Computer modeling of spatial mechanisms for kinematic analysis and synthesis
by John Michael Lowell

A thesis submitted in partial fullfillment of the requirements for the degree of MASTER OF SCIENCE in Mechanical Engineering
Montana State University

Abstract:
A methodology is presented from which a general purpose computer code for determining the kinematics (position, velocity, and acceleration) of spatial mechanisms may be generated. This methodology permits the equations which describe the mechanism kinematics to be assembled and solved by a program which only needs to read a concise data base describing the mechanism and type of solution desired.

The inherent flexibility in this method permits modification of the mechanism configuration and characteristics by altering the data base describing the mechanism. Changing relatively few parameters in the data base makes possible the synthesis of motion. That is, for a given desired path or configuration of the mechanism, the values of the mechanism parameters (coordinates) can be determined without alterations in computer code. The method presented is well suited for multi-positional kinematic analysis and synthesis of complex spatial mechanisms with multiple degrees of freedom.

Two examples of kinematic analysis using this method are presented. The first is a three degree of freedom closed loop mechanism. The second is the robot arm used on the space shuttle, which is a six-degree-of freedom mechanism. Examples of position, velocity, and acceleration solutions are given for both mechanisms.

STATEMENT OF PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Montana State University, I agree that the Library shall make it freely available for inspection. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by my major professor, or, in his absence, by the Director of Libraries. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature John M. Lowell

Date May 28, 1982

COMPUTER MODELING OF SPATIAL MECHANISMS FOR KINEMATIC ANALYSIS AND SYNTHESIS

by

JOHN MICHAEL LOWELL

A thesis submitted in partial fullfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Mechanical Engineering

Approved:

Dennis O. Blackketter
Chairperson, Graduate Committee

Dennis O. Blackketter
Head, Major Department

Michael Malone
Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

May, 1982

ACKNOWLEDGEMENTS

The author wishes to thank Dr. D. O. Blackketter and Dr. H. W. Townes for their advice and assistance in performing this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Symbol	Description
{AA}	column matrix containing second time derivative of {PA}
$\{AA_I\}$	second time derivative of $\{PA_I\}$ (Equation 2.12)
$[a_I]$	3 X 4 acceleration coefficient array for vector I (Equation 2.11)
c_I	4 X 1 cosine squared constraint coefficient array (Equation 2.5)
c'_I	first time derivative of c_I (Equation 2.9)
c''_I	second time derivative of c_I (Equation 2.13)
$[d_I]$	3X4 displacement coefficient array for vector I (Equation 2.2)
[D]	n X m assembled displacement matrix for model (Equation 3.1)
i	number of vectors in model
{IA}	array used to identify known and unknown accelerations
{IP}	array used to identify known and unknown geometry parameters
{IV}	array used to identify known and unknown velocities
j	number of vector paths in the model
[LI]	path information matrix, stores information as to which vector is in which path and its direction
m	number of kinematic equations, m is equal to j times three plus i

Symbol	Description
np	number of parameters defining model, np is equal to i times four
{PA}	column matrix containing all known and unknown geometry parameters
$\{PA_I\}$	1 X 4 parameter array for vector I (Equation 2.3)
{PD}	path displacement matrix, contains X, Y, and Z displacement from beginning to end of each vector path
{VA}	column matrix containing first time derivative of {PA}
$\{VA_I\}$	time derivative of $\{PA_I\}$ (Equation 2.8)
$[v_I]$	3 X 4 velocity coefficient array for vector I (Equation 2.7)

Subscripts

a	denotes parameter in direction angle coordinate system
c	denotes parameter in direction cosine coordinate system
I	corresponding to vector I

ABSTRACT

A methodology is presented from which a general purpose computer code for determining the kinematics (position, velocity, and acceleration) of spatial mechanisms may be generated. This methodology permits the equations which describe the mechanism kinematics to be assembled and solved by a program which only needs to read a concise data base describing the mechanism and type of solution desired.

The inherent flexibility in this method permits modification of the mechanism configuration and characteristics by altering the data base describing the mechanism. Changing relatively few parameters in the data base makes possible the synthesis of motion. That is, for a given desired path or configuration of the mechanism, the values of the mechanism parameters (coordinates) can be determined without alterations in computer code. The method presented is well suited for multi-positional kinematic analysis and synthesis of complex spatial mechanisms with multiple degrees of freedom.

Two examples of kinematic analysis using this method are presented. The first is a three degree of freedom closed loop mechanism. The second is the robot arm used on the space shuttle, which is a six-degree-of freedom mechanism. Examples of position, velocity, and acceleration solutions are given for both mechanisms.

CHAPTER I

INTRODUCTION

Rapid advances in robotics and automated equipment for industrial, underwater, and space applications have placed demands on engineers to design and analyze increasingly sophisticated three-dimensional mechanisms. While the use of vector mechanics to generate the equations necessary for the kinematic analysis of three-dimensional mechanisms is a well known procedure, the derivation and solution of the resulting equations is a lengthy process which typically has required a problem specific computer code for solution.

This thesis does not enlarge on the body of knowledge related to mechanism theory, but describes a methodology by which a three-dimensional, rigid body mechanism can be described with a relatively concise data base. This data base is used as input to a computer code which can assemble and solve the equations which determine the position, velocity, and acceleration of each parameter used to represent the mechanism model. The computer code is not problem dependent and can be used for either analysis or synthesis by changing the data base. Synthesis as used in this thesis refers to determining the configuration of the mechanism required to obtain a given position, velocity, or acceleration of some member or point in the mechanism. This

computer code frees the designer from the necessity of developing programs which are only useful for a single mechanism and difficult to modify.

In application the method presented resembles the finite element technique [1] in the way mechanisms are modeled and the governing equations assembled. Matrix equations for the position, velocity, and acceleration relations of an arbitrary vector are derived using vector mechanics. The overall mechanism equations are assembled from these blocks of equations using information from a data base which describes the mechanism configuration and type of solution desired.

Kinematic problems are typically formulated as problems in vector analysis. However, a variety of solution techniques can be applied to the resulting vector equations. Some of the most commonly used solution techniques include graphical solutions, vector algebra, and complex number methods. While these techniques are methods of vector analysis, each method of solution has inherent advantages and disadvantages.

Historically, graphical techniques [2,3] have been the predominating technique for solution of the kinematics of planar mechanisms. Since solution by graphical methods

requires drawing of the position, velocity and acceleration polygons, its accuracy is limited. Also, graphical methods become very tedious when multiple position solutions are desired. Graphical methods are difficult to use on spatial mechanisms.

Solutions by algebraic methods, whether based on complex numbers [2,3] or vector algebra [2,3], have several advantages over graphical techniques. Accuracy is limited only by the accuracy of the problem data and the numerical evaluation of the solution. Once the problem has been put in an algebraic form, a kinematic solution can be obtained at different positions of the mechanism by solving sets of simultaneous equations. However, an algebraic solution usually requires tedious mathematical manipulations to obtain a solution. Algebraic methods are also applicable to spatial mechanisms [3], although the algebra is more complicated than for planar mechanisms.

Chace, at the University of Michigan, was one of the first to use vector notation to obtain closed form solutions to three-dimensional vector equations [4]. The Chace approach consists of defining solutions to a vector tetrahedron equation in spherical coordinates according to the unknowns in the equations. The resulting nine possible

solutions are reduced to explicit closed-forms and can be quickly evaluated. Chace used this technique to develop the ADAMS (Automatic Dynamic Analysis of Mechanical Systems) computer code, one of the more widely used industrial programs for analysis of three-dimensional mechanisms.

Garcia de Jalon, Serna, and Viadero [5] have offered a new technique using matrix methods and Lagrangian coordinates to solve the kinematics of spatial mechanisms. In this technique the members of a mechanism are represented by three or more points which are the Lagrangian coordinates for the member. Geometric constraint equations requiring constant distance between points, constant area of planar surfaces, and constant volume of solids are specified. Time derivatives of these equations are used for kinematic analysis. However, this method requires a previously known position which limits it's use for multi-position analysis.

The purpose of this thesis is to present a method using classical vector mechanics from which a general purpose computer code may be developed to perform kinematic analysis and synthesis of spatial mechanisms.

CHAPTER II

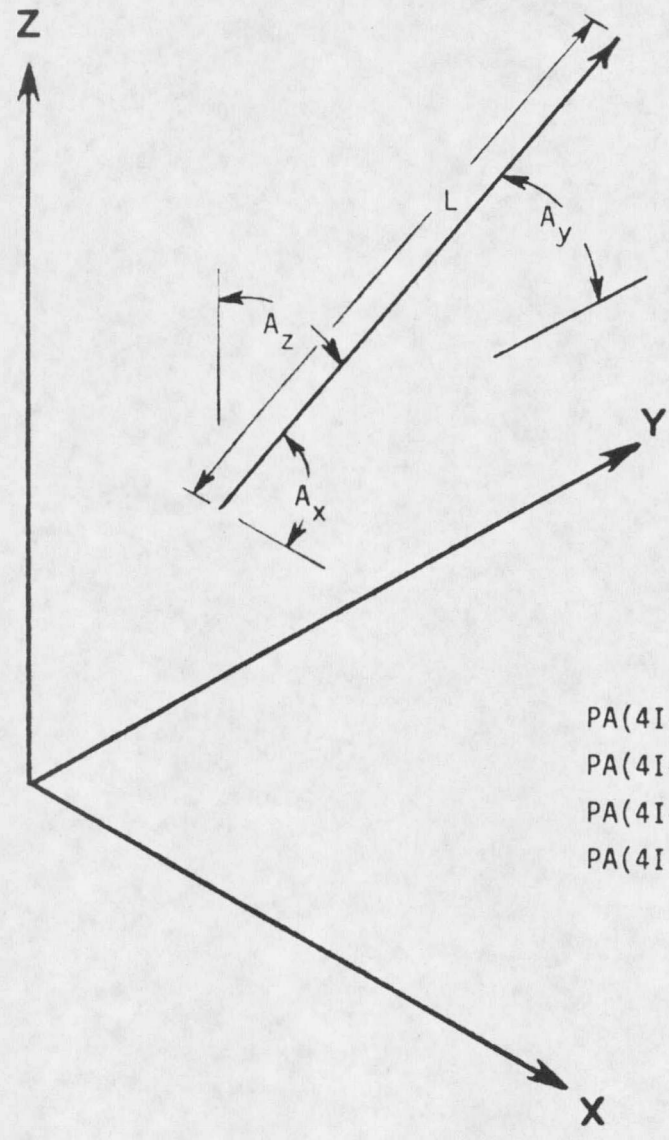
DEVELOPMENT OF VECTOR EQUATIONS

An arbitrary vector in a three-dimensional Cartesian coordinate system will be defined in terms of four parameters. One parameter is the length of the vector and the other three parameters define its direction. Direction can be defined using either direction angles or cosines. The three direction parameters are not independent as any one can be calculated given the other two. Equations defining the position, velocity, and acceleration relationships for an arbitrary vector will be developed in terms of these four parameters and their derivatives. One or more vectors can be used to define a member or locate points of interest in a mechanism.

With vectors defined in this manner, one has the option of developing the kinematic equations in terms of either the direction angles or the direction cosines. These two coordinate systems yield different forms of the vector path equations. Use of direction cosine coordinates to define vector direction results in nonlinear algebraic equations while use of direction angle coordinates to define vector direction results in nonlinear transcendental equations. Use of the direction angles also causes convergence problems when an iterative method is used to

solve the nonlinear equations for the geometry of the mechanism. The time derivatives of the cosine squared constraint equations expressed in direction angle coordinates are identically satisfied when a vector is parallel to one of the axes. This can result in singular matrices for the velocity and acceleration relations at certain positions of a mechanism. Using direction cosines as parameters results in equations which take less computer time to construct, show better convergence characteristics when using an iterative technique to solve them, and their derivatives do not yield a singular set of equations when a vector is parallel to one of the axes. However, use of cosine variable parameters can result in problems when attempting to convert known and unknown angular velocities and accelerations into and out of the cosine variable coordinates from more commonly used units in other coordinate systems. Due to the advantages of using cosine variables, all equations were developed using the direction cosines.

For convenience and ease of programming, the parameters defining the arbitrary vector I shown in Figure 1 are stored in a parameter array {PA}, and subscripted in the following manner:



PA(4I-3) = L
PA(4I-2) = $\cos(A_x)$
PA(4I-1) = $\cos(A_y)$
PA(4I) = $\cos(A_z)$

FIGURE 1 Schematic of Vector I

PA(I*4-3) = Length of vector I
PA(I*4-2) = Cosine of the angle between X axis and vector I
PA(I*4-1) = Cosine of the angle between Y axis and vector I
PA(I*4) = Cosine of the angle between Z axis and vector I

Two additional matrices, a velocity array {VA}, and an acceleration array {AA} store the first and second time derivatives of the {PA} matrix respectively, and are subscripted in the same manner. With vector I defined in this manner the component relations can be written in matrix form as:

$$[d_I]\{PA_I\} = \begin{Bmatrix} x \text{ component} \\ y \text{ component} \\ z \text{ component} \end{Bmatrix}_I \qquad (2.1)$$

where

$$[d_I] = \begin{bmatrix} PA(n+1) & 0 & 0 & 0 \\ PA(n+2) & 0 & 0 & 0 \\ PA(n+3) & 0 & 0 & 0 \end{bmatrix}_I \qquad (2.2)$$

$$\{PA_I\} = \begin{Bmatrix} PA(n) \\ PA(n+1) \\ PA(n+2) \\ PA(n+3) \end{Bmatrix}_I \qquad (2.3)$$

$$n = 4I - 3 \qquad (2.4)$$

I = vector number

While equation (2.1) defines the component relationships for a single vector, an additional equation is needed as the three direction cosines are not independent. This equation states that the sum of the squares of the direction

cosines must equal one. In matrix form this is:

$$\lfloor C_I \rfloor \{PA_I\} = 1 \tag{2.5}$$

where

$$\lfloor C_I \rfloor = \lfloor 0 \quad PA(n+1) \quad PA(n+2) \quad PA(n+3) \rfloor \quad I$$

The velocity relations for vector I are generated by taking the time derivative of the component relations given in equation (2.1) yielding the following equations:

$$[v_I]\{VA_I\} = \begin{Bmatrix} X \text{ component of velocity} \\ Y \text{ component of velocity} \\ Z \text{ component of velocity} \end{Bmatrix} \tag{2.6}$$

where

$$[v_I] = \begin{bmatrix} PA(n+1) & PA(n) & 0 & 0 \\ PA(n+2) & 0 & PA(n) & 0 \\ PA(n+3) & 0 & 0 & PA(n) \end{bmatrix} \quad I \tag{2.7}$$

$$\{VA_I\} = \begin{Bmatrix} VA(n) \\ VA(n+1) \\ VA(n+2) \\ VA(n+3) \end{Bmatrix} \quad I \tag{2.8}$$

The time derivative of the cosine squared constraint equation (2.5) yields the additional velocity relationship:

$$\lfloor C'_I \rfloor \{VA_I\} = 0 \tag{2.9}$$

where

$$\lfloor C'_I \rfloor = \lfloor 0 \quad 2*PA(n+1) \quad 2*PA(n+2) \quad 2*PA(n+3) \rfloor \quad I$$

The acceleration relations for vector I are generated by taking the second time derivative of the component relations which yields:

$$[v_I]\{AA_I\} + [a_I]\{VA_I\} = \left\{\begin{array}{l}\text{X component of acceleration}\\ \text{Y component of acceleration}\\ \text{Z component of acceleration}\end{array}\right\}_I \tag{2.10}$$

where

$$[a_I] = \begin{bmatrix} VA(n+1) & VA(n) & 0 & 0 \\ VA(n+2) & 0 & VA(n) & 0 \\ VA(n+3) & 0 & 0 & VA(n) \end{bmatrix}_I \tag{2.11}$$

$$\{AA_I\} = \begin{Bmatrix} AA(n) \\ AA(n+1) \\ AA(n+2) \\ AA(n+3) \end{Bmatrix}_I \tag{2.12}$$

The second time derivative of the cosine constraint equation (2.5) yields the additional acceleration equation:

$$\lfloor C''_I \rfloor \{VA_I\} = 0 \tag{2.13}$$

where

$$\lfloor C''_I \rfloor = \lfloor 0 \quad 2*VA(n+1) \quad 2*VA(n+2) \quad 2*VA(n+3) \rfloor_I$$

Equations (2.1) thru (2.13) define the kinematic relationships for an arbitrary vector. These vector relations can then be used to define models of large, complicated three-dimensional mechanisms with a wide variety of linkages.

CHAPTER III

ASSEMBLY AND SOLUTION OF KINEMATIC EQUATIONS

Computer assembly of the kinematic equations for a vector model of a mechanism requires information describing the configuration of the model. The information necessary to generate equations which describe vector paths consists of the vectors contained in a given path and the relative displacement from start to finish. More will be said about how these paths are determined in the following chapter on modeling.

A set of matrices will be used to store the information describing the vector model of the mechanism. A path information matrix, [LI], stores the information as to which vector is contained in which path and its direction. The [LI] matrix, which is number of paths by number of vectors in size, contains one row of information for each path in the mechanism. Each row contains an identifier for each vector in the model, this identifier is a one if the vector is in the particular path in the positive direction, minus one if in the negative direction, and zero if the vector is not in the path. The terms in the [LI] matrix are used as multipliers on the $[d_I]$, $[v_I]$, and $[a_I]$ matrices when assembling the kinematic equations for the mechanism.

A second information matrix, {PD}, contains the X, Y,

and Z displacements from beginning to end of a vector path. This {PD} matrix is three times the number of paths in length. The values of the {PD} array are zero for closed paths and non-zero for open paths. The terms PD(3*J-2), PD(3*J-1), and PD(3*J) represent the differences in the X, Y, and Z coordinates, respectively, from the beginning to end of the J-th path.

Geometry

The generation of the geometry equations for the model from the information stored in the [LI] and {PD} arrays is done in the following manner. The matrix [D] is used to store the assembled displacement equations for all vector paths plus the cosine squared constraint equations and is generated as follows:

(3.1)

$$
[D] = \begin{bmatrix}
[d_1]*LI(1,1) & [d_2]*LI(1,2) & \cdots & [d_{np}]*LI(1,i) \\
[d_1]*LI(2,1) & [d_2]*LI(2,2) & \cdots & [d_{np}]*LI(2,i) \\
\vdots & \vdots & & \vdots \\
[d_j]*LI(j,1) & [d_2]*LI(j,2) & \cdots & [d_{np}]*LI(j,i) \\
c_1 & 0 & \cdots & 0 \\
0 & c_2 & \cdots & 0 \\
0 & 0 & \ddots & \vdots \\
\vdots & \vdots & \ddots & 0 \\
0 & 0 & \cdots & c_i
\end{bmatrix}
$$

where j is number of paths, i is the number of vectors, and

np is the number of parameters in the model.

The geometry equations are then written as:

$$[\ D\ ]\ \{PA\} = \begin{Bmatrix} \{PD\} \\ \{\ 1\} \end{Bmatrix} . \qquad (3.2)$$

This is a set of m nonlinear equations in np parameters, where m is equal to the number of vectors plus three times the number of paths, and np is greater than m. Therefore np minus m parameters must be known. The number of known parameters is the number of degrees of freedom available in the model.

The {IP} array is used to identify the known and unknown parameters in the following manner: if IP(L) equals one, then PA(L) is unknown; if IP(L) equals zero, then PA(L) is known. The information in the {IP} array is used to rearrange equation (3.2) to a m-by-m set of nonlinear equations by moving known quantities to the right-hand side of the equation.

Once equation (3.2) has been rearranged, it may be used to solve for the geometry of the mechanism. The author has used the Newton-Raphson method or generalized Newton's method [6] for solution. Newton's method solves the set of nonlinear equations in (3.2) by solving a related set of linear equations several times to converge to the solution

of the nonlinear equations. Newton's method does not require storage of the [D] matrix and results in very rapid convergence, but does require an initial guess for the values of the unknown parameters. As nonlinear equations in general have several possible solutions, the initial guesses must be relatively close to the values which result in the mechanism being in the desired orientation, otherwise convergence to an alternate orientation may occur. For small movements of the mechanism, the previous position serves as an excellent guess for the current position analysis.

Velocity

Once the geometry of the mechanism is known, the velocity relations for the mechanism are generated by taking the time derivative of equation (3.2) which yields:

$$[\ V\ ]\ \{VA\} = \{0\} \tag{3.3}$$

where (3.4)

$$[V] = \begin{bmatrix} [v_1]*LI(1,1) & [v_2]*LI(1,2) & \cdots & [v_{np}]*LI(1,i) \\ [v_1]*LI(2,1) & [v_2]*LI(2,2) & \cdots & [v_{np}]*LI(2,i) \\ \vdots & \vdots & & \vdots \\ [v_j]*LI(j,1) & [v_2]*LI(j,2) & \cdots & [v_{np}]*LI(j,i) \\ c'_1 & 0 & \cdots & 0 \\ 0 & c'_2 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & c'_i \end{bmatrix}$$

The {IV} array is used to identify known and unknown velocity parameters in the same manner the {IP} array is used to identify the geometry parameters. Using the information in the {IV} array equation (3.3) can be rearranged so that all known quantities move to the right hand side and are summed. Known time derivates of angular parameters are expressed in terms of the cosine coordinate system in the following manner. Let PA_a and VA_a represent the direction angle and it's corresponding derivative respectively, the corresponding time derivative in cosine coordinates, VA_c, is then calculated with the following coordinate transformation:

$$VA_c = -VA_a \sin(PA_a) \qquad (3.5)$$

Equation (3.3) is a linear set of equations in velocities and is solved using an exact technique such as Gauss elimination [7].

Acceleration

With the positions and velocities of the mechanism components known, the acceleration relations can be obtained from the time derivates of equation (3.3) which yield:

$$[\ A\ ]\ \{VA\} + [\ V\ ]\ \{AA\} = \{0\} \qquad (3.6)$$

where (3.7)

$$[A] = \begin{bmatrix} [a_1]*LI(1,1) & [a_2]*LI(1,2) & \cdots & [a_{np}]*LI(1,i) \\ [a_1]*LI(2,1) & [a_2]*LI(2,2) & \cdots & [a_{np}]*LI(2,i) \\ \vdots & \vdots & & \vdots \\ [a_j]*LI(j,1) & [a_2]*LI(j,2) & \cdots & [a_{np}]*LI(j,i) \\ c''_1 & 0 & \cdots & 0 \\ 0 & c''_2 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & c''_i \end{bmatrix}$$

Since [A] and {VA} are functions of known quantities, equation (3.6) can be rewritten as:

$$[\ V\ ]\ \{AA\} = -\{Y\} \tag{3.8}$$

where

$$\{Y\} = [\ A\ ]\ \{VA\}. \tag{3.9}$$

The {IA} array is used to identify known and unknown acceleration parameters in the same manner as the {IP} array identifies geometry parameters. Using the information in the {IA} array equation (3.6) can be rearranged so that all known quantities are moved to the right hand side and added to {Y}. The known accelerations are expressed in the cosine coordinate system in the following manner. Let AA_a represent the second time derivative of PA_a, the corresponding

acceleration in cosine coordinates, AA_c, is given by the following coordinate transformation:

$$AA_c = -AA_a\sin(PA_a) - VA_a^2\cos(PA_a) \qquad (3.10)$$

Once rearranged equation (3.6) is a m-by-m set of linear equations in accelerations and may be solved by Gauss elimination or some other exact technique.

After the kinematic calculations are completed, the calculated first and second time derivatives of the angular parameters can be expressed in terms of the direction angle coordinate system by the following coordinate transformations:

$$VA_a = -VA_c/\sin(PA_a) \qquad (3.11)$$

$$AA_a = -AA_c - VA_a^2\cot(PA_a) \qquad (3.12)$$

One should note that [V] is the total derivative of the [D] matrix and is used in Newton's method to calculate the differential changes in {PA} for each iteration. The [V] matrix is also the coefficient matrix in the velocity and acceleration calculations. Figures 2 and 3 show flow diagrams for a computational scheme which takes advantage of the fact that only the [V] matrix need be stored and can be rearranged using the {IP}, {IV}, and {IA} arrays to solve

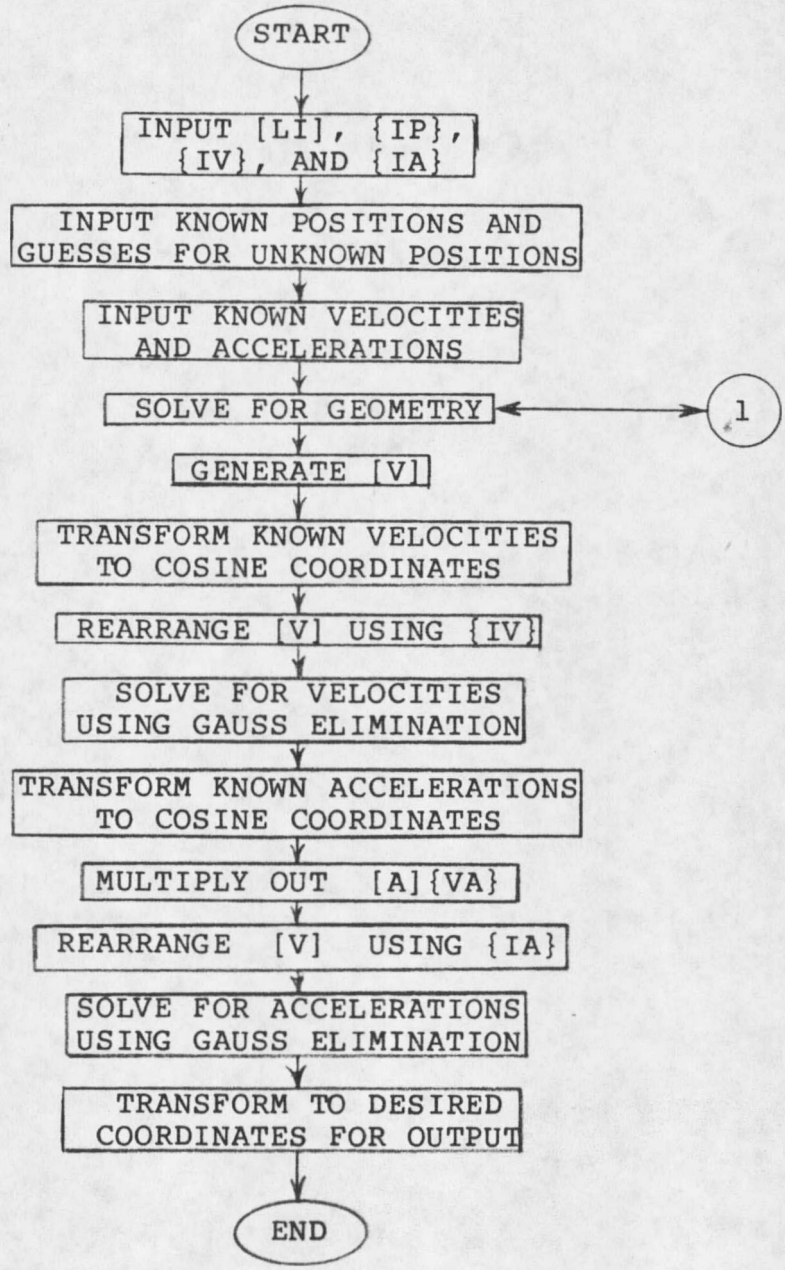


FIGURE 2 Flow Chart of Equation Assembly and Solution Procedure

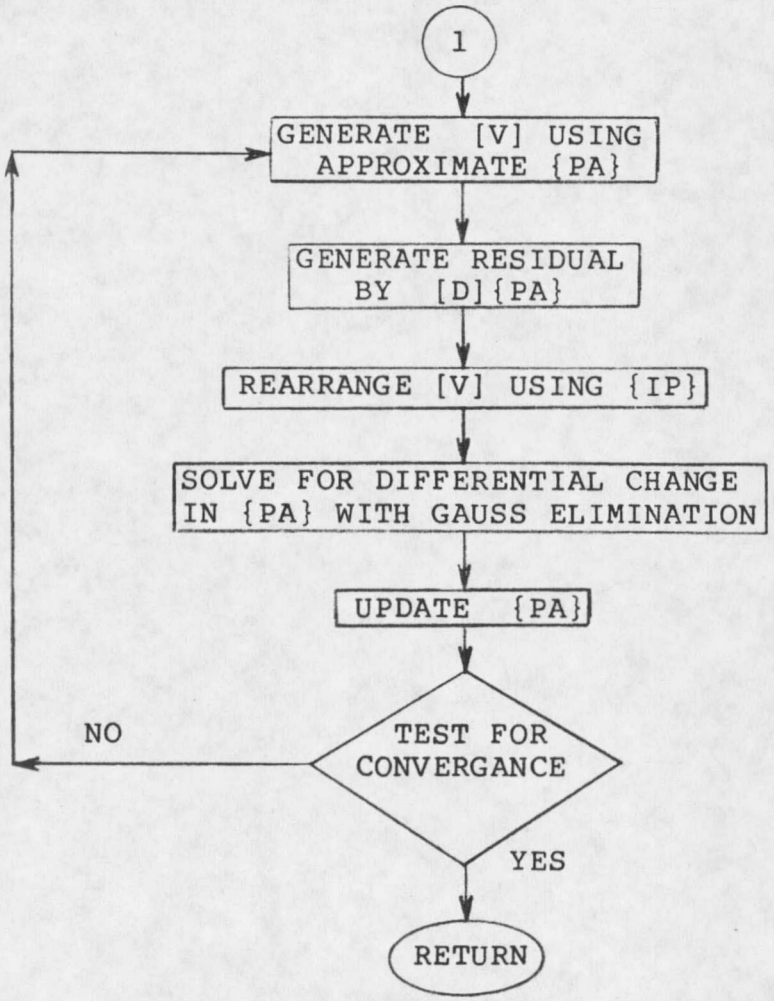


FIGURE 3 Flow Chart for Solution to Geometry using Newton's Method

for unknown positions, velocities, and accelerations. All other coefficient matrices can be reduced by matrix multiplication to a column array and moved to the right-hand side of the equations.

This method is quite general and can be used to solve for a variety of known and unknown parameters since the {IP}, {IV}, and {IA} arrays need not be identical. This procedure allows known geometry parameters to have either known or unknown derivatives.

CHAPTER IV

MODELING AND SYNTHESIS

Use of concise input to define a mechanism model for computer generation of the kinematic equations is an efficient way to perform kinematic analysis. The method used here to define a mechanism model is a vector path method. A vector path is composed of vectors which define the relative position of members, or locate points of interest in the mechanism. It is important to realize that a vector need not be on or define an actual physical link, but may be part of the model to constrain other vectors, locate points of interest, or to be used for synthesis. A mechanism model is defined by a complete set of vector paths. A loop is a path which starts and ends at the same point. The information defining paths and loops is stored in the [LI] and {PD} arrays defined in the previous chapter. The vector path method allows relatively concise input for model definition, yet is adaptable even to very complex mechanisms with complicated internal structures.

Two informal "rules" have been developed to aid in defining a complete set of vector paths when modeling a mechanism. These vector path rules are:

1. A vector path must be made around each "internal" loop in the mechanism. These loops have zero offset and

usually define a solid body other than a rod.

2. If one or more points in the mechanism are "grounded", i.e. attached to the Newtonian coordinate system, a single ground point is selected, and vector paths are generated from this point to every other ground point. The differences in the three Cartesian coordinates of the two points yield the corresponding terms in the path displacement array, {PD}.

Rods with ball joints on each end can be represented by a single vector, while more complicated members are defined by more than one vector. Rods with pinned joints need two additional vectors, one to represent the pin, and the other to define the angle the pin makes with the rod. A plate that has three other members connected to it is represented by the three vectors which form a closed path around the connection points. Complex three-dimensional members with more than three connection points can be modeled as several plates, each containing three vectors which form a loop thru three connection points.

Example 1

A vector model of a tripod, a three element spatial mechanism, is shown in Figure 4. Each of the three vectors shown in the model represents a rod with ball joints on each

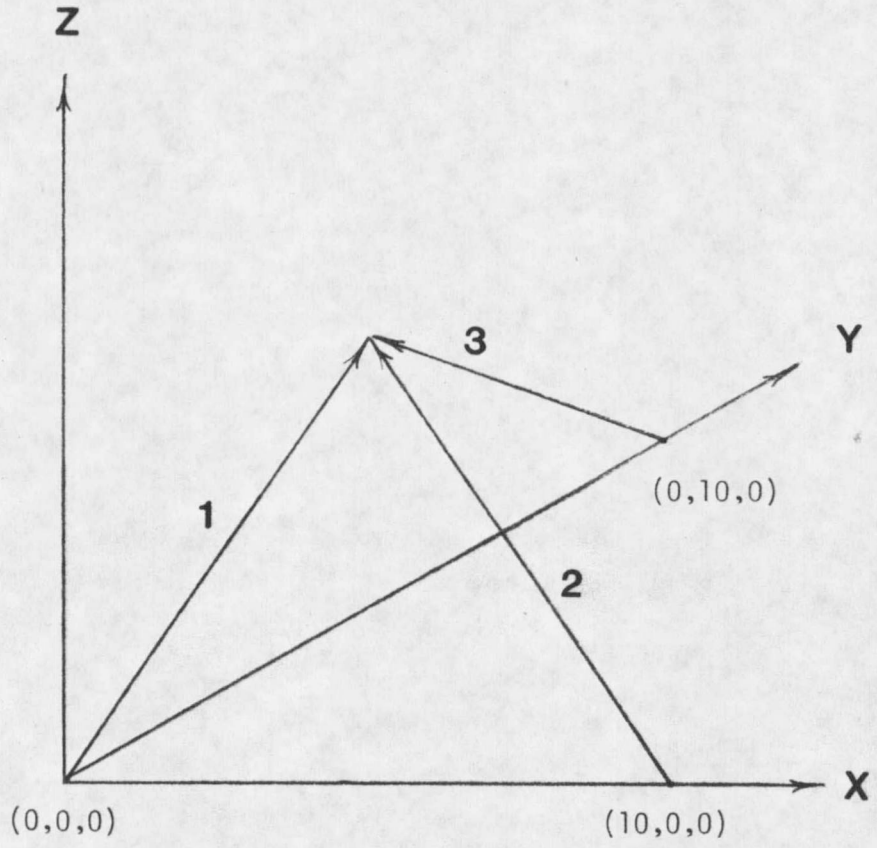


FIGURE 4 Vector Model of Tripod

end. This model contains two vector paths, one from the origin to the point (10,0,0) along vectors one and two, the other from the origin to the point (0,10,0) along vectors one and three. The [LI] information matrix for this configuration is then:

$$[LI] = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

The assembled [V] coefficient matrix from equation (3.4) for this model is:

$$[V] = \begin{bmatrix} [v_1] & -[v_2] & [0] \\ [v_1] & [0] & -[v_3] \\ c'_1 & 0 & 0 \\ 0 & c'_2 & 0 \\ 0 & 0 & c'_3 \end{bmatrix}$$

which is a system of nine equations in twelve parameters.

The [V] matrix is the only coefficient matrix which must be generated and stored as rearranging [V] according to the information in the {IP}, {IV}, and {IA} arrays generates the necessary coefficient matrices to solve for positions, velocities, and accelerations, respectively (see Figures 2 and 3).

Tables 1-3 give an example of the solution to the kinematics of the tripod mechanism when all three lengths

TABLE 1 TRIPOD POSITION SOLUTION

Underlined values are known inputs

Vector Number	Length (inches)	Angle with Axis (Degrees) X	Y	Z
1	10.00	47.16	73.74	47.34
2	8.00	113.58	69.51	32.11
3	12.00	55.48	126.87	55.62

TABLE 2 TRIPOD VELOCITY SOLUTION

Underlined values are known inputs

Vector Number	Length Velocity (in/sec)	Angular Velocities Relative To Coordinate Axis (rad/sec) X	Y	Z
1	1.00E+00	1.75E-01	-2.00E-01	-6.68E-02
2	2.00E+00	-2.73E-02	-2.00E-01	1.24E-02
3	-1.00E+00	3.37E-03	-1.67E-01	-1.75E-01

TABLE 3 TRIPOD ACCELERATION SOLUTION

Underlined values are known inputs

VECTOR NUMBER	LENGTH ACCELERATION (in/sec^2)	ANGULAR ACCELERATION RELATIVE TO COORDINATE AXIS (rad/sec^2) X	Y	Z
1	5.00E-01	-9.86E-02	8.46E-02	1.26E-01
2	-5.00E-01	-4.06E-02	1.15E-01	-6.33E-02
3	7.50E-01	-1.75E-02	-1.22E-02	4.52E-02

are known. Simply altering the {IP}, {IV}, and {IA} information matrices would allow solution to the kinematics problem with a different set of known parameters.

Example 2

The mechanical arm used on the space shuttle shown in Figure 5 [8], is a spatial mechanism with six degrees of freedom. The vector model of the arm is shown in Figure 6, with the [LI] information matrix for the model given in Table 4. This model of the arm consists of seventeen vectors arranged in nine paths. A total of forty four equations in sixty eight parameters must be solved for kinematic analysis. This model of the arm can be used to perform either analysis or synthesis of robotic motions of the arm depending on which twenty four parameters are specified as knowns in the {IP}, {IV}, and {IA} arrays.

For kinematic analysis of the space shuttle arm, the following parameters would typically be known, the lengths of all the vectors except six (a total of sixteen parameters), two of the direction cosines of vector one and the six degrees of freedom in the mechanism, for a total of twenty four known parameters. The six degrees of freedom in the mechanism can be specified in the model in the following manner:

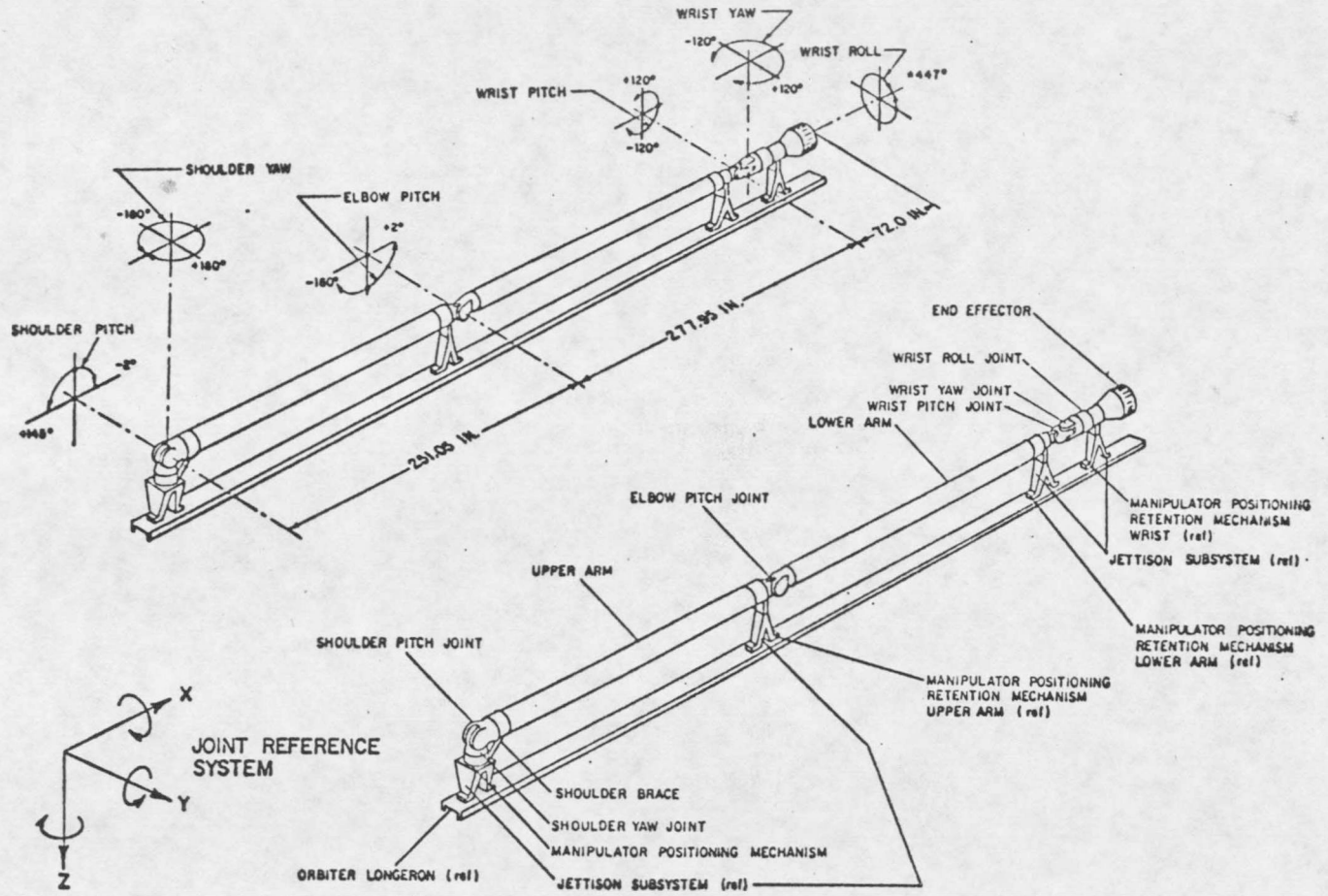


FIGURE 5 Space Shuttle Mechanical Arm

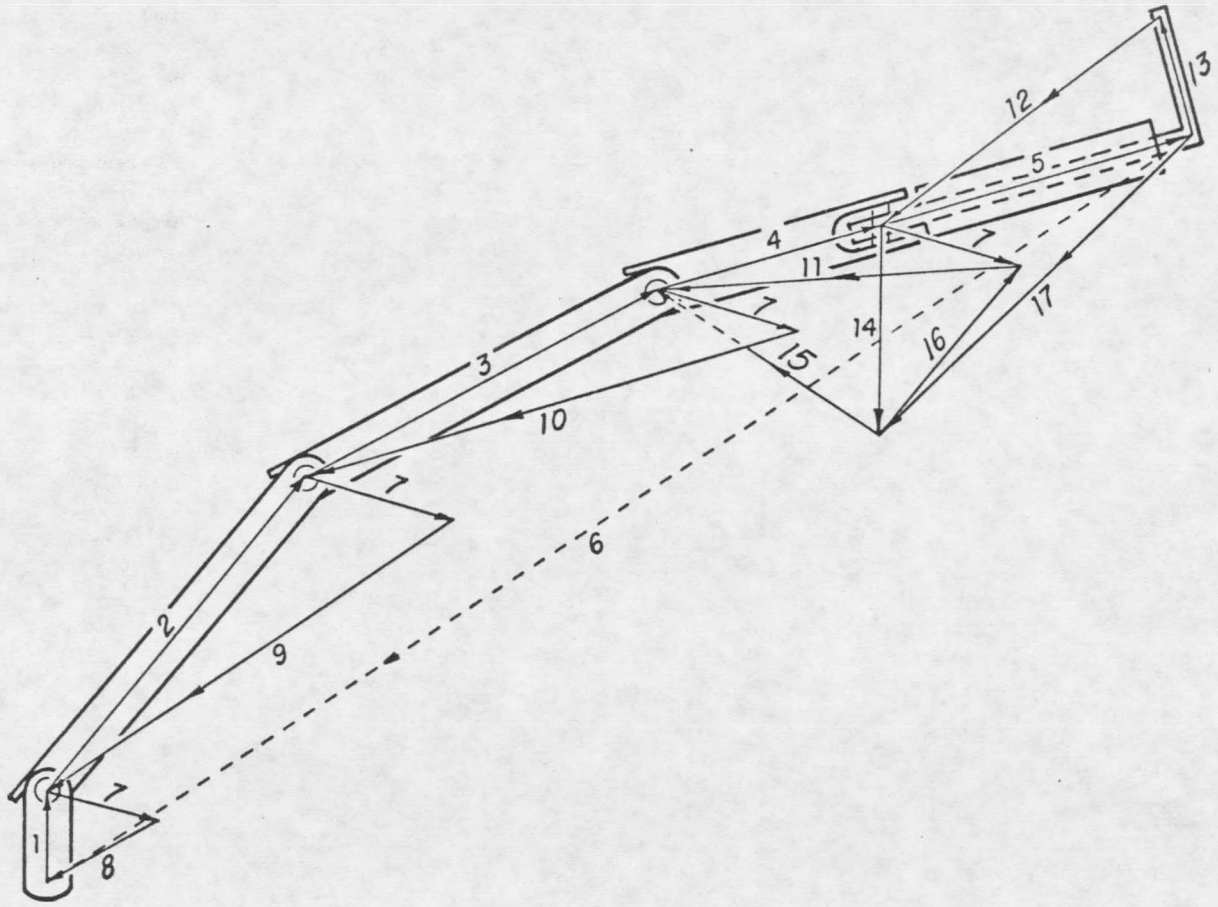


FIGURE 6 Vector Model of Shuttle Arm

TABLE 4 LI ARRAY VALUES FOR SHUTTLE ARM MODEL

Path Number	Vector Number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	LI Array Values																
1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
4	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0
6	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	1	0
7	0	0	0	0	1	0	0	0	0	0	0	0	0	-1	0	0	1
8	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0
9	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

TABLE 5 IP ARRAY VALUES FOR SHUTTLE ARM EXAMPLE

IP = 1 means unknown parameter value
IP = 0 means known parameter value

Vector Number		Direction Cosine Parameters		
	Length	X	Y	Z
1	0	0	0	1
2	0	1	1	0
3	0	1	1	0
4	0	1	1	0
5	0	0	1	1
6	1	1	1	1
7	0	1	0	1
8	0	1	1	1
9	0	1	1	1
10	0	1	1	1
11	0	1	1	1
12	0	1	1	1
13	0	1	0	1
14	0	1	1	1
15	0	1	1	1
16	0	1	1	1
17	0	1	1	1

Shoulder Pitch	-	Z angle of vector two
Shoulder Yaw	-	Y angle of vector seven
Elbow Pitch	-	Z angle of vector three
Wrist Pitch	-	Z angle of vector four
Wrist Yaw	-	X angle of vector five
Wrist Roll	-	Y angle of vector thirteen

These six parameters would then have known velocities and accelerations associated with them. The other known parameters would typically have known velocities and accelerations of zero.

The {IP} array for this set of known parameters is given in Table 5. For this example the {IV} and {IA} arrays were the same as the {IP} array although this was not necessary. Tables 6-8 give an example of position, velocity, and acceleration solutions for the shuttle arm model.

Synthesis of mechanism or servo coordinates of the arm is accomplished by specifying the length and two of the direction cosines for vector six and allowing any three of the degrees of freedom to be unknowns. This is accomplished by appropriate changes in the {IP}, {IV}, and {IA} matrices, with no need to alter any computer code.

TABLE 6 SHUTTLE ARM POSITION SOLUTION

Known Parameters are Underlined

Vector Number	Length (inches)	Angle with Axis (Degrees) X	Y	Z
1	36.00	90.00	90.00	180.00
2	251.00	8.07	92.97	82.50
3	287.00	11.40	92.95	79.00
4	36.00	40.09	92.30	130.00
5	36.00	40.00	85.87	129.70
6	576.75	177.38	87.40	90.37
7	100.00	87.00	3.00	90.00
8	106.28	92.82	159.98	70.20
9	270.19	159.91	108.75	96.97
10	295.44	160.07	106.84	100.34
11	106.28	107.96	157.82	77.43
12	50.91	121.65	139.06	66.90
13	36.00	91.37	5.00	85.19
14	36.00	50.07	91.93	40.00
15	50.91	174.17	87.01	95.00
16	106.28	99.68	18.01	105.04
17	50.91	95.04	94.29	6.62

TABLE 7 SHUTTLE ARM VELOCITY SOLUTION

Known Velocities are Underlined

Vector Number	Length Velocity (in/sec)	Angular Velocities Relative To Coordinate Axis (rad/sec)		
		X	Y	Z
1	0.00E+00	0.00E+00	0.00E+00	0.0E+00
2	0.00E+00	-1.13E-01	-4.89E-02	1.00E-01
3	0.00E+00	8.34E-02	-5.01E-02	-1.00E-01
4	0.00E+00	4.67E-02	-4.00E-02	5.00E-02
5	0.00E+00	1.00E-01	-8.95E-01	-3.06E-02
6	-6.75E+00	1.02E-01	1.03E-01	3.32E-03
7	0.00E+00	5.00E-02	-5.00E-02	-1.43E-07
8	0.00E+00	-4.70E-02	7.19E-03	1.43E-07
9	0.00E+00	-1.15E-02	4.89E-02	-9.28E-02
10	0.00E+00	-9.51E-02	5.01E-02	9.39E-02
11	0.00E+00	-6.01E-02	4.24E-02	-1.33E-02
12	0.00E+00	-1.42E+00	9.73E-01	-4.19E-01
13	0.00E+00	1.64E+00	-1.00E-01	5.70E-01
14	0.00E+00	-5.21E-02	-3.01E-02	5.00E-02
15	0.00E+00	6.85E-02	4.96E-02	-5.00E-02
16	0.00E+00	6.14E-02	2.50E-02	-1.13E-02
17	0.00E+00	-7.40E-02	6.12E-01	3.41E-01

TABLE 8 SHUTTLE ARM ACCELERATION SOLUTION

Known Acceleration Inputs are Underlined

Vector Number	Length Acceleration (in/sec^2)	Angular Accelerations Relative To Coordinate Axis (rad/sec^2)		
		X	Y	Z
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	0.00E+00	-1.11E-01	-5.07E-02	1.00E-01
3	0.00E+00	1.11E-01	-4.87E-02	-1.00E-01
4	0.00E+00	-4.93E-02	-3.35E-02	-5.00E-02
5	0.00E+00	5.00E-02	1.70E+01	9.22E-01
6	2.50E+01	-1.00E+00	-1.01E+00	-3.93E-02
7	0.00E+00	5.00E-02	-5.00E-02	-1.32E-07
8	0.00E+00	-4.70E-02	4.72E-02	1.33E-07
9	0.00E+00	-4.46E-02	5.07E-02	-9.03E-02
10	0.00E+00	-1.04E-01	4.87E-02	9.37E-02
11	0.00E+00	-3.76E-02	3.48E-02	1.38E-02
12	0.00E+00	-7.33E+00	-1.73E+01	-3.03E+01
13	0.00E+00	1.03E+01	1.00E-01	3.87E+01
14	0.00E+00	4.48E-02	-5.20E-02	-5.00E-02
15	0.00E+00	-1.49E-02	6.04E-02	5.00E-02
16	0.00E+00	3.48E-02	3.94E-02	1.06E-02
17	0.00E+00	-2.66E-03	-1.21E+01	-5.57E+00

CHAPTER V

CONCLUSIONS

The method of mechanism modeling presented allows a designer to write a general purpose program for the kinematic modeling of multi-degree of freedom spatial mechanisms. Once written this code can be used to perform kinematic analysis or synthesis of a wide variety of mechanisms with no alterations of the computer code. Only the data base which describes the mechanism and type of solution desired are problem specific.

The method presented uses the direction cosines and their time derivatives to define the kinematics of a mechanism. A pre- and post-processor program can be written to allow inputs and outputs to be in any convenient coordinate system or units.

The method presented has several advantages. It is well suited for multi-positional analysis, permits multiple degrees of freedom, can model very complex spatial mechanisms, permits analysis of different configurations, and can perform limited synthesis of multi-degree of freedom spatial mechanisms.

LITERATURE CITED

LITERATURE CITED

1. Zienkiewicz, O. C., The Finite Element Method in Engineering Science, London: McGraw-Hill Publishing Company, 1971.

2. Shigley, J. E., and J. J. Uicker, Theory of Machines and Mechanisms, New York: McGraw-Hill Publishing Company, 1980.

3. Paul, B., Kinematics and Dynamics of Planar Machinery, Englewood Cliffs: Prentice-Hall, Inc., 1979.

4. Chace, M. A., "Vector Analysis of Linkages", J. Eng. Ind., ASME Trans., ser. B, vol. 85, no. 3, pp. 289-297, 1963.

5. Garcia de Jalon, J., Serna, M. A., Viadero, F., and Flaquer, J., "A Simple Numerical Method for the Kinematic Analysis of Spatial Mechanisms", Journal of Mechanical Design, ASME Trans., vol 104, no. 1, pp 78-82, 1982.

6. Ortega, J. M., and Rheinbolt, W. C., Iterative Solution of Nonlinear Equations in Several Variables, New York: Academic Press, Inc., 1970

7. Hornbeck, R. W., Numerical Methods, New York: Quantum Publishers, Inc., 1975

8. Townes, H. W., Blackketter, D. O., and Lowell, J. M., "Robot Kinematics Using a Microcomputer", presented at Society for Computer Simulation Conference on Modeling and Simulation on Microcomputers, Jan. 28-30, 1982, San Diego, CA. Published in the proceedings.

Lowell, J. M.
Computer modeling of spatial mechanisms for kinematic analysis...

DATE	ISSUED TO
	D. JOHNSON 220 ROBERTS HALL 994-6306