INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17” x 23” black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Efficient dynamic simulation of multiple chain robotic systems

Lilly, Kathryn Weed, Ph.D.
The Ohio State University, 1989

Copyright ©1989 by Lilly, Kathryn Weed. All rights reserved.
EFFICIENT DYNAMIC SIMULATION OF MULTIPLE CHAIN ROBOTIC SYSTEMS

A Dissertation
Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the
Graduate School of the Ohio State University

by

Kathryn W. Lilly, B.S.E.E, M.S.E.E.

The Ohio State University
1989

Dissertation Committee:
Professor David E. Orin
Professor Charles A. Klein
Professor Kenneth J. Waldron

Approved by:
Adviser
Department of Electrical Engineering
To my husband
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. David E. Orin, for his guidance, support, and encouragement throughout my doctoral work. I would also like to thank Dr. Kenneth J. Waldon and Dr. Charles A. Klein for their careful reading of this dissertation and for their help and interest along the way. A special note of thanks belongs to Dr. Ulrich Gerlach for his help in the analysis of the operational space inertia matrix.

It is difficult to fully express my gratitude for the love and support that my entire family has given me throughout my educational career. They have listened, encouraged, and advised, and I am proud to be part of them.

Of course, my deepest appreciation is for my husband, Blaine, who is my best friend and partner for life. His love, his patience, and his faith in me are always sources of strength and great comfort.

Finally, I would like to recognize the Fellowship support I received through the GPOP Program and the Presidential Fellowship Program at The Ohio State University. This research was also supported in part by the National Science Foundation under Computational Engineering Grant No. EET-8718434.
VITA

June 11, 1962 ......................... Born—Columbus, Ohio, U.S.A.

1979–1983 ......................... Battelle Scholar
The Ohio State University
Columbus, Ohio, U.S.A.

1983 ................................. B.S.E.E.
The Ohio State University
Columbus, Ohio, U.S.A.

1983–1986 ......................... GPOP Fellow
The Ohio State University
Columbus, Ohio, U.S.A.

1985 ................................. M.S.E.E.
The Ohio State University
Columbus, Ohio, U.S.A.

1986–1988 ......................... Graduate Research Associate
The Ohio State University
Columbus, Ohio, U.S.A.

Fall 1986 ............................. Graduate Teaching Associate
The Ohio State University
Columbus, Ohio, U.S.A.

1988–1989 ......................... Presidential Fellow
The Ohio State University
Columbus, Ohio, U.S.A.
PUBLICATIONS


FIELDS OF STUDY

Major Field: Electrical Engineering

Studies in Robotics: Professors D. E. Orin,
K. J. Waldron, K. H. Hunt,
C. A. Klein

Studies in Computer Engineering: Professors K. W. Olson,
K. J. Breeding, F. Özgüner

Studies in Control Engineering: Professors R. E. Fenton,
R. Moses, Ü. Özgüner,
S. Yurkovich

Studies in Mechanical Engineering: Professors G. L. Kinzel,
E. O. Doebelin, K. Srinivasan
# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS**

**VITA**

**LIST OF FIGURES**

**LIST OF TABLES**

## I. INTRODUCTION

1.1 The Dynamic Simulation Problem .............................................. 1
1.2 Direct Dynamics for Robotic Systems ....................................... 4
1.3 Scope and Contents ............................................................... 5
1.4 Organization ............................................................................. 8

## II. SYSTEM MODELLING AND NOTATION

2.1 Introduction .............................................................................. 11
2.2 Kinematic and Dynamic Parameters of a Robotic Mechanism ..... 12
2.3 General Joint Model ................................................................. 17
2.3.1 Examples ............................................................................. 20
2.4 Summary ................................................................................... 22
III. ALTERNATE FORMULATIONS FOR THE JOINT SPACE INERTIA MATRIX

3.1 Introduction ................................................................. 24
3.2 Previous Work .............................................................. 26
3.3 Additional Notation and Background ............................... 27
3.4 Structurally Recursive Method ....................................... 29
3.5 Inertia Projection Method .............................................. 34
3.6 Modified Composite-Rigid-Body Method ......................... 38
3.7 Spatial Composite-Rigid-Body Method ............................. 39
3.8 Efficient Transformation of Spatial Quantities .................. 42
  3.8.1 Transforming Spatial Vectors ................................. 42
  3.8.2 Transforming Spatial Rigid-Body Inertias .................. 44
  3.8.3 Computing Spatial Composite-Rigid-Body Inertias .... 46
3.9 Computational Requirements ......................................... 46
3.10 Conclusions and Summary .......................................... 49

IV. ALTERNATE FORMULATIONS FOR THE OPERATIONAL SPACE INERTIA MATRIX

4.1 Introduction ................................................................. 50
4.2 Previous Work .............................................................. 52
4.3 The Operational Space Formulation ................................. 54
4.4 The Operational Space Inertia Matrix ............................... 55
4.5 The Explicit Inversion/Multiplication Method .................... 58
4.6 The Unit Force Method ................................................ 61
4.7 The Force Propagation Method ....................................... 64
  4.7.1 Recursive Dynamic Equations for an Open Chain .... 66
4.7.2 Recursive Dynamic Equations for a Constrained Chain 72
4.7.3 Development of the Force Propagation Method 75
4.8 The Inertia Propagation Method 82
4.9 Computational Requirements 92
4.10 Conclusions and Summary 94

V. EFFICIENT DYNAMIC SIMULATION OF A SINGLE CLOSED CHAIN 96

5.1 Introduction 96
5.2 Previous Work 98
5.2.1 Direct Dynamics for Single Open Chains 98
5.2.2 Direct Dynamics for Single Closed Chains 99
5.3 Equations of Motion and Joint Accelerations 101
5.4 Acceleration of the End Effector 103
5.5 Tip Constraints 104
5.5.1 Contact Model 105
5.5.2 Classes of Contacts 106
5.5.3 Examples 107
5.6 Dynamic Simulation Algorithm for a Single Closed Chain 114
5.6.1 Step 1. The Open Chain Solution 116
5.6.2 Step 2. Calculation of the Unknown Contact Forces 117
5.6.3 Step 3. Calculation of the Closed-Chain Joint Accelerations 124
5.6.4 Step 4. Integration for the Next State 124
5.7 Computational Requirements 126
5.8 Conclusions and Summary 130
VI. EFFICIENT DYNAMIC SIMULATION OF SIMPLE CLOSED-CHAIN MECHANISMS

6.1 Introduction ................................................................. 132
6.2 Previous Work ............................................................. 134
6.3 System Configuration and Problem Statement .............. 135
   6.3.1 Model for Simple Closed-Chain Mechanisms ......... 135
   6.3.2 General Problem Statement ............................... 138
6.4 System Dynamic Equations .......................................... 138
   6.4.1 Dynamic Equations of Motion for Each Chain .... 139
   6.4.2 Dynamic Equation for the Reference Member ...... 140
6.5 Dynamic Simulation Algorithm for Simple Closed-Chain
   Mechanisms ................................................................. 142
   6.5.1 The Spatial Acceleration of the Reference Member .. 148
   6.5.2 The Chain Tip Forces and Closed-Chain Joint
          Accelerations ...................................................... 150
   6.5.3 The Simulation Algorithm in Five Steps ............. 151
6.6 An Alternate Form of the Simulation Algorithm ............ 153
6.7 Computational Requirements ........................................ 157
   6.7.1 Number of Computations .................................... 158
   6.7.2 Parallel Implementation of the Simulation Algorithm .. 161
6.8 Conclusions and Summary ............................................. 162

VII. SUMMARY AND CONCLUSIONS .................................... 164
7.1 Summary ................................................................. 164
7.2 Future Work ............................................................ 166

REFERENCES ................................................................. 169
# LIST OF FIGURES

1. Robotic Simulation and Control Block Diagram ................................ 3
2. Models for the Two Types of Simple Closed-Chain Mechanisms ........ 6
3. Parameters Relating Adjacent Link Coordinate Systems ................. 13
4. One-link Manipulator ...................................................................... 30
5. Two-link Manipulator ..................................................................... 31
6. Operational Space Model of an Actuated Chain ................................. 55
7. Serial-Link Chain for the Force Propagation Method ....................... 66
8. Two-Link Articulated Body ............................................................. 67
9. Serial-Link Chain for the Inertia Propagation Method ..................... 82
10. Example of a Single Closed-Chain System ...................................... 101
11. Rigid Connection Between Tip and Inertial Frame .......................... 108
12. Hard Point Contact with Friction .................................................... 110
13. Example of a Type 0 Simple Closed-Chain Mechanism .................... 136
14. Example of a Type 1 Simple Closed-Chain Mechanism .................... 137
# LIST OF TABLES

1. Algorithm for the Structurally Recursive Method (Method I)  . . . 35
2. Algorithm for the Inertia Projection Method (Method II)  . . . 37
3. Algorithm for the Modified Composite-Rigid-Body Method (Method III)  . . . 40
4. Algorithm for the Spatial Composite-Rigid-Body Method (Method IV)  . . . 41
5. Computations for $X^TIX$  . . . 45
6. Computations for the Modified Composite-Rigid-Body Method  . . 47
7. Number of Computations for the Joint Space Inertia Matrix  . . . 48
8. Algorithm for the Explicit Inversion/Multiplication Method  . . . 59
9. Computations for $\Lambda$ and $\Lambda^{-1}$, Explicit Inversion/Multiplication Method  . . . 60
10. Algorithm for the Unit Force Method  . . . 63
11. Computations for $\Lambda$ and $\Lambda^{-1}$, Unit Force Method  . . 65
12. Algorithm for the Force Propagation Method  . . . 80
13. Computations for $\Lambda$ and $\Lambda^{-1}$, Force Propagation Method  . . . 81
14. Additional Equations for $\Omega$, Force Propagation Method  . . . 83
15. Additional Computations for $\Omega$, Force Propagation Method  . . . 84
16. Algorithm for the Inertia Propagation Method  . . . 90
17. Computations for $\Lambda$ and $\Lambda^{-1}$, Inertia Propagation Method  . . . 91
<table>
<thead>
<tr>
<th>Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Comparison of Computations for $\Lambda$ and $\Lambda^{-1}$</td>
<td>93</td>
</tr>
<tr>
<td>19</td>
<td>Computations for the Single Closed-Chain Dynamic Simulation Algorithm, $O(N^3)$</td>
<td>127</td>
</tr>
<tr>
<td>20</td>
<td>Computations for the Single Closed-Chain Dynamic Simulation Algorithm, $O(N)$</td>
<td>128</td>
</tr>
<tr>
<td>21</td>
<td>Dynamic Simulation Algorithm for Simple Closed-Chain Mechanisms</td>
<td>152</td>
</tr>
<tr>
<td>22</td>
<td>Computations Per Chain in the Simple Closed-Chain Dynamic Simulation Algorithm</td>
<td>159</td>
</tr>
<tr>
<td>23</td>
<td>Computations for the Spatial Acceleration of the Reference Member</td>
<td>161</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

1.1 The Dynamic Simulation Problem

The dynamics of a robotic system play a significant role both in its control and simulation. Thus, as the tasks performed by robots become more complex, the need for a better understanding of their dynamic behavior increases. A thorough and complete dynamic analysis is often difficult, however, as well as computationally intensive. Many researchers continue to search for better ways to describe the behavior of robotic mechanisms in an accurate, yet simple, form. The work presented here contributes to this effort.

When studying the control of robots and other similar mechanisms, the primary problem which must be solved is known as Inverse Dynamics. Inverse Dynamics requires the calculation of the actuator torques and/or forces which will produce a prescribed motion in the robotic system. The present state joint positions and rates are known, and the desired joint accelerations are given. An Inverse Dynamics algorithm computes the driving actuator torques and/or forces which will produce the desired joint motion. Through the efforts of well-known researchers, efficient serial solutions to the Inverse Dynamics problem already exist for both open-chain [1,2] and closed-chain configurations [3]. The design of parallel algorithms is also of current interest [4,5].

In the area of simulation, the fundamental problem to be solved is called
Forward or Direct Dynamics. This problem requires the determination of the joint motion which results from a given set of applied joint torques and/or forces. Again, the present state joint positions and rates are known. A Direct Dynamics algorithm calculates the joint accelerations which result from the application of the given actuator forces. Once the accelerations are known, integration is used to determine the next state joint rates and positions for the simulation.

Simulation is an important tool in mechanical design and in the design and testing of advanced control algorithms. Figure 1 shows the basic computational blocks necessary for testing a proposed control algorithm in conjunction with a simulated model of the target system. The Direct Dynamics and Motion Integration blocks provide the target system simulation. The remaining blocks include the control algorithm being tested and the modules necessary for performance evaluation.

In the past, the simulation of robotic mechanisms has been considered strictly an off-line problem, and for the design applications described above, this assumption is appropriate. However, the value of on-line simulation has recently been demonstrated in a number of cases, including the earth-based teleoperation of remote robotic systems in space. Because of the significant time delays involved in signal transmission, the returned image of the robotic system which the earth-bound human operator must use is delayed, making direct control difficult. Preliminary results [6] indicate a considerable improvement in performance when a graphical display of the simulated system (kinematic in this case) is also made available to the operator, on-line, with no time delay. Many of the current teleoperated systems are relatively slow, so that a kinematic model may be sufficient. But for good performance at higher speeds or for force control, a dynamic simulation at real-time rates is necessary.
Figure 1: Robotic Simulation and Control Block Diagram [13]
It has also been suggested that “super” real-time simulation may be used for trajectory planning within advanced control schemes [7]. That is, if the simulation of a robotic system could be accomplished in a fraction of the time needed for the determination of the correct control inputs, the effects of alternate trajectories could be previewed and optimized as a part of the control algorithm itself. Such a capability would naturally improve the overall performance of the entire system.

Previous research in the dynamic simulation of robotic mechanisms includes the examination of both open-chain mechanisms [8,9,10,11] and closed-chain configurations [12,13,14,15,16,17]. Although many of these earlier results are useful and important, further improvements in the computational efficiency of dynamic simulation algorithms are necessary for real-time implementation.

1.2 Direct Dynamics for Robotic Systems

Although the details may be quite different, every research effort in the area of dynamic simulation faces a common task — the efficient and accurate solution of the Direct Dynamics problem. In the development of algorithms for Direct Dynamics, two basic approaches have emerged for both open- and closed-chain systems. The first utilizes the inversion of the $N \times N$ manipulator joint space inertia matrix to solve for the joint accelerations. More accurately, the accelerations are found via the solution of a system of linear algebraic equations, but the method is often referred to as “matrix inversion”. In general, an algorithm based on the inversion of an $N \times N$ matrix has a computational complexity of at least $O(N^3)$. Thus, this first approach may be computationally intensive, especially for systems with a large number of degrees of freedom, and some researchers have criticized it for this reason. It does provide a useful perspective on the problem, however, and remains a viable choice. Various forms of this approach have been applied to
serial open chains [9,11], single closed chains [12,17,18], and simple closed-chain mechanisms [13].

In an attempt to circumvent the computational complexity of matrix inversion, some researchers are pursuing solutions for the joint accelerations which have a linear recursive form. The inversion of the inertia matrix is explicitly avoided. The resulting linear recursive algorithms have a reduced computational complexity which is $O(N)$. This is the second basic approach to the Direct Dynamics problem, and it has been applied to serial open chains [10,19], single closed chains [14], and some more general multibody systems [16,20]. It is believed that the structure of linear recursive algorithms may also facilitate their implementation on parallel computer systems.

1.3 Scope and Contents

As mentioned above, more efficient algorithms are needed to make real-time dynamic simulation a reality. This need is particularly great for robotic systems with multiple chains and closed kinematic loops. Thus, the fundamental goal of this research effort is the development of better and more efficient algorithms for the dynamic simulation of multiple chain robotic systems. In particular, solutions to the Direct Dynamics problem for simple closed-chain mechanisms are investigated. Simple closed-chain mechanisms are comprised of one reference member interacting with $m$ supporting chains. Initially, it is assumed that the chains do not contain internal loops, so that the removal of the reference member breaks all closed loops in the system. However, the basic simulation approach derived in this dissertation may also be generalized to include chains with a more complex structure.

Figure 2 shows the two basic types of simple closed-chain mechanisms. As shown, a Type 0 mechanism has one end of each chain in unpowered contact with
Figure 2: Models for the Two Types of Simple Closed-Chain Mechanisms

Type 0 \((m=3)\)
(Multiple Manipulator or Dexterous Hand)

Type 1 \((m=4)\)
(Multilegged Vehicle)
the reference member, while a Type 1 mechanism has one end of each chain connected to the reference member by means of an actuated joint structure. Many different multiple chain robotic systems with simple closed loops may be represented by one of these two models. For example, multiple manipulators with a common load and a dexterous hand manipulating an object are both Type 0 simple closed-chain mechanisms. Multilegged vehicles, such as walking machines, are more appropriately modelled by the Type 1 mechanism.

In pursuit of the goal described above, it is important to understand the dynamic behavior of single closed chains. Single closed chains are, in fact, of significant interest themselves. Many industrial operations which involve a single manipulator, such as assembly or welding, bring the manipulator in contact with its environment. This creates a closed loop configuration with contact forces acting on the end effector. An efficient new serial algorithm for the dynamic simulation of a single closed chain will be derived in this dissertation. This leads directly to the development of a corresponding algorithm for simple closed-chain mechanisms.

In studying single closed chains, two physical concepts are of special significance. The first, the joint space manipulator inertia matrix, has been a subject of investigation for several years. It is a key component in the joint space dynamics of any system, and its efficient computation is highly desirable. Several new algorithms for computing the joint space inertia matrix, including the most efficient algorithm known at this time, are derived in this dissertation. The second physical concept, the operational space inertia matrix of a manipulator, is a newer subject of investigation. It was introduced by Khatib [21] as part of the "operational space formulation", in which manipulator control is carried out in end effector variables. The operational space inertia matrix defines the relationship between the generalized forces and accelerations of the end effector, effectively reflecting the dynamics
of an actuated chain to its tip. This dissertation will demonstrate its value as a tool in the development of Direct Dynamics algorithms for closed-chain configurations, and a number of new and efficient algorithms, including two linear recursive methods, are derived for its computation.

### 1.4 Organization

In Chapter 2, the general notational and arithmetic concepts necessary for modelling robotic mechanisms and formulating their kinematic and dynamic equations are presented. These include a modified system of spatial notation and arithmetic, the kinematic and dynamic parameters used to describe mechanisms, and the general joint model which will be used throughout this dissertation to describe the interactions between rigid bodies of a system.

In Chapter 3, four efficient serial algorithms for the joint space manipulator inertia matrix are developed. The first, the Structurally Recursive Method, is based on the concept of adding successive single links to the free end of a serial chain. Additional algorithms are derived by expanding and manipulating the equations of this first recursive algorithm. These include the Inertia Projection Method, the Modified Composite-Rigid-Body Method, and the Spatial Composite-Rigid-Body Method. Finally, the computational requirements for all four methods are compared with those of other existing algorithms. It is shown that the Modified and Spatial Composite-Rigid-Body Methods are the most computationally efficient of all those compared.

In Chapter 4, four efficient serial algorithms are developed for computing the operational space inertia matrix of a manipulator. The first, the Explicit Inversion/Multiplication Method, is based on equations developed in the operational space formulation of Khatib [21]. The second algorithm, the Unit Force Method,
is based on the concept of applying spatial unit forces to the end effector along the workspace axes and determining the resulting acceleration of the chain tip. This algorithm is the most efficient method known for small \( N \). The final two algorithms, the Force Propagation and Inertia Propagation Methods, are both linear recursive algorithms with reduced computational complexities of \( O(N) \). This represents a significant improvement in the order of computation required for this task.

Chapter 5 presents the development of an efficient serial algorithm for the dynamic simulation of a single closed chain. The chain may have arbitrary joints and any number of degrees of freedom. Thus, the algorithm is also valid when singular positions occur. Two classes of contacts between the end effector and other rigid bodies are defined, and specific examples of these contact models are given. The operational space inertia matrix is a significant component of this algorithm, and two versions of the algorithm are developed by utilizing different methods for its computation. The first version has a computational complexity of \( O(N^3) \) for a manipulator with \( N \) degrees of freedom. However, a second version is also developed with a reduced order computational complexity of \( O(N) \). The computational complexities of these two versions are tabulated and compared.

Chapter 6 presents the development of an efficient serial algorithm for the dynamic simulation of simple closed-chain mechanisms. The development of this algorithm is based on many of the same concepts used in the previous chapter for a single closed chain. The supporting chains may have general joints and any number of degrees of freedom. Although the simulation algorithm is initially derived for serial-link chains with no internal loops, the basic analytical method developed in this chapter may also be applied when the chains have a more general structure. In this algorithm, the effective operational space inertia of the entire
system, formed by combining the operational space inertia matrix of each chain and the spatial inertia of the reference member, is used to decouple the chains from the reference member. Then the joint accelerations may be computed for each chain separately. When implemented on a single processor, the simulation algorithm has a computational complexity of $O(mN)$ for $m$ chains, each having $N$ degrees of freedom. The computational complexity of the new algorithm may be further reduced to $O(N) + O(\log_2 m)$ if it is implemented on $m$ processors in parallel.

In the final chapter, a summary of the work completed here and some final conclusions concerning the results are given. Possible future extensions to this work and other related technical issues are also discussed.
CHAPTER II
SYSTEM MODELLING AND NOTATION

2.1 Introduction

This chapter summarizes the notation and system modelling concepts which will be used throughout this dissertation in the formulation of the kinematic and dynamic relationships for robotic mechanisms. The assignment of coordinate frames is briefly discussed, and the kinematic and dynamic parameters of a mechanism are defined. Two important modelling tools are also discussed in this chapter: (1) spatial notation, and (2) a general joint model. Spatial notation was introduced by Featherstone in [19], and a modified form of this notation is used here. Spatial notation allows the user to condense the equations describing a physical system into a concise and elegant form. The physical relationships of the system are not changed, but the simplified format makes the corresponding equations easier to manipulate and often leads to gains in computational efficiency.

The second important modelling tool used in this work is the general joint model of Roberson and Schwertassek [22]. This concept allows arbitrary joints and contacts to be described using orthogonal sets of vectors which are associated with the free and constrained modes of motion of the joint or contact. Multiple degree-of-freedom joints are easily included in any analysis. It is a particularly useful tool in the context of closed chains, since the contacts which are part of the system may be modelled in the same manner and using the same notation as the
regular joints of a manipulator.

Many of the notational conventions presented in this chapter are based on concepts introduced by Roberson and Schwertassek in [22] and used by Brandl, Johanni, and Otter in [10]. They are similar in many ways to those described by Featherstone in [23], although there are a few minor differences.

2.2 Kinematic and Dynamic Parameters of a Robotic Mechanism

For manipulator chains with joints characterized by a single axis (revolute, prismatic, cylindric, and screw joints), the following modelling conventions may be used. First, the links of the chain are numbered such that the base of the chain is link 0 and the final link is link \( N \). Between link \( (i - 1) \) and link \( i \) is joint \( i \). Coordinate systems are assigned according to a modified form of the Denavit-Hartenberg convention [24], so that the coordinate frame of a link is nominally positioned at its near end and is attached to that link. That is, the origin of coordinate frame \( i \) is located at joint \( i \), and it is attached to link \( i \). Coordinate axes are aligned such that the unit vector \( \hat{z}_i \) lies along the axis of motion of joint \( i \), and \( \hat{x}_i \) lies along the common normal between the extensions of \( \hat{z}_i \) and \( \hat{z}_{i+1} \). Figure 3 illustrates the assignment of coordinate frames and the four scalar parameters which may be used to define the relative position and orientation of one coordinate frame with respect to an adjacent one. The definitions of these parameters are as follows:

\[
\begin{align*}
a_{i+1} &= \text{the perpendicular distance along } \hat{x}_i \text{ between } \hat{z}_i \text{ and } \hat{z}_{i+1}, \\
d_{i+1} &= \text{the perpendicular distance along } \hat{z}_{i+1} \text{ between } \hat{x}_i \text{ and } \hat{x}_{i+1}, \\
\alpha_{i+1} &= \text{the angle about } \hat{x}_i \text{ between } \hat{z}_i \text{ and } \hat{z}_{i+1}, \\
\theta_{i+1} &= \text{the angle about } \hat{z}_{i+1} \text{ between } \hat{x}_i \text{ and } \hat{x}_{i+1}.
\end{align*}
\]
Figure 3: Parameters Relating Adjacent Link Coordinate Systems
For single-axis joints, the matrix functions used to transform vector quantities between coordinate frames are functions of these four parameters. These transformations will be defined in the following pages. The modelling conventions used for more complex joint structures will be discussed in the next section.

In spatial notation, velocity, acceleration, and force vectors are all $6 \times 1$ column vectors, and each incorporates the appropriate linear and angular components. For example, the velocity of link $i$, $v_i$, may be written [10]:

$$v_i = \begin{bmatrix} (\omega_i)_x \\ (\omega_i)_y \\ (\omega_i)_z \\ (v_i)_x \\ (v_i)_y \\ (v_i)_z \end{bmatrix}, \quad (2.1)$$

where $(\omega_i)_x$, $(\omega_i)_y$, and $(\omega_i)_z$ are the components of the angular velocity of link $i$ about $x_i$, $y_i$, and $z_i$, respectively. The three components, $(v_i)_x$, $(v_i)_y$, and $(v_i)_z$, represent the linear velocity of the $i$th coordinate origin resolved along these same axes. Similarly, the acceleration of link $i$ may be expressed [10]:

$$a_i = \begin{bmatrix} (\alpha_i)_x \\ (\alpha_i)_y \\ (\alpha_i)_z \\ (a_i)_x \\ (a_i)_y \\ (a_i)_z \end{bmatrix}, \quad (2.2)$$

where the individual components now correspond to resolved angular and linear
acceleration vectors. Force vectors may have a corresponding structure [10]:

\[
\mathbf{f}_i = \begin{bmatrix}
(n_i)_x \\
(n_i)_y \\
(n_i)_z \\
(f_i)_x \\
(f_i)_y \\
(f_i)_z
\end{bmatrix},
\]

where \( \mathbf{f}_i \) is a generalized force exerted on link \( i \). The first three components, \((n_i)_x\), \((n_i)_y\), and \((n_i)_z\), represent the elements of a three-dimensional moment vector, while \((f_i)_x\), \((f_i)_y\), and \((f_i)_z\) are the elements of a three-dimensional force vector.

In general, the transformation of a spatial vector quantity from one coordinate system to an adjacent one may be accomplished by the following spatial multiplication [23]:

\[
i^{i+1}\mathbf{p}_i = i^{i+1}\mathbf{X}_i \cdot ^i\mathbf{p}_i,
\]

where \(^i\mathbf{p}_i\) is a vector associated with link \( i \) that is expressed with respect to the \( i \)th coordinate system, \( ^{i+1}\mathbf{p}_i \) is the same vector expressed with respect to the \((i + 1)\)st coordinate system, and \( ^{i+1}\mathbf{X}_i \) is the \( 6 \times 6 \) spatial transformation matrix. This spatial transformation may be defined as follows [10,22,23]:

\[
i^{i+1}\mathbf{X}_i = \begin{bmatrix}
i^{i+1}\mathbf{A}_i & 0 \\
i^{i+1}\mathbf{A}_i & ^{i+1}\mathbf{A}_i
\end{bmatrix},
\]

where

\[
i^{i+1}\mathbf{A}_i = \begin{bmatrix}
c\theta_{i+1} & s\theta_{i+1}c\alpha_{i+1} & s\theta_{i+1}s\alpha_{i+1} \\
-s\theta_{i+1} & c\theta_{i+1}c\alpha_{i+1} & c\theta_{i+1}s\alpha_{i+1} \\
0 & -s\alpha_{i+1} & c\alpha_{i+1}
\end{bmatrix},
\]

15
and
\[
\mathbf{b}_{i+1} = \begin{bmatrix}
a_{i+1} \\
-d_{i+1}s_{\alpha_{i+1}} \\
d_{i+1}c_{\alpha_{i+1}}
\end{bmatrix},
\]
for single-axis joints. As usual, expressions such as \(c\theta\) and \(s\theta\) represent the cosine and sine of the angle \(\theta\), respectively. In general, \(i^{{+1}}A_i\) is the 3 x 3 rotation transformation between the two coordinate systems, and \(\mathbf{b}_{i+1}\) is the 3 x 1 position vector from the origin of frame \(i\) to the origin of frame \((i + 1)\), with components expressed in frame \(i\). The 3 x 3 matrix, \(\hat{\mathbf{b}}_{i+1}\), is an anti-symmetric matrix defined by the rule:
\[
\hat{\mathbf{c}} = \begin{bmatrix}
0 & -c_3 & c_2 \\
c_3 & 0 & -c_1 \\
-c_2 & c_1 & 0
\end{bmatrix}.
\]

Inertia matrices are also expressed as 6 x 6 matrices. An inertia matrix is defined for each individual link in its own coordinate system. For link \(i\), this matrix, \(\mathbf{I}_i\), is represented as follows:
\[
\mathbf{I}_i = \begin{bmatrix}
\hat{\mathbf{I}}_i & \hat{\mathbf{h}}_i \\
\hat{\mathbf{h}}_i^T & \mathbf{M}_i
\end{bmatrix},
\]
where \(\mathbf{M}_i\) is the 3 x 3 diagonal matrix of the mass of link \(i\),
\[
\mathbf{M}_i = \begin{bmatrix}
m_i & 0 & 0 \\
0 & m_i & 0 \\
0 & 0 & m_i
\end{bmatrix},
\]
and \(\tilde{\mathbf{I}}_i\) is the 3 x 3 moment of inertia tensor at the origin of the \(i\)th coordinate system. The matrix \(\tilde{\mathbf{I}}_i\) is symmetric and positive definite, but not necessarily diagonal. The 3 x 3 matrix, \(\hat{\mathbf{h}}_i\), is equal to \(m_i\hat{s}_i\), where \(m_i\) is the mass of link \(i\), and \(\hat{s}_i\) is the position vector of the center of gravity of link \(i\) from the \(i\)th
coordinate origin. Because $\mathbf{I}_i$ and $\mathbf{s}_i$ are defined in coordinate system $i$, the matrix $\mathbf{I}_i$ is constant.

In general, the transformation of a spatial inertia matrix from one coordinate system to an adjacent one may be accomplished by the following spatial matrix multiplication \[23\]:

$$\mathbf{I}_{i+1}^i = \mathbf{I}_{i+1}^{i+1} \mathbf{X}_{i+1}^{i+1} \mathbf{X}_i^i,$$

(2.11)

where $\mathbf{I}_{i+1}^{i+1}$ is the spatial inertia of link $i+1$ expressed in the $(i+1)$st coordinate system, and $\mathbf{I}_{i+1}^i$ is the same inertia matrix expressed in the $i$th coordinate system.

### 2.3 General Joint Model

To include general joints and contacts with multiple degrees of freedom in a multibody system, an extended model of the interconnections and interactions between individual bodies of that system is required. This section will summarize the important features of one such description, that of Roberson and Schwertassek [22], which is consistent with the invariant method discussed in [25]. This particular model is also used by Brandl, Johanni, and Otter in [10]. The notation used here is slightly different from that found in [10] in order to maintain consistency in the presented algorithms. This model will be used extensively throughout this work in the development of all new algorithms.

In the following paragraphs, the term “joint” will be used to represent the physical interaction or connection between two rigid bodies of a system. These connections include unpowered contacts (e.g. hard point contact, sliding contact) as well as “standard” actuated joints of various degrees of freedom found between manipulator links (e.g. revolute, prismatic, cylindric, spherical, screw). Because of the general nature of this mathematical model, “joints” can have any number of degrees of freedom, up to and including six. That is, even the relative motion of
two free bodies in space may be considered as the action of a six degree-of-freedom joint between the two.

When using this general joint model, the location of link coordinate frames is the same as described in Section 2.2. In fact, for single-axis joints, the joint variables, axis alignment, and spatial coordinate transformations all remain unchanged by this new model. The choice of joint variables, axis alignment, and the spatial transformations between coordinate systems become more complex for multi-axis joints, however. These issues will not be discussed in detail here. Details concerning these transformations may be found in [22] and [23].

To describe the relative motion of body $i$ with respect to body $(i-1)$, generalized minimal coordinates must be defined. If joint $i$ has $n_i$ degrees of freedom, then the $n_i \times 1$ vector $q_i^*$ is defined as the relative joint position vector for joint $i$. Its derivative, $\dot{q}_i^*$, is the relative joint velocity vector. For a simple revolute joint, $q_i^*$ is just the scalar $\theta_i$, while for a simple prismatic joint, $q_i^*$ is the scalar $d_i$. The relative velocity of body $i$ with respect to body $(i-1)$ is a linear function of $\dot{q}_i^*$. The spatial relative acceleration of body $i$ with respect to body $(i-1)$ may be written as follows [10]:

$$a_i^* = \phi_i \ddot{q}_i^* + \xi_i,$$

where $\xi_i$ is a function of joint position, velocity, and time. This equation is always linear in $\ddot{q}_i^*$, the relative joint acceleration vector.

The matrix, $\phi_i$, is of special significance. It is of dimension $6 \times n_i$, resolved in the coordinate frame of body $i$, and it has full column rank, $n_i$. It represents the free modes of joint $i$, and its columns make up a basis for this free vector space. This will also be referred to as the motion space of joint $i$. Because $\phi_i$ is resolved in the $i$th frame, it is constant. It is always possible to construct a $6 \times 6$ matrix
as shown [10]:

\[ \begin{bmatrix} \phi_i & \phi_i^c \end{bmatrix}, \quad (2.13) \]

where the six columns form a basis for all of \( \mathbb{R}^6 \). In general, the columns of \( \phi_i^c \) form a basis for the constrained modes of joint \( i \) and are linearly independent of the columns of \( \phi_i \). From this basis, a second dual basis may be constructed as follows:

\[ \begin{bmatrix} \psi_i & \psi_i^c \end{bmatrix}^T = \left[ \begin{bmatrix} \phi_i & \phi_i^c \end{bmatrix} \right]^{-1}. \quad (2.14) \]

This definition leads to the following relationships:

\[
\begin{align*}
(\phi_i)^T \psi_i &= 1, \quad (2.15) \\
(\phi_i)^T \psi_i^c &= 0, \quad (2.16) \\
(\phi_i^c)^T \psi_i &= 0, \quad (2.17) \\
(\phi_i^c)^T \psi_i^c &= 1, \quad (2.18)
\end{align*}
\]

where \( 1 \) and \( 0 \) are the identity and zero matrices of the appropriate dimensions, respectively. It should be noted that \( \psi_i^c \) is a \( 6 \times (6 - n_i) \) matrix which is orthogonal to \( \phi_i \), the matrix of the free modes of joint \( i \). It should be clear that \( \psi_i^c \) represents the constrained modes of joint \( i \), and its columns also form a basis for this vector space. This will be referred to as the constraint space of joint \( i \), and it is orthogonal to the motion space.

The dynamic interaction between two bodies may be described by a resultant force and moment. Thus, when bodies \((i - 1)\) and \( i \) interact, whether through a joint or contact, a generalized force, \( \mathbf{f}_i \), is exerted on body \( i \) by body \((i - 1)\) and with a negative sign on body \((i - 1)\). Its form is given in Eq. (2.3). This force vector may be resolved in the dual basis defined above:

\[ \mathbf{f}_i = \psi_i \tau_i^* + \psi_i^c \tau_i^c, \quad (2.19) \]
where $\tau_i^*$ is the $n_i \times 1$ vector of applied or driving forces, and $\tau_i^c$ is the $(6 - n_i) \times 1$ vector of constraint forces.

When considering the Direct Dynamics problem, $\tau_i^*$, the vector of driving forces, is known for every joint. The appended vector of applied forces for the entire system is called $\tau$. The unknowns to be found are the joint accelerations for the system, $\ddot{q}$, and all the constraint forces, $\tau^c$ (if desired).

2.3.1 Examples

To illustrate the principles of the general joint model described above, the appropriate joint variables and the basis vectors for the motion and constraint vector spaces will be defined for several examples, including the familiar revolute and prismatic joints.

For a revolute joint:

$$\mathbf{q}_i^* = \theta_i. \tag{2.20}$$

Since the motion of a revolute joint is confined to rotation about the $\hat{z}_i$ unit vector, the motion space is:

$$\phi_i = \psi_i = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T, \tag{2.21}$$

while the constraint space of the joint may be written:

$$\phi_i^c = \psi_i^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.22}$$
For a prismatic joint:

\[ q_i^* = d_i. \]  \hspace{1cm} (2.23)

Because the motion of a prismatic joint is confined to sliding along the \( \hat{z}_i \) unit vector, the motion space is:

\[ \phi_i = \psi_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T, \]  \hspace{1cm} (2.24)

while the constraint space of the joint may be written:

\[ \phi_i^c = \psi_i^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \end{bmatrix}. \]  \hspace{1cm} (2.25)

Note that the vector spaces, \( \phi_i \) and \( \psi_i \), may be made the same for these joints, as can their counterpart vector spaces, \( \phi_i^c \) and \( \psi_i^c \). This fact will be used to good advantage in the computational considerations of later chapters.

For a screw joint, the rotation about the joint axis, \( \theta_i \), and the translation along the axis, \( d_i \), are coupled by the screw such that:

\[ d_i = \rho \theta_i, \]  \hspace{1cm} (2.26)

where \( \rho \) is the scalar pitch of the screw. Because the two motions are coupled, there is only one degree of freedom associated with this joint. Either \( \theta_i \) or \( d_i \) may be used. Here, we will choose:

\[ q_i^* = \theta_i. \]  \hspace{1cm} (2.27)

The motion space which corresponds to this degree of freedom is [22]:

\[ \phi_i = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & \rho \end{bmatrix}^T, \]  \hspace{1cm} (2.28)
while the remaining basis vectors for $\mathbb{R}^6$ are:

$$
\phi_i^\xi = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}.
$$

The constraint space for this screw joint may be written:

$$
\psi_i^\xi = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -\rho & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix},
$$

while the remaining basis vectors for $\mathbb{R}^6$ are:

$$
\psi_i = \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}^T.
$$

Note that in this case $\phi_i$ and $\psi_i$ are not the same; likewise, $\phi_i^\xi$ and $\psi_i^\xi$ also differ.

Additional examples of more complex connections may be found in [22].

### 2.4 Summary

This chapter has summarized the notation and important modelling conventions which will be used in the following chapters in the development of new dynamic algorithms. The kinematic and dynamic parameters which describe a robotic mechanism and the assignment of coordinate frames were discussed. The
basic features of the modified spatial notation used here were described, and the spatial transformations necessary for many calculations were defined. The general joint model which will be used to represent both standard actuated joints and unpowered contacts was discussed, and some examples were given. In the next chapter, the development of new algorithms for the dynamic analysis of robotic mechanisms begins with the joint space inertia matrix for a single serial chain.
CHAPTER III

ALTERNATE FORMULATIONS FOR THE JOINT SPACE INERTIA MATRIX

3.1 Introduction

In general, the inertia matrix of a manipulator defines the relationship between certain forces exerted on the system and some corresponding acceleration vector. This relationship is of great importance both in real-time control and in the simulation of multibody systems. In the control realm, for example, the inertia matrix has been used to decouple robot dynamics so that control schemes may be more effectively applied [21]. This may be accomplished either in joint space or operational space, since the inertia matrix may be defined in either domain. The inertia matrix has also been used in the analysis of collision effects [26]. In addition to its use in control applications, the inertia matrix is an explicit and integral part of certain Direct Dynamics algorithms which are used to solve the simulation problem for manipulators and other multibody systems [9,11,12,13,18].

This chapter will present the development of four efficient serial algorithms for the manipulator joint space inertia matrix. The methodology for the first algorithm, the Structurally Recursive Method, is based on the concept of adding successive single links to the free end of a serial chain. The forces experienced by an added link are projected onto the motion space of the manipulator so that the joint space inertia matrix may be more easily identified. The Jacobian matrix is
used to obtain this projection, and an appropriate method for its calculation is also presented. A structurally recursive method for computing the inertia matrix of the entire manipulator is derived by combining the inertial properties of the added link with the inertia matrix of the original smaller chain. The equations for the Structurally Recursive Method for the joint space inertia matrix are developed and summarized in the fourth section, following a review of previous work and the presentation of additional notation in the second and third sections, respectively.

The Structurally Recursive Method is then expanded, and a second, non-recursive algorithm for the manipulator inertia matrix is derived from it. A finite summation, which is a function of individual link inertia matrices and columns of the appropriate Jacobian matrices, is defined for each element of the joint space inertia matrix in the Inertia Projection Method. Further manipulation of this expression and application of the composite-rigid-body inertia concept [9] are used to obtain two additional algorithms, the Modified Composite-Rigid-Body Method and the Spatial Composite-Rigid-Body Method, in the sixth and seventh sections, respectively. These algorithms do make use of recursive expressions and are more computationally efficient.

The eighth section describes efficient computational methods for the transformation of spatial quantities between adjacent coordinate frames and for the calculation of composite-rigid-body inertias. These procedures are used throughout this work to improve the computational efficiency of the algorithms. Other factors which simplify the equations are also discussed.

In the ninth section, the computational requirements for the methods presented here are compared with those of existing methods for the joint space inertia matrix. Both general and specific cases are considered. It is shown that the Modified Composite-Rigid-Body and Spatial Composite-Rigid-Body Methods are the
most computationally efficient of all those compared. The results of this chapter are summarized in the final section.

3.2 Previous Work

Although the definition of the inertia matrix is simple in the physical sense, its calculation is quite complex. A number of different approaches have been investigated in the search for computationally efficient algorithms. Lee and Lee [27] use the generalized d'Alembert equations of motion to describe the dynamic behavior of robot manipulators with revolute joints. Specific expressions are defined for the elements of the joint space inertia matrix as a part of this formulation. The computational complexity of these equations is $O(N^3)$ for an $N$ degree-of-freedom manipulator. This is quite high, and makes this algorithm inefficient. In addition, the restriction of this algorithm to configurations of revolute joints is undesirable.

Angeles and Ma [11] use the notion of a "natural orthogonal complement" to derive an algorithm for the inertia matrix. The orthogonal complement, in this case, is related to the matrix of unit twists, and it can be computed in a recursive manner. The inertia matrix is factored into the product of a $6n \times n$ block lower-triangular matrix and its transpose. Although the orthogonal complement concept is powerful, the overall computational complexity of this algorithm is still $O(N^3)$. This algorithm is also restricted to configurations of revolute and prismatic joints.

Walker and Orin [9] present one of the most familiar and efficient approaches for the computation of the inertia matrix in the so-called Composite-Rigid-Body Method [23]. This algorithm utilizes the concept of composite-rigid-body inertias to simplify the calculation of the manipulator inertia matrix. The computational complexity of this approach, $O(N^2)$, is significantly reduced compared to those described above, but the restriction to revolute and/or prismatic joints remains.
In a recent effort [28], Li uses the theory of Lagrangian mechanics to formulate the dynamic equations of a manipulator. Similarly to Lee and Lee [27] above, this formulation includes an algorithm for computing the elements of the joint space inertia matrix. In this approach, Li is able to further reduce the required computations for the inertia matrix, making this algorithm the most efficient serial algorithm prior to the present work. The computational complexity is \( O(N^2) \), and the equations are applied to revolute and/or prismatic joint configurations only.

Parallel computation methods have also been investigated for the joint space inertia matrix. Amin-Javaheri and Orin [29], as well as Fijany and Bejczy [30], have achieved better performance by developing parallel and/or pipelined algorithms. In both cases, the parallel forms are based to a great extent on the serial Composite-Rigid-Body Method of Walker and Orin [9], and, of course, the improvement in performance is dependent on an increased number of processors.

3.3 Additional Notation and Background

The general dynamic equations of motion for a single \( N \) degree-of-freedom manipulator may be written in the following form [9]:

\[
\tau = H(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) + J(q)^T f, \tag{3.1}
\]

where

\[
\tau = N \times 1 \text{ vector of joint torques (forces)};
\]
\[
q, \dot{q}, \ddot{q} = N \times 1 \text{ vectors of joint positions, velocities, and accelerations};
\]
\[
C(q, \dot{q}) = N \times N \text{ matrix of Coriolis and centripetal force terms};
\]
\[
G(q) = N \times 1 \text{ vector of gravitational forces};
\]
\[
J(q) = 6 \times N \text{ Jacobian matrix};
\]
\[ f = 6 \times 1 \text{ vector of external forces and moments exerted by the last link,} \]

and \( \mathbf{H}(\mathbf{q}) \) is the \( N \times N \) manipulator joint space inertia matrix, which is both symmetric and positive definite. It is the computation of the matrix \( \mathbf{H}(\mathbf{q}) \) which this chapter addresses. Note that Eq. (3.1) is expressed in joint space and that the constraint forces and moments between the links have been eliminated.

In the analysis of the following sections, the gravitational forces exerted on the given manipulator will be neglected. It will also be assumed that the joint velocities of the manipulator are all zero. These two conditions result in the second and third terms on the right side of Eq. (3.1) being zero. These assumptions are made to simplify the derivation. Because it is a function of position only, the inertia matrix will not be affected by these conditions on the system.

The notation used in this chapter is consistent with the kinematic and dynamic models described in Chapter 2. To facilitate the concept of adding links, however, some additional quantities must be defined. The number \( N_i \) is the total number of degrees of freedom of an \( i \)-link manipulator, while \( n_i \) gives the number of degrees of freedom for the single joint \( i \). The \( N_i \times 1 \) vector, \( \ddot{\mathbf{q}}_i \), represents the entire joint acceleration vector for an \( i \)-link manipulator, while the \( n_i \times 1 \) vector, \( \ddot{\mathbf{q}}^*_i \), is a vector of the joint accelerations about the axes of joint \( i \) alone. These vectors are related in the following way:

\[ \ddot{\mathbf{q}}_i = \begin{bmatrix} \ddot{\mathbf{q}}_{i-1} \\ \ddot{\mathbf{q}}^*_i \end{bmatrix}. \]  

A similar relationship exists for the joint torque (force) vector, \( \tau \), as shown below:

\[ \tau_i = \begin{bmatrix} \tau_{i-1} \\ \tau^*_i \end{bmatrix}. \]
Finally, $J_i$, the $6 \times N_i$ Jacobian matrix for an $i$-link manipulator, is related to $J_{i-1}$, the $6 \times N_{i-1}$ Jacobian of an $(i-1)$-link manipulator, as follows [10,31,32]:

$$J_i = \left[ iX_{i-1} J_{i-1} : \phi_i \right],$$

(3.4)

where the constant $6 \times n_i$ matrix, $\phi_i$, represents the basis vectors of the motion space of the added joint, joint $i$, described in coordinate system $i$. This concept, as discussed in Chapter 2, allows joints to be completely general, with any number of degrees of freedom up to and including six. Note once again that for the common link connections, revolute and prismatic joints, $\phi_i$ is a single unit vector with only one nonzero (unity) element. This will be used to greatly simplify calculations later. Finally, note that in this form, the Jacobian is always expressed with respect to the coordinate frame of the outermost link $(i)$.

### 3.4 Structurally Recursive Method

The goal of this section is the derivation of an algorithm to compute the inertia matrix of a serial $N$-link manipulator as a recursive function of the inertia matrices of partial configurations of its links. The methodology used is based on the concept of successively adding individual links to the free end of a serial chain (structural recursion). The forces acting on a single added link may be projected onto the motion space of the augmented manipulator to eliminate certain unknown force components. This is accomplished using the Jacobian matrix. In addition, some of the projected terms may be directly related to the dynamic equations of the original smaller chain and the torque (force) vectors at the joints. Thus, the dynamic properties of the new link may be combined with those of the original chain to give the dynamic equations of the total configuration. From these equations, the inertia matrix of the augmented system is easily identified.
To begin the development of the Structurally Recursive Method, a one-link manipulator, as shown in Fig. 4, is examined. A free-body force equation may be written for the single link as follows:

\[ I_1 a_1 = f_1 - 2X_1^T f_2, \]  

(3.5)

where \( a_1 \) is the spatial acceleration of link 1, \( f_1 \) is the spatial force applied to link 1 by the base, and \( f_2 \) is the spatial force applied by link 1 to whatever additional link or body it contacts at the origin of coordinate frame 2. The terms of this equation are written as quantities in Cartesian space. In order to identify the joint space inertia matrix, the equation must be transformed to joint space, as in Eq. (3.1).

The columns of the Jacobian matrix are nothing more than basis vectors for the motion space of the manipulator. A dot product of these columns with both sides of Eq. (3.5) will project each term onto the manipulator joint or motion space. Thus, for the one-link manipulator, we may write:

\[ J_1^T I_1 a_1 = J_1^T f_1 - J_1^T 2X_1^T f_2. \]  

(3.6)

By definition, the projection of \( f_1 \) onto the motion space of the single joint is \( \tau_1 \). The unknown constraint forces and moments at joint 1 have been eliminated. It
is also true that $a_1 = J_1 \ddot{q}_1$, since the joint velocities are all zero. Thus, Eq. (3.6) becomes:

$$\left( J_1^T I_1 J_1 \right) \ddot{q}_1 = \tau_1 - J_1^T \dot{^2}X_1^T f_2.$$  \hspace{1cm} (3.7)

A comparison of this result with Eq. (3.1) leads to the identification of the one-link manipulator joint space inertia matrix as:

$$H_1 = \left( J_1^T I_1 J_1 \right),$$  \hspace{1cm} (3.8)

and the dynamic equation for the single-link manipulator may be written:

$$H_1 \ddot{q}_1 = \tau_1 - J_1^T \dot{^2}X_1^T f_2.$$  \hspace{1cm} (3.9)

Note that the transformation, $^2X_1^T$, is required to reference all quantities to the same coordinate frame (1).

Figure 5 shows a two-link manipulator formed by adding a second link to the free end of the one-link manipulator of Fig. 4. A free-body force equation may be written in Cartesian space for the second link as follows:

$$I_2 a_2 = f_2 - \dot{^2}X_2^T f_3,$$  \hspace{1cm} (3.10)

where $a_2$ is the spatial acceleration of link 2, $f_2$ is the spatial force applied to link 2 by link 1, and $f_3$ is the spatial force applied by link 2 to whatever additional link
or body it contacts. Similar to the one-link case, the Jacobian matrix may be used to project the force terms onto the motion space of the two-link manipulator:

\[ J_2^T f_2 a_2 = J_2^T f_2 - J_2^T 3X_2^T f_3. \]  

(3.11)

The term \( J_2^T f_2 \) represents the projection of \( f_2 \) onto the basis vectors of the motion space, which are expressed with respect to the coordinate system of link 2. By Eq. (3.4), we may write the two-link Jacobian as:

\[ J_2 = \begin{bmatrix} 2X_1 J_1 & \phi_2 \end{bmatrix}. \]  

(3.12)

By definition, the projection of \( f_2 \) onto \( \phi_2 \) is the joint torque (force) vector, \( \tau_2^* \).

Again, the unknown constraint forces at the added joint (joint 2) have been eliminated. The projection, \( J_1^T 2X_1^T f_2 \), may be recognized as the external force term from the dynamic equation for the one-link manipulator, Eq. (3.9).

Thus, we may write:

\[ J_2^T f_2 = \begin{bmatrix} J_1^T 2X_1^T f_2 \\ \phi_2^T f_2 \end{bmatrix} = \begin{bmatrix} -H_1 \dot{q}_1 + \tau_1 \\ \tau_2^* \end{bmatrix}. \]  

(3.13)

If we define the \( N_2 \times N_2 \) matrix,

\[ H_1^* = \begin{bmatrix} H_1 & 0 \\ 0 & 0 \end{bmatrix}, \]  

(3.14)

then we may write:

\[ J_2^T f_2 = -H_1^* \ddot{q}_2 + \tau_2. \]  

(3.15)

Combining Eqs. (3.11) and (3.15), and using the relationship, \( a_2 = J_2 \ddot{q}_2 \), we may rewrite Eq. (3.11) as:

\[ (J_2^T I_2 J_2) \ddot{q}_2 = -H_1^* \ddot{q}_2 + \tau_2 - J_2^T 3X_2^T f_3. \]  

(3.16)
Finally, combining terms, it follows that:

\[(J_2^T I_2 J_2 + H_1^T) \ddot{q}_2 = \tau_2 - J_2^T \dot{X}_2^T f_3,\]  

(3.17)

and we may identify the joint space inertia matrix for the two-link manipulator as:

\[H_2 = (J_2^T I_2 J_2 + H_1^T).\]  

(3.18)

The foregoing procedure may be generalized to handle the addition of an \(i\)th link to an \((i - 1)\)-link serial chain. Given the dynamic equation for the \((i - 1)\)-link manipulator,

\[H_{i-1} \ddot{q}_{i-1} = \tau_{i-1} - J_{i-1}^T \dot{X}_{i-1}^T f_i,\]  

(3.19)

we may write a free-body force equation for the \(i\)th link:

\[I_i a_i = f_i - i^{th}X_i^T f_{i+1}.\]  

(3.20)

The \(i\)-link Jacobian matrix is used to project the force terms onto the motion space of the augmented manipulator:

\[J_i^T I_i a_i = J_i^T f_i - J_i^T i^{th}X_i^T f_{i+1}.\]  

(3.21)

Referring to Eqs. (3.4) and (3.19), and utilizing the relationship,

\[\phi_i^T f_i = \tau_i^*,\]  

(3.22)

we may write:

\[J_i^T f_i = \begin{bmatrix} J_{i-1}^T \dot{X}_{i-1}^T & f_i \\ \phi_i^T & f_i \end{bmatrix} = \begin{bmatrix} -H_{i-1} \ddot{q}_{i-1} + \tau_{i-1} \\ \phi_i^* \end{bmatrix}.\]  

(3.23)

If we define the \(N_i \times N_i\) matrix,

\[H_{i-1}^* = \begin{bmatrix} H_{i-1} & 0 \\ 0 & 0 \end{bmatrix},\]  

(3.24)
where $H_{i-1}$ is of dimension $N_{i-1} \times N_{i-1}$, then we may write:

$$J_i^T f_i = -H_{i-1}^* \ddot{q}_i + \tau_i. \quad (3.25)$$

Combining Eqs. (3.21) and (3.25), and using the relationship, $a_i = J_i \ddot{q}_i$, yields:

$$(J_i^T I_i J_i + H_{i-1}^*) \ddot{q}_i = \tau_i - J_i^{T,i+1} X_i^T f_{i+1}. \quad (3.26)$$

Thus, the joint space inertia matrix for an $i$-link manipulator is:

$$H_i = (J_i^T I_i J_i + H_{i-1}^*). \quad (3.27)$$

This is the basic recursive matrix equation for the manipulator joint space inertia matrix.

The complete algorithm for the Structurally Recursive Method for calculating the joint space inertia matrix of a serial $N$-link manipulator is given in Table 1. It shall be referred to as Method I. This formulation is straightforward and compact. It may be applied to any serial open-chain manipulator with any number of links and general joints with multiple degrees of freedom. The Jacobian matrix for the given configuration is an additional result at no extra computational cost. For an $N$-link configuration with simple revolute and/or prismatic joints, the total computational complexity of this algorithm is $O(N^3)$. This comes as a result of the need to compute $N$ Jacobians, each requiring as many as $O(N)$ scalar operations, and $N$ successive inertia matrices, each requiring as many as $O(N^2)$ scalar operations. A more detailed discussion of the computational complexity will be given in Section 3.9.

3.5 Inertia Projection Method

In this section, we will examine the components of successive inertia matrices as defined by the algorithm given in Section 3.4. First, the expansion of the
Table 1: Algorithm for the Structurally Recursive Method (Method I)

Given:

\[ \phi_i, I_i, \dot{x}_{i-1} \text{ for } i = 1, \ldots, N, \]
\[ J_0 = 0, \]
\[ H_0 = 0; \]

Define:

\[ H_{i-1}^+ = \begin{bmatrix} H_{i-1} & 0 \\ 0 & 0 \end{bmatrix}; \]

Calculate for \( i = 1, \ldots, N: \)

\[ J_i = \begin{bmatrix} \dot{x}_{i-1} & J_{i-1} & \phi_i \end{bmatrix}, \]
\[ H_i = (J_i^T I_i J_i + H_{i-1}^+). \]

Equations for the Structurally Recursive Method (Table 1) leads to an expression for \( H_{ij} \), the \( n_i \times n_j \) submatrix of \( H \), in the form of a summation. Its terms involve projections of individual link inertias onto the preceding joint axis vectors, which are represented by the columns of the appropriate Jacobian matrices. The resulting equations together define the Inertia Projection Method. Further study of the expression for \( H_{ij} \) results in other, more computationally efficient methods which are similar to the Composite-Rigid-Body Method first presented in [9] and further discussed in [23]. These will be presented in later sections of this chapter.

The inertia matrix of a one-link manipulator, as found in Section 3.4, may be written as follows:

\[ H_1 = (J_1^T I_1 J_1) = [\phi_1^T I_1 \phi_1]. \] (3.28)
For a two-link serial chain, $H_2$ is:

$$
H_2 = (J_2^T I_2 J_2 + H_1^T)
$$

$$
= \begin{bmatrix}
\phi_1^T 2X_1^T I_2 2X_1 \phi_1 & \phi_2^T 2X_1^T I_2 \phi_2 \\
\phi_2^T 2X_1^T I_2 \phi_1 & \phi_2^T 2X_1^T I_2 \phi_2
\end{bmatrix}
+ \begin{bmatrix}
\phi_1^T I_1 \phi_1 & 0 \\
0 & 0
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\sum_{k=1}^2 \phi_1^T kX_1^T I_k kX_1 \phi_1 & \phi_1^T 2X_1^T I_2 \phi_2 \\
\phi_2^T I_2 2X_1 \phi_1 & \phi_2^T 2X_1^T I_2 \phi_2
\end{bmatrix}
$$

Note the summation form which may be used for the upper left submatrix. The terms of the summation are functions of the two link inertia matrices, $I_1$ and $I_2$, as well as the first columns of the two Jacobian matrices, $J_1$ and $J_2$. The remaining elements of $H_2$ are single terms which depend only on $I_2$ and $J_2$. In general, each term represents the projection of a particular link inertia onto a specific pair of the preceding joint axis vectors.

The inertia matrix for a three-link serial chain may be written:

$$
H_3 = (J_3^T I_3 J_3 + H_2^T)
$$

$$
= \begin{bmatrix}
\phi_1^T 2X_1^T 3X_2^T I_3 3X_2 2X_1 \phi_1 & \phi_2^T 2X_1^T 3X_2^T I_3 3X_2 \phi_2 & \phi_3^T 2X_1^T 3X_2^T I_3 \phi_3 \\
\phi_2^T 3X_2^T I_3 3X_2 2X_1 \phi_1 & \phi_2^T 3X_2^T I_3 3X_2 \phi_2 & \phi_2^T 3X_2^T I_3 \phi_3 \\
\phi_3^T 3X_2^T I_3 3X_2 \phi_1 & \phi_3^T 3X_2^T I_3 \phi_2 & \phi_3^T I_3 \phi_3
\end{bmatrix}
+ \begin{bmatrix}
\sum_{k=1}^2 \phi_1^T kX_1^T I_k kX_1 \phi_1 & \phi_1^T 2X_1^T I_2 \phi_2 & 0 \\
\phi_2^T I_2 2X_1 \phi_1 & \phi_2^T I_2 \phi_2 & 0 \\
0 & 0 & 0
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\sum_{k=1}^3 \phi_1^T kX_1^T I_k kX_1 \phi_1 & \sum_{k=2}^3 \phi_1^T kX_1^T I_k kX_2 \phi_2 & \phi_1^T 2X_1^T I_3 \phi_3 \\
\sum_{k=2}^3 \phi_2^T kX_2^T I_k kX_1 \phi_1 & \sum_{k=2}^3 \phi_2^T kX_2^T I_k kX_2 \phi_2 & \phi_2^T 2X_2^T I_3 \phi_3 \\
\phi_3^T I_3 3X_1 \phi_1 & \phi_3^T I_3 3X_2 \phi_2 & \phi_3^T I_3 \phi_3
\end{bmatrix}
$$

Extrapolating from these expanded versions of the equations for the Structurally Recursive Method, a general expression for the $(i,j)$ submatrix of the $N$-link ma-
Table 2: Algorithm for the Inertia Projection Method (Method II)

Given:

\[ i\phi_i, I_i, i^iX_{i-1} \text{ for } i = 1, \ldots, N; \]

Step 1. Calculate for all \( i = 1, \ldots, N \):

For \( j = i + 1, \ldots, N \):

\[ j^j\phi_j = j^jX_{j-1} j^{-1} \phi_i. \]

Step 2. Calculate for all \( i = 1, \ldots, N \):

For all \( j = i, \ldots, N \):

\[ H_{ij} = \sum_{k=\max(i,j)}^N (k^j\phi_j)^T \mathbf{I}_k (k^j\phi_j). \]

The manipulator inertia matrix, \( H_N \) (or simply \( H \)), may be written as follows:

\[ H_{ij} = \sum_{k=\max(i,j)}^N (k^iX_i \phi_i)^T \mathbf{I}_k (k^jX_j \phi_j), \quad (3.31) \]

where the dimensions of \( H_{ij} \) correspond to the degrees of freedom of the two joints, \( i \) and \( j \). That is, in general, \( H_{ij} \) will be an \( n_i \times n_j \) matrix.

The inertia matrix is symmetric, and so, in terms of submatrices, \( H_{ji} = H_{ij}^T \).

Therefore, only the submatrices on the diagonal and those above or below the diagonal need to be computed. The formal algorithm for computing the entire inertia matrix, \( \mathbf{H} \), of an \( N \)-link manipulator according to this Inertia Projection Method is presented in Table 2. Here, only the upper off-diagonal elements are found. Note that Step 1 simply calculates the Jacobian matrix for each partial manipulator from the 1-link case to the total \( N \)-link manipulator case. This algorithm is mathematically similar in form to that presented by Paul in [33]. The approach used in the present work, however, has led to a simpler, more straight-
forward expression which may be easily manipulated to provide additional insight and lead to other more efficient methods.

It can be shown that, in its present form, Method II has a computational complexity of $O(N^3)$, and the total computation is greater than that of Method I for the same configuration. Step 1 (computation of $N$ Jacobians) requires $O(N^2)$ scalar operations, while Step 2 involves $O(N^2)$ computations with as many as $O(N)$ scalar operations each. By expanding the given summation once again, however, and applying the composite-rigid-body inertia concept [9], it is possible to rewrite the algorithm in a more efficient form and reduce the computational burden.

3.6 Modified Composite-Rigid-Body Method

If the summation for each element as given in Eq. (3.31) is expanded and terms regrouped, the joint space inertia matrix for a three-link manipulator, $H_3$, may be written:

$$H_3 = \begin{bmatrix}
\phi^T_1 K_1 \phi_1 & \phi^T_2 X_1^T K_2 \phi_2 & \phi^T_3 X_1^T X_2^T K_3 \phi_3 \\
\phi^T_2 K_2 X_1^T \phi_1 & \phi^T_2 K_2 \phi_2 & \phi^T_2 X_2^T K_3 \phi_3 \\
\phi^T_3 K_3 X_2^T \phi_1 & \phi^T_3 K_3 X_2^T \phi_2 & \phi^T_3 K_3 \phi_3
\end{bmatrix}, \quad (3.32)$$

where

$$K_3 = I_3, \quad (3.33)$$

$$K_2 = X_2^T I_3 X_2 + I_2 = X_2^T K_3 X_2 + I_2, \quad (3.34)$$

and

$$K_1 = X_1^T (X_2^T I_3 X_2 + I_2)^2 X_1 + I_1 = X_1^T K_2 X_1 + I_1. \quad (3.35)$$

38
\( K_i \) is the spatial composite-rigid-body inertia for bodies \( i \) through \( N \), expressed in the \( i \)th coordinate system. It may be computed recursively using the equation:

\[
K_i = (i+1)X_i^T K_{i+1} i+1 + I_i. \tag{3.36}
\]

The transformation of joint axes from one coordinate system to another may also be done recursively as before:

\[
j\phi_i = j\phi_{j-1} j-1 \phi_i. \tag{3.37}
\]

Then the \((i,j)\) element (or submatrix) of \( H \) may be defined \((i \leq j)\):

\[
H_{ij} = (j\phi_i)^T K_j (j\phi_j), \tag{3.38}
\]

and \( H_{ji} = H_{ij}^T \). Using this approach, the algorithm for the Modified Composite-Rigid-Body Method is formalized in Table 3. Note the Jacobian computation once again in Step 2. By applying the composite-rigid-body concept, the computational complexity has been reduced to \( O(N^2) \) for an \( N \)-link configuration with simple revolute and/or prismatic joints. Step 1, calculation of the composite-rigid-body inertias, is an \( O(N) \) procedure. The \( N \) Jacobians found in Step 2, as discussed earlier, require \( O(N^2) \) operations. Computation of the diagonal and upper off-diagonal elements of \( H \) in Step 3 requires \( O(N^2) \) scalar additions and multiplications. A significant improvement in computational efficiency has been achieved by application of the composite-rigid-body concept, and the simultaneous Jacobian computation has been preserved.

### 3.7 Spatial Composite-Rigid-Body Method

If Eq. (3.32) is examined once again, it is possible to formulate yet another method for computing \( H \). In Method III, after \( K_i \) is found, the transformed joint
Table 3: Algorithm for the Modified Composite-Rigid-Body Method (Method III)

Given:

\[ \{ i \phi_i, I_i, \dot{X}_{i-1} \} \quad \text{for} \quad i = 1, \ldots, N, \]

\[ K_{N+1} = 0; \]

Step 1. Calculate for \( i = N, \ldots, 1 \):

\[ K_i = \left( \dot{X}_{i}^T K_{i+1} \dot{X}_{i} \right) + I_i. \]

Step 2. Calculate for all \( i = 1, \ldots, N \):

For \( j = i + 1, \ldots, N \):

\[ \dot{\phi}_i = \dot{X}_{j-1} \dot{\phi}_i. \]

Step 3. Calculate for all \( i = 1, \ldots, N \):

For all \( j = i, \ldots, N \):

\[ H_{ij} = (\dot{\phi}_i)^T K_j (\dot{\phi}_j). \]
Table 4: Algorithm for the Spatial Composite-Rigid-Body Method (Method IV)

Given:

\[ \phi_i, \mathbf{I}_i, \dot{\mathbf{X}}_{i-1} \quad \text{for} \quad i = 1, \ldots, N, \]

\[ \mathbf{K}_{N+1} = 0; \]

Step 1. Calculate for \( i = N, \ldots, 1 \):

\[ \mathbf{K}_i = \left( \mathbf{i}^{i+1} \mathbf{X}_i^T \mathbf{K}_{i+1} \mathbf{i}^{i+1} \mathbf{X}_i \right) + \mathbf{I}_i, \]

\[ \dot{\mathbf{I}}_i = \mathbf{K}_i \phi_i. \]

Step 2. Calculate for all \( i = 2, \ldots, N \):

For \( j = i - 1, \ldots, 1 \):

\[ \dot{\mathbf{I}}_i = \mathbf{j}^{i+1} \mathbf{X}_j^T \mathbf{j}^{i+1} \mathbf{I}_i. \]

Step 3. Calculate for all \( i = 1, \ldots, N \):

For all \( j = 1, \ldots, i \):

\[ \mathbf{H}_{ji} = \phi_j^T \dot{\mathbf{I}}_i. \]

axes are used to obtain the elements of \( \mathbf{H} \). In a fourth approach, the composite-rigid-body inertia, \( \mathbf{K}_i \), is also calculated recursively for all \( N \). However, in contrast to Method III, the joint axes are not transformed to any other coordinate system. Instead, the portion of \( \mathbf{K}_i \) associated with joint \( i \) is transformed back along the previous links. At each joint \( j \), this vector (or matrix) is projected onto the joint motion space to obtain the corresponding component of \( \mathbf{H} \) which couples joints \( i \) and \( j \).

The algorithm for Method IV is formally presented in Table 4. Note that the Jacobian is no longer computed here because the joint axes are no longer transformed from frame to frame. This approach is, in fact, the Composite-Rigid-
Body Method of Walker and Orin [9], expressed in a spatial form and extended to include multiple-degree-of-freedom joints. It is thus named the Spatial Composite-Rigid-Body Method. It differs only in the respect that the composite-rigid-body inertias computed here are defined at the origin of coordinate frame i, while those in [9] are defined at the center of gravity of the composite-rigid-body. The final results of the two methods are the same.

As is expected, then, the computational complexity of Method IV is also $O(N^2)$ for $N$ links with simple revolute and/or prismatic joints. It requires fewer scalar operations to compute $H$ than in the three previous methods for $N \geq 6$, but the Jacobian matrix for the manipulator is no longer an additional result. The computational complexities of all four methods will be presented and discussed in more detail in Section 3.9. The next section will provide the necessary background by outlining efficient procedures for transforming spatial vectors and rigid-body inertias from one coordinate frame to another. It will also discuss the efficient calculation of composite-rigid-body inertias.

3.8 Efficient Transformation of Spatial Quantities

3.8.1 Transforming Spatial Vectors

As was discussed in Chapter 2, the transformation of a vector from one coordinate system to an adjacent one may be accomplished using the spatial transformation $X$:

$$i + 1 p_i = i + 1 X_i i p_i.$$  \hspace{1cm} (3.39)

The spatial transformation matrix, $i + 1 X_i$, is usually written as a single general transformation, as shown in Eq. (2.5). However, as demonstrated by Featherstone in [23], significant savings can be obtained by using two screw transformations instead, the first on the $x$-axis, followed by a second on the new $z$-axis. Thus, the
transformation, $X$, may be written:

$$X = X_z X_x = \begin{bmatrix} A_z & 0 \\ A_z b_z^T & A_x \end{bmatrix} \begin{bmatrix} A_x & 0 \\ A_z b_z^T & A_x \end{bmatrix}, \quad (3.40)$$

where

$$A_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha & s\alpha \\ 0 & -s\alpha & c\alpha \end{bmatrix}, \quad A_z = \begin{bmatrix} c\theta & s\theta & 0 \\ -s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.41)$$

$b_z = [a \ 0 \ 0]^T$ and $b_x = [0 \ 0 \ d]^T$. This is a feature of the Denavit-Hartenberg coordinates which is sometimes ignored. It is applicable to systems of single-axis joints, such as revolute, prismatic, cylindric, and screw joints.

Multiplication of one screw transformation with a general spatial vector requires (10 scalar multiplications, 6 scalar additions). Thus, the complete transformation of a general vector requires a total of (20 multiplications, 12 additions) if two screw transformations are used. The product of a general transformation between adjacent coordinate systems and a general vector, on the other hand, requires (24 multiplications, 18 additions).

Transforming the unit vector, $\phi_i (\equiv \hat{i}_i)$, is a special case. Since $\phi_i$ has only one nonzero element for revolute and prismatic joints, and this element is unity, transformation of this vector involves only the selection of a single column from $X$, or premultiplication of the appropriate column of $X_x$ by $X_z$. Thus, transformation of $\phi_i$ requires (0 multiplications, 0 additions) if $X$ is available, and at worst (6 multiplications, 2 additions) if $X_z$ and $X_x$ are used. On-line computation of $X$ itself, however, requires (14 multiplications, 4 additions), while $X_z$ requires only (2 multiplications, 0 additions). Note that $X_x$ may be calculated off-line, since the variables involved, $a_i$ and $\alpha_i$ for link $i$, are constants. Finally then, when
general vectors, as well as simple vectors such as $\phi_i$, must be transformed, the most efficient method of transformation must be carefully chosen.

### 3.8.2 Transforming Spatial Rigid-Body Inertias

The spatial inertia matrix of a rigid body may be transformed from one coordinate system to an adjacent one using Eq. (2.11), repeated here for convenience:

$$
\mathbf{I}_{i+1} = \mathbf{X}_i^T \mathbf{I}_i \mathbf{X}_i
$$

If the form of $\mathbf{I}$ as given in Eq. (2.9) is used, then this product is:

$$
X^T I X = \begin{bmatrix}
(A^T \mathbf{I} A + \mathbf{b} A^T \mathbf{h}^T A + A^T \mathbf{h} \mathbf{a}^T + \mathbf{b} A^T \mathbf{M} \mathbf{a}^T)
& (A^T \mathbf{h} A + \mathbf{b} A^T \mathbf{M} A)

& (A^T \mathbf{h} A + A^T \mathbf{M} \mathbf{a}^T)

&A^T \mathbf{M} A
\end{bmatrix}
$$

which includes the $3 \times 3$ congruence transformations $(A^T \mathbf{M} A)$, $(A^T \mathbf{I} A)$, and $(A^T \mathbf{h} A)$. The scalar operations required for the terms shown in Eq. (3.43) are listed in Table 5.

The general vector $\mathbf{b}$ is formed from $\mathbf{b}_x$, $\mathbf{b}_z$, and the link parameters. This simple combination requires only (2 multiplications, 0 additions). $\mathbf{A}$ is the product of matrices $\mathbf{A}_x$ and $\mathbf{A}_z$ defined above. Efficiency may be gained in a congruence transformation by using these two planar rotation matrices individually instead of their product. The matrix $\mathbf{M}$ is invariant to this transformation, and therefore, $(A^T \mathbf{M} A) = \mathbf{M}$. The product, $(A^T \mathbf{h} A)$, is anti-symmetric and requires only (8 multiplications, 4 additions). By decomposing $\mathbf{I}$ into two parts, one of which is invariant under planar rotation [10], the congruence transformation of this matrix requires (23 multiplications, 18 additions), although (2 multiplications, 2 additions) may be eliminated if certain functions of the elements of $\mathbf{I}$ are previously known. In all cases, any terms which are functions of the constant link dimensions, angles, and mass are considered to be computed off-line.
Table 5: Computations for $X^T I X$

<table>
<thead>
<tr>
<th>Component</th>
<th>#Mult</th>
<th>#Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$A^T M A$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$A^T h A$</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>$A^T \bar{h} A$</td>
<td>23</td>
<td>18</td>
</tr>
<tr>
<td>$b A^T M A$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$b A^T M A \bar{b}^T$</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$(b A^T \bar{h}^T A + A^T \bar{h} A \bar{b}^T)$</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>$(X^T I X)_{11}$</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>$(X^T I X)<em>{12} = (X^T I X)</em>{21}^T$</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Total:</td>
<td>49</td>
<td>49</td>
</tr>
</tbody>
</table>

The transformation of $I$ involves other terms which are functions of the expressions discussed above. By using these results carefully, avoiding redundant calculation, and taking into account known zeros in the matrices, the transformation, $X^T I X$, requires only (49 multiplications, 49 additions) as shown. This is a significant improvement over the explicit multiplication of the three $6 \times 6$ matrices. In Table 5, note that the terms $(X^T I X)_{11}$, $(X^T I X)_{12}$, and $(X^T I X)_{21}$ represent the upper left, upper right and lower left submatrices of the product $X^T I X$, respectively.
3.8.3 Computing Spatial Composite-Rigid-Body Inertias

The spatial composite-rigid-body inertia of links \( i \) through \( N \), \( K_i \), was defined in Eq. (3.36) as:

\[
K_i = (i^{+1}X_i^T K_{i+1} i^{+1}X_i) + I_i. \tag{3.44}
\]

It can be easily shown that \( K_{i+1} \) has the same mathematical form as the spatial inertia of a single rigid body. Thus, the congruence transformation of this matrix also requires (49 multiplications, 49 additions) as described above. The addition of \( I_i \) requires only an extra 10 additions, if we consider the symmetry and form of \( K_i \) and \( I_i \). Note that the bottom right submatrix of \( K_i \) is simply the diagonal matrix of the composite mass of links \( i \) through \( N \). Thus, since this composite mass may be computed off-line, 1 scalar addition may be eliminated, and the on-line computation of \( K_i \) requires only (49 multiplications, 58 additions).

It may be shown that this procedure for computing the spatial composite-rigid-body inertia is exactly equivalent to the procedure used by Walker and Orin in [9]. That is, if the composite mass, composite center of mass, and composite moment of inertia matrix are computed for links \( i \) through \( N \), they may be combined to obtain the spatial matrix \( K_i \). After studying this equivalent approach, however, it appears that the congruence transformation method given here is more efficient.

3.9 Computational Requirements

The number of scalar operations required by each of the four methods presented in this chapter have been calculated explicitly. As a specific example, Table 6 lists the computations required by the Modified Composite-Rigid-Body Method for the case of an \( N \)-link manipulator with simple revolute and prismatic joints. Note that the computations listed for the transformation matrix, \( \dot{X}_{i-1} \),
Table 6: Computations for the Modified Composite-Rigid-Body Method

<table>
<thead>
<tr>
<th>Component</th>
<th>#Mult.</th>
<th>#Add.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{i-1}$</td>
<td>$2N - 2$</td>
<td>0</td>
</tr>
<tr>
<td>$K_i$</td>
<td>$49N - 49$</td>
<td>$58N - 58$</td>
</tr>
<tr>
<td>$j\phi_i$</td>
<td>$10N^2 - 24N + 14$</td>
<td>$6N^2 - 16N + 10$</td>
</tr>
<tr>
<td>$H_{ij}$</td>
<td>$2\frac{1}{2}N^2 - 2\frac{1}{2}N$</td>
<td>$2N^2 - 2N$</td>
</tr>
<tr>
<td>Total:</td>
<td>$12\frac{1}{2}N^2 + 24\frac{1}{2}N - 37$</td>
<td>$8N^2 + 40N - 48$</td>
</tr>
</tbody>
</table>

correspond to the use of two screw transformations as discussed in the previous section.

The computational complexities of the four methods presented in this chapter are compared with those of four existing serial algorithms in Table 7. The number of scalar operations (multiplications, additions) is compared for the case of an $N$-link, serial, open-chain manipulator with simple revolute and prismatic joints only. The efficient matrix transformations and other simplifications described in Section 3.8 have been applied in each of Methods I-IV. The computations necessary for determining individual link transformation matrices have also been included in the expressions for these four methods, while this may not be true of the others referenced.

Methods I and II, although simple in development and compact in form, both result in $O(N^3)$ scalar operations. Method II is the least efficient of the four methods presented here. Both Methods I and II, however, also provide the manipulator Jacobian matrix as a simultaneous result at no extra cost, an advantage not found in any of the three previous methods.

Methods III and IV have reduced computational complexities of $O(N^2)$. Method
Table 7: Number of Computations for the Joint-Space Inertia Matrix

<table>
<thead>
<tr>
<th>Method</th>
<th>#Multiplications</th>
<th>#Additions</th>
<th>#Mult. ((N=6))</th>
<th>#Add. ((N=6))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lee/Lee [27]</td>
<td>(\frac{13}{6}N^3 + 18\frac{1}{2}N^2 + 34\frac{1}{3}N + 15)</td>
<td>(\frac{4}{3}N^3 + 16\frac{1}{2}N^2 + 24\frac{1}{6}N + 15)</td>
<td>1376</td>
<td>1042</td>
</tr>
<tr>
<td>Angeles/Ma [11]</td>
<td>(N^3 + 17N^2 - 17N + 8)</td>
<td>(N^3 + 14N^2 - 16N + 5)</td>
<td>734</td>
<td>629</td>
</tr>
<tr>
<td>Walker/Orin [9]</td>
<td>(12N^2 + 56N - 27)</td>
<td>(7N^2 + 67N - 53)</td>
<td>741</td>
<td>601</td>
</tr>
<tr>
<td>Li [28]</td>
<td>(11N^2 + 38N + 8)</td>
<td>(6\frac{1}{2}N^2 + 36\frac{1}{2}N)</td>
<td>632</td>
<td>453</td>
</tr>
<tr>
<td>Lilly/Orin I</td>
<td>(N^3 + 22N^2 - 35N + 12)</td>
<td>(N^3 + 15N^2 - 26N + 10)</td>
<td>810</td>
<td>610</td>
</tr>
<tr>
<td>Lilly/Orin II</td>
<td>(5N^3 + 5\frac{1}{2}N^2 - 22\frac{1}{2}N + 12)</td>
<td>(4N^3 + 3N^2 - 16\frac{5}{6}N + 10)</td>
<td>1155</td>
<td>881</td>
</tr>
<tr>
<td>Lilly/Orin III</td>
<td>(12\frac{1}{2}N^2 + 24\frac{1}{2}N - 37)</td>
<td>(8N^2 + 40N - 48)</td>
<td>560</td>
<td>480</td>
</tr>
<tr>
<td>Lilly/Orin IV</td>
<td>(10N^2 + 41N - 51)</td>
<td>(6N^2 + 52N - 58)</td>
<td>555</td>
<td>470</td>
</tr>
</tbody>
</table>
III is the most efficient approach for the joint space inertia matrix of all those listed for $N < 6$, and computes the manipulator Jacobian matrix as well. Method IV, although the simultaneous Jacobian computation has been eliminated, is the most efficient approach for computing the joint space inertia matrix for $N \geq 6$. Thus, these two methods together provide the most efficient calculation of $H$ for all $N$.

3.10 Conclusions and Summary

In this chapter, four efficient methods for computing the joint space inertia matrix of a serial open-chain manipulator have been presented. The first uses a structurally recursive algorithm for the entire matrix, while the other three define expressions for each matrix element. In all four methods, a general joint model is used which allows multiple-degree-of-freedom joints to connect the links if desired. This generalization permits the application of these methods to more complex robot configurations than the standard revolute/prismatic combinations. For the sake of simplicity in the presentation, however, only these typical configurations have been considered here when evaluating computational efficiency.

More specifically, Methods III and IV have been shown to have improved computational efficiency over previous algorithms for typical robot configurations. Method III is the most efficient algorithm for the calculation of $H$ and $J$ together and the best method for $H$ alone for $N < 6$. Method IV is the most efficient algorithm for $H$ for $N \geq 6$. It should be noted that the efficiency gained by either of these methods may be used to reduce the total computation required for dynamic simulation according to the Walker and Orin approach [9].

The simultaneous computation of the manipulator Jacobian which occurs in Methods I-III is an important feature of these algorithms. Its usefulness will be illustrated with regard to the operational space inertia matrix in the next chapter.
CHAPTER IV

ALTERNATE FORMULATIONS FOR THE OPERATIONAL SPACE INERTIA MATRIX

4.1 Introduction

The operational space inertia matrix, $\Lambda$, like its joint space counterpart, $H$, is an inertial quantity which defines the dynamic relationship between certain forces exerted on a manipulator and a corresponding acceleration vector. In the case of the joint space inertia matrix, the forces of interest are the actuator joint forces and torques, and the corresponding acceleration is the joint acceleration vector. On the other hand, $\Lambda$ traditionally relates the spatial vector of forces and moments exerted at the tip, or end effector, and the spatial acceleration of this same point. Matrix or vector quantities which are defined with respect to the end effector are often said to be in “workspace” or “operational space” coordinates. Hence, $\Lambda$ is called the operational space inertia matrix of a manipulator.

The use of an operational space formulation for the dynamic equations of motion in the analysis, design, and control of robot manipulators has been studied quite extensively by Khatib [21,34,35]. Naturally, the matrix $\Lambda$ is a significant part of this formulation. In later chapters of this dissertation, we will show that the operational space inertia matrix also plays an important role in the development of efficient algorithms for the dynamic simulation of closed-chain robotic systems. In either case, the efficient computation of $\Lambda$ is desirable.
This chapter will present four efficient serial algorithms for computing the operational space inertia matrix of a manipulator. Like the algorithms of the previous chapter, the general joint model is used to allow multiple-degree-of-freedom joints in the configuration. The first algorithm, the Explicit Inversion/Multiplication Method, is based on equations developed in the operational space formulation of Khatib [21]. This algorithm requires the computation of the Jacobian and joint space inertia matrices, which are combined through matrix inversion and multiplication to form A and/or its inverse. Because the inversion of the joint space inertia matrix is required, the computational complexity of this algorithm is $O(N^3)$ for a standard $N$-link manipulator. The equations for the Explicit Inversion/Multiplication Method are given in the fifth section of this chapter, following a brief review of previous work, an introduction to the operational space dynamic formulation, and a derivation of the relationship between $H$ and $A$ in the second, third, and fourth sections, respectively.

The final three algorithms are new and original, developed as a part of this dissertation. The second algorithm, the Unit Force Method, is presented in the sixth section of this chapter. The development of this algorithm is based on the concept of applying spatial unit forces to the end effector along the workspace axes and determining the resulting joint accelerations along the manipulator chain. These results are combined with the manipulator Jacobian to form A and/or its inverse. This algorithm is also $O(N^3)$ for a standard $N$-link manipulator.

In the seventh and eighth sections, two linear recursive algorithms are developed for computing $A^{-1}$ and $A$, respectively. The Force Propagation Method is based on the recursive articulated-body dynamic equations for a constrained chain, which are derived here. The computational structure of this method is a double recursion. The first recursion computes the articulated-body inertia [19] of each
link in the chain, progressing from the tip down to the base member. The second recursion utilizes a new spatial articulated transformation called a "force propagator" which transforms spatial forces across an articulated joint structure. The final result of this second recursion is $\Lambda^{-1}$, the inverse operational space inertia matrix for the entire manipulator. The Inertia Propagation Method, developed in the eighth section, requires only a single recursion to compute $\Lambda$. In this recursion, the operational space inertia matrix at a given link is propagated across the connecting joint by means of an articulated transformation, and combined with the spatial link inertia of the next link in the chain. The recursion progresses from the base member to the tip. Both the Force Propagation and Inertia Propagation Methods have computational complexities of $O(N)$ for a standard $N$-link manipulator. This represents a significant reduction in order from the first two algorithms discussed above.

The computational complexities of all four algorithms are compared in the ninth section of the chapter. A summary of the results and some overall conclusions are given in the final section.

4.2 Previous Work

Because the operational space formulation for robot dynamics is fairly new, few efficient methods exist for computing its components. The conceptual framework for the operational space formulation was presented by Khatib in [21,34,35], where he established a basic definition for the operational space inertia matrix. In [21], Khatib shows that the operational space inertia matrix of a 6 degree-of-freedom manipulator may be computed as follows:

$$\Lambda_6(x) = \Lambda_6(q) = J_6^{-T}(q) H_6(q) J_6^{-1}(q),$$

(4.1)
where

\[ \Lambda_6(x) = 6 \times 6 \text{ operational space inertia matrix}, \]
\[ J_6(q) = 6 \times 6 \text{ Jacobian matrix}, \]
\[ H_6(q) = 6 \times 6 \text{ joint space inertia matrix}. \]

The relationship in Eq. (4.1) was found by examining the quadratic forms of the manipulator kinetic energy matrix in the joint space and operational space dynamic models. With this equation, Khatib shows the explicit dependence of \( \Lambda \) on the generalized system coordinates, either operational space \( (x) \) or joint space \( (q) \).

Khatib also discusses the operational space formulation for redundant manipulators. In this case, the definition for the operational space inertia matrix is:

\[ \Lambda_r(x) = \Lambda_r(q) = [J(q)H^{-1}(q)J^T(q)]^{-1}, \quad (4.2) \]

where

\[ \Lambda_r(x) = 6 \times 6 \text{ operational space inertia matrix for a redundant manipulator}, \]
\[ J(q) = 6 \times N \text{ Jacobian matrix}, \]
\[ H(q) = N \times N \text{ joint space inertia matrix}, \]

and \( N > 6 \). Note that the Jacobian is not a square matrix for redundant systems, so it is no longer invertible. In [21,34,35], Khatib does not consider the form of the operational space inertia matrix for systems with less than 6 degrees of freedom, nor does he consider the computational requirements of the equations given above.

Rodriguez, Kreutz, and Jain [20,36] present a linear recursive algorithm for the operational space inertia matrix, referred to as the "operational space mass matrix", as part of an original operator formulation for open- and closed-chain
multibody dynamics. In general, this operator approach appears to be quite pow-
erful, especially in matrix factorization and inversion, and with it, the authors have
derived linear recursive solutions to several dynamics problems. Unfortunately, the
use of particular mathematical techniques (recursive filtering and smoothing) in
the manipulation of the linear operators, though computationally sound, may make
the physical interpretation of the results somewhat difficult. It is the intent of this
dissertation to emphasize the physical meaning of the equations developed here
and to utilize any resulting insight in their simplification.

4.3 The Operational Space Formulation

The operational space dynamic formulation describes the dynamic behavior of
a robot manipulator in terms of its end effector. The dynamic equations of motion
for a single chain, written in end effector (or operational space) coordinates are
\[ F = A(x) \dot{x} + \mu(x, \ddot{x}) + p(x), \]
where

\[ F = 6 \times 1 \text{ operational force vector}, \]
\[ A(x) = 6 \times 6 \text{ operational space inertia matrix}, \]
\[ x = 6 \times 1 \text{ vector of end effector coordinates}, \]
\[ \dot{x}, \ddot{x} = 6 \times 1 \text{ spatial vectors of end effector rates and accelerations}, \]
\[ \mu(x, \dot{x}) = 6 \times 1 \text{ vector of centripetal and Coriolis forces, and} \]
\[ p(x) = 6 \times 1 \text{ vector of gravity forces.} \]

Mathematically, \( A \) is associated with the kinetic energy matrix of this formulation
[21]. In a more basic physical sense, \( A \) is an inertial quantity which combines the
The relationship between the operational space inertia matrix, $\Lambda$, and its joint space counterpart, $H$, may be established by investigating the relationship between the dynamic equations of their respective formulations. The joint space dynamic
equations of motion for a single chain may be written [21]:

\[ \Gamma = H(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q), \]  

(4.4)

where \( C(q, \dot{q}) \dot{q} \) and \( G(q) \) represent the centripetal/Coriolis and gravity forces in joint space, respectively, and, from Eq. (3.1),

\[ \Gamma = \tau - J^T(q)f \]  

(4.5)

represents the generalized joint forces which result from the driving actuator torques and forces, \( \tau \), and the spatial force vector applied by the end effector, \( f \). Note that when \( \tau = 0 \), \( F \) and \( f \) represent the same force vector, but with opposite signs.

From Eq. (4.4), we may solve for the joint accelerations as follows:

\[ \ddot{q} = H^{-1} (\Gamma - C \dot{q} - G). \]  

(4.6)

The generalized joint force vector, \( \Gamma \), is related to the operational force vector, \( F \), as follows [21]:

\[ \Gamma = J^T(q)F. \]  

(4.7)

If this expression is used in Eq. (4.6), the result is:

\[ \ddot{q} = (H^{-1}J^T)F - H^{-1}(C \dot{q} + G), \]  

(4.8)

\[ = \Omega F - B_{\ddot{q}}(q, \dot{q}), \]  

(4.9)

where \( B_{\ddot{q}}(q, \dot{q}) \) is a joint space bias vector which is a function only of the present state, and the matrix

\[ \Omega = (H^{-1}J^T) \]  

(4.10)

relates the joint acceleration vector and the operational force vector in the absence of bias terms. The spatial velocity of the end effector is related to the joint velocity
vector of a manipulator as follows:

\[ \dot{x} = J(q) \dot{q}. \]  

(4.11)

By definition, then, the spatial acceleration of the end effector is:

\[ \ddot{x} = J(q) \ddot{q} + J(q) \dot{q}. \]  

(4.12)

Combining Eqs. (4.8) and (4.12), we may express the spatial end effector acceleration vector as follows:

\[ \ddot{x} = (JH^{-1}J^T)F - JH^{-1}(C\ddot{q} + G) + J\dot{q}, \]  

(4.13)

\[ = (JH^{-1}J^T)F - B_x(x, \dot{x}), \]  

(4.14)

where \( B_x(x, \dot{x}) \) is an operational space bias vector which is also a function only of the present state. Solving for \( F \) gives:

\[ F = (JH^{-1}J^T)^{-1} \ddot{x} + (JH^{-1}J^T)^{-1} B_x(x, \dot{x}), \]  

(4.15)

\[ = (JH^{-1}J^T)^{-1} \ddot{x} + B_F(x, \dot{x}), \]  

(4.16)

where \( B_F(x, \dot{x}) \) represents the bias gravity and velocity dependent forces.

If we compare Eq. (4.16) with Eq. (4.3), we may identify the operational space inertia matrix as follows:

\[ \Lambda(x) = \Lambda(q) = \left[J(q)H^{-1}(q)J^T(q)\right]^{-1}. \]  

(4.17)

Relationships between the other terms in the two formulations may also be established in a similar manner.

From Eq. (4.17), we see that the operational space inertia matrix is a function of position only. It is always a \( 6 \times 6 \) symmetric matrix, independent of \( N \). Note,
however, that $\Lambda$ is only defined if the bracketed term has full rank, that is, if the product,

$$J(q) H^{-1}(q) J^T(q),$$

has rank 6. It is easy to determine the conditions under which this occurs. The joint space inertia matrix, $H$, is an $N \times N$, symmetric, positive definite matrix with rank $N$. Thus, $H$ is of full rank and invertible. However, the Jacobian matrix, $J$, is a $6 \times N$ rectangular matrix for which the rank varies with position. If $N \geq 6$, and the manipulator is not in a singular position, then the rank of $J$ will be equal to 6, the product above will have rank 6, and $\Lambda$ will be defined. If $N < 6$, or the manipulator has reached a singular position, then $J$ will have rank less than 6, and the product in Eq. (4.18) will not be invertible. From these considerations, we may conclude that the inverse operational space inertia matrix,

$$\Lambda^{-1}(q) = J(q) H^{-1}(q) J^T(q),$$

is defined for all $N$, regardless of singularities, while the operational space inertia matrix,

$$\Lambda(q) = [J(q) H^{-1}(q) J^T(q)]^{-1},$$

is only explicitly defined, in general, for $N \geq 6$. These conclusions are consistent with the results of Khatib [21].

4.5 The Explicit Inversion/Multiplication Method

Perhaps the simplest and most obvious way to compute the operational space inertia matrix (or its inverse) is by the explicit inversion and multiplication of the Jacobian and joint space inertia matrices as shown in Eqs. (4.19) and (4.20). We will call this the Explicit Inversion/Multiplication Method. Although we will see
Table 8: Algorithm for the Explicit Inversion/Multiplication Method

Step 1. Calculate J and H.
Step 2. Calculate $H^{-1}$.
Step 3. Calculate:

$$\Omega = (H^{-1}J^T).$$

Step 4. Calculate:

$$\Lambda^{-1} = J(H^{-1}J^T).$$

Step 5. Calculate:

$$\Lambda = (JH^{-1}J^T)^{-1}.$$ 

that this is not the most efficient approach, it is in standard use today and may serve as a benchmark for new computational approaches.

The complete algorithm for the Explicit Inversion/Multiplication Method for calculating the operational space inertia matrix of a serial $N$-link manipulator is given in Table 8. Note that for this method, the computation of both the joint space inertia matrix, $H$, and the Jacobian, $J$, is required. The most efficient algorithm for calculating both quantities is the Modified Composite-Rigid-Body Method (Method III) given in Chapter 3. This method is even more efficient than the combination of the Spatial Composite-Rigid-Body Method (Method IV), which is the most efficient algorithm for $H$ alone, and the Orin/Schrader algorithm [32], which is one of the most efficient algorithms for $J$ alone.

The scalar operations (multiplications, additions) required to compute $\Lambda$ and its inverse using the Explicit Inversion/Multiplication Method are shown in Table 9. All operations are given for an $N$ degree-of-freedom manipulator with simple
Table 9: Computations for $\Lambda$ and $\Lambda^{-1}$, Explicit Inversion/Multiplication Method

<table>
<thead>
<tr>
<th>Calculation</th>
<th>#Mult.</th>
<th>#Add.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J, H$</td>
<td>$12\frac{1}{2}N^2 + 24\frac{1}{2}N - 37$</td>
<td>$8N^2 + 40N - 48$</td>
</tr>
<tr>
<td>$H^{-1}$</td>
<td>$\frac{1}{2}N^3 + N^2 - \frac{1}{2}N$</td>
<td>$\frac{1}{2}N^3 - \frac{1}{2}N^2$</td>
</tr>
<tr>
<td>$H^{-1}J^T$</td>
<td>$6N^2$</td>
<td>$6N^2 - 6N$</td>
</tr>
<tr>
<td>$JH^{-1}J^T$</td>
<td>$21N$</td>
<td>$21N - 21$</td>
</tr>
<tr>
<td>$(JH^{-1}J^T)^{-1}$</td>
<td>141</td>
<td>90</td>
</tr>
<tr>
<td><strong>Totals:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Omega$</td>
<td>$\frac{1}{2}N^3 + 19\frac{1}{2}N^2 + 24N - 37$</td>
<td>$\frac{1}{2}N^3 + 13\frac{1}{2}N^2 + 34N - 48$</td>
</tr>
<tr>
<td>$\Lambda^{-1}$</td>
<td>$\frac{1}{2}N^3 + 19\frac{1}{2}N^2 + 45N - 37$</td>
<td>$\frac{1}{2}N^3 + 13\frac{1}{2}N^2 + 55N - 69$</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>$\frac{1}{2}N^3 + 19\frac{1}{2}N^2 + 45N + 104$</td>
<td>$\frac{1}{2}N^3 + 13\frac{1}{2}N^2 + 55N + 21$</td>
</tr>
</tbody>
</table>
revolute and/or prismatic joints. The Modified Composite-Rigid-Body Method is used to compute \( \mathbf{J} \) and \( \mathbf{H} \). An inversion procedure based on Gaussian elimination is used to compute \( \mathbf{H}^{-1} \) [9,37], in which the number of required computations is reduced by considering the symmetry of \( \mathbf{H} \). A similar reduction in operations is also realized when computing the symmetric product \( \mathbf{J} (\mathbf{H}^{-1} \mathbf{J}^T) \). Note that the final inversion of the product, \( \mathbf{A}^{-1} \), may be done at a constant cost because the operational space inertia matrix is always \( 6 \times 6 \). Of course, the inversion is only possible if the rank of \( \mathbf{A}^{-1} \) is equal to 6. The computational complexity of the complete algorithm is \( O(N^3) \) due to the explicit inversion of the \( N \times N \) joint space inertia matrix. Note that the computations for \( \mathbf{\Omega} \), an intermediate result of this algorithm, are also specified. The significance of \( \mathbf{\Omega} \) in the development of dynamic simulation algorithms for closed-chain systems will be demonstrated in later chapters.

4.6 The Unit Force Method

A more efficient algorithm may be developed for computing \( \mathbf{A} \) and its inverse by considering the dynamics of a simplified single chain system. Because \( \mathbf{A} \) is a function of position only, certain terms in the dynamic equations of motion may be neglected without affecting its value. Specifically, if the actuator torques/forces, \( \tau \), the joint rates, \( \dot{\mathbf{q}} \), the gravity forces, \( \mathbf{G} \), and the motion of the base are all set to zero, the value of \( \mathbf{A} \) will remain unchanged. Using Eqs. (4.4) and (4.7), we may obtain the following simplified dynamic equation under these conditions:

\[
\mathbf{H} \ddot{\mathbf{q}} = \mathbf{J}^T \mathbf{F}.
\]  

(4.21)
If a spatial unit force is applied to the end effector along its first axis of motion, that is, if
\[ F = \hat{e}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \] \hfill (4.22)
then Eq. (4.21) becomes:
\[ \mathbf{H} \ddot{q} = (J^T)^{(1)}, \] \hfill (4.23)
where \((J^T)^{(1)}\) is the first column of \(J^T\). For this unit force, the solution for the joint acceleration vector is:
\[ \ddot{q}^{(1)} = (\mathbf{H}^{-1} J^T)^{(1)}, \] \hfill (4.24)
where, in a similar fashion, \((\mathbf{H}^{-1} J^T)^{(1)}\) represents the first column of \((\mathbf{H}^{-1} J^T)\). Note that the explicit inversion of the joint space inertia matrix, \(\mathbf{H}\), is not actually required here, since the solution for \(\ddot{q}\) in Eq. (4.23) may be obtained via linear system solution. By applying the series of spatial unit force vectors,
\[ F = \{\hat{e}_i : i = 1, \ldots, 6\}, \] \hfill (4.25)
at the end effector, the six columns of \((\mathbf{H}^{-1} J^T)\) may be computed one at a time. Thus, the complete solution for this matrix may be constructed as follows:
\[ (\mathbf{H}^{-1} J^T) = \begin{bmatrix} \ddot{q}^{(1)} & \ddot{q}^{(2)} & \ddot{q}^{(3)} & \ddot{q}^{(4)} & \ddot{q}^{(5)} & \ddot{q}^{(6)} \end{bmatrix}, \] \hfill (4.26)
where the \(\ddot{q}^{(i)}\) are the solutions to the set of equations:
\[ \mathbf{H} \ddot{q}^{(i)} = (J^T)^{(i)}; \quad i = 1, \ldots, 6. \] \hfill (4.27)
Once \((\mathbf{H}^{-1} J^T)\) is known, we may complete the solution for \(\Lambda^{-1}\) via simple matrix multiplication. That is,
\[ \Lambda^{-1} = J (\mathbf{H}^{-1} J^T), \] \hfill (4.28)
Table 10: Algorithm for the Unit Force Method

Step 1. Calculate $\mathbf{J}$ and $\mathbf{H}$.
Step 2. Set for all $i = 1, \ldots, 6$:

$$\mathbf{F}_i = \mathbf{\dot{e}}_i,$$

$$(\mathbf{J}^T)^{(i)} = \mathbf{J}^T \mathbf{F}_i.$$  

Step 3. Solve for all $i = 1, \ldots, 6$:

$$\mathbf{H} \mathbf{\ddot{q}}^{(i)} = (\mathbf{J}^T)^{(i)}.$$  

Then,

$$\mathbf{\Omega} = (\mathbf{H}^{-1} \mathbf{J}^T) = \begin{bmatrix} \mathbf{\ddot{q}}^{(1)} & \mathbf{\ddot{q}}^{(2)} & \mathbf{\ddot{q}}^{(3)} & \mathbf{\ddot{q}}^{(4)} & \mathbf{\ddot{q}}^{(5)} & \mathbf{\ddot{q}}^{(6)} \end{bmatrix}.$$  

Step 4. Calculate:

$$\mathbf{\Lambda}^{-1} = \mathbf{J} (\mathbf{H}^{-1} \mathbf{J}^T).$$  

Step 5. Calculate:

$$\mathbf{\Lambda} = (\mathbf{J} \mathbf{H}^{-1} \mathbf{J}^T)^{-1}.$$  

and finally,

$$\mathbf{\Lambda} = (\mathbf{J} \mathbf{H}^{-1} \mathbf{J}^T)^{-1}. \quad (4.29)$$  

Because this approach is based on the application of spatial unit forces at the end effector, we will call it the Unit Force Method. The complete algorithm for the Unit Force Method for calculating the operational space inertia matrix of a serial $N$-link manipulator is given in Table 10. Note, once again, that the Jacobian and joint space inertia matrices, $\mathbf{J}$ and $\mathbf{H}$, must be computed first, and that $\mathbf{\Omega}$ is an intermediate result of the algorithm.
The scalar operations (multiplications, additions) required to compute $\Lambda$ and its inverse using the Unit Force Method are shown in Table 11. All operations are given for an $N$ degree-of-freedom manipulator with simple revolute and/or prismatic joints. As before, the Modified Composite-Rigid-Body Method of Chapter 3 is used to compute $J$ and $H$. The computational complexity of this algorithm is also $O(N^3)$. However, the coefficient of $N^3$ has been reduced by avoiding the explicit inversion of $H$. Step 3 of the algorithm is completed using an approach which combines the Cholesky decomposition method [11] and Gaussian elimination [37]. The initial decomposition of $H$ is only done once for all six iterations of Step 3. A back substitution procedure is then carried out six times, once for each column of $J^T$. The computations required for $\Omega$ are also specified in this table.

4.7 The Force Propagation Method

Although the two preceding algorithms for $\Lambda$ are straightforward, the computational order of each is high. This section will present the development of a linear recursive algorithm for computing $\Lambda^{-1}$, the inverse operational space inertia matrix of a single chain, which has a reduced computational order. We will begin our analysis with the recursive articulated-body dynamic equations for an unconstrained manipulator as derived by Featherstone [23]. These equations will then be extended to describe the dynamics of a manipulator which is constrained at the tip. From these new recursive equations for a constrained chain, the Force Propagation Method for computing $\Lambda^{-1}$ will be derived.

The analysis of this section will be based on the simple configuration shown in Fig. 7. This figure depicts a serial-link manipulator chain with no internal closed loops. The joints are arbitrary, and they are modelled using the general joint model of Chapter 2. The base member is fixed to the inertial frame. The spatial
Table 11: Computations for $\mathbf{A}$ and $\mathbf{A}^{-1}$, Unit Force Method

<table>
<thead>
<tr>
<th>Calculation</th>
<th>#Mult.</th>
<th>#Add.</th>
<th>#Mult. $(N = 6)$</th>
<th>#Add. $(N = 6)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{J, H}$</td>
<td>$12\frac{1}{2}N^2 + 24\frac{1}{2}N - 37$</td>
<td>$8N^2 + 40N - 48$</td>
<td>560</td>
<td>480</td>
</tr>
<tr>
<td>$\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(6)}$</td>
<td>$\frac{1}{6}N^3 + 6\frac{1}{2}N^2 - \frac{2}{3}N$</td>
<td>$\frac{1}{6}N^3 + 6N^2 - \frac{6}{6}N$</td>
<td>266</td>
<td>215</td>
</tr>
<tr>
<td>$\mathbf{JH}^{-1}\mathbf{J}^T$</td>
<td>$21N$</td>
<td>$21N - 21$</td>
<td>126</td>
<td>105</td>
</tr>
<tr>
<td>$(\mathbf{JH}^{-1}\mathbf{J}^T)^{-1}$</td>
<td>141</td>
<td>90</td>
<td>141</td>
<td>90</td>
</tr>
<tr>
<td><strong>Totals:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathbf{\Omega}$</td>
<td>$\frac{1}{6}N^3 + 19N^2 + 23\frac{5}{6}N - 37$</td>
<td>$\frac{1}{6}N^3 + 14N^2 + 33\frac{5}{6}N - 48$</td>
<td>826</td>
<td>695</td>
</tr>
<tr>
<td>$\mathbf{\Lambda}^{-1}$</td>
<td>$\frac{1}{6}N^3 + 19N^2 + 44\frac{5}{6}N - 37$</td>
<td>$\frac{1}{6}N^3 + 14N^2 + 54\frac{5}{6}N - 69$</td>
<td>952</td>
<td>800</td>
</tr>
<tr>
<td>$\mathbf{\Lambda}$</td>
<td>$\frac{1}{6}N^3 + 19N^2 + 44\frac{5}{6}N + 104$</td>
<td>$\frac{1}{6}N^3 + 14N^2 + 54\frac{5}{6}N + 21$</td>
<td>1093</td>
<td>890</td>
</tr>
</tbody>
</table>
force vector, \( f \), represents the vector of forces and moments applied by the tip of the chain to the environment. For an open chain, \( f \) is identically zero. For a constrained chain, however, \( f \) is unknown, and in general, nonzero.

4.7.1 Recursive Dynamic Equations for an Open Chain

Recursive dynamic equations for a single open chain are derived by Featherstone in [23] and later presented again by Brandl, et al. in [10]. The formulation of these equations is based on the concept of treating a chain of rigid bodies connected by joints as a single "articulated body". The recursive equations allow the dynamics of the entire chain to be resolved to a single link called the "handle" of the articulated body. All interactions with the articulated body are assumed to occur through this link. The articulated-body dynamic equation for the handle incorporates the dynamic information of the entire chain in a single equation. Thus, the articulated-body formulation may be used to simplify the dynamic analysis of multibody systems by reducing the number of equations necessary to describe the
Featherstone defines the articulated-body inertia of a link as the $6 \times 6$ matrix which relates a spatial force applied to the link and the spatial acceleration of the link, taking into account the dynamics of the rest of the articulated body. This relationship is linear, and, for link $i$, it may be written as follows [23]:

$$I_i^* a_i = f_i,$$

where $I_i^*$ is the articulated-body inertia of link $i$ (notation from [10]), $a_i$ is the spatial acceleration of link $i$, and $f_i$ is the spatial force exerted on link $i$. For simplicity, the bias force terms due to the joint velocities, applied joint torques and/or forces, and gravity are assumed to be zero.

A number of important recurrence relations in the articulated-body formulation may be derived by analyzing a simple two-link articulated body as shown in Fig. 8. This articulated body is made up of links 1 and 2 with spatial link inertias of $I_1$ and $I_2$, respectively. The two links are connected by a general joint, joint
2, which is characterized by its motion space, $\phi_2$. Recall from Chapter 2 that $\phi_2$ represents the free modes of this joint. In this analysis, it is assumed that the two bodies are initially at rest and experience no applied joint forces or torques on or about $\phi_2$ ($\tau_2 = 0$). The tip force, $f$, is zero because the chain is unconstrained. The only spatial force vector exerted on the articulated body is $f_1$, which is applied to link 1, the handle of the articulated body. For the moment, we will assume that the physical parameters describing the links and joints, including the spatial link inertias, are given in absolute coordinates.

The development of the recursive articulated-body dynamic equations begins with the simple free-body dynamic equations of the two individual links. In the notation of this dissertation, we may write the free-body equation for link 1 as follows:

$$ I_1 \mathbf{a}_1 = f_1 - f_2, \quad (4.31) $$

where $\mathbf{a}_1$ is the spatial acceleration of link 1, and $f_2$ is the spatial force exerted on link 2 by link 1. Similarly, the free-body equation for link 2 may be written:

$$ I_2 \mathbf{a}_2 = f_2, \quad (4.32) $$

where $\mathbf{a}_2$ is the spatial acceleration of link 2, and $f$ is assumed to be zero for the unconstrained case. The spatial acceleration of link 2 is related to the spatial acceleration of link 1 as follows:

$$ \mathbf{a}_2 = \mathbf{a}_1 + \phi_2 \ddot{q}_2, \quad (4.33) $$

where $\ddot{q}_2$ is the vector of relative joint accelerations in the free directions of joint 2.

The component of a spatial vector which lies on or along a specific axis of a joint may be determined by performing a simple dot product between that spatial
vector and the vector which represents the joint axis. This may be referred to as “projecting the spatial vector onto the joint axis”. Thus, we may project the terms of Eq. (4.32) onto the motion space of joint 2 via the following spatial dot product:

$$\phi_2^T I_2 a_2 = \phi_2^T f_2,$$

(4.34)

where

$$\phi_2^T f_2 = \tau_2 = 0.$$  

(4.35)

We may combine Eqs. (4.33) through (4.35) to solve for the vector of relative joint accelerations, $\ddot{q}_2$, in terms of the spatial acceleration of link 1, $a_1$, as follows:

$$\ddot{q}_2 = - [\phi_2^T I_2 \dot{\phi}_2] a_1.$$  

(4.36)

By combining Eqs. (4.33) and (4.36), we may express $a_2$ as a function of $a_1$:

$$a_2 = [1 - \phi_2 (\phi_2^T I_2 \phi_2)^{-1} \phi_2^T I_2] a_1.$$  

(4.37)

This is a simple recursive acceleration equation for the unconstrained articulated body. Equation (4.37) may be combined with Eqs. (4.31) and (4.32) to give the following dynamic equation for link 1, the handle of the unconstrained articulated body:

$$[I_1 + I_2 - I_2 \phi_2 (\phi_2^T I_2 \phi_2)^{-1} \phi_2^T I_2] a_1 = f_1,$$

(4.38)

or

$$I_1^* a_1 = f_1,$$

(4.39)

where

$$I_1^* = I_1 + I_2 - I_2 \phi_2 (\phi_2^T I_2 \phi_2)^{-1} \phi_2^T I_2$$

(4.40)

is the articulated-body inertia of link 1. Note from Eq. (4.32) that the articulated-body inertia of link 2, $I_2^*$, is just the spatial link inertia of link 2. That is,

$$I_2^* = I_2.$$  

(4.41)

69
Thus,

\[ I_i^* = I_1 + I_2^* - I_2^* \phi_2 \cdot (\phi_2^T I_2^* \phi_2)^{-1} \phi_2^T I_2^* \]  

(4.42)

is a simple recursive equation for the articulated-body inertia of the first link of the unconstrained two-link articulated body.

The equations given above may be generalized for the case of an unconstrained articulated body with an arbitrary number of links. The dynamic equation for link \( i \) as the handle of the unconstrained articulated body, ignoring bias terms, is written in Eq. (4.30), repeated here for convenience:

\[ I_i^* a_i = f_i, \]  

(4.43)

where \( I_i^* \) is the articulated-body inertia of link \( i \). The defining recursive equation for \( I_i^* \) is:

\[ I_i^* = I_i + I_{i+1}^* - I_{i+1}^* \phi_{i+1} (\phi_{i+1}^T I_{i+1}^* \phi_{i+1})^{-1} \phi_{i+1}^T I_{i+1}^*. \]  

(4.44)

The starting value of this recursion is:

\[ I_N^* = I_N \]  

(4.45)

for a manipulator chain with \( N \) links. If we define:

\[ K_i^* = \phi_i (\phi_i^T I_i^* \phi_i)^{-1} \phi_i^T, \]  

(4.46)

and

\[ L_i^* = (1 - K_i^* I_i^*), \]  

(4.47)

then Eq. (4.44) may be written more simply as:

\[ I_i^* = I_i + I_{i+1}^* L_{i+1}^*. \]  

(4.48)
Note the symmetry of both $\mathbf{K}_i^*$ and $\mathbf{I}_i^*$. The quantity $\mathbf{K}_i^*$ is, in essence, an inverse inertia, while the form of $\mathbf{L}_i^*$ makes it dimensionless. The recursive acceleration equation, Eq. (4.37), may also be generalized as follows:

$$a_i = \left[ \mathbf{I} - \phi_i (\phi_i^T \mathbf{I}_i^* \phi_i)^{-1} \phi_i^T \mathbf{I}_i^* \right] a_{i-1}, \quad (4.49)$$

$$= (1 - \mathbf{K}_i^* \mathbf{I}_i^*) a_{i-1}, \quad (4.50)$$

or

$$a_i = (\mathbf{L}_i^*) a_{i-1}. \quad (4.51)$$

The starting value for this recursion is simply $a_0$, the spatial acceleration of the base member.

In Eq. (4.51), $\mathbf{L}_i^*$ acts as a spatial transformation which propagates the spatial acceleration vector, $a_{i-1}$, across joint $i$. We will call a matrix which transforms spatial vectors across actuated joint structures a spatial "articulated transformation". In general, an articulated transformation is a nonlinear function of the articulated-body inertia and is a dimensionless $6 \times 6$ matrix. Featherstone calls the articulated transformation, $\mathbf{L}_i^*$, the "acceleration propagator" [23]. It relates the spatial acceleration of one link of an articulated body to the spatial acceleration of a neighboring link in the same articulated body (ignoring bias terms). Also note the appearance of this same articulated transformation in the recursive equation for $\mathbf{I}_i^*$, Eq. (4.48). In that expression, $\mathbf{L}_i^* \mathbf{I}_{i+1}$ propagates the articulated-body inertia of link $(i+1)$ across joint $(i+1)$ before it is combined with the spatial inertia of link $i$. It is obvious that articulated transformations are valuable in the formulation of these simple articulated-body dynamic equations. In the following pages, the significance of articulated transformations will become even more evident.

Equations (4.43), (4.46)-(4.48), and (4.51) summarize the fundamental and relevant recursive dynamic equations for an unconstrained single chain. Additional
recursive equations for such a configuration may be found in [10] and [23].

4.7.2 Recursive Dynamic Equations for a Constrained Chain

We will now extend the recursive articulated-body dynamic equations for an open chain to describe the dynamics of a chain which is constrained at the tip. For this task, we will refer again to Fig. 8. Now, however, we will assume that \( f \), the spatial force exerted by the tip, is nonzero.

The free-body equations for links 1 and 2 are now written as follows:

\[
I_1 a_1 = f_1 - f_2, \hspace{1cm} (4.52)
\]

and

\[
I_2 a_2 = f_2 - f, \hspace{1cm} (4.53)
\]

respectively. The spatial accelerations of the two links are still related by Eq. (4.33). By projecting Eq. (4.53) onto the motion space of joint 2, and combining this result with Eqs. (4.33) and (4.35), we may once again solve for the vector of relative joint accelerations as follows:

\[
\ddot{q}_2 = -\left[ (\phi_2^T I_2 \phi_2)^{-1} \phi_2^T I_2 \right] a_1 - \left[ (\phi_2^T I_2 \phi_2)^{-1} \phi_2^T I_2 \right] f. \hspace{1cm} (4.54)
\]

Note that \( \ddot{q}_2 \) is now a function of both \( a_1 \) and \( f \). Combining Eqs. (4.33) and (4.54), we may express \( a_2 \) as a function of these same two vectors:

\[
a_2 = \left[ 1 - \phi_2 (\phi_2^T I_2 \phi_2)^{-1} \phi_2^T I_2 \right] a_1 - \left[ \phi_2 (\phi_2^T I_2 \phi_2)^{-1} \phi_2^T I_2 \right] f. \hspace{1cm} (4.55)
\]

Note the similarity between Eqs. (4.37) and (4.55). Equation (4.55) may be combined with Eqs. (4.52) and (4.53) to obtain the dynamic equation for link 1, the handle of the constrained articulated body:

\[
\left[ I_1 + I_2 - I_2 \phi_2 (\phi_2^T I_2 \phi_2)^{-1} \phi_2^T I_2 \right] a_1 = f_1 - (1 - I_2 K_2) f, \hspace{1cm} (4.56)
\]
or

\[ I_1^* a_1 = f_1 - f^*. \]  \hspace{1cm} (4.57)

Note that \( I_1^* \), the articulated-body inertia of link 1, is the same whether the chain is constrained or unconstrained. The force vector, \( f^* \), defined by the equation:

\[ f^* = (1 - I_2 K_2) f, \]  \hspace{1cm} (4.58)

represents the spatial contact force vector, propagated from the tip of link 2 to the tip of link 1. The coefficient matrix, \((1 - I_2 K_2)\), transforms the force across the connecting joint, joint 2. Note the similarity between this transformation matrix and the acceleration propagator discussed earlier.

The equations given above may be generalized for the case of a constrained articulated body with an arbitrary number of links. The dynamic equation for link \( i \) as the handle of the constrained articulated body may be written as follows:

\[ I_i^* a_i = f_i - f^*_{i+1}, \]  \hspace{1cm} (4.59)

where, as in the unconstrained case,

\[ I_i^* = I_i + I_{i+1}^* L_{i+1}^* \]  \hspace{1cm} (4.60)

is the recursive equation for the articulated-body inertia of link \( i \). Once again, the starting value for this recurrence relation is:

\[ I_N^* = I_N. \]  \hspace{1cm} (4.61)

From Eq. (4.58), we may define the following recurrence relation for \( f_i^* \):

\[ f_i^* = (1 - I_i^* K_i^*) f_{i+1}^*. \]  \hspace{1cm} (4.62)
The vector \( \mathbf{f}_i^* \) represents the spatial contact force vector, \( \mathbf{f} \), propagated down from the tip of the chain, across joint \( i \), to the tip of link \( i - 1 \). We will now define the spatial articulated transformation,

\[
P_i^* = (1 - \mathbf{I}_i^* \mathbf{K}_i^*),
\]

as the "force propagator" across joint \( i \). This matrix transforms a spatial force from one link of an articulated body across the connecting joint structure to the tip of the preceding link. Note that

\[
P_i^* = (1 - \mathbf{I}_i^* \mathbf{K}_i^*),
\]

\[
= (1 - \mathbf{K}_i^* \mathbf{I}_i^*)^T,
\]

\[
= (\mathbf{L}_i^*)^T.
\]

The force propagator across joint \( i \) is simply the transpose of the acceleration propagator across joint \( i \). Thus, we may rewrite Eq. (4.62) as follows:

\[
\mathbf{f}_i^* = (\mathbf{L}_i^*)^T \mathbf{f}_{i+1}^*.
\]

At the tip of the chain, the spatial contact force is applied directly by the last link. Thus, the starting value for the recursive force equation is:

\[
f_{N+1}^* = \mathbf{f} = \mathbf{f}_{N+1},
\]

where \( \mathbf{f}_{N+1} \) is the spatial force exerted by link \( N \) on any constraining body (link \( N + 1 \)). Note that the recursions given in Eqs. (4.60) and (4.67) both begin at the last link of the chain and proceed back toward the base.

The recursive acceleration equation, Eq. (4.55), may also be generalized as follows:

\[
a_i = \left[ 1 - \phi_i \left( \phi_i^T \mathbf{I}_i^* \phi_i \right)^{-1} \phi_i^T \right] a_{i-1} - \left[ \phi_i \phi_i^T \mathbf{I}_i^* \phi_i \right] \mathbf{f}_{i+1}^*,
\]

where \( \phi_i \) represents the spatial coordinate vector of link \( i \).
The spatial acceleration of link $i$ is now a function of the spatial acceleration of the preceding link, $a_{i-1}$, and the spatial contact force propagated back to the tip of link $i$. Note that, given $f_{i+1}^*$, this recursion progresses from the base of the chain to the last link.

Equations (4.59), (4.60), (4.67), and (4.70) summarize the fundamental and relevant recursive dynamic equations for a constrained single chain. These equations will now be used to derive the Force Propagation Method for computing the inverse operational space inertia matrix of a single chain.

4.7.3 Development of the Force Propagation Method

Given the recursive dynamic equations for a constrained chain, we will now begin the development of a linear recursive algorithm for $\Lambda^{-1}$, the inverse operational space inertia matrix of a single chain. First, we will define a new quantity, $(\Lambda^*)^{-1}$, an inertial matrix which relates the spatial acceleration of a link and the propagated spatial contact force exerted at the tip of the same link. We may write a defining equation for this matrix (at link $i$) as follows:

$$a_i = - (\Lambda_i^*)^{-1} f_{i+1}^*,$$  \hspace{1cm} (4.71)

where $a_i$ is the spatial acceleration of link $i$, and $f_{i+1}^*$ is the propagated spatial contact force exerted at the tip of link $i$. Note that when this equation is written for the last link of the chain, link $N$, we obtain:

$$a_N = - (\Lambda_N^*)^{-1} f_{N+1}^*,$$  \hspace{1cm} (4.72)

or, referring to Eq. (4.68),

$$a_N = - (\Lambda_N^*)^{-1} f_{N+1} = - (\Lambda_N^*)^{-1} f.$$  \hspace{1cm} (4.73)

75
By definition, \( \Lambda^{-1} \) relates the spatial acceleration of the last link or tip, which we may call \( a_N \), and the spatial force exerted by that link on the environment, \( f \). That is,

\[
a_N = -\Lambda^{-1} f. \tag{4.74}
\]

By comparing Eqs. (4.73) and (4.74), we may conclude that:

\[
(\Lambda_N^*)^{-1} = \Lambda^{-1}. \tag{4.75}
\]

Thus, at the last link of the chain, \((\Lambda_N^*)^{-1}\) and \(\Lambda^{-1}\), the inverse operational space inertia matrix of the chain, are equivalent.

Our goal in this section is to develop a recursive algorithm for \((\Lambda^*)^{-1}\) which will, by virtue of the previous discussion, lead to the computation of \(\Lambda^{-1}\). To determine an initial condition for this recursion, we will begin at the base of the chain. For the base, link 0, Eq. (4.71) has the form:

\[
a_0 = -(\Lambda^*_0)^{-1} f_1^*, \tag{4.76}
\]

where \( f_1^* \) is the spatial contact force propagated back across joint 1 to the "tip" of the base member. Since the base is fixed to the inertial frame, we have:

\[
a_0 = 0. \tag{4.77}
\]

In general, \( f_1^* \) is finite and nonzero. Thus, we conclude:

\[
(\Lambda^*_0)^{-1} = 0. \tag{4.78}
\]

This is the initial condition in the recursion for \((\Lambda^*)^{-1}\).

Let us now proceed to the first link in the chain, link 1. From Eq. (4.70), we may write the spatial acceleration of link 1 as follows:

\[
a_1 = (L_1^*) a_0 - (K_1^*) f_2^*. \tag{4.79}
\]
Since \( a_0 = 0 \), this becomes:
\[
a_1 = -(K^*_1) f^*_2. \tag{4.80}
\]

From Eq. (4.71),
\[
a_1 = -(\Lambda^*_1)^{-1} f^*_2. \tag{4.81}
\]

Thus,
\[
(\Lambda^*_1)^{-1} = K^*_1, \tag{4.82}
\]

where \( K^*_1 \) is a function of \( I^*_1 \). At link 2, we may write the spatial acceleration as follows:
\[
a_2 = (L^*_2) a_1 - (K^*_2) f^*_3. \tag{4.83}
\]
Substituting Eq. (4.81) for \( a_1 \), we obtain:
\[
a_2 = -(L^*_2)(\Lambda^*_1)^{-1} f^*_2 - (K^*_2) f^*_3. \tag{4.84}
\]

We will now make use of the force propagator concept developed earlier. The propagated spatial contact force, \( f^*_2 \), is related to the propagated spatial contact force, \( f^*_3 \), by means of the force propagator, \( (L^*_2)^T \), as follows:
\[
f^*_2 = (L^*_2)^T f^*_3. \tag{4.85}
\]
Using this relationship in Eq. (4.84) gives:
\[
a_2 = -(L^*_2)(\Lambda^*_1)^{-1} (L^*_2)^T f^*_3 - (K^*_2) f^*_3, \tag{4.86}
\]
\[
= - [K^*_2 + (L^*_2)(\Lambda^*_1)^{-1}(L^*_2)^T] f^*_3. \tag{4.87}
\]

Since
\[
a_2 = -(\Lambda^*_2)^{-1} f^*_3, \tag{4.88}
\]
then
\[
(\Lambda^*_2)^{-1} = [K^*_2 + (L^*_2)(\Lambda^*_1)^{-1}(L^*_2)^T]. \tag{4.89}
\]
It appears that $(\Lambda_i^*)^{-1}$ is propagated across joint $2$ via the pre- and postmultiplication by the acceleration and force propagators, $(L_2^*)$ and $(L_2^*)^T$, respectively. This propagated matrix is then combined with $K_2^*$, which is associated with the inverse articulated-body inertia of link $2$.

In general, we may define the following recursive equation for $(\Lambda^*)^{-1}$:

\[ (\Lambda_i^*)^{-1} = K_i^* + (L_i^*) (\Lambda_{i-1}^*)^{-1} (L_i^*)^T, \]  

(4.90)

where $K_i^*$ and $L_i^*$ are defined in Eqs. (4.46) and (4.47), respectively. The starting value for this recursion, as discussed above, is:

\[ (\Lambda_0^*)^{-1} = 0. \]  

(4.91)

In the previous development, it was assumed that all vector and matrix quantities were expressed in absolute coordinates. Fortunately, the transformation of Eq. (4.90) to local (link) coordinates is a simple matter. The following matrix multiplication may be used to transform $(\Lambda^*)^{-1}$ between coordinate systems:

\[ (i^* \Lambda_{i-1}^*)^{-1} = i^* X_{i-1} (i^{-1} \Lambda_{i-1}^*)^{-1} i^* X_{i-1}^T, \]  

(4.92)

where $(i^{-1} \Lambda_{i-1}^*)^{-1}$ is expressed in the $(i - 1)$st coordinate frame, and $(i^* \Lambda_{i-1}^*)^{-1}$ is expressed in the $i$th coordinate frame. Therefore, Eq. (4.90) may be rewritten in link coordinates as follows:

\[ (i^* \Lambda_i^*)^{-1} = i^* K_i^* + (i^* L_i^*) i^* X_{i-1} (i^{-1} \Lambda_{i-1}^*)^{-1} i^* X_{i-1}^T (i^* L_i^*)^T. \]  

(4.93)

As shown in Eq. (4.75), the final iteration of this recursion at link $N$ gives the inverse operational space inertia matrix, $\Lambda^{-1}$, for the chain. That is,

\[ \Lambda^{-1} = N \Lambda_N^{-1} = (N^* \Lambda_N^*)^{-1}. \]  

(4.94)
If $\Lambda^{-1}$ is desired at some point on link $N$ other than the $N$th coordinate origin, then the matrix transformation of Eq. (4.92) may be invoked once again. For example, if $\Lambda^{-1}$ is desired at the origin of a coordinate system called $N+1$ which is fixed to link $N$, it may be computed from $^N\Lambda_N^{-1}$ as follows:

$$
\Lambda^{-1} = ^{N+1}\Lambda_N^{-1} = ^{N+1}X_N (^N\Lambda_N^{-1})^{N+1}X^T_N.
$$

(4.95)

Based on the use of the force propagator concept, we will call this recursive approach the Force Propagation Method. The complete algorithm for the Force Propagation Method for computing the inverse operational space inertia matrix of a serial $N$-link manipulator is given in Table 12. The structure of the algorithm is a double recursion. In the first recursion, the articulated-body inertia of each link is calculated, starting at the last link and moving back to the first link (Backward Recursion). Note that we do not need to calculate the articulated-body inertia of the base member. The (transposed) force propagators are also computed in the first recursion. The second recursion begins at link 1 and moves out to the last link (Forward Recursion), computing $(\Lambda_i^*)^{-1}$ at each link. When this recursion reaches link $N$, the inverse operational space inertia matrix of the entire chain, $\Lambda^{-1}$, is known. The operational space inertia matrix, $\Lambda$, may be computed by explicit matrix inversion if $\Lambda^{-1}$ is nonsingular.

The scalar operations (multiplications, additions) required to compute $\Lambda^{-1}$ and $\Lambda$ using the Force Propagation Method are shown in Table 13. These scalar operations are given for an $N$ degree-of-freedom manipulator with simple revolute and/or prismatic joints. Note that $I_N$, $K_N^*$, and $L_N^*$ may all be computed off-line, and that the initial condition, $(\Lambda_0^*)^{-1} = 0$, allows some simplification in the first iteration of the Forward Recursion. The computational complexity of the complete algorithm is $O(N)$, an improvement over the previous two algorithms. The efficient
Table 12: Algorithm for the Force Propagation Method

Given:

\[ \phi_i, I_i, X_{i-1} \text{ for } i = 1, \ldots, N; \]
\[ I_N^* = I_N; \]
\[ K_N^* = \phi_N (\phi_N^T I_N \phi_N)^{-1} \phi_N^T; \]
\[ L_N^* = (1 - K_N I_N); \]
\[ (0 \Lambda^*_0)^{-1} = 0; \]

**Backward Recursion**

Calculate for all \( i = (N - 1), \ldots, 1: \)

\[ I_i^* = I_i + (i+1)X_i^T (I_{i+1}^* L_{i+1}^* )^{i+1} X_i, \]
\[ K_i^* = \phi_i (\phi_i^T I_i^* \phi_i)^{-1} \phi_i^T, \]
\[ L_i^* = (1 - K_i^* I_i^*). \]

**Forward Recursion**

Calculate for all \( i = 1, \ldots, N: \)

\[ (i \Lambda_i^*)^{-1} = K_i^* + (L_i^* )^i X_{i-1} (i-1 \Lambda_{i-1}^*)^{-1} i X_{i-1}^T (L_i^* )^T, \]
\[ \Lambda^{-1} = (N \Lambda_N^*)^{-1}. \]
Table 13: Computations for $\Lambda$ and $\Lambda^{-1}$, Force Propagation Method

<table>
<thead>
<tr>
<th>Recursion</th>
<th>#Mult. $(N = 6)$</th>
<th>#Add. $(N = 6)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backward</td>
<td>136$N - 156$</td>
<td>126$N - 141$</td>
</tr>
<tr>
<td>Forward</td>
<td>167$N - 264$</td>
<td>153$N - 271$</td>
</tr>
<tr>
<td>Totals:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Lambda^{-1}$</td>
<td>303$N - 420$</td>
<td>279$N - 412$</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>303$N - 279$</td>
<td>279$N - 322$</td>
</tr>
</tbody>
</table>

Coordinate transformations described in Chapter 3 are utilized in every case.

Note that $\Omega$ is not an immediate result of the equations for $\Lambda^{-1}$, unlike the previous two algorithms. However, a double recursion may also be defined to compute $\Omega$ which uses many of the computational terms already defined in Table 12. We may generalize Eq. (4.54) for the $i$th joint acceleration vector, $\ddot{q}_i$, (in absolute coordinates), as follows:

$$\ddot{q}_i = -[(\dot{\phi}_i^T \dot{I}_i^* \phi_i)^{-1} \phi_i^T] I_i^* a_{i-1} - [(\phi_i^T I_i^* \phi_i)^{-1} \phi_i^T] f_{i+1}^*.$$

Combining Eq. (4.96) with Eqs. (4.67) and (4.71), we may relate $\ddot{q}_i$ to $f_{i+1}^*$ directly:

$$\ddot{q}_i = -[(\phi_i^T I_i^* \phi_i)^{-1} \phi_i^T] [I - (I_i^*) (\Lambda_i^{-1})^{-1} (I_i^*)^T] f_{i+1}^*,$$

where

$$f_{i+1}^* = (L_{i+1}^*)^T f_{i+2}^*,$$

$$= (L_{i+1}^*)^T (L_{i+2}^*)^T \cdots (L_N^*)^T f_{N+1}^*.$$

Thus, we have derived a relationship between the relative joint acceleration vector at joint $i$ and the spatial contact force vector at the tip of the chain. The quantity
Figure 9: Serial-Link Chain for the Inertia Propagation Method

Ω is the composite matrix which relates the relative joint acceleration vector for the entire chain to this same spatial contact force. Given the recursive equations for $\Lambda^{-1}$ in Table 12, the additional equations required to compute $\Omega$ are presented in Table 14. The additional scalar operations required to compute $\Omega$ are shown in Table 15.

4.8 The Inertia Propagation Method

In this section, a linear recursive algorithm will be developed for computing $\Lambda$, the operational space inertia matrix of a single chain. The following analysis will be based on the simple configuration shown in Fig. 9. This figure depicts a serial-link chain similar to that of Fig. 7. In this case, however, the base member of the chain is not fixed to the inertial frame, but is unconstrained. The following recursive algorithm is also valid for a constrained base member, as long as the operational space inertia matrix of the base member, $\Lambda_0$, is known.

By definition, the operational space inertia matrix of an $i$-link manipulator,
Table 14: Additional Equations for $\Omega$, Force Propagation Method

Given:

$$D_N^* = 1;$$

**Backward Recursion**

Calculate for all $i = (N-1), \ldots, 1$:

$$D_i^* = i+1X_i^T (L_{i+1}^*)^T D_{i+1}^*.$$  

**Forward Recursion**

Calculate for all $i = 1, \ldots, N$:

$$\Omega_{(i)} = (\phi_i^T \Lambda_i^* \phi_i)^{-1} \phi_i^T \left[ 1 - (\Lambda_i^*)^T X_{i-1} (i-1 \Lambda_i^* - 1)^{-1} X_{i-1} (L_i^*)^T \right] D_i^*.$$  

Then,

$$\Omega = \begin{bmatrix} \Omega_{(1)} \\ \Omega_{(2)} \\ \vdots \\ \Omega_{(N)} \end{bmatrix}.$$  

83
Table 15: Additional Computations for $\Omega$, Force Propagation Method

<table>
<thead>
<tr>
<th>Recursion</th>
<th>#Mult.</th>
<th>#Add.</th>
<th>#Mult. $(N = 6)$</th>
<th>#Add. $(N = 6)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backward</td>
<td>$150N - 280$</td>
<td>$102N - 192$</td>
<td>620</td>
<td>420</td>
</tr>
<tr>
<td>Forward</td>
<td>$83N - 77$</td>
<td>$65N - 65$</td>
<td>421</td>
<td>325</td>
</tr>
<tr>
<td>Total:</td>
<td>$233N - 357$</td>
<td>$167N - 257$</td>
<td>1041</td>
<td>745</td>
</tr>
</tbody>
</table>

$\Lambda_i$, is the matrix which relates the spatial acceleration of link $i$ and the spatial force vector exerted at the tip of link $i$. Thus, we may write:

$$\Lambda_i \mathbf{a}_i = -\mathbf{f}_{i+1}, \quad (4.100)$$

where, according to the convention of this dissertation, $\mathbf{f}_{i+1}$ is the spatial force exerted by link $i$ on a constraining body or link.

We will begin our analysis at the base of the chain shown in Fig. 9. For the unconstrained base member, link $0$, the free-body dynamic equation is:

$$I_0 \mathbf{a}_0 = -\mathbf{f}_1, \quad (4.101)$$

and Eq. (4.100) takes the form:

$$\Lambda_0 \mathbf{a}_0 = -\mathbf{f}_1. \quad (4.102)$$

These two equations imply that

$$\Lambda_0 = I_0 \quad (4.103)$$

for the unconstrained base, where $I_0$ is the spatial link inertia of the base member. If the base member is fixed to the inertial frame, then Eqs. (4.101)–(4.103) still hold, but $I_0$ is now the spatial inertia of the earth, which is considered to be
infinite. In this case, we may treat the earth as a "big base" which is floating in space, and approximate $I_0$ as a very large spatial inertia (mass and inertia tensor). If the magnitude of $I_0$ ($\Lambda_0$) is large enough relative to the spatial inertias of the links of the chain, the numerical results for $\Lambda$ at the tip of the chain will be very accurate. This concept has been tested for a specific manipulator (PUMA 560); the results will be summarized at the end of this section when the computational requirements for this algorithm are considered.

Equation (4.103) is the initial condition for this new recursive algorithm. Having defined an initial condition for $\Lambda$ at the base member, we may proceed to link 1. Once again, we will assume that the joint velocities, actuator torques and/or forces, and gravity forces are all zero, and all vectors and matrices are initially defined in absolute coordinates. As before, we may write the free-body dynamic equation for link 1 as follows:

$$ I_1 a_1 = f_1 - f_2, \quad (4.104) $$

where $I_1$ is the spatial inertia matrix of link 1, and $f_2$ is the spatial force vector exerted on link 2 by link 1. Combining the kinematic equation for $a_1$,

$$ a_1 = a_0 + \phi_1 \ddot{q}_1, \quad (4.105) $$

and Eq. (4.102), we obtain an expression for $f_1$:

$$ f_1 = \Lambda_0 (-a_1 + \phi_1 \ddot{q}_1). \quad (4.106) $$

Because

$$ \phi_1^T f_1 = \tau_1 = 0, \quad (4.107) $$

we may write:

$$ \phi_1^T \Lambda_0 (-a_1 + \phi_1 \ddot{q}_1) = 0, \quad (4.108) $$
from which we may solve for the relative joint acceleration vector at joint 1:

$$\ddot{q}_1 = \left[ (\phi_1^T \Lambda_0 \phi_1)^{-1} \phi_1^T \Lambda_0 \right] a_1. \quad (4.109)$$

Combining this result with Eq. (4.106), we may express $f_1$ as follows:

$$f_1 = -\Lambda_0 a_1 + \Lambda_0 \phi_1 \ddot{q}_1, \quad (4.110)$$

$$= -[\Lambda_0 - \Lambda_0 \phi_1 (\phi_1^T \Lambda_0 \phi_1)^{-1} \phi_1^T \Lambda_0] a_1. \quad (4.111)$$

If we use this definition for $f_1$ in Eq. (4.104), we obtain:

$$I_1 a_1 = -[\Lambda_0 - \Lambda_0 \phi_1 (\phi_1^T \Lambda_0 \phi_1)^{-1} \phi_1^T \Lambda_0] a_1 - f_2, \quad (4.112)$$

and upon simplifying, we find:

$$[I_1 + \Lambda_0 - \Lambda_0 \phi_1 (\phi_1^T \Lambda_0 \phi_1)^{-1} \phi_1^T \Lambda_0] a_1 = -f_2. \quad (4.113)$$

Since, by Eq. (4.100), $\Lambda_1$ is defined by the following equation:

$$\Lambda_1 a_1 = -f_2, \quad (4.114)$$

we may identify $\Lambda_1$ from Eq. (4.113) as:

$$\Lambda_1 = I_1 + \Lambda_0 - \Lambda_0 \phi_1 (\phi_1^T \Lambda_0 \phi_1)^{-1} \phi_1^T \Lambda_0, \quad (4.115)$$

$$= I_1 + \Lambda_0 \left[ I - \phi_1 (\phi_1^T \Lambda_0 \phi_1)^{-1} \phi_1^T \Lambda_0 \right]. \quad (4.116)$$

If we define:

$$K_{\Lambda_0} = \phi_1 (\phi_1^T \Lambda_0 \phi_1)^{-1} \phi_1^T, \quad (4.117)$$

and

$$L_{\Lambda_0} = \left[ I - \phi_1 (\phi_1^T \Lambda_0 \phi_1)^{-1} \phi_1^T \Lambda_0 \right], \quad (4.118)$$

$$= (I - K_{\Lambda_0} \Lambda_0), \quad (4.119)$$

86
then we may rewrite Eq. (4.116) as follows:

$$\Lambda_1 = I_1 + \Lambda_0 L_{\Lambda_0}. \quad (4.120)$$

In this simple recursion, the operational space inertia matrix of the base member, $\Lambda_0$, is propagated across joint 1 by $L_{\Lambda_0}$, a new spatial articulated transformation which is very similar in form to the acceleration propagator of the previous section. The propagated matrix is combined with $I_1$, the spatial inertia of link 1 to form $\Lambda_1$, the operational space inertia matrix of the two-link partial chain comprised of links 0 and 1. Note the similarity between this recursive procedure and the structural recursion used to derive the Structurally Recursive Method (Method I) in Chapter 3.

The recursive equations given above may be generalized for an arbitrary link of the chain (in absolute coordinates). First, define:

$$K_{\Lambda_{i-1}} = \phi_i (\phi_i^T \Lambda_{i-1} \phi_i)^{-1} \phi_i^T, \quad (4.121)$$

and

$$L_{\Lambda_{i-1}} = (1 - K_{\Lambda_{i-1}} \Lambda_{i-1}). \quad (4.122)$$

Then we may write the following recursive equation for $\Lambda_i$:

$$\Lambda_i = I_i + \Lambda_{i-1} - \Lambda_{i-1} \phi_i (\phi_i^T \Lambda_{i-1} \phi_i)^{-1} \phi_i^T \Lambda_{i-1}, \quad (4.123)$$

or

$$\Lambda_i = I_i + \Lambda_{i-1} L_{\Lambda_{i-1}}. \quad (4.124)$$

Note the remarkable similarity between Eq. (4.124) and Featherstone's recursive equation for $I_i^*$, the articulated-body inertia of link $i$, shown in Eq. (4.60). These two equations are almost identical in mathematical form, except for a few minor differences. In Eq. (4.124), the operational space inertia matrix takes the place.
of the articulated-body inertia matrix. Also, the direction of the recursion for \( A \) is in the opposite direction, progressing from the base, link 0, to the tip, link \( N \). (The recursion for the articulated-body inertia begins at the tip and progresses back to the base member.) Because of this difference in direction, and the fact that \( A \) and \( I^* \) are, in general, desired at opposite ends of any given link, the indices in Eqs. (4.121) and (4.122) also differ slightly from those in the corresponding articulated-body equations, Eqs. (4.46) and (4.47). Nevertheless, it appears that this recursive algorithm for \( A \) is, for the most part, a reformulation of the articulated-body inertia equations with a few minor changes. In fact, for a chain with an unconstrained base, the two algorithms are basically equivalent. Under this condition, the operational space inertia matrix computed at the constrained tip of the chain will be the articulated-body inertia matrix of the constrained link viewed as the base of an open chain.

The transformation of Eq. (4.124) to local (link) coordinates is straightforward. We may write Eq. (4.123) in link coordinates as follows:

\[
^iA_i = ^iI_i + ^iA_{i-1} - ^iA_{i-1} \left[ (^i\phi_i)^T (^iA_{i-1} ^i\phi_i)^{-1} (^i\phi_i)^T \right] ^iA_{i-1}, \tag{4.125}
\]

where

\[
^iA_{i-1} = ^{i-1}X_i^T (^{i-1}A_{i-1}) ^{i-1}X_i \tag{4.126}
\]

is the operational space inertia matrix at link \( i-1 \) expressed in the \( i \)th coordinate system. If we define:

\[
K'_{A_{i-1}} = \phi_i \left[ \phi_i^T^{i-1}X_i^T (^{i-1}A_{i-1}) ^{i-1}X_i \phi_i \right]^{-1} \phi_i^T, \tag{4.127}
\]

and

\[
L'_{A_{i-1}} = \left[ 1 - (K'_{A_{i-1}})^T (^{i-1}A_{i-1}) (^{i-1}A_{i-1}) ^{i-1}X_i \right], \tag{4.128}
\]

88
then we may rewrite Eq. (4.124) as follows:

\[ ^i\Lambda_i = ^iI_i + i^{-1}X_i^T(i^{-1}\Lambda_{i-1})i^{-1}X_i(L_{A_{i-1}}). \]  

(4.129)

Note the position of the coordinate transformations in this equation compared to the link coordinate version of the articulated-body inertia equation given in Table 12. This difference in position is due to a difference in the order of application of the transformations across joints and links in the two algorithms. As a chain is traversed in the recursion for \( \Lambda \), the operational space inertia matrix of a link is first transformed across the present link (via \( X \)) and then transformed across the next joint (via \( L \)). In contrast, as a chain is traversed in the recursion for \( I^* \), the articulated-body inertia is first transformed across a joint and then transformed back across the preceding link.

Note that when the recursion of Eq. (4.129) reaches link \( N \), the operational space inertia matrix of the entire chain, \( \Lambda \), is known. That is,

\[ \Lambda = ^N\Lambda_N. \]  

(4.130)

If \( \Lambda \) is desired at some point on the last link other than the \( N \)th coordinate origin, then the coordinate transformation of Eq. (4.126) may be invoked for this purpose.

Based on the propagation of the operational space inertia matrix, we will call this approach the Inertia Propagation Method. The complete algorithm for the Inertia Propagation Method for computing the operational space inertia matrix of a serial \( N \)-link manipulator is given in Table 16. The structure of the algorithm is a single forward recursion, starting at the base member and moving out to the last link of the chain. In each iteration, the operational space inertia matrix is propagated across a connecting joint and combined with the spatial inertia of the next link. At link \( N \), the final desired value of \( \Lambda \) is computed. If needed, this matrix may be inverted explicitly to obtain \( \Lambda^{-1} \).
Table 16: Algorithm for the Inertia Propagation Method

Given:

\[ i\dot{\phi}_i, I_i, i^{-1}X_i \] for \( i = 1, \ldots, N; \)

\[ ^0\Lambda_0 = I_0; \]

**Forward Recursion**

Calculate for all \( i = 1, \ldots, N: \)

\[
K'_{\Lambda_{i-1}} = \phi_i \left[ \phi_i^T i^{-1}X_i^T (i^{-1}\Lambda_{i-1}) i^{-1}X_i \phi_i \right]^{-1} \phi_i^T,
\]

\[
L'_{\Lambda_{i-1}} = \left[ 1 - (K'_{\Lambda_{i-1}}) i^{-1}X_i^T (i^{-1}\Lambda_{i-1}) i^{-1}X_i \right],
\]

\[ i\Lambda_i = I_i + i^{-1}X_i^T (i^{-1}\Lambda_{i-1}) i^{-1}X_i (L'_{\Lambda_{i-1}}). \]

Then:

\[ \Lambda = ^N\Lambda_N. \]
Table 17: Computations for $A$ and $A^{-1}$, Inertia Propagation Method

<table>
<thead>
<tr>
<th>Totals</th>
<th>#Mult.</th>
<th>#Add.</th>
<th>#Mult. ($N = 6$)</th>
<th>#Add. ($N = 6$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Lambda$</td>
<td>$155N$</td>
<td>$150N$</td>
<td>930</td>
<td>900</td>
</tr>
<tr>
<td>$A^{-1}$</td>
<td>$155N + 141$</td>
<td>$150N + 90$</td>
<td>1071</td>
<td>990</td>
</tr>
</tbody>
</table>

The scalar operations (multiplications, additions) required to compute $\Lambda$ using the Inertia Propagation Method are shown in Table 17. All operations are given for an $N$ degree-of-freedom manipulator with simple revolute and/or prismatic joints. The computational complexity of this algorithm, like the Force Propagation Method, is $O(N)$. The efficient coordinate transformations described in Chapter 3 are also utilized here. Although this algorithm is almost identical to the recursive articulated-body inertia equations used in the Backward Recursion of the Force Propagation Method, the number of operations shown in Table 17 differs slightly. The additional operations required for this algorithm are basically a result of the difference in the order of application of the link and articulated joint transformations. Note that no computations are tabulated here for $\Omega$ as in the previous three algorithms. It appears that this matrix cannot be computed naturally as a part of the Inertia Propagation Method.

The "big base" concept discussed earlier for a chain with a constrained base was tested for a specific manipulator, the PUMA 560, to determine the computational accuracy of the Inertia Propagation Method under this condition. Arbitrary joint positions were assigned, and all required spatial link and joint parameters were calculated using the industrial specifications for the PUMA [38]. The spatial inertia of the earth, $I_0$, was assigned various values which differed in their relative
size compared to the largest (and most massive) link of the PUMA robot. Using the given information, \( \Lambda \) was calculated in two different ways: by the Explicit Inversion/Multiplication Method and the Inertia Propagation Method. The final results for \( \Lambda \) were then compared to determine the accuracy of the Inertia Propagation Method for a chain with a fixed base. It was determined that values of \( I_0 \) with a magnitude at least \( 10^6 \) times as large as the spatial inertia of the most massive link of the manipulator result in an error in \( \Lambda \) of less than 0.01 percent. These results demonstrate the validity of the “big base” concept and the accuracy of the Inertia Propagation Method under this condition.

4.9 Computational Requirements

The number of scalar operations required by each of the four algorithms presented in this chapter have been calculated and listed separately in the appropriate section. For convenience, we will refer to the Explicit Inversion/Multiplication Method, the Unit Force Method, the Force Propagation Method, and the Inertia Propagation Method as Methods I, II, III, and IV, respectively. The computational requirements for computing the operational space inertia matrix, \( \Lambda \), and its inverse, \( \Lambda^{-1} \), using each of the four algorithms listed above, are compared in Table 18. The computational complexities are given for the case of an \( N \)-link, serial manipulator with simple revolute and prismatic joints. The efficient matrix transformations and other simplifications described in Chapter 3 have been applied in each case. The computations necessary for determining the individual link transformation matrices have also been included.

Methods I and II both require \( O(N^3) \) scalar operations for either \( \Lambda^{-1} \) or \( \Lambda \). Method I, based on the original formulation of Khatib, is always less efficient than Method II. Method II, despite the order of its computational complexity, is the
Table 18: Comparison of Computations for $\Lambda$ and $\Lambda^{-1}$

<table>
<thead>
<tr>
<th>Method</th>
<th>$#\text{Mult.}$</th>
<th>$#\text{Add.}$</th>
<th>$#\text{Mult.}$</th>
<th>$#\text{Add.}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(N = 6)$</td>
<td>$(N = 6)$</td>
<td>$(N = 6)$</td>
<td>$(N = 6)$</td>
</tr>
<tr>
<td>Lilly/Orin I</td>
<td>$\frac{1}{2}N^3 + 19\frac{1}{2}N^2 + 45N + 104$</td>
<td>$\frac{1}{2}N^3 + 13\frac{1}{2}N^2 + 55N + 21$</td>
<td>1184</td>
<td>945</td>
</tr>
<tr>
<td>Lilly/Orin II</td>
<td>$\frac{1}{6}N^3 + 19N^2 + 44\frac{5}{6}N + 104$</td>
<td>$\frac{1}{6}N^3 + 14N^2 + 54\frac{5}{6}N + 21$</td>
<td>1093</td>
<td>890</td>
</tr>
<tr>
<td>Lilly/Orin III</td>
<td>$303N - 279$</td>
<td>$279N - 322$</td>
<td>1539</td>
<td>1352</td>
</tr>
<tr>
<td>Lilly/Orin IV</td>
<td>$155N$</td>
<td>$150N$</td>
<td>930</td>
<td>900</td>
</tr>
</tbody>
</table>

Computation for $\Lambda^{-1}$

<table>
<thead>
<tr>
<th>Method</th>
<th>$#\text{Mult.}$</th>
<th>$#\text{Add.}$</th>
<th>$#\text{Mult.}$</th>
<th>$#\text{Add.}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(N = 6)$</td>
<td>$(N = 6)$</td>
<td>$(N = 6)$</td>
<td>$(N = 6)$</td>
</tr>
<tr>
<td>Lilly/Orin I</td>
<td>$\frac{1}{2}N^3 + 19\frac{1}{2}N^2 + 45N - 37$</td>
<td>$\frac{1}{2}N^3 + 13\frac{1}{2}N^2 + 55N - 69$</td>
<td>1043</td>
<td>855</td>
</tr>
<tr>
<td>Lilly/Orin II</td>
<td>$\frac{1}{6}N^3 + 19N^2 + 44\frac{5}{6}N - 37$</td>
<td>$\frac{1}{6}N^3 + 14N^2 + 54\frac{5}{6}N - 69$</td>
<td>952</td>
<td>800</td>
</tr>
<tr>
<td>Lilly/Orin III</td>
<td>$303N - 420$</td>
<td>$279N - 412$</td>
<td>1398</td>
<td>1262</td>
</tr>
<tr>
<td>Lilly/Orin IV</td>
<td>$155N + 141$</td>
<td>$150N + 90$</td>
<td>1071</td>
<td>990</td>
</tr>
</tbody>
</table>
most efficient approach for $A$ when $N < 6$, and for $A^{-1}$ when $N \leq 7$. Note that both Methods I and II require the computation of the Jacobian and joint space inertia matrices.

Methods III and IV have reduced computational complexities of $O(N)$. This is a significant improvement over the first two algorithms. Method IV is the most efficient approach for $A$ when $N \geq 6$, and for $A^{-1}$ when $N > 7$. Recall, however, that Method IV is based on the explicit knowledge of the spatial link inertia of the base member, $I_0$. This assumption may not be the most appropriate in all cases. If the base is fixed to the inertial frame, then the $O(N)$ solution of Method III may be used. This approach is more efficient than Method II for $A$ and $A^{-1}$ when $N > 12$.

A few observations concerning the computation of $\Omega$ will also be made here. Recall that $\Omega$ is an immediate result of both Methods I and II in the calculation of $A^{-1}$ and/or $A$. The required computations are shown in Tables 9 and 11, respectively. Recall also that $\Omega$ may be computed using some of the partial results of Method III, as well as some additional computations. In this case, the number of operations required to compute $\Omega$ alone are $(501N - 742)$ scalar multiplications and $(413N - 636)$ scalar additions. This $O(N)$ approach is the most efficient for $\Omega$ alone for $N \geq 21$, while the equations of Method II lead to the fewest operations for $N < 21$. If both $\Omega$ and $A$ (or $A^{-1}$) are desired, Method II is again the most efficient approach for $N < 21$, while Method III is the best for $N \geq 21$.

4.10 Conclusions and Summary

In this chapter, four efficient methods for computing the operational space inertia matrix have been presented. The first method is based on equations developed in the operational space formulation of Khatib [21], while the second uses the
concept of spatial unit forces to derive the relationship between the spatial contact force vector and the end effector acceleration vector. The third and fourth methods use linear recursive algorithms to solve for $\Lambda^{-1}$ and $\Lambda$, respectively. In all four methods, a general joint model is used which allows multiple-degree-of-freedom joints and contacts if desired. This generalization allows these methods to be applied to more complex robot configurations than the standard revolute/prismatic combinations. For the sake of simplicity, however, only these typical configurations have been considered when evaluating computational efficiency.

The $O(N)$ algorithms of Methods III and IV, with their reduced computational order, represent a significant contribution in the development of efficient algorithms for the calculation of the operational space inertia matrix. Method IV is the most efficient algorithm known for $\Lambda$ for $N \geq 6$. Although Methods I and II are $O(N^3)$, Method II provides the best solution for $\Lambda$ for $N < 6$. Method II is also the most efficient algorithm for computing $\Omega$ and $\Lambda$ together for small $N$. The efficient computation of $\Lambda$ and/or $\Lambda^{-1}$ is an important requirement in operational space control. In the next two chapters, we will also demonstrate its desirability in the simulation of closed-chain systems.
CHAPTER V

EFFICIENT DYNAMIC SIMULATION OF A SINGLE CLOSED CHAIN

5.1 Introduction

Single chain robot manipulators may be found in many different applications today. In industry, single robot manipulators are used for welding, painting and some assembly operations. Their use in forging and machining continues. Robot manipulators are also part of new plans for space exploration and space station operation. In many of these applications, the manipulator tip comes in contact with various objects in its environment, instantaneously creating a closed-chain configuration.

Although a single closed chain is a simple example of a closed-chain robotic mechanism, its real-time dynamic simulation is not trivial. The dynamics of the chain must be combined with the kinematic constraints which are imposed by the tip contact. In general, both the contact forces at the tip and the joint accelerations must be computed to completely solve the system.

In this chapter, a new, efficient serial algorithm for the dynamic simulation of a single closed chain is developed. The algorithm is valid for a manipulator with any number of degrees of freedom, and it is still applicable when the manipulator is in a singular position. Arbitrary joints are allowed, including multiple-degree-of-freedom joints. The end effector of the chain is assumed to be in contact with
some rigid object in its environment which has known motion, so that a single closed kinematic loop is formed. The operational space inertia matrix of the chain is used to solve for the unknown contact force at the tip, which is then used in the solution for the joint accelerations. Two computational versions of the algorithm are formulated for a standard manipulator with simple revolute and/or prismatic joints. The computational complexity of the first version is $O(N^3)$, where $N$ is the number of degrees of freedom in the chain. The second version has a reduced order computational complexity of $O(N)$.

A review of previous work related to the dynamic simulation of single closed chains is given in the second section of this chapter. The next three sections discuss several steps in the development of the new algorithm. In particular, in the third section, the equations of motion for a single chain are used to partition the joint acceleration vector into two terms, one known and one unknown. The unknown term is a function of the contact forces and moments at the tip. The end effector acceleration vector is partitioned in a similar way in the fourth section, making use of the operational space inertia matrix. In the fifth section, two classes of contacts are defined which may be used to model interactions between the end effector and other rigid bodies. Specific examples are provided.

In the sixth section, the complete dynamic simulation algorithm for a single closed chain is presented as a series of steps. Each step is explained in detail, particularly the step which computes the unknown contact forces and moments. The integration of the joint rates and accelerations to obtain the next state positions and rates is also briefly discussed. The computational requirements of both versions of this algorithm are tabulated and compared in the seventh section of this chapter. A summary of the results and some overall conclusions are given in the final section.
5.2 Previous Work

The new dynamic simulation algorithm for a single closed chain presented in this chapter makes use of Direct Dynamics algorithms for single open chains. Therefore, this section begins with a review of some important open-chain Direct Dynamics algorithms, followed by a summary of similar research efforts related to single closed chains.

5.2.1 Direct Dynamics for Single Open Chains

One of the most efficient open-chain Direct Dynamics algorithms (based on inertia matrix inversion) is presented by Walker and Orin in [9]. The authors discuss four methods for obtaining the joint accelerations of an open kinematic chain. Of the four methods, the third (later named the Composite-Rigid-Body Method [23]) is the most efficient. It is based on the computation of composite-rigid-body inertias. Although its computational complexity is $O(N^3)$, the coefficient of $N^3$ is small, making it quite efficient. However, the application of this algorithm is limited to configurations of revolute and prismatic joints.

A similar $O(N^3)$ method, presented by Angeles and Ma in [11], uses the concept of an "orthogonal complement" to construct the joint space inertia matrix. The Cholesky decomposition of this matrix is used in solving the appropriate linear system for the joint accelerations. The computational complexity of this algorithm is slightly better than that in [9], but the algorithm is still not the most efficient. It, too, is restricted to configurations of simple revolute and prismatic joints.

Perhaps the most familiar linear recursive approach for open-chain Direct Dynamics is the $O(N)$ Articulated-Body Method of Featherstone [19]. The spatial notation introduced by Featherstone is both useful and powerful. The concept of articulated-body inertias is also significant, and it is the central feature of the
method. The algorithm computes the articulated-body inertia of each set of links of the mechanism in a recursive manner, starting at the tip and working back to the base. A second recursion, beginning at the base and working out to the tip, calculates the joint acceleration at each joint. Single degree-of-freedom joints are considered. Although the computational complexity is \( O(N) \), this algorithm is less efficient than the Composite-Rigid-Body Method for \( N < 12 \).

Brandl, Johanni, and Otter [10] have developed an \( O(N) \) approach which is very similar to that of Featherstone. The method differs only in its use of a more general joint model (allowing multiple-degree-of-freedom joints) and in the optimization of vector and matrix transformations. Of all the algorithms considered, this one appears to be the most efficient for computing the joint accelerations for an open-chain mechanism. The optimized mathematical transformations are very important, and they may be used in any general application where vector and matrix quantities must be transformed between coordinate frames.

5.2.2 Direct Dynamics for Single Closed Chains

One of the first Direct Dynamics algorithms for single closed-chain robotic mechanisms is presented by Orin and McGhee in [12]. This algorithm is based on the inertia matrix inversion approach. The dynamic equations of motion for the chain are augmented with kinematic constraint equations at the tip of the chain. For a chain with \( N \) degrees of freedom and \( n_c \) acceleration constraints at the tip, the simultaneous solution for the joint accelerations and the unknown tip contact forces and moments requires the inversion of an \((N + n_c) \times (N + n_c)\) matrix. Although this approach is straightforward, it is not very efficient.

In [18], Featherstone introduces a similar approach which combines the \( N \) dynamic equations for the chain with \( n_c \) simultaneous quadratic equations in the
unknown contact forces. The main difficulty with this approach is the solution of the quadratic equations, which have, in general, $2^{nc}$ roots. Even after the unknown contact forces have been found, an $N \times N$ inverse is still required to solve for the joint accelerations. Although this approach is quite general, it is computationally intensive, and the quadratic equations are not easily dealt with.

McClamroch [39] and Huang [40] both discuss the modelling of constrained robot systems using singular systems of differential equations. Given a singular system with a particular form, a reduction approach is developed which ultimately decouples the system into simplified differential and algebraic equations. A single manipulator carrying an inertial load and a single chain with its tip in contact with a rigid surface are two examples discussed in both works. The approach of [39] and [40] is mathematically general, and therefore, computational complexity is not considered.

In [17], Kankaanranta and Koivo derive a dynamic model of a single constrained manipulator which is useful both for simulation and control. This approach is similar to that developed by McClamroch and Huang above. Using this model, the $n_c$ unknown contact forces at the tip are found from a set of $n_c$ linear simultaneous algebraic equations. Knowing the contact forces, the joint accelerations are found from the $N$ dynamic equations of motion. Although this approach is quite efficient, it is developed only for the specific case of a six degree-of-freedom manipulator in a nonsingular position. The more complex issues involved when considering manipulators in singular positions, or with reduced degrees of freedom, are not addressed.

Linear recursive approaches for Direct Dynamics for single closed chains have been presented by Lathrop in [14] and are included in the work of Brandl, Johanni, and Otter [16]. In [14], general endpoint constraints are propagated from the tip
down to the base, where two $6 \times 6$ inverses are necessary to match the propagated
tip constraints to the existing base constraints. A second recursion from base to
tip solves for the joint accelerations. In [16], the open-chain Direct Dynamics algo­
rithm of [10] is extended to include closed kinematic loops. Once again, a general
joint model is used, and for a single closed chain, this algorithm requires no inver­
sion greater than $6 \times 6$. Although a great variety of configurations and constraints
may be modelled in these two algorithms, specific and detailed examples need to
be investigated to determine their actual practicality.

5.3 Equations of Motion and Joint Accelerations

In this section, we begin the development of a new serial algorithm for the
efficient dynamic simulation of a single closed-chain robotic system. Figure 10
illustrates one example of such a system: a serial-link robot chain with a fixed
base and the end effector in contact with the inertial frame. In general, we will
assume that the motion (velocity and acceleration) of the base member and the
contacted body are known (not necessarily zero). The total number of degrees of
degrees of freedom for the chain is arbitrary; that is, it may be less than, equal to, or greater
than six. The algorithm is applicable even when the chain is in a singular position.
The joints of the system are also arbitrary, and multiple-degree-of-freedom joints
are allowed. As might be expected, we begin with the dynamic equations of motion
for the single chain.

The dynamic equations of motion for a single N degree-of-freedom chain may
be written (as given in Chapter 3):

$$\tau = H(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) + J^T(q) f, \quad (5.1)$$

where f is the $6 \times 1$ vector of contact forces and moments exerted by the last link.
Note that if f is known, the mechanism may be treated as an open chain with an
additional known input torque or force at each joint actuator which is a result of
the contact force vector. The Jacobian matrix transforms f to joint space.

The equations of motion for the chain may be used to partition the joint
acceleration vector into the difference of two terms. For a given position, the first
term corresponds to the chain in an open, unconstrained configuration with the
contact force removed, while the second term is a function of the contact force
vector. This fact allows us to find a partial solution for the closed-chain joint
accelerations before the contact force is explicitly determined.

Using Eq. (5.1), we may solve for the vector of joint accelerations for the
closed chain, $\ddot{q}$, as follows:

$$\ddot{q} = H^{-1}(\tau - C \dot{q} - G - J^T f), \quad (5.2)$$

$$= H^{-1}(\tau - C \dot{q} - G) - (H^{-1} J^T) f, \quad (5.3)$$

$$= \ddot{q}_{open} - \ddot{q}_{constrained}, \quad (5.4)$$
where $\ddot{q}_{\text{open}}$ is the vector of joint accelerations for the open, unconstrained chain, and $\ddot{q}_{\text{constrained}}$ is the vector of joint accelerations resulting from the contact force vector, $f$. Noting from Chapter 4 that

$$(H^{-1}J^T) = \Omega,$$  \hspace{1cm} (5.5)

we may rewrite the above equation:

$$\ddot{q} = \ddot{q}_{\text{open}} - \Omega f.$$  \hspace{1cm} (5.6)

Given the present state of the manipulator, $q$ and $\dot{q}$, the input joint actuator torques and forces, $\tau$, and the motion of the base, the open-chain term, $\ddot{q}_{\text{open}}$, is completely defined. An appropriate Direct Dynamics algorithm for open-chain manipulators may be used to determine its value. With the same given information, the coefficient of $f$ in the constrained term, $\Omega$, is also known. The efficient computation of $\Omega$ was discussed in Chapter 4.

### 5.4 Acceleration of the End Effector

A similar difference expression may be developed for the end effector acceleration vector, $\ddot{x}$, as follows. The end effector velocity, $\dot{x}$, is defined:

$$\dot{x} = J \dot{q}.$$  \hspace{1cm} (5.7)

Then, also by definition, its time derivative, $\ddot{x}$, is:

$$\ddot{x} = J \ddot{q} + \dot{J} \dot{q}.$$  \hspace{1cm} (5.8)

Using Eq. (5.2), we may write:

$$\ddot{x} = J H^{-1}(\tau - C \dot{q} - G) + \dot{J} \dot{q},$$  \hspace{1cm} (5.9)

$$= J H^{-1}(\tau - C \dot{q} - G) + \dot{J} \dot{q} - (J H^{-1}J^T) f,$$  \hspace{1cm} (5.10)

$$= [J H^{-1}(\tau - G) + (\dot{J} - J H^{-1}C) \dot{q}] - (J H^{-1}J^T) f,$$  \hspace{1cm} (5.11)

$$= \ddot{x}_{\text{open}} - \ddot{x}_{\text{constrained}}.$$  \hspace{1cm} (5.12)
where $\ddot{x}_{\text{open}}$ is the tip acceleration vector for the open, unconstrained chain, and $\ddot{x}_{\text{constrained}}$ is the tip acceleration vector resulting from the contact force vector, $f$. Noting that

$$ (JH^{-1}J^T) = \Lambda^{-1}, $$

(5.13)

we may rewrite the above equation as:

$$ \ddot{x} = \ddot{x}_{\text{open}} - \Lambda^{-1}f, $$

(5.14)

where $\Lambda^{-1}$ is the inverse operational space inertia matrix for the single chain.

As was true for the joint accelerations, if the present state, driving actuator torques and/or forces, and motion of the base are known, the open-chain term, $\ddot{x}_{\text{open}}$, is known. Its value may also be determined by using an appropriate Direct Dynamics algorithm for open-chain manipulators. Because the joint positions are assumed known, the inverse operational space inertia matrix, $\Lambda^{-1}$, is defined. The efficient computation of $\Lambda$ and its inverse was the primary topic of Chapter 4.

### 5.5 Tip Constraints

In this section, the interactions between the tip of the chain and other rigid bodies are discussed. We begin with a general model of a contact, using the mathematical tools and notation of Section 2.3. We may use this model to specify the structure of the end effector acceleration and contact force vectors. Two classes of contacts between the tip and other rigid bodies are defined. In the first class of contacts, the unknown components of the contact force vector are independent of the known components. This class includes rigid connections and the cases of hard point contact and soft finger contact with no slipping. The second class of contacts describes those interactions in which the unknown components of the contact force vector are linear functions of the known components. This class includes the case.
of a sliding contact on a surface with a characteristic coefficient of friction. Finally, examples of these two classes of contacts are described in detail.

5.5.1 Contact Model

At the tip contact, the motion and constraint vector spaces may be defined using the general joint model discussed in Section 2.3. For convenience, we will assume that the two dual bases used to partition the spatial acceleration and force vectors at the tip,

\[
\left[ \begin{array}{c}
\phi_{N+1} \\
\phi^c_{N+1}
\end{array} \right] \quad \text{and} \quad \left[ \begin{array}{c}
\psi_{N+1} \\
\psi^c_{N+1}
\end{array} \right],
\]

(5.15)

are the same. Thus, we may define:

\[
\phi_{N+1} = \text{vector space of free (unconstrained) directions}
\]

for the contact, \([6 \times (6 - n_c)]\),

\[
\phi^c_{N+1} = \text{vector space of constrained directions for}
\]

the contact, \([6 \times n_c]\),

where

\[
n_c = \text{number of degrees of constraint at the contact point.}
\]

This contact occurs between body \(N\) (the last link or end effector) and body \(N+1\) (the rigid body in the environment). The subscript \(N+1\) is used for this \((N+1)\)st "joint" of the system. To simplify notation, we will drop this subscript for the remainder of this chapter.

The two vector spaces defined for the contact are orthogonal, so the following relationships are true:

\[
\phi^T(\phi^c) = 0,
\]

(5.16)
If we assume that the columns of \( \phi \) and \( \phi^c \) are orthonormal, then we may also write:

\[
\phi^T \phi = 1,
\]

and

\[
(\phi^c)^T (\phi^c) = 1,
\]

where \( 1 \) is the \([ (6 - n_c) \times (6 - n_c) ] \) or \([ n_c \times n_c ] \) identity matrix, respectively.

We may resolve both the end effector acceleration and general contact force vectors in the two orthogonal vector spaces of the contact. The end effector acceleration vector may be written as follows:

\[
\ddot{x} = \phi \dot{g} + \phi^c \dot{g}^c,
\]

where \( g \) and \( g^c \) are \((6 - n_c) \times 1 \) and \( n_c \times 1 \) vectors, respectively. In a similar manner, the vector of contact forces may be written:

\[
f = \phi \dot{h} + \phi^c \dot{h}^c,
\]

where \( h \) and \( h^c \) are \((6 - n_c) \times 1 \) and \( n_c \times 1 \) vectors, respectively. If we refer again to Section 2.3, we see that the vector \( f \) is literally \( f_{N+1} \), the general force vector exerted on body \( N + 1 \) by body \( N \). The component vectors, \( h \) and \( h^c \), correspond to \( \tau^*_N \) and \( \tau^c_{N+1} \), respectively. (Recall here that \( \psi = \phi \) and \( \psi^c = \phi^c \).)

5.5.2 Classes of Contacts

Two classes of contacts are considered in this analysis. The characteristics of the two classes are described with reference to Eqs. (5.20) and (5.21) above. The
classes are defined as follows:

**Class I Contact**

(a) \(g^c\) and \(h\) are completely specified,

(b) \(g\) and \(h^c\) are unknown.

**Class II Contact**

(a) \(g^c\) is completely specified,

(b) \(g\) and \(h^c\) are unknown,

(c) \(h = Bh^c + d\), where \(B\) and \(d\) are known.

For every unknown component of the contact force vector, there is a corresponding known value of relative linear or angular acceleration of the end effector in (or along) the same direction which is a result of the constraint [12]. Because the acceleration of the contacted body is known, and the components of the relative acceleration are known in the constrained directions, the components of the absolute end effector acceleration in the constrained directions are also known. That is, for every component of the unknown \(h^c\), there is a known component of \(g^c\). Also, as with any "joint", the components of force in the free directions, \(h\), are known (or their relationship to the constrained components is known), and the relative accelerations in the free directions, \(g\), are unknown. Examples of the two classes of contacts are given next.

**5.5.3 Examples**

For ease of explanation, the following examples refer to contacts between the end effector and an inertial frame, the motion of which is zero. These examples may be easily extended to contacts between the end effector and any other body
for which the motion is completely specified, including the base of the chain itself. More comments on this point are made throughout the following discussion.

The purpose of these examples is to illustrate the modelling principles presented above. For each example, the free and constrained vector spaces are defined, and the known and unknown components of the end effector acceleration and general contact force vectors are specified. From this information, the class of the contact may be identified.

**Example 1: Rigid Connection**

Figure 11 illustrates a rigid connection between the tip of the manipulator and the fixed inertial frame. No relative motion is possible between the tip and this frame. The coordinate frame at the tip, relative to which the acceleration and contact force vectors are expressed, may be assigned such that the unit vector \( \hat{z} \) is normal to the contact surface at the point of contact and is pointing away from the surface. For this example, the number of degrees of constraint at the tip is:

\[
    n_c = 6. \tag{5.22}
\]

108
The free and constrained vector spaces of the contact may be defined as follows:

\[ \phi = 0, \quad (5.23) \]

and

\[ \phi^c = I_{6 \times 6}, \quad (5.24) \]

where \( I_{6 \times 6} \) is the \( 6 \times 6 \) identity matrix. Since the end effector acceleration is the same as that of the inertial frame (which is zero), we may write:

\[ \ddot{x} = g^c = 0. \quad (5.25) \]

Because the tip is completely constrained, all six components of the contact force vector are unknown, or:

\[ f = h^c. \quad (5.26) \]

The vectors in the free directions, \( g \) and \( h \), are null vectors. The distribution of known and unknown components for this example makes it a Class I Contact.

Note that if this rigid connection exists between the chain tip and another body with a spatial acceleration of \( a_b \), then the known components of \( \ddot{x} \) are defined as follows:

\[ g^c = a_b, \quad (5.27) \]

while the constrained contact forces, \( h^c \), are still unknown.

**Example 2: Hard Point Contact with Friction**

For this example, we will initially assume that the tip of the manipulator is already in motion relative to the contact surface (slipping). The coefficient of friction is finite. In this case, we may assume that the contact forces applied in the directions of motion are already large enough to overcome static friction. We will also examine the same contact when the coefficient of friction is negligible (frictionless surface), and when the manipulator tip is not slipping on the surface.
Figure 12: Hard Point Contact with Friction

Figure 12 illustrates a hard point contact between the tip and the inertial frame with a surface coefficient of friction, $\mu$. Motion is assumed to occur only in the plane normal to the $\hat{z}$ unit vector, that is, in the plane spanned by the $\hat{x}$ and $\hat{y}$ unit vectors.

Because it is a hard point contact, the tip cannot exert any moments on the inertial frame. Thus, the first three components of the spatial contact force vector, $n_x, n_y,$ and $n_z$, are zero. By definition, the linear forces which the tip exerts in the plane of motion, $f_x$ and $f_y$, are proportional to the normal force, $f_z$, as follows:

\begin{align*}
    f_x &= (\mu k_x)f_z, \\
    f_y &= (\mu k_y)f_z,
\end{align*}

where $k_x$ and $k_y$ are functions of the angle between the direction of motion in the plane and the corresponding coordinate axis. Because the tip is constrained by the contact surface in the normal direction, the normal force, $f_z$, is unknown. Combining all the above information, we may write the spatial contact force vector.
as follows:

\[
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
(\mu k_x) \\
(\mu k_y) \\
1
\end{bmatrix}
\]

The \( \ddot{z} \) component of the linear end effector acceleration, \( a_z \), is known and is equal to zero. All other components of \( \ddot{x} \) are unknown. Thus, we may express \( \ddot{x} \) as follows:

\[
\ddot{x} = \begin{bmatrix}
a_x \\
a_y \\
a_z \\
a_x \\
a_y \\
0
\end{bmatrix}
\]

The number of degrees of constraint for this contact is:

\[
n_c = 1.
\]

When \( \mu \) is finite and nonzero, the free and constrained vector spaces of the contact may be defined as follows:

\[
\phi = \begin{bmatrix}
1_{5 \times 5} \\
0_{1 \times 5}
\end{bmatrix},
\]

and

\[
\phi^c = \begin{bmatrix}
0_{5 \times 1} \\
1_{1 \times 1}
\end{bmatrix}.
\]
The known and unknown components of the end effector acceleration are:

\[ g^c = a_z = 0, \]  

and

\[ g = \begin{bmatrix} \alpha_x & \alpha_y & a_x & a_y \end{bmatrix}^T. \]  

The unknown component of the spatial contact force vector is the normal force, \( f_z \). That is,

\[ h^c = f_z. \]  

The known components may be written as a function of this unknown as follows:

\[
h = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\mu k_x) f_x \\ (\mu k_y) f_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\mu k_x) \\ (\mu k_y) \end{bmatrix} h^c = Bh^c + d, \]  

where \( d \) is a \( 5 \times 1 \) zero vector.

The characteristics of this interaction make it a Class II Contact. Once again, note that if this contact exists between the chain tip and another body whose spatial acceleration is known, say \( a_t \), then all the above assignments would remain the same except for the known vector, \( g^c \). Under this condition, \( g^c \) would be equal to the \( z \) component of the linear part of \( a_t \).

If the coefficient of friction, \( \mu \), is negligible (frictionless surface), then this contact becomes a Class I Contact. That is, if

\[ \mu \to 0, \]  

then

\[ f_x, f_y \to 0, \]  

112
so that

\[ g^c = 0, \quad (5.41) \]
\[ g = \begin{bmatrix} \alpha_x & \alpha_y & a_x & a_y \end{bmatrix}^T, \quad (5.42) \]
\[ h^c = f_z, \quad (5.43) \]

and

\[ h = \begin{bmatrix} n_x \\ n_y \\ n_z \\ f_x \\ f_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (5.44) \]

If \( \mu \) is not negligible, and the manipulator tip is not slipping, then a Class I Contact is also formed. We have:

\[ a_x, a_y \to 0. \quad (5.45) \]

In this case, the free and constrained vector spaces become:

\[ \phi = \begin{bmatrix} 1_{3 \times 3} \\ 0_{3 \times 3} \end{bmatrix}, \quad (5.46) \]

and

\[ \phi^c = \begin{bmatrix} 0_{3 \times 3} \\ 1_{3 \times 3} \end{bmatrix}. \quad (5.47) \]

The known and unknown components of the end effector acceleration and contact force vectors are now:

\[ g^c = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad (5.48) \]
\[ g = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix}, \quad (5.49) \]
\[ h^c = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \quad (5.50) \]

and
\[ h = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \quad (5.51) \]

As before, these examples are easily adapted to represent contacts between the tip and another body with known nonzero acceleration. Note that in all of the contact examples illustrated above, the dual bases defined in Eq. (5.15) and used for partitioning the spatial tip acceleration and force vectors are, in fact, the same.

5.6 Dynamic Simulation Algorithm for a Single Closed Chain

This section presents an efficient serial algorithm for the dynamic simulation of a single closed chain based on the concepts developed in the previous three sections. For this simulation problem, the following information is given for the chain or contacted body, as indicated:

**Given:**

\[ q, \dot{q} = \text{present state joint positions and rates}, \]
\[ \tau = \text{vector of driving actuator torques and/or forces}, \]
\[ v_0, a_0 = \text{velocity and acceleration of the base}, \]
\[ v_b, a_b = \text{velocity and acceleration of the contacted body}, \]
(g^c, h) or (g^c, B, d) = known characteristics of the contact.

In the solution of this closed-chain simulation problem, the following unknown quantities are desired:

Unknown:

\[ h^c = \text{unknown contact forces and moments}, \]
\[ \ddot{q} = \text{closed-chain joint accelerations}, \]
\[ q^+, \dot{q}^+ = \text{next state joint positions and rates}. \]

The reaction torques/forces at the powered joints and the relative accelerations of the end effector in the free directions of the contact are also unknown, but their solution is not usually required. They may also be computed as desired.

The new simulation algorithm developed here may be presented as a series of four computational steps. These steps will be referred to as follows:

1. The Open Chain Solution,
2. Calculation of the Unknown Contact Forces,
3. Calculation of the Closed-Chain Joint Accelerations,
4. Integration for the Next State.

In the first step, the Direct Dynamics problem is solved for an open-chain configuration of the system. That is, the joint acceleration and end effector acceleration vectors are computed assuming that the contact force vector is zero. The coefficients of \( f \) in Eqs. (5.6) and (5.14), \( \Omega \) and \( \Lambda^{-1} \), are also computed. These matrices are both functions of the joint positions only.

In the second step, the dynamic equations of the end effector are combined with the contact model to determine the unknown components of the contact force.
vector. The computations required for this second step differ slightly for the two classes of contacts discussed in Section 5.5, but the basic conceptual approach is the same. Once the contact force vector is completely defined, a full solution for the closed-chain joint accelerations may be found from the corresponding equations of motion. This is the third step. In the fourth and final step, the joint accelerations and rates are integrated to find the next state joint rates and positions. The next four subsections explain each of these four steps in some detail.

5.6.1 Step 1. The Open Chain Solution

In this step, the solution to the Direct Dynamics problem for an open, unconstrained configuration of the manipulator is found. This corresponds to solving for the two open-chain terms, $\ddot{q}_{open}$ and $\dot{x}_{open}$, in Eqs. (5.6) and (5.14), respectively. As was mentioned in the previous sections, because these two vectors are functions only of the present state joint positions, rates, and the actuator torques and/or forces, they may be found from the given information. The coefficients of the contact force vector in these same two equations, $\Omega$ and $\Lambda^{-1}$, may also be calculated in this step. They are also functions of the present state joint positions.

Let us first consider the calculation of the open-chain acceleration terms, $\ddot{q}_{open}$ and $\dot{x}_{open}$. These two vectors may be found using an appropriate open-chain Direct Dynamics algorithm. The most efficient algorithm known for computing both of these vectors is that of Brandl, Johanni, and Otter [10]. As was mentioned previously, this algorithm is based on a general joint model which allows multiple-degree-of-freedom joints. The computational complexity of this algorithm is $O(N)$ for an $N$ degree-of-freedom manipulator with simple revolute and/or prismatic joints. The efficient computation of $\Omega$ and $\Lambda^{-1}$ was discussed in detail in Chapter 4. The most efficient method known for the computation of both $\Omega$ and $\Lambda^{-1}$ for
$N < 21$ is the Unit Force Method (Method II), which is $O(N^3)$ for an $N$ degree-of-freedom manipulator with revolute and/or prismatic joints. For $N \geq 21$, the $O(N)$ Force Propagation Method (Method III) is the most efficient. The use of these two methods will be discussed further in Section 5.7.

5.6.2 Step 2. Calculation of the Unknown Contact Forces

In this second step, the unknown components of the contact force vector, $h^c$, are computed. The dynamic equations of motion expressed in end effector (or operational) space are combined with the contact model at the tip to accomplish this task. As was previously noted, the equations derived for this step differ slightly for the two classes of contacts which we have defined, but the fundamental method for finding $h^c$ is the same. First we consider a manipulator with a Class I Contact between the tip and another rigid body.

**Class I Contact**: $(g^c, h)$ known, $(g, h^c)$ unknown

From Eqs. (5.14) and (5.20):

\[
\ddot{x} = \ddot{x}_{open} - \Lambda^{-1}f
\]

\[
= \phi g + \phi^c g^c.
\]

Combining this equation with Eq. (5.21) gives:

\[
\phi g + \phi^c g^c = \ddot{x}_{open} - \Lambda^{-1}(\phi h + \phi^c h^c).
\]

Since $(\phi^c)^T(\phi^c) = 1$, and $(\phi^c)^T(\phi) = 0$, we may premultiply Eq. (5.54) by $(\phi^c)^T$ to eliminate the unknown quantity, $g$. Thus, we have,

\[
(\phi^c)^T(\phi g + \phi^c g^c) = (\phi^c)^T \left[ \ddot{x}_{open} - \Lambda^{-1}(\phi h + \phi^c h^c) \right],
\]

or

\[
g^c = (\phi^c)^T \ddot{x}_{open} - (\phi^c)^T \Lambda^{-1}(\phi h + \phi^c h^c).
\]
Note that if it is necessary to model $f$ using the dual basis composed of $\psi$ and $\psi^c$, the same elimination may be carried out using $(\psi^c)^T$ instead. The only unknown quantity in Eq. (5.56) is $h^c$, the unknown components of the contact force vector. A solution for $h^c$ may be found by solving the following linear system of algebraic equations:

$$
([\phi^c]^T \Lambda^{-1} \phi^c] h^c = \left[([\phi^c]^T \ddot{x}_{\text{open}} - (\phi^c)^T \Lambda^{-1} \phi h - g^c].
$$

(5.57)

An analytical solution for $h^c$ may also be written:

$$
h^c = \left([\phi^c]^T \Lambda^{-1} \phi^c\right)^{-1} \left([\phi^c]^T \ddot{x}_{\text{open}} - (\phi^c)^T \Lambda^{-1} \phi h - g^c].
$$

(5.58)

The dimension of the linear system in Eq. (5.57) is $n_c$. Thus, only an $n_c \times n_c$ inverse is needed to solve for the unknown components of the contact force vector. Once $h^c$ is known,

$$
f = \phi h + \phi^c h^c
$$

(5.59)

is completely specified.

Now we consider the case of a Class II Contact between the tip and another rigid body. The solution is slightly more complex due to the nature of the contact model, but the basic approach is the same.

**Class II Contact:** $(g^c, B, d)$ known, $(g, h^c)$ unknown

As in the previous case,

$$
\phi g + \phi^c g^c = \ddot{x}_{\text{open}} - \Lambda^{-1} f.
$$

(5.60)

However, the contact force vector must now be expressed as follows:

$$
f = \phi h + \phi^c h^c,
$$

(5.61)

$$
= \phi (Bh^c + d) + \phi^c h^c,
$$

(5.62)
or

\[ f = (\phi B + \phi^c)h^c + \phi d. \] \quad (5.63)

Combining Eqs. (5.60) and (5.63), we obtain:

\[ \phi g + \phi^c g^c = \bar{x}_{open} - \Lambda^{-1}[(\phi B + \phi^c)h^c + \phi d]. \] \quad (5.64)

Again, we may use \((\phi^c)^T\) to eliminate the unknown vector, \(g\), as follows:

\[ (\phi^c)^T(\phi g + \phi^c g^c) = (\phi^c)^T\bar{x}_{open} - (\phi^c)^T\Lambda^{-1}[(\phi B + \phi^c)h^c + \phi d], \] \quad (5.65)

or

\[ g^c = (\phi^c)^T\bar{x}_{open} - (\phi^c)^T\Lambda^{-1}(\phi B + \phi^c)h^c - (\phi^c)^T\Lambda^{-1}\phi d. \] \quad (5.66)

Again, \(h^c\) is the only unknown quantity in this equation. Its solution may be found from the following linear system of algebraic equations:

\[ [(\phi^c)^T\Lambda^{-1}\phi B + (\phi^c)^T\Lambda^{-1}(\phi^c)]h^c = [(\phi^c)^T\bar{x}_{open} - (\phi^c)^T\Lambda^{-1}\phi d - g^c]. \] \quad (5.67)

An analytical expression for \(h^c\) may be written:

\[ h^c = [(\phi^c)^T\Lambda^{-1}\phi B + (\phi^c)^T\Lambda^{-1}(\phi^c)]^{-1}[(\phi^c)^T\bar{x}_{open} - (\phi^c)^T\Lambda^{-1}\phi d - g^c]. \] \quad (5.68)

The dimension of the linear system in Eq. (5.67) is \(n_c\). Thus, even though the contact model is more complex, an \(n_c \times n_c\) inverse is still all that is needed to solve for the unknown components of the contact force vector. Once \(h^c\) is known,

\[ f = \phi h + \phi^c h^c \] \quad (5.69)

is completely specified.

In both cases above, a solution for \(h^c\) is found by solving a linear system of algebraic equations of the form:

\[ P h^c = s, \] \quad (5.70)
where \( P \) is an \( n_c \times n_c \) square matrix, and \( h^c \) and \( s \) are \( n_c \times 1 \) column vectors. As long as \( s \) lies in the range of \( P \), denoted \( \mathcal{R}(P) \), there will exist at least one solution for \( h^c \). Physically interpreted, as long as the trajectory of the closed-chain system satisfies the kinematic constraints at the tip, the acceleration term (represented by \( s \)) will lie in the constrained motion space of the end effector (represented by \( P \)), and the constrained contact forces, \( h^c \), will be defined.

To determine the number and type of solutions which exist for \( h^c \), we must examine the rank of the matrix \( P \). The rank of \( P \), call it \( r \), must be equal to or less than \( n_c \). If \( r \) is equal to \( n_c \), then there exists a single unique solution for \( h^c \). If \( r \) is less than \( n_c \), then multiple solutions exist for \( h^c \). In fact, if \( r < n_c \), then \( h^c \) has infinitely many solutions. To choose the best of these many values, we must examine this situation in more detail.

Assume that \( r < n_c \). Via elementary row operations, Eq. (5.70) may be transformed into the following form:

\[
P' h^c = s',
\]

(5.71)

where

\[
P' = \begin{bmatrix}
U \\
0
\end{bmatrix},
\]

\[
s' = \begin{bmatrix}
w \\
0
\end{bmatrix},
\]

with

\[
U = r \times n_c \text{ matrix in echelon form,}
\]

\[
w = r \times 1 \text{ vector,}
\]

120
and the zero matrices in $P'$ and $s'$ are $(n_c - r) \times n_c$ and $(n_c - r) \times 1$, respectively. Equation (5.71) is equivalent to the reduced system:

$$U h^c = w,$$  \hspace{1cm} (5.72)

which represents $r$ equations in $n_c$ unknowns.

The matrix $U$ has rank $r$, so there will be $r$ basic variables and $(n_c - r)$ free variables in the solution for $h^c$. In fact, we may further reduce the system in Eq. (5.72) into row-reduced echelon form as follows:

$$\begin{bmatrix} 1_r & U_r \end{bmatrix} \begin{bmatrix} h^c_b \\ h^c_f \end{bmatrix} = [w_r],$$  \hspace{1cm} (5.73)

where

$1_r = r \times r$ identity matrix,

$U_r = r \times (n_c - r)$ matrix in echelon form,

$h^c_b = r \times 1$ vector of basic variables,

$h^c_f = (n_c - r) \times 1$ vector of free variables,

$w_r = r \times 1$ transformed $w$ vector.

Note that permutations of the columns of $U$ may be necessary to obtain the row-reduced echelon form shown in Eq. (5.73). Of course, in order to maintain consistency in the equations, column permutations in $U$ must be accompanied by corresponding row permutations in $h^c$.

From Eq. (5.73), we may now express the basic variables in terms of the free variables as follows:

$$h^c_b = w_r + (-U_r) h^c_f.$$  \hspace{1cm} (5.74)
Thus, a general solution for $h^c$ may be written:

$$h^c = \begin{bmatrix} h^c_b \\ h^c_f \end{bmatrix} = \begin{bmatrix} w_r \\ 0 \end{bmatrix} + \begin{bmatrix} -U_r \\ 1_{(n_c-r)} \end{bmatrix} h^c_f,$$

(5.75)

or

$$h^c = (h^c)_{part.} + (h^c)_{hom.},$$

(5.76)

where

$$(h^c)_{part.} = \text{the particular solution},$$

$$(h^c)_{hom.} = \text{the homogeneous solution}.$$

By definition, $(h^c)_{hom.}$ lies in the nullspace of $U$. That is,

$$U (h^c)_{hom.} = 0.$$

(5.77)

The nullspace of $P$ is the same as the nullspace of $U$ [41]. Therefore, assuming that all the column permutations needed for $U$ are included in $P$, we may write:

$$P (h^c)_{hom.} = 0.$$

(5.78)

In general, then,

$$Ph^c = P [(h^c)_{part.} + (h^c)_{hom.}],$$

(5.79)

$$= P(h^c)_{part.} + 0,$$

(5.80)

$$= P(h^c)_{part.},$$

(5.81)

$$= s.$$  

(5.82)

Thus, a correct solution for $h^c$ is obtained for any value of $(h^c)_{hom.}$ The simplest such solution, of course, exists when $(h^c)_{hom.}$ is identically zero. This condition implies that the free variables are all zero. That is,

$$h^c_f = 0.$$

(5.83)
In this case, the general solution for the unknown contact forces is:

\[
\mathbf{h}^c = (\mathbf{h}^c)_{\text{part.}} = \begin{bmatrix} \mathbf{w}_r \\ 0 \end{bmatrix}.
\]  

(5.84)

In addition to being straightforward and simple, this solution also requires the least amount of computation.

The existence of multiple solutions for \( \mathbf{h}^c \) may be interpreted physically as follows. The rank of \( \mathbf{P} \) may be less than \( n_c \) under several conditions: (1) the given manipulator has fewer than six degrees of freedom, and/or (2) the manipulator is in a singular position. In either case, we say that the mobility of the end effector is decreased. As discussed briefly by Khatib [21], for any configuration in which the mobility of the end effector is less than six, there exists one or more “singular directions” for the end effector. In these directions, the end effector presents infinite inertial resistance to any motion, i.e., infinite mass for displacements or infinite inertia for rotations. So, any force or moment exerted on or by the end effector in these directions will have no effect on its motion, no matter how great the magnitude. These directions correspond to the free variables of \( \mathbf{h}^c \), and because they have no effect on the motion of the system, they may be assigned any value. Since any value is correct, the simplest assumption to make is that the free variables are zero.

When \( r < n_c \), the solution approach discussed above for \( \mathbf{h}^c \) is appropriate for a simulation in which only the motion of the mechanism is desired, and the exact magnitudes of the constraint forces are not required. If an exact solution is desired for \( \mathbf{h}^c \), then a more detailed model of the system will be needed which includes some measures of stiffness.

The greatest amount of computation for Step 2 is required when \( \mathbf{P} \) has full rank. In this case, a single unique solution must be found for \( \mathbf{h}^c \) from \( n_c \) lin-
ear simultaneous equations. The computational complexity of this linear system solution is $O(n_c^3)$.

5.6.3 Step 3. Calculation of the Closed-Chain Joint Accelerations

The formulation of the simulation algorithm described here makes this third step particularly straightforward. With the contact force vector, $f$, completely defined, the vector of closed-chain joint accelerations, $\dot{q}$, may be found using Eq. (5.6), which is repeated here for convenience:

$$\dot{q} = \dot{q}_{open} - \Omega f.$$  (5.85)

The open-chain term, $\dot{q}_{open}$, and the force vector coefficient, $\Omega$, are known from Step 1, and $f$ is known from Step 2. The computational complexity of this step is $O(N)$. With this step, the Direct Dynamics problem has been completely solved for this single closed chain system.

5.6.4 Step 4. Integration for the Next State

The final step in any dynamic simulation algorithm is the numerical integration of the joint accelerations and rates to determine the next state joint rates and positions. Many different integration methods may be found in the literature to accomplish this task. Familiar examples include the fourth-order Runge-Kutta method and various predictor-corrector methods. Because a predictor-corrector method needs only two joint rate or acceleration evaluations for each iteration and a fourth-order Runge-Kutta method needs four such evaluations for the same degree of accuracy, a predictor-corrector method is more efficient [42].

A problem faced in this last step of any simulation algorithm is the drift due to numerical integration. Typically, the joint positions obtained through integration and transformed to an end effector position do not satisfy the physical constraints
of the robot system. To compensate for this, end effector position and rate feedback may be used to modify the general end effector contact force vector as follows:

\[
\mathbf{f}_m = \mathbf{f} + \mathbf{K}_v (\mathbf{x}_c - \dot{\mathbf{x}}) + \mathbf{K}_p (\mathbf{x}_c - \mathbf{x}),
\]

(5.86)

where

- \( \mathbf{f}_m \) = modified contact force vector,
- \( \mathbf{f} \) = original contact force vector,
- \( \mathbf{x}_c, \dot{\mathbf{x}}_c \) = constrained end effector position and rate vectors,
- \( \mathbf{x}, \dot{\mathbf{x}} \) = calculated end effector position and rate vectors,
- \( \mathbf{K}_v, \mathbf{K}_p \) = rate and position feedback gains.

This modification is modeled after a similar treatment of the integration drift problem described in [42,43]. Note that \( \mathbf{x}_c \) and \( \dot{\mathbf{x}}_c \) are known and constant for the given application. The two terms, \( \mathbf{x} \) and \( \dot{\mathbf{x}} \), may be found from the present state joint variables. The position difference term, \( (\mathbf{x}_c - \mathbf{x}) \), may be found from a position deviation matrix as described in [42]. Thus, the entire modified contact force vector is known as a function of the present state only.

The new modified value of the contact force vector may now be used in the calculation of the joint accelerations as follows:

\[
\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_{open} - \Omega \mathbf{f}_m.
\]

(5.87)

It is expected that the negative rate and position feedback in the modified force term will eliminate the drift problem in the integration step. The presence of these feedback terms may be interpreted physically as the presence of a spring and damper combination inserted at the contact between the end effector and the
other rigid body. Any difference between the calculated and constrained end effector rates and positions results in a “restoring force” from the spring/damper combination which “restores” the end effector/rigid body contact to its desired condition. Small values for the feedback gains, $K_v$ and $K_p$, correspond to a “soft” spring/damper combination. Such a combination is desirable because high integration rates are not required to track its behavior.

5.7 Computational Requirements

In this section, the computational requirements of the new simulation algorithm for a single closed chain are summarized and discussed. The number of required scalar operations is tabulated for each step, with the exception of the integration step. The operations required for integration are usually not included in the overall computational complexity of a simulation algorithm.

Tables 19 and 20 list the computational requirements for the new single closed-chain dynamic simulation algorithm, using the most efficient algorithms known for each calculation for different values of $N$. The computations are tabulated in terms of the matrix and vector quantities which are found in the first three steps of the algorithm. The required scalar operations (multiplications, additions) are given for an $N$-link, serial manipulator with simple revolute and prismatic joints only. The efficient matrix transformations and other simplifications described in Chapter 3 have been applied in each step, and the computations necessary to determine the individual link transformation matrices have also been included.

The algorithms used to compute the open-chain acceleration terms, $\dot{q}_{\text{open}}$ and $\ddot{x}_{\text{open}}$, the unknown contact forces and/or moments, $h^c$, and the closed-chain joint accelerations, $\ddot{q}$, are the same in both tables. The open-chain Direct Dynamics algorithm of Brandl, Johanni, and Otter [10] is used to compute $\dot{q}_{\text{open}}$ and $\ddot{x}_{\text{open}}$. 

126
Table 19: Computations for the Single Closed-Chain Dynamic Simulation Algorithm, $O(N^3)$

<table>
<thead>
<tr>
<th>Calculation</th>
<th>#Mult.</th>
<th>#Add.</th>
<th>#Mult.</th>
<th>#Add.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(N = 6, n_c = 3)$</td>
<td>$(N = 6, n_c = 3)$</td>
<td>$(N = 6, n_c = 3)$</td>
<td>$(N = 6, n_c = 3)$</td>
</tr>
<tr>
<td>$\dot{q}<em>{\text{open}}, \ddot{x}</em>{\text{open}}$</td>
<td>$250N - 182$</td>
<td>$220N - 167$</td>
<td>$1318$</td>
<td>$1153$</td>
</tr>
<tr>
<td>$\Omega, \Lambda^{-1}$</td>
<td>$\frac{1}{6}N^3 + 19N^2 + 44\frac{5}{6}N - 37$</td>
<td>$\frac{1}{6}N^3 + 14N^2 + 54\frac{5}{6}N - 69$</td>
<td>$952$</td>
<td>$800$</td>
</tr>
<tr>
<td>$h^c, (f)$</td>
<td>$-\frac{2}{3}n_c^3 + 6n_c^2 + 5\frac{3}{2}n_c$</td>
<td>$-\frac{2}{3}n_c^3 + 5\frac{1}{2}n_c^2 + 6\frac{1}{6}n_c$</td>
<td>$53$</td>
<td>$50$</td>
</tr>
<tr>
<td>$\dot{q}$</td>
<td>$6N$</td>
<td>$6N$</td>
<td>$36$</td>
<td>$36$</td>
</tr>
<tr>
<td>Total:</td>
<td>$(\frac{1}{6}N^3 + 19N^2 + 300\frac{5}{6}N - 219)$</td>
<td>$(\frac{1}{6}N^3 + 14N^2 + 280\frac{5}{6}N - 236)$</td>
<td>$2359$</td>
<td>$2039$</td>
</tr>
</tbody>
</table>
Table 20: Computations for the Single Closed-Chain Dynamic Simulation Algorithm, $O(N)$

<table>
<thead>
<tr>
<th>Calculation</th>
<th>#Mult.</th>
<th>#Add.</th>
<th>#Mult. $(N = 6, n_c = 3)$</th>
<th>#Add. $(N = 6, n_c = 3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_{open}, \bar{x}_{open}$</td>
<td>$250N - 182$</td>
<td>$220N - 167$</td>
<td>$1318$</td>
<td>$1153$</td>
</tr>
<tr>
<td>$\Omega, \Lambda^{-1}$</td>
<td>$400N - 621$</td>
<td>$320N - 528$</td>
<td>$1779$</td>
<td>$1392$</td>
</tr>
<tr>
<td>$h^c, (f)$</td>
<td>$-\frac{2}{3}n_c^3 + 6n_c^2 + 5\frac{2}{3}n_c$</td>
<td>$-\frac{2}{3}n_c^3 + 5\frac{1}{2}n_c^2 + 6\frac{1}{6}n_c$</td>
<td>$53$</td>
<td>$50$</td>
</tr>
<tr>
<td>$\bar{q}$</td>
<td>$6N$</td>
<td>$6N$</td>
<td>$36$</td>
<td>$36$</td>
</tr>
<tr>
<td>Total:</td>
<td>$656N - 803$</td>
<td>$546N - 695$</td>
<td>$3186$</td>
<td>$2631$</td>
</tr>
</tbody>
</table>
This is the most efficient open-chain algorithm known for computing both of these quantities for all $N$, and its computational complexity is $O(N)$ as shown. The unknown contact forces and/or moments, $h^c$, are found by linear system solution of the algebraic equations given in Eq. (5.57) or (5.67), depending on the class of the tip contact. The computational complexity of this task is $O(n_c^3)$, where $n_c \leq 6$. Because the solution for $h^c$ requires more operations when a Class II contact is involved, the operations needed for solving Eq. (5.67) are listed in both tables (worst case). Note that the characteristic matrix in Eq. (5.67) is not necessarily symmetric. Finally, the closed-chain joint accelerations, $\ddot{\mathbf{q}}$, are found using Eq. (5.6).

The two tables differ only in the algorithm used to compute the inverse operational space inertia matrix, $\mathbf{\Lambda}^{-1}$, and the coefficient $\Omega$. In Chapter 4, the efficient computation of these two quantities was discussed in some detail. It was determined that the Unit Force Method (Method II) is the most efficient algorithm for these two matrices together for $N < 21$. The Force Propagation Method (Method III) is the best solution for $\mathbf{\Lambda}^{-1}$ and $\Omega$ for $N \geq 21$. The scalar operations required for Method II are used in Table 19, while those required for Method III are used in Table 20.

Note that the number of operations listed for $\Omega$ and $\mathbf{\Lambda}^{-1}$ in Table 20 is less than the total given for these two quantities in the $O(N)$ Force Propagation Method in Chapter 4. This reduction was achieved through a little insight. First, we note that the first recursion in the open-chain Direct Dynamics algorithm of Brandl, Johanni, and Otter [10] computes the articulated-body inertia of each link in the chain, starting at the tip and moving back to the base. This same recursion is the first recursion in the Force Propagation Method for computing $\mathbf{\Lambda}^{-1}$. That is, there is an overlap of computations between the solution for the open-chain
acceleration terms, $q_{open}$ and $x_{open}$, and the calculation of the inverse operational space inertia matrix, $\Lambda^{-1}$, for this case. This fact was taken into account when the operations were tabulated. The operations listed for $\Omega$ and $\Lambda^{-1}$ in Table 20 include only the second recursion for $\Lambda^{-1}$ and the additional operations needed for $\Omega$. The recursion which computes the articulated-body inertias is included in the computations for $q_{open}$ and $x_{open}$.

The computational complexity of the complete dynamic simulation algorithm for a single closed chain shown in Table 19 is $O(N^3)$ for a given $n_c \leq 6$. This is a result of the $O(N^3)$ operations needed for the Unit Force Method for $\Omega$ and $\Lambda^{-1}$. In Table 20, however, the computational complexity of the complete simulation algorithm has been reduced to $O(N)$ by using the Force Propagation Method. The development of an $O(N)$ algorithm for the simulation of a single closed chain is an important result for several reasons. First, because of the reduced order of complexity, this $O(N)$ approach becomes more efficient than the $O(N^3)$ approach for $N \geq 15$. In addition, the structure of $O(N)$ algorithms often facilitates their implementation on parallel computer systems. Hence, an $O(N)$ dynamic simulation algorithm is a significant and valuable accomplishment.

The operations required for the special case of $N = 6$ and $n_c = 3$ are given in the last two columns of each table. This value of $n_c$ might correspond to a hard point contact between the manipulator tip and a constraining body or surface when the tip is not slipping.

5.8 Conclusions and Summary

In this chapter, a new efficient serial algorithm for the dynamic simulation of a single closed chain has been developed. The main feature of this algorithm is the use of the operational space inertia matrix to project the dynamics of the chain
to its tip, or end effector. This projection allows the knowledge of the contact characteristics to be used to good advantage in solving for the unknown contact forces and/or moments at that point. Once the contact force vector is completely determined, the joint accelerations for the closed chain may be found.

Two computational versions of this algorithm were developed based on the two different algorithms used to calculate the matrices $\Lambda^{-1}$ and $\Omega$. The first version has a computational complexity which is $O(N^3)$, and it is the most efficient of the two for $N < 15$. The second version has a reduced order computational complexity which is only $O(N)$. This version of the algorithm is the most efficient known for $N \geq 15$. The development of an $O(N)$ algorithm for the dynamic simulation of a closed chain is a significant accomplishment in itself. $O(N)$ algorithms are ultimately more efficient than higher order algorithms, and their implementation on parallel computer systems may be easier. In the next chapter, we will show that the same techniques used to develop these efficient algorithms for a single closed chain may be extended to the case of multiple closed-chain systems.
CHAPTER VI

EFFICIENT DYNAMIC SIMULATION OF SIMPLE
CLOSED-CHAIN MECHANISMS

6.1 Introduction

A simple closed-chain mechanism is one example of a multiple chain robotic system. Many different robotic systems may be modelled as simple closed-chain mechanisms, including multiple manipulators with a common load, a dexterous hand with its fingers manipulating an object, and multilegged vehicles. These systems all share the same basic structure, characterized by a group of actuated chains all supporting a common reference member. Naturally, the real-time simulation of these mechanisms is even more difficult than for a single closed chain. Fortunately, it is possible to extend many of the concepts used in solving the single closed chain simulation problem to this more complex case.

In this chapter, a new, efficient serial algorithm for the dynamic simulation of simple closed-chain mechanisms is derived. The development of this algorithm is based on many of the same concepts used in the previous chapter for a single closed chain. The common reference member is modelled as a single rigid body. The supporting chains may have general joints and any number of degrees of freedom. For convenience, it is assumed that the chains themselves have no internal closed loops. Note, however, that the fundamental simulation method developed in this chapter may also be applied when the chains have a more general structure.
A brief review of previous work related to the dynamic simulation of simple closed-chain mechanisms is given in the second section of this chapter. Next, a model for simple closed-chain mechanisms is described, and the basic problem statement is discussed. The dynamic equations of motion for the individual chains and the common reference member are summarized in the fourth section. The fifth section presents the development of an efficient serial algorithm for the dynamic simulation of a simple closed-chain mechanism. In many ways, this algorithm is an extension of the algorithm for a single closed chain given in Chapter 5. It is valid for both Type 0 and Type 1 simple closed-chain mechanisms. Initially, spatial acceleration vectors are matched at the coupling point between each chain and the reference member. The resulting equations are combined with the general joint model to solve for the spatial acceleration of the reference member and the unknown forces exerted on it by each chain. The operational space inertia matrix of each chain plays an important role in these calculations. Once the spatial tip forces are defined, the system is effectively decoupled, and the closed-chain joint accelerations may be computed for each chain separately.

An alternate form of the algorithm is then presented which is based on resolving the dynamics of each chain to the reference member. This dynamic analysis also leads to a solution for the spatial acceleration of the reference member and the resolved force vectors exerted on it by each chain. Once again, after the resolved forces are completely defined, the closed-chain joint accelerations may be computed for each chain separately. This alternate approach is less efficient than the initial approach discussed above, but it provides a slightly different perspective which is useful and interesting.

The computational requirements of the new algorithm, as well as the potential for its parallel implementation, are also discussed. A summary of the results and
some overall conclusions are given in the final section of the chapter.

6.2 Previous Work

While considerable effort has been spent studying the simulation problem for single closed chains, fewer results are available for more complex multiple chain robotic systems. Existing algorithms for simple closed-chain mechanisms are, in general, difficult to apply and/or computationally inefficient [13,16,20]. Some of the relevant references will now be briefly discussed.

In [13], Oh and Orin extend the basic method of Orin and McGhee [12] to include simple closed-chain mechanisms with $m$ chains of $N$ links each. The dynamic equations of motion for each chain are combined with the net force and moment equations for the reference member and the kinematic constraint equations at the chain tips to form a large system of linear algebraic equations. The unknowns are the joint accelerations for all the chains, the constraint forces applied to the reference member, and the spatial acceleration of the reference member. To find the joint accelerations, this system must be solved as a whole via standard elimination techniques. Although this approach is straightforward, its computational complexity of $O(m^3N^3)$ is high.

In [16], Brandl, Johanni, and Otter extend their simulation algorithm for open chains [10] to multibody systems with kinematic loops. The rigid bodies of the system are connected by general joints. The computational complexity of the algorithm is $O(n_B) + O(n_L)$ in the best case, where $n_B$ is the total number of bodies in the system, and $n_L$ is the number of kinematic loops. In the worst case, the algorithm requires $O(n_B) + O(n_L^2)$ operations. While this algorithm is quite general, its practicality has yet to be evaluated through detailed examples.

In [20], Rodriguez and Kreutz utilize their linear operator methods to derive
a simulation algorithm for multiple arms rigidly grasping a common object. For \( m \) actuated chains, each having \( N \geq 6 \) degrees of freedom, the computational complexity of this algorithm is \( O(mN) + O(m^3) \). This is similar to the results found in [16] above. Unfortunately, as discussed in Chapter 4, the manipulation techniques used in this work make the physical interpretation of the results somewhat difficult.

6.3 System Configuration and Problem Statement

Multiple chain robotic systems can take many forms, some of them quite complex. Simple closed-chain mechanisms are a subset of multiple chain systems with specific structural characteristics. In this section, a model for simple closed-chain mechanisms is described, and the nature of the simulation problem for these mechanisms is discussed.

6.3.1 Model for Simple Closed-Chain Mechanisms

The structure of a simple closed-chain mechanism is characterized by \( m \) actuated chains which support a single common reference member [13]. The supporting chains are assumed to be serial-link chains, free of internal closed loops. Therefore, the removal of the reference member breaks all closed loops in the system. Each chain may have an arbitrary number of links and degrees of freedom. The interactions which occur between bodies or links in the system (including those at the support surface and at the reference member) may be described using the general joint model of Chapter 2. This includes both powered joint structures and unpowered contacts. The motion of the support surface is assumed to be known.

As was mentioned briefly in Chapter 1, there are two basic types of simple closed-chain mechanisms called Type 0 and Type 1, respectively [13]. These two
types are defined based on the nature of the interactions which occur between the links of the chain and the reference member or support surface. For both types, the support surface acts as the “base” of each chain. We will refer to the link which interacts with the support surface as link 1, and the link which interacts with the reference member will be called the last link or end effector (link N). The far end of link N is the “tip” of the chain.

Figure 13 illustrates a typical Type 0 mechanism which may be used to model multiple manipulators or dexterous hands. Note that the support surface, shown here as a fixed inertial frame for a multiple manipulator configuration, might also represent the moving “palm” of a dexterous hand. In either case, for a Type 0 mechanism, the first link of each chain is connected to the support surface by an actuated joint structure, while the last link interacts with the reference member through an unpowered contact.

Figure 14 illustrates a Type 1 simple closed-chain mechanism which may be used to model multilegged vehicles. For a Type 1 mechanism, the first link of each
Figure 14: Example of a Type 1 Simple Closed-Chain Mechanism

chain interacts with the support surface through an unpowered contact, while the last link is connected to the reference member by an actuated joint structure. For both Type 0 and Type 1 mechanisms, the reference member (object, load, or body) is numbered 0, while the chains are numbered arbitrarily from 1 to \( m \). Chain \( k \) (\( k = 1, \ldots, m \)) has \( N_k \) degrees of freedom, where \( N_k \) may be less than, equal to, or greater than 6.

Note that the use of the general joint model eliminates many of the confusing notational issues which often arise in the initial stages of analysis of complex robotic systems. Details, such as the location of coordinate frame origins, become application dependent. They may be specified at a later stage in the analysis in the most convenient way for the problem at hand. This fact allows us to study the given problem at a higher analytical level, with less encumbering mathematics and better physical insight.
6.3.2 General Problem Statement

The dynamic simulation problem for simple closed-chain mechanisms, though more complex than the single closed chain problem, is still very similar in its basic form. The present state of the entire system is given, and certain force and acceleration components are known at the general joints. The motion of the system which results from the known applied forces is desired. The additional complexity arises because the chains of the system are coupled through a common reference member. The goal of this chapter is the development of an analytical approach which will decouple the system, allowing each chain to be analyzed independently.

To be more specific, in this dynamic simulation problem, the position and velocity of each body in the system is known from given information. This includes the links of each chain and the reference member. The motion of the support surface (velocity and acceleration) is also given. At the general joints, the torques and/or forces which are applied in the free directions and the relative accelerations in the constrained directions are known. Unknown quantities include the spatial acceleration of the reference member, the relative accelerations at the general joints in the free directions, and the torques and/or forces in the constrained directions at the general joints. For simulation, the joint accelerations and the acceleration of the reference member are desired, since they are needed to compute the next state positions and rates. Because the chains and the reference member are coupled, the unknown forces which the chains apply to the reference member will also be computed in order to obtain these desired accelerations.

6.4 System Dynamic Equations

As in the single closed chain problem, we will begin our analysis with the dynamic equations of motion for the entire simple closed-chain system. First, we
will consider the dynamic equations for each supporting chain, and then we will formulate an appropriate dynamic equation for the reference member alone.

6.4.1 Dynamic Equations of Motion for Each Chain

Each chain in the simple closed-chain mechanism is governed by the dynamic equations of motion for a single chain. These are:

\[ \tau_k = H_k \ddot{q}_k + C_k \dot{q}_k + G_k + J_k^T f_k; \quad k = 1, \ldots, m, \] (6.1)

where

\[ \tau_k = N_k \times 1 \text{ applied general joint torque/force vector,} \]
\[ \dot{q}_k, \ddot{q}_k = N_k \times 1 \text{ general joint rate and acceleration vectors,} \]

and

\[ f_k = 6 \times 1 \text{ spatial force vector exerted by chain } k \text{ on the reference member.} \]

The quantities $H_k$, $C_k$, $G_k$, and $J_k$ are, of course, the joint space inertia, centripetal/Coriolis, gravity, and Jacobian matrices for chain $k$, respectively. They are all functions of the general joint position and rate vectors, $q_k$ and $\dot{q}_k$. Recall that the "base" of each chain is the support surface, and the "tip" of each chain touches the reference member. The components of $q_k$, $\dot{q}_k$, and $\tau_k$ correspond to the general joints of each chain, starting with the joint between link 1 and the support surface and ending with the joint between links $N - 1$ and $N$. The only unknowns in Eq. (6.1) are the general joint accelerations, $\ddot{q}_k$, and the components of the force vector, $f_k$, in the constrained directions of the general joint at the tip of chain $k$. 139
As in Chapter 5, we may use the dynamic equations of motion to partition the general joint acceleration and spatial tip acceleration vectors of each chain into the difference of two terms, one known and one unknown. For each chain, we may write:

\[ \ddot{q}_k = (\ddot{q}_k)_{\text{open}} - (H_k^{-1}J_k^T) f_k, \quad (6.2) \]

\[ = (\ddot{q}_k)_{\text{open}} - \Omega_k f_k, \quad (6.3) \]

where \((\ddot{q}_k)_{\text{open}}\) is the vector of general joint accelerations for chain \(k\) in an open, unconstrained configuration, and \(\Omega_k\) is a function of the general joint positions for chain \(k\). Likewise, for each chain:

\[ \ddot{x}_k = (\ddot{x}_k)_{\text{open}} - (J_k H_k^{-1}J_k^T) f_k, \quad (6.4) \]

\[ = (\ddot{x}_k)_{\text{open}} - \Lambda_k^{-1} f_k, \quad (6.5) \]

where \((\ddot{x}_k)_{\text{open}}\) is the spatial tip acceleration vector for chain \(k\) in an open, unconstrained configuration, and \(\Lambda_k^{-1}\) is the inverse operational space inertia matrix for chain \(k\), defined at the tip of the chain.

As in the single closed chain case, the open-chain terms, \((\ddot{q}_k)_{\text{open}}\) and \((\ddot{x}_k)_{\text{open}}\), are completely defined for each chain given the present state general joint positions and rates, \(q_k\) and \(\dot{q}_k\), the applied general joint torques/forces in the free directions, \(\tau_k\), and the motion of the support surface. An appropriate open-chain Direct Dynamics algorithm may be used to calculate these terms. Because the general joint positions are known, \(\Omega_k\) and \(\Lambda_k^{-1}\) are also defined. The efficient computation of \(\Omega_k\) and \(\Lambda_k^{-1}\) for a single serial-link chain was discussed in detail in Chapter 4.

### 6.4.2 Dynamic Equation for the Reference Member

The dynamic behavior of the reference member may be described using a spatial force balance equation for that body. The sum of the spatial forces exerted
by each chain on the reference member and any other external spatial forces (including gravity) are equal to the resultant force on the reference member. Using spatial notation, we may write the force balance equation as follows:

\[ F_0 = \sum_{k=1}^{m} 0f_k + g_0, \quad (6.6) \]

where

\[ F_0 = 6 \times 1 \text{ resultant spatial force vector applied to the reference member,} \]
\[ 0f_k = 6 \times 1 \text{ spatial force vector applied by chain } k \]
\[ \text{to the reference member,} \]

and

\[ g_0 = 6 \times 1 \text{ external spatial force vector applied to the reference member (including gravity).} \]

Each force term in Eq. (6.6) is defined with respect to the coordinate frame attached to the reference member (frame 0). By definition, we may also write the resultant vector, \( F_0 \), as follows:

\[ F_0 = I_0 a_0 + v_0 \times I_0 v_0, \quad (6.7) \]
\[ = I_0 a_0 + b_0, \quad (6.8) \]

where

\[ I_0 = 6 \times 6 \text{ spatial inertia of the reference member,} \]
\[ a_0 = 6 \times 1 \text{ spatial acceleration of the reference member,} \]
\[ v_0 = 6 \times 1 \text{ spatial velocity of the reference member.} \]
Both \( v_0 \) and \( a_0 \) refer to the motion of the coordinate origin of frame 0. The spatial inertia matrix, \( I_0 \), is also defined at this point, and it is known and constant. Because \( v_0 \) is given for the present state, the velocity-dependent term, \( b_0 \), is known. If we combine Eqs. (6.6) and (6.8), we finally obtain the following dynamic equation for the reference member:

\[
\sum_{k=1}^{m} 0 f_k + g_0 = I_0 a_0 + b_0.
\]

(6.9)

In this equation, the only unknowns are \( a_0 \) and the components of \( 0 f_k \) in the constrained directions of the general joint at the tip of chain \( k \).

### 6.5 Dynamic Simulation Algorithm for Simple Closed-Chain Mechanisms

In developing an efficient algorithm for the dynamic simulation of simple closed-chain mechanisms, we are naturally led to consider the relationship between the physical structure of the robotic system and the computational structure of the desired algorithm. Intuitively, it seems apparent that the structural parallelism present in a simple closed-chain mechanism should lead to computational parallelism in the solution of the Direct Dynamics problem for that mechanism.

More specifically, in a simple closed-chain mechanism, the \( m \) actuated chains act on the reference member in parallel, and their motion is coupled with that of the reference member. If the reference member is removed, the chains may function independently. Computationally, the physical removal of the reference member corresponds to solving for the forces which it exerts on each chain. Once these forces are known, the system is equivalent to a group of independent chains with known tip forces. The general joint accelerations may then be computed for each chain separately. Given enough processors (one per chain), the computations for each chain may be carried out in parallel.
We may illustrate the basic methodology of the new simulation algorithm developed here by first examining a simple special case. Consider \( m \) serial-link manipulators rigidly grasping a common object. Each manipulator has six degrees of freedom, and no chain is in a singular position. For simplicity, we will express all of the relevant equations in absolute coordinates. Because each chain tip is rigidly attached to the reference member, we may write:

\[
\dot{x}_k = a_0
\]  

(6.10)

for each chain \( k, k = 1, \ldots, m \). Thus, the operational space dynamic equation for each chain, as given in Eq. (6.5), takes the form:

\[
a_0 = (\dot{x}_k)_{open} - \Lambda_k^{-1} f_k.
\]  

(6.11)

Because no chain is in a singular position, and each chain has a full six degrees of freedom, \( \Lambda_k \) is defined. We may therefore solve for the spatial tip force exerted by chain \( k \) on the reference member, \( f_k \), as follows:

\[
f_k = \Lambda_k \left[ (\dot{x}_k)_{open} - a_0 \right].
\]  

(6.12)

Thus, we have established an explicit relationship between the spatial tip force, \( f_k \), and the spatial acceleration of the reference member, \( a_0 \). This expression may be used in the reference member dynamic equation, given in Eq. (6.9), to obtain:

\[
\sum_{k=1}^{m} \Lambda_k \left[ (\dot{x}_k)_{open} - a_0 \right] = I_0 a_0 + b_0 - g_0.
\]  

(6.13)

The only unknown in Eq (6.13) is \( a_0 \), the spatial acceleration of the reference member. Collecting terms, we may write:

\[
\left[ I_0 + \sum_{k=1}^{m} \Lambda_k \right] a_0 = \left[ \sum_{k=1}^{m} \Lambda_k (\dot{x}_k)_{open} - b_0 + g_0 \right].
\]  

(6.14)

143
We may now solve for $a_0$ from this linear system of algebraic equations using any linear system solver. Note that the characteristic matrix is only $6 \times 6$. With $a_0$ known, we may also solve explicitly for the spatial tip force $f_k, k = 1, \ldots, m$, using Eq. (6.12). Thus, the motion of the reference member and the spatial force exerted at the tip of each chain are completely defined. The simple closed-chain mechanism is effectively decoupled. Each manipulator may now be treated as an independent chain with a known spatial tip force. The joint accelerations for each chain may be computed separately using an appropriate Direct Dynamics algorithm and then integrated to obtain the next state.

The method outlined above is quite straightforward. Of course, the illustrated example represents a special case. We will now develop a similar approach for a general simple closed-chain mechanism. Consider a mechanism with $m$ chains, each with an arbitrary number of degrees of freedom, $N$. The interaction between each chain tip and the reference member is arbitrary and will be modelled using the general joint model of Chapter 2. To begin, we will derive an explicit relationship between the spatial acceleration of each chain tip and the spatial acceleration of the reference member. The spatial acceleration of the tip of chain $k$ is denoted by $\ddot{x}_k$. The relative spatial acceleration between the tip of chain $k$ and the reference member, $\ddot{x}_k^r$, resolved in the orthogonal vector spaces of the general joint between them, may be written:

$$\ddot{x}_k^r = (\phi)_k \alpha_k + (\phi^c)_k \alpha^c_k,$$  \hspace{1cm} (6.15)

where $(\phi)_k$ and $(\phi^c)_k$ are the motion space and constraint space of the general joint at the tip of chain $k$, respectively. The quantities $\alpha_k$ and $\alpha^c_k$ are the corresponding components of relative acceleration in the free and constrained directions. For each chain, $(\phi)_k$, $(\phi^c)_k$, and $\alpha^c_k$ are known, while $\alpha_k$ is unknown. The sum of $\ddot{x}_k$ and
\( \ddot{x}_k^r \) is just the spatial acceleration of the reference member on the far side of the general joint between it and chain \( k, a_k^b \). Thus, we may write:

\[
a_k^b = \ddot{x}_k + \ddot{x}_k^r, \tag{6.16}
\]

\[
= \ddot{x}_k + (\phi)_k \alpha_k + (\phi^c)_k \alpha_k^c. \tag{6.17}
\]

We may also express \( a_k^b \) in terms of the spatial acceleration of the reference member, \( a_0 \), as follows:

\[
a_k^b = X_k^0 a_0 + \zeta_k^0, \tag{6.18}
\]

where \( X_k^0 \equiv N X_0^k \) is the spatial transformation between coordinate frame 0 and the coordinate frame associated with the general joint at the tip of chain \( k \). The quantity \( \zeta_k^0 \) is the \( 6 \times 1 \) bias acceleration vector which is a function of the position and spatial velocity of the reference member. Because the present state of the entire system is given, both \( X_k^0 \) and \( \zeta_k^0 \) are known.

Equating the two expressions above for \( a_k^b \), we obtain the following:

\[
\ddot{x}_k + (\phi)_k \alpha_k + (\phi^c)_k \alpha_k^c = X_k^0 a_0 + \zeta_k^0. \tag{6.19}
\]

This equation matches the spatial accelerations at the coupling point between chain \( k \) and the reference member, giving an explicit relationship between \( \ddot{x}_k \) and \( a_0 \) when the coupling is arbitrary. The unknowns in Eq. (6.19) are \( \ddot{x}_k, \alpha_k, \) and \( a_0 \). All other vectors and matrices are known from the initial information given for the simulation problem.

To decouple the chains and the reference member, we need an explicit mathematical relationship between the spatial force exerted by chain \( k \) on the reference member, \( f_k \), and the spatial acceleration of the reference member, \( a_0 \). Equation (6.19) relates the spatial acceleration of the reference member and the spatial acceleration of the tip of chain \( k \). We may eliminate \( \alpha_k \), the unknown components
of the relative acceleration, by projecting Eq. (6.19) onto the constraint space of the corresponding general joint as follows:

\[(\phi^c)_k^T [\dot{x}_k + (\phi)_k \alpha_k + (\phi^c)_k \alpha_k^c] = (\phi^c)_k^T [X_k \dot{a}_0 + \zeta_0^k]. \tag{6.20}\]

Since

\[(\phi^c)_k^T (\phi)_k = 0, \tag{6.21}\]

and

\[(\phi^c)_k^T (\phi^c)_k = 1, \tag{6.22}\]

we have:

\[(\phi^c)_k^T \ddot{x}_k + \alpha_k^c = (\phi^c)_k^T [X_k \dot{a}_0 + \zeta_0^k]. \tag{6.23}\]

Equation (6.5) defines \(\ddot{x}_k\) in terms of the desired force vector, \(f_k\). If we combine Eqs. (6.5) and (6.23), we obtain:

\[(\phi^c)_k^T [(\ddot{x}_k)^\text{open} - A_k^{-1} f_k] = (\phi^c)_k^T [X_k \dot{a}_0 + \zeta_0^k] - \alpha_k^c, \tag{6.24}\]

or

\[[(\phi^c)_k^T A_k^{-1}] f_k = [\alpha_k^c - (\phi^c)_k \zeta_0^k + (\phi^c)_k^T (\ddot{x}_k)^\text{open}] - [(\phi^c)_k^T X_k \dot{a}_0]. \tag{6.25}\]

The first bracketed term on the right side of Eq. (6.25) is completely known. The only unknowns in this equation are the constraint components of the force vector, \(f_k\), and the spatial acceleration, \(a_0\). We may now pursue an explicit relationship between these two vectors.

Like the relative acceleration vector, \(f_k\) may also be resolved in the orthogonal vector spaces of the general joint at the tip of chain \(k\) as follows:

\[f_k = (\phi)_k h_k + (\phi^c)_k h_k^c, \tag{6.26}\]
where $h_k$ is the vector of known force components in the free directions, and $h_k^c$ is the vector of unknown force components in the constrained directions. (For simplicity, we will assume that all unpowered contacts are Class I as described in Chapter 5.) Combining Eqs. (6.25) and (6.26), we obtain:

$$
(\phi^c)_k^T \Lambda_k^{-1} [(\phi)_k h_k + (\phi^c)_k h_k^c] = [\alpha_k^c - (\phi^c)_k^T \zeta_k + (\phi^c)_k^T (\bar{x}_k)_{open}] - [(\phi^c)_k^T X^c_k] a_0.
$$

If the spatial acceleration of the reference member is known, we may find an explicit solution for the unknown force components at the tip of chain $k$ from the following set of linear algebraic equations:

$$
(\phi^c)_k^T \Lambda_k^{-1} (\phi^c)_k h_k^c = \left\{ \alpha_k^c - (\phi^c)_k^T \left[ \zeta_k - (\bar{x}_k)_{open} + \Lambda_k^{-1} (\phi)_k h_k \right] \right\} - [(\phi^c)_k^T X^c_k] a_0,
$$

$$
= S_k - [(\phi^c)_k^T X^c_k] a_0,
$$

where $S_k$ is known. Even when $a_0$ is unknown, we may still find a solution for $h_k^c$ in terms of the unknown $a_0$. The solution will have the following form:

$$
h_k^c = M_k \left[ S_k - (\phi^c)_k^T X^c_k a_0 \right],
$$

$$
= M_k S_k - [M_k (\phi^c)_k^T X^c_k] a_0.
$$

If $(n_c)_k$ is the number of degrees of constraint for the general joint at the tip of chain $k$, then $M_k$ is the $(n_c)_k \times (n_c)_k$ transformation matrix which solves for $h_k^c$. The basic solution procedure for $h_k^c$, even with $a_0$ unknown, is identical to the approach discussed in Section 5.6.2 for a single closed chain. Thus, this general solution is still valid for a chain in a singular position or a chain with less than six original degrees of freedom. The solution procedure requires $O[(n_c)_k^2]$ scalar operations.
Note that if chain \( k \) is rigidly grasping the reference member, then the constraint space for this general joint, \((\phi^c)_{k}\), is the \(6 \times 6\) identity matrix. In this case, also note that \(\alpha_k^c\) and \(h_k\) are identically zero for each chain. If chain \( k \) has six degrees of freedom and is not in a singular position, then \(M_k\) will be exactly equal to \(\Lambda_k\), the operational space inertia matrix for chain \( k \), and the solution for \(h_k^c\) will be:

\[
h_k^c = \Lambda_k \left[ S_k - X_{0k}^k a_0 \right]. \tag{6.32}
\]

This solution corresponds to the simple example discussed at the beginning of this section, but now expressed in local coordinates.

Given the general solution for \(h_k^c\) in Eq. (6.31), the force vector, \(f_k\), may now be written:

\[
f_k = (\phi)_k h_k + (\phi^c)_k h_k^c, \tag{6.33}
\]

\[
= [(\phi)_k h_k + (\phi^c)_k M_k S_k] - [(\phi^c)_k M_k (\phi^c)_k^T X_{0k}^k] a_0, \tag{6.34}
\]

\[
= P_k - R_k a_0, \tag{6.35}
\]

where \(P_k\) and \(R_k\) are of dimension \(6 \times 1\) and \(6 \times 6\), respectively, and both are known. We now have an explicit equation relating the force vector exerted by chain \( k \) and the spatial acceleration of the reference member. We may combine this information with the dynamic equation for the reference member to solve for \(a_0\) explicitly.

### 6.5.1 The Spatial Acceleration of the Reference Member

The dynamic equation for the reference member given in Eq. (6.9) may be rewritten as follows:

\[
\sum_{k=1}^{m} (X_{0k}^k)^T f_k + g_0 = I_0 a_0 + b_0. \tag{6.36}
\]

148
where $f_k$ is the spatial force exerted by chain $k$ on the reference member, expressed in the coordinate frame of the general joint at the chain tip. If the expression for $f_k$ in Eq. (6.35) is used in Eq. (6.36), we obtain:

$$\sum_{k=1}^{m} (X_k^0)^T (P_k - R_k a_0) = I_0 a_0 + b_0 - g_0. \quad (6.37)$$

Summing like terms, we may write:

$$m \sum_{k=1}^{m} (X_k^0)^T R_k a_0 = \sum_{k=1}^{m} (X_k^0)^T P_k - b_0 + g_0, \quad (6.38)$$

or, expanding $R_k$,

$$\left[ I_0 + \sum_{k=1}^{m} (X_k^0)^T \phi^c_k M_k (\phi^c)^T_k (X_k^0) \right] a_0 = \left[ \sum_{k=1}^{m} (X_k^0)^T P_k - b_0 + g_0 \right]. \quad (6.39)$$

In Eq. (6.39), the $6 \times 1$ spatial acceleration vector of the reference member, $a_0$, is the only unknown. We may find a solution for $a_0$ from the given set of linear algebraic equations, or we may write the following analytical expression for $a_0$:

$$a_0 = \left[ I_0 + \sum_{k=1}^{m} (X_k^0)^T \phi^c_k M_k (\phi^c)^T_k (X_k^0) \right]^{-1} \left[ \sum_{k=1}^{m} (X_k^0)^T P_k - b_0 + g_0 \right]. \quad (6.40)$$

Because the required inversion is always $6 \times 6$, the computational cost of solving for $a_0$ is constant.

The coefficient matrix of $a_0$ in Eq. (6.39) represents the combined inertial properties of all the chains and the reference member. The inertial properties of each chain are first projected to the tip of that chain by computing the inverse operational space inertia matrix, $\Lambda_k^{-1}$. Along the free directions of the general joint which connects the chain tip and the reference member, the projected inertia of the chain is not felt by the reference member. Along the constrained directions of the joint, however, the corresponding components of $\Lambda_k^{-1}$ are reflected across to the reference member. These components, spatially transformed to the coordinate
origin of frame 0, are combined with the spatial inertia of the reference member, $I_0$. This combination represents the effective operational space inertia of the simple closed-chain mechanism defined at the coordinate origin of frame 0. It is the effective inertia “felt” by the reference member in the present state. The bracketed term on the right side of Eq. (6.39) represents the spatial forces which act on the reference member at the given instant.

### 6.5.2 The Chain Tip Forces and Closed-Chain Joint Accelerations

Once the spatial acceleration of the reference member is known, the spatial force vector applied by chain $k$ to the reference member is defined by Eq. (6.35). That is, with $a_0$ given, we may compute $f_k$ as follows:

$$f_k = P_k - R_k a_0.$$  

(6.41)

Recall that $f_k$ is defined with respect to the coordinate frame of the general joint between chain $k$ and the reference member. The explicit knowledge of $f_k$ allows us to treat chain $k$ as an independent chain with a known tip force. We may now solve for the general closed-chain joint accelerations for chain $k$ using Eq. (6.3), repeated here for convenience:

$$\ddot{q}_k = (\ddot{q}_k)_{\text{open}} - \Omega_k f_k.$$  

(6.42)

The application of Eq. (6.42) to every actuated chain in the simple closed-chain mechanism results in a complete solution to the Direct Dynamics problem for this robotic system. The next state positions and velocities may be computed by integrating the appropriate quantities for each chain and the reference member. As discussed in Chapter 5, small amounts of negative position and rate feedback may be employed to counteract the drift which is a result of the integration process.
6.5.3 The Simulation Algorithm in Five Steps

Like the dynamic simulation algorithm for a single closed chain, the algorithm developed here for simple closed-chain mechanisms may also be presented as a series of steps. In this case, five steps are required, and they are as follows:

1. The Open Chain Solution,
2. Calculation of the Spatial Acceleration of the Reference Member,
3. Calculation of the Spatial Chain Tip Forces,
4. Calculation of the Closed-Chain Joint Accelerations,
5. Integration for the Next State.

The fundamental computations required in each of these steps are summarized in Table 21.

In Step 1, the Direct Dynamics problem is solved for each chain of the mechanism assuming that the reference member has been removed and each chain is in an open, unconstrained state. The general open-chain acceleration vectors, \((\ddot{q}_k)_{\text{open}}\) and \((\ddot{x}_k)_{\text{open}}\), are computed for each chain, along with the position-dependent matrices, \(\Omega_k\) and \(A_k^{-1}\). In Step 2, Eq. (6.38) is used to find an explicit solution for \(a_0\), the spatial acceleration of the reference member, via linear system solution. The quantities \((M_k S_k)\) and \(\left[M_k (\phi^c)^T X_k \right]\), required for both \(P_k\) and \(R_k\), are computed in the determination of the explicit relationship between \(h_k^F\) and \(a_0\). This relationship is found by linear system solution using Eq. (6.29), with the solution taking the form of Eq. (6.31), repeated for convenience in Table 21. In Step 3, this solution is used in Eq. (6.41) to solve for the spatial force vector exerted on the reference member by each chain. In Step 4, the general closed-chain joint accelerations are computed for each chain using Eq. (6.42), given the spatial tip force
Table 21: Dynamic Simulation Algorithm for Simple Closed-Chain Mechanisms

**Step 1.** Compute \((\ddot{q}_k)^{open}, (\dot{x}_k)^{open}, \Omega_k, \) and \(\Lambda_k^{-1}; \ k = 1, \ldots, m.\)

**Step 2.** Solve for \(a_0:\)

\[
\left[ I_0 + \sum_{k=1}^{m} (X_0^k)^T R_k \right] a_0 = \left[ \sum_{k=1}^{m} (X_0^k)^T \mathbf{P}_k - b_0 + g_0 \right],
\]

where \((M_k S_k)\) and \([M_k (\phi^c)^T X_0^k]\) are defined by:

\[
h_k^c = M_k S_k - [M_k (\phi^c)^T X_0^k] a_0,
\]

and

\[
\mathbf{P}_k = [(\phi)_k h_k + (\phi^c)_k M_k S_k],
\]

\[
R_k = [(\phi^c)_k M_k (\phi^c)^T X_0^k].
\]

**Step 3.** Solve for \(f_k; \ k = 1, \ldots, m:\)

\[
f_k = \mathbf{P}_k - R_k a_0.
\]

**Step 4.** Solve for \(\ddot{q}_k; \ k = 1, \ldots, m:\)

\[
\ddot{q}_k = (\ddot{q}_k)^{open} - \Omega_k f_k.
\]

**Step 5.** Integrate to obtain the next state positions and rates for the system.
vector. In Step 5, the appropriate rates and accelerations are integrated to obtain the next state positions and rates for all bodies in the system. The computational requirements of this algorithm will be discussed in detail in a later section of this chapter.

Note that the first step may be carried out for all chains in parallel, if enough processors are available (one per chain). Once the second step is complete and $a_0$ is known, the third, fourth, and fifth steps may also be carried out for all chains simultaneously. Thus, taking advantage of the structural parallelism inherent in the simple closed-chain system has led to parallelism in the computational structure of the simulation algorithm.

Perhaps more importantly, note that the basic approach developed here may be applied equally well to a system of chains which have internal closed loops. Of course, the computation of all vectors and matrices directly associated with the chains, such as the open-chain acceleration terms, $(\ddot{q}_k)_{open}$ and $(\ddot{x}_k)_{open}$, the inverse operational space inertia matrix, $\Lambda^{-1}$, and the coefficient matrix, $\Omega_k$, would require extended algorithms to account for the more complex chain structure.

6.6 An Alternate Form of the Simulation Algorithm

In the development of the simulation algorithm in the previous section, the objective was to decouple the simple closed-chain mechanism by computing the spatial force vectors exerted by the chains on the reference member. The spatial tip forces computed in that algorithm are real, measurable forces, associated with the general joints which connect the reference member and each chain tip. Once these forces are known, the chains are effectively decoupled from the reference member, and the general joint accelerations may be computed for each chain separately.

An alternate method of decoupling the system may be achieved by resolving
the dynamics of each chain to the reference member from the start. We begin by examining the operational space dynamics of each chain as seen at the coordinate origin of frame 0. (If desired, we may assume that this point is coincident with the center of gravity of the reference member.) Conceptually, this notion is equivalent to inserting a fictitious massless and inertialess link between each chain tip and the coordinate origin of frame 0, and analyzing the operational space dynamics at the tip of each augmented chain. Assume that such a link, link $N+1$, is added to each chain. To add a new link, we must also add a new joint between links $N$ and $N+1$. This new joint is defined to be equivalent to the general joint which coupled the original chain tip and the reference member. It retains the same motion and constraint vector spaces, but it is now included, along with the fictitious link, as part of the augmented chain. The far end of each fictitious link is rigidly attached to the reference member at the coordinate origin of frame 0.

Under these conditions, the spatial tip acceleration of augmented chain $k$, $\ddot{x}_k^q$, is exactly equal to the spatial acceleration of the reference member. That is,

$$\ddot{x}_k^q = a_0; \quad k = 1, \ldots, m.$$  \hfill (6.43)

If we combine Eq. (6.43) and the dynamic equation for chain $k$ given in Eq. (6.5), we may write

$$a_c = (\ddot{x}_k^q)_{\text{open}} - (\Lambda_k^q)^{-1} f_k^o$$  \hfill (6.44)

for each chain, where $(\ddot{x}_k^q)_{\text{open}}$ is the open-chain tip acceleration vector for the augmented chain, $(\Lambda_k^q)^{-1}$ is the inverse operational space inertia matrix for the augmented chain, and $f_k^o$ is the resolved spatial force exerted on the reference member by the augmented chain. Since the connection between the tip of each augmented chain and the reference member is immobile (a rigid grasp), all six components of $f_k^o$ are unknown. A solution for $f_k^o$ in terms of $a_0$ may be found.
via linear system solution from Eq. (6.44). Similar to the analysis of the previous section, the solution will have the following form:

\[ f^q_k = M^q_k \left[ (\ddot{x}_k^q)_{open} - a_0 \right], \]  

(6.45)

where \( M^q_k \) is a \( 6 \times 6 \) transformation matrix. If the augmented chain has at least six degrees of freedom, and if it is not in a singular position, the matrix \( M^q_k \) will be exactly equal to the operational space inertia matrix of the augmented chain. That is,

\[ M^q_k = \Lambda^q_k. \]  

(6.46)

For convenience, let us assume that this condition is true for each chain. Then, we may write:

\[ f^q_k = \Lambda^q_k \left[ (\ddot{x}_k^q)_{open} - a_0 \right]; \quad k = 1, \ldots, m. \]  

(6.47)

The dynamic equation for the reference member may now be written as follows:

\[ \sum_{k=1}^{m} f^q_k + g_0 = I_0 a_0 + b_0, \]  

(6.48)

since

\[ ^0 f^q_k = f^q_k; \quad k = 1, \ldots, m. \]  

(6.49)

If we combine Eqs. (6.47) and (6.48), we obtain:

\[ \sum_{k=1}^{m} \Lambda^q_k \left[ (\ddot{x}_k^q)_{open} - a_0 \right] = I_0 a_0 + b_0 - g_0. \]  

(6.50)

Summing like terms, we obtain the following equation:

\[ \left[ I_0 + \sum_{k=1}^{m} \Lambda^q_k \right] a_0 = \left[ \sum_{k=1}^{m} \Lambda^q_k (\ddot{x}_k^q)_{open} - b_0 + g_0 \right]. \]  

(6.51)

The only unknown in Eq. (6.51) is the spatial acceleration of the reference member, \( a_0 \). Its solution may be found using any linear system solution method. Note that
the coefficient matrix of $a_0$ will always be a $6 \times 6$ matrix. Thus, the computational cost of solving for $a_0$ is still constant. We may also write the following analytical solution for $a_0$:

$$a_0 = \left[ I_0 + \sum_{k=1}^{m} \Lambda_k^q \right]^{-1} \left[ \sum_{k=1}^{m} \Lambda_k^q \left( \dot{x}_k^q \right)_{open} - b_0 + g_0 \right]. \quad (6.52)$$

The coefficient matrix of $a_0$ in Eq. (6.51) represents the effective operational space inertia of the entire simple closed-chain mechanism as seen by the reference member at the origin of its own coordinate system. The operational space inertia of the reference member is just its spatial inertia matrix, $I_0$. Note that the operational space inertias of the augmented chains (acting in parallel on the reference member) add in a simple sum. This is a general rule for inertia matrices. For actuated chains connected in series, the combination rule is not as simple. In this case, extended versions of the recursive algorithms of Chapter 4 may be applied.

With the spatial acceleration of the reference member known, the force vector exerted by each augmented chain on the reference member, $f_k^q$, may be computed using Eq. (6.47). The explicit knowledge of $f_k^q$ allows us to treat each augmented chain as an independent chain with a known tip force. The general closed-chain joint accelerations for each augmented chain may now be computed using the equation:

$$\ddot{q}_k^q = (\ddot{q}_k^q)_{open} - \Omega_k^q f_k^q,$$  

(6.53)

where $(\ddot{q}_k^q)_{open}$ and $\ddot{q}_k^q$ are the open-chain and closed-chain joint acceleration vectors for the augmented chain $k$, respectively, and $\Omega_k^q$ is the joint space force coefficient for the same augmented chain. Note that both joint acceleration vectors include the components of relative acceleration in the free directions at the added joint. The application of Eq. (6.53) to every augmented chain in the simple closed-chain mechanism results in a complete solution to the Direct Dynamics problem.
for the mechanism. The appropriate rates and accelerations may be integrated to obtain the next state of the system.

The equations developed in this alternate formulation have a slightly simpler form than those of the previous section. Unfortunately, this approach has some disadvantages. First, the addition of the new link and joint to each chain results in added computations throughout the algorithm. For example, the open-chain acceleration term, \((\ddot{\mathbf{q}}_k^{\text{open}})\), includes additional acceleration components corresponding to the added joint. It was not necessary to compute these components in the algorithm of the previous section. Likewise, the computation of \(\mathbf{\Omega}_k^{\mathbf{e}}\) and \((\Lambda_k^{\mathbf{e}})^{-1}\) requires additional operations to account for the extra link. The second disadvantage of this formulation is related to the nature of the spatial force vectors at each chain tip. In the previous algorithm, the spatial forces exerted by each chain on the reference member are real and measurable. Here, the computed forces are fictitious and not measurable, since no real connections exist between the chain tips and the coordinate origin of the reference member. Thus, the value of this alternate formulation of the simulation algorithm is primarily conceptual and not computational.

6.7 Computational Requirements

In this section, we will consider the computational requirements of the initial dynamic simulation algorithm for simple closed-chain mechanisms presented in Section 6.5. First, the number of scalar operations required for each chain of the mechanism will be tabulated, followed by the number of operations required to compute the spatial acceleration of the reference member. The computational complexity of the complete algorithm will then be discussed. The parallel implementation of this algorithm will also be considered.
6.7.1 Number of Computations

Table 22 lists the number of scalar operations (multiplications, additions) required in the simulation algorithm for each chain of a simple closed-chain mechanism. The operations are tabulated for the case of an \( N \) degree-of-freedom chain with simple revolute and/or prismatic joints only. The \( O(N) \) Direct Dynamics algorithm of [10] is used to compute the open-chain terms, \( \ddot{q}_{\text{open}} \) and \( \ddot{x}_{\text{open}} \). The \( O(N) \) Force Propagation Method of Chapter 4 is used to compute \( \Omega \) and \( \Lambda^{-1} \). For small values of \( N \), the \( O(N^3) \) Unit Force Method is more efficient for this latter task, but its higher order computational complexity is not desirable. All \( \ddot{q}_{\text{open}} \), \( \ddot{x}_{\text{open}} \), \( \Omega \), and \( \Lambda^{-1} \) are computed in Step 1 of the simulation algorithm.

In Step 2 of the simulation algorithm, the spatial acceleration of the reference member is calculated using Eq. (6.38). For this task, \( X^T P \) and \( X^T R \) must be computed for each chain. The number of operations required to compute \( P \), \( R \), \( X^T P \), and \( X^T R \) are also listed in Table 22. In this case, the number of operations is a function of the number of degrees of constraint at the general joint between the chain tip and the reference member \( (n_c) \). This number can never be greater than six. The computational complexity of these calculations is \( O(n_c^3) \) due to the linear system solution required in the computation of both \( P \) and \( R \) (see Table 21).

The spatial force vector, \( f \), exerted by each chain on the reference member, and the closed-chain joint accelerations for the chain, \( \ddot{q} \), are calculated in Steps 3 and 4 of the simulation algorithm, respectively. The appropriate equations are given in Table 21. The operations required to calculate these vectors complete the table.

The operations required for the special case of \( N = 6 \) and \( n_c = 3 \) are given in the last two columns of Table 22. This value of \( n_c \) might correspond to a hard point.
Table 22: Computations Per Chain in the Simple Closed-Chain Dynamic Simulation Algorithm

<table>
<thead>
<tr>
<th>Calculation</th>
<th>#Mult.</th>
<th>#Add.</th>
<th>#Mult.</th>
<th>#Add.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>($N = 6, n_c = 3$)</td>
<td>($N = 6, n_c = 3$)</td>
<td>($N = 6, n_c = 3$)</td>
<td>($N = 6, n_c = 3$)</td>
</tr>
<tr>
<td>$\bar{q}<em>{open}, \bar{x}</em>{open}$</td>
<td>$250N - 182$</td>
<td>$220N - 167$</td>
<td>$1318$</td>
<td>$1153$</td>
</tr>
<tr>
<td>$\Omega, \Lambda^{-1}$</td>
<td>$400N - 621$</td>
<td>$320N - 528$</td>
<td>$1779$</td>
<td>$1392$</td>
</tr>
<tr>
<td>$P, R$</td>
<td>$\frac{1}{6}n_c^3 + 6\frac{1}{2}n_c^2 + 5\frac{1}{3}n_c + 26$</td>
<td>$\frac{1}{6}n_c^3 + 6n_c^2 + \frac{5}{3}n_c + 10$</td>
<td>$105$</td>
<td>$71$</td>
</tr>
<tr>
<td>$X^TP, X^TR$</td>
<td>$36n_c + 20$</td>
<td>$36n_c - 24$</td>
<td>$128$</td>
<td>$84$</td>
</tr>
<tr>
<td>$f$</td>
<td>$36$</td>
<td>$36$</td>
<td>$36$</td>
<td>$36$</td>
</tr>
<tr>
<td>$\bar{q}$</td>
<td>$6N$</td>
<td>$6N$</td>
<td>$36$</td>
<td>$36$</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td>$656N - 767$</td>
<td>$546N - 659$</td>
<td>$3402$</td>
<td>$2772$</td>
</tr>
<tr>
<td></td>
<td>$+$ ($\frac{1}{6}n_c^3 + 6\frac{1}{2}n_c^2 + 41\frac{1}{3}n_c + 46$)</td>
<td>$+$ ($\frac{1}{6}n_c^3 + 6n_c^2 + 36\frac{5}{6}n_c - 14$)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
contact between the manipulator tip and a constraining body or surface when the tip is not slipping.

Note that some of the entries in Table 22 may differ slightly for a Type 1 simple closed-chain mechanism under certain conditions. When all chains of the Type 1 mechanism (such as the legs of a multilegged vehicle) are in contact with the support surface, the number of computations will be the same. However, a special situation occurs if the chains break contact with the support surface while maintaining a connection to the reference member. For these chains, the most efficient computation of $\Omega$ and $\Lambda^{-1}$ is achieved using the $O(N)$ articulated-body inertia equations [19], as discussed in Chapter 4. The computational requirements of these equations are considerably less than those listed in the table, given the proper coordinate frame assignment. Other table entries may also change in a similar manner for this special configuration of a Type 1 mechanism. Also note that the chains of a Type 0 simple closed-chain mechanism may break contact with the reference member. Any such unconstrained chain may be simulated independently of the rest of the system using an appropriate open-chain simulation algorithm.

Given the computations required for each individual chain, the number of scalar operations needed to compute the spatial acceleration of the reference member, $a_0$, is given in Table 23. Equation (6.38) is used to obtain the solution, which requires $O(m)$ spatial additions and a single $6 \times 6$ symmetric linear system solution. Thus, the number of operations required for $a_0$ is a function only of $m$, the number of chains in the simple closed-chain mechanism. The example of three chains ($m = 3$) is given in the last two columns of this table.

To determine the total number of scalar operations required to simulate the entire simple closed-chain mechanism, the number of operations required for a single chain is simply multiplied by $m$, the number of chains, and added to the
Table 23: Computations for the Spatial Acceleration of the Reference Member

<table>
<thead>
<tr>
<th>Calculation</th>
<th>#Mult.</th>
<th>#Add.</th>
<th>#Mult.</th>
<th>#Add.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(m = 3)</td>
<td></td>
<td>(m = 3)</td>
</tr>
<tr>
<td>(a_0)</td>
<td>86</td>
<td>27m + 71</td>
<td>86</td>
<td>152</td>
</tr>
</tbody>
</table>

computations required for \(a_0\) for the same number of chains. Thus, the computational complexity of the complete simulation algorithm is \(O(mN)\) for a given value of \(n_c \leq 6\). Note that if the \(O(N^3)\) Unit Force Method were used to compute \(\Omega\) and \(\Lambda^{-1}\) for each chain instead of the \(O(N)\) Force Propagation Method, then the computational complexity of the algorithm would be \(O(mN^3)\). All other computational requirements would remain the same. This \(O(mN^3)\) form of the simulation algorithm is more efficient for \(N < 21\), but its increased computational order may not always be desirable.

### 6.7.2 Parallel Implementation of the Simulation Algorithm

The total computational complexity discussed in the previous section only considered the execution of the simulation algorithm on a single processor. In order to speed up the simulation, parallel processing may be investigated.

If a single processor is used for the entire system of \(m\) chains, the computational complexity of the simulation algorithm is \(O(mN)\) for a given \(n_c \leq 6\). Given \(a_0\), all computations for each chain may be carried out independently. Thus, if \(m\) processors are available, the computational tasks associated with each chain may be performed in parallel, and the computational complexity of the operations required for the \(m\) chains may be reduced to \(O(N)\). Of course, the computations required to compute \(a_0\) must also be considered. These operations may also be
implemented in parallel on the \( m \) available processors. Equation (6.38) requires \( O(m) \) spatial additions to compute \( a_0 \). On \( m \) parallel processors, this task may be carried out in \( O(\log_2 m) \) operations by using the recursive doubling approach \[29\]. Thus, on \( m \) parallel processors, the computational complexity of the entire dynamic simulation algorithm may be reduced to \( O(N) + O(\log_2 m) \).

6.8 Conclusions and Summary

In this chapter, a general and efficient dynamic simulation algorithm for simple closed-chain mechanisms was derived. The algorithm is applicable to both Type 0 and Type 1 mechanisms. Both types of mechanisms are modelled in a convenient and general manner through the use of the general joint concept. The operational space inertia matrix of each chain is used to project the dynamic properties of the chain to its tip where it is coupled to the reference member. By combining the operational space inertia of each chain with the model of the general joint at each chain tip, a solution may be found for the spatial acceleration of the reference member and the spatial force vector exerted on it by each chain. Once the force vectors are completely defined, the system is effectively decoupled, and the joint accelerations for each chain may be computed separately.

The computational complexity of the new simulation algorithm is \( O(mN) \) for a given \( n_c \leq 6 \) when implemented on a single processor. The linear dependence on \( N \) is a significant improvement over previous simulation algorithms such as that presented in \[13\]. The computational complexity of the new algorithm may be further reduced to \( O(N) + O(\log_2 m) \) if it is implemented on \( m \) processors in parallel.

This chapter concludes the development of efficient dynamics algorithms for multiple chain robotic systems. Although the algorithms developed in this dis-
ertation were specifically designed for and applied to the simulation of simple closed-chain mechanisms, many of the results are also applicable for more general systems. The following chapter will summarize the contributions of this dissertation and discuss possible extensions to this research.
CHAPTER VII
SUMMARY AND CONCLUSIONS

7.1 Summary

In this dissertation, the dynamic behavior of multiple-chain robotic systems is investigated. Particular emphasis is placed on the class of multiple-chain robotic systems known as simple closed-chain mechanisms which are characterized by a group of actuated chains supporting a single common reference member. Simple closed-chain mechanisms include multiple manipulators with a common load, dexterous hands manipulating an object, and multilegged vehicles. The primary goal of this study in dynamics is the development of a general and efficient algorithm for the dynamic simulation of these complex mechanisms. This goal is successfully accomplished, along with the derivation of efficient algorithms for the computation of important inertial quantities and the dynamic simulation of single closed chains.

Prior to the development of simulation algorithms for closed-chain mechanisms, the efficient computation of the joint space inertia matrix for a single chain is considered. The joint space inertia matrix is an important part of many dynamics algorithms, both for simulation and control. A general joint model is used here which allows general multiple-degree-of-freedom joints to connect the links of the chain if desired. Four alternate methods are derived for computing the joint space inertia matrix, including the most efficient serial algorithm known at this time. Three of the four algorithms presented also include the simultaneous calculation
of the robot Jacobian matrix at no extra computational cost. Efficient transformations for spatial vectors and matrices are used to reduce the computational requirements of these algorithms.

The derivation of efficient algorithms for basic dynamics computations continues with an examination of the operational space inertia matrix. The operational space inertia matrix is a primary component of the operational space formulation developed for motion and force control [21]. This matrix is also useful in the decoupling of multiple closed-chain robotic systems. Four alternate methods are derived for the efficient computation of the operational space inertia matrix, including two linear recursive methods. Once again, a general joint model is used in all cases. The linear recursive algorithms derived here represent a significant reduction in computational complexity compared to previously existing algorithms.

Next, the results described above are utilized in the development of an efficient serial algorithm for the dynamic simulation of a single closed chain. The general joint model is used to define two possible classes of contacts between the end effector and other rigid bodies. The operational space inertia matrix of the chain is the key to solving for the unknown contact forces exerted by the chain on the environment. The determination of these forces leads to a solution for the joint accelerations of the closed chain. By using a linear recursive algorithm to compute the operational space inertia matrix, an efficient $O(N)$ algorithm is developed for the dynamic simulation of a single closed chain with $N$ degrees of freedom. It is believed that this algorithm may be implemented in near real time with state-of-the-art computing hardware.

Finally, the simulation algorithm for a single closed chain is extended to the case of simple closed-chain mechanisms. The operational space inertias of all the chains are combined with the spatial inertia of the common body to form the effec-
tive operational space inertia of the entire system as seen by the common member. This effective inertia is used to solve for the spatial acceleration of the reference member and the spatial forces exerted on it by each chain. Once these spatial forces are known, the system is decoupled, and the closed-chain joint accelerations for each chain may be computed independently using open-chain algorithms. On a single processor, this algorithm may be implemented in $O(mN)$ operations for a system of $m$ chains, each with $N$ degrees of freedom. On $m$ processors, this algorithm may be implemented in $O(N)$ operations. The development of an $O(N)$ algorithm for this simulation problem represents a significant contribution towards the real-time simulation of general multiple-chain robotic systems.

7.2 Future Work

This dissertation presents the development of efficient serial algorithms for the implementation of important dynamics computations for multiple-chain robotic systems. In particular, efficient algorithms for computing the joint space and operational space inertia matrices are presented, and efficient dynamic simulation algorithms are derived for single closed chain and simple closed-chain mechanisms. Although these results are significant, considerable work still remains.

The algorithms for the joint space and operational space inertia matrices developed here are designed for serial-link chains with no internal closed loops. Because many manipulators have a more complex structure, these algorithms should be extended to include general configurations of links. The properties of the operational space inertia matrix for manipulators with reduced numbers of degrees of freedom ($N < 6$) should also be investigated further. Parallel forms of the algorithms for $H$ and $A$ should also be pursued, with the linear recursive algorithms derived for the operational space inertia matrix receiving first consideration.
As discussed briefly in Chapter 6, some inherent parallelism exists in the dynamic simulation algorithm developed for simple closed-chain mechanisms. This may be attributed to the structural parallelism in the mechanism itself. Parallel algorithms for these mechanisms may also be formulated.

Other research issues associated with this work which may be investigated in the future include the following:

1. The inclusion of compliance and flexible structure models in the formulation of dynamic simulation algorithms;

2. The investigation of robust integration techniques for the next state computation;

3. The generalization of dynamic simulation algorithms to consider time-varying topologies;

4. The programming of specific examples to evaluate algorithm performance;

5. Investigation of the stability of numerical solutions when approaching singular positions;

6. The realization and application of super real-time dynamic simulation for simple closed-chain mechanisms; and

7. The derivation of dynamic simulation algorithms for more complex multiple-chain robotic systems.

In conclusion, it is believed that the efficient algorithms developed in this dissertation represent a significant contribution towards the real-time simulation of general multiple-chain robotic systems. In addition, many of the basic computational results presented here are believed to be valuable for real-time control
applications as well. It is hoped that the interpretation of the algorithms, as well as the equations themselves, may provide some physical insight into the general dynamic behavior of complex robotic systems.
REFERENCES


172


