Arm Dynamics Simulation

Neil M. Swartz

CMU-RI-TR-82-17

Vision Laboratory
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

November 1982

This work is supported by the Office of Naval Research, Contract No. N00014-82-K-0503.

Table of Contents

Notation:

- The vector sign ($\vec{\ }$) indicates a three dimensional spatial vector.
- An underbar beneath a symbol indicates an "arm vector" which has one component for each joint in the arm. For example, $\underline{\theta}$ is equal to ($\theta_1, \theta_2, ..., \theta_N$) when there are N joints in the arm.
- Subscripts indicate one of two things:
  1. The coordinate frame in which the symbol is referenced and the frame to which the symbol refers.
  2. The joint number that is referred to by the symbol.

  The type of symbol will determine whether the subscript is a link coordinate frame or a joint number. For example, $\vec{\omega}_i$ is the angular velocity of link *i* in the link *i* coordinate system and θ_i is the position of joint *i*.
- A hat on a symbol ($\hat{\ }$) indicates a constant.
- A star superscript (*) indicates a value related to the center of mass. Other values are related to the link coordinate frame.
- A dot and a double dot over a variable ($\dot{\ }$ and $\ddot{\ }$) indicate first and second time derivatives respectively.
- An A with a superscript and a subscript represents a transformation matrix from the superscripted coordinate system to the subscripted coordinate system; so $A_1^2 \omega_2$ is the angular velocity of the link 2 coordinate origin in link 1 coordinates.
- Boldfaced letters represent matrices or tensors, so **J** is a moment of inertia tensor.

Note that a reference to the velocity or acceleration of a link actually refers to the velocity or acceleration of the link's coordinate system origin.

Abstract

The ability to mathematically model the movement of a robot manipulator is a prerequisite to the understanding of the key factors that influence a manipulator's performance. This paper presents a manipulator model which has been used to simulate and control a real robot arm. A method of describing the arm by its rotational characteristics, a set of equations called the *Inverse Arm*, and an algorithm called *Forward Arm* are presented. The Forward Arm simulates the movement of an arm, and the Inverse Arm provides a means of computing the correct voltages to apply to an arm to achieve a desired movement.

Introduction

A mathematical model of a physical system, such as a robot manipulator(arm), is one of the most useful tools available for studying the system's behavior. The model, usually in the form of a computer program, can be used to study the system in several ways. The development of an accurate model leads to a full understanding of all of the key elements of the system. The model provides a means of testing the system under conditions that would be dangerous or impossible for the real system. Larger systems that contain the modeled system can be tested with the model instead of the real system.

As part of the Carnegie-Mellon University Direct-Drive Manipulator Project (CMU DD Arm Project) we have developed a mathematical model of the manipulator. The mathematical equations are based on a Newton-Euler analysis of free-body dynamics developed for robotic manipulators. [8] [11]

This paper describes the structure of the model that simulates the dynamic motions of our manipulator. The model is divided into three parts.

- *A detailed description of the structure of the arm.* The description of the structure is contained in a **Manipulator Database** which consists of two parts: the kinematic and the dynamic. The kinematic description specifies the relative positions between the links of the arm and gives the axes of rotation for each of the joints. This description is easily determined from the mechanical drawings of the arm. The dynamic description contains the moment of inertia, the center of mass, and the mass for each of the links. A computer program was written to calculate these values from a database file(the **Parts Database**) that contains a description of every piece of the arm.

- *The Inverse Arm.* This is a set of equations which, when evaluated, yield the motor voltages required to produce certain accelerations. This is the inverse of a real arm which produces accelerations given the voltages. The *Inverse Arm* part of the model is needed for the third part which is the *Forward Arm.*

- *The Forward Arm.* This part of the model contains an algorithm which can compute values for the acceleration of the joints in the arm when the motor voltages are specified. When the accelerations are integrated over a period of time, the new positions and velocities can be determined.

Arm Description

The CMU DD Arm consists of seven links, numbered 0 to 6, going from the base (link 0) down to the hand (link 6). There are six joints numbered 1 to 6. The odd numbered joints are *rotational* joints which rotate in the same manner as the turning of a screw. The even numbered joints are *pivotal* joints, which move in a manner similar to that of a person's elbow.

Each link has a coordinate frame associated with it. The Denavit-Hartenburg convention [5] for assigning coordinate frames to manipulator links is used to specify the coordinate frames of the manipulator because it simplifies the evaluation of the equations used in the *Inverse Arm* and *Forward Arm* parts of the model.

The Denavit-Hartenburg convention specifies that link *i+1* rotates around the Z axis of link *i*, denoted $\mathbf{Z}_i$, when joint *i+1* turns. Link 1, therefore, rotates around $\mathbf{Z}_0$ at joint 1 and link 2 around $\mathbf{Z}_1$ at joint 2, etc. The X axis of each link's coordinate system points along the common normal of the link's Z axis and the Z axis of the previous link. If there is no common normal, such as when the two Z axes intersect, the direction of the X axis is arbitrary, so long as it is perpendicular to the Z axis. The Y-axis is perpendicular to both the X and Z axes and completes a right-handed coordinate frame. The coordinate frames specified by the Denavit-Hartenburg convention for the CMU DD Arm are shown in figure 1. [4]

Kinematic Description

The kinematic description specifies the general organization of a manipulator. The description is a database which contains three pieces of information, denoted $\boldsymbol{\alpha}$, **r**, and **a**, for each joint. For any joint *i*, $\boldsymbol{\alpha}$ specifies the angle of rotation necessary to bring $\mathbf{Z}_{i-1}$ parallel to or collinear with $\mathbf{Z}_i$. The parameter **r** specifies the distance along the $\mathbf{Z}_{i-1}$ axis from the link *i-1* coordinate system to the link *i* coordinate system. The **a** parameter specifies the distance from link *i-1* to link *i* along the $\mathbf{X}_{i-1}$ axis. Once the coordinate frame origins are determined in the Denavit-Hartenburg convention the $\boldsymbol{\alpha}$, **r**, and **a** parameters can be obtained from the mechanical drawings.

Dynamic Description

Obtaining a dynamic description of the arm requires a greater effort than the kinematic description. The dynamic description consists of the moment of inertia tensor, center of mass vector, and the mass scalar of each link. To determine this data we have developed a **Parts Database**[1] for the CMU-DD Arm which has information on each of the six links. Within the database, each *link* is separated into several *parts* which are numbered in the mechanical drawings. Each *part* is broken down into several *sections*. Each *section* is described as a cylinder, semi-cylinder, rectangle, sphere, or prism. The characteristics of each of the *parts* in our manipulator can be approximated by combinations of these shapes. The database contains dimensions, densities, and locations of each *section* relative to the *part* that it belongs to. With this information we can determine the moment of inertia, center of mass, and mass of each *section*. Once the information is calculated for each *section*, it can then be determined for each *part* using the formulas for transforming moments of inertia. [7] [10][2] The dynamic description of each *link* can be computed in a similar manner from the *part* information. The resultant description is saved as part of the **Manipulator Database** and is referred to by the rest of the arm model. It must be recalculated if the construction of the arm changes.

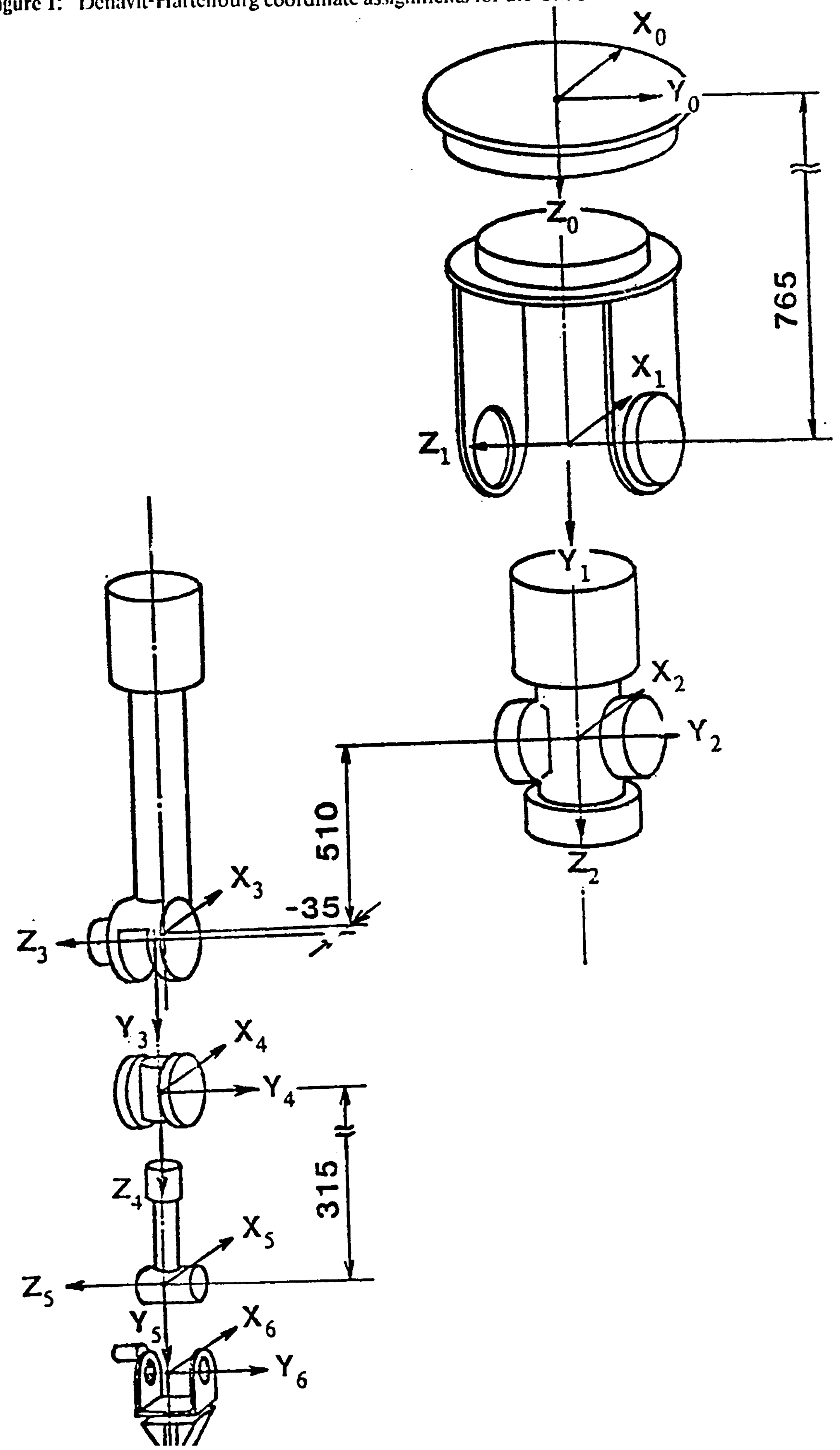
There are other pieces of data related to the dynamic description of the arm which can only be experimently determined. The most important of these are the motor constants. Each motor has a resistance, **R**, and a torque constant, **Kt**. The resistance of the motor relates the current in each motor's armature to the voltage applied across each motor's terminals. The torque constant relates the torque produced by the motor to the current in the armature and it relates the speed of the motor to the back EMF(Electro-Motive Force).

---

[1]The link coordinate systems used in the parts database are assigned for convenience. The software which generates the dynamic description changes these coordinate systems to those of the Denavit-Hartenburg convention when the **Manipulator Database** is produced.

[2]Chapter 10.5 The Inertia Tensor

Figure 1: Denavit-Hartenburg coordinate assignments for the CMU DD Arm



Inverse Arm

The *Inverse Arm* model is an application of a Newton-Euler analysis of free-body motion. The purpose of this model is to allow us to compute the motor voltages required to produce given accelerations when we know the current state of the arm and all of its parameters. The model has six major steps.

1. Calculation of the velocities and accelerations of each of the links.

2. Finding the linear acceleration of each of the link's center of mass.

3. Computing the net force and moment exerted on each link.

4. Calculation of the local forces and moments on each link.

5. Finding the torque required for each motor.

6. Computing the motor voltage required to produce the computed torques.

The last step is done separately from the first five so that the *Forward Arm* program can use the first five steps of the *Inverse Arm* to find torques.

In this paper a reference to the velocity or acceleration of a link actually refers to the velocity or acceleration of the coordinate system imbedded in the link.

Link Velocities and Accelerations

There are two forms of link velocities and accelerations which are considered here, angular and linear. We have four equations which can be solved iteratively from link 1 to link N to find the angular velocity, angular acceleration, linear velocity, and linear acceleration of each of the links in the arm. Link 0 is assumed to have no angular or linear velocity and no angular acceleration(i.e. $\vec{\omega}_0 = \vec{v}_0 = \dot{\vec{\omega}}_0 = [0\ 0\ 0]^T$). It does, however, have a linear acceleration equal to a Z directed gravitational acceleration (i.e. $\dot{\vec{v}}_0 = [0\ 0\ g]^T$, $g = \pm 9.80621$ meters/second^2, depending upon whether Z points up or down). Since we know $\vec{\omega}_0$, $\vec{v}_0$, $\dot{\vec{\omega}}_0$, and $\dot{\vec{v}}_0$ we can use the following four equations to solve for $\vec{\omega}_1$, $\vec{v}_1$, $\dot{\vec{\omega}}_1$, and $\dot{\vec{v}}_1$. We can then apply the equations repeatedly to solve for the velocities and accelerations of links 2, 3, etc. up through link N.

The angular velocity of link *i+1*, $\vec{\omega}_{i+1}$, is related to the angular velocity of link *i*, $\vec{\omega}_i$, and the rate at which the joint between them, θ_{i+1}, turns by

$$\vec{\omega}_{i+1} = A^i_{i+1}\,(\,\vec{\omega}_i + \hat{Z}_0\,\dot{\theta}_{i+1}) \qquad i = 0,1,\ldots,N\text{-}1 \qquad (1)$$

where $\hat{Z}_0 = [0\ 0\ 1]^T$ and N=6 in the CMU DD Arm. The Denavit-Hartenburg convention dictates that the axis of rotation of joint *i+1* is along the Z axis of the link *i* coordinate frame, so the rate of turning of joint *i+1* is multiplied by $\hat{Z}_0$ and added to the angular velocity of link *i* to give the angular velocity of link *i+1*. The coordinate frame is changed from link *i* to link *i+1* by premultipling by A^i_{i+1}.

The angular acceleration of link *i+1*, $\dot{\vec{\omega}}_{i+1}$, is given by:

$$\dot{\vec{\omega}}_{i+1} = A^i_{i+1} (\dot{\vec{\omega}}_i + \hat{Z}_0\ddot{\theta}_{i+1} + \vec{\omega}_i \times \hat{Z}_0\dot{\theta}_{i+1}) \qquad i = 0,1,...,N-1. \qquad (2)$$

Since joint *i+1* rotates around the Z axis of link *i*, the acceleration of joint *i+1* is multiplied by $\hat{Z}_0$ before being changed to link *i+1* coordinates. The cross product term comes from the geometry of the situation. [10][3] A coordinate transformation from link *i* to link *i+1* coordinates is performed by a premultiplication by A^i_{i+1}.

The linear velocity of link *i+1*, $\vec{v}_{i+1}$, is related to the linear velocity of link *i*, $\vec{v}_i$, and the cross product of $\vec{\omega}_{i+1}$ and the vector, $\hat{p}_{i+1}$, which points from the link *i* to the link *i+1* coordinate system.

$$\vec{v}_{i+1} = \vec{\omega}_{i+1} \times \hat{p}_{i+1} + A^i_{i+1}\vec{v}_i \qquad i = 0,1,...,N-1 \qquad (3)$$

where $\hat{p}_{i+1}$ is given by $[\ a_{i+1} \quad r_{i+1}\sin(\alpha_{i+1}) \quad r_{i+1}\cos(\alpha_{i+1})\ ]^T$ in the Denavit-Hartenburg convention. The linear velocity of the link *i* coordinate system is transformed to the link *i+1* coordinate system by a premultiplication by A^i_{i+1}. The cross product need not be transformed because it is already expressed in link *i+1* coordinates. Note that the linear velocity of each link is not used in later calculations. This equation need not be evaluated, but is included for completeness.

The linear acceleration of link *i+1*, $\dot{\vec{v}}_{i+1}$, is given by

$$\dot{\vec{v}}_{i+1} = \dot{\vec{\omega}}_{i+1} \times \hat{p}_{i+1} + \vec{\omega}_{i+1} \times (\vec{\omega}_{i+1} \times \hat{p}_{i+1}) + A^i_{i+1}\dot{\vec{v}}_i \qquad i = 0,1,...,N-1. \qquad (4)$$

The first term is, again, due to the geometry of the situation and the second is called the *Centripetal* acceleration. This equation is a limited case of the Coriolis theorem. [10] Because there are no translational joints in our arm, the coriolis term of the Coriolis theorem is zero.

Linear Acceleration of the Centers of Mass

The calculation of the linear acceleration of the center of mass of each of the links is very similar to the linear acceleration of the coordinate system calculation. The equation relating the linear acceleration of the center of mass of a link to $\vec{\omega}_i$, $\dot{\vec{\omega}}_i$, $\hat{s}^*_i$, and $\dot{\vec{v}}_i$ is

$$\dot{\vec{v}}^*_i = \dot{\vec{\omega}}_i \times \hat{s}^*_i + \vec{\omega}_i \times (\vec{\omega}_i \times \hat{s}^*_i) + \dot{\vec{v}}_i \qquad i = 1,2,...,N \qquad (5)$$

where $\hat{s}^*_i$ is a vector pointing to the center of mass of link *i* from its coordinate origin. Again, we see that there are no coriolis accelerations in the arm. Note that these calculations can be performed in any order, but must be performed after equations 1 through 4 have been evaluated for all of the links.

Net Forces and Moments

The net force is the sum of all of the forces acting on a link. Likewise, the net moment is the sum of all of the moments. Since we know what the accelerations are we can calculate the net forces and net moments for each link using Newton's law and its analog in rotational dynamics. Newton's law in this context is

$$\vec{F}_i = m_i\dot{\vec{v}}^*_i \qquad i = 1,2,...,N \qquad (6)$$

where m_i is the mass of the link.

[3]Chapter 7.2 Moving Origin of Coordinates

Newton's law can be derived from the fact that the net force is equal to the rate of change of the momentum. In a similar manner, we can derive the rotational analogy of Newton's law from the fact that the net moment is equal to the rate of change of the rotational momentum or

$$\vec{N}_i = \frac{d\vec{L}_i}{dt}$$

where

$$\vec{L}_i = J_i^* \dot{\vec{\omega}}_i.$$

$\vec{L}_i$ is the rotational momentum and J_i^* is the moment of inertia of link *i* around its center of mass. Since we need to express the moment with respect to link coordinate origins we change the coordinates of the moment. The moment coordinate change formula is:

$$\vec{N}_i = J_i^* \dot{\vec{\omega}}_i + \vec{\omega}_i \times (J_i^* \vec{\omega}_i) \qquad i = 1,2,...,N. \qquad (7)$$

This is a form of Euler's equation of motion for a rigid body. [10][4]

Local Forces and Moments

Each link is connected to two other links (except the hand and the base) which exert forces on that link. The sum of these two forces is the net force. For any link *i*, the force, $\vec{f}_i$, that link *i-1* exerts on it is called the local force. The net force, $\vec{F}_i$, on link *i* is the sum of the local force, $\vec{f}_i$, and the negative of the local force on the next link, $\vec{f}_{i+1}$, that is

$$\vec{F}_i = \vec{f}_i - A_i^{i+1} \vec{f}_{i+1}$$

or

$$\vec{f}_i = A_i^{i+1} \vec{f}_{i+1} + \vec{F}_i \qquad i = N,...,2,1. \qquad (8)$$

Note that we must change the coordinate system of $\vec{f}_{i+1}$ before adding it to $\vec{F}_i$ by premultiplying it by A_i^{i+1} We can calculate the local forces by solving this equation iteratively starting at the hand, where $\vec{f}_{N+1}$ is the external force exerted on the hand, and working our way up the arm.

The net moment, $\vec{N}_i$, of link *i* has four components.

1. The local moment of the link, $\vec{n}_i$, which is the moment exerted by link *i-1* on link *i*.

2. The negative of the local moment of the next link transformed to the link *i* coordinate system, that is

$$- A_i^{i+1} \vec{n}_{i+1}.$$

3. The moment caused by the local force acting on the link at a distance away from the the origin.

[4]Chapter 11.2 Euler's Equations of Motion for a Rigid Body

$$- \hat{p}_i \times A_i^{i+1} \vec{f}_{i+1}.$$

This is the negative of the cross product of the position vector which points from the *i-1* to the *i* coordinate origin and the local force on the next link transformed to the link *i* coordinate system.

4. The moment caused by the net force acting on the center of mass of the link.

$$-(\hat{p}_i + \hat{s}_i^*) \times \vec{F}_i.$$

This is the negative of the cross product of the vector pointing from the link *i-1* coordinate system origin to the center of mass of link *i* and the net force on link *i*.

By rearranging these components to solve for the local moment we get

$$\vec{n}_i = A_i^{i+1} \vec{n}_{i+1} + \hat{p}_i \times (A_i^{i+1} \vec{f}_{i+1}) + (\hat{p}_i + \hat{s}_i^*) \times \vec{F}_i + \vec{N}_i \qquad (9)$$

$$i = N,...,2,1.$$

where $\vec{n}_{N+1}$ is the external moment exerted on the hand.

We can iteratively solve this equation from the hand back to the base to find the local moments on each link.

Joint Torques

The local moment of link *i* is the moment that the link exerts on joint *i-1*. The component of the local moment that is along the Z_{i-1} axis is the torque exerted on joint *i-1*. The torque required for a joint to compensate for the local moment and friction is given by

$$\tau_i = \vec{n}_i \cdot (A_i^{i-1} \hat{Z}_0) + b_i \dot{\theta}_i \qquad i = N-1,...,1 \qquad (10)$$

where b_i is the friction coefficient of joint *i*. The friction, b_i, in each of the joints is related to the velocity of the joint by some nonlinear function. Since the friction in the joints of the CMU DD arm is very small, we neglect this term in the simulation. [2] [1]

Defining the InvArm Function

We can define the function which evaluates equations 1 through 10 as

$$\underline{\tau} = \text{InvArm}(\underline{\theta}, \underline{\dot{\theta}}, \underline{\ddot{\theta}})$$

where $\underline{\tau} = (\tau_1,\tau_2,...,\tau_N)$, $\underline{\theta} = (\theta_1,\theta_2,...,\theta_N)$, $\underline{\dot{\theta}} = (\dot{\theta}_1,\dot{\theta}_2,...,\dot{\theta}_N)$, and $\underline{\ddot{\theta}} = (\ddot{\theta}_1,\ddot{\theta}_2,...,\ddot{\theta}_N)$. This function call is an actual procedure in the software which implements the algorithms discussed in this paper. The InvArm function will be used later in the *Forward Arm*.

DAC Output

Once we know the torques, $\underline{\tau}$, that we must produce, we have to calculate what motor voltages we must apply in order to generate these torques. In the computer program, the motor voltage calculation is performed separately from the above five steps because when the *Forward Arm* uses the *Inverse Arm* it requires the torques as output. The motor voltages, computed from the torques and velocities, can be used for a feedforward compensation control system. [2]

We assume that the inductance of the motors is negligible so that the equation relating voltage to the armature resistance and motor speed is

$$V_i = R_i I_i + Kt_i \dot{\theta}_i$$

where R_i is the resistance of motor *i*, Kt_i is a back EMF constant for motor *i*, and I_i is the current in motor *i*. The torque that the motor produces is related to the current in the motor and is given by

$$\tau_i = Kt_i I_i.$$

We can rewrite these equations as

$$V_i = R_i \tau_i / Kt_i + Kt_i \dot{\theta}_i. \tag{11}$$

Forward Arm

The purpose of this model is to allow us to simulate the movement of a manipulator. We can specify the voltages applied to each of the motors and calculate the resulting movement of the arm. The *Forward Arm* model algorithm is taken from a paper on manipulator dynamic simulation written by Walker and Orin. [11] We use the third method given in the paper.

The Walker-Orin algorithm is a method for calculating the acceleration of each joint in a manipulator. We use a third order Runge-Kutta integration algorithm to compute the velocities and positions of the joints from the accelerations. We have added a model of the motor dynamics so that motor voltages can be converted to torques. The *Forward Arm* model, which consists of these three parts (calculation of acceleration, integration, and motor dynamics), takes as input a voltage schedule which is a list of input voltages to be applied to the motors of the arm over a period of time. The output of the model is the positions, velocities, and accelerations that the joints of the arm undergo with the specified input.

We will first describe the motor dynamics equations and then describe the Walker-Orin algorithm in detail. This discussion will be completed with a description of the Runge-Kutta algorithm as it applies to this problem.

Motor Dynamics

The motors have characteristics, such as back EMF, which can be modeled as a control system around a Walker-Orin arm model. (see figure 2)

The voltages applied to the terminals of the motors have the back EMFs of the motors, given by $Kt_i \dot{\theta}_i$, subtracted from them. The result is multiplied by Kt_i/R_i to give the torque that is actually generated and applied to the joint of the arm. The inductance of the motor is negligible in most cases, so it is not included in this analysis. The torque, τ_i, in terms of the applied voltage, V_i, and the joint velocity, $\dot{\theta}$, is given by

$$\tau_i = Kt_i (V_i - Kt_i \dot{\theta}_i) / R_i \qquad i = 1,2,\ldots,N. \tag{12}$$

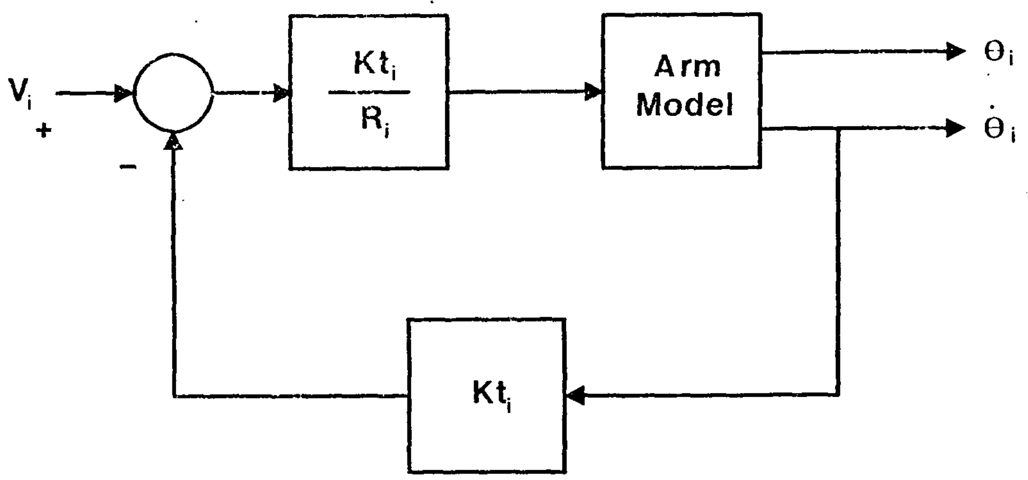


Figure 2: Control System Model of a Motor

The Walker-Orin Simulation Method

The dynamics of any manipulator can be summed up in one general second-order differential equation. [6]

$$\underline{\tau} = H(\underline{\theta})\underline{\ddot{\theta}} + C(\underline{\theta},\underline{\dot{\theta}})\underline{\dot{\theta}} + G(\underline{\theta}) + K_f(\underline{\theta})^T \vec{f}_{hand} + K_n(\underline{\theta})^T \vec{n}_{hand} \tag{13}$$

where $H(\underline{\theta})$ is an $N \times N$ symmetric nonsingular moment of inertia matrix, $C(\underline{\theta}, \underline{\dot{\theta}})$ is an $N \times N$ matrix specifying the centrifugal and coriolis effects, $G(\underline{\theta})$ is a vector of size N specifying the effects of gravity, $K_f(\underline{\theta})$ and $K_n(\underline{\theta})$ are $3 \times N$ Jacobian matrices specifying the torques created at each joint due to external forces and moments exerted on the hand, $\vec{f}_{hand}$ is a spatial vector specifying the external force exerted on the hand, and $\vec{n}_{hand}$ is a vector specifying the moments exerted on the hand about the X, Y, and Z axes.

The purpose of this part of the *Forward Arm* model is to compute the accelerations of each of the joints given the torques applied to the joints and the current state (positions and velocities) of the arm. There are three parts to this computation: computing the bias vector, finding the H matrix, and solving for the joint accelerations. The Walker-Orin paper [11] gives four methods of finding the accelerations. The first and third steps, that is computing the bias vector and solving for the joint accelerations, are common to the first three methods presented in the Walker-Orin paper. The difference between the three methods is in their algorithms for computing the H matrix. The third method given for computing the H matrix is used here for speed.

Computing the Bias Vector

If we let

$$\underline{B} = C(\underline{\theta}, \dot{\underline{\theta}})\dot{\underline{\theta}} + G(\underline{\theta}) + K_f(\underline{\theta})^T \vec{f}_{hand} + K_n(\underline{\theta})^T \vec{n}_{hand} \tag{14}$$

then equation 13 becomes

$$H(\underline{\theta})\ddot{\underline{\theta}} = (\underline{\tau} - \underline{B}). \tag{15}$$

$\underline{B}$ is called a bias vector which corresponds to the torque required to maintain the current state without any acceleration. The bias vector can be computed with the InvArm function by setting $\ddot{\underline{\theta}} = 0$ and calling the routine to calculate the torque. If we knew the H matrix then we could solve equation 15 as a set of N simultaneous equations in N unknowns. The $\ddot{\underline{\theta}}$ would be the unknowns.

Calculating the H matrix

The H matrix represents an effective moment of inertia for the arm. It is a function entirely of the arm position since the velocity effects are accounted for in other terms. The simplest means of calculating the H matrix is to set one element of $\ddot{\underline{\theta}}$ to 1 and all of the rest to 0. We can use the InvArm function with $\dot{\underline{\theta}} = \underline{k} = G(\underline{\theta}) = 0$ to compute the torque vector for that acceleration. The computed torque vector is equal to the column of the H matrix corresponding to the element of $\ddot{\underline{\theta}}$ that was set to a 1 since $\underline{\tau} = H(\underline{\theta})\ddot{\underline{\theta}}$ when $\dot{\underline{\theta}} = G(\underline{\theta}) = \underline{k} = 0$ from equation 15. This is Walker-Orin's method 1 which is simple, but computationally slow.

Method 3 uses a different approach. The same assumption about the acceleration is made, but the calculation of torques proceeds differently. The H matrix is symmetrical so only the diagonal and top half of the off diagonal elements are computed. If $\ddot{\underline{\theta}}$ is set up as before, with the j^{th} element set to 1 and the rest 0, we notice that the manipulator can be viewed as a singly jointed arm with a "hand" that is made up of links *j* through *N* and the base made up of links *0* through *j-1*. The characteristics of the "hand," its mass, center of mass, and moment of inertia, can be calculated iteratively using

$$M_j = M_{j+1} + m_j \qquad j = N\text{-}1,\ldots,2,1 \tag{16}$$

$$\vec{c}_j^{\,*} = \frac{1}{M_j}[\, m_j \hat{s}_j^{\,*} + M_{j+1} A_j^{j+1}(\vec{c}_{j+1}^{\,*} + \hat{p}_{j+1})\,] \qquad j = N\text{-}1,\ldots,2,1 \tag{17}$$

$$\begin{aligned} E_j^* = {} & A_j^{j+1} E_{j+1}^* A_{j+1}^j + M_{j+1}[\,|A_j^{j+1}(\vec{c}_{j+1}^{\,*} + \hat{p}_{j+1}) - \vec{c}_j^{\,*}|^2 I \\ & - (A_j^{j+1}(\vec{c}_{j+1}^{\,*} + \hat{p}_{j+1}) - \vec{c}_j^{\,*})(A_j^{j+1}(\vec{c}_{j+1}^{\,*} + \hat{p}_{j+1}) - \vec{c}_j^{\,*})^T\,] \\ & + J_j^* + m_j[\,|\hat{s}_j^{\,*} - \vec{c}_j^{\,*}|^2 I - (\hat{s}_j^{\,*} - \vec{c}_j^{\,*})(\hat{s}_j^{\,*} - \vec{c}_j^{\,*})^T\,] \qquad j = N\text{-}1,\ldots,2,1 \end{aligned} \tag{18}$$

where m_j is the mass of link *j*, M_j is the mass of the "hand" that is links *j* through *N* inclusive, $\vec{c}_j^{\,*}$ is the composite center of mass of the "hand," E_j^* is the composite moment of inertia of the "hand," and I is the identity matrix. The boundary conditions at link N are

$$M_N = m_N$$

$$\vec{c}_N^{\,*} = \hat{s}_N^{\,*}$$

$$E^*_N = J^*_N$$

The composite mass, M_j, is the sum of the masses of links *j* through *N*. The composite center of mass, $\vec{c}^*_j$, is computed relative to the origin of link *j* in link *j* coordinates. $\vec{c}^*_j$ is a weighted sum of the center of mass of link *j*, $\hat{s}^*_j$, and the center of mass of composite link *j+ 1*, $\vec{c}^*_{j+1}$, divided by the total mass of the composite link *j*, M_j. $\hat{s}^*_j$ is weighted by its mass, m_j and $\vec{c}^*_{j+1}$ is weighted by M_{j+1}. The sum of the position vector, $\hat{p}_{j+1}$, which points from the link *j* origin to the *j + 1* origin and $\vec{c}^*_{j+1}$ is transformed from *j+1* coordinates to *j* coordinates to give $\vec{c}^*_{j+1}$ relative to the *j* coordinate origin in *j* coordinates.

The composite moment of inertia, E^*_j, is the sum of the moment of inertia of link *j*, J^*_j, and the composite moment of inertia of links *j* through *N*, E^*_{j+1}, both moved to the center of mass of composite link *j*. E^*_{j+1} must undergo a coordinate transformation before it is added to J^*_j.

To move a moment of inertia tensor we use a form of the parallel axis theorem which gives the moment of inertia of an object around an arbitrary location when the moment of inertia around the center of mass is known. [10][5]

$$I_0 = I^*_g + M[\vec{R}^2 1 - \vec{R}\vec{R}^T]$$

where I^*_g is the moment of inertia tensor around the center of mass, M is the mass of the object, $\vec{R}$ is the vector pointing from an arbitrary location to the center of mass, and 1 is the identity matrix.

E^*_{j+1} is first transformed to *j* coordinates by pre- and post-multiplication by A^{j+1}_j and its inverse. Then it is moved to $\vec{c}^*_j$. $\vec{R}$ is $A^{j+1}_j(\vec{c}^*_{j+1} + \hat{p}_{j+1}) - \vec{c}^*_j$ for this move which points from the center of mass of composite link *j* to the center of mass of composite link *j+ 1*. J^*_j is moved to the center of mass of composite link j by the same means. $\vec{R}$ is $\hat{s}^*_j - \vec{c}^*_j$ in this case which is a vector from the center of mass of composite link *j* to the center of mass of link *j*.

The net force on the "hand" is the force acting on the center of mass which is the "hand's" mass times its linear acceleration. Since the angular acceleration about a joint axis is assumed to be 1, the linear acceleration is the cross product of the angular acceleration vector (which is just the Z axis of link *j-1* expressed in *j* coordinates) and the vector from the axis to the center of mass of the link. The net force is given by

$$\vec{F}_j = M_j [A^{j-1}_j \hat{Z}_0 \times (\vec{c}^*_j + \hat{p}_j)] \qquad j = N,...,2,1 \qquad (19)$$

The net moment of the "hand" is the moment around the "hand's" center of mass which is the component of the moment of inertia matrix, E^*_j, which is in the direction of the joint axis, or

$$\vec{N}_j = E^*_j A^{j-1}_j \hat{Z}_0 \qquad j = N,...,2,1. \qquad (20)$$

The force that link *j -1* exerts on the hand, called the local force, is equal to the net force since the only force exerted on the "hand" is the local force. The moment exerted on the "hand" by link *j-1*, called the local moment, is the net moment plus the moment produced by a force acting at a distance from the rotational axis. The force is the net force and the distance is the sum of the center of mass vector of link *j*, $\vec{c}^*_j$, and the position vector, $\hat{p}_j$, which points from the link *j -1* to the link *j* coordinate axes.

[5]Chapter 10.5

$$\vec{f}_j = \vec{F}_j \qquad j = N,...,2,1 \qquad (21)$$

$$\vec{n}_j = N_j + (\vec{c}_j^{\,*} + \hat{p}_j) \times \vec{F}_j \qquad j = N,...,2,1 \qquad (22)$$

The torque required in the joint is the (j, j) element of the H matrix which is equal to the component of the local moment in the direction of the link *j-1* Z axis, given by

$$H_{jj} = \vec{n}_j \cdot (A_j^{j-1} \hat{Z}_0) \qquad j = N,...,1. \qquad (23)$$

This gives us the diagonal of the H matrix.

The off diagonal elements can be computed by calculating the torques needed in joints *1* through *j-1* to maintain this static situation. The local forces on each link from *1* to *j-1* are the same as the local force, f_j, on link *j* when transformed to the correct coordinate system. The local moment of link *i* is the sum of the local moment of link *i+1* transformed to *i* coordinates and the moment caused by the local force of link *i+1* acting at a distance $\hat{p}_{i+1}$.

$$\vec{f}_i = A_i^{i+1} \vec{f}_{i+1} \qquad i = j-1,...,1 \qquad (24)$$

$$\vec{n}_i = A_i^{i+1} \vec{n}_{i+1} + \hat{p}_i \times (A_i^{i+1} \vec{f}_{i+1}) \qquad i = j-1,...,1. \qquad (25)$$

The elements of the off diagonals of the H matrix are the components of the local moments which are in the same direction as the Z axes of the previous links.

$$H_{ij} = H_{ji} = \vec{n}_i \cdot (A_i^{i-1} \hat{Z}_0) \qquad i = j-1,...,2,1 \qquad j = N,...,2,1. \qquad (26)$$

Once we have the H matrix we can calculate the acceleration vector for the given torque, $\underline{\tau}$, and computed bias vector, $\underline{B}$ using equation 15. Since the H matrix is symmetric an algorithm tailored to such matrices is used to solve the simultaneous equations. [9]

Defining the ForArm Function

Like the InvArm function we can define a ForArm function which returns the acceleration of the arm joints given the positions, velocities, and joint torques. This function is useful in explaining the Runge-Kutta integration and corresponds to a real function in the modeling software. We define ForArm as

$$\ddot{\underline{\theta}} = \text{ForArm} (\underline{\theta}, \dot{\underline{\theta}}, \underline{\tau})$$

where $\underline{\theta}$, $\dot{\underline{\theta}}$, $\ddot{\underline{\theta}}$, and $\underline{\tau}$ are defined in section .

Runge-Kutta Integration

The ability to calculate the acceleration for a given set of conditions allows us to compute future positions and velocities of the joints using a Runge-Kutta integration algorithm. A third order Runge-Kutta provides enough accuracy without sacrificing too much speed.

If we let

$$F(\underline{\theta}, \dot{\underline{\theta}}, \underline{V}) = \text{ForArm} (\underline{\theta}, \dot{\underline{\theta}}, [\, \underline{Kt} (\underline{V} - \underline{Kt}\, \dot{\underline{\theta}}) / \underline{R} \,])$$

where $\underline{V}$ is the arm vector of applied motor voltages (see section) then the first order differential equations that we are trying to solve are

$$\frac{d(\dot{\underline{\theta}})}{dt} = F(\underline{\theta}, \dot{\underline{\theta}}, \underline{V})$$

and

$$\frac{d(\underline{\theta})}{dt} = \dot{\underline{\theta}}.$$

To use the third order Runge-Kutta method, we calculate two sets of three coefficients which represent the current values and estimated future values of $\dot{\underline{\theta}}$ and $\ddot{\underline{\theta}}$. The new values computed for $\underline{\theta}$ and $\dot{\underline{\theta}}$ are the old values plus a time increment multiplied by a weighted sum of the current and estimated future values of $\dot{\underline{\theta}}$ and $\ddot{\underline{\theta}}$. [3] The coefficients to be calculated for each new set of values for $\underline{\theta}$ and $\dot{\underline{\theta}}$ are

$$\vec{c}_1 = \dot{\underline{\theta}}$$

$$\underline{k}_1 = F(\underline{\theta}, \dot{\underline{\theta}}, \underline{V})$$

$$\underline{c}_2 = \dot{\underline{\theta}} + \frac{h}{2}\underline{k}_1$$

$$\underline{k}_2 = F(\underline{\theta} + \frac{h}{2}\underline{c}_1, \underline{c}_2, \underline{V})$$

$$\underline{c}_3 = \dot{\underline{\theta}} + 2h\underline{k}_2 - h\underline{c}_1$$

$$\underline{k}_3 = F(\underline{\theta} + 2h\underline{c}_2 - h\underline{c}_1, \underline{c}_3, \underline{V})$$

where h is a time step. The determination of the correct size for the time step h is a difficult procedure. There are algorithms for adjusting the stepsize according to the change in $\dot{\underline{\theta}}$ and $\ddot{\underline{\theta}}$, but these are not currently implemented. A fixed stepsize of 1 millisecond which is approximately $1/15^{th}$ of the smallest mechanical time constant in the arm is currently being used.

The iterative equations for calculating the weighted sum and obtaining new values for $\underline{\theta}$ and $\underline{\theta}$ are:

$$\underline{\theta}^{n+1} = \underline{\theta}^{n} + \frac{h}{6}(\underline{c}_1 + 4\,\underline{c}_2 + \underline{c}_3)$$

$$\dot{\underline{\theta}}^{n+1} = \dot{\underline{\theta}}^{n} + \frac{h}{6}(\underline{k}_1 + 4\,\underline{k}_2 + \underline{k}_3)$$

Programs Available

We have implemented the model of arm dynamics which has been described. The software resides on the CMU-750R VAX in the /usrr0/nms/sim directory. Currently, the software is in the form of an executable testbed called "sim." The sim program is CI based with full help functions. The sim program can do the *Inverse Arm* on single or multiple sets of data. It can do the *Forward Arm* on single sets of data or do a simulation sequence.

The sim system is run by typing /usr/nms/sim/sim. The user is then at the CI user interface level. There are several variables which the user can set to different values. They are

gravity	The value of the gravitational acceleration. Defaults to -9.80621 meters/second^2
DACTimePeriod	The time between changes in the reference input to the control system of the arm

The following variables are vectors which have one component for each joint in the arm being modeled.

theta	The joint angle of each joint in the arm starting from the base.
omega	The joint angle velocities. This is the first time derivative of theta.
alpha	The joint accelerations. This is the output of the *Forward Arm* model.
torque	The torque each motor is producing. This is calculated from the voltage.
voltage	The voltage applied or that should be applied to each joint to satisfy the other conditions
DACOut	The Digital to Analog Converter output voltage which is connected to the reference input of the control system of the arm

Any of these variable can be set by saying: variable = value1 value2 ... valueN, where N is the number of joints in the arm. The torque and voltage variables are tied together so that setting one recalculates the other.

The commands allow the user to apply the *Inverse Arm* or *Forward Arm* models to different sets of data. Possible commands are

inverse_arm	Runs the *Inverse Arm* model on the theta, omega, and alpha variables. The result is put in the torque and voltage variables.
forward_arm	Runs the *Forward Arm* model of the theta, omega, and voltage variables. The result is put in the alpha variable. This command does not perform Runge-Kutta integration.

simulate	Simulates the motion of an arm. Prompts the user as to whether it should start a new simulation, set the theta and omega variables to zero before doing a simulation, and whether a voltage schedule file is used. If a voltage schedule file is used the user is prompted for its name. If a schedule file is not used the voltage is assumed to be held constant at the value given by the voltage variable and the user is prompted for the number of timesteps over which the simulation is to occur.
plot_simulation	Plots the simulation on the dover. Generates a poof file (see MAN POOF) called #doverplot.poof and a press file called #poof.press.
print_simulation	Prints the simulation on the user's screen if no file is currently open. Prompts the user for start, finish, and step values.
write_graphic_file	Writes a file of simulation results which is suitable for input to the graphics simulation display package on the PERQ.
open_output_file	Opens a file for writing with print_simulation. Prompts user for filename.
close_output_file	Closes the open file used for simulation output. Further output is sent to the terminal after the file is closed.
database_read	Reads a database file for an arm which contains mass, center of mass, moment of inertia, Denavit-Hartenburg parameters, and other related factors which characterize an arm. This command is done implicitly for the CMU DD Arm when the sim program is started up.

Further information about the internal workings of the system is in the program documentation.

General Equation of Arm Dynamics

$$\underline{\tau} = H(\underline{\theta})\ddot{\underline{\theta}} + C(\underline{\theta},\dot{\underline{\theta}})\dot{\underline{\theta}} + G(\underline{\theta}) + K_f(\underline{\theta})^T \vec{f}_{hand} + K_n(\underline{\theta})^T \vec{n}_{hand} \quad (13)$$

$$\underline{B} = C(\underline{\theta},\dot{\underline{\theta}})\dot{\underline{\theta}} + G(\underline{\theta}) + K_f(\underline{\theta})^T \vec{f}_{hand} + K_n(\underline{\theta})^T \vec{n}_{hand} \quad (14)$$

$$H(\underline{\theta})\ddot{\underline{\theta}} = (\underline{\tau} - \underline{B}) \quad (15)$$

Where:

$H(\underline{\theta})$	≙ An N×N symmetric nonsingular moment of inertia matrix.
$C(\underline{\theta},\dot{\underline{\theta}})$	≙ An N×N matrix specifying the centrifugal and coriolis effects.
$G(\underline{\theta})$	≙ An arm vector specifying the effects due to gravity.
$K_f(\underline{\theta})$	≙ A 3×N Jacobian matrix specifying the torques created at each joint due to an external force exerted on the hand.
$K_n(\underline{\theta})$	≙ A 3×N Jacobian matrix specifying the torques created at each joint due to an external moment exerted on the hand.
$\vec{f}_{hand}$	≙ A spatial vector specifing the force exerted on the hand externally.
$\vec{n}_{hand}$	≙ A vector specifying the external moments around the X, Y, and Z axes.
$\underline{B}$	≙ A bias vector which can be computed by equations 1 through 10 with $\ddot{\underline{\theta}}$ set equal to 0.

Equations for Evaluating the Dynamic Motion of a Manipulator

$$\vec{\omega}_{i+1} = A^{i}_{i+1} (\vec{\omega}_i + \hat{Z}_0 \dot{\theta}_{i+1}) \qquad i = 0,1,\ldots,N\text{-}1 \quad (1)$$

$$\dot{\vec{\omega}}_{i+1} = A^{i}_{i+1} (\dot{\vec{\omega}}_i + \hat{Z}_0 \ddot{\theta}_{i+1} + \vec{\omega}_i \times \hat{Z}_0 \dot{\theta}_{i+1}) \qquad i = 0,1,\ldots,N\text{-}1 \quad (2)$$

$$\vec{v}_{i+1} = \vec{\omega}_{i+1} \times \hat{p}_{i+1} + A^{i}_{i+1} \vec{v}_i \qquad i = 0,1,\ldots,N\text{-}1 \quad (3)$$

$$\dot{\vec{v}}_{i+1} = \dot{\vec{\omega}}_{i+1} \times \hat{p}_{i+1} + \vec{\omega}_{i+1} \times (\vec{\omega}_{i+1} \times \hat{p}_{i+1}) + A^{i}_{i+1} \dot{\vec{v}}_i \qquad i = 0,1,\ldots,N\text{-}1 \quad (4)$$

$$\dot{\vec{v}}^{*}_i = \dot{\vec{\omega}}_i \times \hat{s}^{*}_i + \vec{\omega}_i \times (\vec{\omega}_i \times \hat{s}^{*}_i) + \dot{\vec{v}}_i \qquad i = 1,2,\ldots,N \quad (5)$$

$$\vec{F}_i = m_i \dot{\vec{v}}^{*}_i \qquad i = 1,2,\ldots,N \quad (6)$$

$$\vec{N}_i = J_i \dot{\vec{\omega}}_i + \vec{\omega}_i \times (J_i \vec{\omega}_i) \qquad i = 1,2,\ldots,N \quad (7)$$

$$\vec{f}_i = A^{i+1}_{i} \vec{f}_{i+1} + \vec{F}_i \qquad i = N,\ldots,2,1 \quad (8)$$

$$\vec{n}_i = A^{i+1}_{i} \vec{n}_{i+1} + \hat{p}_i \times (A^{i+1}_{i} \vec{f}_{i+1}) + (\hat{p}_i + \hat{s}^{*}_i) \times \vec{F}_i + \vec{N}_i \qquad i = N,\ldots,2,1 \quad (9)$$

$$\tau_i = \vec{n}_i \cdot (A^{i-1}_{i} \hat{Z}_0) + b_i \dot{\theta}_i \qquad i = N\text{-}1,\ldots,1 \quad (10)$$

Where:

$\vec{\omega}_i , \dot{\vec{\omega}}_i$	$\triangleq$ The angular velocity and acceleration of the link *i* coordinate system.
$\vec{v}_i , \dot{\vec{v}}_i$	$\triangleq$ The linear velocity and acceleration of the link *i* coordinate system.
$\dot{\vec{v}}^{*}_i$	$\triangleq$ The linear acceleration of the center of mass of link *i*.
$\vec{F}_i , \vec{N}_i$	$\triangleq$ The net force and moment exerted on link *i*.
$\vec{f}_i , \vec{n}_i$	$\triangleq$ The force and moment exerted on link *i* by link *i - 1*.
τ_i	$\triangleq$ The input torque for joint *i*.
b_i	$\triangleq$ A number representing viscous damping or friction.
$\theta_i , \dot{\theta}_i , \ddot{\theta}_i$	$\triangleq$ The angle, and its derivatives, through which joint *i* is turned.
$\hat{p}_i$	$\triangleq$ Vector from link *i - 1* origin to link *i* coordinate system origin.
$\hat{s}^{*}_i$	$\triangleq$ Vector from link *i* coordinate system origin to center of mass of link *i*.
J_i	$\triangleq$ Moment of inertia matrix of link *i*.
m_i	$\triangleq$ Mass of link *i*.

Given:

$$A_i^{i+1} = \begin{bmatrix} \cos\theta_{i+1} & -\cos\alpha_{i+1}\sin\theta_{i+1} & \sin\alpha_{i+1}\sin\theta_{i+1} \\ \sin\theta_{i+1} & \cos\alpha_{i+1}\cos\theta_{i+1} & -\sin\alpha_{i+1}\cos\theta_{i+1} \\ 0 & \sin\alpha_{i+1} & \cos\alpha_{i+1} \end{bmatrix}$$

A_{N+1}^{N} = I. The identity matrix
The N + 1th coordinate system is the hand coordinate system which is identical to the link N coordinate system.

$$A_{i+1}^{i} = (A_i^{i+1})^T = (A_i^{i+1})^{-1}$$

$$\hat{p}_i = \begin{bmatrix} a_i \\ r_i\sin(\alpha_i) \\ r_i\cos(\alpha_i) \end{bmatrix}$$

$$\hat{Z}_0 = [0\,0\,1]^T$$

$$\vec{\omega}_0, \dot{\vec{\omega}}_0, \vec{v}_0 = [0\,0\,0]^T$$

$$\dot{\vec{v}}_0 = [0\,0\,g]^T$$

$g = 9.80621$ meters/sec^2 if the Z-axis of the link 0 coordinate frame is pointing vertically up.

$g = -9.80621$ meters/sec^2 if the Z-axis of the link 0 coordinate frame is pointing vertically down.

$\vec{f}_{N+1}$ ≙ The external force on the hand.

$\vec{n}_{N+1}$ ≙ The external moment on the hand.

Equations for Calculation of H Matrix in Forward Arm Problem

$$M_j = M_{j+1} + m_j \qquad j = N\text{-}1,\ldots,2,1 \quad (16)$$

$$\vec{c}_j^{\,*} = \frac{1}{M_j}[\, m_j \hat{s}_j^{\,*} + M_{j+1} A_j^{j+1}(\vec{c}_{j+1}^{\,*} + \hat{p}_{j+1})\,] \qquad j = N\text{-}1,\ldots,2,1 \quad (17)$$

$$E_j^* = A_j^{j+1} E_{j+1}^* A_{j+1}^j + M_{j+1}[\, | A_j^{j+1}(\vec{c}_{j+1}^{\,*} + \hat{p}_{j+1}) - \vec{c}_j^{\,*} |^2 I$$
$$- (A_j^{j+1}(\vec{c}_{j+1}^{\,*} + \hat{p}_{j+1}) - \vec{c}_j^{\,*})(A_j^{j+1}(\vec{c}_{j+1}^{\,*} + \hat{p}_{j+1}) - \vec{c}_j^{\,*})^T\,]$$
$$+ J_j^* + m_j[\, | \hat{s}_j^{\,*} - \vec{c}_j^{\,*} |^2 I - (\hat{s}_j^{\,*} - \vec{c}_j^{\,*})(\hat{s}_j^{\,*} - \vec{c}_j^{\,*})^T\,] \qquad j = N\text{-}1,\ldots,2,1 \quad (18)$$

$$\vec{F}_j = M_j\,[\, A_j^{j-1}\hat{Z}_0 \times (\vec{c}_j^{\,*} + \hat{p}_j)] \qquad j = N,\ldots,2,1 \quad (19)$$

$$\vec{N}_j = E_j^* A_j^{j-1}\hat{Z}_0 \qquad j = N,\ldots,2,1 \quad (20)$$

$$\vec{f}_j = \vec{F}_j \qquad j = N\text{-}1,\ldots,2,1 \quad (21)$$

$$\vec{n}_j = N_j + (\vec{c}_j^{\,*} + \hat{p}_j) \times \vec{F}_j \qquad j = N,\ldots,2,1 \quad (22)$$

$$H_{jj} = \vec{n}_j \cdot (A_j^{j-1}\hat{Z}_0) \qquad j = N\text{-}1,\ldots,1 \quad (23)$$

$$\vec{f}_i = A_i^{i+1}\vec{f}_{i+1} \qquad i = j\text{-}1,\ldots,1 \quad (24)$$

$$\vec{n}_i = A_i^{i+1}\vec{n}_{i+1} + \hat{p}_i \times (A_i^{i+1}\vec{f}_{i+1}) \qquad i = j\text{-}1,\ldots,1 \quad (25)$$

$$H_{ij} = H_{ji} = \vec{n}_i \cdot (A_i^{i-1}\hat{Z}_0) \qquad i = j\text{-}1,\ldots,2,1 \qquad j = N,\ldots,2,1 \quad (26)$$

where

M_j	$\triangleq$ The cumulative mass of links j through N with $M_N = m_N$.
$\vec{c}_j^{\,*}$	$\triangleq$ The center of mass of composite link *j* with $\vec{c}_N^{\,*} = \hat{s}_N^{\,*}$
E_j^*	$\triangleq$ The moment of inertia of composite link j around its center of mass with $E_N^* = J_N^*$
H_{ij}	$\triangleq$ The N X N matrix relating angular accelerations to joint torques.

References

[1] Haruhiko Asada and Takeo Kanade.
Design of Direct-Drive Mechanical Arms.
ASME Journal on Dynamics Systems, Measurement, and Control, 1982.

[2] Haruhiko Asada, Takeo Kanade, and Ichiro Takeyama.
Control of a Direct-Drive Arm.
1982.
CMU Robotics Institute internal document CMU-RI-TR-82-4.

[3] B. Carnahan, H. A. Luther, and J. O. Wilkes.
Applied Numerical Methods.
John Wiley and Sons, Inc., 1969, pages 363.

[4] DD Arm Project Staff.
Definition of Link Coordinate Systems for the CMU DD Arm.
1982.
Robotics Institute white paper RVL010.

[5] J. Denavit and R. S. Hartenburg.
A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices.
ASME Journal of Applied Mechanics :215-221, June, 1955.

[6] John M. Hollerbach.
A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity.
IEEE Transactions on Systems, Man, and Cybernetics SMC-10(11):730-736, November, 1980.

[7] Takeo Kanade.
Transforming Inertia Matrix.
1982.
Robotics Institute white paper RVL006.

[8] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul.
On-Line Computational Scheme for Mechanical Manipulators.
Journal of Dynamic Systems, Measurement, and Control 102:69-76, June, 1980.

[9] J. C. Nash.
A Halstead Press Book: Compact Numerical Methods for Computers: Linear Algebra and Function Minimization.
John Wiley and Sons, Inc., New York, 1979.

[10] Keith R. Symon.
Mechanics Third Edition.
Addison-Wesley Publishing Company, Reading, Massachusetts, 1971.

[11] M. W. Walker and D. E. Orin.
Efficient Dynamic Computer Simulation of Robotic Mechanisms.
ASME Journal on Dynamics Systems. Measurement, and Control, September, 1982.