SRC TR 85-12

DYNAMAN
A Tool for Manipulator Design and Analysis

by

N. Sreenath and P.S. Krishnaprasad

TR-85-12

DYNAMAN

A tool for manipulator design and analysis

by

N. Sreenath and P.S. Krishnaprasad

TR - 85 - 12

DYNAMAN

A tool for manipulator design and analysis

N.Sreenath

&

P.S.Krishnaprasad

Systems Research Center

&

Dept. of Electrical Engineering

University of Maryland, College Park, MD-20742

This work is supported in part by the National Science Foundation under grant OIR-8500108 to the Systems Research Center and by the Minta Martin Fund for Aeronautical Research.

Table of Contents

Abstract 1

1 . Introduction 1

2 . Formulation 4

3 . Program description 13

4 . Validation 15

5 . Future work 16

6 . Summary 17

7 . Acknowledgement 18

Appendix A 19

Appendix B 24

Appendix C 25

List of figures

Page No.

(i) N-link manipulator schematic 4

(ii) Denavit-Hartenberg parameters 4

(iii) Two-link manipulator with revolute joints 24

(iv) Two-link manipulator - one revolute and one prismatic joint 25

DYNAMAN: A Tool for Manipulator Design and Analysis

Abstract:

This report describes a method to formulate the dynamic equations of a robot manipulator with N links and with either revolute or prismatic joints using a Newton-Euler formulation. A general-purpose tool for analysis and design of robot manipulators and related control problems is being developed in the form of a software package DYNAMAN using this method. At present, DYNAMAN has the capability to do the following:

(1) generate dynamic models of a robot manipulator given the number of links N and the types (revolute/prismatic) of the manipulator joints.

(2) generate automatically upon request the requisite FORTRAN code for numerical simulation of the dynamic model generated.

(3) generate symbolically the manipulator Jacobian.

The package is written in MACSYMA as implemented for operation under the Berkeley 4.2BSD Unix operating system for the Vax. This report describes in brief some of the main features of the DYNAMAN package. Detailed documentation is provided within the body of the program.

1. Introduction:

A careful study of the dynamics of a robot manipulator is a necessary first step in designing the controls required in order to accomplish any given manipulator task. Such a study is also essential for the mechanical design of prototype arms and design of the joints for proper structural stiffness.

The formulation of the dynamic equations by hand is a very tedious process, and many researchers have taken up the possibility of computer-aided methods to generate these equations. These efforts have led to a variety of simulation tools. We mention here a few such tools:

Among the programs using numerical methods to generate the equations :

NBOD - A program written exclusively in FORTRAN [1]. This program is mainly used to simulate the dynamics of a multi - rigid body system, connected in the form of a tree structure [2].

Among those using symbolic methods to generate the equations :

(i) TOAD (Tele Operator Arm Design) : This program is written in PL1/FORMAC for revolute and prismatic joints [3].

(ii) OSSAM (OHIO State Symbolic and Algebraic Manipulator) : It is written in LISP and was designed to reduce the use of memories. It can be implemented on a small computer such as PDP-11 [4].

(iii) EDYLMA : (Equation Dynamiques Litterales d' un Manipulateur Artticule') Written for rotating joints only, in PL1/FORMAC [5, 6].

(iv) EGAM : (Equation Generation of Articulated Mechanism) Written in PL1/FORMAC for both revolute and prismatic joints. An evolution of EDYLMA [5, 6].

(v) DYMIR : Written in REDUCE, it can take into account revolute, as well as, prismatic joints [7]. It also can take into account elasticity and transmission losses at the joints.

Symbolic programs in general are more efficient in terms of running time as compared to the programs utilizing numerical models. Furthermore symbolic programs have the ability to compute "sensitivity parameters" like the influence of mass, length, or inertia of a link of the model. This would be very useful for design purposes. A symbolic tool also allows us to simulate some special control techniques that are impossible to realize using numerical methods [7].

We use here a Symbolic method for the generation of equations of motion for a N-link robot-manipulator.

In general the derivation of equations of motion for an N-link manipulator is done by using either a Lagrangian formulation or a Newton-Euler formulation. Here, we use a Newton-Euler approach, which is closely related to the *method of nested bodies* [8], which has played a useful role in spacecraft dynamics. The formulation is implemented symbolically using VAXIMA (MACSYMA as implemented on VAX 11/785) in the package DYNAMAN (for DYNamics and Analysis of MANipulators). DYNAMAN automatically generates a FORTRAN listing, for a direct numerical simulation of the dynamic equations generated. It also includes a TUTORIAL input program for feeding in the data. A HOWTORUN file exists for a novice user.

In the next section we describe the formulation of equations of motion in detail. Section 3 describes the program input/output format for DYNAMAN. Section 4 lists two examples which were used to validate our program. In section 5 we discuss our plans for future enhancements of the package. The notation used is described in Appendix A. The data files of the two examples used for validating the software, along with a brief description of the problems, are listed in Appendix B and C respectively.

2. Formulation:

In this section we derive the equations of motion of a N - link manipulator using a Newton-Euler method known as the *nested body approach*. It is originally due to Velman [8].

We consider here a manipulator consisting of N links and N joints. The links are assumed to be rigid bodies and the joints have one d.o.f (degree of freedom) each. The joints are either revolute or prismatic in nature. The schematic of a general N - link manipulator is as given in figure(i). The manipulator base is fixed at $\mathbf{O}_0$, the origin of the inertial co-ordinate system. The origin of the i'th co-ordinate system $\mathbf{O}_i$ is fixed at joint $i+1$. The i'th co-ordinate system is assumed to be fixed with respect to link $i-1$. "Joint $N+1$" is any fixed point on link N.

We use Denavit-Hartenberg [9] parameters to describe the manipulator kinematics. Figure (ii) illustrates in detail the convention of Denhavit-Hartenberg parameters. We give here a brief description of the notation used, and of the parameters.

The rotation or the translation of link i with respect to link i-1 is assumed to be along axis Z_i, i.e., Z_i is the degree of freedom axis. The axis X_i is defined as the axis which is normal to both Z_i and Z_{i-1} axes. Y_i is the axis which is normal to both X_i and Z_i axes, and the direction selected according to the right-hand rule. X_1 is assigned arbitrarily. We now define the D-H parameters for the i'th link.

r_i	The perpendicular distance between $\mathbf{O}_i$ on the Z_i axis and the $X_{i-1} - Y_{i-1}$ plane.
a_i	The perpendicular distance between $\mathbf{O}_i$ and the Z_{i-1} axis.
α_i	Angle between Z_i, and the line parallel to Z_i passing through $\mathbf{O}_i$.
θ_i	Angle between X_{i-1} and the line formed by the projection of X_i on the $X_{i-1} - Y_{i-1}$ plane.

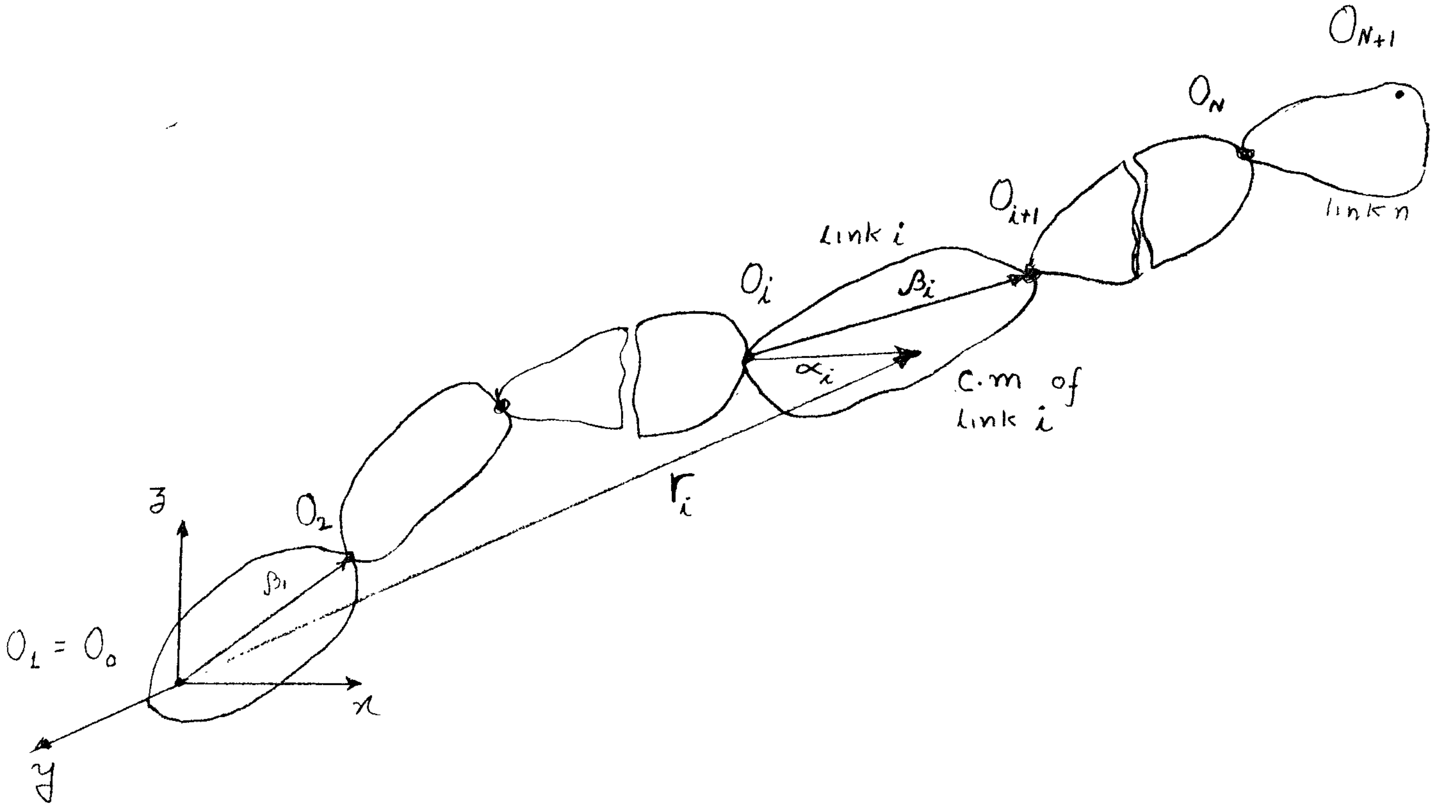

Fig.(1) N - link manipulator schematic

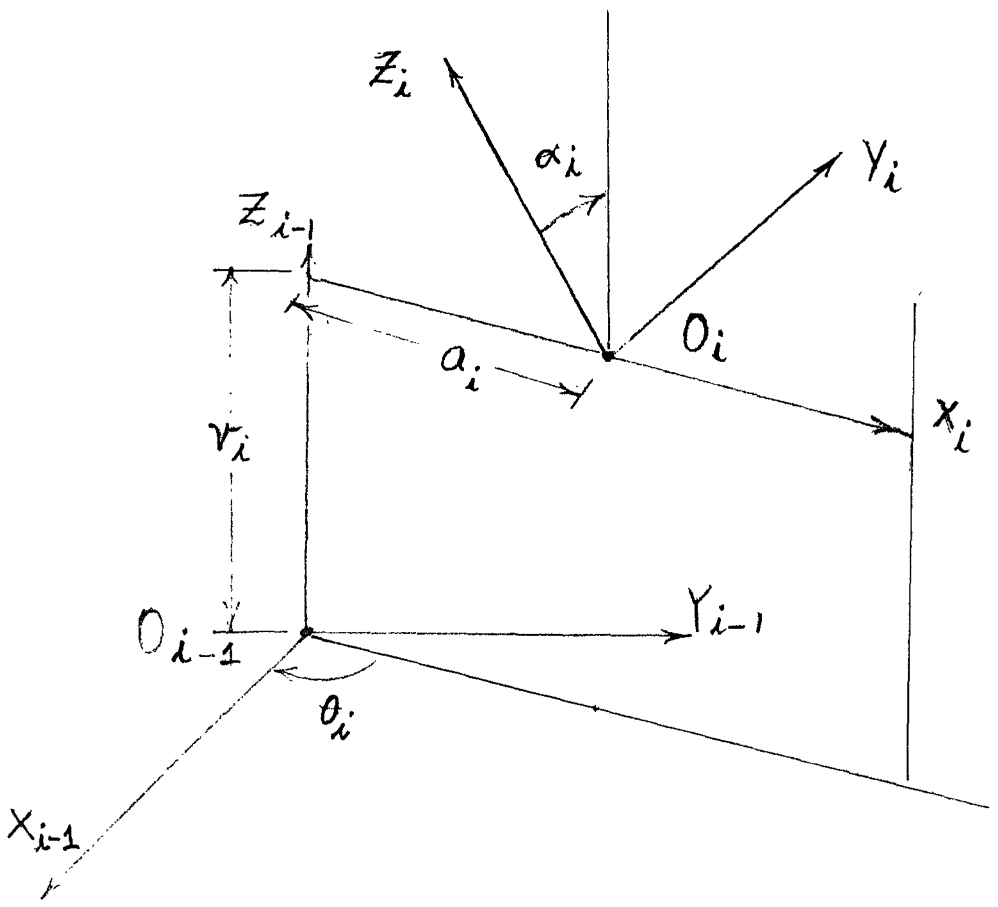

Fig.(11) Denavit-Hartenberg parameters

Referring to figure(i) we define the transformation matrix $\mathbf{A}_i$ from i'th to the $(i\text{-}1)$'th co-ordinate system to be

$$A_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) \end{bmatrix}, \qquad i = 1,\dots,N.$$

This is the "rotational part" of the change of co-ordinates. The transformation matrix $\mathbf{ACB}_i$ from i'th to the inertial co-ordinate system is given by

$$\mathbf{ACB}_i = \prod_{j=1}^{j=i} \mathbf{A}_j \,, \qquad i = 1,\dots,N.$$

If the i'th joint is revolute then,

$$\beta_i = \mathbf{ACB}_i \cdot \begin{bmatrix} a_i \\ 0 \\ r_i \end{bmatrix},$$

$$\dot{\beta}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\ddot{\beta}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

If the i 'th joint is prismatic then,

$$\beta_i = \mathbf{ACB}_i \ . \begin{bmatrix} a_i \, cos\,(\theta_i) \\ a_i \sin(\theta_i) \\ r_i \end{bmatrix} ,$$

$$\dot{\beta}_i = \mathbf{ACB}_i \ . \begin{bmatrix} 0 \\ 0 \\ \dot{r}_i \end{bmatrix} ,$$

$$\ddot{\beta}_i = \mathbf{ACB}_i \ . \begin{bmatrix} 0 \\ 0 \\ \ddot{r}_i \end{bmatrix} .$$

Relative angular velocity in the inertial co-ordinate system $\tilde{\omega}_i$ is given by the following formulas ;

for a revolute joint , if we let,

$$thd_i = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_i \end{bmatrix} ,$$

then $\tilde{\omega}_i = \mathbf{ACB}_i \ . \ thd_i$;

for a prismatic joint we have,

$$\tilde{\omega}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} .$$

Now the inertial angular velocity of the i'th link is given by,

$$\omega_i = \sum_{j=1}^{j=i} \tilde{\omega}_j \quad ,$$

Since αoi_i (see appendix A) is fixed in the i'th co-ordinate system we have

$$\alpha o_i = \mathbf{ACB}_i \, . \, \alpha oi_i \quad ,$$

$$\alpha_i = \beta_i + \alpha o_i \quad ,$$

$$\dot{\alpha}_i = \dot{\beta}_i \quad ,$$

$$\ddot{\alpha}_i = \ddot{\beta}_i \quad .$$

From fig(i) we have, the position vector of link i center of mass (c.m.) Γ_i is given by

$$\Gamma_i = \Gamma_{i-1} + \beta_{i-1} - \alpha_{i-1} + \alpha_i \quad . \tag{1}$$

Inertial velocity of the link i , $\dot{\Gamma}_i$ is given by :

$$\dot{\Gamma}_i = \dot{\Gamma}_{i-1} + \omega_{i-1} \times (\beta_{i-1} - \alpha_{i-1}) + \omega_i \times \alpha_i + \dot{\alpha}_i \quad . \tag{2}$$

Inertial accleration of the link i , $\ddot{\Gamma}_i$ is given by ,

$$\ddot{\Gamma}_i = \ddot{\Gamma}_{i-1} + \omega_{i-1} \times \omega_{i-1} \times (\beta_{i-1} - \alpha_{i-1}) + \omega_i \times \omega_i \times \alpha_i + \dot{\omega}_{i-1} \times (\beta_{i-1} - \alpha_{i-1}) \quad (3)$$

$$+ \dot{\omega}_{i-1} \times \alpha_{i-1} + 2\, \omega_i \times \dot{\alpha}_i + \ddot{\alpha}_i \quad .$$

The above recursion relations are most useful in simplifying the force balance and the torque balance equations.

Transform all forces & torques to the inertial co-ordinate system

$$\mathbf{F}_i^{ext} = \mathbf{ACB}_i \ . \ \mathbf{F}_i^{ext_i} \ ,$$

$$\mathbf{T}_i^{ext} = \mathbf{ACB}_i \ . \ \mathbf{T}_i^{ext_i} \ ,$$

$$\mathbf{F}_i^{motor} = \mathbf{ACB}_i \ . \ \mathbf{F}_i^{motor_i} \ ,$$

$$\mathbf{T}_i^{motor} = \mathbf{ACB}_i \ . \ \mathbf{T}_i^{motor_i} \ .$$

Definition: Nest i is defined as consisting of the set of links numbered from i to N.

Linear momentum $\mathbf{G}_i$ of link i is given by

$$\mathbf{G}_i = m_i \dot{\mathbf{r}}_i \ .$$

Linear momentum of nest i is

$$\sum_{j=i}^{N} \mathbf{G}_i = \sum_{j=i}^{N} m_j \dot{\mathbf{r}}_j \ .$$

Differentiating the above expression with respect to time we have the rate of change of linear momentum in nest i as,

$$\sum_{j=i}^{N} \dot{\mathbf{G}}_i = \sum_{j=i}^{N} m_j \ddot{\mathbf{r}}_j \ . \qquad (4)$$

Also we know that the total force acting on nest i is

$$\mathbf{F}_i^{total} = \mathbf{F}_i^c + \mathbf{F}_i^{motor} + \sum_{j=i}^{N} \mathbf{F}_j^{ext} + \sum_{j=i}^{N} m_j \ g \ . \qquad (5)$$

From Newton's laws we have the force balance equations :

rate of change of linear momentum of nest i = total force acting on nest i

$$\sum_{j=i}^{N} \dot{\mathbf{G}}_j = \mathbf{F}_i^{total} \; . \tag{6}$$

Substituting for $\dot{\mathbf{G}}_j$ from equation (4) and for $\mathbf{F}_i^{total}$ from equation (5) we have

$$\sum_{j=i}^{N} m_j \ddot{\Gamma}_j = \mathbf{F}_i^{c} + \mathbf{F}_i^{motor} + \sum_{j=i}^{N} \mathbf{F}_j^{ext} + \sum_{j=i}^{N} m_j \, g \; . \tag{7}$$

The angular momentum $\mathbf{L}_i$ of link i about the origin $\mathbf{O}_0$ is

$$\mathbf{L}_i = \phi_i \, . \, \omega_i + m_i \Gamma_i \times \dot{\Gamma}_i \; .$$

Rate of change of angular momentum of nest i is given by,

$$\dot{\mathbf{L}}_i^{total} = \sum_{j=i}^{N} \dot{\mathbf{L}}_i = \sum_{j=i}^{N} (\phi_j \, . \, \dot{\omega}_j + \omega_j \times \phi_j \, . \, \omega_j + m_j \Gamma_j \times \ddot{\Gamma}_j) \; . \tag{8}$$

The total torque acting on nest i is

$$\mathbf{T}_i^{total} = \sum_{j=i}^{N} \Gamma_j \times (m_j \, g + \mathbf{F}_i^{ext}) + (\Gamma_i - \alpha_i) \times (\mathbf{F}_i^{c} + \mathbf{F}_i^{motor})$$

$$+ \mathbf{T}_i^{c} + \mathbf{T}_i^{motor} + \sum_{j=i}^{N} \mathbf{T}_j^{ext} \; . \tag{9}$$

But from the equation (7) we have,

$$\mathbf{F}_i^{c} + \mathbf{F}_i^{motor} = \sum_{j=i}^{N} m_j \ddot{\Gamma}_j - \sum_{j=i}^{N} \mathbf{F}_j^{ext} - \sum_{j=i}^{N} m_j \, g \; . \tag{10}$$

Substituting in the equation (10) we have,

$$\mathbf{T}_i^{total} = \sum_{j=i}^{N} m_j (\Gamma_i - \alpha_i) \times \ddot{\Gamma}_j + \sum_{j=i}^{N} (\Gamma_j - \Gamma_i + \alpha_i) \times (m_j g + \mathbf{F}_j^{ext})$$

$$+ \mathbf{T}_i^c + \mathbf{T}_i^{motor} + \mathbf{T}_i^{ext} \ . \qquad (11)$$

We now have the torque balance equations from the Euler equations,

rate of change of angular momentum of nest i = total torque acting on nest i ,

Using this and substituting for $\dot{\mathbf{L}}_i^{total}$ from equation (8), and for $\mathbf{T}_i^{total}$ from equation (11) and simplifying, we get

$$\sum_{j=i}^{N} (\phi_j \ . \dot{\omega}_j + m_j (\Gamma_j - \Gamma_i + \alpha_i) \times \ddot{\Gamma}_j) = - \sum_{j=i}^{N} \omega_i \times \phi_i \ . \omega_i + \sum_{j=i}^{N} m_j (\Gamma_j - \Gamma_i + \alpha_i) \times (g + \mathbf{F}_j^{ext})$$

$$+ (\Gamma_i - \alpha_i) \times \mathbf{F}_i^{motor} + \mathbf{T}_i^c + \mathbf{T}_i^{motor} + \mathbf{T}_i^{ext} (12) \ .$$

where $\ddot{\Gamma}_i$ is as derived before.

We collect together all the dynamic equations in a box.

if joint i is revolute,

$$\sum_{j=i}^{N} (\phi_j \ . \dot{\omega}_j + m_j (\Gamma_j - \Gamma_i + \alpha_i) \times \ddot{\Gamma}_j) = - \sum_{j=i}^{N} \omega_i \times \phi_i \ . \omega_i + \sum_{j=i}^{N} m_j (\Gamma_j - \Gamma_i + \alpha_i) \times (g + \mathbf{F}_j^{ext})$$

$$+ (\Gamma_i - \alpha_i) \times \mathbf{F}_i^{motor} + \mathbf{T}_i^c + \mathbf{T}_i^{motor} + \mathbf{T}_i^{ext} \ . \qquad (13)$$

if joint i is prismatic,

$$\sum_{j=i}^{N} m_j \ddot{\Gamma}_j = \mathbf{F}_i^c + \mathbf{F}_i^{motor} + \sum_{j=i}^{N} \mathbf{F}_j^{ext} + \sum_{j=i}^{N} m_j \ g \ .$$

We now have 3 x N equations for N links. Since from general principles we need only N second order equations to explicitly describe the dynamics of the N - link manipulator, we use the *principle of virtual work* to reduce the set of 3 $\times N$ equations to a set of N equations , one for each link.

We let ,

$$q_i = \mathbf{ACB}_i \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} ,$$

denote the axis of the degree of freedom.

By the *principle of virtual work* we know that the work done along constrained d.o.f. axes is equal to ' 0 '

Thus we have :

$$q_i^T \cdot \mathbf{T}_i^c = 0 \ , \tag{14}$$

$$q_i^T \cdot \mathbf{F}_i^c = 0 \ . \tag{15}$$

(We simply note that if there are damping forces/torques involved, equations (14) and (15) can be suitably modified.)

Taking dot product of q_i with the dynamic equations in equation (13) we get :

if the joint is revolute ,

$$\boxed{\begin{aligned} q_i^T \cdot \phi_i \cdot \dot{\omega}_i + q_i^T \cdot m_i \alpha_i \times \ddot{\Gamma}_i = & - q_i^T \cdot \omega_i \times \phi_i \cdot \omega_i + q_i^T \cdot \sum_{j=i}^{N} m_j (\Gamma_j - \Gamma_i + \alpha_i) \times g \\ & + q_i^T \cdot \sum_{j=i}^{N} (\Gamma_j - \Gamma_i + \alpha_i) \times \mathbf{F}_i^{ext} + q_i^T \cdot (\Gamma_i - \alpha_i) \times \mathbf{F}_i^{motor} \\ & + q_i^T \cdot \mathbf{T}_i^{motor} + q_i^T \cdot \mathbf{T}_i^{ext} \ . \qquad (16) \end{aligned}}$$

if the joint is prismatic ,

$$\boxed{q_i^T \cdot m_i \, \mathring{\ddot{\Gamma}}_i = q_i^T \cdot \sum_{j=i}^{N} (m_j \, g + \mathbf{F}_j^{ext}) + q_i^T \cdot \mathbf{F}_i^{motor} \ . \qquad (16)}$$

Combining the above equations we get,

$$\mathbf{B}(x)\,.\,\ddot{x} = \mathbf{f}(x\,,\dot{x}) \;+\; \mathbf{e}(x\,,\mathbf{F}^{ext}\,,\mathbf{T}^{ext}\,,\mathbf{F}^{motor}\,,\mathbf{T}^{motor})\;. \tag{17}$$

where $$x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}$$

and,

$x = \theta_i$ *if the joint is revolute*

$= r_i$ *if the joint is prismatic*

The functions $\mathbf{B}(.)$, $\mathbf{f}(.,.)$ and $\mathbf{e}(.,.,.,.,.)$ can be determined from equation (16).

3. Program Description:

The formulation of section 2 is implemented using the symbolic language VAXIMA (MACSYMA as implemented in the VAX 11/780). The package is called **DYNAMAN** - short for **DYN**amics and **A**nalysis of **MAN**ipulators. The program itself is extensively documented for clarity. The program is written such that it is user friendly. While using the program the user need only answer appropriately the questions posed by the program, and select various options.

The input of data to the package DYNAMAN can be done by a semi-sophisticated user, by making use of the built-in TUTORIAL program. The user can just say '*y*' when the program asks the relevant question. If the data already exists in a VAXIMA compatible form in a file, then the user has to give the name of the data file when the program asks for it.

The data for DYNAMAN consists of the following:

(i). Number of links of the manipulator - N.

(ii). Denavit-Hartenberg Parameters of the links of the manipulator - a,r,α,θ.

(iii). The position vectors of the center of mass of a link from the origin corresponding to the the local link coordinate system - αoi.

(iv). The physical parameters like the mass & moment of inertia tensor of the link - m ,ϕ .

(v). The gravity vector in the inertial (or base) coordinate system - go.

(vi). Torques & forces due to mechanisms at the joints, acting on the link - $\mathbf{T}^{motor}$,$\mathbf{F}^{motor}$.

(vii). External torques & forces acting on the link - $\mathbf{T}^{ext}$,$\mathbf{F}^{ext}$.

If $\mathbf{T}^{motor}$,$\mathbf{F}^{motor}$,$\mathbf{T}^{ext}$ or $\mathbf{F}^{ext}$ is not specified for a particular link, Then the final equations will carry these in symbolic form.

The following is a brief description of the input and output options available to the user.

(i) Selecting the TUTORIAL program or using a VAXIMA compatible data file.

(ii) Directing the output to any output file of his/her choice.

(iii) Choice of getting the intermediate calculations printed.

(iv) Choice of generating a FORTRAN file, for a direct simulation of the dynamic equations, automatically.

A HOWTORUN file exists for a novice.

4. VALIDATION

The following two examples were used to validate our program. The examples were taken from Horn and Freund [10, 11].

Example I

The dynamic equations of a 2-link planar manipulator [10] (fig(iii)) was generated. Both the joints were revolute. The dynamic equations were found to be :

$$\frac{\ddot{\theta}_1 (3\cos(\theta_2) + 5)}{6} + \frac{(3\cos(\theta_2) + 5)\ddot{\theta}_2}{3} = \frac{\mathrm{T}_1^{motor}}{l^2 m} + (\dot{\theta}_2^2 \sin^2(\theta_2) + 2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1))$$

$$- \frac{(\cos(\theta_1 + \theta_2) + 3 \cos(\theta_1)) g}{2l}$$

$$\frac{1}{3} \ddot{\theta}_2 + \frac{1}{6} \ddot{\theta}_1 (3\cos(\theta_2) + 2) = \frac{\mathrm{T}_2^{motor}}{l^2 m} - \frac{((\dot{\theta}_1^2 \sin(\theta_2)\, l + \cos(\theta_1 + \theta_2)\, g)}{2\, l}$$

The results were verified using the results from Horn's paper [10] and also by hand calculation. The requisite details including the data file are given in Appendix B.

Example II

This example was taken from Freund's paper [11]. Here we have a two link manipulator (fig(iv)) with the first joint being prismatic and the second one being revolute. The resulting dynamic equations are as below.

$$\frac{\ddot{\theta}_1 (l_r - 4 r_2 l_r + 4 r_2^2) m_r}{4} + \ddot{\theta}_1 \mathrm{I}_{disc} = \mathrm{T}_1^{motor} + (\dot{\theta}_1 \dot{r}_2 l_r - 2 r_2 \dot{\theta}_1 \dot{r}_2) m_r$$

$$\ddot{r}_2 m_r = \mathrm{F}_2^{motor} - \frac{\dot{\theta}_1^2 l_r - 2 r_2 \dot{\theta}_1^2}{2} m_r$$

The results were in agreement with those of Freund's [11]. Appendix C describes the example in greater detail along with the data file.

5. FUTURE WORK

Symbolic algebraic manipulation codes are becoming increasingly useful tools in many areas of applied science. The growing availability of work-stations (such as the SYMBOLICS 3600 Lisp Machine, Texas Instruments EXPLORER etc.) specifically designed for efficient symbolic processing is expected to lead to new and imaginative use of symbolic algebraic computation software, in research.

Our own efforts in this area will be directed towards enhancing **DYNAMAN** so as to be able to use the automatic programming facility and optimization-based design tools such as DELIGHT.MaryLin in an integrated manner. We intend to create the necessary interface software for this purpose. We also have plans to develop a COMMON LISP version of **DYNAMAN** that can be run on a Lisp machine.

6. SUMMARY :

DYNAMAN a software package for the automatic generation of dynamic equations of a robot manipulator was created. The software also has the capability of generating FORTRAN code for the simulation of the dynamic equations so formed. The formulation used was based on the method of nested bodies. The package was validated and tested successfully. It was also run in a SYMBOLICS 3600 Lisp machine to test its performance in an environment with large virtual memory.

7. ACKNOWLEDGEMENT :

We would like to thank Anthony Lavigna and David MacEnany for their kind assistance on matters concerning MACSYMA and its capabilities.

APPENDIX A

NOTATION :

Parameters

a_i	Denavit-Hartenberg parameter 'a' of the i'th link.
r_i	Denavit-Hartenberg parameter 'r' of the i'th link.
α_i	Denavit-Hartenberg parameter 'α' of the i'th link.
θ_i	Denavit-Hartenberg parameter 'θ' of the i'th link.

Origins

O_i	Origin of i'th co-ordinate system.
O_0	Origin of the inertial or the base co-ordinate system.

Vectors

αoi_i	Vector from $\mathbf{O}_{i+1}$ to c.m. (center of mass) of link i in the i'th co-ordinate system (fixed on link i).
αi_i	Vector from $\mathbf{O}_{i+1}$ to c.m. of link i in the inertial co-ordinate system.
α_i	Vector from $\mathbf{O}_i$ to c.m. of link in the inertial co-ordinate system (this is different from Denavit-Hartenberg parameter α).
$\dot{\alpha}_i$	Inertial velocity associated with vector α_i.
$\ddot{\alpha}_i$	Inertial accleration associated with vector α_i.
βi_i	Vector from $\mathbf{O}_i$ to $\mathbf{O}_{i+1}$ in the i'th co-ordinate system.
β_i	Vector from $\mathbf{O}_i$ to $\mathbf{O}_{i+1}$ in the inertial co-ordinate system.
$\dot{\beta}_i$	Inertial velocity associated with vector β_i.
$\ddot{\beta}_i$	Inertial accleration associated with vector β_i.
Γ_i	Vector from inertial origin $\mathbf{O}_0$ to c.m. of link i.
$\dot{\Gamma}_i$	Inertial velocity of link i c.m.
$\ddot{\Gamma}_i$	Inertial accleration of link i c.m.
$\tilde{\omega}_i$	Relative angular velocity of link i w.r.t. link i-1 in the inertial co-ordinate system.

ω_i	Inertial angular velocity of link i.
$\dot{\omega}_i$	Inertial angular accleration of link i.

Momentum

$\mathbf{G}_i$	Linear momentum of link i.
$\dot{\mathbf{G}}_i$	Rate of change of linear momentum of link i.
$\mathbf{L}_i$	Angular momentum of link i.
$\dot{\mathbf{L}}_i$	Rate of change of angular momentum of link i.
$\dot{\mathbf{L}}_i^{total}$	Rate of change of angular momentum of nest i.

Torques & Forces

$\mathbf{T}_i^{motor_i}$	Torque due to the mechanisms at joint i in the i'th co-ordinate system (Equal to 0 if joint i is prismatic).
$\mathbf{F}_i^{motor_i}$	Force due to the mechanisms at joint i in the i'th co-ordinate system (Equal to 0 if joint i is revolute).
$\mathbf{F}_i^{ext_i}$	Total external force acting on link i through its c.m. in the i'th co-ordinate system.
$\mathbf{T}_i^{ext_i}$	Total external torque acting on link i in the i'th co-ordinate system.

$\mathbf{F}_i^{ext}$	Total external force acting on link i (through the c.m. of link i) in the inertial co-ordinate system.
$\mathbf{T}_i^{ext}$	Total external torque acting on i'th link in the inertial co-ordinate system.
$\mathbf{F}_i^{c}$	Total constraint force acting on link i in the inertial co-ordinate system at joint i.
$\mathbf{T}_i^{c}$	Total constraint torque acting on link i in the inertial co-ordinate system.
$\mathbf{T}_i^{total}$	Total torque acting on nest i in the inertial co-ordinate system.

Matrices

$\mathbf{A}_i$	Transformation matrix from i'th to the (i-1)'th co-ordinate system (3 $\times$ 3).
$\mathbf{ACB}_i$	Transformation matrix from i'th co-ordinate system to the inertial co-ordinate system (3 $\times$ 3).

Physical Parameters

m_i	Mass of link i.
ϕ_i	Inertia tensor of link i.
go	Gravity Vector in the inertial co-ordinate system.
$jtyp$	Type of joint.
N	Number of links.

APPENDIX B

EXAMPLE 1 :

A planar manipulator with two equal links and with uniform mass distribution is considered. Fig(iii) shows a schematic of the manipulator. The following symbols are used in the body of the data file for example 1.

m	Mass of each manipulator link.
l	Length of each link.
θ_1	Angle between x axis and link 1.
θ_2	Angle between link 1 and link 2.
tm1	Torque due to motor at joint 1.
tm2	Torque due to motor at joint 2.
ϕ	Moment of inertia tensor of links about their own center of mass (c.m.).
g	Acceleration due to gravity.

A listing of the data file follows. The output file and also the FORTRAN code for numerical simulation of the dynamic equations, which was generated on request, is also given.

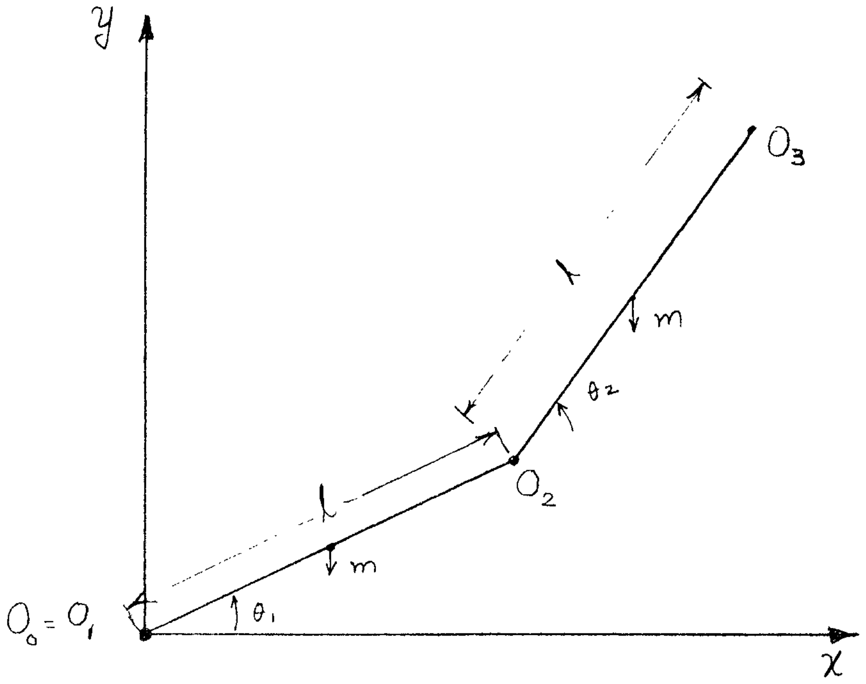

Fig.(III) Two-link manipulator with revolute joints

```
/*                                                              */
/*              DATA   FOR   TWO   LINK   MANIPULATOR           */
/*                                                              */
/*                     (example from Horn's paper)              */
/*                                                              */
/*                    File  :  rdata                            */
/*                                                              */


n:2;
jtyp[1]:r;
jtyp[2]:r;

go:matrix([Ø],[-g],[Ø]);
alpha[1]:Ø;
alpha[2]:Ø;


aoi[1]:matrix([-1/2],[Ø],[Ø]);
aoi[2]:matrix([-1/2],[Ø],[Ø]);

ai[1]:1;
rz[1]:Ø;
ai[2]:1;
rz[2]:Ø;


m[1]:m;
m[2]:m;

phi[1]:matrix([Ø,Ø,Ø],[Ø,m*l*l/12,Ø],[Ø,Ø,m*l*l/12]);
phi[2]:matrix([Ø,Ø,Ø],[Ø,m*l*l/12,Ø],[Ø,Ø,m*l*l/12]);

fexti[1]:fexti[2]:zeromatrix(3,1);
texti[1]:texti[2]:zeromatrix(3,1);


fmotori[1]:fmotori[2]:zeromatrix(3,1);
tmotori[1]:matrix([Ø],[Ø],[tm1]);
tmotori[2]:matrix([Ø],[Ø],[tm2]);
```

```
            ********************************************
              This File has been created by  :  VAXIMA
              For Vaxima INPUT-DATAFILE      : rdata
            ********************************************
FORTRAN  code has been generated on request


FORTRAN code in file :  rfor .f
                         ****************************

                           THE FINAL EQUATIONS ARE :

                         ****************************

                                      Main-Eqn

                                          1




                       2
                      d                    2
(3 cos(theta ) + 2) (--- (theta )) 1  m
            2           2        2
                      dt
----------------------------------------
                      6

       2
      d                               2
   (--- (theta )) (3 cos(theta ) + 5) 1  m
       2       1               2
      dt
 + --------------------------------------- =
                        3

                        d                2       d                                  d
tm1 + ((sin(theta ) (-- (theta ))  + 2 (-- (theta )) sin(theta ) (-- (theta )))
                 2   dt        2         dt       1            2   dt        2

  2
 1  + (- cos(theta  + theta ) - 3 cos(theta )) g 1) m/2
                  2        1                1
```

```
                                         Main-Eqn

                                             2




  2                          2
 d              2           d                                     2
(--- (theta )) l  m   (--- (theta )) (3 cos(theta ) + 2) l  m
   2        2                2        1             2
 dt                          dt
-------------------- + ---------------------------------------- =
          3                                6

                       d            2               2
                  (- (-- (theta ))  sin(theta ) l  - cos(theta  + theta ) g l) m
                      dt        1              2             2        1
           tm2 + ------------------------------------------------------------
                                              2
```

```
                ****************************************

                     The Equations in a short form

           In short notation dx = dx/dt  &  ddx = d/dt(dx/dt)

                                x  = theta
                                 1        1

                                x  = theta
                                 2        2



[ ddx  ]
[    1 ]
[      ] = matrix([(- 18 cos(x ) tm2 - 12 tm2 + 12 tm1
[ ddx  ]                      2
[    2 ]

          2                            2                    2            2
 + ((9 dx  cos(x ) sin(x ) + (6 dx  + 12 dx  dx  + 6 dx ) sin(x )) l
         1       2       2         2         1   2        1       2

                                                           2           2
 + (9 cos(x ) cos(x  + x ) g - 18 cos(x ) g) l) m)/((9 sin (x ) + 7) l  m)],
           2       2    1               1                    2

[- (- 6Ø tm2 + cos(x ) (18 tm1 - 36 tm2) + 12 tm1
                    2

           2                   2
 + (((9 dx  + 18 dx  dx  + 18 dx ) cos(x ) sin(x )
           2        1   2        1       2       2

          2                    2            2
 + (6 dx  + 12 dx  dx  + 3Ø dx ) sin(x )) l
         2        1   2        1       2

 + ((24 cos(x  + x ) - 18 cos(x )) g + cos(x ) (9 cos(x  + x ) - 27 cos(x )) g)
             2    1             1            2          2    1             1

              2            2
 l) m)/((9 sin (x ) + 7) l  m)])
                 2

                ****************************************
```

```
c   *************************************************************
c        Program to simulate the DYNAMICS  of a MANIPULATOR
c   *************************************************************
c
c
c        This Program is  Created by       : VAXIMA
c        For Vaxima Input-File             :  rdata
c        Corresponding Vaxima Output-File :  rout
c
c
c
      real l, m, g
      real tm1, tm2
       parameter(nlink=2, n2link=4)
      double precision xsave(1000,n2link)
      double precision  xs(n2link)
      common/param/l, m, g
      common/tork/tm1, tm2
c        Enter the parameters
         write(6,*) ' l, m, g  '
         read(5,*)  l, m, g
c        Enter the initial conditions
      write(6,*) 'Enter Initial Conditions'
      read(5,*) (xs(i),i=1, 4 )
      write(6,*) 'Enter Time-step   Max-time'
      read(5,*) dt,tmax
      t=0.
      itmax=tmax/dt
      do 100 ioo=1,itmax
      call rk4(xs,t.dt)
      t=t+dt
      write(6,*) (xs(i),i=1, 4 )
100   continue
      stop
       end
c
c
c
c      Runge-Kutta method
              subroutine rk4(x,t,dt)
       parameter(nst=4)
      double precision  x(nst)
              double precision tx(nst)
c     1 step integration.
              double precision d1(nst),d2(nst),d3(nst),d4(nst)
      call xdot(x,t,d1)
              do 1 i=1,nst
1     tx(i)=x(i)+0.5*dt*d1(i)
              tt=t+dt/2.
              call xdot(tx,tt,d2)
              do 2 i=1,nst
2     tx(i)=x(i)+0.5*dt*d2(i)
              call xdot(tx,tt,d3)
              do 3 i=1,nst
3     tx(i)=x(i)+dt*d3(i)
              tt=t+dt
              call xdot(tx,tt,d4)
              do 4 i=1,nst
```

```
4      x(i)=x(i)+dt*(d1(i)+2.*d2(i)+2.*d3(i)+d4(i))/6.
              return
              end
c
c
c
       subroutine xdot(fx,t,fdx)
       real l, m, g
       real tm1, tm2
        parameter(n2lin=4, nlin=2)
       double precision fx(n2lin),fdx(n2lin),dx(nlin),x(nlin)
       common/param/l, m, g
       common/tork/tm1, tm2
       do 1Ø ipp=1,nlin
       dx(ipp)=fx(nlin+ipp)
       x(ipp)=fx(ipp)
1Ø       continue
       do 2Ø ipp=1,nlin
       fdx(ipp)=dx(ipp)
2Ø       continue
              fdx(3) =(-18*cos(x(2))*tm2-12*tm2+12*tm1+((9*dx(1)**2*cos(x
     1   (2))*sin(x(2))+(6*dx(2)**2+12*dx(1)*dx(2)+6*dx(1)**2)*sin(x(2))
     2   )*l**2+(9*cos(x(2))*cos(x(2)+x(1))*g-18*cos(x(1))*g)*l)*m)/((9*
     3   sin(x(2))**2+7)*l**2*m)
              fdx(4) =-(-6Ø*tm2+cos(x(2))*(18*tm1-36*tm2)+12*tm1+(((9*dx(
     1   2)**2+18*dx(1)*dx(2)+18*dx(1)**2)*cos(x(2))*sin(x(2))+(6*dx(2)*
     2   *2+12*dx(1)*dx(2)+3Ø*dx(1)**2)*sin(x(2)))*l**2+((24*cos(x(2)+x(
     3   1))-18*cos(x(1)))*g+cos(x(2))*(9*cos(x(2)+x(1))-27*cos(x(1)))*g
     4   )*l)*m)/((9*sin(x(2))**2+7)*l**2*m)
       return
       end
```

APPENDIX C

EXAMPLE 2 :

This is a two link manipulator, consisting of a revolute joint followed by a prismatic joint (Fig(iv)). In the data file that follows we have the special symbols for various mechanical variables as below:

m_d	Mass of the disc.
m_r	Mass of the rod.
l_r	Length of the rod.
$tm\,1$	Torque due to the mechanism at joint 1 ($T_1^{motor} = [0,0,tm\,1]^T$).
$fm\,2$	Force due to the mechanism at joint 2 ($F_2^{motor} = [0,0,fm\,2]^T$).
ϕ_1	Moment of inertia tensor of the disc about its own center of mass.
ϕ_2	Moment of inertia tensor of the rod about its center of mass.
g	Accleration due to gravity.

A listing of the data file is attached, along with the corresponding output file and the FORTRAN code for the numerical simulation of the dynamic equations so generated.

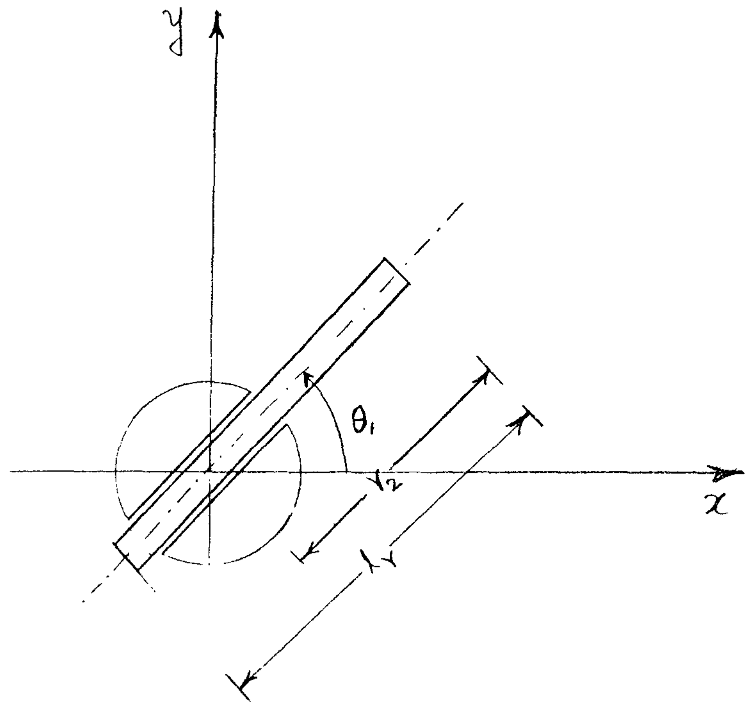

Fig.(iv) Two-link manipulator - one revolute and one prismatic joint

```
/*      DATA  FOR TWO LINK MANIPULATOR                                */
/*                                                                    */
/*               (example from Freund's paper in the black book)     */
/*                                                                    */
/*      File  :  prdata                                               */

n:2;

jtyp[1]:r;
jtyp[2]:p;

go : matrix([Ø],[Ø],[-g]);

alpha[1]:Ø;
alpha[2]:-%pi/2;

m[1]:md;
m[2]:mr;

ah[1]:Ø;
rz[1]:Ø;
ah[2]:Ø;

theta[2]:Ø;

aoi[1]:matrix([Ø],[Ø],[Ø]);
aoi[2]:matrix([Ø],[Ø],[-lr/2]);


phi[1]:matrix([idsc,Ø,Ø],[Ø,idsc,Ø],[Ø,Ø,idscz]);
phi[2]:matrix([mr*lr*lr/12,Ø,Ø],[Ø,mr*lr*lr/12,Ø],[Ø,Ø,Ø]);

fexti[1]:fexti[2]:zeromatrix(3,1);
texti[1]:texti[2]:zeromatrix(3,1);

fmotori[1]:zeromatrix(3,1);
fmotori[2]:matrix([Ø],[Ø],[fm2]);
tmotori[1]:matrix([Ø],[Ø],[tm1]);
tmotori[2]:zeromatrix(3,1);
```

```
          ***********************************************
            This File has been created by  :  VAXIMA
            For Vaxima INPUT-DATAFILE       : prdata
          ***********************************************
FORTRAN  code has been generated on request


FORTRAN code in file :  prfor .f
                         ******************************

                          THE FINAL EQUATIONS ARE :

                         ******************************

                                    Main-Eqn

                                        1



  2
 d                     2                  2
(--- (theta )) (lr  - 4 rz  lr + 4 rz ) mr
   2       1              2          2
 dt                                                     2
                                                       d
------------------------------------------ + (--- (theta )) idscz =
                     4                             2        1
                                                 dt

                  d               d                           d                d
        tm1 + ((-- (theta )) (-- (rz )) lr - 2 rz  (-- (theta )) (-- (rz ))) mr
                 dt       1    dt    2               2  dt       1    dt    2
```

```
                         Main-Eqn

                              2


                    d         2                d               2
  2               ((-- (theta  ))  lr - 2 rz  (-- (theta  ))  ) mr
 d                  dt         1              2  dt        1
(--- (rz  )) mr = fm2 - ------------------------------------------------
  2     2                                      2
 dt

          ****************************************

               The Equations in a short form

        In short notation dx = dx/dt  &  ddx = d/dt(dx/dt)

                       x  = theta
                        1         1

                        x  = rz
                         2     2


                [ 4 tm1 + (4 dx  dx  lr - 8 dx  dx  x ) mr ]
                [              1   2           1   2  2    ]
                [ ---------------------------------------- ]
    [ ddx  ]    [      2                2                   ]
    [    1 ]    [  (lr  - 4 x  lr + 4 x ) mr + 4 idscz     ]
    [      ] =  [             2          2                 ]
    [ ddx  ]    [                                          ]
    [    2 ]    [         2          2                     ]
                [      (dx  lr - 2 dx  x ) mr - 2 fm2      ]
                [         1          1  2                  ]
                [    - ------------------------------      ]
                [                  2 mr                    ]

          ****************************************
```

```
c   ********************************************************
c       Program to simulate the DYNAMICS  of a MANIPULATOR
c   ********************************************************
c
c
c       This Program is  Created by       : VAXIMA
c       For Vaxima Input-File             :  prdata
c       Corresponding Vaxima Output-File :  prout
c
c
c
      real idscz, lr, mr
      real tm1, fm2
       parameter(nlink=2, n2link=4)
      double precision xsave(1ØØØ,n2link)
      double precision  xs(n2link)
      common/param/idscz, lr, mr
      common/tork/tm1, fm2
c        Enter the parameters
         write(6,*) ' idscz, lr, mr   '
         read(5,*)  idscz, lr, mr
c        Enter the initial conditions
      write(6,*) 'Enter Initial Conditions'
      read(5,*) (xs(i),i=1, 4 )
      write(6,*) 'Enter Time-step   Max-time'
      read(5,*) dt,tmax
      t=Ø.
      itmax=tmax/dt
      do 1ØØ ioo=1,itmax
      call rk4(xs,t,dt)
      t=t+dt
      write(6,*) (xs(i),i=1, 4 )
1ØØ   continue
      stop
       end
c
c
c
c      Runge-Kutta method
             subroutine rk4(x,t,dt)
       parameter(nst=4)
      double precision  x(nst)
             double precision tx(nst)
c     1 stop integration.
             double precision d1(nst),d2(nst),d3(nst),d4(nst)
      call xdot(x,t,d1)
             do 1 i=1,nst
1     tx(i)=x(i)+Ø.5*dt*d1(i)
             tt=t+dt/2.
             call xdot(tx,tt,d2)
             do 2 i=1,nst
2     tx(i)=x(i)+Ø.5*dt*d2(i)
             call xdot(tx,tt,d3)
             do 3 i=1,nst
3     tx(i)=x(i)+dt*d3(i)
             tt=t+dt
             call xdot(tx,tt,d4)
             do 4 i=1,nst
4     x(i)=x(i)+dt*(d1(i)+2.*d2(i)+2.*d3(i)+d4(i))/6.
             return
             end
```

```
c
c
c
      subroutine xdot(fx,t,fdx)
      real idscz, lr, mr
      real tm1, fm2
       parameter(n2lin=4, nlin=2)
      double precision fx(n2lin),fdx(n2lin),dx(nlin),x(nlin)
      common/param/idscz, lr, mr
      common/tork/tm1, fm2
      do 1Ø ipp=1,nlin
      dx(ipp)=fx(nlin+ipp)
      x(ipp)=fx(ipp)
1Ø      continue
      do 2Ø ipp=1,nlin
      fdx(ipp)=dx(ipp)
2Ø      continue
             fdx(3) =(4*tm1+(4*dx(1)*dx(2)*lr-8*dx(1)*dx(2)*x(2))*mr)/((
     1   lr**2-4*x(2)*lr+4*x(2)**2)*mr+4*idscz)
             fdx(4) =-((dx(1)**2*lr-2*dx(1)**2*x(2))*mr-2*fm2)/mr/2.Ø
      return
      end
```

REFERENCES

[1] H.P.Frisch, "The NBOD2 User's and Programmer's Manual," in *NASA TP 1145,*, (Feb, 1978).

[2] H.P.Frisch, "A vector-dyadic development of the equations of motion for a N-coupled rigid bodies and point masses," in *NASA TND-7767,* (Oct, 1974).

[3] R.Sturges , "Teleoperators arm design program," in *Report 3.2746,* , MIT Draper Lab. Cambridge Mass. (1973).

[4] S.R.Dillon, "Computer assisted equation generation in linkage dynamics," in *Thesis, Ohio State University,* (1973).

[5] A.Liegeois , W.Khalil, J.M.Dumas, and M.Renauld, "Mathematical and computer models of interconnected mechanical systems," pp. 5-17 in *2nd International symposium on theory and practice of robots and manipulators,* (Sept. 1976).

[6] W.Khalil, "Modelization et commande par calculeteurs du manipulateurs," in *Thesis,Universite' des Sciences et Technique du Languedoc,* (1976).

[7] G.Cesareo and F. Nicolo, "DYMIR: A code for generating dynamic models of robots," in *preprint,* (1984).

[8] J.R.Velman, "Simulation results for a Dual-spin spacecraft ," pp. 1-12 in *Proc. of the Symp. on Attitude Stabilization and Control of Dual-spin Spacecrafts (Aug. 1967), SAMSO TR-68-191, Airforce System Command, Space and Missile System Organization and Aerospace Corp.,* (Nov. 1967).

[9] J.Denavit and R.S.Hartenberg, "A kinematic notation for lower-pair mechanism based on matricies," *Trans. ASME Journal of Applied Mechanics* vol. 77, pp. 215-221 (June,1955).

[10] B.K.P.Horn, "Kinematics, statics and dynamics of two-dimensional manipulators," pp. 273-310 in *Artificial Intelligence : An MIT Perspective,* ed. Patrick Winston and Richard Brown, , MIT Press, Cambridge, Mass. (1979).

[11] Freund, "Fast non-linear control with arbitrary pole-placement for industrial robots and manipulators," pp. 147-168 in *Robot Motion: Planning and Control,* ed. M.Brady, J.M.Hollerbach, T.L.Johnson, T. Lorenzo-Perez and M.T.Mason, , MIT Press,Cambridge, Mass. (1982).