ROBOT DYNAMICS ALGORITHMS

Roy Featherstone

Ph. D. Thesis

University of Edinburgh

1984

Abstract

In this dissertation I introduce a new notation for representing rigid-body dynamics, and use it to describe a number of methods for calculating robot dynamics efficiently.

The notation (called spatial notation) is based on the use of 6-dimensional vectors (called spatial vectors) which combine the linear and angular aspects of rigid-body dynamics. Spatial vectors are similar to quantities called screws and motors. The use of spatial notation allows a more concise treatment of problems in rigid-body dynamics than is possible by the conventional vector approach by reducing the number of quantities required to describe a system and the number and size of equations relating those quantities.

I consider both forward and inverse robot dynamics, though I am concerned mainly with forward dynamics. Three basic algorithms are described: the recursive Newton-Euler method for inverse dynamics, and the composite-rigid-body and articulated-body methods for forward dynamics. The articulated-body method is new, and is based on the use of quantities called articulated-body inertias which relate the force applied to a member of a linkage to the acceleration induced in that member, taking into account the effect of the rest of the linkage. Once the basic algorithms have been introduced, I consider some aspects of their implementation on a computer, then I describe various extensions of the basic algorithms to cater for generalisations of the robot's structure, including multiple-degree-of-freedom joints and branched kinematic chains. Finally, I consider the problem of simulating contact and impact between the robot and its environment.

Acknowledgements

I wish to acknowledge the help of Robin Popplestone, my supervisor, who originally suggested robot dynamics as a topic for research. He offered many 'high level' comments and much assistance over the years, and has kept a keen interest in the ideas and concepts I was developing. I also owe a great debt to John Hallam and Stephen Cameron, who were always willing to offer helpful comments and criticisms, and who selflessly undertook to proof-read this thesis very thoroughly. They spotted many embarrassing errors. Finally, I wish to acknowledge the funding of the S.E.R.C. which enabled me to carry out this work in the first place.

I hereby declare that this thesis was composed by myself and that it describes my own original work.

Contents

1.	Introduction	1
1.1	Scope and Contents	1
1.2	The Robot Dynamics Problem	2
1.3	Spatial Notation	5
1.4	Readers' Guide	6
2.	Spatial Kinematics	8
2.1	Introduction	8
2.2	Spatial Velocity	9
2.3	Line Vectors and Free Vectors	11
2.4	Screws and Motors	15
2.5	Spatial Vectors	19
2.6	Rigid-Body Coordinate Transformations	22
2.7	Differentiation in Moving Coordinates	25
2.8	Spatial Acceleration	29
3.	Spatial Dynamics	32
3.1	Introduction	32
3.2	Spatial Rigid-Body Momentum	33
3.3	Spatial Rigid-Body Inertia	34
3.4	The Equations of Motion	38
3.5	Inverse Inertia	39
3.6	Articulated-Body Inertias	40
3.7	Other Representations for Inertia	45
3.8	The Spatial Scalar Product	47
3.9	The Spatial Transpose Operator	49
3.10	Spatial Vector Analysis	51
3.11	Summary of Spatial Algebra	58
4.	Inverse Dynamics - The Recursive Newton-Euler Method	61
4.1	Introduction	61
4.2	Recurrence Relations	63
4.3	The Robot System Model	67
4.4	The Recursive Newton-Euler Formulation	70

4.5	Summary	74
5.	Forward Dynamics - The Composite-Rigid-Body Method	75
5.1	Introduction	75
5.2	Using an Inverse Dynamics Algorithm to Calculate Forward Dynamics	77
5.3	The Composite-Rigid-Body Method	80
6.	Forward Dynamics - The Articulated-Body Method	84
6.1	Introduction	84
6.2	The Articulated-Body Method in Outline	87
6.3	The Calculation of Articulated-Body Inertias	89
6.4	The Articulated-Body Dynamics Algorithm	94
7.	Coordinate Systems and Efficiency	102
7.1	Introduction	102
7.2	The Placement of Link Coordinates	102
7.3	Efficient Spatial Arithmetic	105
7.4	The Computational Requirements of the Algorithms	112
8.	Branched Kinematic Chains, Multiple-Degree-of-Freedom Joints and Moving Bases	118
8.1	Introduction	118
8.2	Branched Kinematic Chains	121
8.3	Screw Joints	125
8.4	Multiple-Degree-of-Freedom Joints	127
8.5	Robots with Floating Bases	133
9.	Contact, Impact, Kinematic Loops and Friction	140
9.1	Introduction	140
9.2	Contact	142
9.3	The Kinematics of Contact	152
9.4	Impact	154
9.5	Joint End Stops	157
9.6	Coulombic Friction at the Joints	159
9.7	The Articulated-Body Method for Kinematic Loops	160
	Bibliography	167

Chapter 1

Introduction

1.1 Scope and Contents

In this dissertation I am concerned with mathematical formulations of robot dynamics which produce efficient algorithms when implemented on a computer. The mathematical formulations are derived with the aid of spatial notation - a new notation which greatly facilitates the analysis of robot dynamics by reducing the size and number of equations involved.

The dissertation is organised into two parts, with part one dealing with the spatial notation and its underlying spatial vector algebra, and part two dealing with robot dynamics algorithms. Three basic algorithms are described: the recursive Newton-Euler method for inverse dynamics, and the composite-rigid-body and articulated-body methods for forward dynamics. The first two are my own versions of existing algorithms, but the articulated-body method is new. Various extensions to these algorithms are also described.

Forward and inverse dynamics are the two main problems of robot dynamics. Forward dynamics is the problem of finding the instantaneous acceleration of a robot given the applied forces, and finds application in simulation; whereas inverse dynamics is the problem of

finding the joint forces required to produce a given acceleration, and finds application in control. The material presented in this dissertation was motivated by the work I have been doing on robot dynamics simulation, and therefore leans towards the simulation side of robot dynamics. I have developed a robot dynamics simulation program, described in [Featherstone 1982; Featherstone 1983b], which has been used to test all of the basic dynamics algorithms and many of the extensions.

Although the dynamics algorithms presented here are designed to be of practical use, no specific applications are considered. In particular, I do not describe any kind of robot control system using inverse dynamics, nor any robot dynamics simulator using forward dynamics. References on these applications are cited in chapters 4 and 5.

1.2 The Robot Dynamics Problem

In principle, solving the robot dynamics problem presents no difficulty - the real problem is the practical one of finding formulations of robot dynamics that lead to efficient computation algorithms. Robot dynamics warrants special attention over the more general area of mechanism dynamics due to the relatively simple nature of robot kinematics, the large number of degrees of freedom and the special needs of robot control systems.

In its simplest form the equation of motion for a system of rigid bodies can be expressed as

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{\tau}) ,$$

where $\underline{x}$ is the system state vector (generalised positions and velocities of the members) and $\underline{\tau}$ the input force vector. To solve this differential equation requires procedures for evaluating $\underline{f}$ and for performing numerical integration. For complex rigid-body systems the main problem is the evaluation of $\underline{f}$.

With simple problems the most appropriate way is to evaluate the symbolic equations of motion with the appropriate numerical parameters, but this is impractical for larger problems since the size of the symbolic equations grows very rapidly with increasing problem size. A better approach for larger systems is to evaluate $\underline{f}$ directly from the system model.¹ I call an algorithm for doing this a forward dynamics algorithm, and one that finds $\underline{\tau}$ in terms of $\underline{x}$ and $\dot{\underline{x}}$ an inverse dynamics algorithm.

The execution time for different algorithms varies enormously. To see why, one must realise that symbolic execution of a forward dynamics algorithm produces the symbolic equations of motion, and so its numeric execution is equivalent to a structured evaluation of the equations of motion. The equations are very lengthy and contain a great many terms differing only slightly one from another. A naive evaluation performs a lot of duplicated calculation, which can be avoided by a structured evaluation which computes common sub-expressions as intermediate results for use in subsequent stages of the calculation. The efficiency of a forward dynamics algorithm depends on how much duplication it avoids.

1. The system model is a description of a rigid-body system in terms of its elements (rigid bodies, joints, springs, dampers, etc.) and how those elements are connected.

The key to efficient calculation of robot dynamics is to find a set of common sub-expressions which is effective in eliminating most of the repetition inherent in the equations of motion, or better, to design a forward dynamics algorithm such that its intermediate results are the desired common sub-expressions. The same arguments apply also to inverse dynamics.

A technique that has been found to be particularly effective in producing efficient dynamics algorithms is that of formulating the dynamics recursively in terms of recurrence relations [Brady et al. 1982]. The task is broken down into a number of partially ordered stages, each of which is solved by the application of a formula to each link in turn. Where appropriate, the formula defines the quantity of interest for the link in question in terms of the values of that quantity for one or more of the link's immediate neighbours, in which case it is known as a recurrence relation. The practice of using previously calculated results in subsequent applications of recurrence relations ensures that the intermediate results are frequently occurring common sub-expressions of the overall dynamics equations (but not necessarily the optimum set).

All the dynamics algorithms described in this dissertation are formulated in this manner. A practical example of how recurrence relations work is given in chapter 4.

1.3 Spatial Notation

A particle has three degrees of freedom, so its velocity, acceleration, etc. can each be described by a 3-dimensional vector, and it has one vector equation of motion. A rigid body has six degrees of freedom, yet rigid-body dynamics is expressed using 3-dimensional vectors. So the velocity, acceleration, etc. of a rigid body are each represented by a pair of vectors (one linear and one angular) and there are two equations of motion. The reason for this is that linear and angular quantities are considered to be physically different quantities; but this does not preclude their amalgamation into 6-dimensional vectors as a matter of algebraic and notational convenience.

The spatial notation is based on the use of 6-dimensional vectors called spatial vectors. The use of spatial notation results in a considerable reduction in the number of quantities and in the number and size of equations required to express, solve, or express the solution to a problem in spatial rigid body dynamics. This is especially so when handling inertias. I use the spatial notation to express all of the robot dynamics algorithms presented in this dissertation, enabling me to present them concisely and (I hope) clearly. The underlying spatial vector algebra is described in chapters 2 and 3. Spatial vectors are very similar to mathematical quantities known as screws and motors, which are described briefly in chapter 2 together with comments on their unsuitability for describing robot dynamics.

1.4 Readers' Guide

This dissertation is organised into two parts. Part one describes the spatial vector algebra and comprises chapters 2 and 3, while part two describes the robot dynamics algorithms and comprises chapters 4 to 9.

In part one, chapter 2 introduces the basic concept of a spatial vector and deals with the kinematic aspects of the spatial vector algebra, while chapter 3 is concerned with the representation of inertia, the equations of motion and spatial vector analysis. Part one may be read in isolation by anyone interested only in the spatial notation, though part two does offer many examples of its use and some hints on efficient computer implementation of spatial formulae. Readers familiar with the use of screws or motors in kinematics will probably find most of the material in chapter 2 familiar and most of the material in chapter 3 unfamiliar.

In part two, all of the algorithms are described using spatial notation, so a proper understanding of part two can only be obtained by reading part one first. However it is possible to get the gist of part two by taking the spatial formulae on trust and appealing to the superficial similarity between spatial formulae and their counterparts in conventional vector algebra.

Chapters 4 to 6 describe three robot dynamics algorithms in their simplest form. Chapter 4 describes the recursive Newton-Euler method for inverse dynamics, chapter 5 the composite-rigid-body method for forward dynamics, and chapter 6 the articulated-body method for forward dynamics. Chapter 4 also defines the system model used in the

descriptions of the algorithms, and gives an example of how recurrence relations work. Only the articulated-body algorithm is new. The concept of an articulated-body inertia is introduced in chapter 3, but the important points are summarised in chapter 6.

Chapter 7 is concerned with computational considerations such as the choice of link coordinate systems, the computer representation of spatial quantities, and a comparison of the computational requirements of the basic dynamics algorithms. This chapter will be of interest to anyone wishing to implement the algorithms described here.

Finally, chapters 8 and 9 deal with a number of extensions to the basic algorithms. Chapter 8 covers generalisations of the robot structure, such as branched kinematic chains and multiple-degree-of-freedom joints, while chapter 9 covers various dynamic effects, like contact and impact between the robot and other objects. These two chapters are organised into fairly self-contained topics, and are suitable for selective reading.

Chapter 2

Spatial Kinematics

2.1 Introduction

In this chapter I introduce the basic concept of a spatial vector, and show how it relates to line vectors, free vectors, screws and motors. I cover the representation of velocities, forces, line vectors, free vectors, and revolute, prismatic and screw joint axes as spatial vectors. I then go on to describe spatial coordinate systems, the representation of rigid-body coordinate transformations, and the differentiation of spatial vectors in moving coordinate systems. The representation of inertia and the equations of motion for a rigid body will be dealt with in chapter 3.

For a proper understanding of spatial algebra, the reader must be prepared to accept that quantities of a physically different nature, like linear and angular velocity, can be combined as components of a single entity; and it is important to regard spatial quantities as objects in their own right rather than as conglomerations of more familiar objects.

2.2 Spatial Velocity

A rigid body has six degrees of freedom - three translational and three rotational. So the location, displacement, or velocity of a rigid body can be described completely by six numbers - three describing the translational aspect and three the rotational aspect. As an example, I now show how the total velocity of a rigid body can be expressed as a 6-dimensional vector.

The instantaneous velocity of a rigid body may be described by the linear velocity $\underline{v}_p$ of some point P of the rigid body and the angular velocity $\underline{\omega}$ of the rigid body (see figure 2.1). $\underline{\omega}$ is independent of the choice of P , but $\underline{v}_p$ is not. The instantaneous velocity of the rigid body may be described completely by the pair of vectors $(\underline{\omega}, \underline{v}_p)$ given the point P .

Figure 2.1 Rigid Body Velocity

The rigid body may be considered to be rotating with angular velocity $\underline{\omega}$ about an axis passing through P whilst simultaneously translating with linear velocity $\underline{v}_P$. The choice of P is arbitrary. It need not be physically within the rigid body, so long as it has the same motion as the rigid body. Moreover, we need not choose the same point relative to the rigid body from one instant to the next.

Let us choose the point at the origin as standard, and represent the velocity of the rigid body by the pair $(\underline{\omega}, \underline{v}_O)$, where $\underline{v}_O$ is given by

$$\underline{v}_O = \underline{v}_P + \overrightarrow{OP} \times \underline{\omega} . \quad (1)$$

The body may now be considered to be rotating about an axis passing through the origin whilst simultaneously translating with linear velocity $\underline{v}_O$. The factor $\overrightarrow{OP} \times \underline{\omega}$ added to the linear velocity at P has had the effect of shifting the axis of rotation from passing through P to passing through O.

The pair $(\underline{\omega}, \underline{v}_O)$ defines the spatial velocity of the rigid body. We shall represent spatial vector quantities as 6-element column vectors and denote all spatial quantities with carets. Thus in spatial notation we say that $\hat{\underline{v}}$ is the spatial velocity of the rigid body, where $\hat{\underline{v}} = [\omega_x \ \omega_y \ \omega_z \ v_{ox} \ v_{oy} \ v_{oz}]^T$ which will normally be abbreviated to $\hat{\underline{v}} = [\underline{\omega}^T \ \underline{v}_O^T]^T$.

2.3 Line Vectors and Free Vectors

An 'ordinary' 3-dimensional vector has magnitude and direction, but a vector may also have positional properties, depending on the nature of the physical (or mathematical) entity it represents. In particular, two kinds of vector occur in rigid-body dynamics: line vectors and free vectors.

A line vector has magnitude and direction, and is constrained to lie in a definite line (i.e. a line in a particular position). Line vectors arise naturally in mechanics to represent quantities like angular velocity, where there is a definite axis of rotation, and forces acting on a rigid body, which have a definite line of action but may act at any point along that line.

A free vector has magnitude and direction only, and is used to represent quantities like linear velocity and couple which have no definite line or axis associated with them.

Figure 2.2 Definition of a line vector

A line vector may be represented as follows. Suppose a line vector has magnitude and direction given by the vector $\underline{a}$ and its line of action passes through the point A (see figure 2.2). Since the direction of the line is the same as that of the line vector, we may say that it has magnitude and a line of action. The equation of the line of action is given by

$$(\underline{r} - \overrightarrow{OA}) \times \underline{a} = \underline{0},$$

where $\underline{r}$ is the position of any point on the line. This equation is more normally stated in the form

$$\underline{r} \times \underline{a} = \underline{a}_0 \quad (2)$$

where $\underline{a}_0 = \overrightarrow{OA} \times \underline{a}$. $\underline{a}$ and $\underline{a}_0$ are known as the Plücker vectors of the line and determine it completely. $\underline{a}$ is called the resultant vector and $\underline{a}_0$ the moment vector. Although six numbers are specified, only five are independent since $\underline{a} \cdot \underline{a}_0 = 0$; and if the vectors represent just the line then one number is arbitrary since $\lambda \underline{a}$ and $\lambda \underline{a}_0$ represent the same line as $\underline{a}$ and $\underline{a}_0$, though not the same line vector.

$\underline{a}_0$ depends on the choice of origin, for if the origin is moved from O to P then the Plücker vectors become $\underline{a}$ and $\underline{a}_p$ where $\underline{a}_p = \overrightarrow{PA} \times \underline{a}$. So we have

$$\underline{a}_0 = \underline{a}_p + \overrightarrow{OP} \times \underline{a}, \quad (3)$$

which is the same shifting formula as for spatial velocity. The only difference between the line vector and the spatial velocity vector, $[\underline{\omega}^T \underline{v}_0^T]^T$, is that it is not necessary for $\underline{\omega} \cdot \underline{v}_0$ to be zero. A line vector can be expressed as a spatial vector $\hat{\underline{a}} = [\underline{a}^T \underline{a}_0^T]^T$; indeed any spatial vector where $\underline{a} \cdot \underline{a}_0 = 0$ represents a line vector.

A free vector with magnitude and direction given by the vector $\underline{b}$ is of course adequately represented by $\underline{b}$. We can put free vectors on the same algebraic footing as line vectors by expressing them in the form $\underline{\hat{b}} = [\underline{0}^T \underline{b}^T]^T$, for such a quantity obeys the shifting formula of eq. (3) and yet is independent of the choice of origin.

2.3.1 Line and Free Vector Components of Spatial Vectors

A general spatial vector is the sum of a line vector and a free vector. There are an infinite number of line vectors and free vectors that add up to the same spatial vector, so by convention we constrain the line vector's line of action to pass through the origin of the coordinate system in which the spatial vector is represented. A general spatial vector $\underline{\hat{a}} = [\underline{a}^T \underline{a}_0^T]^T$ may then be uniquely broken down into its line and free vector components, $[\underline{a}^T \underline{0}^T]^T$ and $[\underline{0}^T \underline{a}_0^T]^T$.

A general spatial vector is also equivalent to a screw, for, if we let $\underline{\hat{a}} = [\underline{a}^T \underline{a}_0^T]^T$ be a general spatial vector, we can express $\underline{a}_0$ as $\underline{a}_0 = \underline{a}' + \rho \underline{a}$ where $\underline{a} \cdot \underline{a}' = 0$ and $\rho = \frac{\underline{a} \cdot \underline{a}_0}{\underline{a} \cdot \underline{a}}$. So a general spatial vector can be expressed uniquely as the sum of a line vector and a parallel free vector: $\underline{\hat{a}} = [\underline{a}^T \underline{a}_0'^T]^T + [\underline{0}^T \rho \underline{a}^T]^T$. Such a combination is known as a screw in analogy with the motion of a nut on a screw thread - it rotates about an axis and translates parallel to that axis. The parameter ρ is known as the pitch and is an invariant of the screw.

2.3.2 The Representation of Joint Axes

A revolute (or turning) joint has a definite axis about which rotation is possible. This axis may be represented by a line vector. Since axes do not have magnitude, a line vector of unit magnitude is used by convention to represent them.

A prismatic (or sliding) joint is characterised only by its direction, having no definite axis, so a unit free vector is the appropriate representation.

A screw joint combines rotation about a definite axis with translation along it, so it can be represented by a line vector and a parallel free vector. The pitch of the screw determines the ratio of the magnitudes of the line and free vectors, and a 'unit' screw axis is defined as one having a unit line vector. Further discussion of screw joints and the representation of joints allowing more than one degree of motion freedom is deferred until chapter 8.

Once a joint axis is represented as a spatial vector, the type of motion allowed by the joint becomes irrelevant. There is no need to have separate equations to deal with each different joint type.

Example 1

Let $\hat{\underline{s}} = [\underline{s}^T \underline{s}_0^T]^T$ be the axis of a joint connecting bodies b_1 and b_2 , and let $\dot{q}$ be the (scalar) joint velocity indicating the relative velocity of b_2 with respect to b_1 , then the relative spatial velocity of b_2 with respect to b_1 is $\hat{\underline{s}}\dot{q} = [\underline{s}^T\dot{q} \underline{s}_0^T\dot{q}]^T$. This is true regardless of what type of joint is represented.

2.3.3 The Representation of Forces

A force acting on a point in a rigid body has the same effect as a force of equal magnitude and direction acting on any other point in a line parallel to the force and passing through the first point. A force acting on a rigid body may therefore be represented as a line vector. A spatial force with magnitude and direction given by $\underline{f}$ and acting in the line passing through P is given by $\hat{\underline{f}} = [\underline{f}^T (\overrightarrow{OP} \times \underline{f})^T]^T$. The moment vector is a couple. The rules for shifting the line of action of a force by adding a couple are the same as the corresponding rules for shifting line vectors. A pure couple has no line of action, and may therefore be represented as a free vector. A general spatial force can be represented as the sum of a force acting through the origin and a couple. Any system of forces acting on a rigid body can be replaced by a single, statically equivalent, spatial force.

2.4 Screws and Motors

The spatial vector quantities used in this dissertation bear a close resemblance to mathematical quantities called screws and motors. In the following paragraphs I define these quantities and compare them with spatial vectors. Whilst it is perfectly possible to perform the dynamic analysis of rigid-body systems using screws or motors they do put the user to a certain amount of relative inconvenience, both conceptual and notational. This is particularly so when dealing with inertias.

2.4.1 Screws

A screw, as defined by Ball [1900],¹ is a directed line with which a linear magnitude, called the pitch, is associated. A screw is essentially a geometrical concept. Any finite or infinitesimal displacement of a rigid body may be represented in terms of a twist of a certain magnitude about a screw - five numbers determine the screw and one the twist magnitude, making a total of six. Similarly, any system of forces acting on a rigid body may be replaced by a single wrench on a screw (a force and a parallel couple). Ball treats twists and wrenches as magnitudes associated with screws, but some people allow screws to have intrinsic magnitudes.

Screws are used extensively in kinematics [Hunt 1978; Bottema and Roth 1979; Woo and Freudenstein 1970; Yuan and Freudenstein 1971; Yuan et al. 1971; Dimentberg 1965; Lipkin and Duffy 1982] but less so in dynamics [Woo and Freudenstein 1971] although they were originally devised by Ball to study the dynamics of a rigid body. Ball used principal screws of inertia, in analogy to principal axes of rotational inertia, to represent rigid body inertias. This method is inconvenient for reasons described in chapter 3, and to my knowledge has not been used by anyone other than Ball. Screws are represented in a variety of ways, including the use of real and dual numbers.

1. This is not the earliest definition of a screw.

2.4.2 Motors

The term motor was originally introduced by Clifford [1882] to mean a magnitude associated with a screw. He said that a motor consists of a rotor (line vector) part and a vector (free vector) part. Nowadays the term is generally used to mean a vector of three dual numbers - i.e. a dual vector. A dual number (also introduced by Clifford) is one of the form $a + \epsilon b$, where a and b are real numbers and ϵ is an algebraic unit having the formal property that $\epsilon^2 = 0$.¹ The motor representation of a spatial vector $\begin{bmatrix} \underline{a}^T & \underline{a}_0^T \end{bmatrix}^T$ is $\underline{a} + \epsilon \underline{a}_0$.

The usefulness of dual numbers derives from the principle of transference [Dimentberg 1965; Rooney 1975] which allows one to obtain valid motor formulae from the corresponding vector formulae by the process of dualisation - the replacement of real numbers representing distances, angles, etc. in the formula with dual numbers. Thus the motor algebra may readily be obtained from the vector algebra, and since there is a 1:1 correspondence between motors and screws with magnitude, they offer a very convenient way to approach the screw algebra. This is the approach taken by Dimentberg, who speaks of reducing (i.e. representing) a screw to a point P as representing it in the form $\underline{a} + \epsilon \underline{a}_P$. However, the principle of transference only applies if the functions and operators are analytic (in the same sense as for functions of a complex variable), and the only representation of inertia available in dual numbers is the inertia binor (see chapter 3) which is not analytic, so the principle of transference does not apply to dynamic equations.

1. The symbol ω is often used in place of ϵ .

The motor and dual-number algebras are described by Dimentberg [1965] and Brand [1953] and to a lesser extent by Bottema and Roth [1979] (who call motors dual vectors) and Hunt [1978]. Dual numbers are also used extensively in kinematics [Yuan and Freudenstein 1971; Yuan et al. 1971; Yang and Freudenstein 1964; Duffy 1980] but much less so in dynamics [Yang 1971].

There is one important difference between the use of dual numbers and real numbers to represent screw algebra, and that is that the scalar product of two dual vectors produces a dual number result whereas with real numbers you get a real number result. The use of dual numbers also has serious repercussions on the representation of inertia, as mentioned above.

Mises [1924a; 1924b] uses 6-element real vectors to represent motors, and develops a real number motor algebra, including motor dyadics, which is very similar to spatial vector algebra. The main difference between the two is that spatial vector algebra uses the operators of matrix algebra plus the spatial transpose operator (defined in chapter 3), whereas Mises uses a number of specially defined operators.

In summary then, whilst screws and motors are very effective tools for kinematics, the use of screws in dynamic analysis is hampered by their geometric definition, which usually requires the magnitude to be kept separate from the screw, and the use of motors is hampered by the properties of the dual-number algebra.

2.5 Spatial Vectors

A spatial vector is a member of a 6-dimensional vector space over the real numbers. To establish a correspondence between spatial vectors and the vectors of the 3-dimensional Euclidean space, certain spatial vectors are identified as line vectors and certain others as free vectors. To make the correspondence explicit, a Cartesian coordinate frame is placed in the 3-dimensional space and a standard basis is defined in the spatial vector space. This basis, denoted $\{\hat{e}_i\}$, consists of the six vectors $\hat{e}_1 \dots \hat{e}_6$, where $\hat{e}_1 \dots \hat{e}_3$ are unit line vectors along the x, y and z axes of the Cartesian coordinate frame and $\hat{e}_4 \dots \hat{e}_6$ are unit free vectors parallel to the x, y and z axes of that frame. There is a unique standard basis for each Cartesian coordinate frame in 3-dimensional space. Any spatial vector $\hat{a}$ may be expressed as

$$\hat{a} = \sum_{i=1}^6 a_i \hat{e}_i, \quad (4)$$

and may be represented as the 6×1 matrix of components $[a_1 \dots a_6]^T$.

To clarify the distinction between the spatial vector itself and its matrix representation, I will occasionally denote the matrix representation with a tilde, thus:

$$\tilde{a} = [a_1 \dots a_6]^T. \quad (5)$$

$\hat{a}$ is a primitive entity, but $\tilde{a}$ is a realisation of $\hat{a}$ in a particular coordinate system. The expression $\hat{a} + \hat{b}$ is independent of any coordinate system, whereas $\tilde{a} + \tilde{b}$ is only meaningful if $\tilde{a}$ and $\tilde{b}$ are

realisations in the same coordinate system. Most of the time I will ignore this distinction and use $\hat{a}$ to denote both the vector and the representation.

2.5.1 Operations on Spatial Vectors

For an expression involving spatial quantities to make physical sense there must be a physical interpretation for the mathematical operations as well as for the spatial quantities. The interpretation of a mathematical operation depends on what operation it is and what its operands are. For the most part the restrictions are pretty obvious and much as you would expect, for example it does not make sense to add a velocity to a force, but there are a few cases where interpretations valid for ordinary vectors do not carry over to spatial vectors. (This is particularly so with the scalar product defined in chapter 3.) This topic is dealt with rigorously by Woo and Freudenstein [1970] (and more intuitively by Ball [1900]), but here I merely list the permissible interpretations of operations on those spatial quantities we have already met.

- o Addition is defined between spatial forces acting on the same rigid body, the sum being the resultant force.
- o Addition is defined between spatial velocities representing the velocity of A with respect to B and B with respect to C, the sum being the velocity of A with respect to C.
- o Multiplication of a spatial vector representing a joint axis by a scalar representing the joint velocity (i.e. the first time derivative of the joint variable) gives a spatial vector

representing the relative velocity between the two links connected by the joint.

Example 2

A robot has $n+1$ links numbered $0..n$ and n joints numbered $1..n$. Link 0 is stationary, and link $i-1$ is connected to link i by joint i . The axis of joint i is $\hat{s}_i$ and its scalar velocity (i.e. joint rate) is $\dot{q}_i$ measured from link $i-1$ to link i . The relative velocity of link i with respect to link $i-1$ is then $\hat{s}_i \dot{q}_i$, and the absolute velocity of link i is

$$\hat{v}_i = \sum_{j=1}^i \hat{s}_j \dot{q}_j .$$

In particular, the velocity of link n , the end effector, is

$$\hat{v}_n = \sum_{j=1}^n \hat{s}_j \dot{q}_j . \text{ This can be expressed in matrix form as}$$

$$\hat{v}_n = \begin{bmatrix} \hat{s}_1 & . & . & . & \hat{s}_n \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = \hat{J} \dot{q} .$$

The $6 \times n$ matrix, $\hat{J}$, composed of the axis vectors is the Jacobian of the transformation between the manipulator's joint space and the Cartesian velocity of the end effector expressed in the same coordinates as the $\hat{s}_i$.

2.6 Rigid-Body Coordinate Transformations

A rigid-body transformation, or Euclidean displacement, is one which leaves the distance between any two points invariant. A rigid-body transformation in three dimensions is composed of a rotation (three degrees of freedom) and a translation (also three degrees of freedom). The image of a Cartesian coordinate frame under a rigid-body transformation is another Cartesian coordinate frame.

Consider two standard bases, $\{\hat{\underline{e}}_i\}$ and $\{\hat{\underline{f}}_i\}$, where $\{\hat{\underline{e}}_i\}$ corresponds to a Cartesian coordinate frame at O and $\{\hat{\underline{f}}_i\}$ to one at P. The spatial transformation $\hat{Y}: \hat{\underline{e}}_i \rightarrow \hat{\underline{f}}_i$ corresponds to the rigid-body transformation that maps O coordinates to P coordinates, and is therefore a linear mapping of spatial vector space. $\hat{Y}$ will be referred to as a spatial rigid-body transformation. Using tildes to denote matrix representations, suppose that a vector $\hat{\underline{a}}$ is represented as $\tilde{\underline{a}}_O = [a_{O1} \dots a_{O6}]^T$ in $\{\hat{\underline{e}}_i\}$ and as $\tilde{\underline{a}}_P = [a_{P1} \dots a_{P6}]^T$ in $\{\hat{\underline{f}}_i\}$, so

$$\hat{\underline{a}} = \sum a_{Oi} \hat{\underline{e}}_i = \sum a_{Pi} \hat{\underline{f}}_i. \quad (6)$$

The mapping $\tilde{X}: \tilde{\underline{a}}_O \rightarrow \tilde{\underline{a}}_P$ is a spatial rigid-body coordinate transformation. $\tilde{X}$ is a passive transformation in that it changes the representation of a spatial vector, as opposed to an active transformation, like $\hat{Y}$, which maps a vector to a different vector (its image). $\tilde{X}$ is linear, being the coordinate transformation induced by $\hat{Y}$, so it may be expressed as a 6x6 matrix. The rest of this section is devoted to finding $\tilde{X}$ in terms of the rigid-body transformation that maps O coordinates to P coordinates.

To translate the origin of coordinates from O to P we use the shifting formula of eq. (3) (or eq. (1)), which gives

$$\hat{\underline{a}}_p = \begin{bmatrix} \underline{a} \\ \underline{a}_o + \overrightarrow{PO} \times \underline{a} \end{bmatrix}, \quad \text{where } \hat{\underline{a}}_o = \begin{bmatrix} \underline{a} \\ \underline{a}_o \end{bmatrix}. \quad (7)$$

This can be rearranged to express $\hat{\underline{a}}_p$ as the product of a 6×6 matrix with $\hat{\underline{a}}_o$ thus:

$$\hat{\underline{a}}_p = \begin{bmatrix} 1 & 0 \\ \overrightarrow{PO} \times & 1 \end{bmatrix} \hat{\underline{a}}_o. \quad (8)$$

Here 0 is the 3×3 zero matrix, 1 is the 3×3 identity matrix, and $\overrightarrow{PO} \times$ is a 3×3 anti-symmetric matrix obtained by applying the cross operator, $\times$, to $\overrightarrow{PO}$. The cross operator is defined as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \quad (9)$$

The product of the matrix $\underline{a} \times$ with the vector $\underline{b}$ is equal to the vector cross product $\underline{a} \times \underline{b}$ (which I will write as $\underline{a} \times \underline{b}$ when I wish to emphasise the identity of $\underline{a} \times$ as an object in its own right). The cross operator is a linear operator with the following properties:

$$\lambda (\underline{a} \times) = (\lambda \underline{a}) \times, \quad (\text{where } \lambda \text{ is a scalar}) \quad (10)$$

$$\underline{a} \times + \underline{b} \times = (\underline{a} + \underline{b}) \times, \quad (11)$$

$$(\underline{a} \times)^T = -\underline{a} \times, \quad (12)$$

$$(\underline{a} \times) \underline{b} = -(\underline{b} \times) \underline{a}, \quad (13)$$

$$(E \underline{a}) \times = E \underline{a} \times E^{-1}, \quad (\text{where } E \text{ is a } 3 \times 3 \text{ orthog. matrix}) \quad (14)$$

$$(\underline{a} \times \underline{b}) \times = \underline{a} \times \underline{b} \times - \underline{b} \times \underline{a} \times. \quad (15)$$

Unless indicated otherwise, expressions such as $\underline{a} \times \underline{b} \times \underline{c}$ are to be interpreted as $(\underline{a} \times)(\underline{b} \times)\underline{c}$, which is equivalent to the vector triple product $\underline{a} \times (\underline{b} \times \underline{c})$ but not to $(\underline{a} \times \underline{b}) \times \underline{c}$.

To effect a coordinate rotation, say from O coordinates to O' , consider the spatial quantity $\hat{\underline{a}}$ to be the sum of a line vector through the origin and a free vector, so $\hat{\underline{a}} = \begin{bmatrix} \underline{a} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \underline{a}_0 \end{bmatrix}$. The representation of $\hat{\underline{a}}$ in the rotated coordinate system is the sum of the rotated representations of the line vector and the free vector.

Let E be the 3×3 orthogonal rotation matrix corresponding to the coordinate rotation, then the representation of the line vector, which still acts through the origin, in the rotated coordinate system is $[(E\underline{a})^T \ 0^T]^T$, and the representation of the free vector is $[0^T \ (E\underline{a}_0)^T]^T$. So if $\hat{\underline{a}}_0$ is a spatial vector represented in O and $\hat{\underline{a}}_0'$ is the same vector in O' then the coordinate transformation is

$$\hat{\underline{a}}_0' = \begin{bmatrix} E & 0 \\ 0 & E \end{bmatrix} \hat{\underline{a}}_0. \quad (16)$$

Spatial coordinate transformations will be given leading and following subscripts indicating the destination and source coordinate systems respectively. Thus ${}_a\hat{X}_b$ is a spatial transformation which operates on a vector represented in coordinates b and produces a representation of the same vector in coordinates a according to the formula $\hat{\underline{v}}_a = {}_a\hat{X}_b \hat{\underline{v}}_b$.

Spatial coordinate transformations may be combined by matrix multiplication, thus

$${}_a\hat{X}_b {}_b\hat{X}_c = {}_a\hat{X}_c. \quad (17)$$

The form of a general spatial coordinate transformation, which involves both rotation and translation, may be obtained by multiplying the transformation matrices of eqs. (8 and 16). The spatial transformation corresponding to a shift of origin from O to P

followed by a rotation about P is

$${}_P\hat{X}_O = \begin{bmatrix} E & 0 \\ 0 & E \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \underline{r}^{\times T} & 1 \end{bmatrix} = \begin{bmatrix} E & 0 \\ E \underline{r}^{\times T} & E \end{bmatrix}, \quad (18)$$

where $\underline{r} = \overrightarrow{OP}$. The inverse transformation is

$${}_O\hat{X}_P = \begin{bmatrix} 1 & 0 \\ \underline{r}^{\times} & 1 \end{bmatrix} \begin{bmatrix} E^{-1} & 0 \\ 0 & E^{-1} \end{bmatrix} = \begin{bmatrix} E^{-1} & 0 \\ \underline{r}^{\times} E^{-1} & E^{-1} \end{bmatrix} \quad (19)$$

which is not the transpose of ${}_P\hat{X}_O$, so unlike 3×3 rotation matrices general spatial transformations are not orthogonal.

There are several ways of representing spatial displacements/coordinate transformations, most of which are described in [Rooney 1978]. From the computational point of view, the use of a 6×6 matrix of real numbers is one of the least efficient, both in terms of storage space and computation time. However, the machine implementation of an operation and its mathematical representation are largely independent, and chapter 7 describes methods for achieving the desired result (i.e. a change of representation) on a computer which are far more efficient than using 6×6 matrices.

2.7 Differentiation in Moving Coordinates

The absolute time derivative of a spatial vector $\hat{\underline{a}}$ represented in the coordinate system $\{\hat{\underline{e}}_i\}$ is

$$\begin{aligned} \frac{d}{dt} \hat{\underline{a}} &= \frac{d}{dt} \left(\sum_{i=1}^6 a_i \hat{\underline{e}}_i \right) \\ &= \sum_{i=1}^6 \left(\left(\frac{d}{dt} a_i \right) \hat{\underline{e}}_i + a_i \left(\frac{d}{dt} \hat{\underline{e}}_i \right) \right). \end{aligned} \quad (20)$$

If the coordinate system is stationary then $\frac{d}{dt} \hat{e}_i = \hat{0}$ and the absolute derivative of $\hat{a}$ is just the component-wise derivative. The problem to be discussed here is that of differentiating a spatial vector represented in a moving coordinate system.

The simplest approach is to transform the vector to a stationary coordinate system, differentiate it, and transform the derivative back to the moving coordinate system. This makes the (correct) assumption that the derivative of a spatial vector can be transformed like a spatial vector. A more rigorous approach is to obtain an expression for the derivative of a moving vector which can be substituted for $\frac{d}{dt} \hat{e}_i$ in eq. (20) and the problem solved from there; but this approach is more difficult to follow and will not be described here.¹

Let P be a moving coordinate system, O a stationary one, and $\hat{x}_p$ and $\hat{x}_o$ the transformations between the two. Using $\frac{d}{dt}$ to denote absolute differentiation and $\frac{d'}{dt}$ to denote apparent, or component-wise differentiation, the absolute derivative of a vector $\hat{a}$ represented in P coordinates is given by

$$\begin{aligned} \frac{d}{dt} \hat{a} &= {}_p\hat{x}_o \frac{d'}{dt} ({}_o\hat{x}_p \hat{a}) \\ &= \frac{d'}{dt} \hat{a} + {}_p\hat{x}_o \left(\frac{d'}{dt} {}_o\hat{x}_p \right) \hat{a}. \end{aligned} \quad (21)$$

The problem is to evaluate ${}_p\hat{x}_o \left(\frac{d'}{dt} {}_o\hat{x}_p \right)$. This is done by considering the components of ${}_o\hat{x}_p$. If the transformation from O to P coordinates is achieved by a translation $\underline{r}$ followed by a rotation E^{-1} (a 3x3 orthogonal matrix), then

1. The differentiation of moving screws and motors are described in [Woo and Freudenstein 1970] and [Brand 1953] respectively. The reasoning is the same for spatial vectors.

$$\begin{aligned} {}_0\hat{X}_p &= \begin{bmatrix} E & 0 \\ \underline{r} \times E & E \end{bmatrix}, \\ \frac{d'}{dt} {}_0\hat{X}_p &= \begin{bmatrix} \frac{d'}{dt} E & 0 \\ (\frac{d'}{dt} \underline{r}) \times E + \underline{r} \times \frac{d'}{dt} E & \frac{d'}{dt} E \end{bmatrix}. \end{aligned} \quad (22)$$

Let P coordinates have velocity $\hat{\underline{v}}$ expressed in O coordinates as $\hat{\underline{v}}_O = [\underline{\omega}^T \underline{v}_O^T]^T$. Now $\frac{d'}{dt} \underline{r} = \underline{v}_p = \underline{v}_O - \underline{r} \times \underline{\omega}$ and $\frac{d'}{dt} E = \underline{\omega} \times E$ (differentiation of an orthogonal matrix, e.g. see [Bottema and Roth 1979]), so

$$\begin{aligned} \frac{d'}{dt} {}_0\hat{X}_p &= \begin{bmatrix} \underline{\omega} \times E & 0 \\ (\underline{v}_O - \underline{r} \times \underline{\omega}) \times E + \underline{r} \times (\underline{\omega} \times E) & \underline{\omega} \times E \end{bmatrix} \\ &= \begin{bmatrix} \underline{\omega} \times E & 0 \\ \underline{v}_O \times E + \underline{\omega} \times \underline{r} \times E & \underline{\omega} \times E \end{bmatrix} \quad (\text{by eq. (15)}) \\ &= \begin{bmatrix} \underline{\omega} \times & 0 \\ \underline{v}_O \times & \underline{\omega} \times \end{bmatrix} \begin{bmatrix} E & 0 \\ \underline{r} \times E & E \end{bmatrix} \\ &= \begin{bmatrix} \underline{\omega} \times & 0 \\ \underline{v}_O \times & \underline{\omega} \times \end{bmatrix} {}_0\hat{X}_p. \end{aligned} \quad (23)$$

Introducing the spatial cross operator, $\hat{\times}$, defined by

$$\begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} \hat{\times} = \begin{bmatrix} \underline{a} \times & 0 \\ \underline{b} \times & \underline{a} \times \end{bmatrix}, \quad (24)$$

eq. (23) becomes

$$\frac{d'}{dt} {}_0\hat{X}_p = \hat{\underline{v}}_O \hat{\times} {}_0\hat{X}_p. \quad (25)$$

The spatial cross operator is the spatial analogue of the cross operator described earlier. It is linear and obeys rules equivalent to those given in eqs. (10, 11, 13-15) but with spatial vectors and

coordinate transformations in place of the ordinary vectors and 3×3 matrices. The expression $\underline{\hat{a}} \times \underline{\hat{b}}$ is the spatial equivalent of the motor product of motors [Brand 1953; Dimentberg 1965]. If we let $\underline{\hat{v}}_P = {}_P\hat{X}_O \underline{\hat{v}}_O$ be the velocity of P coordinates represented in P coordinates then we get

$$\frac{d}{dt} \underline{\hat{a}} = \frac{d'}{dt} \underline{\hat{a}} + \underline{\hat{v}}_P \times \underline{\hat{a}}. \quad (26)$$

In summary, the absolute derivative of a vector represented in a moving coordinate system is the sum of its apparent derivative and the cross product of the absolute velocity of the coordinate system with the vector being differentiated.

Eq. (26) may be used to find the derivative of a moving vector as follows. Let the vector $\underline{\hat{a}}$, represented in stationary coordinates, be moving with velocity $\underline{\hat{v}}$. At any instant let there be a coordinate system instantaneously coincident with the stationary coordinate system and moving with velocity $\underline{\hat{v}}$. Since the two coordinate systems coincide, $\underline{\hat{a}}$ has the same representation in both. The representations of the derivative are also the same, so we have

$$\begin{aligned} \frac{d}{dt} \underline{\hat{a}} &= \frac{d'}{dt} \underline{\hat{a}} + \underline{\hat{v}} \times \underline{\hat{a}} \\ &= \underline{\hat{v}} \times \underline{\hat{a}} \end{aligned} \quad (27)$$

(since $\underline{\hat{a}}$ is stationary in the moving coordinate system). The same argument can be used to find the apparent derivative of a vector moving in a moving coordinate system.

2.8 Spatial Acceleration

The absolute spatial acceleration of a rigid body is just the absolute derivative of its spatial velocity. However, the 'spatial acceleration of a rigid body differs somewhat from its 'conventional' acceleration.

Figure 2.3. Acceleration of a rigid body

Consider the rigid body shown in figure 2.3, which has angular velocity $\underline{\omega}$ and for which the linear velocity of the point P is $\underline{\dot{r}}$. The spatial velocity in stationary coordinates at O is $\underline{\hat{v}}_O = [\underline{\omega}^T (\underline{\dot{r}} + \underline{r} \times \underline{\omega})^T]^T$. The absolute spatial acceleration is the component-wise derivative of this, i.e.

$$\underline{\hat{a}}_O = \begin{bmatrix} \underline{\dot{\omega}} \\ \underline{\ddot{r}} + \underline{\dot{r}} \times \underline{\omega} + \underline{r} \times \underline{\dot{\omega}} \end{bmatrix} = \begin{bmatrix} \underline{a} \\ \underline{a}_O \end{bmatrix} . \quad (28)$$

The linear component of the spatial acceleration at P is $\underline{a}_P = \underline{a}_O - \underline{r} \times \underline{\dot{\omega}} = \underline{\ddot{r}} + \underline{\dot{r}} \times \underline{\omega}$, which differs from the conventional acceleration of P (i.e. $\underline{\ddot{r}}$) by a factor of $\underline{\dot{r}} \times \underline{\omega}$. This extra term

serves to make spatial acceleration transform like a vector - something which conventional acceleration does not do in a moving coordinate system.

Example 3

A rigid body traversing a helical course at constant speed has constant screw (and hence spatial) velocity, so its spatial acceleration is zero despite the fact that every point in the rigid body not on the axis of rotation has non-zero linear acceleration. (See figure 2.4.)

Example 4

Continuing with the robot defined in example 2, let us find the spatial acceleration of the end effector given that $\ddot{q}_i$ is the scalar joint acceleration about joint i . From example 2, the velocity of the end effector is given by $\hat{\underline{v}}_n = \sum_{i=1}^n \hat{\underline{s}}_i \dot{q}_i$. The acceleration, $\hat{\underline{a}}_n$, is the derivative of this, so

$$\hat{\underline{a}}_n = \frac{d}{dt} \sum_{i=1}^n \hat{\underline{s}}_i \dot{q}_i = \sum_{i=1}^n \left(\frac{d}{dt} \hat{\underline{s}}_i \dot{q}_i + \hat{\underline{s}}_i \ddot{q}_i \right).$$

Now $\hat{\underline{s}}_i$ is moving with velocity $\hat{\underline{v}}_i$, so $\frac{d}{dt} \hat{\underline{s}}_i = \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i$ and

$$\hat{\underline{a}}_n = \sum_{i=1}^n \left(\hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{q}_i + \hat{\underline{s}}_i \ddot{q}_i \right).$$

This is not the most efficient way to calculate the acceleration.

Example 5

A more efficient method is as follows. If $\hat{\underline{v}}_i$ is the absolute spatial velocity of link i , then

Chapter 3

Spatial Dynamics

3.1 Introduction

In the previous chapter I introduced the basic notions of spatial quantities and dealt with the kinematic aspects of spatial algebra. In this chapter I describe the spatial momentum and spatial inertia of a rigid body, and discuss the properties and representation of spatial inertia at length. The concepts of inverse inertias and articulated-body inertias are introduced. A scalar product is defined between force-type and motion-type vectors, the implementation of which as a matrix operation leads to the definition of the spatial transpose operator. Finally I introduce the elements of spatial vector analysis, and conclude with a summary of spatial vector algebra.

Much of the material in this chapter dealing with the representation of inertia, its properties and the equations of motion has been described by others, particularly Mises [1924a; 1924b], in connection with screws and motors, but the material on inverse inertia, articulated-body inertias, the spatial transpose operator and resolving vectors is new.

Figure 3.1 Spatial momentum of a rigid body

3.2 Spatial Rigid-Body Momentum

The spatial momentum of a rigid body is a vector composed of the linear and angular momenta of the rigid body. Consider a rigid body with mass m , position of centre of mass at P , and rotational inertia about the centre of mass I^* , as shown in figure 3.1. If the body has a pure translational velocity of $\vec{v}_P$ then its linear momentum is $\vec{p} = m \vec{v}_P$ and its moment of momentum, which is defined as the moment of the linear momentum about the origin, is given by $\vec{p}_O = \vec{OP} \times \vec{p}$. Thus the linear momentum of a rigid body behaves like a line vector with magnitude and direction given by $m \vec{v}_P$ and with line of action passing through P ; $\vec{p}$ and $\vec{p}_O$ being the Plücker vectors of the line vector.

If instead the body has a pure angular velocity $\vec{\omega}$ about an axis passing through P , then it has no linear momentum but only an intrinsic angular momentum of $I^* \vec{\omega}$, which is independent of the choice of origin and therefore behaves like a free vector.

Combining the above, let the rigid body have a spatial velocity $\hat{\underline{v}}_O = [\underline{\omega}^T \underline{v}_O^T]^T$. The linear momentum is given by

$$\underline{p} = m \underline{v}_p = m (\underline{v}_O + \overrightarrow{PO} \times \underline{\omega})$$

and the angular momentum is given by the sum of the moment of the linear momentum and the intrinsic angular momentum due to $\underline{\omega}$, which is

$$\begin{aligned} \underline{p}_O &= \overrightarrow{OP} \times \underline{p} + I^* \underline{\omega} \\ &= I^* \underline{\omega} + \overrightarrow{OP} \times m (\underline{v}_O + \overrightarrow{PO} \times \underline{\omega}) . \end{aligned}$$

So we have the following expression for the spatial momentum of a rigid body in terms of its spatial velocity and its inertia properties:

$$\hat{\underline{p}}_O = \begin{bmatrix} \underline{p} \\ \underline{p}_O \end{bmatrix} = \begin{bmatrix} m (\underline{v}_O + \overrightarrow{PO} \times \underline{\omega}) \\ I^* \underline{\omega} + \overrightarrow{OP} \times m (\underline{v}_O + \overrightarrow{PO} \times \underline{\omega}) \end{bmatrix} . \quad (1)$$

Yang's approach to dynamic analysis [Yang 1971] is based on this equation, but we must go one step further and obtain an explicit expression for the inertia.

3.3 Spatial Rigid-Body Inertia

Let the spatial inertia of a rigid body be defined as the tensor which transforms the spatial velocity of a rigid body into its spatial momentum. So we postulate a linear equation of the form $\hat{\underline{p}} = \hat{I} \hat{\underline{v}}$, where $\hat{I}$ is the spatial inertia of the rigid body and is represented by a 6x6 matrix. From eq. (1) we have

$$\begin{aligned}
\hat{\underline{p}}_O &= \begin{bmatrix} m \overrightarrow{PO} \times \underline{\omega} + m \underline{v}_O \\ (I^* + \overrightarrow{OP} \times m \overrightarrow{PO} \times) \underline{\omega} + \overrightarrow{OP} \times m \underline{v}_O \end{bmatrix} \\
&= \begin{bmatrix} m \overrightarrow{PO} \times & m \mathbf{1} \\ I^* + \overrightarrow{OP} \times m \overrightarrow{PO} \times & \overrightarrow{OP} \times m \end{bmatrix} \hat{\underline{v}}_O \\
&= \hat{I}_O \hat{\underline{v}}_O,
\end{aligned}$$

$$\rightarrow \hat{I}_O = \begin{bmatrix} m \overrightarrow{PO} \times & m \mathbf{1} \\ I^* + \overrightarrow{OP} \times m \overrightarrow{PO} \times & \overrightarrow{OP} \times m \end{bmatrix}. \quad (2)$$

This matrix can be expressed more concisely with the introduction of the 3×3 matrices $M = m \mathbf{1}$, $H = \overrightarrow{OP} \times m$ and $I = I^* + \overrightarrow{OP} \times m \overrightarrow{PO} \times$, whence

$$\hat{I}_O = \begin{bmatrix} H^T & M \\ I & H \end{bmatrix}. \quad (3)$$

M , H and I are matrix representations of the zeroth, first and second moments of mass of the rigid body about the origin: the zeroth moment is just the mass; the first moment is $\underline{h} = m \overrightarrow{OP}$, and $H = \underline{h} \times$; and the second moment is the rotational inertia of the rigid body about the origin, which is I by the parallel axes theorem ($\overrightarrow{OP} \times m \overrightarrow{PO} \times$ being the rotational inertia about the origin of a point mass m at P). M and I are symmetric.

When the coordinate origin coincides with the rigid body's centre of mass, then $\overrightarrow{OP} = \underline{0}$ and the spatial inertia takes on the special form

$$\hat{I}_P = \begin{bmatrix} 0 & M \\ I^* & 0 \end{bmatrix}. \quad (4)$$

3.3.1 Properties of Spatial Rigid-Body Inertias

Unlike rotational inertia, the matrix representing spatial inertia is neither symmetric nor positive definite in the conventional sense. However, when the spatial transpose operator is introduced the spatial inertia matrix will be found to be symmetric and positive definite with respect to that operator.

The rigid-body inertia matrix is always non-singular, for if it were singular then it would be possible for the rigid body to have zero momentum with non-zero velocity.

The coordinate transformation rule for spatial inertia is determined from its definition as a linear mapping of spatial vectors. If $\hat{\underline{p}}_a$, $\hat{\underline{I}}_a$ and $\hat{\underline{v}}_a$ are the momentum, inertia and velocity of a rigid body represented in coordinates a, and $\hat{\underline{p}}_b$, $\hat{\underline{I}}_b$ and $\hat{\underline{v}}_b$ are the same represented in coordinates b, then we have that $\hat{\underline{p}}_a = \hat{\underline{I}}_a \hat{\underline{v}}_a$ and $\hat{\underline{p}}_b = \hat{\underline{I}}_b \hat{\underline{v}}_b$. Now $\hat{\underline{p}}_a = {}_a\hat{X}_b \hat{\underline{p}}_b$ and $\hat{\underline{v}}_b = {}_b\hat{X}_a \hat{\underline{v}}_a$, where ${}_a\hat{X}_b$ and ${}_b\hat{X}_a$ are the usual vector transformation matrices, so

$$\begin{aligned} \hat{\underline{I}}_a \hat{\underline{v}}_a &= {}_a\hat{X}_b \hat{\underline{I}}_b \hat{\underline{v}}_b \\ &= {}_a\hat{X}_b \hat{\underline{I}}_b {}_b\hat{X}_a \hat{\underline{v}}_a, \\ \rightarrow \hat{\underline{I}}_a &= {}_a\hat{X}_b \hat{\underline{I}}_b {}_b\hat{X}_a. \end{aligned} \tag{5}$$

This is just the usual tensor transformation rule, and applies to any quantity defined as a linear mapping of vectors.

The derivative with respect to time of a spatial inertia represented in a stationary coordinate frame is just the component-wise derivative. A formula for differentiating a spatial inertia

represented in a moving coordinate frame is most easily obtained via the transformation approach. If a is a coordinate system moving with velocity $\hat{v}$, O a stationary one and $\hat{I}$ the inertia to be differentiated, then, using subscripts to indicate the coordinate system of representation, we have from eq. (5) that

$$\frac{d}{dt} \hat{I}_a = {}_a\hat{X}_O \frac{d'}{dt} \hat{I}_O {}_O\hat{X}_a .$$

But

$$\begin{aligned} \frac{d'}{dt} \hat{I}_O &= \frac{d'}{dt} ({}_O\hat{X}_a \hat{I}_a {}_a\hat{X}_O) \\ &= {}_O\hat{X}_a \hat{v}_a \hat{x} \hat{I}_a {}_a\hat{X}_O + {}_O\hat{X}_a \frac{d'}{dt} \hat{I}_a {}_a\hat{X}_O - {}_O\hat{X}_a \hat{I}_a \hat{v}_a \hat{x} {}_a\hat{X}_O \end{aligned}$$

(using eq. (2.25) and eq. (2.14) applied to spatial cross), so

$$\frac{d}{dt} \hat{I}_a = \frac{d'}{dt} \hat{I}_a + \hat{v}_a \hat{x} \hat{I}_a - \hat{I}_a \hat{v}_a \hat{x} . \quad (6)$$

If a rigid body has inertia $\hat{I}$ and is moving with an absolute velocity $\hat{v}$, then the absolute derivative of $\hat{I}$ is given by

$$\frac{d}{dt} \hat{I} = \hat{v} \hat{x} \hat{I} - \hat{I} \hat{v} \hat{x} . \quad (7)$$

This can be verified from eq. (6).

The inertia of a composite rigid body is the sum of the inertias of its components. If a body b is a rigid assembly composed of rigid bodies b_i then the inertia, $\hat{I}$, of b is given in terms of the inertias, $\hat{I}_i$, of b_i by

$$\hat{I} = \sum_i \hat{I}_i . \quad (8)$$

This single spatial equation takes the place of three equations in the conventional approach: one to compute the composite mass, one to compute the composite centre of mass, and one to compute the

rotational inertia about the new centre of mass.

3.4 The Equations of Motion

In the absence of an applied force, the momentum of a rigid body is conserved. So

$$\hat{\mathbf{I}} \hat{\mathbf{v}} = \text{constant} . \quad (9)$$

If a force is applied, then the rate of change of momentum is equal to the net applied force:

$$\begin{aligned} \hat{\mathbf{f}} &= \frac{d}{dt} (\hat{\mathbf{I}} \hat{\mathbf{v}}) \\ &= \hat{\mathbf{I}} \hat{\mathbf{a}} + \hat{\mathbf{v}} \hat{\times} \hat{\mathbf{I}} \hat{\mathbf{v}} , \end{aligned} \quad (10)$$

where $\hat{\mathbf{a}}$ is the rigid body's spatial acceleration. It will often be convenient to regard eq. (10) as an inhomogeneous linear equation

$$\hat{\mathbf{f}} = \hat{\mathbf{I}} \hat{\mathbf{a}} + \hat{\mathbf{p}} , \quad (11)$$

where $\hat{\mathbf{p}}$ is the force which must be applied to the rigid body to produce zero spatial acceleration, and is called the bias force. Eq. (9) combines the laws of conservation of linear and angular momentum, and eq. (10) combines Newton's and Euler's equations for linear and angular motion of a rigid body.

Example 1

In chapter 2, example 3, we learned that a rigid body traversing a helical course with constant speed has constant screw velocity and hence zero spatial acceleration; but a force is required to keep the body on this course, and we are now able

to calculate it:

$$\begin{aligned}\underline{\hat{f}} &= \underline{\hat{I}} \underline{\hat{a}} + \underline{\hat{v}} \times \underline{\hat{I}} \underline{\hat{v}} \\ &= \underline{\hat{v}} \times \underline{\hat{I}} \underline{\hat{v}} .\end{aligned}$$

$\underline{\hat{f}}$ will only be zero if $\underline{\hat{v}}$ is zero, if $\underline{\hat{v}}$ is purely linear, or if $\underline{\hat{I}} \underline{\hat{v}}$ is coaxial with $\underline{\hat{v}}$. This last case will occur if the screw axis of $\underline{\hat{v}}$ is one of the principal axes of the rotational inertia of the rigid body.

3.5 Inverse Inertia

The inverse inertia of a rigid body relates impulse (or momentum) to velocity, so if $\underline{\hat{p}} = \underline{\hat{I}} \underline{\hat{v}}$ then $\underline{\hat{v}} = \underline{\hat{\phi}} \underline{\hat{p}}$, where $\underline{\hat{\phi}} = \underline{\hat{I}}^{-1}$ is the inverse inertia of the rigid body. Inverse inertias obey the same rules for transformation and differentiation as do inertias, but their composition to form the inverse inertia of a composite body is accomplished by the harmonic sum, thus:

$$\underline{\hat{\phi}} = \left(\sum_i \underline{\hat{\phi}}_i^{-1} \right)^{-1} . \quad (12)$$

If a force $\underline{\hat{f}}$ is applied to a constrained body producing an acceleration $\underline{\hat{a}}$, then the acceleration may be expressed as

$$\underline{\hat{a}} = \underline{\hat{\phi}} \underline{\hat{f}} + \underline{\hat{b}} , \quad (13)$$

where $\underline{\hat{b}}$ is a vector depending on the velocity, motion constraints, etc., and is called the bias acceleration.

The main advantage in using inverse inertias comes in dealing with bodies constrained to have fewer than six degrees of motion freedom. For such a body, the equation $\hat{\Phi} \hat{f} = \hat{0}$ has non-trivial solutions, so $\hat{\Phi}$ is singular and $\hat{I}$ does not exist. In fact the rank of the inverse inertia equals the number of degrees of motion freedom possessed by the body, and the solutions to $\hat{\Phi} \hat{f} = \hat{0}$ form the space of possible reaction forces generated by the motion constraints. The main disadvantage of inverse inertias is the inconvenience of the combination rule: the harmonic sum is difficult to perform, especially if some of the matrices are singular.

The form of an unconstrained rigid-body inverse inertia with mass m , centre of mass at P and central inertia tensor I^* is

$$\hat{\Phi} = \begin{bmatrix} I^{*-1} \overrightarrow{PO}_x & I^{*-1} \\ \frac{1}{m} 1 + \overrightarrow{OP}_x I^{*-1} \overrightarrow{PO}_x & \overrightarrow{OP}_x I^{*-1} \end{bmatrix}. \quad (14)$$

This equation is identical to eq. (2) with $\frac{1}{m} 1$ replacing I^* and I^{*-1} replacing $m 1$.

3.6 Articulated-Body Inertias

I define an articulated body as a collection of rigid bodies connected by joints. All interactions between an articulated body and the rest of a rigid-body system must occur through a single member of the articulated body, which is called the handle. An articulated-body inertia gives the relationship between a force applied to the handle (by the rest of the system) and its acceleration, taking into account the dynamics of the articulated body. The relationship is

linear and may be expressed in the form

$$\underline{\hat{f}} = \hat{I}^A \underline{\hat{a}} + \underline{\hat{p}}, \quad (15)$$

where $\underline{\hat{f}}$ is the applied force, also known as the test force, $\underline{\hat{a}}$ is the handle's acceleration, $\hat{I}^A$ is the articulated-body inertia and $\underline{\hat{p}}$ is its associated bias force, which is the force which must be applied to the handle to give it zero acceleration.

The basic idea of the articulated-body inertia is that it allows the group of bodies making up the articulated body to be treated as if they were a single rigid-body-like element of the system. This is possible because all interactions with the rest of the system occur through the handle and the form of eq. (15) is the same as the rigid-body equation of motion, eq. (11). The use of articulated-body inertias allows the simplification of a rigid-body system by reducing the apparent number of elements. This is the basic idea behind the articulated-body dynamics algorithm described in chapter 6.

Eq. (15) assumes that the handle has six degrees of motion freedom. If this is not the case then the more general equation

$$\underline{\hat{a}} = \hat{\Phi}^A \underline{\hat{f}} + \underline{\hat{b}} \quad (16)$$

is required, where $\hat{\Phi}^A$ is the articulated-body inverse inertia and $\underline{\hat{b}}$ is the acceleration of the handle in the absence of a test force. An equation of this form always exists.

It is possible to generalise the notion of an articulated body to allow more than one handle. In this case the relationship between a test force applied to one handle and the acceleration of another would be described by a cross-inertia, which would normally need to

be expressed as an inverse inertia. For more details on this topic see chapter 9.

Articulated-body inertias depend only on the rigid-body inertias of the members and the instantaneous kinematics of the connections between the members. Velocity effects and the various forces acting on and within the articulated body affect only the bias force.

3.6.1 Properties of Articulated-Body Inertias

Articulated-body inertias are defined in terms of a linear relationship between a force and an acceleration. They are tensors and obey the same rules for transformation of representation and differentiation in moving coordinates as do rigid-body inertias. They have the same physical dimensions as rigid-body inertias. However, an expression of the form $\hat{\mathbf{I}}^A \hat{\mathbf{v}}$, where $\hat{\mathbf{I}}^A$ is an articulated-body inertia and $\hat{\mathbf{v}}$ is a velocity, is meaningless.

Articulated-body inertias, when they exist, are symmetric,¹ positive-definite matrices (with respect to spatial transpose); articulated-body inverse inertias are symmetric, positive semi-definite; and inverse cross-inertias are generally non-symmetric, indefinite and singular.

Articulated-body inertias are general symmetric matrices, not conforming to the special form of rigid-body inertias. Physically, this implies that there is no such thing as a centre of mass for an articulated body and that the apparent mass has directional

1. I have not proved this statement in the general case, but I can prove it for loop-free articulated bodies.

Figure 3.2 Articulated body with directional apparent mass

properties analogous to those of rotational inertia. To see this consider the simple articulated body shown in figure 3.2. It consists of two bodies: a sphere of mass m_1 with a cylindrical shaft through the middle, and a cylinder of mass m_2 which fits in the shaft. The shaft is parallel to the y axis and the two bodies are connected kinematically by a cylindric joint. Both bodies are initially at rest and their centres of mass coincide at the centre of the sphere.

A force $\hat{f}_z$ applied to the sphere along a line parallel to the z axis and passing through the centre of the sphere causes both bodies to accelerate with acceleration $\begin{bmatrix} 0^T & \frac{1}{m_1 + m_2} \hat{f}_z^T \end{bmatrix}^T$, but a force $\hat{f}_y$ applied to the sphere in the y direction through the centre of the sphere causes only the sphere to accelerate, so its acceleration is $\begin{bmatrix} 0^T & \frac{1}{m_1} \hat{f}_y^T \end{bmatrix}^T$. In fact the 3×3 matrix in the mass quadrant of the inertia for this articulated body, taking the sphere as the handle, is

$$M = \begin{bmatrix} m_1+m_2 & 0 & 0 \\ 0 & m_1 & 0 \\ 0 & 0 & m_1+m_2 \end{bmatrix} .$$

If the shaft were not parallel to one of the coordinate axes then M would not be diagonal, but it would still be symmetric. This articulated body exhibits a centre of mass, since any pure force applied to the sphere in a line passing through its centre imparts a purely linear acceleration to it, but if the cylindric joint is replaced by a screw joint then $\hat{f}_y$ causes the sphere to rotate, and a centre of mass no longer exists.

3.6.2 The Calculation of Articulated-Body Inertias

The calculation of articulated-body inertias (and cross-inertias) is very difficult in the general case; indeed only the inverse inertias can be guaranteed to exist. However, if the connectivity of the articulated-body is such that there are no kinematic loops and no connections to ground (an un-movable frame) then the calculation becomes relatively easy. The basic calculation method for this case is described in chapter 6. The presence of kinematic loops introduces the need to calculate inverse cross-inertias, and the presence of kinematic connections to ground causes some members to have less than six degrees of motion freedom, also necessitating the use of inverse inertias. A calculation method for these cases is described at the end of chapter 9.

3.7 Other Representations for Inertia

Ball [1900] analysed the spatial motion of a constrained rigid body using screws. He represented the spatial inertia in terms of a mass and a number of principal screws of inertia. The principal screws are effectively eigenvectors of inertia, and the number of screws required equals the number of degrees of motion freedom possessed by the rigid body. This approach has some advantages in dealing with the dynamics of a single constrained body. Its disadvantages are that it needs a variable number of quantities of two different types (scalar and screw) to describe a spatial inertia, and that combining inertias to form composite inertias is very difficult. Ball extends his analysis to multi-body systems with the theory of screw chains, but the method of analysis is unsuitable for computer implementation.

Spatial rigid-body inertia may be expressed as a 4×4 symmetric matrix, which is used, in conjunction with 4×4 transformation matrices, to express the kinetic energy of a moving rigid body. This representation has become popular for use in the Lagrangian approach to robot and mechanism dynamics, where finding an expression for the kinetic energy of the system is one of the first steps, and it was originally introduced by Uicker for this purpose [Uicker 1965; Uicker 1967; Paul 1981; Hollerbach 1980; Waters 1979]. However, this representation of inertia does not appear to be used for any other purpose, and it does not appear to be applicable to vectorial mechanics. Furthermore, it is not possible to represent articulated-body inertias this way.

Spatial inertia can not be represented directly in terms of dual numbers since the mass part of the inertia transforms the dual part of the velocity motor into (part of) the principal part of the momentum motor. This problem can be surmounted by defining a quantity called a binor [Dimentberg 1965], which is an ordered pair of 3×3 matrices of dual numbers. The operation of multiplying the binor (A, A^+) by a motor $\underline{a} + \epsilon \underline{a}_0$ is defined as

$$(A, A^+) (\underline{a} + \epsilon \underline{a}_0) = A \underline{a} + A^+ \underline{a}_0.$$

The form of the inertia binor is then $(H^T + \epsilon I, M + \epsilon H)$. Binors are linear operators, but not analytic. It is therefore not possible to arrive at the inertia binor via the principle of transference. One needs to define the operation of binor transformation, since motor transformation matrices will not work on a binor, and one runs into difficulty when trying to differentiate a binor. Compared with the 6×6 matrix, inertia binors are very awkward to use.

It is possible to perform dynamic analysis using motors without the need to represent inertia as a quantity in its own right by dealing directly with rigid-body momenta, as done by Yang [1971], or by using methods which do not require a vectorial equation of motion, as done by Woo and Freudenstein [1971]. Most of the algorithms presented in this dissertation do need to operate explicitly on inertias.

Mises' motor dyadic [Mises 1924a] is a 6×6 matrix of real numbers that differs from my own representation only in the arrangement of the quadrants. Mises' inertia dyadic is given by $\begin{bmatrix} M & H^T \\ H & I \end{bmatrix}$, which is

symmetric in the conventional sense. The difference is accounted for by Mises' definition of special multiplication operators between dyadics and motors. My arrangement is designed to fit as neatly as possible into the framework of matrix algebra. The only new operator required is the spatial transpose, which is the subject of the next section.

3.8 The Spatial Scalar Product

So far we have not defined a scalar product on spatial vectors. Such an operation needs to be able to combine the linear and angular components of two spatial vectors in such a way as to produce a single scalar that has some physical significance. The only sensible choice is for the scalar to be the work done by a spatial force over a spatial displacement, or to be more accurate the virtual work done by the force over an infinitesimal displacement, since finite displacements are not vectors. This is the definition used by Mises for his scalar product of motors, and is also the same as the dual part of the scalar product of motors in dual-number algebra [Brand 1953; Dimentberg 1965] and as the virtual coefficient of two screws [Ball 1900].

A linear force $\underline{f}$ acting over an infinitesimal linear displacement $\underline{\delta}$ performs an infinitesimal amount of work (virtual work) $\underline{f} \cdot \underline{\delta}$. Similarly a couple $\underline{t}$ acting over an infinitesimal angular displacement $\underline{\theta}$ performs an infinitesimal amount of work $\underline{t} \cdot \underline{\theta}$. Work is a scalar, so it is permissible to add work due to linear and angular effects.

Now let a spatial force $\hat{f} = [\underline{f}^T \underline{f}_0^T]^T$ act over an infinitesimal displacement $\hat{\delta} = [\underline{\delta}^T \underline{\delta}_0^T]^T$. $\underline{f}$ acts through the origin, and the linear displacement of the origin is $\underline{\delta}_0$, so the linear component of work is $\underline{f} \cdot \underline{\delta}_0$. Similarly, the angular component of work is $\underline{f}_0 \cdot \underline{\delta}$, so we define the spatial scalar product as follows:

$$\hat{f} \hat{\delta} = \underline{f} \cdot \underline{\delta}_0 + \underline{f}_0 \cdot \underline{\delta}. \quad (17)$$

This is a true scalar, independent of the choice of origin, as can easily be verified by substituting $\underline{f}_p$ and $\underline{\delta}_p$ in place of $\underline{f}_0$ and $\underline{\delta}_0$ and expanding in terms of $\underline{f}_0$, $\underline{\delta}_0$ and $\overrightarrow{OP}$.

This scalar product is defined only between a force-type vector and a motion-type vector.¹ This is the applicability restriction on the spatial scalar product. The spatial scalar product of a vector with itself is not defined, so we can not use the normal definition of length for a spatial vector. Two vectors satisfying $\hat{a} \hat{b} = 0$ are said to be orthogonal. (In screw theory they are said to be reciprocal.)

An important property of this scalar product is that it is not positive definite,² i.e. if we override the applicability restriction and allow the expression $\hat{a} \hat{a}$ we find that it may be positive, negative, or zero for non-zero $\hat{a}$, so we still can not use the normal definition of length. The effects of this property are largely

1. A force-type vector is one where the linear component is the line vector and a motion-type vector is one where the angular component is the line vector. There is a distinction between the two similar to that between polar and axial vectors, for if we change the rotation convention to the left-handed corkscrew then the representation of the spatial vector $\hat{a}$ becomes either $\hat{a}'$ or $-\hat{a}'$ depending on whether it represents a force-type vector or a motion-type vector.

2. See Lang [1972] (ch. 6) for the definition and properties of such a scalar product.

overshadowed by the effects of the applicability restriction.

3.9 The Spatial Transpose Operator

The scalar product operation can be expressed in matrix form as

$$\underline{\hat{a}} \cdot \underline{\hat{b}} = \underline{\hat{a}}^T \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \underline{\hat{b}},$$

but I shall eliminate the intermediate matrix by introducing the spatial transpose operator, s , defined on a spatial vector as follows:

$$\underline{\hat{a}}^s = \left[\begin{array}{c} \underline{\hat{a}} \\ \underline{\hat{a}}_0 \end{array} \right]^s = \left[\begin{array}{cc} \underline{\hat{a}}^T & \underline{\hat{a}}^T \end{array} \right]. \quad (18)$$

The scalar product is then represented by

$$\underline{\hat{a}} \cdot \underline{\hat{b}} = \underline{\hat{a}}^s \underline{\hat{b}}. \quad (19)$$

The spatial transpose operator is to have the same algebraic properties as the ordinary transpose operator, so $\underline{\hat{a}}^{ss} = \underline{\hat{a}}$, $(\underline{\hat{a}} + \underline{\hat{b}} + \dots)^s = \underline{\hat{a}}^s + \underline{\hat{b}}^s + \dots$, and so on. The spatial transpose of a scalar is just the original scalar, and the definition of the spatial transpose of a tensor $\hat{A}$ is obtained by requiring that

$$\underline{\hat{a}}^s \hat{A} \underline{\hat{b}} = (\underline{\hat{a}}^s \hat{A} \underline{\hat{b}})^s = \underline{\hat{b}}^s \hat{A}^s \underline{\hat{a}}$$

for all $\underline{\hat{a}}$ and $\underline{\hat{b}}$ (since $\underline{\hat{a}}^s \hat{A} \underline{\hat{b}}$ is a scalar). This results in the following definition:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^s = \begin{bmatrix} D^T & B^T \\ C^T & A^T \end{bmatrix}. \quad (20)$$

This can be memorised by observing that it is equivalent to an ordinary transpose about the broken diagonal through B and C as shown

below.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{matrix} A \\ C \end{matrix} \rightarrow \begin{bmatrix} D^T & B^T \\ C^T & A^T \end{bmatrix} \begin{matrix} D^T \\ C^T \end{matrix}$$

As a welcome bonus, use of the spatial transpose operator clears up some of the anomalies between spatial algebra and ordinary vector algebra. In particular, coordinate transformation matrices are spatially orthogonal, i.e. $\hat{X}^{-1} = \hat{X}^S$ (or ${}_a\hat{X}_b^S = {}_b\hat{X}_a$), the spatial cross product matrix is anti-symmetric, i.e. $(\hat{v}\hat{x})^S = -\hat{v}\hat{x}$, and spatial rigid-body inertias are both symmetric and positive definite, i.e. $\hat{I} = \hat{I}^S$ and $\hat{x}^S \hat{I} \hat{x} > 0 \quad \forall \hat{x} \neq \hat{0}$.

Example 2

The kinetic energy of a rigid body with inertia $\hat{I}$ and velocity $\hat{u}$ is equal to the amount of work done in accelerating it from rest to $\hat{u}$, so

$$KE = \int_{\hat{v}=\hat{0}}^{\hat{v}=\hat{u}} \hat{f}^S \hat{v} \, dt$$

Now

$$\hat{f} = \frac{d}{dt} (\hat{I} \hat{v}) = \hat{I} \hat{a} + \hat{v}\hat{x} \hat{I} \hat{v}$$

and

$$\begin{aligned} \hat{f}^S \hat{v} &= \frac{1}{2} (\hat{f}^S \hat{v} + \hat{v}^S \hat{f}) \\ &= \frac{1}{2} \left(\frac{d}{dt} (\hat{I} \hat{v})^S \hat{v} + \hat{v}^S (\hat{I} \hat{a} + \hat{v}\hat{x} \hat{I} \hat{v}) \right) \\ &= \frac{1}{2} \left(\frac{d}{dt} (\hat{I} \hat{v})^S \hat{v} + \hat{v}^S \hat{I} \hat{a} \right) \end{aligned}$$

$$= \frac{1}{2} \left(\frac{d}{dt} (\hat{\mathbf{I}} \hat{\mathbf{v}})^s \hat{\mathbf{v}} + (\hat{\mathbf{I}} \hat{\mathbf{v}})^s \frac{d}{dt} \hat{\mathbf{v}} \right),$$

so

$$KE = \frac{1}{2} \hat{\mathbf{u}}^s \hat{\mathbf{I}} \hat{\mathbf{u}}.$$

3.10 Spatial Vector Analysis

This section covers a number of topics with the general theme of resolving vectors into components on generalised bases. The main topics are the definition of spatial vector sub-spaces, a technique for resolving a vector into components which are elements of given sub-spaces, and the tensor product of spatial vectors.

3.10.1 Coordinates, Bases and Sub-Spaces

A set of spatial vectors $\hat{\mathbf{e}}_i$ is said to be linearly independent if the only solution to $\sum a_i \hat{\mathbf{e}}_i = \hat{\mathbf{0}}$ is for all a_i to be zero. Since spatial vector space is 6-dimensional, there can be at most six vectors in a linearly independent set. A set of r linearly independent spatial vectors span an r -dimensional sub-space of spatial vector space and define a basis on that sub-space. Any element of the sub-space may be expressed in the form

$$\hat{\mathbf{a}} = \sum_{i=1}^r a_i \hat{\mathbf{e}}_i, \quad (21)$$

where the numbers a_i are unique for each $\hat{\mathbf{a}}$ and are its coordinates in the basis $\{\hat{\mathbf{e}}_i\}$. The members of a basis, and hence all members of a

sub-space, must be of the same type: motion or force. A standard basis, as defined in chapter 2, is special in that its vectors are given a physical interpretation as line vectors and free vectors. No physical interpretation exists for the vectors of a general basis except through their representations in a standard basis. Unless a sub-space is indicated, a basis is assumed to span the entire spatial vector space.

Eq. (21) may be expressed in matrix form as

$$\underline{\hat{a}} = \hat{E} \underline{a} , \quad (22)$$

where $\hat{E}$ is a $6 \times r$ matrix whose i^{th} column is $\underline{\hat{e}}_i$, and $\underline{a}$ as an $r \times 1$ vector whose i^{th} element is a_i . $\hat{E}$ represents both a vector sub-space (the column space of the matrix) and the particular basis $\{\underline{\hat{e}}_i\}$ spanning that sub-space. $\underline{a}$ is just a vector of coefficients identifying a particular member of the sub-space.

The matrix $\hat{A} \hat{E}$, where $\hat{A}$ is a spatial transformation, represents the r -dimensional sub-space spanned by $\{\hat{A} \underline{\hat{e}}_i\}$. Any matrix $\hat{E} A$, where A is a non-singular $r \times r$ matrix, is equivalent to $\hat{E}$ in that it spans the same vector sub-space, but with a different set of basis vectors. The spatial transpose of a sub-space matrix is a straightforward extension of the spatial transpose of a vector, and is given by

$$\hat{E}^s = \begin{bmatrix} E \\ E_0 \end{bmatrix}^s = \begin{bmatrix} E^T & E_0^T \end{bmatrix} , \quad (23)$$

where E and E_0 are $3 \times r$ matrices. It is convenient to extend the definition of spatial transpose to apply to any non-spatial vector or matrix. For such a quantity it is defined to be the same operation as ordinary transpose. So, for example,

$$\hat{\underline{a}}^S = (\hat{\underline{E}} \underline{a})^S = \underline{a}^T \hat{\underline{E}}^S . \quad (24)$$

If $\hat{S}$ is an r -dimensional sub-space of the spatial vector space, then $\hat{S}^\perp$ is the sub-space orthogonal to $\hat{S}$, defined as the space of all vectors orthogonal to every element of $\hat{S}$. $\hat{S}^\perp$ has dimension $6-r$ and is of the opposite type to $\hat{S}$. If $\hat{S} \underline{\alpha}$ is a vector in $\hat{S}$ and $\hat{S}^\perp \underline{\beta}$ a vector in $\hat{S}^\perp$ then $(\hat{S} \underline{\alpha})^S \hat{S}^\perp \underline{\beta} = 0$; but this is true for all $\underline{\alpha}$ and $\underline{\beta}$, so we have the matrix equation

$$\hat{S}^S \hat{S}^\perp = 0 , \quad (25)$$

where here 0 is an $r \times (6-r)$ matrix of zeroes. Any two sub-spaces which satisfy this equation are said to be orthogonal, regardless of whether their dimensions add up to 6.

Vector sub-space matrices are used in chapter 8 to represent multiple-degree-of-freedom joints.

3.10.2 Resolving Vectors into Sub-Spaces and Components

It often happens that we wish to resolve a vector into its components in two sub-spaces. This is only possible in general if the two sub-spaces span the spatial vector space, and the solution is unique only if the two sub-spaces have no non-zero element in common. In our case it is also necessary for both sub-spaces to be of the same type as the vector to be resolved. So, given an r -dimensional sub-space $\hat{S}$, if we wish to resolve a vector (of the same type as the elements of $\hat{S}$) into unique components in $\hat{S}$ and $\hat{T}$, it is necessary and sufficient that $\hat{T}$ be a $6-r$ dimensional sub-space of the same type as and having no non-zero element in common with $\hat{S}$. I will describe the

general procedure for doing this shortly, but first I describe the special case of $r=1$.

Take two vectors $\underline{\hat{a}}$ and $\underline{\hat{b}}$ such that $\underline{\hat{a}}^S \underline{\hat{b}} \neq 0$. We wish to express a third vector $\underline{\hat{c}}$ as the sum of two vectors $\underline{\hat{c}}_1$ and $\underline{\hat{c}}_2$, where $\underline{\hat{c}}_1 = \underline{\hat{a}} \alpha$ and $\underline{\hat{b}}^S \underline{\hat{c}}_2 = 0$. $\underline{\hat{c}}_1$ is a member of the sub-space generated by $\underline{\hat{a}}$, and $\underline{\hat{c}}_2$ is a member of the orthogonal sub-space to $\underline{\hat{b}}$. These spaces satisfy the above conditions, so we are justified in saying that

$$\underline{\hat{c}} = \underline{\hat{c}}_1 + \underline{\hat{c}}_2 = \underline{\hat{a}} \alpha + \underline{\hat{c}}_2 .$$

The problem is solved once we know α . To find α simply premultiply by $\underline{\hat{b}}^S$ giving

$$\begin{aligned} \underline{\hat{b}}^S \underline{\hat{c}} &= \underline{\hat{b}}^S \underline{\hat{a}} \alpha + \underline{\hat{b}}^S \underline{\hat{c}}_2 = \underline{\hat{b}}^S \underline{\hat{a}} \alpha \\ \alpha &= \frac{\underline{\hat{b}}^S \underline{\hat{c}}}{\underline{\hat{b}}^S \underline{\hat{a}}} . \end{aligned} \tag{26}$$

The denominator is non-zero by hypothesis.

This can be generalised as follows. Let $\hat{S}$ and $\hat{T}$ be two r -dimensional sub-spaces such that for every member $\hat{S}_{\underline{\alpha}}$ of $\hat{S}$ there exists at least one member $\hat{T}_{\underline{\beta}}$ of $\hat{T}$ satisfying $(\hat{T}_{\underline{\beta}})^S \hat{S}_{\underline{\alpha}} \neq 0$. In terms of matrices, this means that the $r \times r$ matrix $\hat{T}^S \hat{S}$ is non-singular. Now the sub-spaces $\hat{S}$ and $\hat{T}^\perp$ span the spatial vector space, since the sum of their dimensions is 6 and they have no non-zero element in common, so it is possible to express a vector $\underline{\hat{a}}$ as $\underline{\hat{a}} = \hat{S}_{\underline{\alpha}} + \hat{T}^\perp_{\underline{\beta}}$. As before, $\underline{\alpha}$ may be found by pre-multiplying by $\hat{T}^S$ thus:

$$\hat{T}^S \underline{\hat{a}} = \hat{T}^S \hat{S}_{\underline{\alpha}} + \hat{T}^S \hat{T}^\perp_{\underline{\beta}} = \hat{T}^S \hat{S}_{\underline{\alpha}}$$

$$\rightarrow \underline{a} = (\hat{T}^S \hat{S})^{-1} \hat{T}^S \underline{\hat{a}}. \quad (27)$$

This process is very useful for resolving forces acting on a rigid body.

3.10.3 Reciprocal Bases and Orthogonal Bases

The applicability restriction prevents us from considering bases which are orthogonal with respect to the spatial scalar product, but we can consider reciprocal bases. Given the basis $\{\underline{\hat{e}}_i\}$, the vectors of the reciprocal basis $\{\underline{\hat{f}}_i\}$ are defined by

$$\hat{e}_i^S \hat{f}_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}. \quad (28)$$

It is always possible to find a reciprocal basis (using a variant of the Gram-Schmidt orthogonalisation process based on the above procedures). The reciprocal basis is useful for finding the components of a vector on a given basis:

$$\text{if } \underline{\hat{v}} = \sum \underline{\hat{e}}_i v_i \quad \text{then} \quad v_i = \hat{f}_i^S \underline{\hat{v}}. \quad (29)$$

It is possible for a basis to be orthogonal with respect to an inner product defined between vectors of the same type. For example, we can say that the motion-type basis $\{\underline{\hat{e}}_i\}$ is orthogonal with respect to the inertia $\hat{I}$ if $\hat{e}_i^S \hat{I} \hat{e}_j = 0 \quad \forall \quad i \neq j$, and similarly for a force-type basis and an inverse inertia. Such a basis can always be found using a variant of the Gram-Schmidt orthogonalisation process described in [Lang 1972]. If the matrix is positive definite then so is the inner product.

Example 3

A rigid body with inertia $\hat{I}$ is initially at rest and its motion is constrained to the sub-space spanned by $\hat{S}$. Find the acceleration of the body in response to an applied force $\hat{f}$. To solve this problem, we hypothesise that $\hat{f}$ may be split into two components, $\hat{f}_1$ and $\hat{f}_2$, where $\hat{f}_1$ is responsible for the motion and $\hat{f}_2$ is absorbed by the reaction forces produced by the motion constraints. The acceleration is constrained to lie in $\hat{S}$, and may therefore be expressed as $\hat{S} \underline{\alpha}$. If $\hat{f}_1$ is responsible for causing the motion then $\hat{f}_1 = \hat{I} \hat{S} \underline{\alpha}$, and is therefore a member of the sub-space $\hat{I} \hat{S}$. $\hat{f}_2$ does no work, so $\hat{S}^s \hat{f}_2 = 0$ meaning that $\hat{f}_2$ is a member of $\hat{S}^\perp$. Now $\hat{S}^s \hat{I} \hat{S}$ is non-singular, so the hypothesised decomposition is permissible, and the acceleration is given by

$$\underline{\alpha} = (\hat{S}^s \hat{I} \hat{S})^{-1} \hat{S}^s \hat{f}$$

$$\rightarrow \hat{S} \underline{\alpha} = \hat{S} (\hat{S}^s \hat{I} \hat{S})^{-1} \hat{S}^s \hat{f}.$$

Notice that $\hat{S} (\hat{S}^s \hat{I} \hat{S})^{-1} \hat{S}^s$ is the articulated-body inverse inertia of the constrained body.

3.10.4 Inertia Dyads and the Tensor Product

The tensor product, $\hat{\underline{a}} \hat{\underline{b}}$, of two spatial vectors $\hat{\underline{a}}$ and $\hat{\underline{b}}$ is defined as follows: for any third vector $\hat{\underline{c}}$,

$$(\hat{\underline{a}} \hat{\underline{b}}) \hat{\cdot} \hat{\underline{c}} = \hat{\underline{a}} (\hat{\underline{b}} \hat{\cdot} \hat{\underline{c}}). \quad (30)$$

The tensor product of two vectors is represented as the 6×6 matrix $\hat{\underline{a}} \hat{\underline{b}}^s$. Such a quantity is known as a dyad and is a matrix of rank 1

(rank used in the matrix sense). The sum of r independent dyads is a matrix of rank r ; a set of dyads $\hat{\underline{a}}_i \hat{\underline{b}}_i^S$ being independent if the vectors $\hat{\underline{a}}_i$ are independent and the vectors $\hat{\underline{b}}_i$ are independent. Any linear transformation of spatial vectors can be expressed as the sum of six (or fewer) dyads, for a linear transformation is completely specified by its effect on six linearly independent vectors. For example, if $\{\hat{\underline{e}}'_i\}$ is the image of $\{\hat{\underline{e}}_i\}$ under some transformation $\hat{A}$, then the transformation can be expressed as $\hat{A} = \sum_{i=1}^6 \hat{\underline{e}}'_i \hat{\underline{f}}_i^S$, where $\{\hat{\underline{f}}_i\}$ is the reciprocal basis to $\{\hat{\underline{e}}_i\}$.

A rigid-body inertia, or indeed any symmetric articulated-body inertia or inverse inertia, can be represented as the sum of six symmetric inertia dyads. Let $\{\hat{\underline{s}}_i\}$ be an orthogonal basis with respect to the inertia $\hat{I}$, i.e. $\hat{\underline{s}}_i^S \hat{I} \hat{\underline{s}}_j = 0 \quad \forall i \neq j$, then $\hat{I}$ transforms $\{\hat{\underline{s}}_i\}$ to $\{\hat{I} \hat{\underline{s}}_i\}$, and $\{\frac{\hat{I} \hat{\underline{s}}_i}{\hat{\underline{s}}_i^S \hat{I} \hat{\underline{s}}_i}\}$ is a reciprocal basis to $\{\hat{\underline{s}}_i\}$, so $\hat{I}$ may be represented as

$$\hat{I} = \sum_{i=1}^6 \frac{\hat{I} \hat{\underline{s}}_i \hat{\underline{s}}_i^S \hat{I}}{\hat{\underline{s}}_i^S \hat{I} \hat{\underline{s}}_i}. \quad (31)$$

We are free to make the $\hat{\underline{s}}_i$ orthonormal with respect to $\hat{I}$, in which case the denominator disappears. Similarly, the inverse inertia may be expressed as

$$\hat{I}^{-1} = \sum_{i=1}^6 \frac{\hat{\underline{s}}_i \hat{\underline{s}}_i^S}{\hat{\underline{s}}_i^S \hat{I} \hat{\underline{s}}_i} \quad (32)$$

by virtue of the identity $\hat{I}^{-1} = \hat{I}^{-1} \hat{I} \hat{I}^{-1}$, and we have that

$$\sum_{i=1}^6 \frac{\hat{I} \hat{s}_i \hat{s}_i^S}{\hat{s}_i^S \hat{I} \hat{s}_i} = \sum_{i=1}^6 \frac{\hat{s}_i \hat{s}_i^S \hat{I}}{\hat{s}_i^S \hat{I} \hat{s}_i} = \hat{I} \quad (33)$$

Example 4

From example 3 we learned that the inverse inertia of a body with rigid-body inertia $\hat{I}$ and constrained to move in an r -dimensional sub-space $\hat{S}$ is given by

$$\hat{\Phi} = \hat{S} (\hat{S}^S \hat{I} \hat{S})^{-1} \hat{S}^S.$$

Since here $\hat{S}$ represents only the vector sub-space, we may choose the basis as we please, and in particular we may choose the vectors $\hat{s}_i$ making up $\hat{S}$ to be orthogonal with respect to $\hat{I}$. If we do this then $\hat{S}^S \hat{I} \hat{S}$ becomes diagonal, so $(\hat{S}^S \hat{I} \hat{S})^{-1}$ is also diagonal, and we may expand the above equation as

$$\hat{\Phi} = \sum_{i=1}^r \frac{\hat{s}_i \hat{s}_i^S}{\hat{s}_i^S \hat{I} \hat{s}_i}.$$

This shows clearly that $\hat{\Phi}$ is the sum of r dyads, one for each degree of freedom, and hence that the rank of an inverse inertia is the degree of motion freedom of the body to which it refers.

3.11 Summary of Spatial Algebra

A spatial vector is an element of a 6-dimensional vector space over the real numbers. It may be interpreted geometrically as having a line vector component and a free vector component. Analogous linear and angular physical quantities (e.g. linear and angular velocity, force, etc.) may be amalgamated to form spatial vectors if it

makes physical sense to regard one of them as a line vector and the other as a free vector. A spatial vector representing a physical quantity may be classified as either motion or force type depending on whether the line vector part is the angular or linear component.

A spatial vector is represented as a 6×1 column vector of components in a standard basis, which comprises three unit line vectors forming the axes of a Cartesian coordinate frame and three unit free vectors parallel to the coordinate axes. The representation of a spatial vector may be transformed from one standard basis to another by a rigid-body coordinate transformation, which is represented as a 6×6 matrix. The basic operations of addition and multiplication by a scalar may be performed on spatial vectors provided there is a physical interpretation for such an operation. Transformations are combined and applied to vectors by matrix multiplication.

A formula for differentiating spatial vectors in moving coordinates is defined, which is closely analogous to the corresponding formula for differentiating an ordinary vector in rotating coordinates.

A scalar product is defined between motion and force type vectors which is not positive definite. To implement this as a matrix operation, the spatial transpose operator is defined and used in place of ordinary transpose. All spatial quantities are subject to spatial transpose, but apart from this the normal rules of matrix algebra apply. The use of spatial transpose makes coordinate transformations orthogonal, spatial rigid-body inertias symmetric, and cross product matrices anti-symmetric.

The spatial inertia of a rigid body is defined as a tensor which is spatially symmetric and positive definite, and which is represented as a 6×6 matrix. Rigid-body inertias may be summed to form the inertia of a composite rigid body. An equation of motion is defined relating the force applied to a rigid body to the rate of change of its momentum. The concepts of articulated-body inertias and inverse inertias, which are the apparent inertias of collections of rigid bodies treated as if they were one, are introduced. The facility with which inertias can be manipulated is the distinguishing feature of spatial algebra and what makes it more suitable for analysing the dynamics of rigid-body systems than screws or motors.

Spatial quantities are denoted in this thesis by carets. Normally no notational distinction is made between an object and its representation, but if a distinction is needed for clarity then representations are denoted by tildes.

This concludes the description of the spatial notation and underlying spatial algebra. In the rest of this dissertation it will be used as a tool for analysing robot dynamics and expressing dynamics calculation algorithms. These two chapters have covered all the material necessary for the analyses in the remainder of this dissertation.

Chapter 4

Inverse Dynamics - The Recursive Newton-Euler Method

4.1 Introduction

In this chapter I describe the recursive Newton-Euler formulation for inverse robot dynamics - the calculation of the forces required to produce a given acceleration. I start with inverse dynamics partly because it is an easier problem to solve than forward dynamics, thus making it a good introduction to the subject of robot dynamics, and partly because it is an essential part of the composite-rigid-body method for forward dynamics, which is the subject of the next chapter.

Inverse dynamics is primarily of use in control (e.g. see [Horn and Raibert 1977; Luh et al. 1980b]) where it is used to provide estimates of the joint forces required to drive the robot with accelerations decided upon by the control system. In this application the ability to perform the inverse dynamics computation in real time is of prime importance.

The recursive Newton-Euler formulation is not new, and is described in [Luh et al. 1980a]. It is the most efficient currently known algorithm for the calculation of the inverse dynamics of a robot consisting of an open-loop kinematic chain with general

kinematic and inertia parameters. More efficient formulations do exist for robots with special geometries and/or simplified inertia parameters (e.g. see [Hollerbach and Sahar 1983; Kane and Levinson 1983]).

Before describing the algorithm itself, I give an example of how the use of recurrence relations can produce computationally efficient algorithms, and I define the system model of a robot which will be used to describe this algorithm and the ones in chapters 5 and 6.

4.1.1 History of Robot Dynamics

There have been two main approaches to formulating robot dynamics: the Lagrange approach and the Newton-Euler approach. They are equivalent in that they solve the same problem, but the route to the solution is different for each. In both approaches the key to computational efficiency is the use of recursive formulations, which not only reduce the computational requirement substantially over non-recursive formulations, but also reduce the computational complexity from (typically) $O(n^4)$ to $O(n)$.

In the Lagrange approach, the Lagrangian of the manipulator is expressed in terms of joint variables and velocities (which are the generalised coordinates and their derivatives), and the expression is substituted into the Euler-Lagrange equation which is expanded by symbolic differentiation to give the generalised forces (joint forces) in terms of the generalised coordinates, velocities and accelerations. This method was first applied to robot dynamics by Kahn [1969], who adapted Uicker's Lagrangian formulation for more

general mechanisms [Uicker 1965]. Kahn's formulation is computationally inefficient, but the efficiency of the more recently developed recursive Lagrangian formulation [Hollerbach 1980] is comparable with that of the recursive Newton-Euler formulation.

In the Newton-Euler approach, the equations of motion (Newton's + Euler's equations) are applied to each link and the resulting equations are combined with constraint equations from the joints in such a way as to give the joint forces in terms of the joint accelerations. This method was originally developed for multi-body satellite dynamics [Hooker and Margulies 1965; Hooker 1970], and has been applied to robot mechanisms by Stepanenko and Vukobratović [1976], Vukobratović [1978], Vukobratović and Potkonjak [1979] and Orin et al. [1979]. Some of these formulations are partially recursive, but it was Luh et al. [1980a] who developed the fully recursive Newton-Euler formulation.

Other relevant contributions include [Armstrong 1979; Horn and Raibert 1977; Huston et al. 1978; Mufti 1979; Mufti 1983; Silver 1982; Turney et al. 1980; Waters 1979; Walker and Orin 1982]. A more detailed account of robot dynamics history may be found in [Brady et al. 1982].

4.2 Recurrence Relations

All of the algorithms described in this dissertation use recurrence relations to achieve computational efficiency.

A recurrence relation is an equation defining a member of a sequence in terms of its predecessors. Usually it is possible to generate any member of a sequence by the appropriate number of iterations of the recurrence relation, given suitable starting values; in which case the sequence is defined recursively by the recurrence relation and its starting values.

In the context of robot dynamics, the elements of the sequences would be, for example, the spatial velocities of the robot's links, or their accelerations, and a recurrence relation giving the velocity or acceleration of link i in terms of that of link $i-1$ could be used to calculate the velocity or acceleration of each link in turn.

It turns out that the proper use of recurrence relations leads to algorithms which are substantially faster than 'equation-of-motion' type methods, for reasons described in chapter 1. For example, the recursive Newton-Euler method is approximately 100 times faster than the Uicker/Kahn method for a six-degree-of-freedom robot [Hollerbach 1980].

In view of the dramatic effect that recurrence relations have on the efficiency of robot dynamics computations, it is worthwhile spending a few moments discussing how they work. The question we should ask ourselves is not 'why are recurrence relations so efficient?' but rather 'why are other methods so inefficient?'. I illustrate the answer with a simple example.

Let the matrix B be defined as the product of n matrices A_i thus:

$$B = \prod_{i=1}^n A_i . \quad (1)$$

The task is to compute the numeric value of $\dot{B}$ ($=\frac{d}{dt}B$) given numeric values for A_i and $\dot{A}_i$. Symbolic differentiation of eq. (1) gives the following equation for $\dot{B}$ in terms of A_i and $\dot{A}_i$:

$$\dot{B} = \sum_{i=1}^n P_i \quad \text{where} \quad P_i = \left(\prod_{j=1}^{i-1} A_j \right) \dot{A}_i \left(\prod_{j=i+1}^n A_j \right). \quad (2)$$

To calculate $\dot{B}$ directly from eq. (2) requires the calculation of n matrices P_i , each requiring $n-1$ multiplications, and their summation, requiring $n-1$ additions. The total amount of computation is $n(n-1)$ matrix multiplications and $n-1$ matrix additions, or $n(n-1)m + (n-1)a$ for short.

This is the 'algebraic equation' approach to computing the answer, and is characteristic of many early robot dynamics calculation schemes. The problem is solved algebraically giving a symbolic equation, and the numeric solution is obtained by substituting numeric values into the symbolic equation and evaluating it.

This approach is usually highly inefficient because of the large amount of repeated calculation in evaluating the solution. The repetition is not particularly obvious, but is brought about by the fact that an algebraic solution to a problem like this contains a great many terms which are almost identical but which are each calculated independently of the others. In this case, P_i differs from P_j by only 2 factors out of n , but each is calculated independently. For example, consider the case when $n=5$. The computation performed is the following:

$$\dot{B} = \dot{A}_1 A_2 A_3 A_4 A_5 +$$

$$A_1 \dot{A}_2 A_3 A_4 A_5 +$$

$$A_1 A_2 \dot{A}_3 A_4 A_5 +$$

$$A_1 A_2 A_3 \dot{A}_4 A_5 +$$

$$A_1 A_2 A_3 A_4 \dot{A}_5 .$$

Notice that $A_1 A_2$ is calculated three times, and in two of those cases the result is multiplied by A_3 . Similar remarks apply to A_3 , A_4 and A_5 . This repetition escalates as n increases.

Calculating the answer by means of recurrence relations offers a way of eliminating these unnecessary calculations. To solve the problem using recurrence relations, introduce the sequence of partial results $B_i = \prod_{j=1}^i A_j$, in which case B_{i+1} is given in terms of B_i by the recurrence relation

$$B_{i+1} = B_i A_{i+1} . \quad (3)$$

Repeated application of eq. (3), starting with $B_1 = A_1$, allows us to calculate all the B_i , and hence B ; but the task is to calculate $\dot{B}$, not B , so we differentiate eq. (3) to get a recurrence relation in $\dot{B}_i$:

$$\dot{B}_{i+1} = \dot{B}_i A_{i+1} + B_i \dot{A}_{i+1} . \quad (4)$$

Now we can calculate $\dot{B}$ from eq. (4), starting with $\dot{B}_1 = \dot{A}_1$. To calculate $\dot{B}$ this way requires $n-1$ iterations of eq. (4), where one iteration requires $2m+1a$, but we also need the values of $B_1 \dots B_{n-1}$, which requires $n-2$ iterations of eq. (3) (or none if $n < 2$). So the overall computational requirement is $(3n-4)m + (n-1)a$.

size	algebraic	recurrence relation
n	n^2-n	$3n-4$
1	0	0
2	2	2
3	6	5
5	20	11
10	90	26
20	380	56

Table 4.1 Comparison of the computational requirements of the algebraic and recurrence relation methods

Table 4.1 shows the computational requirements (for multiplication only) of the two methods for various values of n . Notice that the computational complexity of the recurrence relation method is lower than that of the algebraic equation method ($O(n)$ vs. $O(n^2)$), so the improvement increases with problem size. The degree of improvement is greater with a more complicated problem, like robot dynamics, where the computational complexity is improved from $O(n^4)$ to $O(n)$.

4.3 The Robot System Model

The dynamics algorithms in this and the following chapters are described using a system model of the robot. This is a description of the robot in terms of the number of links, their inertias, and the joints connecting them.

The first step in defining a system model is to label the various components (joints and links) and specify which is connected to which. Normally the connectivity¹ has to be given explicitly, but

1. Connectivity is used here in the topological sense.

Figure 4.1 Robot connectivity

the robot we will be considering consists of a single un-branched, open-loop kinematic chain, as shown in figure 4.1, for which the connectivity is implicit in the numbering scheme.

Let there be n movable links, numbered $1..n$, and n one-degree-of-freedom joints, also numbered $1..n$, such that joint i connects link i to link $i-1$. Joint 1 connects link 1 to the immovable base member, which will be called link 0 for convenience.

Figure 4.2 Robot geometry and parameters

The next step is to define the geometry of the linkage, the types of joint and the inertia parameters (see figure 4.2). To do this, a set of link coordinate systems is introduced, one attached to each link, where the coordinate system attached to link i is called i -coordinates. Since the position of a link is fixed in its own coordinate system, we can express the inertia of link i and the axis of any joint embedded in link i as a constant in i -coordinates. Let $\hat{I}_i'$ be the spatial inertia of link i and let $\hat{s}_i'$ describe the motion allowed by joint i , both expressed in i -coordinates. $\hat{s}_i'$ is either a line vector or a free vector, depending on whether joint i is revolute or prismatic (screw joints are considered in chapter 8).

Having introduced link coordinates, we need to be able to transform representations of objects into and out of link coordinates. To do this, the adjacent-link coordinate transformations ${}^i\hat{X}_{i-1}$ are introduced. Each such transformation is a function of one joint variable, and needs to be computed on demand. The details of how this is done depend on how the link coordinate systems are chosen, and are left to chapter 7. Other coordinate transformations can be computed from the adjacent-link transformations. It will be convenient at this stage to make 0-coordinates synonymous with absolute coordinates.

Finally, we need to define the system variables. These are as follows: q_i is the joint variable for joint i , $\dot{q}_i$ its velocity, $\ddot{q}_i$ its acceleration and Q_i the joint force. They may be treated as the components of the n -dimensional vectors $\underline{q}$, $\underline{\dot{q}}$, $\underline{\ddot{q}}$ and $\underline{Q}$. We will always assume that $\underline{q}$ and $\underline{\dot{q}}$ are known and that we are either trying to find $\underline{\ddot{q}}$ given $\underline{Q}$ (forward dynamics) or $\underline{Q}$ given $\underline{\ddot{q}}$ (inverse dynamics) at some particular instant.

To simplify the task of describing the dynamics algorithms, they will each be derived initially in absolute coordinates. To facilitate this the quantities $\hat{\underline{s}}_i$ and $\hat{\underline{I}}_i$ are introduced, which are the joint motion and spatial inertia of link i expressed in absolute coordinates. Together they define the instantaneous kinematics and inertia of the system. They may be calculated at any instant from $\hat{\underline{s}}'_i$, $\hat{\underline{I}}'_i$ and ${}^0\hat{\underline{X}}_i$, the latter being calculated (using a recurrence relation) from ${}^0\hat{\underline{X}}_{i-1}$ and ${}_{i-1}\hat{\underline{X}}_i$.

In chapter 8 many enhancements to this basic system model will be discussed, including branched kinematic chains, multiple-degree-of-freedom joints and moving bases, while chapter 9 allows for the possibility of kinematic loops.

4.4 The Recursive Newton-Euler Formulation

The problem is stated as follows: given a system model of a robot as outlined in the previous section and the values of the desired joint accelerations, find the joint forces required to produce those accelerations. The solution to this problem by the recursive Newton-Euler method consists of three steps:

- 1) calculate the velocity and acceleration of each link,
- 2) calculate the net force acting on each link from its motion and inertia,
- 3) calculate the joint forces required to produce the forces in step 2.

Step 1: Link Motion

Let $\hat{\underline{v}}_i$ be the absolute velocity of link i and $\hat{\underline{a}}_i$ its absolute acceleration. The velocity of each link may be calculated from the recurrence relation

$$\hat{\underline{v}}_i = \hat{\underline{v}}_{i-1} + \hat{\underline{s}}_i \dot{q}_i \quad (\hat{\underline{v}}_0 = \hat{\underline{0}}) \quad (5)$$

(The starting value is shown in brackets.) The recurrence relation for accelerations is obtained by differentiating eq. (5) and is given by

$$\hat{\underline{a}}_i = \hat{\underline{a}}_{i-1} + \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{q}_i + \hat{\underline{s}}_i \ddot{q}_i \quad (\hat{\underline{a}}_0 = \hat{\underline{0}}) \quad (6)$$

Successive iterations of eqs. (5) and (6) allow us to calculate the velocity and acceleration of every link.

Step 2: Net Forces on the Links

The net force acting on link i is given by the spatial equation of motion as the link's rate of change of momentum:

$$\begin{aligned} \hat{\underline{f}}_i^* &= \frac{d}{dt} (\hat{\underline{I}}_i \hat{\underline{v}}_i) \\ &= \hat{\underline{I}}_i \hat{\underline{a}}_i + \hat{\underline{v}}_i \hat{\times} \hat{\underline{I}}_i \hat{\underline{v}}_i \end{aligned} \quad (7)$$

Step 3: Joint Forces

First we must find the total force transmitted from link i-1 to link i through joint i, which includes the component due to bearing reaction forces. If $\hat{\underline{f}}_i$ is this force, then we have

$$\begin{aligned} \hat{\underline{f}}_i - \hat{\underline{f}}_{i+1} &= \hat{\underline{f}}_i^* \\ \rightarrow \hat{\underline{f}}_i &= \hat{\underline{f}}_{i+1} + \hat{\underline{f}}_i^* \quad (\hat{\underline{f}}_n = \hat{\underline{f}}_n^*) \end{aligned} \quad (8)$$

This recurrence relation allows us to calculate all the total joint forces starting from the end effector. The scalar joint forces are then given by¹.

$$Q_i = \hat{\underline{s}}_i^s \hat{\underline{f}}_i \quad (9)$$

The equations (5-9) form the complete Newton-Euler formulation for inverse dynamics.

Known external forces acting on individual links may be included in eq. (8) thus (see figure 4.3):

$$\hat{\underline{f}}_i = \hat{\underline{f}}_{i+1} + \hat{\underline{f}}_i^* - \hat{\underline{f}}_i^x,$$

where $\hat{\underline{f}}_i^x$ is the external force acting on link i . Gravity may be dealt with in this manner, but it is more efficient to simulate gravity by giving the robot a fictitious acceleration. This can be done by using $\hat{\underline{a}}_0 = -\hat{\underline{g}}$ as the starting value of eq. (6), where $\hat{\underline{g}}$ is the gravitational acceleration vector.

4.4.1 The Recursive Newton-Euler Method in Link Coordinates

A more efficient formulation of the recursive Newton-Euler method can be obtained by performing those calculations pertinent to link i in i -coordinates. The increased efficiency is due mainly to it being easier to transform three vector quantities ($\hat{\underline{v}}_i$, $\hat{\underline{a}}_i$ and $\hat{\underline{f}}_i$) to an

1. This equation is derived from the power balance equation, which in spatial form is $\hat{\underline{v}}_i^s \hat{\underline{f}}_i = \dot{Q}$.

adjacent coordinate system than it is to transform one vector and a tensor ($\hat{\underline{s}}'_i$ and $\hat{\underline{I}}'_i$) from i coordinates to absolute coordinates; there are also other factors involved, and a full discussion on this topic is postponed until chapter 7.

Converting a formulation in absolute coordinates to one in link coordinates is very simple. Firstly, all the quantities defined as being represented in absolute coordinates are replaced by or redefined as quantities represented in link coordinates. So $\hat{\underline{I}}_i$ is replaced by $\hat{\underline{I}}'_i$, $\hat{\underline{v}}_i$ is redefined as represented in i -coordinates, and so on. Secondly, quantities on the right-hand side of an equation which are represented in a coordinate system other than that of the quantity on the left-hand side are transformed accordingly.

The result of applying these rules to eqs. (5-9) is as follows:

$$\hat{\underline{v}}_i = {}_i\hat{X}_{i-1} \hat{\underline{v}}_{i-1} + \hat{\underline{s}}'_i \dot{q}_i, \quad (\hat{\underline{v}}_0 = \underline{\hat{0}}) \quad (10)$$

$$\hat{\underline{a}}_i = {}_i\hat{X}_{i-1} \hat{\underline{a}}_{i-1} + \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}'_i \dot{q}_i + \hat{\underline{s}}'_i \ddot{q}_i, \quad (\hat{\underline{a}}_0 = \underline{\hat{0}}) \quad (11)$$

$$\hat{\underline{f}}_i^* = \hat{\underline{I}}'_i \hat{\underline{a}}_i + \hat{\underline{v}}_i \hat{\times} \hat{\underline{I}}'_i \hat{\underline{v}}_i, \quad (12)$$

$$\hat{\underline{f}}_i = {}_i\hat{X}_{i+1} \hat{\underline{f}}_{i+1} + \hat{\underline{f}}_i^*, \quad (\hat{\underline{f}}_n = \hat{\underline{f}}_n^*) \quad (13)$$

$$Q_i = \hat{\underline{s}}'_i{}^S \hat{\underline{f}}_i. \quad (14)$$

4.5 Summary

I have described the Newton-Euler formulation for inverse dynamics, and in the process introduced the basic robot system model and explained how recurrence relations work. To evaluate the inverse dynamics for an n -degree-of-freedom robot requires $n-1$ iterations of eq. (8) and n iterations of each of eqs. (5-7 and 9), so it is clearly an $O(n)$ algorithm. In the next chapter I describe the composite-rigid-body method for forward dynamics, which uses inverse dynamics as a building block, and in chapter 8 I deal with extensions to the basic algorithm to cater for floating bases, branched kinematic chains, etc..

Figure 4.3 Forces acting on a link

Chapter 5

Forward Dynamics - The Composite-Rigid-Body Method

5.1 Introduction

In this chapter I describe the composite-rigid-body method for forward robot dynamics - the calculation of the acceleration produced by a given set of forces. Forward dynamics is used for simulation, so it is not necessary for it to execute at real-time speed, although efficient algorithms are still useful to minimise the cost of simulation. The composite-rigid-body algorithm is used in the simulators described in [Featherstone 1982; Featherstone 1983b; Swartz 1982]; for other approaches see [Thomas and Tesar 1982; Backhouse and Rees-Jones 1981; Benati et al. 1980].

The composite-rigid-body method is not new, and is described in [Walker and Orin 1982] as method 3.¹ The method is based on finding the force required to impart unit acceleration about joint i to links $i..n$ treated as a single composite rigid body, hence my choice of name. It is the most efficient currently known general purpose algorithm for performing forward dynamics on robots with a small number of joints; but for robots with a large number of joints it is slower

1. This paper first appeared as WP-2B in Proc. 1981 J.A.C.C., Charlottesville, VA. I developed the algorithm independently in 1980, but did not publish it.

than the articulated-body method, which is the subject of the next chapter. The crossover point comes at about $n=9$, so the composite-rigid-body algorithm is faster for most practical situations.

5.1.1 Forward Dynamics

A robot's equations of motion may be written in the form

$$\underline{Q} = H(\underline{q}) \underline{\ddot{q}} + \underline{C}(\underline{q}, \underline{\dot{q}}) , \quad (1)$$

where $\underline{Q}$ is the vector of generalised forces, $\underline{q}$, $\underline{\dot{q}}$ and $\underline{\ddot{q}}$ are the joint variables, velocities and accelerations, H is an $n \times n$ matrix whose elements are functions of $\underline{q}$ only (n being the number of joints), and $\underline{C}$ is a vector function of $\underline{q}$ and $\underline{\dot{q}}$. The term $H \underline{\ddot{q}}$ accounts for inertial acceleration forces, and $\underline{C}$ for Coriolis, centrifugal and gravitational forces. (Gravitational terms are often treated separately.) An equation of this form can be derived from both the Lagrange and Newton-Euler approaches. I refer to H as the joint-space inertia matrix.

The task of solving eq. (1) for $\underline{\ddot{q}}$ can be broken down into three parts: the calculation of H , the calculation of $\underline{C}$, and the solution of the simultaneous equations $H \underline{\ddot{q}} = \underline{Q} - \underline{C}$ for $\underline{\ddot{q}}$. In non-recursive formulations, each element of H and $\underline{C}$ is expressed as a sum or multiple sum of simple expressions (e.g. see [Luh et al. 1980a]). Their evaluation involves a lot of duplicated calculation. This is particularly so with the calculation of the elements of $\underline{C}$, each of which is an $O(n^3)$ operation. In the following sections I describe more efficient methods for calculating H and $\underline{C}$, culminating with the composite-rigid-body method, which is the most efficient known method

for calculating H .

5.2 Using an Inverse Dynamics Algorithm to Calculate Forward Dynamics

We can use an inverse dynamics algorithm (such as the one described in the previous chapter) to calculate both H and $\underline{C}$. With an $O(n)$ inverse dynamics algorithm the calculation of H can be achieved in $O(n^2)$ steps and $\underline{C}$ in $O(n)$. These are the minimum possible computational complexities, since H and $\underline{C}$ contain n^2 and n elements respectively. This approach produces a substantial increase in efficiency over non-recursive methods, particularly with the calculation of $\underline{C}$, but the calculation of H is not as fast as with the composite-rigid-body method.

5.2.1 Calculating $\underline{C}$

If we set the generalised acceleration vector in eq. (1) to zero, then the equation simplifies to:

$$\underline{Q} = \underline{C}(\underline{q}, \dot{\underline{q}}) . \quad (2)$$

The vector $\underline{C}$ may thus be interpreted as the vector of generalised forces which produces zero generalised acceleration, given the robot's current position, velocity, etc.. We can use an inverse dynamics algorithm to calculate $\underline{C}$ by asking it to calculate the forces required to produce zero acceleration. If we view the inverse dynamics algorithm as a function, IDA, taking $\underline{q}$, $\dot{\underline{q}}$ and $\ddot{\underline{q}}$ as arguments, then we may compute $\underline{C}$ by

$$\underline{C}(\underline{q}, \dot{\underline{q}}) = \text{IDA}(\underline{q}, \dot{\underline{q}}, \underline{0}) . \quad (3)$$

Any inverse dynamics algorithm will do, but obviously an efficient one is to be preferred. The result is that we can calculate $\underline{C}$ quickly and in $O(n)$ steps.

5.2.2 Calculating $\underline{H}$

Using the inverse dynamics function, we can calculate $\underline{Q}$ of eq. (1) for any acceleration vector of our choice, and we can also calculate $\underline{C}$, so we can calculate the product of $\underline{H}$ with any chosen vector as follows:

$$\begin{aligned} \text{IDA}(\underline{q}, \dot{\underline{q}}, \underline{u}) &= \underline{Q} \\ &= \underline{H} \underline{u} + \underline{C} \\ &= \underline{H} \underline{u} + \text{IDA}(\underline{q}, \dot{\underline{q}}, \underline{0}) \\ \longrightarrow \underline{H} \underline{u} &= \text{IDA}(\underline{q}, \dot{\underline{q}}, \underline{u}) - \text{IDA}(\underline{q}, \dot{\underline{q}}, \underline{0}) \\ &= \text{IDA2}(\underline{q}, \underline{u}) \quad (\text{since } \underline{H} \underline{u} \text{ is independent of } \dot{\underline{q}}). \end{aligned} \quad (4)$$

IDA2 is just a simplified version of IDA which ignores velocity effects, gravity, and anything else which is independent of the joint accelerations.

If we let $\underline{u} = \underline{\delta}_i$, where $\underline{\delta}_i$ is defined such that its i^{th} element is one and the rest are zero, then we can calculate column i of $\underline{H}$ from eq. (4); so we can calculate the whole of $\underline{H}$ with n iterations of eq. (4). This is Walker and Orin's first method for calculating $\underline{H}$.

Their second method is basically the same, but achieves a greater efficiency by noting that H is symmetric and arranging to calculate only the lower triangle.

Having calculated $\underline{C}$ and H , the final step is to solve the simultaneous equations

$$H \ddot{\underline{q}} = \underline{Q} - \underline{C} . \quad (5)$$

The complexity of this step is $O(n^3)$, thus making the complexity of the overall solution $O(n^3)$, but the coefficient of n^3 in the computational requirement is much smaller than that of any of the lower powers of n , and for values of n around $n=6$ (or even $n=12$) its contribution is insignificant. The largest part of the computational burden for small n goes in calculating H .

Since H is symmetric and positive definite, specialised techniques like the Cholesky decomposition can be used to solve the equations, though it is possible for H to become singular if some of the links are given zero (or singular) spatial inertias.

Walker and Orin also describe a method for calculating $\ddot{\underline{q}}$ which bypasses the need to calculate H explicitly (method 4). This method uses an iterative technique for solving eq. (5) that converges to an exact solution in a maximum of n iterations. At each iteration this method requires the value of a vector $H \underline{u}$, where $\underline{u}$ is chosen by the iteration algorithm. The complexity of this method is $O(n^2)$, but the coefficient of n^2 is large enough that it ends up being less efficient than the composite-rigid-body method for all but very large values of n (e.g. $n > 300$).

5.3 The Composite-Rigid-Body Method

This is a method for calculating the joint-space inertia matrix which is more efficient than any of the methods described in the previous section. It is described in [Walker and Orin 1982] as method 3. The method works by treating links $i..n$ as a composite rigid body which is accelerating about joint i , and uses the inertia of the composite body to calculate the forces involved.

It can be seen from eq. (4) that $H \underline{u}$ is the vector of joint forces required to impart a joint-space acceleration of $\underline{u}$ to a stationary robot which is free from gravitational or other external influences; in particular column i of H (which is $H \underline{\delta}_i$) is the vector of joint forces required to produce a unit acceleration about joint i only. Since no motion occurs about any of the joints $i+1..n$, they can all be replaced by rigid connections without having any effect on the dynamics lower down the chain. So the moving part of the robot simplifies to a single composite rigid body (see figure 5.1).

Figure 5.1 Acceleration about joint i only

Using the system model as described in chapter 4, let $\hat{I}_i^c$ be the inertia of the composite rigid body comprising links $i..n$, then

$$\hat{I}_i^c = \sum_{j=i}^n \hat{I}_j ,$$

which expressed as a recurrence relation is

$$\hat{I}_i^c = \hat{I}_{i+1}^c + \hat{I}_i : \quad (\hat{I}_n^c = \hat{I}_n) \quad (6)$$

The force required to give $\hat{I}_i^c$ an acceleration of $\hat{s}_i$ (i.e. unit acceleration about joint axis i) is

$$\hat{f}_i = \hat{I}_i^c \hat{s}_i . \quad (7)$$

Now links $1..i-1$ are in static equilibrium, so there is no net force acting on any of them. Therefore every joint from 1 to i transmits $\hat{f}_i$ on to the next link. So for each joint from 1 to i we have that the scalar joint force is the component of $\hat{f}_i$ on the appropriate joint axis, and these quantities are the first i elements of column i of H . Thus we have:

$$H_{ji} = \hat{s}_j^s \hat{f}_i , \quad (j \leq i) \quad (8)$$

which, making use of the symmetry of H , enables us to calculate the whole of the matrix H . An equation which shows the symmetry more clearly is

$$H_{ij} = H_{ji} = \hat{s}_j^s \hat{I}_r^c \hat{s}_i , \quad \text{where } r = \max(i,j) . \quad (9)$$

Eqs. (6-8) form the composite-rigid-body method for calculating the joint-space inertia matrix. To complete the forward dynamics algorithm we need to calculate $\underline{C}$ and to solve $H \ddot{\underline{q}} = \underline{Q} - \underline{C}$, both of which were covered in the previous section.

5.3.1 The Composite-Rigid-Body Method in Link Coordinates

The advantage of calculating the joint-space inertia matrix in link coordinates is less clear than was the case with the inverse dynamics algorithm of the previous chapter, since in absolute coordinates we need only to transform the quantities $\underline{\hat{s}}'_i$ and $\hat{I}'_i$ to serve for the calculation of both $\underline{C}$ and H , whereas in link coordinates we need to transform $\underline{\hat{f}}_i$ and $\hat{I}^c_i$ (the former involving $O(n^2)$ transformations) in addition to the transformations carried out by the inverse dynamics algorithm. Nevertheless, the link coordinate version is faster for $n \leq 16$ (see chapter 7).

The link coordinate version of the algorithm is obtained as described in chapter 4, except that since $\underline{\hat{f}}_i$ is needed in several coordinate systems we replace it with $\underline{\hat{f}}_{ij}$ where the second subscript indicates the coordinate system in which it is represented. The link coordinate version is as follows:

$$\hat{I}^c_i = {}_i\hat{X}_{i+1} \hat{I}^c_{i+1} {}_i\hat{X}_i + \hat{I}'_i, \quad (\hat{I}^c_n = \hat{I}'_n) \quad (10)$$

$$\underline{\hat{f}}_{ii} = \hat{I}^c_i \underline{\hat{s}}'_i, \quad (11)$$

$$\underline{\hat{f}}_{ij} = {}_j\hat{X}_{j+1} \underline{\hat{f}}_{ij+1}, \quad (j < i) \quad (12)$$

$$H_{ij} = H_{ji} = \underline{\hat{s}}'^s_j \underline{\hat{f}}_{ij}, \quad (j \leq i) \quad (13)$$

This is a good example of the conciseness of spatial notation compared with conventional vector notation (cf. [Walker and Orin 1982]).

5.3.2 The Lagrange Approach to the Composite-Rigid-Body Algorithm

In this section I show briefly how eq. (9) may be obtained from the Lagrange equation. This is how I originally developed the algorithm. From eq. (1) it follows that

$$H_{ij} = \frac{\partial Q_i}{\partial \dot{q}_j} . \quad (14)$$

Lagrange's equation for Q_i is

$$Q_i = \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} , \quad (15)$$

where the Lagrangian is given by

$$\begin{aligned} L &= KE - PE \\ &= \sum_{k=1}^n \frac{1}{2} \hat{\underline{v}}_k^s \hat{\underline{I}}_k \hat{\underline{v}}_k - PE . \end{aligned} \quad (16)$$

So H_{ij} is given by

$$\begin{aligned} H_{ij} &= \frac{\partial}{\partial \dot{q}_j} \left(\frac{d}{dt} \frac{\partial}{\partial \dot{q}_i} \left(\sum_{k=1}^n \frac{1}{2} \hat{\underline{v}}_k^s \hat{\underline{I}}_k \hat{\underline{v}}_k - PE \right) - \frac{\partial L}{\partial q_i} \right) \\ &= \frac{\partial}{\partial \dot{q}_j} \left(\frac{d}{dt} \left(\sum_{k=1}^n \hat{\underline{s}}_i^s \hat{\underline{I}}_k \hat{\underline{v}}_k \right) - \frac{\partial L}{\partial q_i} \right) \\ &= \frac{\partial}{\partial \dot{q}_j} \left(\sum_{k=1}^n \left((\hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i)^s \hat{\underline{I}}_k \hat{\underline{v}}_k + \hat{\underline{s}}_i^s (\hat{\underline{I}}_k \hat{\underline{a}}_k + \hat{\underline{v}}_k \hat{\times} \hat{\underline{I}}_k \hat{\underline{v}}_k) \right) - \frac{\partial L}{\partial q_i} \right) \\ &= \sum_{k=r}^n \hat{\underline{s}}_i^s \hat{\underline{I}}_k \hat{\underline{s}}_j \quad (r = \max(i, j)) \\ &= \hat{\underline{s}}_i^s \hat{\underline{I}}_r^c \hat{\underline{s}}_j . \end{aligned} \quad (17)$$

Chapter 6

Forward Dynamics - The Articulated-Body Method

6.1 Introduction

In this chapter I describe the articulated-body method for forward dynamics - an alternative to the composite-rigid-body method. This method has a computational complexity of $O(n)$ as compared with $O(n^3)$ for the composite-rigid-body method, though it does not become faster than the composite-rigid-body method until n reaches a value of approximately 9.

The articulated-body method is new, though an early version is described in [Featherstone 1983a]. The main difference between the versions described here and in that paper is that the version described here solves the forward dynamics problem in its entirety, whereas the version described in the paper neglects velocity dependent terms and assumes that the forces required to compensate for them have been calculated by an inverse dynamics algorithm.

The articulated-body method uses a different approach to solving forward dynamics from the methods described in the previous chapter. The basic idea is to regard the robot as a one-joint robot whose only moving link is in fact an articulated body representing the rest of the robot. The forward dynamics problem for this one-joint robot is

easily solved once the apparent inertia of the moving link is known. Once the acceleration of the first joint is known, the process can be repeated for the second and subsequent joints. So the articulated-body method consists of the calculation of a series of articulated-body inertias which are used to solve the forward dynamics one joint at a time.

The articulated-body method is quite unlike any other published robot dynamics algorithm,¹ though it does bear some resemblance to Armstrong's method [Armstrong 1979]. This is another $O(n)$ algorithm for calculating forward dynamics, which uses recursion coefficients playing a similar role to articulated-body inertias, but the approach is somewhat different and in its basic form is only applicable to robots with spherical joints.

6.1.1 Articulated-Body Inertias

Articulated-body inertias were described in chapter 3, so here I will just go over the important points.

A collection of rigid bodies connected by joints is called an articulated body. To define an articulated-body inertia, we single out a particular member of the articulated body, called the handle, and define the articulated-body inertia as the relationship between a test force $\hat{f}$ applied to the handle and the acceleration $\hat{a}$ of the handle according to

1. See the note at the end of this chapter.

$$\underline{\hat{f}} = \hat{I}^A \underline{\hat{a}} + \underline{\hat{p}} . \quad (1)$$

$\hat{I}^A$ is the articulated-body inertia and $\underline{\hat{p}}$ is the associated bias force, which is the value of the test force which must be applied to the handle in order to give it zero acceleration. It is not always possible to express the relationship between $\underline{\hat{f}}$ and $\underline{\hat{a}}$ in the form of eq. (1), but a sufficient condition is that there be no kinematic connection with ground, i.e. that the articulated body is a 'floating' system. The articulated bodies used in the articulated-body method all satisfy this condition.

The equation of motion of a single rigid body can also be written in the form of eq. (1), in which case $\hat{I}^A$ is its rigid-body inertia and $\underline{\hat{p}}$ takes the place of the velocity product term. It is therefore possible to replace a member of a system of rigid bodies with the handle of an articulated body without affecting the equations of motion of the system. This works provided that the handle is the only point of interaction between the system and the articulated body. However, the real usefulness of articulated-body inertias lies in their power to 'simplify' systems of rigid bodies by allowing several members to be treated as one articulated body, so reducing the apparent number of members in the system.

In the following sections I describe first the outline of the articulated-body method, then how to calculate articulated-body inertias, and I finish with a detailed account of the articulated-body method.

6.2 The Articulated-Body Method in Outline

The basic idea of the articulated-body algorithm is to treat an n -joint robot as a one-joint robot whose only moving link is in fact the handle of an articulated body comprising all the remaining links of the robot. To find the acceleration of the first joint requires the solution of the forward dynamics of a one-joint robot, which is much easier than the general case. Having solved for the acceleration of joint 1, we can treat link 1 as the (moving) base of an $n-1$ joint robot and repeat the process for joint 2, and so on.

So to carry out the articulated-body method we need two capabilities: the ability to calculate articulated-body inertias, and the ability to solve the dynamics of a one-joint robot with a moving base. I leave the problem of how to calculate articulated-body inertias to the next section and concentrate here on the forward dynamics problem. We could use one of the methods of chapter 5 to solve this problem, but the one-joint case is so simple that it is easily solved directly.

Let a one-joint robot have a base moving with velocity $\hat{\underline{v}}$ and acceleration $\hat{\underline{a}}$, a link with inertia $\hat{\underline{I}}$, a joint with axis $\hat{\underline{s}}$, and let the joint velocity, acceleration and force be $\dot{q}$, $\ddot{q}$ and Q respectively (see figure 6.1). All quantities are measured in absolute coordinates. The velocity and acceleration of the link are given by

$$\hat{\underline{v}}_1 = \hat{\underline{v}} + \hat{\underline{s}} \dot{q} \quad (2)$$

Figure 6.1 One-joint robot with moving base

and

$$\hat{a}_1 = \hat{a} + \hat{v} \times \hat{s} \dot{q} + \hat{s} \ddot{q}, \quad (3)$$

and the equation of motion of the link is

$$\hat{f} = \hat{I} \hat{a}_1 + \hat{v}_1 \times \hat{I} \hat{v}_1, \quad (4)$$

where $\hat{f}$ is the net force applied to the link (ignoring gravity). $\hat{f}$ is applied via the joint, so its component on the joint axis is Q , i.e.

$$\hat{s}^S \hat{f} = Q. \quad (5)$$

Substituting eqs. (3 and 4) into eq. (5), we obtain an expression for $\ddot{q}$ in terms of Q :

$$\begin{aligned} Q &= \hat{s}^S (\hat{I} \hat{a}_1 + \hat{v}_1 \times \hat{I} \hat{v}_1) \\ &= \hat{s}^S (\hat{I} (\hat{a} + \hat{v} \times \hat{s} \dot{q} + \hat{s} \ddot{q}) + \hat{v}_1 \times \hat{I} \hat{v}_1) \\ \rightarrow \ddot{q} &= \frac{Q - \hat{s}^S (\hat{I} (\hat{a} + \hat{v} \times \hat{s} \dot{q}) + \hat{v}_1 \times \hat{I} \hat{v}_1)}{\hat{s}^S \hat{I} \hat{s}} \end{aligned} \quad (6)$$

Eq. (6) is the equation of motion for a one-joint robot with a moving base.

If we replace the single rigid body of the link by an articulated body comprising links 1..n of an n-joint robot, where link 1 is the handle, and we are able to calculate the articulated-body inertia $\hat{I}^A$ and the bias force $\hat{p}$, then we can replace eq. (4) with

$$\hat{f} = \hat{I}^A \hat{a}_1 + \hat{p} \quad (7)$$

and the equation of motion becomes

$$\ddot{q} = \frac{Q - \hat{s}^S (\hat{I}^A (\hat{a} + \hat{v} \times \hat{s} \dot{q}) + \hat{p})}{\hat{s}^S \hat{I}^A \hat{s}} \quad (8)$$

Articulated-body inertias are positive definite (see chapter 3) so the denominator is always greater than zero.

Eq. (8) is the equation of motion of the first joint of an n-joint robot. To solve for the motion of the rest of the robot, treat link 1 as the moving base of an n-1 joint robot, since we now know its velocity and acceleration, and repeat the whole process. n iterations will solve the problem completely.

6.3 The Calculation of Articulated-Body Inertias

Let us start by considering a simple articulated body comprising two members. Calculation methods for more complicated articulated bodies may be obtained by straightforward generalisations of the solution for this particular example.

Figure 6.2 Simple articulated body

Let the articulated body comprise two members, b_1 and b_2 , having inertias of $\hat{I}_1$ and $\hat{I}_2$ respectively and connected by a joint with axis $\hat{s}$ (see figure 6.2). The bodies are initially at rest and experience no applied force other than the test force, so the bias force will be zero in this case. A test force $\hat{f}$ is applied to b_1 , which is chosen as the handle, causing accelerations of $\hat{a}_1$ and $\hat{a}_2$ in b_1 and b_2 respectively.

Let $\hat{I}_1^A$ be the articulated-body inertia of b_1 in the articulated body comprising b_1 and b_2 , then

$$\hat{f} = \hat{I}_1^A \hat{a}_1. \quad (9)$$

$\hat{f}$ can be split into two components, $\hat{f}_1$ and $\hat{f}_2$, where $\hat{f}_1$ is the net force acting on b_1 and $\hat{f}_2$ is the force transmitted to b_2 through the joint bearings. The equations of motion for the individual bodies are then

$$\hat{f}_1 = \hat{I}_1 \hat{a}_1 \quad (10)$$

and

$$\hat{\underline{f}}_2 = \hat{\underline{I}}_2 \hat{\underline{a}}_2 , \quad (11)$$

and we also have that

$$\hat{\underline{f}} = \hat{\underline{f}}_1 + \hat{\underline{f}}_2 . \quad (12)$$

Now the joint imposes constraints on both $\hat{\underline{a}}_2$ and $\hat{\underline{f}}_2$ which will allow us to express them in terms of $\hat{\underline{a}}_1$, and hence allow us to find $\hat{\underline{f}}$ in terms of $\hat{\underline{a}}_1$. $\hat{\underline{a}}_2$ is given by

$$\hat{\underline{a}}_2 = \hat{\underline{a}}_1 + \hat{\underline{s}} \alpha , \quad (13)$$

where α is the scalar joint acceleration, and $\hat{\underline{f}}_2$ does no work in the direction of motion of the joint, so

$$\hat{\underline{s}}^S \hat{\underline{f}}_2 = 0 . \quad (14)$$

We can find α in terms of $\hat{\underline{a}}_1$ by substituting eqs. (11 and 13) into eq. (14), giving

$$\begin{aligned} \hat{\underline{s}}^S \hat{\underline{I}}_2 (\hat{\underline{a}}_1 + \hat{\underline{s}} \alpha) &= 0 \\ \rightarrow \alpha &= - \frac{\hat{\underline{s}}^S \hat{\underline{I}}_2 \hat{\underline{a}}_1}{\hat{\underline{s}}^S \hat{\underline{I}}_2 \hat{\underline{s}}} . \end{aligned} \quad (15)$$

Once we know α , we can find $\hat{\underline{f}}$ in terms of $\hat{\underline{a}}_1$:

$$\begin{aligned} \hat{\underline{f}} &= \hat{\underline{I}}_1 \hat{\underline{a}}_1 + \hat{\underline{I}}_2 \hat{\underline{a}}_2 \\ &= \hat{\underline{I}}_1 \hat{\underline{a}}_1 + \hat{\underline{I}}_2 (\hat{\underline{a}}_1 - \hat{\underline{s}} \frac{\hat{\underline{s}}^S \hat{\underline{I}}_2 \hat{\underline{a}}_1}{\hat{\underline{s}}^S \hat{\underline{I}}_2 \hat{\underline{s}}}) \\ &= (\hat{\underline{I}}_1 + \hat{\underline{I}}_2 - \frac{\hat{\underline{I}}_2 \hat{\underline{s}} \hat{\underline{s}}^S \hat{\underline{I}}_2}{\hat{\underline{s}}^S \hat{\underline{I}}_2 \hat{\underline{s}}}) \hat{\underline{a}}_1 . \end{aligned} \quad (16)$$

Eq. (16) has the same form as eq. (9), and since both are valid for

all $\hat{\underline{f}}$ it follows that

$$\hat{\underline{I}}_1^A = \hat{\underline{I}}_1 + \hat{\underline{I}}_2 - \frac{\hat{\underline{I}}_2 \hat{\underline{s}} \hat{\underline{s}}^S \hat{\underline{I}}_2}{\hat{\underline{s}}^S \hat{\underline{I}}_2 \hat{\underline{s}}} . \quad (17)$$

Note two interesting points about eq. (17). Firstly, it is a symmetry preserving equation, that is, if $\hat{\underline{I}}_1$ and $\hat{\underline{I}}_2$ are symmetric (which they are) then $\hat{\underline{I}}_1^A$ is also symmetric. Secondly, we can use it to show that $\hat{\underline{I}}_1^A$ is bounded below by $\hat{\underline{I}}_1$ and bounded above by $\hat{\underline{I}}_1 + \hat{\underline{I}}_2$ in the sense that for any vector $\hat{\underline{x}}$ we have

$$\hat{\underline{x}}^S \hat{\underline{I}}_1 \hat{\underline{x}} \leq \hat{\underline{x}}^S \hat{\underline{I}}_1^A \hat{\underline{x}} \leq \hat{\underline{x}}^S (\hat{\underline{I}}_1 + \hat{\underline{I}}_2) \hat{\underline{x}} .$$

This is a stronger statement than merely saying that $\hat{\underline{I}}_1^A$ is positive definite, and is proved by letting $\hat{\underline{x}} = \hat{\underline{x}}_1 + \hat{\underline{x}}_2$ where $\hat{\underline{x}}_1 = \hat{\underline{x}}_1 \hat{\underline{s}}$ and $\hat{\underline{x}}_2^S \hat{\underline{I}}_2 \hat{\underline{s}} = 0$.

6.3.1 Allowing for Velocities and Active Joint Forces

Now let us consider what happens when we allow the bodies to have non-zero velocities and allow an active force to be present at the joint. In this case the bias force will be non-zero, so we must replace eq. (9) by the inhomogeneous version

$$\hat{\underline{f}} = \hat{\underline{I}}_1^A \hat{\underline{a}}_1 + \hat{\underline{p}}_1 . \quad (18)$$

Let b_1 and b_2 have velocities $\hat{\underline{v}}_1$ and $\hat{\underline{v}}_2$ respectively (consistent with the joint constraints), then velocity product terms appear in the equations of motion of b_1 and b_2 , which become

$$\hat{\underline{f}}_1 = \hat{\underline{I}}_1 \hat{\underline{a}}_1 + \hat{\underline{p}}_1^v , \quad (19)$$

$$\hat{\underline{f}}_2 = \hat{\underline{I}}_2 \hat{\underline{a}}_2 + \hat{\underline{p}}_2^v, \quad (20)$$

where $\hat{\underline{p}}_1^v$ and $\hat{\underline{p}}_2^v$ are the bias forces due to velocity product effects and are given by

$$\hat{\underline{p}}_1^v = \hat{\underline{v}}_1 \hat{\times} \hat{\underline{I}}_1 \hat{\underline{v}}_1, \quad (21)$$

$$\hat{\underline{p}}_2^v = \hat{\underline{v}}_2 \hat{\times} \hat{\underline{I}}_2 \hat{\underline{v}}_2. \quad (22)$$

It is convenient to extend the notion of bias force to rigid bodies.

A velocity product term also appears in eq. (13) which becomes

$$\hat{\underline{a}}_2 = \hat{\underline{a}}_1 + \hat{\underline{v}}_1 \hat{\times} \hat{\underline{v}}_2 + \hat{\underline{s}} \alpha. \quad (23)$$

If we let Q be the active joint force then this appears in eq. (14), which becomes

$$\hat{\underline{s}}^s \hat{\underline{f}}_2 = Q. \quad (24)$$

The equations are now a bit more complicated, but we follow exactly the same line of reasoning as before, ending up with the following equations:

$$\hat{\underline{I}}_1^A = \hat{\underline{I}}_1 + \hat{\underline{I}}_2 - \frac{\hat{\underline{I}}_2 \hat{\underline{s}} \hat{\underline{s}}^s \hat{\underline{I}}_2}{\hat{\underline{s}}^s \hat{\underline{I}}_2 \hat{\underline{s}}}, \quad (25)$$

$$\hat{\underline{p}}_1 = \hat{\underline{p}}_1^v + \hat{\underline{p}}_2^v + \hat{\underline{I}}_2 \left(\hat{\underline{v}}_1 \hat{\times} \hat{\underline{v}}_2 + \hat{\underline{s}} \frac{Q - \hat{\underline{s}}^s (\hat{\underline{I}}_2 \hat{\underline{v}}_1 \hat{\times} \hat{\underline{v}}_2 + \hat{\underline{p}}_2^v)}{\hat{\underline{s}}^s \hat{\underline{I}}_2 \hat{\underline{s}}} \right). \quad (26)$$

Eq. (25) is the same as eq. (17), demonstrating that articulated-body inertias do not depend on the forces present in the system. Eq. (26) looks complicated, but its evaluation need not be as time-consuming as it might seem since there is some duplication of expressions both within eq. (26) and between eqs. (26 and 25).

6.3.2 More General Articulated Bodies

The inertia of a more general articulated body can be obtained by replacing the rigid bodies b_1 and b_2 in the example with the handles of two (independent) articulated bodies. Eqs. (25 and 26) then become the general formulae for calculating the inertia of a larger articulated body recursively in terms of the inertias of two smaller articulated bodies. An articulated-body inertia for an articulated body containing n joints can be calculated in n iterations of eqs. (25 and 26), one per joint.

This process is limited to the calculation of inertias for articulated bodies free from kinematic loops. The presence of kinematic loops complicates matters considerably, and I defer consideration of this problem until the end of chapter 9.

6.4 The Articulated-Body Dynamics Algorithm

I come now to the detailed description of the articulated-body method for forward dynamics, using the system model as defined in chapter 4. This method consists of two steps:

- 1) the calculation of a series of articulated-body inertias, one for each link, and
- 2) using the articulated-body inertias to calculate the joint accelerations.

Figure 6.3 Articulated body i

Step 1: Calculating the articulated-body inertias

Define articulated body i as the articulated body comprising links i..n of the robot, connected by joints i+1..n (see figure 6.3). Link i is always the handle of articulated body i. Let $\hat{I}_i^A$ and $\hat{p}_i$ be the articulated-body inertia and associated bias force of link i in articulated body i. Our target is to find a pair of recurrence relations which will enable us to calculate $\hat{I}_i^A$ and $\hat{p}_i$ from $\hat{I}_{i+1}^A$ and $\hat{p}_{i+1}$.

Since velocity effects are to be included in $\hat{p}_i$, we must start by calculating the velocities and velocity product forces at each link in turn. Let $\hat{v}_i$ be the absolute velocity of link i and let $\hat{p}_i^v$ be the bias force on link i due to velocity product forces, then

$$\hat{v}_i = \hat{v}_{i-1} + \hat{s}_i \dot{q}_i \quad (\hat{v}_0 = \hat{0}) \quad (27)$$

and

$$\hat{p}_i^v = \hat{v}_i \hat{\times} \hat{I}_i \hat{v}_i. \quad (28)$$

Terms due to external forces acting on link i may also be included in

eq. (28). The equation of motion for link i then takes the form

$$\underline{\hat{f}} = \underline{\hat{I}}_i \underline{\hat{a}} + \underline{\hat{p}}_i^v .$$

Apply a test force $\underline{\hat{f}}$ to link i in articulated body i , producing an acceleration of $\underline{\hat{a}}_1$ in link i and $\underline{\hat{a}}_2$ in link $i+1$. Splitting $\underline{\hat{f}}$ into components $\underline{\hat{f}}_1$ and $\underline{\hat{f}}_2$, where $\underline{\hat{f}}_1$ acts on link i and $\underline{\hat{f}}_2$ is passed on to link $i+1$, and treating link $i+1$ as the handle of articulated body $i+1$, we have the following relations:

$$\underline{\hat{f}} = \underline{\hat{I}}_i^A \underline{\hat{a}}_1 + \underline{\hat{p}}_i , \quad (29)$$

$$\underline{\hat{f}}_1 = \underline{\hat{I}}_i \underline{\hat{a}}_1 + \underline{\hat{p}}_i^v , \quad (30)$$

$$\underline{\hat{f}}_2 = \underline{\hat{I}}_{i+1}^A \underline{\hat{a}}_2 + \underline{\hat{p}}_{i+1} \quad (31)$$

and

$$\underline{\hat{f}} = \underline{\hat{f}}_1 + \underline{\hat{f}}_2 . \quad (32)$$

Joint $i+1$ constrains $\underline{\hat{a}}_2$ such that it can be described in terms of $\underline{\hat{a}}_1$ and a single unknown scalar, and imposes one degree of constraint on $\underline{\hat{f}}_2$. $\underline{\hat{a}}_2$ is given by

$$\underline{\hat{a}}_2 = \underline{\hat{a}}_1 + \underline{\hat{v}}_{i+1} \hat{x} \underline{\hat{s}}_{i+1} \dot{q}_{i+1} + \underline{\hat{s}}_{i+1} \alpha , \quad (33)$$

where α is the unknown joint acceleration, and the component of $\underline{\hat{f}}_2$ on the axis of joint $i+1$ is Q_{i+1} , i.e.

$$\underline{\hat{s}}_{i+1}^S \underline{\hat{f}}_2 = Q_{i+1} . \quad (34)$$

We need to find α in terms of $\underline{\hat{a}}_1$. This can be done by substituting eqs. (31) and (33) into eq. (34), giving

$$\underline{\hat{s}}_{i+1}^S (\underline{\hat{I}}_{i+1}^A (\underline{\hat{a}}_1 + \underline{\hat{v}}_{i+1} \hat{x} \underline{\hat{s}}_{i+1} \dot{q}_{i+1} + \underline{\hat{s}}_{i+1} \alpha) + \underline{\hat{p}}_{i+1}) = Q_{i+1}$$

$$\rightarrow \alpha = \frac{Q_{i+1} - \hat{s}_{i+1}^S (\hat{I}_{i+1}^A (\hat{a}_1 + \hat{v}_{i+1} \hat{x} \hat{s}_{i+1} \dot{q}_{i+1}) + \hat{p}_{i+1})}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}} \quad (35)$$

Now we know α , we can express $\hat{a}_2$ in terms of $\hat{a}_1$ and hence we can express $\hat{f}$ in terms of $\hat{a}_1$ using eqs. (30-33) as follows:

$$\begin{aligned} \hat{f} &= \hat{I}_i \hat{a}_1 + \hat{p}_i^v + \hat{I}_{i+1}^A \hat{a}_2 + \hat{p}_{i+1} \\ &= \hat{I}_i \hat{a}_1 + \hat{p}_i^v + \hat{I}_{i+1}^A (\hat{a}_1 + \hat{v}_{i+1} \hat{x} \hat{s}_{i+1} \dot{q}_{i+1} + \hat{s}_{i+1} \alpha) + \hat{p}_{i+1} \\ &= \hat{I}_i \hat{a}_1 + \hat{p}_i^v + \hat{p}_{i+1} + \hat{I}_{i+1}^A \left(\hat{a}_1 + \hat{v}_{i+1} \hat{x} \hat{s}_{i+1} \dot{q}_{i+1} \right. \\ &\quad \left. + \hat{s}_{i+1} \frac{Q_{i+1} - \hat{s}_{i+1}^S (\hat{I}_{i+1}^A (\hat{a}_1 + \hat{v}_{i+1} \hat{x} \hat{s}_{i+1} \dot{q}_{i+1}) + \hat{p}_{i+1})}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}} \right) \\ &= \left(\hat{I}_i + \hat{I}_{i+1}^A - \frac{\hat{I}_{i+1}^A \hat{s}_{i+1} \hat{s}_{i+1}^S \hat{I}_{i+1}^A}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}} \right) \hat{a}_1 \\ &\quad + \hat{p}_i^v + \hat{p}_{i+1} + \hat{I}_{i+1}^A \hat{v}_{i+1} \hat{x} \hat{s}_{i+1} \dot{q}_{i+1} \\ &\quad + \frac{\hat{I}_{i+1}^A \hat{s}_{i+1} (Q_{i+1} - \hat{s}_{i+1}^S (\hat{I}_{i+1}^A \hat{v}_{i+1} \hat{x} \hat{s}_{i+1} \dot{q}_{i+1} + \hat{p}_{i+1}))}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}} \quad (36) \end{aligned}$$

On comparing eq. (36) with eq. (29) we get the following expressions for $\hat{I}_i^A$ and $\hat{p}_i$:

$$\hat{I}_i^A = \hat{I}_i + \hat{I}_{i+1}^A - \frac{\hat{I}_{i+1}^A \hat{s}_{i+1} \hat{s}_{i+1}^S \hat{I}_{i+1}^A}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}}, \quad (37)$$

$$\begin{aligned} \hat{p}_i &= \hat{p}_i^v + \hat{p}_{i+1} + \hat{I}_{i+1}^A \hat{v}_{i+1} \hat{x} \hat{s}_{i+1} \dot{q}_{i+1} \\ &\quad + \frac{\hat{I}_{i+1}^A \hat{s}_{i+1} (Q_{i+1} - \hat{s}_{i+1}^S (\hat{I}_{i+1}^A \hat{v}_{i+1} \hat{x} \hat{s}_{i+1} \dot{q}_{i+1} + \hat{p}_{i+1}))}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}} \quad (38) \end{aligned}$$

The starting values for these recurrence relations are simply

$$\hat{\underline{I}}_n^A = \hat{\underline{I}}_n \quad (39)$$

and

$$\hat{\underline{p}}_n = \hat{\underline{p}}_n^V, \quad (40)$$

since articulated body n is just a single rigid body.

Step 2: Calculating the joint accelerations

Having found all the articulated-body inertias, we can now start calculating the joint accelerations. At the i^{th} stage of this computation, link $i-1$ is treated as the moving base of a one-joint robot and articulated body i as its only apparent link. So, if we let $\hat{\underline{a}}_i$ be the absolute acceleration of link i , we calculate $\ddot{q}_i$ assuming that $\hat{\underline{a}}_{i-1}$ is known, which then allows us to calculate $\hat{\underline{a}}_i$ ready for the next iteration. $\hat{\underline{a}}_i$ is given in terms of $\hat{\underline{a}}_{i-1}$ by

$$\hat{\underline{a}}_i = \hat{\underline{a}}_{i-1} + \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{q}_i + \hat{\underline{s}}_i \ddot{q}_i, \quad (41)$$

and the total force, $\hat{\underline{f}}$, transmitted through link i is related to $\hat{\underline{a}}_i$ by

$$\begin{aligned} \hat{\underline{f}} &= \hat{\underline{I}}_i^A \hat{\underline{a}}_i + \hat{\underline{p}}_i \\ &= \hat{\underline{I}}_i^A (\hat{\underline{a}}_{i-1} + \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{q}_i + \hat{\underline{s}}_i \ddot{q}_i) + \hat{\underline{p}}_i. \end{aligned} \quad (42)$$

The component of $\hat{\underline{f}}$ on the axis of joint i is Q_i , so we have

$$\hat{\underline{s}}_i^S \hat{\underline{f}} = Q_i$$

$$\begin{aligned}
&\rightarrow \hat{\underline{s}}_i^S (\hat{\underline{I}}_i^A (\hat{\underline{a}}_{i-1} + \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{\underline{q}}_i + \hat{\underline{s}}_i \ddot{\underline{q}}_i) + \hat{\underline{p}}_i) = Q_i \\
&\rightarrow \ddot{\underline{q}}_i = \frac{Q_i - \hat{\underline{s}}_i^S (\hat{\underline{I}}_i^A (\hat{\underline{a}}_{i-1} + \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{\underline{q}}_i) + \hat{\underline{p}}_i)}{\hat{\underline{s}}_i^S \hat{\underline{I}}_i^A \hat{\underline{s}}_i} .
\end{aligned} \tag{43}$$

Substituting the value for $\ddot{\underline{q}}_i$ back into eq. (41) gives us $\hat{\underline{a}}_i$ for the next iteration, and the starting value for eq. (41) is

$$\hat{\underline{a}}_0 = \hat{\underline{0}} . \tag{44}$$

Gravity may be catered for by giving the base a fictitious acceleration of $-\hat{\underline{g}}$, where $\hat{\underline{g}}$ is the gravitational acceleration vector, in which case the starting value for eq. (41) is

$$\hat{\underline{a}}_0 = -\hat{\underline{g}} . \tag{45}$$

6.4.1 Common Sub-Expressions

If the articulated-body algorithm is carried out exactly in accordance with the above equations then a significant amount of duplicate calculation is performed due to the presence of common sub-expressions both within and between the equations. This repetition may be eliminated by introducing the following partial results which should be computed once and stored for later use:

$$\hat{\underline{c}}_i = \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{\underline{q}}_i , \tag{46}$$

$$\hat{\underline{h}}_i = \hat{\underline{I}}_i^A \hat{\underline{s}}_i , \tag{47}$$

$$d_i = \hat{\underline{s}}_i^S \hat{\underline{h}}_i \tag{48}$$

and

$$\underline{u}_i = \underline{Q}_i - \underline{\hat{h}}_i^s \underline{\hat{c}}_i - \underline{\hat{s}}_i^s \underline{\hat{p}}_i . \quad (49)$$

In terms of these partial results, the equations of the articulated-body algorithm become:

$$\underline{\hat{v}}_i = \underline{\hat{v}}_{i-1} + \underline{\hat{s}}_i \dot{\underline{q}}_i , \quad (\underline{\hat{v}}_0 = \underline{\hat{0}}) \quad (50)$$

$$\underline{\hat{p}}_i^v = \underline{\hat{v}}_i \hat{\times} \underline{\hat{I}}_i \underline{\hat{v}}_i , \quad (51)$$

$$\underline{\hat{I}}_i^A = \underline{\hat{I}}_i + \underline{\hat{I}}_{i+1}^A - \frac{\underline{\hat{h}}_{i+1} \underline{\hat{h}}_{i+1}^s}{d_{i+1}} , \quad (\underline{\hat{I}}_n^A = \underline{\hat{I}}_n) \quad (52)$$

$$\underline{\hat{p}}_i = \underline{\hat{p}}_i^v + \underline{\hat{p}}_{i+1} + \underline{\hat{I}}_{i+1}^A \underline{\hat{c}}_{i+1} + \frac{u_{i+1}}{d_{i+1}} \underline{\hat{h}}_{i+1} , \quad (\underline{\hat{p}}_n = \underline{\hat{p}}_n^v) \quad (53)$$

$$\underline{\hat{a}}_i = \underline{\hat{a}}_{i-1} + \underline{\hat{c}}_i + \underline{\hat{s}}_i \ddot{\underline{q}}_i , \quad (\underline{\hat{a}}_0 = \underline{\hat{0}}) \quad (54)$$

$$\ddot{\underline{q}}_i = \frac{u_i - \underline{\hat{h}}_i^s \underline{\hat{a}}_{i-1}}{d_i} . \quad (55)$$

6.4.2 Computation in Link Coordinates

There is a slight benefit from performing this computation in link coordinates, though it depends entirely on the greater scope for optimisation in link coordinates than in absolute coordinates (see chapter 7). The conversion to link coordinates is effected by straightforward application of the rules laid down in chapter 4, and result in the following equations:

$$\underline{\hat{c}}_i = \underline{\hat{v}}_i \hat{\times} \underline{\hat{s}}_i' \dot{\underline{q}}_i , \quad (56)$$

$$\underline{\hat{h}}_i = \underline{\hat{I}}_i^A \underline{\hat{s}}_i' , \quad (57)$$

$$d_i = \underline{\hat{s}}_i'^s \underline{\hat{h}}_i , \quad (58)$$

$$u_i = Q_i - \hat{h}_i^s \hat{c}_i - \hat{s}_i^s \hat{p}_i, \quad (59)$$

$$\hat{v}_i = {}_i\hat{X}_{i-1} \hat{v}_i + \hat{s}_i' \dot{q}_i, \quad (\hat{v}_0 = \hat{o}) \quad (60)$$

$$\hat{p}_i^v = \hat{v}_i \hat{\otimes} \hat{I}_i' \hat{v}_i, \quad (61)$$

$$\hat{I}_i^A = \hat{I}_i' + {}_i\hat{X}_{i+1} \left(\hat{I}_{i+1}^A - \frac{\hat{h}_{i+1} \hat{h}_{i+1}^s}{d_{i+1}} \right) {}_{i+1}\hat{X}_i, \quad (\hat{I}_n^A = \hat{I}_n') \quad (62)$$

$$\hat{p}_i = \hat{p}_i^v + {}_i\hat{X}_{i+1} \left(\hat{p}_{i+1} + \hat{I}_{i+1}^A \hat{c}_{i+1} + \frac{u_{i+1}}{d_{i+1}} \hat{h}_{i+1} \right), \quad (\hat{p}_n = \hat{p}_n^v) \quad (63)$$

$$\hat{a}_i = {}_i\hat{X}_{i-1} \hat{a}_{i-1} + \hat{c}_i + \hat{s}_i' \ddot{q}_i, \quad (\hat{a}_0 = \hat{o}) \quad (64)$$

$$\ddot{q}_i = \frac{u_i - \hat{h}_i^s {}_i\hat{X}_{i-1} \hat{a}_{i-1}}{d_i}. \quad (65)$$

Note: Since writing this thesis it has been brought to my attention that an algorithm described by Vereshchagin ("Computer Simulation of the Dynamics of Complicated Mechanisms of Robot-Manipulators", Engineering Cybernetics, no. 6, pp. 65-70, 1974) is practically identical to the articulated-body algorithm, although it is derived using a different approach.

Chapter 7

Coordinate Systems and Efficiency

7.1 Introduction

In this chapter I deal with the details of the efficient computer implementation of the algorithms presented in the last three chapters. In particular, I cover the choice of link coordinate systems, the implementation of efficient spatial arithmetic, and a comparison of the computational requirements of the algorithms. These details have been deliberately omitted from earlier chapters to avoid cluttering the descriptions of the algorithms.

I do not describe the implementation of any simulator or control system based on these algorithms, nor any programs or program fragments illustrating the implementation of the algorithms in terms of computer code.

7.2 The Placement of Link Coordinates

The robot system model described in chapter 4 states that there is a coordinate system attached to each link, and refers to the one attached to link i as i -coordinates. No indication is given as to where it should be placed relative to the link. Any placement is

permissible, but certain placements are computationally more convenient than others by virtue of giving some of the system parameters a special form, thus allowing short cuts to be taken in some of the computations. The savings can be quite substantial. The two most convenient placements are:

- 1) to align the coordinate axes with the principal axes of inertia of the link, and
- 2) to align one of the coordinate axes with the axis of one of the joints imbedded in the link.

With choice 1 the equation of motion of the link takes on a special form, while with choice 2 the joint axis vector takes on a special form and the task of calculating the transformations between adjacent link coordinate systems can be simplified. From the computational point of view, choice 2 is more profitable.

A coordinate system placement that is particularly effective in reducing the computational cost is that due to Denavit and Hartenberg [1955]. The scheme makes use of the fact that any two lines in space have a common perpendicular, which is normally unique, and places the coordinate systems such that their z axes are aligned with the joint axes and their x axes with the common perpendiculars.

A coordinate placement based on this scheme is shown in figure 7.1 (see also [Featherstone 1982]). O_i is the origin of i -coordinates, which has its z axis aligned with the axis of joint i and its x axis aligned with the common perpendicular between the axes of joints i and $i+1$, both of which are imbedded in link i . The transformation from i -coordinates to $i+1$ coordinates is effected by a screw

Figure 7.1 Link coordinate systems

transformation on the x axis, comprising a rotation α_i and a translation a_i , followed by a screw transformation on the new z axis comprising a rotation θ_{i+1} , a translation s_{i+1} and a rotation or translation of q_{i+1} depending on whether joint $i+1$ is revolute or prismatic. The four parameters a_i , α_i , s_{i+1} and θ_{i+1} give the relative position of $i+1$ coordinates with respect to i -coordinates when the joint variable q_{i+1} is zero. O'_0 is the base coordinate system (rather than O_0) to allow arbitrary placement with respect to the first joint axis, and O_{n+1} is the end effector coordinate system which can be placed arbitrarily with respect to the last joint axis. A total of $4n+6$ parameters define the kinematics of the robot.

This scheme is adequate for straight kinematic chains and single kinematic loops, but has some problems in coping with arbitrary mechanisms. A more general scheme for coordinate placement is described in [Sheth and Uicker 1971].

7.3 Efficient Spatial Arithmetic

The straightforward way of implementing a spatial arithmetic capability on a computer is to use a standard matrix arithmetic package and augment it with functions to perform spatial transposes on vectors and tensors. (Two operations are needed.) In this case a spatial vector is represented by a 6×1 matrix and a spatial tensor by a 6×6 matrix. The main disadvantage with this scheme is the representation of spatial transformations. A spatial rigid-body transformation is determined by six independent parameters, but there are 36 in a 6×6 matrix. The redundancy in this representation leads to considerable inefficiencies in storing, combining and applying

transformations [Rooney 1978]. The representation of rigid-body inertias, which have ten independent parameters, is also inefficient.

To improve the efficiency, spatial quantities may be stored in a computer as aggregates of scalar and 3-dimensional vector and matrix quantities, and a small suite of special purpose procedures may be defined to implement the necessary arithmetic operations in terms of these quantities. There are adequate facilities in most programming languages to hide the implementation details from the programmer, who should be allowed to think of spatial quantities as objects in their own right.

Quantity	Mathematical Form	Computer Representation	No. of Real Numbers	
			Math.	Comp.
vector	$\begin{bmatrix} \underline{v} \\ \underline{v}_o \end{bmatrix}$	$\text{spv}(\underline{v}, \underline{v}_o)$	6	6
transform	$\begin{bmatrix} E & 0 \\ E\underline{r}_x^T & E \end{bmatrix}$	$\text{spx}(E, \underline{r})$	36	12
artic.-body inertia	$\begin{bmatrix} H^T & M \\ I & H \end{bmatrix}$	$\text{abi}(M, H, I)$	36	27
rigid-body inertia	$\begin{bmatrix} \underline{h}_x^T & m1 \\ I & \underline{h}_x \end{bmatrix}$	$\text{rbi}(m, \underline{h}, I)$	36	13

Table 7.1 Computer representation of spatial quantities

The suggested computer representations for spatial quantities are given in table 7.1. The notation $\text{obj}(a, b, \dots)$ is intended to represent a computer record, structure, or aggregate representing obj and containing the quantities listed, which are not necessarily quantities of the same type. The implementation of operations like addition on the computer representations is obvious, so I shall concentrate on operations involving transformations.

7.3.1 Efficient Transformations

Let $\hat{X}$, $\hat{X}_1$ and $\hat{X}_2$ be spatial transformations defined by

$$\hat{X} = \begin{bmatrix} E & 0 \\ E \underline{r}^{\times T} & E \end{bmatrix}, \quad \hat{X}_1 = \begin{bmatrix} E_1 & 0 \\ E_1 \underline{r}_1^{\times T} & E_1 \end{bmatrix} \quad \text{and} \quad \hat{X}_2 = \begin{bmatrix} E_2 & 0 \\ E_2 \underline{r}_2^{\times T} & E_2 \end{bmatrix}. \quad \text{Given}$$

that $\hat{X} = \hat{X}_1 \hat{X}_2$, we have that

$$\begin{aligned} \begin{bmatrix} E & 0 \\ E \underline{r}^{\times T} & E \end{bmatrix} &= \begin{bmatrix} E_1 & 0 \\ E_1 \underline{r}_1^{\times T} & E_1 \end{bmatrix} \begin{bmatrix} E_2 & 0 \\ E_2 \underline{r}_2^{\times T} & E_2 \end{bmatrix} \\ &= \begin{bmatrix} E_1 E_2 & 0 \\ E_1 \underline{r}_1^{\times T} E_2 + E_1 E_2 \underline{r}_2^{\times T} & E_1 E_2 \end{bmatrix} \\ &= \begin{bmatrix} E_1 E_2 & 0 \\ E_1 E_2 (E_2^{-1} \underline{r}_1 + \underline{r}_2)^{\times T} & E_1 E_2 \end{bmatrix}. \end{aligned}$$

So, in terms of the computer representations of the transformations, we have that

$$\text{spx}(E_1, \underline{r}_1) \text{spx}(E_2, \underline{r}_2) = \text{spx}(E_1 E_2, \underline{r}_2 + E_2^{-1} \underline{r}_1).$$

The implementation of this operation requires one 3×3 matrix multiplication, one transpose, one matrix-vector multiplication and one vector addition. The total computational requirement is $36m + 27a$, where m stands for scalar multiplication and a for scalar addition. This compares favourably with the computational requirement for a 6×6 matrix multiplication, which is $216m + 180a$. Other operations involving transformations may be treated in a similar way, and the results are summarised in table 7.2. (It is assumed that the operations $\underline{a} \times \underline{b}$ and $\underline{a} \times B$ take $9m + 3a$ and $18m + 9a$ respectively.) The figures in the improvement column indicate the factor by which the computational

Operation	Result	Cost	Improvement
$\text{spx}(E_1, \underline{r}_1) \text{ spx}(E_2, \underline{r}_2)$	$\text{spx}(E_1 E_2, \underline{r}_2 + E_2^{-1} \underline{r}_1)$	$36m + 27a$	6
$\text{spx}(E, \underline{r}) \text{ spv}(\underline{v}, \underline{v}_0)$	$\text{spv}(E \underline{v}, E(\underline{v}_0 - \underline{r} \times \underline{v}))$	$24m + 18a$	1.5
$\text{spx}(E, \underline{r}) \text{ abi}(M, H, I)$	$\text{abi}(E M E^{-1}, E(H - \underline{r} \times M) E^{-1}, E(I - \underline{r} \times H^T + (H - \underline{r} \times M) \underline{r} \times) E^{-1})$	$216m + 162a$	2
$\text{spx}(E, \underline{r}) \text{ rbi}(m, \underline{h}, I)$	$\text{rbi}(m, E(\underline{h} - m \underline{r}), E(I + \underline{r} \times \underline{h} \times + (\underline{h} - m \underline{r}) \times \underline{r} \times) E^{-1})$	$84m + 69a$	5

Table 7.2 Implementation of transformation operations

cost is reduced over the use of 6-dimensional vectors and matrices.¹ In view of the large improvement in efficiency I recommend the use of this representation.

7.3.2 Symmetry

The process of reducing the computational requirement can be taken much further. For instance, we can make use of the fact that M and I are symmetric, and implement special functions for the addition and multiplication of symmetric matrices. The resulting improvement in efficiency is about 20%, which is quite small, and the numbers are given in table 7.3.

1. In most cases both multiplication and addition requirements are reduced by the same factor, but where not the improvement figure is calculated assuming multiplication takes 4 times as long as addition.

Operation	Cost		Improve- ment
	before	after	
$\text{spx}(E, \underline{r}) \text{ abi}(M, H, I)$	$216m+162a$	$177m+132a$	1.2
$\text{spx}(E, \underline{r}) \text{ rbi}(m, \underline{h}, I)$	$84m+69a$	$69m+57a$	1.2

Table 7.3 Making use of symmetry

7.3.3 Screw Transformations on Coordinate Axes

Recall that Denavit-Hartenberg style coordinates allow the transformation from one link coordinate system to an adjacent one to be performed by two screw transformations, one on the x axis and one on the z axis. When the screw axis is one of the coordinate axes, the transformation takes on a particularly simple form (e.g. see [Rooney 1978]) and can be represented as $\text{scr}(\theta, d, w)$, where θ is the angle of rotation about the axis, d the displacement along it, and w the specification of which axis (x, y or z) is involved. For example, $\text{scr}(\theta, d, z)$ is equivalent to $\text{spx}(E, \underline{r})$ where $E = \begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $\underline{r} = \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}$, the abbreviations s and c standing for $\sin(\theta)$ and $\cos(\theta)$ respectively. The application of $\text{scr}(\theta, d, z)$ to $\text{spv}(\underline{v}, \underline{v}_0)$ proceeds as follows:

$$\begin{bmatrix} c & s & 0 & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -sd & cd & 0 & c & s & 0 \\ -cd & -sd & 0 & -s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_{ox} \\ v_{oy} \\ v_{oz} \end{bmatrix} = \begin{bmatrix} cv_x + sv_y \\ cv_y - sv_x \\ v_z \\ c(v_{ox} + v_y d) + s(v_{oy} - v_x d) \\ c(v_{oy} - v_x d) - s(v_{ox} + v_y d) \\ v_{oz} \end{bmatrix}$$

The computational requirement for this operation is $10m+6a$, which is less than half of the requirement for the general transformation, so it is possible to save time by performing transformations between adjacent coordinate systems as pairs of these transformations.

Operation	Cost	Improvement
$scr(\theta, d, w) \text{ spx}(\underline{E}, \underline{r})$	$15m+9a$	1.25
$scr(\theta, d, w) \text{ spv}(\underline{v}, \underline{v}_0)$	$10m+6a$	1.25
$scr(\theta, d, w) \text{ abi}(\underline{M}, \underline{H}, \underline{I})$	$70m+64a$	1.5
+symmetry	$54m+50a$	1.6
$scr(\theta, d, w) \text{ rbi}(\underline{m}, \underline{h}, \underline{I})$	$31m+25a$	1.35
+symmetry	$23m+20a$	1.5

Table 7.4 Coordinate-axis-screw transformations

The computational requirements for the operations of table 7.2 using coordinate-axis-screw transformations are given in table 7.4. The figures in the cost column refer to one transformation, but those in the improvement column refer to using two coordinate-axis-screw transformations instead of one general one. To save having to compute trigonometric functions every time a transformation is performed, the aggregate should include the values $s=\sin(\theta)$ and $c=\cos(\theta)$, and for transforming matrices it is assumed that s^2 , c^2 and

sc are also available (otherwise an additional $3m$ overhead per transformation is incurred). Notice that the combination of coordinate-axis-screw transformations and symmetric matrices is more effective than either separately. An additional bonus with this method is that we do not have the overhead of constructing explicitly the transformations between adjacent link coordinate systems.

Even this is not the last word in optimisation, for if we take the operation $\text{scr}(\theta, d, w) \text{ rbi}(m, \underline{h}, I)$ and find an expression for each element of the result, then, by judicious use of partial results, we can reduce the computational cost to a mere $19m+18a$.

7.3.4 Parallel Computation

It has been assumed throughout the preceding discussion that the calculations are to be performed on a serial computer. Suppose we have a computer capable of performing three arithmetic operations in parallel so that, in terms of the time taken, the cost of a vector addition is reduced from $3a$ to a , that of a scalar product from $3m+2a$ to $m+a$, and that of a vector product from $6m+3a$ to $2m+a$. However, we can not ask for three arbitrary scalar operations to be performed in parallel. This gives 3-dimensional vector operations an advantage over scalar operations with the result that the optimisations based on symmetric matrices and coordinate-axis-screw transformations no longer give any improvement in computation speed. However, the aggregate representation retains its margin of efficiency over the 6-dimensional vector and matrix representation. On a machine capable of performing six arithmetic operations in parallel, the aggregate representation is still faster, though with a reduced margin of

efficiency. On a machine capable of performing 36 or more operations in parallel it is better to use the 6-dimensional vector representation.

7.4 The Computational Requirements of the Algorithms

The tally of arithmetic operations is a good guide to the actual amount of effort involved in the computation process. The assumption is that these operations (especially multiplication) take significantly longer to execute than the average machine instruction, and so account for most of the execution time. The computational requirement for a particular algorithm depends on the degree of optimisation, and to compare the relative performance of algorithms requires that they be implemented fairly.

In this section I give the computational requirements for each of the algorithms presented in the last three chapters, and compare the solution of the forward dynamics problem via the composite-rigid-body method with that via the articulated-body method. The choice of link coordinate systems is as described in section 2, and the implementation of the spatial arithmetic is as described in section 3, with maximum optimisation. Optimisations on operations other than transformations are also taken into account, and these are described below along with a discussion on the merits of calculation in link and absolute coordinates.

7.4.1 Link Coordinates versus Absolute Coordinates

To compute the algorithms in absolute coordinates requires the overhead of computing $\hat{\mathbf{I}}_i$ and $\hat{\mathbf{s}}_i$ from $\hat{\mathbf{I}}_i^A$ and $\hat{\mathbf{s}}_i^A$ (see chapter 4, section 4). This involves the calculation of the n transformations ${}^0\hat{\mathbf{X}}_i$ and performing n vector and n rigid-body inertia transformations. To compute the algorithms in link coordinates requires that every passed-on quantity be transformed to an adjacent coordinate system. In terms of the number of quantities that need to be transformed in each case, it would seem that absolute coordinates are more efficient. However, link coordinates have a number of points in their favour:

- 1) a transformation to an adjacent coordinate system can be made more efficient than a general transformation,
- 2) $\hat{\mathbf{s}}_i^A$ can be made to have a special form in link coordinates which leads to considerable savings, and
- 3) there is no overhead for calculating ${}^0\hat{\mathbf{X}}_i$.

The second point is based on the fact that $\hat{\mathbf{s}}_i^A$ can be made equal to one of the coordinate axes, so much multiplication can be avoided on operations like $\mathbf{Q}_i = \hat{\mathbf{s}}_i^{A^T} \hat{\mathbf{f}}_i$ (eq. (4.14)), which simplifies to selecting an element from $\hat{\mathbf{f}}_i$, and $\hat{\mathbf{h}}_i = \hat{\mathbf{I}}_i^A \hat{\mathbf{s}}_i^A$ (eq. (6.57)), which simplifies to selecting a column (or row) from $\hat{\mathbf{I}}_i^A$. The third point is only valid if the locations of the links are of no interest.

A disadvantage of performing the calculation in link coordinates is that if external forces are being applied to the links then they must be transformed to link coordinates before their effects can be

taken into account. This involves an additional transformation overhead which is not incurred if the calculations are being performed in absolute coordinates.

In the case of the Newton-Euler algorithm, computation in link coordinates has the considerable advantage of not requiring the transformation of any tensors, and is therefore substantially more efficient than computation in absolute coordinates.

In the case of the articulated-body algorithm, computation in link coordinates requires the transformation of articulated-body inertias, which is expensive, whereas computation in absolute coordinates requires the transformation of rigid-body inertias, which is somewhat cheaper. Link coordinates are still faster, but only by about 10% or so, and the need to take into account external forces on the links or the location of the end effector can easily render link coordinates less efficient.

In the case of the composite-rigid-body algorithm the situation is more complicated, since the link coordinate version requires $O(n^2)$ transformations whereas the absolute coordinate version requires only $O(n)$. As a result, the absolute coordinate version has a considerably lower coefficient of n^2 in its computational requirement, but because of the overheads associated with the use of absolute coordinates it has a considerably larger $O(n)$ dependency. The absolute coordinate version is more efficient for values of $n > 16$, but its $O(n)$ dependency is so large that solving forward dynamics by this method is always less efficient than the articulated-body algorithm.

Method	Cost		Cost n=6
	multiplications	additions	
Newton-Euler	$130n-68$	$101n-56$	$712m+550a$
composite-rigid-body	$10n^2+31n-41$	$6n^2+40n-46$	$505m+410a$
fwd. dyn. via CRBM	$\frac{1}{6}n^3+11\frac{1}{2}n^2+161\frac{1}{3}n-109$	$\frac{1}{6}n^3+7n^2+138\frac{5}{6}n-102$	$1309m+1019a$
articulated body	$300n-267$	$279n-259$	$1533m+1415a$

Table 7.5 Comparison of dynamics algorithms

7.4.2 Comparison of the Algorithms

Table 7.5 shows the computational requirements of the three algorithms in terms of the numbers of arithmetic operations expressed as polynomials in n . The numbers exclude the cost of calculating the $n-1$ sines and cosines required for the transformations. The computational requirement of performing the complete forward dynamics calculation via the composite-rigid-body method is also included for comparison with the articulated-body method. This is the sum of the computational requirements of the composite-rigid-body method, a simplified version of the Newton-Euler method and the Cholesky decomposition method for solving a set of simultaneous equations where the matrix is symmetric and positive definite. In all cases the algorithms are performed in link coordinates using all the optimisations described above, but the numbers in the table do not represent the minimum possible cost. They do, however, give a reasonably fair comparison of the relative costs of the algorithms.

The basic conclusions we can draw are that forward dynamics takes longer to compute than inverse dynamics, by a factor of up to 2.3, and that the articulated-body method for forward dynamics is only faster than the composite-rigid-body method if $n > 9$, so the composite-rigid-body method is better for most practical problems. If we allow $200\mu\text{s}$ computation time for a sine/cosine calculation, $10\mu\text{s}$ for multiplication, $2\mu\text{s}$ for addition and a $1\mu\text{s}$ overhead per operation for shuffling data, we get an execution time of less than 12ms for the Newton-Euler algorithm on a 6-degree-of-freedom robot, which is fast enough for use in real-time control.

Various other factors may tip the balance in favour of one or another algorithm. For example, if the robot has a floating base (e.g. as in a satellite-mounted robot) then the articulated-body algorithm becomes relatively much more efficient than the composite-rigid-body algorithm (see chapter 8), but the presence of kinematic loops is detrimental to the articulated-body algorithm (see chapter 9). Another factor that should be taken into account is the possibility of partitioning the calculation of forward dynamics into parts that need frequent re-calculation and parts that do not (as done in [Featherstone 1982] for example). This practice favours the composite-rigid-body method.

7.4.3 Approximations and Special-Purpose Algorithms

All the discussion so far has been on the implementation and optimisation of exact dynamics algorithms for robots with general geometries. Most industrial robots have very special geometries, with adjacent joint axes being almost always parallel or

perpendicular and frequently intersecting. It is possible to write special-purpose algorithms which take advantage of the special features of some particular robot geometry to reduce the computational cost. Another way to reduce the cost is to use approximations to the inertia parameters. A link's centre of mass is typically close to a link coordinate axis, and there is usually little error in assuming that its principal axes of inertia are parallel with the link coordinate axes. The reduction in computational cost over the general algorithms depends on the particular robot, but typically a factor of 2 or more can be expected. For more details see [Hollerbach and Sahar 1983; Kane and Levinson 1983].

Chapter 8

Branched Kinematic Chains Multiple-Degree-of-Freedom Joints and Moving Bases

8.1 Introduction

In this chapter I describe various extensions of the robot system model defined in chapter 4 to allow the description of a wider class of robot mechanism, and the modifications to the dynamics algorithms necessary to cater for these extensions. The material is organised into three topics:

- 1) generalising the robot's connectivity to allow branched kinematic chains,
- 2) allowing screw joints and joints with more than one degree of freedom, and
- 3) allowing the base member to move so that its acceleration is not known in advance.

Each extension is easily combined with the others, but for the sake of modularity I describe each independently of the others. Where possible, modifications to the basic dynamics algorithms are described in terms of rules for modifying the equations presented in

chapters 4 to 6 rather than by re-stating or re-deriving entire algorithms.

The implementation of branched kinematic chains is accomplished by including an explicit description of the link connectivity as part of the system model and making a few systematic modifications to the equations of the dynamics algorithms. Their properties and performances are not affected significantly. An example of a branched kinematic chain is a robot with a multi-fingered gripper. Kinematic loops are specifically excluded at this stage since they introduce many more problems than tree structures, so we will not yet be able to cope with a multi-fingered gripper manipulating an object. Kinematic loops are considered in the next chapter.

Screw joints can be implemented without any change to the system model or the dynamics algorithms simply by allowing $\hat{s}_i$ to represent a screw axis. They do, however, raise the issue of magnitudes of unit screw forces and displacements. This issue is clarified under multiple-degree-of-freedom joints with the introduction of explicit actuator models.

Multiple-degree-of-freedom joints introduce more significant changes to both the system model and the dynamics algorithms: the notion of a powered joint needs to be defined more carefully, and a number of quantities in the dynamics algorithms which were previously scalars become vectors or matrices. The task of finding the joint displacement can become problematical since it is not always possible to integrate the velocity. Although multiple-degree-of-freedom joints can be synthesised from the appropriate number of single-degree-of-freedom joints, this is not always convenient and can lead

to computational inefficiencies.

A moving-base robot can be treated as a fixed-base robot by introducing a fictitious, un-powered joint between the base and some fixed point. If the base has complete motion freedom then the joint has six degrees of freedom. The forward dynamics algorithms cope with this easily enough, but inverse dynamics needs significant modification because of the unknown base acceleration. The modified algorithm is a mixture of forward and inverse dynamics. Examples of robots with moving bases include robots mounted on spacecraft and legged robots leaping through the air.

The algorithms described in this chapter have not been tested on a computer (unlike those of chapters 4 to 6, which have).

Figure 8.1 Robot with branched chain structure

8.2 Branched Kinematic Chains

By a branched kinematic chain I mean that the links may be connected in any manner corresponding to a topological tree. Basically, this means that there are no kinematic loops and that no part of the robot is entirely disconnected from the rest. Figure 8.1 gives an example. The base link (which is still considered immobile) is chosen as the root of the tree, and the outermost links, which have only one joint, are the leaves. I shall call the neighbour of a given link its predecessor or successor depending on whether it is nearer to or further from the base, the joint connecting a link to its predecessor its inner joint, and those connecting a link to its successors its outer joints. Every link except the base has exactly one inner joint, but may have any number of outer joints.

The main difference between such a tree structure and the unbranched chains we have been using up to now is that for the tree structure we need to specify the connectivity explicitly whereas for the unbranched chain it was implicit in the numbering scheme. The connectivity is defined by introducing for each link the quantity λ_i , which is the identification number of the preceding link.¹ For the tree structure shown in figure 8.1 the values of λ_i for $i=1..6$ are $(0,1,2,1,4,4)$. The λ_i alone are enough to define the connectivity, but for notational convenience in expressing iterations going from the base to the leaves I introduce for each link the additional quantity μ_i , which is the set of identification numbers of all the successors of link i . For the tree structure shown in figure 8.1 the

¹. These are the elements of the body connection array defined by [Huston et al. 1978].

values of μ_i are $\mu_0=\{1\}$, $\mu_1=\{2,4\}$, $\mu_2=\{3\}$, $\mu_4=\{5,6\}$, $\mu_3,\mu_5,\mu_6=\{\}$. The leaves are distinguished by having empty μ_s .

A systematic enumeration of the links is no longer necessary, but it is still convenient and will be retained. The numbering scheme I shall use is to call the base link 0, and number the other links from 1 to n so that every link has a higher identification number than its predecessor. The inner joint for link i is called joint i. (The numbering on figure 8.1 follows this scheme.) This numbering scheme degenerates into our original numbering scheme when applied to an un-branched kinematic chain, but there is not normally a unique numbering for a branched kinematic chain. The adjacent-link coordinate transformation across joint i is now ${}_{\lambda_i} \hat{X}_i$.

8.2.1 Modifying the Algorithms

The equations of the Newton-Euler and articulated-body algorithms may be classified into three groups: those that iterate from the base to the end effector, those that iterate from the end effector to the base, and those that perform computations local to each link. Members of the last group are independent of the connectivity and need no modification.

Outward iterating equations carry motion-type quantities from the base to the end effector. An example is the link velocity calculation of eq. (4.5):

$$\underline{\hat{v}}_i = \underline{\hat{v}}_{i-1} + \underline{\hat{s}}_i \dot{q}_i \quad (\underline{\hat{v}}_0 = \underline{\hat{o}}) \quad (1)$$

In this context the expression i-1 really means 'the predecessor of

link i' , so modifying these equations to work on branched kinematic chains is just a matter of replacing instances of $i-1$ with λ_i . The link velocity calculation for a tree structure becomes

$$\underline{\hat{v}}_i = \underline{\hat{v}}_{\lambda_i} + \underline{\hat{s}}_i \dot{q}_i \quad (\underline{\hat{v}}_0 = \underline{\hat{0}}) \quad (2)$$

If we stick to the original order of calculation, where i goes from 1 to n , then $\underline{\hat{v}}_{\lambda_i}$ will always be calculated before $\underline{\hat{v}}_i$ (since $\lambda_i < i$).

Inward iterating equations carry forces or inertias from the end effector to the base. With a tree structure, any given link receives one such quantity from each of its successors, and since these quantities are additive the total amount is the sum of all the contributions. For example, the calculation of total joint forces for the Newton-Euler method (eq. (4.8)) is given by

$$\underline{\hat{f}}_i = \underline{\hat{f}}_{i+1} + \underline{\hat{f}}_i^* \quad (\underline{\hat{f}}_n = \underline{\hat{f}}_n^*) \quad (3)$$

This equation states that the total force required to support the motion of all links from i onwards is the sum of the forces required to support the motion of link i and the motion of all following links. In the context of a tree structure, link i supports directly all links j where $j \in \mu_i$, so $\underline{\hat{f}}_{i+1}$ in eq. (3) must be replaced by $\sum_{j \in \mu_i} \underline{\hat{f}}_j$ giving

$$\underline{\hat{f}}_i = \sum_{j \in \mu_i} \underline{\hat{f}}_j + \underline{\hat{f}}_i^* \quad (4)$$

Notice that there is now no need for a separate initial condition since the calculation of $\underline{\hat{f}}_i$ when link i is a leaf is handled correctly by virtue of μ_i being the empty set for such a link. Again, if we stick to the original order of calculation, where i goes

from n down to 1, we will always calculate all the $\hat{f}_j$ before $\hat{f}_i$.

A more complicated example is the calculation of articulated-body inertias. For an un-branched chain this is given by eq. (6.37):

$$\hat{I}_i^A = \hat{I}_i + \hat{I}_{i+1}^A - \frac{\hat{I}_{i+1}^A \hat{s}_{i+1}^S \hat{s}_{i+1}^S \hat{I}_{i+1}^A}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}^S} . \quad (\hat{I}_n^A = \hat{I}_n) \quad (5)$$

The quantity that gets passed back from the successor link appears as an expression rather than a single quantity, so the branched-chain version of this equation is

$$\hat{I}_i^A = \hat{I}_i + \sum_{j \in \mu_i} \left(\hat{I}_j^A - \frac{\hat{I}_j^A \hat{s}_j^S \hat{s}_j^S \hat{I}_j^A}{\hat{s}_j^S \hat{I}_j^A \hat{s}_j^S} \right) . \quad (6)$$

The order of traversal of the tree can be controlled by the numbering scheme. This has no effect on the amount of computation performed, but it can affect the amount of temporary storage needed.

The computation of the joint-space inertia matrix does not fit into the scheme outlined above, but it is still easily modified. The joint-space inertia matrix is defined by eq. (5.8) as

$$H_{ji} = \hat{s}_j^S \hat{f}_i , \quad (j \leq i) \quad (7)$$

where $\hat{f}_i$ is the force required to impart unit acceleration to the composite rigid body comprising links $i..n$, and H_{ji} is the joint force required at joint j to support $\hat{f}_i$ at joint i . In the case of a tree structure, this equation only applies if joint j is supporting the sub-tree containing the moving links. For example, referring back to figure 8.1, if $i=5$ then eq. (7) only applies to joints 1, 4

and 5, and the values of H_{25} and H_{35} are zero since neither joint 2 nor joint 3 supports the motion of joint 5.

To express this concisely, I introduce the quantities v_i which, for each joint, list all the links in the sub-tree supported by that joint. v_i is defined by

$$j \in v_i \text{ iff } j = i \text{ or } (j \in \mu_k \text{ and } k \in v_i) . \quad (8)$$

For the tree structure shown in figure 8.1 $v_1 = \{1, 2, 3, 4, 5, 6\}$, $v_2 = \{2, 3\}$, and so on. Armed with this definition, we may express the tree structure version of eq. (7) as

$$H_{ji} = \begin{cases} \hat{s}_j^s \hat{f}_i & j \in v_i \\ 0 & j \notin v_i \end{cases} . \quad (9)$$

It is possible to do without the v_i by setting every element of H initially to zero and using eq. (7) with i ranging from n down to 1 and j taking the values i , λ_i , λ_{λ_i} , etc., to compute the non-zero elements of H .

8.3 Screw Joints

The representation of screw joint axes was discussed in chapter 2 along with revolute and prismatic ones. Although the system model defined in chapter 4 specified that joints should be either revolute or prismatic, they can just as easily be screw joints, without affecting any of the dynamic equations, simply by allowing $\hat{s}_i'$ to represent a screw axis.

The issue raised by screw joints is really one of units and magnitudes. We have assumed that the unit joint force acting over the unit joint displacement performs the unit amount of work. Moreover, we have assumed that the joint force for a revolute joint is a pure couple in the appropriate direction and that for a prismatic joint a pure force. The most logical choice of joint force for a screw joint of pitch h is a screw force of pitch $1/h$, but if we use the standard definition of a unit screw, which is that the line-vector part should have unit magnitude, then the unit screw force acting over the unit screw displacement performs $1+h^2$ times the unit work. So we define the unit joint force for a screw joint of pitch h as a screw force of magnitude $1/(1+h^2)$ and pitch $1/h$.

It is tempting to say that the standard definition of a unit screw is inappropriate, and that a better definition is to set the magnitude of the line-vector part of a screw of pitch h to $1/\sqrt{1+h^2}$, but this definition is only appropriate if we make the pitch of the joint force $1/h$. Remember that, from the dynamics point of view, a spatial vector does not have a true magnitude, and whatever we decide to call a unit vector is largely a matter of convenience.

The relationship between the magnitude of $\hat{s}_i$ and the units of measurement of q_i and Q_i is quite simple: if we scale $\hat{s}_i$ by a factor of α then the unit of measurement of q_i is scaled by α and that of Q_i by $1/\alpha$.

The situation will be made clearer in the next section when I introduce an explicit model of the actuation system and relax the condition that the unit joint force acting over the unit joint displacement should perform the unit work.

8.4 Multiple-Degree-of-Freedom Joints

A multiple-degree-of-freedom joint can be simulated by a chain of the appropriate number of single-degree-of-freedom joints, but there are occasions when this can be inconvenient or lead to computational inefficiencies, and the introduction of fictitious links with zero inertia can cause numerical problems in the dynamics computations. Consequently, I describe in this section the direct representation of multiple-degree-of-freedom joints. This requires a number of modifications to the joint model, including an explicit model of the actuation system for a powered joint, and it introduces a few problems of its own, notably that of determining the joint displacement from the velocity.

8.4.1 The Joint Model

The motion allowed by a single-degree-of-freedom joint is defined by its axis $\hat{s}$. The instantaneous motion allowed by an r -degree-of-freedom joint is defined by the r -dimensional sub-space $\hat{S}$, which, in general, is a function of the joint displacement. For most practical cases, however, $\hat{S}$ is independent of the joint displacement. The generalised velocity of the joint is now given by the r -dimensional vector $\dot{\underline{q}}$ and the spatial velocity of the joint by

$$\underline{\hat{v}} = \hat{S} \dot{\underline{q}} . \quad (10)$$

The columns of $\hat{S}$ are important, since they define the motions to which each element of $\dot{\underline{q}}$ refers. Similarly, the acceleration of the joint is given by the r -dimensional vector $\ddot{\underline{q}}$ and the spatial

acceleration by

$$\underline{\hat{a}} = \hat{S} \ddot{\underline{q}} + \dot{\hat{S}} \dot{\underline{q}}, \quad (11)$$

though it will often be the case that $\dot{\hat{S}}$ is zero in some link coordinate frame. The joint displacement can not normally be obtained by integrating $\dot{\underline{q}}$ since there is in general no such thing as a spatial displacement vector. Joint displacement must be represented as a transformation and related to the velocity by a differential equation. Since I am primarily interested in instantaneous dynamics, I shall defer further discussion on this point until the end of the section.

A powered joint is deemed to have an actuation system which is capable of generating spatial forces in the sub-space $\hat{F}$. For a joint to be 'properly powered' the actuation space must be of the same dimension as the motion space, and it must not have any member in common with the reaction-force space $\hat{S}^\perp$. In other words, $\hat{S}^S \hat{F}$ must be non-singular. It is possible to conceive of joints where the actuation system has fewer degrees of freedom than the joint (under powered), or more (over powered), but I shall restrict my discussion to properly-powered joints.

The actuator force for an r -degree-of-freedom, properly-powered joint is given by the r -dimensional vector $\underline{Q}$ and the spatial force produced by the actuation system by $\hat{F} \underline{Q}$. The columns of $\hat{F}$ define the significance of each element of $\underline{Q}$. This force is deemed to act on the 'successor' link and be supplied by the 'predecessor' link. The power balance equation now takes on a more complicated form: if $\hat{\underline{f}}$ and $\hat{\underline{v}}$ are the spatial force and velocity at the joint, then

$$P = \hat{\underline{v}}^S \hat{\underline{f}} = \underline{\underline{q}}^T \hat{\underline{S}}^S \hat{\underline{F}} \underline{\underline{Q}} . \quad (12)$$

To find $\underline{\underline{Q}}$ in terms of $\hat{\underline{f}}$, use

$$\begin{aligned} \frac{\partial P}{\partial \underline{\underline{q}}} &= \hat{\underline{S}}^S \hat{\underline{f}} = \hat{\underline{S}}^S \hat{\underline{F}} \underline{\underline{Q}} \\ \rightarrow \underline{\underline{Q}} &= (\hat{\underline{S}}^S \hat{\underline{F}})^{-1} \hat{\underline{S}}^S \hat{\underline{f}} . \end{aligned} \quad (13)$$

$\hat{\underline{S}}^S \hat{\underline{F}}$ is non-singular, so $\underline{\underline{Q}}$ may always be found. Note that $\hat{\underline{S}}^S \hat{\underline{F}}$ is coordinate-system independent.

In a single-degree-of-freedom joint $\hat{\underline{S}}^S \hat{\underline{F}}$ is a scalar, and in our original model it was assumed that this scalar was unity. It was assumed, for example, that $\hat{\underline{F}}$ was a pure couple for a revolute joint and a pure force for a prismatic joint. Whilst we can now generalise our notion of a powered joint, the case where $\hat{\underline{S}}^S \hat{\underline{F}}$ is the identity matrix is still the most convenient.

We are at liberty to choose the columns of $\hat{\underline{F}}$ as we please. In particular, we can make them reflect the actual arrangement of actuators in a real joint. Moreover, for $r < 6$ the actuation-force space itself can not be determined completely from the motion constraints, so there is some degree of choice here too.

8.4.2 Modifying the Dynamics Algorithms

Multiple-degree-of-freedom joints introduce more substantial changes to the equations of the dynamics algorithms, though there is no change in the procedures for deriving them. The rules for modifying the equations are as follows:

- 1) replace the quantities $\hat{s}_i$, $\dot{q}_i$ and $\ddot{q}_i$ with $\hat{S}_i$, $\dot{\hat{q}}_i$ and $\ddot{\hat{q}}_i$;
- 2) include a term with $\dot{\hat{S}}_i$ wherever $\hat{S}_i$ is to be differentiated; and
- 3) replace occurrences of Q_i with the expression $\hat{S}_i^S \hat{F}_i \underline{Q}_i$.

These rules introduce new terms and factors, and they cause some quantities and expressions which were originally scalars to become vectors or matrices. In some cases the form of the final equations can be quite different, so it is advisable to apply the modification rules to the initial equations and re-derive the final ones. For example, the equation for calculating articulated-body inertias (eq. (6.37)) becomes

$$\hat{I}_i^A = \hat{I}_i + \hat{I}_{i+1}^A - \hat{I}_{i+1}^A \hat{S}_{i+1} (\hat{S}_{i+1}^S \hat{I}_{i+1}^A \hat{S}_{i+1})^{-1} \hat{S}_{i+1}^S \hat{I}_{i+1}^A. \quad (14)$$

The denominator of the original equation has become an $r \times r$ matrix, so it appears as an inverse, splitting the original numerator. Similarly, the equation for calculating the joint-space inertia matrix (eq. (5.9)) becomes

$$H_{ji} = H_{ij}^T = \hat{S}_j^S \hat{I}_r^C \hat{S}_i. \quad (r = \max(i, j)) \quad (15)$$

The transpose is necessary since H_{ji} is now a rectangular matrix, but it is not obvious from the original equation where the transpose should go. It is interesting to note that in the case of eq. (14) if $r=6$ then $\hat{S}_{i+1}^{-1}$ exists and the equation simplifies to $\hat{I}_i^A = \hat{I}_i$, which is what you would expect.

Assessing the computational requirements of the algorithms is now more complicated than it was before, since some parts depend on the number of joints, and other parts on the number of degrees of freedom of the entire system or of individual joints. For example, the

number of link-to-link transformations is a function of the number of joints, the size of the joint-space inertia matrix is given by the total number of degrees of freedom, and the amount of calculation involved in evaluating eq. (14) depends on the number of degrees of freedom of joint $i+1$.

8.4.3 Calculating Joint Displacements

The displacement of a single-degree-of-freedom joint is a scalar, but that of a multiple-degree-of-freedom joint can not in general be represented as a vector. Consequently, it is not possible to obtain the displacement simply by integrating the velocity vector. The general solution to this problem is to represent the joint displacement as a spatial transformation $\hat{X}$, and express its derivative in terms of the joint velocity $\underline{\hat{v}}$, giving rise to a differential equation

$$\dot{\hat{X}} = \underline{\hat{v}} \hat{X} . \quad (16)$$

This solves the problem of obtaining the joint displacement, but it introduces two new problems. Firstly, due to numerical errors in the integration process, $\hat{X}$ will drift from being a rigid-body transformation, distorting all subsequent bodies. Secondly, even if $\hat{X}$ can be maintained as a rigid-body transformation, it will gradually depart from the geometric constraints which define the joint unless corrective action is taken.

To maintain $\hat{X}$ as a rigid-body transformation it must be kept in the form

$$\hat{X} = \begin{bmatrix} E & 0 \\ E \underline{r}^{\times T} & E \end{bmatrix},$$

but using the component representation described in the previous chapter, $\hat{X} = \text{spx}(E, \underline{r})$, this boils down to maintaining the orthogonality of E . To maintain the geometric accuracy of the transformation requires that it be checked against the geometry of the joint. The degree of difficulty involved depends on the joint, but it can be very simple. For example, for a spherical joint $\underline{r}$ is constant if the pivot point coincides with the origin of the coordinate system of the following link, so provided we set it to the correct value initially and do not modify it the transformation will always be geometrically correct.

It is possible to represent finite rotations as unit quaternions [Rooney 1977]. This is a more compact representation than the 3×3 orthogonal matrix, but the cost of applying a transformation is higher. The reason for mentioning this here is that if rotation is represented as a unit quaternion then it is easier to calculate the displacement from the velocity and maintain the correctness of the transformation. If q is a quaternion representing the rotational part of a joint displacement and $\underline{\omega}$ is the joint's angular velocity, then $\dot{q}$ is given by

$$\dot{q} = \frac{1}{2} \underline{\omega} q. \quad (17)$$

The correctness of the transformation is maintained by normalising q .

8.5 Robots with Floating Bases

By a floating base I mean that the base member is free to move so that its acceleration depends on the motion of the rest of the robot. In our original model it was stipulated that the base should be stationary, though in fact it is possible to permit non-zero velocity and acceleration provided that they are known in advance. This affects only the starting values for the velocity and acceleration recurrence relations.

To allow for a floating base in our robot model, we must give the base an inertia $\hat{I}_0$, and introduce a fictitious 6-degree-of-freedom joint between the base and a fixed reference point (e.g. as done in [Raibert et al. 1983]). A partially constrained base can be modelled by a joint with fewer than six degrees of freedom, but I will assume here that the base has complete motion freedom. The base velocity and acceleration will be denoted by $\hat{v}_0$ and $\hat{a}_0$ respectively.

The effect of allowing a floating base is different on each of the basic dynamics algorithms. The articulated-body algorithm is the least affected since articulated bodies are floating structures anyway. Assuming that no external force is being applied to the base, its acceleration is given by

$$\begin{aligned} \hat{I}_0^A \hat{a}_0 + \hat{p}_0 &= \hat{0} \\ \rightarrow \hat{a}_0 &= - (\hat{I}_0^A)^{-1} \hat{p}_0, \end{aligned} \tag{18}$$

where $\hat{I}_0^A$ and $\hat{p}_0$ are calculated from eqs. (6.37 and 6.38). This equation takes the place of eq. (6.44). A constant computational cost is incurred which is approximately equal to the incremental cost of one

extra joint.

The simplest way to adapt the calculation of forward dynamics via the composite-rigid-body algorithm to cope with a floating base is to introduce $\hat{S}_0$, the 6-degree-of-freedom motion space of the fictitious joint between the base and the reference point, and use the techniques described in the previous section for dealing with multiple-degree-of-freedom joints. This is a good example of how the direct representation of multiple-degree-of-freedom joints can lead to computational savings, since we can take advantage of the special nature of a 6-degree-of-freedom joint in a way that would not be possible if it were represented as a series of six one-degree-of-freedom joints.

The problem can still be solved (though less elegantly) without recourse to multiple-degree-of-freedom-joint techniques as follows. If we perform the inverse dynamics calculation assuming that $\hat{\underline{a}}_0 = \hat{\underline{0}}$, the resulting vector, $\underline{C}$, gives the joint forces needed to account for all dynamic effects except the joint and base accelerations. A readily-available by-product of this computation is $\hat{\underline{f}}$, the force which must be applied to the base in the situation where the joint and base accelerations are all zero. Ignoring velocity and external effects, the force required at the base to support a unit acceleration on joint i is $\hat{\underline{I}}_i^c \hat{\underline{s}}_i$, so the force required to support the joint acceleration $\underline{\ddot{q}}$ is $\sum_i \hat{\underline{I}}_i^c \hat{\underline{s}}_i \ddot{q}_i$. This can be expressed more concisely as $\hat{\underline{K}} \underline{\ddot{q}}$, where $\hat{\underline{K}}$ is a $6 \times n$ matrix the i^{th} column of which is $\hat{\underline{I}}_i^c \hat{\underline{s}}_i$. The equation relating base acceleration to applied force is then

$$\hat{\underline{I}}_0^c \hat{\underline{a}}_0 + \hat{\underline{K}} \underline{\ddot{q}} + \hat{\underline{f}} = \hat{\underline{0}}. \quad (19)$$

This gives us six equations in $n+6$ unknowns. The force required at joint i to compensate for non-zero acceleration of the base is

$\hat{\underline{s}}_i^s \hat{\underline{I}}_i^c \hat{\underline{a}}_0$, and the vector of all such forces can be expressed as $\hat{\underline{K}}^s \hat{\underline{a}}_0$. The joint forces are then given by the equation

$$\hat{\underline{K}}^s \hat{\underline{a}}_0 + H \ddot{\underline{q}} + \underline{C} = \underline{Q} . \quad (20)$$

This gives us another n equations, and we can combine eqs. (19 and 20) to produce the following set of simultaneous equations:

$$\begin{bmatrix} \hat{\underline{I}}_0^c & | & \hat{\underline{K}} \\ \hline \hat{\underline{K}}^s & | & H \end{bmatrix} \begin{bmatrix} \hat{\underline{a}}_0 \\ \hline \ddot{\underline{q}} \end{bmatrix} = \begin{bmatrix} -\hat{\underline{f}} \\ \hline \underline{Q} - \underline{C} \end{bmatrix} . \quad (21)$$

The augmented matrix is not symmetric, as it would be if we were using the multiple-degree-of-freedom joint approach, but this is purely because of the way we have arranged the elements in the vectors. Swapping the first three rows with the second three will make the matrix symmetric.

The extra calculation required to set up eq. (21) is approximately equal to the cost of adding one extra joint in the calculation of the joint-space inertia matrix, but the cost of solving the equations has risen by an amount equal to the addition of six extra joints. The incremental cost of a floating base is therefore greater for the composite-rigid-body method than for the articulated-body method.

Inverse dynamics is profoundly affected by the presence of a floating base. The nature of the problem is changed into a hybrid between forward and inverse dynamics: the accelerations of joints $1..n$ and the force on joint 0 are given, and the task is to find the forces at joints $1..n$ and the acceleration of joint 0. The algorithm is quite different to the one described in chapter 4 and is described

below in detail. The most general form of hybrid algorithm is one which for each joint computes either the force from the acceleration or the acceleration from the force depending on the value of a flag setting. A general hybrid algorithm based on the articulated-body algorithm is also described below, and one based on the composite-rigid-body algorithm is described in chapter 9 under joint end stops.

8.5.1 Inverse Dynamics of a Robot with a Floating Base

The base acceleration will be calculated from the composite-rigid-body inertia of the whole robot and the force required to produce zero base acceleration. Once the base acceleration is known, the rest follows easily. Let $\hat{\underline{f}}_i$ be the force exerted on link i from link $i-1$, then $\hat{\underline{f}}_i$ gives the overall rate of change of momentum for links $i..n$, so

$$\begin{aligned}
 \hat{\underline{f}}_i &= \frac{d}{dt} \left(\sum_{j=i}^n \hat{\underline{I}}_j \hat{\underline{v}}_j \right) \\
 &= \sum_{j=i}^n \left(\hat{\underline{I}}_j \hat{\underline{a}}_j + \hat{\underline{v}}_j \hat{\times} \hat{\underline{I}}_j \hat{\underline{v}}_j \right) \\
 &= \left(\sum_{j=i}^n \hat{\underline{I}}_j \right) \hat{\underline{a}}_0 + \sum_{j=i}^n \left(\hat{\underline{I}}_j \hat{\underline{a}}_j^r + \hat{\underline{v}}_j \hat{\times} \hat{\underline{I}}_j \hat{\underline{v}}_j \right) \\
 &= \hat{\underline{I}}_i^c \hat{\underline{a}}_0 + \hat{\underline{p}}_i .
 \end{aligned} \tag{22}$$

$\hat{\underline{a}}_i^r$ is the relative acceleration of link i with respect to link 0; $\hat{\underline{I}}_i^c$ is the composite-rigid-body inertia of links $i..n$; and $\hat{\underline{p}}_i$ is the bias force required to produce zero base acceleration. Velocities and relative accelerations are calculated from

$$\hat{\underline{v}}_i = \hat{\underline{v}}_{i-1} + \hat{\underline{s}}_i \dot{\underline{q}}_i \quad (\hat{\underline{v}}_0 \text{ known}) \quad (23)$$

and

$$\hat{\underline{a}}_i^r = \hat{\underline{a}}_{i-1}^r + \hat{\underline{s}}_i \ddot{\underline{q}}_i + \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{\underline{q}}_i, \quad (\hat{\underline{a}}_0^r = \hat{\underline{0}}) \quad (24)$$

composite-rigid-body inertias from

$$\hat{\underline{I}}_i^c = \hat{\underline{I}}_{i+1}^c + \hat{\underline{I}}_i, \quad (\hat{\underline{I}}_n^c = \hat{\underline{I}}_n) \quad (25)$$

and bias forces from

$$\hat{\underline{p}}_i = \hat{\underline{p}}_{i+1} + \hat{\underline{I}}_i \hat{\underline{a}}_i^r + \hat{\underline{v}}_i \hat{\times} \hat{\underline{I}}_i \hat{\underline{v}}_i. \quad (\hat{\underline{p}}_{n+1} = \hat{\underline{0}}) \quad (26)$$

Assuming no external force is applied to the base, the acceleration is computed from

$$\begin{aligned} \hat{\underline{I}}_0^c \hat{\underline{a}}_0 + \hat{\underline{p}}_0 &= \hat{\underline{0}} \\ \rightarrow \hat{\underline{a}}_0 &= - (\hat{\underline{I}}_0^c)^{-1} \hat{\underline{p}}_0. \end{aligned} \quad (27)$$

Once $\hat{\underline{a}}_0$ is known we can calculate all the $\hat{\underline{f}}_i$ from eq. (22), so it remains only to calculate the joint forces, which are given by

$$\underline{Q}_i = \hat{\underline{s}}_i^s \hat{\underline{f}}_i. \quad (28)$$

The computational requirement of this algorithm is considerably more than that of the Newton-Euler algorithm for a robot with a fixed base, but it is less than that of either of the forward dynamics algorithms for robots with a floating base, and its computational complexity is still $O(n)$.

8.5.2 Articulated-Body Hybrid Dynamics

The articulated-body algorithm described in chapter 6 can be modified to solve problems where the forces at some of the joints and the accelerations of the rest are given. For each joint, the equations in chapter 6 apply if the force is given and the acceleration is to be determined, but if the acceleration is given and the force is to be determined then the equations given below should be used. (Quantities are as defined in chapter 6.)

The relationship between the force applied to the handle of articulated body i and its acceleration is given by

$$\begin{aligned}\hat{\underline{f}} &= \hat{\underline{I}}_i^A \hat{\underline{a}} + \hat{\underline{p}}_i \\ &= \hat{\underline{I}}_i \hat{\underline{a}} + \hat{\underline{p}}_i^V \\ &\quad + \hat{\underline{I}}_{i+1}^A (\hat{\underline{a}} + \hat{\underline{v}}_{i+1} \hat{\underline{s}}_{i+1} \dot{\underline{q}}_{i+1} + \hat{\underline{s}}_{i+1} \ddot{\underline{q}}_{i+1}) + \hat{\underline{p}}_{i+1} ,\end{aligned}\quad (29)$$

so we have that

$$\hat{\underline{I}}_i^A = \hat{\underline{I}}_i + \hat{\underline{I}}_{i+1}^A \quad (30)$$

and

$$\hat{\underline{p}}_i = \hat{\underline{p}}_i^V + \hat{\underline{p}}_{i+1} + \hat{\underline{I}}_{i+1}^A (\hat{\underline{v}}_{i+1} \hat{\underline{s}}_{i+1} \dot{\underline{q}}_{i+1} + \hat{\underline{s}}_{i+1} \ddot{\underline{q}}_{i+1}) . \quad (31)$$

Eqs. (30 and 31) are used in place of eqs. (6.37 and 6.38). Eq. (6.41) can be used immediately, since there is no need to calculate $\ddot{\underline{q}}_i$ first, and the joint force can be calculated from

$$\underline{Q}_i = \hat{\underline{s}}_i^S (\hat{\underline{I}}_i^A \hat{\underline{a}}_i + \hat{\underline{p}}_i) . \quad (32)$$

Eq. (30) shows that, for the purposes of calculating inertias, if the

relative acceleration of two bodies is known then they are effectively rigidly connected.

Chapter 9

Contact, Impact, Kinematic Loops and Friction

9.1 Introduction

In this chapter I am concerned mainly with the problem of modelling contact and impact between a robot and its environment for the purposes of computer simulation of these phenomena. A simulator capable of a limited form of contact and impact modelling is described in [Featherstone 1983b].

When some part of a robot is in contact with an object a joint is formed having the special property that the reaction force due to kinematic constraints must be a physically possible contact force. An impossible contact force indicates that contact has been broken or that the mode of contact has changed. An impact occurs when two rigid bodies come into contact or their mode of contact changes so that the number of motion constraints increases. An impulse normally occurs between impacting rigid bodies causing their velocities to change so that they either bounce apart or come into sustained contact.

Contact between the robot and its environment normally results in the formation of kinematic loops. A kinematic loop occurs when a circuit can be traced from one link back to itself without traversing

any joint more than once. For example, a robot with a fixed base closes a kinematic loop when it makes contact with a fixed part of its environment, which is rigidly connected (however indirectly) to the base. The techniques described here for solving the dynamics of robots in contact are equally applicable to solving the dynamics of robot mechanisms containing kinematic loops.

The study of general methods for calculating the dynamics of rigid-body systems containing kinematic loops is really the province of mechanism dynamics, and much work has been done in this area (e.g. see [Paul 1975; Smith 1973; Smith et al. 1973; Uicker 1965; Uicker 1967; Orlandea et al. 1977]). Kinematic loops introduce two complications: firstly they introduce constraint equations (called loop equations) between the joint variables, so there is no longer a one-to-one correspondence between the joint variables and the motion freedom of the system, and secondly they introduce unknown reaction forces, the number of forces being equal to the number of motion constraints. The method I will use to solve the dynamics of such a system is to solve the open-loop dynamics by the composite-rigid-body method and augment the open-loop equations with the acceleration constraint equations. This method is essentially the same as that described in [Smith 1973] and the method of Lagrange multipliers described in [Paul 1975] (though these papers deal with planar mechanisms). Impact equations are derived following the same procedure as for contact, but with impulses in place of forces and velocity changes in place of accelerations. The effects of friction during impact and contact are ignored.

What makes the simulation of contact and impact more difficult than just allowing for kinematic loops are the geometric and kinematic problems of detecting the making, breaking and changing modes of contact. The solutions to these problems are outside the scope of this thesis, so I confine myself to a description in broad outline of how they might be solved.

A joint in contact with or striking a mechanical end stop is a special case of contact and impact which is much easier to solve than the general one. I show how this problem can be solved by the use of hybrid forward and inverse dynamics. The same mechanism can be used for the simulation of stick-slip motion caused by Coulombic friction in the joints.

Finally, I describe how articulated-body methods could be applied to problems involving kinematic loops. This is a speculative topic in that I do not describe any explicit algorithms, but describe instead the techniques and equations from which such algorithms could be developed.

9.2 Contact

Contact will be modelled as a multiple-degree-of-freedom joint connecting the two bodies in contact. The instantaneous motion space is given by $\hat{S}$ and the instantaneous reaction force space by $\hat{R} = \hat{S}^\perp$. The joint is assumed to be un-powered and to impose r degrees of motion constraint. The contact force is given by $\hat{\underline{f}}_C = \hat{R} \underline{f}$, where $\underline{f}$ is the vector of unknown constraint forces. The distinguishing feature of a contact is that only certain values of $\hat{\underline{f}}_C$ are physically

possible. If a calculation of the dynamics returns an impossible value for $\hat{\underline{f}}_c$ (as determined by a kinematic test procedure) then this indicates that a partial or complete loss of contact has occurred, so the dynamics should be recalculated with modified contact kinematics. During sustained contact the relative velocity of the two bodies satisfies the condition $\hat{\underline{R}}^s \hat{\underline{v}} = \underline{0}$ which can be differentiated to get a constraint on the relative acceleration.¹

The dynamics algorithm is based on the use of eq. (5.5), which is augmented by the inclusion of $\underline{f}$ into the vector of unknowns, and the inclusion of the acceleration constraint equations. I describe the method first for the simple case of a robot in contact with a fixed object, and then go on to describe more general cases.

I assume that the dynamics algorithms which follow are to be used in a simulator program where they are packaged into functions and called by a supervisory function. I assume also that this supervisor calls upon geometric and kinematic functions to handle collision detection, contact break detection, and the calculation of $\hat{\underline{R}}$ and $\dot{\hat{\underline{R}}}$ for use by the dynamics functions. All that a dynamics function has to do is work out the instantaneous dynamics of the system given the hypothesised contact kinematics.

9.2.1 Robot in Contact with Fixed Object

Consider the simple case of a straight-chain robot with a fixed base contacting a fixed object in its environment, as shown in figure

1. These are known as the first and second order loop equations respectively.

Figure 9.1 Robot in contact with fixed object

9.1. The object is fixed relative to the base and can be thought of as an extension of the base, so a kinematic loop is formed from the base through to the end effector and back to the base via the contact. Let the robot have n degrees of freedom, and let the contact impose r degrees of constraint. To solve the problem we must find the n unknown joint accelerations in $\ddot{\underline{q}}$ and the r unknown reaction forces in $\underline{f}$.

Let the robot's end effector apply a force $\hat{\underline{f}}_c$ to the object, which can be expressed as

$$\hat{\underline{f}}_c = \hat{\underline{R}} \underline{f} . \quad (1)$$

For sustained contact, the end effector velocity, $\hat{\underline{v}}_e$, must satisfy

$$\hat{\underline{R}}^S \hat{\underline{v}}_e = \underline{0} . \quad (2)$$

Differentiating this equation gives the constraint on the end effector acceleration, $\hat{\underline{a}}_e$, as

$$\hat{\underline{R}}^S \hat{\underline{a}}_e + \dot{\hat{\underline{R}}}^S \hat{\underline{v}}_e = \underline{0} . \quad (3)$$

The joint forces can be thought of as being made up from two components: that part which accounts for the motion of the robot in the absence of any contact and that part which accounts for the contact force in isolation. Since no dynamic effects are included in the second force, it is the joint force required to compensate statically for the contact force. Calling these components $\underline{Q}_m$ and $\underline{Q}_f$ respectively, we have that

$$\underline{Q} = \underline{Q}_m + \underline{Q}_f , \quad (4)$$

and the open-loop dynamic equation (taken from eq. (5.5)) becomes

$$H \ddot{\underline{q}} = \underline{Q}_m - \underline{C} . \quad (5)$$

In this instance, eq. (5) is more useful in the form

$$H \ddot{\underline{q}} + \underline{Q}_f = \underline{Q} - \underline{C} , \quad (6)$$

which puts all the unknowns on the left-hand side.

Eqs. (1, 3 and 6) essentially describe the dynamics, but to tie them together we need to relate the spatial quantities $\hat{\underline{a}}_e$ and $\hat{\underline{f}}_c$ to the joint-space quantities $\ddot{\underline{q}}$ and $\underline{Q}$. This is done by using the Jacobian matrix, which is the $6 \times n$ matrix defined by

$$\hat{J} = [\hat{\underline{s}}_1 \dots \hat{\underline{s}}_n] . \quad (7)$$

The Jacobian gives the end effector velocity in terms of the joint velocity as

$$\hat{\underline{v}}_e = \hat{J} \dot{\underline{q}} , \quad (8)$$

and the joint force required to support the force applied by the end effector as

$$\underline{Q}_f = \hat{J}^S \hat{f}_c . \quad (9)$$

$\hat{J}$ obeys the same rules for transformation and transpose as a subspace, and it performs a similar function in that it relates a non-spatial quantity to a spatial one, but it is a distinct kind of quantity. Differentiating eq. (8) gives the end effector acceleration in terms of the joint accelerations as

$$\underline{\hat{a}}_e = \hat{J} \ddot{\underline{q}} + \dot{\hat{J}} \underline{\dot{q}} ; \quad (10)$$

but $\dot{\hat{J}} \underline{\dot{q}}$ is the acceleration of the end effector due to velocity product effects, which is a readily available by-product of the inverse dynamics calculation of $\underline{C}$, so I will call it $\underline{\hat{a}}_e^{vp}$ and eq. (10) becomes

$$\underline{\hat{a}}_e = \hat{J} \ddot{\underline{q}} + \underline{\hat{a}}_e^{vp} . \quad (11)$$

Substituting eqs. (9 and 1) into eq. (6) gives

$$H \ddot{\underline{q}} + \hat{J}^S \hat{R} \underline{\hat{f}} = \underline{Q} - \underline{C} , \quad (12)$$

a set of n equations in $n+r$ unknowns, and substituting eq. (11) into eq. (3) and rearranging to bring $\ddot{\underline{q}}$ to the left-hand side gives

$$\hat{R}^S \hat{J} \ddot{\underline{q}} = - \hat{R}^S \underline{\hat{a}}_e^{vp} - \hat{R}^S \underline{\hat{v}}_e , \quad (13)$$

another r equations. Amalgamating eqs. (12 and 13) into a single set of $n+r$ equations gives

$$\left[\begin{array}{c|c} H & \hat{J}^S \hat{R} \\ \hline \hat{R}^S \hat{J} & 0 \end{array} \right] \left[\begin{array}{c} \ddot{\underline{q}} \\ \underline{\hat{f}} \end{array} \right] = \left[\begin{array}{c} \underline{Q} - \underline{C} \\ - \hat{R}^S \underline{\hat{a}}_e^{vp} - \hat{R}^S \underline{\hat{v}}_e \end{array} \right] . \quad (14)$$

The matrix is symmetric but not positive definite, and it is possible for the matrix to be singular. The singularities occur when the kinematic loop is in a locking position. A version of this algorithm

is implemented in the simulator described in [Featherstone 1983b].

9.2.2 Robot in Contact with Moving Object

I now generalise the situation by having the robot in contact with a general moving object. This object has non-zero velocity and acceleration, but more importantly its acceleration is affected by the contact force.

Let the object have velocity $\hat{\underline{v}}_o$ and acceleration $\hat{\underline{a}}_o$, and let the relationship between the force applied to the object and its acceleration be given by

$$\hat{\underline{a}}_o = \hat{\underline{I}}_o^{-1} \hat{\underline{f}}_c + \hat{\underline{b}}_o, \quad (15)$$

where $\hat{\underline{I}}_o^{-1}$ is an articulated-body inverse inertia (for generality) and $\hat{\underline{b}}_o$ a bias acceleration. The velocity constraint equation is now

$$\hat{\underline{R}}^s (\hat{\underline{v}}_e - \hat{\underline{v}}_o) = \underline{0}, \quad (16)$$

and the acceleration constraint equation is

$$\hat{\underline{R}}^s (\hat{\underline{a}}_e - \hat{\underline{a}}_o) + \dot{\hat{\underline{R}}}^s (\hat{\underline{v}}_e - \hat{\underline{v}}_o) = \underline{0}. \quad (17)$$

Substituting eqs. (1, 11 and 15) into eq. (17) and rearranging to put the unknowns on the left-hand side gives

$$\hat{\underline{R}}^s \hat{\underline{J}} \ddot{\underline{q}} - \hat{\underline{R}}^s \hat{\underline{I}}_o^{-1} \hat{\underline{R}} \underline{f} = \dot{\hat{\underline{R}}}^s (\hat{\underline{v}}_o - \hat{\underline{v}}_e) + \hat{\underline{R}}^s (\hat{\underline{b}}_o - \hat{\underline{a}}_e^{vp}), \quad (18)$$

which replaces eq. (13). The rest is unchanged, so the overall equation of motion is

$$\begin{bmatrix} H & \hat{J}^s \hat{R} \\ \hat{R}^s \hat{J} & -\hat{R}^s \hat{I}_o^{-1} \hat{R} \end{bmatrix} \begin{bmatrix} \ddot{\underline{q}} \\ \underline{f} \end{bmatrix} = \begin{bmatrix} \underline{Q} - \underline{C} \\ \hat{R}^s (\hat{\underline{v}}_o - \hat{\underline{v}}_e) + \hat{R}^s (\hat{\underline{b}}_o - \hat{\underline{a}}_e^{vp}) \end{bmatrix} \quad (19)$$

Eq. (14) is a special case of this equation where $\hat{I}_o^{-1}$, $\hat{\underline{v}}_o$ and $\hat{\underline{b}}_o$ are all zero.

9.2.3 Tree Structure Robots

If we allow the robot to have a tree structure and make the contact occur between link i and the object, then the following modifications are required. The velocity of link i , $\hat{\underline{v}}_i$, is given in terms of the joint velocities by

$$\hat{\underline{v}}_i = \hat{J}_i \dot{\underline{q}}, \quad (20)$$

where $\hat{J}_i$ is the Jacobian for link i and is defined as follows: if joint j lies on the path connecting link i to the base then column j of $\hat{J}_i$ is $\hat{\underline{s}}_j$ else it is $\hat{\underline{0}}$. The definition given earlier is a special case of this one. The quantities $\hat{\underline{v}}_i$, $\hat{\underline{a}}_i$ and $\hat{J}_i$ replace $\hat{\underline{v}}_e$, $\hat{\underline{a}}_e$ and $\hat{J}$, but otherwise there is no difference.

9.2.4 Kinematic Loops in the Robot Mechanism

To model a robot mechanism containing kinematic loops, treat the joints closing the loops as contacts between different links of the robot. Let there be a contact between links i and j of a tree structured robot. Let $\hat{\underline{v}}_i$ and $\hat{\underline{v}}_j$ be the velocities of the links and let $\hat{\underline{v}}_{ij}$ be the relative velocity

$$\hat{\underline{v}}_{ij} = \hat{\underline{v}}_i - \hat{\underline{v}}_j \quad (21)$$

Similarly, let $\hat{J}_i$ and $\hat{J}_j$ be the Jacobians of links i and j and let $\hat{J}_{ij}$ be defined by

$$\hat{J}_{ij} = \hat{J}_i - \hat{J}_j, \quad (22)$$

so that

$$\hat{\underline{v}}_{ij} = \hat{J}_{ij} \dot{\underline{q}} \quad (23)$$

This equation is differentiated to give the relative acceleration:

$$\begin{aligned} \hat{\underline{a}}_{ij} &= \hat{J}_{ij} \ddot{\underline{q}} + \dot{\hat{J}}_{ij} \dot{\underline{q}} \\ &= \hat{J}_{ij} \ddot{\underline{q}} + \hat{\underline{a}}_{ij}^{vp}, \end{aligned} \quad (24)$$

where

$$\hat{\underline{a}}_{ij}^{vp} = \hat{\underline{a}}_i^{vp} - \hat{\underline{a}}_j^{vp} \quad (25)$$

Both $\hat{\underline{a}}_i^{vp}$ and $\hat{\underline{a}}_j^{vp}$ should be available from the partial results of performing the inverse dynamics calculation of $\underline{C}$. The velocity constraint equation is now

$$\hat{R}^s \hat{\underline{v}}_{ij} = \underline{0}, \quad (26)$$

which on differentiating and substituting for $\hat{\underline{a}}_{ij}$ gives

$$\begin{aligned} \hat{R}^s \hat{\underline{a}}_{ij} + \dot{\hat{R}}^s \hat{\underline{v}}_{ij} &= \underline{0} \\ \rightarrow \hat{R}^s \hat{J}_{ij} \ddot{\underline{q}} &= -\dot{\hat{R}}^s \hat{\underline{v}}_{ij} - \hat{R}^s \hat{\underline{a}}_{ij}^{vp} \end{aligned} \quad (27)$$

Let $\hat{\underline{f}}_c$ be the contact force applied by link i to link j ; then the force applied by link j to link i is $-\hat{\underline{f}}_c$, and the value of $\underline{Q}_f$ needed to support both these forces is

$$\begin{aligned}
\underline{Q}_f &= \hat{J}_i^s \hat{\underline{f}}_c - \hat{J}_j^s \hat{\underline{f}}_c \\
&= \hat{J}_{ij}^s \hat{\underline{f}}_c .
\end{aligned} \tag{28}$$

Eqs. (27 and 28) replace eqs. (13 and 9) respectively, and the final equation of motion becomes

$$\begin{bmatrix} H & | & \hat{J}_{ij}^s \hat{\underline{R}} \\ \hline \hat{\underline{R}}^s \hat{J}_{ij} & | & 0 \end{bmatrix} \begin{bmatrix} \ddot{\underline{q}} \\ \underline{f} \end{bmatrix} = \begin{bmatrix} \underline{Q} - \underline{C} \\ \hline - \hat{\underline{R}}^s \hat{\underline{a}}_{ij}^{vp} - \hat{\underline{R}}^s \hat{\underline{v}}_{ij} \end{bmatrix} . \tag{29}$$

Once again, eq. (14) is a special case of this one.

If the loop is closed by a proper joint rather than a contact then there is no need for the geometric checks on $\underline{f}$. If the joint is powered then a few extra terms are needed. Inter-robot contact can also be treated in this way, regarding the separate robots as branches of a single tree robot which have only the base in common.

9.2.5 Multiple Kinematic Loops

If there are l independent kinematic loops closed simultaneously, then each loop will have its own loop equation, independent of the others, and $\underline{Q}_f$ is the sum of all the forces required to account for all the loop closure forces. So, for loop k , we have that

$$\hat{\underline{R}}_k^s \hat{J}_k \ddot{\underline{q}} = - \hat{\underline{R}}_k^s \hat{\underline{v}}_k - \hat{\underline{R}}_k^s \hat{\underline{a}}_k^{vp} , \tag{30}$$

where $\hat{\underline{R}}_k^s$ is the reaction space for loop k , $\hat{J}_k$ the Jacobian for loop k , and so on. The joint force needed to support all the contact forces is

$$\underline{Q}_f = \sum_k \hat{J}_k^s \hat{R}_k \underline{f}_k \quad (31)$$

Putting these equations together, we get

$$\begin{bmatrix} H & \hat{J}_1^s \hat{R}_1 & \hat{J}_2^s \hat{R}_2 & \dots & \hat{J}_1^s \hat{R}_1 \\ \hat{R}_1^s \hat{J}_1 & & & & \\ \hat{R}_2^s \hat{J}_2 & & & & \\ \vdots & & & & \\ \hat{R}_1^s \hat{J}_1 & & & & \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \underline{f}_1 \\ \underline{f}_2 \\ \vdots \\ \underline{f}_1 \end{bmatrix} = \begin{bmatrix} \underline{Q} - \underline{C} \\ -\hat{R}_1^s \hat{a}_1^{vp} - \hat{R}_1^s \hat{v}_1 \\ -\hat{R}_2^s \hat{a}_2^{vp} - \hat{R}_2^s \hat{v}_2 \\ \vdots \\ -\hat{R}_1^s \hat{a}_1^{vp} - \hat{R}_1^s \hat{v}_1 \end{bmatrix} \quad (32)$$

We can use eq. (32) to simulate an arbitrary kinematic network. Do not be deceived by the size of the zero area in the matrix - if the robot is to have any motion freedom at all, then the number of constraints must be less than the number of open-loop degrees of freedom (in general), so the size of the zero portion is less than that of H.

9.2.6 Floating-Base Robots

No additional complications arise from progressing to floating-base robots provided that such a robot is treated as a fixed-base robot with an un-powered fictitious joint between its real base and a fixed reference point.

9.3 The Kinematics of Contact

The most difficult problems facing a contact modelling system are the kinematic and geometric ones of detecting the making and breaking of contact and determining its kinematics. The necessary operations are:

- 1) collision detection (including increased contact constraints),
- 2) the calculation of $\hat{R}$ and $\dot{\hat{R}}$ assuming sustained contact, and
- 3) contact break detection (including partial contact breaks).

The first step in tackling this problem is to describe the shapes of the robot's links and the objects in its environment in a form suitable for computer processing. This can be done using the techniques of geometric body modelling [Requicha and Voelcker 1982; Baer et al. 1979]. Given the shapes of the bodies and their positions at any instant, it is possible to determine whether any of them intersect. This is the interference detection problem, and is receiving much attention [Boyse 1979; Tilove 1981]. The collision detection problem is that of determining at what instant two moving bodies begin to intersect, and is being studied in connection with the problem of finding collision-free paths for moving objects [Cameron 1984; Boyse 1979; de Pennington et al. 1983; Udupa 1977]. For the purposes of dynamics simulation, it is convenient to regard any event where the mode of contact changes so that the number of motion constraints between two bodies increases as a collision. As far as I know, no work is being done on this 'increased contact detection' problem.

The calculation of $\hat{R}$ and $\dot{\hat{R}}$ can be performed given descriptions of the surfaces in contact and the velocities of the bodies, and the detection of partial or complete contact breaks (i.e. a change in the mode of contact reducing the number of motion constraints) can be tackled using the techniques developed by Ohwovoriole [1980]. Both are difficult problems to solve in general.

The general methods would be very time-consuming if implemented on a computer, but fortunately there are many special cases which are much easier to handle. For example, figure 9.2 shows two bodies, b_1 and b_2 , where a vertex of b_1 is in contact with a plane surface on

Figure 9.2 Single point contact

b_2 . b_1 may apply any force to b_2 passing through the point of contact and in the direction of the inward-pointing surface normal $\hat{n}$. In this case $\hat{R}$ is a 6×1 matrix equal to $\hat{n}$, and the test for broken contact is that $\underline{f}$, which is now a scalar, be negative. The treatment of an r -point contact ($r \leq 6$), as shown in figure 9.3, is a straightforward extension of one-point contact provided that the contact normals are linearly independent. In this case $\hat{R}$ is the $6 \times r$ matrix composed of the vectors $\hat{n}_1 \dots \hat{n}_r$, and the test for a partial contact break is that any element of $\underline{f}$ be negative. The point contact

Figure 9.3 Multiple point contact

corresponding to the negative element is broken. $\dot{\hat{R}}$ is composed of the vectors $\dot{\hat{n}}_1 \dots \dot{\hat{n}}_r$ which are given by $\dot{\hat{n}}_i = \hat{v}_i \hat{\times} \hat{n}_i$, where $\hat{v}_i$ is the linear velocity of the i^{th} point of contact.

These are commonly occurring special cases, and it is probably worthwhile taking these and other similar cases into account to reduce the computational cost of the kinematic and geometric calculations (which, at the time of writing, would greatly exceed the cost of the dynamics calculations).

9.4 Impact

An impact occurs when two rigid bodies come into contact or their mode of contact changes to one with increased motion constraint. In general, when this happens the relative velocity of the two bodies does not satisfy the new motion constraints, and an impulse occurs

between them causing an instantaneous change in velocity so that they either bounce apart or their velocities are made to comply with the constraints. The parameter controlling the response to an impact is the coefficient of restitution, e . This is the ratio of the absolute magnitudes before and after impact of the components of the relative velocity violating the motion constraints of the contact. I assume that e is a scalar. If two bodies have relative velocity $\underline{\hat{v}}$ just before impact and $\underline{\hat{v}} + \delta \underline{\hat{v}}$ just after, where $\delta \underline{\hat{v}}$ is the change in velocity due to the impulse, and the reaction force space at the moment of impact is $\underline{\hat{R}}$, then

$$\begin{aligned} \underline{\hat{R}}^S \underline{\hat{v}} &= -e \underline{\hat{R}}^S (\underline{\hat{v}} + \delta \underline{\hat{v}}) \\ \rightarrow \underline{\hat{R}}^S \delta \underline{\hat{v}} &= - (1 + e) \underline{\hat{R}}^S \underline{\hat{v}}. \end{aligned} \quad (33)$$

This equation only partially determines $\delta \underline{\hat{v}}$ - to determine it completely requires a knowledge of the inertias of the bodies. If $e=0$ then a perfectly inelastic collision takes place and the velocity after the impact complies with the constraints. If $e>0$ then a rebound will occur, and if $e=1$ the collision is perfectly elastic.

The impulsive equation of motion for an open-loop system can be obtained from eq. (6) as

$$H \delta \underline{\dot{q}} + \underline{Q}'_f = \underline{Q}', \quad (34)$$

where $\delta \underline{\dot{q}}$ is the change in joint velocities, $\underline{Q}'_f$ is the joint-space impulse needed to produce the impulse applied to the object, and $\underline{Q}'$ is the joint-space impulse vector. It is possible for impulses to appear about the joint axes if, for example, the joint hits a mechanical end stop (see later section), but I will assume here that $\underline{Q}' = \underline{0}$. The equations governing the motion of the robot during impact are

derived in the same way as the equations for contact, but they are slightly simpler since we can ignore forces and there is no need to differentiate the velocity constraint equation.

Consider first the situation where the robot's end effector strikes a fixed object. This is the impact version of the first situation described under contact. The impact velocity equation is

$$\hat{R}^S \delta \hat{\underline{v}}_e = - (1 + e) \hat{R}^S \hat{\underline{v}}_e, \quad (35)$$

where $\delta \hat{\underline{v}}_e$ is the change in end effector velocity and is related to $\delta \dot{\underline{q}}$ by

$$\delta \hat{\underline{v}}_e = \hat{J} \delta \dot{\underline{q}}. \quad (36)$$

So we have that

$$\hat{R}^S \hat{J} \delta \dot{\underline{q}} = - (1 + e) \hat{R}^S \hat{\underline{v}}_e. \quad (37)$$

$\underline{Q}'_f$ is given by

$$\underline{Q}'_f = \hat{J}^S \hat{\underline{f}}'_c = \hat{J}^S \hat{R} \underline{f}', \quad (38)$$

where $\hat{\underline{f}}'_c$ is the spatial impulse applied to the object and $\underline{f}'$ the r-vector of unknown impulse coefficients. Substituting eq. (38) into eq. (34) and amalgamating the result with eq. (37) gives

$$\begin{bmatrix} H & \hat{J}^S \hat{R} \\ \hat{R}^S \hat{J} & 0 \end{bmatrix} \begin{bmatrix} \delta \dot{\underline{q}} \\ \underline{f}' \end{bmatrix} = \begin{bmatrix} 0 \\ -(1+e) \hat{R}^S \hat{\underline{v}}_e \end{bmatrix}. \quad (39)$$

The matrix part of this equation is the same as that of eq. (14), only the vectors have changed.

Equations can be obtained in a similar manner for the impact equivalents of the other situations described under contact, but rather than reproduce all the working I just give the final equations below.

Robot striking moving object:

$$\begin{bmatrix} H & | & \hat{J}^s \hat{R} \\ \hline \hat{R}^s \hat{J} & | & -\hat{R}^s \hat{I}_O^{-1} \hat{R} \end{bmatrix} \begin{bmatrix} \delta \dot{q} \\ \underline{f'} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \hline -(1+e) \hat{R}^s (\hat{v}_e - \hat{v}_O) \end{bmatrix} \quad (40)$$

Impact between links i and j of a robot:

$$\begin{bmatrix} H & | & \hat{J}_{ij}^s \hat{R} \\ \hline \hat{R}^s \hat{J}_{ij} & | & 0 \end{bmatrix} \begin{bmatrix} \delta \dot{q} \\ \underline{f'} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \hline -(1+e) \hat{R}^s \hat{v}_{ij} \end{bmatrix} \quad (41)$$

Impact in a system with many loops:

$$\begin{bmatrix} H & | & \hat{J}_1^s \hat{R}_1 & \hat{J}_2^s \hat{R}_2 & \dots & \hat{J}_1^s \hat{R}_1 \\ \hline \hat{R}_1^s \hat{J}_1 & & & & & \\ \hat{R}_2^s \hat{J}_2 & & & & & \\ \vdots & & & & & \\ \hat{R}_1^s \hat{J}_1 & & & & & \end{bmatrix} \begin{bmatrix} \delta \dot{q} \\ \underline{f'_1} \\ \underline{f'_2} \\ \vdots \\ \underline{f'_1} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \hline -(1+e) \hat{R}_1^s \hat{v}_1 \\ -(1+e) \hat{R}_2^s \hat{v}_2 \\ \vdots \\ -(1+e) \hat{R}_1^s \hat{v}_1 \end{bmatrix} \quad (42)$$

9.5 Joint End Stops

Joint end stops present special cases of contact and impact which are easier to handle than the general cases. Considering only one-degree-of-freedom joints, the effect of mechanical end stops is to impose upper and lower limits on the range of joint motion. When a joint reaches an end stop an impact occurs, and if a joint is at an

end stop then acceleration in one direction is prevented by a reaction force from the end stop. The geometric problems of collision detection etc. are now trivial, and since the constraints are already expressed in joint space there is no need to use the Jacobian.

Suppose that joint i is in contact with an end stop. Assuming that the robot does not accelerate away from the end stop, the equation of motion is

$$H \ddot{\underline{q}} + \underline{Q}_f = \underline{Q} - \underline{C} , \quad (43)$$

where element i of $\ddot{\underline{q}}$ is zero, element i of $\underline{Q}_f$ is the unknown force applied to the end stop, and all other elements of $\underline{Q}_f$ are zero. Rearranging this equation to get all the unknowns into a single vector produces

$$\begin{bmatrix} H & 0 & \vdots \\ \vdots & 1 & \vdots \\ \vdots & 0 & H \end{bmatrix} \begin{bmatrix} \ddot{\underline{q}} \\ Q_{fi} \\ \ddot{\underline{q}} \end{bmatrix} = \underline{Q} - \underline{C} , \quad (44)$$

where column i of H has been replaced by a column of zeroes except for the element on the diagonal, which is one, and the position held by $\ddot{q}_i$ is now occupied by Q_{fi} . The inertia matrix is still positive definite, but no longer symmetric. If Q_{fi} turns out to have the wrong sign, then the dynamics must be recalculated assuming that the joint is no longer in contact with the end stop.

The procedure is easily extended for the case where there are several joints simultaneously at end stops, one column of H being modified for each such joint. What we are doing is solving a hybrid forward-inverse dynamics problem using an algorithm based on the composite-rigid-body method. We could also solve this problem using

the hybrid dynamics algorithm described in chapter 8.

The impact version of eq. (44) is

$$\begin{bmatrix} H & 0 \\ & 1 \\ & 0 & H \end{bmatrix} \begin{bmatrix} \delta \dot{q} \\ Q'_{fi} \\ \delta \dot{q} \end{bmatrix} = \begin{bmatrix} H^i \end{bmatrix} (1 + e) \dot{q}_i, \quad (45)$$

(where H^i is column i of H) assuming that

$$\delta \dot{q}_i = - (1 + e) \dot{q}_i. \quad (46)$$

Since joint end stops are nothing more than contacts between adjacent links, they may easily be combined with other forms of contact. However, it is not possible simply to replace eqs. (6 and 34) with eqs. (44 and 45).

Figure 9.4 Friction vs. joint velocity

9.6 Coulombic Friction at the Joints

I consider friction here because the simulation of stick-slip motion due to Coulombic friction at the joints is easily done using the same mechanism as for joint end stops. To keep things simple, I define the Coulombic friction force acting at a one-degree-of-freedom

joint as follows: if the joint is moving then the frictional force is C in the direction opposed to the motion, but if the joint velocity is zero then the frictional force is that required to achieve zero acceleration provided that its magnitude does not exceed C , otherwise it is C in the direction opposed to the acceleration. The situation is shown in figure 9.4: if the joint velocity is non-zero then the frictional force is defined, but if not then it may take on any value within a given range depending on prevailing dynamic conditions.

To implement this model, each joint is considered to be either free or stuck. The hybrid dynamics is used to calculate the accelerations of all the free joints and the forces required at each of the stuck joints to see if they remain stuck. If any of the sticking forces are too large then the joints affected are freed and the dynamics redone. This system has been implemented on the simulator described in [Featherstone 1983b].

9.7 The Articulated-Body Method for Kinematic Loops

I do not have a complete algorithm for calculating the dynamics of systems containing kinematic loops via an articulated-body method, but I describe here in outline how such an algorithm should work and I give equations for most of the necessary calculations. I do not believe that such an algorithm would be efficient enough to be of practical use except in highly unusual circumstances.

The problem with the articulated-body method of chapter 6 is that it relies on the fact that all interaction between an articulated body and the rest of a system occurs through a single member, the

handle. To allow for kinematic loops we must be able to work with articulated bodies with more than one handle. To do this we need to know the articulated-body inertias of every handle and the cross-inertias between every pair of handles. In the rest of this section I describe how these quantities can be used and how they can be calculated.

9.7.1 Calculating the Motion of a Multi-Handled Articulated Body

Suppose that an articulated-body has h handles, and for each handle we know the articulated-body inertia $\hat{I}_i$ and the bias acceleration $\hat{b}_i$ and for each pair of handles we know the inverse cross-inertias $\hat{I}_{ij}^{-1}$ and $\hat{I}_{ji}^{-1}$. $\hat{I}_{ij}^{-1}$ gives the contribution to the acceleration of handle number i caused by a force acting on handle number j according to

$$\hat{a}_i = \hat{I}_{ij}^{-1} \hat{f}_j + \hat{b}_i. \quad (47)$$

If forces are applied to several handles then the acceleration of any one handle is the sum of the contributions of each of the forces. The equation of motion for all the handles is

$$\begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_h \end{bmatrix} = \begin{bmatrix} \hat{I}_1^{-1} & \hat{I}_{12}^{-1} & \cdots & \hat{I}_{1h}^{-1} \\ \hat{I}_{21}^{-1} & \hat{I}_2^{-1} & \cdots & \hat{I}_{2h}^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{I}_{h1}^{-1} & \hat{I}_{h2}^{-1} & \cdots & \hat{I}_h^{-1} \end{bmatrix} \begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_h \end{bmatrix} + \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{b}_h \end{bmatrix}. \quad (48)$$

Suppose that handles i and j are connected to form a loop, then we have that

$$\hat{\underline{f}}_i = -\hat{\underline{f}}_j = \hat{\underline{R}} \underline{f} \quad (49)$$

and

$$\hat{\underline{R}}^s (\hat{\underline{a}}_i - \hat{\underline{a}}_j) + \dot{\hat{\underline{R}}}^s (\hat{\underline{v}}_i - \hat{\underline{v}}_j) = \underline{0} \quad (50)$$

The motion of a system containing kinematic loops can be found by solving the set of simultaneous equations formed by eq. (48) and a number of equations like eqs. (49 and 50). An algorithm based on these equations would be $O(l^3)$, where l is the number of loops, so it is conceivable that it would run faster than the other algorithms described in this chapter for a system with a large number of links but only a few loops.

Figure 9.5 Part of an articulated body

9.7.2 Calculating Several Articulated-Body Inertias from one Articulated Body

Referring to figure 9.5, let b_1 and b_2 be the handles of the open-loop articulated bodies B_1 and B_2 which form the articulated body B when connected as shown. We are interested in calculating the

articulated-body inertias of several handles in the one articulated body B. Let B_1 and B_2 have articulated-body inertias $\hat{I}_1$ and $\hat{I}_2$ respectively, and let the articulated-body inertias of B with the handles b_1 and b_2 be $\hat{I}_1^B$ and $\hat{I}_2^B$ respectively. From eq. (6.17) we have that

$$\hat{I}_1^B = \hat{I}_1 + \hat{I}_2 - \frac{\hat{I}_2 \underline{\hat{s}} \underline{\hat{s}}^s \hat{I}_2}{\underline{\hat{s}}^s \hat{I}_2 \underline{\hat{s}}} \quad (51)$$

and

$$\hat{I}_2^B = \hat{I}_1 + \hat{I}_2 - \frac{\hat{I}_1 \underline{\hat{s}} \underline{\hat{s}}^s \hat{I}_1}{\underline{\hat{s}}^s \hat{I}_1 \underline{\hat{s}}} \quad (52)$$

$\hat{I}_2^B$ can be expressed in terms of $\hat{I}_1^B$ and $\hat{I}_2$ as

$$\hat{I}_2^B = \hat{I}_1^B + \frac{\hat{I}_2 \underline{\hat{s}} \underline{\hat{s}}^s \hat{I}_2}{\underline{\hat{s}}^s \hat{I}_2 \underline{\hat{s}}} - \frac{\hat{I}_1^B \underline{\hat{s}} \underline{\hat{s}}^s \hat{I}_1^B}{\underline{\hat{s}}^s \hat{I}_1^B \underline{\hat{s}}} \quad (53)$$

The corresponding equation for bias forces is

$$\begin{aligned} \hat{p}_2^B = \hat{p}_1^B - \hat{I}_2 \underline{\hat{s}} \frac{Q - \underline{\hat{s}}^s \hat{p}_2}{\underline{\hat{s}}^s \hat{I}_2 \underline{\hat{s}}} \\ + \hat{I}_1^B \left(\underline{\hat{v}}_2 \hat{x} \underline{\hat{v}}_1 + \underline{\hat{s}} \frac{2Q - \underline{\hat{s}}^s \left(\hat{I}_1^B \underline{\hat{v}}_2 \hat{x} \underline{\hat{v}}_1 + \hat{p}_1^B \right)}{\underline{\hat{s}}^s \hat{I}_1^B \underline{\hat{s}}} \right) \end{aligned} \quad (54)$$

With eqs. (53 and 54) we can calculate every articulated-body inertia of B by the following route:

- 1) calculate one articulated-body inertia of B using the method of chapter 6,

- 2) use the partial results from 1 and eqs. (53 and 54) to calculate the articulated-body inertia of B for every other link (of interest).

Making good use of partial results, it takes only about twice as long to calculate the articulated-body inertia for every link as it does for one.

9.7.3 Calculating Cross-Inertias

The simplest way to calculate cross-inertias is to calculate them from acceleration propagators. The acceleration propagator $\hat{A}_{ij}$ gives the acceleration of link i (ignoring bias accelerations) in terms of the acceleration of link j, which was induced by a test force on link j, according to

$$\underline{\hat{a}}_i = \hat{A}_{ij} \underline{\hat{a}}_j . \quad (55)$$

The inverse cross-inertia is expressed in terms of the acceleration propagator as

$$\hat{I}_{ij}^{-1} = \hat{A}_{ij} \hat{I}_j^{-1} . \quad (56)$$

Referring to figure 9.5, if an acceleration $\underline{\hat{a}}_1$ is induced in b_1 by a force acting on some link in B_1 , then the acceleration $\underline{\hat{a}}_2$ induced in b_2 is given by

$$\underline{\hat{a}}_2 = \underline{\hat{a}}_1 + \underline{\hat{s}} \alpha . \quad (57)$$

The force $\underline{\hat{f}}$ which produces this acceleration is given by

$$\underline{\hat{f}} = \hat{I}_2 \underline{\hat{a}}_2 \quad (58)$$

and satisfies

$$\underline{\hat{s}}^S \underline{\hat{f}} = 0, \quad (59)$$

from which we get

$$\begin{aligned} \underline{\hat{s}}^S \hat{I}_2 (\underline{\hat{a}}_1 + \underline{\hat{s}} \alpha) &= 0 \\ \rightarrow \alpha &= - \frac{\underline{\hat{s}}^S \hat{I}_2 \underline{\hat{a}}_1}{\underline{\hat{s}}^S \hat{I}_2 \underline{\hat{s}}} \\ \rightarrow \underline{\hat{a}}_2 &= (\hat{1} - \frac{\underline{\hat{s}} \underline{\hat{s}}^S \hat{I}_2}{\underline{\hat{s}}^S \hat{I}_2 \underline{\hat{s}}}) \underline{\hat{a}}_1 \\ \rightarrow \hat{A}_{21} &= \hat{1} - \frac{\underline{\hat{s}} \underline{\hat{s}}^S \hat{I}_2}{\underline{\hat{s}}^S \hat{I}_2 \underline{\hat{s}}}. \end{aligned} \quad (60)$$

$\hat{I}_2^B$ may be substituted for $\hat{I}_2$ in eq. (60) since $\hat{I}_2^B \underline{\hat{s}} = \hat{I}_2 \underline{\hat{s}}$. An equation like eq. (60) can be used to calculate acceleration propagators from any link to any other once a sufficient number of articulated-body inertias are known.

9.7.4 Conclusion

An algorithm can be devised for calculating the forward dynamics of an arbitrary kinematic network, based on eqs. (48-50, 53-54, 56 and 60) and the equations and techniques from chapters 6 and 8. Such an algorithm would be complicated and too inefficient for practical use. However, this algorithm would exhibit a computational complexity of $O(n+1^3)$, where n is the number of links and 1 the number of

loops, so it would be more efficient than the other algorithms described in this chapter for a system with a large number of links and a small number of loops.

There are other uses for the techniques described here. For example, the articulated-body inertia of the end effector of a robot is a good indicator of its performance in complying with external position constraints: the smaller the inertia the faster the complying movement can be performed and the less the force required to do it. The techniques described here can be used to calculate this quantity, thus allowing the possibility of minimising it as part of the design process, or of using it in the planning of tasks involving compliant motions. These are promising areas for future research.

Bibliography

[Armstrong 1979]

W. W. Armstrong, "Recursive Solution to the Equations of Motion of an n Link Manipulator", Proc. 5th World Congress on the Theory of Machines and Mechanisms, Montreal, vol. 2, pp. 1343-1346, July 1979.

[Backhouse and Rees Jones 1981]

C. Backhouse, J. Rees Jones, "Analogue Computer Simulation of a Robot Manipulator", I. Mech. E. Journal of Mechanical Engineering Science, vol. 23, no. 3, pp. 121-129, 1981.

[Baer et al. 1979]

A. Baer, C. Eastman, M. Henrion, "Geometric Modelling - A Survey", Computer-Aided Design, vol. 11, no. 5, 1979, pp. 253-272.

[Ball 1900]

R. S. Ball, "A Treatise on the Theory of Screws", Cambridge Univ. Press, London, 1900.

[Benati et al. 1980]

M. Benati, S. Gaglio, P. Morasso, V. Tagliasco, R. Zaccaria, "Anthropomorphic Robots" (in two parts), Biological Cybernetics, vol. 38, pp. 125-150, 1980.

[Bottema and Roth 1979]

O. Bottema, B. Roth, "Theoretical Kinematics", North Holland, Amsterdam, 1979.

[Boyse 1979]

J. W. Boyse, "Interference Detection Among Solids and Surfaces", Comm. ACM, vol. 22, no. 1, 1979.

[Brady et al. 1982]

J. M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, M. T. Mason, (editors) "Robot Motion: Planning and Control", MIT Press, Cambridge Mass. & London, 1982. (Note. This book contains the following papers which are also listed separately in this bibliography: Hollerbach [1980], Luh et al. [1980a], Luh et al. [1980b], Walker and Orin [1982].)

[Brand 1953]

L. Brand, "Vector and Tensor Analysis", 4th ed., Wiley / Chapman and Hall, New York / London, 1953.

[Cameron 1984]

S. A. Cameron, "Modelling Solids in Motion" (Ph. D. Thesis), Department of Artificial Intelligence, University of Edinburgh, 1984.

[Clifford 1882]

W. K. Clifford, "Preliminary Sketch of Biquaternions" (1873), appears in W. K. Clifford, "Mathematical Papers", Macmillan, London, 1882. (Reprinted Chelsea Pub. Co., New York, 1968.)

[Denavit and Hartenberg 1955]

J. Denavit, R. S. Hartenberg, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices", Trans. ASME Jnl. Applied Mechanics, vol. 22, pp. 215-221, 1955.

[Dimentberg 1965]

F. M. Dimentberg, "The Screw Calculus and its Applications in Mechanics" (in Russian), Izd. Nauka, Moscow, 1965. English translation: Foreign Technology Div., Wright-Patterson AFB, Ohio, part no. AD 680 993, 1968.

[Duffy 1980]

J. Duffy, "Analysis of Mechanisms and Robot Manipulators", Edward Arnold, London, 1980.

[Featherstone 1982]

R. Featherstone, "A Program for Simulating Robot Dynamics", Dept. of Artificial Intelligence, University of Edinburgh, Working Paper no. 116, 1982.

[Featherstone 1983a]

R. Featherstone, "The Calculation of Robot Dynamics using Articulated-Body Inertias", Robotics Research, vol. 2, no. 1, pp. 13-30, 1983.

[Featherstone 1983b]

R. Featherstone, "Update on the Robot Simulator Program", Dept. of Artificial Intelligence, University of Edinburgh, Working Paper no. 155, 1983.

[Hollerbach 1980]

J. M. Hollerbach, "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity", IEEE Trans. Systems, Man and Cybernetics, vol. SMC-10, no. 11, pp. 730-736, 1980.

[Hollerbach and Sahar 1983]

J. M. Hollerbach, G. Sahar, "Wrist-Partitioned Inverse Kinematic Accelerations and Manipulator Dynamics", Robotics Research, vol. 2, no. 4, pp. 61-76, 1983.

[Hooker and Margulies 1965]

W. W. Hooker, G. Margulies, "The Dynamical Attitude Equations for an n-Body Satellite", Jnl. Astronautical Sciences, vol. 12, no. 4, pp. 123-128, 1965.

[Hooker 1970]

W. W. Hooker, "A Set of r Dynamical Attitude Equations for an Arbitrary n-Body Satellite having r Rotational Degrees of Freedom", AIAA Journal, vol. 8(2), no. 7, pp. 1205-1207, 1970.

[Horn and Raibert 1977]

B. K. P. Horn, M. H. Raibert, "Configuration Space Control", MIT AI Lab., AI Memo no. 458, 1977.

[Hunt 1978]

K. H. Hunt, "Kinematic Geometry of Mechanisms", Clarendon Press, Oxford, 1978.

[Huston et al. 1978]

R. L. Huston, C. E. Passerello, M. W. Harlow, "Dynamics of Multirigid-Body Systems", Trans. ASME Jnl. Applied Mechanics, vol. 45, pp. 889-894, 1978.

[Kahn 1969]

M. E. Kahn, "The Near Minimum-Time Control of Open-Loop Articulated Kinematic Chains" (Ph. D. Thesis), Stanford AI Lab. AIM 106, 1969.

[Kane and Levinson 1983]

T. R. Kane, D. A. Levinson, "The Use of Kane's Dynamical Equations in Robotics", Robotics Research, vol. 2, no. 3, pp. 3-21, 1983.

[Lang 1972]

S. Lang, "Linear Algebra", Addison-Wesley, Reading Mass., 1972.

[Lipkin and Duffy 1982]

H. Lipkin, J. Duffy, "Analysis of Industrial Robots via the Theory of Screws", Proc. 12th International Symposium on Industrial Robots, Paris, June 9th-11th, pp. 359-370, 1982.

[Luh et al. 1980a]

J. Y. S. Luh, M. W. Walker, R. P. C. Paul, "On-Line Computational Scheme for Mechanical Manipulators", Trans. ASME Jnl. Dynamic Systems Measurement and Control, vol. 102, no. 2, pp. 69-76, 1980. (Errata on p. 173.)

[Luh et al. 1980b]

J. Y. S. Luh, M. W. Walker, R. P. C. Paul, "Resolved Acceleration Control of Mechanical Manipulators", IEEE Trans. Automatic Control, vol. AC-25, no. 3, pp. 468-474, 1980.

[Mises 1924a]

R. von Mises, "Motorrechnung, ein neues Hilfsmittel der Mechanik" [Motor Calculus: a new Tool for Mechanics] (in German), Zeitschrift für Angewandte Mathematik und Mechanik, vol. 4, no. 2, pp. 155-181, 1924.

[Mises 1924b]

R. von Mises, "Anwendungen der Motorrechnung" [Applications of Motor Calculus] (in German), Zeitschrift für Angewandte Mathematik und Mechanik, vol. 4, no. 3, pp. 193-213, 1924.

[Mufti 1979]

I. H. Mufti, "Dynamics of Multi-Body Systems", National Research Council of Canada, Ottawa, paper MK-29, NRC no. 18017, 1979.

[Mufti 1983]

I. H. Mufti, "Equations of Motion for Rigid Multibody Systems", National Research Council of Canada, Ottawa, paper DM-3, NRC no. 22797, 1983.

[Ohwovoriote 1980]

M. S. Ohwovoriote, "An Extension of Screw Theory and its Application to the Automation of Industrial Assemblies" (Ph. D. Thesis), Stanford AI Lab. memo AIM-338, 1980.

[Orin et al. 1979]

D. E. Orin, R. B. Mc Ghee, M. Vukobratović, G. Hartoch, "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilising Newton-Euler Methods", Mathematical Biosciences, vol. 43, pp. 107-130, 1979.

[Orlande et al. 1977]

N. Orlande, D. A. Calahan, M. A. Chace, "A Sparsity-Oriented Approach to the Dynamic Analysis and Design of Mechanical Systems" (in 2 parts), Trans. ASME Jnl. Engineering for Industry, vol. 99, no. 3, pp. 773-784, 1977.

[Paul 1975]

B. Paul, "Analytical Dynamics of Mechanisms - A Computer Oriented Overview", Mechanism and Machine Theory, vol. 10, pp. 481-507, 1975.

[Paul 1981]

R. Paul, "Robot Manipulators: Mathematics, Programming and Control", MIT Press, Cambridge Mass. & London, 1981.

[de Pennington et al. 1983]

A. de Pennington, M. S. Bloor, M. Balila, "Geometric Modelling: A Contribution Towards Intelligent Robots", Proc. 13th International Symposium on Industrial Robots, Chicago, April 17th-21st, vol. 1, section 7, pp. 35-54, 1983.

[Raibert et al. 1983]

M. H. Raibert, H. B. Brown Jr., M. Chepponis, E. Hastings, S. S. Murthy, F. C. Wimberly, "Dynamically Stable Legged Locomotion", The Robotics Institute, Carnegie-Mellon University, CMU-RI-TR-83-1, 1983.

- [Requicha and Voelcker 1982]

A. A. G. Requicha, H. B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment", IEEE Computer Graphics and Applications, vol. 2, no. 2, pp. 9-24, 1982.

[Rooney 1975]

J. Rooney, "On the Principle of Transference", Proc. 4th World Congress on the Theory of Machines and Mechanisms, Newcastle upon Tyne, vol. 5, pp. 1089-1094, Sept. 8th-13th, 1975.

[Rooney 1977]

J. Rooney, "A Survey of Representations of Spatial Rotation about a Fixed Point", Environment and Planning B, vol. 4, pp. 185-210, 1977.

[Rooney 1978]

J. Rooney, "A Comparison of Representations of General Spatial Screw Displacement", Environment and Planning B, vol. 5, pp. 45-88, 1978.

[Sheth and Uicker 1971]

P. N. Sheth, J. J. Uicker Jr., "A Generalised Symbolic Notation for Mechanisms", Trans. ASME Jnl. Engineering for Industry, vol. 93, pp. 102-112, 1971.

[Silver 1982]

W. Silver, "On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators", Robotics Research, vol. 1, no. 2, pp. 60-70, 1982.

[Smith 1973]

D. A. Smith, "Reaction Force Analysis in Generalised Machine Systems", Trans. ASME Jnl. Engineering for Industry, vol. 95, pp. 617-623, 1973.

[Smith et al. 1973]

D. A. Smith, M. A. Chace, A. C. Rubens, "The Automatic Generation of a Mathematical Model for Machinery Systems", Trans. ASME Jnl. Engineering for Industry, vol. 95, pp. 629-635, 1973.

[Stepanenko and Vukobratović 1976]

Yu. Stepanenko, M. Vukobratović, "Dynamics of Articulated Open Chain Active Mechanisms", Mathematical Biosciences, vol. 28, pp. 137-170, 1976.

[Swartz 1982]

N. M. Swartz, "Arm Dynamics Simulation", The Robotics Institute, Carnegie-Mellon University, CMU-RI-TR-82-17, 1982.

[Thomas and Tesar 1982]

M. Thomas, D. Tesar, "Dynamic Modelling of Serial Manipulator Arms", Trans. ASME Jnl. Dynamic Systems Measurement and Control, vol. 104, pp. 218-228, 1982.

[Tilove 1981]

R. B. Tilove, "Exploiting Spatial and Structural Locality in Geometric Modelling" (Ph. D. Thesis), College of Engineering and Applied Science, University of Rochester, 1981. (Also published as TM 38, Production Automation Project, Univ. of Rochester.)

[Turney et al. 1980]

J. L. Turney, T. N. Mudge, C. S. G. Lee, "Equivalence of Two Formulations for Robot Arm Dynamics", Center for Robotics and Integrated Manufacturing (CRIM), Robot Systems Division, University of Michigan, 1980.

[Udupa 1977]

S. M. Udupa, "Collision Detection and Avoidance in Computer-Controlled Manipulators", Proc. 5th International Joint Conference on Artificial Intelligence, vol. 2, Boston Mass., 1977.

[Uicker 1965]

J. J. Uicker Jr., "On the Dynamic Analysis of Spatial Linkages Using 4x4 Matrices" (Ph. D. Thesis), Northwestern University, Evanston, Illinois, 1965.

[Uicker 1967]

J. J. Uicker Jr., "Dynamic Force Analysis of Spatial Linkages", Trans. ASME Jnl. Applied Mechanics, vol. 34, pp. 418-424, 1967.

[Vukobratović 1978]

M. Vukobratović, "Dynamics of Active Articulated Mechanisms and Synthesis of Artificial Motion" (in 3 parts), Mechanism and Machine Theory, vol. 13, pp. 1-56, 1978.

[Vukobratović and Potkonjak 1979]

M. Vukobratović, V. Potkonjak, "Contribution of the Forming of Computer Models for Automatic Modelling of Spatial Mechanisms Motions" (in 2 parts), Mechanism and Machine Theory, vol. 14, pp. 179-188, 1979.

[Walker and Orin 1982]

M. W. Walker, D. E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms", Trans. ASME Jnl. Dynamic Systems Measurement and Control, vol. 104, pp. 205-211, 1982.

[Waters 1979]

R. C. Waters, "Mechanical Arm Control", MIT AI Lab. AIM 549, 1979.

[Woo and Freudenstein 1970]

L. S. Woo, F. Freudenstein, "Application of Line Geometry to Theoretical Kinematics and the Kinematic Analysis of Mechanical Systems", Journal of Mechanisms, vol. 5, pp. 417-460, 1970.

[Woo and Freudenstein 1971]

L. S. Woo, F. Freudenstein, "Dynamic Analysis of Mechanisms using Screw Coordinates", Trans. ASME Jnl. Engineering for Industry, vol. 93, pp. 273-276, 1971.

[Yang and Freudenstein 1964]

A. T. Yang, F. Freudenstein, "Application of Dual Number Quaternion Algebra to the Analysis of Spatial Mechanisms", Trans. ASME Jnl. Applied Mechanics, vol. 31, pp. 300-308, 1964.

[Yang 1971]

A. T. Yang, "Inertia Force Analysis of Spatial Mechanisms", Trans. ASME Jnl. Engineering for Industry, vol. 93, pp. 27-33, 1971.

[Yuan and Freudenstein 1971]

M. S. C. Yuan, F. Freudenstein, "Kinematic Analysis of Spatial Mechanisms by Means of Screw Coordinates. Part 1 - Screw Coordinates", Trans. ASME Jnl. Engineering for Industry, vol. 93, pp. 61-66, 1971.

[Yuan et al. 1971]

M. S. C. Yuan, F. Freudenstein, L. S. Woo, "Kinematic Analysis of Spatial Mechanisms by Means of Screw Coordinates. Part 2 - Analysis of Spatial Mechanisms", Trans. ASME Jnl. Engineering for Industry, vol. 93, pp. 67-73, 1971.