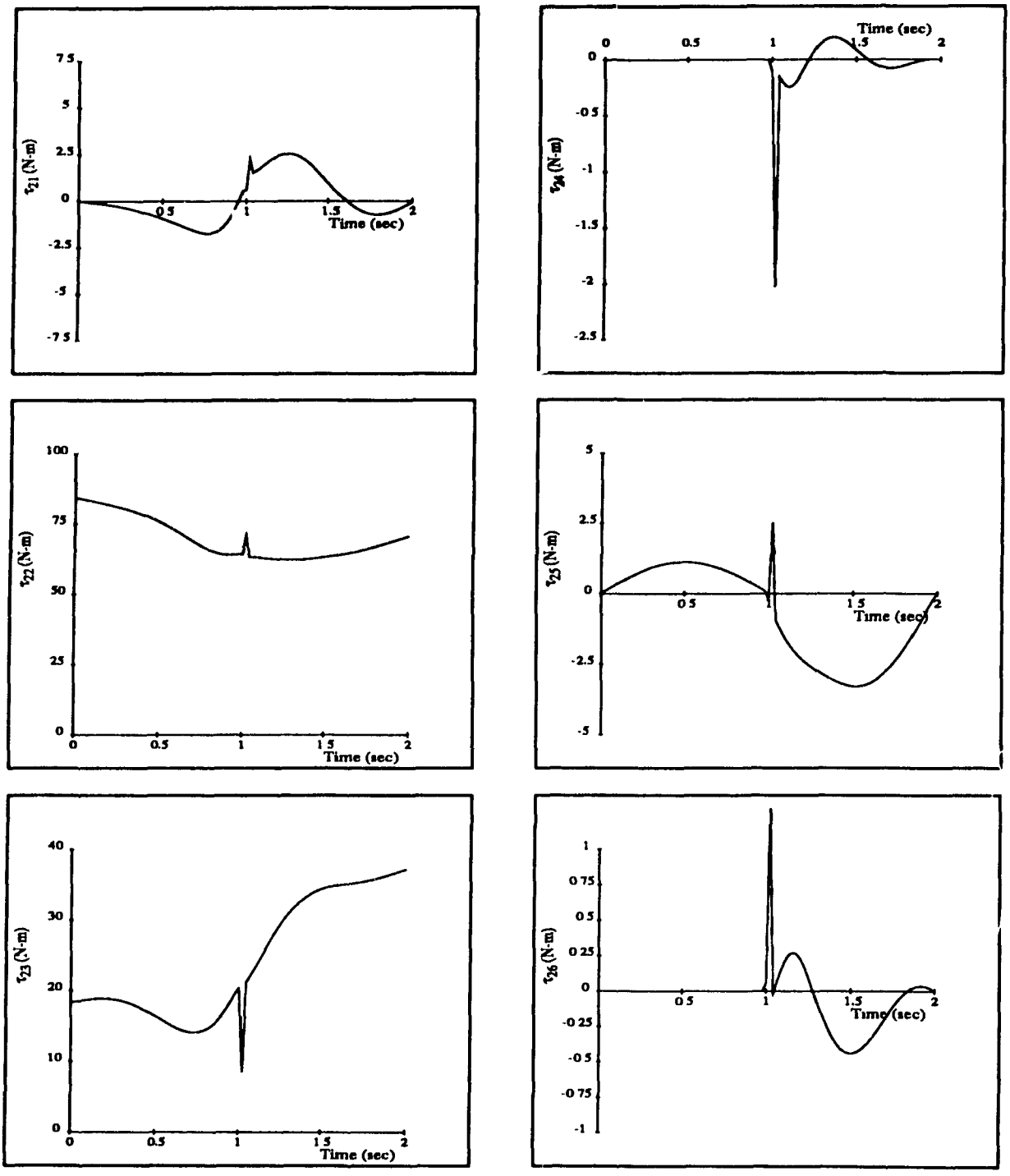

Figure 7.14 *Actuator Torques for Manipulator # 2 ($\rho = 10,\ \Delta = \infty$)*

Chapter 8

Conclusions

The fingers of mechanical hands, the legs of walking machines and multiple manipulators handling a common payload were treated in the present work as 'cooperating robotic devices'. One of the characteristics of these systems was found to be that their kinematic structure, or topology, varies with time. Since the number of actuators installed in these systems is chosen to ensure the controllability of its constituent robotic devices when they act independently, the system will tend to have more actuators than necessary at other times. This situation was therefore described as redundant actuation.

The number of actuators required in a mechanical system is dependent on its number of degrees of freedom. The determination of this property was therefore investigated in detail, as was the freedom allowed and constraint imposed by the interbody joints in the system. This analysis of kinematic structure was followed by an analysis of the motion of the systems in question, where analogies were drawn between them and serial and parallel manipulators.

The dynamic equations of motion of these systems were then studied and it was found that the relationship between the motion of the system and the wrenches acting within it can be written in a number of ways. In all cases, the inverse dynamics problem—that of finding the wrenches acting in the system for a known motion— could be formulated as an underdetermined system of linear equations. Furthermore, this system of

equations was found to be underdetermined by an amount equal to the degree of actuation redundancy existing in the system. When studying the behavior of these equations upon changes in the topology of the system, it was found that the left-hand side matrix of coefficients suffered discontinuous changes at these times. Other constraints were imposed in the solution to account for the limitations of passive contacts, actuator capabilities and interbody joint strength.

In order to find an 'ideal' solution to the underdetermined problem mentioned above, optimization techniques were investigated. In this context, the inverse dynamics equations were treated as linear equality constraints, while the friction forces, actuator torque bounds and joint limits were treated as inequality constraints. The characteristics of various optimization problems were investigated with emphasis on two considerations: 1) the speed with which the algorithm could provide a solution (since the optimization must be performed in real time in order to provide force setpoints for the controller), and 2) the uniqueness and continuity of the solution. Quadratic programming appeared to be preferable, particularly in the latter respect. Numerical techniques for solving the optimization problem were then investigated. In the case of a problem which could be written with only equality constraints, a technique which included symbolic preprocessing of the problem was developed. For more general problems, an existing inequality-constrained quadratic optimization algorithm was modified to efficiently include equality constraints. This method may be viewed as an extension of the pseudo-inverse solution commonly used for these systems to include inequality constraints.

Any optimization problem must optimize an objective function but the choice of this function can be difficult. A number of objective functions were compared for linear and quadratic programming and it became more apparent that the continuity properties of the latter were far superior. As well, the quadratic programming algorithm implemented in the present work proved to be faster for all but the lowest-dimensional problems. Various objective functions were discussed, namely, minimum 'internal force,' minimum norm of the actuator torques, minimum-norm of the joint constraint wrenches, minimum

power losses and minimum solution discontinuities upon changes in the topology of the system. It was found that there are a number of possible interpretations for the minimum 'internal force' approach, and that it would be useful to include a weighting matrix to ensure that its results are invariant with changes in units. As well, it was found that although the minimization of power losses may appear to be an attractive objective, it may result in excessive stresses being imposed on a grasped object.

Finally, the utility of redundant actuation in systems with fixed topology was outlined. It was shown that redundant actuation could be used to perform the complete dynamic balancing of a four-bar linkage. This, when performed on a linkage which was initially force-balanced, resulted in a linkage which exerted no dynamical forces at either of its frame supports. Redundant actuation was also used to reduce the effects of shocks on mechanisms. Although this approach has certain limitations, it may warrant further investigation.

8.1 Recommendations for Future Work

In the course of the research performed for the present work, a number of interesting avenues were noted which warrant further investigation.

Although the present work delved into a number of interesting objective functions, it is the opinion of the author that a universal choice for this objective function which would apply equally well to walking machines, mechanical hands and multiple manipulators handling a common payload is not possible. Rather, the choice of objective function will be governed by the particular characteristics of the system and, perhaps, its task. For example, Nakamura's minimization of strain energy might be well suited to delicate payloads while power loss minimization might be well suited to space-based systems with limited energy availability. Therefore, a series of more specific investigations would be useful to identify the 'ideal' objective functions for particular systems and tasks.

Throughout this work, it was assumed that the bodies which make up the systems under consideration were rigid. However, space-based robotic systems are known to have links with significant structural flexibility, while joint flexibility is acknowledged to be a major concern for industrial robotic systems. The effect of flexibility in the bodies and joints in a redundantly-actuated system would therefore be a useful topic to investigate. Conceptually, one of the effects of flexibility is the introduction of new degrees of freedom in the system. Since actuation redundancy is a result of having more actuators than degrees of freedom, it is possible that the flexibility could be controlled by the redundant actuators, or that the actuation redundancy would be reduced by flexibility. This is closely related to the passive compliance approach suggested by certain authors.

Another pervasive assumption in this thesis was that the robotic devices making up the system were not kinematically redundant. If the system were kinematically redundant, the motion of all its members would *not* be fully specified by specifying the motion of its moving pole. Two approaches could be taken: 1) the kinematic redundancy could first be resolved using existing techniques (e.g., minimum condition number of the Jacobian), and the force optimization problem could then be resolved independently, or 2) the two problems could be treated as one larger force optimization problem. The latter approach deserves some investigation.

The present work was primarily concerned with finding setpoints for a force-motion controller assumed to be installed in the system. No consideration was given to the manner in which these force setpoints could be combined with motion setpoints to achieve stable, consistent control. Although this is by no means a neglected research topic, certain concerns have been expressed about the validity of existing methods of combining setpoints. It should be apparent from the present work that the subspaces in which internal force and motion can be controlled are orthogonal. It is conjectured that two independent controllers could be designed in these subspaces rather than attempting to couple the setpoints in a single task space controller.

The fact that redundant actuation can be useful for systems with fixed topology is little acknowledged. Further work could be done in the complete dynamic balancing of linkages to extend this technique to more complex mechanisms. As well, an evaluation of redundant actuation in systems with fixed topology from a more global perspective is required. For example, redundant actuators add weight to the system, while they allow a lighter structure to be constructed by reducing the forces. It would be of practical importance to compare the trade-offs between these two effects.

References

Alberts, T. E., and Soloway, D. I., 1988, 'Force Control of a Multi-Arm Robot System', *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, pp. 1490–1496.

Anderson, K., and Angeles, J., 1989, 'Kinematic Inversion of Robotic Manipulators in the Presence of Redundancies', The International Journal of Robotics Research, Vol. 8, No. 6, pp. 80–97.

Angeles, J., 1982, *Spatial Kinematic Chains*, Springer-Verlag, New York.

Angeles, J., 1986, 'Automatic Computation of the Screw Parameters of Rigid-Body Motions. Part I: Finitely-Separated Positions', *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 108, No. 1, pp. 32–38.

Angeles, J., Anderson, K., and Gosselin, C., 1987, 'An Orthogonal-Decomposition Algorithm for Constrained Least-Squares Optimization', *Proceedings of the 13th ASME Design Automation Conference*, Boston, pp. 215–220.

Angeles, J., Alivizatos, A., and Zsombor-Murray, P. J., 1988, 'The Synthesis of Smooth Trajectories for Pick-and-Place Operations', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-18, No. 1, pp. 173–178.

Angeles, J., and Gosselin, C., 1988, 'Détermination du degré de liberté des chaînes cinématiques simples et complexes' ('On the Determination of the Degree of Freedom of Simple and Complex Kinematic Chains'), *Transactions of the Canadian Society of Mechanical Engineering*, Vol. 12, No. 4, pp. 219–226.

Angeles, J., Ma, O., and Rojas, A., 1989, 'An Algorithm for the Inverse Dynamics of n-Axis General Manipulators Using Kane's Equations', *Computers & Mathematics, with Applications*, Vol. 17, No. 12, pp. 1545–1561.

Angeles, J., and Lee, S., 1989, 'The Modelling of Holonomic Mechanical Systems Using a

Natural Orthogonal Complement', *Transactions of the CSME*, Vol. 13, No. 4, pp. 81–89.

Angeles, J., 1989, *Rational Kinematics*, Springer-Verlag, New York.

Armstrong, B., Khatib, O., and Burdick, J., 1986, 'The Explicit Dynamic Model and Inertial Parameters of the Puma 560 Arm', *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Fransisco, pp. 510–518.

Asada, H., and Slotine, J.-J. E., 1986, *Robot Analysis and Control*, John Wiley and Sons, New York.

Baker, J. E., 1981, 'On Mobility and Relative Freedoms in Multiloop Linkages and Structures', *Mechanism and Machine Theory*, Vol. 16, No. 6, pp. 583–597.

Bagci, C., 1979, 'Shaking Force Balancing of Planar Linkages with Force Transmission Irregularities Using Balancing Idler Loops', *Mechanism and Machine Theory*, Vol. 14, No. 4, pp. 267–284.

Bekker, M. G., 1960, *Off-The-Road Locomotion*, University of Michigan Press, Ann Arbor.

Berkof, R. S., and Lowen, G. G., 1969, 'A New Method for Completely Force Balancing Simple Linkages', *Transactions of the ASME*, ser. B, Vol. 91, No. 1, pp. 21–26.

Berkof, R. S., 1973, 'Complete Force and Moment Balancing of Inline Four-Bar Linkages', *Mechanism and Machine Theory*, Vol. 8, No. 3, pp. 397–410.

Bessanov, A. P., and Umnov, N. V., 1983, 'The Stabilization of the Position of the Body of Walking Machines', *Mechanism and Machine Theory*, Vol. 18, No. 4, pp. 261–265.

Beveridge, G. S., and Schechter, R. S., 1970, *Optimization: Theory and Practice*, McGraw-Hill, New York.

Boot, J. C. G., 1964, *Quadratic Programming*, North-Holland Publishing Company, Amsterdam.

Borduas, H., Gossain, D., Kong, A., Quittner, E., and Shaffer, D., 1989, 'Concept Design of the Special Purpose Dexterous Manipulator for the Space Station Mobile Servicing System', *Canadian Aeronautics and Space Journal*, Vol. 35, No. 4, pp. 197–204.

Cai, C., 1988, 'Instantaneous Robot Motion with Contact Between Surfaces', Doctoral Dissertation, Stanford University.

Carignan, C. R., and Akin, D. L., 1989, 'Optimal Force Distribution for Payload Posi-

tioning Using a Planar Dual-Arm Robot', *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 111, No. 2, pp. 205–210.

Cheng, F. T., 1989, 'Efficient Algorithms for Optimal Force Distribution in Multiple-Chain Robotic Systems', Ph. D. Thesis, The Ohio State University.

Cheng, F. T., and Orin, D. E., 1989, 'Efficient Algorithm for Optimal Force Distribution in Multiple-Chain Robotic Systems—The Compact-Dual LP Method', *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Ariz., pp. 943–950.

Crossley, F. R. E., and Umholtz, F. G., 1977, 'Design for a Three-Fingered Hand', *Mechanism and Machine Theory*, Vol. 12, No. 1, pp. 85–93.

Cutkosky, M., 1985, *Robotic Grasping and Fine Manipulation'*, Kluwer Academic Publishers, Boston.

Daneshmend, L., 1990, personal communication.

Danowski, P., 1989, 'Simulation einer Stabheuschrecke' ('Simulation of a Walking Stick'), Masters Thesis, Technical University of Munich.

Davies, T. H., 1981, 'Kirchoff's Circulation Law Applied to Multi-Loop Kinematic Chains', *Mechanism and Machine Theory*, Vol. 16, No. 3, pp. 171–183.

Dobrjanskyj, L., and Freudenstein, F., 1967, 'Some Applications of Graph Theory to the Structural Analysis of Mechanisms', *Transactions of the ASME Journal of Engineering for Industry*, Vol. 89, No. 1, pp. 153–158.

Electro-Craft Corporation, 1980, *DC Motors Speed Controls Servo Systems*, Electro-Craft Corporation, 1600 Second St. South, Hopkins, Minn. 55343.

Esroy, M., 1976, 'Systematik und Katalog für Räumliche Elementenpaare' ('Classification of and Catalog for Spatial Pairs of Elements'), *Mechanism and Machine Theory*, Vol. 11, No. 5, pp. 331–342.

Feng, G., 1989, 'Complete Shaking Force and Shaking Moment Balancing of Four Types of Six-Bar Linkages', *Mechanism and Machine Theory*, Vol. 24, No. 4, pp. 275–287.

Freund, R. M., 1985, 'Postoptimal Analysis of a Linear Program Under Simultaneous Changes in Matrix Coefficients', *Mathematical Programming Study 24*, pp. 1–13.

Gao, X. C., and Song, S. M., 1990, 'Stiffness Matrix Method for Foot Force Distribution

of Walking Vehicles', *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, pp. 1470–1475.

Gardner, J. F., Kumar, V., Ho, J. H , 1989, 'Kinematics and Control of Redundantly Actuated Close Chains', *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Ariz., pp. 418–424.

Gill, P. E., and Murray, W., 1978, 'Numerically Stable Methods for Quadratic Programming', *Mathematical Programming*, Vol. 14, No. 3, pp. 349–372.

Gill, P. E., Murray, W., and Wright, M. H., 1981, *Practical Optimization*, Academic Press.

Goldfarb, D., and Idnani, A., 1983, 'A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs', *Mathematical Programming*, Vol. 27, No. 1, pp. 1–33.

Golub, G. H., and Van Loan, C. F., 1983, *Matrix Computations*, John Hopkins University Press, Baltimore.

Gosselin, C., 1988, 'Kinematic Analysis, Optimization and Programming of Parallel Robotic Manipulators', Doctoral Dissertation, McGill University.

Gurfinkel, V. S., Gurfinkel, E. V., Shneider, A. Yu., Devjanin, E. A., and Lensky, A. V., 1981, 'Walking Robot with Supervisory Control', *Mechanism and Machine Theory*, Vol. 16, No. 1, pp. 31–36.

Gray, H., *Anatomy of the Human Body*, 1985, Thirtieth American Edition, ed. C. D. Clemente, Lea and Febiger, Philadelphia.

Haftka, R. T., and Kamat, M. P., 1985, *Elements of Structural Optimization* Martinus Nijhoff Publishers, Dordrecht, The Netherlands.

Harary, F., 1972, *Graph Theory*, Addison-Wesley, Reading, Mass.

Hartenberg, R. S., and Denavit, J., 1964, *Kinematic Synthesis of Linkages*, McGraw-Hill, New York.

Haug, E. J., Wu, S. C., and Yang, S. M., 1986, 'Dynamics of Mechanical Systems with Coulomb Friction, Stiction, Impact and Constraint Addition-Deletion—I, II and III', *Mechanism and Machine Theory*, Vol. 21, No. 5, pp. 401–425.

Hayati, S., 1986, 'Hybrid Position/Force Control of Multi-Arm Cooperating Robots',

Proceedings of the 1986 IEEE International Conference on Robotics and Automation, San Fransisco, pp. 82–89.

Hayward, V., 1988, 'An Analysis of Redundant Manipulators From Several View-Points', *NATO Advanced Research Workshop—Robots with Redundancy: Design, Sensing and Control*, June 1988, Salo, Lago di Garda, Italy.

Hayward, V., and Hayati, S., 1988, 'Kali: An Environment for the Programming and Control of Cooperative Manipulators', *Proceedings of the 1988 American Control Conference*, Atlanta, pp. 473–478.

Hervé, J. M., 1978, 'Analyse structurelle des mécanismes par groupe des déplacements' ('Structural Analysis of Mechanisms Using Displacement Sets'), *Mechanism and Machine Theory*, Vol. 13, No. 4, pp. 437–450.

Hirose, S., 1984, 'A Study of Design and Control of a Quadruped Walking Vehicle', *The International Journal of Robotics Research*, Vol. 3, No. 2, pp. 113–133.

Hsu, P., Li, Z., and Sastry, S., 1988, 'On Grasping and Coordinated Manipulation by a Multifingered Robot Hand', *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, Pa., pp. 384–389.

Hu, Y.-R., and Goldenberg, A. A., 1990, 'Dynamic Control of Multiple Coordinated Redundant Manipulators with Torque Optimization', *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, pp. 1000–1005.

Hunt, K. H., 1978, *Kinematic Geometry of Mechanisms*, Clarendon Press, Oxford.

Hussaini, S. S., and Jakopac, D. E., 1986, 'Multiple Manipulators and Robotic Workcell Coordination', *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Fransisco, pp. 1236–1241.

IMSL, 1987, 'MATH/LIBRARY User's Manual', IMSL, 2500 ParkWest Tower One, 2500 CityWest Blvd., Houston, 77042-3020.

Iwata, T., Oda, M., and Nakamura, T., 1989, 'Development of the 2nd Generation Space Robot in NASDA', 40th Congress of the International Astronautical Federation, Oct. 7–12, India.

Jacobsen, S. C., Wood, J. E., Knutti, D. F., and Biggers, K. B., 1984, 'The Utah/M.I.T. Dextrous Hand: Work in Progress', *The International Journal of Robotics Research*, Vol. 3, No. 4, pp. 21–50.

Jacobsen, S. C., Iversen, E. K., Knutti, D. F., Johnson, R. T., and Biggers, K. B., 1986, *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Fransisco, pp. 1520–1532.

Kahng, J., and Amirouche, F. M., 1987, 'Impact Force Analysis in Mechanical Hand Design', *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, N.C., pp. 2061–2067.

Kane, T. R., 1968, *Dynamics*, Holt, Rinehart and Winston, New York.

Kerr, J., and Roth, B., 1986, 'Analysis of Multifingered Hands', *The International Journal of Robotics Research*, Vol. 4, No. 4, pp. 3–17.

Klein, C. A., and Briggs, R. L., 1980, 'Use of Active Compliance in the Control of Legged Vehicles', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-10, No. 7, pp. 393–400.

Klein, C. A., Olson, K. W., and Pugh, D. R., 1983, 'Use of Force and Attitude Sensors for Locomotion of a Legged Vehicle Over Irregular Terrain', *International Journal of Robotic Research*, Vol. 2, No. 2, pp. 3–17.

Klein, C. A., and Chung, T.-S., 1987, 'Force Interaction and Allocation for the Legs of a Walking Vehicle', *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 6, pp. 546–555.

Klein, C. A., and Kittivatcharapong, S., 1990, 'Optimal Force Distribution for the Legs of a Walking Machine with Friction Cone Constraints', *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 1, pp. 73–85.

Kochev, I. S., 1988, 'A New General Method For Full Force Balancing of Planar Linkages', *Mechanism and Machine Theory*, Vol. 23, No. 6, pp. 475–480.

Kopf, C. D., 1988a, 'Dynamic Two-Arm Hybrid Position/Force Control', Masters Thesis, University of California at Santa Barbara, June 1988.

Kopf, C. D., 1988b, 'Two-Arm Hybrid Position/Force Control with Dynamic Compensation', *Proceedings of the Second International Symposium on Robotics and Manufacturing*, Albuquerque, New Mexico, pp. 371–381.

Kuhn, H. W., and Tucker, A. W., 1951, 'Nonlinear Programming', *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, ed. J. Neyman, University of California Press, pp. 481–492.

Kumar, V., and Waldron, K. J., 1988, 'Force Distribution in Closed Kinematic Chains', *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, pp. 114–119.

Kumar, V., and Waldron, K. J., 1989, 'Suboptimal Algorithms for Force Distribution in Multifingered Grippers', *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 4, pp. 491–498.

Lallemand, J.-P., 1988, *Techniques de la robotique, Tome 1: architectures et commande*, ed. J.-D.Boissonnat, B. Faverjon and J.-P. Merlet, Hermès, Paris.

Lawson, C. L. and Hanson R. J., 1974, *Solving Least Squares Problems*, Prentice Hall, Englewood Cliffs, N. J.

Leahy Jr., M. B., and Saridis, G. N., 'Compensation of Industrial Manipulator Dynamics', *The International Journal of Robotics Research*, Vol. 8, No. 4, pp. 73–84.

Lötstedt, P., 1984, 'Numerical Simulation Of Time-Dependent Contact and Friction Problems in Rigid Body Mechanics', SIAM Journal on Scientific and Statistical Computing, Vol. 5, No. 2, pp. 370–393.

Luh, J. Y. S., Walker, M. W., and Paul, R. P. C., 1980, 'On-Line Computational Scheme for Mechanical Manipulators', *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 102, No. 2, pp. 69–76.

Luh, J. Y. S., and Zheng, Y. F., 1985, 'Computation of Input Generalized Forces for Robots with Closed Kinematic Chain Mechanisms', *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 2, pp. 95–103.

Luh, J. Y. S., and Zheng, Y. F., 1988, 'Load Distribution Between Two Coordinating Robots by Nonlinear Programming', *Proceedings of the 1988 American Control Conference*, Atlanta, pp. 479–482.

Mason, M. T., and Salisbury, J. K., 1985, *Robot Hands and the Mechanics of Manipulation*, The MIT Press, Cambridge, Mass.

Mathlab Group, Laboratory for Computer Science, MIT, 1983, 'MACSYMA Reference Manual', MACSYMA Group, Symbolics Inc., 257 Vassar St., Cambridge Mass. 02139.

McCain, H. G., 1990, 'NASA's First Dexterous Space Robot', *Aerospace America*, Feb. 1990, pp. 12–30.

McGhee, R. B., and Orin, D. E., 1976, 'A Mathematical Programming Approach to

Control of Joint Positions and Torques in Legged Locomotion Systems', *Proceedings of the Second International CISM-IFToMM Symposium on the Theory and Practice of Robots and Manipulators*, ed. A. Morecki and K. Kedzior, Sept. 14–17, Warsaw, Poland, pp. 231–239.

McGhee, R. B., and Iswandhi, G. I., 1979, 'Adaptive Locomotion of a Multilegged Robot Over Rough Terrain', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-9, No. 4, pp. 176–182.

Merlet, J-P., 1990, 'Assembly-Modes and Minimal Polynomial Formulation of the Direct Kinematics of Parallel Manipulators', *Proceedings of the CSME Mechanical Engineering Forum, Vol. III*, June 3–9, Toronto, pp. 343–348.

Miura, H., and Shimoyama, I., 1984, 'Dynamic Walk of a Biped', *The International Journal of Robotics Research*. Vol. 3, No. 2, pp. 60–74.

Mosher, R. S., 1969, 'Exploring the Potential of a Quadruped', (SAE paper 690191) *SAE Transactions*, Vol. 78, pp. 836–843.

Nahon, M., and Angeles, J., 1989a, 'Force Optimization in Redundantly-Actuated Closed Kinematic Chains', *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Ariz., pp. 951–956.

Nahon, M., and Angeles, J., 1989b, 'Optimization of Dynamic Forces in the Design of Mechanical Hands', *Advances in Design Automation—1989, DE Vol. 19-3*, presented at the 15th ASME Design Automation Conference, Montreal, Canada, pp. 289–295.

Nahon, M., and Angeles, J., 1990a, 'Smoothing of Force Discontinuities in Robotic Systems with Time-Varying Topologies', *Proceedings of the CSME Mechanical Engineering Forum, Vol. III*, June 3–9, Toronto, pp. 411–416.

Nahon, M., and Angeles, J., 1990b, 'Cooperative Control of Multiple Space Manipulators', *Proceedings of the Sixth CASI Conference on Astronautics*, Nov.19–21, Ottawa, pp.28–37.

Nahon, M., and Angeles, J., 1991a, 'Real-Time Force Optimization in Parallel Kinematic Chains Under Inequality Constraints', *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento.

Nahon, M., and Angeles, J., 1991b, 'Minimization of Power Losses in Cooperating Manipulators', *Proceedings of the 1991 American Control Conference*, Boston.

Nakamura, Y., Nagai, K., and Yoshikawa, T., 1987, 'Mechanics of Coordinative Manipulation by Multiple Robotic Mechanisms', *Proceedings of the 1987 IEEE International*

Conference on Robotics and Automation, Raleigh, N.C., pp. 991–998.

Nakamura, Y., 1988a, 'Dynamics of Closed Link Robots with Actuational Redundancy', *Proceedings of the Second International Symposium on Robotics and Manufacturing*, Albuquerque, New Mexico, pp. 309–318.

Nakamura, Y., 1988b, 'Minimizing Object Strain Energy for Coordination of Multiple Robotic Manipulators", *Proceedings of the 1988 American Control Conference*, Atlanta, pp. 499–504.

Nakamura, Y., and Ghodoussi, M., 1989, 'Dynamic Computation of Closed-Link Robot Mechanisms with Nonredundant and Redundant Actuators', *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 3, pp. 294–302.

Nakamura, Y., and Ropponen, T., 1989, 'A Closed Link Manipulator with Kinematic and Actuation Redundancies', Robotics Research—1989, Proceedings of the Winter Annual Meeting of the ASME, DSC-Vol. 14, pp. 203–210.

Okada, T., 1979, 'Computer Control of Multijointed Finger System', *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Aug. 20–23, Tokyo, pp. 693–695.

Orin, D. E., and Oh, S. Y., 1981, 'Control of Force Distribution in Robotic Mechanisms Containing Closed Kinematic Chains', *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 103, No. 2, pp. 134–141.

Orin, D. E., 1982, 'Supervisory Control of a Multilegged Robot', *The International Journal of Robotics Research*, Vol. 1, No. 1, pp. 79–91.

Park, Y. C., and Starr, G. P., 1989, 'Finger Force Computations for Manipulation of An Object by a Multifingered Robot Hand', *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Ariz., pp. 930–935.

Parker, J. K., and Paul, F. W., 1985, 'Impact Force Contro' in Robot Hand Design', *Proceedings of the 1985 ASME Winter Annual Meeting*, Vol. 15, pp. 57–65

Pfeiffer, F., Weidemann, H.-J., and Danowski, P., 1990, 'Dynamics of the Walking Stick Insect', *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, pp. 1458–1463.

Pfeiffer, F., 1991 'Dynamical Systems with Time-Varying or Unsteady Structure', *Zeitschrift für Angewandte Mathematik und Mechanik*, Vol. 71, No. 4/5/6.

Pieper, D. L., 1968, 'The Kinematics of Manipulators Under Computer Control', Doctoral Dissertation, Stanford University.

Popov, E. P., 1982, *Modern Robot Engineering*, MIR Publishers, Moscow.

Powell, M. J. D., 1983, *ZQPCVX: A Fortran Subroutine for Convex Programming*, Report DAMTP NA-17, Department of Applied Mathematics and Theoretical Physics, University of Cambridge.

Powell, M. J. D., 1985, 'On the Quadratic Programming Algorithm of Goldfarb and Idnani' *Mathematical Programming Study 25*, pp. 46–61.

Press, W., H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W., T., 1986, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, England.

Raibert, M. H., Brown, H. B. Jr., and Chepponis, M., 1984, 'Experiments in Balance with a 3D One-Legged Hopping Machine', *The International Journal of Robotics Research*, Vol. 3, No. 2, pp. 75–92.

Raibert, M. H., Chepponis, M., and Brown, H. B. Jr., 1986, 'Running on Four Legs as Though They Were One', *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 2, pp. 70–82.

Rao, C. R., and Mitra, S. K., 1971, *Generalized Inverse of Matrices and its Applications*, John Wiley and Sons, New York.

Roberts, A. W., and Varberg, D. E., 1973, *Convex Functions*, Academic Press, New York.

Rodriguez, G., Kreutz, K., and Jain, A., 1989, 'A Spatial Operator Algebra for Manipulator Modeling and Control', *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Ariz., pp. 1374–1379.

Rosen, J. B., 1960, 'The Gradient Projection Method for Nonlinear Programming, Part I. Linear Constraints', SIAM Journal of Applied Mathematics, Vol. 8, pp. 181–217.

Russell, M. Jr., 1983, 'Odex I: The First Functionoid', *Robotics Age*, Vol. 5, No. 5, pp. 12–18.

Salisbury, J. K., and Craig, J. J., 1982, 'Articulated Hands: Force Control and Kinematic Issues', *The International Journal of Robotics Research*, Vol. 1, No. 1, pp. 4–17.

Salisbury, J. K., and Roth, B., 1983, 'Kinematic and Force Analysis of Articulated Me-

chanical Hands', *Transactions of the ASME, Journal of Mechanisms, Transmissions and Automation in Design*, Vol. 105, No. 1, pp. 35–41.

Santoso, B., 1987, 'Design and Control of a Multi-Fingered Robot Hand with Tactile Feedback', Doctoral Dissertation, Katholieke Universiteit Leuven.

Shaw, F. S., 1972, *Virtual Displacements and Analysis of Structures*, Prentice-Hall, Englewood Cliffs, New Jersey.

Shipman, P., Walker, A., and Bichell, D., 1985, *The Human Skeleton*, Harvard University Press, Cambridge, Mass.

Skinner, F., 1975, 'Designing a Multiple Prehension Manipulator', *Mechanical Engineering*, Vol. 97, No. 9, pp. 30–37.

Spring, K. W., 1986, 'Euler Parameters and the Use of Quaternion Algebra in the Manipulation of Finite Rotations: A Review', *Mechanism and Machine Theory*, Vol. 21, No. 5, pp. 365–373.

Strang, G., 1976, *Linear Algebra and Its Applications*, Academic Press, New York.

Sutherland, I. E., and Ullner, M. K., 1984, 'Footprints in the Asphalt', *The International Journal of Robotics Research*, Vol. 3, No. 2, pp. 29–36.

Takanishi, A., Takeya, T., Karaki, H., and Kato, I., 1990, 'A Control Method for Dynamic Biped Walking Under Unknown External Force', *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, July 3–6, Tsuchiura, Japan.

Tarn, T. J., Bejczy, A. K., and Li, Z. F., 1988, 'Connected Robot Arms as Redundant Systems', *NATO Advanced Research Workshop—Robots with Redundancy: Design, Sensing and Control*, June 1988, Salo, Lago di Garda, Italy.

Todd, D. J., 1985 *Walking Machines: An Introduction to Legged Robots*, Kogan Page, London.

Tournassoud, P., 1988, *Géométrie et intelligence artificielle pour les robots*, Hermès, Paris.

Tomović, R., and Stojiljković, Z., 'Multifunctional Terminal Device with Adaptive Grasping Force', *Automatica*, Vol. 11, No. 6, pp. 567–570.

Tsai, L.-W., and Morgan, A. P., 1985, 'Solving the Kinematics of the Most General Six- and Five-Degree-of-Freedom Manipulators by Continuation Methods', *ASME Journal of Mechanisms, Transmissions and Automation in Design*, Vol. 107, pp. 189–200.

Tsai, L.-W., and Lee, J.-J., 1989, 'Kinematic Analysis of Tendon-Driven Robotic Mechanisms Using Graph Theory', *Transactions of the ASME, Journal of Mechanisms, Transmissions and Automation in Design*, Vol. 111, No. 1, pp. 59–65.

Umnov, N. V., and Pogrebniac, A. Y., 1975, '**Перемещение Корпуса Шагающего Экипажа При Ходьбе**' ('Body Movement of Walking Machines During Walking'), *Proceedings of the Fourth World Congress on the Theory of Machines and Mechanisms*, Sept. 8–12, Newcastle-Upon-Tyne, pp. 365–369.

Vanderplaats, G. N., 1984, *Numerical Optimization Techniques for Engineering Design*, McGraw-Hill, New York.

Waldron, K. J., 1966, 'The Constraint Analysis of Mechanisms', Journal of Mechanisms, Vol. 1, No. 2, pp. 101–114.

Waldron, K. J., Vohnout, V. J., Pery, A., and McGhee, R. B., 1984, 'Configuration Design of the Adaptive Suspension Vehicle', *The International Journal of Robotics Research*, Vol. 3, No. 2, pp. 37–48.

Waldron, K. J., 1986, 'Force and Motion Management in Legged Locomotion', *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 4, pp. 214–220.

Waldron, K. J., Klein, C. A., Pugh, D., Vohnout, V. J., Ribble, E., Patterson, M., and McGhee, R. B., 1987, 'Operational Experience with the Adaptive Suspension Vehicle', *Proceedings of the Seventh World Congress on the Theory of Machines and Mechanisms*, Sevilla, Spain, Sept. 1987, pp. 1495–1498.

Wang, Y., and Mason, M. T., 1987, 'Modelling Impact Dynamics for Robotic Operations', *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, N.C., pp. 678–685.

West, H. H., 1980, *The Analysis of Structures*, McGraw-Hill, New York.

Whitney, D. E., 1972, 'The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators', *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 94, No. 4, pp. 303–309.

Wilde, D. J., and Beightler, C. S., 1967, *Foundations of Optimization*, Prentice Hall, Englewood Cliffs, New Jersey, 1967.

Williams, R. J., and Seireg, A., 1979, 'Interactive Modeling and Analysis of Open or Closed Loop Dynamic Systems with Redundant Actuators', *Transactions of the ASME, Journal of Mechanical Design*, Vol. 101, No. 3, pp. 407–416.

Wittenburg, J., 1977, *Dynamics of Systems of Rigid Bodies*, B. G. Teubner, Stuttgart.

Youcef-Toumi, K., and Gutz, D. A., 1989, 'Impact and Force Control', *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Ariz., pp. 410–416.

Zheng, Y. F., and Hemami, H., 1985, 'Mathematical Modeling of a Robot Collision with its Environment', *Journal of Robotic Systems*, Vol. 2, No. 3, pp. 289–307.

Zheng, Y. F., and Luh, J. Y. S., 1986, 'Joint Torques for Control of Two Coordinated Moving Robots', *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Fransisco, pp. 1375–1380.

Zheng, Y. F., 1987, 'Collision Effects on Two Coordinating Robots in Assembly and the Effect Minimization', *IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-17*, No. 1, pp. 108–116.

Zheng, Y. F., and Luh, J. Y. S., 1989, 'Optimal Load Distribution for Two Industrial Robots Handling a Single Object', *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 111, No. 2, pp. 233–237.

Appendix A

Inverse Dynamics of Serial Manipulators

A number of inverse dynamics algorithms for serial manipulators have been proposed in the literature. One of the most efficient of these is the recursive Newton-Euler algorithm put forward by Luh *et al.* (1980). It is presented here not because it is the most efficient of the existing algorithms, but rather because it is relatively easy to understand. In cases where the available computational power is restricted, the reader is encouraged to investigate more efficient algorithms such as that of Angeles *et al.* (1989).

Luh *et al.*'s algorithm is composed of two recursive procedures

1. An outward recursion from the manipulator's base to its tip in which the motion of each link of the manipulator in Cartesian space is determined as a function of the motion of the preceding links and joints.

2. An inward recursion from the manipulator's tip to its base to calculate the actuator and constraint wrenches acting on each link.

The equations which are implemented to accomplish this will first be presented in vector form, and will then be given in component form where the transformations necessary during numerical implementation are more apparent.

A.1 Vector Form of the Equations

The algorithm starts by specifying the motion of the base of the manipulator—ω_0, $\dot{\omega}_0$ and $\mathbf{a}_0$—where ω_0 and $\mathbf{a}_0$ are the angular velocity and translational acceleration of the base link, respectively. The outward recursion then consists of (for $i = 1$ to n, where n is the number of links in the manipulator):

When joint i is revolute:

$$\omega_i = \omega_{i-1} + \dot{q}_i \mathbf{e}_{i-1} \tag{A.1a}$$

$$\dot{\omega}_i = \dot{\omega}_{i-1} + \ddot{q}_i \mathbf{e}_{i-1} + \omega_{i-1} \times \dot{q}_i \mathbf{e}_{i-1} \tag{A.1b}$$

$$\mathbf{a}_i = \mathbf{a}_{i-1} + \dot{\omega}_i \times \mathbf{r}_i + \omega_i \times (\omega_i \times \mathbf{r}_i) \tag{A.1c}$$

When joint i is prismatic:

$$\omega_i = \omega_{i-1} \tag{A.2a}$$

$$\dot{\omega}_i = \dot{\omega}_{i-1} \tag{A.2b}$$

$$\mathbf{a}_i = \mathbf{a}_{i-1} + \dot{\omega}_i \times \mathbf{r}_i + \omega_i \times (\omega_i \times \mathbf{r}_i) + \ddot{q}_i \mathbf{e}_{i-1} + 2\omega_{i-1} \times \dot{q}_i \mathbf{e}_{i-1} \tag{A.2c}$$

where ω_i denotes the angular velocity of the i-th link, $\mathbf{a}_i$ denotes the translational acceleration of the i-th link at the origin of the i-th frame, q_i is the i-th joint variable, $\mathbf{e}_i$ is a vector aligned with the i-th joint axis and $\mathbf{r}_i$ represents the vector from the origin of frame $(i-1)$ to the origin of frame i.

For both types of joints, the translational acceleration of the i-th link at its centroid, $\mathbf{a}_{ci}$, is given by:

$$\mathbf{a}_{ci} = \mathbf{a}_i + \dot{\omega}_i \times \mathbf{c}_{i,i} + \omega_i \times (\omega_i \times \mathbf{c}_{i,i}) \tag{A.3}$$

where $\mathbf{c}_{i,i}$ is a vector from the origin of frame i to the centroid of link i.

This completes the outward recursion procedure. The inward recursion makes use of the Cartesian motions of all the links found during the outward recursion (ω_i, $\dot{\omega}_i$

and $\mathbf{a}_{ci}$ for $i = 1, \ldots, n$) and the fact that the external wrench acting on link n is known (and denoted by the force vector $\mathbf{f}_{n,n+1}$ and the moment vector $\mathbf{n}_{n,n+1}$). The inward recursion therefore consists of (for $i = n$ to 1):

$$\mathbf{f}_{i-1,i} = \mathbf{f}_{i,i+1} + m_i \mathbf{a}_{ci} \tag{A.4a}$$

$$\mathbf{n}_{i-1,i} = \mathbf{n}_{i,i+1} - \mathbf{c}_{i,i} \times \mathbf{f}_{i,i+1} + \mathbf{c}_{i-1,i} \times \mathbf{f}_{i-1,i} + \mathbf{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i \boldsymbol{\omega}_i \tag{A.4b}$$

where $\mathbf{f}_{i-1,i}$ and $\mathbf{n}_{i-1,i}$ are, respectively, the force and moment applied by link $i-1$ to link i, m_i is the mass of link i, $\mathbf{I}_i$ is its centroid moment of inertia, while $\mathbf{c}_{i-1,i}$ is a vector from the origin of frame $i-1$ to the centroid of link i.

The final equation to be implemented is simply one which picks the component of $\mathbf{f}_{i-1,i}$ or $\mathbf{n}_{i-1,i}$ which corresponds to the actuator force or torque at that joint.

When joint i is revolute:

$$\tau_i = \mathbf{e}_{i-1}^T \mathbf{n}_{i-1,i} \tag{A.5a}$$

When joint i is prismatic:

$$\tau_i = \mathbf{e}_{i-1}^T \mathbf{f}_{i-1,i} \tag{A.5b}$$

Note that the remaining components of $\mathbf{f}_{i-1,i}$ and $\mathbf{n}_{i-1,i}$ constitute the constraint wrench acting at that joint.

A.2 Component Form of the Equations

The preceding equations were given in frame-invariant form. When they are implemented numerically, each vector must be resolved into its components in a particular frame. Of course, whenever an operation is performed between two vectors, they must both be expressed in the same frame. The equations of the preceding section are now given explicitly in component form along with all transformations which must be performed during the recursions.

The motion of the base of the manipulator is assumed to be given as components in frame 0 (the base frame)—$[\omega_0]_0$, $[\dot{\omega}_0]_0$ and $[\mathbf{a}_0]_0$—where the notation $[\star]_i$ implies that the vector inside the square brackets is expressed as components in frame i. Gravitational effects are most efficiently included in this model by adding the acceleration of gravity to the base acceleration $[\mathbf{a}_0]_0$. The outward recursion is described below:

For $i = 1$ to n,

When joint i is revolute:

$$[\omega_i]_i = \mathbf{Q}^T_{i-1,i}([\omega_{i-1}]_{i-1} + \dot{q}_i[\mathbf{e}_{i-1}]_{i-1}) \tag{A.6a}$$

$$[\dot{\omega}_i]_i = \mathbf{Q}^T_{i-1,i}([\dot{\omega}_{i-1}]_{i-1} + \ddot{q}_i[\mathbf{e}_{i-1}]_{i-1} + [\omega_{i-1}]_{i-1} \times \dot{q}_i[\mathbf{e}_{i-1}]_{i-1} \tag{A.6b}$$

$$[\mathbf{a}_i]_i = \mathbf{Q}^T_{i-1,i}[\mathbf{a}_{i-1}]_{i-1} + [\dot{\omega}_i]_i \times [\mathbf{r}_i]_i + [\omega_i]_i \times ([\omega_i]_i \times [\mathbf{r}_i]_i) \tag{A.6c}$$

When joint i is prismatic:

$$[\omega_i]_i = \mathbf{Q}^T_{i-1,i}[\omega_{i-1}]_{i-1} \tag{A.7a}$$

$$[\dot{\omega}_i]_i = \mathbf{Q}^T_{i-1,i}[\dot{\omega}_{i-1}]_{i-1} \tag{A.7b}$$

$$[\mathbf{a}_i]_i = \mathbf{Q}^T_{i-1,i}([\mathbf{a}_{i-1}]_{i-1} + \ddot{q}_i[\mathbf{e}_{i-1}]_{i-1} + 2[\omega_{i-1}]_{i-1} \times \dot{q}_i[\mathbf{e}_{i-1}]_{i-1}) + [\dot{\omega}_i]_i \times [\mathbf{r}_i]_i + [\omega_i]_i \times ([\omega_i]_i \times [\mathbf{r}_i]_i) \tag{A.7c}$$

The orthogonal rotation matrix $\mathbf{Q}^T_{i-1,i}$ transforms components in frame $i-1$ into components in frame i. It represents the composite rotation about the joint angle θ_i and about the link's twist angle α_i (Hartenberg and Denavit, 1964), and can be written as

$$\mathbf{Q}^T_{i-1,i} = \begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 \\ -\cos\alpha_i \sin\theta_i & \cos\alpha_i \cos\theta_i & \sin\alpha_i \\ \sin\alpha_i \sin\theta_i & -\sin\alpha_i \cos\theta_i & \cos\alpha_i \end{bmatrix} \tag{A.8}$$

Since $\mathbf{Q}^T_{i-1,i}$ is orthogonal, its inverse is equal to its transpose. Therefore, the matrix which transforms components in frame i into components in frame $i-1$ is nothing

but $\mathbf{Q}_{i-1,i}$. Furthermore, it should be noted that the vector $\mathbf{e}_i$ expressed as components in frame i is nothing but $[\mathbf{e}_i]_i = [0 \quad 0 \quad 1]^T$ thereby making any operation in which $\mathbf{e}_i$ is involved trivial.

For both types of joints,

$$[\mathbf{a}_{ci}]_i = [\mathbf{a}_i]_i + [\dot{\boldsymbol{\omega}}_i]_i \times [\mathbf{c}_{i,i}]_i + [\boldsymbol{\omega}_i]_i \times ([\boldsymbol{\omega}_i]_i \times [\mathbf{c}_{i,i}]_i) \tag{A.9}$$

The component form of the inward recursion equations are (for $i = n$ to 1):

$$[\mathbf{f}_{i-1,i}]_{i-1} = \mathbf{Q}_{i-1,i}([\mathbf{f}_{i,i+1}]_i + m_i[\mathbf{a}_{ci}]_i) \tag{A.10a}$$

$$\begin{aligned}[\mathbf{n}_{i-1,i}]_{i-1} &= \mathbf{Q}_{i-1,i}([\mathbf{n}_{i,i+1}]_i - [\mathbf{c}_{i,i}]_i \times [\mathbf{f}_{i,i+1}]_i + [\mathbf{c}_{i-1,i}]_i \times \mathbf{Q}_{i-1,i}^T[\mathbf{f}_{i-1,i}]_{i-1} \\ &\quad + [\mathbf{I}_i]_i[\dot{\boldsymbol{\omega}}_i]_i + [\boldsymbol{\omega}_i]_i \times [\mathbf{I}_i]_i[\boldsymbol{\omega}_i]_i)\end{aligned} \tag{A.10b}$$

Finally, when joint i is revolute:

$$\tau_i = [\mathbf{e}_{i-1}^T]_{i-1}[\mathbf{n}_{i-1,i}]_{i-1} \tag{A.11a}$$

and when joint i is prismatic:

$$\tau_i = [\mathbf{e}_{i-1}^T]_{i-1}[\mathbf{f}_{i-1,i}]_{i-1} \tag{A.11b}$$

When the above equations are implemented for a six-link manipulator ($n = 6$), and the operations involving $[\mathbf{e}_i]_i$ are appropriately reduced due to the particular form of that vector, the complete inverse dynamics problem can be solved with 810 scalar multiplications and 684 scalar additions.

Appendix B

The Quadratic-Programming Algorithm

This appendix gives the listing of the quadratic-programming algorithm developed in the present work. The first subroutine—**quad**—is called by a main program with the following arguments:

- `c` The vector **c** in eq.(5.9) of dimension `n`,
- `a` The composite matrix $\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}$ in eqs.(5.4a) and (5.4b) of dimension `m` $\times$ `n`,
- `b` The composite vector $\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$ in eqs.(5.4a) and (5.4b) of dimension `m`,
- `x` The vector of design variables **x** of dimension `n`,
- `n` The dimension of the vector of design variables,
- `m` The total number of equality and inequality constraints,
- `meq` The number of equality constraints,
- `ia` The row dimension of `a` and `Linv` in the calling program, and
- `Linv` The `n` $\times$ `n` Cholesky decomposition of the weighting matrix **W** in eq.(5.9).

Subroutine **quad** will make use of the 10 subroutines which follow it in the listing. The last subroutine given in the listing performs the Cholesky decomposition of a matrix and can be used by the main program to obtain the matrix `Linv` which would subsequently be passed to subroutine **quad**.

The listing is included in the following pages.

```
      subroutine quad (c, a, b, x, n, m, meq, ia, Linv)
c
c  Routine for quadratic programming using the method of Goldfarb
c  and Idnani (1983), modified to include equality constraints.
c  n.b. Some local arrays are dimensioned 100.  If either m or n
c  is larger than 100, these must be redimensioned.
c
c  Calling arguments:
c   c    = Linear weighting vector of dimension n (input)
c   a    = Matrix of constraints of dimension m X n (input)
c          The first meq rows correspond to equality constraints;
c          The last (m-meq) rows correspond to inequality constraints
c   b    = Vector of rhs of constraint equations of dimension m (input)
c   x    = Vector of design variables of dimension n (output)
c   n    = Dimension of vector of design variables (input)
c   m    = Total number of constraints (equality + inequality) (input)
c   meq  = Number of equality constraints (input)
c   ia   = Row dimension of a and Linv in calling program (input)
c   Linv = Matrix inverse of Cholesky decomposition of quadratic
c          weighting matrix of dimension n X n (input)
c
c  c = c,      a =  | A_1 |,    b =  | b_1 |
c                   | A_2 |          | b_2 |
c
      implicit undefined (a-z)
      real*8 Linv(ia,n), u(100), up(100), s(100), a(ia,n), x(n), b(m)
      real*8 np(100), z(100), d(100), rr(100,100), c(n), x2(100)
      real*8 jt(100,100), temp(100,100), r(100), norm(100), uu(100)
      real*8 nn(100,100), Linvc(100), Linvt(100,100)
      real*8 t1, t2, t, ztnp, test, eps, sum
      integer*4 ia, iloc, m, n, p, q, i, j, l, aa(100), meq, dummy
      integer*4 ier
      logical active(100), found
c
      data iloc/100/ , eps/1.d-20/
c
c Initialize--do only at start of each new problem (Step 0):
c Transpose A to get N and premultiply it by Linv:
      call transp (a, ia, nn, iloc, meq, n)
      call mult (Linv, ia, nn, iloc, rr, iloc, n, n, meq)
      call multvec (Linv, ia, c, Linvc, n, n)
      call transp (Linv, ia, Linvt, iloc, n, n)
      call multvec (Linvt, iloc, Linvc, x2, n, n)
      do 22 i = 1, meq
        sum = 0.d0
        do 23 j = 1, n
   23     sum = sum + a(i,j)*x2(j)
   22   d(i) = sum + b(i)
c Perform Householder reduction on N to get R, and find min-norm soln:
      call hhred2 (rr, iloc, n, meq, uu, ier)
      if (ier.ne.0) write (*,*) 'ier = ', ier
      call trfor2 (rr, iloc, meq, d, ier)
      if (ier.ne.0) write (*,*) 'ier = ', ier
      do 3 i = meq + 1, n
    3   d(i) = 0.d0
      call hhbak2 (rr, iloc, n, meq, uu, d)
      call multvec (Linvt, iloc, d, x, n, n)
      do 4 i = 1, n
    4   x(i) = x(i) - x2(i)
      if (m.eq.meq) return
c Store V^(-1) in jt and premultiply it by Q^T to get J^T:
      do 5 i = 1, n
        do 6 j = 1, n
    6     jt(i,j) = Linv(i,j)
    5 continue
      do 7 i = 1, n
```



```
    7   call hhfor2 (rr, iloc, n, meq, uu, jt(1,i))
c Put zeroes in below the main diagonal of R:
      do 8 i = 2, meq
        do 8 j = 1, i-1
    8     rr(i,j) = 0.d0
      do 9 i = meq+1, n
        do 9 j = 1, meq
    9     rr(i,j) = 0.d0
      do 10 i = meq+1, m
        u(i) = 0.d0
   10   up(i) = 0.d0
      do 30 i = meq+1, m
        aa(i) = 0
   30   active(i) = .false.
      q = meq
      do 40 i = meq+1, m
        sum = 0.d0
        do 45 j = 1, n
   45     sum = sum + a(i,j)*a(i,j)
   40   norm(i) = 1.d0/sqrt(sum)
c
c Check inactive constraints (Step 1):
 1000 call check (a, ia, x, b, s, m, n, meq, active, found, p, norm)
      if (.not.found) then
        do 50 i = 1, n
   50     np(i) = a(p,i)
        do 60 i = meq+1, q
   60     up(i) = u(i)
        up(q+1) = 0.d0
c
c Determine a new S-pair (Step 2):
c (Step 2a):
 2000   if (q.eq.0) then
          do 70 i = meq+1, m
   70       u(i) = 0.d0
        endif
        do 80 i = 1, n
          sum = 0.d0
          do 85 j = 1, n
   85       sum = sum + jt(i,j)*np(j)
   80     d(i) = sum
        do 90 i = 1, n
          sum = 0.d0
          do 95 j = q+1, n
   95       sum = sum + jt(j,i)*d(j)
   90     z(i) = sum
        if (q.gt.0) then
          do 100 i = q, 1, -1
            sum = 0.d0
            do 110 j = i+1, q
  110         sum = sum + rr(i,j)*r(j)
  100       r(i) = (d(i) - sum)/rr(i,i)
        endif
c
c (Step 2bi):
        t1 = 1.d60
        do 120 j = meq+1, q
          if (r(j).gt.0.d0) then
            test = up(j)/r(j)
            if (test.lt.t1) then
              l = j
              t1 = test
            endif
          endif
  120   continue
c
```

```
c (Step 2bii):
        sum = 0.d0
        do 130 i = 1, n
  130     sum = sum + z(i)*z(i)
        if (sum.lt.eps) then
           t2 = 1.d60
        else
           ztnp = 0.d0
           do 140 i = 1, n
  140        ztnp = ztnp + z(i)*np(i)
           t2 = - s(p)/ztnp
        endif
c
c (Step 2biii):
        if (t1.lt.t2) then
           t = t1
        else
           t = t2
        endif
c
c (Step 2ci)
        if (t.eq.1.d60) then
           write (*,*) 'The QPP is not feasible'
           stop
        endif
c
c (Step 2cii):
        if (t2.eq.1.d60) then
           do 150 i = meq+1, l-1
  150        up(i) = up(i) - t*r(i)
           do 160 i = l+1, q
  160        up(i-1) = up(i) - t*r(i)
           up(q) = t
           if (l.ne.q)
     *        call remove (rr, iloc, jt, temp, iloc, l, n, q)
           active(aa(l)) = .false.
           do 170 i = l, q-1
  170        aa(i) = aa(i+1)
           aa(q) = 0
           q = q - 1
           call check (a, ia, x, b, s, m, n, meq, active, found,
     *                 dummy, norm)
           go to 2000
        endif
c
c (Step 2ciii):
        do 180 i = 1, n
  180     x(i) = x(i) + t*z(i)
        if (t.eq.t2) then
           do 190 i = meq+1, q
  190        up(i) = up(i) - t*r(i)
           up(q+1) = t
           do 200 i = meq+1, q+1
  200        u(i) = up(i)
           call adjoin (d, rr, iloc, jt, iloc, n, q)
           q = q + 1
           active(p) = .true.
           aa(q) = p
           go to 1000
        else
           do 210 i = meq+1, l-1
  210        up(i) = up(i) - t*r(i)
           do 220 i = l+1, q
  220        up(i-1) = up(i) - t*r(i)
           up(q) = t
           if (l.ne.q)
```



```
     *        call remove (rr, iloc, jt, temp, iloc, l, n, q)
           active(aa(l)) = .false.
           do 230 i = l, q-1
  230        aa(i) = aa(i+1)
           aa(q) = 0
           q = q - 1
           call check (a, ia, x, b, s, m, n, meq, active, found,
     *                 dummy, norm)
           go to 2000
        endif
c
c Solution Found:
        else
           return
        endif
c
        end
c
c*******************************************************************
c
        subroutine check (a, ia, x, b, s, m, n, meq, active, found, p,
     *                    norm)
c
c Routine to check the inequality constraints:
c
        implicit undefined (a-z)
        real*8 a(ia,*), x(*), b(*), s(*), norm(*)
        real*8 eps, smallest, test, sum
        integer*4 m, n, meq, i, j, ia, p
        logical active(*), found
        data eps/-1.d-06/
c
        found = .true.
        smallest = eps
        do 10 i = meq+1, m
          if (.not.active(i)) then
             sum = -b(i)
             do 20 j = 1, n
  20           sum = sum + a(i,j)*x(j)
             s(i) = sum
             test = sum*norm(i)
             if (test.lt.smallest) then
                found = .false.
                smallest = test
                p = i
             endif
          endif
  10    continue
c
        return
        end
c
c*******************************************************************
c
        subroutine adjoin (d, rr, ic, jt, ia, n, q)
c
c Routine to add an active constraint
c
c  inputs:
c     d      Previous vector d used to update rr
c     rr     Previous upper-triangular reduction of constraints--updated
c     jt     Transpose of previous matrix J--updated on output
c     ic     Row dimension of rr in calling program
c     ia     Row dimension of jt in calling program
c     n      order of jt
c     q      Previous number of active constraints
```

```
c
      implicit undefined (a-z)
      real*8 rr(ic,*), jt(ia,*), d(*)
      real*8 beta, delta, u(100), t, gamma
      integer*4 ic, ia, n, q, i, j
c
c Perform Householder reduction on vector d2:
      delta = 0.d0
      do 10 i = 1, n-q
        u(i) = d(i+q)
   10   delta = delta + u(i) * u(i)
      delta = sqrt(delta)
      if (u(1).lt.0.d0) delta = -delta
      u(1) = u(1) + delta
      beta = delta*u(1)
      d(q+1) = -delta
      if (beta.eq.0.d0) write (*,*) 'ier = ', -1
c
c Update R:
      do 20 i = 1, q
   20   rr(i,q+1) = d(i)
      rr(q+1,q+1) = -delta
c
c update J^T by applying the reduction matrix to J_2^T:
      do 30 i = 1, n
        t = d(q+1)
        if (t.ne.0.d0) then
           beta = u(1)*t
           d(q+1) = u(1)
           gamma = 0.d0
           do 40 j = q+1, n
   40        gamma = gamma + d(j) * jt(j,i)
           gamma = gamma/beta
           do 50 j = q+1, n
   50        jt(j,i) = jt(j,i) + gamma * d(j)
        endif
        d(q+1) = t
   30 continue
c
      return
      end
c
c*********************************************************************
c
      subroutine remove (rr, ic, jt, temp, ia, l, n, q)
c
c Routine to remove an active constraint
c
c  inputs:
c     rr      Previous upper-triangular reduction of constraints--updated
c     jt      Transpose of previous matrix J--updated on output
c     temp    Work matrix at least n x n.
c     ic      Row dimension of rr in calling program
c     ia      Row dimension of jt and temp in calling program
c     l       Index of the constraint to be removed
c     n       order of jt
c     q       Previous number of active constraints
c
      implicit undefined (a-z)
      real*8 rr(ic,*), jt(ia,*), temp(ia,*), u(100)
      integer*4 ic, ia, l, n, q, i, j, ier
c
      do 10 i = 1, q-l+1
        do 10 j = 1, q-l
   10     temp(i,j) = rr(i+l-1, j+1)
c
```



```
      call hhred2 (temp, ia, q-l+1, q-l, u, ier)
      if (ier.ne.0) write (*,*) 'ier = ', ier
c
c Update R:
      do 20 i = 1, l-1
        do 20 j = 1, q-1
   20     rr(i,j) = rr(i,j+1)
c
      do 30 i = 1, q-1
        do 30 j = i, q-1
   30     rr(i,j) = temp(i-l+1,j-l+1)
c
c Update J:
      do 40 j = 1, n
   40   call hhfor2 (temp, ia, q-l+1, q-l, u, jt(1,j))
c
      return
      end
c
c*********************************************************************
c
      subroutine mult (a, ma, b, mb, c, mc, m, nn, n)
      implicit undefined (a-z)
      real*8 a(ma,*), b(mb,*), c(mc,*), sum
      integer*4 ma, mb, mc, m, nn, n, i, j, k
c
      do 10 i = 1, m
        do 20 j = 1, n
          sum = 0.d0
          do 30 k = 1, nn
   30       sum = sum + a(i,k)*b(k,j)
          c(i,j) = sum
   20   continue
   10 continue
c
      return
      end
c
c*********************************************************************
c
      subroutine multvec (a, ma, b, c, m, n)
c
c     Subroutine to multiply a matrix times a vector
c
      implicit undefined (a-z)
      real*8 a(ma,*), b(*), c(*), sum
      integer*4 ma, m, n, i, j
c
      do 10 i = 1, m
        sum = 0.d0
        do 20 j = 1, n
   20     sum = sum + a(i,j)*b(j)
        c(i) = sum
   10 continue
c
      return
      end
c
c*********************************************************************
c
      subroutine transp (a, ma, b, mb, m, n)
      implicit undefined (a-z)
      real*8 a(ma,*), b(mb,*)
      integer*4 ma, mb, m, n, i, j
c
      do 10 i = 1, m
```

```
        do 10 j = 1, n
          b(j,i) = a(i,j)
   10   continue
c
        return
        end
c
c*********************************************************************
c
      subroutine hhred2 (a, ia, m, n, u, ierr)
c
c     Matrix Householder Decomposition
c
c  inputs:
c       a       m by n matrix for which the householder decomposition
c               is required (destroyed)
c     ia        row dimension of a as declared in the calling segment
c     m, n      dimensions of matrix a
c
c  outputs:
c     ierr      error flag (-1 if a is singular)
c     aij       upper-triangular reduction of a (ie. u) for i =< j
c               ith component of jth reflection vector for i > j
c     uj        jth component of jth reflection vector for j =< n
c
      implicit undefined (a-z)
      real*8 a(ia,*), u(*), alpha, beta, gamma
      integer*4 ia, m, n, ierr, i, j, k
c
      ierr = 0
      do 100 k = 1, n
        alpha=0.d0
        do 10 i = k, m
          u(i) = a(i,k)
   10     alpha = alpha + u(i) * u(i)
        alpha = sqrt(alpha)
        if (u(k).lt.0.d0) alpha = -alpha
        u(k) = u(k) + alpha
        beta = alpha*u(k)
        a(k,k) = -alpha
        if (beta.eq.0.d0) then
           ierr = -1
           go to 100
        endif
        if (k.ne.n) then
           do 20 j = k+1, n
             gamma = 0.d0
             do 30 i = k, m
   30          gamma = gamma + u(i) * a(i,j)
             gamma = -gamma/beta
             do 20 i = k, m
   20          a(i,j) = gamma * u(i) + a(i,j)
        endif
  100 continue
c
      return
      end
c
c*********************************************************************
c
      subroutine hhfor2 (a, ia, m, n, u, b)
c                                                    -1
c     Matrix Householder Forwards Multiplication h    b
c
c  Inputs:
c     aij     upper-triangular reduction of a (ie. u) for i =< j
```



```
c               ith component of jth reflection vector for i > j
c     ia        row dimension of a as declared in the calling segment
c     n         order of matrix a
c     uj        jth component of jth reflection vector for j =< n
c     bij       ith element of jth b-vector i < n and j < p
c
c  Outputs:                                    -1
c     bij       ith element of jth h    b -vector i < n and j < p
c
      implicit undefined (a-z)
      real*8 a(ia,*), u(*), b(*), t, betan, gamma
      integer*4 ia, m, n, i, k
c
      do 10 k = 1, n
        t = a(k,k)
        if (t.ne.0.d0) then
           betan = u(k)*t
           a(k,k) = u(k)
           gamma = 0.d0
           do 20 i = k, m
   20        gamma = gamma + a(i,k) * b(i)
           gamma = gamma/betan
           do 30 i = k, m
   30        b(i) = b(i) + gamma * a(i,k)
        endif
        a(k,k) = t
   10 continue
c
      return
      end
c
c*********************************************************************
c
      subroutine hhbak2 (a, ia, m, n, u, b)
c                                                    -t
c     Matrix Householder Backwards Multiplication h    b
c
c  Inputs:
c     aij       upper-triangular reduction of a (ie. u) for i =< j
c               ith component of jth reflection vector for i > j
c     ia        row dimension of a as declared in the calling segment
c     n         order of matrix a
c     uj        jth component of jth reflection vector for j =< n
c     bij       ith element of jth b -vector i < n and j < p
c
c  Outputs:                                    -t
c     bij       ith element of jth h    b -vector i < n and j < p
c
      implicit undefined (a-z)
      real*8 a(ia,*), u(*), b(*), t, betan, gamma
      integer*4 ia, m, n, k, l
c
      do 10 k = n, 1, -1
        t = a(k,k)
        if (t.ne.0.d0) then
           betan = u(k)*t
           a(k,k) = u(k)
           gamma = 0.d0
           do 20 l = k, m
   20        gamma = gamma + a(l,k) * b(l)
           gamma = gamma/betan
           do 30 l = k, m
   30        b(l) = b(l) + gamma * a(l,k)
        endif
        a(k,k)=t
   10 continue
```

```
c
      return
      end
c
c*********************************************************************
c
      subroutine trfor2 (a, ia, n, b, ierr)
c                                                    -t
c     Upper Triangular Matrix Forwards Elimination u   b
c
c  Inputs:
c     aij      upper-triangular matrix a for i =< j
c              (only the upper triangle need be given)
c     ia       row dimension of a as declared in the calling segment
c     n        order of matrix a
c     bij      ith element of jth b -vector i < n and j < p
c
c  Outputs:                          -t
c     bij      ith element of jth u  b  -vector i < n and j < p
c     ierr     error flag (-1 if a singular)
c
      implicit undefined (a-z)
      real*8 a(ia,*), b(*), t
      integer*4 ia, n, ierr, i, k, km1
c
      ierr = -1
      do 10 k = 1, n
        km1 = k - 1
        if (a(k,k).eq.0.d0) return
        t = 0.d0
        if (km1.ne.0) then
           do 20 i =1, km1
   20        t = t + a(i,k) * b(i)
        endif
        b(k) = (b(k) - t)/a(k,k)
   10 continue
      ierr = 0
c
      return
      end
c
c*********************************************************************
c
      subroutine decomp (a, b, ia, m)
c
c     Cholesky Decomposition Algorithm :
c
c     If (a) is a positive definite mxm matrix, then (a) has a
c     factorization of the form a = bbt(t = transpose),where b
c     is a lower-triangular matrix.
c
c     Input :  matrix (a) dimension mxm
c
c     Output : matrix (b) dimension mxm (lower-triangular)
c
      implicit undefined (a-z)
      real*8 a(ia,*), b(ia,*), g, gg, f, ff
      integer ia, m, i, j, k, ii, c1
c
      do 5 i = 1, m
        do 5 j = 1, m
          b(i,j) = 0.d0
 5    continue
c
      b(1,1) = sqrt(a(1,1))
      do 10 i = 2, m
```



```
        b(i,1) = a(i,1)/b(1,1)
 10   continue
c
      ii = 3
      do 15 j = 2, m
        c1 = j - 1
        gg = 0.d0
        do 20 k = 1, c1
          g = b(j,k)*b(j,k)
          gg = gg + g
 20     continue
c
        b(j,j) = sqrt(a(j,j) - gg)
        if (j.ne.m) then
           do 25 i = ii, m
             ff = 0.d0
             do 30 k = 1, c1
               f = b(i,k) * b(j,k)
               ff = ff + f
   30        continue
             b(i,j) = (a(i,j) - ff)/b(j,j)
 25        continue
           ii = ii + 1
        endif
 15   continue
c
      return
      end
```