An experimental system for the kinematic analysis of two- and three-dimensional mechanical linkages is outlined.

The structure of the programmed system, the input language, and the method of storage allocation are described.

The class of problems treated by the system is discussed in brief, as are the basic vector equations used in obtaining solutions for position, velocity, acceleration, and force of linkage elements.

A computer-aided linkage analysis system

by F. Bitonti, D. W. Cooper, D. N. Frayne, and H. H. Hansen

A significant portion of mechanical engineering effort is spent in the kinematic analysis of mechanisms such as gears, cams, and linkages. Although linkages present a more complex problem of analysis than most other basic mechanisms, they are widely used because of their reliability, speed, and force-transmission properties. Engineers continually seek improvements in existing linkages and devise linkages for new mechanical systems. Linkage analyses have traditionally been performed on the drafting board, but this is difficult and time consuming, and complete analyses are not feasible for the more involved linkage systems encountered in practice.

This paper describes an experimental tool for the analysis of proposed two- or three-dimensional linkages. Called KAM (Kinematic Analysis Method), the tool consists of a programmed system for the IBM 1620. Based on mathematical procedures due to Chace,[1] the system can provide position, motion, and force analyses for a wide class of linkages. The user describes a proposed linkage to KAM in a language modeled after the APT language.[2] This language functions solely as a means of describing the connectivity of parts in a linkage; the action statements that request calculations are specified by other means. From a linkage description in KAM language, the KAM program uses a storage-allocation technique described by Ross[3] to form a tree-organized model of the linkage within computer memory.

The data required by KAM consist primarily of the coordinates of points in the linkage at design position and the magnitudes of input positions, motions, and forces. Position, motion, and force results are displayed in standardized formats. To calculate special parameters of interest, or to exhibit the results in a special way, the user can provide supplementary programs that further process the normal output.

Class of KAM applications

In a linkage, the rigid members function together as an integrated mechanism because member actions are limited by physical interconnections. In a *planar* linkage, action is confined to two dimensions; in a *spatial* linkage, action in three dimensions may occur.

For analytical purposes, a physical connection can be treated as a vector *constraint*, and a fixed-length line between two constraints can be formulated as a vector *link*. A rigid member with two constraints is normally treated as a link. A rigid member with more than two constraints may be treated as a *body* in which three selected points suffice to define a coordinate system. If additional points on the member are of analytical concern, these *points of interest* can be related to the coordinate system of the member body.

In the simplest case, a linkage can be treated as one vector loop. In other cases, a useful analysis requires more than one loop. Although a linkage must have at least one independent loop that requires position data as an input, it may also have dependent loops that are specified by way of incident loops.

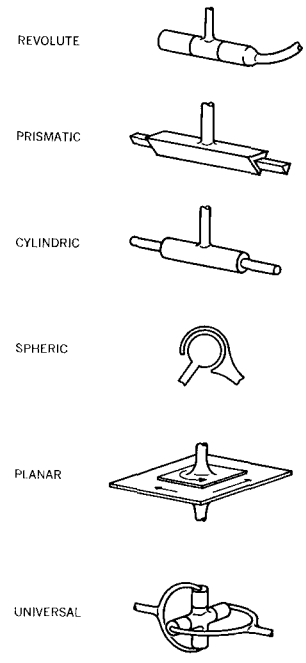
Because it restricts the action of a pair of links, a constraint in a linkage is also called a *pair*. The constraints considered in KAM are of six kinds: revolute, prismatic, cylindric, spheric, planar, and universal. Each of these pairs requires one or more independent variables to specify the relative position of two links connected by the pair. These variables are the degrees of freedom (d.f.) of the pair. The general nature of these pairs is shown in Figure 1.

The Gruebler criterion[4] for a linkage is a function of the number of links, number of pairs, and the d.f. of each pair. The criterion equals zero for a *locked* linkage, one for a *constrained* linkage, and two or more for an *unconstrained* linkage.

KAM is basically designed to treat constrained linkages of three, four, or five links. This permits the analysis of a wide variety of linkages of practical interest. Moreover, by means of a few special rules, certain linkages with two degrees of freedom can be reduced to the category of constrained linkages.

Although computers have previously been applied to linkage analysis, the programs have used techniques of very limited power, and have often limited their scope to position solutions. Denavit and Hartenberg[4,5] have described a computer procedure that yields a position, motion, and static force analysis of virtually

Figure 1 Six constraints



any single-loop linkage from simple input parameters, but their method depends upon iterative processing of a product of several large matrices. By contrast, the method used in KAM gives closed-form position solutions to eight categories of single-loop, two or three-dimensional linkages. The solutions can also be used step-by-step for position determination of multiloop linkages. Most practical linkages fall into either one of the eight categories or into a combination of them. The method reduces motion and force analysis to the solution of simultaneous linear equations.

A useful classification of existing and potential areas for the application of computers to mechanical design has been made by Knappe,[6] who distinguishes between kinematic type synthesis, kinematic size synthesis, kinematic analysis, component design, methematical model generation, and dynamic analysis.

Very little exists in the way of defined logical or mathematical approaches for treating type synthesis. Ordinarily, the major engineering effort is in the redesign of existing mechanisms, and a computer procedure for type synthesis would find limited application—this is true, for example, in the automotive industry. Recently, kinematic size synthesis has received considerable attention. For example, computer procedures have been developed to select gear trains for a needed gear ratio,[7] and the synthesis of cams seems to be adequately handled by polynomial methods.[8] Linkages present the most formidable synthesis problem. For linkages, Freudenstein and Sandor[9] have developed new synthesis methods, and industry has made some use of these methods. However, except for the simplest 4-bar planar mechanisms, mathematical procedures for synthesis rely on linearization of highly non-linear equations and subsequent use of iteration to a solution. No one procedure as yet combines the desirable features of broad application, simplicity of use, and high reliability. Again, the need for size synthesis of linkages in industry is largely restricted to those engineers responsible for original design effort.

A look at the packaging function of automotive engineering demonstrates the value of KAM for geometric problems. Starting with given specifications, such as wheelbase and maximum height and length, and a selection of components such as engine, transmission, and suspensions, the packaging group must "fit" components with a chassis and a body. Fitting is partly a routing and placement problem and partly a kinematic analysis problem. Several of the first components to be placed in the package design are linkages, for example, the driveline, the steering, and the front and rear suspensions. The space required for motion of these components is one of the most difficult factors to deal with in the packaging effort.

At other stages in the packaging effort, and also in design of the individual components, the position, motion, and forces of a host of other linkages must be determined. Throttle and brake linkages, door and hood hinges, window lifts, windshield wipers, and even the human body as a linkage have all been

Table 2 KAM program for FRBAR example

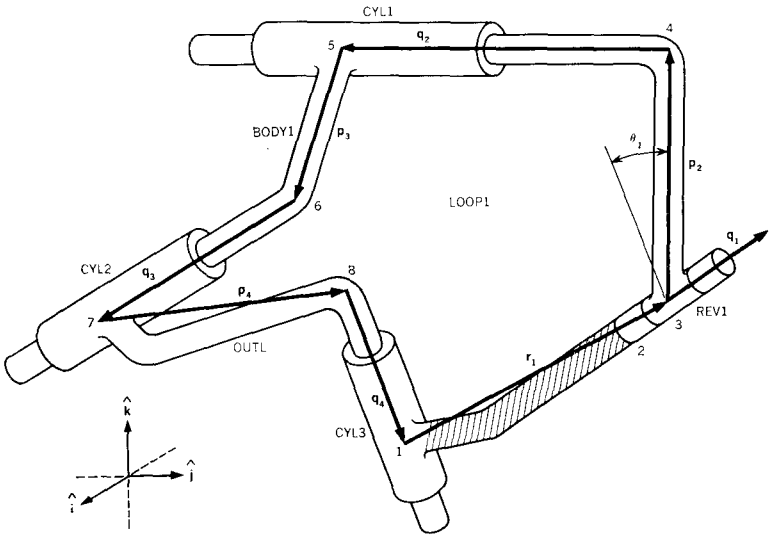
```
REMARK/FRBAR BENT LINK
   DIMENSION/PT (8)
   BASE/PT (2), PT (3) PT(1)
REMARK/BODIES
   BODY1 = BODY/PT (5), PT (4), PT (6)
   DRIVE = BODY/PT (3), PT (2), PT (4)
   OUTL = BODY/PT (7), PT (6), PT (8)
REMARK/CONSTRAINTS
   REV1 = REVLUT/PT (3), PT (2)
   CYL1 = CYLNDR/PT (5), PT (4)
   CYL2 = CYLNDR/PT (7), PT (6)
   CYL3 = CYLNDR/PT (1), PT (8)
REMARK/FORCES AND TORQUES
   T21 = TORQUE/DRIVE (PT (3)), PT (2), PT (3)
   F14 = FLOAD/OUTL (PT (1)), PT (8), PT (1)
   T14 = TLOAD/OUTL (PT (1)), PT (8), PT (1), F14
REMARK/MOTION INPUT
   W21 = MOTION/REV1, PT (2), PT (3)
REMARK/MOTION OUTPUT STATEMENTS FOR FRBAR
   P5INR2 = MOTOUT/BODY1, PT (5)
   P7INC2 = MOTOUT/OUTL, PT (7)
   P1INC3 = MOTOUT/OUTL, PT (8)
REMARK/LOOPS
   LOOP1 = LOOP/REV1, DRIVE, CYL1, BODY1, CYL2, OUTL,CYL3
REMARK/POSITION INPUT
   THETA = INANG/LOOP1 (REV1)
   FRBAR = SYSTEM/LOOP1
   END
```

the plex

Although the rules for using this language will not be detailed here, the use of many of the terms can be seen from an example. The linkage description of Table 2 suffices to describe a single-loop, bent-link mechanism consisting of one revolute and three cylindric constraints; the structure of such a mechanism is suggested by Figure 2. A detailed analysis of this particular mechanism can be found in Reference 10.

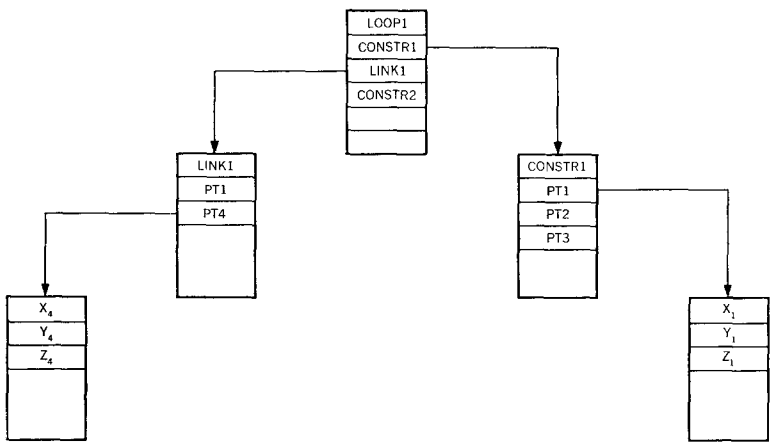
The KAM program organizes statements descriptive of the linkage into the "plex" form for physical models, as suggested by Ross.[3] The plex in KAM consists of a collection of FORTRAN arrays. The columns of the arrays are referred to as beads or lists. Connectivity is achieved by means of pointers contained in topology beads. Metric data beads are necessary for the storage of dimensional data, and have direct correspondence with the topology beads. Topology beads are required for the loops, constraints, links, and bodies; essentially, they provide an unscaled representation of the linkage. This facilitates checking the linkage against kinematic rules, selecting mathematical procedures and accessing data related to any portion of the linkage. Because the topology and metric data are separated, many variations in dimensions and linkage inputs can be analyzed without changes in linkage description.

Figure 2 Three-dimensional four-bar linkage with turn slide pairs



The KAM system allows up to five kinematic loops in a linkage description. A LOOP statement defines, not the topology of the several loops in the system, but the topology of one loop only. This is desirable for KAM because position solution is closely tied to individual loops. A tree-organized plex is useful here in that it enables connectivity to flow between constraints and links within a loop, as well as from the constraints and links to related point data. This organization is shown schematically in Figure 3 for a single loop. On the other hand, the force and motion solutions are system oriented rather than loop oriented. If several loops are named, the list of constraint and link pointers created for each loop is sufficient to represent individual loop topology but not to capture overall linkage topology. To overcome this limitation, the tree organization was generalized by means of "back pointers."

Figure 3 Tree-structured loop



canonical forms

Each of the six allowable KAM constraints can be defined as a combination of the two elemental geometric conditions, viz., fixed length, and fixed angle. These two conditions could have been used instead of the six pair types. But the six constraints are justified in that they are commonly used and understood by engineers, and permit greater conciseness in constraint definition as well. However, the engineer's KAM description of pairs must be reduced by processing combinations of the two basic conditions. Because knowledge of the pair types, their connectivity, and the static position of pair axes in a loop is sufficient for mathematical representation of a loop, the reduction is accomplished in KAM with the aid of six "canonical forms." The canonical forms define the simplest set of input data for each pair type and provide an unambiguous procedure for reduction of pair data to the two basic geometric conditions. As provided, the canonical forms also eliminate the need for consideration of certain details of the physical arrangement. Specifically, bends and twists in the links can be neglected, and attention need not be given to which of two adjacent links is rigidly attached to a connecting pair. An illustration of canonical form data is found in Reference 10.

table of algorithms

Since KAM employs the several mathematical procedures of Chace[1] for position solution, one of the functions of the system must be that of solution procedure selection. Unlike the force and motion phases, in which the solution procedure consists of the reduction of linear equations, position solution may be one of eight non-linear procedures depending upon the order and type of constraints in the kinematic loop. One approach to this problem is to logically determine, by programming, the resultant unknowns for any configuration and to make a selection of solution procedure. Considering the six constraints and a maximum of five links in a loop, the possible linkages are in the hundreds. Practical considerations, based on degrees of freedom, reduce this number greatly. Further investigation led to the conclusion that an enumeration of the linkages and their associated canonical form data and vector unknowns was feasible. This table of KAM algorithms lists all of the constraint configurations (linkages) that can be solved by Chace's position solution procedures.

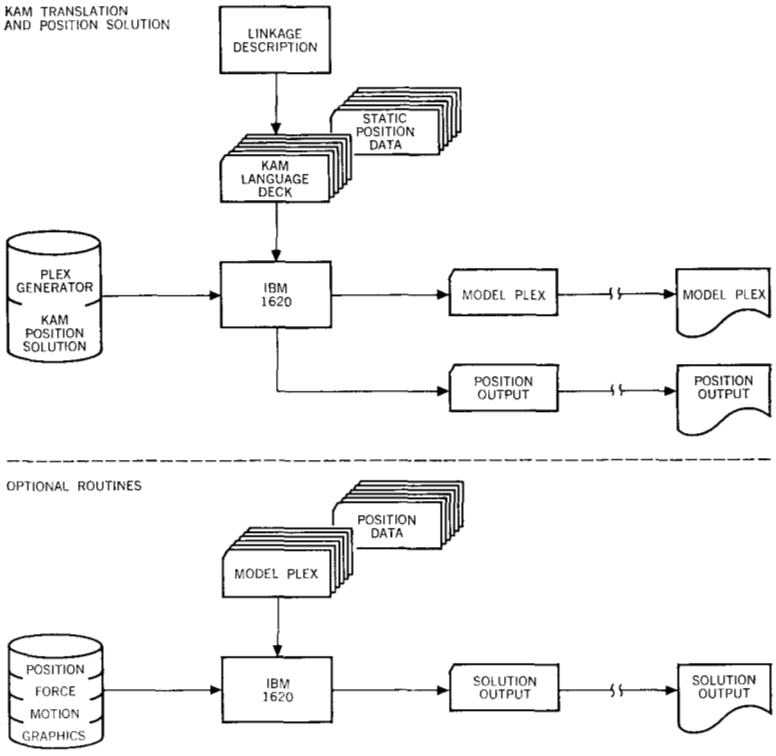
KAM processing

Following the generation of a plex, the KAM processing phases consist of position, force, and motion calculations. The latter two, being dependent upon position data, are optional and are performed only if requested by the user. The overall operation of KAM is indicated in Figure 4.

In addition to the linkage description, KAM input includes (1) coordinates of each linkage point at design position, (2) magnitudes of incremental changes to be made in angles or positions, (3) applicable ranges for angles or distances, and (4) magnitudes of the externally applied forces and moments.

KAM output consists principally of numerical data for position,

Figure 4 Overall KAM logic



force, torque, velocity, and acceleration. All output is in standard format to facilitate further processing if desired. Normally, the results of an engineering analysis are most useful in graphic form. Typical needs of a linkage designer might be a plot of input angles or lengths versus positions of some key point, or a plot of input angles versus the projected angles of a link. Such graphs are not provided by KAM, but can easily be drawn from the KAM output. However, KAM can provide a line sketch of the entire system, as well as perspective views of such a sketch.

The set of seven programs mentioned in Table 3 are included in KAM to facilitate the programming of FORTRAN output programs. Two of these programs assist in the display of results on the 1627 Plotter. The other five calculate geometric parameters based on the coordinates of points in the linkage system.

KAM is an open-ended system, as illustrated by the ability to integrate FORTRAN output programs. The system design recognizes that the KAM language is purely descriptive, and that the linkage descriptions, and the position, motion, and force analyses, are subject to complete standardization. The lower half of Figure 4 illustrates the optional calculations which the user can request for the particular design to be evaluated. Because this flexibility is most needed in the area of output, once KAM has calculated positions, motions, and forces for the described linkage, the display of results is left to programs provided by

Table 3 Output subroutines

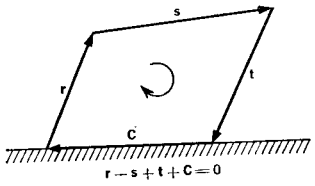
SKETCH	Draws point and line sketch of either of the three plan views of the linkage.
PERVEW	Draws a true perspective point and line sketch of the linkage.
LOCUS	Calculates the three coordinates of points on a locus of revolution.
ANGLE	Calculates the angle between two lines, or a line and a reference axis.
XLGTH	Calculates the length of a line from a point to a plane, or the length of a line in space.
PANGLE	Calculates the angle between a line and one of the reference planes.
DIRECN	Calculates the polar and azimuthal angles of a vector.

the user. The user's output program may make full use of the algorithmic capabilities of the FORTRAN system, as well as those of the special KAM subroutines. Programs that perform additional calculations on the results of the standard position, motion, and force solutions can be included.

system adaptation

The KAM system was written in FORTRAN in order to keep the system as machine independent as possible. Moreover, the use of FORTRAN ensured a system that could be modified and expanded with ease. Each major function of the system is written as a self-contained program. KAM could be readily adapted to any disk system, assuming that a disk-oriented FORTRAN processor were available. Conversion to a combination disk/tape system would undoubtedly require procedural changes for the input and output of numeric data, whereas conversion to a solely tape-oriented system would require a more extensive system redesign to facilitate program segmenting and data handling.

Figure 5 Typical vector representation of simple 4-bar mechanism



KAM was written for the IBM 1620, with the aid of the FORTRAN II-D system. The use of an on-line 1311 Disk File enables the KAM program to be segmented into a series of main subprograms, all of which are stored in the disk file. Some of the needed subroutines are stored in core memory at all times; others are loaded from the disk only on call. Sketching is accomplished with the IBM 1627 Plotter.

Analytical foundations

position solution

The previously cited work by Chace forms a mathematical nucleus for position solution in KAM. Each link is treated as a vector, with the ground being a link that closes the loop as shown in Figure 5. This is represented by the three-dimensional vector equation,

$$\mathbf{r} + \mathbf{s} + \mathbf{t} + \mathbf{C} = 0, \tag{1}$$

where **r**, **s**, and **t** are potentially unknown and **C** is the sum of all known vectors. Chace derived a set of nine closed-form solutions (Tetrahedron Solutions) to this Tetrahedron Equation. These

Table 4 Potentially unknown vector components

Vector	r	s	t
Polar Angle	ϕ_r	ϕ_s	ϕ_t
Azimuthal Angle	Θ_r	Θ_s	Θ_t
Length	r	s	t

solutions represent all possible combinations of unknown spherical coordinates among the vectors. Table 4 summarizes the spherical coordinates, any three of which may be unknown. Table 5 summarizes the nine solutions (called cases) and the known and unknown quantities. To illustrate, in Case 3, two vectors are dependent upon unknown coordinates; the length and azimuthal angle of the vector **r** and the azimuthal angle of the vector **s** are unknowns. To solve the case for these unknowns, required quantities are the sum of all known vectors in the loop; the polar unit vectors about which **r** and **s** rotate; the polar angles between $\hat{\omega}_r$ and **r** and between $\hat{\omega}_s$ and **s**; and the length of **s**.

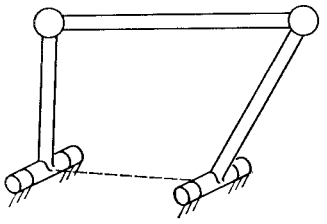
linkage types

KAM is designed to accept problems which require Cases 1 through 8. Because an application of Case 9 is not readily found in industry, it was omitted from KAM.

Just as loop solution types are classified, so are problem types. They are

I Single loop,
  a) Simple 4-bar loop
  b) Bent-link loop
II Multiloop series,
  a) Single input
  b) Two or more inputs
III Multiloop parallel
  a) Single system of parallel loops
  b) Other combinations of above

Figure 6 Simple 4-bar loop



The simple 4-bar loop of Figure 6 is a typical example that can be solved quite readily by graphic or geometric means. If the two revolutes are parallel, the mechanism is planar, i.e., two-dimensional in action. The solutions for planar linkages are special cases of the spatial (three-dimensional) cases of Table 5 and can be treated without enlarging the number of case solutions.

A more complex 4-bar linkage has bent links, as shown in Figure 2. As an example of a typical loop solution, this linkage will be examined in more detail. With the input of θ_1 at the revolute constraint, all unit vectors can be calculated. This procedure is outlined in Table 6. Also, since the link lengths remain unchanged, the link vectors $\mathbf{r}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$ are completely known. The only remaining unknowns are the lengths of the constraint vectors $\mathbf{q}_2$, $\mathbf{q}_3$, and $\mathbf{q}_4$. From Table 5, Case 6 is known to apply, and the solution can be completed easily.

Type II problems consist of a connected series of Type I problems. The single-input Type II category is exemplified by throttle linkages, clutch linkages, and the like. For multiple inputs, a typical example is the front suspension of an automobile, for which two independent suspension and steering loops must be solved prior to a dependent tie-rod loop. Following this, other data can be found, e.g., steer effect, camber, caster, jounce and

Table 5 Categorization of solutions to Equation 1

Case Number	*Unknown*	*Known*			*Degree of Polynomial Solution*
		Vectors	*Unit Vectors*	*Scalars*	
1	r, θ_r, ϕ_r	C			1 (trivial)
2	$r, \theta_r; s$	C	$\hat{s}, \hat{\omega}_r$	ϕ_r	2
3	$r, \theta_r; \theta_s$	C	$\hat{\omega}_r, \hat{\omega}_s$	$\phi_r; s, \phi_s$	4
4	$\theta_r, \phi_r; s$	C	$\hat{s}$	r	2
5	$\theta_r, \phi_r; \theta_s$	C	$\hat{\omega}_s$	$r; s, \phi_s$	2
6	$r; s; t$	C	$\hat{r}, \hat{s}, \hat{t}$		1
7	$r; s; \theta_t$	C	$\hat{r}, \hat{s}, \hat{\omega}_t$	t, ϕ_t	2
8	$r; \theta_s; \theta_t$	C	$\hat{r}, \hat{\omega}_s, \hat{\omega}_t$	$s, \phi_s; t, \phi_t$	4
9	$\theta_r; \theta_s; \theta_t$	C	$\hat{\omega}_r, \hat{\omega}_s, \hat{\omega}_t$	$r, \phi_r; s, \phi_s; t, \phi_t$	8

Notes

Equation 1 is commutative (the solution for unknown $r; \theta_s; \theta_t$ is the same as for unknown $s; \theta_r; \theta_t$, etc.)

If a single vector depends upon a known azimuthal angle and an unknown polar angle, it can be restated as dependent upon an unknown azimuthal angle and a known polar angle.

Table 6 Typical solution

KNOWN:	
$\hat{q}_1$	base constraint unit vector (revolute)
$\hat{q}_4$	base constraint unit vector (cylindric)
$\hat{q}_1 \cdot \hat{q}_2$	dot products of q_1 and q_2
$\hat{q}_3 \cdot \hat{q}_4$	other dot products
$\hat{q}_1 \cdot \hat{p}_2$	
$\hat{q}_2 \cdot \hat{p}_2$	
$\hat{q}_2 \cdot \hat{q}_3$	
$\hat{q}_2 \cdot \hat{p}_3$	
$\hat{p}_3 \cdot \hat{q}_3$	
p_2, p_3, p_4, r_1	length of all links
INPUT:	
θ_1	angular input to base constraint (revolute $\hat{q}_1$)
IMMEDIATELY FIND:	
$\hat{q}_2$	with simultaneous solution of $(\hat{p}_2 \cdot \hat{q}_2)$ and $(\hat{q}_1 \cdot \hat{q}_2)$
$\hat{q}_3$	with simultaneous solution of $(\hat{q}_4 \cdot \hat{q}_3)$ and $(\hat{q}_2 \cdot \hat{q}_3)$
$\hat{p}_3$	with simultaneous solution of $(\hat{p}_3 \cdot \hat{q}_3)$ and $(\hat{p}_3 \cdot \hat{q}_2)$
$\hat{p}_1$	with simultaneous solution of $(\hat{p}_4 \cdot \hat{q}_3)$ and $(\hat{p}_4 \cdot \hat{q}_4)$

rebound envelope. Loops are solved one at a time; independent loops are solved first and followed by the solution of dependent loops.

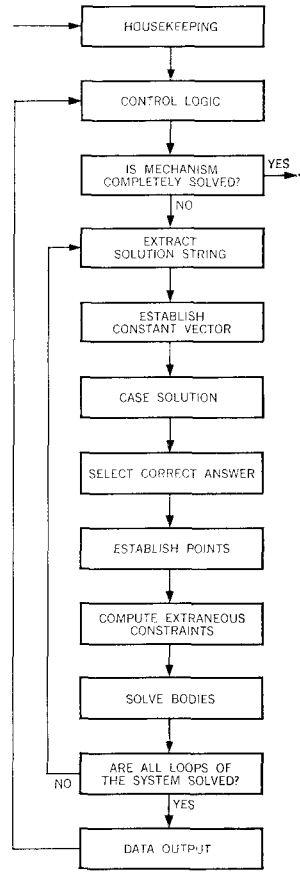
More complicated than Types I and II are the mechanisms of Type III. Type IIIb was considered but not included in the design of KAM. A typical example of Type IIIa, the automotive rear suspension, has five vector loops that must be solved simultaneously. It is interesting to note that KAM can perform a force analysis of such a mechanism, provided that position data is available. Reference 10 further illustrates Type II and Type III mechanisms.

The first phase of position solution selects the method and determines the sequence of mathematical solution. To accomplish this, the program draws upon the fundamental systems concepts, namely, the plex organization of topology and the table of KAM algorithms. The system topology is examined loop by loop for a match with an entry in the table of KAM algorithms. Following a match, the Tetrahedron Solution required for position solution is extracted from the table and placed in the loop bead of the plex. Additional data, such as the required simultaneous dot products and vector unknowns, are extracted from the table and placed in the plex. All bodies named in the program are examined and appropriate data are entered in the plex; one by one, each body becomes fixed in space as the result of solving one or more loops. In addition to the originally given topology, the plex at this point contains data that indicate a solution method and sequence for the second phase of the position solution.

Figure 7 shows a general outline of the second phase of position solution. The first steps in the housekeeping function are to read the plex, and then the data, into memory. The point data that are read into the position solution section of KAM may constitute any set of points in the range of the given mechanism. However, the KAM system treats this set as the design or static condition of the mechanism, and then, by scanning the linkage data, relates all parameters and variables describing the mechanism to the set. In housekeeping, each loop is scanned in sequence of solution and all vectors for links and constraints are computed. After these vectors are found and stored in the metric arrays of the plex, a second scan is undertaken and angular data pertinent to the description of loops are computed from the results of the first pass and the connectivity indicated by the plex. In a third scan, points of interest on defined bodies are assigned polar coordinates relative to a reference frame defined by the three points that fix the body in space.

One housekeeping scan finds universal joints in the mechanism. These constraints must be treated by a simultaneous solution of two dot products, and each solution has two answers. Therefore, a solution mode based on static position is established for later use in selecting correct answers as the mechanism is moved through its range.

Figure 7 General flow chart for position solution, phase 2



Some links may be bent, in which case a simultaneous solution of dot products establishes the axes as constant vectors. By the same operation, the axes of certain constraints can be found and treated either as constant vectors or as known unit vectors of slide-type constraints. Since two answers are possible for each operation, the solution mode is established from static data, as for the universal constraint.

All but two of Chace's Tetrahedron Solutions have more than one answer: two have four, one has eight, and the rest have two answers. However, the actual static position of a mechanism is mathematically unique, and defines criteria from which the single correct answer can be selected for subsequent positions of the mechanism. These criteria are determined and stored in the plex. Because the solution criteria may change for different sets of point data, they must be calculated for each set of input point data. The same reasoning holds for the solution criteria established for the simultaneous solution of dot products.

Length incrementation in KAM simply requires a change in the scalar magnitude of a vector, the unit vector remaining unchanged. However, for angular input, the vector changes orientation and must be re-established as a constant vector for the case solution. A reference frame, established from the revolute and adjacent input link for each loop that requires angular input, is saved in the plex. In KAM, a given loop can have either angular input or linear input, but not both.

The control logic performs the looping functions of a FORTRAN "DO" statement and establishes increments for subsequent use in solution and output functions of KAM. Data required for these purposes are stored in the plex. After a loop has been selected for solution by the control logic, the solution string found by phase 1 is extracted from the plex.

Links and constraints not classified as unknown by the solution string are treated as known vectors and summed into one constant vector as required.

Once the required data have been established, a branch is taken to one of eight case solutions. If an error occurs in the case solution indicating that the mechanism is going through an imaginary range, an error message yields pertinent data, the input variable is incremented, and calculations are resumed. This process continues until either a real range is again entered or the complete range of the mechanism is satisfied.

For case solutions with more than one answer, a test is made on each answer, using the same function that determined the solution mode at housekeeping time. If the answer is inappropriate, then the case-solution error routine is executed.

After the correct answer is selected, the point data in the loop are found by adding each vector, in sequence, to a given fixed or ground point. These new points become fixed points for dependent loops or bodies subsequently to be solved. Some loops have constraints that are not required in solving the loop,

but are required to fix bodies in space. These constraints, as well as those bodies which have three fixed points, were found by phase one and their solution points placed in the plex. Phase two solves them at this time.

After the body solutions, a test determines whether all loops have been solved. If no, control passes back to the block that extracts the solution string. If yes, an output routine is entered. Subsequently, the control logic tests whether the range is satisfied. If not, the linkage is solved for a new value of the input variable, and so on until the range is satisfied.

motion solution

The velocities and accelerations of linkage elements yield important knowledge about the performance characteristics of the linkage. They may also be used in computing other element properties, such as inertia and stress. The mathematical and programming procedures needed to obtain these motions are discussed here. The mathematical procedure, again based on Chace's work, permits motion analysis of all linkages within the scope of KAM. In contrast to approximate solutions, such as seen in "finite differencing" techniques, the mathematical methods to be discussed reduce the problem to a set of simultaneous linear equations that give an exact solution at any linkage position. A principle used throughout is that each order of motion is dependent only on motion quantities having the same or lower order of motion. For example, velocity (first order motion) is dependent on input velocity and on position (zeroth order motion).

motion equations

Two fundamental conditions hold for the motion of a linkage. These, a linear condition and an angular condition, are written as two vector equations for each loop in the mechanism. In final form, the equations yield the same coefficient matrix for both velocity and acceleration, although the constant vector differs for each. Thus the coefficient matrix, once found, can be used to obtain both orders of motion.

velocity equations

Represent each link by a position vector $\mathbf{r}_i$; the sum of such vectors is zero for each loop in the mechanism. For a summary of notation, see Table 7. The velocity vector is the time-derivative of r_i. Thus,

$$\mathbf{V} = D\mathbf{r} = (Dr)\hat{\mathbf{r}} + \boldsymbol{\omega} \times \mathbf{r}$$

$$\mathbf{V}_i = (Dr_i)\hat{\mathbf{r}}_i + \left[\sum_{i=2}^{j} \boldsymbol{\omega}_{i,i-1}\right] \times \mathbf{r}_i.$$

Since all unit vectors, $\hat{\boldsymbol{\omega}}_{i,i-1}$, are known from the position solution, all absolute rotational velocity vectors, $\boldsymbol{\omega}_{i1}$ are expressed as a sum of relative rotational velocities, $\boldsymbol{\omega}_{i,i-1}$. Thus for example, $\boldsymbol{\omega}_{31} = \boldsymbol{\omega}_{21} + \boldsymbol{\omega}_{32}$. The fundamental linear condition for velocity, for a vector loop of n links, can be written

$$\sum_{i=1}^{n} (Dr_i)\hat{\mathbf{r}}_i + \sum_{j=2}^{n} \left(\sum_{i=2}^{j} \boldsymbol{\omega}_{i,i-1}\right) \times \mathbf{r}_i = 0. \qquad (2)$$

This equation contains terms of translational velocity due to

Table 7 Summary of notation

u	Magnitude: a scalar, non-directional quantity
$\hat{\mathbf{u}}$	Unit vector: a vector of unit magnitude
$\mathbf{u}$	Product of a magnitude u and a unit vector $\hat{\mathbf{u}}$. Thus $\mathbf{u} = u\hat{\mathbf{u}}$
$\mathbf{a} \cdot \mathbf{b}$	Dot product of $\mathbf{a}$ and $\mathbf{b}$. If β is the smaller angle between the two vectors, $\mathbf{a} \cdot \mathbf{b} = ab\ (\cos \beta)$
$\mathbf{r}$, $\mathbf{s}$, $\mathbf{t}$, $\mathbf{c}$	Postion vectors relative to the nth reference frame
$\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$	Ground reference frame: three mutually perpendicular unit vectors that obey the right-hand rule and have a fixed orientation relative to ground
θ, ϕ	Angles specifying the direction of a unit vector with respect to a ground reference frame
$\mathbf{p}$, $\mathbf{q}$	Dummy variable vectors
$\hat{\lambda}$, $\hat{\mu}$, $\hat{\nu}$	A reference frame having instantaneous position defined relative to $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$
D	Shorthand for the derivative d/dt
$\hat{\boldsymbol{\omega}}_{i,i-1}$	Unit vector of rotation of link i with respect to link $i-1$
$\boldsymbol{\omega}_{i1}$	Absolute angular velocity of link i with respect to ground
$\boldsymbol{\omega}_I$	An input velocity, also written $\boldsymbol{\omega}_{21}$
Dr_I	An input scalar, also written Dr_2
$\boldsymbol{\alpha}$, $\boldsymbol{\alpha}_i$, $\boldsymbol{\alpha}_{ij}$	Angular acceleration; subscripts as for $\boldsymbol{\omega}$; for spatial motion this $\boldsymbol{\alpha}$ represents only partial angular acceleration[1]
$\mathbf{f}_{i,j}$	Force exerted on body or link i by body or link j with components $f^i_{i,j}$, $f^j_{i,j}$, $f^k_{i,j}$, $f^\lambda_{i,j}$, $f^\mu_{i,j}$, $f^\nu_{i,j}$ in the direction indicated by superscripts
$\boldsymbol{\tau}_{i,j}$	Torque exerted on body or link i by body or link j with components $\tau^i_{i,j}$, $\tau^j_{i,j}$, $\tau^k_{i,j}$, $\tau^\lambda_{i,j}$, $\tau^\mu_{i,j}$, $\tau^\nu_{i,j}$ in the direction indicated by superscripts
$\mathbf{F}_o$	Force output
$\mathbf{F}_I$	Force input
$\boldsymbol{\tau}_o$	Torque output
$\boldsymbol{\tau}_I$	Torque input
$\mathbf{r}_{i,j}$	Vector from center point of constraint i to center point of constraint j
$\mathbf{r}_o$ and $\mathbf{r}_I$	Vector from center point of constraint i to output and input forces

rotation ($\boldsymbol{\omega} \times \mathbf{r}$) as well as terms of pure translational velocity ($(Dr)\hat{\mathbf{r}}$). Many of the terms may be either zero or known inputs; the existence of terms is dependent on constraint types in the linkage and on the location of fixed links in the loop.

A second condition expresses the fundamental angular condition for motion.

$$\sum_{i=2}^{n} (\omega_{i,i-1})\hat{\boldsymbol{\omega}}_{i,i-1} + (\omega_{1,n})\hat{\boldsymbol{\omega}}_{1,n} = 0. \tag{3}$$

Equation 3 states that the sum of the relative angular velocities of a linkage loop is zero.

For solution purposes, (2) and (3) are rewritten as Equations 4 and 5:

$$\sum_{i=3}^{n} (Dr_i)\hat{\mathbf{r}}_i + \sum_{i=3}^{n} \left[\omega_{i,i-1}\hat{\boldsymbol{\omega}}_{i,i-1} \times \sum_{i=j}^{n} \mathbf{r}_i \right]$$
$$= -(Dr_2)\hat{\mathbf{r}}_2 - \boldsymbol{\omega}_{21} \times \sum_{j=2}^{n} \mathbf{r}_i. \tag{4}$$

In (4), Dr_2 or ω_{21} represent possible scalar inputs; in any independent loop, one of these will be zero because the linkage has only a single freedom. In a dependent loop, Dr_2 and ω_{21} may be unknowns and would appear on the left side of the equation.

$$\sum_{i=3}^{n} \omega_{i,i-1}\hat{\boldsymbol{\omega}}_{i,i-1} + \omega_{1,n}\hat{\boldsymbol{\omega}}_{1,n} = -\boldsymbol{\omega}_{21} \tag{5}$$

Again, ω_{21} represents the scalar input and would be zero for pure translational input.

acceleration equations

The acceleration equations are obtained by taking derivatives of (2) and (3). For each link vector $\mathbf{r}_i$, the acceleration is written:

$$\mathbf{a}_i = (D^2 r_i)\hat{\mathbf{r}}_i + \left(\sum_{i=2}^{j} \boldsymbol{\alpha}_{i,i-1} \times \mathbf{r}_i\right) + \mathbf{C}_i,$$

where

$$\mathbf{C}_i = 2Dr_i(\boldsymbol{\omega}_{j1} \times \hat{\mathbf{r}}_i) + \boldsymbol{\omega}_{j1} \times (\boldsymbol{\omega}_{j1} \times \mathbf{r}_i) + \sum_{i=3}^{j} (\boldsymbol{\omega}_{i-1,1} \times \boldsymbol{\omega}_{i,i-1}) \times \mathbf{r}_i$$

and

$$\sum_{i=2}^{j} \boldsymbol{\alpha}_{i,i-1} \times \mathbf{r}_i = \boldsymbol{\alpha}_{j1} \times \mathbf{r}_j.$$

Thus, for a vector loop of n links:

$$\sum_{i=2}^{n} (D^2 r_i)\hat{\mathbf{r}}_i + \sum_{i=2}^{n} \left(\sum_{i=2}^{j} \boldsymbol{\alpha}_{i,i-1}\right) \times \mathbf{r}_i = -\sum_{i=2}^{n} \mathbf{C}_i. \tag{6}$$

Note the similarity in (2) and (6). The first and second terms of (6) express the normal and tangential accelerations, respectively. All r_i, Dr_i, and $\boldsymbol{\omega}_{i1}$ are known from the position and velocity solutions.

The differentiation of (3) establishes a second equation for acceleration.

$$\sum_{i=2}^{n} \boldsymbol{\alpha}_{i,i-1} + \boldsymbol{\alpha}_{1,n} = -\mathbf{C}, \tag{7}$$

where

$$\mathbf{C} = \sum_{j=2}^{n-1} (\boldsymbol{\omega}_j \times \boldsymbol{\omega}_{j+1,j})$$

and

$$\hat{\boldsymbol{\alpha}}_{i,i-1} = \hat{\boldsymbol{\omega}}_{i,i-1}; \qquad \boldsymbol{\alpha}_{i,i-1} = D\boldsymbol{\omega}_{i,i-1}.$$

Again, note the similarity between (7) and the angular velocity equation (3).

The number of scalar equations for acceleration is identical to the number of scalar equations for velocity. Since the coefficients of the velocity and acceleration equations are identical, only the vector $\mathbf{C}$ need be found to obtain the linear equations that define the accelerations.

motion of an arbitrary point

The equations discussed above are used to obtain the relative motions of link i with respect to link $i-1$ for all links in the mechanism. To obtain the absolute motion of link i with respect to ground, the relative motions are summed. Once these are established, the motion of points of interest can be computed by summing all motions from ground to the point.

Equations 2 and 3 are written for each loop in the mechanism. For an n-loop spatial linkage, $6n$ scalar equations are obtained by taking scalar products of the original $2n$ vector equations and the $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$ unit vectors. Thus, $6n$ unknown motions can be computed for spatial linkages. For planar linkages, a computed auxiliary reference frame ($\hat{\boldsymbol{\lambda}}$, $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$) allows the plane of action to be oriented arbitrarily in space, and the $2n$ vector equations give $3n$ scalar equations. Scalar products between (2) and $\hat{\boldsymbol{\lambda}}$ and $\hat{\mathbf{u}}$ yield $3n$ equations, while the scalar product of (3) and $\hat{\mathbf{v}}$ yields n equations.

Unknown motions can be classified into two categories.

scalar unknowns

- Unknown translational motions: Dr_i for velocity and D^2r_i for acceleration, $(i = 2, \cdots n)$.
- Unknown relative angular motions: $\omega_{i,i-1}$ for velocity and $D\omega_{i,i-1}$ for acceleration, $(i = 2, \cdots n)$.

For a single independent spatial loop, any combination of the scalars $D^m r_i$ (for $m = 1, 2$) and $D^m \omega_{i,i-1}$ (for $m = 0, 1$), up to a maximum of six, may be unknowns. However, a maximum of three $D^m r_i$ and six $D^m \omega_{i,i-1}$ scalars can be present in this case.

A spatial loop with more than six unknowns can be solved using additional "constraints" on the loop. In a multiloop system, the added constraints on a given loop take the form of another loop that shares a portion of the given loop. The number of equations, hence the number of unknowns, is limited by the total number of loops. Thus, linkages comprised of a series of loops can have more than one degree of freedom and more unknowns. Again, if its motion is determinate, a single loop linkage can have only *one* degree of freedom. Spin degrees of freedom, created when two ball joints connect a link, create extraneous degrees of freedom that are eliminated in KAM by replacing one of the ball joints by a universal joint.

For both planar and spatial mechanisms, the number of equations for linear and angular conditions is shown in Table 8. Also illustrating the general matrix form, the matrix is divided into two sections, the top being used for the linear conditions and the bottom for the angular conditions.

Each constraint in a vector loop describing a linkage introduces one, two, or three unknowns, as shown in Table 9. The directions of the unit vectors associated with these unknown magnitudes are known from the position solution. In conjunction with the link vectors, the components of the unit vectors representing direction become equation coefficients. For example, a revolute constraint introduces one unknown and this unknown may appear

in six equations. Thus, a revolute constraint for one spatial loop introduces six coefficients into the matrix.

the motion program

For KAM, the input for both position and motion must be applied at the same constraint. In addition, the input constraint for motion must be a ground constraint with one degree of freedom, i.e., a revolute or a prismatic. The linear motion of points of interest is represented by a vector to the point from the ground. All other outputs represent the relative and absolute motions of link i with link $i-1$ and link i with respect to ground, respectively.

The KAM motion processor executes the mathematical computations outlined above, solves the resulting set of linear equations, and outputs the velocities and accelerations of the linkage elements. This processor consists of the phases for input and initialization, for a scan, for a determination of velocities, and for a determination of unknown accelerations.

Table 8 The general form of the coefficients matrix

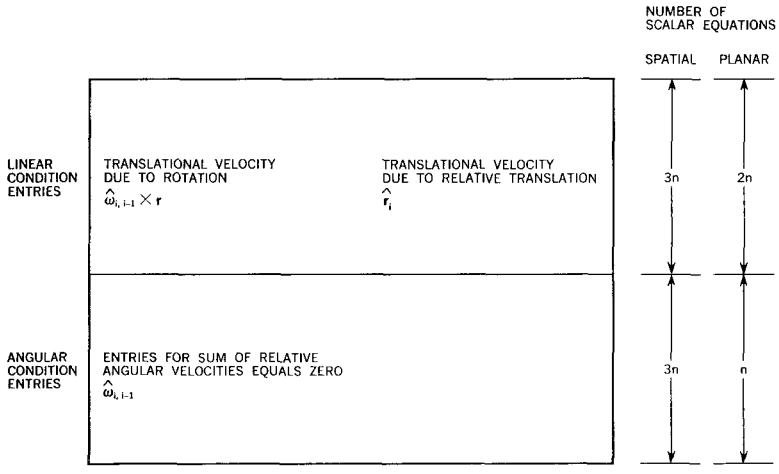


Table 9 Reference data for the constraint types

Type	*Degrees of Freedom*	*Velocity component of motion*	*Acceleration component of motion*	*Form of force torque expressions*
revolute	1	$\omega_{i,i-1}$	$D\omega_{i,i-1}$	$\mathbf{f} = f^i\,\hat{\mathbf{i}} + f^j\,\hat{\mathbf{j}} + f^k\,\hat{\mathbf{k}}$ $\tau = \tau^\lambda\,\hat{\boldsymbol{\lambda}} + \tau^\mu\,\hat{\boldsymbol{\mu}}$
prismatic	1	Dr_i	D^2r_i	$\mathbf{f} = f^\lambda\,\hat{\boldsymbol{\lambda}} + f^\mu\,\hat{\boldsymbol{\mu}}$ $\tau = f^i\,\hat{\mathbf{i}} + f^j\,\hat{\mathbf{j}} + f^k\,\hat{\mathbf{k}}$
cylindric	2	$\omega_{i,i-1}$, Dr_i	$D\omega_{i,i-1}$, D^2r_i	$\mathbf{f} = f^\lambda\,\hat{\boldsymbol{\lambda}} + f^\mu\,\hat{\boldsymbol{\mu}}$ $\tau = \tau^\lambda\,\hat{\boldsymbol{\lambda}} + \tau^\mu\,\hat{\boldsymbol{\mu}}$
spheric	3	$\omega_{i,i-1}$, $\omega_{i+1,i}$, $\omega_{i+2,i+1}$	$D\omega_{i,i-1}$, $D\omega_{i+1,i}$, $D\omega_{i+2,i+1}$	$\mathbf{f} = f^i\,\hat{\mathbf{i}} + f^j\,\hat{\mathbf{j}} + f^k\,\hat{\mathbf{k}}$ $\tau = 0$
planar	3	$\omega_{i,i-1}$, Dr_i, Dr_{i+1}	$D\omega_{i,i-1}$, D^2r_i, D^2r_{i+1}	$\mathbf{f} = f^\nu\,\hat{\boldsymbol{\nu}}$ $\tau = \tau^\lambda\,\hat{\boldsymbol{\lambda}} + \tau^\mu\,\hat{\boldsymbol{\mu}}$
universal	2	$\omega_{i,i-1}$, $\omega_{i+1,i}$	$D\omega_{i,i-1}$, $D\omega_{i+1,i}$	$\mathbf{f} = f^i\,\hat{\mathbf{i}} + f^j\,\hat{\mathbf{j}} + f^k\,\hat{\mathbf{k}}$ $\tau = \tau^\nu\,\hat{\boldsymbol{\nu}}$

The input and initialization phase reads the model plex. The number of input motions is determined from the model, and a corresponding number of velocity and acceleration magnitudes are read. The direction of motion is established, by the right-hand rule, from points in the plex. Point coordinates for the particular linkage position being analyzed are read as needed.

A motion topology table, a shorthand plex particularly suited to motion analysis, is constructed during a scan of the plex. The scan phase also tests various characteristics of the linkage to govern the content of the motion table. Entries are made in the motion table for each degree of freedom of each constraint of each loop.

velocity phase

The velocity phase of the motion processor builds the motion coefficient matrix and computes the velocity constant vector; entries are made for degree of freedom within constraint within loop. When an input constraint is found, the entire vector is computed. Depending on the type of freedom indicated in the motion table, coefficient entries can be of three types. These are pure rotational velocity, pure translational velocity, or translational velocity due to rotation. Finally, the set of linear equations is solved for scalar unknowns, and velocity vectors are then formed with the aid of unit vectors obtained from the data portion of the plex.

acceleration phase

The motion processor enters its acceleration phase with all positions and velocities computed and available. The acceleration phase uses the coefficient matrix (actually stored in its inverse form) given by the velocity phase. The acceleration phase uses the motion table for control of the calculations, as does the velocity phase. However, to obtain the acceleration constant vector, individual constraint routines are used because of the great number and variety of terms to be computed. These constraint routines are based on the entries in a special table. The program proceeds from constraint to constraint in the motion table and then repeats the process for all loops.

A matrix multiplication is performed to obtain the desired solution vector; the relative acceleration vectors are computed, stored, and then summed to obtain the absolute accelerations of each link.

In addition to the motion of all links, the program can calculate the motion of up to three "points of interest," such as points on a coupler. One subprogram serves to compute both the velocity and acceleration of these points, the logic being shared by unique arithmetic operations for each type of motion. The program is executed immediately after the velocity phase, as well as following the acceleration phase. The output of this phase consists of the motion vector drawn from the first constraint in the loop to the point and represents the absolute linear motion of the point. The magnitude of this vector is also output.

force solution

Applied to a linkage, the two well-known conditions for static equilibrium of each body or link are that, (1) the sum of the

Planar mechanisms are statically indeterminant for forces perpendicular to the plane of motion, and therefore for moments in the plane of motion. From each force vector equation, two scalar equations are obtained by taking respective dot products with the two unit reference vectors that lie in the plane. From each moment vector equation, one scalar equation is obtained by taking a dot product with the reference vector that is perpendicular to the plane.

Because Equations 8 and 9 are vector equations, they do not yield linear equations in the unknowns as a mere consequence of dot products. However, if each unknown vector is expanded into 1, 2, or 3 components along appropriately chosen directions before the dot product is taken, linear equations can be assured. Having known directions, the output forces and torques become one-dimensional vectors with unknown magnitudes. The other forces and torques, because they act as pairs of known type, can be characterized by the type. Pairs are listed in Table 9 along with the expressions for the resulting 0, 1, 2, or 3 dimensional vectors that may be used to represent forces and torques for each pair type.

If force and torque are to be determined, extraneous degrees of freedom, such as the spin freedom in a link between two ball joints, must be removed—either by making one of the ball joints a universal joint or by defining an output torque with direction parallel to the axis of spin (such a torque will always be zero in value). Thus, a single-loop linkage can have only one output quantity. Multiloop linkages may have more than one degree of freedom, and therefore more than one resisting force and torque. In either case, the linkage may have more degrees of restraint than degrees of freedom. This can be allowed for if relationships exist between the resisting forces and torques. Further, if these relationships are linear, they can be added to the set of linear equations defining force analysis without affecting the general procedure. Non-linear relationships between resisting forces and torques, like non-linear friction forces, cannot be handled by the general procedure.

The KAM method will allow one linear relationship between resisting elements in the form $\alpha = C_1\beta + C_2$ where α denotes any output force or torque, β denotes any output force or torque other than α, and C_1, C_2 are constants that define the linear relationships between α and β.

The KAM force and torque processor requires, as input data, the coordinates for position points and the magnitudes of the input forces and torques. It calculates the entries to a matrix, solves the linear equations specified in the matrix, builds up the solution vectors (unknowns of the scalar equations are components of these vectors), and outputs the solution vectors.

Programming techniques were developed to simplify the force and torque processor. Determining the exact form of the equations for any particular link or body are a large number of conditions

that govern equation form. Some of the conditions are:

- planar or spatial loop
- link or body
- 6 possible constraint types at constraint i
- 6 possible constraint types at constraint $i + 1$
- 4 possible input conditions (no input, force input, torque input or both)
- 4 possible output conditions (no output, force output, torque output or both)
- link position in loops (first, floating, or last)
- constraint does or does not join other loops

These conditions alone give rise to 13,824 different forms for the equations.

A preliminary scan of the plex is made, loop by loop, to determine the number, say n, of unique links or bodies. Any link or body may appear in more than one loop; further, more than one link can appear in a body (the term "unique" excludes both multiple counting and the counting of links within bodies). This first scan sets the number of equations at $6n$ for spatial mechanisms and $3n$ for planar mechanisms.

In the next scan, all matrix coefficients for forces acting on each unique link or body are established in the following order:

- input and output forces
- the force acting at the i constraint in the loop
- forces acting at all body constraints
- the force acting at the $i + 1$ constraint

In each case, the coefficients in the force equations are established first; then, the coefficients due to the force moments are established in the torque equations.

The final scan for coefficient entries for torques is made in a similar order, but with the significant difference that no entries are made in the force equations, only to the moment equations.

The matrix entries made during each scan are suggested, for the forces in a spatial linkage, by Table 10. Such a table is designed to reflect each of the conditions listed above. Some symmetry in the conditions is already considered in the table. For example, the table contains entries for unknown vectors of three dimensions, rather than entries for six different constraints. Other symmetries in the tables are exploited to simplify programming. For example, only one procedure for entering the array

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

was actually programmed, although this matrix and its negative are required under several different combinations of conditions.

Table 10 Matrix for force unknowns in spatial linkage

Force Location	*Force Dimensions* 3			2		1	
Unknowns	f^i	f^j	f^k	f^λ	f^μ	f	f_o (output)
At i	1	0	0	$\hat{\mathbf{i}} \cdot \hat{\boldsymbol{\lambda}}_i$	$\hat{\mathbf{i}} \cdot \hat{\mathbf{u}}_i$	$\hat{\mathbf{i}} \cdot \hat{\mathbf{f}}$	——
	0	1	0	$\hat{\mathbf{j}} \cdot \hat{\boldsymbol{\lambda}}_j$	$\hat{\mathbf{j}} \cdot \hat{\mathbf{u}}_j$	$\hat{\mathbf{j}} \cdot \hat{\mathbf{f}}$	——
	0	0	1	$\hat{\mathbf{k}} \cdot \hat{\boldsymbol{\lambda}}_k$	$\hat{\mathbf{k}} \cdot \hat{\mathbf{u}}_k$	$\hat{\mathbf{k}} \cdot \hat{\mathbf{f}}$	——
At $i+1$	−1	0	0	$-\hat{\mathbf{i}} \cdot \hat{\boldsymbol{\lambda}}_i$	$-\hat{\mathbf{i}} \cdot \hat{\mathbf{u}}_i$	$-\hat{\mathbf{i}} \cdot \hat{\mathbf{f}}$	——
	0	−1	0	$-\hat{\mathbf{j}} \cdot \hat{\boldsymbol{\lambda}}_j$	$-\hat{\mathbf{j}} \cdot \hat{\mathbf{u}}_j$	$-\hat{\mathbf{j}} \cdot \hat{\mathbf{f}}$	——
	0	0	−1	$-\hat{\mathbf{k}} \cdot \hat{\boldsymbol{\lambda}}_k$	$-\hat{\mathbf{k}} \cdot \hat{\mathbf{u}}_k$	$-\hat{\mathbf{k}} \cdot \hat{\mathbf{f}}$	——
Body Constraints	−1	0	0	$-\hat{\mathbf{i}} \cdot \hat{\boldsymbol{\lambda}}_i$	$-\hat{\mathbf{i}} \cdot \hat{\mathbf{u}}_i$	$-\hat{\mathbf{i}} \cdot \hat{\mathbf{f}}$	$-\hat{\mathbf{i}} \cdot \hat{\mathbf{f}}_o$
	0	−1	0	$-\hat{\mathbf{j}} \cdot \hat{\boldsymbol{\lambda}}_j$	$-\hat{\mathbf{j}} \cdot \hat{\mathbf{u}}_j$	$-\hat{\mathbf{j}} \cdot \hat{\mathbf{f}}$	$-\hat{\mathbf{j}} \cdot \hat{\mathbf{f}}_o$
	0	0	−1	$-\hat{\mathbf{k}} \cdot \hat{\boldsymbol{\lambda}}_k$	$-\hat{\mathbf{k}} \cdot \hat{\mathbf{u}}_k$	$-\hat{\mathbf{k}} \cdot \hat{\mathbf{f}}$	$-\hat{\mathbf{k}} \cdot \hat{\mathbf{f}}_o$
				To Torque Equations			
At $i+1$	0	r_k	$-r_j$	$-\hat{\mathbf{i}} \cdot (\hat{\mathbf{r}} \times \hat{\boldsymbol{\lambda}})$	$-\hat{\mathbf{i}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{u}})$	$-\hat{\mathbf{i}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{f}})$	
	$-r_k$	0	r_i	$-\hat{\mathbf{j}} \cdot (\hat{\mathbf{r}} \times \hat{\boldsymbol{\lambda}})$	$-\hat{\mathbf{j}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{u}})$	$-\hat{\mathbf{j}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{f}})$	
	r_j	$-r_i$	0	$-\hat{\mathbf{k}} \cdot (\hat{\mathbf{r}} \times \hat{\boldsymbol{\lambda}})$	$-\hat{\mathbf{k}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{u}})$	$-\hat{\mathbf{k}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{f}})$	
Body Constraints	0	r_k	$-r_j$	$-\hat{\mathbf{i}} \cdot (\hat{\mathbf{r}} \times \hat{\boldsymbol{\lambda}})$	$-\hat{\mathbf{i}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{u}})$	$-\hat{\mathbf{i}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{f}})$	$-\hat{\mathbf{i}} \cdot (\hat{\mathbf{r}}_o \times \hat{\mathbf{f}}_o)$
	$-r_k$	0	r_i	$-\hat{\mathbf{j}} \cdot (\hat{\mathbf{r}} \times \hat{\boldsymbol{\lambda}})$	$-\hat{\mathbf{j}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{u}})$	$-\hat{\mathbf{j}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{f}})$	$-\hat{\mathbf{j}} \cdot (\hat{\mathbf{r}}_o \times \hat{\mathbf{f}}_o)$
	r_j	$-r_i$	0	$-\hat{\mathbf{k}} \cdot (\hat{\mathbf{r}} \times \hat{\boldsymbol{\lambda}})$	$-\hat{\mathbf{k}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{u}})$	$-\hat{\mathbf{k}} \cdot (\hat{\mathbf{r}} \times \hat{\mathbf{f}})$	$-\hat{\mathbf{k}} \cdot (\hat{\mathbf{r}}_o \times \hat{\mathbf{f}}_o)$

Summary

A technique for analyzing two- and three-dimensional linkages by solving vector equations for position, motion, and force was programmed for a digital computer. A model plex that treats a linkage as one or more vector loops is generated. The program selects the applicable equations for position solution. Algorithms are simplified by storing linkage information in the form of a tree-organized plex.

The vector equations employed in finding the velocities and accelerations are not obtained through differentiation of the position solution equations, but consist of vector relationships expressing fundamental linear and angular conditions. Motion can be calculated for each consecutive position computation, given an input motion. Because the problem of finding velocity and acceleration for linkage elements reduces to one of solving a set of simultaneous linear equations, the solutions are exact and iteration is not required.

Vector algebra is also used to obtain linear equations that define forces and torques. The coefficients for the linear equations are generated by an algorithm that scans the linkage plex and related tables. As in the case of motion, analysis reduces to the problem of solving a set of simultaneous linear equations.

CITED REFERENCES AND FOOTNOTE

1. M. A. Chace, *Development and Application of Vector Mathematics for Kinematic Analysis of Three Dimensional Mechanisms,* Doctor's Dissertation, University of Michigan (1964).

2. S. A. Brown, C. E. Drayton, and B. Mittman, "A description of the APT language," *Communications of the ACM* **6**, No. 11 (November 1963).
3. D. T. Ross and J. E. Rodriguez, "Theoretical foundations for the computer-aided design system," *AFIPS Conference Proceedings* **23**, Spring Joint Computer Conference (1963).
4. For a description of the criterion, see R. S. Hartenberg and J. Denavit, "Analysis of spatial linkages by matrix methods," The Technological Institute, Northwestern University (September 1963).
5. J. Denavit and R. S. Hartenberg, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of Applied Mechanics*, ASME Transactions (1955).
6. L. F. Knappe, "A computer oriented mechanical design system," ASME Paper 64-MECH-30 (1964).
7. H. G. ApSimon, "Algorithm for a gear-train problem," *IBM Systems Journal* **3**, No. 1, 95–103 (1964).
8. D. A. Stoddart, "Polydyne cam design," *Machine Design* (January 1953).
9. F. Freudenstein and G. N. Sandor, "Synthesis of path generating mechanisms by means of a programmed digital computer," ASME Transactions, *Journal of Engineering for Industry* (May 1956).
10. *Kinematic Analysis Method*, SP-272, Society of Automotive Engineers (May 1965). This bulletin contains six related but separate papers presented at the SAE Mid-Year Meeting by F. Bitonti, D. W. Cooper, D. N. Frayne, and H. Hansen.

This paper describes a production control system for fabrication and assembly processes. Major functions of the system, from long-range planning to detailed production scheduling and monitoring, are discussed in several parts. Techniques of mathematical statistics are applied to the calculation of parameter values in sequencing decisions. The first four parts of the paper appeared in Volume 4, No. 2, of the IBM Systems Journal.

Part V develops detailed sequences for individual work activities, considering not only delivery and resource constraints, but also the requirements for work already in progress. Part VI defines a formal statistical analysis of the process data and a logical decision rule that automatically resolves conflicts among the various production service facilities. A procedure for adjusting the resource allocations by comparing planned expenses with actual costs is discussed in Part VII.

Fabrication and assembly operations

Although the parts published in this issue are largely self-contained, each subject belongs within the frame of the production control system described in Part I.