

# AN ABSTRACT OF THE THESIS OF

Charles Eugene O'Kins for the M.S. in Mechanical Engineering  
(Name) (Degree) (Major)

Date thesis is presented 29 Sept 1964

Title THE DEVELOPMENT OF ANALYTIC TECHNIQUES AND  
FORTRAN PROGRAMS FOR THE EVALUATION OF PLANE  
MOTION MECHANISMS

Abstract approved Redacted for Privacy  
(Major professor)

A mechanism is a constrained kinematic chain composed of gears, links, cams, or the like. Mechanisms are the building blocks of all machines, and, as such, their evaluation is of considerable importance to the mechanical designer. Because the mathematical analyses of systems with four or more moving members has been prohibitively complex, the principal method of mechanism evaluation has been graphical. With the development and widespread distribution of high-speed digital computer systems, however, mathematical methods for complex-linkage evaluation have become practicable. Because of the somewhat universal nature of the digital computer language called FORTRAN, it is possible for an analyst to develop a system of analysis which not only he, but any other person as well, may use.

In this paper are developed analytic techniques and computer programs which evaluate the principal class of mechanisms -- plane motion linkages with a single degree of freedom, employing turning joints, and having either an angular or translational input motion. The fundamental premise is that a link may be represented by a complex vector, and a linkage may be represented by a set of these vectors in the form of closed polygons. The position vectors, which are known and are considered to be functions of time, sum to zero about a closed path. There exist, in a single degree of freedom linkage, one-half as many independent closed paths as  $n$ -links free to rotate (i. e., links which are neither the crank nor the frame). Therefore, one-half- $n$  independent vector sums may be written. By separating the sums into their real and imaginary parts,  $n$  independent equations result. The first and second time-derivatives of the  $n$ -set provides two  $n$ -sets of linear algebraic equations. These two sets are solved, by the computer, for the  $n$ -unknown angular rates and the  $n$ -unknown angular accelerations. With these values of angular kinematic quantities, the computer estimates the angular rotations over a particular time interval. Through use of a simple iterative process, these estimated angular values are improved. If the angular transition is not large, convergence is rapid. After calculation of angular rates and accelerations in this new position, the process is repeated, as before, until the desired range of operation of the linkage has been traversed.

Having determined the linkage's angular kinematic values for a particular position (or instant of time), the computer uses the values to calculate the translational kinematic data of any specified point on the linkage. The translational values are placed in an absolute reference by using the drive-member frame pin as the datum and summing the translational vector components to the desired point.

The mathematical system of equations and logic are validated by the successful evaluation of an eleven-bar, ram-drive linkage system.

It is acknowledged that these methods and the associated computer programs, although a significant improvement over widely practiced linkage evaluation methods, are but the first step in the use of the digital computer in mechanical design. Work must also be done to develop computer methods to assist the mechanical designers with dynamic evaluation, stress analysis, bearing loading, space budgeting, force analysis, and so forth.

THE DEVELOPMENT OF ANALYTIC TECHNIQUES AND  
FORTRAN PROGRAMS FOR THE EVALUATION OF  
PLANE MOTION MECHANISMS

by

CHARLES EUGENE O'KINS

A THESIS

submitted to

OREGON STATE UNIVERSITY

in partial fulfillment of  
the requirements for the  
degree of

MASTER OF SCIENCE

June 1965

APPROVED:

Redacted for Privacy

---

Associate Professor of Mechanical Engineering

In Charge of Major

Redacted for Privacy

---

Head of Mechanical and Industrial Engineering Department

Redacted for Privacy

---

Dean of Graduate School

Date thesis is presented: 29 Sept 1964

Typed by Carol Baker

## TABLE OF CONTENTS

	Page
INTRODUCTION	1
LINKAGE SYMBOLISM AND DEGREES OF FREEDOM	4
DEVELOPMENT OF ANALYTIC TECHNIQUES	9
Complex Vectors Representing Linkages	9
Angular Kinematics of Four-bar Crank Linkages	13
Mathematical Singularities	18
Translational Kinematics of Four-bar Crank Linkage	22
KINEMATIC EVALUATION OF COMPLEX LINKAGES	26
Drawing the Vector Polygon	26
Vector Loops and Degrees of Freedom	27
Writing Equations of Angular Kinematics	30
Determining Translational Kinematics	34
THE COMPUTER PROGRAM TO PERFORM KINEMATIC EVALUATION	36
The Closure Program	39
The Basic Program	42
The Matrix Evaluation Subroutines	50
Subroutine ACOE	53
Subroutine BCOE	56
Subroutine GAUSS	62
The Angle Computation Subroutine	64
Translational Kinematics Subroutine	65
VALIDATING TEST CASES	70
Eleven-bar Linkage Test Case	70
Required Data Cards	72
Results of Computer Analysis	74
The Mathematical Singularity Test Cases	83

## TABLE OF CONTENTS (con't)

	Page
Type I Singularity	83
Type II Singularity	86
CONCLUDING REMARKS	89
BIBLIOGRAPHY	92
APPENDICES	93
Appendix A. Program CLOSURE	93
Appendix B. Program MAIN	94
Appendix C. Subroutine ACOE	95
Appendix D. Subroutine BCOE	96
Appendix E. Subroutine GAUSS	97
Appendix F. Subroutine ANGLE	98
Appendix G. Subroutine KINE	98

## LIST OF FIGURES

Figure	Page
1. Links showing degrees of freedom	5
2. Systematic reduction of degrees of freedom	5
3. A linkage showing polygon loops	7
4. (a). A link and its vector representation	10
(b). A linkage and its vector polygon representation	10
5. Complex vector in x-y plane	11
6. Vectors serially added	12
7. A four-bar linkage and its vector equivalent showing a vector loop $L$	14
8. Type I singularity	19
9. Type II singularity	20
10. Four-bar crank linkage with coupler points D and E	24
11. The vector polygon representing a single degree of freedom, eleven-bar linkage with a ram drive	28
12. Schematic flow diagram for program MAIN	43
13. Schematic flow diagram for subroutine BCOE	57
14. Schematic flow diagram for subroutine GAUSS	63
15. Schematic flow digaram for subroutine KINE	66
16. Eleven-bar linkage vector diagram	71



## LIST OF TABLES

Table		Page
I.	Vector loop table to show loop independence	30
II.	Vector loops and mechanism links related through signs and subscripts. The table values are called NORDER(I, J) in the program	54
III.	Eleven-bar linkage data	75
	Part 1. Input data key, input data, output data key	75
	Part 2. Output data, position one	75
	Part 3. Output data, position two	76
	Part 4. Output data, position nine	76
	Part 5. Output data, position 17	77
	Part 6. Output data, position 18	77
IV.	Type I singularity output data	85
V.	Type II singularity output data	87

# THE DEVELOPMENT OF ANALYTIC TECHNIQUES AND FORTRAN PROGRAMS FOR THE EVALUATION OF PLANE MOTION MECHANISMS

## INTRODUCTION

A mechanism is a constrained kinematic chain in which one link is stationary or considered so during motion analysis. A kinematic chain is any connected group of elements, such as gears, links, or cams, whose parts have motion(3, p. 3). These mechanisms are the building blocks of any and all machines. Because of this, mechanism analysis is of primary concern to the mechanical designer.

Mechanism analysis has two fundamental forms: graphics and mathematics. Graphics is by far the most widely used, particularly for mechanisms of more than four moving members. For example, in Faïres' text Kinematics, he says, "Mathematical analyses of linkages with four or more [movable] links are usually rather complex and have been traditionally avoided"(3, p. 114). Even manufacturing firms which have specialized in products requiring complex linkages perform most of their design and evaluation on a drafting board.

Graphical solutions to linkage design problems, although quite common and frequently effective, have several limitations. First, the designer must be well versed in the specialized techniques, a requirement necessitating a great deal of experience. Secondly, graphical methods have obvious accuracy limitations. Finally,

performing the graphical evaluation of a complex linkage is usually quite time consuming, resulting in considerable expense and competitively dangerous delays. The effect of these limitations is that designers tend to avoid the complex design and, instead, select a simpler design which may only crudely approximate the desired performance.

Examples of this are pandemic. A particular, but typical, case was presented to the Oregon State University's Engineering Experiment Station. An Oregon firm was engaged in the manufacture of a product of international distribution. Several other firms manufactured a similar product and all employed the same fundamental linkage design basic to the product. The simple linkage, although traditional, had several undesirable operating features. This linkage design was retained, apparently, because any improved linkage would also have been more complex. The Oregon firm<sup>1</sup> decided to develop a greatly improved, but more complex, linkage design. After considerable time and expense, the company could see quite clearly that the traditional graphical methods of design and evaluation were insufficient. They formally asked the Experiment Station if a mathematical technique of complex-linkage evaluation could be developed which could also be programmed for computer analysis.

---

<sup>1</sup> Company name and product withheld upon request.

As an answer to this specific inquiry, and to contribute to the more general need of linkage designers, this paper has been prepared. In subsequent sections, fundamental linkage theory will be discussed. This theory will be limited to that which will lead to the development of a mathematical technique of linkage evaluation using well-known vector properties. After the development of the mathematics, digital computer programs, written in FORTRAN for the IBM 1410 system, will be presented and explained. The program presented will be for use with linkages having turning joints, a single degree of freedom, up to ten moving members, and either a crank or ram/ piston drive link. It is believed that this is no strict limitation and will encompass the vast majority of linkage designs. The program presentation will be supplemented with the computer solution of several significant test cases, the most important being a ten-moving-bar system with a ram input. Finally, suggestions will be made as to how the mathematics and computer programs could be expanded to cover a wider range of problems surrounding linkage design.

## LINKAGE SYMBOLISM AND DEGREES OF FREEDOM

Mechanically speaking, a link is a device by which motion is physically transmitted from a source to an object and is symbolically represented by a bar or line. For most mechanisms, the link is actually a rigid bar, but, even when it is not, the symbolism is usually adequate. A system of interconnected links is called a linkage. Common to all linkages is a driver which is the source of motion. Some linkages have more than one driver, but this is not common and, frequently, is a result of two linkage systems being brought together. Mathematically at least, the system can be separated into two distinct systems with one driver for each. For instance, consider a fork-lift truck lifting linkage: it has a vertical drive and a fork tilt drive, but the two are (usually) independent and can be considered separately. The number of drivers required is equal to the system's degrees of freedom.

The degrees of freedom describe the number of independent motions a body may have. A rigid bar in space has six -- three translational and three rotational. Restricting the body's motion to a plane reduces the degrees of freedom to three -- two translational and one rotational. An unconstrained, plane motion, rigid link then has three degrees of freedom. Connecting two such rigid links by a turning joint leaves one of the links with its three degrees unaffected

but reduces the other's degrees of freedom to one, as shown in Figure 1(b).

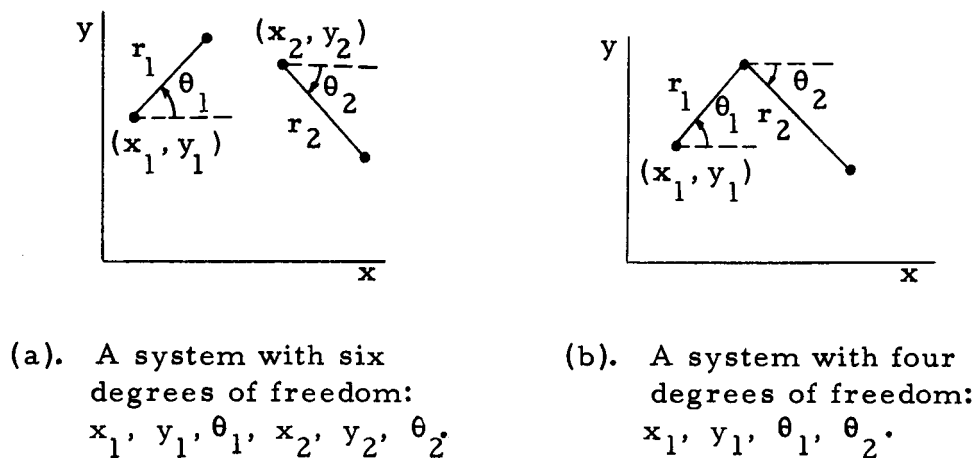


Figure 1. Links showing degrees of freedom.

By adding two more rigid links to the system of Figure 1(b) it becomes a four bar system with six degrees of freedom -- two translational and four angular. See Figure 2(a).

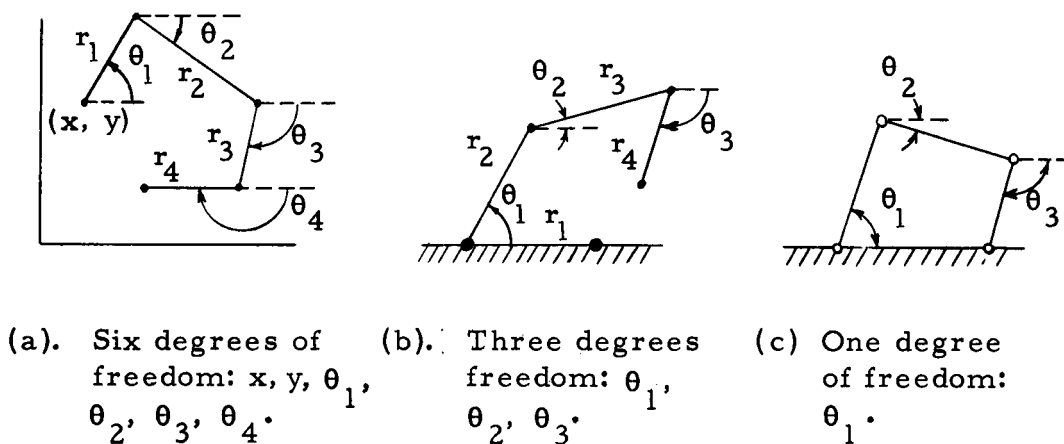


Figure 2. Systematic reduction of degrees of freedom.

For the system to be a linkage, one bar, by definition, must be completely constrained, i. e. one bar is made the frame. With this act, three degrees of freedom are lost, reducing the total from six to three. See Figure 2(b). Next, closing the system by pinning the free end of the link chain to the frame, reduces the degrees of freedom, as before, by two, leaving a system total of one degree. The result is the traditional four-bar linkage. This method of link-by-link study of freedom and constraint will provide an answer but, for more complex systems, is quite laborious and subject to human error. It is presented here only to provide understanding of the concept of degrees of freedom.

Authors on mechanism use a variety of methods and formulas to determine a system's degrees of freedom. One such is Gruebler's equation as modified by Bottema (1, p. 162) :

$$X = 3(N-1) - \sum q_i \cdot p_i$$

where  $X$  = degrees of freedom

$N$  = number of links

$q_i$  = multiplicity of constraint

$p_i$  = number of pins with a  $q_i$  multiplicity of constraint

The multiplicity of constraint  $q_i$ , although not defined further by Bottema, appears to be, for turning joints at least,

$$q_i = 2(M-1)$$

The term  $M$  is the multiplicity of connection of a particular pin; e. g. a pin connecting three bars has a multiplicity of three. Gruebler's equation is modified to:

$$X = 3(N-1) - \sum 2(M_i - 1)p_i$$

An alternate method suggested by Paul (6, p. 196) is based on a topological analysis of the linkage system and does not require the determination of the number of pins or multiplicity. Instead, the analyst determines, in addition to the moveable links ( $N - 1$ ), the number of independent loops which can be made in a linkage system. A loop, in this sense, is the circumscription of a polygon formed by a group of serially connected links. For independence, the loop must not be capable of being formed by combining two other loops. In Figure 3, for example, only two independent loops

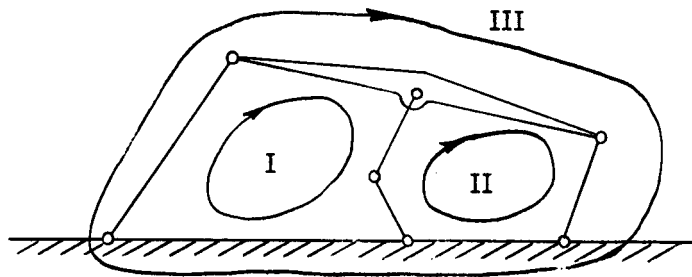


Figure 3. A linkage showing polygon loops.

are possible, I and II, I and III, or II and III. The combination I, II, and III can not be used because any two can form the third.



Paul's equation for the degrees of freedom is:

$$X = (N - 1) - 2L$$

where  $L$  is the number of independent loops.

Later in this report, Paul's equation will be modified and used to determine the number of possible independent loops in a complex linkage once Gruebler's equation has determined the degrees of freedom.

## DEVELOPMENT OF ANALYTIC TECHNIQUES

The subsequent mathematical development will be a utilization of the concept of defining links by complex vectors as first put forth by Block in 1940 (3, p. 249) who suggested its use in four-bar synthesis. To date, his method has been mostly restricted to its original usage. In this paper, however, it will be developed for kinematic evaluation -- first for the simple case of the four-bar linkage where the validity will be obvious, and then for the complex case.

### Complex Vectors Representing Linkages

A mechanical link is an object whose kinematically important dimensions are its distance (or distances) between joints and angular direction (or directions). A vector has only magnitude and direction, as does the distance between two pins on a link. A link then is quite conveniently represented by a vector; a linkage is conveniently represented by a collection of vectors. See Figure 4.

The magnitude of the vector can appropriately have a one-to-one correspondence to the distance between the link pins. A vector has a distinct direction, but, its direction magnitude (argument) is arbitrary. Its sense (which way the arrowhead points) is a matter of convenience, as is the reference line from which the angle is

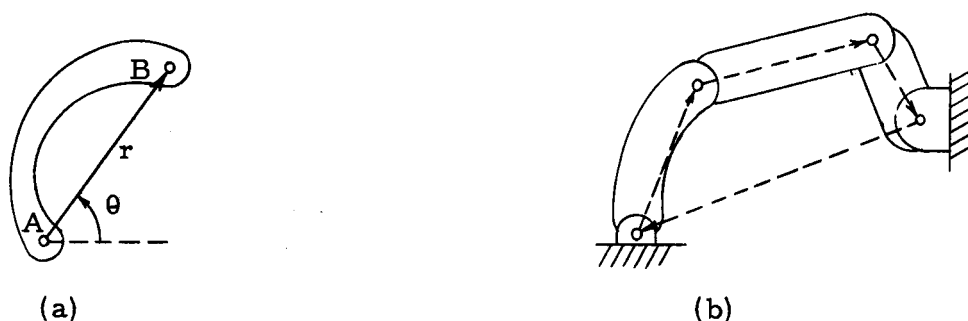


Figure 4. (a). A link and its vector representation.

(b). A linkage and its vector polygon representation.

measured. The one restriction is that all vectors representing links within a single system must have the same reference. Traditionally, the "horizontal" is made the angular datum and angles are measured counterclockwise from the right. That would place the vector  $\overline{AB}$ , in Figure 4 (a), somewhere in the first quadrant.

The linkage type of interest in this report is the one operating in a single plane; vectors restricted to a single plane can be mathematically described by complex vector notation. Mathematically, the complex vector is defined as (1) the sum of two orthogonal components, one real and one imaginary, or (2) as the product of a real magnitude and the Napierian base  $e$  exponentiated by the imaginary unit  $i$  and the vector direction.

$$z = x + iy \quad (1)$$

$$z = re^{i\theta} \quad (2)$$

Where  $x$  and  $y$  are the usual cartesian coordinates,  $i$  is the imaginary element,  $r$  is the vector magnitude, and  $\theta$  is the vector direction, in radians.

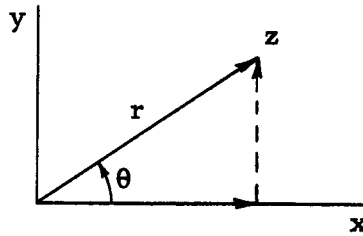


Figure 5. Complex vector in x-y plane.

Referring to Figure 5 and using standard definitions:

$$x = r \cos \theta \quad (3)$$

$$y = r \sin \theta \quad (4)$$

Then, from (1), (2), (3), and (4)

$$z = r \cos \theta + i r \sin \theta = r e^{i\theta} \quad (5)$$

Given a closed string of vectors as in Figure 6, it is easy to see that:

$$z_1 = x_1 + i y_1$$

$$z_2 = (x_2 - x_1) + i(y_2 - y_1)$$

$$z_3 = (x_3 - x_2) + i(y_3 - y_2)$$

$$\text{and } z_4 = x_3 + i y_3$$

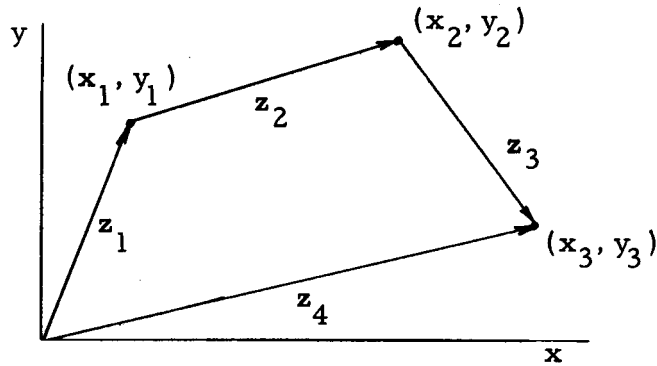


Figure 6. Vectors serially added.

If  $z_2$  were added to  $z_1$ ,  $z_3$  to  $z_2$ , and  $(-) z_4$  to  $z_3$ , the closed path would have been traversed and the sum would be:

$$\begin{aligned}
 \sum_i z_i &= z_1 + z_2 + z_3 + (-) z_4 \\
 &= x_1 + iy_1 + (x_2 - x_1) + i(y_2 - y_1) \\
 &\quad + (x_3 - x_2) + i(y_3 - y_2) - x_3 - iy_3 \\
 \sum_i z_i &= 0
 \end{aligned}$$

By inductive reasoning it can be seen that vector addition over any closed path, regardless of the number of elements, sums to zero. Next, consider the situation in which the complex vectors  $z_i$  are functions of time, and for all time represent the mechanism, such that,

$$\sum_i z_i(t) = 0 \tag{6}$$

Except under certain special conditions to be discussed later, the

derivatives of  $z_i$ , with respect to time, exist. These must also sum to zero.

$$\sum_i \frac{d^{(n)}}{dt^{(n)}} z_i = 0 \quad (7)$$

Equations (6) and (7) are the basis of the analytic techniques and computer programs to be developed. Their rigorous proof is of only didactical importance. In this paper it will be assumed they are true and will be validated by successful application to linkage analysis.

For a more detailed introduction to complex vectors see references (2, p. 1) and (7, p. 527).

### Angular Kinematics of Four-bar Crank Linkages

The linkage of Figure 7(a) is the traditional four-bar system. It is the fundamental form for a single degree of freedom linkage. It can be shown that simpler or more complex systems are merely modifications of this basic form. For this reason, its somewhat obvious properties will be developed first. Following this will be the application of the same properties to a more complex system. Figure 7(b) shows the four-bar vector equivalent and the vector loop  $L$ . The vector loop  $L$  is mathematically the sum of elements of the closed system it traverses; i. e.

$$L = z_1 + z_2 - z_3 - z_4 = 0$$

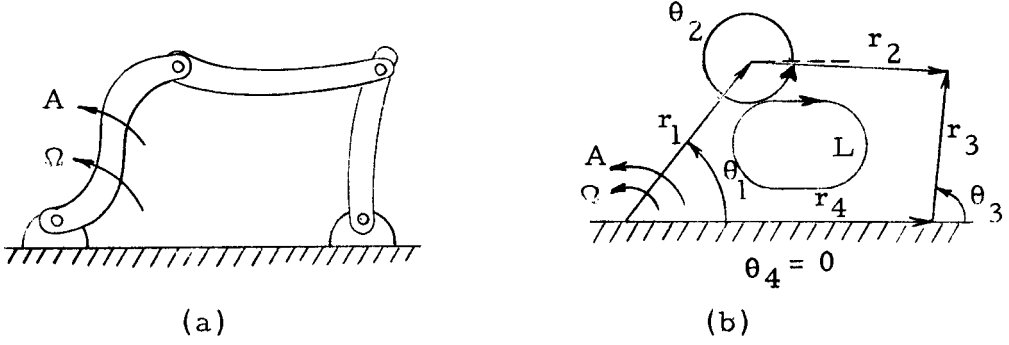


Figure 7. A four-bar linkage and its vector equivalent showing a vector loop  $L$ .

$$\text{from (2), } L = r_1 e^{i\theta_1} + r_2 e^{i\theta_2} - r_3 e^{i\theta_3} - r_4 e^{i\theta_4} = 0$$

$$\begin{aligned} \text{from (5), } L = & r_1 (\cos\theta_1 + i\sin\theta_1) + r_2 (\cos\theta_2 + i\sin\theta_2) \\ & - r_3 (\cos\theta_3 + i\sin\theta_3) - r_4 (\cos\theta_4 + i\sin\theta_4) = 0 \end{aligned} \quad (8)$$

Differentiating equation (8) with respect to time and equating like coefficients (for convenience in writing let  $S_i = \sin\theta_i$  and  $C_i = \cos\theta_i$ ) gives:

$$\begin{aligned} \frac{dL}{dt} = & r_1 (-S_1 + iC_1) \frac{d\theta_1}{dt} + r_2 (-S_2 + iC_2) \frac{d\theta_2}{dt} - r_3 (-S_3 + iC_3) \frac{d\theta_3}{dt} \\ & - r_4 (-S_4 + iC_4) \frac{d\theta_4}{dt} = 0 \end{aligned}$$

Then

$$r_1 S_1 \omega_1 + r_2 S_2 \omega_2 - r_3 S_3 \omega_3 - r_4 S_4 \omega_4 = 0 \quad (9)$$

and

$$r_1 C_1 \omega_1 + r_2 C_2 \omega_2 - r_3 C_3 \omega_3 - r_4 C_4 \omega_4 = 0 \quad (10)$$

where the  $\frac{d\theta_i}{dt}$  have been represented by  $\omega_i$ . Since the driver's angular velocity,  $\Omega$ , is equal to  $\omega_1$  and  $\omega_4$  is zero, equations (9) and (10) can be written as:

$$r_2 S_2 \omega_2 - r_3 S_3 \omega_3 = -r_1 S_1 \Omega \quad (11)$$

$$r_2 C_2 \omega_2 - r_3 C_3 \omega_3 = -r_1 C_1 \Omega \quad (12)$$

It is presumed that the vector magnitudes (which in this example are constant) and original directions are known; therefore, equations (11) and (12) provide two equations to solve for the two unknown angular rates,  $\omega_2$  and  $\omega_3$ .

By similar differentiation of (11) and (12), two more equations can be written:

$$r_2 (S_2 a_2 + C_2 \omega_2^2) - r_3 (S_3 a_3 + C_3 \omega_3^2) = -r_1 (S_1 A + C_1 \Omega^2) \quad (13)$$

$$r_2 (C_2 a_2 - S_2 \omega_2^2) - r_3 (C_3 a_3 - S_3 \omega_3^2) = -r_1 (C_1 A - S_1 \Omega^2) \quad (14)$$

where

$$a_i = \frac{d\omega_i}{dt}.$$

In these equations,  $\omega_2$  and  $\omega_3$  are presumed known from (11) and (12), and  $A$ , the angular acceleration of the driver, is set equal to zero (i. e.  $\Omega = \text{constant}$ ). Equations (13) and (14) then may be rewritten as:

$$r_2 S_2 a_2 - r_3 S_3 a_3 = -r_1 C_1 \Omega^2 - r_2 C_2 \omega_2^2 + r_3 C_3 \omega_3^2 \quad (15)$$

$$r_2 C_2 a_2 - r_3 C_3 a_3 = r_1 S_1 \Omega^2 + r_2 S_2 \omega_2^2 - r_3 S_3 \omega_3^2 \quad (16)$$



From (15) and (16), the unknown angular accelerations  $\alpha_2$  and  $\alpha_3$  can be computed.

The links having angular velocity and acceleration, not all of which are zero, will move in time to new relative positions. Determining these new positions is the next step in the evaluation; for this purpose, extrapolated approximations will be made. The approximating assumption is that, over reasonably small increments of time, angular acceleration may be considered constant.

By definition,

$$\frac{d\theta}{dt} = \omega$$

and

$$\frac{d\omega}{dt} = \alpha$$

then

$$\int_{\omega_0}^{\omega} d\omega = \int_{t_0}^t \alpha dt$$

assuming

$$\alpha = \text{constant} = \alpha_0$$

$$\omega = \alpha_0(t - t_0) + \omega_0$$

Solving for  $\theta$

$$\int_{\theta_0}^{\theta} d\theta = \int_{t_0}^t \omega dt = \int_{t_0}^t [\alpha_0(t - t_0) + \omega_0] dt$$

and letting  $t - t_0 = \Delta t$ , then

$$\theta = \alpha_0 \frac{\Delta t^2}{2} + \omega_0 \Delta t + \theta_0 \quad (17)$$

To gain insight as to the amount of initial error introduced by assuming constant angular acceleration over  $\Delta t$ , a comparison can be made to the value of  $\theta$  obtained by assuming  $\alpha$  changes linearly with time. Let  $\alpha$  be the angular acceleration at the end of the interval  $\Delta t$  and let  $\alpha_0$  be the initial. Then, by taking the average of the two, rewrite (17) as

$$\theta = \frac{(\alpha + \alpha_0)}{2} \frac{\Delta t^2}{2} + \omega_0 \Delta t + \theta_0 \quad (18)$$

The difference between (18) and (17) is:

$$\theta_\epsilon = (\alpha - \alpha_0) \frac{\Delta t^2}{4}$$

Setting the time increment at 0.5 seconds produces:

$$\theta_\epsilon = \frac{1}{16} (\alpha - \alpha_0)$$

Setting  $\Delta t$  to 0.1 second produces:

$$\theta_\epsilon = \frac{1}{400} (\alpha - \alpha_0)$$

Quite obviously, even large differences between successive values of angular acceleration will have little effect if the time increment is made small enough. Even this error, however, is overcome to a large extent if, after calculating new angular rates, the new rates are used to obtain better approximations of angles. The equation suggested for iterative convergence after the initial approximation is:

$$\theta = (\omega + \omega_0) \frac{\Delta t}{2} + \theta_0 \quad (19)$$

Note that if the angular acceleration is linear, then

$$\omega = (\alpha + \alpha_0) \frac{\Delta t}{2} + \omega_0$$

then

$$\theta = [(\alpha + \alpha_0) \frac{\Delta t}{2} + 2\omega_0] \frac{\Delta t}{2} + \theta_0$$

producing equation (18) again. This means that, in the limit, equation (19) would yield the same result as would be obtained if new values of  $\alpha$  were known and equation (18) were used.

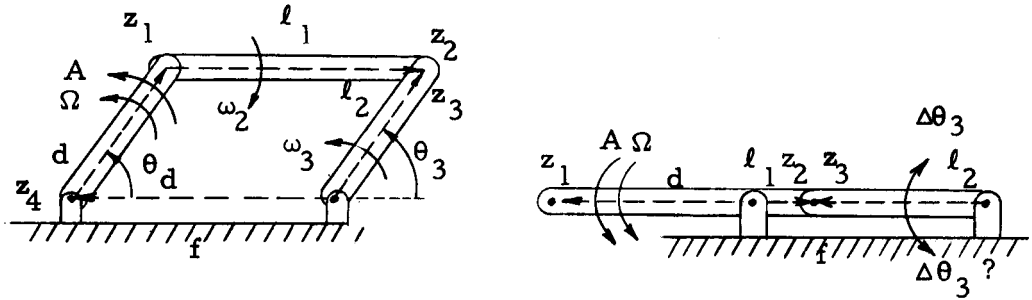
Since, however, new correct values of  $\alpha$  can not be computed without knowing correct values of  $\omega$ , there is no convenient alternative to using (19), after using (17) for the first approximation.

When iteration has brought about the desired convergence of the angular values, new values of  $\omega$  and  $\alpha$  can be computed. It would seem that the linkage cycling process could be repeated indefinitely, however, under certain conditions mathematical singularities appear.

### Mathematical Singularities

Singularities in linkage analysis occur when a link is in a kinematic state which allows it, mathematically, to have zero or two possibilities for the next step; i. e., the time derivatives of  $\sum z$  are not continuous. As an example of two possibilities for the next

state, consider the Type I singularity in Figure 8. The driver  $d$  and the link  $\ell_2$  have the same magnitude and direction. Motion



(a). A four-bar linkage in a non-singular state.

(b). A four-bar linkage in a singular state where there may be two possibilities for the next positions of some links.

Figure 8. Type I singularity.

of the driver uniquely defines the motion of the followers

$\ell_1$  and  $\ell_2$ . When  $d$  has progressed to  $\theta_d = 180^\circ$ ,  $\theta_3$  also is  $180^\circ$ , and all links are aligned. At this point, an incremental counterclockwise change in  $\theta_d$  can, with equal likelihood, cause  $\theta_3$  to become smaller or larger; i. e.  $\ell_1$  may remain horizontal and  $\ell_2$  continue to rotate counterclockwise, or  $\ell_1$  may rotate, driving  $\ell_2$  clockwise. In the equation of motion, this condition appears as a division by zero.

Summing position vectors:

$$\begin{aligned} z_1 + z_2 - z_3 - z_4 &= 0 \\ r_1 C_1 + r_2 C_2 - r_3 C_3 - r_4 C_4 &= 0 \\ r_1 S_1 + r_2 S_2 - r_3 S_3 - r_4 S_4 &= 0 \end{aligned}$$

Writing the velocity equations:

$$\begin{aligned} r_2 S_2 \omega_2 - r_3 S_3 \omega_3 &= -r_1 S_1 \Omega \\ r_2 C_2 \omega_2 - r_3 C_3 \omega_3 &= -r_1 C_1 \Omega \\ \therefore \omega_3 &= \frac{r_1 \Omega}{r_3} \frac{(S_2 C_1 - C_2 S_1)}{(S_2 C_3 - C_2 S_3)} \end{aligned}$$

Noting that, in Figure 8(b):

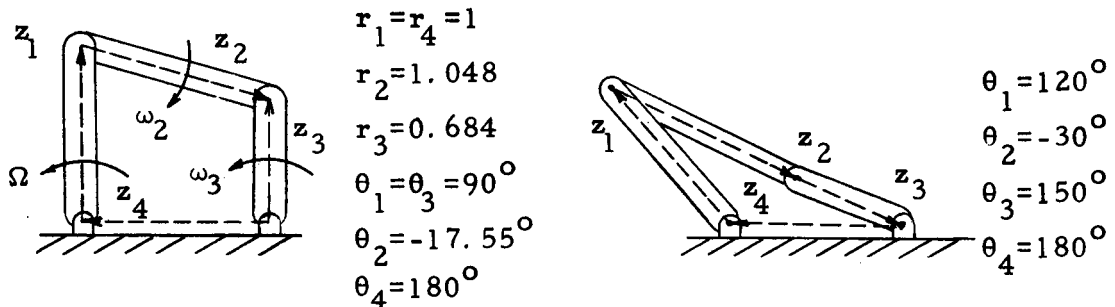
$$\begin{aligned} \theta_1 &= 180^\circ, \quad C_1 = -1, \quad S_1 = 0 \\ \theta_2 &= 0^\circ, \quad C_2 = 1, \quad S_2 = 0 \\ \theta_3 &= 180^\circ, \quad C_3 = -1, \quad S_3 = 0 \end{aligned}$$

then:

$$\omega_3 = \frac{r_1 \Omega}{r_3} \left( \frac{0}{0} \right)$$

This condition will occur whenever any two connected binary links become aligned, (a condition sometimes called dead center) and where neither is a driver.

For a singularity resulting from zero possibilities for the next position, consider the type II singularity in Figure 9.



(a). A four-bar mechanism in a non-singular state. (b). A four-bar mechanism in a singular state where there are no possibilities for the next position of any link.

Figure 9. Type II singularity.

In Figure 9(a), the linkage is obviously in a non-singular state, but in Figure 9(b), links 2 and 3 are aligned and have the angles noted in the figure. The sines and cosines are

$$C_1 = -0.5, \quad S_1 = 0.866$$

$$C_2 = 0.866, \quad S_2 = -0.5$$

$$C_3 = -0.866, \quad S_3 = 0.5$$

The equation for  $\omega_3$  is the same as the previous case; i. e.

$$\omega_3 = \frac{r_1 \Omega}{r_3} \frac{(S_2 C_1 - C_2 S_1)}{(S_2 C_3 - C_2 S_3)}$$

then

$$\omega_3 = \frac{r_1 \Omega}{r_3} \frac{(-0.5)(-0.5) - (0.866)(0.866)}{(-0.5)(-0.866) - (0.866)(0.5)}$$

The denominator is zero and  $\omega_3$  is undefined. For the driver to turn more counterclockwise, the combined length of  $z_2$  and  $z_3$  must increase.

It is interesting to note that the Type I singularity is the limiting case of the Type II singularity.

The conditions of singularity are sometimes overcome in practice by ensuring sufficient linkage inertia, avoiding starting in dead-center positions, or avoiding the drive position causing it. For kinematic analysis, however, the members have no inertia and particular care must be taken to detect singularities. This is accomplished by checking continuously for division by zero. Unfortunately, on

the computer, the incremental step may seldom exactly coincide with a division by zero condition, but sudden rises in angular velocity or acceleration also indicate a singularity. This condition will exist if the denominator approaches zero more rapidly than the numerator, as in the linkage of Figure 9. Finally, checking linkage directions for possible binary linkage alignment conditions will also detect singular conditions. Singular conditions indicate the necessity of adding inertia to the system(e. g. , a flywheel) to ensure smooth operation or of placing limits on the operating positions.

### Translational Kinematics of Four-bar Crank Linkage

Once the time histories of angular position, velocity, and acceleration are known, the determination of translational position, velocity, and acceleration is a relatively simple matter. Repeating equations (3) and (4)

$$x = r \cos \theta \quad (3)$$

$$y = r \sin \theta \quad (4)$$

and differentiating both with respect to time twice(holding  $r$  constant) provides translational velocity and acceleration equations:

$$\dot{x} = -r \sin \theta \omega$$

$$\dot{y} = r \cos \theta \omega$$

$$\ddot{x} = -r(\sin \theta \alpha + \cos \theta \omega^2)$$

$$\ddot{y} = r(\cos \theta \alpha - \sin \theta \omega^2)$$

After proper substitution, the translational equations become

$$x = r \cos \theta \quad (3)$$

$$y = r \sin \theta \quad (4)$$

$$\dot{x} = -y\omega \quad (20)$$

$$\dot{y} = x\omega \quad (21)$$

$$\ddot{x} = -y\alpha - \dot{y}\omega \quad (22)$$

$$\ddot{y} = x\alpha + \dot{x}\omega \quad (23)$$

Since vectors may be added by adding corresponding components, equations (3), (4), (20), (21), (22), and (23) may be used repeatedly to find the translational position, velocity, and acceleration of any point on the linkage or of any point referenced to any particular link.

The first step is to establish a translational reference origin. For convenience and consistency, the frame pin of the driver crank may be chosen. Referring to Figure 10, for example, point (or pin) A is made the origin. If, for example, the translation position, velocity, and acceleration of coupler points D and E are desired, the angular position, velocity, and acceleration of the links are computed first. A vector path is then selected from the origin, pin A, to points D and E. A convenient path to point E is  $z_4$ ,  $z_3$ , and  $z_E$ ; for point D, perhaps  $z_1$ ,  $z_2$ , and  $z_D$  is best. For clarification, the kinematic equations defining point D will be presented. It



is important to note that the position, velocity, and acceleration of each pin with respect to the previous pin along the path will be determined; then the "relative" values will be added to give the position, velocity, and acceleration of the point D with respect to the point A, the origin.

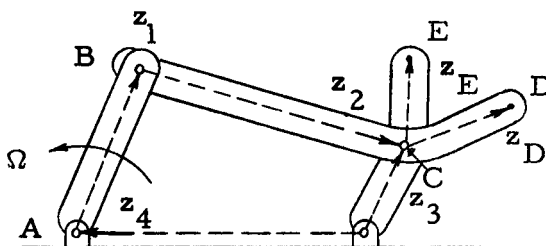


Figure 10. Four-bar crank linkage with coupler points D and E.

$$x_{B/A} = r_1 C_1$$

(B/A read as "of B with respect to A" )

$$y_{B/A} = r_1 S_1$$

$$\dot{x}_{B/A} = -y_{B/A} \omega_1$$

$$\dot{y}_{B/A} = x_{B/A} \omega_1$$

$$\ddot{x}_{B/A} = -y_{B/A} \alpha_1 - \dot{y}_{B/A} \omega_1$$

$$\ddot{y}_{B/A} = x_{B/A} \alpha_1 + \dot{x}_{B/A} \omega_1$$

$$x_{C/B} = r_2 C_2$$

$$y_{C/B} = r_2 S_2$$

$$\dot{x}_{C/B} = -y_{C/B} \omega_2$$

$$\dot{y}_{C/B} = x_{C/B} \omega_2$$

$$\ddot{x}_{C/B} = -y_{C/B} \alpha_2 - \dot{y}_{C/B} \omega_2$$

$$\ddot{y}_{C/B} = x_{C/B} \alpha_2 + \dot{x}_{C/B} \omega_2$$

$$x_{D/C} = r_D C_D$$

$$y_{D/C} = r_D S_D$$

$$\dot{x}_{D/C} = -y_{D/C} \omega_2$$

$$\dot{y}_{D/C} = x_{D/C} \omega_2$$

$$\ddot{x}_{D/C} = -y_{D/C} \alpha_2 - \dot{y}_{D/C} \omega_2$$

$$\ddot{y}_{D/C} = x_{D/C} \alpha_2 + \dot{x}_{D/C} \omega_2$$

$$x_{D/A} = x_{B/A} + x_{C/B} + x_{D/C}$$

$$y_{D/A} = y_{B/A} + y_{C/B} + y_{D/C}$$

$$\dot{x}_{D/A} = \dot{x}_{B/A} + \dot{x}_{C/B} + \dot{x}_{D/C}$$

etc.

.

.

.

## KINEMATIC EVALUATION OF COMPLEX LINKAGES

Since the development in the previous section relied on no peculiarity of a four-bar system, the mathematical relations can be readily extended to linkages with any number of members and with either a crank or ram driver. The degrees of freedom are also non-restrictive, but, as stated, the development will be for the more common single degree of freedom system.

In this section, as an example of a complex linkage, an eleven-bar plane motion linkage with a single degree of freedom and a ram drive will be presented and then analyzed, using the equations previously developed. This particular configuration has been chosen because, first, it represents a linkage of unusual complexity and yet for which the developed techniques apply, and second, it represents a problem comparable to that of the manufacturer's mentioned in the introduction.

### Drawing the Vector Polygon

The first step in an analysis of a complex linkage is drawing the vector polygon representing the system. It is unnecessary to be precise in the vector magnitudes and direction; in fact, it is frequently desirable to exaggerate in order to obtain a clear schematic (it is presumed, of course, that a drawing or a set of

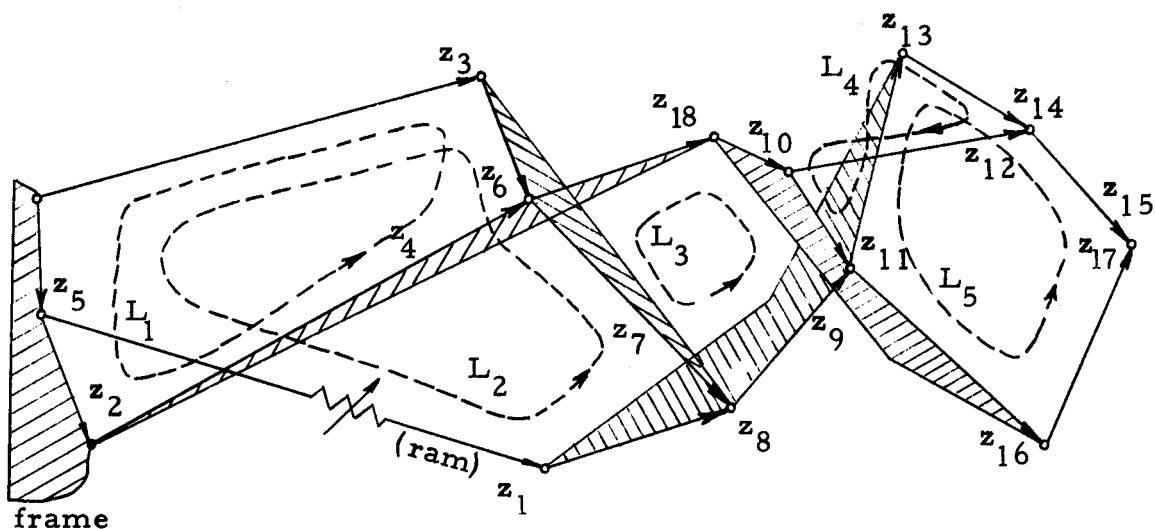
data exist which do give acceptably precise values). The vector polygon for the eleven-bar example is shown in Figure 11.

The distance between any two pins on a link is represented by a simple vector; the vector subscript and sense is arbitrary. If a bar has  $n$ -pins,  $n-1$  vectors are drawn; if  $n > 2$ , shading is included to show rigidity between vectors.

### Vector Loops and Degrees of Freedom

A vector loop is a closed path formed by serial vectors. All such loops must be independent; that is, no loop can be formed by combining two other loops. Each loop provides two equations, one for the imaginary components and one for the real. From elementary algebra it is known that to solve a set of linear equations there must be as many independent equations as there are unknowns. Referring to the linkage of Figure 11, it is seen that two sets of ten unknowns exist,  $\omega_i$  and  $\alpha_i$ . Each set, then, requires five vector loops to provide the necessary ten equations. These loops are shown as dashed contours marked  $L_1$ ,  $L_2$ ,  $L_3$ ,  $L_4$ , and  $L_5$ .

For some linkage arrangements it is not at all simple to determine the possible number of vector loops. A mathematical method, eliminating guesswork, would be to first compute the degrees of freedom using the modified Gruebler equation (page 7 ):



Known:

1.  $@t_0, r_j (j = 1, 18)$
2.  $@t_0, \theta_j (j = 1, 18)$
3.  $\dot{r}_1 = \text{constant}$
4.  $\omega_5 = \omega_2 = 0$
5.  $a_5 = a_2 = 0$
6.  $\omega_6 = \omega_7$
7.  $a_6 = a_7$
8.  $\omega_4 = \omega_{18}$
9.  $a_4 = a_{18}$
10.  $\omega_{10} = \omega_{11} = \omega_{16}$
11.  $a_{10} = a_{11} = a_{16}$
12.  $\omega_8 = \omega_9 = \omega_{13}$
13.  $a_8 = a_9 = a_{13}$

Unknown:

1.  $\omega_1, a_1$
2.  $\omega_3, a_3$
3.  $\omega_4, a_4$
4.  $\omega_6, a_6$
5.  $\omega_8, a_8$
6.  $\omega_{10}, a_{10}$
7.  $\omega_{12}, a_{12}$
8.  $\omega_{14}, a_{14}$
9.  $\omega_{15}, a_{15}$
10.  $\omega_{17}, a_{17}$

Figure 11. The vector polygon representing a single degree of freedom, eleven-bar linkage with a ram drive.

$$X = 3(N-1) - \sum 2(M_i - 1) p_i \quad (24)$$

Then substituting this value of  $X$ , the degrees of freedom, into Paul's equation (page 8 ) for degrees of freedom and solving for  $L$ , the number of independent loops becomes:

$$L = \frac{1}{2} (N - 1 - X) \quad (25)$$

Heretofore, a ram drive was considered a single link of variable length; however, to use Paul's equation, a ram must be thought of as two rigid links with slider contact (having multiplicity of two). The eleven-bar example becomes a twelve-bar system. Solving Gruebler's equation:

$$N = 12$$

$$p_2 = 14$$

$$p_3 = 1$$

$$\therefore X = 3(12-1) - 2(2-1)(14) - 2(3-1)(1) = 1$$

Using  $X = 1$  in Paul's equation establishes the number of possible independent loops in the system of Figure 11.

$$L = (1/2)(12-1-1) = 5$$

The answer of five independent loops indicates that, as might be expected, a set of equations does exist which will allow solving for the unknowns. These loop equations are:

$$L_1 = z_5 + z_2 + z_4 - z_6 - z_3 = 0 \quad (26)$$

$$L_2 = -z_7 - z_6 - z_3 + z_5 + z_1 + z_8 = 0 \quad (27)$$

$$L_3 = -z_{18} + z_7 + z_9 - z_{11} - z_{10} = 0 \quad (28)$$

$$L_4 = z_{11} + z_{13} + z_{14} - z_{12} = 0 \quad (29)$$

$$L_5 = -z_{13} + z_{16} + z_{17} - z_{15} - z_{14} = 0 \quad (30)$$

A convenient way to determine if all equations are independent is to set up a chart, as in Table I, and check to see if any row is a combination of any other rows. The table method is nearly always reliable.

Table I. Vector loop table to show loop independence.

$z$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$L_1$		x	-x	x	x	-x												
$L_2$	x		-x		x	-x	-x	x										
$L_3$							x		x	-x	-x							-x
$L_4$											x	-x	x	x				
$L_5$													-x	-x	-x	x	x	

The table will also indicate if all vectors have been used, and will provide a quick check on the loop equations.

### Writing Equations of Angular Kinematics

To obtain the angular equations of motion, loop equations

(26) through (30) are differentiated with respect to time. The resulting equations, (33) through (42), provide ten equations with ten unknown  $\omega$  terms, the angular velocities. Vectors on the same link are assigned the same angular velocity term and the coefficients are grouped accordingly. The basic form of the differential equations is:

$$\sum \dot{x} = \sum \text{known products} \quad (31)$$

$$-\sum \dot{y} = -\sum \text{known products} \quad (32)$$

The  $\dot{y}$ -component equations have been multiplied by -1 to facilitate the writing of a computer algorithm to be explained in a later section.

The ordering of terms is the same as in Table I.

$$-r_3 S_3 \omega_3 + r_4 S_4 \omega_4 - r_6 S_6 \omega_6 = 0 \quad (33)$$

$$-r_3 C_3 \omega_3 + r_4 C_4 \omega_4 - r_6 C_6 \omega_6 = 0 \quad (34)$$

$$r_1 S_1 \omega_1 - r_3 S_3 \omega_3 - (r_6 S_6 + r_7 S_7) \omega_6 + r_8 S_8 \omega_8 = \dot{r}_1 C_1 \quad (35)$$

$$r_1 C_1 \omega_1 - r_3 C_3 \omega_3 - (r_6 C_6 + r_7 C_7) \omega_6 + r_8 C_8 \omega_8 = -\dot{r}_1 S_1 \quad (36)$$

$$r_7 S_7 \omega_6 + r_9 S_9 \omega_8 - (r_{10} S_{10} + r_{11} S_{11}) \omega_{10} - r_{18} S_{18} \omega_{14} = 0 \quad (37)$$

$$r_7 C_7 \omega_6 + r_9 C_9 \omega_8 - (r_{10} C_{10} + r_{11} C_{11}) \omega_{10} - r_{18} C_{18} \omega_{14} = 0 \quad (38)$$

$$r_{11} S_{11} \omega_{10} - r_{12} S_{12} \omega_{12} + r_{13} S_{13} \omega_8 + r_{14} S_{14} \omega_{14} = 0 \quad (39)$$

$$r_{11} C_{11} \omega_{10} - r_{12} C_{12} \omega_{12} + r_{13} C_{13} \omega_8 + r_{14} C_{14} \omega_{14} = 0 \quad (40)$$

$$-r_{13} S_{13} \omega_8 - r_{14} S_{14} \omega_{14} - r_{15} S_{15} \omega_{15} + r_{16} S_{16} \omega_{10} + r_{17} S_{17} \omega_{17} = 0 \quad (41)$$

$$-r_{13} C_{13} \omega_8 - r_{14} C_{14} \omega_{14} - r_{15} C_{15} \omega_{15} + r_{16} C_{16} \omega_{10} + r_{17} C_{17} \omega_{17} = 0 \quad (42)$$



These equations show that all products containing unknowns sum to zero, with the exception of the equations containing the drive link. In this example, the drive link is a ram which has both a variable magnitude and a variable direction. For such a link,

$$\dot{x}_d = \dot{r}_d C_d - r_d S_d \omega_d$$

$$\dot{y}_d = \dot{r}_d S_d + r_d C_d \omega_d$$

In these equations, only  $\omega_d$  is unknown. Grouping is according to equations (31) and (32).

To obtain the set of equations which will provide angular acceleration, equations (33) through (42) are differentiated. Similar to (31) and (32), (43) and (44) are used to establish grouping.

$$\sum \ddot{x} = \sum \text{known products} \quad (43)$$

$$\sum \ddot{y} = \sum \text{known products} \quad (44)$$

Equations (45) through (54) are equations (33) through (42) differentiated. For these equations, angular rates,  $\omega$ , are assumed known from (33) through (42).

$$-r_3 S_3 a_3 + r_4 S_4 a_4 - r_6 S_6 a_6 = r_3 C_3 \omega_3^2 - r_4 C_4 \omega_4^2 + r_6 C_6 \omega_6^2 \quad (45)$$

$$-r_3 C_3 a_3 + r_4 C_4 a_4 - r_6 C_6 a_6 = -r_3 S_3 \omega_3^2 + r_4 S_4 \omega_4^2 - r_6 S_6 \omega_6^2 \quad (46)$$

$$r_1 S_1 a_1 - r_3 S_3 a_3 - (r_6 S_6 + r_7 S_7) a_6 + r_8 S_8 a_8 =$$

$$-2\dot{r}_1 S_1 \omega_1 - r_1 C_1 \omega_1^2 + r_3 C_3 \omega_3^2 + (r_6 C_6 + r_7 C_7) \omega_6^2 - r_8 C_8 \omega_8^2 \quad (47)$$

$$r_1 C_1 a_1 - r_3 C_3 a_3 - (r_6 C_6 + r_7 C_7) a_6 + r_8 C_8 a_8 =$$

$$-2\dot{r}_1 C_1 \omega_1 + r_1 S_1 \omega_1^2 - r_3 S_3 \omega_3^2 - (r_6 S_6 + r_7 S_7) \omega_6^2 + r_8 S_8 \omega_8^2 \quad (48)$$

$$r_7 S_7 a_6 + r_9 S_9 a_8 - (r_{10} S_{10} + r_{11} S_{11}) a_{10} - r_{18} S_{18} a_{18} =$$

$$-r_7 C_7 \omega_6^2 - r_9 C_9 \omega_8^2 + (r_{10} C_{10} + r_{11} C_{11}) \omega_{10}^2 + r_{18} C_{18} \omega_{18}^2 \quad (49)$$

$$r_7 C_7 a_6 + r_9 C_9 a_8 - (r_{10} C_{10} + r_{11} C_{11}) a_{10} - r_{18} C_{18} a_{18} =$$

$$r_7 S_7 \omega_6^2 + r_9 S_9 \omega_8^2 - (r_{10} S_{10} + r_{11} S_{11}) \omega_{10}^2 - r_{18} S_{18} \omega_{18}^2 \quad (50)$$

$$r_{11} S_{11} a_{10} - r_{12} S_{12} a_{12} + r_{13} S_{13} a_{18} + r_{14} S_{14} a_{14} =$$

$$-r_{11} C_{11} \omega_{10}^2 + r_{12} C_{12} \omega_{12}^2 - r_{13} C_{13} \omega_{18}^2 - r_{14} C_{14} \omega_{14}^2 \quad (51)$$

$$r_{11} C_{11} a_{10} - r_{12} C_{12} a_{12} + r_{13} C_{13} a_{18} + r_{14} C_{14} a_{14} =$$

$$r_{11} S_{11} \omega_{10}^2 - r_{12} S_{12} \omega_{12}^2 + r_{13} S_{13} \omega_{18}^2 + r_{14} S_{14} \omega_{14}^2 \quad (52)$$

$$-r_{13} S_{13} a_{18} - r_{14} S_{14} a_{14} - r_{15} S_{15} a_{15} + r_{16} S_{16} a_{10} + r_{17} S_{17} a_{17} =$$

$$r_{13} C_{13} \omega_{18}^2 + r_{14} C_{14} \omega_{14}^2 + r_{15} C_{15} \omega_{15}^2 - r_{16} C_{16} \omega_{10}^2 - r_{17} C_{17} \omega_{17}^2 \quad (53)$$

$$-r_{13} C_{13} a_{18} - r_{14} C_{14} a_{14} - r_{15} C_{15} a_{15} + r_{16} C_{16} a_{10} + r_{17} C_{17} a_{17} =$$

$$r_{13} S_{13} \omega_{18}^2 - r_{14} S_{14} \omega_{14}^2 - r_{15} S_{15} \omega_{15}^2 + r_{16} S_{16} \omega_{10}^2 + r_{17} S_{17} \omega_{17}^2 \quad (54)$$

Note the coriolis acceleration term in (47) and (48) which contained the driver vector.

With angular rate and acceleration being provided by equations (33) through (42) and (45) through (54), respectively, new angular positions may be calculated for a time lapse of  $\Delta t$  by using:

$$\theta = \alpha_0 \frac{\Delta t^2}{2} + \omega_0 \Delta t + \theta_0 \quad (17)$$

With these approximated angles for the new position, approximate values of  $\omega$  may be calculated using (33) through (42). These new values of  $\omega$  are used in (19) to provide a better approximation of the angles.

$$\theta = (\omega + \omega_0) \frac{\Delta t}{2} + \theta_0 \quad (19)$$

After sufficient iteration to effect desired convergence, final new values of angular rate and acceleration are computed. The procedure is repeated until the desired range of operation has been evaluated.

### Determining Translational Kinematics

To find the translational position, velocity, or acceleration of any point, the equations presented for the four-bar system are used.

$$x = r \cos \theta \quad (3)$$

$$y = r \sin \theta \quad (4)$$

$$\dot{x} = -y\omega \quad (20)$$

$$\dot{y} = x\omega \quad (21)$$

$$\ddot{x} = -y\alpha - \dot{y}\omega \quad (22)$$

$$\ddot{y} = x\alpha + \dot{x}\omega \quad (23)$$

These equations apply for a point on a single link. As before, the translational origin is placed at the frame pin of the drive link. To determine the relative position, velocity, and acceleration of a point, say  $p_i$ , on a particular link, say  $l_j$ , a vector path is chosen from the origin to that point. Each equation, (3) and (4), and (20) through (23), is applied to each vector step and the corresponding components summed.

## THE COMPUTER PROGRAM TO PERFORM KINEMATIC EVALUATION

In the previous section, kinematic equations for an eleven-bar linkage were developed as an example of the method of evaluating a complex linkage. To obtain specific kinematic values in this example it would have been necessary to:

- 1) perform around 50 arithmetic operations to obtain the coefficients for the rate matrix,
- 2) solve a 10-by-10 set of linear algebraic equations for angular rates,
- 3) perform around 80 arithmetic operations to compute better angles (except for initial position),
- 4) repeat steps 1), 2), and 3)  $n$  times for convergence,
- 5) perform around 120 arithmetic operations to obtain the coefficients for the acceleration matrix,
- 6) solve a 10-by-10 set of linear algebraic equations for angular accelerations,
- 7) perform about 10 arithmetic operations for each link step in the vector path leading to the point whose translational kinematics was desired,
- 8) perform about 140 arithmetic operations to estimate angles for next position,
- 9) repeat steps 1) through 8) as many times as necessary to traverse the desired range.

It is also necessary to remember that each basic operation of 1) through 8) and each position of the linkage is entirely dependent on previous calculations. Obviously, carrying insufficient significant figures, making any arithmetic error, or having an error in the link lengths, original angles, or driver input value would, in all probability, invalidate the results. Steps 2) and 6) alone, for, say ten driver positions requiring three iterations each for convergence would require the error-free solution of a 10-by-10 set of linear equations thirty-eight times. Consideration of human error and slowness, even with an electric calculator, indicates that the only reasonable way to perform the calculations is with a high-speed digital computer.

The fundamental advantage in using a digital computer to perform these calculations is the changing of the task from the nearly impossible (for a highly complex linkage) to the relatively simple. Next, once a general program has been prepared, even an inexperienced person could perform the analysis with relative ease, even with little understanding of the theory or mathematics. Accuracy, too, would be enhanced, since most computers can carry anywhere from one to about 20 significant figures and perform endless computations without error. The cost of making the analysis becomes virtually insignificant, also, because for example, computers capable of performing 200 to 500 man-years of work in one

hour rent for around \$500 per hour. Additionally, this time can be purchased on an hourly basis without the necessity of computer rental for longer periods of time. The final advantage of significance is that a design can be evaluated immediately for each design change, eliminating the competitively dangerous long delay usually associated with a highly complex linkage design evaluation.

The complete and working programs for performing kinematic analysis of any single degree of freedom, plane-motion linkage with turning joints and with either a crank or ram constant input are presented in the appendices. These programs are coded in the FORTRAN language specified for the IBM 1410 computer (5, p. 7). This language is, with only minor exceptions, the same as the FORTRAN II language which is of virtually universal use on modern, high speed computers.

The programs are presented as working tools for the design analyst. The input data required for operation has, for the most part, been kept simple and as closely related to the fundamental vector-sum equations as was possible. Subsequent sections will present an explanation of the program section's input, output, mathematics, and logic.

The programs, although general in nature, do, of course, have a scope of applicability; i. e., they apply only over a certain class of problems. In order for a linkage to be evaluated by these

programs, the following conditions must hold:

- 1) All connections must be turning joints (excluding the sliding contact of the ram's piston and cylinder).
- 2) The number of moving links must not exceed ten.
- 3) The driving link must be either a crank or ram/ piston device.
- 4) Angular rate of driver must be constant for crank driver; length change rate of driver must be constant for ram/ piston.

The foregoing conditions can be broadened, if necessary, by certain programming modifications. If, however, a set of linear equations significantly greater than ten is to be solved for the unknown variables, a computer of greater speed and capacity than the IBM 1410 may have to be used.

### The Closure Program

Fundamental to the mathematical technique previously developed for linkage analysis is that the sum of the position vectors about the linkage vector-polygon is zero; i. e. :

$$\sum z_j = 0 \quad (6)$$

where

$$z_j = r_j(\cos\theta_j + i\sin\theta_j) \quad (5)$$



and  $r_j$  is the distance (in any desired units) between adjacent pins on a link and  $\theta_j$  is the angle, in radians, from a right-directed horizontal line to the vector, measured counterclockwise.

Equation (6) is mathematically exact, but  $r_j$  and  $\theta_j$  are measured quantities and, as such, are inexact; the result is:

$$\sum z_j = \epsilon$$

where  $\epsilon$  would not normally be zero if the  $r$ 's and  $\theta$ 's were scaled from a drawing.

The CLOSURE program in Appendix A calculates  $\epsilon$  for each closed vector path specified and computes the change, if any, in length and direction of a particular vector in the loop required to make  $\epsilon = 0$ . By doing this the program accomplishes two things: it detects gross errors in the measurement of the vector magnitude and angles before using them to calculate the kinematic data, and it provides a means of adjusting for unavoidable measurement error. Since the data will normally be from a flexible design, having one link absorb the error appears justified.

The program, in Appendix A, should be referred to for exact information concerning data format specifications.

The following are the input and output variable definitions, in order of appearance.

**Input:**

- 1) MATRIX, MAXR, R, THETA -- see definitions in next subsection under program input.
- 2) NUM -- control data: the number of position vectors in the jth loop equation.
- 3) NORDER -- control data: the sign and subscript of each position vector in the jth loop equation. Not the same as NORDER for program MAIN.
- 4) MAKEUP -- control data: the signed subscript of the position vector in the jth loop equation which is to be adjusted so that  $\epsilon = 0$ .

**Output:**

- 1) L, I -- subscripting.
- 2) B(L) -- the error sum of the vector products, called  $\epsilon$  in the text, for the 2J-1 component equation.
- 3) MI -- absolute values of NORDER(J, K).
- 4) RERR -- difference between the computed vector magnitude and R(MI). (See item 13 in next subsection under program input.)
- 5) RAD -- the adjusted R(MI).
- 6) ANGERR -- difference between the computed angle and THETA(MI).
- 7) ANGLE -- the adjusted THETA(MI).

### The Basic Program

The purpose of the basic program, called MAIN, is to perform the mathematical operations necessary for complete kinematic evaluation of any linkage, if it is of the kind prescribed. The program attains this end through arithmetic and logical operations and the use of five subroutines -- ACOE, BCOE, GAUSS, ANGLE, and KINE. These subroutines are based upon previously developed mathematics. The logical flow of program MAIN is shown in Figure 12, the flow schematic. For obvious reasons, minor operations are not included.

Program MAIN reads control and computational data. Current values of angles are written out. Subroutine ACOE is called and computes the A-coefficients of the angular rate matrix, such as (33) through (42). Subroutine BCOE is called and, using MODE 1, computes the sum of known products for the velocity vector matrix. Subroutine GAUSS uses the output of ACOE and BCOE to compute angular rates from the rate matrix. If the rate matrix is singular (as indicated by division by zero), the adjusted A-coefficient array (see section on GAUSS) is written out and GAUSS is exited. This triggers the program to restore the original A-coefficient array by recalling ACOE; the original array is written out and all further calculations on these linkage data are terminated.

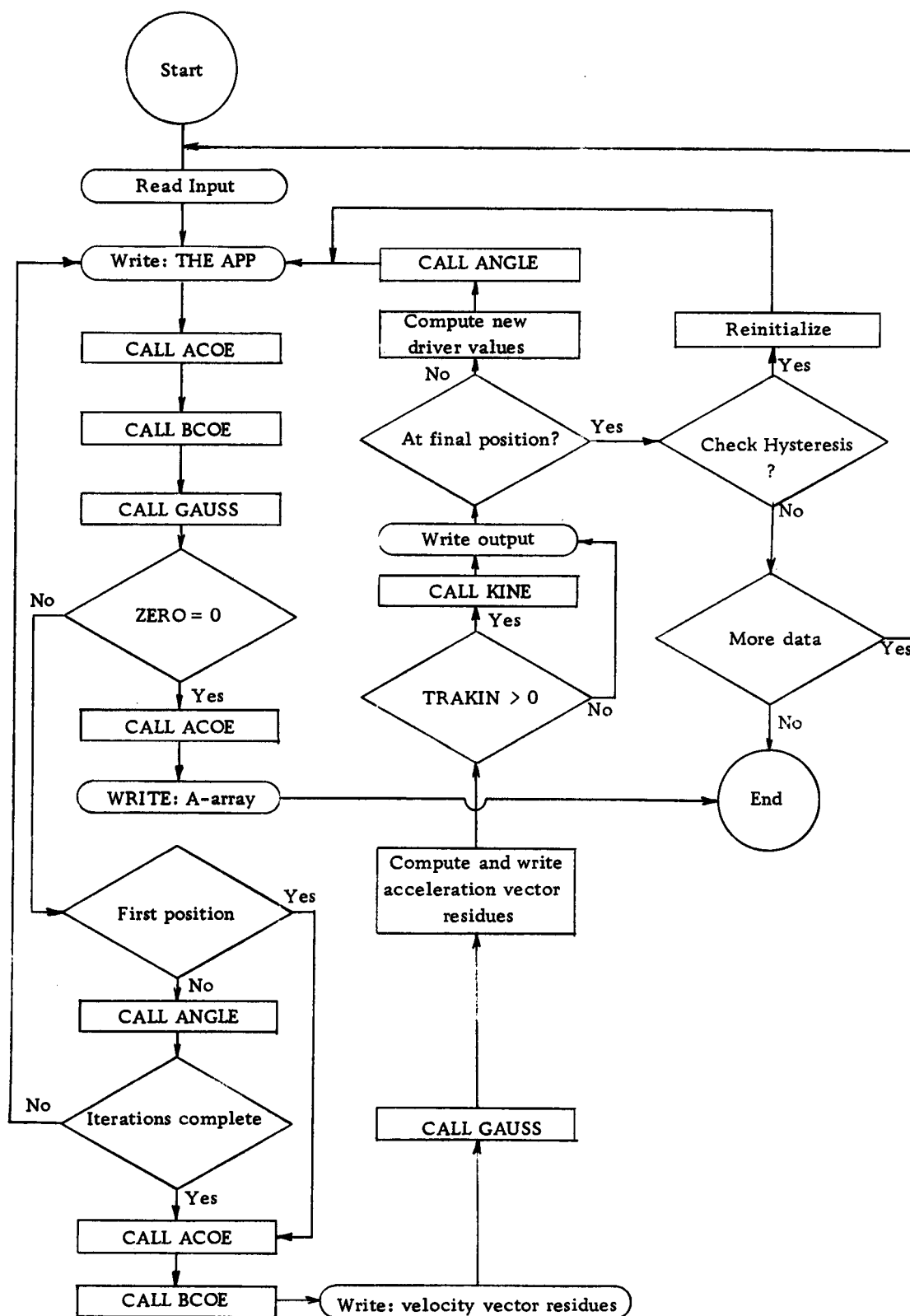


Figure 12. Schematic flow diagram for program MAIN

If, however, the rate matrix is non-singular, the control proceeds (except for the first step) to subroutine ANGLE which computes better approximations of the angles. If the prescribed iterations have not been accomplished, control shifts back to the angle write-out statement prior to ACOE and the entire sequence is repeated. Upon completing the required iterations, the program proceeds to the calculation of angular accelerations. ACOE is called again and using the most recent angular approximations establishes the A-coefficient array to be used in the acceleration matrix such as (45) through (54). BCOE, in MODE 2, is called to compute the known product sums for the acceleration matrix and the velocity vector residues (see paragraph describing BCOE). Upon exiting BCOE, the program writes out the velocity vector residues and calls GAUSS to compute the angular accelerations. After determining the angular accelerations, the acceleration vector residues are computed.

The calculation of these residues is the only significant computation not performed by a subroutine. The fundamental algorithms used are:

$$\rho(\ddot{\mathbf{x}}) = \frac{100 \sum \ddot{\mathbf{x}}}{\sum |\ddot{\mathbf{x}}|}$$

and

$$\rho(\ddot{\mathbf{y}}) = \frac{100 \sum \ddot{\mathbf{y}}}{\sum |\ddot{\mathbf{y}}|}$$

The program statements accomplishing these calculations are found between statements 66 and 75 in program MAIN. Following the calculations, the residues are written out.

At this point, if translational kinematics are desired, subroutine KINE reads in the control cards, makes the calculations, and writes out the results. The computations being complete, the angular position, rate, and acceleration are written out. A decision is made as to whether the last desired position has been reached. If not, control cycles to the beginning and the procedure (except for the reading in of data) is repeated. If the end position has been reached, a decision is made as to whether it is desired to reverse the drive link and compute back to the original position. The last instruction cycles the program back to read the next data set; if there is none, the program exits.

The data card format and other program details can be found in Appendix B. The input/output data definitions for program MAIN only, are:

Input, in order of appearance:

- 1) MATRIX -- control data: a number equal to the set size of the unknown angular rates or angular accelerations; also, it is equal to twice the number of z-loop equations; maximum of 10.

- 2) IPTYPE -- control data: if set equal to 1, equations for ram drive are selected; if set equal to 2, equations for crank drive are selected.
- 3) MXSTEP -- control data: the number of discrete positions desired for the drive link, including the original position; if HYST > 0, twice this number are computed.
- 4) NUMCYC -- control data: the number of times, except for the original position, that ACOE-BCOE-GAUSS-ANGLE will be cycled through for convergence of the angles.
- 5) MAXR -- control data: the number of r-values (or  $\theta$ -values) to be read into storage; maximum of 20.
- 6) DELTEE -- computational data: the value, in time units, of one time step; used to compute new crank angle or ram length.
- 7) DRIVER -- computational data: the rate value of the drive member; for IPTYPE = 1, it is the ram lengthening or shortening rate, in units consistent with r-values and DELTEE; for IPTYPE = 2, it is the crank angular rate, positive CCW and in degrees/DELTEE units. DRIVER times DELTEE is used as the change in the drive member.
- 8) SIG -- computational data: A subscripted variable (equal to 0, or  $\pm 1$ ) providing the sign of the drive link as it appears in  $\Sigma z = 0$ .
- 9) IFRAME -- control data: a subscripted, unsigned variable providing the subscripts of the frame vectors (fixed links).

- 10) HYST -- control data: if set to 0, program terminates after last step; if set to 1, program reverses polarity of DRIVER and computes last position back to original position. Used to determine computational hysteresis.
- 11) TRAKIN -- control data: if 0, subroutine KINE is bypassed; if 1, KINE is called.
- 12) ALL -- control data: if set to 1, and KINE is called, the translational kinematics of every vector step is printed out. If set to 0, just the data of the last vector on the path is printed by KINE. See subroutine KINE explanation.
- 13) R -- computational data: a subscripted, unsigned variable containing the values of the vector magnitudes. Order of appearance is order of subscripting. The first value must be for the drive link. Units must be consistent with DRIVER, if IPTYPE = 1.
- 14) THETA -- computational data: a subscripted, unsigned variable containing the values of the vector directions, positive CCW from the right horizontal. Order must be consistent with r-values. Units are degrees.
- 15) NORDER -- control data: a double subscripted variable containing the signed subscript of the vectors as they appear in the loop equation. From card-to-card, corresponding fields must refer to vectors on same link (See Table II). On the same card, fields 1 and 2 are for link number 1, 3 and 4 are for link number 2, etc., and must contain subscripts of vectors on same



link.

If there are no paired vectors in the loop equation, the second field of the pair is left blank or set to zero. If there is no vector from a particular link in the loop equation, both fields are left blank or set to zero. No equation may have three or more vectors from the same link. (See section on ACOE).

Output, in order of appearance:

- 1) THEAPP -- the current approximations of angular position, in degrees.
- 2) "ACOE (I, I) IS ZERO" -- the statement printed by GAUSS if a division by zero is detected, I is the appropriate subscript.
- 3) AY -- the adjusted A-coefficient array printed out by GAUSS if division by zero is attempted.
- 4) BEE -- the adjusted B-value array printed with AY.
- 5) COLSWP -- the array which indicates the column pivoting performed in order to maximize the initial elements of the adjusted matrices. Printed with AY.
- 6) AY -- the restored A-coefficient array written by MAIN if division by zero is attempted by GAUSS.
- 7) BUFF1 -- a data buffer containing velocity vector residues computed in BCOE and written out by MAIN.
- 8) BUFF1 -- the same buffer as 7) but now containing acceleration vector residues computed and written out by MAIN.

- 9) (Subroutine KINE) -- translational kinematics. See KINE description.
- 10) NMSTEP -- the numerical value specifying which discrete position of the driver is currently being calculated.
- 11) R(1), THETA2(1) -- the magnitude and direction of the drive vector.
- 12) THETA2 -- the final value of angles, in degrees, for current NMSTEP. Order of output gives subscript.
- 13) BUFF1 -- the final values of angular rates, in degrees per unit time, for current NMSTEP. Order of output gives subscript.
- 14) ALPHA -- the values of angular accelerations, in degrees per time unit squared, for current NMSTEP. Order of output gives subscript.

Program MAIN, as it appears in Appendix B, is restricted to ten moving members including ram drive, or ten moving members and crank drive. The total number of vectors describing the linkage is 20, and the total frame vectors allowable is five. To broaden the application to more complex single degree of freedom linkages it is only necessary to modify the DIMENSION specifications of the program and the subroutines. For example, if a 20 member system is desired, the DIMENSION statements might be rewritten as AY(20,20), BEE(20), X(20), BUFFI(20), R(40), etc. Note, however, that as programmed for the IBM 1410, the program, subroutines, and built

in functions use virtually the entire core; to expand to a 20 member system would require tape storage or the use of a larger capacity computer.

### The Matrix Evaluation Subroutines

The matrix evaluation subroutines, ACOE, BCOE, and GAUSS, along with the other subroutines, ANGLE and KINE, were given distinct identities for several reasons: the saving of core storage, ease of checkout, and the enhancement of understanding the total operation by simple presentation of distinct pieces.

The three matrix evaluation subroutines provide the means of calculating the angular rates and angular accelerations of the mechanism links if the link lengths and directions are known, as well as the drive motion. Basically, it is a programmed method of solving  $n$  equations for  $n$  unknowns, such as (45) through (54). There exist several established mathematical methods for doing this. One in particular appears well suited to handle a large matrix with at least some zero coefficients, and which can conveniently be programmed for automatic solution. The method, generally attributed to Gauss (4, p. 428), (7, p. 28), is commonly called "Gauss reduction". The technique is that of starting with a set of equations, as in (55), and solving it in five operations.

$$\begin{array}{cccccc}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n & = & b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n & = & b_2 \\
\cdot & & \cdot & & \cdot & \\
\cdot & & \cdot & & \cdot & \\
\cdot & & \cdot & & \cdot & \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n & = & b_n
\end{array} \tag{55}$$

Step one: Determine  $a_{ij}$  of greatest absolute magnitude. If  $i \neq 1$ , exchange  $i$ th row with first row. If  $j \neq 1$ , exchange  $j$ th column with first column. This is necessary to prevent division by zero in Step Two in case  $a_{11} = 0$ , and is also beneficial in reducing computation error by ensuring division by the largest possible number.

Step two: Divide new row one elements by new  $a_{11}$ .

Step three: Multiply modified row one by  $a_{21}$  and difference with corresponding elements of row two. Do likewise with  $a_{31}$  and row three,  $a_{41}$  and row four, etc.

Step four: the new set of equations will look like (56), where primes indicate the result of the prescribed operations.

$$\begin{array}{cccccc}
x_1 + a'_{12}x_2 + a'_{13}x_3 + \dots + a'_{1n}x_n & = & b'_1 \\
0 + a'_{22}x_2 + a'_{23}x_3 + \dots + a'_{2n}x_n & = & b'_2 \\
\cdot & & \cdot & & \cdot & \\
\cdot & & \cdot & & \cdot & \\
\cdot & & \cdot & & \cdot & \\
0 + a'_{n2}x_2 + a'_{n3}x_3 + \dots + a'_{nn}x_n & = & b'_n
\end{array} \tag{56}$$

Ignore row and column one and repeat steps one, two, and three on the remaining array. Continue operations until set (55) is a unit diagonal like (57).

$$\begin{array}{ccccccc}
 x_1 + a''_{12}x_2 + a''_{13}x_3 + \dots + a''_{1n}x_n & = & b''_1 \\
 0 + x_2 + a''_{23}x_3 + \dots + a''_{2n}x_n & = & b''_2 \\
 \cdot & & \cdot & & \cdot & & \cdot \\
 \cdot & & \cdot & & \cdot & & \cdot \\
 \cdot & & \cdot & & \cdot & & \cdot \\
 0 + 0 + 0 + \dots + x_n & = & b''_n
 \end{array} \tag{57}$$

Step five: With the unit diagonal matrix (57), first use  $x_n = b''_n$  to solve for  $x_{n-1}$  in the previous row; i. e.

$$\begin{aligned}
 x_{n-1} + a''_{n-1,n}x_n &= b''_{n-1} \\
 x_{n-1} &= b''_{n-1} - (a''_{n-1,n})b''_n
 \end{aligned}$$

Using  $x_n$  and  $x_{n-1}$ , solve for  $x_{n-2}$ . Continue until all  $x$ s are determined. When performing this operation, the fact that the columns may have been switched  $n-1$  times during step one must be accounted for. This is accomplished by maintaining a record of which columns were switched for which adjusted matrix and, after solving for the current  $x_i$ , switching it back to its original position before determining  $x_{i-1}$ . The result will be the column matrix of  $x$  in its original order.

The foregoing Gauss reduction is performed by the

program as outlined. The outlined  $a_{ij}$  coefficients are computed by the subroutine ACOE, the original  $b_s$  are computed by BCOE, and Gauss reduction is performed by Gauss. See Appendices C, D, E.

Subroutine ACOE. This subroutine, shown in detail in Appendix C, computes the  $a$ -coefficients for use in the Gauss reduction subroutine, GAUSS. All data, except the matrix size specification, MATRIX, is implicitly transferred from MAIN to ACOE through COMMON. These data are R, NORDER, and THETA2. The resulting values of the  $a_{ij}$ 's, called AY(I, J), are transferred back to MAIN by way of COMMON. The equations used to perform the calculations result from the observation that a vector-loop about a link polygon provides two sets of equations. The only difference in the two equations is that one uses  $\cos\theta$  products whereas the other uses  $\sin\theta$  products; the only difference between any two terms of a single equation is the values of R and THETA. For example, considering equations (33) through (42) and (45) through (54), the coefficients of  $\omega_i$  and  $a_i$  are identical and follow a pattern predictable from the original vector loop equations (26) through (30). First,  $z_2$  and  $z_5$  are ignored because they are frame vectors and cannot move. By assigning a number to each link, a table such as Table II can be prepared wherein the link members and loop equations are related through vector signs and subscripts. In the table, each row represents a vector loop equation (without frame vectors) and each column

represents a link. The table numbers are the signs and subscripts of the vectors as they appear in the loop equations.

Table II. Vector loops and mechanism links related through vector signs and subscripts. The table values are called NORDER (I, J) in the program.

Link No.	1	2	3	4	5	6	7	8	9	10
L <sub>1</sub>		-6	-3	4						
L <sub>2</sub>	1	-7, -6	-3		8					
L <sub>3</sub>		7		-18	9	-11, -10				
L <sub>4</sub>					13	11	14	-12		
L <sub>5</sub>					-13	16	-14		17	-15

(Note: for programming reasons, with a ram drive, link no. 1 must be the driver; for a crank drive, the drive link does not appear in the table.) The table numbers are introduced to the program by the array NORDER(M, N). The absolute values of NORDER become the subscripts of R and THETA2, and the sign becomes the sign of the a-coefficients. Notice that each row of the table, or NORDER(M, N), is used to provide calculation of two rows of a-coefficients. These control values are used in the basic equations:

$$a_{ij} = r_k \cos \theta_k$$

$$a_{i+1,j} = r_k \sin \theta_k$$

which in the program appear as:

$$AY(I, J) = SIGN * R(K) * COS(THETA2(K)) \quad (58)$$

$$AY(I+1, J) = SIGN * R(K) * SIN(THETA2(K)) \quad (59)$$

where  $K = |NORDER(M, N)| \quad (60)$

$$SIGN = NORDER(M, N) / K \quad (61)$$

$$I = 2M$$

$$J = 2N-1$$

If two signed numbers appear in a single location in the table, they also appear in two successive positions in NORDER, which results in two calculations each of (58) and (59) and like values being summed.

An example would clarify. Consider row  $L_3$  of Table II; it represents the vector loop equation (28). To obtain the coefficients of the corresponding equations (37) and (38), a set of values of NORDER are prepared.

$$\begin{aligned} NORDER(3, J) = & -18, 0; 7, 0; 0, 0; 0, 0; 9, 0; -11, -10; \\ & 0, 0; 0, 0; 0, 0; 0, 0. \end{aligned}$$

The program reads the first value of NORDER, which is NORDER(3, 1) = -18. Using (60) and (61), the equations (58) and (59) become:

$$AY(6, 1) = -R(18) * COS(THETA2(18))$$

$$AY(7, 1) = -R(18) * SIN(THETA2(18))$$

The variables R(18) and THETA 2(18) are read from COMMON



and the calculations are completed. Since  $NORDER(3, 2) = 0$ ,  $AY(6, 2)$  and  $AY(7, 2)$  are stored unaltered. The process is continued until ten values each of  $AY(6, J)$  and  $AY(7, J)$  have been computed, six of which are identically zero because six sets of  $NORDER$  were zero.

In this way the entire set of a-coefficients is computed and then stored for use in subroutine GAUSS.

Subroutine BCOE. This subroutine, presented in detail in Appendix D, computes the b-values required to solve for  $X$  in the matrix product  $A \cdot X = B$ , represented by set (55). Unlike the a-coefficients, which are the same whether solving for angular rates or angular accelerations or whether the drive link is a crank or a ram, the b-values are different for each case. Consequently, four different sets of equations are required. Figure 13 presents the program schematic.

The proper set of equations is selected according to the control arguments  $MODE$  and  $IPTYPE$ . The number of values computed is controlled by  $MATRIX$ . The argument  $DRIVER$  is the value of the drive link rate, whether it is unit length per unit time for a ram, or radians per unit time for a crank. All other data appear as arrays and are implicitly transferred through  $COMMON$ . The arrays are:  $AY$ , from  $ACOE$ ;  $OMEGA2$ , angular rates used to compute b-values for  $\alpha$ ;  $R$ ;  $THETA2$ ; and  $SIGN$ , equal to zero or  $\pm 1$ , which is used to give the b-values the proper sign.

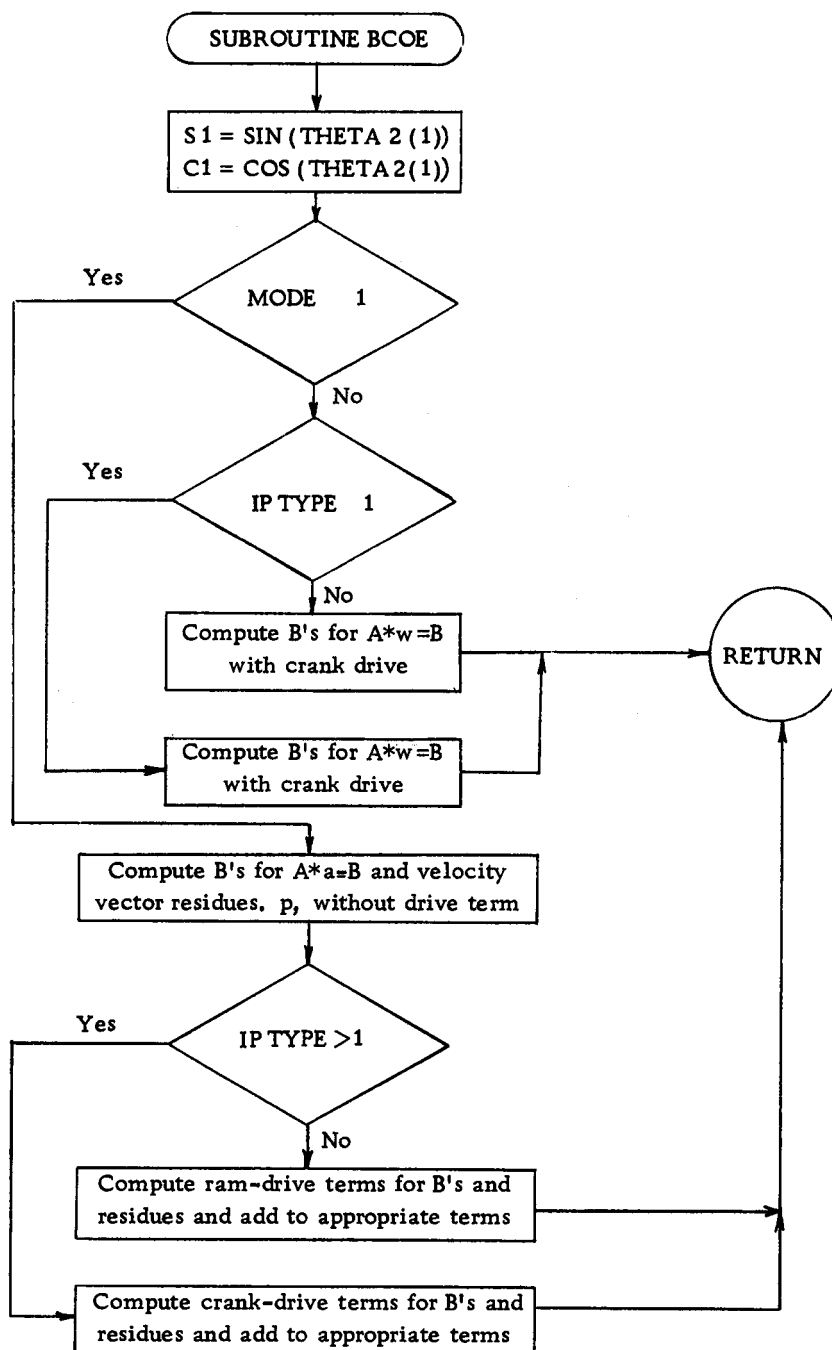


Figure 13. Schematic flow diagram for subroutine BCOE.

The equations defining the b-values for  $A*\omega = B$  with ram drive are (see (35), (36)):

$$b_i = \pm \dot{r}_1 \sin\theta_1$$

$$b_{i+1} = \pm \dot{r}_1 \cos\theta_1$$

or

$$BEE(I) = -SIGN(I)*DRIVER*S1$$

$$BEE(I+1) = -SIGN(I)*DRIVER*C1$$

where

$$S1 = \sin(THETA2(1))$$

$$C1 = \cos(THETA2(1))$$

These equations require that the angle of the driver always be subscripted (1). This means the driver angle must be the first one in the input data. The SIGN(I) term carries the sign of driver as it appears in the original loop equation. If the drive link is not in the ith loop equation, SIGN(I) = 0.

The equations defining the b-values for  $A*\omega = B$  with crank drive are (see(11), (12)):

$$b_i = \pm r_1 \Omega \cos\theta_1$$

$$b_{i+1} = \pm r_1 \Omega \sin\theta_1$$

or

$$PROD = SIGN(I)*R(1)*DRIVER$$

$$BEE(I) = -PROD*C1$$

$$BEE(I+1) = -PROD*S1$$

As before, the subscript of the drive link data must be (1). The

term SIGN(I) has the previous definition.

The majority of b-values for the  $A*\omega = B$  matrix are zero; they have a non-zero value only when the drive link is in that particular vector loop. The b-values for  $A*a = B$ , however, are rarely zero because, although only two or so have a drive link term, everyone has  $n$  omega-squared products (where  $n$  = no. of vectors in that vector loop). This may be observed in equations (45) through (54). For this reason, in the program the omega-squared terms are summed and then, if appropriate, the drive-link terms are added. The basic equations are:

$$b_i = \sum r_k \sin \theta_k \omega_j^2$$

$$b_{i+1} = \sum r_k \cos \theta_k \omega_j^2$$

It can be seen that the coefficients of the omega-squared terms are a-coefficients. In the program this fact is used and the resulting equations are:

$$PROD1 = AY(I+1, J)*OMEGA2(J)$$

$$BEE(I) = BEE(I) + PROD1*OMEGA2(J)$$

$$PROD2 = AY(I, J)*OMEGA2(J)$$

$$BEE(I+1) = BEE(I+1) - PROD2*OMEGA2(J)$$

After summing these terms, the drive-link terms are added. For a ram-drive the drive-link terms are (see (49), (48)):

$$b_i = b_i \pm 2\dot{r}_1 \cos\theta_1 \omega_1$$

$$b_{i+1} = b_{i+1} \pm 2\dot{r}_1 \sin\theta_1 \omega_1$$

Or, in the program language:

$$\text{PROD1} = \text{SIGN(I)} * \text{DRIVER} * \text{C1}$$

$$\text{PROD2} = \text{SIGN(I)} * \text{DRIVER} * \text{S1}$$

$$\text{BEE(I)} = \text{BEE(I)} - 2.0 * \text{PROD1} * \text{OMEGA2(1)}$$

$$\text{BEE(I+1)} = \text{BEE(I+1)} - 2.0 * \text{PROD2} * \text{OMEGA2(1)}$$

The ram-drive term is the familiar coriolus term. The subscript of R, THETA2, and OMEGA2 must be (1). The R and THETA2 subscript of (1) is effected by placing the drive-link data in the first field of the input data; the OMEGA subscript of (1) is effected by assigning the number 1 to the ram-drive-link, as in Table II. Note, however, that for a crank-drive, the angular rate is prescribed and, consequently, the drive-link is assigned no number and does not appear in the table.

For a crank-drive, the driver term to be added to the b-values results from the equations (see (15), (16)):

$$b_i = b_i \pm r_1 \cos\theta_1 \Omega^2$$

$$b_{i+1} = b_{i+1} \mp r_1 \sin\theta_1 \Omega^2$$

```

or:      PROD2 = SIGN(I)*C1*DRIVER*R(1)

          PROD1 = SIGN(I)*S1*DRIVER*R(1)

          BEE(I) = BEE(I) + PROD1*DRIVER

          BEE(I+1) = BEE(I+1) - PROD2*DRIVER

```

After calculation, the b-values are returned to program MAIN by way of COMMON for use with subroutine GAUSS.

An additional feature of this subroutine is that during calculations to obtain b-values for  $A*a = B$ , some of the results are used to calculate the residual,  $\rho$ , in

$$\sum r_i \cos \theta_{ij} = \rho(\dot{y})$$

$$\sum r_i \sin \theta_{ij} = \rho(\dot{x})$$

In addition, the following calculations are made.

$$\sum |r_i \cos \theta_{ij}| = \text{total y-path}$$

$$\sum |r_i \sin \theta_{ij}| = \text{total x-path}$$

The ratio of  $\rho$  to total path times 100 gives the percentage of residual error. In the program

$$\rho_i = \text{BUFFER}(I)$$

$$\text{total path} = \text{TOTPTH}(I)$$

$$\frac{\rho}{\text{total path}} \times 100 = \text{BUFFER}(I) = \text{BUFFER}(I) / \text{TOTPTH}(I) * 100.0$$

The final value of BUFFER is the percentage residual and is printed out in program MAIN. These values can be used to determine effective closure of velocity vectors and accuracy of calculations.

Subroutine GAUSS. This subroutine, presented in detail in Appendix E, performs the Gauss reduction on the matrix product  $AX=B$ , as generalized by set 55, using previously calculated a-coefficients and b-values from ACOE and BCOE, respectively. The argument MATRIX controls the size of the matrix solved. The arrays AY(I,J) and BEE(I) are implicitly transferred through COMMON. (Note: this subroutine is not restricted to linkage analysis; it is useable on any linear, square matrix). The program schematic is shown in Figure 14. The program symbols and steps agree well with those used in presentation of the theory of Gauss reduction; they require no further explanation. The phrase "adjusted matrix" used in the program schematic, Figure 14, may, however, require defining. It is best explained by example. Set (55) represents the original matrix; then set (55) manipulated so the maximum  $|a_{ij}|$  is in the first row and column is the first adjusted matrix. On this matrix perform steps 2) and 3) of the Gauss reduction, drop row and column one and perform step 1). The result is the second adjusted matrix (similar to set (56) with row and column one and  $a''_{22}$  being maximum  $|a_{ij}|$ ). If the original matrix has n rows or columns,

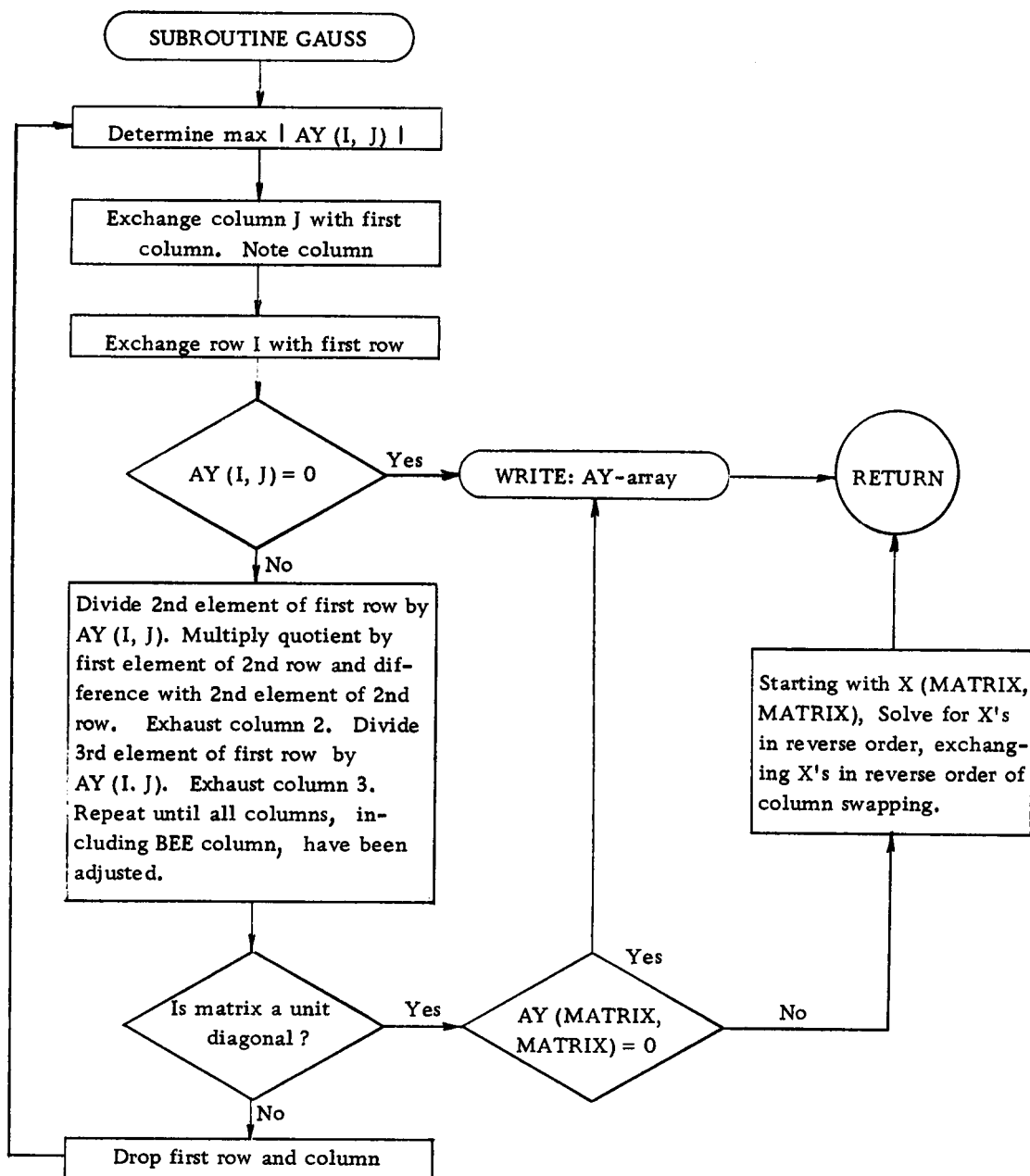


Figure 14. Schematic flow diagram for subroutine GAUSS



there will be  $n$  adjusted matrices, the first having  $n \times n$  terms and the  $n$ th having one term.

In the Gauss reduction, the first row of the matrix is divided by the first element of that row,  $a'_{11}$ . Under certain conditions, such as at a linkage singularity, the remaining coefficients of the adjusted matrix are zero, including  $a'_{11}$ . This warns of a singular matrix. The reason for indeterminacy in the system may be due to a mathematical singularity or a result of an error in the input data. To assist in analyzing the nature of the problem, instructions are included to write out the AY-array, BEE-array, and COLSWP.

If division by zero is not encountered, the subroutine computes the complete  $\dot{x}$ -array, which may be either angular rate or acceleration, and transfers the data and control back to program MAIN.

### The Angle Computation Subroutine

The angle computation subroutine, called ANGLE, is presented in detail in Appendix F. Its purpose is to calculate new approximate link-vector directions from acceleration data and to refine these angular approximations using the angular velocities calculated from the approximations.

The argument MATRIX controls the size of the NORDER array to be used in making the calculations. If argument CNTRL=1, angle increments are calculated using equation (17), repeated here

for convenience:

$$\theta = \alpha_0 \frac{\Delta t^2}{2} + \omega_0 \Delta t + \theta_0 \quad (17)$$

Equation (17) presumes  $\alpha$  = constant over time increment DELTEE; consequently, this mode is used only for the initial approximations.

If CNTROL = 2, refined approximations are made using the average between new and old values of  $\omega$ ; this equation is (19), repeated here for convenience:

$$\theta = (\omega + \omega_0) \frac{\Delta t}{2} + \theta_0 \quad (19)$$

Equation (19) requires that values obtained from (17) are used in ACOE - BCOE - GAUSS to obtain  $\omega$ . The data arrays NORDER, X, THETA, OMEGA, AND THETA2 are implicitly transferred by way of COMMON.

Since some values of NORDER are repeated (see Table II), some angle approximations are repeated. The time-cost of repeating a few calculations is outweighed by the convenience of simpler logic and less stored data.

Angle values are returned to MAIN by way of COMMON.

### Translational Kinematics Subroutine

The last subroutine to be discussed computes the translational kinematic data and is called KINE. The program appears in detail in Appendix G; the program schematic is shown in Figure 15.

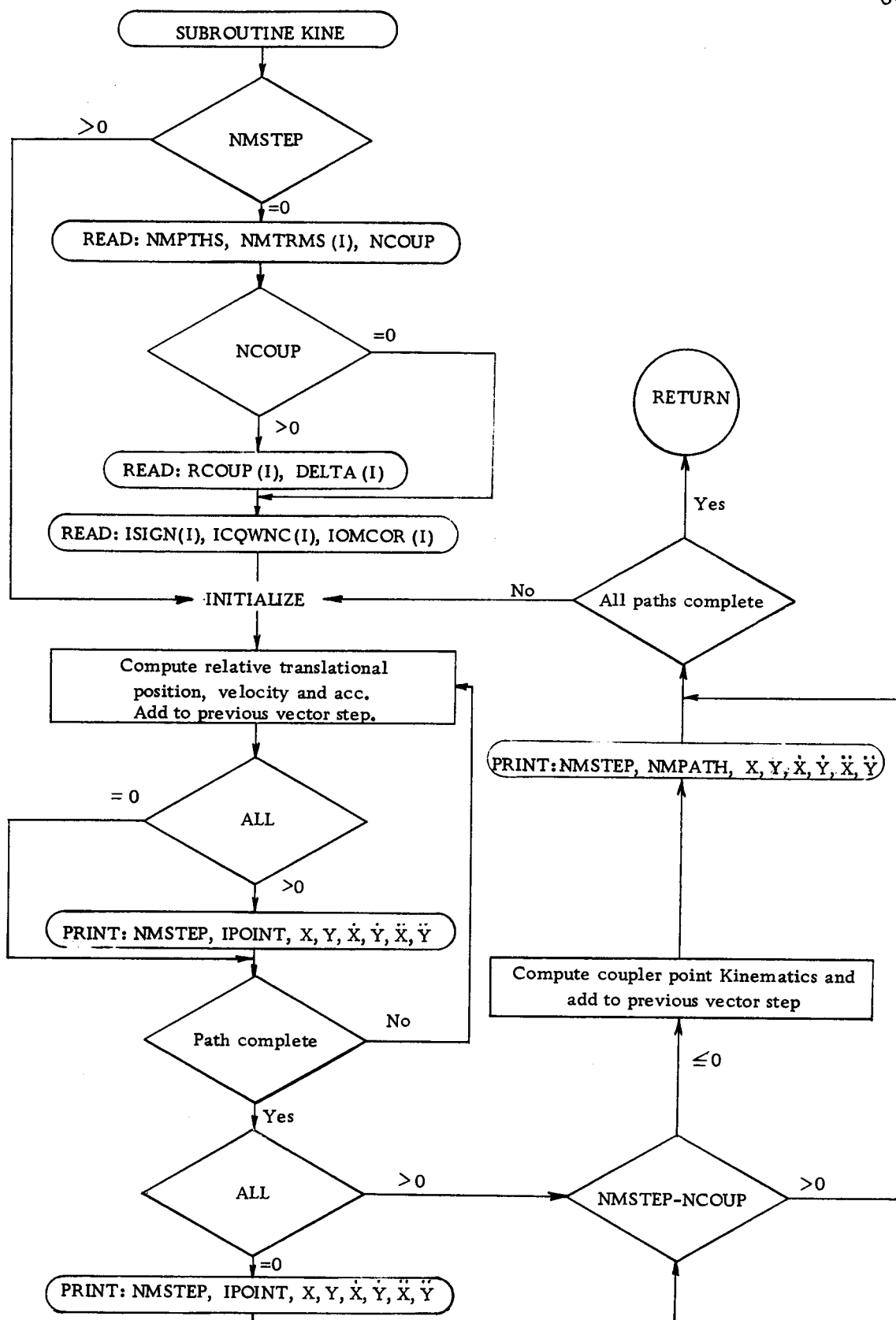


Figure 15. Schematic flow diagram for subroutine KINE

Fundamentally the program is merely an algorithm based upon equations (3), (4), and (20) through (23). These equations provide the translational position, velocity, and acceleration of one end of a vector relative to the other end. The translational kinematics of one point, then, can be described relative to another point by adding like kinematic components along a vector path from the reference point to the point in question. If the computed data is to be used in equations of dynamics, the reference point must not have motion relative to absolute space. This is achieved by placing the reference point for the linkage at a frame pin. Velocity and acceleration magnitudes are unaffected by the reference point as long as it is fixed. Position data, however, is affected. For consistency, it is suggested that the frame pin of the drive link be used as reference for all linkages.

Data from program MAIN is transferred through the subroutine's argument (i. e., NMSTEP, ALL) and by way of COMMON.

The data read in is defined as follows:

- 1) NMPTHS -- control data: instructions as to how many distinct vector paths will be traversed.
- 2) NMTRMS -- control data: instructions as to how many vectors there are in each path.
- 3) NCOUP -- control data: instructions as to how many coupler points are to be defined kinematically. (A coupler point is any point other than the origin or terminus of a vector; in linkage parlance, a coupler point is any point on a moving member which is not at a joint. )

- 4) RCOUP -- computational data: the distance, in consistence units, from the last point on a vector to the coupler point. RCOUP(N) will be associated with path N; note, however, that if there is one coupler point it will be associated with path number one, and so forth.
- 5) DELTA -- computational data: the angle, in degrees, which, when added to the direction of the last vector step, gives the direction of the coupler point. Ordering must be consistent with RCOUP.
- 6) ISIGN -- computational data: must be  $\pm 1$ . It provides polarity to the traversed vector along the vector path. Used with ICQWNC and IOMCOR.
- 7) ICQWNC -- control data: This array provides the subscripts of the vectors traversed along a path.
- 8) IOMCOR -- control data: the array correlating the vectors, on a vector path, with the appropriate OMEGA subscripts. If the vector is a frame vector, the IOMCOR value is zero.

Caution: For reasons of program simplicity, no provisions are made for a path including the drive-link; consequently, such a path cannot be used.

Output of subroutine KINE is:

- 1) NMSTEP -- defined in MAIN
- 2) IPOINT -- the subscript of the vector for which translational data will be provided. If positive, the data refers to the vector head; if negative, it refers to the tail.

- 3) X -- (not the same as in GAUSS) the right-positive horizontal distance from the reference point, or beginning of vector path.
- 4) Y -- the cartesian compliment of X; positive up.
- 5) XDOT -- the horizontal velocity of IPOINT; positive right.
- 6) YDOT -- the vertical velocity of IPOINT; positive up.
- 7) XDBLDT -- the horizontal acceleration of IPOINT; positive right.
- 8) YDBLDT -- the vertical acceleration of IPOINT; positive up.

If ALL = 1, the items listed will be read out for each vector step. If ALL = 0, only the data for the last point will be output. If NCOUP > 0, kinematic data for the last point will be followed by data for the coupler point, until NCOUP is exhausted.

## VALIDATING TEST CASES

In this section, the computer results of the analysis of three linkages will be presented. The first linkage test case is that of the eleven-bar (ten moving-bar) system schematically represented in Figure 11 and which was much discussed in the section "Kinematic Evaluation of Complex Linkages." The analysis of this linkage will be used to validate the analytic techniques, as well as demonstrate the capability of the FORTRAN programs and their methods of use. The last two test cases are shown in Figures 8 and 9. These two cases, discussed in detail in the subsection "Mathematical Singularities" will be used to demonstrate program response to a linkage configuration having either no solution or two; i. e., a singular state.

### Eleven-bar Linkage Test Case

This linkage is comparable in complexity to the industrial problem cited in the introduction. It also represents the most complex linkage which may be analyzed by the established programs, because of storage limitations. Consequently, it is a significant test case.

A scaled vector polygon of the test linkage is shown by the solid-line diagram in Figure 16. This polygon, or something similar, is required in the analysis to provide initial-position data. After

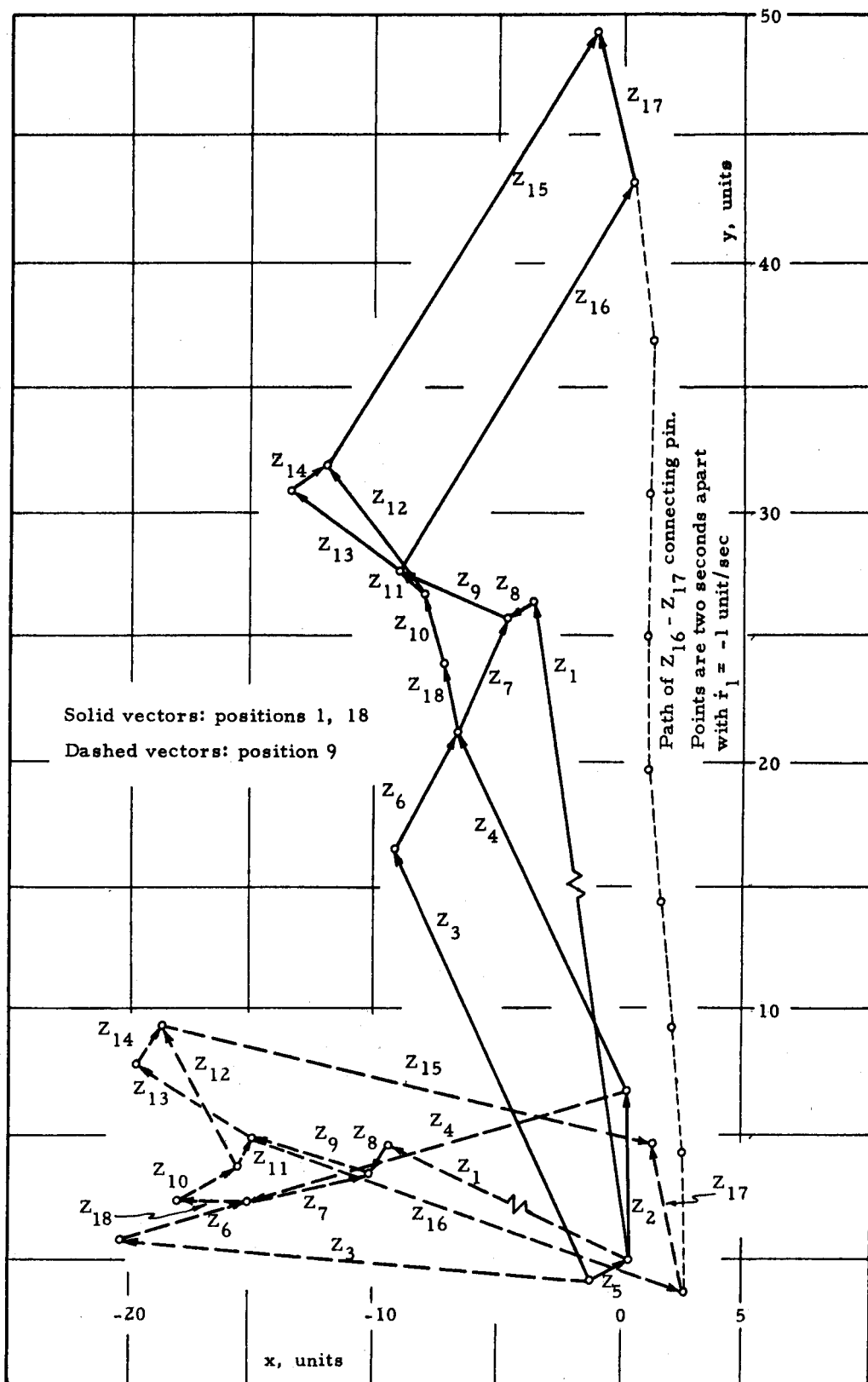


Figure 16. Eleven-bar linkage vector diagram.



scaling lengths and angles, a new exaggerated polygon such as in Figure 11 should be drawn. The new polygon facilitates the writing of accurate vector loop equations for angular analyses, and simplifies the selection of a vector path for translational analyses.

Required Data Cards. Reference to the appendices and the previous section shows that the required MAIN program input data is:

Card 1 -- MATRIX, IPTYPE, MXSTEP, NUMCYC, MAXR,  
 DELTEE, DRIVER, SIG, IFRAME, HYST,  
 TRAKIN, ALL

Cards 2,3 -- (R(I), I = 1, MAXR)

Cards 4,5 -- (THETA(I), I=1, MAXR)

Cards 6 to 10 -- ((NORDER(I, J), J = 1, MATRIX), I = 1,  
 MATRIX)

Figure 11 shows a possibility of five vector loops.

Gruebler's equation, (24), proves the system has one degree of freedom. Substitution into Paul's equation, (25), provides the answer of five vector loops, which agrees with observation. That there are five vector loops means  $MATRIX = 10$ . The ram drive requires that  $IPTYPE = 1$ . Eighteen vectors means  $MAXR = 18$ . If the loop equations are written as (26) through (30), a table such as Table II can be made which shows that  $SIG = 0, 0, 1, 1, 0, 0, 0, 0, 0, 0$ . The schematic in Figure 11 shows the frame vectors to be  $z_2$  and  $z_5$ ;

therefore, IFRAME = 2, 5. Figure 16 shows the ram contracts with a rate of 1 unit/ second; therefore, DRIVER = -1.0. The remaining data for card 1, MXSTEP, NUMCYC, DELTEE, HYST, TRAKIN, and ALL, are arbitrary. MXSTEP is selected as 9 and NUMCYC is set to 4. DELTEE is chosen as 2.0 (two seconds per interval). Since an estimate of repeatability is desired, HYST is made equal to 1. The translational position, velocity, accelerations of certain points is desired so TRAKIN = 1. To minimize calculations, the translational kinematics of every intermediate step should be printed; consequently, ALL is set to 1. These data are entered on the first input data card as shown in Table III, Part I. Positions and decimalization is determined by program MAIN in Appendix B.

The next data of concern are the R's, the vector lengths, and the THETA's, the vector argument in degrees and measured positive counterclockwise from the horizontal. These data are determined from Figure 16 and are entered on cards 2 through 5, also shown in Table III, Part I.

The remaining data required by program MAIN are the NORDER values -- that is, the vector subscripts in the velocity vector loops. These are taken from Table II and entered on cards 6 through 10, as shown in Table III, Part 1.

Since translational kinematics were desired and TRAKIN is set to 1, subroutine KINE requires additional input data to describe the vector paths desired.

Card 1 -- NMPTHS, (NMTRMS(I), I = 1, NMPTHS), NCOUP

Card 2 -- (ISIGN(J), J=1, NMTRMS)

Card 3 -- (ICQWNC(J), J = 1, NMTRMS)

Card 4 -- (IOMCOR(J), J = 1, NMTRMS)

The total number of cards varies, as with program MAIN, according to what is desired. In this test case, a path is selected which will include every pin but the forward end of the ram (this pin's position is given separately in MAIN). Consequently, NMPTHS = 1 and there is only one value of NMTRMS, which is 15. No coupler point kinematics is desired so NCOUP is left blank (this also means no cards are provided which give RCOUP or DELTA). The vector path is described by ISIGN and ICQWNC, cards 2 and 3. The path selected is, starting at the frame pin of the ram,  $z_2 + z_4 + z_{18} + z_{10} + z_{11} + z_{16} + z_{17} - z_{15} - z_{14} - z_{13} - z_9 - z_7 - z_6 - z_3 + z_5$ . ISIGN(I), therefore, becomes 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, and ICQWNC(I) is 2, 4, 18, 10, 11, 16, 17, 15, 14, 13, 9, 7, 6, 3, 5. To be able to make the correct calculations, the vector steps must be correlated to the angular velocity terms. This is done in card 4. The values to enter are established by correlating card 3 to Table II; IOMCOR(I) becomes 0, 4, 4, 6, 6, 6, 9, 10, 7, 5, 5, 2, 2, 3, 0 (zeros are entered for frame vectors). The resulting data cards are shown in Table III, Part 1.

Results of Computer Analysis. The preceding data cards were added to the computer program decks described in Appendices B through G. The ensemble was submitted to the Oregon State

Table III. Eleven-bar linkage data.

Part 1. Input data key, input data, output data key

## PROGRAM MAIN INPUT DATA CARDS

1 MATRIX, IPTYPE, MXSTEP, NUMCYC, MAXR, DELTEE, DRIVER, SIG, IFRAME, HYST, TRAKIN, ALL  
 2,3 R(1) THROUGH (18) IN UNITS  
 4,5 THETA2(1) THROUGH (18) IN DEGREES  
 6 THROUGH 10 NORDER(1,1) THROUGH (5,20)  
 SUBROUTINE KINE INPUT DATA CARDS  
 1 NMPHIS, NMTBMS(NMPHIS), NCOUP  
 2 ISIGN(1) THROUGH (NMTRMS(1))  
 3 ICQWNC(1) THROUGH (NMTRMS(1))  
 4 IOMCOR(1) THROUGH (NMTRMS(1))

10	1	9	4	18	2.0	-1.0	0	0	1	1	0	0	0	0	0	0	25111
26.80	6.70	19.10	16.05	1.80	5.30	5.10	1.30	4.80	2.80								
1.40	6.60	5.60	1.90	20.40	18.40	6.14	2.83										
98.00	90.00	114.40	115.77	31.30	62.50	64.20	217.70	158.20	105.70								
141.70	126.63	142.50	32.20	57.50	58.40	104.27	97.71										
0	0	-6	0	-3	0	4	0	0	0	0	0	0	0	0	0	0	0
1	0	-6	-7	-3	0	0	0	8	0	0	0	0	0	0	0	0	0
0	0	7	0	0	0	-18	0	9	0	-10	-11	0	0	0	0	0	0
0	0	0	0	0	0	0	0	13	0	11	0	14	0	-12	0	0	0
0	0	0	0	0	0	0	0	-13	0	16	0	-14	0	0	0	17	0
1	15																
1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	
2	4	18	10	11	16	17	15	14	13	9	7	6	3	5			
0	4	4	6	6	6	9	10	7	5	5	2	2	3	0			

## OUTPUT DATA

1,3,5,7 THEAPP (1) THROUGH (10) (1ST, 2ND, 3RD, 4TH APPROX), IN DEGREES  
 2,4,6,8 THEAPP (11) THROUGH (18)  
 9 BUFF1(1) THROUGH (10) (VELOCITY PERCENT RESIDUAL)  
 10 BUFF1(1) THROUGH (10) (ACCELERATION PERCENT RESIDUAL)  
 11 THROUGH 25 NMSTEP, IPOINT, X, Y, XDOT, YDOT, XDBLDT, YDBLDT  
 26 NMSTEP, R(1), THETA2(1)  
 27,28 THETA2(1) THROUGH (18) IN DEGREES  
 29 BUFF1(1) THROUGH (10) (BUFF1 = OMEGA) IN DEGREES/SEC  
 30 ALPHA(1) THROUGH (10) IN DEGREES/SEC\*SEC

Part 2. Output data, position one

98.00	90.00	114.40	115.77	31.30	62.50	64.20	217.71	158.20	105.70
141.70	126.63	142.50	32.20	57.50	58.40	104.27	97.71		
-.80E-05	.37E-05	-.81E-05	-.17E-06	.74E-05	-.35E-05	.18E-05	-.18E-05	.00E-99	.14E-05
.00E-99	-.12E-05	-.52E-06	.00E-99	-.14E-05	.12E-05	.20E-05	.22E-05	.00E-99	.14E-05
1	2	.00	6.70	.00	.00	.00	.00		
1	4	-6.97	21.15	-2.56	-1.23	.78	-.18		
1	18	-7.35	23.95	-3.05	-1.30	.90	-.25		
1	10	-8.11	26.65	-2.52	-1.15	.75	-.41		
1	11	-9.21	27.52	-2.35	-.93	.74	-.51		
1	16	.42	43.19	.75	-2.84	-.67	-.49		
1	17	-1.08	49.14	1.13	-2.75	-.99	-.60		
1	-15	-12.04	31.93	-1.99	-.75	.44	-.72		
1	-14	-13.65	30.92	-2.09	-.59	.54	-.83		
1	-13	-9.21	27.51	-2.35	-.93	.74	-.51		
1	-9	-4.75	25.73	-2.48	-1.27	.83	-.20		
1	-7	-6.97	21.14	-2.56	-1.23	.78	-.18		
1	-6	-9.42	16.44	-2.63	-1.19	.73	-.15		
1	-3	-1.53	-.95	-.00	-.00	.00	.00		
1	5	.00	-.01	-.00	-.00	.00	.00		
1		26.80	97.99						
97.99	89.99	114.39	115.76	31.29	62.49	64.19	217.69	158.19	105.69
141.69	126.62	142.49	32.19	57.49	58.39	104.26	97.70		
5.54	-.93	8.69	10.15	-4.33	-11.36	-5.68	-5.73	-3.63	-10.41
-1.19	-.66	-1.81	-2.24	3.84	3.79	4.33	3.77	3.16	3.60

Table III. (continued)

## Part 3. Output data, position two

106.69	90.00	126.17	131.60	31.30	59.29	60.99	216.72	157.22	90.55
126.56	122.71	141.52	29.48	43.86	43.25	103.32	113.54		
107.81	90.00	129.65	133.57	31.30	60.08	61.78	213.16	153.66	87.50
123.50	119.29	137.96	25.97	40.89	40.19	100.21	115.51		
107.62	90.00	129.45	133.32	31.30	59.93	61.63	213.57	154.06	87.83
123.83	119.67	138.36	26.34	41.22	40.52	100.58	115.26		
107.63	90.00	129.46	133.33	31.30	59.95	61.65	213.51	154.01	87.79
123.79	119.63	138.31	26.30	41.19	40.49	100.52	115.27		
.26E-03	.11E-03	.33E-03	.80E-04	-.84E-02	-.46E-03	-.24E-02	-.10E-02	-.31E-03	.48E-03
-.13E-05	-.27E-05	.32E-06	-.13E-05	.27E-05	.10E-05	-.14E-05	-.28E-06	.17E-05	.26E-05
2	2	.00	6.70	.00	.00	.00	.00		
2	4	-11.01	18.37	-1.50	-1.42	.35	-.02		
2	18	-12.22	20.93	-1.83	-1.57	.41	-.05		
2	10	-12.11	23.73	-1.52	-1.59	.34	-.08		
2	11	-12.89	24.89	-1.38	-1.50	.33	-.12		
2	16	1.09	36.84	-.02	-3.09	-.14	.06		
2	17	-.02	42.88	-.01	-3.09	-.22	.04		
2	-15	-15.37	29.44	-1.39	-1.52	.23	-.15		
2	-14	-17.07	28.60	-1.39	-1.51	.25	-.20		
2	-13	-12.89	24.87	-1.38	-1.50	.33	-.12		
2	-9	-8.58	22.77	-1.38	-1.49	.37	-.03		
2	-7	-11.00	18.28	-1.50	-1.42	.35	-.02		
2	-6	-13.65	13.69	-1.63	-1.34	.34	-.01		
2	-3	-1.51	-1.04	-.00	.00	.00	-.00		
2	5	.02	-.11	-.00	.00	.00	-.00		
2	24.80	107.62							
107.62	89.99	129.45	133.33	31.29	59.94	61.64	213.51	154.01	87.79
123.79	119.63	138.31	26.30	41.19	40.49	100.53	115.27		
4.08	-1.61	6.36	7.40	.14	-6.53	-.20	-1.25	-.10	-5.88
-.44	-.16	-.77	-.86	1.09	1.39	1.56	1.13	.80	1.28

## Part 4. Output data, position nine

153.25	90.00	174.56	195.92	31.30	15.93	17.63	224.45	164.95	28.06
64.06	119.28	149.25	58.84	-12.83	-19.24	102.43	177.86		
154.09	90.00	174.23	195.87	31.30	14.91	16.61	223.66	164.16	27.50
63.51	118.48	148.45	57.84	-13.40	-19.79	101.65	177.81		
154.34	90.00	174.13	195.87	31.30	14.57	16.27	223.41	163.91	27.31
63.32	118.23	148.21	57.54	-13.60	-19.98	101.41	177.81		
154.42	90.00	174.10	195.87	31.30	14.46	16.16	223.33	163.83	27.25
63.25	118.15	148.13	57.44	-13.66	-20.04	101.33	177.81		
.80E-02	-.11E 00	.15E-01	-.85E-01	-.43E-02	.37E-01	.30E-02	-.99E-03	-.47E-03	-.52E-03
-.38E-05	-.84E-05	.19E-05	-.19E-05	.49E-05	.58E-06	.00E-99	.13E-05	.19E-05	.17E-05
9	2	.00	6.70	.00	.00	.00	.00		
9	4	-15.43	2.31	.19	-.67	.00	.09		
9	18	-18.26	2.42	.18	-.79	.01	.10		
9	10	-15.77	3.70	.32	-1.06	.01	.02		
9	11	-15.14	4.95	.45	-1.12	.04	-.00		
9	16	2.13	-1.36	-.21	-2.97	-.32	-.40		
9	17	.93	4.66	.13	-2.90	-.09	-.38		
9	-15	-18.88	9.48	.62	-.90	.24	.12		
9	-14	-19.91	7.88	.60	-.89	.17	.17		
9	-13	-15.15	4.92	.45	-1.12	.04	-.00		
9	-9	-10.54	3.58	.39	-1.35	-.01	-.18		
9	-7	-15.44	2.17	.19	-.67	.00	.09		
9	-6	-20.58	.85	.00	.04	.03	.37		
9	-3	-1.58	-1.11	.00	.00	.00	-.00		
9	5	-.04	-.18	.00	.00	.00	-.00		
9	10.80	154.44							
154.44	89.99	174.08	195.86	31.29	14.43	16.13	223.29	163.79	27.23
63.23	118.11	148.09	57.40	-13.69	-20.06	101.30	177.80		
5.73	-8.04	-.14	2.49	-2.86	-6.13	-.60	-2.94	-3.34	-5.80
2.10	-2.90	-1.14	-.31	-2.24	-1.56	-2.57	-2.15	-2.16	-1.61

Table III.(continued)

## Part 5. Output data, position 17

107.80	90.00	129.70	133.63	31.30	59.98	61.68	212.85	153.35	87.09
123.09	118.99	137.64	25.51	40.53	39.78	100.02	115.57		
107.65	90.00	129.49	133.56	31.30	59.97	61.67	213.43	153.93	87.72
123.72	119.55	138.23	26.21	41.12	40.42	100.46	115.30		
107.63	90.00	129.46	133.34	31.30	59.95	61.65	213.51	154.01	87.79
123.79	119.62	138.31	26.29	41.18	40.49	100.52	115.28		
107.63	90.00	129.46	133.34	31.30	59.95	61.65	213.52	154.02	87.79
123.80	119.63	138.32	26.30	41.19	40.49	100.53	115.27		
-.14E-04	.00E-99	-.65E-04	-.20E-04	.92E-03	.37E-04	.42E-03	.18E-03	.56E-04	-.90E-04
.00E-99	-.27E-05	.11E-05	.13E-05	.18E-05	.18E-06	.57E-06	.44E-06	.88E-06	.89E-06
17 2	.00	6.70	.00	.00	.00	.00	.00	.00	.00
17 4	-11.01	18.37	1.50	1.42	.35	-.02			
17 18	-12.22	20.93	1.83	1.57	.41	-.05			
17 10	-12.11	23.73	1.52	1.59	.34	-.08			
17 11	-12.89	24.89	1.38	1.50	.33	-.12			
17 16	1.09	36.84	.02	3.09	-.14	.06			
17 17	-.02	42.87	.01	3.09	-.22	.04			
17 -15	-15.37	29.44	1.39	1.52	.23	-.15			
17 -14	-17.07	28.60	1.39	1.51	.25	-.20			
17 -13	-12.89	24.87	1.38	1.50	.33	-.12			
17 -9	-8.58	22.77	1.38	1.49	.37	-.03			
17 -7	-11.00	18.28	1.50	1.42	.35	-.02			
17 -6	-13.65	13.69	1.63	1.34	.34	-.01			
17 -3	-1.51	-1.04	.00	-.00	.00	-.00			
17 5	.02	-.11	.00	-.00	.00	-.00			
17	24.80	107.62							
107.62	89.99	129.46	133.33	31.29	59.95	61.65	213.51	154.01	87.79
123.79	119.63	138.31	26.30	41.19	40.49	100.53	115.27		
-4.08	1.61	-6.36	-7.40	-.14	6.53	.20	1.25	.10	5.88
-.44	-.16	-.77	-.86	1.09	1.39	1.55	1.13	.80	1.28

## Part 6. Output data, position 18

98.57	90.00	115.19	116.80	31.30	62.86	64.56	215.40	155.90	103.66
139.67	124.42	140.20	29.84	55.54	56.36	102.34	98.74		
98.17	90.00	114.61	116.04	31.30	62.63	64.33	217.32	157.82	105.45
141.45	126.27	142.12	31.85	57.23	58.15	103.92	97.98		
98.04	90.00	114.46	115.84	31.30	62.53	64.23	217.66	158.16	105.67
141.67	126.59	142.45	32.16	57.46	58.37	104.21	97.78		
98.01	90.00	114.42	115.79	31.30	62.51	64.21	217.70	158.19	105.69
141.69	126.62	142.49	32.19	57.49	58.39	104.26	97.73		
-.56E-02	.13E-02	.28E-02	.80E-04	.18E-01	.26E-03	.36E-03	.21E-03	.68E-03	-.56E-03
.71E-06	.63E-06	.53E-06	.11E-05	.28E-05	.63E-06	.20E-05	.83E-06	.32E-06	.45E-05
18 2	.00	6.70	.00	.00	.00	.00	.00	.00	.00
18 4	-6.98	21.15	2.56	1.23	.78	-.18			
18 18	-7.36	23.95	3.05	1.30	.90	-.25			
18 10	-8.11	26.65	2.52	1.15	.75	-.40			
18 11	-9.21	27.52	2.35	.93	.74	-.51			
18 16	.42	43.19	-.75	2.84	-.67	-.49			
18 17	-1.08	49.14	-1.13	2.75	-.99	-.60			
18 -15	-12.04	31.93	1.99	.75	.44	-.72			
18 -14	-13.65	30.92	2.09	.60	.53	-.83			
18 -13	-9.21	27.51	2.35	.93	.74	-.51			
18 -9	-4.75	25.73	2.48	1.27	.83	-.20			
18 -7	-6.97	21.14	2.56	1.23	.78	-.18			
18 -6	-9.42	16.44	2.63	1.19	.73	-.15			
18 -3	-1.53	-.95	.00	-.00	.00	.00			
18 5	.00	-.01	.00	-.00	.00	.00			
18	26.80	98.00							
98.00	89.99	114.40	115.77	31.29	62.50	64.20	217.69	158.19	105.69
141.69	126.62	142.49	32.19	57.49	58.39	104.26	97.71		
-5.53	.93	-8.68	-10.15	4.32	11.35	5.68	5.73	3.63	10.41
-1.19	-.66	-1.80	-2.24	3.84	3.78	4.32	3.77	3.15	3.59

University IBM 1410 digital computer. The computer required about five minutes for loading the routines and an additional 25 minutes to perform the calculations and punch the output data cards (time could be reduced by tape output or on-line printing). The computer rental charge is \$50 per hour, making the cost of kinematically evaluating this extremely complex linkage only \$25, or about one engineering man-hour. The time and cost are impressively small and are a significant improvement over the usual expense and time associated with non-computer methods of linkage analysis.

Because MXSTEP was set at 9 and HYST at 1, the computer performed 18 sets of calculations, nine each way. This moved the linkage's mathematical model through its entire "range of operation" and back to its original position. Although a design analyst working with this linkage might be interested in the data from every step, the program capabilities can be effectively demonstrated by displaying only the first, second, ninth, seventeenth, and eighteenth steps. Positions one and nine are the extremes, and positions 17 and 18 are the return positions of two and one, respectively.

Data for these positions appear in Table III, Parts 2 through 6, and are presented exactly as punched by the computer. The key for determining what each field represents is displayed in Table III, Part 1. An even better key would be the program itself.

Lines three and four of step one and lines nine and ten of

the other steps give the amount, in percent of total path, that the velocity and acceleration vector loops failed to close. These values, called residues ( $\rho$ ) provide an estimate of the error in determining angular rates and accelerations. Even values are  $\rho(\dot{x})$  or  $\rho(\ddot{x})$  and odd-values are  $\rho(\dot{y})$  or  $\rho(\ddot{y})$ . The largest velocity residue is found in the x-velocity components of loop one, step nine, and is only -0.11 percent. Generally, the velocity residues are smaller than  $\pm 0.01$  percent. Very few of the acceleration residues are greater than  $\pm 0.00001$  percent. The reason for the acceleration residues being smaller is that angles used in angular acceleration computation have enjoyed one more iteration than those used to calculate angular velocity. Additionally, there are a greater number of terms used to calculate acceleration vector loops which would tend to dilute the effect of a particular term being in error. It is also interesting to note that, whereas step nine residues are larger than residues of step one, step 18 residues are generally an order of magnitude smaller than for step nine. This would imply that residues are more affected by the values of angles, rates, etc., of a particular position than they are by cumulative errors stepwise.

It is pertinent to subsequent discussion that a computer output peculiarity be noted. The computer carries eight significant figures during computation, but truncates the results when the values are printed out. The amount of truncation depends on the value and



the field specification of the program. Most output fields in the programs specify only two decimal places. This causes all lesser significant figures to be dropped. For example, consider the four numbers that might be carried in core storage as 98.090000, 98.099999, 98.100000, and 98.109999. If the program called for these numbers to be printed or punched, they would appear as 98.09, 98.09, 98.10 and 98.10, respectively. Quite obviously, there is actually more difference between the first and second and the third and fourth numbers than between the second and third, although the output would belie the fact. Consequently, any two numbers (except the residues) agreeing within  $\pm 0.01$  should be considered to be in complete agreement.

Considering the foregoing comments, data from steps one and 18 can be compared to provide the most convincing validation of the analytic techniques and the FORTRAN angular programs which have been developed in this paper -- except for the residues, there is complete agreement between the two sets of data, although they are separated by 17 steps. The magnitude of travel over these 17 steps can be better appreciated by referring to Figure 16. In this diagram, the solid vectors represent steps one and 18, and the dashed vectors represent step nine. The mathematical model moved from the extended position, to the folded, back to the extended.

The data used for drawing the two vector arrays of

Figure 16 were obtained from subroutine KINE translational data for steps one and nine. The translational vector path over this linkage was carefully selected to begin and terminate (item marked 5) at the frame pin of the ram; in addition, the path was allowed to cross over itself at two points, 4 and -7, and 11 and -13. (It will be recalled that these numbers are the vector subscripts of the traversed vectors. The minus sign means the step terminated at the vector tail, whereas, the lack of sign means termination at the head. Refer, also, to ISIGN and ICQWNC). As a result, if the translational kinematics are to be valid, the pairs 4 and -7, and 11 and -13, must have identical values. The item marked 5 must have all zero values since it represents the origin. Steps one and 18 show this to be true, but steps 2, 9, and 17 indicate a slight variation between position values for pairs 4 and -7, and 11 and -13; they also show that the position value for item 5 is not zero as it should be. However, that these positional variations are in reality quite small and are cumulative over the vector path, can be seen from studying the y-position data of step nine, the data most in variance. First, the crossover pair 11 and -13 are separated on the vector path by very little and, as a result, have values with little difference. Pairs 4 and -7 are separated by nearly the entire path and are different by almost as much as the error in item 5 which does represent the entire path. These facts indicate the error is cumulative over the path. Next, the

smallness of the error is shown by the error in the y-position data of step nine being only 0.4 percent of the total path. That the error is not cumulative from linkage step to step is proven by the data of steps 1, 2, 17, and 18; the errors are identical for steps one and 18 and for steps two and 17. Apparently, the error is a function only of the angles. This agrees to a large extent with the variation observed in the velocity and acceleration residues previously discussed. One strange fact with no obvious explanation is that whereas the y-position error for step nine is about 0.4 percent, the x-position error is only 0.04 percent, one order of magnitude smaller. Similar differences appear in the other steps. The only difference in calculation is that the cosine functions are used to determine x and sine functions are used to determine y. This, also, agrees with the velocity residues which show about one order of magnitude difference between sums using cosine functions and sums using sine functions. For example, consider line nine, of step nine, which displays the velocity vector loop residues; the odd values employ cosine functions whereas the even values rely on sine functions. The correlation is striking and suggests that the computer built-in subroutines for determining sines and cosines are sufficiently different in structure that the sine generator has more inherent error than the cosine generator.

### The Mathematical Singularity Test Cases

The two test case linkages to be discussed in this subsection are shown pictorially in Figures 8 and 9, representing the Type I and Type II singularities, respectively.

Type I Singularity. For the Type I singularity, the linkage is defined by the position vectors  $z_1$  (the driver),  $z_2$  (the floater),  $z_3$  (the follower), and  $z_4$  (the frame). The lengths are 1.0, 2.0, 1.0, and 2.0, respectively. The initial angles are  $170^\circ$ ,  $0^\circ$ ,  $-10^\circ$  (opposite sense to that shown in Figure 8) and  $180^\circ$ , respectively. Quite obviously, if a step is calculated at  $\theta_1 = 180^\circ$ , the program, being appropriately coded, will detect an attempt to divide by zero, and, after printing out the existing AY-array and then upon restoring the original AY-array and printing it, will terminate further calculations. In a complex linkage analysis, however, the probability is very small that the calculations would exactly (to eight significant figures) fall on the singular position. This suggests that a practical demonstration would have the singular position fall exactly between two calculated positions. Consequently, arbitrarily setting the crank rate at  $5^\circ/\text{sec}$  and counterclockwise, the time interval is made 0.8 seconds. This will place the singular position between the  $178^\circ$  and  $182^\circ$  positions of the crank. To complete the demonstration, the program will be

instructed to continue to  $186^\circ$  on the crank, and then return, if it can, to the original position. Table IV presents the results of the calculations using the foregoing criteria. The key to reading the data is:

Line 1 --  $\theta_1, \theta_2, \theta_3, \theta_4$

Line 2 --  $\rho(\dot{y}), \rho(\dot{x})$

Line 3 --  $\rho(\ddot{y}), \rho(\ddot{x})$

Line 4 -- step number,  $r_1, \theta_1$

Line 5 --  $\theta_1, \theta_2, \theta_3, \theta_4$

Line 6 --  $\omega_2, \omega_3$

Line 7 --  $a_2, a_3$

The data shows no abrupt rise of any values for step four which is just across the singularity, nor even for the next step, number 5 (step 6 is the same position as step 5, but with the crank reversed), but on the reverse path anomalies develop. First, angular data for each step are significantly different from the values calculated for the comparable position during counterclockwise movement of the crank; i. e., hysteresis has developed. Next, the x-velocity residues are beginning to be appreciably large, that is, two to three percent as compared to the maximum of 0.1 percent for the considerably more complex eleven-bar system. Finally, the simple system failed to return to its original state. It can be deduced that a more complex system would have even more obvious anomalies which

Table IV. Type I singularity output data.

Column 1				Column 2			
170.00	.00	-10.00	180.00	186.00	.00	5.99	180.00
.58E-06	.32E-05			-.19E-03	.17E-01		
.37E-05	.66E-06			-.56E-06	.65E-06		
1	1.00	169.99		6	1.00	185.99	
169.99	.00	-9.99	179.99	185.99	.00	5.99	179.99
-.00	5.00			.00	-5.00		
-.00	.00			.00	-.00		
174.00	-.00	-6.00	180.00	182.00	.00	1.98	180.00
.00E-99	.10E-03			-.22E-03	.18E 00		
.62E-06	.13E-05			-.23E-05	.13E-05		
2	1.00	173.99		7	1.00	181.99	
173.99	-.00	-5.99	179.99	181.99	.00	1.98	179.99
-.00	5.00			.01	-5.02		
-.00	.00			.06	-.13		
178.00	-.00	-1.99	180.00	178.00	.04	-2.08	180.00
.57E-05	.39E-02			-.38E-02	.31E 01		
.15E-05	.13E-05			-.12E-05	.00E-99		
3	1.00	177.99		8	1.00	177.99	
177.99	-.00	-1.99	179.99	177.99	-.02	-1.95	179.99
-.00	5.00			-.09	-4.80		
-.00	.00			.48	-.96		
182.00	-.00	2.00	180.00	174.00	.05	-6.10	180.00
.92E-04	-.76E-01			-.23E-01	.21E 01		
.74E-06	.65E-06			-.21E-05	.65E-06		
4	1.00	181.99		9	1.00	173.99	
181.99	.00	1.99	179.99	173.99	-.07	-5.84	179.99
.00	4.99			-.04	-4.91		
-.01	.02			.07	-.14		
186.00	-.00	6.00	180.00	170.00	-.09	-9.81	180.00
.58E-03	.53E-01			.37E-02	.12E 00		
.58E-06	.00E-99			.00E-99	.64E-06		
5	1.00	185.99		10	1.00	169.99	
185.99	.00	5.99	179.99	169.99	-.07	-9.84	179.99
.00	4.99			.04	-5.09		
-.00	.00			-.04	.09		

should warn the analyst that a singularity had been traversed (or that there were errors in the input data).

Type II Singularity. The Type II singularity, depicted in Figure 9, is the result of the linkage being moved to a state such that it has become a structure. This type of singularity is easily detected since the singular state is generally preceded by a sudden rise in angular rate and acceleration. For this test case the values 1.0, 1.048, 0.684, and 1.0 are used for the vector moduli,  $r_1$ ,  $r_2$ ,  $r_3$ , and  $r_4$ , respectively, and  $110.00^\circ$ ,  $-19.87^\circ$ ,  $-58.60^\circ$ , and  $180.00^\circ$  are used for the initial values of  $\theta_1$ ,  $\theta_2$ ,  $\theta_3$ , and  $\theta_4$  respectively. The moduli are the same as in Figure 9 and have already been shown to create a singularity at  $\theta_1 = 120^\circ$ . The crank velocity was set to  $5^\circ/\text{second}$  counterclockwise, the time interval was made 0.8 seconds, the number of steps was selected as five, the hysteresis mode was chosen, and the number of iterations desired for convergence was set at 3.

The results of this linkage analysis are presented in Table V. The key for reading the table is the same as for the Type I data, except the iteration produces three lines of angles at the beginning. They quite clearly show the approach to, and the passing over of, a singularity. Beginning with the step prior to passing over the singularity, angular velocity and acceleration increase sharply. The data

Table V. Type II Singularity output data.

<u>Column 1</u>				<u>Column 2</u>			
110.00	-19.87	-58.60	180.00	126.00	-1.70	-65.07	180.00
.00E-99	.16E-05			126.00	-1.70	-65.07	180.00
.18E-05	-.20E-05			126.00	-1.70	-65.07	180.00
1	1.00	109.99		.00E-99	-.49E-05		
109.99	-19.86	-58.59	179.99	.12E-05	.00E-99		
-1.50	8.96			6	1.00	125.99	
-.86	1.19			125.99	-1.70	-65.07	179.99
				-1.02	-6.46		
				-.08	-.03		
114.00	-21.35	-51.04	180.00	122.00	-2.55	-70.26	180.00
114.00	-21.46	-50.86	180.00	122.00	-2.55	-70.26	180.00
114.00	-21.48	-50.83	180.00	122.00	-2.55	-70.26	180.00
.59E-02	.48E-02			.10E-05	-.54E-05		
.25E-05	.85E-06			.00E-99	.00E-99		
2	1.00	113.99		7	1.00	121.99	
113.99	-21.49	-50.83	179.99	121.99	-2.55	-70.26	179.99
-2.54	10.45			-1.09	-6.50		
-1.92	2.82			-.09	-.05		
118.00	-24.14	-41.56	180.00	118.00	-3.46	-75.48	180.00
118.00	-24.73	-40.65	180.00	118.00	-3.46	-75.48	180.00
118.00	-25.04	-40.19	180.00	118.00	-3.46	-75.48	180.00
.25E 00	.41E 00			.00E-99	-.12E-05		
.12E-05	-.16E-05			.57E-06	-.13E-05		
3	1.00	117.99		8	1.00	117.99	
117.99	-25.22	-39.92	179.99	117.99	-3.46	-75.48	179.99
-6.78	16.81			-1.16	-6.55		
-14.23	21.64			-.09	-.06		
122.00	-35.20	-19.54	180.00	114.00	-4.42	-80.75	180.00
122.00	-23.54	-37.39	180.00	114.00	-4.42	-80.75	180.00
122.00	-30.74	-26.28	180.00	114.00	-4.42	-80.75	180.00
.20E 02	.50E 02			.00E-99	-.50E-05		
.71E-05	.13E-05			.55E-06	-.16E-05		
4	1.00	121.99		9	1.00	113.99	
121.99	-15.02	-50.42	179.99	113.99	-4.42	-80.75	179.99
32.27	-43.07			-1.25	-6.61		
-61.35	92.85			-.10	-.08		
126.00	-8.84	-55.17	180.00	110.00	-5.46	-86.07	180.00
126.00	-2.06	-64.79	180.00	110.00	-5.46	-86.08	180.00
126.00	-1.71	-65.07	180.00	110.00	-5.46	-86.08	180.00
-.24E-02	.98E-03			-.11E-04	-.48E-05		
.63E-06	.00E-99			.53E-06	-.19E-05		
5	1.00	125.99		10	1.00	109.99	
125.99	-1.70	-65.06	179.99	109.99	-5.46	-86.07	179.99
1.02	6.46			-1.33	-6.68		
-.08	-.03			-.11	-.10		



for the first step beyond the singularity shows velocity residues as high as 50 percent. Angular velocities increase in magnitude but change sign. Angular accelerations increase fourfold over their previously high values. Beyond this point, angular velocities and accelerations drop to very low values. When the crank returns to its original position, the other data show no similarity to the original values. It is much as if the linkage had snapped upon encountering the singularity, which it probably would have, had it been an actual, rather than a mathematical, model. The analyst should encounter no difficulty in detecting a singularity of this sort.

### CONCLUDING REMARKS

The expressed purpose of this paper was to develop analytic techniques and computer programs for the evaluation of plane motion mechanisms. Inasmuch as a mechanism has been defined as a constrained kinematic chain, any evaluation of it must also be kinematic. In addition, the term mechanism has, in this paper, been used to define the principal class of kinematic chains known as linkages. Further, in avoidance of pedantic discussion of rare applications, the developments were restricted to single degree of freedom linkages with turning joints and either a crank or ram drive. The stated purpose, tempered by these practical considerations, has been fulfilled by the preceding sections and has been validated by the kinematic analysis of the eleven-bar linkage and the singularity test cases.

The digital computer programs presented in this paper can be used directly by any analyst having access to an IBM 1410 digital computer. The general language used in programming also makes it quite easy to convert the program to a wide variety of other scientific digital computers. Very likely, the only modifications would be to the input/output instructions. Since digital computers are becoming increasingly available, virtually every linkage analyst could gain access to a computer which would accept the linkage analysis computer programs.

The programs and associated mathematical theory,

however, should not be accepted as an ultimate in linkage analysis but rather as a sound basis for additional development. The program could be modified to accept a variable velocity driver. With greater storage available (such as magnetic tape units, larger core storage, etc.) the programs could be altered to handle more complex linkages or linkages with more than one degree of freedom. Instructions could be added which would cause the computer to modify the linkage geometry and then re-evaluate, and, in this way, obtain a spectrum of linkage designs.

Linkage analysis, albeit important, is but one phase of mechanical design. Having determined the motion and geometric characteristics of the design, the next step would be to analyze the dynamics, static loading, member stress, space budgeting, bearing loading, and so forth. These analyses all would employ data obtained from the linkage analysis programs, but would, however, require the introduction of a third dimension in order to describe volume, mass, moments of inertia, etc.

With these developments, the design analyst would be relieved of an even greater burden of routine calculations allowing him to work more as an engineer and less as a technician. The advance in the development of a complete program would be the conversion of the analyst's "judgement" to computer logic. With this, the computer would analyze the linkage in precise detail, make a decision as to

whether the design were optimum, and if not, make a logical change, and re-evaluate. The system would then be approaching the true goal -- automatic design.

## BIBLIOGRAPHY

1. Bottema, O. On Gruebler's formula of mechanisms. Applied Scientific Research A(21):162-4. 1950.
2. Churchill, R.V. Complex variables and applications. New York, McGraw-Hill, 1960. 297 p.
3. Faires, V.M. Kinematics. New York, McGraw-Hill, 1959, 468 p.
4. Hildebrand, F.B. Introduction to numerical analysis. New York, McGraw-Hill, 1956. 511 p.
5. International Business Machines Corporation. IBM 1410/ 7010 operating system (1410-PR-155)-FORTRAN. (Form C28-0328-1). Poughkeepsie, 1963. 49 p.
6. Paul, B. Unified criterion for degree of constraint of plane kinematic chains. ASME Transactions - Journal of Applied Mechanics 27E(1):196-200. 1960.
7. Wylie, C.R. Advanced engineering mathematics. New York, McGraw-Hill, 1960. 696 p.

## APPENDICES

## Appendix A. Program CLOSURE

```

MON$$      EXEQ FORTRAN.....CLOS
            DIMENSION R(20), THETA(20), NORDER(5,20), B(10), MAKEUP(5)
            1      , Nums(10)
5  FORMAT (2I3)
6  FORMAT (10F8.2)
7  FORMAT (20I3)
36 FORMAT (10F8.3)
45 FORMAT (5I3)
100 FORMAT (2I4,4F8.2)
1  READ (1,5) MATRIX, MAXR
   READ (1,6) (R(I), I = 1, MAXR)
   READ (1,6) (THETA(I), I = 1, MAXR)
   MAT1 = MATRIX/2
   DO 10 I = 1, MAT1
   READ (1,7)  NUM, (NORDER(I,J), J = 1, NUM)
   Nums(I) = NUM
10 CONTINUE
   DO 15 I = 1, MAXR
15 THETA(I) = THETA(I)/57.2985
   DO 20 I = 1, MATRIX
20 B(I) = 0.0
   DO 40 I = 2, MATRIX, 2
   L = I - 1
   K = I/2
   NUM = Nums(K)
   DO 35 J = 1, NUM
30 M1 = IABS (NORDER(K,J))
   SIGN = NORDER(K,J)/M1
   B(L) = B(L) + SIGN*R(M1)*SIN (THETA(M1))
   B(I) = B(I) + SIGN*R(M1)*COS (THETA(M1))
35 CONTINUE
40 CONTINUE
   WRITE (3,36) (B(I), I = 1, MATRIX)
   READ (1,45) (MAKEUP(I), I = 1, MAT1)
   DO 105 I = 2, MATRIX, 2
   J = I/2
   M1 = IABS (MAKEUP(J))
   SIGN = MAKEUP(J)/M1
   A = R(M1)*COS (THETA(M1))*SIGN
   C = R(M1)*SIN (THETA(M1))*SIGN
   BOT=-(B(I) - A)
   TOP=-(B(I-1) -C)
   ANGLE = 0.0
   IF (BOT)68,56,52
52 IF (TOP)76,54,76
54 ANGLE = 0.0
   GO TO 72
56 IF (TOP)60,64,58
58 ANGLE = 1.5708
   GO TO 62
60 ANGLE = -1.5708
62 RAD = TOP
   GO TO 78
64 ANGLE = 0.0
   RAD = 0.0
   GO TO 78
68 IF (TOP)74,70,74
70 ANGLE = 3.14159
72 RAD = BOT
   GO TO 78
74 ANGLE = 3.14159
76 ANGLE = ANGLE + ATAN(TOP/BOT)
   IF (ABS(COS(ANGLE))- 0.707)80,80,81
80 RAD = TOP/SIN(ANGLE)
   GO TO 79
81 RAD = BOT/COS(ANGLE)
79 IF (SIGN)82,82,78
82 ANGLE = ANGLE + 3.14159
78 ANGERR = ANGLE - THETA(M1)
95 RERR = RAD - R(M1)
   ANGERR=ANGERR*57.2985
   ANGLE = ANGLE*57.2985
   WRITE (3,100) I, M1, RERR, RAD, ANGERR, ANGLE
105 CONTINUE
   GO TO 1
END

```

## Appendix B. Program MAIN

```

MONSS      EXEQ FORTRAN,,,3,PCH,,MAIN
DIMENSION AY(10,10), BEE(10), X(10), BUFF1(10), R(20), THETA(20),
1  NORDER(5,20), OMEGA(10), ALPHA(10), THETA2(20),SIG (10)
2      , IFRAME(5), THEAPP(20)
COMMON AY, BEE, X, BUFF1, R, THETA, NORDER, OMEGA, ALPHA, THETA2
1      , SIG
5  FORMAT (5I3,2F10.4,10F2.0,17X, 5I1, 3F1.0)
16 FORMAT (20I3)
15 FORMAT (10F8.2)
65 FORMAT (10E8.2)
97 FORMAT (3X,13,2X,2F8.2)
98 FORMAT (10F8.4)
1  READ (1,5) MATRIX, IPTYPE, MXSTEP, NUMCYC, MAXR, DELTEE, DRIVER,
1  SIG ,IFRAME, HYST,      TRAKIN, ALL
READ (1,15) (R(I), I = 1, MAXR)
READ (1,15) (THETA(I), I = 1, MAXR)
MAT1 = MATRIX/2
MAT6 = MATRIX*2
DO 18 I = 1, MAT1
18 READ (1,16) (NORDER(I,J), J = 1, MAT6)
NMSTEP = 0
ZERO = 1.0
IF (IPTYPE - 1)21,21,19
19 DRIVER = DRIVER/57.2958
21 DO 17 I = 1, MAXR
THETA (I) = THETA(I)/57.2958
17 THETA2(I) = THETA(I)
20 ITERAT = 0
30 DO 33 I = 1, MAXR
33 THEAPP(I) = THETA2(I)*57.2985
WRITE (2,15) (THEAPP(I), I = 1, MAXR)
32 CALL ACOE (MATRIX)
CALL BCOE (MATRIX, DRIVER,C1,S1, IPTYPE, 1)
CALL GAUSS (MATRIX,ZERO)
IF (ZERO)38,38,40
38 CALL ACOE (MATRIX)
WRITE (3, 98)((AY(I,J),J= 1, MATRIX),I=1,MATRIX)
GO TO 130
40 ITERAT = ITERAT + 1
44 IF (NMSTEP)50,50,45
45 CALL ANGLE (MATRIX, 2.0, DELTEE)
IF (NUMCYC - ITERAT)50,50,30
50 DO 60 LM = 1, MATRIX
60 OMEGA(LM) = X(LM)
CALL ACOE (MATRIX)
CALL BCOE (MATRIX, DRIVER,C1,S1, IPTYPE, 2)
WRITE (2,65) (BUFF1(I), I = 1, MATRIX)
66 CALL GAUSS (MATRIX,ZERO)
CALL ACOE (MATRIX)
M = -1
DO 68 I = 1, MATRIX
BEE(I) = 0.0
BUFF1(I) = 0.0
M = -M
S = -M
IPM = I + M
DO 68 J = 1, MATRIX
PROD1 = AY(I,J)*X(J)
PROD2 = S*AY(IPM,J)*OMEGA(J)*OMEGA(J)
BEE(I) = BEE(I) + PROD1 + PROD2
68 BUFF1(I) = BUFF1(I) + ABS(PROD1) + ABS(PROD2)
PROD3 = 2.0*DRIVER*OMEGA(1)
PROD4 = R(1)*DRIVER*DRIVER
DO 75 I = 2, MATRIX, 2
IF (IPTYPE - 1)72,72,73
72 PROD1 = PROD3*C1*SIG(I)
PROD2 = PROD3*S1*SIG(I)
GO TO 74
73 PROD1 = -PROD4*S1*SIG(I)
PROD2 = PROD4*C1*SIG(I)
74 BUFF1(I-1) = (BEE(I-1) + PROD1)/(BUFF1(I-1) + ABS(PROD1))*100.0
75 BUFF1(I ) = (BEE(I ) + PROD2)/(BUFF1(I ) + ABS(PROD2))*100.0
WRITE (2,65) (BUFF1(I), I = 1, MATRIX)

```

(continued on next page)



## Appendix B. Program MAIN (continued)

```

      DO 80 I = 1, MAXR
        THETA(I) = THETA2(I)
      80 THETA2(I) = THETA2(I)*57.2958
        NMSTEP = NMSTEP + 1
        IF (TRAKIN)82,82,81
      81 CALL KINE (NMSTEP, ALL)
      82 WRITE (2,97) NMSTEP, R(1), THETA2(1)
        WRITE (2,15) (THETA2(I), I = 1, MAXR)
        DO 95 I = 1, MATRIX
          ALPHA(I) = X(I) * 57.2958
      95 BUFF1(I) = OMEGA(I) * 57.2958
        WRITE (2,15) (BUFF1(I), I = 1, MATRIX)
        WRITE (2,15) (ALPHA(I), I = 1, MATRIX)
        DO 200 I = 1, 5
          MJ = IFRAME(I)
          IF (MJ)200,200,201
      201 THETA2(MJ) = THETA(MJ)
      200 CONTINUE
          IF (NMSTEP - MXSTEP)100,128,128
      100 IF (IPTYPE - 1)110,110,120
      110 R(1) = R(1) + DRIVER*DELTEE
          GO TO 125
      120 THETA(1) = THETA(1) + DRIVER*DELTEE
          THETA2(1) = THETA(1)
      125 CALL ANGLE (MATRIX, 1.0, DELTEE)
          GO TO 20
      128 IF (HYST)130,130,129
      129 DRIVER = -DRIVER
          MXSTEP = 2*MXSTEP
          HYST = 0
          DO 127 I = 1, MAXR
      127 THETA2(I) = THETA(I)
          GO TO 20
      130 GO TO 1
      END

```

## Appendix C. Subroutine ACOE

```

      MONSS      EXEC FORTRAN,,,3,PCH,,ACOE
      SUBROUTINE ACOE (MATRIX)
      DIMENSION AY(10,10), DUM1(20), BUFF1(10), R(20), DUM2(20),
1      NORDER(5,20), DUM3(20), THETA2(20)
      COMMON AY, DUM1, BUFF1, R, DUM2, NORDER, DUM3, THETA2
      MAT2 = MATRIX*2
      MAT1=MATRIX-1
      DO 100 I = 1, MAT1, 2
        K = (I + 1)/2
        DO 100 J = 2, MAT2, 2
          JJ = J/2
          IF (NORDER(K, J-1))20,10,20
      10 AY(I,JJ) = 0.0
          AY(I+1, JJ) = 0.0
          GO TO 99
      20 M1 = IABS (NORDER(K, J-1))
          SIGN1 = NORDER(K, J-1)/M1
          PROD1 = SIGN1*R(M1)
          AY(I,JJ) = PROD1*COS (THETA2(M1))
          AY(I+1, JJ) = PROD1*SIN (THETA2(M1))
          IF (NORDER(K,J))30,99,30
      30 M2 = IABS (NORDER(K,J))
          SIGN2 = NORDER(K,J)/M2
          PROD2 = SIGN2*R(M2)
          AY(I,JJ) = AY(I,JJ)+ PROD2*COS (THETA2(M2))
          AY(I+1, JJ) = AY(I+1, JJ) + PROD2*SIN (THETA2(M2))
      99 TRANS = TRANS
      100 CONTINUE
      RETURN
      END

```

## Appendix D. Subroutine BCOE

```

MON$$      EXEC FORTRAN,,,3,PCH,,BCOE
SUBROUTINE BCOE (MATRIX, DRIVER,C1,S1, IPTYPE, MODE)
  DIMENSION  AY(10,10), BEE(10), OMEGA2(10), BUFFER(10), R(20),
1  DUM1(20), IDUM(100),DUM2(20), THETA2(20), SIGN(10)
2  , TOTPTH(10)
  COMMON  AY, BEE, OMEGA2, BUFFER, R, DUM1, IDUM,DUM2,THETA2,SIGN
  C1 = COS(THETA2(1))
  S1 = SIN(THETA2(1))
  MAT = MATRIX - 1
  DO 10 I = 1, MATRIX
    TOTPTH(I) = 0.0
    BUFFER(I) = 0.0
10 BEE(I) = 0.0
    IF (MODE - 1)20,20,90
20 IF (IPTYPE - 1)30,30,60
30 DO 50 I = 1, MAT, 2
    IF (SIGN(I))40,50,40
40 BEE(I) = -SIGN(I)*DRIVER*S1
    BEE(I+1) = SIGN(I)*DRIVER*C1
50 CONTINUE
    GO TO 200
60 DO 80 I = 1, MAT, 2
    IF (SIGN(I))70,80,70
70 PROD = SIGN(I)*R(1)*DRIVER
    BEE(I) = -PROD*C1
    BEE(I+1) = -PROD*S1
80 CONTINUE
    GO TO 200
90 DO 130 I = 1, MAT, 2
    DO 130 J = 1, MATRIX
      IF (AY(I+1,J))100,110,100
100 PROD1 = AY(I+1,J)*OMEGA2(J)
      BEE(I) = BEE(I) + PROD1*OMEGA2(J)
      BUFFER(I+1) = BUFFER(I+1) + PROD1
      TOTPTH(I+1) = TOTPTH(I+1) + ABS(PROD1)
110 IF (AY(I,J))120,129,120
120 PROD2 = AY(I,J)*OMEGA2(J)
      BEE(I+1) = BEE(I+1) - PROD2*OMEGA2(J)
      BUFFER(I) = BUFFER(I) + PROD2
      TOTPTH(I) = TOTPTH(I) + ABS(PROD2)
129 TRANS = TRANS
130 CONTINUE
    IF (IPTYPE -1)140,140,170
140 DO 160 I = 1, MAT, 2
    IF (SIGN(I))150,160,150
150 PROD1 = SIGN(I)*DRIVER*C1
    PROD2 = SIGN(I)*DRIVER*S1
    BEE(I) = BEE(I) - 2.0*PROD1*OMEGA2(1)
    BEE(I+1) = BEE(I+1) - 2.0*PROD2*OMEGA2(1)
    BUFFER(I) = BUFFER(I) + PROD2
    BUFFER(I+1) = BUFFER(I+1) - PROD1
    TOTPTH(I) = TOTPTH(I) + ABS(PROD2)
    TOTPTH(I+1) = TOTPTH(I+1) + ABS(PROD1)
160 CONTINUE
    DO 165 I = 1, MATRIX
165 BUFFER(I) = BUFFER(I)/TOTPTH(I)*100.0
    GO TO 200
170 DO 190 I = 1, MAT, 2
    IF (SIGN(I))180,190,180
180 PROD2 = SIGN(I)*C1*DRIVER*R(1)
    PROD1 = SIGN(I)*S1*DRIVER*R(1)
    BEE(I) = BEE(I) + PROD1*DRIVER
    BEE(I+1) = BEE(I+1) - PROD2*DRIVER
    BUFFER(I) = BUFFER(I) + PROD2
    BUFFER(I+1) = BUFFER(I+1) + PROD1
    TOTPTH(I) = TOTPTH(I) + ABS(PROD2)
    TOTPTH(I+1) = TOTPTH(I+1) + ABS(PROD1)
190 CONTINUE
    DO 195 I = 1, MATRIX
195 BUFFER(I) = BUFFER(I)/TOTPTH(I)*100.0
200 RETURN
  END

```

## Appendix E. Subroutine GAUSS

```

MONSS      EXEQ FORTRAN,.,.,3,PCH,GAUSS
SUBROUTINE GAUSS (MATRIX,ZERO)
  DIMENSION AY(10,10), BEE(10), X(10), COLSWP(10)
  COMMON  AY, BEE, X, COLSWP
  4  FORMAT (1X, 12E10.4)
104  FORMAT (7H ACOEF(,12,1H,,12, 9H) IS ZERO)
  MM = MATRIX - 1
  DO 200 JAY = 1, MM
    AMAX = ABS (AY(JAY, JAY))
    COLSWP(JAY) = 0.0
    DO 50 KCOL = JAY, MATRIX
      DO 50 KROW = JAY, MATRIX
        IF (AMAX - ABS (AY(KROW, KCOL)))10,49,49
10  AMAX = ABS (AY(KROW, KCOL))
        COLSWP(JAY) = KCOL
        MROW = KROW
49  TRANS = TRANS
        MCL = COLSWP(JAY)
50  CONTINUE
        IF (MCL)90,90,60
60  DO 70 KAY = JAY, MATRIX
        BUF = AY(KAY, JAY)
        AY(KAY, JAY) = AY(KAY,MCL)
        AY(KAY, MCL) = BUF
70  CONTINUE
        DO 80 KAY = JAY, MATRIX
        BUF = AY(JAY, KAY)
        AY(JAY, KAY) = AY(MROW,KAY)
        AY(MROW, KAY) = BUF
80  CONTINUE
        BUF = BEE(JAY)
        BEE(JAY) = BEE(MROW)
        BEE(MROW) = BUF
90  NN = JAY + 1
        IF (AY(JAY,JAY))100,103,100
103  WRITE (3,104) JAY, JAY
        ZERO = 0.0
        GO TO 320
100  DO 110 ICOL = NN, MATRIX
105  AY(JAY,ICOL) = AY(JAY, ICOL)/AY(JAY, JAY)
        DO 110 IROW = NN, MATRIX
110  AY(IROW, ICOL) = AY(IROW, ICOL) - AY(IROW, JAY)*AY(JAY, ICOL)
        BEE(JAY) = BEE(JAY)/AY(JAY, JAY)
        DO 200 IROW = NN, MATRIX
200  BEE(IROW) = BEE(IROW) - AY(IROW, JAY)*BEE(JAY)
        IF (AY(MATRIX,MATRIX))220,210,220
210  WRITE (3,104) MATRIX, MATRIX
        ZERO = 0.0
        GO TO 320
220  X(MATRIX) = BEE(MATRIX)/AY(MATRIX, MATRIX)
        DO 300 II = 1, MM
          N = MATRIX - II
          X(N) = BEE(N)
          M = N + 1
          DO 250 MJJ = M, MATRIX
250  X(N) = X(N) - AY(N, MJJ ) * X(MJJ )
          IF (COLSWP(N))300,300,260
260  MCL = COLSWP(N)
          BUF = X(N)
          X(N) = X(MCL)
          X(MCL) = BUF
300  CONTINUE
        GO TO 325
320  TRANS = TRANS
        DO 315 I = 1, MATRIX
315  WRITE (3,4) (AY(I,J), J = 1, MATRIX), BEE(I), COLSWP(I)
325  RETURN
      END

```

## Appendix F. Subroutine ANGLE

```

MONSS      EXEQ FORTRAN,.,.,3,PCH,.,ANGLE
SUBROUTINE ANGLE (MATRIX, CNTRL, DELTEE)
DIMENSION DUM1(110), X(10), BUFF1(10), DUM2(20), THETA(20),
1          NORDER(5,20), OMEGA(10), DUM3(10), THETA2(20)
COMMON DUM1, X, BUFF1, DUM2, THETA, NORDER, OMEGA, DUM3, THETA2
M1 = MATRIX/2
M2 = MATRIX*2 - 1
10 K = 0
DO 25 J = 1, M2, 2
K = K + 1
IF (CNTRL - 1.0)11,11,12
11 PROD = OMEGA(K)*DELTEE + X(K)*DELTEE*DELTEE/2.0
GO TO 13
12 PROD = (OMEGA(K) + X(K))*DELTEE/2.0
13 DO 25 I = 1, M1
IF (NORDER(I,J))15,24,15
15 INDEX = IABS (NORDER(I,J))
THETA2(INDEX) = PROD + THETA(INDEX)
IF (NORDER(I,J+1))20,24,20
20 INDEX = IABS (NORDER(I,J+1))
THETA2(INDEX) = PROD + THETA(INDEX)
24 TRANS = TRANS
25 CONTINUE
50 RETURN
END

```

## Appendix G. Subroutine KINE

```

MONSS      EXEQ FORTRAN,.,.,3,PCH,.,KINE
SUBROUTINE KINE (NMSTEP, ALL)
DIMENSION DUM1(110), R(20), THETA(20), IDUM2(100), OMEGA(10),
1          ALPHA(10), BUFF(10), ISIGN(100), ICQWNC(100), IOMCOR(100),
2          NMTRMS(24), RCOUP(10), DELTA(10)
COMMON DUM1, ALPHA, BUFF, R, THETA, IDUM2, OMEGA
11 FORMAT (26I3)
13 FORMAT (10F8.2)
33 FORMAT (2I4,6F8.2)
IF (NMSTEP - 1)10,10,20
10 READ (1,11) NMPTHS, (NMTRMS(I), I = 1, NMPTHS), NCOUP
IF (NCOUP)14,14,12
12 READ (1,13) (RCOUP(I), I = 1, NCOUP)
READ (1,13) (DELTA(I), I = 1, NCOUP)
14 IM = 1
IN = NMTRMS(1)
DO 19 I = 1, NMPTHS
READ (1,11) (ISIGN(J), J = IM, IN)
READ (1,11) (ICQWNC(J), J = IM, IN)
READ (1,11) (IOMCOR(J), J = IM, IN)
IF (NMPTHS - I)19,19,17
17 IM = IM + NMTRMS(I)
IN = IN + NMTRMS(I+1)
19 CONTINUE
20 K = 1
L = NMTRMS(1)

```

(continued on next page)

## Appendix G. Subroutine KINE (continued)

```

      DO 40 I = 1, NMPTS
        X = 0.0
        Y = 0.0
        XDOT = 0.0
        YDOT = 0.0
        XDBLDT = 0.0
        YDBLDT = 0.0
        DO 30 J = K,L
          M = ICQWNC(J)
          N = IOMCOR(J)
          SGN = ISGN(J)
          A = SGN*R(M)*COS (THETA(M))
          B = SGN*R(M)*SIN (THETA(M))
          IF (N)21,21,22
21      THEDOT = 0.0
          OMEDOT = 0.0
          GO TO 23
22      THEDOT = OMEGA(N)
          OMEDOT = ALPHA(N)
23      C = -B*THEDOT
          D = A*THEDOT
          E = -D*THEDOT - B*OMEDOT
          F = C*THEDOT + A*OMEDOT
          X = X + A
          Y = Y + B
          XDOT = XDOT + C
          YDOT = YDOT + D
          XDBLDT = XDBLDT + E
          YDBLDT = YDBLDT + F
          IF (ALL)30,30,25
25      IPOINT = ISGN(J)*M
          WRITE (2,33) NMSTEP,IPOINT, X, Y, XDOT, YDOT, XDBLDT, YDBLDT
30      CONTINUE
          IF (ALL)36,36,37
36      IPOINT = ISGN(L)*M
          WRITE (2,33) NMSTEP,IPOINT, X, Y, XDOT, YDOT, XDBLDT, YDBLDT
37      IF (I - NCOUP)31,31,32
31      ANG = THETA(M) + DELTA(I)/57.2958
          A = RCOUP(I)*COS (ANG)
          B = RCOUP(I)*SIN (ANG)
          C = -B*THEDOT
          D = A*THEDOT
          E = -D*THEDOT - B*OMEDOT
          F = C*THEDOT + A*OMEDOT
          X = X + A
          Y = Y + B
          XDOT = XDOT + C
          YDOT = YDOT + D
          XDBLDT = XDBLDT + E
          YDBLDT = YDBLDT + F
          WRITE (2,33) NMSTEP, I, X, Y, XDOT, YDOT, XDBLDT, YDBLDT
32      IF (NMPTS - I)40,40,35
35      K = K + NMTRMS(I)
          L = L + NMTRMS(I+1)
40      CONTINUE
      RETURN
      END

```