DYNAMIC ANALYSIS OF ANTHROPOMORPHIC MANIPULATORS IN COMPUTER ANIMATION

A thesis submitted to the
Council for National Academic Awards
in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

by

STEPHANIA M. LOIZIDOU

UNIVERSITY OF NORTH LONDON

JULY 1992

I would like to dedicate this work to my parents
for their encouragement and support throughout
my studies.

DECLARATION

Whilst registered as a candidate for this degree the author has not been registered for any other award.

S. Loizidou.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my special thanks to my supervisor Dr. Gordon Clapworthy, for his invaluable help and encouragement during the course of this study.

Many thanks also go to Dr. Nick Alexandrou and Ms Elli Georgiadou for their detailed comments on the draft version.

I am also grateful to Prof. John Charalambous for his support.

Special thanks are extended to Marios Michaelides for producing most of the diagrams.

Stephania Loizidou

Dynamic Analysis of Anthropomorphic Manipulators in Computer Animation

ABSTRACT

The thesis examines the motion control and motion coordination, of articulated bodies, in particular human models, through the use of dynamic analysis.

The work concentrates particularly on the use of dynamic analysis as a tool for creating an animated walking sequence. The aim is to relieve the animator of the tedious specification of unknown parameters of motion control, and instead, to investigate the possibility of shifting this specification and much of the work to the animation software. Using this idea, a higher level of control could be used to coordinate the movement.

This has been examined through the use of a Hybrid Inverse and Direct Dynamics System, HIDDS, based on Featherstone's formulation. The inverse system is used to get an initial estimate of the values of the forces and torques needed to move the model to the required position. Once the values of these forces/torques are determined, a direct dynamics system is called to compute the accelerations produced from the influence of the input forces/torques. Double integration over time then gives the new positions. Applications of the hybrid technique can create animation sequences through the use of the graphics editor AnthroPI (Anthropomorphic Programming Interface), especially created for the implementation of the desired results.

The problem of ground reaction forces has also been studied and formulated. An algorithm proposed by Kearney for a one-legged hopping machine was extended to the two-legged anthropomorphic model by the introduction of the virtual-leg concept.

The dynamics approach showed promise in creating motion sequences which are both natural and realistic and HIDDS proved to be a useful experimental tool on which further research can be based. At its early stage, it provides a middle-level of control, which we believe has the potential to be upgraded to a higher level.

CONTENTS

Title *i*

Dedication *ii*

Declaration *iii*

Acknowledgements *iv*

Abstract *v*

CHAPTER 1 6

PRELUDE

1.1 Introduction 7

1.2 An Outline of the Thesis 9

CHAPTER 2 12

COMPUTER ANIMATION : A SURVEY

2.1 Introduction 13

2.2 The Animation Sequence 15

2.2.1 Geometric Modelling - *(The Creation of Human Shapes)* 16

2.2.2 Dynamic Modelling - *(The Motion of the Human Skeleton)* 17

2.2.3 Kinematic Models 18

2.2.4 Dynamic Models 20

2.3 Classification of Figure Animation Systems 23

2.4 Conclusions 25

CHAPTER 3 27

EXISTING DYNAMIC FORMULATIONS

3.1 Introduction - Newton-Euler and Lagrangian Formulations 28

3.2 The Newton-Euler Method 29

3.3 The Lagrangian Method 32

3.4 Comparison of the available Dynamic Formulations 33

3.5 Formulations based on Alternative Methods 35

3.6 Conclusions 38

CHAPTER 4 39

KINEMATIC ASPECTS OF SPATIAL ALGEBRA

4.1 Introduction 40

4.2 Relationship between Spatial Vectors, Line Vectors, and Free Vectors 41

4.2.1 Introduction 41

4.2.2 Line and Free Vectors 41

4.2.3 Line and Free Vector Components of Spatial Vectors 43

4.3 Operations on Spatial Vectors 44

4.3.1 Introduction 44

4.3.2 The Representation of Spatial Rigid-Body Coordinate Transformations 45

4.3.3 The Spatial Scalar Product 48

4.3.4 The Spatial Transpose Operator 49

4.3.5 The Spatial Cross Operator 51

4.4 Representation using Spatial Quantities 52

4.4.1 Introduction 52

4.4.2 Spatial Velocity 52

4.4.3 Spatial Acceleration 54

4.4.4 Representation of Joint Axes 54

4.4.5 Representation of Forces 58

4.5 Differentiation in Moving Coordinates 59

4.5.1 Introduction 59

4.5.2 The Derivative of a Vector in a Moving Coordinate System 59

4.6 Concluding Remarks 62

Appendix 4A - Addition of Spatial Vectors and Multiplication of a Spatial Vector by a Scalar 65

Appendix 4B - Apparent Derivative of the Joint Axis 67

Appendix 4C - Illustration of the form of the Spatial Transpose Operator 68

Appendix 4D - Differentiation of an Orthogonal Matrix 70

Appendix 4E - Properties of the Cross Operator, $\times$ 73

CHAPTER 5 79

DYNAMIC ASPECTS OF SPATIAL ALGEBRA

5.1 Introduction 80

5.2 Spatial Rigid-Body Momentum and Spatial Rigid-Body Inertia 81
5.2.1 Introduction 81
5.2.2 Definition of Spatial Momentum 81
5.2.3 Definition of Spatial Inertia 82
5.2.4 Properties of the Spatial (Rigid-Body) Inertias 84

5.3 The Equations of Motion 86

5.4 Articulated-Body Inertias 87
5.4.1 Properties of Articulated-Body Inertias 88

5.5 The Calculation of Robot Dynamics using Articulated-Body Inertias 89
5.5.1 Introduction 89
5.5.2 Degrees of Freedom of Motion and Branched Chains 89
5.5.3 Spatial Rigid-Body Inertia Matrices 91
5.5.4 Initial development of the Direct Dynamics Algorithm 93
5.5.5 Transformation Matrices 98
5.5.6 The Articulated-Body Dynamics Algorithm 100

5.6 Conclusions 101

Appendix 5A - Calculation of Inertia Tensors 103
Appendix 5B - Joint Axis Representation 104
Appendix 5C - Comparison of the Computational Requirements of Dynamics Formulations 105

CHAPTER 6 107
ADAPTATIONS OF THE ARTICULATED-BODY METHOD TO COPE WITH AN ARBITRARY ANTHROPOMORPHIC MANIPULATOR
6.1 Introduction 108

6.2 Extensions of the Dynamics Algorithm 108
6.2.1 Modifications to Allow Multiple-Degree-of-Freedom Joints 109
6.2.2 Modifications to Allow Branched Kinematic Chains 111
6.2.3 A Moving Base rather than a Stationary One 113

6.3 The Modified Algorithm 117
6.3.1 New Description of the Model 119

6.4 Conclusions 122

CHAPTER 7 123
GROUND REACTION FORCES
7.1 Introduction 124
7.2 Ground Reaction Forces Formulation 124
7.2.1 Introduction 124
7.2.2 Contact Kinematics 127
7.2.3 The Analysis - A Single Point of Contact 128
7.2.4 Re-formulation of the Algorithm 132
7.3 Conclusions 133

CHAPTER 8 134
AnthroPI - ANTHROPOMORPHIC PROGRAMMING INTERFACE
8.1 Introduction 135
8.2 AnthroPI - An Outline of the System 136
8.2.1 Calculations of Inertia Tensors 139
8.2.2 Order of Rotations 139
8.2.3 Joint Limits 140
8.3 User Interface 145
8.4 Conclusions 147

CHAPTER 9 148
THE GAIT CONTROLLER
9.1 Overview 149
9.2 Levels of Control 150
9.3 A Generic Locomotion Pattern 153
9.3.1 Introduction 153
9.3.2 Human Locomotion 153
9.4 Higher Levels of Control in Human Walking 154
9.4.1 Introduction 154
9.4.2 Control Principles 156
9.4.3 The Determinants of Gait 159
9.4.4 Time Durations and Locomotion Parameters 162
9.4.5 Virtual-Leg Concept 163

9.5 Implementation Details : A Hybrid Dynamics Algorithm 168

9.6 Refining the Movement 173

9.7 Conclusions 174

CHAPTER 10 178

POSTSCRIPT

10.1 Conclusions 179

10.2 Future Directions 182

REFERENCES 186

CHAPTER 1
PRELUDE

1.1 INTRODUCTION

'The eye draws us into different parts of reality, it crowns mathematics, its sciences are the most reliable, it has measured the distance and size of the stars, it has created architecture, perspective and divine painting'

Leonardo da Vinci

Animation is the process of creating images that appear to move. Long before movie cameras were invented, Eadweard Muybridge lined up a series of still cameras to photograph a horse as it ran down a racetrack (to prove that when a horse is galloping, there are times when it has all four feet off the ground simultaneously). Since our culture is predominantly visual, the movement of animals and humans, and more generally the movement of rigid objects, has always been an interesting area of research.

Computer animation is the process of creating visual movement through the use of a computer. Today, computer animation is used in many different and diverse areas of our everyday life and its number of applications is enormous.

Perhaps the fastest-growing use of computer animation is in the *film industry*. The computer approach promises to improve realism and lower production costs. One very desirable but not fully realised approach is to use computer-generated animation to replace the hand-built models. Sophisticated mechanisms are available to model the objects and to represent light and shadows. The main drawback though, is that a long time is required to enter all the coordinate information for the model.

Advertising and *arcade games* for entertainment, is where large amounts of money are being spent in computer animation today. This is probably because special effects of computer animation are so novel and sometimes the computer influence is not readily detectable. In addition to the current preoccupation with the use of virtual reality techniques in computer games and computer art, there are a number of international aviation and robotics initiatives which are driving important technological developments.

The use of animation in the *medical sciences* is becoming important in helping

doctors and researchers to visualise the composition of a particular organ or bone structure. Doctors can "fly" around inside our bodies, having first scanned them with whole body scanners to obtain cross sections.

Computer animation has a promising future in *educational fields* by improving teaching programs on personal computers. This is because human beings usually learn faster and better when presented with an image of the items about which they are learning. Essentially, animation allows designers and engineers to visualise complex processes and to make better decisions regarding them. *Civil engineers* and *architects* are largely aided by the use of computer animation programs as they can use these to display the results of mathematical simulations of both the interior and exterior of building designs, bridges and so on. Additionally, one growing application of computer animation is *biological simulations*, i.e. the simulation of how molecules are formed.

Computer animation is the most broad, complex and technically sophisticated area of the computer graphics world. Computer animation makes it possible to display and visualise a 3D scene and it allows fly-throughs which give a 3D-realistic feeling. In addition, the dynamic nature of computer-animated images introduces four-dimensional displays; with the fourth dimension being time. Animation systems are capable of displaying dynamic images that show the behaviour of the subject over a period of time, or under the effect of certain changes in conditions. Furthermore, the unique application of computer animation to visualise entities that cannot be seen either with the naked eye or with scientific aids provides a valuable tool in perceiving images that otherwise can only be imagined.

Among the vast number of recently developed applications in computer animation, the animation of human figures is by no means the least important, as it allows one to perform and visualise the complex movement of the human body and adds to the large list of applications of the diverse areas mentioned above. Nowadays, the ability to build, display and manipulate models of human figures, as well as generate realistic movement of these models, gives animation techniques the power to play an important role in computer graphics.

1.2 AN OUTLINE OF THE THESIS

This thesis examines the motion control and motion coordination of anthropomorphic articulated figures using dynamic analysis. The aim is to permit the animator to control the motion at a high level, without the need for a detailed specification of the motion, a task which would be handled by the motion-control system.

This is an interesting problem closely related to man's everyday life. It is a complicated issue, since the articulations contained in the human figure have enormous complexity. Besides, there exists a large number of variations in the movement of individuals, so that exact specification and control of their movements is, in most cases, practically impossible. The fact that human movement patterns are extremely familiar, and the viewer can easily recognise when a motion appears artificial and unnatural, complicates the matter further.

The investigation has resulted in the development of a movement-simulation system, in which much of the burden of generating explicit motion descriptions for articulated figures is shifted to the animation software, but which, nevertheless, provides the animator with the means to adjust, or to induce subtle variations into, the motion. It provides an experimental tool on which further investigations can be conducted for the creation of an automatic control technique which, with further development, will offer great potential for animating complex motion, with minimal user input, and will produce accurate and realistic results.

The coordination and control of the motion can be achieved through the use of a Hybrid Inverse and Direct Dynamics System (HIDDS) : inverse dynamics involves the calculation of the forces/torques required at the joints of a model, in order to produce a given set of relevant accelerations. Once enough information is given for the system to be able to calculate the motion of every link in the mechanism, it is then a straight-forward matter to calculate the forces involved. Upon calculation of the forces, direct dynamics is applied to determine the motion of the model by calculating the joint accelerations and the joint positions of the next phase. In addition, kinematic control techniques can be used as an alternative approach to dynamics to aid the development of e.g. an animated walking sequence.

Chapter 2 describes a number of methods for animating general rigid bodies and, in particular methods for animating articulated figures such as models of humans and animals. Emphasis is placed on the development of methods using dynamic analysis and the discussion shows why and how, dynamic analysis offers an attractive alternative to other established methods.

In Chapter 3, an outline comparison is made of the different formulations which are most frequently used in dynamic analysis, i.e. the Newton-Euler and the Lagrangian algorithms. Consideration is given to the economy and efficiency of the various algorithms that have been proposed. The large number of different methods and diverse approaches in existence show that solving the dynamic equations of a complex, coordinated human motion, is still quite difficult without a unifying, outstandingly-superior algorithm.

Chapters 4 and 5 of the thesis describe the use of spatial algebra in dynamic analysis. The kinematic and dynamic aspects of spatial algebra are described, with the objective of introducing the necessary background material for the development of an animated sequence using Featherstone's direct dynamics formulation.

Chapter 4 serves as an introduction to the use of spatial notation in the analysis of spatial rigid-body dynamics. Firstly, it defines the spatial vector notation, then represents well-known physical vector quantities using this spatial notation, and finally gives the basic background of the kinematic aspects of spatial algebra.

Once this spatial notation has been properly defined, together with its representations and operations, Chapter 5 investigates all the dynamic aspects of spatial algebra; it considers the application of spatial theory to more advanced topics, such as the representation of inertia, the equations of motion and spatial vector analysis.

In Chapter 5 all the quantities are represented using spatial notation and a proper understanding of Chapter 5 and subsequent chapters, can only be obtained by reading Chapter 4 first. Chapters 4 and 5 will provide the necessary background theory of Featherstone's formulation.

Chapter 6 contains the necessary adaptations and extensions, to enable the formulation to accommodate articulated models of any structure.

Chapter 7 tackles the problem of ground reaction forces, and Chapter 8 is concerned with the description of the interface environment, AnthroPI (Anthropomorphic Programming Interface). Chapter 9 then considers the control and coordination of motion at the highest level. Once the parameters of a locomotion pattern and the gait determinants are described, the effort is concentrated into motion specification which is another major problem with regard to motion control of articulated bodies. This is a necessary requirement for the development of a purely dynamically-animated walking sequence.

The innovative idea of introducing a hybrid algorithm to achieve a fully dynamically animated walking sequence has proven that it is feasible to specify motion easily and conveniently and hence, produce accurate, realistic and natural effects. Although such a hybrid system is still in its infancy, it proved to be promising as most of the problems associated with its description have been identified and tackled.

CHAPTER 2
COMPUTER ANIMATION : A SURVEY

2.1 INTRODUCTION

A variety of methods for animating general rigid bodies and in particular, articulated figures such as humans and animals, have been developed but each has a number of drawbacks. Recently *dynamic analysis* of motion has been introduced as a means of animating articulated bodies.

When considering the physical properties and the physical principles governing moving objects, it can be seen that the motion of such objects is determined by their own nature and their interaction with the environment; this holds true, in particular for models of human animation. Therefore, describing such models dynamically shows great promise for creating realistic animations.

The development sequence of methods for animating articulated objects, which has led to the dynamic approach, will first be considered.

One of the earliest approaches was to *record* or *digitise* the motion of a human. A number of problems are associated with this technique. The most serious one is the need of a properly instrumented human actor to perform all the actions required for animation. Another problem with the approach is that the human actor cannot perform dangerous actions or simulate the motion of non-human characters. A resulting approach, related to the above, was to use human body positions as *keyframes* in an animation sequence, but this method still requires the collection of human movement data and has all the problems of key framing. For example, key-frame animation for a complex object such as an articulated figure is realistic only if a large number of keyframes are used so that the intermediate motion is well-defined. An extension of the above methods is *rotoscoping* where the joint coordinates of the model are digitised in a frame by frame fashion. A complete description of the method is given in Section 2.2.2.

A more recent approach was based on developing a *kinematic model* of the human body. This model is based on the simplification of the skeleton of the human figure, that is, representing the topology and physical properties of the figure, as well as describing the characteristics of its motion, which is achieved by a hierarchy of motor programs.

Incorporating *dynamics* into the model simplifies interaction with, and control of, the model. Once the model has been constructed, a wide range of movements can be achieved during the animation, by applying forces or torques to joints of the model. Complete dynamics-based models have the promise of producing realistic motion, though they are computationally expensive, especially for long sequences.

For example, dynamic analysis is performed when an engineer needs to understand how the individual parts within a complex assembly move, and what forces they endure when the assembly is acted upon by an external force. To work with such a tool, an engineer who, for example, is designing an aeroplane landing gear, might sketch out a model of the assembly; he would input the forces, such as the pressure generated by the impact of landing, the speed of the plane and other relevant information. The system would then calculate the reaction forces, motion, velocities and accelerations for the individual parts and for the landing gear assembly as a whole.

It follows that commonly-used approaches are fairly low level and require a considerable amount of user input to design the motion. As the motion becomes more complex, with many objects interacting in a variety of ways, the provision of more automatic and high-level approaches to motion control becomes increasingly important and necessary. Accordingly, a number of objectives should, if possible, be fulfilled. For example, the automatic production of computer-generated human models and their natural behaviour are necessary requirements for realistic human figure animation. Additionally, the improvement of the realism of motion, the reduction of its complexity, and the simplification of the method of motion description are essential features.

Methods which include a high proportion of features which are generated automatically will allow the production of sophisticated animation with less user effort : e.g. *dynamic analysis*, *path planning* and *collision avoidance*, *stimulus response*, *control* and *learning algorithms*.

Consequently, there are three main factors that lead to the introduction of dynamics in animation control :

1) dynamics frees the animator from the description of the motion by using the physical properties of the objects,

2) the produced motion is physically realistic, and

3) bodies can react automatically to internal and external environmental influences such as collisions, forces and torques.

All these advantages of a dynamic approach show promise in systems based on physical characteristics, which form an integral part in the process of animating a human model.

2.2 THE ANIMATION SEQUENCE

Tost & Pueyo [1] and Wyvill [4] have produced surveys about the development of an animation sequence. This sequence consists of three distinct but inter-related stages, namely, Geometric Modelling, Dynamic Modelling and Rendering.

Geometric Modelling is simply the creation of data to define the geometric skin or primitive of the object. *Dynamic Modelling* involves the movements of the objects in the scene to be animated as well as the movement of the 'virtual camera'. *Rendering* is the production of the final image and involves lighting, shading, hidden surface removal, and addition of texture.

Rendering does not impose any extra problems when used for human figure animation, since it is independent of the modelling and animation of the object, whereas, geometric modelling and dynamic modelling present particular difficulties. In the next sections, descriptions of geometric and dynamic modelling will be given, together with their applications to human models.

Before we embark on these descriptions, some terminology needs to be introduced.

The structure of the human model consists of joints connected by links. The connectivity can be described using a tree structure representation which clearly defines the hierarchy of joints, the nodes of the tree, together with the branches coming out from a specific node, the links. The tree consists of a number of kinematic chains, and in each single chain, the first link is called the *root* of the chain and the last link is called the *end-effector*.

The terms *distal* and *proximal* segments, which define the relative position of one link with respect to another, have to be defined. The proximal link

defines a neighbour of the current link and is the one which precedes a referred joint, whereas the distal link is the one which succeeds it. Therefore, the proximal link will be the one nearer to the root if the order of traversing the tree is from the root down to the end-effectors.

The *world space position* is the position of a coordinate system relative to a fixed reference system and *a degree of freedom* determines the direction along which a joint can move. *Translational* degrees of freedom are the degrees of freedom associated with linear motion, while *rotational* degrees of freedom are those associated with revolute motion.

2.2.1 Geometric Modelling

THE CREATION OF THE HUMAN SHAPES

In the first stage, the objects, and in particular the human models, to be viewed are built. From a geometrical point of view, the problem of constructing the objects is mainly one of entering their shapes. That is, generating the body and storing it using a geometric model structured in some way in order to define relationships between the different elements. A variety of techniques have been used to describe the human body, Badler & Smoliar [56].

Stick Models consist of a hierarchical set of rigid objects (limbs) connected at joints, forming an articulated body. The main advantage of the stick figure is that the specification of movement is very easy. It is represented using lines. The major drawback of these systems is that they produce rather unrealistic visualisations as the lack of volume makes the perception of the depth, twists and contacts difficult to represent.

Surface Models can be used to construct a surface (model the external shape of the body - the skin) made up of planar or curved patches. For this, the tedious time-consuming task of entering the significant points or vertices that configure the surface is needed. The advantage of curved patches is that they model the skin in a smoother, more realistic manner and considerably fewer are required than planar patches. The disadvantages are that computations concerning their precise position, intersections and other activities in the rendering process, such as hidden surface removal, are greatly increased.

Volume Models approximate the structure and the shape of the body with a collection of elemental volume primitives such as ellipsoids, spheres or cylinders. These models solve the problem of ambiguity of appearance associated with stick figure models. On the other hand, when realism is required, they cannot compete with surface models since, although they require only a small number of primitives, these cannot be combined naturally and realistically. As with surface models, volume modelling requires hidden surface removal which tends to be extremely time consuming.

2.2.2 Dynamic Modelling

THE MOTION OF THE HUMAN SKELETON

This second stage is involved with the specification and computation of the movements. That is, moving the objects in the scene to be animated. Models of movement can be classified using one of the following techniques.

1. Rotoscoping,
2. Kinematic Models, and
3. Dynamic Models.

Rotoscoping is the process whereby the live action of characters and/or objects is recorded on film and then projected on to animation cells, (i.e. cartoons of the animation industry), in a frame by frame fashion. In each recorded frame the positions taken are digitised. One such example is taking footage of live actors and using it as a guide for animated sequences to generate the motion of the figure, McGovern [31].

Rotoscoping is not very flexible. Once the animation sequence is created, one cannot change it to create a different one. Additionally, rotoscoping is not suitable for the creation of imaginary sequences, and a lot of effort is needed to collect the film data. On the other hand, its main positive characteristic is that it does produce realistic animation. Using rotoscoping in 2D is an easy process which can be interpreted correctly, but in 3D it is a fairly long and complicated procedure which requires the collection of 3D information.

Detailed consideration will be given to the other two techniques, namely, kinematic models and dynamic models.

Kinematics studies how individual parts of an object move relative to one another, without considering what has caused the movement. It therefore produces motion from positions, speeds and accelerations, Wilhelms [2]. *Dynamics* relates the forces and torques acting on masses to their accelerations. Applications of dynamics are based on well-known techniques from physics and robotics and, therefore, dynamically-controlled motion produces very realistic results, freeing the animator from the description of the motion and allowing for bodies that can react automatically to internal and external environmental constraints, Armstrong et al [3].

It can be concluded that dynamic animation has certain benefits lacking in kinematic systems. Motion is automatically constrained to respond to environmental conditions. For example, it is difficult to animate kinematically a body, such as a human figure, to respond naturally to collisions. Using dynamics, such collisions can be automatically simulated by applying an opposing force against the body it collides with.

However, the use of dynamics implies that the physical properties of the body, such as the mass, moments of inertia and principal axes, have to be defined, a requirement that is not necessary when kinematic models are employed.

Of course, both methods (kinematics and dynamics) have their difficulties because of our unwillingness to accept mildly unrealistic motion for bodies as familiar as humans and animals. Therefore, there are systems that combine more than one of the above three methods, especially rotoscoping with kinematic models in order to improve the depiction of the motion, although these systems do not necessarily offer the best solution.

2.2.3 Kinematic Models

Most animation systems are kinematically based, Wilhelms [2]. Such systems rely on user input, facilitated by an interactive system, of which *keyframe interpolation* is the most common.

Keyframe interpolation provides the automatic generation of intermediate frames, called *inbetweens*, by interpolating a set of keyframes supplied by the animator. To interpolate the inbetween positions, keyframing uses

splines, usually cubic splines as these use a small number of coefficients, to provide second order derivative continuity and therefore guarantee smooth animation. A way of producing improved motion is to interpolate parameters describing the position of the object rather than the positional values of points on the object. This popular technique, is called *parametric keyframe interpolation.* In a parametric model, the animator creates keyframes by specifying the appropriate set of parameter values. Parameters are then interpolated, and images are individually constructed using the interpolated parameters.

Kinematics treats the phenomenon of motion without regard to what caused it. It calculates positions without considering forces. In kinematics, there is no reference to mass; the concern is only with relative positions and their changes. It is strictly a motion analysis tool used to study how individual parts move relative to one another. Therefore, kinematic models produce motion from positions, speeds and accelerations, which need to be specified by the animator.

Motion is achieved by a hierarchy of motor programs. Problems associated with this are that a separate set of motor programs is required for each type of motion, and that the model can only react to the environment in a restricted way. It is difficult to specify realistic motion kinematically, particularly in cases where the body is moving in complex patterns, in a complex environment, or with great freedom, unless complex laws are found.

When considering hierarchical structures such as the human model, there exist two types of kinematic control, namely, direct and inverse kinematics.

Direct Kinematics finds the world space position of a distal segment of the body given the joint positions of the segments proximal to it, i.e. it computes the position and orientation of the end-effector of the manipulator relative to the base frame.

Inverse Kinematics is the simplest automatic control of motion. In a typical animation system based on inverse kinematics, the animator specifies the discrete joint positions and motions of the outermost links; then the system computes backwards the necessary joint angles and orientations for the other parts of the body to put the specified parts in the desired positions and through the desired motions. Inverse kinematics provides a means of

constraining articulated bodies with reference to the world, e.g. keeping a body realistically in contact with the ground when it moves.

The inverse kinematics problem is not as simple as that of direct kinematics and, because the kinematic equations of the inverse problem are nonlinear, their solution is not always easy or even possible.

In recent years interesting applications of kinematics have been made in areas as diverse as animal locomotion, art, biomechanics, geology, robots and manipulators, space mechanics, structural chemistry and surgery. The large list of applications is due to the fact that everything that moves has kinematical aspects. Intelligent control of motion can be greatly aided by combining knowledge from related fields such as robotics, artificial intelligence, psychology, biology and physics.

2.2.4 Dynamic Models

Motion in the real world is produced by the action of *forces* and *torques* on objects which have mass. Equations of motion are used to relate the forces and torques acting on the masses to the accelerations produced under the influence of these forces.

Force is a physical quantity. It is a vector and has a meaning independent of any coordinate system. To represent it analytically, it is usual to introduce a system of coordinates using a frame of three mutually-orthogonal unit vectors. Coordinate frames may be of various types such as Cartesian, spherical, or cylindrical.

One can also use coordinates not restricted to a particular type of frame. A coordinate whose particular nature is not specified is known as a *generalised coordinate*. Generalised coordinates are chosen in such a way that specification of the coordinate values completely defines the system configuration.

Typically motion is described in terms of *degrees of freedom*, i.e. the number of independent coordinates needed to specify the positions of all components of the system.

The generalised forces, that is the net forces or torques active at each degree of freedom, are the forces represented in a generalised coordinate system and they are expressed in the dynamics equations as functions of the mass distribution, acceleration, and velocity of all distal segments.

Dynamics equations increase in complexity when the masses involved are extended bodies, not points, and when multiple masses interact.

For example, a *particle* can be described as a point in three-dimensional space (x, y, z). The location and motion descriptors of such a point are each designated by three variables, and thus the point has three translational degrees of freedom of motion.

Rigid bodies, on the other hand, are defined by an infinite number of points which cannot move relative to each other, though they may move as a whole relative to the world space. These bodies are assumed to have mass, centre of mass, and mass distribution.

The motion of a rigid body is specified by six degrees of freedom, three translational and three rotational. Motion of a rigid body is usually visualised as motion of a frame that is fixed to the body, with all points defining the body moving with that frame. An arbitrary rotation of these bodies can be formed as a combination of at most three rotations about the coordinate axes, x, y, and z.

Articulated bodies are made up of segments, which are individual rigid bodies whose relative motions are somewhat restricted. The dynamics of these bodies is more complex due to this segmentational interaction.

The number of dynamics equations necessary to specify a system depends upon the number of degrees of freedom in the system. For an articulated body, the number of degrees of freedom is the sum of the degrees of freedom at each joint (at most three rotational degrees of freedom, one for each rotation about the three coordinate axes), plus the number of degrees of freedom connecting the body to the world (up to three translational degrees of freedom, one for each direction in space).

Simulating articulated body motion with dynamics is difficult because of the many degrees of freedom and the complex coordinated motion possible, so that

acceptable results are not produced easily. Because of the interaction taking place between masses in an articulated body, the equations are complex and coupled. For these bodies, there exists the need for a common mathematical basis for advanced modelling, and for a compact representation and control method.

Armstrong & Green [5] suggest that the model must have the following characteristics. It should produce realistic motion sequences, the amount of information provided by the animator must be minimal and proportional to the complexity of the motion, and it should be able to react to and act upon its environment.

Forces and torques responsible for motion come from a variety of *external* and *internal sources*, for example, gravitational forces or known externally-applied mechanical forces. These forces, some of which have been modelled by different researchers using springs and dampers, include those due to joint limits, contact with the ground, and collisions with other objects.

It is difficult to control directly torques and forces acting on the body. Furthermore, controlling the forces and torques of more than two or three limbs at any one time is, in some cases, practically impossible. Therefore, while the user is manipulating a small subset of limbs, the animation system must be able to control most of them automatically.

Coupled with computer animation, dynamics has a large list of applications, for example in the design and control of robots and mechanical manipulators, in the advertising and entertainment industry, and in the biomechanical exploration of the engineering principles underlying animal motion and motion in sports. Dynamic analysis is used in CAD/CAM and robotic applications to simulate and control vehicles or mechanical manipulators.

Dynamic animation has three main drawbacks. Firstly, the *cost* of the analysis is high, secondly, numerical *instability* can be a problem when the bodies are complex and finally, *control* of the motion is difficult. Accordingly, dynamic analysis is computationally much more expensive than kinematic animation; a lot of CPU time is required to solve the equations of motion for a complex articulated body using numerical methods, and when the system dynamics are complex, as they are in articulated bodies, numerical instability can be a problem.

Dynamics also requires the initial determination of object and environmental characteristics. This means that the design of a specific user interface is essential, since the animator does not think in terms of forces or torques to apply to a link or the body as a whole, in order to generate a motion. Perhaps the most serious disadvantage occurs in simulating controlled motion.

Dynamic model formulations can be solved using direct dynamics and/or inverse dynamics.

The *Direct Dynamics* problem involves the determination of the motion of a system from a set of applied forces. That is, if the generalised forces are known, the equations can be solved for accelerations. It is the most frequent direction of the formulation used in animation and is an essential part of the process of simulating a human figure motion.

On the other hand, *Inverse Dynamics* is the problem of determining the forces desired to produce a prescribed motion in a system. That is, if the accelerations are known, the equations can be solved for the generalised forces. This is the direction commonly used for dynamic control in robotics.

2.3 CLASSIFICATION OF FIGURE ANIMATION SYSTEMS

Using terminology that has been covered already and introducing some new concepts, an attempt will be made to classify human body animation systems by outlining the most important classes. General classifications have been given by Tost & Pueyo [1].

We can distinguish animation systems *according to the type of model* used to structure the human body. As has been mentioned already, the human model can be represented by **stick figures** that consist mostly of lines. Alternatively, it can be represented by **surface models** which are planar or curved patches, or by **volume models** which are made out of elemental volume primitives such as ellipsoids, spheres or cylinders.

According to the motion model, systems can be classified as being **dynamic**, where the motion is calculated using forces/torques and accelerations, or, as being **kinematic**, where we talk about positions and speeds.

An alternative classification will be *according to their application.*

Examples include those that are produced for **commercial or entertainment films** and used for didactic and publicity purposes, or for advertising. The objective is to create a character or 'reconstruct' a real person so that the visual impact can be taken into consideration. There are also applications where a simplified model is satisfactory, and these applications are used in engineering, medicine, or choreography for **industrial or scientific simulations** in which human beings are involved.

Zeltzer [6] gives a distinction *according to the movement specification.* **Guiding systems,** keyframe and interpolation, define explicitly a set of frames giving the parameters that define them, and intermediate frames are calculated by interpolation. These systems require that the details of motion must be specified in advance. The animator can control the motion at any level of detail and needs to have no computer science knowledge. On the other hand, a large amount of data is needed for complex movement since the motion needs to be described explicitly. Guiding is thus unsuited for controlling complex motion or complicated figures. The second class is **program level,** i.e. languages, where motion is specified algorithmically in some programming notation. This needs computer scientist users as the user faces all the problems associated with software development. Although programming is difficult, it is powerful in the sense that it provides control of the motion. **Task level,** i.e. motor program handling, is the third class and is described further in Zeltzer [73]. For this, high level commands are used to perform predefined or computable movements. The animation system must schedule the execution of motor programs to control characters. The behaviours need to be described implicitly, in terms of events and relationships. Task level programs provide easy control over complex movement by trading off explicit control over the details of the movement. The major drawback of these systems is their accuracy, because the user has no control of the motion of individual elements.

Finally, systems can be classified as *interactive or scripted.* The **interactive** method implies that the user and motion control system participate in a loop. The user describes a motion, the computer quickly provides an animation using it, the user modifies the motion as necessary, and the interaction continues. Thus the user can design motion in real time

while watching the animation develop on the graphics screen. One example is *keyframing* where the user specifies a sequence of positions and the times when they occur, and the computer interpolates between these positions to produce the animation. In particular, 3D object-based keyframing is a convenient and successful motion-control method. The user can see the total configuration of the system at given times, easily noting collisions or undesirable interactions. Disadvantages are that it is low level, requires the user specifically to control each degree of freedom, and does not allow easy visualisation of the motion between keyframes. In **scripted systems**, the user creates a written script describing the motion that the control system later interprets to produce the animation. Since animation is algorithmic in nature, it is difficult for it to be specified interactively, therefore sequences using scripting systems are easier to produce than those using interactive systems, but on the other hand, movements within them cannot be specified by direct manipulation, and an explicit description of each frame does not exist.

2.4 CONCLUSIONS

Various computer animation techniques for both the creation and motion control of the human figure have been presented. It was shown that no overall general method exists, as the use of a specific technique is dependent upon the application, the amount of realism required, the cost of the method and the amount of computational time.

Methods that tackle the problem of motion control originate from very different disciplines, such as biomechanics, robotics, AI, cybernetics and physics. These techniques are discussed in more detail in Chapter 3.

Subsequently, the thesis is concerned with the animation of articulated human models and in particular, with the motion control of such objects. This task, because of the large number of articulations and links that the human body contains, as well as the realism required, is a difficult one. Research in the field concludes that the incorporation of dynamics into the model simplifies interaction with, and control of, the model. Once the model has been constructed, a wide range of motions can be achieved by applying forces or torques to joints in the model, during the animation.

Dynamically-controlled motion sequences of worlds created by computer animation can be produced without intensive animator action. Although a number of difficulties exist, early results have shown that these methods have the potential, with further development, to create highly-realistic motion.

CHAPTER 3
EXISTING DYNAMIC FORMULATIONS

3.1 INTRODUCTION - NEWTON-EULER AND LAGRANGIAN FORMULATIONS

In the literature associated with this field of study, most investigators have chosen to use algorithms based on Lagrangian and Newton-Euler methods and, although occasionally individuals have tried to employ other methods, this pattern remains true today. In this chapter we compare the different formulations which have arisen from the applications of Newton-Euler and Lagrangian methods, that is, the most frequently used sets of dynamic equations of motion for typical manipulators. A consideration of formulations based on alternative methods is also included.

With the evolution of techniques, the problem of computing simple manipulator dynamics appears, in principle, to have been solved, and, with recent developments in hardware, a real-time solution now seems to be a possibility. With this, the impetus for applying approximations or tabular techniques, both of which result in computational economies at the expense of large memories, to solve the dynamics appears to have been lost. With both the recursive Lagrangian formulation and the recursive Newtonian formulation of the dynamics, algorithms have been implemented for which the number of additions and multiplications varies *linearly* with the number of joints. Extensions of these two techniques are required so that, if possible, they can be applied to any structure and particularly to dynamically-complex manipulators having more than one degree of freedom.

The Newton-Euler method might be said to be a '*force-balance*' approach to dynamics, whereas the Lagrangian method is an '*energy-based*' approach to dynamics. Of course, for the same manipulator, both will give sets of equations of motion which will have similar computational costs.

A survey and some sort of analysis concerning the economy and the efficiency of the two formulations is outlined in this chapter. An attempt is made to establish whether one of the methods is easier to apply and/or is more efficient than the other (Section 3.4).

One of the earliest ideas, discussed by different researchers in the area, indicated that formulations based on the Newton-Euler method were the least computationally intensive, with real time computation. Later Silver [33] showed that the end result of applying both methods is that the number of

additions and multiplications varies *linearly* with the number of joints and therefore, the Lagrangian method can be made roughly as efficient as the Newton-Euler method.

The economy associated with this linearity is also dependent upon the number of coefficients, so that for example, an algorithm of order $O(n^2)$ could sometimes be faster than an O(n), if the coefficients are small or n itself is small. However, as the human figure has a large number of joints and links, for a realistic approximation, n will have to be moderately large and linear dependency will be desirable.

The two methods, Newton-Euler and Lagrangian (together with their existing formulations), are described separately, so that their advantages and disadvantages can be compared. A third method, proposed by Wilhelms [2], that uses the Gibbs-Appell formulation, is also described, together with formulations by Raibert [15], and Featherstone [16]. An algorithm by Hashimoto & Kimura [22] using parallel processing is also discussed.

3.2 THE NEWTON-EULER METHOD

The Newton-Euler method is based directly on the laws governing the dynamics of rigid bodies. That is,

(1) The vector force acting on a given link is related to the acceleration of its centre of mass by Newton's second law

$$F = m\,\dot{v}$$

where m is the total mass of the body and $\dot{v}$ is the acceleration of the centre of mass.

(2) The total vector moment (torque) about the centre of mass is related to the angular velocity and angular acceleration of the body by Euler's equation

$$N = I.\dot{\omega} + \omega \times (I.\omega)$$

where ω is the angular velocity, and I is the inertia tensor.

The Newton-Euler approach applied successively to the links of a hierarchy yields a set of *recursive equations*. The results, however, have the

disadvantage that the input forces/torques for all joints are referenced to the base coordinates which form the implicit reference coordinate frame, while in practice, they are referred to their own coordinates, i.e. the internal coordinate system of the link. Further, although manipulators have rigid links, the inertia term of each link with respect to the base coordinates varies when the link moves and hence the computation is difficult. Therefore, the end result of referencing to link coordinates avoids a great deal of coordinate transformation and allows the inertia tensor to be fixed in each link coordinate system.

Armstrong [5, 9], Luh et al [7, 12], Walker & Orin [11], Orin et al [32], Hahn [13], Barzel & Barr [14] and Stepanenko & Vukobratovic [17, 18] have used the Newton-Euler method in their formulations.

Stepanenko & Vukobratovic [17] developed a recursive Newton-Euler formulation in the context of human limb dynamics to measure forces. This formulation was revised and extended by Vukobratovic et al in [18].

Orin et al [32] were the first to suggest that the forces and moments in the Newton-Euler formulation be referred to the internal coordinate system of the link, that is, all input joint torques were referenced in their own local coordinates. One coordinate system was attached to each link of the manipulator; therefore, the coordinates moved together with the links.

Armstrong [9] and Luh et al [7, 12] extended this idea by calculating the linear and angular velocities and accelerations in link coordinates as well. Armstrong's dynamics (direct formulation) referred to coordinate systems located at the joints, while Luh's (inverse) dynamics referred to coordinate systems located at the link centres of mass. The formulation developed by Luh et al [7] for mechanical manipulators was based on systematic computation and was independent of the type of manipulator configuration. Barzel & Barr [14] also used body coordinates in their inverse dynamics system to gain benefits similar to those mentioned above.

Walker & Orin [11] presented four different formulations that are based on the inverse Newton-Euler problem : given relative joint positions, rates and accelerations, the relative joint torques can be determined. Three of the algorithms they proposed were of order $O(n^3)$, while the fourth was of order

$O(n^2)$. One might expect that the algorithm of order $O(n^2)$ to be the most efficient. However, they claimed that the most efficient among their formulations is one of order $O(n^3)$, where a recursive procedure is used for computing the mass, centre of mass and the moment of inertia matrix. They showed that the $O(n^2)$ algorithm required $76n^2+120n-21$ multiplications and $56n^2+87n-6$ additions, whereas, the best of their $O(n^3)$ algorithms required $\frac{1}{6}n^3+\frac{13}{2}n^2+\frac{192}{3}n-49$ multiplications and $\frac{1}{6}n^3+8n^2+\frac{5}{6}166n-64$ additions. All these methods used local coordinates.

It is therefore concluded that following Orin's proposal to use local coordinates, all the succeeding algorithms have adopted the same idea.

Essentially the process involves the successive transformation of velocities and accelerations from the base of the manipulator to the end-effector, link by link, using the relationships of moving coordinate systems, as explained by Symon [10]. Forces are then transformed back from the outermost link to the base to obtain the joint torques. Because of the nature of the formulation and the systematic calculation of the generalised forces, computations are simple, which makes it possible to achieve a short computation time.

In conclusion, the formulation presented by Luh et al, produces a relatively accurate numerical solution and requires a short average computing time. Using this formulation, the amount of computation increases linearly with the number of joints.

The efficiency of Newton-Euler formulations is due to two factors : the possible *recursive structure* of the computation and the *representation chosen for rotational dynamics.* Employing equations in a computational algorithm is attractive because the equations can be adapted and applied to any manipulator. Once the inertia tensors, the link masses, the position vectors and the transformation matrices are specified for a particular manipulator, the equations may be applied directly to compute the joint torques corresponding to any motion. In the following section, the Lagrangian formulations are described.

3.3 THE LAGRANGIAN METHOD

The Newton-Euler approach is based on elementary dynamic formulae and on an analysis of forces and moments of constraint acting between the links. An alternative to the Newton-Euler method, is the Lagrangian dynamic method, which is based on the principle of conservation of energy. This provides a means of deriving the equations of motion from a scalar called the *Lagrangian*, which is defined as the difference between the kinetic and potential energy of a mechanical system. The method, which solves for the generalised forces F_i gives :

$$F_i = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{a}_i} \right) - \frac{\partial L}{\partial a_i}$$

where $L = K - P$, K is the kinetic energy,

P the potential energy, and

a_i is the displacement of the ith link from its initial position.

The recursive Lagrangian method may be the most convenient and efficient dynamic algorithm for formulations using homogeneous coordinates and 4x4 transformation matrices combining translation with rotation.

The standard formulation was derived by Uicker [28], who used 4x4 matrices to set up the Lagrangian-based inverse dynamics for a general linkage problem. Kahn [29] particularised the formulation to open-loop kinematic chains. In the Uicker/Kahn formulation, the numbers of multiplications and additions have an n^4 dependence.

Hollerbach [8] provided a detailed discussion of these formulations and also described a formulation developed by Waters [30], which was a Lagrangian dynamics algorithm with backward recursion (backward in the sense of the link numbering direction). With this formulation, the numbers of additions and multiplications were reduced to an n^2 dependence, primarily by means of a more efficient Coriolis and centrifugal force computation.

Hollerbach proceeded to develop a more efficient Lagrangian formulation of manipulator dynamics. The efficiency derived from recurrence relations for

the velocities, accelerations, and generalised forces. This recursive formulation was computed similarly to the recursive formulations of the Newton-Euler dynamics mentioned in the previous section and also had a *linear* computational cost. The n^2 cost in Waters' formulation was eliminated without significantly changing the coefficients of the other terms. Analytically, Hollerbach's formulation requires (412n-277) multiplications and (320n-201) additions.

Since the formulation of Hollerbach reduced the computational cost to an n dependence, a further reduction can be achieved only by reducing the size of the coefficients. The greatest such reduction can be obtained through a reformulation of the Lagrangian dynamics in terms of 3x3 matrices, rather than 4x4 rotation-translation matrices.

Thus, Hollerbach's formulation made it possible in principle, even using modest computers, to compute the Lagrangian dynamics in real time for a small number of joints.

Li [21] proposed an even faster inverse Lagrangian dynamics formulation, than Hollerbach, for robot manipulators. Li's approach was recursive in nature and was independent of the manipulator configuration. Once again, all computations were referenced to the link coordinate systems. The formulation can be applied to the real-time control and dynamic simulation of robot manipulators. For a manipulator with n degrees of freedom, the number of operations of the method is (120n-104) multiplications and (98n-94) additions.

3.4 COMPARISON OF THE AVAILABLE DYNAMIC FORMULATIONS

It is thus concluded that recursive formulations based either on the Lagrangian or Newton-Euler dynamics methods offer efficient algorithms for dynamics calculation.

Specifically, there are three parts to the Lagrangian reformulation :

1) backward recursion of the velocities and accelerations working from the base of the manipulator to the end link,

2) forward recursion of the generalised forces working from the end link to the base of the manipulator, and

3) the use of 3x3 rotation matrices instead of 4x4 rotation - translation matrices, using homogeneous coordinates.

This reformulation brings Lagrangian dynamics *into line* with the recursive Newton-Euler dynamics developed by Luh et al and Armstrong, whose computational complexity varies linearly with the number of joints and whose formulation contains parts *1)* and *2)* above. In the Newton-Euler method, some formulations use 3x3 matrices and others use 4x4.

Thus, in principle, the problem of computing manipulator dynamics in real time has been solved. It seems unlikely that any improvements in efficiency can be made to the present recursive formulations.

The choice between the formulations is therefore dependent upon the application, Silver [33]. This choice depends upon such factors as the control and specification of motion, the efficiency of the algorithm with respect to applicability of the problem being investigated, the composition of the equations, or upon whether the hierarchical structure of the mechanism is kinematically simple or complex and its chains are open or closed, or upon the nature of the internal and external environmental constraints, i.e. the description of external influences such as ground reaction, collision forces and torques, etc.

The formulations mentioned for Newton-Euler method give an efficient set of recursive equations, but are less well structured for control purposes. Hollerbach [8] and Li [21] use the Lagrangian method which is well structured and which can also be applied in matrix notation.

It can clearly be seen that formulations based on both methods have exactly the same pattern of computation. To compute the generalised forces for example, we first iterate from the base to the end-effector to compute the linear and angular velocities and accelerations, and then iterate from the end-effector to the base to compute the forces. For these formulations, the quantities associated with a given link are expressed in a coordinate system attached to that link, and then are transformed to the coordinates of the previous link as the iteration proceeds.

3.5 FORMULATIONS BASED ON ALTERNATIVE METHODS

In addition to what has been described above, Wilhelms [2] used the Gibbs-Appell direct dynamics formulation which had a cost of $O(n^4)$. She claimed that the control and environmental simulation that the formulation used were quite sophisticated, offered considerable generality and was useful for complex mechanisms. However, the $O(n^4)$ cost shows that the algorithm is not practical unless a configuration with only a small number of links is used which is not the case if an anthropomorphic model with many links and joints is considered. Wilhelms claimed that the formulation can be improved to $O(n^3)$ although this is not a great improvement when compared with the linear formulations.

Raibert [15] used the Space Configuration method, where he introduced tables to increase the speed of computation at the cost of large memory requirements, although he tabulated only the position-dependent terms. The computational complexity of his inverse dynamics formulation is $O(n^3)$, and is therefore practical only for a small number of links, while alternative techniques to tabulation were found to work better.

Featherstone [16] described an algorithm that uses articulated-body inertias. Featherstone claimed that the articulated-body method was the most efficient algorithm then developed for direct dynamics simulation. Featherstone's algorithm is linearly related to the number of joints and analytically, it requires (199n-198) multiplications and (174n-173) additions, This proves that his formulation is quite fast and is actually more efficient than Armstrong's algorithm.

Hashimoto & Kimura [22] presented a parallel algorithm for inverse dynamics. This is the so-called resolved Newton-Euler algorithm, which is based on a new description of the Newton-Euler formulation. The computational time is linear, (60n+193) floating-point operations for a manipulator with n joints, and the calculation is based on a network of processing elements which concurrently calculate the resolved Newton-Euler algorithm. With the model that we are presenting in Chapter 6, it is evident that the human figure can easily be broken down into different branches, and each single branch could be handled separately by a different processor. However, it is not clear, because of communication overheads, that such an approach necessarily

provides greater efficiency and this may depend on the precise structure of the figure being used.

Table 3.1 summarises the different formulations based on the existing methods that have been described and compared in this chapter.

Inventor	Type of Formulation	Cost	Origin
Armstrong	Newton-Euler (Direct Dynamics)	O(n)	Space Research
Walker	Newton-Euler (Inverse Dynamics)	$O(n^3)$	Robotics
Hollerbach	Lagrangian (Inverse Dynamics)	O(n)	Robotics
Wilhelms	Gibbs-Appell (Direct Dynamics)	$O(n^4)$	Biomechanics
Raibert	Space Configuration (Inv. Dynam.)	$O(n^3)$	Robotics
Featherstone	Articulated Inertias(Dir. Dynam.)	O(n)	Robotics
Li	Lagrangian (Inverse Dynamics)	O(n)	Robotics
Hashimoto & Kimura	Newton-Euler (Inverse Dynamics)	O(n)	Robotics

Table 3.1 - Existing Dynamic Formulations

3.6 CONCLUSIONS

An outline comparison has been made of the different formulations most frequently employed for dynamic analysis. It has been shown that Li's formulation provides the fastest inverse dynamics algorithm, and Featherstone's the fastest direct dynamics algorithm.

Both formulations have originated from robotics so their application is restricted to the use of unbranched kinematic chains. The fact that even a simple representation of the human figure contains branched chains requires the extension of these algorithms to be able to handle such chains, which in most cases is not easy to achieve. This is discussed in full in Chapter 6.

Because the articulated human figure has many links and joints, the articulated-body method was chosen as the basis of our simulation formulation. Although the method was originally proposed for use in robotics, it can easily be extended to accommodate the branched chains and multiple-degree-of-freedom joints required for modelling the realistic movement of the human figure. The implementation of this method is discussed in Chapters 4, 5 and 6.

In addition, this research has been carried out in parallel with the work of Vasilonikolidakis [64], who proposed an algorithm based on Hollerbach's inverse-dynamics formulation, which was also used for anthropomorphic manipulators. Thus, the chosen formulation enables comparisons to be drawn between the benefits associated with the two approaches of dynamic analysis.

The number of different methods and diverse approaches in existence shows that the problem of solving the dynamic equations to create complex movement is still quite difficult, without any unifying algorithm having claims to be the outstandingly best method. Furthermore, research is still going on in the area, which promises impending results.

CHAPTER 4
KINEMATIC ASPECTS OF SPATIAL ALGEBRA

4.1 INTRODUCTION

This chapter is concerned with the basic concept of a spatial vector, explaining why there is a need for it, and with the representation of physical vector quantities as spatial vectors to present computationally efficient mathematical formulations. The original idea about spatial vector algebra has been proposed by Featherstone [16, 19].

Having introduced the subject area, as well as explaining its usefulness, we go on to relate spatial vectors with line and free vectors. This enables us to form a spatial vector by the composition of its line and free vector components. We then present the operations defined upon the spatial vectors, so that the representation of well-known quantities using this spatial notation can be possible. Finally, the derivative of a vector in a moving coordinate system is formulated.

A particle has three degrees of freedom, so its velocity, acceleration, etc., can be represented by a 3-dimensional vector, and it has one vector equation of motion.

A rigid-body on the other hand, has six degrees of freedom, three translational and three rotational, yet rigid-body dynamics is conventionally described using 3-dimensional vectors.

So the location, displacement, velocity and acceleration of a rigid-body are each represented completely by a pair of vectors (one linear and one angular; six numbers, three describing the translational aspect and three describing the rotational aspect). There are two equations of motion.

The reason for this representation is that the linear and angular quantities (translation + rotation) are considered to be physically different quantities, but this does not preclude their amalgamation into 6-dimensional vectors as a matter of algebraic notational convenience. The spatial notation can be used to represent the combined linear and angular components (translation and rotation) of the velocity or acceleration of rigid-body systems, as single entities.

Thus, algorithms for dynamic formulations can be represented using a six-dimensional vector notation called *spatial notation*, described by

Featherstone [16, 19], which greatly facilitates the analysis of dynamics by reducing the size and number of equations involved. It simplifies the process of formulating the algorithms to be expressed clearly and concisely.

A single spatial vector can hold the information which otherwise requires two 3-dimensional vectors, and a single spatial equation can replace two 3-dimensional vector equations. The result is that the formulation algorithms can be described by fewer equations, relating fewer quantities, and the equations are usually shorter than their 3-dimensional counterparts.

Spatial vectors are represented by 6x1 matrices of components, and *spatial tensors* by 6x6 matrices. The operations of spatial algebra are implemented using the operations of standard arithmetic, with the exception that a different transpose operator is used.

In the following analysis spatial quantities are indicated by a caret, ^.

4.2 RELATIONSHIP BETWEEN SPATIAL VECTORS, LINE VECTORS AND FREE VECTORS

4.2.1 Introduction

This section describes the relationship between spatial vectors, line vectors and free vectors. The definitions of line and free vectors are first given, followed by their physical realisations. The decomposition of the spatial vector into its line and free vector components is then proposed.

4.2.2 Line and Free Vectors

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

A *line vector* has magnitude and direction and is restricted to lie in a definite line (i.e. line in a particular direction). Line vectors in mechanics 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.

For example, a line vector with magnitude and direction given by $\underline{m}$, restricted to lie in a line passing through the origin, is represented by the 6-element column vector $[m_x \; m_y \; m_z \; 0 \; 0 \; 0]^T$ or $[\underline{m}^T \; \underline{0}^T]^T$ for short. This is an example of a spatial quantity.

A line vector, $\underline{m}$, restricted to lie in the line defined by the equation

$$\underline{r} \times \underline{m} = \underline{m}_0 \qquad (\underline{m}.\underline{m}_0 = 0)$$

where $\underline{r}$ is a vector from the origin to any point on the line, is represented by the 6-vector

$$[\underline{m}^T \; \underline{m}_0{}^T]^T$$

where $\underline{m}$ defines the magnitude and direction of the line vector, independent of the origin, and $\underline{m}_0$ defines the position of the line.

We can clearly see that, when $\underline{m}_0 = \underline{0}$, then from the definition of the cross product, it follows that either $\underline{r} = \underline{m}$, or $\underline{r}$ is parallel to $\underline{m}$. Here, only the first possibility applies since, $\underline{r} = \underline{m}$ means that the line vector is restricted to pass through the origin and it therefore has the form $[\underline{m}^T \; \underline{0}^T]^T$, as mentioned above. On the other hand $\underline{r}$ parallel to $\underline{m}$ is a contradiction, since $\underline{r}$ is the position vector of the line.

To express the line vector in a parallel coordinate system with origin P, we use the equation

$$\underline{s} \times \underline{m} = \underline{m}_P \qquad (\underline{m}.\underline{m}_P = 0)$$

where $\underline{s}$ is the position of any point on the line as measured from P (see Figure 4.1).

Clearly, for any particular point on the line

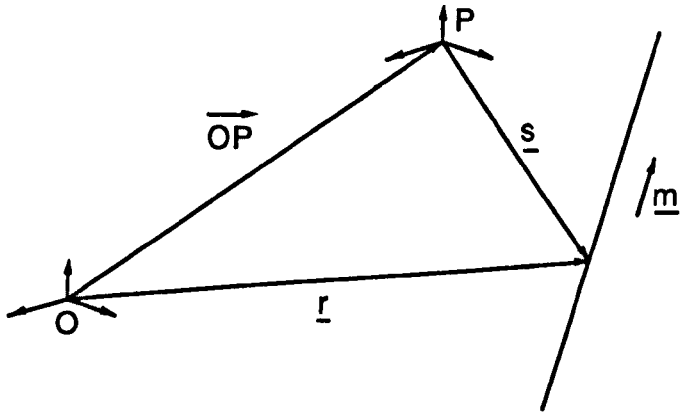
$$\underline{s} = \underline{r} + \overrightarrow{PO},$$

from which we obtain the translation rule

$$\underline{m}_P = \underline{m}_O + \overrightarrow{PO} \times \underline{m}. \qquad (4.1)$$

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

We can put free vectors on the same algebraic footing as line vectors, for example, a free vector, $\underline{n}$, may be represented by the 6-vector $[\underline{0}^T \ \underline{n}^T]$, for such a quantity obeys the translational rule and yet is independent of the choice of origin.



Translation of a coordinate system origin

Figure 4.1

4.2.3 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 ways in which a given spatial vector can be constructed from the sum of a line vector and a free vector, but if we constrain the line vector to pass through a given point then the line and free vector components of a spatial vector are determined uniquely.

The decomposition of a spatial vector into its line and free components is particularly simple if the line vector is restricted to pass through the origin, in which case the line and free vector components of a spatial vector $[\underline{m}^T \ \underline{m}_0^T]^T$ are $[\underline{m}^T \ \underline{0}^T]^T$ and $[\underline{0}^T \ \underline{m}_0^T]^T$ respectively.

A general spatial vector, $[\underline{m}^T \ \underline{m}_0^T]^T$ can be expressed uniquely as the sum of a line vector and a parallel free vector, see Appendix 4A,

$$\begin{bmatrix} \underline{m} \\ \underline{m}_0' \end{bmatrix} + \begin{bmatrix} \underline{0} \\ p\underline{m} \end{bmatrix}$$

where $\underline{m}_0' = \underline{m}_0 - p\underline{m}$ and $p = \underline{m}.\underline{m}_0 \ / \ \underline{m}.\underline{m}$.

A question might arise here, with regards to the importance of $\underline{m}_0'$, that is, the vector which makes $[\underline{m}^T \ \underline{m}_0'^T]^T$ a line vector whereas $[\underline{m}^T \ \underline{m}_0^T]^T$ does not. Referring to the definition of a line vector, given above, we conclude that a vector expressed by $\hat{\underline{m}} = [\underline{m}^T \ \underline{m}_0^T]^T$ is a line vector, *if and only if*, $\underline{m}.\underline{m}_0 = 0$.

Now, for $[\underline{m}^T \ \underline{m}_0'^T]^T$ to be a line vector, $\underline{m}.\underline{m}_0'$ must be equal to zero (a scalar). Indeed,

$$\begin{aligned} \underline{m}.\underline{m}_0' &= \underline{m}.(\underline{m}_0 - p\underline{m}) = \underline{m}.(\underline{m}_0 - (\underline{m}.\underline{m}_0 \ / \ \underline{m}.\underline{m})\underline{m}) \\ &= \underline{m}.\underline{m}_0 - \underline{m}.(\underline{m}.\underline{m}_0 / \ \underline{m}.\underline{m})\underline{m} \\ &= \underline{m}.\underline{m}_0 - \underline{m}.\underline{m}(\underline{m}.\underline{m}_0 / \underline{m}.\underline{m}) \\ &\quad \text{since } \underline{m}.\underline{m}_0 \text{ and } \underline{m}.\underline{m} \text{ are scalars,} \\ &= \underline{m}.\underline{m}_0 - \underline{m}.\underline{m}_0 \\ &= 0, \end{aligned}$$

which is the required result for $[\underline{m}^T \ \underline{m}_0'^T]^T$ to be a line vector.

4.3 OPERATIONS ON SPATIAL VECTORS

4.3.1 Introduction

For an expression involving spatial quantities to make physical sense there must be a physical interpretation for the mathematical operations in the expression as well as for the spatial quantities themselves. For example some permissible operations, that we have seen already, using spatial vectors, are :

- addition between spatial forces acting on the same body,
- addition between spatial velocities, and
- multiplication of a spatial vector, representing a joint axis, by a scalar representing the joint velocity (the first time derivative of the joint variable).

In the following subsections, additional spatial operations are defined. These include the transformation and the transpose spatial operator, and the scalar and cross product spatial operator.

4.3.2 The representation of Spatial Rigid-Body Coordinate Transformations

A rigid-body transformation, or Euclidean displacement, is one which leaves the distance between any two points invariant. In three dimensions this consists of a *rotation* and a *translation*, each having three degrees of freedom.

Consider the transformations of spatial vector space which correspond to rigid-body transformations in 3-dimensional Euclidean space.

To translate the origin of coordinates from O to P we use the translation rule Eq. 4.1, which gives,

$$\hat{\underline{m}}_P = \begin{bmatrix} \underline{m} \\ \underline{m}_0 + \overrightarrow{PO} \times \underline{m} \end{bmatrix} \qquad \text{where} \quad \hat{\underline{m}}_0 = \begin{bmatrix} \underline{m} \\ \underline{m}_0 \end{bmatrix}.$$

Next, let us introduce the *cross operator* which is defined as :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \tag{4.2}$$

From the definition of the operator, it follows that, the product of the matrix $\underline{a}\times$ with the vector $\underline{b}$ is equal to the vector cross product $\underline{a}\times\underline{b}$. This cross operator is a linear operator. It has some properties which are listed in Appendix 4E.

Rearranging, we can now express $\hat{\underline{m}}_P$ as the product of a 6x6 matrix with $\hat{\underline{m}}_O$, shown in Eq. 4.3 :

$$\hat{\underline{m}}_P = \begin{bmatrix} \underline{1} & \underline{0} \\ \overrightarrow{PO}\times & \underline{1} \end{bmatrix} \hat{\underline{m}}_O \qquad (4.3)$$

where $\underline{1}$ and $\underline{0}$ are the 3x3 identity and zero matrices respectively, and $\overrightarrow{PO}\times$ is a 3x3 anti-symmetric matrix obtained by applying the cross operator, $\times$, to $\overrightarrow{PO}$.

To effect a coordinate rotation from one coordinate system to another, consider the spatial vector $\hat{\underline{m}}$, to be the sum of a line vector through the origin and a free vector :

$$\hat{\underline{m}} = \begin{bmatrix} \underline{m} \\ \underline{0} \end{bmatrix} + \begin{bmatrix} \underline{0} \\ \underline{m}_O \end{bmatrix}.$$

The representation of $\hat{\underline{m}}$ in the rotated coordinate system is the sum of the rotated representations of the line vector and the free vector. Thus, the rotation of a 6-vector is accomplished by rotating its 3-vector components separately.

If E is the 3x3 orthogonal matrix that transforms the representation of a 3-vector in one coordinate system to that of another coordinate system rotated with respect to the first, then the representation of the line vector, which still acts through the origin in the rotated coordinate system, is $[(E\underline{m})^T\ \underline{0}^T]^T$, and the representation of the free vector is $[\underline{0}^T\ (E\underline{m}_O)^T]^T$.

So, the corresponding 6x6 rotation matrix is given by

$$\begin{bmatrix} E & \underline{0} \\ \underline{0} & E \end{bmatrix}.$$

Spatial transformations can be *combined* by *multiplying* the transformation matrices in the *correct order*. To obtain a consistency here, correct order has to be the pre-multiplication of matrices. For instance, the transformation formula for velocity, (defined later, in Section 4.5), is given by $\hat{\underline{v}}_O = {}_O\hat{X}_P\ \hat{\underline{v}}_P$, where the general form for a spatial transformation produced by a translation $\underline{r}$ followed by a rotation E is,

$$\begin{bmatrix} E & \underline{0} \\ \underline{0} & E \end{bmatrix} \begin{bmatrix} \underline{1} & \underline{0} \\ \underline{r}\times^T & \underline{1} \end{bmatrix} = \begin{bmatrix} E & \underline{0} \\ E\underline{r}\times^T & E \end{bmatrix} \tag{4.4}$$

where $\underline{r} = \overrightarrow{OP}$.

If you compare Eq. 4.3 with Eq. 4.4, for example, a question might arise as to whether you need $\underline{r}\times^T$ (transpose) and not $\underline{r}\times$. This is because $\underline{r}$, is now equal to $\overrightarrow{OP}$ and not to $\overrightarrow{PO}$. Thus, by the anti-symmetric property, the minus sign is preserved, which requires the introduction of the transpose operator.

The *inverse* transformation is

$$\begin{bmatrix} \underline{1} & \underline{0} \\ \underline{r}\times & \underline{1} \end{bmatrix} \begin{bmatrix} E^{-1} & \underline{0} \\ \underline{0} & E^{-1} \end{bmatrix} = \begin{bmatrix} E^{-1} & \underline{0} \\ \underline{r}\times E^{-1} & E^{-1} \end{bmatrix} \tag{4.5}$$

which is *not* the transpose of the above, so, unlike 3x3 rotation matrices, general spatial transformations are not orthogonal. Actually, to be more precise, general translational matrices are not orthogonal which implies that general transformation matrices are not orthogonal, but rotational matrices usually are.

Now, considering the inverse of each component matrix, it can be seen clearly that the inverse of $\begin{bmatrix} E & \underline{0} \\ \underline{0} & E \end{bmatrix}$ is $\begin{bmatrix} E^{-1} & \underline{0} \\ \underline{0} & E^{-1} \end{bmatrix}$, whereas the inverse of $\begin{bmatrix} \underline{1} & \underline{0} \\ \underline{r}\times^T & \underline{1} \end{bmatrix}$ is $\begin{bmatrix} \underline{1} & \underline{0} \\ \underline{r}\times & \underline{1} \end{bmatrix}$ because of the minus sign resulting from the anti-symmetric property.

Spatial coordinate transformations will be given leading and following subscripts indicating the destination and the source coordinate systems respectively.

Thus, ${}_0\hat{X}_P$ is a spatial transformation which operates on a vector represented in coordinate system P and produces a representation of the same vector in coordinate system 0.

Since spatial coordinate transformations may be combined by matrix multiplication, it follows that,

$$\hat{X}_{OP}\,\hat{X}_{PQ} = \hat{X}_{OQ}.$$

4.3.3 The Spatial Scalar Product

In ordinary vector algebra, the component of a vector, $\underline{a}$, in the direction of a unit vector, $\underline{s}$, is given by the scalar product, $\underline{a}.\underline{s}$.

The equivalent scalar product on spatial vectors, requires us 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.

If $\hat{\underline{a}} = [\underline{a}^T\ \underline{a}_0^T]^T$ and $\hat{\underline{s}} = [\underline{s}^T\ \underline{s}_0^T]^T$,

then the spatial scalar product based on the work done by a spatial force over a spatial displacement, e.g. the virtual work done by a force over an infinitesimal displacement, since finite displacements are not vectors, is given by

$$\hat{\underline{a}} \cdot \hat{\underline{s}} = \underline{a}.\underline{s}_0 + \underline{a}_0.\underline{s}.$$

Some *physical restrictions* apply on the spatial scalar product, that is, this scalar product is defined only between a *force-type vector* (one where the linear component is the line vector), and a *motion-type vector* (one where the angular component is the line vector).

The spatial scalar product of a vector with itself is not defined, so we can not use the normal definition of magnitude for a spatial vector. Two vectors satisfying $\hat{\underline{a}} \cdot \hat{\underline{s}} = 0$ are said to be orthogonal. The important property of this scalar product is that it is *not* positive definite. If we allow the expression $\hat{\underline{a}} \cdot \hat{\underline{a}}$, by overriding the restriction given above, we find that it may be positive, negative, or zero for non-zero $\hat{\underline{a}}$, so we still can not use the standard definition of magnitude.

In matrix algebra, the ordinary scalar product is given by $\underline{a}^T\underline{s}$, but the spatial equivalent is *not* given by $\hat{\underline{a}}^T\hat{\underline{s}}$. We therefore need to introduce the

spatial transpose operator ^{S}, defined by

$$\hat{\underline{a}}^S = \begin{bmatrix} \underline{a} \\ \underline{a}_0 \end{bmatrix}^S = [\underline{a}_0^T \; \underline{a}^T].$$

The definition of this spatial transpose operator is given in the next section.

4.3.4 The Spatial Transpose Operator

The definition of spatial transpose may be extended to scalars and 6x6 matrices by allocating it the same properties as ordinary transpose. The spatial transpose of a scalar is just the ordinary scalar. The spatial transpose of products and sums obey the rules

$$(\hat{A}\,\hat{B})^S = \hat{B}^S\,\hat{A}^S, \text{ and}$$

$$(\hat{A} + \hat{B})^S = \hat{A}^S + \hat{B}^S.$$

From the above definitions, we have that

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

for any vector, $\hat{\underline{a}}$, and matrix $\hat{A}$.
(since $\hat{\underline{a}}^S\hat{A}\hat{\underline{a}}$ is a scalar, and $(\hat{\underline{a}}^S)^S = \hat{\underline{a}}$).

This results in the following spatial transpose definition :

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^S = \begin{bmatrix} D^T & B^T \\ C^T & A^T \end{bmatrix}. \qquad (4.6)$$

To illustrate this form , let $\hat{\underline{a}} = \begin{bmatrix} \underline{a}_1 \\ \underline{a}_2 \end{bmatrix}$ and $\hat{\underline{b}} = \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \end{bmatrix}$,

therefore, $\hat{\underline{a}}^S = [\underline{a}_2^T \; \underline{a}_1^T]$ and $\hat{\underline{b}}^S = [\underline{b}_2^T \; \underline{b}_1^T]$.

Now, let $\begin{bmatrix} A & B \\ C & D \end{bmatrix}^S = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}$.

Let, $M = \hat{\underline{a}}^S \begin{bmatrix} A & B \\ C & D \end{bmatrix} \hat{\underline{b}}$

$$= [\underline{a}_2^T \quad \underline{a}_1^T] \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \end{bmatrix}$$

$$= [\underline{a}_2^T A + \underline{a}_1^T C \quad \underline{a}_2^T B + \underline{a}_1^T D] \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \end{bmatrix}$$

$$= \underline{a}_2^T A \underline{b}_1 + \underline{a}_1^T C \underline{b}_1 + \underline{a}_2^T B \underline{b}_2 + \underline{a}_1^T D \underline{b}_2. \qquad (i)$$

But M is a scalar, therefore $M = M^S$. It then follows that,

$$M = \hat{\underline{b}}^S \begin{bmatrix} A & B \\ C & D \end{bmatrix}^S (\hat{\underline{a}}^S)^S$$

$$= [\underline{b}_2^T \quad \underline{b}_1^T] \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} \begin{bmatrix} \underline{a}_1 \\ \underline{a}_2 \end{bmatrix}$$

$$= [\underline{b}_2^T W + \underline{b}_1^T Y \quad \underline{b}_2^T X + \underline{b}_1^T Z] \begin{bmatrix} \underline{a}_1 \\ \underline{a}_2 \end{bmatrix}$$

$$= \underline{b}_2^T W \underline{a}_1 + \underline{b}_1^T Y \underline{a}_1 + \underline{b}_2^T X \underline{a}_2 + \underline{b}_1^T Z \underline{a}_2$$

$$= (\underline{b}_2^T W \underline{a}_1)^T + (\underline{b}_1^T Y \underline{a}_1)^T + (\underline{b}_2^T X \underline{a}_2)^T + (\underline{b}_1^T Z \underline{a}_2)^T$$

since all these quantities are scalars,

$$= \underline{a}_1^T W^T \underline{b}_2 + \underline{a}_1^T Y^T \underline{b}_1 + \underline{a}_2^T X^T \underline{b}_2 + \underline{a}_2^T Z^T \underline{b}_1. \qquad (ii)$$

Thus, for arbitrary $\underline{a}_1$, $\underline{a}_2$, $\underline{b}_1$, $\underline{b}_2$, equations (*i*) and (*ii*) are satisfied only if,

$$W^T = D, \qquad Y^T = C, \qquad X^T = B, \qquad \text{and} \qquad Z^T = A.$$

That is, $\begin{bmatrix} A & B \\ C & D \end{bmatrix}^S = \begin{bmatrix} D^T & B^T \\ C^T & A^T \end{bmatrix}$.

(See also Appendix 4C).

4.3.5 The Spatial Cross Operator

If $\underline{a}$ and $\underline{b}$ are two 3x1 vectors amalgamated to form a spatial vector then the cross operator, $\hat{\times}$, applied to the spatial vector is defined as

$$\begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} \hat{\times} = \begin{bmatrix} \underline{a}\times & \underline{0} \\ \underline{b}\times & \underline{a}\times \end{bmatrix}$$

where $\times$ is the cross operator defined by

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$

Following this, we represent the breakdown into the conventional vector combinations by,

$$\begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} \hat{\times} \begin{bmatrix} \underline{c} \\ \underline{d} \end{bmatrix} = \left(\begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} \hat{\times} \right) \begin{bmatrix} \underline{c} \\ \underline{d} \end{bmatrix} = \begin{bmatrix} \underline{a}\times & \underline{0} \\ \underline{b}\times & \underline{a}\times \end{bmatrix} \begin{bmatrix} \underline{c} \\ \underline{d} \end{bmatrix}$$

$$= \begin{bmatrix} \underline{a}\times\underline{c} \\ \underline{b}\times\underline{c} + \underline{a}\times\underline{d} \end{bmatrix}.$$

4.4 REPRESENTATIONS USING SPATIAL QUANTITIES

4.4.1 Introduction

Accepting the fact that quantities of a physically different nature, like the linear and angular velocity, can be combined as components of a single entity, it is possible to represent common quantities using the spatial vector notation.

In the next sections we are looking at how, for example, velocities, accelerations, etc. can be represented using this notation. The representation of the forces as well as the representation of the joint axis, to help us give a precise expression of the spatial velocities and accelerations, are developed in the subsequent sections.

4.4.2 Spatial Velocity

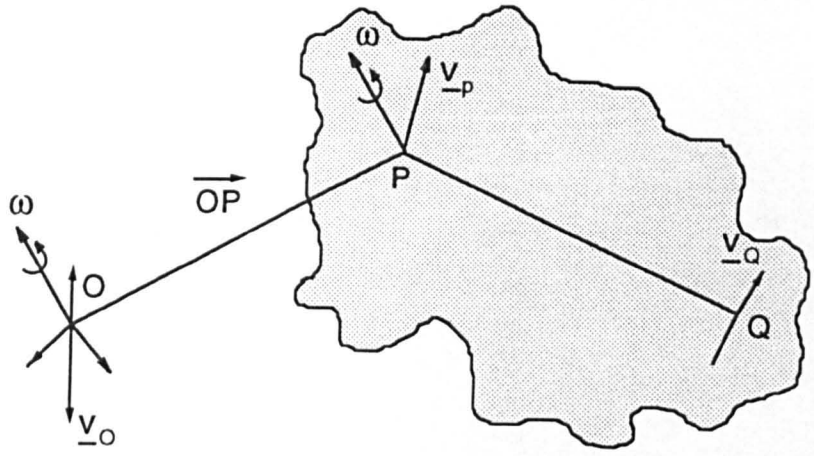
Consider how the velocity of a rigid body can be expressed as a spatial vector. The instantaneous spatial velocity of a rigid body may be described by the linear velocity, $\underline{v}_P$, of some point P moving with the rigid body, and its angular velocity, ω.

The body is considered to be rotating with angular velocity ω and at the same time it is moving (translating) with linear velocity $\underline{v}_P$: here $\underline{v}_P$ applies only to the point P, while ω applies to the body as a whole, and is independent of the choice of P. This is illustrated in Figure 4.2. The choice of P is arbitrary. It need not be physically within the rigid body, as long as it has the same motion as the body.

The instantaneous velocity $\underline{v}_Q$ of any other point Q in the rigid body can be expressed in terms of ω and $\underline{v}_P$ as

$$\underline{v}_Q = \underline{v}_P + \overrightarrow{QP} \times \omega \qquad (4.7)$$

i.e. this quantity obeys the translational rule defined above so we may express the spatial velocity of the rigid body in any coordinate system.



Representation of rigid-body velocity

Figure 4.2

Thus, the velocity is described completely by the pair of vectors $(\omega, \underline{v}_P)$ given the point P, and the body may be considered to be rotating with angular velocity ω about an axis passing through P whilst simultaneously translating with linear velocity $\underline{v}_P$.

The factor $\overrightarrow{QP} \times \omega$ added to the linear velocity at P has had the effect of shifting the axis of rotation from passing through P to passing through Q.

The pair $(\omega, \underline{v}_P)$ defines the spatial velocity of the rigid body. Thus, in spatial notation, we say that $\hat{\underline{v}}$ is the spatial velocity of the rigid body, where $\hat{\underline{v}} = [\omega_x \ \omega_y \ \omega_z \ v_{P_x} \ v_{P_y} \ v_{P_z}]^T$ abbreviated to $\hat{\underline{v}} = [\omega^T \ \underline{v}_P{}^T]^T$.

The spatial velocity pair is given in that order rather than the alternative $(\underline{v}_P, \omega)$, because it is then possible for $\hat{\underline{v}}_0 = [\omega^T \ \underline{v}_0{}^T]^T$ to obey the translation rule Eq. 4.1 which is of the form

$$\underline{v}_P = \underline{v}_0 + \overrightarrow{PO} \times \omega$$

i.e. the corresponding transformation rule for velocities. See also the definition of the spatial cross operator $\hat{\times}$, which interchanges the order of the translational and rotational components of the spatial vector.

4.4.3 Spatial Acceleration

Using the expression of the spatial velocity let us now define expressions for the spatial acceleration.

The absolute acceleration of a rigid body is the absolute derivative of its spatial velocity. Therefore, similarly to the spatial velocity, the spatial acceleration of a rigid body may be expressed as an angular acceleration about an axis passing through the origin, together with a linear acceleration.

Spatial accelerations will obey the transformation rule if both the body and coordinate system are stationary.

Consider a rigid body with angular velocity ω, and with linear velocity of a point P in the rigid body, $\dot{\underline{r}}$, where $\underline{r}$ is the position vector of P. The spatial velocity of the body in stationary coordinates at O is

$$\hat{\underline{v}}_0 = \begin{bmatrix} \omega \\ \dot{\underline{r}} + \underline{r}\times\omega \end{bmatrix}$$

and its absolute spatial acceleration is the component-wise derivative of this, which is shown in Eq. 4.8,

$$\hat{\underline{a}}_0 = \begin{bmatrix} \dot{\omega} \\ \ddot{\underline{r}} + \dot{\underline{r}}\times\omega + \underline{r}\times\dot{\omega} \end{bmatrix}. \qquad (4.8)$$

The angular component of the spatial acceleration is the angular acceleration of the body, but the linear component is the rate of change over time of the velocity of whichever point in the rigid body happens to be passing through the origin.

4.4.4 Representation of Joint Axes

The essential feature of a joint is that it allows some degree of relative motion between the two bodies that it connects. In the case of a one-degree-of-freedom joint, the relative position of the two bodies is a function of a scalar called the joint variable, and we can express the

relative velocity of the two bodies in terms of a joint velocity which is the derivative of the joint variable.

The joint axis is a spatial vector which defines the direction and nature of motion allowed by the joint, and the relative velocity of the two bodies is obtained by multiplying the axial joint (a vector), by the speed of the joint velocity (a scalar).

Once a joint axis is represented as a spatial vector, the type of motion allowed by the joint becomes irrelevant. For example, it is irrelevant if the representation is for revolute, prismatic, or screw joints; a revolute joint allows only rotation between the bodies it connects, a prismatic only translation, whereas screw joints combine rotation with translation. There is no need to have separate equations to deal with each different joint type.

Let us express the relationship between the linear and angular velocities of two bodies connected by one-degree-of-freedom joint.

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 can be expressed in terms of a joint axis and a scalar called the joint velocity (or rate),

$$\hat{\underline{s}}\,\dot{q} = [s^T\dot{q} \quad s_0^T\dot{q}]^T.$$

That is, the joint axis is a spatial vector which defines the direction and nature of motion allowed by the joint and the spatial relative velocity of two bodies is obtained by multiplying the joint axis by the joint velocity, (s and s_0 are two 3x1 vectors), i.e. $\hat{\underline{s}}\,\dot{q}$, where

$\hat{\underline{s}} = [s^T \ s_0^T]^T$ is the joint axis

and

$\dot{q}$ is the magnitude of the velocity, a scalar.

Now, $\hat{\underline{s}}$ is given with respect to the origin, and the spatial relative velocity of b_2 with respect to b_1 is

$$\hat{\underline{v}}_2 = \hat{\underline{v}}_1 + \hat{\underline{s}}\,\dot{q}. \qquad (4.9)$$

To illustrate this, consider two bodies having a common joint. See Figure 4.3. The joint allows the relative motion of the two bodies about the joint axis $\hat{\underline{s}}$. That is, the joint is the only point that stays in the same position while it allows the rotation of the two bodies.

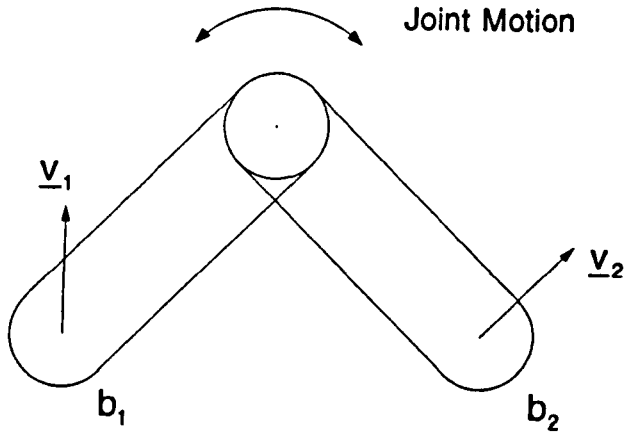


Figure 4.3

Now the relative velocity $\hat{\underline{v}}_2$ of body b_2 with respect to body b_1 is $\hat{\underline{v}}_1$ plus the joint axis multiplied by the relative velocity of it.

To extend this to a robot having n+1 links numbered 0..n and n joints numbered 1..n, (where link 0 is stationary, and link i-1 is connected to link i by joint i) consider again the axis of joint i being $\hat{\underline{s}}_i$ and its scalar joint velocity being $\dot{q}_i$, measured from link i-1 to link i.

Thus, the relative velocity of a link j with respect to link j-1 is then

$$\hat{\underline{s}}_j\,\dot{q}_j,$$

and the absolute velocity of link j is

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

The velocity of a link j with respect to link j-1 as given above, implies that the acceleration, $\hat{\underline{a}}_j$, the derivative of $\hat{\underline{v}}_j$ is given by,

$$\hat{\underline{a}}_j = \frac{d}{dt} \sum_{i=1}^{j} \hat{\underline{s}}_i \dot{q}_i$$

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

Now, $\hat{\underline{s}}_i$ is moving with velocity $\hat{\underline{v}}_i$, so $\frac{d}{dt} \hat{\underline{s}}_i = \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i$, which is the apparent derivative of $\hat{\underline{s}}$ (see equation in Appendix 4B), and the acceleration is given by,

$$\hat{\underline{a}}_j = \sum_{i=1}^{j} \left(\hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{q}_i + \hat{\underline{s}}_i \ddot{q}_i \right) \qquad (4.10)$$

where the definition of $\hat{\times}$, the spatial cross operator, is as given earlier.

Another method to calculate the acceleration is as follows.

If $\hat{\underline{v}}_i$ is the absolute spatial velocity of link i, then

$$\hat{\underline{v}}_i = \hat{\underline{v}}_{i-1} + \hat{\underline{s}}_i \dot{q}_i.$$

Differentiating this equation gives,

$$\hat{\underline{a}}_i = \hat{\underline{a}}_{i-1} + \hat{\underline{s}}_i \ddot{q}_i + \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{q}_i \qquad (4.11)$$

where $\hat{\underline{a}}_i$ and $\hat{\underline{a}}_{i-1}$ are the absolute spatial accelerations of links i and i-1 respectively. Given that

$$\hat{\underline{v}}_0 = \hat{\underline{a}}_0 = \hat{\underline{0}}, \qquad (4.12)$$

successive applications of these formulae with i taking values from 1 to n, will give the velocity and acceleration of each link in turn.

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

$$\hat{\underline{v}}_n = \sum_{i=1}^{n} \hat{\underline{s}}_i \dot{q}_i. \tag{4.13}$$

We can therefore find the spatial acceleration of the end-effector in terms of the scalar joint accelerations $\ddot{q}$,

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

4.4.5 Representation of Forces

Having represented in spatial notation the velocity, acceleration and joint axes, we can go further and try to represent forces using the same notation.

A force acting on a rigid body has magnitude and a line of action. It may therefore be represented as a line vector. A spatial force with magnitude and direction given by $\underline{f}$ and acting along a line passing through a point P is given by

$$\hat{\underline{f}} = [\underline{f}^T \quad (\overrightarrow{OP} \times \underline{f})^T]^T. \tag{4.14}$$

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 magnitude and direction only, and may therefore be represented as a free vector.

A system of forces acting on a rigid body may be reduced to a single force acting in a line passing through the origin, together with a couple which is a free vector.

If several spatial forces are acting on a single rigid body, then their resultant is a single spatial force which is simply the vector sum of the individual forces.

4.5 DIFFERENTIATION IN MOVING COORDINATES

4.5.1 Introduction

The previous sections studied the fundamental aspects of spatial quantities and dealt with the kinematics of spatial algebra.

In the next section we are going to be looking at how a spatial vector can be *differentiated.*

4.5.2 The Derivative of a Vector in a Moving Coordinate System

The derivative of a spatial vector can be transformed like a spatial vector. Thus, to find the derivative of a vector in a moving coordinate system, we transform the vector to a stationary coordinate system, differentiate it at the stationary coordinate system, and transform the derivative back to the moving coordinate system.

For example, let P be a moving coordinate system and O a stationary one. Let ${}_O\hat{X}_P$ and ${}_P\hat{X}_O$ be 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 represented in P coordinates is given by

$$\frac{d}{dt}\,\hat{\underline{m}} = {}_P\hat{X}_O\,\frac{d'}{dt}\,({}_O\hat{X}_P\,\hat{\underline{m}})$$

i.e. transform the vector $\hat{\underline{m}}$, to the stationary coordinate system O differentiate it there and transform its derivative back to the moving coordinate system P.

Now, $$\frac{d}{dt}\,\hat{\underline{m}} = \frac{d'}{dt}\,\hat{\underline{m}} + {}_P\hat{X}_O\,(\frac{d'}{dt}\,{}_O\hat{X}_P)\,\hat{\underline{m}}. \qquad (4.15)$$

To evaluate ${}_P\hat{X}_O\,(\,\frac{d'}{dt}\,{}_O\hat{X}_P\,)$ consider 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

$$ {}_0\hat{X}_P = \begin{bmatrix} E & \underline{0} \\ \underline{r}\times E & E \end{bmatrix} $$

and

$$ \frac{d'}{dt}\,{}_0\hat{X}_P = \begin{bmatrix} \frac{d'}{dt}(E) & \underline{0} \\ (\frac{d'}{dt}\underline{r})\times E + \underline{r}\times\frac{d'}{dt}(E) & \frac{d'}{dt}(E) \end{bmatrix}. \qquad (4.16) $$

Let P coordinates have velocity $\hat{\underline{v}}$ expressed in O coordinates as $\hat{\underline{v}}_0 = [\ \omega^T\ \underline{v}_0^T\ ]^T$ and $\underline{r}$ is a vector from O to P coordinates.

Now, $\frac{d'}{dt}\,\underline{r} = \underline{v}_P = \underline{v}_0 - \underline{r}\times\omega$ (the translation rule)

and $\frac{d'}{dt}\,E = \omega\times E$

(differentiation of an orthogonal matrix, Bottema & Roth [20] - see Appendix 4D).

So, considering each component of the above matrix separately, we have,

$$ \frac{d'}{dt}\,E = \omega\times E $$

$$ \begin{aligned} (\frac{d'}{dt}\,\underline{r})\times E + \underline{r}\times\frac{d'}{dt}\,E &= (\underline{v}_0 - \underline{r}\times\omega)\times E + \underline{r}\times(\omega\times E) \\ &= \underline{v}_0\times E - (\underline{r}\times\omega)\times E + \underline{r}\times(\omega\times E) \\ &= \underline{v}_0\times E - [(\underline{r}\times\omega\times - \omega\times\underline{r}\times)E] + \underline{r}\times(\omega\times E) \\ &\quad \text{(using property 6 - see Appendix 4E)} \\ &= \underline{v}_0\times E - \underline{r}\times\omega\times E + \omega\times\underline{r}\times E + \underline{r}\times\omega\times E \\ &= \underline{v}_0\times E + \omega\times\underline{r}\times E. \end{aligned} $$

Thus, $$ \frac{d'}{dt}\,{}_0\hat{X}_P = \begin{bmatrix} \omega\times & \underline{0} \\ \underline{v}_0\times & \omega\times \end{bmatrix} \begin{bmatrix} E & \underline{0} \\ \underline{r}\times E & E \end{bmatrix} $$

$$ = \begin{bmatrix} \omega\times & \underline{0} \\ \underline{v}_0\times & \omega\times \end{bmatrix} {}_0\hat{X}_P. \qquad (4.17) $$

Again, introducing the *spatial cross operator*, $\hat{\times}$, defined by

$$\begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix}\hat{\times} = \begin{bmatrix} \underline{a}\times & \underline{0} \\ \underline{b}\times & \underline{a}\times \end{bmatrix}$$

it follows that,

$$\frac{d'}{dt}\,{}_0\hat{X}_P = \hat{\underline{v}}_0 \,\hat{\times}\, {}_0\hat{X}_P \tag{4.18}$$

where $\hat{\underline{v}}_0 = \begin{bmatrix} \omega \\ \underline{v}_0 \end{bmatrix}$.

The spatial cross operator is the spatial analogue of the cross operator described earlier. It is a linear operator, with spatial vectors and coordinate transformations in place of the ordinary vectors and 3x3 matrices.

Now, let $\hat{\underline{v}}_P = {}_P\hat{X}_0\,\hat{\underline{v}}_0$ be the velocity of P coordinates represented in O coordinates, then we get

$$\frac{d}{dt}\,\hat{\underline{m}} = \frac{d'}{dt}\,\hat{\underline{m}} + \hat{\underline{v}}_P \,\hat{\times}\, \hat{\underline{m}}. \tag{4.19}$$

Proof :

$$\frac{d}{dt}\,\hat{\underline{m}} = {}_P\hat{X}_0\,\frac{d'}{dt}\,({}_0\hat{X}_P\,\hat{\underline{m}})$$

$$= \frac{d'}{dt}\,\hat{\underline{m}} + {}_P\hat{X}_0\,(\frac{d'}{dt}\,{}_0\hat{X}_P)\,\hat{\underline{m}}$$

but

$$\frac{d}{dt}\,{}_0\hat{X}_P = \hat{\underline{v}}_0 \,\hat{\times}\, {}_0\hat{X}_P,$$

therefore

$$\frac{d}{dt}\,\hat{\underline{m}} = \frac{d'}{dt}\,\hat{\underline{m}} + {}_P\hat{X}_0\,(\hat{\underline{v}}_0 \,\hat{\times}\, {}_0\hat{X}_P)\,\hat{\underline{m}}.$$

Using the tensor transformation rule :

$$\hat{\underline{v}}_P = {}_P\hat{X}_0\,\hat{\underline{v}}_0\,{}_0\hat{X}_P$$

it follows that,

$$\frac{d}{dt}\hat{\underline{m}} = \frac{d'}{dt}\hat{\underline{m}} + \hat{\underline{v}}_P \hat{\times} \hat{\underline{m}} \qquad \text{(that is, Eq. 4.19).}$$

By applying this rule to $\hat{\underline{v}}_0 \hat{\times}$ we get

$$\hat{\underline{v}}_P \hat{\times} = {}_P\hat{X}_0 \, (\hat{\underline{v}}_0 \hat{\times}) \, {}_0\hat{X}_P .$$

So, 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.

The last equation may be used to find the derivative of a moving vector as follows.

Let the vector $\hat{\underline{m}}$, represented in stationary coordinates O, be moving with velocity $\hat{\underline{v}}$. Let P be a coordinate system moving with velocity $\hat{\underline{v}}$, so $\hat{\underline{m}}$ is stationary in P, then,

$$\begin{aligned} \frac{d}{dt}\hat{\underline{m}} &= {}_0\hat{X}_P \left(\frac{d'}{dt}\hat{\underline{m}}_P + \underline{v}_P \hat{\times} \underline{m}_P\right) \\ &= {}_0\hat{X}_P \, (\hat{\underline{v}}_P \hat{\times} \hat{\underline{m}}_P) \\ &= \hat{\underline{v}} \hat{\times} \hat{\underline{m}} \end{aligned} \qquad (4.20)$$

since $\hat{\underline{m}}$ is represented in O coordinates.

For the special case, if we consider O to be a moving coordinate system and $\hat{\underline{v}}$ the apparent velocity of ω in O, then the last equation gives the *apparent derivative* of $\hat{\underline{m}}$ in O.

4.6 CONCLUDING REMARKS

A spatial vector, that is a 6-dimensional vector space over the real numbers, can be interpreted geometrically as one having a *line vector* component and a *free vector* component. Thus, linear and angular physical quantities, e.g. linear and angular velocity, acceleration, force, etc, may be formed from a spatial vector if it makes physical sense to regard one of them as a line vector and the other as a free vector.

The spatial notation unites the rotational and translational components of motion into a single vector quantity. For example, the spatial representation of a joint axis specifies the axis about which rotations occur, and also the location of that axis in 3 space. Similarly, the spatial velocity contains both the angular and linear velocity, and is transformed and manipulated as a single quantity. The general form of a spatial vector is as follows,

$$\hat{\underline{a}} = \begin{bmatrix} \underline{a} \\ \underline{a}_0 \end{bmatrix}$$

where $\underline{a}$ is a 3-dimensional vector which specifies a magnitude and direction, and $\underline{a}_0$ specifies the location of $\underline{a}$ in the following manner :

$$\underline{r} \times \underline{a} = \underline{a}_0$$

where $\underline{r}$ is the 3-vector from the coordinate frame origin to the location of vector $\underline{a}$.

In a similar fashion, the *spatial force* expresses linear force $\underline{f}$ and rotational force (torque) $\underline{\tau}$ as

$$\hat{\underline{f}} = \begin{bmatrix} \underline{f} \\ \underline{\tau} \end{bmatrix}.$$

Spatial acceleration expresses linear acceleration $\underline{a}$ and rotational acceleration $\underline{\alpha}$ as

$$\hat{\underline{a}} = \begin{bmatrix} \underline{\alpha} \\ \underline{a} \end{bmatrix}.$$

Spatial velocity expresses linear velocity $\underline{v}$ and rotational velocity $\underline{\omega}$ as

$$\hat{\underline{v}} = \begin{bmatrix} \underline{\omega} \\ \underline{v} \end{bmatrix}.$$

Notice that the rotational and translational components are reversed between velocity and acceleration on the one hand and force on the other. This is because of the way the spatial quantities translate.

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.

The scalar product between a motion-type vector and a force-type vector is defined but the scalar product between two motion-type vectors or between two force-type vectors is not defined. To implement the scalar product as a matrix operation, a new *spatial transpose operator* needs to be introduced and used in the place of the ordinary transpose operator. This is given by

$$\hat{\underline{a}}^S = \begin{bmatrix} \underline{a} \\ \underline{a}_0 \end{bmatrix}^S = [\underline{a}_0^T \quad \underline{a}^T] \quad \text{and}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^S = \begin{bmatrix} D^T & B^T \\ C^T & A^T \end{bmatrix}.$$

The use of spatial transpose makes spatial coordinate transformations *orthogonal*, spatial rigid-body inertias *symmetric*, and spatial cross-product matrices *anti-symmetric*.

Transformation of spatial vectors from one coordinate system to another is possible which is represented by a 6x6 matrix. These transformations can be combined and therefore be applied to vectors by matrix multiplication and are given leading and following subscripts to indicate the order of the transformation.

The *derivative* of a spatial vector represented in a stationary coordinate system is its component-wise derivative. The derivative 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.

APPENDIX 4A -

Addition of Spatial Vectors and Multiplication of Spatial Vector by a Scalar

To show these, we are going to use the spatial vector as the combination of its line and free vector components.

A general spatial vector, $\hat{\underline{m}}$ is of the form

$$\hat{\underline{m}} = [\underline{m}^T \ \underline{m}_0{}^T]^T.$$

Taking a line vector, with magnitude and direction given by $\underline{m}$, restricted to lie in a line passing through the origin, and giving its 6-dimensional column vector representation, we get,

$$[\underline{m}^T \ \underline{0}^T]^T.$$

Now, taking a free vector, $\underline{m}_0$ and representing it by the 6-vector notation, we have a spatial vector of the form

$$[\underline{0}^T \ \underline{m}_0{}^T]^T.$$

If we try to add the line and free vectors together, i.e.

$$[\underline{m}^T \ \underline{0}^T]^T + [\underline{0}^T \ \underline{m}_0{}^T]^T,$$

or using the expanded notation, we obtain,

$$\begin{bmatrix} \underline{m} \\ \underline{0} \end{bmatrix} + \begin{bmatrix} \underline{0} \\ \underline{m}_0 \end{bmatrix} = \begin{bmatrix} \underline{m} \\ \underline{m}_0 \end{bmatrix} = [\underline{m}^T \ \underline{m}_0{}^T]^T,$$

that is, the initial general spatial vector.

Thus, adding a line vector and a free vector together gives a spatial vector. Let us now consider the addition of two general spatial vectors.

(a) Addition of two spatial vectors.

If we take two spatial vectors, $\hat{\underline{s}}$ and $\hat{\underline{m}}$, both represented in the same coordinate system, then from the definition of the representation of spatial vector we have

$$\hat{\underline{s}} = [\underline{s}^T \; \underline{s}_0{}^T]^T$$

where

$$\underline{s}_0 = \underline{r} \times \underline{s} = \begin{bmatrix} \underline{i} & \underline{j} & \underline{k} \\ r_1 & r_2 & r_3 \\ s_1 & s_2 & s_3 \end{bmatrix}$$

$$= \underline{i}(r_2 s_3 - r_3 s_2) - \underline{j}(r_1 s_3 - r_3 s_1) + \underline{k}(r_1 s_2 - r_2 s_1)$$

($\underline{r}$ is the position vector of any point on the axis).

Therefore,

$$\hat{\underline{s}} = [\underline{s}^T \; \underline{s}_0{}^T]^T$$

$$= [s_1, \; s_2, \; s_3, \; r_2 s_3 - r_3 s_2, \; r_3 s_1 - r_1 s_3, \; r_1 s_2 - r_2 s_1].$$

Similarly,

$$\hat{\underline{m}} = [\underline{m}^T \; \underline{m}_0{}^T]^T$$

$$= [m_1, \; m_2, \; m_3, \; r_2 m_3 - r_3 m_2, \; r_3 m_1 - r_1 m_3, \; r_1 m_2 - r_2 m_1].$$

Now, adding them together, i.e. $\hat{\underline{s}} + \hat{\underline{m}}$, we get another spatial vector which has magnitude equal to the addition of magnitudes of $\hat{\underline{s}}$ and $\hat{\underline{m}}$, since both of them are represented in the same coordinate system. That is,

$$\hat{\underline{s}} + \hat{\underline{m}} = [s_1 + m_1, \; s_2 + m_2, \; s_3 + m_3, \; r_2(s_3 + m_3) - r_3(s_2 + m_2),$$
$$r_3(s_1 + m_1) - r_1(s_3 + m_3), \; r_1(s_2 + m_2) - r_2(s_1 + m_1)]$$

$$= [(\underline{s} + \underline{m})^T \; (\underline{s}_0 + \underline{m}_0)^T],$$

which is another spatial vector in the same coordinate system as $\hat{\underline{s}}$ or $\hat{\underline{m}}$.

(b) Multiplication of a spatial vector by a scalar k.

$\hat{\underline{s}}$ k will give a vector in the direction of $\hat{\underline{s}}$ with magnitude equal to the original magnitude of $\hat{\underline{s}}$ times the scalar k.

That is, addition of spatial vectors and multiplication of a spatial vector by a scalar follow normal rules.

APPENDIX 4B -

Apparent Derivative of $\hat{\underline{s}}$

If we let $\hat{\underline{v}}_P = {}_P\hat{X}_0 \, \hat{\underline{v}}_0$ be the velocity of P represented in P coordinates then we get

$$\frac{d}{dt} \hat{\underline{s}} = \frac{d'}{dt} \hat{s} + \hat{\underline{v}}_P \times \hat{\underline{s}}$$

(see differentiation of a spatial vector, Section 4.5).

Now, let the vector $\hat{\underline{s}}$, represented in stationary coordinates O, be moving with velocity $\hat{\underline{v}}$.

Let P be a coordinate system moving with velocity $\hat{\underline{v}}$, so $\hat{\underline{s}}$ is stationary in P, then,

$$\frac{d}{dt} \hat{\underline{s}} = {}_0\hat{X}_P \left(\frac{d'}{dt} \hat{\underline{s}}_P + \hat{\underline{v}}_P \times \hat{\underline{s}}_P\right)$$

$$= {}_0\hat{X}_P \left(\hat{\underline{v}}_P \times \hat{\underline{s}}_P\right)$$

$$= \hat{\underline{v}} \times \hat{\underline{s}}$$

that is, the apparent derivative of $\hat{\underline{s}}$ in O.

APPENDIX 4C -

Illustration of the form of the Spatial Transpose Operator

To demonstrate that the spatial transpose of products and sums obeys the same rules as the standard transpose operator, consider first $(\hat{A}\ \hat{B})^S$. Using the product of two matrices and the definition of the spatial transpose operator, Eq. 4.6,

$$(\hat{A}\ \hat{B})^S = \left[\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix}\begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix}\right]^S = \begin{bmatrix} A_1A_2+B_1C_2 & A_1B_2+B_1D_2 \\ C_1A_2+D_1C_2 & C_1B_2+D_1D_2 \end{bmatrix}^S$$

$$= \begin{bmatrix} (C_1B_2+D_1D_2)^T & (A_1B_2+B_1D_2)^T \\ (C_1A_2+D_1C_2)^T & (A_1A_2+B_1C_2)^T \end{bmatrix}. \qquad (4C.1)$$

On the other hand, using the spatial transpose operator,

$$\hat{B}^S\ \hat{A}^S = \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix}^S \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix}^S = \begin{bmatrix} D_2^T & B_2^T \\ C_2^T & A_2^T \end{bmatrix}\begin{bmatrix} D_1^T & B_1^T \\ C_1^T & A_1^T \end{bmatrix}$$

$$= \begin{bmatrix} D_2^TD_1^T+B_2^TC_1^T & D_2^TB_1^T+B_2^TA_1^T \\ C_2^TD_1^T+A_2^TC_1^T & C_2^TB_1^T+A_2^TA_1^T \end{bmatrix}. \qquad (4C.2)$$

It follows that, Eq. 4C.1 and Eq. 4C.2 are equal, and hence $(\hat{A}\ \hat{B})^S = \hat{B}^S\ \hat{A}^S$.

Similarly,

$$(\hat{A} + \hat{B})^S = \left[\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} + \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix}\right]^S = \begin{bmatrix} A_1+A_2 & B_1+B_2 \\ C_1+C_2 & D_1+D_2 \end{bmatrix}^S$$

$$= \begin{bmatrix} (D_1+D_2)^T & (B_1+B_2)^T \\ (C_1+C_2)^T & (A_1+A_2)^T \end{bmatrix} \qquad (4C.3)$$

and

$$\hat{A}^S + \hat{B}^S = \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix}^S + \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix}^S = \begin{bmatrix} D_1^T & B_1^T \\ C_1^T & A_1^T \end{bmatrix} + \begin{bmatrix} D_2^T & B_2^T \\ C_2^T & A_2^T \end{bmatrix}$$

$$= \begin{bmatrix} D_1^T+D_2^T & B_1^T+B_2^T \\ C_1^T+C_2^T & A_1^T+A_2^T \end{bmatrix}. \qquad (4C.4)$$

Eq. 4C.3 and Eq. 4C.4 are equal, therefore, the required result has been proved. Hence $(\hat{A} + \hat{B})^S = \hat{A}^S + \hat{B}^S$.

APPENDIX 4D -

Differentiation of an Orthogonal Matrix

i.e. $\frac{d'}{dt} E = \omega \times E$

where E is an *orthogonal* matrix (a rotational matrix) and ω is a *skew matrix.*

Proof :

A rotation about the origin O is given by

$$\underline{P} = E\ \underline{p} \qquad (4D.1)$$

where E is an orthogonal matrix and $\underline{P}$ and $\underline{p}$ are the position vectors, represented by column matrices, of a point P in the fixed space and the moving space respectively.

From this, it follows that the vector of a point P of the moving space is

$$\dot{\underline{P}} = \dot{E}\ \underline{p}. \qquad (4D.2)$$

Using Eq. 4D.1 to eliminate $\underline{p}$, $\dot{\underline{P}}$ can be expressed in terms of $\underline{P}$ as,

$$\dot{\underline{P}} = \dot{E}\ E^{-1}\ \underline{P}. \qquad (4D.3)$$

The matrix $\dot{E}\ E^{-1}$ which appears in Eq. 4D.3 has an important property.

If we differentiate, $E\ E^T = I$, we obtain

$$\dot{E}\ E^T + E\ (\dot{E}^T) = 0. \qquad (4D.4)$$

In view of the general relation

$$(M_1\ M_2)^T = M_2^T\ M_1^T,$$

the obvious property

$$(\dot{M}^T) = (\dot{M})^T$$

and the fact that E is an orthogonal matrix, i.e. $E^{-1} = E^T$, the second term of Eq. 4D.4 is equal to

$$E\,(\dot{E}^T) = (E^{-1})^T\,(\dot{E})^T = (\dot{E}\,E^{-1})^T.$$

Now, since the first term of Eq. 4D.4 is $\dot{E}\,E^{-1}$ we conclude that this product is a skew matrix.

We denote it by ω, which means that Eq. 4D.3 may be written as

$$\dot{\underline{P}} = \omega\,\underline{P} \qquad (4D.5)$$

where the skew matrix ω is defined by

$$\omega = \dot{E}\,E^{-1}. \qquad (4D.6)$$

This matrix ω is called the angular velocity matrix.

Now, for three-dimensional space (n = 3) the number of independent elements of the skew matrix ω is equal to three and, if we write it as follows

$$\omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

we find a correspondence between ω and the vector with Cartesian components $(\omega_x, \omega_y, \omega_z)$ which we shall denote by $\underline{\omega}$.

If (p_x, p_y, p_z) are the elements of the column vector $\underline{p}$ we have

$$\omega\,\underline{p} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

$$= \begin{bmatrix} \omega_y p_z - \omega_z p_y \\ \omega_z p_x - \omega_x p_z \\ \omega_x p_y - \omega_y p_x \end{bmatrix}.$$

Therefore, we have *an equivalence* between the matrix multiplication and the vector product :

$$\omega \underline{p} = \omega \times \underline{p} \qquad (4D.7)$$

which makes it possible to treat spatial quantities by means of vector algebra.

Thus, from Eq. 4D.6, we have

$$\dot{E} = \omega E$$

and using Eq. 4D.7, we obtain,

$$\dot{E} = \omega \times E \qquad (4D.8)$$

i.e. $\frac{d'}{dt} E = \omega \times E$

which is the required result.

APPENDIX 4E

Properties of the Cross Operator, ×

The linear cross operator is defined by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

and has the following *properties* :

1. $k(\underline{a}\times) = (k\underline{a})\times$ (k is a scalar)

2. $\underline{a}\times + \underline{b}\times = (\underline{a}+\underline{b})\times$

3. $(\underline{a}\times)^T = -\underline{a}\times$

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

5. $(E\underline{a})\times = E\underline{a}\times E^{-1}$ (E is a 3x3 orthogonal matrix)

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

Let us now try to prove the above properties.

For simplicity, let

$$\underline{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad \text{and} \quad \underline{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

Therefore,

$$\underline{a}\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \quad \text{and} \quad \underline{b}\times = \begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix}.$$

1. $k(\underline{a}\times) = (k\underline{a})\times$ (k is a scalar)

$$k(\underline{a}\times) = k\begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -a_3k & a_2k \\ a_3k & 0 & -a_1k \\ -a_2k & a_1k & 0 \end{bmatrix}$$

$$= (k\underline{a})\times.$$

2. $\underline{a}\times + \underline{b}\times = (\underline{a} + \underline{b})\times$

$$\underline{a}\times + \underline{b}\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -(a_3+b_3) & a_2+b_2 \\ a_3+b_3 & 0 & -(a_1+b_1) \\ -(a_2+b_2) & a_1+b_1 & 0 \end{bmatrix}$$

$$= (\underline{a} + \underline{b})\times.$$

3. $(\underline{a}\times)^T = -\underline{a}\times$

Trivial for Antisymmetry.

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

$$(\underline{a}\times)\underline{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} -a_3b_2 + a_2b_3 \\ a_3b_1 - a_1b_3 \\ -a_2b_1 + a_1b_2 \end{bmatrix} \qquad (4E.1)$$

and $$-(\underline{b}\times)\underline{a} = -\begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = -\begin{bmatrix} -a_2b_3 + b_2a_3 \\ b_3a_1 - b_1a_3 \\ -b_2a_1 + b_1a_2 \end{bmatrix}. \qquad (4E.2)$$

Eq. 4E.1 and Eq. 4E.2 are identical, therefore, the property has been proved.

5. $(E\underline{a})\times = E\underline{a}\times E^{-1}$ (E is a 3x3 orthogonal matrix)

Since E is an orthogonal matrix, $E\,E^T = I$, and therefore,

$$E^{-1} = E^T \quad \text{and} \quad |E| = 1.$$

Also, $$E^T = E^{-1} = \frac{adj(E)}{|E|} = adj(E)$$

which implies that,

$$E = [adj(E)]^T = cof(E). \tag{4E.3}$$

In particular, $$e_{ij} = cof(e_{ij}). \tag{4E.4}$$

Considering that, $$\underline{a}\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

if $$\varepsilon_{ijk} = \begin{cases} 0 & \text{if } i=j \text{ or } j=k \text{ or } k=i \\ 1 & \text{if } i,j,k \text{ are cyclic (e.g. } \varepsilon_{123} = 1) \\ -1 & \text{if } i,j,k \text{ are non-cyclic (e.g. } \varepsilon_{132} = -1) \end{cases}$$

then $$(\underline{a}\times)_{ij} = -\sum_{k=1}^{3} \varepsilon_{ijk}\, a_k \tag{4E.5}$$

(Example : $(\underline{a}\times)_{13} = -[\varepsilon_{131}\, a_1 + \varepsilon_{132}\, a_2 + \varepsilon_{133}\, a_3] = +a_2$).

Now, $$cof(e_{ij}) = (e_{i+1,j+1}\, e_{i+2,j+2} - e_{i+1,j+2}\, e_{i+2,j+1})$$

where the subscripts of e are subject to modular arithmetic.

Consider, $$T = \sum_{j=1}^{3}\sum_{k=1}^{3} \varepsilon_{jkl}\, e_{ij}\, e_{mk}.$$

We see that j=l, k=l, j=k do not contribute to the summation, leaving only 2 terms $e_{jkl} = 0$ if j=k, k=l, or j=l.

i.e. $T = e_{1,l+1}\, e_{m,l+2} - e_{1,l+2}\, e_{m,l+1}$.

Now, if i = m, $T = 0$

if i = m+1, $T = e_{m+1,l+1}\, e_{m,l+2} - e_{m+1,l+2}\, e_{m,l+1}$

if i = m-1, $T = e_{1,l+1}\, e_{l+1,l+2} - e_{1,l+2}\, e_{l+1,l+1}$

i.e. if i = m+1 then $T = -\text{cof}(e_{m-1,l})$

if i = m-1 then $T = +\text{cof}(e_{l-1,l})$.

Therefore, $T = + \sum_{n=1}^{3} \varepsilon_{lmn}\, \text{cof}(e_{nl})$

since, if i = m+1, $\varepsilon_{lmn} \neq 0$ only if n = m-1 and $\varepsilon_{m+1,m,m-1} = -1$

and if i = m-1, $\varepsilon_{lmn} \neq 0$ only if n = i-1 and $\varepsilon_{l,l+1,l-1} = +1$.

Using Eq. 4E.4, it follows that,

$$\sum_{j=1}^{3} \sum_{k=1}^{3} \varepsilon_{jkl}\, e_{lj}\, e_{mk} = \sum_{n=1}^{3} \varepsilon_{lmn}\, e_{nl}. \qquad (4E.6)$$

Now, $(E\underline{a})_r = \sum_{l=1}^{3} e_{rl}\, a_l = f_r$

and therefore, $[(E\underline{a})\times]_{lm} = - \sum_{n=1}^{3} \varepsilon_{lmn}\, f_n$ (from Eq. 4E.5)

$$= -\sum_{n=1}^{3}\sum_{l=1}^{3} \varepsilon_{lmn}\, e_{nl}\, a_l \qquad (4E.7)$$

also, $[(\underline{a}\times)E^T]_{jm} = \sum_{k=1}^{3} (\underline{a}\times)_{jk}\, e_{mk}$

$$= -\sum_{k=1}^{3}\sum_{l=1}^{3} \varepsilon_{jkl}\, a_l\, e_{mk} \qquad \text{(from Eq. 4E.5)}$$

That is,

$$[E(\underline{a}\times)E^T]_{lm} = \sum_{j=1}^{3} e_{lj}\, [(a\times)E^T]_{jm}$$

$$= -\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3} e_{lj}\, \varepsilon_{jkl}\, a_l\, e_{mk}$$

$$= -\sum_{l=1}^{3} a_l \left(\sum_{j=1}^{3}\sum_{k=1}^{3} \varepsilon_{lmn}\, e_{nl}\right)$$

$$= \sum_{l=1}^{3} a_l \left(\sum_{n=1}^{3} \varepsilon_{lmn}\, e_{nl}\right) \qquad \text{(from Eq. 4E.6)}$$

$$= [(E\underline{a})\times]_{lm} \qquad \text{(from Eq. 4E.7)}$$

and therefore,

$$E(\underline{a}\times)E^T = (E\underline{a})\times.$$

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

Now $$(\underline{a}\times\underline{b})\times = \begin{bmatrix} -a_3b_2 + a_2b_3 \\ a_3b_1 - a_1b_3 \\ -a_2b_1 + a_1b_2 \end{bmatrix} \times$$

$$= \begin{bmatrix} 0 & a_2b_1-a_1b_2 & a_3b_1-a_1b_3 \\ -a_2b_1+a_1b_2 & 0 & a_3b_2-a_2b_3 \\ -a_3b_1+a_1b_3 & -a_3b_2+a_2b_3 & 0 \end{bmatrix}$$

and $$\underline{a}\times\underline{b}\times - \underline{b}\times\underline{a}\times = (\underline{a}\times)(\underline{b}\times) - (\underline{b}\times)(\underline{a}\times)$$

$$= \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_3b_3-a_2b_2 & a_2b_1 & a_3b_1 \\ a_1b_2 & -a_3b_3-a_1b_1 & a_3b_2 \\ a_1b_3 & a_2b_3 & -a_2b_2-a_1b_1 \end{bmatrix} -$$

$$\begin{bmatrix} -a_3b_3-a_2b_2 & b_2a_1 & b_3a_1 \\ b_1a_2 & -a_3b_3-a_1b_1 & a_2b_3 \\ a_3b_1 & a_3b_2 & -a_2b_2-a_1b_1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & a_2b_1-b_2a_1 & a_3b_1-b_3a_1 \\ a_1b_2-b_1a_2 & 0 & a_3b_2-a_2b_3 \\ a_1b_3-a_3b_1 & a_2b_3-a_3b_2 & 0 \end{bmatrix}$$

which demonstrates property 6, and therefore, completes the proofs of the properties of the cross operator.

CHAPTER 5
DYNAMIC ASPECTS OF SPATIAL ALGEBRA

5.1 INTRODUCTION

The previous chapter studied the fundamental aspects of spatial quantities and dealt with the kinematics of spatial algebra. This chapter investigates the use of this spatial notation in the dynamic analysis - principally the concept of inertia.

In the next sections, the spatial momentum of a rigid body is defined as the mapping between velocity and momentum, and its representation as a 6x6 matrix is deduced.

Additionally, the equations of motion for a rigid body are given and the concepts of inverse inertias and articulated-body inertias are introduced, describing their principal properties and uses.

Having introduced the principal ideas of momentum and spatial inertias, as well as the concept of articulated-body inertias, we will go further on to deal with the calculation of robot dynamics, a direct dynamics method for articulated-body inertias, originally suggested by Featherstone for use in robotics.

The main points to be discussed here, are the degrees of freedom of motion and branched chains, the rigid-body inertia matrices and the transformation matrices, so that the development of the articulated-body dynamics algorithm will follow.

Necessary extensions of the algorithm are introduced in the next chapter, so that modifications of the method could be made, for example, to allow for multiple-degree-of-freedom joints and branched kinematic chains, which are topics that are discussed in Chapter 6.

5.2 SPATIAL RIGID-BODY MOMENTUM AND SPATIAL RIGID-BODY INERTIA

5.2.1 Introduction

When an object is moving it is said to have an amount of momentum given, by definition, by

momentum = mass . velocity.

Newton defines the force acting on an object as *the rate of change of its momentum*, the momentum being the product of its mass and velocity. Momentum is thus a vector quantity. Its direction is that of the velocity.

Momentum can be classified as being linear or angular. In linear or straight-line motion, an important property of the moving object is its *linear momentum.* When the body spins, or rotates about an axis, its *angular momentum* plays an important part in its motion.

Newton's first law, *'every body continues in its state of rest or uniform motion in a straight line, unless it is acted by a resultant force'*, expresses the idea of inertia.

The inertia of the body is its reluctance to start moving, or to stop once it has begun moving. Thus an object at rest begins to move only when a force acts on it.

Mass is a measure of the inertia of a body. If the direction of an object is changed, or its velocity is changed slightly when a force acts on it, its inertial mass is high.

5.2.2 Definition of Spatial Momentum

Consider a rigid body having

- mass m,
- centre of mass at a point P,
- rotational inertia, $\underline{I}^*$, about the centre of mass,
- spatial velocity, $\hat{\underline{v}}_0 = [\omega^T \ \underline{v}_0^T]^T$,
- linear velocity of the centre of mass $\hat{\underline{v}}_P = \hat{\underline{v}}_0 + \overrightarrow{PO} \times \omega$.

The linear momentum of the rigid body is to be defined as the product of the mass and the velocity of the body. Newton's second law of motion states that *'the rate of change of momentum is proportional to the impressed force and takes place in the direction of the straight line along which the force acts'*.

That is, a force acts on the rigid body which is equal to the change in momentum per second. $\underline{P} = m\,\underline{v}_P$ and the line of action is through the centre of mass of the body.

Now, using the spatial notation, we get

$$\hat{\underline{P}}_0 = [\ \underline{P}^T\ (\overrightarrow{OP} \times \underline{P})^T\ ]^T$$

where $\overrightarrow{OP} \times \underline{P}$ is the vector representing the body's moment of momentum about the origin.

The body's angular momentum is given by $\underline{I}^* \omega$, which behaves like a free vector and can thus be represented as a spatial vector

$$\hat{\underline{P}}^* = [\ \underline{0}^T\ (\underline{I}^*\ \omega)^T\ ]^T.$$

The total momentum is the sum of its linear and angular components and is therefore given by

$$\hat{\underline{P}} = \begin{bmatrix} m\underline{v}_P \\ \overrightarrow{OP} \times m\underline{v}_P \end{bmatrix} + \begin{bmatrix} \underline{0} \\ \underline{I}^*\ \omega \end{bmatrix}$$

$$= \begin{bmatrix} m(\underline{v}_0 + \overrightarrow{PO} \times \omega) \\ \underline{I}^*\omega + \overrightarrow{OP} \times m\ (\underline{v}_0 + \overrightarrow{PO} \times \omega) \end{bmatrix}. \qquad (5.1)$$

5.2.3 Definition of Spatial Inertia

The spatial rigid-body inertia is defined as the mapping between the spatial velocity of a rigid body and its spatial momentum.

Now, $\hat{\underline{P}} = \hat{\underline{I}}\,\hat{\underline{v}}$

where $\hat{\underline{I}}$ is the 6x6 spatial inertia.

From the equation for momentum Eq. 5.1, we get,

$$\hat{\underline{P}} = \begin{bmatrix} m\,\overrightarrow{PO} \times \omega + m\,\underline{v}_0 \\ (\underline{I}^* + \overrightarrow{OP} \times m\,\overrightarrow{PO} \times)\omega + \overrightarrow{OP} \times m\,\underline{v}_0 \end{bmatrix}$$

$$= \begin{bmatrix} m\,\overrightarrow{PO} \times & m\,\underline{1} \\ \underline{I}^* + \overrightarrow{OP} \times m\,\overrightarrow{PO} \times & \overrightarrow{OP} \times m \end{bmatrix} \hat{\underline{v}}_0$$

$$= \hat{\underline{I}}_0\,\hat{\underline{v}}_0.$$

So, the spatial inertia is given above by the 6x6 matrix, having 3x3 matrices as components

$$\hat{\underline{I}}_0 = \begin{bmatrix} m\,\overrightarrow{PO} \times & m\,\underline{1} \\ \underline{I}^* + \overrightarrow{OP} \times m\,\overrightarrow{PO} \times & \overrightarrow{OP} \times m \end{bmatrix}$$

or, simplifying it we get,

$$\hat{\underline{I}}_0 = \begin{bmatrix} \underline{H}^T & \underline{M} \\ \underline{I} & \underline{H} \end{bmatrix}$$

where,

$\underline{M} = m\,\underline{1}$ - the zeroth moment of mass about the origin,
$\underline{H} = \overrightarrow{OP} \times m$ - the first moment of mass about the origin,
$\underline{I} = \underline{I}^* + \overrightarrow{OP} \times m\,\overrightarrow{PO} \times$ - the second moment of inertia or inertia tensor; and
$\underline{I}^*$ is the rotational inertia about the centre of mass.

If $\overrightarrow{OP} = \underline{0}$, i.e. if the coordinate origin coincides with the centre of mass of the rigid body, then

$$\hat{\underline{I}}_0 = \begin{bmatrix} \underline{0} & \underline{M} \\ \underline{I}^* & \underline{0} \end{bmatrix}$$

which gives

$$\hat{\underline{P}} = \begin{bmatrix} \underline{0} & \underline{Mv}_0 \\ \underline{I}^*\omega & \underline{0} \end{bmatrix}.$$

5.2.4 Properties of the Spatial (Rigid-Body) Inertias

1) The spatial inertia matrix, (unlike rotational inertia), is neither symmetric nor positive definite in the conventional sense. However, the use of the spatial transpose operator makes the spatial inertia matrix symmetric and positive definite with respect to the spatial operator. That is,

$$\hat{\underline{I}} = \hat{\underline{I}}^S \qquad \text{and} \qquad \hat{\underline{x}}^S \hat{\underline{I}}\, \hat{\underline{x}} > 0 \qquad \text{for all} \qquad \hat{\underline{x}} > \hat{\underline{0}}.$$

2) It is always non-singular; otherwise it would have been possible for the rigid-body to have zero momentum with non-zero velocity.

3) Spatial inertias can be transformed from one coordinate system to another using the definition of a linear mapping of spatial vectors,

i.e. if $\hat{\underline{P}}_P$, $\hat{\underline{I}}_P$ and $\hat{\underline{v}}_P$ are represented in P coordinates

and $\hat{\underline{P}}_0$, $\hat{\underline{I}}_0$ and $\hat{\underline{v}}_0$ are represented in O coordinates,

then $\hat{\underline{P}}_P = \hat{\underline{I}}_P\, \hat{\underline{v}}_P$ and $\hat{\underline{P}}_0 = \hat{\underline{I}}_0\, \hat{\underline{v}}_0$.

Now, $\hat{\underline{P}}_P = {}_P\hat{X}_0\, \hat{\underline{P}}_0$ and $\hat{\underline{v}}_0 = {}_0\hat{X}_P\, \hat{\underline{v}}_P$

so $\hat{\underline{I}}_P\, \hat{\underline{v}}_P = \hat{\underline{P}}_P = {}_P\hat{X}_0\, \hat{\underline{P}}_0 = {}_P\hat{X}_0\, \hat{\underline{I}}_0\, \hat{\underline{v}}_0 = {}_P\hat{X}_0\, \hat{\underline{I}}_0\, {}_0\hat{X}_P\, \hat{\underline{v}}_P$

and therefore

$$\hat{\underline{I}}_P = {}_P\hat{X}_0 \, \hat{\underline{I}}_0 \, {}_0\hat{X}_P .$$

4) (a) The derivative of a spatial inertia with respect to time exists and is just the component-wise derivative (in a stationary coordinate frame).

(b) The derivative in a moving coordinate frame can be obtained using the transformation approach.

Thus, if P is a coordinate frame moving with velocity $\hat{\underline{v}}$, and O a stationary one, then

$$\frac{d}{dt} \hat{\underline{I}}_P = {}_P\hat{X}_0 \, (\frac{d'}{dt} \hat{\underline{I}}_0) \, {}_0\hat{X}_P$$

But
$$\frac{d'}{dt} \hat{\underline{I}}_0 = \frac{d'}{dt} ({}_0\hat{X}_P \, \hat{\underline{I}}_P \, {}_P\hat{X}_0)$$

$$= \frac{d'}{dt} ({}_0\hat{X}_P) \, \hat{\underline{I}}_P \, {}_P\hat{X}_0 + {}_0\hat{X}_P \, \frac{d'}{dt} (\hat{\underline{I}}_P) \, {}_P\hat{X}_0 + {}_0\hat{X}_P \, \hat{\underline{I}}_P \, \frac{d'}{dt} ({}_P\hat{X}_0)$$

$$= {}_0\hat{X}_P \, \hat{\underline{v}}_P \hat{\times} \, \hat{\underline{I}}_P \, {}_P\hat{X}_0 + {}_0\hat{X}_P \, \frac{d'}{dt} \hat{\underline{I}}_P \, {}_P\hat{X}_0 - {}_0\hat{X}_P \, \hat{\underline{I}}_P \, \hat{\underline{v}}_P \hat{\times} \, {}_P\hat{X}_0,$$

using that,

$$\frac{d'}{dt} \, {}_0\hat{X}_P = \hat{\underline{v}}_0 \hat{\times} \, {}_0\hat{X}_P$$

and

$$(E \, \underline{a}) \times = E \, \underline{a} \times E^{-1} \qquad \text{(See Appendix 4E - property 5).}$$

Therefore,

$$\frac{d}{dt} \hat{\underline{I}}_P = \frac{d'}{dt} \hat{\underline{I}}_P + \hat{\underline{v}}_P \hat{\times} \, \hat{\underline{I}}_P - \hat{\underline{I}}_P \, \hat{\underline{v}}_P \hat{\times} .$$

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

$$\frac{d}{dt}\hat{\underline{I}} = \hat{\underline{v}} \times \hat{\underline{I}} - \hat{\underline{I}}\,\hat{\underline{v}} \times.$$

5) The inertia of a composite rigid body is the sum of the inertias of its components. That is,

$$\hat{\underline{I}} = \sum_i \underline{I}_i.$$

6) If $\hat{\underline{I}}$ is the rigid-body inertia, then $\hat{\underline{I}}^{-1}$ is the inverse inertia of the rigid body and relates momentum to velocity.

If $\hat{\underline{P}} = \hat{\underline{I}}\,\hat{\underline{v}}$ then $\hat{\underline{v}} = \hat{\underline{I}}^{-1}\,\hat{\underline{P}}$.

Inverse inertias obey the same rules for transformation and differentiation as do inertias, but their composition to form inverse inertia of a composite sum is accomplished by the harmonic sum.

5.3 THE EQUATIONS OF MOTION

(a) If there is *no force* applied to the body then from Newton's second law we find that

$$\hat{\underline{I}}\,\hat{\underline{v}} = \text{constant},$$

i.e. the combination of linear and angular momenta will remain unchanged. This is due to the principle of conservation of linear momentum which states that, *'If no external forces act on a system of colliding objects, the total momentum of the objects in a given direction remains constant'.*

Further, the conservation of angular momentum, which corresponds to the conservation of linear momentum, states that *'the angular momentum about an axis of a given rotating body or system of bodies is constant if no external torque acts about that axis'.*

(b) If a force is applied, then the rate of change of momentum is equal to the net applied force,

$$\hat{\underline{f}} = \frac{d}{dt}(\hat{\underline{I}}\,\hat{\underline{v}})$$

$$= \hat{\underline{I}}\,\hat{\underline{a}} + \hat{\underline{v}} \times \hat{\underline{I}}\,\hat{\underline{v}}$$

where $\hat{\underline{a}}$ is the spatial acceleration of the rigid body and $\hat{\times}$ is the spatial cross operator.

$\hat{\underline{v}} \times \hat{\underline{I}}$ is called the apparent derivative of $\hat{\underline{I}}$, Bottema & Roth [20]. This is because the absolute derivative of a spatial 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.

Therefore, if a rigid body is moving with absolute velocity $\hat{\underline{v}}$ and inertia $\hat{\underline{I}}$ then the absolute derivative of $\hat{\underline{I}}$ is

$$\frac{d}{dt}\hat{\underline{I}} = \hat{\underline{v}} \times \hat{\underline{I}} - \hat{\underline{I}}\,\hat{\underline{v}} \times.$$

Taking the product of the derivative of $\hat{\underline{I}}$ with the velocity $\hat{\underline{v}}$ results in making the second term, of the right hand side of the above equation, zero.

Let $\hat{\underline{p}} = \hat{\underline{v}} \times \hat{\underline{I}}\,\hat{\underline{v}}$

then $\hat{\underline{f}} = \hat{\underline{I}}\,\hat{\underline{a}} + \hat{\underline{p}}$

$\hat{\underline{p}}$ is called the *bias force* and is the force that must be applied to the rigid body to produce zero spatial acceleration.

5.4 ARTICULATED-BODY INERTIAS

An articulated body is a collection of rigid bodies connected by joints which restrict the free relative motion of the member elements.

An articulated-body inertia is the relationship between a spatial force applied to a particular member of an articulated body and the spatial acceleration of that member, taking into account the effect of the rest of the articulated body.

The relationship is linear and may be expressed in the inhomogeneous form

$$\hat{\underline{f}} = \hat{\underline{I}}^A \hat{\underline{a}} + \hat{\underline{p}}$$

where, $\hat{\underline{I}}^A$ is the articulated-body inertia of the member,

$\hat{\underline{f}}$ is the applied-test force, and

$\hat{\underline{p}}$ is the associated bias force, which is the force to be applied to that member of the articulated body to give it zero acceleration.

If $\hat{\underline{I}}^A$ and $\hat{\underline{p}}$ are known for any particular member of an active articulated mechanism, then the acceleration of that member may be calculated.

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 each were a single rigid-body-like element of the system and permits the simplification of a rigid-body system by reducing the apparent number of elements in the system.

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.

5.4.1 Properties of Articulated-Body Inertias

Articulated-body inertias defining the relationship between a force and an acceleration, are tensors and obey the same rules for transformation of representation and differentiation in moving coordinates as do rigid-body inertias.

Articulated-body inertias are symmetric, positively definite matrices (with respect to the spatial transpose operator). Being symmetric implies that there is no such thing as a centre of mass for an articulated body and that the apparent mass has directional properties analogous to those of rotational inertia.

To calculate articulated-body inertias is very difficult in the general case. However, if the connectivity of the articulated body is such that there are no kinematic loops and no connections to the ground then the calculation

becomes relatively easy. The only inertia that is always guaranteed to exist is the inverse inertia, [19].

5.5 THE CALCULATION OF ROBOT DYNAMICS USING ARTICULATED-BODY INERTIAS

5.5.1 Introduction

Our target in this section is the development of a linearly computational, direct dynamics method, for the calculation of accelerations of a robot in response to given actuator forces. This is based on *recursive formulae*, involving the use of *articulated-body inertias*.

As already stated, articulated bodies are made out of rigid-bodies linked together. It is thus obvious, that the dynamics of such bodies is more complex than that of simple rigid bodies, owing to this segmentational interaction. Articulated bodies require the formulation of a common mathematical basis for advanced modelling as well as a compact representation and control method. It follows that the spatial notation, originally introduced in Chapter 4, can be used here, since using matrices, constraints can be put together with the constraint equations.

In the next sections the concepts of degrees of freedom of motion and branched chains are introduced. The form of articulated-body inertia matrices and transformation matrices are discussed, so that the articulated-body dynamics algorithm is finally deduced.

5.5.2 Degrees of Freedom of Motion and Branched Chains

Typically motion is described in terms of *degrees of freedom*, i.e. the direction along which a joint can move, and these can be translational or rotational. A translational joint is one that allows linear motion, and a rotational joint is one that allows angular motion.

It has already been discussed in Chapter 2 that *a particle* with coordinates (x, y, z) has three degrees of freedom of motion. This is because the location and motion of such a point is designated by the three variables x, y and z.

Rigid bodies are defined by an infinite number of points that must move together. They may not move relative to each other, though they may move as a whole relative to the world space. The motion of a rigid body is specified by six degrees of freedom, three translational and three rotational.

Essentially a human body is represented by segments consisting of rigid extended masses connected by three-degrees-of-freedom joints. If we first consider the body to have a fixed immobile base, then all the joints, as well as the root, will have three rotational degrees of freedom.

Articulated bodies are rigid bodies whose motion relative to each other is somewhat restricted. The number of dynamics equations necessary to specify a system depends upon the number of degrees of freedom in the system. For an articulated body, the number of degrees of freedom is the sum of the degrees of freedom at each joint plus, the number of degrees of freedom connecting the body to the world (three translational degrees of freedom for a mobile root).

The body segments are described as a tree structure branching from the fixed, massless world segment. Even a simple representation of the human body model has branched trees. Therefore, we need a way of being able to define multiple-degrees-of-freedom joints branching from the kinematic chains (open loop kinematic trees). This could be achieved by treating the multiple-degree-of-freedom joints, e.g. shoulders, as an appropriate sequence of one-degree-of-freedom joints joined by massless, dimensionless segments, Wilhelms [2]. Section 6.2.1 of this thesis deals with this concept.

In a *branched kinematic chain* the manner in which the links are connected corresponds to a topological tree. Basically, this means that there are no kinematic loops and that no part of the human model is entirely disconnected from the rest. The base link is chosen as the root of the tree, and is considered to be immobile. The outermost links are its leaves.

Neighbouring links are two links connected by a joint and a neighbour of a given link will be called its *predecessor* or its *successor* depending on whether it is nearer to or further away from the base link. *Inner joint* is the joint connecting a link to its predecessor and *outer joint* a joint connecting a link to its successor. Thus, every link except from 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 is that for the tree structures we need to specify the connectivity explicitly whereas for the unbranched chain the connectivity is implicitly in the numbering scheme. This is further considered in Section 6.2.2.

5.5.3 Spatial Rigid-Body Inertia Matrices

The spatial inertia of a rigid body, referred to its centre of mass is given by :

$$\begin{bmatrix} \underline{0} & \underline{M} \\ \underline{I}^{*} & \underline{0} \end{bmatrix}$$

a 6x6 matrix.

$\underline{M}$ is the 3x3 diagonal mass matrix - $\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}$ where m is a scalar.

$\underline{I}^{*}$ is the 3x3 inertia tensor matrix - $\begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$,

Appendix 5A gives numerical calculations for I_{xx}, I_{yy}, I_{zz}, the moments of inertia, and I_{xy}, I_{yz}, I_{xz}, the products of inertia.

Therefore, the spatial inertia of a rigid body referred to its centre of mass is the 6x6 matrix,

$$\hat{\underline{I}} = \begin{bmatrix} 0 & 0 & 0 & m & 0 & 0 \\ 0 & 0 & 0 & 0 & m & 0 \\ 0 & 0 & 0 & 0 & 0 & m \\ I_{xx} & I_{xy} & I_{xz} & 0 & 0 & 0 \\ I_{xy} & I_{yy} & I_{yz} & 0 & 0 & 0 \\ I_{xz} & I_{yz} & I_{zz} & 0 & 0 & 0 \end{bmatrix}.$$

The spatial inertia matrix of a rigid body, $\hat{\underline{I}}$, referred to the origin, is the relationship between the force and acceleration referred to the origin and it is thus given by,

$$\hat{\underline{I}} = \begin{bmatrix} \underline{H}^T & \underline{M} \\ \underline{I} & \underline{H} \end{bmatrix}$$

where, $\underline{H}$ is $\underline{c}x\underline{M}$ and $\underline{c}$ is the position vector,

and $\underline{I}$, the inertia tensor about the origin of the rigid body, is equal to $\underline{I}^* + \underline{c}x\underline{M}\ \underline{c}x^T$.

Now, $\underline{c}x\underline{M}\ \underline{c}x^T$ is the inertia tensor of a point mass, and

$\underline{c}x$, is the anti-symmetric 3x3 matrix corresponding to $\underline{c}$.

That is, $$\underline{c} = \begin{bmatrix} 0 & -c_3 & c_2 \\ c_3 & 0 & -c_1 \\ -c_2 & c_1 & 0 \end{bmatrix}.$$

Therefore,

$$\hat{\underline{I}} = \begin{bmatrix} 0 & c_3m & -c_2m & m & 0 & 0 \\ -c_3m & 0 & c_1m & 0 & m & 0 \\ c_2m & -c_1m & 0 & 0 & 0 & m \\ I_{xx}+c_3^2m+c_2^2m & I_{xy}-c_1c_2m & I_{xz}-c_1c_3m & 0 & -c_3m & c_2m \\ I_{xy}-c_1c_2m & I_{yy}+c_3^2m+c_1^2m & I_{zy}-c_2c_3m & c_3m & 0 & -c_1m \\ I_{xz}-c_1c_3m & I_{yz}-c_2c_3m & I_{zz}+c_2^2m+c_1^2m & -c_2m & c_1m & 0 \end{bmatrix}.$$

5.5.4 Initial development of the Direct Dynamics Algorithm

The method, as originally developed by Featherstone [16], considers n links, connected by n single-degree-of-freedom joints forming a branch-free kinematic chain. It is a direct dynamics method and the objective is to calculate the values of $\ddot{q}_i$, the accelerations of all links i, given the values of $\hat{I}_i$, $\hat{s}_i$ and Q_i. That is, each link is characterised by a spatial rigid-body inertia $\hat{I}_i$; and each joint is characterised by an axis $\hat{s}_i$; Q_i, is the active force applied about the axis of joint i and $\ddot{q}_i$, is the acceleration of link i relative to link (i-1) about a joint axis. The following is a list of the notation used by the algorithm.

$\hat{I}_i$ - is the spatial rigid-body inertia of link i, a 6x6 matrix,

$\hat{s}_i$ - the axis of joint i (unit line vector for revolute joints) - 6x1,

Q_i - the active (scalar) force applied about the axis of joint i, and

$\ddot{q}_i$ - the (scalar) acceleration of link i relative to link i-1.

In the equation of the inhomogeneous inertias, $\hat{f} = \hat{I}_i^A \hat{a}_i + \hat{p}_i$,

$\hat{p}_i$ - is the bias force - of the form 6x1 matrix,

$\hat{a}_i$ - the absolute acceleration of link i,

$\hat{f}$ - the spatial force applied to link i - 6x1, and

$\hat{I}_i^A$ - the articulated-body inertia of link i - 6x6.

Also, from the following equations,

$\hat{p}_i^v$ - is the bias force due to velocity-product effects - 6x1 matrix,

$\hat{v}_i$ - the absolute velocity of link i, and

$\dot{q}_i$ - the relative velocity of link i.

Note : The acceleration is non-zero since there are active (externally applied) forces within the linkage.

Let $\hat{\underline{f}}$ be a spatial force applied to link i, producing an acceleration $\hat{\underline{a}}_1$. They are related by

$$\hat{\underline{f}} = \hat{\underline{I}}_1^A \hat{\underline{a}}_1 + \hat{\underline{p}}_1 \tag{5.2}$$

$\hat{\underline{f}}$ can be considered to be made up of two components, $\hat{\underline{f}}_1$ and $\hat{\underline{f}}_2$, where $\hat{\underline{f}}_1$ is the spatial force required to produce an acceleration of $\hat{\underline{a}}_1$ in link i alone, and $\hat{\underline{f}}_2$ is responsible for accelerating the rest of the links, assuming that no previous links, i.e. i-1, i-2, are connected to the structure. Therefore we have,

$$\hat{\underline{f}} = \hat{\underline{f}}_1 + \hat{\underline{f}}_2 \tag{5.3}$$

and

$$\hat{\underline{f}}_1 = \hat{\underline{I}}_1 \hat{\underline{a}}_1. \tag{5.4}$$

It can clearly be seen that $\hat{\underline{f}}_2$ is transmitted to link i+1 through joint i+1. $\hat{\underline{f}}_2$ produces an acceleration in link i+1 of $\hat{\underline{a}}_2$, and the relationship between $\hat{\underline{f}}_2$ and $\hat{\underline{a}}_2$ is given by

$$\hat{\underline{f}}_2 = \hat{\underline{I}}_{i+1}^A \hat{\underline{a}}_2 + \hat{\underline{p}}_{i+1}. \tag{5.5}$$

Also

$$\hat{\underline{a}}_2 = \hat{\underline{a}}_1 + \hat{\underline{s}}_{i+1} \alpha \tag{5.6}$$

(α is an unknown scalar),

since $\hat{\underline{a}}_2$ differs from $\hat{\underline{a}}_1$ by a component in the direction of the axis of joint i+1 only, as links i and i+1 are connected by joint i+1. Additionally, link i+1 has only one degree of freedom of motion relative to link i.

Now,

$$\hat{\underline{s}}_{i+1}^S \hat{\underline{f}}_2 = Q_{i+1} \tag{5.7}$$

where Q_{i+1} is the component of $\hat{\underline{f}}_2$ in the direction of $\hat{\underline{s}}_{i+1}$ since the joint bearing transmits no force in this direction, and ^{S} is the spatial transpose operator (see Appendix 5B). Substituting Eq. 5.6 into Eq. 5.5 we have

$$\hat{\underline{f}}_2 = \hat{\underline{I}}^A_{1+1} (\hat{\underline{a}}_1 + \hat{\underline{s}}_{1+1} \alpha) + \hat{\underline{p}}_{1+1} \tag{5.8}$$

and substituting this into Eq. 5.7 we get

$$\hat{\underline{s}}^S_{1+1} (\hat{\underline{I}}^A_{1+1} (\hat{\underline{a}}_1 + \hat{\underline{s}}_{1+1} \alpha) + \hat{\underline{p}}_{1+1}) = Q_{1+1} \tag{5.9}$$

from which we obtain the following expression for α,

$$\alpha = \frac{Q_{1+1} - \hat{\underline{s}}^S_{1+1} \hat{\underline{I}}^A_{1+1} \hat{\underline{a}}_1 - \hat{\underline{s}}^S_{1+1} \hat{\underline{p}}_{1+1}}{\hat{\underline{s}}^S_{1+1} \hat{\underline{I}}^A_{1+1} \hat{\underline{s}}_{1+1}}. \tag{5.10}$$

Substituting $\hat{\underline{f}}_1$ and $\hat{\underline{f}}_2$ in Eq. 5.3 using Eq 5.4 and Eq 5.8 we get

$$\hat{\underline{f}} = \hat{\underline{I}}_1 \hat{\underline{a}}_1 + \hat{\underline{I}}^A_{1+1} (\hat{\underline{a}}_1 + \hat{\underline{s}}_{1+1} \alpha) + \hat{\underline{p}}_{1+1}$$

and substituting for α using Eq. 5.10 gives

$$\hat{\underline{f}} = \hat{\underline{I}}_1 \hat{\underline{a}}_1 + \hat{\underline{I}}^A_{1+1} \left[\hat{\underline{a}}_1 + \hat{\underline{s}}_{1+1} \frac{Q_{1+1} - \hat{\underline{s}}^S_{1+1} \hat{\underline{I}}^A_{1+1} \hat{\underline{a}}_1 - \hat{\underline{s}}^S_{1+1} \hat{\underline{p}}_{1+1}}{\hat{\underline{s}}^S_{1+1} \hat{\underline{I}}^A_{1+1} \hat{\underline{s}}_{1+1}} \right] + \hat{\underline{p}}_{1+1}.$$

Rearranging this equation to collect terms in $\hat{\underline{a}}_1$ gives

$$\hat{\underline{f}} = \left[\hat{\underline{I}}_1 + \hat{\underline{I}}^A_{1+1} - \frac{\hat{\underline{I}}^A_{1+1} \hat{\underline{s}}_{1+1} \hat{\underline{s}}^S_{1+1} \hat{\underline{I}}^A_{1+1}}{\hat{\underline{s}}^S_{1+1} \hat{\underline{I}}^A_{1+1} \hat{\underline{s}}_{1+1}} \right] \hat{\underline{a}}_1 +$$

$$\left[\hat{\underline{p}}_{1+1} + \frac{\hat{\underline{I}}^A_{1+1} \hat{\underline{s}}_{1+1} (Q_{1+1} - \hat{\underline{s}}^S_{1+1} \hat{\underline{p}}_{1+1})}{\hat{\underline{s}}^S_{1+1} \hat{\underline{I}}^A_{1+1} \hat{\underline{s}}_{1+1}} \right]$$

which has the same form as Eq. 5.2, and, therefore, we can equate the expressions inside the brackets with $\hat{\underline{I}}^A_1$ and $\hat{\underline{p}}_1$, respectively.

Furthermore, $\hat{\underline{a}}_1$, the absolute acceleration of link i, is composed by summing joint accelerations and is therefore given by

$$\hat{\underline{a}}_1 = \hat{\underline{a}}_{1-1} + \hat{\underline{s}}_1 \ddot{q}_1 \qquad \text{(where } \hat{\underline{a}}_0 = 0) \tag{5.11}$$

using Eq. 5.2. Now, $\hat{\underline{f}}$ is transmitted through joint i in the direction of $\hat{\underline{s}}_i$ and therefore

$$\hat{\underline{s}}_i^S \hat{\underline{f}} = Q_i. \tag{5.12}$$

Thus, $\hat{\underline{s}}_i^S [\hat{\underline{I}}_i^A (\hat{\underline{a}}_{i-1} + \hat{\underline{s}}_i \ddot{q}_i) + \hat{\underline{p}}_i] = Q_i,$

from which, an expression for $\ddot{q}_i$ can be obtained :

$$\ddot{q}_i = \frac{Q_i - \hat{\underline{s}}_i^S \hat{\underline{I}}_i^A \hat{\underline{a}}_{i-1} - \hat{\underline{s}}_i^S \hat{\underline{p}}_i}{\hat{\underline{s}}_i^S \hat{\underline{I}}_i^A \hat{\underline{s}}_i}. \tag{5.13}$$

Following what has been discussed above, in the next section, the equations making up the algorithm are sequentially presented.

The Recursive algorithm for calculating $\ddot{q}_i$

In the following equations let some constants be defined so that the common sub-expressions $\hat{\underline{c}}_i$, $\hat{\underline{h}}_i$, d_i and u_i can be computed once and stored for later use.

1/ *The absolute velocity and the bias force due to velocity-product forces*

$$\hat{\underline{v}}_i = \hat{\underline{v}}_{i-1} + \hat{\underline{s}}_i \dot{q}_i \quad \text{with initial value } \hat{\underline{v}}_0 = \hat{\underline{0}}, \tag{5.14}$$

$\dot{q}$ is the relative velocity of link i,

and $$\hat{\underline{p}}_i^v = \hat{\underline{v}}_i \hat{\times} \hat{\underline{I}}_i \hat{\underline{v}}_i. \tag{5.15}$$

2/ *The common sub-expression*

$$\hat{\underline{c}}_i = \hat{\underline{v}}_i \hat{\times} \hat{\underline{s}}_i \dot{q}_i, \tag{5.16}$$

3/ Some constants, the articulated-body inertia and the bias force for values of i from n-1 down to 1 using the common sub-expressions

$$\hat{\underline{h}}_i = \hat{\underline{I}}_i^A \, \hat{\underline{s}}_i, \tag{5.17}$$

$$d_i = \hat{\underline{s}}_i^S \, \hat{\underline{h}}_i, \text{ and} \tag{5.18}$$

$$u_i = Q_i - \hat{\underline{h}}_i^S \, \hat{\underline{c}}_i - \hat{\underline{s}}_i^S \, \hat{\underline{p}}_i. \tag{5.19}$$

Initial value : $\hat{\underline{I}}_n^A = \hat{\underline{I}}_n$;

$$\hat{\underline{I}}_i^A = \hat{\underline{I}}_i + \hat{\underline{I}}_{i+1}^A - \frac{\hat{\underline{h}}_{i+1} \, \hat{\underline{h}}_{i+1}^S}{d_{i+1}} \tag{5.20}$$

and

$$\hat{\underline{p}}_i = \hat{\underline{p}}_i^v + \hat{\underline{p}}_{i+1} + \hat{\underline{I}}_{i+1}^A \, \hat{\underline{c}}_{i+1} + \frac{u_{i+1}}{d_{i+1}} \hat{\underline{h}}_{i+1} \tag{5.21}$$

with initial value, $\hat{\underline{p}}_n = \hat{\underline{p}}_n^v$.

4/ The acceleration of link i relative to link (i-1), and the absolute acceleration of link i for values of i from 1 to n

$$\ddot{q}_i = \frac{u_i - \hat{\underline{h}}_i^S \, \hat{\underline{a}}_{i-1}}{d_i}, \tag{5.22}$$

$$\hat{\underline{a}}_i = \hat{\underline{a}}_{i-1} + \hat{\underline{c}}_i + \hat{\underline{s}}_i \, \ddot{q}_i \tag{5.23}$$

with initial value

$$\hat{\underline{a}}_0 = \hat{\underline{0}},$$

for an immobile fixed base, or

$$\hat{\underline{a}}_i = -(\hat{\underline{I}}_i^A)^{-1} \, \hat{\underline{p}}_i,$$

for a mobile base (complete discussion is given later in Section 6.2.3).

Up to this stage, all the values of $\hat{\underline{I}}_i$ and $\hat{\underline{s}}_i$ are required in *absolute coordinates.*

Greater efficiency can be achieved by doing the calculations in *link coordinates.* This avoids having to transform the $\hat{\underline{I}}_i$ and $\hat{\underline{s}}_i$ from link to absolute coordinates, but it does introduce the need to transform the $\hat{\underline{I}}^A_i$, $\hat{\underline{p}}_i$ and $\hat{\underline{a}}_i$ from one link-coordinate system to the next (e.g. aligning coordinate-system axes with joint axes).

This implies the need of the 6x6 transformation matrix

$$\hat{X}_{i+1},$$

which transforms from the coordinate system of link (i+1) to that of link i (see next section below).

5.5.5 Transformation Matrices

As pointed out earlier on, greater efficiency can be achieved if all calculations are done in link coordinates. With each link, let there be associated a coordinate system which moves with that link.

Let $\hat{X}_{i+1}$ be the spatial transformation matrix from the coordinate system of link i+1 to that of link i; a function of the joint variable $\ddot{q}_{i+1}$.

Since all the links are considered to be rotational, the transformation matrix is derived as follows.

Rotation of a 6-vector is accomplished by rotating its 3-vector components separately. If E is the 3x3 orthogonal matrix that transforms the representation of a 3-vector in one coordinate system to that in another *rotated* with respect to the first, then the corresponding 6x6 rotation matrix is given by

$$\begin{bmatrix} E & \underline{0} \\ \underline{0} & E \end{bmatrix}$$

where, rotation in the z-direction through an angle a, for example, is represented by

$$E = \begin{bmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The actual spatial transformation matrix used in the method involves a translation $\underline{r}$, (the translation part involves the use of the position vectors $\underline{r}_1$ from the coordinates of one link to the coordinates of the next one, related to a fixed origin), followed by a rotation E (as given above), combined by multiplying the transformation matrices in the correct order.

Thus,

$$\hat{X}_{1+1} = \begin{bmatrix} E & \underline{0} \\ \underline{0} & E \end{bmatrix} \begin{bmatrix} \underline{1} & \underline{0} \\ \underline{r}x^T & \underline{1} \end{bmatrix} = \begin{bmatrix} E & \underline{0} \\ E\underline{r}x^T & E \end{bmatrix}$$

where,

$$\underline{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}, \qquad \underline{r}x = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

the anti-symmetric matrix of $\underline{r}$, and

$$\underline{r}x^T = \begin{bmatrix} 0 & r_z & -r_y \\ -r_z & 0 & r_x \\ r_y & -r_x & 0 \end{bmatrix}$$

its transpose. Therefore,

$$\hat{X}_{1+1} =$$

$$\begin{bmatrix} \cos(a) & \sin(a) & 0 & 0 & 0 & 0 \\ -\sin(a) & \cos(a) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -r_z\sin(a) & r_z\cos(a) & r_x\sin(a)-r_y\cos(a) & \cos(a) & \sin(a) & 0 \\ -r_z\cos(a) & -r_z\sin(a) & r_y\sin(a)+r_x\cos(a) & -\sin(a) & \cos(a) & 0 \\ r_y & -r_x & 0 & 0 & 0 & 1 \end{bmatrix}$$

As it was already stated, ${}_{i}\hat{X}_{i+1}$ is the transformation from link i+1 to link i. Now, ${}_{i+1}\hat{X}_{i}$ is the inverse transformation from link i to link i+1. It can be shown that the inverse of the matrix ${}_{i}\hat{X}_{i+1}$ i.e. ${}_{i+1}\hat{X}_{i}$ is its transpose which establishes that the matrix is orthogonal.

For example, the transpose operator defined in Chapter 4, is given as,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{S} = \begin{bmatrix} D^{T} & B^{T} \\ C^{T} & A^{T} \end{bmatrix}$$

where A, B, C, D are 3x3 matrices, and therefore,

$${}_{i}\hat{X}_{i+1} = {}_{i+1}\hat{X}_{i}^{-1} = {}_{i+1}\hat{X}_{i}^{S}.$$

5.5.6 The Articulated-Body Dynamics Algorithm

Introducing the transformation matrices defined above, the equations now become, accordingly,

$$\hat{\underline{v}}_{i} = {}_{i}\hat{X}_{i-1}\,\hat{\underline{v}}_{i-1} + \hat{\underline{s}}_{i}\,\dot{q}_{i}, \qquad (\hat{\underline{v}}_{0} = \hat{\underline{0}}) \tag{5.24}$$

$$\hat{\underline{p}}_{i}^{v} = \hat{\underline{v}}_{i} \times \hat{I}_{i}\,\hat{\underline{v}}_{i}, \tag{5.25}$$

where, no transformation is applied to the bias force, since this is calculated locally within each link. Furthermore,

$$\hat{I}_{i}^{A} = \hat{I}_{i} + {}_{i}\hat{X}_{i+1}\left[\,\hat{I}_{i+1}^{A} - \frac{\hat{\underline{h}}_{i+1}\,\hat{\underline{h}}_{i+1}^{S}}{d_{i+1}}\,\right]{}_{i+1}\hat{X}_{i}, \qquad (\hat{I}_{n}^{A} = \hat{I}_{n}) \tag{5.26}$$

$$\hat{\underline{p}}_{i} = \hat{\underline{p}}_{i}^{v} + {}_{i}\hat{X}_{i+1}\left[\,\hat{\underline{p}}_{i+1} + \hat{I}_{i+1}^{A}\,\hat{\underline{c}}_{i+1} + \frac{u_{i+1}}{d_{i+1}}\,\hat{\underline{h}}_{i+1}\,\right], \qquad (\hat{\underline{p}}_{n} = \hat{\underline{p}}_{n}^{v}) \tag{5.27}$$

$$\ddot{q}_{i} = \frac{u_{i} - \hat{\underline{h}}_{i}^{S}\,{}_{i}\hat{X}_{i-1}\,\hat{\underline{a}}_{i-1}}{d_{i}}, \tag{5.28}$$

and

$$\hat{\underline{a}}_i = {}_i\hat{X}_{i-1}\,\hat{\underline{a}}_{i-1} + \hat{\underline{c}}_i + \hat{\underline{s}}_i\,\ddot{q}_i \qquad (\hat{\underline{a}}_0 = \hat{\underline{0}}) \qquad (5.29)$$

modified to include the necessary coordinate transformations. Note that the common sub-expressions, $\hat{\underline{c}}_i$, $\hat{\underline{h}}_i$, d_i and u_i are given local to each link, and therefore no transformation needs to be applied on them.

5.6 CONCLUSIONS

A new, matrix-based notation has been introduced which represents articulated-body inertias and other spatial quantities. Spatial quantities are represented by physical vector quantities.

This notation is used to develop the algorithm and results in a compact representation of the equations, i.e. substantial reduction in the size of the equations and in the number of steps necessary to derive them.

In the equations, with each link we associate a local coordinate system which moves with that link. The spatial transformation ${}_i\hat{X}_{i+1}$ operates on a vector represented in coordinate system i+1 to produce a representation of the same vector in coordinate system i. Introducing this concept achieves greater efficiency since all calculations are done in link coordinates rather than in absolute coordinate systems. This avoids having to transform the $\hat{\underline{I}}_i$ and $\hat{\underline{s}}_i$ from link to absolute coordinates, but it does introduce the need to transform the $\hat{I}^A_i$, $\hat{\underline{p}}_i$ and $\hat{\underline{a}}_i$ from one link coordinate system to the next.

The new algorithm has computational requirement that varies linearly with the number of joints, that is, its computational complexity is O(n).

The method is more efficient than either Armstrong's O(n) method and Hollerbach's O(n) method (see Appendix 5C), or Walker and Orin's $O(n^2)$ method, and of course, becomes more beneficial than the most efficient of Walker & Orin's $O(n^3)$ methods, for bodies with more than 12 joints.

Revolute and prismatic joints can be handled with this method.

The method calculates first the homogeneous articulated-body inertias for each link, using a fixed-step iteration that starts at the end-effector and works towards the base; secondly, it calculates the joint accelerations in an inner fixed-step iteration, this time working from the base toward the end-effector.

The method is easily extended to cope with a mobile base (to be considered in the next chapter), but is not easily extended to cope with closed-loop kinematic chains (articulated bodies do not fall in this category anyway).

APPENDIX 5A -

Calculation of Inertia Tensors

The moments of inertia with respect to the X-, Y-, and Z-axes are defined as,

$$I_{xx} = \int (y^2 + z^2)dm, \qquad I_{yy} = \int (z^2 + x^2)dm, \qquad I_{zz} = \int (x^2 + y^2)dm,$$

respectively, and the products of inertia are defined as

$$I_{xy} = \int xy\ dm, \qquad I_{yz} = \int yz\ dm, \quad \text{and} \quad I_{xz} = \int xz\ dm.$$

The integration is taken over the mass (m) of the body. If we are free to choose the orientation of the reference frame, it is possible to cause the products of inertia to be zero. That is, if the X-axis or the Y-axis or both are axes of symmetry, then $I_{xy} = 0$. Similarly for I_{yz} and I_{xz}.

If we consider a box with dimensions a, b and c and mass m, then the values of the inertia tensors are as follows,

$$I_{xx} = \frac{m(b^2+c^2)}{12}, \qquad I_{yy} = \frac{m(8a^2+b^2)}{12}, \qquad I_{zz} = \frac{m(8a^2+c^2)}{12}$$

$$I_{xy} = \frac{mac}{4}, \qquad I_{yz} = \frac{mbc}{16}, \quad \text{and} \quad I_{xz} = \frac{mab}{4},$$

where, a corresponds to the length of the box, X-axis; b corresponds to its width, Z-axis; and, c is its height, Y-axis. The origin of each local coordinate system of a link is attached to the proximal joint of that link.

APPENDIX 5B -

Joint Axis Representation

The concept of the joint axis has already been introduced in Chapter 4 (Section 4.4.4).

Accordingly, a revolute joint axis is a directed line with a definite position and can be represented by a unit line vector lying on the joint axis.

Now, if $\underline{s}$ is a unit vector giving the direction of the axis then, the equation of the line is

$$\underline{s}_0 = \underline{r} \times \underline{s} = \begin{bmatrix} \underline{i} & \underline{j} & \underline{k} \\ r_1 & r_2 & r_3 \\ s_1 & s_2 & s_3 \end{bmatrix}$$

$$= \underline{i}(r_2 s_3 - s_2 r_3) - \underline{j}(r_1 s_3 - s_1 r_3) + \underline{k}(r_1 s_2 - s_1 r_2)$$

where, $\underline{r}$ is the position of any point on the axis.

It is thus clear that, the 6-vector representation of the joint axis $\hat{\underline{s}}$, is

$$\hat{\underline{s}} = [\underline{s}^T \ \underline{s}_0^T]^T$$

$$= [s_1, \ s_2, \ s_3, \ r_2 s_3 - r_3 s_2, \ r_3 s_1 - r_1 s_3, \ r_1 s_2 - r_2 s_1]^T.$$

APPENDIX 5C -

Comparison of the Computational Requirements of Dynamics Formulations

A general outline of the 3x3 Lagrangian formulation used by Hollerbach is given in this section together with a comparison of the computational requirements of Hollerbach's method versus Featherstone's method.

Hollerbach's method uses an efficient Lagrangian Formulation for manipulation of *inverse dynamics*, namely the problem of computing the joint torques required to produce given joint positions, velocities and accelerations. The Lagrangian dynamic formulation, based on the principle of conservation of energy, provides a means of deriving the equations of motion from a scalar called the *Lagrangian*, which is the difference between the kinetic and potential energy of a mechanical system, already described in Chapter 3.

The method is efficient because the number of multiplications and additions varies linearly with the number of joints, i.e. the formulation makes it possible to compute the Lagrangian dynamics in real time.

Specifically, there are three parts of the reformulation :

(1) backward recursion of the velocities and accelerations working from the base of the manipulator to the end link,

(2) forward recursion of the generalised forces working from the end link to the base of the manipulator, and

(3) use of 3x3 rotation matrices.

In Featherstone's method steps (1) and (2) are interchanged as the formulation is a direct dynamics one, rather than an inverse one. In step (3), 6x6 matrices are used instead of 3x3.

Furthermore, the following table displays the comparison of the computational costs of the two methods.

(1) Featherstone O(n) algorithm versus (2) Hollerbach O(n) method			
Number of Joints	n	6	12
(1) O(n) - 6x6			
Multiplications	199n-198	996	2,190
Additions	174n-173	871	1,915
(2) O(n) - 3x3			
Multiplications	412n-277	2,195	4,667
Additions	320n-201	1,719	3,639

Table 5.1 - Comparison of Computational Requirements of Dynamics Formulations

As it can be seen, Featherstone's O(n) direct dynamics method is more efficient than Hollerbach's O(n) inverse dynamics method.

Table 5.1 illustrates the superiority of the first method over the second as Hollerbach's method uses 3x3 matrices rather than 6x6 which Featherstone is using. Usually greater reduction of computational requirements can be obtained by reducing the size of the coefficients.

CHAPTER 6

ADAPTATIONS OF THE ARTICULATED-BODY METHOD TO COPE WITH AN ARBITRARY ANTHROPOMORPHIC MANIPULATOR

6.1 INTRODUCTION

The equations of motion making up Featherstone's articulated-body algorithm have been presented in the previous chapter. As has already been stated, this method was originally developed for applications in robotics, and therefore standard features such as the use of single-degree-of-freedom joints and unbranched chains were adequate. Starting with the basic idea of his method, we have tried to extend and adapt the method to cope with the animation of any human-like, articulated, model.

In this chapter, we describe the modifications made to the initial algorithm, with the aim of formulating the dynamic analysis which will be applied to anthropomorphic manipulators.

Accordingly, branched kinematic chains and multiple-degree-of-freedom joints are essential requirements, as is the introduction of a mobile base in the structure.

6.2 EXTENSIONS OF THE DYNAMICS ALGORITHM

In this section we are concerned with generalisations of the basic algorithm discussed above, that is, modifications to allow branched kinematic chains where more than one branch comes out from a specific joint, multiple-degree-of-freedom joints to allow for rotations about the three coordinate axes, and the introduction of a moving base.

To achieve the first objective - the inclusion of *branched kinematic chains* - the link connectivity needs to be described explicitly as part of the system model. Some systematic modifications have to be made to the equations of the dynamics algorithm, but the properties and performance of the algorithm are not greatly affected.

On the other hand, implementing *multiple-degree-of-freedom joints* introduces significant changes to both the system model and the dynamics algorithm. It is more convenient to synthesise multiple-degree-of-freedom joints from the appropriate number of single-degree-of-freedom joints, although this may sometimes lead to computational inefficiencies. The modifications of the algorithm to allow both objectives are introduced below.

6.2.1 Modifications to Allow Multiple-Degree-of-Freedom Joints

The simplified human model is shown in Figure 6.1. Joints are indicated by circles and are labelled using a lettering character, whereas links, the rods, are numbered. Links 3 and 5 represent the upper limbs, link 1 represents the torso and 8 and 10 the lower limbs.

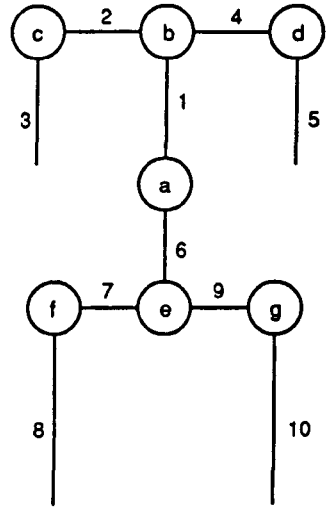


Figure 6.1

The model consists of ten links connected together by seven joints. In Figure 6.2a, the model of Figure 6.1 is drawn in the form of a tree representation. Links are indicated by hexagons, and w indicates that the model is attached to a fixed, world coordinate frame.

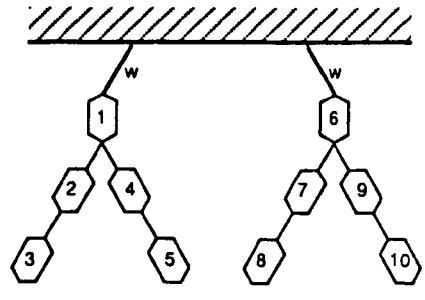


Figure 6.2a

To allow for multiple-degree-of-freedom joints each joint has three degrees of freedom, which are represented as a sequence of three single-degree-of-freedom joints, resulting in a total of twenty-one joints. The mass of each link is divided equally among the three links which make up the corresponding multiple-degree-of-freedom joint, so that the three links together are considered to be equivalent to the one, which has the total mass. The joints form a linkage tree consisting of rigid links connected together, as shown in Figure 6.2b.

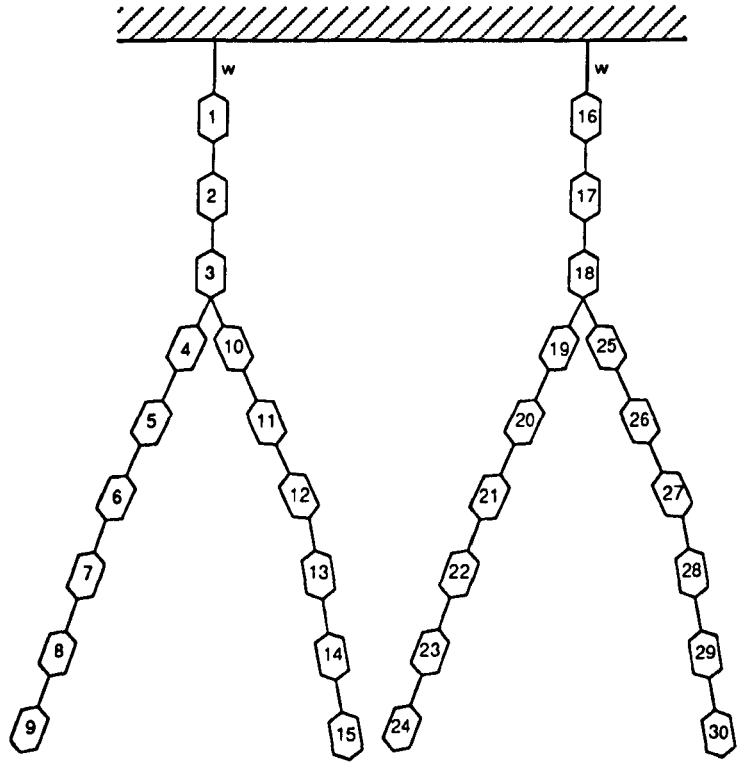


Figure 6.2b

In Figure 6.1 there is an immobile base, joint 'a', to which links 1 and 6 of Figure 6.2a, or links 1 and 16 of Figure 6.2b, are attached. Introducing this immobile base attached to both the corresponding links preserves the symmetry in the system, which makes the calculations easier to perform and test. Subsequently, applying a force to link 3, of Figure 6.2a, will cause that link to accelerate. The resulting motion must be equivalent to the motion produced by applying a force to the symmetric link 5. The same will happen if we apply a force to link 8 or 10.

6.2.2 Modifications to Allow Branched Kinematic Chains

To define a tree structure, we need to specify the connectivity explicitly, whereas for the unbranched chain it was implicit in the numbering scheme. Thus, for each link i we introduce a quantity λ_i which is the identification number of the preceding link. For example, from Figure 6.2b, $\lambda_4 = \lambda_{10} = 3$.

While the λ_i's alone are sufficient to define the connectivity, for expressing iterations going from the base to the leaves of the model, it is convenient to introduce for each link the additional quantity μ_i, which is the set of identification numbers of all the successors of link i. For example, $\mu_3 = \{4, 10\}$.

Thus, each link has only one preceding link, λ_i, but may have any number of succeeding links. For each leaf in the tree, μ_i is empty.

It is obvious that systematic use of these two quantities defines completely the position of each link relative to the other links, thus describing the overall structure of the model.

To modify the equations of the algorithm, it is best if we classify them into three groups :

(a) equations that iterate from the base link to the end-effector,

(b) equations that iterate from the end-effector to the base, and

(c) equations that perform computations local to each link.

(a) Outward-iterating equations carry motion-type equations from the base to the end-effector

As an example, consider Eq. 5.14 in which the link velocity is given by

$$\hat{\underline{v}}_i = \hat{\underline{v}}_{i-1} + \hat{\underline{s}}_i \dot{q}_i \qquad \text{(where } \hat{\underline{v}}_0 = \hat{\underline{0}}\text{)}.$$

In this expression the velocity of link i is given in terms of the velocity of the 'predecessor' of link i, so modifying these equations to work on branched kinematic chains is simply a matter of replacing instances of i-1 with λ_i. Therefore, for a tree structure model, the link velocity equation becomes,

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

(b) Inward-iterating equations carry forces or inertias from the end-effector to the base

One such equation is the calculation of articulated-body inertias. For an unbranched chain, this was given in Eq. 5.20 as,

$$\hat{\underline{I}}^A_i = \hat{\underline{I}}_i + \hat{\underline{I}}^A_{i+1} - \frac{\hat{\underline{I}}^A_{i+1}\,\hat{\underline{s}}_{i+1}\,\hat{\underline{s}}^S_{i+1}\,\hat{\underline{I}}^A_{i+1}}{\hat{\underline{s}}^S_{i+1}\,\hat{\underline{I}}_{i+1}\,\hat{\underline{s}}_{i+1}} \qquad (\hat{\underline{I}}^A_n = \hat{\underline{I}}_n).$$

That is, there is a whole expression that is passed back from the successor link. Within the tree structure, any given link receives one such expression from each of its successors, and since these quantities are additive, the total amount is the sum of all the contributions. Therefore, the equation is rewritten as,

$$\hat{\underline{I}}^A_i = \hat{\underline{I}}_i + \sum_{j \in \mu_i} \left(\hat{\underline{I}}^A_j - \frac{\hat{\underline{I}}^A_j\,\hat{\underline{s}}_j\,\hat{\underline{s}}^S_j\,\hat{\underline{I}}^A_j}{\hat{\underline{s}}^S_j\,\hat{\underline{I}}^A_j\,\hat{\underline{s}}_j} \right).$$

Here no initial condition is needed, since when link i is a leaf, the calculation of $\hat{I}_i^A$ is handled correctly by the nature of the definition of μ_i, as being the empty set for such a link.

Note also, that the order of traversal of the tree can be controlled by the numbering scheme of the model. It is possible to number the links in branched kinematic chains in many different ways. For consistency, it is practical if every link has a higher identification number than its predecessor, so that the link's numbering system can determine their positional order. In the same way, the joints are labelled in an increasing alphabetical order and the adjacent-link coordinate transformation across joint i is now, $\lambda_i \hat{X}_i$.

(c) *Computations local to each link*

Members of this group are independent of the connectivity and need no modification. Examples are the common sub-expressions, Eqs. 5.16 - 5.19, and the bias force $\hat{p}_i^v$, Eq. 5.15.

6.2.3 A Moving Base rather than a Stationary One

Since the method was originally developed for use in robotics, a non-moving, fixed base was satisfactory. For the development of a mobile base, the first requirement is the need of 'inverting' a 6x6 matrix, the matrix of the articulated-body inertia of the base link.

The Gauss-Jordan method, discussed by Burden & Faires [24], is a simple and efficient method which provides a good numerical approximation of the inverse matrix, saving the expensive computations associated with analytic matrix inversion.

Following the theory, let $A = (a_{ij})$ be an nxn matrix. Assuming a unique inverse, A^{-1}, exists, we need to find an nxn matrix $X = (x_{ij})$ such that AX=I.

Using the Gauss-Jordan method, A is invertible if, and only if, it can be reduced to the nxn identity matrix I using elementary row operations. When this reduction can be made, the matrix (x_{ij}) obtained by reducing the following partitioned matrix

$$\left(\begin{array}{cccc|cccc} a_{11} & a_{12} & \cdots & a_{1n} & 1 & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & a_{2n} & 0 & 1 & \cdots & 0 \\ \vdots & & & & \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} & 0 & 0 & \cdots & 1 \end{array}\right)$$

to the form

$$\left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & x_{11} & x_{12} & \cdots & x_{1n} \\ 0 & 1 & \cdots & 0 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & & \vdots & & & \\ 0 & 0 & \cdots & 1 & x_{n1} & x_{n2} & \cdots & x_{nn} \end{array}\right)$$

is the inverse of A,

$$A^{-1} = (x_{ij}) = \left(\begin{array}{cccc} x_{11} & x_{12} & & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{array}\right).$$

The mobile, floating base is essential for the purposes of this research, as it provides generality to the model. The computational costs of the articulated-body algorithm are not affected significantly by its use.

As a result of introducing the mobile base, the structure of the model needs to be changed slightly, see Figure 6.3, where two joints associated with the root are introduced.

As has been stated earlier, each joint had three rotational degrees of freedom, one each for the x, y, and z directions of rotation. Having a mobile base introduces three rotational degrees of freedom for the motion of the root, together with three additional translational degrees of freedom for its connection with the environment. Therefore, the tree structure of Figure 6.3 has 7 joints, all of which have three rotational degrees of freedom, in addition to the six degrees of freedom of the root. Joints 'a' and 'b' constitute the six joints of the root, where joint 'a' represents the three translational degrees of freedom and joint 'b' represents the three rotational degrees of freedom, a total of 27 degrees of freedom.

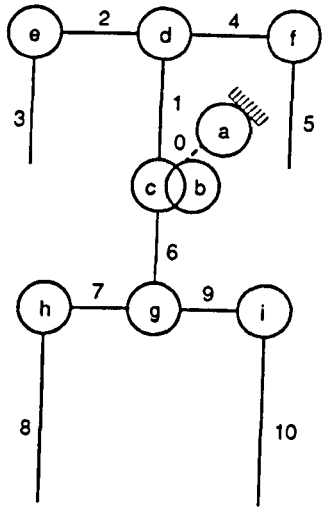


Figure 6.3

The tree representation for the new hierarchical structure is shown in Figure 6.4a and the extended version for multiple-degrees-of-freedom joints is illustrated in Figure 6.4b, where link 0, the mobile base, is attached to a fixed frame w, which is the world, or reference, coordinate system. Figure 6.4a has ten physical links 1..10, in addition to the two links of the mobile root (a translational link α, plus a dimensionless, rotational, link β). Link α corresponds to the translational links 0..2 of Figure 6.4b and link β corresponds to the rotational links 3..5.

Using Eq. 5.2, the acceleration of the base is given by,

$$\hat{\underline{a}}_0 = (\hat{\underline{I}}_0^A)^{-1} (\hat{\underline{f}}_0 - \hat{\underline{p}}_0).$$

That is, the spatial acceleration of the root body (link 0) is computed from the inverse of its articulated body inertia, calculated by the Gauss-Jordan method described above, and the difference between the applied force and the bias force associated with the base.

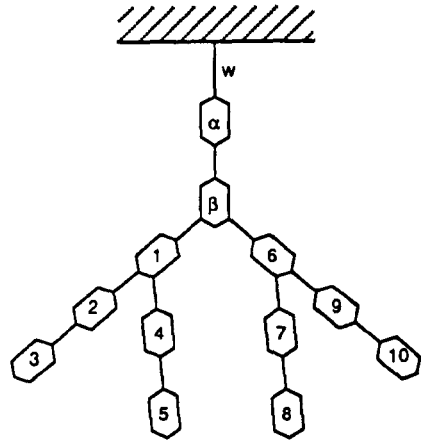


Figure 6.4a

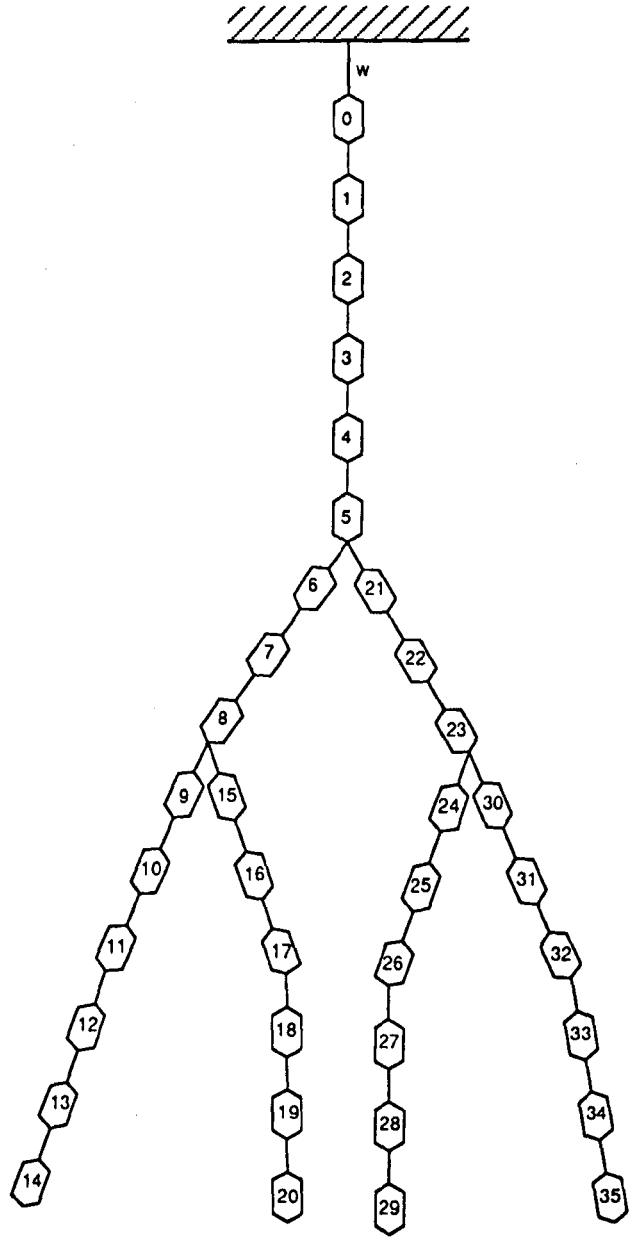


Figure 6.4b

6.3 THE MODIFIED ALGORITHM

The following section shows the effects produced in Eqs. 5.16 - 5.19 i.e. the common sub-expressions which can be computed once and stored for later use, and in Eqs. 5.24 - 5.29 by the extensions suggested in Section 6.2.

Outward iterations

The absolute velocity $\hat{\underline{v}}_i$ and the bias force due to velocity-product force $\hat{\underline{p}}_i^v$ are calculated next, Eq. 5.24,

$$\hat{\underline{v}}_i = {}_i\hat{X}_{\lambda_i}\, \hat{\underline{v}}_{\lambda_i} + \hat{\underline{s}}_i\, \dot{q}_i \qquad (\hat{\underline{v}}_0 = \hat{\underline{0}}) \qquad (6.1)$$

where $\dot{q}_i$ is the velocity of link i relative to link i-1, and i takes the values 0, the root of the hierarchy, to N, the end-effector. The number of links N is equal to 3n, where n represents the original number of links of a mobile model (twelve as introduced in Figure 6.4a). That is, N = 36, i.e. link numbers 0..35, if the hierarchical model of Figure 6.4b is employed.

Local calculations

These are given local to each link and thus need no modification, so Eqs. 5.25 and 5.16 remain unchanged.

$$\hat{\underline{p}}_i^v = \hat{\underline{v}}_i \times \hat{\underline{I}}_i\, \hat{\underline{v}}_i \qquad (6.2)$$

$$\hat{\underline{c}}_i = \hat{\underline{v}}_i \times \hat{\underline{s}}_i\, \dot{q}_i\ . \qquad (6.3)$$

Inward iterations

We can calculate the common sub-expressions $\hat{\underline{h}}_i$, d_i and u_i, the articulated-body inertia and the bias force for values of i from N-1 down to 1. As mentioned in Section 6.2.2, no initial condition is needed.

$$\hat{\underline{h}}_i = \hat{\underline{I}}_i^A\, \hat{\underline{s}}_i, \qquad (6.4)$$

$$d_i = \hat{\underline{s}}_i^S \hat{\underline{h}}_i, \text{ and} \tag{6.5}$$

$$u_i = Q_i - \hat{\underline{h}}_i^S \hat{\underline{c}}_i - \hat{\underline{s}}_i^S \hat{\underline{p}}_i, \tag{6.6}$$

$$\hat{\underline{I}}_i^A = \hat{\underline{I}}_i + \sum_{j \in \mu} {}_i\hat{X}_j \left(\hat{\underline{I}}_j^A - \frac{\hat{\underline{h}}_j \hat{\underline{h}}_j^S}{d_j} \right) {}_j\hat{X}_i, \tag{6.7}$$

$$\hat{\underline{p}}_i = \hat{\underline{p}}_i^v + \sum_{j \in \mu_i} {}_i\hat{X}_j \left(\hat{\underline{p}}_j + \hat{\underline{I}}_j^A \hat{\underline{c}}_j + \frac{u_j}{d_j} \hat{\underline{h}}_j \right). \tag{6.8}$$

Mobile base

The acceleration of the base link is given by

$$\hat{\underline{a}}_0 = (\hat{\underline{I}}_0^A)^{-1} (\hat{\underline{f}}_0 - \hat{\underline{p}}_0), \tag{6.9}$$

which requires the inversion of the matrix (6x6) of the articulated-body inertia of the base link.

Outward iterations

The acceleration of link i relative to link i-1, $\ddot{q}_i$, and the absolute acceleration of link i, $\hat{\underline{a}}_i$, for values of i from 1 to N, can be calculated, i.e. Eq. 5.28 and Eq. 5.29.

$$\ddot{q}_i = \frac{u_i - \hat{\underline{h}}_i^S {}_i\hat{X}_{\lambda_i} \hat{\underline{a}}_{\lambda_i}}{d_i}, \tag{6.10}$$

$$\hat{\underline{a}}_i = {}_i\hat{X}_{\lambda_i} \hat{\underline{a}}_{\lambda_i} + \hat{\underline{c}}_i + \hat{\underline{s}}_i \ddot{q}_i, \tag{6.11}$$

where λ_i is the identification number of the preceding link and μ_i the set of identification numbers of all the successors of link i.

Featherstone's formulation has also been used by McKenna, Schroder and Zeltzer [23, 25, 50, 51] for the dynamic analysis of a cockroach. However, Schroder based his implementation on the use of quaternions, 4-dimensional coordinates, as discussed by Shoemake [55].

6.3.1 New Description of the Model

Once the extended and modified algorithm was applied and tested on the simple structure of the model described above, a more sophisticated structure was introduced and used throughout the development of this work to test and illustrate the results. A useful extension was the introduction of the head which was not included in the old structure.

Therefore, the specification of the model had to be amended. For full control of the model another five joints were introduced : one on each arm representing the elbows, one on each leg representing the knees, and one for the neck. This is necessary and allows our anthropomorphic model to have the basic links/joints of the human figure, see Figure 6.5.

The number of degrees of freedom of the body increased from 27 to 42 and the new tree representations are given in Figures 6.6a and 6.6b.

The model described above is adequate enough for its purposes and it contains the basic links/joints of an anthropomorphic figure. Of course, more complicated structures will result in higher computational costs, an outcome which we are trying, if possible, to avoid.

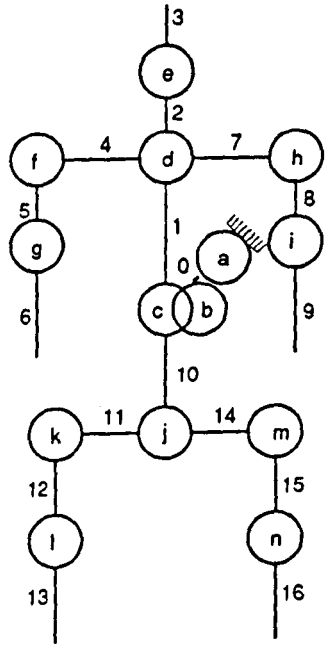


Figure 6.5

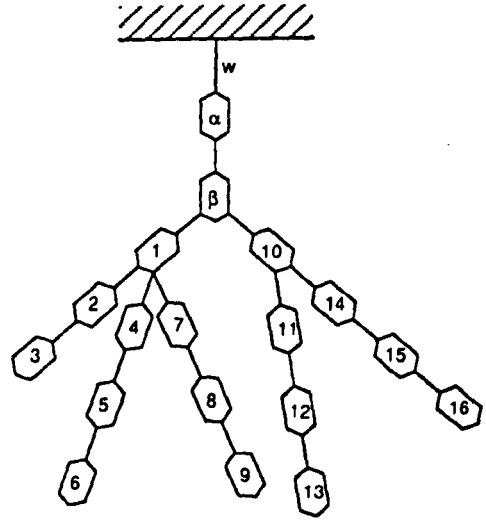


Figure 6.6a

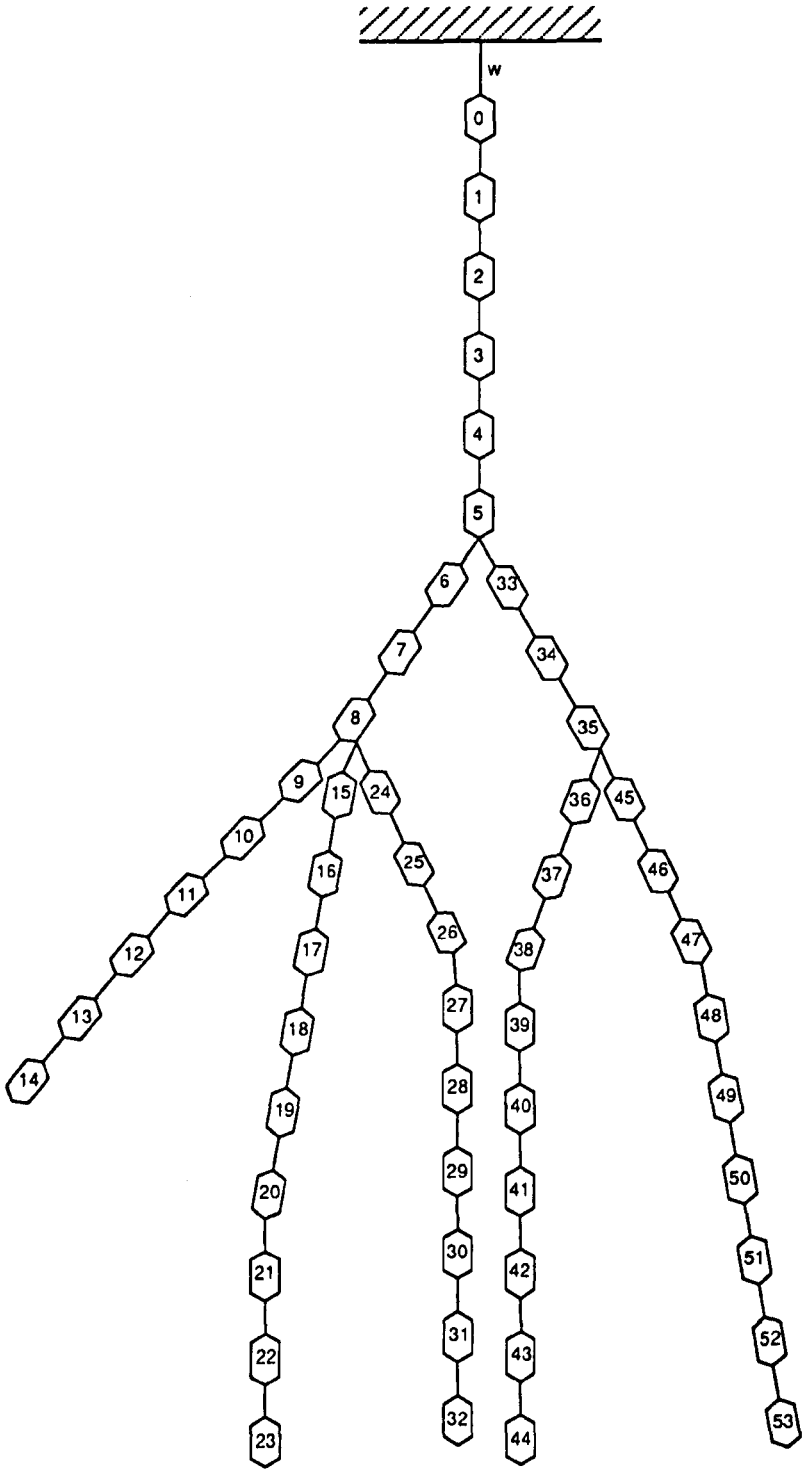

w
0
1
2
3
4
5
6
33
7
34
8
35
9
15
24
36
45
10
16
25
37
46
11
17
26
38
47
12
18
27
39
48
13
19
28
40
49
14
20
29
41
50
21
30
42
51
22
31
43
52
23
32
44
53

Figure 6.6b

6.4 CONCLUSIONS

The equations of the modified algorithm are described above. With each link we associate a local coordinate system which moves with that link, and we introduce the transformation matrix ${}_{i}\hat{X}_{i+1}$, as described in Chapter 5. For each link i the identification number of the preceding link, the one nearer to the root, is denoted by λ_i, and the set of identification numbers of all the successors of link i is denoted by μ_i. Therefore, every link except the base has exactly one inner joint, but may have any number of outer joints.

The algorithm operates basically as follows.

Beginning with the end-effectors, and operating towards the root body, the articulated-body inertia and bias force are computed. This involves the calculation of the velocity-dependent force, the four common sub-expressions and finally, the articulated-body inertia and the bias force.

The second step, involves the computation of the acceleration of the root body, using the inverse of its articulated-body inertia and its bias force.

Finally, the joint accelerations of the bodies are computed outwards from the root body to the leaf, together with the spatial acceleration of body i that is needed to compute the joint acceleration of body i+1.

The extension of the method to handle branched kinematic chains implies that there are multiple leaf bodies, one at the end of each branch. Articulated-body inertia and bias forces simply sum at a branch node, that is the parent body where two or more child branches converge. This requires the change in the body-numbering subscript notation, as described above.

In the following chapters, the implementation details of this algorithm are considered, through the description of the interface environment built for the purposes of this research. Associated problems with regard to the motion of the human model are tackled.

CHAPTER 7
GROUND REACTION FORCES

7.1 INTRODUCTION

In this chapter, a detailed description of the ground reaction forces is given. This includes an outline of the very important issue of ground contact, the period of time where a point on the model and the floor surface coincide. A reformulation of the algorithm described in Chapter 6, to include these contact reaction forces, is necessary and a description of it is introduced below.

This is the first time where the floor surface is introduced and, as one might think, the interaction of the figure with the environment is quite a significant feature for the development of an animation sequence. Thus, if we want to create a walking sequence of the model, it is necessary to include the reaction forces at the point of contact. In contrast with running, where there is a period when both feet are off the ground, in walking at least one leg is in contact with the floor at all times and there is a period (double-support phase) when both legs are in contact with the ground.

7.2 GROUND REACTION FORCES FORMULATION

7.2.1 Introduction

The forces producing the displacement of the segments of the lower extremities are muscular, gravitational, and those due to the interaction with the environment. This section is concerned with the description of the external reaction forces caused by the physical contacts of our model with the ground.

During simulation of the two phases, human models must be prevented from penetrating the ground surface by the use of counteracting reaction forces. In real life, the prevention from moving through the ground is done automatically, but for the animation of any model, the reaction forces must be calculated by considering the present state of the entire body.

By assuming firm contact of the foot with the ground, the action of gravity enables the required floor reactions to be built up to enable movements such

as walking. Without gravity, locomotion would, at its best, be a difficult and haphazard operation. Contact with the ground would be undefined, and acceleration of the body would of necessity have to come entirely from transfer of momentum.

Early studies, Saunders et al [67], have been reported on the use of a *force-plate* for direct measurement of all ground reactions, including torques during locomotion. A force-plate is a sensitive electronic scale which records the magnitude of vertical force, torque, and horizontal shears, as well as the centre of pressure on the foot (the location of the resultant vertical forces, i.e. their point of application), McMahon [27]. However, it does not define the lines of action of the fore-and-aft and lateral shears.

A force-plate is exceedingly sensitive to any changes in the normal displacement pattern of the body as a whole and is more accurate than photographic methods, e.g. Muybridge data [65], for the determination of sudden changes, such as deviations of the centre of gravity of the body caused by alterations in accelerations.

Another important issue for which the force-plate has limitations, is the computation of resting force systems within the body. In considering the body as a whole, knowledge of the total weight plus the measured ground reactions is all that is required to determine the behaviour of the centre of gravity of the body. With this knowledge we can calculate the translational velocities and displacements of the centre of gravity of the body as a whole.

Reaction forces can be included in the dynamic analysis, but this of course involves a considerable increase in the computational costs of the algorithm, Moore & Wilhelms [57] and Wilhelms [63]. As an alternative, another set of dynamic equations can be solved to predict reaction forces when a portion of the body, for example a foot, is in contact with the ground. Wilhelms [63] suggested the approximation of this counteracting contact force by the use of springs and dampers in the interest of cost. The method does not provide realistic results since it is based on an initial estimate of the force. In order to start the calculation a portion of the body must be in contact with the floor. In this case, we need to calculate a normal force perpendicular to the ground. This force can vary from less than one to several times the body weight, and it can be distributed over several support points.

The problem can be considered as a kinematic constraint problem which arises from the physical contact of the body with the ground. The ground is modelled simply as a horizontal plane, and the contact forces depend upon the motion constraints.

The vertical floor reaction is not constant but varies above and below the body weight because of vertical upward and downward accelerations of the body. The difference between the vertical component of the floor reaction and the body weight is proportional to the vertical acceleration of the body. Initially, for the purposes of this research, only a single point of contact with the ground was examined.

To solve the problem, the following three steps need to be developed further.

1) The equation of motion of the system without contacts is found, subject to unknown contact forces,

2) the equation of motion is substituted into the constraint equations to give a set of equations in the unknown contact forces, and

3) the set of contact forces can be substituted into the equations of motion to find the accelerations.

Our main concern is the problem of instantaneous contact dynamics (i.e. determining the acceleration given the positions and velocities of the bodies), and the problem of instantaneous contact constraints and forces acting on the system.

To simplify the problem, a couple of assumptions need to be made.

- A finite number of point contacts is needed to describe the state of the contact. At an early stage, we assumed a single point of contact, and

- No Coulomb friction between contacting points is taken into account, Baraff [36].

7.2.2 Contact Kinematics

Following our previous discussion, a contact point occurs when two bodies meet at a single point. At the point of contact of a body with the ground surface, ground reaction forces need to be applied, and these are applied only to the three translational degrees of freedom of the root. Therefore, in our model, the foot which is in contact with the floor has to be the current root of the structure during its contact.

Boulic & Renault [78] refer to two alternative techniques which can be applied to the model after changing the root of its hierarchical structure. The first one was proposed by Risdale et al [40], where the topology of the hierarchical structure is redefined every time the root is changed. The second technique was reported by Sims & Zeltzer [77], where the tree representation remains the same, but the traversal of the hierarchy is redefined. However, in practice the two techniques are essentially the same.

The idea adopted for this research, which is similar to Risdale's, is that the topology of the hierarchical structure is redefined every time the root of the hierarchy changes; the traversal of the hierarchy remains the same (top to bottom). In this fashion, the root has to be changed from alternately being the left or right leg, discussed also by Vasilonikolidakis & Clapworthy [35] and Philips & Badler [34].

Since the root has been allowed to have three translational degrees of freedom, the values of the desired accelerations for the translational degrees of freedom can be found by solving a system of three linear equations, taking into account that these accelerations are independent. Thus, to set a linear force to a predefined value, only the acceleration associated with that degree of freedom has to change.

The contact is characterised by a contact normal, $\hat{\underline{n}}$, which is a unit line vector passing through the point of contact and normal to the surfaces at that point, with an upwards direction. Raibert & Hodgins [37, 38] have studied configurations where a single point of contact occurs. For such configurations a valid set of normal forces is one which satisfies the following conditions, Baraff [36]. Firstly, the normal force at each contact point must be oriented to 'push' the bodies apart, therefore, its value

should be greater than or equal to zero. Secondly, it is necessary for the normal forces to prevent inter-penetration, that is, the acceleration at the point of contact in the direction of $\hat{\underline{n}}$, the unit surface normal, must be non-negative. Thirdly, if two points are separating at a contact point, the normal force at the contact point must be zero.

In the absence of friction, a valid set of contact forces exists for any configuration of bodies. While this set is not necessarily unique, all valid contact forces yield the same accelerations of the bodies in the configuration, Cottle [39]. Contact forces for frictionless configurations i.e. including only static and no dynamic friction, with n contact points can be found by formulating and solving a convex quadratic program of n variables, Baraff [36, 46] and Featherstone [19].

Configurations with friction are more complicated. Contact forces with friction are valid if they satisfy both the previous three conditions for normal forces and the Coulomb friction model which states that if a contact point of a body A is sliding over a fixed body B, e.g. the ground surface, then a frictional force in the opposite direction to that of the motion acts on A. Valid contact forces for configurations with just dynamic friction (and no static friction) can be found, as in the frictionless case, Baraff [36], but this is not an easy problem and is beyond the scope of this research.

Contact dynamics has also been discussed by Miller [59, 76] in his attempt to move a worm convincingly along a straight path.

7.2.3 The Analysis - A Single Point of Contact

Three steps are required for solving a system of dynamic equations, assuming a single point of contact between our model and a fixed ground surface.

1. The accelerations for the translational degrees of freedom are determined by solving the system of Featherstone's dynamics algorithm for the three translational degrees of freedom of the appropriate foot, which becomes the root of the hierarchical structure at that specific instance. One equation for each degree of freedom is employed, and the new positions of these degrees of freedom are found by taking into account the entire articulated body.

2. From these new positions, the ground reaction forces of these degrees of freedom are calculated by analysing the contact force of that point into two components, one parallel to a vector from the centre of mass of the body to the contact point, and one perpendicular to it.

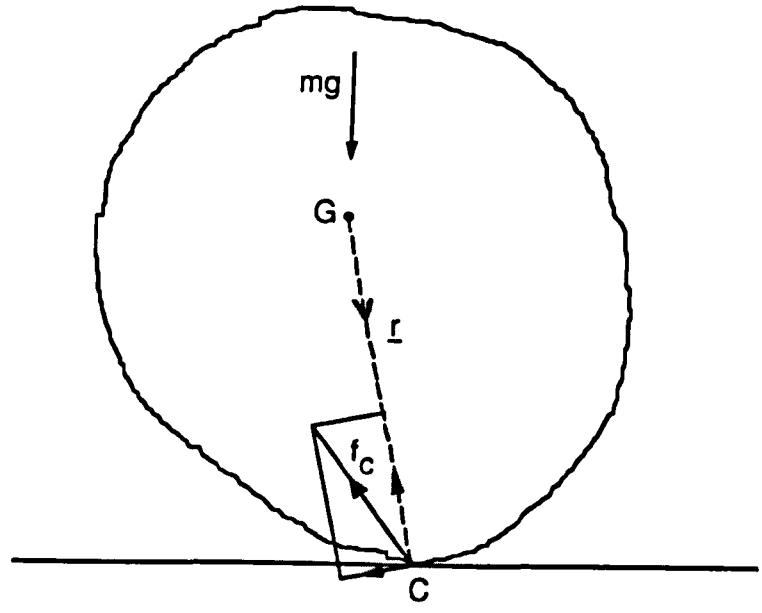
3. The system of dynamic equations are solved again for each joint of the hierarchical structure, considering the contact forces, to calculate the new required position of the foot and generally of the whole body. This should position the whole body into the correct configuration, resulting from the contact of a single point of a model and the ground surface.

Constraint equations need to be introduced to control the two components of the contact force. In this way, the parallel component will act as a linear spring along the leg axis pushing the body away from the ground, whereas the perpendicular component will control the resultant moment about the centre of mass of the body by applying a torque to the body about the mass centre. Using the deformation of a spring to measure a force is a convenient way. Springs that elongate in proportion to an applied force obey Hooke's law and these springs can be constructed and calibrated to measure unknown contact forces.

Figure 7.1 shows a composite body and its ground reaction force that is decomposed into two components as suggested by Kearney et al [48], who developed this idea for a hopping, one-legged machine. In the diagram, $\underline{r}$ is the vector from the centre of mass of the whole to the contact point, called the *virtual leg*. The spring force constraint may be expressed, Kearney et al [48], as

$$\underline{f}_c \cdot \frac{\underline{r}}{|\underline{r}|} = k_1 (|\underline{r}| - r_s) \qquad (7.1)$$

where $\underline{r}$ is the virtual leg and $\underline{f}_c$ represents the reaction force of the ground which acts at the point of contact. k_1 denotes the spring stiffness and the scalar r_s is the resting length of the spring. The parallel component of the contact force contributes to the linear force, normal at the point of contact, which implies that, as the centre of mass moves closer to the contact point, it is pushed away with greater force ($r_s > |\underline{r}|$). Some kind of experimentation is required to determine the values of the spring stiffness and the resting length of the spring, as suggested by Kearney.



Free-body diagram

Figure 7.1

The component of the contact force perpendicular to $\underline{r}$ is used to control the resultant moment about the centre of mass of the whole body, and is constrained to reduce the difference between the angular momentum of the body about its mass centre, $\underline{H}_G$, and the desired angular momentum of the body about its mass centre, $\underline{\tilde{H}}_G$. For example, Kearney's hopping model at take-off should have little or no angular momentum about its centre of mass. Therefore, this component moderates the direction of the push and applies a torque to the body about the mass centre. If the pushing is directed to the left of the centre of mass in Figure 7.1, a clockwise torque is applied to the body. The resultant external moment $\underline{M}_G$, about the centre of mass of the articulated body is

$$\underline{M}_G = \underline{r} \times \underline{f}_C = k_h (\underline{\tilde{H}}_G - \underline{H}_G) \qquad (7.2)$$

where k_h is a scaling factor determining the rate at which corrections are made.

The angular momentum about the mass centre of the whole of an articulated body consisting of n rigid bodies is determined by summing the angular momentum of each link about the centre of mass, and is defined by,

$$\underline{H}_G = \sum_{i=0}^{n} \left[\underline{I}_i^* \, \omega_i^* + (\underline{c}_i^* \times m_i \, \underline{v}_i^*) \right] \qquad (7.3)$$

where $\underline{I}_i^*$ is the inertia tensor of link i about its centre of mass,

ω_i^* the angular velocity of link i,

m_i the mass of link i,

$\underline{v}_i^*$ the velocity of the centre of mass of link i, and

$\underline{c}_i^*$ is the vector from the centre of mass of the whole body to the centre of mass of link i.

The * symbol distinguishes the terms from those introduced in the previous discussion : these are defined about the centre of mass of each link.

The form of the ground reaction force, $\hat{\underline{f}}$, expressed in spatial notation is

$$\hat{\underline{f}} = \begin{bmatrix} \underline{f} \\ \\ \underline{\tau} \end{bmatrix}$$

where $\underline{f}$ represents the three translational vector components of the ground reaction forces upon solution of the four equations, that is, Eq. 7.1 and the three sub-components resulting from Eq. 7.2. This results in a total of four unknowns r_s and $\underline{f}$, in the assumption that $\tilde{\underline{H}}_G$, the desired angular momentum is known (either from biological studies, or experimental results). Additionally, numerical experiments provided values for k_l and k_h. Different tests were carried out to establish acceptable values for the two terms. These showed that the k_h values control the magnitude of the force components whereas no immediate relationship could be established for the k_l values.

The rotational component of this force, $\underline{\tau}$ is chosen so that the resulting vector forms a spatial vector and in particular a line vector, discussed in Section 4.2.2. Thus,

$$\underline{\tau} = \underline{r} \times \underline{f},$$

which defines the position of the line once the magnitude and direction of the force are known, and $\underline{r}$ represents the vector from the origin to the contact point.

7.2.4 Re-formulation of the Algorithm

We therefore need a way of introducing this ground reaction force into the formulation of our algorithm. Recall that $\hat{\underline{p}}_i$ is the bias force of link i, Featherstone [19], and that it is the force that needs to be applied to that particular member of the articulated body to give it zero acceleration, taking into account the effect of the rest of the articulated structure.

Employing the tree structure of Section 6.3.1, the bias force of link i is considered to consist of its original form i.e. Eq. 6.5, minus the net external force, $\hat{\underline{f}}_i^{ext}$, applied to that link,

$$\hat{\underline{p}}_i = \hat{\underline{p}}_i^v - \hat{\underline{f}}_i^{ext} + {}_i\hat{X}_{i+1} \left[\hat{\underline{p}}_{i+1} + \hat{\underline{I}}_{i+1}^A \hat{\underline{c}}_{i+1} + \frac{u_{i+1}}{d_{i+1}} \hat{\underline{h}}_{i+1} \right] \qquad (7.4)$$

with initial condition $\hat{\underline{p}}_n = \hat{\underline{p}}_n^v - \hat{\underline{f}}_n^{ext}$. This is discussed also by McKenna [23, 25] who applied the same idea to a cockroach model.

This external force could be due to gravity and the contact with the ground and is included in the calculation of the spatial acceleration of the root, i.e. Eq. 6.9 (the negative bias force includes the externally applied forces).

In this fashion, we can model the contact between the ground and the human-like figure. It has been assumed that at contact, there is a single vector contact force, $\underline{f}_c$, that acts at the point of contact with the ground. This force and the force of gravity are the only forces that can change the momenta of the links of the model. Furthermore, the movement is controlled by defining constraints on the contact force.

7.3 CONCLUSIONS

The problem of simulating ground reaction forces has been discussed in this chapter, and it raised a number of interesting issues. This is due to the fact that the relative rotations at the joints during the double-support phase are closely related to the torque acting on the foot due to the floor reaction. The normal force can vary between less than one and several times the body weight (vertical upward and downward accelerations of the body) and can be distributed over several support points. We have assumed a single point of contact, which provides a good approximation for the resultant of the contact forces acting at points of a small surface area. However, further development has to take place to include situations where there is more than one point of contact, such as the double-support period in walking.

This is described in Section 9.4, through the use of the virtual-leg concept. To avoid any later misunderstanding, a distinction between the two terms, 'virtual leg' and 'virtual-leg concept' needs to be made. The virtual leg, already introduced in Section 7.2.3, is associated with a single point of contact and represents the vector from the centre of mass of the articulated body to the contact point. On the other hand, the virtual-leg concept, to be discussed in Chapter 9, represents the pair of legs acting in unison as an equivalent virtual leg during the double-support phase.

Reaction forces are needed to oppose the penetration of any of the body parts in the ground. However, they are not automatically calculated in the course of dynamic analysis and our system offers the possibility of including this feature at the expense of computational cost.

By the development of the important issue of ground reaction forces it is now feasible to experiment the creation of a fully dynamically-based, animated, walking sequence, which is the topic of Chapter 9.

CHAPTER 8

AnthroPI - ANTHROPOMORPHIC PROGRAMMING INTERFACE

8.1 INTRODUCTION

In this chapter the description of the animation system AnthroPI (Anthropomorphic Programming Interface) is outlined. AnthroPI, pronounced *anthropee*, is the plural, in Greek, of anthropos and means 'people'.

Issues to be considered include the choice of link origins and coordinate systems; the order of rotations and their constraints; the dimensions and positions of all the links of the model; the calculation of constants such as moments of inertia and transformation matrices; the connectivity of the structure and the solution of the equations developed in Chapter 6.

Although some of these issues have been described in earlier chapters, a brief reminder of them is given again, so that a clear view of the system can be formed.

As has already been described, local coordinate systems are attached to each link. These move with the links, and although they do require the introduction of transformation matrices from one link to the next, they have certain advantages. For example, quantities such as the moments of inertia of each link with respect to their local frame are constant throughout the motion. It is assumed that the origin of the local coordinate system of each link is at the proximal joint of that link.

In addition, an inertial (global) coordinate system is required to associate the root of the hierarchical structure (the three translational degrees of freedom of the root, as described in Chapter 6) with the world. For each coordinate system, the X-axis is constrained to be positioned along the length of the link. The Y-axis is extended along the height of the link and the Z-axis along its width. In this way, the rotation from one joint (the proximal joint of a given link) to the next joint retains this consistency. A right-handed coordinate system is used throughout the calculations whereas a left-handed coordinate system is used for the viewing operations.

Further, quantities such as moments of inertia, masses, and the original positions of the links need to be calculated before commencing the dynamics formulation. In our system AnthroPI, these are usually stored and read from files and provide an initial estimate which can be altered by the animator.

A solution can be sought once the quantities involved in the dynamic formulation are specified. Featherstone's method was chosen because of its superiority to the other methods, as discussed at the end of Chapter 3. This formulation, when used with the closely-related AnthroPI system, provides the necessary means to animate the movement of a human-like model, as well as to display the results.

In the next sections, a detailed description of the system AnthroPI is given.

8.2 AnthroPI - AN OUTLINE OF THE SYSTEM

AnthroPI requires as input the physical and behavioural characteristics of a linked figure and its initial state (initial position, with starting velocity and acceleration).

The physical model includes descriptions of all the links and joints of the structure as well as their connectivity. The structure is considered as a hierarchical tree model, with nodes representing the joints of the human-like figure and branches representing the links. To define the connectivity of the structure a two-dimensional table is constructed which indicates whether a connectivity between a particular joint and a particular link exists. The user can either create his own model, i.e. define the required table, or he/she can employ the experimental model which contains 42 degrees of freedom as described in Section 6.3.1.

The dynamic simulation is treated as an explicit time series analysis. At each time increment, the accelerations associated with all the degrees of freedom of the system are computed. These accelerations are then integrated and, by using information about the current state, a new set of velocities and positions is determined.

Each link has a physical size, mass, centre of mass and moment of inertia. In particular, specification of the dimensions and the density of the links of the figure can determine their inertia tensors. The linkage for each figure forms a tree structure. Each link possesses one joint by which it is attached to its parent link, but it may possess one or more joints by which child links are attached. Links move relative to each other via one to three

rotational degrees of freedom (DOF) associated with each joint. Configuration files are used to store the above-mentioned information about the model.

Joints have associated springs and/or dampers which act to exert internal forces or torques within that joint. Their implementation is discussed in Section 8.2.3.

The links respond to externally applied forces. The data concerning links and joints, the forces, the position and the velocity of the DOF form a complete description of the state of the dynamic system at any given time.

In this way, the dynamic analysis for each time increment can be broken down into four phases; determination of the applied behaviours (i.e. the forces acting on the model), solution of the equations of motion (i.e. dynamic analysis), calculation of the joint angles given the corresponding forces, and evaluation and display of the results. In Figure 8.1, the flow of control of the system is given. The diagram shows the procedural and structural components of the system. Some issues included in the diagram e.g. the gait controller, are introduced in the discussion that follows, which extends into Chapter 9.

Once all the physical parameters are determined, the model can be regarded as a passive linkage moving in response to applied loads. The links and joints of the model have a given position and velocity and they are subjected to internal forces (handled by the system) and external forces (handled by the user). The objective is to calculate the accelerations of the DOF.

The derivation of the equations of motion appears in Chapters 4-6. The recursive solutions have proved efficient for the determination of motion from specified forces. The solution provides the accelerations of the DOF for the current time increment. From these accelerations, the current position and velocity of each DOF, new positions and velocities are calculated for the following time step. If constraints are exceeded during the time increment, the constraining accelerations of the joint angles are determined and explicitly specified, so that the possibility of exceeding physically-possible joint angles is eliminated.

Once a particular motion is produced, displayed and accepted it can be stored in a file so that it can be re-played at any required pace, or it can be loaded later to provide some sort of testing.

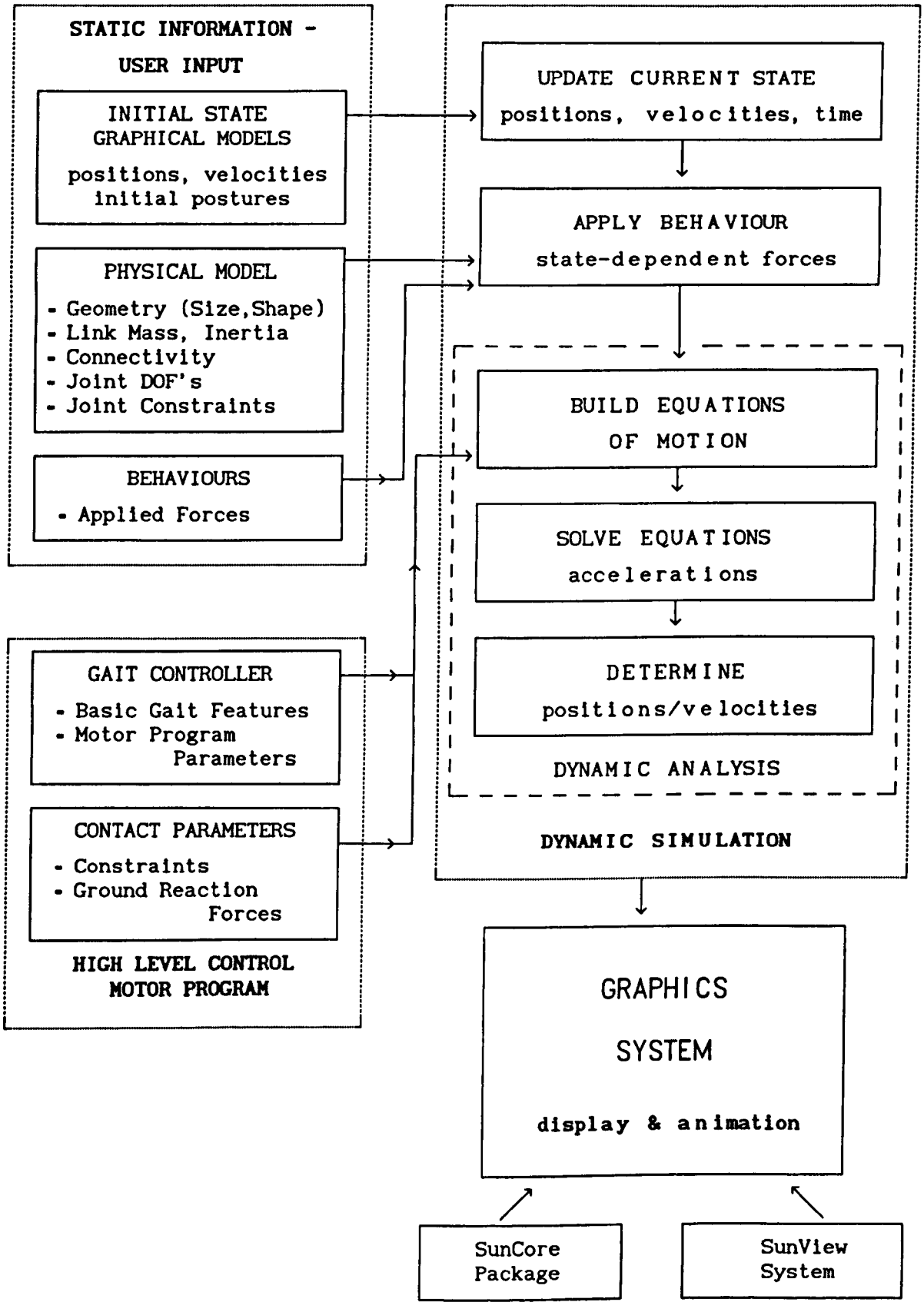


Figure 8.1 : AnthroPI - Flow of Control of the System

8.2.1 Calculation of Inertia Tensors

To calculate the inertia tensors of a body we need to consider its shape and mass distribution. It follows that more sophisticated shapes will require more complex calculations for the inertia tensors. For simplicity, we consider bodies with uniform mass distribution, and their inertia tensors are evaluated with respect to local coordinate systems, so that they remain constant throughout the motion. It is therefore convenient to approximate the shape of the human links with boxes, Vasilonikolidakis & Clapworthy [64], which are quite simple and provide an easy way to compute their inertia tensors.

The determination of the inertia tensors has already been discussed in Section 5.5.3 and explained in Appendix 5A. That is, for movements in three dimensions, moments and products of inertia are expressed in the form of a tensor matrix. For a homogeneous body, the inertia tensor $\underline{I}$ is given by,

$$\underline{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}.$$

8.2.2 Order of Rotations

All the joints of the hierarchical tree representation of the human figure can have up to three rotational degrees of freedom. The order of rotations of the structure can be set to the default value e.g. X-Y-Z for each link at the beginning of the simulation, or can be changed and assigned any order required by the user of the system.

By making the assumption that no two rotations in the same link involve the same axis, the possible combinations of the orders of rotations are six : X-Y-Z, Y-X-Z, Z-X-Y, X-Z-Y, Y-Z-X, and Z-Y-X.

8.2.3 Joint Limits

Positional constraints can be simulated by using appropriate forces to hold the body in position. Each joint has a certain range of angles that it is capable of moving through and angles outside this range will appear unnatural. Joint limits are implemented as associated springs which act to keep the DOF from moving beyond some point to prevent unrealistic movements. For example, attempts to rotate the elbow beyond its physical limits will initiate an internal force in the opposite direction to keep the movement within the physically-possible bounds.

Wilhelms [2] and Armstrong & Green [5] have both used linear springs where the spring force or torque is determined as the product of a spring constant times the distance that the spring is compressed. However, experimental results showed that the requirement to stop a motion quickly is better simulated with an exponential spring. As their name implies, these springs have an exponential relationship between the displacement, x, of the spring from its rest position (or angle) and the component of the force generated, f, which offers the capability of delivering rapidly-increasing forces near the joint limits.

When using a spring to restrict motion, certain issues with regard to its behaviour, need to be taken into consideration. The springs should not act in the central range of the joint angles, but only near their limiting values. Further, the spring should start its action in a small range before the limiting angle is reached, to ensure that the restorative force will restrict the movement to the physical range and prevent unnatural motions. Also, the spring should not act unless the movement of the joint is such that joint limits are being approached.

With linear springs this behaviour is less likely to happen for a given force output, since there is no rapid increase in the force as the joint limits are approached, but exponential springs have the advantage that at small displacements their restorative forces create more natural motions, see Figure 8.2. In addition, linear springs need to be very strong to create similar forces to the exponential springs at large displacements from the rest position of the spring. Exponential springs with a large displacement generate an extremely high force.

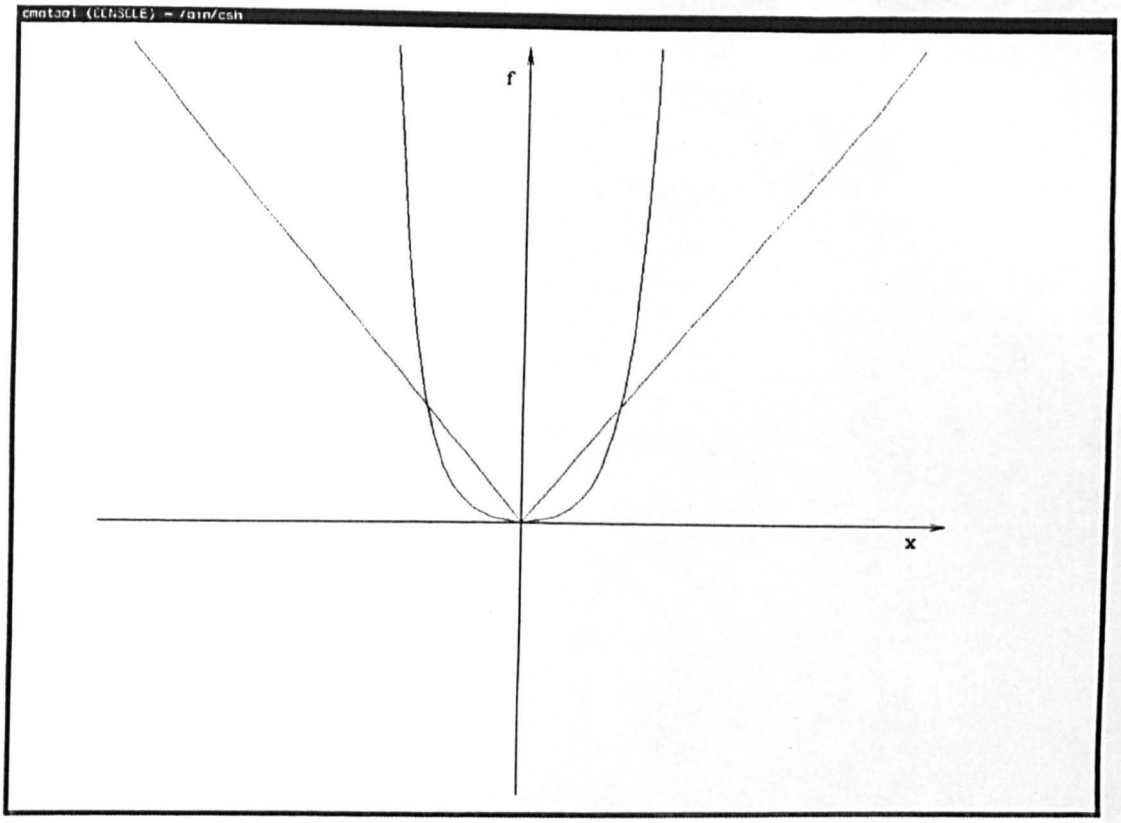


Figure 8.2 : Linear vs Exponential spring force response

Considering the motion of any joint, two directions should be taken into account. The first one is the 'extension' of the joint which reaches its maximum physical range, during which the spring is elongated, and the second the 'recovery' phase during which the joint approaches its minimum physical range and the spring passes through compression. The two directions of motion make the spring behave differently which implies two different functions are necessary to control its movements; one for the 'extension' of the spring, i.e. tension, and the other for the 'recovery' phase, i.e. compression.

To explain the motion of a joint consider, for example, the motion of the elbow. By assuming that the angle is measured from the middle of the physical possible motion, that is 0 lies at the centre of the available joint range, then, -x represents the angles which approach the minimum joint angle towards the upper arm, and +x the angles approaching the maximum angle. At the middle of the curve, i.e. x=0 no limiting force needs to be exerted.

Therefore, possible models that allow for a natural behaviour of the motion of a joint need to take into consideration the following propositions,

(*i*) There should be a smooth blending as the force begins to act, if possible,

(*ii*) There should be a rapid increase in force as the joint limit is approached, and,
(*iii*) No force needs to be acting if the movement is away from the joint limit.

Preliminary thoughts included the use of the exponential spring,

$$f = \exp(c|x|) - 1$$

where the constant c controls how fast the exponential curve will rise. To avoid having to determine appropriate constants for each degree of freedom these constants are replaced in AnthroPI with a variable dependent upon the mass distribution of the link distal to the degree of freedom, although, they can also be controlled by a proportionality factor input by the user. Further, a better way of controlling the movement is, e.g.

$$f = k\ (\exp(c|x|)-1)$$

where k controls linear strength, and c controls exponential rise.

The natural range of a particular joint needs to be first determined and then torques are applied about the joint to keep it in this range. Anthropometric data provided the physical range in which joints can move and to ensure that the joint limits are not exceeded, exponential springs are employed to constrain the movement.

In Figure 8.2 the control function of an exponential spring is shown, where $f = k\ (\exp(c|x|)-1)$ represents the component of the force that needs to be exerted to keep a particular joint within its physical range, i.e. it defines the magnitude of the force rather than its direction, and x represents the angle of the joint.

Exponential (and linear) springs invalidate proposition (*i*), and further experiments were carried out to investigate the behaviour of the x^n functions. However, as the spring needs to be applied at the beginning and the end of the joint's physical range, at the middle of this range, i.e. a 'rest' period, no force needs to be exerted, and these functions need to be explicitly defined to follow this requirement.

A possible solution can be found by specifying the spring force with the following composite control functions defined at the given intervals, see also Figures 8.3a and 8.3b, by assuming that α and β are the lower and upper limits respectively.

Recovery phase towards the minimum joint limit - Compression (x decreasing)

$$f = \begin{cases} k(\alpha+\pi/8-x)^n \quad \underline{or} \quad k[\exp(c(\alpha+\pi/8-x))-1] & \alpha \le x < \alpha+\pi/8 \\ 0 & \alpha+\pi/8 \le x \le \beta \end{cases}$$

Elongation towards the maximum joint limit - Tension (x increasing)

$$f = \begin{cases} 0 & \alpha \le x \le \beta-\pi/8 \\ k(x-\beta+\pi/8)^n \quad \underline{or} \quad k[\exp(c(x-\beta+\pi/8))-1] & \beta-\pi/8 < x \le \beta \end{cases}$$

The parameters k or c, and n need to be defined. In our system these are specified under user control. Early work has tested the case where n=3 and assumed that k and c are some small positive integers. However, these control how fast the force will increase and can be changed by the user. At an early stage, the estimated $\pi/8$ region of action for all the angles is assumed, although, this can be accurately controlled by closer examination of anthropomorphic data. Further, the values of the joint limits, α and β, are determined individually for each joint, and these are stored and read from configuration files which can be altered by the animator, but all the joints are assumed to follow an area of action in proportion to $\beta-\alpha$.

The power functions have the functionality to achieve smooth blending and they offer rapid increase in the force as the joints limits are approached. However, the use of exponential springs for a small range of possible motions is in some cases satisfactory.

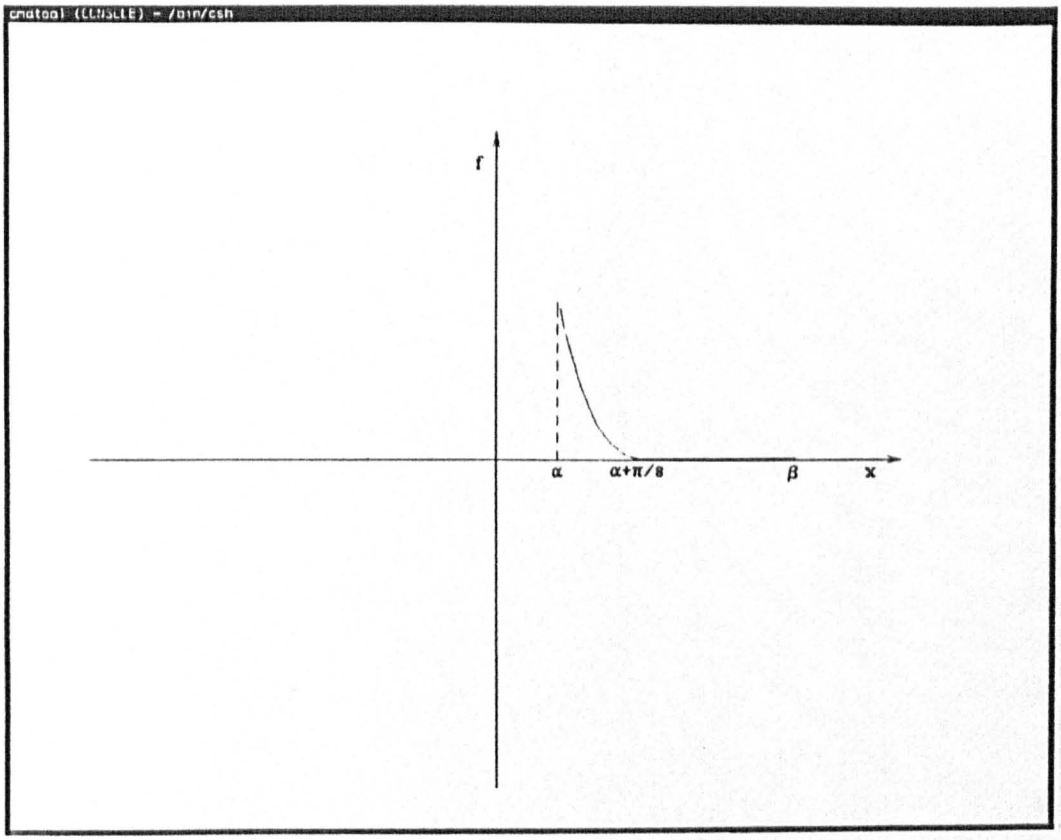


Figure 8.3a

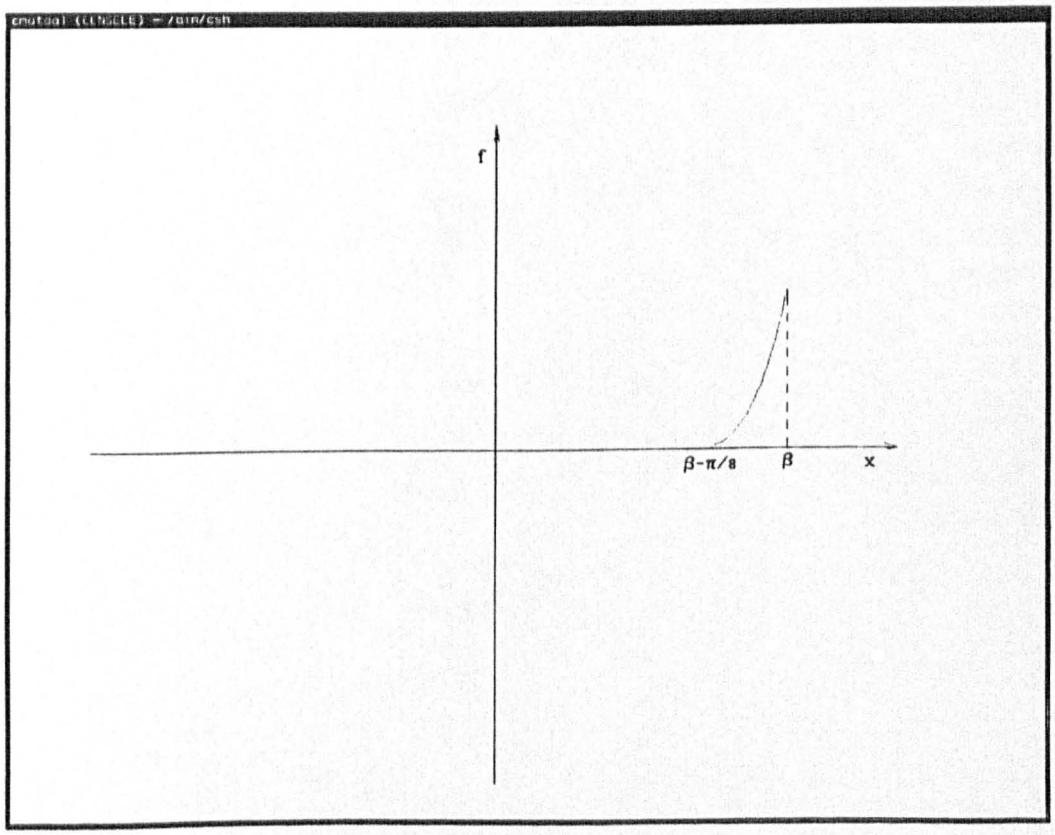


Figure 8.3b

8.3 User Interface

Because AnthroPI is an experimental system the user interface needs to be both easy to use and easy to modify. For these reasons a simple interface consisting of abbreviated commands and window-based menus was developed.

The SunCore package, based on the ACM Core, was mainly used for building the interface. Built in the Sun workstations, SunCore allows its users to create and manage a graphics environment. SunCore can be initialised to suit the requirements of the user, that is, drawing surfaces, input devices, viewing operations and simple commands for creating images.

The AnthroPI screen is divided into three sections. The left half of the screen is a window used for display; it initially uses the SunView package to display a still image, and it uses SunCore to simulate the model. The upper right hand side is a command menu, for creating the image, defining the DOF and for input/output. The lower half consists of a window for describing each degree of freedom, see Figure 8.4. A play-back utility, for playing succeeding frames faster is also available, which runs continuous motion. This requires the storage of each frame in a different file so that these files can be called up and displayed on the screen at any pace. A sequence of 18-22 frames per second gives the illusion of a continuous movement, although this depends largely on the capabilities of the available hardware.

Figure 8.4 : Screen Dump of AnthroPI

8.4 CONCLUSIONS

This chapter has been concerned with the implementation details of the system AnthroPI. The articulated body hierarchy is read from files which include information about the construction of local coordinate systems, inertia tensors, masses and orders of rotation for each link. The user can change these quantities interactively using the interface environment. This offers the facility of changing the root of the hierarchy on-line, which is very useful if an animated walking sequence is to be created, the theme of the next chapter. After the root of the structure is changed, the tree representation is redefined, i.e. the connectivity which is stored in a two-dimensional table is changed, so that the way the tree is traversed remains the same.

AnthroPI is sufficiently general to model any arbitrary articulated body capable of being represented as a tree structure. Figures used for dynamic analysis have generally been capable of 42 degrees of freedom or fewer, because of the high cost of dynamic analysis and the limited computer resources. Also, at this stage, we are more concerned with the correctness and realism of the movement, i.e. motion quality, whereas the geometric modelling of the static figure is beyond the purposes of this project.

AnthroPI is entirely written in the C programming language and uses the SunCore and SunView graphics packages. It runs under Berkeley Unix 4.2 and 4.3 BSD on a network of Sun 3/50-60 and Sparc workstations with Sun operating system 4.1.1, and it is totally interactive. It provides configuration files for initial estimates and menus for selecting and entering data and communicating with the environment as a whole. An on-line help facility is available.

CHAPTER 9
THE GAIT CONTROLLER

9.1 OVERVIEW

> *'... when one is walking rapidly each step takes no more than half a second and in that half second no fewer than fifty-four muscles are set in motion... I at once directed my attention to my legs and tried to discover the infernal machine. I thought I had succeeded in finding it. I could not of course distinguish all its fifty-four parts, but I discovered something terrifically complicated which seemed to get out of order directly I began thinking about it ...'*
>
> *Italo Svevo, Confessions of Zeno*

The problem of creating an animated motion sequence is quite complicated. Activities of articulated bodies, such as humans, are intricate mainly due to the complexity of the model, which has a large number of joints each with several degrees of freedom and interactions between these joints controlled by muscles. As has been discussed in the earlier chapters, simplifying the form of the model is an attractive approach to this complex situation, but this is likely to have a detrimental effect on how natural the model will look, and how realistic its motion will be.

A possible approach, to the research of human figure locomotion, is to combine work on the study of animals (biological systems), Alexander [61], and the construction of machines and control theory (laboratory robots), as discussed by Raibert [60, 68], Craig [66] and Vukobratovic et al [45]. Although the fields are related, the problems involved are not identical.

The study of *biological systems* proves to be much more complicated than one would expect. It is difficult to formulate experiments that will include and control the numerous variables involved.

On the other hand, simple *laboratory robots* are easy to build. Experiments with precise control are possible when careful measurements and manipulations are used. They are easy to study, and the style of their motion is not relevant.

The models used in animation are more complex than most robots and, therefore, present more difficult problems. Moreover, in animation the style

of motion is important, as the eye is unwilling to accept even mildly unrealistic motion for bodies as familiar as humans and animals.

Analysis of living systems and synthesis of laboratory systems are complementary activities. In solving the problem for robotics, we can generate a set of plausible algorithms for the biological system. On the other hand, in understanding the biological behaviour, we can enhance the control of the behaviour for the machine.

We have already considered the concept of ground reaction forces (Chapter 7) which was aided by research in legged machines. The idea was drawn from research in robotics which led to the development of a one-legged hopping machine by Kearney et al [48] at the University of Iowa.

A similar approach has been followed for the development of a locomotor-gait controller. Data from biomechanics, prosthetics and anthropometry [41-44] provided the original estimations and simplifications of motion, which were combined with the use of Featherstone's direct dynamics algorithm, borrowed from research in robotics, to develop a model for human walking.

The most serious problem with dynamic motion specification is *motion control.* To allow the user to specify motion, under internal muscular control, a reliable, convenient, and positionally-based method must be found. For reasons of clarity, we can split the problem into distinct, but related, levels of control which are described in the next section.

9.2 LEVELS OF CONTROL

Knowledge about a dynamically-based animation cycle could be considered at three levels : low, middle and high. The lowest level should calculate the joint angles using the dynamic equations of motion. This would then need higher levels of control, where the values of the locomotion parameters could be determined. At the top level, the form of the motion should be specified easily and conveniently using a flexible set of movement commands that generate a variety of motions. The various *levels* could be specified as follows.

At the *lowest level*, the dynamic equations of motion need to be formulated and solved. The motion is described directly in terms of the generalised coordinates and, depending upon whether an inverse or direct dynamics problem is being solved, either the accelerations or the forces and torques need to be specified explicitly.

The *middle level* is responsible for the specification of gait-specific rules. It determinates the locomotion parameters that 'guide' the dynamic simulation of the limbs, it employs the gait determinants, their attributes and rules associated with them. It is used to define and adjust the character of the movement, especially the movement of the legs and feet.

The *high level* is responsible for the creation and coordination of the motion. At this level the motion is specified generically, employing everyday concepts such as the style of a specific gait (i.e. determining how the motion will look), which automatically define values for the parameters used at the middle level. For this a good user interface is essential.

Thus, a high-level description of walking could be 'walk happily' or 'walk vigorously'. These commands would be translated automatically into the corresponding parametric descriptions of the motion, so that even a non-expert user could control the movement. Of course, the animator will always be able to adjust the form of motion by specifying parameters at the middle level, which will then be translated into the lower level of control, for the solution of the dynamic equations of motion.

Figure 9.1 gives a structural outline of the levels of control as they are described above. Such a high-level control system inevitably involves considerable complexity, and in the literature associated with this field of study, no-one as yet has tried to implement such a system.

This chapter is particularly concerned with the design and implementation of the middle level of control, i.e. describing the motion parametrically, which is outlined in Section 9.4. However, having studied the necessary background theory and implemented the middle level, it is now feasible to create such a high-level control system which, has to be pointed as a goal for future development.

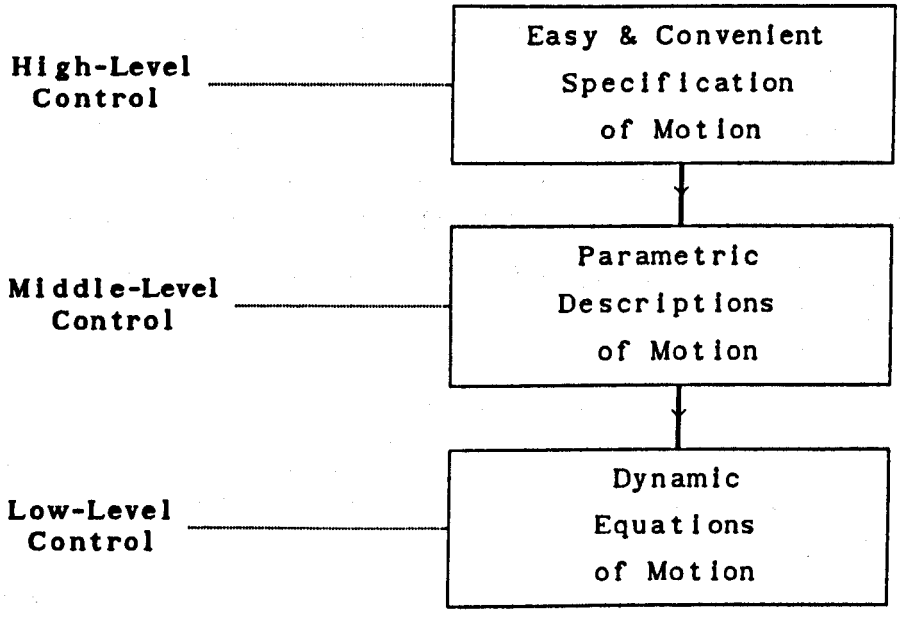


Figure 9.1 - Levels of the Control Hierarchy

9.3 A GENERIC LOCOMOTION PATTERN

9.3.1 Introduction

Legged locomotion is the activity where body translation results from rotational movements in the lower limbs. Certain problems need to be addressed, such as the coordination of the legs, the balancing of the upper body with respect to the movement and the correct timings of the individual leg motions, Bruderlin & Calvert [47].

To reduce the amount of detail necessary to define a motion, a system should possess *goal-directed control* so that tasks can be specified, at a high level, in order to produce fine control of the style of motion. Our initial attempts at an implementation described in the hierarchical model, Figure 9.1, have concentrated on walking, for which the parameters are quantities such as velocity, step-length, step-frequency, and a variety of gait determinants.

9.3.2 Human Locomotion

Motion is an important activity of our every-day life. But the manner of motion, the '*how*' is a question that is asked very rarely.

All people move constantly, though few realise how complex their movements are, and even fewer stop to analyse how these movements come about. Yet it is through their movements that people have the means to interact with their environment, express their feelings and relate meaningfully to one another.

Human locomotion is a phenomenon of most extraordinary complexity. Because of its complexity, the study of human movement in all its forms may be conducted from many points of view. We consider five major theoretical approaches, Galley & Forster [69], to be of immediate concern. They are :

1. *Anatomical* - which describes the structure of the body and its parts and their potential for movement,

2. *Physiological* - which studies the processes involved in the initiation, continuation and control of movement,

3. *Mechanical* - which considers the force, time and distance relationships involved in body movement,

4. *Socio-cultural* - which considers the meanings given to various movements in different human settings, and

5. *Psychological* - which examines the sensations, perceptions and motivations that stimulate movement and the neurological mechanisms which control them.

The idea is that these relevant areas be linked up, whenever possible, so that a wider appreciation of movement and its control is achieved. If the idea is followed up, the concluding remark '*Art and Science meet when they both seek accuracy*', Etienne-Jules Marey (1988), becomes a desirable reality. However, the approaches that are of direct concern here are the mechanical and the physiological.

Human locomotion produces the translation from one point to another by means of a bipedal gait. So many individual motions occur simultaneously that analysis is difficult without some unifying principle. The gait determinants concept and the different parameters involved need to be introduced and examined, so that the higher levels of control can be incorporated.

We, therefore, need to study carefully the primary determinants of human locomotion, as well as record and measure the magnitudes, directions and rates of change of the translations, rotations and forces occurring in the body with respect to the three-coordinate axes in space. These will be the main topics of discussion of the next sections.

9.4 HIGHER LEVELS OF CONTROL IN HUMAN WALKING

9.4.1 Introduction

Walking is a smooth, highly-coordinated, rhythmical, symmetric movement in which the body moves step by step in the required direction at the necessary speed. It incorporates the concepts of *gait*, the manner of walking, and of *locomotion*, the act of moving from place to place.

One of the chief attributes of a normal gait is the wide range of safe and comparatively comfortable walking speeds available. Speed is variable as a person hurries, hesitates, stops or starts. All gaits have certain common characteristics. The lower extremities provide the major action in walking, with additional involvement of the head, trunk and arms.

For simplification, walking is broken up into two phases; the stance and the swing phase, and these are further broken up into subphases. The *stance phase* is the period of double support where both feet are on the ground, whereas the *swing phase* is the single support state where one foot is off the ground. Accordingly, since each lower limb goes through a stance and a swing phase, we can define a stance and a swing leg. In walking, one foot is in contact with the ground at all times.

For bipedal walking, a locomotion (or gait) cycle consists of a step each by the right and left lower limbs. That is, the limbs pass through both the stance and the swing phases and return to their original relative position at the beginning of the cycle. The stance and swing phases of a locomotion cycle are examined separately, which greatly simplifies the control as well as the numerical integration process. For practical purposes, this can be reduced to one step if a symmetric gait in a straight line is assumed, with left and right sides performing symmetric movements shifted in time.

The high-level concepts need to be applied before the impending step, whereas, low-level motion control takes place during the step. In this way, locomotion parameters can be changed from step to step. The dynamics formulation is subject to the step constraints and produces the desired motions during a phase by applying rules about walking directly at the low level. This is the reason why phases are further divided into subphases.

In the following sections, we examine in more detail how this control hierarchy has been implemented in AnthroPI.

9.4.2 Control Principles

The execution of the different components in AnthroPI is based on the following four principles.

1. The control hierarchy as illustrated in Figure 9.2 is applied to each step of a walking sequence where a step is defined as the double plus the single support state (see also Figure 9.3). High-level instructions are decoded before the impending step, and low-level actions are executed during the step. The system offers the capability of changing the locomotion parameters from one step to the next.

2. The dynamic equations of motion guide the lower body kinematics, but the kinematic computations, such as the gait determinants, may affect the dynamics. Of course, it is always the case that dynamics is at the very heart of the control because it guarantees natural-looking rotational movements of the legs and of the whole hierarchical structure of the body, though it is aided by the use of kinematic principles.

3. The upper body movements follow and depend upon the lower body movements. For example, the movements of the torso and arms need to be expressed relative to the movements of the legs (e.g. right arm swings forward at the same time as the left leg is pushed forward). This is the reason why the root of the structure needs to be changed to the corresponding foot in contact with the floor before starting the dynamic analysis.

4. The stance and swing phases of each individual leg are considered separately which greatly simplifies the control. Therefore, for each leg, the simulation of the stance phase is executed first followed by the swing phase dynamics. The symmetry of steps is applied here for the next dynamics calculation. Note that, while one leg is in the swing phase the other is in the stance phase.

Upon definition of the above specific rules, the concept of *determinants of gait* needs to be introduced. Although there is no unique convention in describing the motions of the links during walking, one description given in 1953 by Saunders et al [67], and discussed by McMahon [27, 58], involves the consideration of six major determinants which are related to the function of the hip, knee and ankle during walking. Theoretical definitions of the gait determinants based on Saunders' work are introduced in the following section.

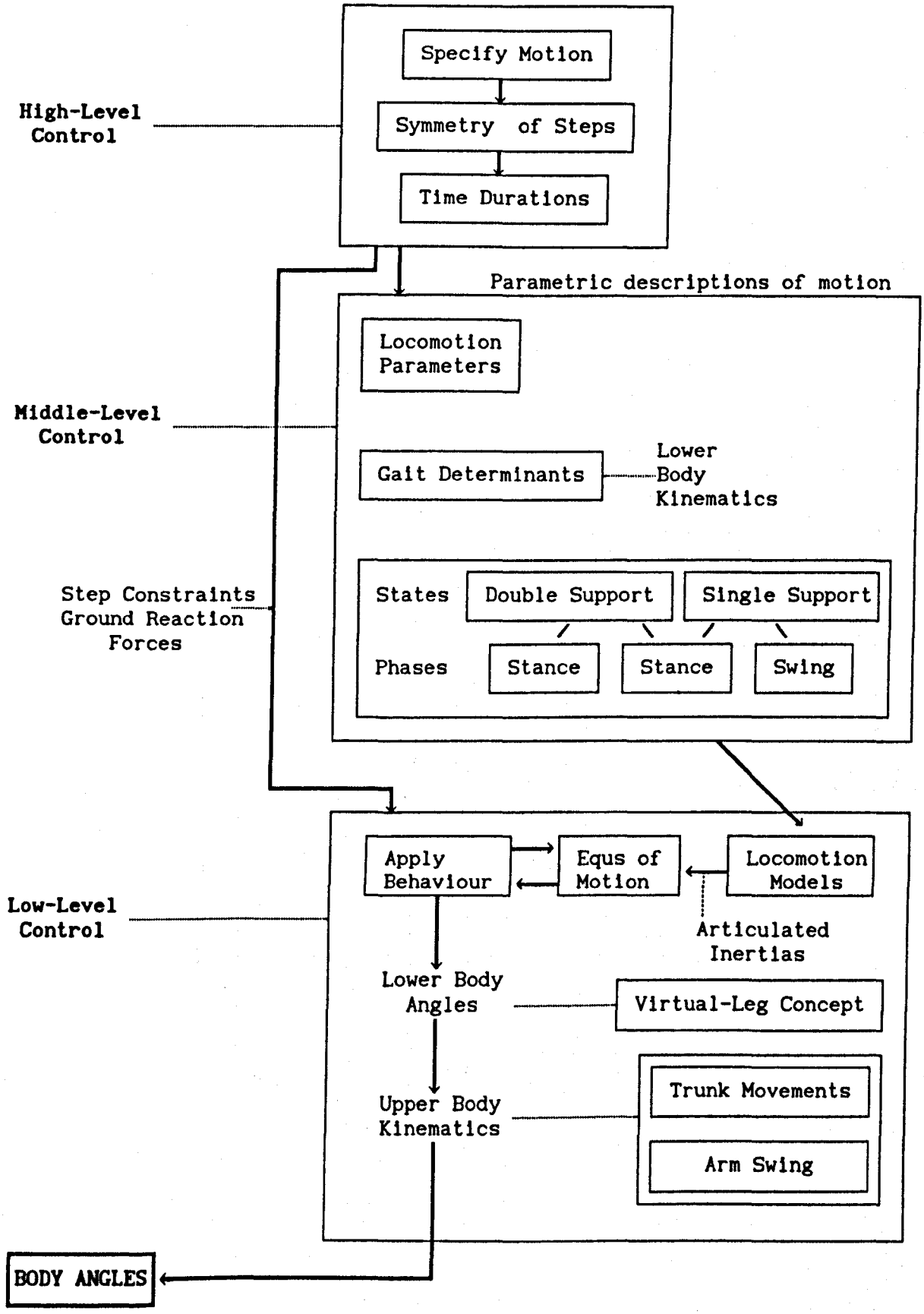
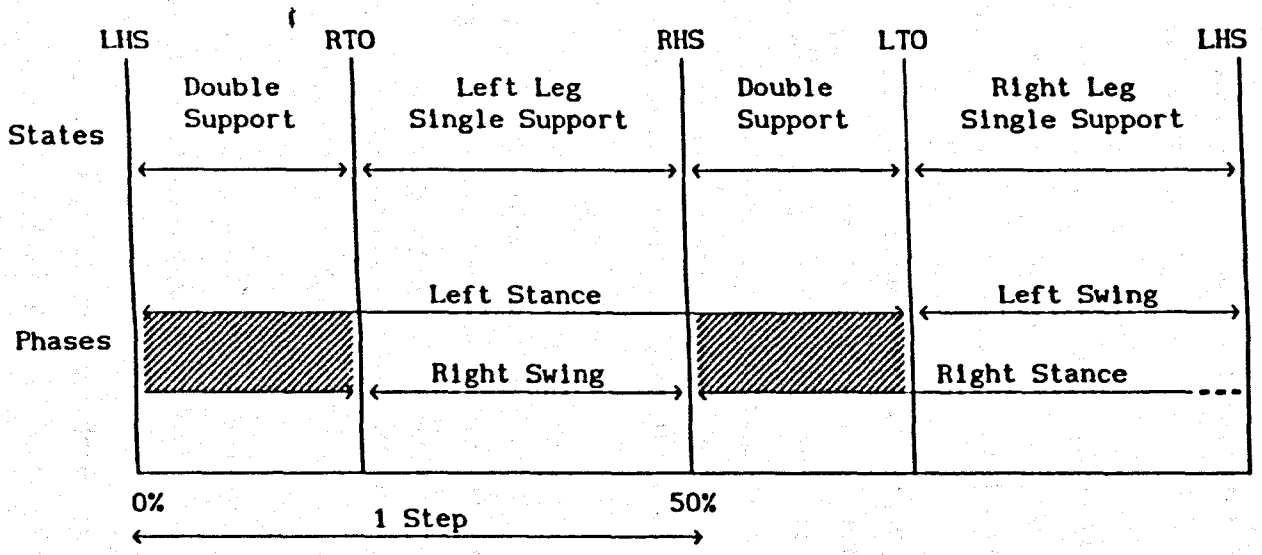


Figure 9.2 : Levels of Control Hierarchy in AnthroPI



LHS = left heel strike RHS = right heel strike

RTO = right toe off LTO = left toe off

Figure 9.3 : Locomotion cycle for bipedal walking

9.4.3 The Determinants of Gait

The gait determinants mainly describe the movements of the pelvis during locomotion. Careful observations of these determinants provide insight into individual variations of the gait.

For analytical purposes, we also consider the behaviour of the centre of gravity in a bipedal model, in which the lower extremities are represented by rigid limbs. This knowledge is needed to link the ground reaction forces with the concept of gait determinants.

The major determinants in normal gait of human locomotion are : pelvic rotation, pelvic tilt, knee motion during the support phase, foot and ankle motion, knee motion during the swing phase, and lateral motion of the pelvis.

Pelvic Rotation. The pelvis rotates alternately to the right and to the left at each hip joint, relative to the line of progression (approximately 4 degrees on either side of a vertical axis with respect to the upper limb of the stance leg). The effects of pelvic rotation are increased length of the leg (it extends the swing leg in the direction of motion) and therefore longer step length and greater radius of the arcs of the hips, resulting in a smoother ride.

Pelvic Tilt. The pelvis tilts downward on the side opposite the weight-bearing leg. This involves a lowering of the swing leg at the hip joint and knee flexion of the swing leg to allow clearance for the swing-through of that leg, preventing it striking the ground as it moves forward. The centre of gravity of the body is lowered.

Knee Flexion during the Support Phase. A characteristic of human locomotion is the passage of the body weight over to the supporting leg while the knee of that leg is undergoing flexion. This is referred to as the period of '*double knee lock*' of the stance leg, since the knee of that leg is first locked in extension, unlocked by flexion, and again locked in extension prior to its final flexion.

Foot and Ankle Motion. This involves the transition from double-support phase to swing phase by smoothing out the path of the centre of gravity. It

comprises the relation of the rotation of the ankle about a radius formed by the heel, and the rotation of the foot about a centre established at the fore part of the foot in association with heel rise.

Knee Flexion during the Swing Phase. At heel contact the knee joint is fully extended, so that the extremity is at its maximum length and the centre of gravity reaches its lowest point in downward displacement. Rapid plantar flexion of the foot, associated with the initiation of knee flexion, maintains the level of the centre of gravity. The termination of this is associated with flexion of the second knee at heel rise.

Lateral Displacement of the Pelvis. The body rocks from side to side since the weight bearing is alternately transferred from one limb to the other. The centre of gravity is displaced laterally over the weight-bearing leg, twice during the cycle of motion.

The three determinants of gait : pelvic rotation, pelvic tilt and knee flexion, all contribute to the flattening of the arc through which the centre of gravity of the body is moved. Pelvic rotation elevates the extremities of the arc, whereas pelvic tilt and knee flexion depress its summits. The deviations of the centre of gravity in the horizontal and the vertical planes are almost equal.

The relative lengthening of the extremities considerably reduces the range of flexion and extension, at the hip joint, required to maintain the same length of stride. It also plays an exceedingly important role in permitting increased velocities of gait, since greater velocities of locomotion are achieved by the lengthening of the stride rather than by increases in cadence.

Individual variations in locomotion are due to exaggerations in one, or another, of the six determinants. Owing to the interactions between the various factors, exaggerations in the range of one determinant are compensated for by reductions in another, so that the final pathway of the centre of gravity remains essentially the same in that it is the most economical to maintain.

The implementation details of the application of these determinants is further discussed. We have experimented with the application of subsets of

these parameters, building on the early work of Vasilonikolidakis & Clapworthy [35].

To construct a model which corresponds as closely as possible to the results produced from physiological and photographical studies, as well as biological research, the gait determinants and the correct time durations were introduced and included in the implementation of the dynamic analysis.

The gait determinants taken into consideration included the pelvic rotation, pelvic tilt, knee flexion during the support phase and knee flexion during the swing phase.

In particular, the motion of the trunk was determined by the rotations that propagated from the root of the motion to the kinematic chains, i.e. the torso and the lower body, in addition to the application of the gait determinants. These were introduced in the model as kinematic constraints if the motion of the anthropomorphic figure was outside the specific, allowable, range. Upper and lower limits were employed. In this way, the gait determinants restricted the motion to the specified limits. Similar techniques were employed for the motion of the legs, e.g. the knee angle, and the motion of the arms.

In particular, implementation of the pelvic rotation involved the left and right rotation at the hip joint where the pelvic angle was in the predetermined range of -4..4 degrees. Similarly, the pelvis was allowed to tilt on the side opposite the weight-bearing leg (-5..5 degrees), and an internal force was applied to generate these movements. The rotation of the pelvis is a maximum at foot contact and a minimum at mid-step, whereas the pelvis tilt is a maximum at mid-step and a minimum at foot contact. Linear interpolation is applied to obtain all the intermediate angles, which is justified since the absolute displacements produced by the determinants are rather small.

Knee flexion during the support phase introduced a kinematic constraint at the knee angle which made the leg flex, lock and flex again. The angle of the knee flexion was dependent upon the step length or the step frequency of the particular gait. For example, a larger step length resulted in a larger knee angle. Knee flexion during the swing phase was necessary in maintaining the level of the centre of gravity and depended upon the step length or the step frequency.

9.4.4 Time Durations and Locomotion Parameters

Walking can be varied by changing the values of the step length and step frequency

$$\text{velocity} = \text{step\_length} \times \text{step\_frequency}. \qquad (9.1)$$

To specify conveniently a desired locomotion we need to define two of the three locomotion parameters : velocity, step length and step frequency. We therefore need to transform the three locomotion parameters into the step constraints for low-level control. Experimental results, Inman et al [52], relate walking speed to the time needed to perform a cycle. Correct time durations of a locomotion cycle have also been calculated by Bruderlin & Calvert [47] and Boulic et al [26].

Experimental data [52] shows that,

$$\text{step\_length} = 0.004 \times \text{step\_frequency} \times \text{body\_height} \qquad (9.2)$$

and therefore

$$(\text{step\_frequency})^2 = \text{velocity} \;/\; (0.004 \times \text{body\_height}). \qquad (9.3)$$

It has also been deduced that the $\text{step\_frequency}_{max}$ = 182 steps per minute. From the step frequency, we can calculate the time for a cycle

$$t_{cycle} = 2 \times t_{step} = 2 \;/\; \text{step\_frequency}, \qquad (9.4)$$

and the times for the double support, t_{ds}, and single support phases (see Figure 9.3), since,

$$t_{step} = t_{stance} - t_{ds} \quad \text{and} \qquad (9.5)$$

$$t_{step} = t_{swing} + t_{ds}. \qquad (9.6)$$

Furthermore, from experimental results [52],

$$t_{ds} = (-0.16 \times \text{step\_frequency} + 29.08) \times t_{cycle} \;/\; 100. \qquad (9.7)$$

In this fashion, the step frequency is used together with symmetry of steps to calculate the time for each phase of a step. This will sufficiently represent the movements of one cycle.

9.4.5 Virtual-Leg Concept

In Chapter 7, the method developed by Kearney et al for calculating the ground reaction forces, was discussed. This formulation was for a composite single body where only one point of contact was assumed.

During the double support phase of bipedal walking, both legs are in contact with the ground. The method thus has to be adapted to deal with the situation in which two points of contact are active. For this it was decided to create a *virtual leg.*

The virtual-leg approach represents the pair of legs acting in unison and is implemented during the double-support phase. Similar proposals have been made by Raibert & Hodgins [49] and Raibert [68], who also refer to work by Sutherland [70].

The two physical legs behave like one virtual leg, located between them. The forces and torques exerted on the body by the two physical legs and by the virtual leg are equal, so the behaviour of the body is the same in both cases.

For the implementation of the virtual-leg concept, a set of operations needs to be defined : positioning, synchronisation and force equalisation.

Force-equalisation for (e.g.) the double-support phase, locates the virtual leg between the two physical legs that it represents. It makes the resulting behaviour simple to analyse and similar to that of the one-legged systems.

Positioning places the physical feet which determine the location of the virtual foot.

Synchronisation ensures coordination of the behaviour of the physical legs.

The virtual-leg approach discards the passive stability that two legs can provide, in favour of active stability. In this fashion, the control system must coordinate the low-level behaviour of the physical legs and must provide locomotion algorithms that specify the desired behaviour for the virtual leg. This is summarised in Figure 9.4.

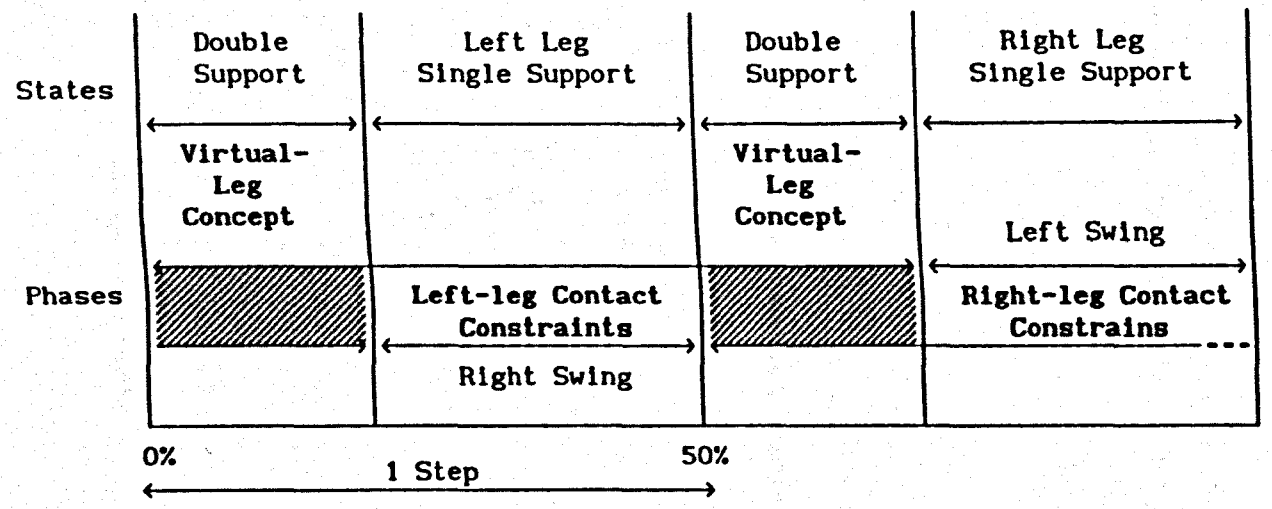


Figure 9.4 : Virtual-Leg Concept

Thus from Eq. 7.2, if we constrain the desired angular momentum at the stance phase to be equal to the original angular momentum of the body at that period, ($\tilde{\underline{H}}_G - \underline{H}_G = 0$) then, only the parallel component of the contact force exists, governed by Eq. 7.1,

$$\underline{f}_c \cdot \frac{\underline{r}}{|\underline{r}|} = k_1 (|\underline{r}| - r_s).$$

Again, as in Eq. 7.1, the spring length r_s is defined to be shorter than $|\underline{r}|$, so that the contact force is directed away from the point of its application.

In Figure 9.5, $\underline{r}_r$ represents the virtual leg of the right leg and similarly, $\underline{r}_l$ the virtual leg of the left leg.

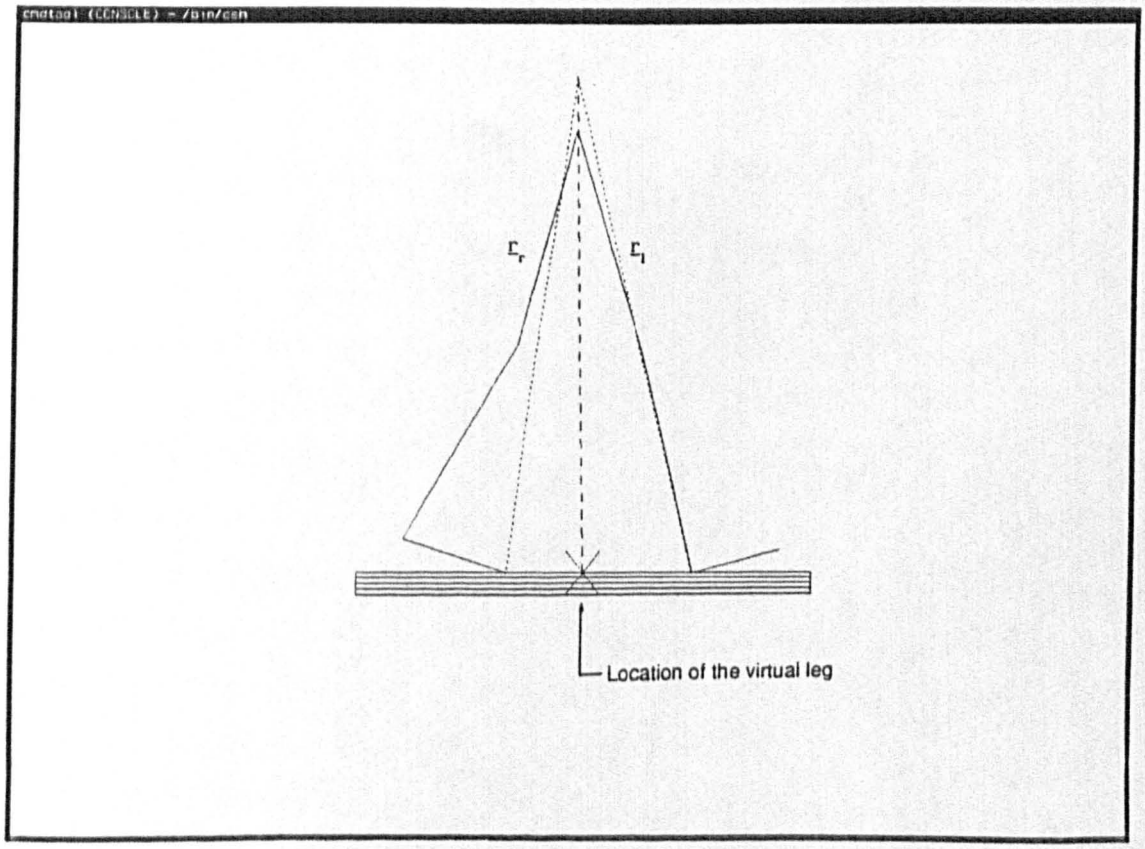


Figure 9.5 : Double-support phase

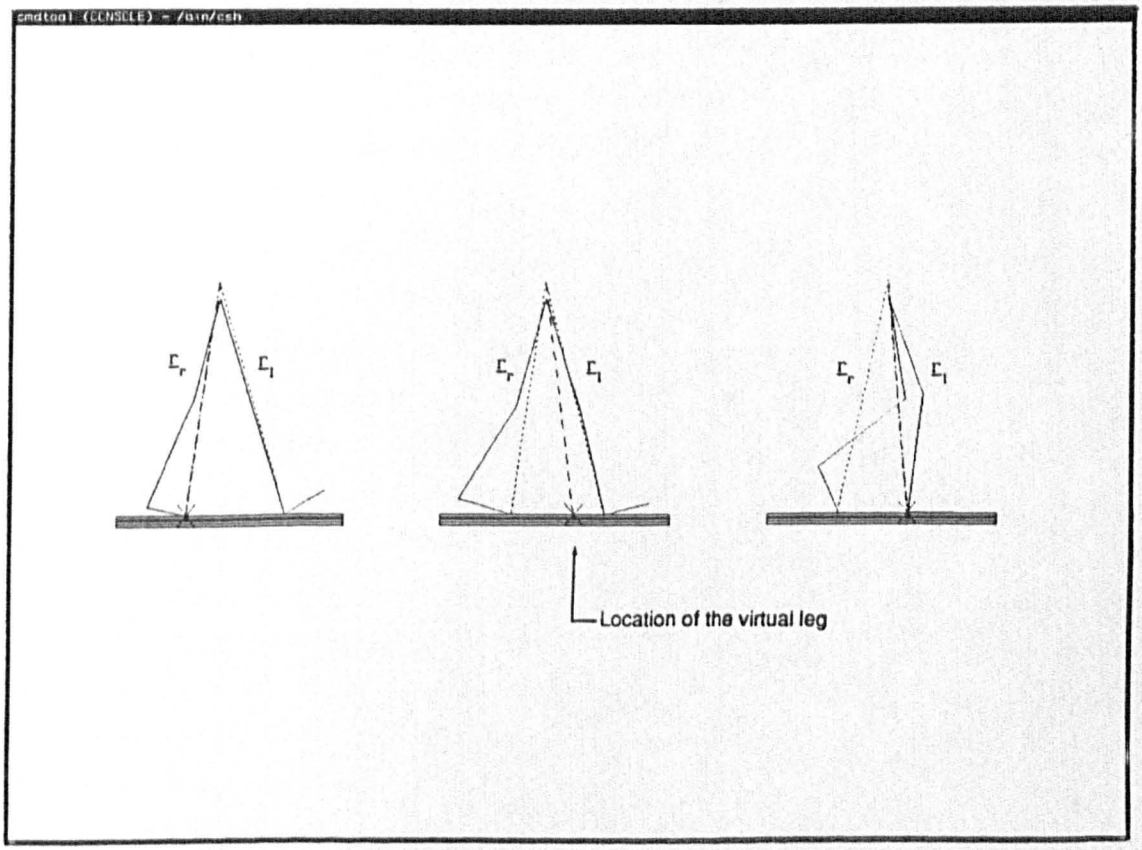
To achieve a smooth positioning of the virtual leg from the double to the single support phase, the following expression is introduced which qualifies a continuous function. This relates the positions of the left and right physical legs, that is,

$$\underline{r} = \alpha \underline{r}_l + (1-\alpha)\underline{r}_r$$

where the coefficient α, takes values in the range

$$0 \leq \alpha \leq 1.$$

Further, α is time dependent and its value changes linearly with time.



(a) (b) (c)

Figure 9.6 : The Virtual Leg - Two Legs in Contact

Figure 9.6 illustrates the position of the virtual leg when the two physical legs are in contact with the ground.

In Figure 9.6(a) the left leg is ready to touch the ground and become the stance leg whereas the right leg is in contact with the ground. The virtual leg is positioned at the right leg and therefore $\alpha = 0$.

Similarly, transition from the double-support position to the case where only the left leg is in contact with the ground, Figure 9.6(c), will result in having the position of this leg identical to the position of the virtual leg.

Transition from the single support phase to the case where both legs are on the ground will result in an inbetween position, Figure 9.6(b) and this can be further interpolated to consider intermediate stages.

Having α linear in time is an easy approach, but more complicated cases where the time relationship is controlled by a cosine or a cubic function need to be further investigated as these could ensure a smoother transition from the single to the double-support phase.

9.5 IMPLEMENTATION DETAILS : A HYBRID DYNAMICS ALGORITHM

Having described, extended and modified Featherstone's direct dynamics algorithm, as well as examined all the concepts related to walking such as the gait determinants, the walking parameters together with the time durations, we are now in a feasible position to produce an animated, walking sequence.

Our original attempt, at creating such a sequence included only the direct dynamics algorithm and the principle of calculating the ground reaction forces. However, deeper examination of this idea showed that finding and coordinating the forces required to cause a specific motion is a very unnatural and difficult problem. Manipulating the forces and torques acting on the body directly will not be a feasible control mechanism until further understanding of the sizes and variations of these quantities required to produce specified forms of motion is acquired.

Furthermore, using the direct dynamics algorithm requires the user to specify a force, or torque, for each degree of freedom that is internally controlled, as well as predict what force or torque will produce the desired motion.

To ease the problem of specifying motion control, a combination of direct and inverse dynamics motion specification, HIDDS - (Hybrid Inverse and Direct Dynamics System), was examined for the determination of initial estimates of the forces that could drive the desirable motion. Once the strength of the forces necessary to drive a specific movement is estimated using HIDDS, some experimentation is required for the creation of a motion sequence so that direct dynamics can determine the exact final positions of the model, resulting from the application of these forces.

Assuming that the desired local positions for part or the entire body are known, either from anthropometric data, or from Muybridge's photographic sequences, or from any other source (e.g. Inman's analysis), inverse dynamics can be applied to the body to find controlling forces/torques, which can be used with direct dynamics on the entire body to find the total body motion.

Using this method, inverse dynamics provides an initial control of forces and torques on the body (starting when the body is still), and direct dynamics ensures that the desired output is the production of realistic motion.

The articulated-body algorithm has been further modified to solve this kind of problem. Description of the reformulation of the algorithm will now follow.

It has been shown in previous chapters that if the forces are given, then the accelerations can be determined (Section 6.3); but if the accelerations are given and the forces are to be determined, then the equations need to be amended in order to calculate these forces. The equations for solving the inverse dynamics problem are given below, where the quantities are as defined in Chapters 4, 5 and 6.

In Eq. 5.2, we deduced that the relationship between the force applied to a particular member of an articulated body i and its acceleration is given by,

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

$$= \hat{\underline{I}}_i \hat{\underline{a}}_i + \hat{\underline{p}}_i^v + \hat{\underline{I}}_{i+1}^A (\hat{\underline{a}}_i + \hat{\underline{v}}_{i+1} \hat{\times} \hat{\underline{s}}_{i+1} \dot{q}_{i+1} + \hat{\underline{s}}_{i+1} \ddot{q}_{i+1})$$

$$+ \hat{\underline{p}}_{i+1} \qquad (9.8)$$

if Eq. 5.23 is used. So collecting the terms $\hat{\underline{I}}_i^A$ and $\hat{\underline{p}}_i$ we have,

$$\hat{\underline{I}}_i^A = \hat{\underline{I}}_i + \hat{\underline{I}}_{i+1}^A.$$

Using the adaptations described in Chapter 6 yields,

$$\hat{\underline{I}}_i^A = \hat{\underline{I}}_i + \sum_{j \in \mu_i} {}_i\hat{X}_j \, \hat{\underline{I}}_j^A \, {}_j\hat{X}_i. \tag{9.9}$$

Similarly, from Eq. 9.8, we have,

$$\hat{\underline{p}}_i = \hat{\underline{p}}_i^v + \hat{\underline{p}}_{i+1} + \hat{\underline{I}}_{i+1}^A \, (\hat{\underline{v}}_{i+1} \times \hat{\underline{s}}_{i+1} \, \dot{q}_{i+1} + \hat{\underline{s}}_{i+1} \, \ddot{q}_{i+1}) \tag{9.10}$$

which results in Eq. 9.11 (shown below) if externally applied forces, such as the ground reaction forces are to be included. The calculations of these forces are as given in Chapter 7. All quantities are as described in earlier chapters.

$$\hat{\underline{p}}_i = \hat{\underline{p}}_i^v - \hat{\underline{f}}_i^{ext} + \sum_{j \in \mu_i} {}_i\hat{X}_j \, [\, \hat{\underline{p}}_j + \hat{\underline{I}}_j^A \, (\hat{\underline{c}}_j + \hat{\underline{s}}_j \, \ddot{q}_j) \,]. \tag{9.11}$$

Eqs. 9.9 and 9.11 are used in place of Eqs. 5.20 and 5.21, and these need to be calculated with recursive iterations from the end-effectors to the base link, to determine the articulated-body inertias and the bias forces, in turn.

Furthermore, in order to calculate the absolute accelerations $\hat{\underline{a}}_i$ for each link, there is no need to calculate the $\ddot{q}_i$'s first, as these will now be given, or can be calculated by double differentiation (over time) of the input positions. Therefore, the absolute accelerations, Eq. 5.23, are calculated by Eq. 9.12 below, which requires the iterations to start from the base to the end links.

$$\hat{\underline{a}}_i = {}_i\hat{X}_{\lambda_i}\, \hat{\underline{a}}_{\lambda_i} + \hat{\underline{v}}_i \times \hat{\underline{s}}_i\, \dot{q}_i + \hat{\underline{s}}_i\, \ddot{q}_i. \qquad (9.12)$$

The base acceleration will be determined from the articulated-body inertia of the whole body and the force required to produce zero base acceleration, as shown in Eq. 6.9,

$$\hat{\underline{a}}_0 = (\hat{I}_0^A)^{-1}\, (\hat{\underline{f}}_0 - \hat{\underline{p}}_0). \qquad (9.13)$$

To calculate the acceleration of the base, the force acting on it has to be supplied by the user. The base force is the only force that needs to be provided by the user. Once the base acceleration is known we can calculate all the $\hat{\underline{f}}_i$'s, so it remains only to calculate the joint forces which are given by,

$$Q_i = \hat{\underline{s}}_i^S\, \hat{\underline{f}}_i \qquad (9.14)$$

already introduced as Eq. 5.12, where

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

Therefore, successive iterations of Eq. 9.14 from the base to the end-effectors will determine the forces acting on all the links.

That is, given a partial description of the desired behaviour of our model, we must determine forces which will yield an appropriate behaviour (inverse dynamics). Once the forces which act on the articulated body are known, we can easily solve the direct dynamics problem - that of describing the model's resulting behaviour.

Thus, we have showed that a combination of direct and inverse dynamics motion specification provides a good motion control technique in the sense that it can provide initial estimates of the forces required to move the model to the desired positions. Once these forces are known, some experimentation can take

place so that direct dynamics on its own can be used successively to produce the required movement. The computational complexity is still linearly related to the number of joints, but since dynamics calculations need to be executed twice (inverse + direct), a double computational cost is incurred.

The nature of the problem is changed into a *hybrid inverse and direct dynamics method* (HIDDS) : inverse dynamics involves the calculation of the forces required at the joints of the model, in order to produce a given set of relevant accelerations. Once we are given enough information to be able to calculate the motion of every link in the model, it is then a straightforward matter to calculate the forces involved, using the equations described above. Upon calculation of the forces, we apply direct dynamics to determine the motion of the system by calculating the joint accelerations of the next phase using the direct dynamics formulation already described in Chapter 6. HIDDS is used routinely to experiment with the values of the resulting initial forces, so that direct dynamics can be used alone to produce the motion for the next phases of the movement.

To our knowledge this is the first time that such a hybrid : inverse and direct dynamics approach has been reported and implemented. It can be used as an experimentation tool to provide an automatic control technique, which we believe will offer great potential for animating complex motion.

Wilhelms [2, 71] used a hybrid kinematics/direct dynamics approach where controlling functions, representing positions over time, had to be entered explicitly before starting the simulation, without knowing intuitively what forces or torques would be necessary to produce the desired motion. This approach is more suitable for low-level specification of motion.

Girard [53, 74] and Girard & Maciejewski [54], also used a hybrid kinematics/dynamics algorithm and therefore all the problems associated with the kinematics descriptions were again present when attempting to control articulated figures. A similar approach has been adopted by Bruderlin & Calvert [47] and is discussed by Calvert [75]. Isaacs & Cohen [62] presented another combination of kinematics and inverse dynamics specifications.

Hahn [13] presented a method that merged kinematics and dynamics for articulated rigid bodies. He used kinematics to control joint (internal)

trajectories and dynamics to model the effects of limb motion (external analysis) and external forces and torques on the body as a whole.

We hope that by using such a hybrid inverse/direct dynamics algorithm we move one step further in the process of specifying motion easily and conveniently, as all the computational burden will be carried by HIDDS, relieving the animator from the tedious specification of unknown parameters. Such a motion-control specification provides the basic tool on which experiments can be based for producing a high-level control system.

9.6 REFINING THE MOVEMENT

Once a specific movement is created by HIDDS and displayed by AnthroPI, the aim is to try improving and correcting the motion. The idea is to retain most of the initial motion, created by HIDDS, while improving its performance as a result of practice and experience. This will provide a system capable of synthesising, and learning, coordinating movements and able to improve the motion. However, at an early stage, this correction has been done subconsciously by the animator since a lot of experimentation was required to e.g. specify forces that would move the model at the required position. Refining the motion is left for future work, since its implementation has not been included in the current system yet.

Both Boulic et al [78] and Mohamed & Armstrong [79] have studied the idea of refining the movement. The former used a direct/inverse kinematics system, whereas the latter used a dynamics system. Both methods showed promise but did not produce conclusive evidence of success.

The proposed system can provide the first approximation for the motion and if the user sees that this is incorrect or imperfect, he/she could modify, recompile and re-execute the torque profiles, thus re-initialising the direct dynamics sub-system. Of course, the same mechanism could be applied as many number of iterations as to create acceptable results. Furthermore, a learning control system could be developed where this correction can take place automatically, on the basis of known criteria governing some form of acceptability of the results.

Therefore, the initial motion could be specified by HIDDS and to refine the movement, direct dynamics could be employed to update and refine the forces/torques acting on the system. The mechanism, which could be executed in a specified number of iterations, can provide automatic control over the motion without disrupting the dynamics of the resulting movement.

9.7 CONCLUSIONS

It is concluded that the use of a hybrid inverse and direct dynamics system provides a convenient experimental tool for the creation of realistic human-like movement.

This hybrid system, while still in its infancy, has the potential to allow the user to specify the motion at a higher level of control, to relieve the animator from the tedious task of entering (on some occasions) meaningless values for the parameters of motion, and to offer the capability of creating acceptable results.

As a tool for creating motion sequences (in particular, a walking cycle), it has proved that it is possible and feasible to coordinate the motion at a higher level and to produce sequences which are both realistic and natural. However, some experimentation is still required before the goals incorporated in such a high-level system become a desirable reality.

The model used for this work, was not very sophisticated, but it was efficient for its purposes, i.e. capable of producing and visualising the movement in an economical way. The inclusion of additional links, like the hand with the fingers, or the foot with the toes, would increase the complexity of the calculations and the computational costs of the algorithm.

The human model had the necessary links and the right dimensions as to be recognised as human-like, but then again fine detail was beyond the purpose of this research. Our objective, was to allow for a model to have the basic links and joints of the human figure. These included, the axial skeleton which consists of the trunk, the head and the neck, the upper extremity consisting of the shoulder, the arm and forearm, and the lower extremity consisting of the hip, the thigh and the leg. Furthermore, the system allows

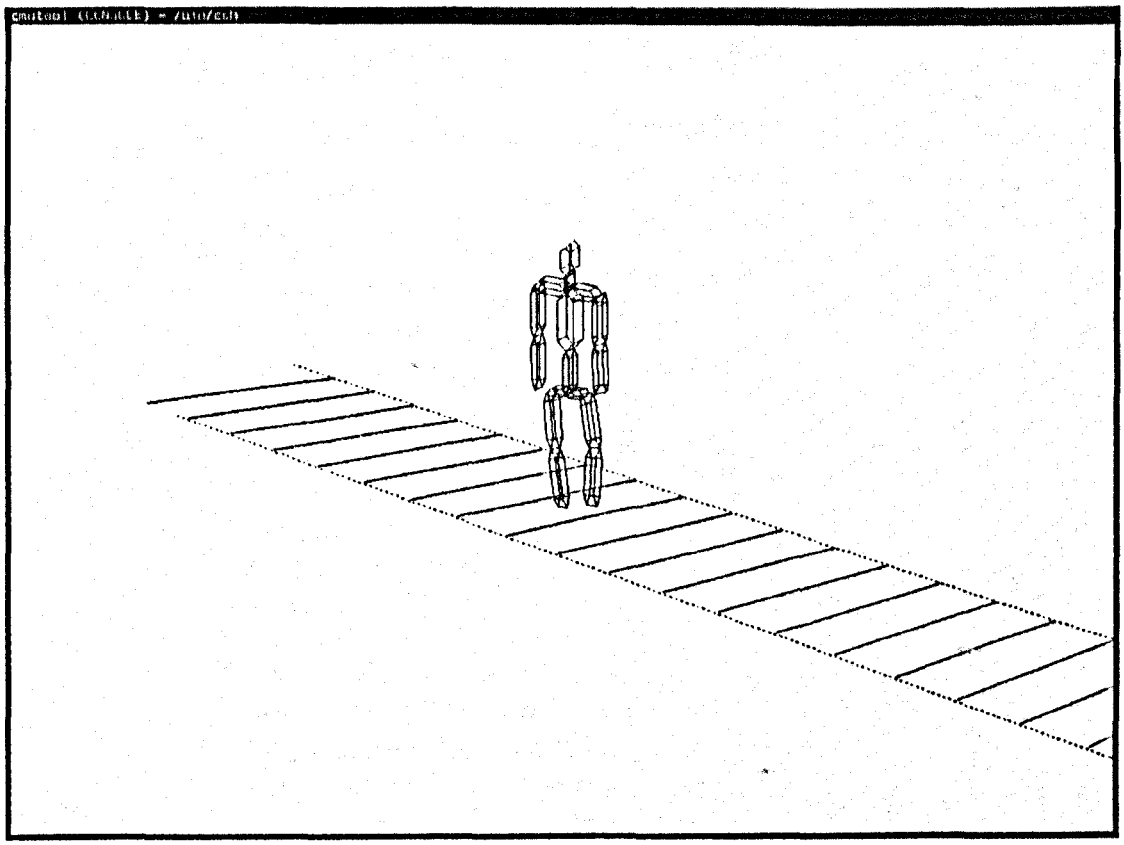
any kind of model to be defined interactively by the animator. This is described through the use of configuration files that specify the initial state of the model, the inertia tensors and masses of the links, the associated rotational/translational components of motion, as well as the connectivity and limitations, if any, of the structure. Additionally, the system allows the user to choose the root of the model.

Employing the human model described above and using HIDDS, a few walking cycles consisting of successive frames were created. Figures 9.7a and 9.7b show single frames of a walking cycle. A step was defined using the correct time durations, the exact values of the walking parameters and the gait determinants. Upon computation of the dynamics for the motion of one of the legs, and using the assumption of a symmetric step, the motion of the other leg was automatically be calculated. The motion of the trunk was separately computed upon consideration of the determinants of the movement, although the trunk was included in the dynamic analysis. The motion of the arms and forearms were also calculated automatically from the motion of the opposite leg.

To overcome some of the control problems, the control technique of freezing (locking) degrees of freedom is also available.

Essentially, the user is asked to determine a small number of walking parameters, such as the number of steps required, and the step frequency. He/she then needs to define the model's desired position and activate the hybrid algorithm. Once the solution of the equations is processed and the result is calculated, the graphics routines are initialised to display the movement. The graphics routines, read the files with the correct motion created, through the use of dynamics, and produce the motion sequence of the anthropomorphic model in an environment useful for visualising the movement.

A utility for saving the results and re-playing the motion, as well as displaying the frames one after the other, in real time, provides the necessary tool to visualise and criticise the correctness and realism of the movement. This will ensure, at a later stage, that the motion sequences produced are natural, realistic and visually acceptable by humans.



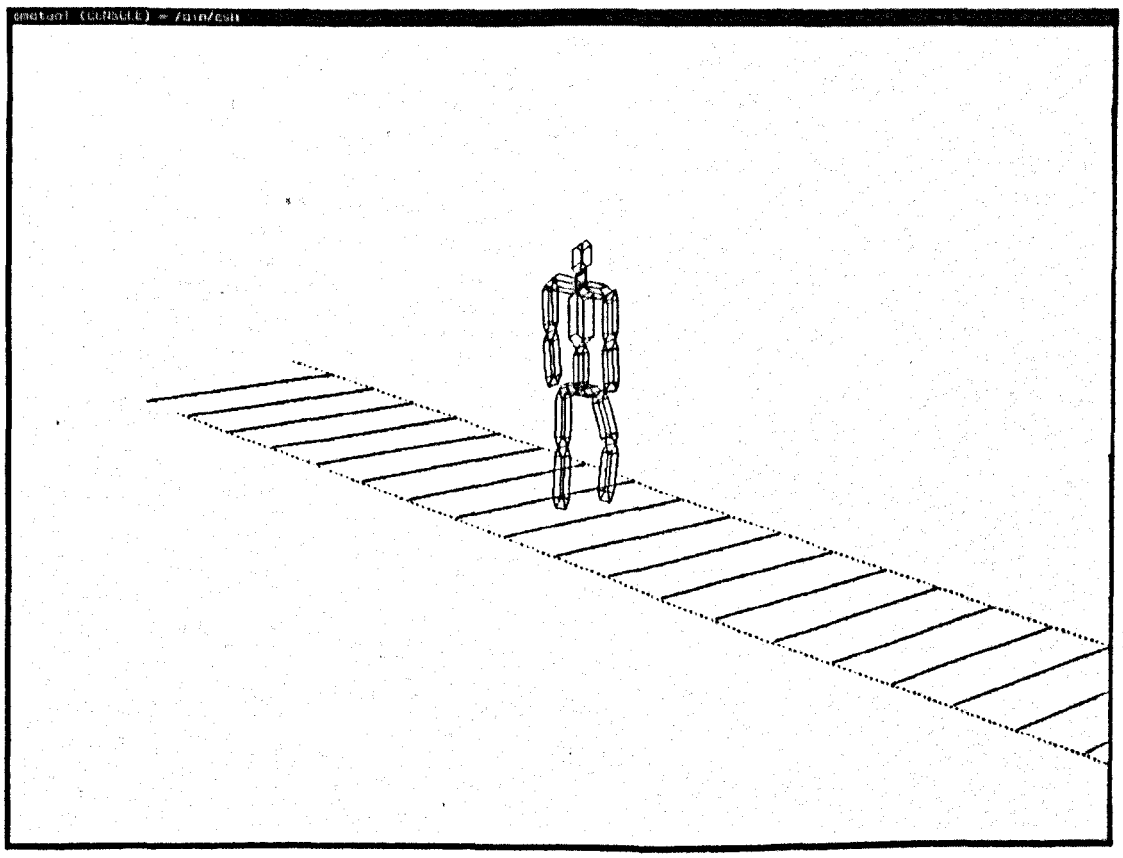
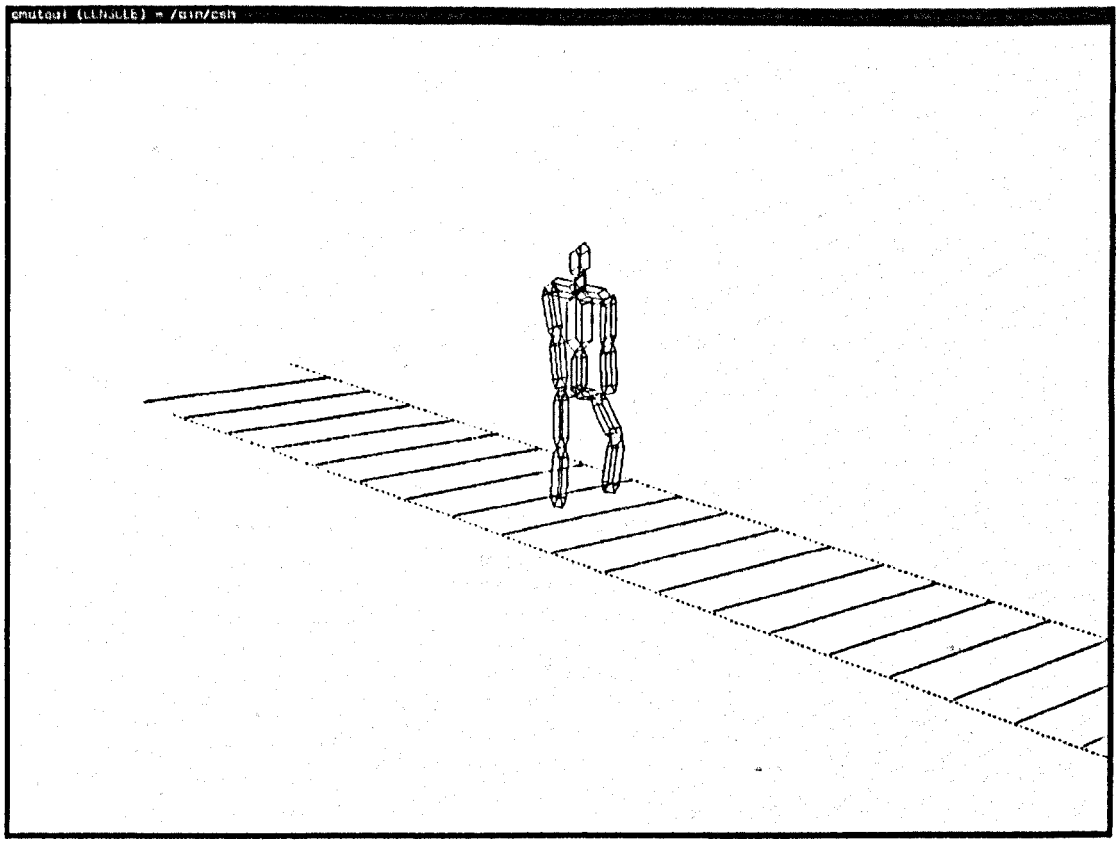


Figure 9.7a : Frames of a Walking Cycle



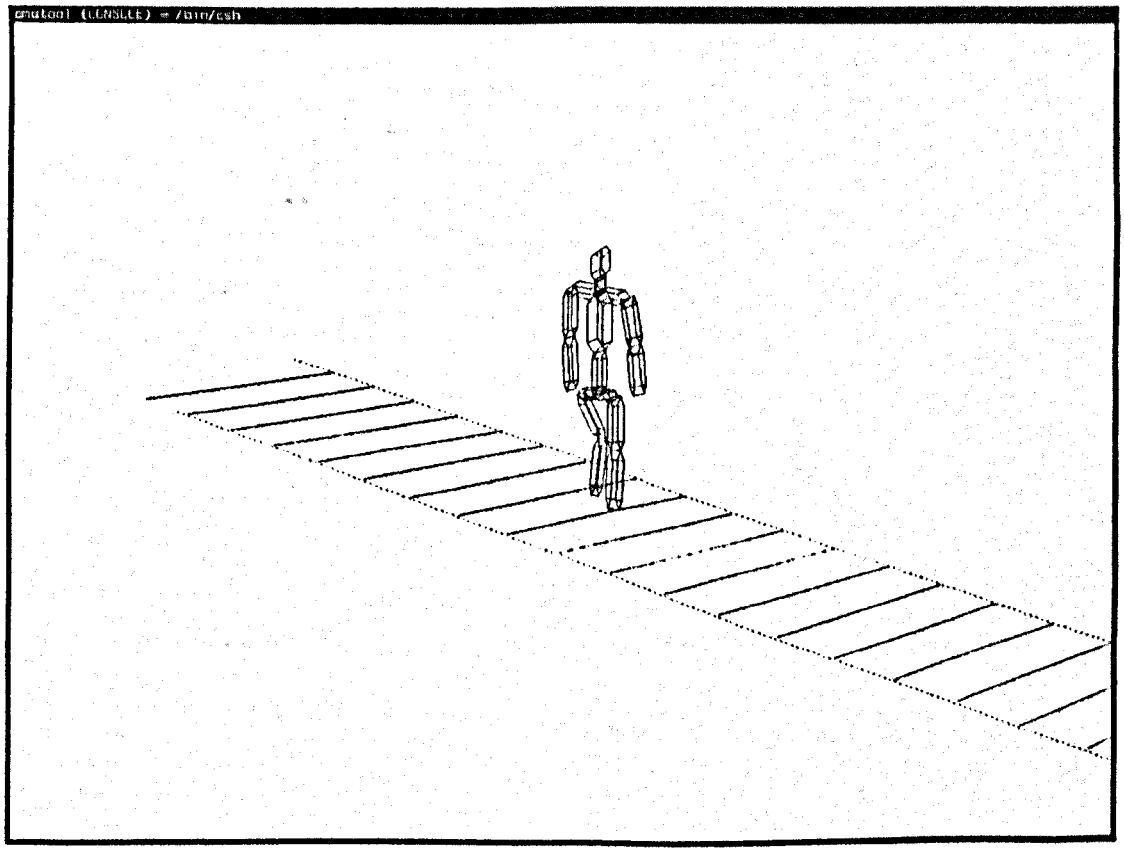


Figure 9.7b : Frames of a Walking Cycle

CHAPTER 10
POSTSCRIPT

10.1 CONCLUSIONS

The main aim of the thesis was to experiment with the use of dynamic analysis in the manipulation and control of anthropomorphic models, for its applicability in computer animation.

Such experimentations developed a technique that was found to be reasonably efficient for human figure animation, as it allows its user to specify motion easily and conveniently, and allows the animator to manipulate and control the movement. This is due to the fact that dynamics is based on physical laws which allow for the production of predictable and useful results.

Once the masses, the inertia tensors and the rotational/translational components of motion of a model are determined, dynamic analysis can be employed to ensure the production of acceptable and realistic results.

Other researchers, in the area, have tried to produce animation sequences by using kinematic control techniques, but this approach inherits all the related problems associated with kinematics. It is difficult to specify realistic motion kinematically, as this involves the tedious specification of, sometimes unknown, parameters of motion, without any particular way of controlling the movement. Dynamics is superior to kinematics when the motion is complex, fast, or involves contact with external forces and torques.

Originally, this work was to use a purely dynamics approach, to avoid the problems mentioned above and to achieve natural motion. In this way, dynamics has to be considered as the basic principle which will provide easy specification of the motion, and will produce accurate and realistic results.

Featherstone's direct dynamics algorithm was extended, implemented and tested for this purpose. The extensions comprised multiple-degree-of-freedom joints, branched kinematic chains and the notion of a mobile base. Appropriate mathematical models were constructed and their formulations were developed.

Initially, the use of direct dynamics on its own was investigated. This is the problem in which the forces, required to produce the motion, are entered into a system of equations and the accelerations, and eventually the

positions, produced under the influence of these forces are calculated. The implementation phase started with the use of Featherstone's extended formulation which, although it produced acceptable motion, had a serious drawback, in that a lot of experimentation was required to determine the forces necessary to place the model in the desired position.

This led us to the conclusion that it was not possible to control the movement easily and conveniently using only direct dynamics. Additional aids were needed to guarantee natural and realistic motion, and a simple method of motion specification.

Thus HIDDS (Hybrid Inverse and Direct Dynamics System) was developed to achieve a fully dynamically animated walking sequence. HIDDS included the use of both inverse and direct dynamics equations of motion. Featherstone's direct dynamics algorithm had to be re-formulated, and used as an inverse dynamics system. Once this re-formulation was applied and tested, it was combined with the direct dynamics approach.

The resulting hybrid system therefore consists of two sub-systems, the direct and the inverse one. A call of HIDDS initialises the values of the inertia tensors, the masses, dimensions and coordinates of all the links and joints of the model, and then transfers the control to the inverse dynamics sub-system. This sub-system requires the determination of the final positions of all the links of the hierarchical structure of the model. It calculates the forces necessary to be applied to each link of the hierarchy, then initialises the evaluation of the direct dynamics sub-system for the calculation of the accelerations and, from these, the positions of the model for the next iteration. Once the initial stage of the motion is reached, HIDDS needs to be used as an experimental tool to provide the 'guide' as to where the motion has to be directed to provide the required results.

The introduction of HIDDS solved most of the problems associated with the use of either one of the direct, or inverse dynamics formulations. Furthermore, it proved to be a useful and convenient tool where further investigations can be based on for the production of natural and realistic movements.

It has therefore been concluded that, the use of either one of the direct or inverse dynamic analysis systems on its own, cannot produce desirable results

and control the movement at a high level. In particular, the use of direct dynamics by itself, has already been discussed and shown to have major drawbacks. Accordingly, Wilhelms, Armstrong and Zeltzer & McKenna have experimented with the use of such a system, but to overcome its problems additional aids, such as the introduction of kinematic techniques, were necessary.

On the other hand, investigations with the use of inverse dynamics on its own, Vasilonikolidakis & Clapworthy and Bruderlin & Calvert, also had serious drawbacks. Using a purely inverse dynamics system, the necessary requirement of directly controlling forces/torques at the joints of a human-like model was very difficult and inconvenient. Inverse dynamics proved to be more applicable to robotics models, where actuators can be used as a means of directing physical devices by applying the calculated forces at the appropriate joints. Thus, the two approaches, direct and inverse dynamics, should be combined and used in conjunction to provide the benefits mentioned above.

The important problem of ground reaction forces was also studied and formulated. It was found that very little literature has been associated with this concept and almost nothing related to the problem when the human figure touched the ground. The method employed here was borrowed from research in robotics, and the original idea was due to Kearney's method. Kearney et al proposed an algorithm for controlling an one-legged hopping model. Some experimentations and assumptions were necessary in applying the technique to our two-legged human model, through the use of the virtual-leg concept. Results produced by the modified algorithm are satisfactory since they allow the model, essentially the foot in contact with the ground, to stay on the ground surface.

For the implementation of the desired results and for the production of a walking cycle, an animation environment, AnthroPI (Anthropomorphic Programming Interface) has been constructed. This provided a convenient system for, firstly, specifying the motion required and initialising the dynamics equations of motion and, secondly, presenting the results graphically. The system is completely interactive and allows its user to specify the motion at any level of control that he/she wishes. It consists of configuration files for initial estimates of all the parameters necessary,

such as masses and inertia tensors; specification of coordinate systems and dimensions; presenting the model as a hierarchical structure; defining the root and the traversal of the tree and for locking degrees of freedom. It also provides menus for selecting different options, calling the system and its sub-systems, as well as displaying the sequence at the required pace. The motion can be stored in, or read from files, if the same sequence needs to be replayed. A play-back facility allows individual frames to be stored in different files and played back successively, in almost real time.

The human model used for this purpose had the necessary links and joints to be recognised as anthropomorphic, and therefore, to enable the visualisation of its movements. Anthropometric data and biological studies provided the correct values for the dimensions of the model.

The use of dynamics showed that such a system has the potential, with further development, to produce expected, realistic, natural and coordinated human motion. The results proved that it is feasible to create an animation system for articulated figures that offers an easy user interface, and it is independent of the configuration of the model. Such a system can be used as the basic platform where further investigations can take place so as to generate realistic and complex movement.

10.2 FUTURE DIRECTIONS

Research needs to be directed towards the completion of the anthropomorphic model to include wrists, hands and fingers, as well as feet and toes. Although, for most movements only a representational hand is necessary, the implementation of fine manual control requires the introduction of fingers. A switch between a 'representational hand' and a 'full hand' model could take place. To achieve a smooth transition between the two models, the masses and the inertia tensors accumulated within a particular joint, e.g. the wrist, need to remain the same, so that the dynamics would not be affected significantly.

To achieve more subtleties in some movements, toes would be needed and these can be introduced at the expense of computational time. For example, the introduction of toes would need to be studied along with the ground reaction

concept which will require reformulation of the contact algorithm as new forces are active when, initially, the toe touches the ground, before the whole foot becomes flat on the floor surface. In addition, implementation details to include all the six gait determinants would also be necessary and desirable. Particularly, the introduction of toes in the model would essentially require the qualification of the ankle motion.

Further, more detailed experimentation with the style of motion needs to be undertaken, so that issues relevant not just to *where*, but to *how* the model is moving are addressed. Although the style of motion is irrelevant in robotics, it is of great importance in the animation of human models, as there exists a large number of possible ways of going from A to B, even for the same person, not to mention the variations between different persons. Additionally, different ways of moving, such as running, jumping and dancing need to be analysed. Running, for example, would require a modification of the dynamic model as well as the high-level concepts to account for the flight phase during which both legs are off the ground.

Alternative, and additional ways, to dynamics for controlling the movement of the human model need to be investigated through, for example, the use of mathematical approximations. For kinematic animation, the user can design control functions, representing positions over time, for each degree of freedom. Alternatively, the velocity and/or the acceleration could be given, and the other two could determine by differentiation or integration of the given parameter. Control functions may represent either kinematic information (positions over time) or dynamic information (forces or torques over time). This provides a rich area for future work.

Another important area of study is collision detection and obstacle avoidance, for which very little information has been reported. A simple algorithm for obstacle avoidance would assume a straight line path from start to goal and test for possible collisions. The system should have an efficient representation of the geometry of the objects in order to plan movements automatically and prevent collisions. When a collision is detected, a new path that avoids the collision can be generated. This process can be repeated until a collision-free path is found. The problem is made more complex when obstacles are themselves in motion. A 'penalty' method could be introduced which would involve the creation of strong collision forces between the

objects as they come in close proximity to each other. To model these forces, a technique similar to the one used for the joint limits using springs could be employed. This has also been discussed by Gascuel et al [80].

Locomotion on rough terrains is another field of possible investigation, where the body needs to adjust its walking style according to the texture and shape of the terrain. Travelling on rough terrain includes many subproblems, such as sensing the terrain, planning a path, selecting a foothold and adjusting the step length, while retaining balance. Bipeds use mostly dynamic balance, in which the figure is falling from support point to support point. The development of a control technique for maintaining postural balance is essential. A simple way to achieve balance in walking is to describe a world-space orientation vector for particular segments, such as the trunk, and apply an external force to counteract any motion away from this orientation.

Further progress in this area needs to be directed towards the development of higher level control techniques. Motion processes can be viewed at the bottom level in a hierarchy of control techniques. Higher levels of this hierarchy would be responsible for more complex motion and for synchronisating the motions of several limbs. The construction of this type of control hierarchy depends on having a useful set of motion processes. These higher-level control schemes would allow the user to describe motion in general terms which the system would then automatically convert to specific low-level joint behaviour.

Related work from the biomechanics of motion needs to be incorporated further into the system. For example, research from the clinical studies and biomechanical experiments provides useful data in determining how the trunk movement is related to the rest of the body during human locomotion. Further, differences between the activities of walking and running are reflected in the different patterns of the muscles controlling the ankle, which could help in the creation of the different gaits of bipeds.

An ideal system should be able to combine all the above related topics so that a learning control system, possibly an expert database, could be developed. The useful mechanism of correcting and refining the motion needs to be incorporated. Furthermore, figures could interact in parallel through

message passing, to reduce computational costs. Such a proposed, high-level system could be used to create visual worlds through the use of a computer, and aid the large list of applications related to this area. The topic has still a broad and fertile prospects for future research. The possibilities are endless.

REFERENCES

REFERENCES

[1] **D. Tost, X. Pueyo**
Human Body Animation : a Survey
The Visual Computer, Vol.3, No.5, pp254-264, 1988.

[2] **J. Wilhelms**
Using Dynamic Analysis for Realistic Animation of Articulated Bodies
IEEE CG&A, pp12-27, June 1987.

[3] **W. Armstrong, M. Green, R. Lake**
Near-Real Time Control of Human Figure Models
IEEE CG&A, pp52-61, June 1987.

[4] **B. Wyvill**
Navigating the Animation Jungle
'Computer Graphics Techniques and Practice', Editors D.. Rogers, R. Earnshaw, Springer-Verlag, 1990.

[5] **W. Armstrong, M. Green**
The Dynamics of Articulated Rigid Bodies for purposes of Animation
The Visual Computer, Vol.1, pp231-240, 1985.

[6] **D. Zeltzer**
Towards an Integrated View of 3-D Computer Animation
The Visual Computer, Vol.1, pp249-259, 1985.

[7] **J. Luh, M. Walker and R. Paul**
On line Computational Scheme for Mechanical Manipulators
Journal of Dynamic Systems, Measurement and Control, Vol.102, pp69-76, June 1980.

[8] **J. Hollerbach**
A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity
IEEE Transactions on Systems, Man, and Cybernetics, Vol SMC-10, No.11, PP730-736, November 1980.

[9] **W. Armstrong**
Recursive Solution to the Equations of Motion of an N-Link Manipulator
Proc. 5th World Congress on Theory of Machines and Mechanics, Vol.2, pp1343-1346, July 1979.

[10] **K. Symon**
Mechanics
Addison-Wesley, 3rd edition, 1971.

[11] **M. Walker, D. Orin**
Efficient Dynamic Computer Simulation of Robotic Mechanisms
Journal of Dynamic Systems, Measurement and Control, Vol.104, pp205-211, September 1982.

[12] **J. Luh, M. Walker, R. Paul**
Newton-Euler Formulation of Manipulator Dynamics for Computer Control
2nd IFAC/IFIP Symposium Information Control Problems, Manufacturing Technology, Stuttgart, pp165-172, 1979.

[13] **J. Hahn**
Realistic Animation of Rigid Bodies
ACM Computer Graphics, Vol.22, No.4, pp299-307, August 1988.

[14] **R. Barzel, A. Barr**
A Modelling System Based on Dynamic Constraints
ACM Computer Graphics, Vol.22, No.4, pp179-188, August 1988.

[15] **M. Raibert**
Manipulator Control using the Configuration Space Method
The Industrial Robot, pp69-73, June 1978.

[16] **R. Featherstone**
The Calculation of Robot Dynamics Using Articulated Body Inertias
International Journal of Robotics Research, Vol.2, No.1, pp13-30, 1983.

[17] **Y. Stepanenko, M. Vukobratovic**
Dynamics of Articulated Open-chain Active Mechanisms
Mathematical Biosciences, Vol.28, No.1/2, pp137-170, 1976.

[18] **M. Vukobratovic, D. Stokic, D. Hristic**
New Control Concept of Anthropomorphic Manipulators
Mechanism and Machine Theory, Vol.12, pp515-530, 1977.

[19] **R. Featherstone**
Robot Dynamics Algorithms
Kluwer Academic Publishers, 1987.

[20] **O. Bottema, B. Roth**
Theoretical Kinematics
North Holland, Amsterdam, 1979.

[21] **C. Li**
A Fast Computational Method of Lagrangian Dynamics for Robot Manipulators
International Journal of Robotics and Automation, Vol.3, No.1, pp14-20, 1988.

[22] **K. Hashimoto, H. Kimura**
A New Parallel Algorithm for Inverse Dynamics
International Journal of Robotics Research, Vol.8, No.1, pp63-76, 1989.

[23] **M. McKenna, D. Zeltzer**
Dynamic Simulation of Autonomous Legged Locomotion
ACM Computer Graphics, Vol.24, No.3, pp29-37, August 1990.

[24] **R. Burden, D. Faires**
Numerical Analysis,
PWS-Kent Publishing Company, Fourth Edition, 1989.

[25] **M. McKenna**
A Dynamic Model of Locomotion for Computer Animation
MSc Thesis, MIT, January 1990.

[26] **R. Boulic, N. Magnenat-Thalmann, D. Thalmann**
Human Free-Walking Model for a Real-Time Interactive Design of Gaits
Computer Animation'90, Springer-Verlag, pp61-79, 1990.

[27] **T. McMahon**
Muscles, Reflexes and Locomotion
Princeton University Press, Princeton NH, 1984.

[28] **J. Uicker**
On the Dynamic Analysis of spatial Linkages using 4x4 Matrices
Ph.D. dissertation, Northwestern University, August 1965.

[29] **M. Kahn**
The Near-minimum-time Control of Open-loop Articulated Kinematic Chains
Stanford Artificial Intelligence Project Memo. AIM-106, December 1969.

[30] **R. Waters**
Mechanical Arm Control
M.I.T. Artificial Intelligence Lab. Memo. 549, October 1979.

[31] **T. McGovern**
The use of Live-Action footage as a tool for the Animator
Tutorial Notes in Character Animation, SIGGRAPH 1987.

[32] **D. Orin, R. McGhee, M. Vukobratovic, G. Hartoch**
Kinematic and Kinetic Analysis of Open-chain linkages utilizing Newton-Euler methods
Math Biosc, Vol.43, pp107-130, 1979.

[33] **W. Silver**
On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators
Robotics Research Vol.1, No.2, pp60-70, 1982.

[34] **C. Philips, N. Badler**
Interactive Behaviors for Bipedal Articulated Figures
ACM Computer Graphics, Vol.25, No.4, pp359-362, July 1991.

[35] **N. Vasilonikolidakis, G. Clapworthy**
Design of Realistic Gaits for the Purpose of Animation
Computer Animation'91, Springer-Verlag, pp101-114, 1991.

[36] **D. Baraff**
Coping with Friction for Non-penetrating Rigid Body Simulation
ACM Computer Graphics, Vol.25, No.4, pp31-40, July 1991.

[37] **M. Raibert, J. Hodgins**
Animation of Dynamic Legged Locomotion
ACM Computer Graphics, Vol.25, No.4, pp349-358, July 1991.

[38] **J. Hodgins, M. Raibert**
Adjusting Step Length for Rough Terrain Locomotion
IEEE Transactions on Robotics and Automation, Vol.7, No.3, pp289-298, June, 1991.

[39] **R. Cottle**
On a problem in Linear Inequalities
Journal of the London Mathematical Society, Vol.43, pp378-384, 1968.

[40] **G. Ridsdale, S. Hewitt, T. Calvert**
The Interactive Specification of Human Animation
Proc. Graphics Interface and Vision Interface'86, 1986.

[41] **A. Thorstensson, L. Oddsson, H. Carlson**
Motor Control of Voluntary Trunk Movements in Standing
Acta Physiol Scand, No.125, pp309-321, 1985.

[42] **A. Thorstensson, J. Nilsson, H. Carlson**
Trunk Movements in Human Locomotion
Acta Physiol Scand, No.121, pp9-22, 1984.

[43] **S. Grillner, J. Halbertsma, J. Nilsson, A. Thorstensson**
The adaptation to speed in Human Locomotion
Brain Research, No.165, pp177-182, 1979.

[44] **J. Nilsson, A. Thorstensson, J. Halbertsma**
Changes in Leg Movements and Muscle Activity with speeds of Locomotion and mode of Progression in Humans
Acta Physiol Scand, No.123, pp457-475, 1985.

[45] **M. Vukobratovic, B. Borovac, D. Stokic**
Legged Locomotion
Springer-Verlag, Berlin, 1990.

[46] **D. Baraff**
Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies
ACM Computer Graphics, Vol.23, No.3, pp223-231, July 1989.

[47] **A. Bruderlin, T. Calvert**
Goal-Directed Dynamic Animation for Human Walking
ACM Computer Graphics, Vol.23, No.3, pp233-242, July 1989.

[48] **J. Kearney, S. Hansen, J. Cremer**
Programming Mechanical Simulations
2nd Eurographics Workshop on Simulation and Animation, Austria, September 1991.

[49] **J. Hodgins, M. Raibert**
Biped Gymnastics
International Journal of Robotics Research, Vol.9, No.2, pp115-132, 1990.

[50] **P. Schroder, D. Zeltzer**
The Virtual Erector Set: Dynamic Simulation with Linear Recursive Constraint Propagation
Proc. 1990 Symposium on Interactive 3D Graphics, March 25-29, Snowbird, Utah, 1990.

[51] **P. Schroder, D. Zeltzer**
The Virtual Erector Set
MSc Thesis, MIT, January 1990.

[52] **V. Inman, H. Ralston, F. Todd**
Human Walking
Williams & Wilkins, Baltimore/London, 1981.

[53] **M. Girard**
Interactive Design of 3D Computer-Animated Legged Animal Motion
IEEE Computer Graphics & Applications, Vol.7, No.6, pp39-50, June 1987.

[54] **M. Girard, A. Maciejewski**
Computational Modeling for the Computer Animation of Legged Figures
ACM Computer Graphics, Vol.19, No.3, pp263-270, July 1985.

[55] **K. Shoemake**
Animating Rotation with Quaternion Curves
ACM Computer Graphics, Vol.19, No.3, pp245-254, July 1985.

[56] **N. Badler, S. Smoliar**
Digital Representations of Human Movement
ACM Computing Surveys, Vol.11, No.1, pp19-38, March 1979.

[57] **M. Moore, J. Wilhelms**
Collision Detection and Response for Computer Animation
ACM Computer Graphics, Vol.22, No.4, pp289-297, August 1988.

[58] **T. McMahon**
Mechanics of Locomotion
International Journal of Robotics Research, Vol.3, No.2, pp4-28, 1984.

[59] **G. Miller**
The Motion Dynamics of Snakes and Warms
IEEE CG&A Vol.22, No.4, pp169-177, August 1988.

[60] **M. Raibert**
Manipulator Control using the Configuration Space Method
The Industrial Robot, pp69-73, June 1978.

[61] **R. Alexander**
Animal Mechanics
Blackwell Scientific Publications, 1983.

[62] **P. Isaacs, M. Cohen**
Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions and Inverse Dynamics
ACM Computer Graphics, Vol.21, No.4, pp215-224, July 1987.

[63] **J. Wilhelms**
Graphical Simulation of the Motion of Articulated Bodies such as Humans and Robots with special emphasis on the use of Dynamic Analysis
Ph.D. Dissertation - University of California, Berkeley, 1985.

[64] **N. Vasilonikolidakis, G. Clapworthy**
Inverse Lagrangian Dynamics for the Purpose of Animating Articulated Bodies
PNL Internal Report, Research Note RN 91/5, 1991.

[65] **E. Muybridge**
The Human Figure in Motion
Dover Publications, Inc., New York, 1955.

[66] **J. Craig**
Introduction to Robotics Mechanics and Control
Addison-Wesley Publishing Company, 2nd edition, 1989.

[67] **J. Saunders, V. Inman, H. Eberhart**
The Major Determinants in Normal and Pathological Gait
Journal of Bone and Joint Surgery, Vol.35-A, No.3, pp543-558, July 1953.

[68] **M. Raibert**
Legged Robots that Balance
The MIT Press, Cambridge, Massachusetts, 1986.

[69] **P. Galley, A. Forster**
Human Movement - An Introductory text for Physiotherapy Students
Churchill Livingstone, 2nd edition, 1987.

[70] **I. Sutherland**
A Walking Robot
Pittsburgh : The Marcian Chronicles, Inc., 1983.

[71] **J. Wilhelms**
VIRYA - A motion Control Editor for Kinematic and Dynamic Animation
Graphics Interface'86, Morgan Kaufmann, pp141-146, May 1986.

[72] **J. Wilhelms**
Towards Automatic Motion Control
IEEE Computer Graphics and Applications, pp11-22, April 1987.

[73] **D. Zeltzer**
Task-level Graphical Simulation : Abstraction, Representation and Control
Make Them Move, Morgan Kaufmann, pp3-33, 1991.

[74] **M. Girard**
Constrained optimization of Articulated Animal Movements in Computer Animation
Make Them Move, Morgan Kaufmann, pp209-222, 1991.

[75] **T. Calvert**
Composition of realistic Animation sequences for Multiple Human Figures
Make Them Move, Morgan Kaufmann, pp35-50, 1991.

[76] **G. Miller**
MacBounce : A Dynamics-Based Modeler for Character Animation
Computer Animation'91, Springer-Verlag, pp149-168, 1991.

[77] **K. Sims, D. Zeltzer**
A Figure Editor and Gait Controller for Task Level Animation
SIGGRAPH '88, Course notes on Synthetic Actors, 1988.

[78] **R. Boulic, O. Renault**
3D Hierarchies for Animation *in*
New Trends in Animation and Visualization, Editors N. Thalmann, D. Thalmann, John Wiley & Sons, 1991.

[79] **A. Mohamed, W. Armstrong**
A Hybrid Numerical/Knowledge-based Locomotion Control System for a Multi-Legged Manipulator Robot
Proceedings of the Eight International Workshop on Expert Systems and Their Applications, Avignon France, Vol.1, pp511-529, 1988.

[80] **M. Gascuel, A. Verroust, C. Puech**
A modelling system for complex deformable bodies suited to animation and collision processing
Journal of Visualisation and Computer Animation, Vol.2, No.3, pp82, 1991.