AUTOMATED GENERATION OF ROBOTIC DYNAMICS SIMULATORS THROUGH SYMBOLIC COMPUTING

by

JUN-TIEN TWU

B. S., National Taiwan University, 1985

A MASTERS THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

Approved by:

K Bachkar

Major Professor

LD
2160
.T4
ME
1987
T88
C.2

TABLE OF CONTENTS

A11207 310504

CHAPTER 1

INTRODUCTION

In this era of the digital computer, scientists and technologists are finding new ways to use the computer as a problem-solving tool in their areas. This is especially true of the engineering sciences. Engineering problems involve enormous, somtimes tedious, calculations; therefore, computers are exceptionally well-suited for applications in this field.

The engineering design process is often one of trial and error; however, computer simulation has done much to reduce the human effort involved and to improve efficiency. The simulation of a design is carried out with computer software -- the so called simulator. Different simulators are available with their own objectives. In general, a simulator is a machine or program to which one feeds the description of the system and obtains the response characteristics of the system. The specific simulator covered in this research is one for simulation of robotic dynamics, both forward and inverse. The input to the simulator will be the robot parameters such as the masses, moments of inertia, and dimensions of each link, joint types and locations, etc. For the forward dynamic

simulation, it also requires joint torques or forces, initial positions and velocities of each link to find the link positions, velocities, and accelerations at various instants of time. In the inverse dynamic simulation, link positions, velocities, and accelerations have to be specified; as the simulation proceeds, it will find the required joint torques or forces at each time interval.

Due to difference among manipulator designs, simulators are often written for each specific design. When a designer wants to change the basic layout of his design, the simulator also needs to be changed. Even minor modifications in the robot system could result in significant modification of the simulator program. This is felt most strongly when the simulator is being used in the design of a robot, since it is likely that the robot will undergo considerable modification during the design process. This lack of versatility is a serious handicap that provides the motivation for the development of more flexible simulators. A good way to do this job is to write an automatic simulator generator, as described in this thesis, which can handle a wide variety of robot manipulator systems. This is best done through symbolic computation using an algebraic manipulation language, such as MACSYMA or REDUCE-3.

There are several methods available for setting the dynamic equations, for example, Lagrange [1,2,3], virtual

work [4], and Newton-Euler [5,6,7]. For the purpose of control, it is desirable to obtain the solution in real-time. To solve the dynamic equations efficiently, several ideas have been presented in the literature. For example, the recursive Lagrange formulation [8] was used to solve the inverse dynamic problem, in which the generalized forces were expressed in recurrence relations involving the generalized coordinates, velocities, and accelerations. By the recurrence of these quantities, it is possible to reduce the number of multiplications and additions involved from quartic to linear dependence on the number of links of the system. There is also a recursive Newton-Euler formulation [5,6,9], which uses a similar idea. Another approach is the tabular solution look-up [10], which reduces computation but increases the input/output (I/O) requirements. The inverse dynamics problem for specific motions of a specific robot is solved once and all the solutions are stored, then one can find the solution of generalized forces at a certain instant by interpolation. For specific robots, it is also possible to simplify problems by dropping Coriolis and centrifugal terms [1,11]. This simplification can be made only after verifying that these terms have insignificant contributions and that certain presumptions apply.

Among these methods, the Lagrangian formulation was chosen as being the most suitable for computer implementation. One coordinate is assigned to each link, thus we are dealing with the minimum set of generalized coordinates. Translational joints and revolute joints are modeled in a unified way; this streamlines the program implementation. The same Lagrange equation formulation is able to represent both forward and inverse dynamic problems; some common terms have the same expressions in both cases which in turn further simplifies the implementation. The forces resulting from springs and dampers, frictional forces, and other externally applied forces are taken into account in the generalized force term.

As stated previously, the forward and inverse dynamic problems are to be solved. For the inverse one, the situation becomes straight forward since all the quantities on the left hand side of the Lagrangian equation are known. The unknown generalized force is found by plugging in suitable values into the Lagrangian equation. For the forward one, we are given the initial positions, velocities and generalized forces. By certain techniques, which will be described later, we can extract the unknown acceleration terms and solve for them. The accelerations obtained are then integrated to find new values of positions and velocities. This step-wise procedure is continued until

the termination time is reached.

Related work in this area has been done in the derivation of Lagrange's equations of motion using MACSYMA [12]. The objective of the work was to find the dynamic equations for any general robot and expand them into symbolic form. The method used to describe the system kinematics was the Denavit-Hartenberg convention [13] with homogeneous coordinate transformations. The dynamic equations are set up following Paul's derivation [14] which is basically a series of matrix and vector multiplications. The matrix form of the dynamic equations was then expanded symbolically using simple commands for matrix multiplication provided by MACSYMA. To simplify the expanded expressions, it was suggested that one use RATSIMP for algebraic simplification, and TRIGREDUCE for trigonometric simplification; one can also define one's own simplifying functions by using the RATSUBST feature of MACSYMA. The benefits of this approach are elimination of the time-consuming and error-prone manual derivation process and the generation of equations which are both insightful and more computationally efficient. The force components are derived individually as inertial, Coriolis, and gravitational force terms. However, only the force terms are derived as explicit functions of other generalized quantities, [12] solves only the inverse

dynamic problem.

The research in this thesis develops a generalized technique for generating simulators for serial open chain robots using the symbolic algebraic manipulation language REDUCE-3. Though the generality for different robots is obtained through symbolic computing, it is also known that the language is very slow in arithmetic. On the other hand, FORTRAN is relatively strong in numeric computing but ill-suited for symbolic computing, the combined use of both languages is used in the implementation. Because of the generality and flexibility, no assumptions, modifications, or simplifications are imposed on the original Lagrangian equation, and hence the solutions obtained from the simulator are exact. Symbolic languages tend to expand all expressions and considerable simplification is often required before these expressions can be evaluated; this is one of the difficulties encountered in [12]. In general, we can not guarantee that all the possible simplifying functions are defined in the simulator generator; further, the simplification process often takes too much memory space in computers. For certain very complicated problems, it might just fail for lack of memory. To avoid excessively lengthy expressions, the individual terms in the Lagrangian equation are treated separately. By expressing them as combinations of other basic quantities, we can derive only these basic expressions symbolically and

set up FORTRAN loops to combine them. Once the lengthy expressions are avoided, the generated FORTRAN simulators are ready for compilation and execution, and no further simplification is necessary. The sensitivity equations are also derived so that the simulators may later be extended to include design optimization capabilities.

The derivation of Lagrange's equations for the forward and inverse dynamic problems are presented in Chapter 2 of this thesis. The design sensitivity equations, which are derived from the equations of motion, are developed in Chapter 3; the potential use of these equations for optimizing the design of new robots and the utilization of existing ones is also discussed. Chapter 4 describes the implementation of the simulator generator with emphasis on the interaction between the symbolic and numerical parts of the code generation/execution. In order to test the efficiency of the simulator generator, several examples were run for both the forward and inverse problems. some of these examples, along with the verification tests that were made to establish the correctness of the results, are presented in Chapter 5. Finally, the conclusions drawn from the present work and some recommendations for future work in this area are discussed in Chapter 6.

CHAPTER 2

SYSTEM EQUATIONS

To analyze the response of a dynamic system, it is necessary to formulate the equations of motion for the system of interest. The equations of motion for a mechanical system can be obtained either from Newtonian mechanics [15] or from Lagrangian mechanics [12,16,17]. The Lagrangian formulation is more popular because it gives a straight forward way of treating constrained dynamic systems; and most mechanical systems can be modelled in terms of rigid bodies and constraints.

In this research, robots with a serial combination of rigid links connected by joints are of interest; such a robot would be completely described if the links and joints are completely specified. In order to specify the relative positions of the links, each link is associated with a cartesian coordinate that is fixed to it; the homogeneous transformation matrices between successive coordinates can be found. These matrix transformations are of the form

$$P_{i-1}=A^{i}_{i-1}P_{i} \qquad (2\text{-}1)$$

where

A^{i}_{i-1} is a 4x4 transformation matrix,

P_{i} is the homogenious representation of a vector, p, in

the i^{th} coordinate system.

The function of a joint between two links is to provide relative motion and is thus associated with the degree-of-freedom of the system. Due to mechanical simplicity, the construction of most practical robots is designed in such a way that there are only single degree-of-freedom joints between the links. Two examples of single degree-of-freedom links are the revolute joint and the translational joint. The relative motion associated with revolute joints is an angular rotation while that for translational joints is a linear displacement. By introducing suitable generalized coordinates in the Lagrangian formulation, we can treat both joint types in a unified way. We can construct a system generalized coordinate vector, q, consisting of the joint coordinates; also, by differentiating it with respect to time, we can define the generalized velocity vector, $\dot{q}$, and the generalized acceleration vector, $\ddot{q}$.

To describe the links fully from the point of view of dynamics, it is necessary to specify the position of the center of mass of each link; and it is convenient to give this specification in the corresponding local coordinate system since this specification is invariant with respect to motion of the system. We also need the mass of each body and the moment of inertia of each body about the local coordinate axis.

Once the above information about the system is provided, it is possible to express the Lagrangian of the system in terms of these parameters. We are presently interested in both the forward and inverse dynamics; therefore, both sets of system equations are required for the purpose of analysis. In deriving the system equations, we must also ensure that they can be conveniently implemented on the computer.

Lagrange's equation of motion can be written as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = Q_j, \qquad j=1,2,\ldots,n \tag{2-2}$$

where

q_j is the generalized coordinate,

$\dot{q}_j$ is the generalized velocity,

Q_j is the generalized force,

n is the degree-of-freedom of the system,

and L is the system Lagrangian defined by

$L=T-V$

where

T is the system kinetic energy,

and V is the system potential energy.

To construct the Lagrangian, the system kinetic and potential energies should be expressed in terms of the generalized coordinates, velocities, and link parameters.

We will first consider the potential energy. Since spring forces are included in the generalized force term in the equation, the only contribution to the potential energy is the gravitational potential energy. Assume that the base coordinate is established in a way that its y-axis is along the vertical; therefore, the gravity acts in the negative y direction and the potential energy of any link is given by

$$V_i = m_i g Y_i \tag{2-3}$$

where

Y_i is the y-coordinate of the center of mass of link, i, as measured from the base coordinate,

m_i is the mass of link, i,

and g is the gravitational acceleration.

Since the coordinates of the center of mass are given with respect to the local coordinate, and the transformation matrix transforms between two successive frames, we can then use a series of transformation matrices to find the coordinate as viewed from the base, given by

$$[X_i, Y_i, Z_i]^T = A_1 A_2 A_3 \ldots A_i [x_i, y_i, z_i]^T \tag{2-4}$$

where the terms within brackets on the left and right hand side are the homogeneous coordinates of the center of mass in the global and local coordinate systems respectively. The A's are 4x4 joint transformation matrices; their products can be written in a simpler way, by defining

$$T_i = A_1 A_2 A_3 \ldots A_i \tag{2-5}$$

T_i is then the transformation matrix from i^{th} local coordinate to the base. The system potential energy would be

$$V = \sum_i V_i = \sum_i m_i g Y_i \tag{2-6}$$

The motion of any link can have both translational and rotational components; therefore, the kinetic energy term is comprised of their respective contributions. It can be written as

$$T = \frac{1}{2} \sum_i [m_i(v_{ix}^2 + v_{iy}^2 + v_{iz}^2) + (I_{ix}W_{ix}^2 + I_{iy}W_{iy}^2 + I_{iz}W_{iz}^2)] \tag{2-7}$$

where

v and W are translational and rotational velocities respectively,

subscripts x, y, and z denote the component along each axis,

i is the link number,

and the I's are the centroidal moments of inertia.

The translational velocity can be found by differentiating Equation 2-4 with respect to time; noting that T_i is an abbreviation for $A_1 A_2 \ldots A_i$, we have

$$[V_{ix}\ V_{iy}\ V_{iz}\ 1] = [\dot{T}_i][x_{ci}\ y_{ci}\ z_{ci}\ 1] \tag{2-8}$$

and from equation 4 we find

$$\dot{T}_i = \dot{A}_1 A_2 \ldots A_i + A_1 \dot{A}_2 \ldots A_i + \ldots + A_1 A_2 \ldots \dot{A}_i \tag{2-9}$$

For the rotational velocity terms, given the moments of inertia in the local coordinates, it is desirable to find the absolute angular velocities for each link expressed in the local coordinate system of that link. It is clear that the velocity of a link is not kinematically dependent on the links that follow it, but only on preceding links and itself; therefore, it involves only the transformation of angular velocities from preceding links to the local frame of interest. We compute the total angular velocity of link ,i , expressed in the i^{th} local coordinate system recursively as follows:

Suppose the total angular velocity vector of link, i-1, is known in its local $i-1^{th}$ coordinate system, it can be converted into the i^{th} coordinate system by premultiplying this vector by a transformation matrix, given by

$$w_i' = R_i^{-1}(W_{i-1}) \qquad (2\text{-}10)$$

where

R_i contains only the rotational terms of A_i,

w_i' is the contribution to angular velocity of link, i, from the preceding links,

and W_{i-1} is the total angular velocity of link, i-1, given by

$$W_{i-1} = w_{i-1} + w_{i-1}' \qquad (2\text{-}11)$$

where

w_{i-1} is the angular velocity of link , i-1, itself.

Since R_i is an orthogonal matrix, R_i^{-1} is just R_i^T which is easy to construct symbolically. Thus for any link, i, we can find the total angular velocity in the following way. Starting from the first link, we find the total angular velocity of link, 1, expressed in local coordinate of link, 2. This is then added to the angular velocity due to relative motion between link, 1 and 2, to obtain the total angular velocity of link, 2. We proceed in the same way until the total angular velocity of link, i, is found.

Note that for a translational joint, the relative angular velocity between the connected bodies is zero. Once the translational and rotational velocities are known, it is easy to explicitly derive the system Lagrangian in symbolic form by using simple loops in a REDUCE program. However, this is not advisable because the velocity terms may be quite complicated and squaring them could lead to expressions that are too large to handle. What we need presently are not the expressions for kinetic energy or the Lagrangian; rather, we are interested in the derivatives of the Lagrangian with respect to generalized coordinates and velocities. We must therefore develop formulas for these quantities which do not require the evaluation of unreasonably large expressions.

The Lagrangian equation can be expanded term by term in the following way:

$$\frac{\partial L}{\partial q_j} = \frac{\partial (T-V)}{\partial q_j}$$

$$= \sum_i m_i v_i \frac{\partial v_i}{\partial q_j} + \sum_i I_i W_i \frac{\partial W_i}{\partial q_j} - \sum_i m_i g \frac{\partial Y_i}{\partial q_j} \qquad (2\text{-}12)$$

and

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) = \frac{d}{dt}\frac{\partial (T-V)}{\partial \dot{q}_j} = \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) \qquad (2\text{-}13)$$

The potential energy term is dropped from the above equation because it is independent of velocities.

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) = \frac{1}{2}\frac{d}{dt}\left[\frac{\partial \sum_i (m_i v_i^2 + I_i W_i^2)}{\partial \dot{q}_j}\right]$$

$$= \sum_i m_i \left[\frac{dv_i}{dt}\frac{\partial v_i}{\partial \dot{q}_j} + v_i \frac{d}{dt}\left(\frac{\partial v_i}{\partial \dot{q}_j}\right)\right] + \sum_i I_i \left[\frac{dW_i}{dt}\frac{\partial W_i}{\partial \dot{q}_j} + W_i \frac{d}{dt}\left(\frac{\partial W_i}{\partial \dot{q}_j}\right)\right] \qquad (2\text{-}14)$$

Now we can find the Lagrangian equation by subtracting Equation 2-12 from Equation 2-14 and setting it equal to the generalized force. The Lagrangian equation is then in a form that can be handled by REDUCE.

For the inverse dynamic problem, we want to find the generalized forces required to satisfy prescribed requirements on the displacements, velocities, and accelerations. We see that the unknown generalized force appears on the right hand side of Equation 2-2, and that all the terms on the left are known quantities; thus the evaluation of the forces is straight forward.

For the forward dynamic problem, we wish to calculate the displacements, velocities, and accelerations given the generalized forces and the initial displacements and velocities. The unknown accelerations are solved algebraically from the equations of motion, and then integrated to obtain velocities and positions. However, the accelerations for which we must solve are mixed with other known quantities on the left side in the $\frac{dv_i}{dt}$ and $\frac{dW_i}{dt}$ terms. Again, there is a perfect similarity between translational and rotational accelerations, because of the use of the generalized coordinates. Since the accelerations are obtained from multiplication of three matrices and there is only a first power of $\ddot{q}$, it follows that the multiples are linear combination of accelerations. All the terms containing accelerations are then extracted, while the other known quantities are moved to the right hand side. Once this is done, the resulting set of linear equations can be solved for the accelerations. Finally, the solved accelerations are integrated in a given time interval to find the velocity and the displacement for the next step by using the Runge-Kutta integration scheme. Provided the generalized forces for the next step, the process is repeated until the end of the simulation.

CHAPTER 3

DESIGN SENSITIVITY ANALYSIS

The utility of a simulator in the design process is greatly enhanced if the simulator has the ability to automatically optimize the system design. It is envisioned that the methods developed in the preceding chapter for analysis will be extended to include optimization capabilities. This will enable the use of the simulator for optimizing the design of new robots as well optimizing task-planning for existing ones.

Symbolic computing is particularly useful in an optimization environment because optimization problems are generally more difficult to standardize; there can be a wide variety of choices of design parameters and optimality criteria. Thus, it is desirable to maintain the symbolic computing approach while extending our analysis method into the realm of optimization.

A method of performing this extension in a manner that will permit convenient implementation in a symbolic manipulation language will be developed in this chapter.

For the purpose of optimizing a design, the design problem should be expressed in a standard format. The standard problem is stated in the following manner.

Suppose the design of the system is specified by a vector, b, of r design variables, i.e.

$$b=[b_1,b_2,\ldots,b_r]^T \tag{3-1}$$

Find thedesign,b, which minimizes a specified cost function, F_0, subject to constraints

$$\begin{aligned} F_i&=0, \qquad i=1,2,\ldots,m \\ F_i&\leq 0, \qquad i=m+1,m+2,\ldots,n \end{aligned} \tag{3-2}$$

The optimal design process solves this standard problem by iterative improvement of the design vector, b. Each iteration of this solution process consists of three distinct steps:

1. System analysis: The behavior of the system is analyzed at the current design. The cost and constraint function values are also computed.
2. Design sensitivity analysis: The derivatives of the cost and constraint functions with respect to the design variables are calculated in this step.
3. Optimization/design update: The cost and constraint function values from the system analysis as well as the derivatives obtained from design sensitivity are supplied to an optimizing algorithm. The algorithm computes the required change in design and proceeds to the next iteration unless a predefined convergence criterion is satisfied.

The system analysis is done by solving the system equations developed in Chapter 2. The design sensitivity

analysis is discussed in this chapter for the future purpose of optimizing and updating the design.

The system equations of motion derived in the preceding chapter are dependent on the design vector, b, of the system. The optimizing process is to choose suitable values for the components of the design vector in order to minimize a cost function, subject to the given constraints. We assume that the cost and constraint functions, F_i, depend on time (t), position (q), velocity ($\dot{q}$), acceleration ($\ddot{q}$), joint generalized force(Q), and design(b). That is, they are of the form

$$F_i=F_i(t,q,\dot{q},\ddot{q},Q,b), \qquad i=0,1,\ldots,n \tag{3-3}$$

Most optimization algorithms require the derivatives of cost and constraint functions with respect to design. The aim of first order design sensitivity analysis is to evaluate these derivatives to first order. These derivatives relate variations in cost/constraints function values to those in the design parameters through the equation:

$$\delta F = l^T \delta b \tag{3-4}$$

where l is the matrix (nxr) of design sensitivity coefficients.

In order to find the design sensitivity coefficients, we have to take the variations of the F_i's, which are of the form

$$\delta F = \frac{\partial F}{\partial b}\delta b + \frac{\partial F}{\partial q}\delta q + \frac{\partial F}{\partial \dot{q}}\delta \dot{q} + \frac{\partial F}{\partial \ddot{q}}\delta \ddot{q} + \frac{\partial F}{\partial Q}\delta Q \qquad (3\text{-}5)$$

Since state, velocity, acceleration and generalized force are all implicit functions of the design, it follows that

$$\delta q = \frac{\partial q}{\partial b}\delta b, \quad \delta \dot{q} = \frac{\partial \dot{q}}{\partial b}\delta b, \quad \delta \ddot{q} = \frac{\partial \ddot{q}}{\partial b}\delta b, \quad \delta Q = \frac{\partial Q}{\partial b}\delta b \qquad (3\text{-}6)$$

Equation 3-6 can be substituted into Equation 3-5 to obtain

$$\delta F = \frac{\partial F}{\partial b}\delta b + \frac{\partial F}{\partial q}\frac{\partial q}{\partial b}\delta b + \frac{\partial F}{\partial \dot{q}}\frac{\partial \dot{q}}{\partial b}\delta b + \frac{\partial F}{\partial \ddot{q}}\frac{\partial \ddot{q}}{\partial b}\delta b + \frac{\partial F}{\partial Q}\frac{\partial Q}{\partial b}\delta b \qquad (3\text{-}7)$$

comparing Equations 3-7 and 3-4, we conclude that

$$1^T = \frac{\partial F}{\partial b} + \frac{\partial F}{\partial q}\frac{\partial q}{\partial b} + \frac{\partial F}{\partial \dot{q}}\frac{\partial \dot{q}}{\partial b} + \frac{\partial F}{\partial \ddot{q}}\frac{\partial \ddot{q}}{\partial b} + \frac{\partial F}{\partial Q}\frac{\partial Q}{\partial b} \qquad (3\text{-}8)$$

As with the dynamics analysis, the sensitivity analysis also has both forward and inverse cases. For the inverse sensitivity analysis, the only unknown on the right hand side of Equation 3-8 is $\frac{\partial Q}{\partial b}$ which can be found by differentiating the Lagrange equation with respect to the design, b. For the forward dynamics problem, we must have initialvalues of $\frac{\partial q}{\partial b}$, $\frac{\partial \dot{q}}{\partial b}$, $\frac{\partial Q}{\partial b}$, and $\frac{\partial Q}{\partial b}$ at each step. Then, by taking the first variation of the Lagrangian equation, we can obtain a second order ordinary

differential equations in $\frac{\partial q}{\partial b}$, which can be numerically integrated along with the equations of motion. At any instant, the values of $\frac{\partial q}{\partial b}$, $\frac{\partial \dot{q}}{\partial b}$, and $\frac{\partial \ddot{q}}{\partial b}$ thus obtained can be substituted into Equation 3-8 to obtain the design sensitivity.

We thus see that the unknown terms are, $\frac{\partial Q}{\partial b}$, for the inverseanalysis, and $\frac{\partial q}{\partial b}$, $\frac{\partial \dot{q}}{\partial b}$, and $\frac{\partial \ddot{q}}{\partial b}$ for the forward analysis. It is desirable to derive these terms so that they are ready for implementation in a computer code. When differentiating a series of matrices, followed by a vector, with respect to another vector, it is possible to avoid three dimensional tensors within the process by using the mathematical technique described below.

Suppose we want to find

$$\frac{\partial (A_1 x)}{\partial b} \tag{3-9}$$

where

A_1 is a two dimensional matrix,

and b and x are vectors.

The technique is given by

$$\frac{\partial (A_1 x)}{\partial b} = \frac{\partial (A_1 \tilde{x})}{\partial b} + A_1 \frac{\partial x}{\partial b} \tag{3-10}$$

The tilde (~) in Equation 3-10 means that the term beneath it is treated as constant when doing differentiation, but after the process, it is again recognized as its original nature.

For a more complicated case, we can generalize the procedure as follows.

$$\frac{\partial}{\partial b}\,(A_1A_2A_3\ldots A_jx) \tag{3-11}$$

can be expressed into a form of a nest within another by

$$(\text{Nest})_k = D_k\,(\text{Nest})_{k+1} + \frac{\partial(D_k\,\tilde{B}_k)}{\partial b}\,,$$

$$k=j,j-1,\ldots,1 \tag{3-12}$$

where

D_k is the dummy that represents transformation matrices A's,

and B_k is the multiple from the $k+1^{\underline{th}}$ matrix to the end in $(A_1A_2A_3\ldots A_jx)$.

The iteration starts with k=j and ends with k=1 and $(\text{Nest})_1$ is the result of the derivation. To start the iteration, $(\text{Nest})_{j+1}$ must be given. Therefore, we define

$$(\text{Nest})_{j+1} = \frac{\partial x}{\partial b} \tag{3-13}$$

Equations 3-12 and 3-13 altogether is the solution of Equation 3-11. Base on this, we can find the terms $\frac{\partial Q}{\partial b}$

and $\frac{\partial \ddot{q}}{\partial b}$ that are necessary for the forward and inverse sensitivity analyses. The key to both terms, as stated previously, is to start with taking the derivative of the Lagrange equation with respect to the design variable.

$$\frac{\partial}{\partial b}\left[\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right)\right]$$

$$= \sum_j \frac{\partial m_j}{\partial b}\frac{dv_j}{dt}\frac{\partial v_j}{\partial \dot{q}_i} + \sum_j m_j \frac{\partial}{\partial b}\left(\frac{dv_j}{dt}\right)\frac{\partial v_j}{\partial \dot{q}_i} + \sum_j m_j \frac{dv_j}{dt}\frac{\partial}{\partial b}\left(\frac{\partial v_j}{\partial \dot{q}_i}\right)$$

$$+ \sum_j \frac{\partial m_j}{\partial b} g \frac{\partial Y_j}{\partial q_i} + \sum_j m_j g \frac{\partial}{\partial b}\left(\frac{\partial Y_j}{\partial q_i}\right) \qquad (3\text{-}14)$$

For the inverse sensitivity problem, Equation 3-14 is just $\frac{\partial Q}{\partial b}$, the unknown. To evaluate 3-14, some of the terms are seen to be very complicated, such as $\frac{\partial}{\partial b}\left(\frac{dv_j}{dt}\right)$ and $\frac{\partial}{\partial b}\left(\frac{\partial v_j}{\partial \dot{q}_i}\right)$, from the experience in dealing with the dynamics analysis. The other terms are either quite simple for REDUCE to handle or has been done in the dynamic problem.

Following the convention described in the second chapter, we can expand $\frac{dv_j}{dt}$ in the following manner.

$$\frac{\partial}{\partial b}\left(\frac{dv_j}{dt}\right) = \frac{\partial}{\partial b}[(\ddot{A}_1A_2\ldots A_jr+\dot{A}_1\dot{A}_2\ldots A_jr+\ldots+\dot{A}_1A_2\ldots\dot{A}_jr)$$
$$+(\dot{A}_1\dot{A}_2\ldots A_jr+A_1\ddot{A}_2\ldots A_jr+\ldots+A_1\dot{A}_2\ldots\dot{A}_jr)$$
$$\vdots$$
$$+(\dot{A}_1A_2\ldots\dot{A}_jr+A_1\dot{A}_2\ldots\dot{A}_jr+\ldots+A_1A_2\ldots\ddot{A}_jr)] \qquad (3\text{-}15)$$

Let's take every $A_1A_2\ldots A_jr$ as a group. The previous described technique can be applied to obtain the derivative for each group. Then, by summing the derivatives for all the n x n groups, we can find the solution.

We can also expand $\frac{\partial v_j}{\partial \dot{q}_i}$ similarly

$$\frac{\partial}{\partial b}\left(\frac{\partial v_j}{\partial \dot{q}_i}\right) = \frac{\partial}{\partial b}\left(\frac{\partial \dot{A}_1}{\partial \dot{q}_i}A_2\ldots A_jr\right.$$
$$+A_1\frac{\partial \dot{A}_2}{\partial \dot{q}_i}\ldots A_jr$$
$$\vdots$$
$$\left.+A_1A_2\ldots\frac{\partial \dot{A}_j}{\partial \dot{q}_i}r\right) \qquad (3\text{-}16)$$

Since $\dot{q}_i$ will exist in $\dot{A}_j$ only when i=j, some of the derivatives in Equation 3-16 are always zero and hence not shown. Actually, the only non zero term in Equation 3-16 is the i^{th} group. Again, the derivative can be evaluated in the previous way.

Once these complicated terms are obtained, along with simpler terms derived by REDUCE, we are ready to calculate $\frac{\partial Q}{\partial b}$ using Equation 3-14. It is then substituted back into Equation 3-8 to obtain $\frac{\delta F}{\delta b}$ which is the objective of the inverse sensitivity analysis.

For the forward sensitivity analysis, we also have to follow Equation 3-14. This time, we are interested in isolating the unknown $\frac{\partial \ddot{q}}{\partial b}$. From Equation 3-14, it is apparent that $\frac{\partial \ddot{q}}{\partial b}$ is mixed with others in $\frac{\partial}{\partial b}\left(\frac{dv_j}{dt}\right)$ only, and hence we are to concentrate on this term.

The term, $\frac{dv_j}{dt}$, can be expanded as in Equation 3-15. The $\ddot{q}$'s exist only in the diagonal groups. Therefore, the diagonal groups shoud be separated into two parts. One with $\ddot{q}$ and the other without, as follows.

$$[\ddot{A}_i]=[\ddot{A}_i^A]\ddot{q}_i+[\ddot{A}_i^B] \tag{3-17}$$

Accordingly, Equation 3-15 becomes the combination of two, i.e.

$$\begin{aligned}\frac{\partial}{\partial b}\ [&(\ddot{A}_1^B A_2 \ldots A_j r+\dot{A}_1\dot{A}_2 \ldots A_j r+\ldots+\dot{A}_1 A_2 \ldots \dot{A}_j r)\\ &+(\dot{A}_1\dot{A}_2 \ldots A_j r+A_1\ddot{A}_2^B \ldots A_j r+\ldots+A_1\dot{A}_2 \ldots \dot{A}_j r)\\ &\qquad\vdots\\ &+(\dot{A}_1 A_2 \ldots \dot{A}_j r+A_1\dot{A}_2 \ldots \dot{A}_j r+\ldots+A_1 A_2 \ldots \ddot{A}_j^B r)]\end{aligned} \tag{3-18}$$

and

$$\frac{\partial}{\partial b}\left[\ddot{A}_1 A_2 \ldots A_j r + A_1 \ddot{A}_2 \ldots A_j r + \cdots + A_1 A_2 \ldots \ddot{A}_j r\right] \qquad (3\text{-}19)$$

Equation 3-18 is composed of constants and is to be moved to the right hand side of the Lagrange equation. The i^{th} group in Equation 3-19 corresponds to the contribution of $\frac{\partial}{\partial b}\left(\frac{dv_j}{dt}\right)$ to the coefficient of $\ddot{q}_i$, $i=1,2,\ldots,j$. Equations 3-18 and 3-19 are evaluated by the previous method again.

After we go through every $\frac{\partial}{\partial b}\left(\frac{dv_j}{dt}\right)$, $j=1,2,\ldots,n$, we can set up a coefficient matrix for $\frac{\partial \ddot{q}_i}{\partial b}$, while moving and combining all the known quantities in Equation 3-14 with the generalized force on the right side of the Lagrange equation. We can solve the n simultaneous equations for $\frac{\partial \ddot{q}_i}{\partial b}$, $i=1,2,\ldots,n$. They are then substituted into Equation 3-8 to obtain the sensitivity, $\frac{\delta F}{\delta b}$. Note that $\frac{\partial \ddot{q}}{\partial b}$ are the only unknowns in that equation.

Finally, $\frac{\partial q}{\partial b}$ and $\frac{\partial \dot{q}}{\partial b}$ would be predicted by integrating $\frac{\partial \ddot{q}}{\partial b}$ at this time, with updated values of generalized forces and the state, velocity, and acceleration obtained from dynamic analysis. We are able to resume the iteration for another $\frac{\partial \ddot{q}}{\partial b}$ until the end of the simulation.

CHAPTER 4

IMPLEMENTATION

In the implementation of system equations, the symbolic manipulation language was chosen for following benefits.

1. All the derivatives involved in the development of the system equation are found automatically and then calculated by the preprocessor so that no further user effort is necessary. On the other hand, if an all-FORTRAN program is used, the user must supply subroutines for evaluating the required values.
2. Most of the errors that occur in the development and use of software arise from mistakes made by either the programmer or the user. By using computers to symbolically derive the equations, the chance of programmer error is greatly reduced. Furthermore, by minimizing the input required from the user, the probability of user error is also lowered. These two factors make REDUCE based programs more reliable.
3. The FORTRAN subroutines generated by REDUCE are problem-dependent because they are constructed specifically for the problem that is being solved. They are much more efficient than the general purpose subroutines. Therefore, the regular trade-off between

the generality and its efficiency does not apply to REDUCE based software.

4. Explicit formulation of expressions can be obtained in FORTRAN code. These could be of interest to the analysts or designers and of help in understanding the behavior of the system.

The system equations of motion have been derived for both the forward and the inverse dynamic problem in Chapter 2. All the terms in these equations can be found by using simple differentiation, and the four basic arithmetic operations in REDUCE. The basic solution procedure to be followed is to feed a system description to the REDUCE program and have the program generate a FORTRAN simulator tailored to the specific robot of interest. We can then feed this FORTRAN program with a set of numeric data and solve the equations of motion numerically. The two step approach was used successfully in the automatic generation of optimizing simulators for general constrained planar mechanical systems [18]. The rationale for this approach is that a symbolic manipulation language has a much higher degree of generality and flexibility, but at the same time, is very slow in arithmetic. To remedy this handicap, many symbolic languages, including REDUCE, provide a facility for writing out the expressions they generate in the form of FORTRAN statements. This approach provides broad generality and flexibility in the symbolic computing step

while the resulting FORTRAN program reflects the efficiency and problem specificity of a special purpose simulator.

The number of additions and multiplications associated with the Lagrange dynamic formulation exhibits quartic variation with respect to the degrees of freedom of the system. If we write the system Equations explicitly as symbolic languages usually do, one can imagine how cumbersome the expressions would be for a system with several degrees of freedom. Further, by expanding all terms, we are placing too many computational demands on the resulting FORTRAN program since the expansions are rarely computationally efficient. To avoid both the huge expressions and the redundant evaluations, we can write out only the frequently used basic terms. Evaluate them once and keep them in memory, then we can use FORTRAN loops to do the multiplications and additions on these known quantities. In the progress of this research, it was found that most of these terms are quite short and can reasonably be handled by the FORTRAN program with every term expanded. Only certain terms such as $\frac{dv}{dt}$, $\frac{\partial v}{\partial q}$, and $\frac{d}{dt}\left(\frac{\partial v}{\partial \dot{q}}\right)$, need special treatment.

It can be seen that the velocity plays an important role in complicating the problem; therefore, it is desirable to express the velocity term in a way that can be

handled by FORTRAN loops. The velocity of the center of mass of link i is obtained by differentiation of the position of the center of mass with respect to time, i.e.

$$v_i=\dot{R}_i \quad , \quad i=1,2,\ldots,n \tag{4-1}$$

Note that v_i and R_i are measured in the base frame.

The position of the center of mass is given in Equation 2-4 by

$$R_i=A_1A_2\ldots A_i[r_i]^T \tag{4-2}$$

The velocity of the center of mass is then

$$v_i=\frac{d}{dt}\{A_1A_2\ldots A_i[r_i]^T\} \tag{4-3}$$

Note that $[r_i]$ is constant in the local frame

$$v_i=[\dot{A}_1A_2\ldots A_i+A_1\dot{A}_2\ldots A_i+\ldots+A_1A_2\ldots\dot{A}_i][r_i] \tag{4-4}$$

now we want to find the time derivative of v_i,

$$\begin{aligned}\frac{dv_i}{dt} = [&(\ddot{A}_1A_2\ldots A_i+\dot{A}_1\dot{A}_2\ldots A_i+\dot{A}_1A_2\ldots\dot{A}_i)\\ &+(\dot{A}_1\dot{A}_2\ldots A_i+A_1\ddot{A}_2\ldots A_i+A_1\dot{A}_2\ldots\dot{A}_i)\\ &\vdots\\ &+(\dot{A}_1A_2\ldots\dot{A}_i+A_1\dot{A}_2\ldots\dot{A}_i+A_1A_2\ldots\ddot{A}_i)][r_i]\end{aligned}$$

$$,i=1,2,\ldots n \tag{4-5}$$

The Equation 4-5 is now in a form that is suitable for programming in FORTRAN loops. The key is to identify which matrices should be used in the different matrix products. If we view the terms in Equation 4-5 as consisting of distinct groups, it is clear that it is made up of ixi

groups, while each group consists of i transformation matrices. We can index the group by means of two indices, namely l and m, and the term number within each group by k. Once each term has been indexed in this manner, the following logic holds:

when l=m

if k=1 then we take $\ddot{A}_k$,

for other k, we take A_k.

when l≠m

if k=1 or m then we take $\dot{A}_k$,

for other k, we take A_k.

Similarly for $\frac{\partial v_i}{\partial q_j}$, we can write

$$\frac{\partial v_i}{\partial q_j} = \left(\frac{\partial \dot{A}_1}{\partial q_j} A_2 \ldots A_i r + \dot{A}_1 \frac{\partial A_2}{\partial q_j} \ldots A_i r + \ldots + \dot{A}_1 A_2 \ldots \frac{\partial A_i}{\partial q_j} r \right)$$

$$+ \left(\frac{\partial A_1}{\partial q_j} \dot{A}_2 \ldots A_i r + A_1 \frac{\partial \dot{A}_2}{\partial q_j} \ldots A_i r + \ldots + A_1 \dot{A}_2 \ldots \frac{\partial A_i}{\partial q_j} r \right)$$

$$\vdots$$

$$+ \left(\frac{\partial A_1}{\partial q_j} A_2 \ldots \dot{A}_i r + A_1 \frac{\partial A_2}{\partial q_j} \ldots \dot{A}_i r + \ldots + A_1 A_2 \ldots \frac{\partial \dot{A}_i}{\partial q_j} r \right) \qquad (4\text{-}6)$$

Since A_i contains only the local generalized coordinate q_i, we can further simplify the Equation by setting the $\frac{\partial A_k}{\partial q_j}$ or $\frac{\partial \dot{A}_k}{\partial q_j}$ terms equal to zero for k≠j. It is easily seen

that only the j^{th} column of groups are non-zero for $\frac{\partial v_i}{\partial q_j}$.

Using the same indexing scheme for l, m, and k as before, the logic for evaluating this term is as follows:

For any fixed j, only m=j is considered

when l=m

if k=l then we take $\frac{\partial \dot{A}_k}{\partial q_j}$,

for other k, we take A_k.

when l≠m

if k=m then we take $\frac{\partial A_k}{\partial q_j}$,

if k=l then we take $\dot{A}_k$,

for other k, we take A_k.

$\frac{d}{dt}(\frac{\partial v_i}{\partial \dot{q}_j})$ can be expanded as

$$\frac{d}{dt}\left(\frac{\partial v_i}{\partial \dot{q}_j}\right) = \frac{d}{dt}\left(\frac{\partial \dot{A}_1}{\partial \dot{q}_j} A_2 \ldots A_i r + A_1 \frac{\partial \dot{A}_2}{\partial \dot{q}_j} \ldots A_i r + \cdots + A_1 A_2 \ldots \frac{\partial \dot{A}_i}{\partial \dot{q}_j} r\right) \qquad (4\text{-}7)$$

Note that the $\frac{\partial A_k}{\partial \dot{q}_j}$ terms are dropped because they are

always zero. The remaining $\frac{\partial \dot{A}_k}{\partial \dot{q}_j}$ terms are not zero only when k=j; therefore, $\frac{\partial v_i}{\partial \dot{q}_j}$ corresponds to the j^{th} group in Equation 4-7.

A further differentiation with respect to time yields

$$\left[\frac{d}{dt}\left(\frac{\partial \dot{A}_1}{\partial \dot{q}_j}\right) A_2 \ldots A_i r + \frac{\partial \dot{A}_1}{\partial \dot{q}_j} \dot{A}_2 \ldots A_i r + \ldots + \frac{\partial \dot{A}_1}{\partial \dot{q}_j} A_2 \ldots \dot{A}_i r\right]$$

$$+\left[\dot{A}_1 \frac{\partial \dot{A}_2}{\partial \dot{q}_j} \ldots A_i r + A_1 \frac{d}{dt}\left(\frac{\partial \dot{A}_2}{\partial \dot{q}_j}\right) \ldots A_i r + \ldots + A_1 \frac{\partial \dot{A}_2}{\partial \dot{q}_j} \ldots \dot{A}_i r\right]$$

$$\vdots$$

$$+\left[\dot{A}_1 A_2 \ldots \frac{\partial \dot{A}_i}{\partial \dot{q}_j} r + A_1 \dot{A}_2 \ldots \frac{\partial \dot{A}_i}{\partial \dot{q}_j} r + \ldots + A_1 A_2 \ldots \frac{d}{dt}\left(\frac{\partial \dot{A}_i}{\partial \dot{q}_j}\right) r\right] \tag{4-8}$$

Now, for $\frac{d}{dt}\left(\frac{\partial v_i}{\partial \dot{q}_j}\right)$, the only non zero terms correspond to the j^{th} row of groups. With l, m, and k defined as before, we have the following logic,

For any fixed j, j=1,2,...,n, only l=j need be considered

when m=l

if k=m then we take $\frac{d}{dt}\left(\frac{\partial \dot{A}_k}{\partial \dot{q}_j}\right)$,

for other k, we take A_k.

when m≠l

if k=l then we take $\frac{\partial \dot{A}_k}{\partial \dot{q}_j}$,

if k=m then we take $\dot{A}_k$,

for other k, we take A_k.

Studying these above expressions, we see that the repeatedly appearing basic terms are the A's, $\dot{A}$'s, $\ddot{A}$'s, $\frac{\partial A_k}{\partial q_j}$, $\frac{\partial \dot{A}_k}{\partial q_j}$, $\frac{d}{dt}\left(\frac{\partial \dot{A}_k}{\partial \dot{q}_j}\right)$, and $\frac{\partial \ddot{A}_k}{\partial \dot{q}_j}$. These expressions are reasonably short and can be explicitly written out in the first part of the FORTRAN program. The matrix multiplication and the logic for evaluating the appropriate matrix products are then programmed in FORTRAN do loops that are also generated by the symbolic language. The other terms in the system equations can also be expanded in the same way; however, it seems to be unnecessary since many elements in the matrices are zero. It would be easier to write them explicitly then to do multiplications on lots of zeros.

As stated in previous chapters, the simulator developed can simulate both forward and inverse dynamics of robot manipulators; therefore,there are two separate programs for the two different purposes.

For the inverse dynamics problem, the FORTRAN program starts by reading in the dimensions and specifications of the robot by calling an input subroutine. Next, we call a subroutine to initialize the joint coordinates, velocities, and accelerations. After this, suitable loops are set up

to obtain the basic terms described earlier and combine them to obtain the required generalized forces.

The basic layout of the forward dynamic simulation is pretty close to the inverse one, except that the big loop now goes into a subroutine and becomes a function. The function in turn is required by DVERK, a subroutine in the IMSL library, that does integration using RUNGE-KUTTA method. Also, what we are interested now is the accelerations; therefore, the $\frac{dv}{dt}$ term developed for the inverse problem is no longer necessary. It is almost impossible to extract the coefficients of the accelerations numerically, fortunately, the REDUCE provides powerful commands that enables us to seperate the terms with and without accelerations. The coefficients are further extracted from the terms with accelerations to form an nxn matrix with known quantities, while all the terms without accelerations are moved to the right hand side to be substracted from the generalized forces. These terms then form a vector with known quantities. The unknown acceleration vector is also established, then the problem becomes solving an n-variable simultaneous equations. This is easily done by calling LEQT2F, another IMSL subroutine. Once the accelerations of links subjected to the corresponding joint force are solved, they are then integrated within specific time step to find new velocities

and positions at next step. The generalized forces are updated before loop for the next step.

CHAPTER 5

NUMERICAL EXAMPLES

The techniques for forward and inverse dynamics that were described in Chapter 2 were implemented in a REDUCE program and this code was used to generate simulators for several example problems. These simulators were then compiled and run to obtain the system response.

The results obtained for some of these examples are presented in this chapter. All the examples were verified by two independent methods:

1) Work-energy balance: A running total of the cumulative work done on the system and the total energy of the system is computed. This information is used to check whether the results obtained are consistent with the work-energy balance equation that must hold for the system. This check is performed independently for the forward and inverse dynamics problems. The trapezoid approach is used to find the work done on each joint. Though it is not the precise solution, just enough for the purpose of checking the results. In all the examples that were run, the check was satisfied within the limits of acceptable numerical error.

2) Cross-check between forward and inverse dynamics: For every system that was considered, a known force input is supplied to the forward dynamics problem and the resulting accelerations, velocities, and positions are obtained. These results are then used as inputs to the inverse dynamics problem for the same system and the forces required to produce this motion are computed. These can then be checked against the original forces that were input to the forward dynamics problem. Ideally, we should obtain exactly the same forces in the two cases; in practice, it was found that they agreed quite accurately, i.e. within the limits of numerical error.

In all the examples presented on the following pages, the results of the work-energy balance and the cross check are presented along with the system response that was obtained by running the simulator.

EXAMPLE 1: 3-R ROBOT

The initial position or home position and coordinate systems are shown in figrue 1. The only load is the self-weight of the members; the simulation time is from 0.00 to 0.18 second.

The input data is given below, and all units are in MKS.

Lengths:

L_1=1.5; L_2=1.0; L_3=0.75;

Masses:

m_1=45; m_2=30; m_3=25;

Moments of inertia:

I_{1x}=0; I_{1y}=0; I_{1z}=8.4375;

I_{2x}=0; I_{2y}=0; I_{1z}=2.5;

I_{3x}=0; I_{3y}=0; I_{3z}=1.17;

Centers of mass:

r_{1x}=0.75; r_{1y}=0; r_{1z}=0;

r_{2x}=0.50; r_{2y}=0; r_{2z}=0;

r_{3x}=0.375; r_{3y}=0; r_{3z}=0;

(1-1) Forward dynamic problem:

Initial conditions:

$q_1=0$; $q_2=0$; $q_3=0$;

$\dot{q}_1=0.2094$; $\dot{q}_2=-0.1396$; $\dot{q}_3=-0.0698$;

Generalized force function at each instant:

$Q_1= 10.0*t^{1/2}+6.0$;

$Q_2= 8.0*t^{1/2}+5.0$;

$Q_3= 6.0*t^{1/2}+4.0$;

where t is the simulation time.

Results:

For link 1:

t	q_1	$\dot{q}_1$	$\ddot{q}_1$
0.00	0.0000e+00	0.2094	0.5179e-01
0.02	0.4200e-02	0.2106	0.6123e-01
0.04	0.8424e-02	0.2118	0.6494e-01
0.06	0.1267e-01	0.2131	0.6764e-01
0.08	0.1695e-01	0.2145	0.6974e-01
0.10	0.2125e-01	0.2159	0.7135e-01
0.12	0.2559e-01	0.2174	0.7249e-01
0.14	0.2995e-01	0.2188	0.7313e-01
0.16	0.3434e-01	0.2203	0.7318e-01
0.18	0.3876e-01	0.2217	0.7254e-01

For link 2:

t	q_2	$\dot{q}_2$	$\ddot{q}_2$
0.00	0.0000e+00	-0.1396	-0.5845
0.02	-0.2919e-02	-0.1526	-0.6994
0.04	-0.6114e-02	-0.1670	-0.7462
0.06	-0.9604e-02	-0.1822	-0.7815
0.08	-0.1341e-01	-0.1980	-0.8104
0.10	-0.1753e-01	-0.2144	-0.8348
0.12	-0.2198e-01	-0.2313	-0.8554
0.14	-0.2678e-01	-0.2485	-0.8724
0.16	-0.3193e-01	-0.2660	-0.8855
0.18	-0.3742e-01	-0.2838	-0.8945

For link 3:

t	q_3	$\dot{q}_3$	$\ddot{q}_3$
0.00	0.0000e+00	-0.6981e-01	2.2970
0.02	-0.8966e-03	-0.1862e-01	2.7667
0.04	-0.7056e-03	0.3828e-01	2.9602
0.06	0.6595e-03	0.9864e-01	3.1076
0.08	0.3260e-02	0.1617	3.2305
0.10	0.7146e-02	0.2272	3.3368
0.12	0.1236e-01	0.2946	3.4303
0.14	0.1894e-01	0.3638	3.5125
0.16	0.2693e-01	0.4346	3.5839
0.18	0.3634e-01	0.5067	3.6444

Work-energy balance check:

t	d(KE)	d(W)
0.00	0.0000e+00	0.0000e+00
0.02	0.8104e-02	0.9018e-02
0.04	0.2120e-01	0.2326e-01
0.06	0.3968e-01	0.4301e-01
0.08	0.6401e-01	0.6873e-01
0.10	0.9464e-01	0.1009
0.12	0.1320	0.1398
0.14	0.1765	0.1860
0.16	0.2285	0.2397
0.18	0.2884	0.3015

where

d(KE) is increase in kenetic energy,

and d(W) is the work done on the system.

(1-2) Inverse dynamics problem:

Initial conditions:

q_1=0.0000e+00; $\dot{q}_1$=0.2094; $\ddot{q}_1$=0.5179e-01;

q_2=0.0000e+00; $\dot{q}_2$=-0.1396; $\ddot{q}_2$=-0.5845;

q_3=0.0000e+00; $\dot{q}_3$=-0.6981e-01; $\ddot{q}_3$=2.2970;

Generalized coordinates, velocities, and accelerations at each instant are specified by the output from the forward dynamic problem.

Results:

t	Q_1	Q_2	Q_3
0.00	0.6000e+01	0.5000e+01	0.4000e+01
0.02	0.7414e+01	0.6131e+01	0.4849e+01
0.04	0.8000e+01	0.6600e+01	0.5200e+01
0.06	0.8449e+01	0.6960e+01	0.5470e+01
0.08	0.8828e+01	0.7263e+01	0.5697e+01
0.10	0.9162e+01	0.7530e+01	0.5897e+01
0.12	0.9464e+01	0.7771e+01	0.6078e+01
0.14	0.9742e+01	0.7993e+01	0.6245e+01
0.16	0.1000e+02	0.8200e+01	0.6400e+01
0.18	0.1024e+02	0.8394e+01	0.6546e+01

The exact values as given by the functions:

t	Q_1	Q_2	Q_3
0.00	0.6000E+01	0.5000E+01	0.4000e+01
0.02	0.7414e+01	0.6131e+01	0.4849e+01
0.04	0.8000e+01	0.6600e+01	0.5200e+01
0.06	0.8450e+01	0.6960e+01	0.5470e+01
0.08	0.8828e+01	0.7263e+01	0.5697e+01
0.10	0.9162e+01	0.7530e+01	0.5897e+01
0.12	0.9464e+01	0.7771e+01	0.6078e+01
0.14	0.9742e+01	0.7993e+01	0.6245e+01
0.16	0.1000e+02	0.8200e+01	0.6400e+01
0.18	0.1024e+02	0.8394e+01	0.6546e+01

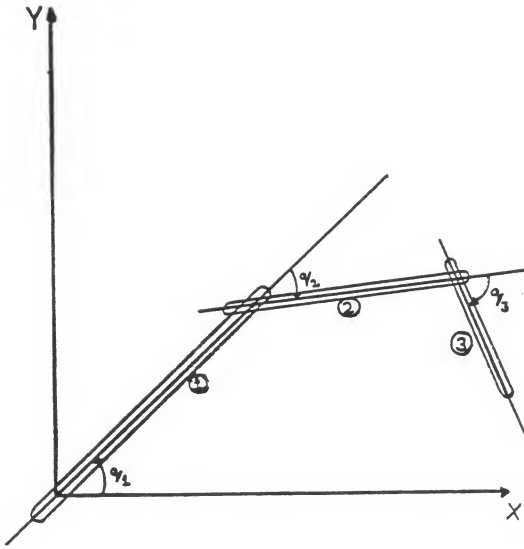

Figure 1

EXAMPLE 2: 4-LINK ROBOT

The initial position or home position and coordinate systems are shown in figrue 2. The only load is the self-weight of the members; the simulation time is from 0.00 to 0.18 second.

The input data is given below, and all units are in MKS.

Lengths:

L_1=1.5; L_2=1.0; L_3=0.75; L_4=0.25;

Masses:

m_1=50; m_2=30; m_3=20; m_4=15;

Moments of inertia:

I_{1x}=9.375; I_{1y}=1.0; I_{1z}=9.375;

I_{2x}=0.15; I_{2y}=2.5; I_{1z}=2.5;

I_{3x}=0.1; I_{3y}=0.9375; I_{3z}=0.9375;

I_{4x}=0.075; I_{4y}=0.8; I_{4z}=0.8;

Centers of mass:

r_{1x}=0; r_{1y}=0.75; r_{1z}=0;

r_{2x}=0.50; r_{2y}=0; r_{2z}=0;

r_{3x}=0.375; r_{3y}=0; r_{3z}=0;

r_{4x}=-0.15; r_{4y}=0; r_{4z}=0;

(2-1) Forward dynamic problem:

Initial conditions:

q_1=0; q_2=0; q_3=0; q_4=0;

$\dot{q}_1$=0.3491; $\dot{q}_2$=0.2618; $\dot{q}_3$=-0.2618; $\dot{q}_4$=0.08727

Generalized force function at each instant:

Q_1= 10.0*t^1/2+6.0;

Q_2= 8.0*t^1/2+5.0;

Q_3= 6.0*t^1/2+4.0;

Q_4= 4.0*t^1/2+3.0;

where t is the simulation time.

Results:

For link 1:

t	q_1	$\dot{q}_1$	$\ddot{q}_1$
0.00	0.0000e+00	0.3491	-0.2486
0.02	-0.6928e-02	0.3436	-0.2920
0.04	0.1374e-01	0.3376	-0.3086
0.06	0.2043e-01	0.3314	-0.3206
0.08	0.2699e-01	0.3249	-0.3302
0.10	0.3343e-01	0.3182	-0.3382
0.12	0.3972e-01	0.3114	-0.3450
0.14	0.4588e-01	0.3045	-0.3508
0.16	0.5190e-01	0.2974	-0.3560
0.18	0.5778e-01	0.2903	-0.3604

For link 2:

t	q_2	$\dot{q}_2$	$\ddot{q}_2$
0.00	0.0000e+00	0.2618	0.4438e-01
0.02	0.5246e-02	0.2628	0.5564e-01
0.04	0.1051e-01	0.2640	0.5920e-01
0.06	0.1580e-01	0.2652	0.6136e-01
0.08	0.2112e-01	0.2664	0.6275e-01
0.10	0.2646e-01	0.2676	0.6361e-01
0.12	0.3183e-01	0.2689	0.6405e-01
0.14	0.3722e-01	0.2702	0.6417e-01
0.16	0.4263e-01	0.2715	0.6400e-01
0.18	0.4808e-01	0.2727	0.6361e-01

For link 3:

t	q_3	$\dot{q}_3$	$\ddot{q}_3$
0.00	0.0000e+00	-0.2618	1.0492
0.02	-0.5009e-02	-0.2386	1.2471
0.04	-0.9527e-02	-0.2130	1.3221
0.06	-0.1352e-01	-0.1862	1.3749
0.08	-0.1697e-01	-0.1584	1.4151
0.10	-0.1985e-01	-0.1299	1.4466
0.12	-0.2216e-01	-0.1008	1.4711
0.14	-0.2388e-01	-0.7130e-01	1.4900
0.16	-0.2501e-01	-0.4144e-01	1.5038
0.18	-0.2554e-01	-0.1135	1.5130

For link 4:

t	q_4	$\dot{q}_4$	$\ddot{q}_4$
0.00	0.0000e+00	0.8727e-01	0.4361
0.02	0.1836e-02	0.9639e-01	0.4716
0.04	0.3858e-02	0.1059	0.4854
0.06	0.6075e-02	0.1157	0.4964
0.08	0.8489e-02	0.1257	0.5062
0.10	0.1110e-01	0.1359	0.5157
0.12	0.1393e-01	0.1463	0.5253
0.14	0.1696e-01	0.1569	0.5352
0.16	0.2020e-01	0.1677	0.5457
0.18	0.2367e-01	0.1787	0.5568

Work-energy balance check:

t	d(KE)	d(W)
0.00	0.0000e+00	0.0000e+00
0.02	0.6028e-01	0.6137e-01
0.04	0.1306	0.1329
0.06	0.2083	0.2119
0.08	0.2930	0.2977
0.10	0.3842	0.3901
0.12	0.4819	0.4890
0.14	0.5861	0.5944
0.16	0.6967	0.7062
0.18	0.8137	0.8244

where

d(KE) is increase in kenetic energy,

and d(W) is the work done on the system.

(2-2) Inverse dynamics problem:

Initial conditions:

q_1=0.0000; $\dot{q}_1$=0.3491; $\ddot{q}_1$=-0.2486;

q_2=0.0000; $\dot{q}_2$=0.2618; $\ddot{q}_2$=0.04438;

q_3=0.0000; $\dot{q}_3$=-0.2618; $\ddot{q}_3$=1.0492;

q_4=0.0000; $\dot{q}_4$=0.08727; $\ddot{q}_4$=0.4361;

Generalized coordinates, velocities, and accelerations at each instant are specified by the output from the forward dynamic problem.

Results:

t	Q_1	Q_2	Q_3	Q_4
0.00	0.6000e+01	0.5000e+01	0.4000e+01	0.3000e+01
0.02	0.7414e+01	0.6131e+01	0.4848e+01	0.3566e+01
0.04	0.7999e+01	0.6600e+01	0.5200e+01	0.3800e+01
0.06	0.8448e+01	0.6960e+01	0.5469e+01	0.3980e+01
0.08	0.8827e+01	0.7263e+01	0.5696e+01	0.4131e+01
0.10	0.9160e+01	0.7530e+01	0.5897e+01	0.4265e+01
0.12	0.9462e+01	0.7772e+01	0.6078e+01	0.4386e+01
0.14	0.9739e+01	0.7994e+01	0.6244e+01	0.4497e+01
0.16	0.9997e+01	0.8201e+01	0.6399e+01	0.4600e+01
0.18	0.1024e+02	0.8395e+01	0.6545e+01	0.4697e+01

The exact values as given by the functions:

t	Q_1	Q_2	Q_3	Q_4
0.00	0.6000E+01	0.5000E+01	0.4000e+01	0.3000e+01
0.02	0.7414e+01	0.6131e+01	0.4849e+01	0.3566e+01
0.04	0.8000e+01	0.6600e+01	0.5200e+01	0.3800e+01
0.06	0.8450e+01	0.6960e+01	0.5470e+01	0.3980e+01
0.08	0.8828e+01	0.7263e+01	0.5697e+01	0.4131e+01
0.10	0.9162e+01	0.7530e+01	0.5897e+01	0.4265e+01
0.12	0.9464e+01	0.7771e+01	0.6078e+01	0.4386e+01
0.14	0.9742e+01	0.7993e+01	0.6245e+01	0.4497e+01
0.16	0.1000e+02	0.8200e+01	0.6400e+01	0.4600e+01
0.18	0.1024e+02	0.8394e+01	0.6546e+01	0.4697e+01

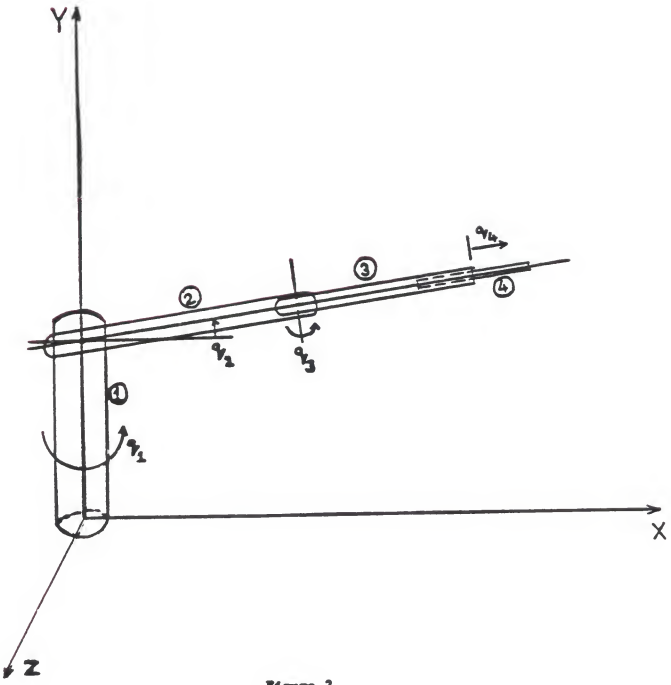

Figure 2

CHAPTER 6

CONCLUSIONS

The major objective of this research was to develop an automated simulator generator for general serial-link open-chain robots by using the symbolic language REDUCE-3. The simulator, based on the Lagrangian formulation, can solve both the forward and the inverse dynamic problems of any robot specified by the user, provided only revolute and translational joints are used.

The system equations are set up for user-specified robot and solved without any modifications or simplifications so that exact simulations (within round-off) can be obtained, the solution of the system equations is done in a way that avoids excessively large explicit expressions and greatly reduces the computing time. A more efficient way of using symbolic languages was also developed through the compromise between REDUCE and FORTRAN languages, as suggested in [18].

Several advantages come with the use of symbolic processing in this application. First of all, it eliminates the time-consuming and tedious manual derivation process. Secondly, the simulators generated would be very reliable since the possibility of programmer error is

minimized. Another benefit is the remarkable flexibility of the simulator generation program, because this program computes all quantities symbolically rather than numerically, it can handle a wide variety of problems without modification. This become important when attempting to extend the scope of the simulators to include controls and optimization.

The work presented in this thesis offers wide scope for future development. The techniques used to express and solve the Lagrangian equation can be applied to various mechanical systems, since most real-world mechanical dynamics problem can be described by the Lagrangian.

The method for describing the geometry of robots can be improved and standardized by adopting the widely used Denavit-Hartenberg convention. The schemes used for calculating complicated terms in the system equation could be further improved by treating the matrices in a more efficient way and by finding a better trade-off between computing time and memory space. Due to the presumption that only one degree of freedom, or one generalized coordinate is associated with each link, the simulators generated becomes clumsy when simulating a multi-degree of freedom link, in which case a single link must be treated as several artificial links. Improvements on this defect can result in better, more efficient simulation of wrist joints and other higher order pairs.

In the context of on-line dynamic control, it was suggested in [19] that the canonical formulation has advantages over Lagrange's equations of motion, since it avoids taking time derivatives, which in turn results in simpler expressions in the equations of motion. Also, the use of super computers, array processors, or parallel processing computers would help in obtaining numeric solutions within the sampling time.

There is ample scope for development in computer-related areas as well. Computer graphics can be used to display the results of dynamic analysis. Interactive commands could also be developed to make the program more user-friendly and to give the user more control over the design process. An expert system for design purposes could be included in the preprocessor to determine the basic profile and mechanical requirements of robots according to their uses.

Since it is nearly impossible to standardize problems in the design of mechanical systems. Automatic simulator generators are extremely attractive as tools for the simulation, analysis, design , and optimization of robot manipulators; they are also a valuable aid in planning the efficient utilization of these manipulators for specific tasks. This is certainly an area worthy of more research work and the one which will become increasingly popular in the near future.

LIST OF REFERENCES

1. Paul, R. P. C., "Modeling Trajectory Calculation and Servoing of a Computer Controlled Arm," A.I. Memo 177, Stanford Artificial Intelligence Laboratory, Stanford University, Stanford, California, September 1972.

2. Uicker, J. J., "On the Dynamic Analysis of Spatial links using 4x4 Matrices," doctoral dissertation, Northwestern University, Evanston, Illinois, 1965.

3. Lewis, R. A., "Autonomous Manipulation of a Robot: Summary of Manipulator Software Functions," Technical Memorandum 33-679, Jet Propulsion Laboratory, Pasadena, California, March 1974.

4. Williams, R. J., and Seireg, A., "Interactive Modeling and Analysis of Open or Closed Loop Dynamic Systems with Redundant Actuators," ASME Journal of Mechanical Design, Volume 101, No.3, July 1979, pp.407-416.

5. Orin, D. E., Mcghee, R. B., Vukobratovic, M., and Hartoch, G., "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods," Mathematical Biosciences, Volume 43, No.1/2, February 1979, pp.107-130.

6. Luh, J. Y. S., Walker M. W., and Paul, R. P. C., "On Line Computational scheme for Mechanical manipulators," ASME Journal of Dynamic Systems, Measurement, and Control, Volume 102, June 1980, pp. 69-76.

7. Stepanenko, Y., and Vukobratovic, M., "Dynamics of Articulated Open-Chain Active Mechanisms," Mathematical Biosciences, Volume 28, No.1/2, 1976.

8. Hollerbach, J. M., "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," IEEE Transactions on Systems, Man, and Cybernetics, Volume SMC-10, No. 11, November 1980.

9. Armstrong, W. W., "Recursive Solution to the Equations of Motion of an N-link Manipulator," in Proc. 5th World Congress, Theory of Machines, Mechanisms, Volume 2, July 1979, pp.1343-1346.

10. Raibert, M. H., and Horn, B. K. P., "Manipulator Control Using the Configuration Space Method," The Industrial Robot, Volume 5, No. 2, June 1978, pp.69-73.

11. Bejczy, A. K., "Robot Arm Dynamics and Control," Technical Memorandum 33-669, Jet Propulsion Laboratory, February 1974.

12. Leu, M. C., Hemati, N., "Automated Symbolic Derivation of Dynamic Equations of Motion for Robotic Manipulators," ASME Journal of Dynamic Systems, Measurement, and Controls, Volume 108, September 1986.

13. Denavit, J., and Hartenberg, R. S., "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices," ASME Journal of Applied Mechanics, Volume 22, 1955, pp.215-221.

14. Paul, R. P., Robot Manipulators: Mathematics, Programming, and Control, MIT Press,1981.

15. Schiehlen, W. O., and Kreuzer, E. J., "Symbolic Computerized Derivation of Equations of Motion," Proceedings of IUTAM Symposium, Munich, Germany, August 29-September 3, 1977.

16. Liegois, A., Khalil, W., Dumas, J. M., and Renaud, M., "Mathematical and Computer Models of Interconnected Mechanical Systems," Proceedings of 2nd International CISM-IFToMM Symposium, Warsaw, September 1976, pp.5-17.

17. Vecchio, L., Nicosia, S., Nicolo, F., and Lentini, D., "Automatic Generation of Dynamic Models of Manipulators," 10th International Symposium on Industrial Robots, Milan, Italy, March 5-7, 1980

18. Krishnaswami, P., "Computer-Aided Optimal Design of Constrained Dynamic Systems," Doctoral Dissertation, The University of Iowa, Iowa City, Iowa, December 1983.

19. Sitchin, A., "On the Use of the Canonical Equations of Motion for the Numerical Solution of Dynamical Systems," Developments in Mechanics, Volume 8. Proceedings of the 14th Midwestern Mechanics Conference.

APPENDIX A

The input to the REDUCE based simulator generator for both examples are given below.

For the 3-R ROBOT in example 1:

```
NUML:=3;
L(1):=RL(1); L(2):=RL(2); L(3):=RL(3);
CM(1,1):=XCM(1); CM(1,2):=0; CM(1,3):=0; CM(1,4)=1;
CM(2,1):=XCM(2); CM(2,2):=0; CM(2,3):=0; CM(2,4)=1;
CM(3,1):=XCM(3); CM(3,2):=0; CM(3,3):=0; CM(3,4)=1;
MAS(1):=RM(1); MAS(2):=RM(2); MAS(3):=RM(3);
MI(1,1):=RMI(1,1); MI(1,2):=RMI(1,2); MI(1,3):=RMI(1,3);
MI(2,1):=RMI(2,1); MI(2,2):=RMI(2,2); MI(2,3):=RMI(2,3);
MI(3,1):=RMI(3,1); MI(3,2):=RMI(3,2); MI(3,3):=RMI(3,3);
DIRECT(1):=X; DIRECT(2):=X; DIRECT(3):=X;
JOINT(1):=REV; JOINT(2):=REV; JOINT(3):=REV;
AXIS(1):=Z; AXIS(2):=Z; AXIS(3):=Z; ;END;
```

For the 4-LINK ROBOT in example 2:

```
NUML:=4;
L(1):=RL(1); L(2):=RL(2); L(3):=RL(3); L(4):=RL(4);
CM(1,1):=0; CM(1,2):=YCM(1); CM(1,3):=0; CM(1,4)=1;
CM(2,1):=XCM(2); CM(2,2):=0; CM(2,3):=0; CM(2,4)=1;
CM(3,1):=XCM(3); CM(3,2):=0; CM(3,3):=0; CM(3,4)=1;
CM(4,1):=XCM(4); CM(4,2):=0; CM(4,3):=0; CM(4,4)=1;
MAS(1):=RM(1); MAS(2):=RM(2); MAS(3):=RM(3); MAS(4):=RM(4);
MI(1,1):=RMI(1,1); MI(1,2):=RMI(1,2); MI(1,3):=RMI(1,3);
MI(2,1):=RMI(2,1); MI(2,2):=RMI(2,2); MI(2,3):=RMI(2,3);
MI(3,1):=RMI(3,1); MI(3,2):=RMI(3,2); MI(3,3):=RMI(3,3);
MI(4,1):=RMI(4,1); MI(4,2):=RMI(4,2); MI(4,3):=RMI(4,3);
DIRECT(1):=Y; DIRECT(2):=X; DIRECT(3):=X; DIRECT(4):=X;
JOINT(1):=REV; JOINT(2):=REV; JOINT(3):=REV; JOINT(4):=TRAN;
AXIS(1):=Y; AXIS(2):=Z; AXIS(3):=Y; ;END;
```

APPENDIX B

The meaning of all the variables in APPENDIX A is given in this section as follows.

NUML : the number of links

L : the length of the link

CM : the location of the center of mass of the link

MAS : the mass of the link

MI : the moment of inertia of the link

DIRECT : the direction in which the link is pointing

JOINT : the joint type

AXIS : the axis of rotation of the link

AUTOMATED GENERATION OF ROBOTIC DYNAMICS SIMULATORS THROUGH SYMBOLIC COMPUTING

by

JUN-TIEN TWU

B. S., National Taiwan University, 1985

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

ABSTRACT

This thesis presents a generalized technique for generating simulators for serial, open-chain robot manipulators using symbolic computing. These simulators can handle both direct dynamics problem as well as the inverse dynamics problem. In both cases, the general governing equations are derived using joint coordinates and the principles of Lagrangian mechanics. These equations are then recast into a form that is better suited for implementation in the symbolic processing language REDUCE. A symbolic processor, written in REDUCE, has been set up to evaluate all the required expressions in these equations and to write these out in the form of a complete FORTRAN simulation program; this FORTRAN simulator is then executed to obtain the system response. Numerical examples are presented to demonstrate the efficiency of the method and to establish the feasibility of the proposed technique of generating simulators automatically.

The detail of design sensitivity analysis is developed so that it is ready for implementing to the symbolic computing language. Combining the sensitivity and dynamics analysis, the simulator generated is then able to be extended to include the design optimization/update part for improved designs.