PROPERTIES OF ROBOT FORWARD DYNAMICS ALGORITHMS
WITH APPLICATIONS TO SIMULATION

By
Benoit P. Cloutier
B. Eng. Ecole Polytechnique de Montreal

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in
THE FACULTY OF GRADUATE STUDIES
COMPUTER SCIENCE

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
March 1995
© Benoit P. Cloutier, 1995
In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Computer Science
The University of British Columbia
2366 Main Mall
Vancouver, Canada
V6T 1Z4

Date: [April 15, 1995]
Abstract

This thesis presents issues related to the dynamic simulation of robot manipulators. The numerical simulation problem is usually treated as two separate problems: the forward dynamics problem for computing system accelerations, and the numerical integration problem for advancing the state in time. Using a compact and unifying notation, existing methods for computing robot forward dynamics are presented and compared. It is shown that the articulated-body method [19, 20] (ABM) is better suited to deal with certain types of numerical problems than the composite rigid-body method [58] (CRBM). Simulation results are presented and the practical implications of these results are considered. In particular, it is shown that the fastest forward dynamics methods are not necessarily best when considered in conjunction with the popular adaptive step-size integration methods. Finally, this thesis also reviews issues related to the difficult task of incorporating contact simulation in the robot simulation process.
Table of Contents

Abstract ................................................................. ii

List of Figures ......................................................... vi

1 Introduction ......................................................... 1
  1.1 Thesis concerns ................................................. 1
  1.2 Thesis Overview ............................................... 2

2 Fundamentals of Robot Manipulator Simulation ................. 3
  2.1 General notation conventions ................................. 3
  2.2 Robot model ..................................................... 5
  2.3 The formulation problem ....................................... 7
  2.4 Dynamics of a single rigid body ............................... 10
    2.4.1 Spatial vectors ............................................. 10
    2.4.2 Rigid body transformation operator ....................... 11
    2.4.3 Coordinate transformation operator ....................... 12
    2.4.4 Newton and Euler's equations of motion ................... 13
  2.5 The inverse dynamics problem .................................. 14
    2.5.1 Generalized spatial operators ............................. 16
  2.6 The motion integration problem ............................... 18

3 Solving the Forward Dynamics Problem ......................... 20
  3.1 $O(n^3)$ Algorithms: inertia matrix inversion methods .......... 20
3.1.1 The composite rigid-body method ........................................ 21
3.1.2 Orthogonal complement based methods ............................... 22
3.2 $O(n^2)$ Algorithms: conjugate gradient methods .................... 23
3.3 $O(n)$ Algorithms: linear recursive methods ......................... 24
3.3.1 The articulated-body method ........................................... 25
3.4 Summary and comparisons ................................................... 27

4 Numerical Properties of Forward Dynamics Algorithms .................. 29
4.1 Review of issues in numerical computation ........................... 29
4.2 Numerical properties of the CRBM ....................................... 30
4.3 Numerical properties of the ABM ........................................ 32
4.4 Impact on motion integration: formulation stiffness .................. 34

5 Implementation and Experimental Verification ............................ 35
5.1 PRRRobot simulation software ............................................. 35
5.1.1 Assumptions and definitions ......................................... 35
5.1.2 Forward dynamics test program .................................... 37
5.1.3 Off-line simulation program ........................................ 38
5.1.4 Animation program .................................................... 40
5.1.5 Implementation issues ................................................ 41
5.2 Results and observations: simulation of ill-conditioned systems .... 44
5.2.1 Performance of the forward dynamics ................................ 44
5.2.2 Performance of the motion integration .............................. 48
5.3 More simulation results ..................................................... 50
5.3.1 "Bad" implementations of the ABM ................................ 50
5.3.2 Conservation of energy ................................................. 50
5.3.3 Simulation of long chains ............................................. 53
6 Extensions: Contact Simulation

6.1 Notation ................................................. 57
6.2 Contact detection ...................................... 59
6.3 Friction modelling ..................................... 60
6.4 Impact simulation ....................................... 61
   6.4.1 Penalty methods .................................. 61
   6.4.2 Analytical methods ............................... 62
   6.4.3 Finite element methods ......................... 65
6.5 Simulation of continuous contact ..................... 66
   6.5.1 Penalty methods .................................. 66
   6.5.2 Analytical methods ............................... 66
   6.5.3 Impulse based methods ......................... 68

7 Conclusion .................................................. 69

Bibliography .................................................. 71
List of Figures

2.1 Notation for link k. .................................................. 6

5.1 N-link PRRRobot manipulator. ................................. 37

5.2 PRRRobot off-line simulator interface and options. ............. 39

5.3 Effect of configuration on the condition number of 2 link chain. Note that the condition number is largest at the singularities in this case. The relative difference for joint 1 accelerations ($\Delta \ddot{\theta}_1$) is also shown; results for joint 2 accelerations are similar. ........................................... 45

5.4 Effect of link length ................................................. 46

5.5 Comparison of calculated joint 1 accelerations (with $l[2]=m[2]=1e5$). Note the smoothness of the ABM solution. ........................................... 47

5.6 Number of forward dynamics evaluations required to simulate 10 seconds of motion for various two link chains of increasing condition number. .. 49

5.7 Joint accelerations for “good” and “bad” implementations of the ABM (the smooth curve is the “good” implementation of the ABM). ........ 51

5.8 Evolution of the total energy. .................................... 52

5.9 Summary of results for the simulation of long regular chains. ....... 54

5.10 Simulation of long chains. ....................................... 55

6.1 Contact simulation notation. ................................... 58
Chapter 1

Introduction

Computer simulation is playing an increasingly important role in the field of robotics. Simulation of robot dynamics has been used in off-line tasks such as robot design testing, robot programming, and controller validation. Recently, the need for real-time performance has become important for on-line applications such as predictive displays for time-delayed teleoperation [53, 9], virtual environments for operator training and development of advanced robot control schemes.

In robotics, the simulation problem requires that the joint trajectory of a robot be determined given prior knowledge of the applied forces and the initial robot state. The simulation problem can be divided into two subproblems:

- **The forward dynamics problem**: computing the joint accelerations given the actuator forces/torques and the current robot joint positions and velocities.

- **The motion integration problem**: computing the joint trajectory (joint positions and velocities) given the joint accelerations and initial conditions.

1.1 Thesis concerns

In recent years, the need for increased performance in robot simulation has led to the development of many different algorithms for solving the forward dynamics problem. The algorithms range in computational complexity from $O(n^3)$ to $O(n)$ where $n$ is the number of degrees of freedom of the robot. Detailed counts of the total number of operations
Chapter 1. Introduction

per forward dynamics evaluation is a common means of comparing the performance of the different algorithms. However, the computational complexity of a forward dynamics method is only half the story; accurate numerical integration of the resulting acceleration is an important factor in the total efficiency of a simulation method. The overall efficiency of a simulation procedure is best measured in terms of the computational effort necessary to achieve a desired accuracy in the solution. In this light, it will be shown how the numerical properties of forward dynamics algorithms can play an important role in the selection and implementation of algorithms for the simulation of robot manipulators. These issues will be first discussed in a theoretical framework and verified experimentally in a number of computer simulations.

Another important issue in the development of robot simulators is the need to simulate contacts of the robot with its environment. Recently, the field of contact dynamics has attracted a significant amount of research efforts. In this thesis, the problems related to incorporating such capabilities into a robot simulation system will be discussed.

1.2 Thesis Overview

This thesis is organized as follows. Chapter 2 presents the problem statement, assumptions and notation conventions. Chapter 3 gives an overview of the algorithms for solving the forward dynamics problem. Chapter 4 explores the numerical properties of two of the most commonly used forward dynamics methods: the composite-rigid body method (CRBM) and the articulated body method (ABM). Chapter 5 presents some numerical results obtained with regards to the forward dynamics problem and the simulation problem. Chapter 6 discusses possible extensions to include contact simulation. Finally, chapter 7 presents some concluding remarks.
Chapter 2

Fundamentals of Robot Manipulator Simulation

This chapter introduces and defines some of the basic concepts that will be explored within the context of this thesis. The first section of the chapter presents an overview of the general notation conventions. As will be demonstrated, a good notation plays a key role in the formulation of clear and intuitive equations of robot dynamics. The second section presents the robot model and some of its associated notation. The third section looks at the problem of formulating the equations of motion of a robot manipulator. The formulation problem is defined and some of the key high level issues related to this problem are introduced. The fourth section of this chapter introduces the spatial notation through the study of the dynamics of a single rigid body. The fifth section deals with multibody dynamics. In particular, a solution to the inverse dynamics problem is presented (the forward dynamics problem is reviewed in detail in chapter 3). Finally, the last section presents a brief overview of the motion integration problem. Notation is introduced gradually throughout each section of this chapter. This notation is the result of a combination of several previously published notations including Craig’s, Featherstone’s and Jain’s [15, 20, 32].

2.1 General notation conventions

The following notation conventions are observed throughout this thesis:

- Uppercase script letters (Latin) are used to designate points, sets of points and coordinate systems (e.g. \( O_B \) designates an origin point on a rigid body \( B \)).
Chapter 2. Fundamentals of Robot Manipulator Simulation

- Lowercase letters (Latin or Greek) are used to represent vector and scalar quantities.

- Uppercase letters are used to represent linear operators (matrix quantities).

- Overlined letters designate a three dimensional vector quantity or linear operator (e.g. \( \omega_B \) is the \( 3 \times 1 \) dimensional angular velocity vector of a body \( B \)).

- Letters topped with a caret symbol (\(^\wedge\)) designate six dimensional spatial vectors or linear operators (the spatial notation will be defined in a later section).

- Underlined letters designate \( n \)-dimensional generalized vectors or linear operators (e.g. \( \dot{\theta} \) is the \( n \times 1 \) dimensional vector of joint angular velocities of a \( n \) degree of freedom robot manipulator).

- Overlines, carets and underlines can be combined to designate \( 3n \) or \( 6n \) dimensional block entities (e.g. \( \hat{v} \) is the \( 6n \times 1 \) dimensional vector of link spatial velocities of an \( n \) degree of freedom robot manipulator, where \( \hat{v}_k \) is the \( 6 \times 1 \) dimensional spatial velocity of link \( k \)).

- Trailing (right) subscripted numbers or lowercase letters are used to designate components of a vector or linear operator (e.g. \( A_{i,j} \) is component \((i,j)\) of the linear operator \( A \)).

- Trailing subscripted uppercase letters can be used to provide an additional description of a quantity (e.g. \( \hat{f}_T \) is the generalized total spatial force acting on the links of a robot manipulator).

- Trailing superscripts are used to define special operators (e.g. \( A^{-1} \) is the inverse of \( A \) and \( A^T \) is the transpose of \( A \)).
Chapter 2. Fundamentals of Robot Manipulator Simulation

- Leading (left) superscripted letters or numbers are used to designate the coordinate frame with respect to which the components of a quantity are expressed. A quantity without a leading superscript is said to be coordinate free. However, we may often assume that \( ^i v_i = v_i \) in order to maximize clarity (such usage should be clear from context).

- Leading subscripts are not used.

- The identity matrix is designated by 1 and the null matrix is designated by 0. The dimensions of these quantities should be clear from the context.

- The \( 3 \times 3 \) skew-symmetric matrix (cross-product matrix) corresponding to a \( 3 \times 1 \) vector \( \vec{v} = [ v_1 \ v_2 \ v_3 ]^T \) is denoted \( \tilde{v} \).

\[
\tilde{v} = \begin{bmatrix}
0 & v_3 & -v_2 \\
-v_3 & 0 & v_1 \\
v_2 & -v_1 & 0
\end{bmatrix} = -(\vec{v})^T
\]

In summary, lowercase and uppercase letters are used to distinguish between vector quantities and linear operators. Overlines, carets and underlines are used to distinguish between three dimensional, six dimensional (spatial) and n-dimensional (generalized) entities.

2.2 Robot model

The robot manipulators considered within the context of this study are composed of a serial chain of \( n \) links having no closed-loops (open-chain configuration) and connected by single degree of freedom (DOF) rotational joints. The proximal link of the chain
is connected to a fixed base element. The distal link of the chain is called the end-effector link. The end of the distal link is called the robot manipulator tip. The links are numbered 1 through $n$ from the base to the tip.

Each link $k$ is considered to be a single rigid body having a uniformly distributed mass $m_k$. The link center of mass is located at the point $C_k$ and $I_{C_k}$ is the link's inertia matrix about $C_k$. To the proximal end of each link is attached a fixed coordinate frame. Link coordinate frames are associated to the robot manipulator according to the Denavit-Hartenberg convention [15] (see figure 2.1). The vector from the location of the origin $O_k$ of link coordinate frame $k$ and the origin $O_{k+1}$ of the successive link coordinate frame is $l_{k,k+1}$. The vector from $O_k$ to the link center of mass $C_k$ is $\bar{r}_k$. $X_{k+1,k}$ is defined as the coordinate transformation matrix from frame $O_{k+1}$ to frame $O_k$.\(^1\)

Each joint $k$ is characterized by a unit joint axis vector, for a revolute joint, $\bar{z}_k = [0 \ 0 \ 1]^T$ in the direction of the $z$ axis of the links coordinate frame. Joints are

\(^1\)Transformation operators will be defined later in this chapter.
equipped with actuators which apply torques $\tau_k$ about the joint axis based upon the robot's particular control law.\textsuperscript{2} The masses of the joints and actuators are considered to be incorporated into the description of the link. Kinematic joint limits and dynamic actuator limits will not be considered.

The manipulator is assumed to be in a gravity field characterized by a constant linear acceleration vector $\vec{g}$. At first, the effects of joint damping friction\textsuperscript{3} and external contact forces will not be considered. However, the topic of incorporating contact simulation into a robot simulation will be briefly introduced in a later chapter of this thesis.

Overall, the type of multibody serial chain considered in this thesis allows one to build basic models of many of today's popular robot manipulator designs. Serial multibody chains also form the basic components of many of the more complex multibody robotic system designs. Hence the importance of developing the capabilities to efficiently simulate these basic types of robot manipulators.\textsuperscript{4}

### 2.3 The formulation problem

A preliminary step in solving the simulation problem for an n-link robot manipulator is to formulate its equations of motion. These equations relate forces and torques to accelerations given the current state of the robot and its inertial properties. This step is often referred to as solving the formulation problem.

There are two approaches to formulating the equations of motion of multibody systems: elimination methods and augmentation methods [52]. Elimination based methods (or formulations in configuration space) are based on defining the equations of motion in

\textsuperscript{2}Issues related to robot control will not be discussed within the context of this thesis. Many of our analyses will consider the robot manipulator to be an unactuated multibody chain controlled by a gravity law.

\textsuperscript{3}For more information on the impact of friction on robot simulation see [17]

\textsuperscript{4}The simplicity of the proposed robot model will allow us to better concentrate on specific issues related to robot simulation.
terms of generalized coordinates (e.g. joint angles) which span the constraint manifold. For open-chain problems, these methods result in a compact set of ordinary differential equations (ODE). However, these methods are not well designed to incorporate constraints that are not integrable (non-holonomic constraints) or discontinuous constraints (e.g. contacts with the environment).

Augmentation based methods (or formulations in descriptor space) start from a formulation of the dynamics in Cartesian space and restrict the motion of the bodies with constraint forces (Lagrange multipliers) applied at each joint. These methods result in a relatively large set of differential algebraic equations (DAE) (differential equations of motion with algebraic constraint equations). These equations are generally more difficult to integrate than ODEs and can be viewed as very stiff ODEs. One of the advantages of these methods is their capability of handling both holonomic and non-holonomic constraints and allowing the possibility of adding and removing such constraints on-line (without a complete reformulation of the equations).

The main focus of this thesis is the study of algorithms for simulating the motion of robot manipulators in open-chain configurations, for example robots without closed kinematic loops and robots whose motion is not constrained by contacts with the environment. For the simulation of robots in open-chain configurations, elimination based methods are usually preferred since they lead to compact sets of equations which can be directly integrated. There are two popular elimination based methods for formulating the equations of motion in configuration space: Lagrangian methods and Newton-Euler methods. Lagrangian methods are based on formulating the equations of motion of the robot using an analysis of the total energy of the system. Newton-Euler methods are based on a detailed analysis of the forces at each link. Both of these methods have been

---

Chapter 6 will introduce issues related to the simulation of contacts between a robot and its environment.
shown to lead to equivalent sets of equations [54]. They lead to equations having the following configuration space form:

\[ \tau = H(\theta)\ddot{\theta} + b(\theta, \dot{\theta}) \]

where:

- \( \tau \) is the \( n \times 1 \) vector of torques (forces) applied by the joint actuators,
- \( \ddot{\theta} \) is the \( n \times 1 \) vector of joint variables (\( \dot{\theta} \) and \( \ddot{\theta} \) are the joint velocities and accelerations),
- \( H \) is the \( n \times n \) joint-space inertia matrix (JSIM) or generalized inertia matrix,
- \( b \) is the \( n \times 1 \) bias vector representing the torques (forces) due to gravity, centrifugal and Coriolis accelerations, and any external moments and forces acting on the end effector of the manipulator.

For complex manipulators having a large number of degrees of freedom, manually writing out the equations of motion can be a very complicated and time consuming task. Many researchers have developed algorithms that automate the generation of the equations of motion (e.g. [44]) and several commercial software packages are available today (e.g. SD/FAST, ADAMS, DADS).

Once the equations of motion are formulated, they can either be solved for joint torques (the inverse dynamics problem) or for joint accelerations (the forward dynamics problem). Inverse dynamics can be easily solved in \( O(n) \) time as will be demonstrated later in this chapter. Forward dynamics is a more difficult problem and will be discussed in detail in the following chapter.
2.4 Dynamics of a single rigid body

In this section, the equations of motion of a single rigid body are developed. These equations form the basis of the following sections which develop the equations of motion for a serial chain of rigid bodies. This section also serves to introduce the spatial notation which will be used throughout this study.

2.4.1 Spatial vectors

A coordinate free spatial notation [20, 32] will be used to describe the motion of rigid bodies and the applied forces. In general, spatial vector quantities can be viewed as $6 \times 1$ column vectors which group the linear (translational) and angular (rotational) components of motion and force, while a spatial operator is a linear transformation of spatial vectors, with a $6 \times 6$ coordinate matrix.

The spatial velocity $\mathbf{v}_O$ and the spatial acceleration $\mathbf{a}_O$ of a point $O$ on a rigid body $B$ are defined as

$$
\mathbf{v}_O = \begin{bmatrix} \mathbf{v}_O \\ \mathbf{\omega}_O \end{bmatrix}, \quad \mathbf{a}_O = \begin{bmatrix} \mathbf{a}_O \\ \mathbf{\dot{\omega}}_O \end{bmatrix} = \begin{bmatrix} \mathbf{\dot{v}}_O \\ \mathbf{\ddot{w}}_O \end{bmatrix}.
$$

Where $\mathbf{v}_O$ and $\mathbf{a}_O$ are the $3 \times 1$ dimensional linear velocity and acceleration of the point $O$ on the rigid body, and $\mathbf{\omega}_O$ and $\mathbf{\ddot{\omega}}_O$ are the $3 \times 1$ dimensional angular velocity and acceleration of the rigid body. Unless otherwise specified, all velocities and accelerations are assumed to be relative to a global fixed inertial frame.

Similarly, the net spatial force $\mathbf{f}_O$ acting on a rigid body $B$ at a point $O$ is defined as

$$
\mathbf{f}_O = \begin{bmatrix} \mathbf{f}_O \\ \mathbf{r}_O \end{bmatrix}.
$$

Where $\mathbf{f}_O$ is the $3 \times 1$ dimensional force (linear) acting on the rigid body $B$ and $\mathbf{r}_O$ is the $3 \times 1$ dimensional moment (torque) acting about $O$ on the rigid body $B$. 
These definitions of spatial velocity, acceleration and force differ slightly from the definitions given by Featherstone. In essence, our definitions specify two different types of spatial vectors: spatial motion vectors (a combination of a free vector with a line vector) and spatial force vectors (a combination of a line vector with a free vector). Mappings between two spatial vectors of the same type are discussed in the following subsection. Mappings from/to spatial motion vectors to/from spatial force vectors are commonly referred to as equations of motion.

2.4.2 Rigid body transformation operator

The rigid body transformation $\Phi_{O,C}$ is defined as a spatial operator relating spatial velocities/forces at a point $O$ on a rigid body to spatial velocities/forces at a second point $C$ on the body. This transformation is a function of the vector $l_{O,C}$ connecting the two points.

$$\Phi_{O,C} \equiv \begin{bmatrix} 1 & l_{O,C} \\ 0 & 1 \end{bmatrix}, \quad \Phi_{O,C}^T \equiv \begin{bmatrix} 1 & 0 \\ (l_{O,C})^T & 1 \end{bmatrix}.$$ 

Important properties of this operator include

$$\Phi_{O,C}^{-1} = \Phi_{C,O}, \quad \Phi_{O,C} = \Phi_{O,P} \Phi_{P,C}.$$ 

where $P$ is a third point on the rigid body. Using this operator, the relationship between spatial forces at $C$ and $O$ is

$$f_O = \Phi_{C,O}^T f_C.$$ 

Which corresponds to the following well known relationships between forces and torques on a rigid body

$$f_O = f_C, \quad \tau_O = \tau_C + l_{O,C} f_C.$$ 

---

6 This transformation should not be confused with the coordinate transformation operator which will be discussed in the following subsection.

7 The coordinates of $l_{O,C}$ are assumed to be expressed with respect to frame $O$. 
Similarly, $\dot{\Phi}_{O,C}$ can be used to express the relationship between the spatial velocities at $C$ and $O$

$$\dot{v}_O = \dot{\Phi}_{O,C} \dot{v}_C.$$ 

Which corresponds to the following well known relationships between linear and angular velocities

$$\ddot{v}_O = \ddot{v}_C + \omega_C I_{C,O}, \quad \ddot{\omega}_O = \ddot{\omega}_C.$$ 

The transformation of spatial accelerations is a little more complicated. It is given by

$$\ddot{a}_O = \dot{\Phi}_{O,C} \ddot{a}_C + \dot{\rho}_C,$$

where $\dot{\rho}_C$ is the velocity dependent Coriolis and centrifugal spatial acceleration term at the center of mass

$$\dot{\rho}_C \equiv \dot{\Phi}_{O,C} \dot{\omega}_C = \begin{bmatrix} \dot{\omega}_C \dot{\omega}_C I_{C,O} \\ 0 \end{bmatrix}.$$ 

### 2.4.3 Coordinate transformation operator

The above notation is said to be coordinate free since the quantities are not bound or referenced to a specific coordinate frame. This type of notation allows us to obtain a more compact expression of the equations of motion.

However, in order to solve the equations of motion, the components of these quantities must be compared and computed with respect to common frames of reference. Therefore, coordinate frame transformation operators (rotation matrices) must be applied to the quantities considered. These operators are defined as follows:

$$\ddot{X}_{O,C} = \dot{\Phi}_{O,C} \dot{R}_{O,C},$$

where

$$\dot{R}_{O,C} = \begin{bmatrix} E_{O,C} & 0 \\ 0 & E_{O,C} \end{bmatrix},$$
where $\hat{E}_{O,C}$ is the standard orthogonal rotation transformation matrix between the coordinate frame $O$ and the coordinate frame $C$ [15].

2.4.4 Newton and Euler's equations of motion

If $C$ is the location of the center of mass on the rigid body, then the spatial inertia of the body about this point is defined as

$$M_c = \begin{bmatrix} m & 0 \\ 0 & I_c \end{bmatrix},$$

where $m$ is the total mass of the body and $I_c$ is the body's moment of inertia about its center of mass.

The relationship between the net force applied at the center of mass and the resulting linear acceleration of the body is given by Newton's equation:

$$f_c = \frac{d(m \ddot{v}_c)}{dt} = m \ddot{a}_c.$$

The relationship between the net torque and angular acceleration is given by Euler's equation:

$$\tau_c = \frac{d(I_c \ddot{\omega}_c)}{dt} = I_c \ddot{\omega}_c + \ddot{\omega}_c I_c \ddot{\omega}_c.$$

These equations result in the following spatial relationship

$$\dot{f}_c = \dot{M}_c \ddot{a}_c + \dot{\gamma}_c,$$

where $\dot{\gamma}_c$ is the velocity dependent spatial gyroscopic force acting at the center of mass:

$$\dot{\gamma}_c = \dot{M}_c \ddot{v}_c.$$

Using the previously defined spatial rigid body transformation, the relationship between the spatial forces and accelerations can be rewritten for any point $O$ on the rigid body as follows:

$$\dot{f}_O = \dot{M}_O \ddot{a}_O + \dot{\gamma}_O.$$
where
\[ \dot{M}_o \equiv \dot{X}_{c,o}^T \dot{M}_c \dot{X}_{c,o} \]
and
\[ \dot{\gamma}_o \equiv \dot{X}_{c,o}^T (\dot{M}_c \dot{\gamma}_o + \dot{\gamma}_c) = \begin{bmatrix} \ddot{\omega}_o \dot{I}_o \dot{\omega}_o \\ m \ddot{\omega}_o \ddot{\omega}_o \dot{I}_o \end{bmatrix}. \]

2.5 The inverse dynamics problem

In order to demonstrate the extension of our notation to multibody chains, this section presents an \( O(n) \) recursive algorithm [39] for solving the inverse dynamics problem. As will be shown, inverse dynamics can play a direct role in solving the forward dynamics problem. This section also serves to introduce and define certain generalized spatial operators.

The notation for developing the dynamics of multibody chains is a natural extension of the notation used in the analysis of the dynamics of a single rigid body. The following is a summary of the notation associated with multibody chains:

**Link properties:**

- \( \tilde{t}_{i,i+1} \) is the vector from the origin of link \( i \) to the origin of link \( i + 1 \),
- \( \tilde{p}_i \) is the vector from the origin of link \( i \) to the center of mass of link \( i \),
- \( \hat{M}_i \) is the spatial inertia matrix of link \( i \),
- \( \hat{X}_{i+1,i} \) is the transformation matrix from link \( i + 1 \) to link \( i \),

**Joint properties:**

- \( \hat{\theta}, \hat{\dot{\theta}}, \hat{\ddot{\theta}} \) are the generalized relative joint angles, rates and accelerations, where \( \theta_i, \dot{\theta}_i, \ddot{\theta}_i \) are the position, velocity and acceleration of joint \( i \).
• $\tau_i$ is the generalized joint torque, where $\tau_i$ is the torque applied to link $i$ about its joint axis.

• $\dot{s}_i$ is the unit axis of the joint between link $i - 1$ and link $i$ (for a revolute joint $\dot{s}_i = [0\; s_i]^T$).

**Generalized velocities and accelerations:**

• $\dot{\hat{v}}_i$ is the generalized spatial velocity, where $\hat{v}_i$ is the spatial velocity of the origin of link $i$.

• $\dot{\hat{a}}_i$ is the generalized spatial acceleration, where $\hat{a}_i$ is the spatial acceleration of the origin of link $i$.

• $\dot{\hat{p}}_i$ is the generalized spatial Coriolis and centrifugal acceleration, where $\hat{p}_i$ is the spatial Coriolis and centrifugal acceleration of the origin of link $i$.

**Generalized forces:**

• $\hat{f}_i$ is the generalized spatial inter-link force, where $\hat{f}_i$ is the spatial force exerted on link $i$ by link $i - 1$ at joint $i$.

• $\hat{\Gamma}_i$ is the generalized spatial gyroscopic force, where $\hat{\Gamma}_i$ is the spatial gyroscopic force acting on link $i$.

• $\hat{f}_{Ti}$ is the generalized spatial net force, where $\hat{f}_{Ti}$ is the net spatial force exerted on link $i$.

The inverse dynamics algorithm is based on work done by Luh, Walker and Paul [39]. The algorithm proceeds in two recursive iterations: an outward iteration from base to tip to compute the spatial velocities and accelerations of each link in the chain, and an inward iteration from tip to base to compute the link forces and joint torques. The
proposed notation leads to a compact formulation of the familiar recursive Newton-Euler formulation of inverse dynamics.

The outward recursive relationships between the spatial velocity/acceleration of link $i+1$ and the spatial velocity/acceleration of link $i$ are

$$
\dot{\mathbf{v}}_{i+1} = \dot{\mathbf{X}}_{i+1,i} \mathbf{v}_i + \ddot{s}_{i+1} \mathbf{\theta}_{i+1},
$$

$$
\dot{\mathbf{a}}_{i+1} = \dot{\mathbf{X}}_{i+1,i} \mathbf{a}_i + \ddot{s}_{i+1} \mathbf{\theta}_{i+1} + \dot{\mathbf{p}}_{i+1},
$$

where

$$
\dot{\mathbf{p}}_{i+1} = \dot{\mathbf{X}}_{i+1,i} \mathbf{v}_i + \ddot{s}_{i+1} \mathbf{\theta}_{i+1} = \begin{bmatrix}
\ddot{\omega}_i \mathbf{I}_{i+1} + \dot{\mathbf{\omega}}_i \mathbf{I}_{i+1} \\
\ddot{\mathbf{\omega}}_i \mathbf{I}_{i+1} \mathbf{\theta}_{i+1}
\end{bmatrix},
$$

for $i = 0$ to $n - 1$ with $\dot{\mathbf{v}}_0 = 0$ and $\dot{\mathbf{a}}_0 = 0$ (note: the effect of gravity can be included by setting $\dot{\mathbf{a}}_0 = \mathbf{g}$).

The inward recursive relationship to compute inter link spatial forces from the tip to the base is

$$
\dot{\mathbf{f}}_i = \dot{\mathbf{M}}_i \mathbf{a}_i + \dot{\mathbf{g}}_i + \dot{\mathbf{X}}_{i+1,i} \mathbf{f}_{i+1},
$$

where the joint torques are given by the projection

$$
\tau_i = \mathbf{s}_i^T \mathbf{f}_i,
$$

for $i = n$ to $1$ with $\mathbf{f}_{n+1} = 0$.

### 2.5.1 Generalized spatial operators

Rodriguez, Jain and Kreutz-Delgado [50] introduced a powerful generalized spatial operator algebra which allows the equations of motion of multibody chains to be expressed in a much more compact high-level form. Generalized spatial operators are $n \times n$ generalized matrices whose elements are spatial entities.\(^8\) The following is a list of some key

---

\(^8\)Generalized spatial operator are similar to the "super matrices" defined by Ellis [18]
Chapter 2. Fundamentals of Robot Manipulator Simulation

generalized spatial operators:

\[ \hat{M} = \text{diag} \begin{bmatrix} \hat{M}_1 & \hat{M}_2 & \ldots & \hat{M}_n \end{bmatrix} \]

\[ \hat{S} = \text{diag} \begin{bmatrix} \hat{s}_1 & \hat{s}_2 & \ldots & \hat{s}_n \end{bmatrix} \]

\[ \hat{\Phi} = \begin{bmatrix} \Phi_{2,1} & 1 & 0 & \ldots & 0 \\
\hat{\Phi}_{3,1} & \hat{\Phi}_{3,2} & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\hat{\Phi}_{n,1} & \hat{\Phi}_{n,2} & \hat{\Phi}_{n,3} & \ldots & 1 \end{bmatrix} \]

\[ \hat{X} = \begin{bmatrix} \hat{X}_{2,1} & 1 & 0 & \ldots & 0 \\
\hat{X}_{3,1} & \hat{X}_{3,2} & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\hat{X}_{n,1} & \hat{X}_{n,2} & \hat{X}_{n,3} & \ldots & 1 \end{bmatrix} \]

Generalized spatial operators can be interpreted as encapsulating recursive operations. Three structures are of particular interest: block lower triangular operators represent recursions from base to tip, block upper triangular operators represent recursions from tip to base and block diagonal operators are non recursive or memoryless operations. For example, \( \hat{y} = \hat{X} \hat{\dot{x}} \) is equivalent to the recursion: \( \hat{y}_{i+1} = \hat{x}_{i+1} + \hat{X}_{i+1,i} \hat{\dot{y}}_i \), and \( \hat{y} = \hat{X}^T \hat{\dot{x}} \) is equivalent to the recursion: \( \hat{y}_i = \hat{x}_i + \hat{X}^T_{i+1,i} \hat{\dot{y}}_{i+1} \). Generalized spatial operators may also be referred to as composite spatial operators since they act on the whole chain at a given configuration. For example, \( \hat{\Phi} \) is often referred to as a composite rigid body transformation.

These generalized spatial operators can be used to express the previously described inverse dynamics algorithm:

\[ \hat{u} = \hat{X} \hat{S} \hat{\dot{\theta}} \]
\[
\begin{align*}
\hat{a} &= \dot{X}(\dot{S}\ddot{\theta} + \ddot{p}) \\
\hat{f} &= \dot{X}^T(\dot{M}\hat{a} + \dot{\hat{r}}) \\
\tau &= \hat{S}^T\dot{f}
\end{align*}
\]
which can be rewritten to obtain the familiar state space expression for the equations of motion of robot manipulators:
\[
\tau = H\ddot{\theta} + b
\]
where
\[
\begin{align*}
H &= \hat{S}^T\dot{X}^T\hat{M}\dot{X}\hat{S} \\
b &= \hat{S}^T\dot{X}^T(\dot{\hat{M}}\dot{X}\ddot{p} + \dot{\hat{r}})
\end{align*}
\]

2.6 The motion integration problem

Although the motion integration problem is not the focus of this study, we introduce here some of the basic issues related to this part of the simulation process.

In robot simulation, the motion integration problem is defined as follows: given the current state of the robot (joint positions and velocities) and the current joint accelerations at time \(t\) compute the state of the robot at a time \(t + h\), where \(h\) is called the integration step size. For numerical simulation, the second order differential equations of motion are converted to a coupled first order differential equation:
\[
\frac{d}{dt} \begin{pmatrix} \theta(t) \\ \dot{\theta}(t) \end{pmatrix} = \begin{pmatrix} \ddot{\theta}(t) \\ \dddot{\theta}(t) \end{pmatrix} = f(\theta, \dot{\theta}, t)
\]

Numerical integration is not an exact procedure. It introduces truncation errors which are a function of the size of the time step and the smoothness of the derivative function. It is therefore desirable for an integrator to exert control over the step size; adjusting the
time step so as to maintain a desired level of accuracy. Such integration schemes are said to be adaptive step size methods.

In this thesis, we are principally interested in the interaction of the forward dynamics and the motion integration with respect to adaptive step size integration. In the context of an adaptive step size integrator, a forward dynamics function will be said to exhibit operational stiffness if there is a sharp increase in the number of derivative function evaluations necessary to reach a stated accuracy [30].

These issues will be discussed in chapter 4 and illustrated in chapter 5. For a more general introduction to numerical integration see [25] and for further discussions of the relationship between motion integration and robot simulation see [41, 2].
Chapter 3

Solving the Forward Dynamics Problem

Existing algorithms for solving the forward dynamics problem for open chain robot manipulators can be classified into three categories based on their computational complexity:

1. $O(n^3)$ algorithms based on inertia matrix inversion,
2. $O(n^2)$ conjugate gradient methods,
3. $O(n)$ linear recursive methods.

A good overview of the relationship between the different types of forward dynamics algorithms can be found in [32]. Jain uses a spatial operator algebra to analyze and compare the different classes of forward dynamics algorithms from a unified perspective.\(^1\)

3.1 $O(n^3)$ Algorithms: inertia matrix inversion methods

The $O(n^3)$ methods for solving the forward dynamics problem are said to be based on inertia matrix inversion. The starting point of these methods is the following state space linear equation:

$$H \ddot{\theta} = \tau - b$$  \hspace{1cm} (3.1)

These methods follow three basic steps.

1. Calculate the bias vector ($b$). This generalized vector represents the action of forces due to gravity, centrifugal and Coriolis accelerations, and any external forces

\(^1\)The algorithms reviewed in this chapter are for sequential computation. A review of parallel algorithms for robot forward dynamics can be found in [22].
acting on the end effector of the manipulator (e.g.: contact forces). The bias vector is a function of the current joint configuration $\theta$ and joint velocities $\dot{\theta}$. It can be calculated by solving the inverse dynamics problem for $\ddot{\theta} = 0$ ($b$ are joint torques necessary to cause zero acceleration). For this purpose, an efficient $O(n)$ inverse dynamics algorithm such as the one presented in the previous chapter can be used [39].

2. Calculate the joint space inertia matrix $H$ (or generalized inertia matrix). It is in the method by which this is performed that most $O(n^3)$ algorithms differ from one another. One of the most efficient methods for computing $H$ is known as the Composite Rigid Body Method (CRBM) [58]. This method has a computational complexity of $O(n^2)$. It will be described below.

3. Solve the linear equation for the joint accelerations (inertia matrix inversion step).
   This step is usually accomplished by such techniques as LU decomposition or Cholesky decomposition [28, 56]. The fact that the joint space inertia matrix is symmetric positive definite means that solving this step should not require any pivoting. The computational complexity of this step is $O(n^3)$.

Although inertia matrix inversion methods have an overall computational complexity of $O(n^3)$, the coefficient of $n^3$ is very small (e.g. 1/3) making these methods efficient for small values of $n$.

3.1.1 The composite rigid-body method

Walker and Orin [58] present three methods for computing the joint space inertia matrix (step 2 of the above procedure). Their third method, known as the composite rigid-body method (CRBM), is the most efficient of the three. In this method, a composite rigid-body $j$ is defined as a single rigid body formed by the chain of links from $j$ to $n$, where
joints \( j + 1 \) to \( n \) are considered fixed. The composite rigid body \( n \) is the distal link of the chain. The CRBM proceeds in three steps.

1. For \( j = n \) to 1, consider the composite rigid-body \( j \). Compute the position of its center of mass, its total mass and inertia matrix with respect to joint \( j \) (using the parallel axis theorem). These calculations depend only on the current geometric configuration of the manipulator.

2. For \( j = 1 \) to \( n \), consider joint \( j \) as the only joint in motion (set \( \ddot{\theta}_j = 1 \) and \( \ddot{\theta}_i = 0 \), for all \( i \neq j \)), compute the total forces and moments applied to the center of mass of each composite rigid-body \( j \).

3. For \( j = n \) to 1, for \( i = j \) to 1, calculate the forces and moments applied to link \( i - 1 \) by the composite rigid-body \( i \). If joint \( i \) is rotational, then the \((i, j)\)th component of \( H \) is equal to the \( z \) component of the computed moment. This step takes advantage of the symmetry of \( H \).

Using the generalized spatial notation, the joint space inertia matrix can be defined as follows [32]:

\[
H = \dot{\mathbf{s}}^T \hat{\mathbf{x}}^T \hat{\mathbf{m}} \hat{\mathbf{x}} \dot{\mathbf{s}}. \tag{3.2}
\]

### 3.1.2 Orthogonal complement based methods

Two \( O(n^3) \) methods for solving the forward dynamics were developed by Angeles and Ma [3]. In these methods, the concept of a natural orthogonal complement \( \mathbf{\mathbf{T}}_n \) (a \( 6n \times n \) matrix of unit twists) is used to construct the generalized inertia matrix \( H \) of the manipulator. The use of a natural orthogonal complement describing the constraints of the manipulator allows the authors to obtain a factorization of the generalized inertia matrix into \( H = \)
\( \hat{P}^T \hat{P} \), where \( \hat{P} \) is obtained through a factorization of the generalized extended mass matrix.

Their first method computes the generalized inertia matrix explicitly by calculating the product \( H = \hat{P}^T \hat{P} \) and uses Cholesky decomposition to solve the linear equations of motion for joint accelerations. Their second method does not explicitly form \( H \). Instead, the following system of equations is solved:

\[
P^T y = \tau - b \tag{3.3}
\]

\[
P \ddot{q} = y \tag{3.4}
\]

\( \hat{P} \) is a \( 6N \times N \) block lower triangular matrix; in brief, this method solves an underdetermined system followed by an overdetermined system (using the minimum-norm solution and the least squares approximation).

They observe that this second method, although less efficient than their first has better numerical properties and is not subject to as potentially high condition numbers as other \( O(n^3) \) methods. The problems related to ill-conditioning of the generalized inertia matrix will be further discussed in the following chapter.

### 3.2 \( O(n^2) \) Algorithms: conjugate gradient methods

Conjugate gradient methods provide a general \( O(n^2) \) means for solving a set of linear equations of the form: \( Ax = b \). Conjugate gradient methods are guaranteed to converge to the solution in, at most, \( n \) iterations. In each iteration the matrix \( A \) is accessed only through multiplication by a vector \( y \). The product \( Ay \) can be obtained in \( O(n) \) time using inverse dynamics.

Walker and Orin (method 4 [58]) describe an \( O(n^2) \) algorithm for solving the forward dynamics problem based on a conjugate gradient method. The procedure follows the
same basic steps as described in previous section except that in the last step, a conjugate gradient method is used to solve the linear equation. A similar $O(n^2)$ procedure is also described by Fijany and Scheid [23].

3.3 $O(n)$ Algorithms: linear recursive methods

In general, $O(n)$ methods are based on propagating the coefficients of motion and force along the robot manipulator in an inward (tip to base) and outward (base to tip) recursive procedure. These methods can be viewed as forming an equation of the following form:

$$\ddot{\theta} = F(\theta, r, \alpha)$$

(3.5)

where $F$ (the forward dynamics function) is expressed without reference to the joint space inertia matrix.

One of the first such procedures was developed by Vereshchagin [57]. Later Armstrong [4] also demonstrated the existence of a linear recursive relationship between the motion of and forces applied to link $i$, and the motion of and forces applied to links $i - 1$ and $i + 1$ of a robot manipulator. This recursive relationship allows the joint accelerations to be determined in $O(n)$ time. However, the coefficient of $n$ is rather large.

One of the most familiar linear recursive methods is Featherstone's articulated-body method (ABM) [19, 20]. Featherstone obtains a reduction in the number and size of equations by the introduction of a spatial notation for accelerations, inertias and forces which combines the translational and rotational components of motion. Featherstone also defines the concept of articulated-body inertias: “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”. In other terms, the spatial articulated-body inertia $k$ is: the effective spatial inertia at the
Chapter 3. Solving the Forward Dynamics Problem

$k$th joint of links $k$ to $n$ taken as a single non-actuated articulated-body. The articulated body method will be presented in detail later in this section.

Another $O(n)$ method is presented by Brandl, Johanni and Otter [11]. This method is very similar to Featherstone's algorithm, but uses a more general joint model to allow for the modeling of multiple degree of freedom joints. It also obtains a better performance than Featherstone's by optimizing vector and matrix coordinate transformations.

Finally, Rodriguez, Jain and Kreutz-Delgado [49, 50] also present a recursive $O(n)$ method. This method is based on developing an analogy between robot dynamics and techniques from linear filtering and smoothing theory (Kalman filtering and Bryson-Frazier smoothing). The method uses a specially developed spatial algebra based on a spatial notation similar to the one introduced by Featherstone. Although the resulting computations are very similar to the ones of Featherstone, the means by which the equations are derived are very different. This method does not require an initial step to calculate the bias terms (inverse dynamics step), the bias term is naturally incorporated in the formulation in such a way that only a single inward filtering and outward smoothing iteration is necessary to solve the forward dynamics problem. Featherstone in [20] also developed an AB algorithm that did not require pre-computation of the bias term.

3.3.1 The articulated-body method

The articulated-body method proceeds in three steps.

1. Calculate the bias vector ($b$). This is accomplished using an inverse dynamics procedure (as was done in step 1 of the $O(n^3)$ procedure).

\footnote{For multibody dynamics formulations supporting multiple degree of freedom joints see also [48, 36].}
2. Inward recursion. Moving from the tip to the base, calculate the inertias of each articulated-body $I^A_i$ and the associated correction forces $\hat{p}_i$ (these forces incorporate the effects of non-zero forces in the joints of the articulated-bodies). These quantities enable the calculation of the spatial forces applied at each joint.

$$ I^A_i = \tilde{M}_i + \dot{X}_{i+1,i}^T (I^A_i - \frac{\dot{I}^A_{i+1}}{\dot{s}_{i+1}} \dot{X}_{i+1,i}) \dot{X}_{i+1,i}, \quad I_n = \tilde{M}_n \tag{3.6} $$

$$ \hat{p}_i = \dot{X}_{i+1,i}^T (\hat{p}_{i+1} + \frac{\dot{I}^A_{i+1}}{\dot{s}_{i+1}} ((\tau_{i+1} - b_{i+1}) - \dot{s}^T_{i+1} \dot{p}_{i+1})), \quad \hat{p}_n = 0 \tag{3.7} $$

3. Outward recursion. Calculate the link accelerations $\dot{a}_i$ and joint accelerations $\ddot{\theta}_i$ by a recursive propagation from the base to the tip of the robot.

$$ \ddot{a}_i = \dot{X}_{i,i-1}^T \ddot{a}_{i-1} + \dot{s}_i \ddot{\theta}_i, \quad \ddot{a}_0 = 0 \tag{3.8} $$

$$ \ddot{\theta}_i = \frac{(\tau_i - b_i) - \dot{s}^T_i \dot{I}^A_i \dot{X}_{i,i-1} \ddot{a}_{i-1} - \dot{s}^T_i \dot{p}_i}{\dot{s}^T_i \dot{I}^A_i \dot{s}_i} \tag{3.9} $$

Although this algorithm has a computational complexity of $O(n)$ it is shown to be less efficient than Walker and Orin's composite-rigid body method for $n < 9$ [20].

**Physical interpretation: force and acceleration propagators**

The inward recursive iterations of the articulated body method serve to encode the inertial properties of the arm and to propagate bias forces from the tip to the base. The force propagator $\hat{P}_{Fi}$ can be defined as

$$ \hat{P}_{Fi} = 1 - \frac{\dot{I}^A_i \dot{s}_i \dot{s}_i^T}{\dot{s}_i^T \dot{I}^A_i \dot{s}_i}. \tag{3.10} $$

Using the force propagator, the tip to base recursion of the articulated-body method can be rewritten as follows

$$ \dot{I}^A_i = \dot{I}_i + \dot{X}_{i+1,i}^T \hat{P}_{Fi+1} \dot{I}^A_{i+1} \dot{X}_{i+1,i}, \tag{3.11} $$
and
\[ \hat{\dot{r}}_i = \hat{\dot{X}}_{i+1,i}^T (\hat{P}_{A,i} \hat{\dot{r}}_{i+1} + \frac{\hat{f}_{i+1}^A \hat{s}_{i+1}(\tau_{i+1} - b_{i+1})}{\hat{s}_{i+1}^T \hat{I}_{i+1}^A \hat{s}_{i+1}}). \] (3.12)

The outward recursive iteration of the articulated body method propagates accelerations from the base to the tip. The unitless acceleration propagator operator \( \hat{P}_{A,i} \) can be defined as
\[ \hat{P}_{A,i} = (1 - \frac{\hat{s}_i \hat{s}_i^T \hat{I}_i^A}{\hat{s}_i^T \hat{I}_i^A \hat{s}_i}). \] (3.13)

Using the acceleration propagator, the tip to base recursion can be rewritten as follows
\[ \hat{a}_i = \hat{P}_{A,i} \hat{\dot{X}}_{i,i-1} \hat{a}_{i-1} + \frac{\hat{s}_i(\tau_i - b_i - \hat{s}_i^T \hat{\dot{r}}_i)}{\hat{s}_i^T \hat{I}_i^A \hat{s}_i}. \] (3.14)

It is interesting to note that this operator is in fact a weighted least squares projection operator. In other words, the acceleration propagator could be interpreted as an operator that projects motion vectors along the direction that moves the least mass. This would be in agreement with Gauss’ principle of least constraint [34, 57].

### 3.4 Summary and comparisons

In an attempt to better compare the various forward dynamics algorithms, Jain [32] used the generalized spatial notation, to obtain a closed-form expression for the JSIM based on generalized spatial operators. Jain showed that the interpretation of the generalized operators in this expression leads directly to the ABM algorithm. In [5], using a unified formulation of multibody dynamics and basic linear algebra, it is shown that the CRBM and ABM are in fact simply different elimination methods for solving the same system of algebraic equations of motion.

At a first glance, a good way to compare the performance of the CRBM and the ABM is to look at the number of operations required. The following table summarizes

\[ \text{This type of spatial transformation can be called a spatial articulated transformation as opposed to a composite-rigid body transformation [32].} \]
the operation counts for the two methods. These results were obtained by Featherstone [20] and account for the fact that each method has been optimized so as to remove redundant calculations.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of multiplications</th>
<th>Number of additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABM</td>
<td>$300n - 267$</td>
<td>$279n - 259$</td>
</tr>
<tr>
<td>CRBM</td>
<td>$n^3/6 + 11.5n^2 +$</td>
<td>$n^3/6 + 7n^2 +$</td>
</tr>
<tr>
<td>(Cholesky)</td>
<td>$(160 + 1/3)n - 109$</td>
<td>$(138 + 5/6)n - 102$</td>
</tr>
</tbody>
</table>

Based on this table, in terms of the number of operations, the ABM is more efficient than the CRBM for $n > 9$. However, as will be demonstrated in the following chapters, the performance of a simulation program does not only depend on the efficiency of the forward dynamics algorithm but also on its numerical performance. Most of the literature on forward dynamics does not discuss stability and other numerical issues related to the implementation of these methods [18, 55].
In this chapter, the focus will be placed on the analysis of the numerical properties of two of the most popular methods for performing forward dynamics:

- the Composite Rigid-Body Method (CRBM)
- the Articulated-Body Method (ABM)

In addition to examining the numerical properties of these forward dynamics methods, we will examine the effects on the whole simulation process (i.e. including motion integration).

4.1 Review of issues in numerical computation

Roundoff errors

Computers store floating-point numbers with finite precision. The machine accuracy $\epsilon_m$ is defined as the smallest floating-point number that when added to 1.0 gives a result different from 1.0. A computer operation between two floating-point numbers adds a fractional error of at least $\epsilon_m$ to the results. This error is called the roundoff error. For example, subtraction of two nearly equal numbers can result in relatively large roundoff errors. A good numerical algorithm should attempt to avoid such inaccuracies.

One of the important issues in numerical computation is determining how roundoff errors accumulate in successive/iterative computations. An algorithm will be numerically unstable if roundoff error is propagated in such a way that the errors are magnified.
Some of the current methods for estimating the numerical stability of algorithms include interval analysis and statistical analysis. A very good reference on the topic of roundoff errors in numerical computation is [60].

Matrix computations

The condition number $\kappa(A) = \|A\| \|A^{-1}\|$ is a popular way of measuring the sensitivity of a linear system $Ax = b$. The two-norm condition number of a matrix corresponds to the ratio of the largest singular value to the smallest singular value [28]. A large condition number indicates a poorly conditioned system. The condition number is only a worst case estimate of the potential sensitivity to errors, it can only provide a bound on the relative errors. If $\kappa(A)$ is the condition number of $A$ and $\delta b$ is a perturbation in $b$ then $\delta x$, the perturbation in $x$, will be bound by:

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A) \frac{\|\delta b\|}{\|b\|}.$$ \hfill (4.1)

Truncation errors

Approximate numerical algorithms result in truncation error. In the case of robot simulation, truncation error is introduced only in the numerical integration process since forward dynamics is an ”exact” algorithm. Adaptive step size integration schemes provide a good means by which truncation error can be controlled.

4.2 Numerical properties of the CRBM

As previously discussed, an $O(n^3)$ forward dynamics algorithm typically calculates the joint space inertia matrix (JSIM) completely, and then solves an n-dimensional linear system. Forming the JSIM can contribute to a loss of precision due to round-off errors (round-off errors occur when relatively "large" quantities are added to smaller ones during
the computation of the JSIM). Using different linear solution algorithms does not help since once the JSIM is formed, the precision is lost and cannot be easily recovered. Furthermore, the JSIM is symmetric positive definite (SPD) and therefore should not require any pivoting to better solve the linear system unless roundoff errors are such that the matrix is no longer SPD.

However, the JSIM could be poorly conditioned. In robot simulation, ill-conditioning of the JSIM is more common than generally appreciated. Popular robot designs have intersecting joint axes at the base of the manipulator and at its wrist, leading to one or more links of small mass and link length. In addition, there are typically large discrepancies in link length and mass between links of the regional structure and those of the orientation structure. Some ill-conditioning may be caused by the process of modeling, when the masses of some links are neglected or poorly approximated. Finally, the condition number is also a function of the configuration of the manipulator.

Ill-conditioning of the JSIM is an important indication of difficulty. This follows from the well known properties of the sensitivity of linear systems solutions [28]. Suppose the final step of the CRBM is subjected to $O(\epsilon)$ perturbations

$$ (H + \epsilon G )\ddot{q} (\epsilon ) = \tau - b + \epsilon g , \quad (4.2) $$

where $G \in \mathbb{R}^{N \times N}$, $g \in \mathbb{R}^{N}$ and $\kappa(H) = \|H\|\|H^{-1}\|$ is the condition number, and the relative error in quantity $x$ is written as

$$ \rho(x) = \|x(\epsilon) - x\|/\|x\| \quad (4.3) $$

then the relative error in $\ddot{q}$ is bounded above as follows

$$ \rho(\ddot{q}) \leq \kappa(H)[\rho(H) + \rho(\tau - b)] + O(\epsilon^2) \quad (4.4) $$

We observe that this bound is met in practice, as can be seen in the experimental results of chapter 5.
4.3 Numerical properties of the ABM

We now show that the \(O(n)\) articulated-body method naturally possesses good numerical properties for ill-conditioned problems. A precise analysis of roundoff error propagation in this case is difficult to perform but the results of this section and the numerical experiments of the following chapter clearly demonstrate the phenomenon.

In this section we will consider an example which will illustrate the numerical differences between the forward dynamics methods and will provide an explanation for the ABM's observed robustness to ill-conditioning. A more detailed analysis of this phenomenon in terms of underlying matrix factorizations can be found in [5]. In related work, [18] showed that the AB algorithm is equivalent to performing Gaussian elimination on a symmetric positive definite matrix, and therefore stable as long as the positive definiteness is maintained. However, they did not observe any numerical superiority of the ABM over other methods such as CRBM.

Consider the example of a two link manipulator. To compute \(\ddot{\theta}_1\) using the ABM (see previous chapter), we solve

\[
\ddot{\theta}_1 = \frac{c_2}{\dot{s}_1^T \hat{I}_1^A \dot{s}_1}
\]

where \(c_2\) is a computed bias force and \(\hat{I}_1^A\) is the articulated-body inertia of link 1

\[
c_2 = (\tau_1 - b_1) - \dot{s}_1^T \ddot{X}_{2,1}^T \hat{M}_2 \ddot{s}_2 (\ddot{s}_2^T \hat{I}_2 \ddot{s}_2)^{-1} (\tau_2 - b_2).
\]

\[
\hat{I}_1^A = M_1 + \ddot{X}_{2,1}^T (\hat{M}_2 - \hat{M}_2 \ddot{s}_2 \ddot{s}_2^T \hat{M}_2 (\ddot{s}_2^T \hat{M}_2 \ddot{s}_2)^{-1}) \ddot{X}_{2,1}.
\]

Writing out the expression for \(\ddot{\theta}_1\) gives:

\[
\ddot{\theta}_{1,ABM} = \frac{(\tau_1 - b_1) - \dot{s}_1^T \ddot{X}_{2,1}^T \hat{M}_2 \ddot{s}_2 (\ddot{s}_2^T \hat{I}_2 \ddot{s}_2)^{-1} (\tau_2 - b_2)}{\dot{s}_1^T (M_1 + \ddot{X}_{2,1}^T (\hat{M}_2 - \hat{M}_2 \ddot{s}_2 \ddot{s}_2^T \hat{M}_2 (\ddot{s}_2^T \hat{M}_2 \ddot{s}_2)^{-1}) \ddot{X}_{2,1}) \ddot{s}_1}
\]
Now consider the CRBM. For the two link robot manipulator, the joint space inertia matrix can be written as (see, e.g. [36])

\[
H = \begin{pmatrix}
\hat{s}_1^T (\dot{M}_1 + \dot{\hat{X}}_{2,1}^T \dot{\hat{M}}_2 \hat{X}_{2,1}) \hat{s}_1 & \hat{s}_1^T \dot{\hat{X}}_{2,1}^T \dot{\hat{M}}_2 \dot{s}_2 \\
\hat{s}_2^T \dot{\hat{M}}_2 \hat{X}_{2,1} \hat{s}_1 & \hat{s}_2^T \dot{\hat{M}}_2 \dot{s}_2
\end{pmatrix}
\] (4.9)

To solve for \(\ddot{q}_1\) using the \(2 \times 2\) matrix \(H\), we eliminate the upper right element by a block-row operation (multiply the second row by \(\hat{s}_2^T \dot{\hat{M}}_2 \dot{s}_2\) and subtract it from the first row). This gives the following expression for \(\ddot{\hat{q}}_1\):

\[
\ddot{\hat{q}}_{1,CRBM} = \frac{(\tau_1 - b_1) - \hat{s}_1^T \dot{\hat{X}}_{2,1}^T \dot{\hat{M}}_2 \hat{s}_2 (\hat{s}_2^T \dot{\hat{M}}_2 \hat{s}_2)^{-1} (\tau_2 - b_2)}{\hat{s}_1^T (\dot{M}_1 + \dot{\hat{X}}_{2,1}^T \dot{\hat{M}}_2 \hat{X}_{2,1}) \hat{s}_1 - \hat{s}_1^T \dot{\hat{X}}_{2,1}^T \dot{\hat{M}}_2 \hat{s}_2 (\hat{s}_2^T \dot{\hat{M}}_2 \hat{s}_2)^{-1} \hat{s}_2^T \dot{\hat{M}}_2 \hat{X}_{2,1} \hat{s}_1}.
\] (4.10)

This expression is of course equivalent to equation 4.8. The numerator is computed in exactly the same order in both cases. The difference between the ABM and CRBM lies in the order in which the denominator \((\hat{s}_1^T \dot{\hat{I}}_1^A \hat{s}_1)\) is computed.

Suppose the distal inertia \((\dot{M}_2)\) is much larger than the proximal inertia \((\dot{M}_1)\) (ill-conditioned chain). In the CRBM we must, while forming \(H\), add the small \(\dot{M}_1\) to the large \(\dot{\hat{X}}_{2,1}^T \dot{\hat{M}}_2 \hat{X}_{2,1}\). Then the other large term \(\hat{s}_1^T \dot{\hat{X}}_{2,1}^T \dot{\hat{M}}_2 \hat{s}_2 (\hat{s}_2^T \dot{\hat{M}}_2 \hat{s}_2)^{-1} \hat{s}_2^T \dot{\hat{M}}_2 \hat{X}_{2,1} \hat{s}_1\) is subtracted during the solution process. This may cause cancellation error which appears as a random function of time \(t\), even though all the quantities are smooth functions of \(t\). This error does not appear in the ABM algorithm.

However, it is important to take note that the ABM can be implemented so as to perform poorly in the presence of ill-conditioning. For example, if the articulated body inertia is computed in the following manner:

\[
\dot{\hat{I}}_1^A = (\dot{M}_1 + \dot{\hat{X}}_{2,1}^T \dot{\hat{M}}_2 \hat{X}_{2,1}) - \dot{\hat{X}}_{2,1} \frac{T \dot{\hat{M}}_2 \hat{s}_2 T \dot{\hat{M}}_2 \hat{s}_2}{{\hat{s}_2}^T \dot{\hat{M}}_2 \hat{s}_2} \dot{\hat{X}}_{2,1}
\]

then cancellation error may also appear.
4.4 Impact on motion integration: formulation stiffness

The numerical simulation problem is usually treated as two separate problems: the forward dynamics problem for computing system accelerations, and the numerical integration problem for advancing the state in time. However, when the noise caused by cancellation error in the CRBM becomes relatively large, the forward dynamics presents a nonsmooth solution profile. This profile may slow down an unsuspecting adaptive step size motion integration routine. In such cases, the interaction between the forward dynamics and the numerical integration is an important factor that must be considered when evaluating the overall efficiency of the robot simulation process.

Define the formulation stiffness in multibody simulation as any operational stiffness encountered by the motion integration routine that is due to a poor numerical formulation of the simulation algorithm (i.e. any stiffness that can be avoided by a reformulation of the solution algorithm). Based on this definition, we may state the following: the CRBM will exhibit formulation stiffness when used in the simulation of ill-conditioned multibody chains. The ABM should perform better than the CRBM in ill-conditioned situations. This will be experimentally verified in the following chapter.
In order to observe the numerical properties discussed in the previous chapter, three computer programs were developed: a program for testing only the forward dynamics component, an off-line robot simulation program, and an animation program used to visualize the simulation results. The first part of this chapter describes the implementation of these programs and the following sections present some of the results obtained with this program. In section 5.2, we first focus on the numerical performance of the forward dynamics component before broadening the analysis to incorporate the whole simulation process (forward dynamics and motion integration). Section 5.3 presents some other interesting operational differences between the CRBM and ABM algorithms.

### 5.1 PRRRobot simulation software

Robot simulation software was developed for comparing the properties of the CRBM and ABM algorithms. The software consists of a specialized program to test and compare forward dynamics (called `prrrfdt`), an off-line robot simulation program (called `prrrsim`) and an animation program (called `prrranim`) used to visualize the simulation results. Both the ABM and CRBM algorithms were implemented for the basic case of a general n-link planar, revolute jointed, rectangular linked robot.

#### 5.1.1 Assumptions and definitions

In general, a robot simulation program manages the following types of data:
• Robot parameters: characteristics of the robot that remain constant throughout the simulation (e.g. link lengths, weights, widths, joint friction coefficients and link moments of inertias).

• Robot variables (state): characteristics of the robot that may vary at each time step (e.g. joint positions, velocities and accelerations).

• Environment parameters: characteristics of the environment that remain constant throughout the simulation (e.g. gravity, environment friction coefficients).

• Environment variables (state): characteristics of the robot that may vary at each time step (for simulation in dynamic environments, e.g. positions of moving obstacles).  

• Simulation parameters: e.g. numerical integration accuracy, minimal step size.

• Simulation statistics and results: e.g. history of joint positions, number of forward dynamics function evaluations, execution time, energy measures.

The implementations consider a special class of robot manipulators. These robots have the following attributes:

• robot joints are single axis revolute joints and have a full 360 degree freedom of motion (inter-link collisions are not considered) the joints are also considered to be free of friction,

• motion of the robot motion is constrained to a two dimensional plane (i.e. all joint axis are parallel),

\[1\text{At this stage, robot interactions with the environment are not considered. Therefore, environment variables are non-existent and environment parameters are greatly simplified. See chapter 6 for a discussion of the issues involved in simulating environment interactions.}\]
5.1.2 Forward dynamics test program

The PRRRobot forward dynamics test program systematically compares the different forward dynamics solutions for a two link PRRRobot by performing a static sweep through the possible joint configurations. For example, the program may be configured to compute the forward dynamics for discrete values of \( \theta_2 \in [0, 2\pi] \) while maintaining all other input variables constant. For each configuration of the chain, the program computes the joint accelerations using the articulated-body method (ABM) and the composite rigid body method (CRBM). The relative difference between the two solutions is then
computed as follows:

\[ \Delta_r \ddot{q}_i = \frac{|(\ddot{q}_{i,ABM} - \ddot{q}_{i,CRBM})/\ddot{q}_{i,ABM}|}{\ddot{q}_{i,ABM}} \]

where:

- \( \ddot{q}_{i,ABM} \) is the computed joint i acceleration using the ABM,
- \( \ddot{q}_{i,CRBM} \) is the computed joint i acceleration using the CRBM,
- \( \Delta_r \ddot{q}_i \) is the relative difference between the joint i accelerations computed using the ABM and the CRBM. If we consider \( \ddot{q}_{i,ABM} \) to be the "correct" solution, then \( \Delta_r \ddot{q}_i \) represents the relative error in \( \ddot{q}_{i,CRBM} \).

In this chapter, the prrrfidd program will be used to illustrate the effect of ill-conditioning on the CRBM and ABM forward dynamics algorithms.

### 5.1.3 Off-line simulation program

The off-line simulation program (prrrsim) reads the robot description from an input file and writes the simulation results to an output animation file and to various statistics files. The user may choose from two distinct forward dynamics algorithms (ABM or CRBM), various linear solution methods (e.g. Cholesky or LU decomposition) and various numerical integration methods (e.g. Runge-Kutta with adaptive or fixed step size). Figure 5.2 illustrates some of the run-time options provided by the program.

**Performance measures**

The PRRRobot simulation program generates statistics which allow monitoring and comparison of the numerical performance of each forward dynamics method over the whole simulation process. The statistics are generated at a user-specified frequency and are compiled into several output files. These statistics include:
Figure 5.2: PRRRobot off-line simulator interface and options.
• The evolution of joint positions \( q(t) \), velocities \( \dot{q}(t) \) and accelerations \( \ddot{q}(t) \).

• The evolution of the total energy \( E(t) \) (kinetic and potential) of the robot. This can be used to confirm the conservation of energy throughout a simulation.

• The number of forward dynamics function calls needed to advance the solution by \( \Delta t \) (a user-specified update frequency). This number is a good measure of the evolution of the computational effort required to simulate a particular motion.

• The evolution of the condition number \( \kappa(H(t)) \) of the joint space inertia matrix of the robot. The two-norm condition number of a matrix is equal to the ratio of its largest singular value to its smallest singular value. In this implementation, the singular values are found using the singular value decomposition Numerical Recipes function \([46]\) (not a particularly efficient technique, but it does the trick).

• The evolution of the relative difference between joint accelerations \( \Delta \ddot{q}_i(t) \) calculated using the CRBM vs. joint accelerations calculated using the ABM based on the current CRBM solution (see equation 5.1.2). \(^2\)

5.1.4 Animation program

The animation program \(\text{prrranim}\) was developed to enable simple visualization of the simulation results. This graphical display program was implemented using VOGL, a portable graphics library based on SGI's GL and available from the University of Melbourne via ftp.

\(^2\)To compute the relative difference at time \( t \), both the ABM and CRBM accelerations are computed using the joint positions and velocities of the CRBM solution at time \( t \).
5.1.5 Implementation issues

This section presents some of the details related to the implementation of the forward dynamics and simulation programs. The validity of the implementations was confirmed by comparing the obtained results to those of a commercially available simulation package (SD/FAST\(^3\)). \(^4\)

Computation of inverse dynamics

As was mentioned in chapter 3, the calculation of the inverse dynamics is used to compute the bias vector in both the ABM and CRBM forward dynamics algorithms. An efficient inverse dynamics procedure for a general \(n\) link PRRobot was implemented. The inverse dynamics procedure is based on a recursive Newton-Euler formulation of the dynamics developed by Luh, Walker and Paul [39]. For the purposes of this project, the equations have been simplified by taking advantage of the attributes of the special class of robots being considered (PRRRobots). In particular, the following simplifications were possible:

- For 2D motion, the rotational quantities: inertia matrices, torques, angular velocities and accelerations can be represented by scalars instead of matrices and vectors (these scalars implicitly represent the component of rotation about the \(z\) axis). In addition, these values are invariant to changes in coordinate systems since all coordinate systems have the same \(z\) axis. \(^5\)

- In 2D, forces, linear velocities and accelerations can be represented by 2D vectors instead of 3D vectors.

\(^3\)SD/FAST is a trademark of Symbolic Dynamics, Inc., 561 Bush Street, Mountain View, CA 94041 USA.

\(^4\)All programs were implemented in C and are highly portable. Versions of these programs were developed for PC compatibles, for SUN type workstations and for SCI workstations. For more details on the software implementations, the reader is encouraged to examine the documented source code.

\(^5\)Inertia matrices are transformed based on the parallel axis theorem.
• In 2D, the gyroscopic torque \((\omega \times I\omega)\) is zero.

• By taking advantage of the rectangular structure of the links and the uniform mass distribution assumption, the calculation of the positions of the link center of mass and the calculation of the link inertias is also greatly simplified.

Composite rigid-body forward dynamics

Walker and Orin's composite-rigid body algorithm for computing the joint-space inertia matrix was implemented. The implementation takes full advantage of the simple structure of the special class of planar robots being considered. In addition to the possible simplifications which were stated in the previous section, the calculation of the composite-rigid-body inertias simplify to simple scalar operations.

Once the JSIM is computed, and once the bias forces are calculated using the inverse dynamics algorithm, a linear system of equations must be solved in order to obtain the joint accelerations. For this step, the simulation program offers a choice of several different linear system solvers ([46]):

• Cholesky decomposition (CD) having \(O(1/6n^3)\) running time (CD can be used since the JSIM is positive definite),

• LU decomposition with partial pivoting having \(O(1/3n^3)\) running time,

• Gauss Jordan elimination (GJE) with full pivoting,

• Singular value decomposition (SVD).

Although theoretically pivoting is not required, given that the JSIM is symmetric positive definite, it is possible that the numerical condition of the JSIM could degrade to a point where it becomes no longer positive definite. It is only in this case that algorithms such
as LU decomposition with partial pivoting would be preferred over Cholesky decomposition. The Numerical Recipes [46] implementations of these algorithms were used in our simulation program.

Articulated-body forward dynamics

Featherstone's articulated-body algorithm [19] was implemented. Due to the simplified class of robots being considered, the following simplifications were possible:

- In 3D, spatial articulated-body inertias and spatial coordinate transformations are represented by $6 \times 6$ matrices (36 components). However, for the planar robots being considered, only 9 components need actually be computed ($\mathbf{I}_{1,3}^A, \mathbf{I}_{1,4}^A, \mathbf{I}_{1,5}^A, \mathbf{I}_{2,3}^A, \mathbf{I}_{2,4}^A, \mathbf{I}_{2,5}^A, \mathbf{I}_{6,3}^A, \mathbf{I}_{6,4}^A$ and $\mathbf{I}_{6,5}^A$).

- In 2D, spatial forces and motions can be represented using 3-vectors instead of 6-vectors (e.g. spatial forces can be represented by three components: the linear forces in $x$ and $y$ and a torque about $z$).

Motion integration

The main ODE integration method used is a fifth-order Runge-Kutta method with adaptive step size control. Other numerical integration methods are also provided, including two integrators for stiff ODE: an adaptive step size Fourth-order Rosenbrock method and an adaptive step size semi-implicit method (Burlirsch-Stoer). These methods will not be discussed in this study. They were found to be much less efficient and difficult to adjust. The simulation program uses the Numerical Recipes [46] routines to perform all numerical integration.
Chapter 5. Implementation and Experimental Verification

5.2 Results and observations: simulation of ill-conditioned systems

5.2.1 Performance of the forward dynamics

In this section, simple two link chains are used to illustrate the effect of a poorly conditioned JSIM on individual computations of the forward dynamics. Ill-conditioning is obtained by increasing the length and mass of the second link while keeping the first link's dimensions constant. To illustrate the effect of joint configuration on the condition number, the forward dynamics test program `prrfstdt` was used.

The forward dynamics were computed under the following conditions: gravity of 9.8 $m/s^2$, no friction, no applied actuator torques, zero initial joint positions and velocities, unit link widths, first link of unit length and mass, and second (distal) link length and mass were varied from 1.0 to 1.0e6. Cholesky decomposition was used to solve the linear system in the CRBM (note that similar results were obtained using LU decomposition with pivoting). For all computations, a double precision floating point representation (15 digits of precision) was used. Thus giving rise to relative errors ($\rho$) in the representation on the order of $10^{-16}$.

Figure 5.3 shows, for one particular two link chain ($l_2 = m_2 = 10^2$), the effect of the configuration on the condition number and computed accelerations. First, observe that although the configuration has a direct effect on the condition number ($\text{cnum} = \kappa(H)$), the magnitude of the condition number remains on the order of $10^4$ (such high condition numbers are actually not uncommon, for example the Stanford Arm is known to exhibit condition numbers as high as 11934 [3]).

Next, observe that the size of the relative error between the CRBM and the ABM solution ($\Delta_\mathcal{r}\hat{\theta}_1$) follows the condition number and is on the order of $10^{-12}$. As expected, this indicates that each forward dynamics implementation deals with ill-conditioning in a different way. It also illustrates the relationship between the condition number and the
Figure 5.3: Effect of configuration on the condition number of 2 link chain. Note that the condition number is largest at the singularities in this case. The relative difference for joint 1 accelerations ($\Delta_1, \dot{\theta}_1$) is also shown; results for joint 2 accelerations are similar.
relative error described in the previous chapter

\[ \Delta_{r} \ddot{\theta}_1 \approx 10^{-12} \leq 10^{4} \times 10^{-16} = \kappa H^2 \rho. \]  

(5.1)

Figure 5.4 summarizes the results obtained for several different lengths (and masses) of the distal link. Once again, this illustrates the direct relationship between the condition number and the relative error in the CRBM (note the logarithmic scales).

Figure 5.5 depicts the computed accelerations for the ABM and CRBM for a particularly ill-conditioned linkage. This figure shows that the actual solution given by the ABM is smooth and that the solution given by the CRBM is not. In effect, this illustrates the poor behaviour of the CRBM in ill-conditioned situations and the exceptional behaviour of the ABM in these situations.

In summary, these results show that the ABM does in fact deal better with ill-conditioned systems than the CRBM. Specifically, the size of the relative error in the CRBM solution is a function of the condition number of the JSIM and the floating point
Chapter 5. Implementation and Experimental Verification

Figure 5.5: Comparison of calculated joint 1 accelerations (with $l[2]=m[2]=1e5$). Note the smoothness of the ABM solution.
representation being used.

5.2.2 Performance of the motion integration

Now that we have illustrated the numerical properties of the forward dynamics, we consider how these properties carry over into the numerical motion integration.

Results for the simulation of 10 seconds of motion of various two link planar arms are summarized in Figure 5.6. The forward dynamics were computed under similar conditions to those described in the previous section, with the exception that, in this case, each chain has the same total mass and length, only the ratio of the link lengths and masses is varied (this ensures that each system has the same total energy).

Figure 5.6 displays the link length ratio ($l_1/(l_1 + l_2)$), the order of magnitude of the maximum condition number ($\kappa_{\text{max}}$), the order of magnitude of the maximum relative error in individual evaluations of the CRBM forward dynamics ($\Delta, \ddot{q}_{1,\text{max}}$), the number of forward dynamics function evaluations for each method ($n_{\text{ABM}}, n_{\text{CRBM}}$), and the ratio $r_{\text{CRBM}} = n_{\text{CRBM}}/n_{\text{ABM}}$. For these simulations, an integration step size accuracy of $1e-6$ was used.

Notice that when the relative errors in the computed accelerations become greater than the desired step accuracy, the number of function evaluations in the CRB based simulation increases dramatically (in the last case, the integration for the CRBM failed to converge within a specified maximum number of iterations). This increase in the number of forward dynamics function evaluations is due to the adaptive step size integrator perceiving and reacting to the fluctuations in the computed accelerations (see figure 5.5). This effectively illustrates the phenomenon we refer to as "formulation stiffness".

Although the results of figure 5.6 demonstrate an extreme case, similar problems can arise at much lower condition numbers. For example, in time critical simulations, single precision floating point representations (7 digits of precision) might be used in order to
Chapter 5. Implementation and Experimental Verification

<table>
<thead>
<tr>
<th>( \frac{l_1}{l_1 + l_2} )</th>
<th>( \kappa_{\text{max}} )</th>
<th>( \Delta_r \theta_{1\text{max}} )</th>
<th>( n_{\text{ABM}} )</th>
<th>( n_{\text{CRBM}} )</th>
<th>( r_{\text{CRBM}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e-1</td>
<td>1e2</td>
<td>1e-14</td>
<td>6526</td>
<td>6553</td>
<td>1.00</td>
</tr>
<tr>
<td>1e-3</td>
<td>1e6</td>
<td>1e-10</td>
<td>31424</td>
<td>31469</td>
<td>1.00</td>
</tr>
<tr>
<td>1e-5</td>
<td>1e8</td>
<td>1e-8</td>
<td>23787</td>
<td>23756</td>
<td>1.00</td>
</tr>
<tr>
<td>1e-6</td>
<td>1e9</td>
<td>1e-7</td>
<td>23825</td>
<td>23888</td>
<td>1.00</td>
</tr>
<tr>
<td>1e-7</td>
<td>1e10</td>
<td>1e-6</td>
<td>23797</td>
<td>23779</td>
<td>1.00</td>
</tr>
<tr>
<td>1e-8</td>
<td>1e11</td>
<td>1e-5</td>
<td>23812</td>
<td>24395</td>
<td>1.02</td>
</tr>
<tr>
<td>4e-9</td>
<td>1e11</td>
<td>1e-5</td>
<td>23828</td>
<td>25392</td>
<td>1.07</td>
</tr>
<tr>
<td>2e-9</td>
<td>1e11</td>
<td>1e-5</td>
<td>23792</td>
<td>31015</td>
<td>1.30</td>
</tr>
<tr>
<td>1e-9</td>
<td>1e12</td>
<td>1e-4</td>
<td>23792</td>
<td>64065</td>
<td>2.69</td>
</tr>
<tr>
<td>1e-10</td>
<td>1e13</td>
<td>1e-3</td>
<td>23822</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Figure 5.6: Number of forward dynamics evaluations required to simulate 10 seconds of motion for various two link chains of increasing condition number.

accelerate the mathematical computations. In such cases, the “formulation stiffness” would occur at much lower condition numbers. The situation would also be worse if a specific application required a very high level of step accuracy. Furthermore, the examples in this section dealt only with two link chains, however, preliminary studies have shown that these effects are made worse by the addition of more links to the chain.

Finally, we note that we have performed similar experiments with a commercial dynamics simulator (SD/FAST) where we compared simulations using Kane’s formulation and using SD/FAST’s \( O(n) \) formulation which is similar to the ABM. While the specifics of the algorithms used in this code are not known to us, and the situation is complicated by symbolic simplification, we have observed the same difference in formulation stiffness between the two methods.

In summary, these results have illustrated that the performance of a simulation algorithm does not depend only on the individual performances of the forward dynamics algorithm and the integration algorithm, but on the combined performance of these algorithms. Hence, the true measure of performance is given by the following expression
for the total number of operations ($n_{total}$):

$$n_{total}(t_s, \varepsilon) = n_{single} \times n_{FD}(t_s, \varepsilon)$$

where:

- $n_{FD}$ is the total number of forward dynamics function evaluations required to advance the solution by $t_s$ seconds with a step accuracy of $\varepsilon$,
- $n_{single}$ is the total number of operations required for a single forward dynamics function evaluation.

5.3 More simulation results

5.3.1 "Bad" implementations of the ABM

As was discussed in the previous chapter, it is possible to implement the ABM algorithm so that its performance degrades in the presence of ill-conditioning. Figure 5.7 compares the well implemented ABM with an implementation of the ABM that computes the articulated body inertia using

$$\dot{\hat{I}}^A_i = (\dot{\hat{M}}_i + \hat{X}_{i+1,i}^T \dot{\hat{M}}_i \hat{X}_{i+1,i}) - \hat{X}_{i+1,i}^T \frac{T \hat{M}_{i+1} \hat{s}_{i+1,i+1} T \hat{M}_{i+1} \hat{X}_{i+1,i}}{\hat{s}_{i+1,i+1} T \hat{M}_{i+1} \hat{s}_{i+1,i+1}}$$

Accelerations were computed under conditions similar to those of figure 5.5. This result serves to confirm the observations of the previous chapter.

5.3.2 Conservation of energy

In this section, using a simple example, we show that both the CRBM and ABM based implementations respect the conservation of energy principle. Figure 5.8 shows the evolution of the total energy for the simulation of a well-conditioned four link chain. This
Figure 5.7: Joint accelerations for “good” and “bad” implementations of the ABM (the smooth curve is the “good” implementation of the ABM).
Chapter 5. Implementation and Experimental Verification

Figure 5.8: Evolution of the total energy.

The figure shows that the total energy is conserved within a 5 decimal margin for the duration of the simulation for each forward dynamics method (this error is in keeping with the desired step accuracy). However, as the simulation advances, it seems to be more difficult to conserve energy. This behaviour is typical of any finite precision numerical simulation and is due to the propagation of the truncation error from step to step.

For all simulations presented in this chapter energy was conserved in a similar fashion, regardless of ill-conditioning. The condition of the JSLM did not significantly affect the correctness of the results. Only the computational efficiency was degraded.
5.3.3 Simulation of long chains

This section presents results and observations related to the simulation of long chains. The competing forward dynamics methods (ABM and CRBM) are compared for regular chains varying in length from 2 to 14 links (each individual link is of equal length and mass). Observations are made regarding the execution time, the number of forward dynamics evaluations and the conservation of energy.

Simulations were executed for 60 seconds of motion, with an integration step accuracy of $10^{-4}$, for unactuated chains swinging from a horizontal position (initial joint positions and velocities are zero) in a constant gravity field ($9.8\text{m/s}^2$). For each chain, a verification of the evolution of the joint angles in time confirmed that each forward dynamics method did indeed obtain the same joint angle trajectories to within the desired accuracy.

Figure 5.9 presents the results obtained in each simulation, where:

- $n$ is the number of links,
- $E_{\text{init}}$ is the total energy (potential and kinetic) of the robot at the initial configuration,
- $t_{\text{exec}}$ is the total execution time (timing results were obtained on a SUN 4/75 workstation),
- $n_{FD}$ is the number of forward dynamics function evaluations,
- $E_{\text{dev}}$ is the deviation (in percentage) of the total energy from $E_{\text{init}}$ calculated using the average total energy $E_{\text{avg}}$ as follows: $E_{\text{dev}} = 100 \times (E_{\text{avg}} - E_{\text{init}})/E_{\text{init}}$.

Figure 5.10 summarizes the results by showing the evolution of the number of forward dynamics function evaluations and the evolution of the total execution time as a function of the number of links.
Chapter 5. Implementation and Experimental Verification

54

<table>
<thead>
<tr>
<th></th>
<th>ABM</th>
<th>CRBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>(E_{\text{init}}(J))</td>
<td>(t_{\text{exec}}(s))</td>
</tr>
<tr>
<td>2</td>
<td>3920</td>
<td>0.68</td>
</tr>
<tr>
<td>4</td>
<td>15680</td>
<td>2.18</td>
</tr>
<tr>
<td>6</td>
<td>35280</td>
<td>4.85</td>
</tr>
<tr>
<td>8</td>
<td>62720</td>
<td>8.75</td>
</tr>
<tr>
<td>10</td>
<td>98000</td>
<td>11.95</td>
</tr>
<tr>
<td>12</td>
<td>141120</td>
<td>15.18</td>
</tr>
<tr>
<td>14</td>
<td>192080</td>
<td>20.78</td>
</tr>
</tbody>
</table>

Figure 5.9: Summary of results for the simulation of long regular chains.

Observations

As the measure of energy deviation indicates, the simulations seem to conserve energy quite well regardless of the number of links.

For these problems, the implementation of the ABM seems to require less execution time than the CRBM even for small values of \(n\). This would indicate the need for further optimization of the CRBM code. However, it is also possible that the ABM is better suited to this particular type of planar problem than the CRBM. Further study is indicated.

As expected, the total execution time also increases with the number of links. Furthermore, in almost all the test cases, real-time performance is obtained. The number of calls to the forward dynamics function seems to increase with the number of links in a logarithmic fashion (see figure 5.10). This would indicate that as the number of links increases, the set of differential equations becomes more difficult to solve to the desired accuracy (smaller steps must be taken). This measure suggests that the operational stiffness of the differential equations increases with the number of links. This stiffness could simply be attributed to the increasing complexity of the resulting physical motion and should not necessarily be attributed to the forward dynamics formulation.
Figure 5.10: Simulation of long chains.
In summary, the increase in running time stems from two sources of additional work:

1. as \( n \) increases, each call to the forward dynamics function is longer to evaluate (complexity: \( O(n) \) for ABM and \( O(n^3) \) for CRBM),

2. as \( n \) increases, more calls to the forward dynamics function are required (complexity: \( O(f(n)) \), where \( f(n) \) appears to be logarithmic).

This second source of work explains the fact that the running time for the ABM simulation does not seem to progress linearly with the number of links. Based on these observations, the effective running time for the ABM simulation can be expressed as \( O(n \times f(n)) \) and \( O(n^3 \times f(n)) \) for the CRBM simulation.
Chapter 6

Extensions: Contact Simulation

One of the basic functions of a robot manipulator is to act on and modify its environment. In general, to accomplish these goals, the robot must come in contact with its environment. When the end-effector of a robot comes in contact with its environment, constraints are imposed on the robot's motion. The simulation of interactions between the robot and its environment is a natural extension to a robot simulator and is currently one of the most challenging problems of rigid body dynamic simulation. This chapter presents an overview of the principal problems involved in contact simulation.

The contact simulation problem can be divided into the following subproblems: contact detection and contact resolution. Contact resolution can be further divided into impact simulation and continuous contact simulation. In the following sections each of these issues will be considered individually.

6.1 Notation

This chapter will first consider the general problem of resolving the interactions between two single rigid bodies interacting at a finite number of contact points. Extending the proposed solutions to a robot manipulator interacting with a rigid body is straightforward and will also be considered. However, extending solutions to a non-finite number of contact points is an open problem.

The following notation is introduced to deal with contact simulation between two rigid bodies $B_1$ and $B_2$ colliding at a single contact point $P$ (see figure 6.1).
Chapter 6. Extensions: Contact Simulation

58

Figure 6.1: Contact simulation notation.

- \( \hat{n} \) is the unit contact normal vector directed from \( B_2 \) to \( B_1 \), and let the right subscript \( N \) denote the normal component of a vector with respect to \( \hat{n} \) (i.e. \( v_N = \hat{n}^T \hat{v} \)) and \( T \) denote the tangential components of a vector with respect to \( \hat{n} \).

- \( \hat{M}_1 \) and \( \hat{M}_2 \) are the spatial inertias of \( B_1 \) and \( B_2 \).

- \( \check{\hat{v}}_1^- \) and \( \check{\hat{v}}_2^- \) are the known spatial velocities of \( B_1 \) and \( B_2 \) before the impact, and \( \check{\hat{v}}_1^+ \) and \( \check{\hat{v}}_2^+ \) are the unknown spatial velocities of \( B_1 \) and \( B_2 \) after the impact.

- \( \hat{v}^- = \check{\hat{v}}_2^- - \check{\hat{v}}_1^- \) is the relative velocity of the contact point before impact, and \( \hat{v}^+ = \check{\hat{v}}_2^+ - \check{\hat{v}}_1^+ \) is the relative velocity of the contact point after impact.

- \( \Delta \hat{v}_i = \check{\hat{v}}_i^+ - \check{\hat{v}}_i^- \) is the absolute velocity change of body \( B_i \) due to the impact, and \( \Delta \hat{v} = \hat{v}^+ - \hat{v}^- \) is the change in relative velocity due to the impact.
6.2 Contact detection

From a geometrical perspective the contact detection problem can be defined as the problem of determining all points of intersection between a set of objects at a time $t$ given the current position and orientation of the objects. In the context of contact simulation, the specific time at which two objects collide $t_c$ is also of interest.

For a relatively complex environment, three-dimensional collision detection can be a very difficult and time consuming process. Resolution of this problem is regarded by many as the performance bottleneck of the contact simulation process (see simulation results given by Hahn [29]). Conventional methods result in $O(n^2)$ complexity, where $n$ is the number of bodies [43]. However, more efficient algorithms based on efficient distance computation methods have recently been proposed [27, 37].

Cremer [16] uses a simple iterative approach in order to determine the time of collision. By assuming that the integration time step is small enough so that no collisions are missed, the time of collision is obtained by performing a binary search of the time step. The iterative search can be improved by making use of the current acceleration of the contact point to better divide the search space [7]. However, assuming that the integration time step is small enough so that no collisions are missed may impose the choice of an unnecessarily small integration time step. More precise methods perform a search for intersections between the four-dimensional space-time swept volumes of the objects in order to determine the exact time and point of collision [26, 13, 42].

For the contact simulation process, it is common to assume that all objects can be modelled as convex polyhedra (non-convex objects can be decomposed into unions of convex polyhedra). Reducing the number of objects tested to a minimum is perhaps the most basic way of reducing the cost of collision detection. A spatial segmentation based on bounding volumes can be used to determine objects that are near to another.
Also, knowledge of velocity and acceleration bounds can be used to reduce the number of collision tests. Using such bounds, it is possible to construct swept bounding volumes which can be tested for intersection in a preprocessing phase [42].

Once contact points have been determined, they must be organized into an efficient data structure for the simulation process. Mirtich and Canny [42] propose an interesting approach based on constructing a "collision heap" which gives a prioritization of the contact events in the simulation. This prioritization of contact events can facilitate the choice of appropriate integration time steps.

### 6.3 Friction modelling

When two bodies are in contact, friction creates a force that opposes relative motion. According to Coulomb's law, the friction force is proportional to the normal force acting at the contact point. The Coulomb friction model is expressed as follows:

\[
\bar{v}_T \neq 0 \Rightarrow \bar{f}_T = -\mu_d|f_N|\frac{\bar{v}_T}{\|\bar{v}_T\|}
\]

\[
\bar{v}_T = 0 \Rightarrow \|\bar{f}_T\| \leq \mu_s|f_N|.
\]

The first condition is referred to as dynamic friction and the second is referred to as static friction. \(\bar{v}_T\) is the tangential velocity of the point of contact, \(\bar{f}_T\) is the friction force, \(f_N\) is the component of the total force along the normal, \(\mu_s\) is the static friction coefficient and \(\mu_d\) is the dynamic friction coefficient. The coulomb friction model defines an exact friction force only in the dynamic case. In the case of static friction, all that is known is an upper bound on the friction force (cf. friction cone). In the following, we will assume \(\mu_s = \mu_d = \mu\).

Although the Coulomb friction model is approximate, it is the most widely used. As will be discussed in the following sections, modelling friction in contact simulation is a very difficult task.
6.4 Impact simulation

Impact simulation is defined as the simulation of the motion interactions of two colliding bodies for which the point of contact has a non-zero relative normal velocity $\Delta v_N$. This section reviews some of the popular methods for impact simulation.

6.4.1 Penalty methods

Penalty methods\(^1\) are approximate methods which handle collisions by applying a force at the contact point which penalizes interpenetration. In general, the penalty force is obtained using a spring model. A spring is inserted in the normal direction between the colliding bodies at the point of contact. If $d$ is the interpenetration depth then the resulting penalty force is $f = kd$ (linear spring model). To be effective, penalty methods must necessarily permit a certain degree of interpenetration.\(^2\) The spring constant $k$ is chosen \textit{ad hoc} so as to be large enough to prevent too much interpenetration. However, the larger the spring constant, the stiffer the resulting differential equations of motion and the smaller the required integration step size.

Due to the approximate nature of the numerical simulation process, a spring contact model cannot guarantee the conservation of energy. In order to obtain a more stable contact model and in order to simulate frictional effects during impact, it is common to add a damping component to the spring model. In [1], Anderson uses such a spring-damper system to simulate force feedback in a telerobotics application.

The elasticity of an impact can be specified by using two different spring constants in order to simulate the loss of energy during a collision. One for the compression phase ($\Delta v_N \leq 0$) and the other for the relaxation phase ($\Delta v_N > 0$), e.g. $k_{relax} = e k_{compress}$

---

\(^1\)This approach is based on penalty function methods used in nonlinear optimization.

\(^2\)Certain methods minimize interpenetration by extending the boundaries of the bodies so as to begin applying a penalty force when two bodies come within a distance $\epsilon$ of each other. However, this does not necessarily result in a more realistic simulation.
where $0 < e < 1$ is the coefficient of "restitution" [43]. Another variation of the spring model consists of using a non-linear spring constant [21].

Furthermore, by adding tangential spring-damping components, one can also simulate the effects of frictional forces. An in depth analysis of penalty methods can be found in Baraff [7].

### 6.4.2 Analytical methods

Analytical methods generally assume that the impact between two rigid bodies occurs instantaneously. The collision results in a discontinuity in the velocity of the bodies. When an impact occurs, the motion integration routine must be stopped and reset with new post-impact initial velocity conditions. This discontinuity in velocity is a result of an impulsive constraint force that acts normal to the contact surface

\[
\Delta v_{1N} = \frac{P_N}{(\hat{n}^T \hat{l}_1 \hat{n})}
\]

\[
\Delta v_{2N} = -\frac{P_N}{(\hat{n}^T \hat{l}_2 \hat{n})},
\]

The impulsive constraint force is computed using the coefficient of restitution. The coefficient of restitution (or elasticity) is a function of the materials of the two bodies in contact. There are two principal interpretations of this coefficient: Newton’s and Poisson’s. This section, presents rigid body simulation systems based on each of these interpretations. Good references on the topic of impact modelling include [33, 10, 59].

#### Analytical methods based on Newton’s law of restitution

Newton’s law of restitution states that the coefficient of restitution relates the relative normal velocity after impact to the relative normal velocity before impact:

\[
v^+_N = -ev^-_N.
\]
Using this relationship and equations 6.3 and 6.4 relating velocity variations to applied impulses, a linear set of equations is obtained. For the frictionless case, this set of equations can be directly resolved to obtain the change in velocity [16, 29].

In [29] the equations include a Coulomb friction model for tangential impulses. Tangential impulses are related to normal or constraint impulses in a fashion similar to equations 6.1 and 6.2. The solution algorithm first assumes that the bodies stick, and if the impulse is larger than the static friction bound then it is assumed that the bodies are sliding and the frictional impulses are determined. This method assumes the relative tangential velocity does not change during a collision (the problem of reversed sliding [59] is not considered). This method is similar to that used by Moore and Wilhelms [43].

Baraff [7] uses Newton's interpretation of the coefficient of restitution to obtain the following constraints:

\[(v_N^+ + \varepsilon v_N^-) \geq 0,\] \hspace{1cm} (6.6)

\[p_N \geq 0,\] \hspace{1cm} (6.7)

\[p_N (v_N^+ + \varepsilon v_N^-) = 0.\] \hspace{1cm} (6.8)

These constraints can be used to obtain a LCP formulation similar to that used for continuous contact simulation and can be solved by quadratic programming techniques.

**Analytical methods based on Poisson's hypothesis of restitution**

Poisson's hypothesis of restitution distinguishes between a compression and a relaxation phase of impact. The hypothesis states that the coefficient of restitution relates the total normal impulse to the normal impulse at maximum compression

\[p_N = (1 + \varepsilon)p_{Nc},\] \hspace{1cm} (6.9)

where \(p_{Nc}\) is the total normal impulse up to the point of maximum compression and \(p_N\) is the total normal impulse due to the whole impact.
As with the previous analytical methods, the velocity variations are related to the total impulse by equations 6.3 and 6.4. In the frictionless case, Poisson’s hypothesis can be used to obtain a closed form expression for the total normal impulse [20]. In this case, the resulting equations are the same as for Newton’s interpretation of restitution.

Mirtich and Canny use a detailed impact model based on Poisson's hypothesis. This model incorporates a complete friction model that deals with the transitions between sticking and sliding during impact. The method consists of numerically integrating the velocity during impact using the cumulative normal impulse as the integration parameter (instead of time). When the velocity reaches zero, the maximum compression has been reached and the total impulse at compression is obtained. Then, by multiplying by \(1+e\) the total impulse at the end of the impact is obtained. This represents the value for the “time” of end of collision, integration continues to this point and the variation in velocity is obtained. This approach is based in part on the work of Keller [33].

**Incorporating friction**

Dealing with friction in the impact model is a very difficult problem. Wang and Mason present a treatment of impact with friction for planar collisions involving single contact points. One of their main findings is that Poisson’s hypothesis is better than Newton’s law of restitution for modelling impact with friction (they show that Newton’s law of restitution does not always conserve energy). This result is based on the fact that Poisson’s hypothesis is expressed as a dynamic law and is consistent with energy conservation. Brach [10] states that the fundamental problem in modelling impacts with friction is the interpretation of the coefficient of restitution and friction. Brach uses Newton’s law of restitution and resolves the increase in energy by using a slightly different interpretation of the coefficient of friction and restitution. Baraff discusses an approximate method for simulating impact with friction [7].
Simultaneous impacts

It is very difficult to model multiple-contact point collisions. In general, most models assume that no two collisions occur at the same time and treat simultaneous collisions serially. Cremer [16] presents some of the advantages and disadvantages of using a simultaneous model of collisions. In general, the problem of resolving simultaneous collisions with friction is currently unsolved.

Multibody systems

Extending impact simulation to include collisions with multibody systems, generally involves propagating the velocity discontinuities through each joint of the multibody system. These issues are discussed in [6] and [43].

Summary

Even though resolving impacts using analytical methods may seem like a lot of work, it is often more efficient than a spring model since it only has to be computed once for each impact [43] (spring models involve several evaluations over many small steps of integration to give realistic results).

6.4.3 Finite element methods

Finite element simulation methods (FEM) consist of simulating contacts at a very detailed (elementary) level. These methods can be used to study the stresses and strains occurring during the compression and restitution phase of a collision, however such methods are too computationally expensive for real-time simulation. Furthermore, it is generally believed that such detail is unnecessary and that simple empirical rules can sufficiently describe the collision behaviour [42].
6.5 Simulation of continuous contact

Baraff defines the continuous contact simulation problem as follows [8]: "Given a system of rigid bodies, contacting at n points and nowhere colliding, with known spatial and velocity variables, calculate the contact forces that should arise between the bodies to prevent interpenetration (...) the contact forces must also obey constraints due to friction." In this section, several approaches to continuous contact simulation will be reviewed.

6.5.1 Penalty methods

Penalty methods similar to those used for impact modeling can be used in continuous contact simulation. The main advantage of using such models is that they are very simple to implement. The use of such a model during impact and continuous contact may also ensure smoother transitions between the impact and continuous contact motions. For certain applications, these types of models may generate results which give an acceptable level of realism. However, many applications demand a greater level of realism; hence the interest in analytical methods.

6.5.2 Analytical methods

Analytical methods or constraint-based methods impose kinematic constraints on the configuration of contacting bodies such that interpenetration is strictly prohibited. Forces at a contact point are separated into normal forces and tangential forces. Normal or constraint forces are workless forces which prevent interpenetration, while tangential forces or friction forces oppose motion along the contact surface.

For two bodies in continuous contact with no friction, the following local motion
constraints can be written for each contact point:

\[ d(v_N)/dt \geq 0, \quad (6.10) \]
\[ f_N \geq 0, \quad (6.11) \]
\[ f_N d(v_N)/dt = 0. \quad (6.12) \]

These constraints define a convex linear complementary problem (LCP) which may be solved by quadratic programming (QP) techniques. Lotstedt [38] was one of the first to formulate the contact simulation problem in such a manner. Recently, Baraff [7] has used such a formulation to simulate problems with and without friction. Baraff studied the inconsistencies related to frictional contact simulation [47, 59] and showed that determining whether a configuration is consistent is NP-hard.

Recently, Lee et al. [35] showed how Baraff's contact simulation methods can be extended to the context of a robot simulation system. An LCP formulation is also used by Featherstone [20] to solve the problem of constrained robot dynamic simulation. Featherstone combines the dynamic equations of the robot manipulator with \( n \) simultaneous quadratic equations in the unknown contact forces.

For an equality constrained problem, constraint forces can be computed by solving a linear system of equations. Such a formulation is used by Lilly [36] to simulate the motion of a robot whose end-effector is constrained by continuous contact. Lilly uses an operational space formulation of the robot's inertia matrix to obtain an expression for contact constraints which results in a linear system of algebraic equations. This approach is similar to the contact simulation methods used by Witkin et al. [61] (this work also considers multiple contact points).

For a robot manipulator constrained by a contact with the environment, the constraining forces can be much larger than the joint forces resulting in a system of stiff
ordinary differential equations. Therefore, equality constrained contact problems are very well suited to DAE formulations. DAE formulations of continuous contact problems can be found in [40, 21].

Baiardi et al. [6] present the differential algebraic equations of motion for a kinematic chain in contact with a hard motionless surface. Three formulations are obtained for constraints under the following conditions: friction free, with static friction and with dynamic friction. The transitions between each of these conditions are also explored.

One of the drawbacks with analytical approaches is in dealing with changing contact profiles. Constraints must be monitored in order to keep track of the changing geometry of the contact surface. Another disadvantage is that impacts must be dealt with separately.

6.5.3 Impulse based methods

Mirtich and Canny [42] propose an interesting approach to contact simulation called impulse-based simulation which resolves all types of interactions as impacts. Under this approach, continuous contact is modelled as a series of numerous impacts called micro-collisions. This approach is both conceptually simple and computationally efficient.

The underlying assumption is that collision forces in nature are based on local properties like contact velocity, and not on the global state configuration of bodies in contact. Mirtich and Canny use an impulse simulation algorithm based on [33].

One of the advantages of this approach to contact simulation is that it unifies all types of contacts (bouncing, rolling, sliding, etc..) under a single model. No artificial boundary is created between the different modes of motion. It is also claimed that this approach results in physically accurate behaviour.
In this thesis, we have explored several key issues involved in dynamic simulation of robot manipulators. We have showed how a good notation can play an important role in the formulation and understanding of the equations of motion of robot dynamics. We have identified an important phenomenon which we call “formulation stiffness”. Different formulations of the forward dynamics problem were shown to deal differently with ill-conditioned joint-space inertias. In particular, we have shown that the articulated-body method can be less sensitive to ill-conditioning, while the composite rigid body method produces errors that are proportional to the condition number of the joint space inertia matrix. This is important because the errors in the CRBM solution can incapacitate explicit adaptive-step size integration methods by requiring very small integration step sizes. In this light, we demonstrated with experimental results that the articulated-body method is superior to the composite rigid body method for ill-conditioned problems. We thus highlight the importance of measuring the performance of simulation algorithms by taking into account both the computational complexity of the forward dynamics and its numerical properties. Awareness of such issues is an essential step in properly identifying the performance and limitations of computer simulation.

Of particular interest to the author is the possible contribution of robot dynamic simulation to the development of advanced telerobotics operator interfaces. Recently, many have showed how graphical simulations can be used to improve operator interfaces to telerobotics control systems.
Graphical robot simulations can be used to create what are known as predictive displays [53, 9]. Predictive displays are used to bridge the time delay gap in situations where the remote robot is situated far from the operator (e.g. space telerobotics can involve time delays of the order of 10 seconds). Experimental results indicate that the use of predictive displays improves the performance of the human operator [53, 9, 14, 31] (in comparison with simple “do and wait” methods). A graphical simulation of the robot and its environment may also serve to complement available camera views of the worksite [12]. Recently, Funda and Paul have shown how simulation can be used to create a preview display for teleprogramming [24, 45]. In addition, a robot simulation interface can include elements such as virtual tools and fixtures which can further aid robot operators [51].

Most of the current systems used to enhance telerobotics interfaces incorporate only a kinematic simulation of the robot manipulator. Therefore, it is of particular interest to see how dynamic simulation, including simulation of contacts with the environment, could be used to aid telerobotics.
Bibliography


Bibliography


