5 A FORTRAN Program for Analysis of Planar Kinematics

In this chapter, a FORTRAN program for planar kinematic analysis is presented. The program contains only two types of lower-pair joints — a revolute joint and a translational joint. With only these two basic joints, a large class of mechanisms can be analyzed. The program is organized in a form that can be expanded to include other kinematic pairs, such as those formulated in Chap. 4. The exercises given at the end of this chapter provide a pattern to assist in expansion of the program.

The subroutines LINEAR and LU that were discussed in Chap. 3, are used in this program. They can be appended to this program without any modification. The program is written in standard FORTRAN and should run on most FORTRAN compilers without any difficulty. However, it is possible that some compilers will require minor modification of the program.

5.1 KINEMATIC ANALYSIS PROGRAM (KAP)

The main routine of the kinematic analysis program performs three major tasks:

1. Reads input data, either directly or by making calls to other input subroutines.
2. Splits the working arrays A and IA into smaller subarrays by defining pointers (addresses in arrays).
3. Makes calls to subroutine KINEM for kinematic analysis.

A detailed explanation of these three major tasks follows.

Input/Output. Input is provided to the program through the console in unformatted form: READ(1,*).... The output is written on the console screen with specified format; e.g., WRITE(1,230)....

Working Arrays. There are two main working arrays in this program—one real array A and one integer IA. The two arrays are dimensioned to 500 and 200, respectively (MAXA = 500 and MAXIA = 200), which is sufficient for most small problems. For larger problems—those with larger numbers of bodies or kinematic joints, for example—these dimensions can be increased. The program computes the minimum dimensions that are required for the two arrays and provides a warning message when the existing dimensions are not sufficient.

Number of Elements. The first request the program makes is to

```
ENTER NB, NR, NT, NG, NS, ND, NP
```

These data are defined as follows:

NB	Number of bodies in the system, including ground
NR	Number of revolute joints in the system
NT	Number of translational joints in the system
NG	Number of bodies that are attached to (or considered as) ground
NS	Number of simple constraints in the system
ND	Number of driving constraints (driving links)
NP	Number of points of interest

Further explanation of these variables and their functions is given in the following sections. The program computes the number of coordinates N and the total number of constraint equations M, including the driving constraints, from this first set of data. If N is not equal to M, then an error message is given by the program.

Subarrays. The working arrays A and IA are divided into smaller subarrays, according to the number of elements in the problem. The subarrays and their corresponding pointers and lengths are shown in Fig. 5.1. Pointers N1, N2, . . . locate the beginnings of subarrays of A, and pointers M1, M2, . . . locate the beginnings of subarrays IA; e.g., subarray TJ begins at A(N2), so TJ(2) is the same as A(N2+1), and so forth. Some pointers that have not been used, such as N8, N9, or M8, or other additional pointers can be included for further expansion of the program. The function of the subarrays is explained in the following sections.

Input Data. The main program makes calls to other subroutines, such as INBODY and INRVLT, to read additional information for the problem at hand. These subroutines are discussed in Sec. 5.1.1.

Time Parameters. The last prompt given by the program is

```
ENTER STARTING TIME, FINAL TIME, AND TIME INCREMENT
```

These values are assigned to the variable names T0, TE, and DT. During the analysis, the time parameter T is incremented by DT from T0 to TE.

Working Array A				Working Array IA			
Pointer	Subarray	Length	Description	Pointer	Subarray	Length	Description
N1	RJ	4*NR	Revolute joints	M1	IRJ	2*NR	Revolute joints
N2	TJ	7*NT	Translational joints	M2	ITJ	2*NT	Translational joints
N3	GR	3*NG	Ground	M3	IGR	6*NG	Ground
N4	SM	NS	Simple constraints	M4	ISM	2*NS	Simple constraints
N5	DR	3*ND	Drivers	M5	IDR	2*ND	Drivers
N6	RB	2*NB	Bodies				
N7	PI	2*NP	Points of interest	M7	IPI	NP	Points of interest
N10	Q	N	$\mathbf{q}$	M10	ICOL	N	Column pointers for $\mathbf{\Phi}_q$
N11	QD	N	$\dot{\mathbf{q}}$				
N12	QDD	N	$\ddot{\mathbf{q}}$				
N14	FQ	M*N	$\mathbf{\Phi}_q$				
N15	F	M	$\mathbf{\Phi}, \dot{\mathbf{\Phi}}, \ddot{\mathbf{\Phi}}$				
N16	W	N	Work array				
.							
.							
.							

Figure 5.1 Subarrays of A and IA with their corresponding pointers and lengths.

Kinematic Analysis. After all of the input data are read by the program, the main program makes a call to subroutine KINEM for kinematic analysis. Prior to this call, three variables are defined and given specific values:

NRMAX	Maximum number of iterations allowed for the Newton-Raphson algorithm at each time step
FEPS	Maximum error allowed for constraint violation
EPSLU	Error tolerance for the L-U factorization algorithm

The user may modify the program to assign values to these variables, by a READ statement, depending on the accuracy required in the response computation.

The main routine of the KAP program is as follows:

```
C..........KINEMATIC ANALYSIS...........
C
C............Main Program...............
C
      COMMON /CONST / NRMAX,FEPS,EPSLU
      COMMON /MPNTR / M1,M2,M3,M4,M5,M6,M7,M8,M9,M10
      COMMON /NELMNT/ NB,NR,NT,NG,NG3,NS,ND,NP
      COMMON /NPNTR / N1,N2,N3,N4,N5,N6,N7,N10,N11,N12,N14,N15,N16
      COMMON /ROWCOL/ IR,IC,M,N
      COMMON /TIME  / T0,TE,DT,T
      DIMENSION A(500),IA(200)
C.....For more space in A and IA arrays, increase the dimension and
C.....update MAXA and MAXIA accordingly
      MAXA =500
      MAXIA=200
C.....Read number of bodies, revolute joints, translational joints,
C.....grounded bodies, simple constraints, drivers, points of interest
   10 WRITE(1,200)
      READ(1,*  ) NB,NR,NT,NG,NS,ND,NP
C.....Determine number of coordinates N and number of constraints M
      N=3*NB
      M=2*(NR+NT)+3*NG+NS+ND
C.....N must be equal to M (including the driver constraints)
      IF (M.EQ.N) GOTO 20
      WRITE(1,210) N,M
      GOTO 10
C.....Define pointers and split A and IA into subarrays
C.....Subarrays for storing input data
   20 N1=1
      N2=N1+4*NR
      N3=N2+7*NT
      N4=N3+3*NG
      N5=N4+  NS
      N6=N5+3*ND
      N7=N6+2*NB
      M1=1
      M2=M1+2*NR
      M3=M2+2*NT
      M4=M3+6*NG
      M5=M4+2*NS
      M7=M5+2*ND
C.....Subarrays for vector of coordinates, velocities, etc.
      N10=N7+2*NP
      N11=N10+N
      N12=N11+N
      N14=N12+N
      N15=N14+N*M
      N16=N15+M
      NUSED=N16+M-1
      M10=M7+NP
      MUSED=M10+N-1
C.....Check for sufficient storage space in A and IA arrays
      IF(NUSED.LE.MAXA .AND. MUSED.LE.MAXIA)GOTO 30
      WRITE(1,220) NUSED,MUSED
      STOP
C.....Read initial estimates on the coordinates
   30 CALL INBODY (A(N10),NB)
C.....Read revolute joints data
      IF (NR.GT.0) CALL INRVLT (A(N1),IA(M1),NR)
C.....Read translational joints data
      IF (NT.GT.0) CALL INTRAN (A(N2),IA(M2),NT,A(N10),NB)
C.....Read ground constraints data
      NG3=3*NG
      IF (NG.GT.0) CALL INGRND (A(N3),IA(M3),NG,A(N10),NG3,NB)
C.....Read simple constraints data
      IF (NS.GT.0) CALL INSMPL (A(N4),IA(M4),NS,A(N10),NB)
C.....Read driver constraints dataA
      IF (ND.GT.0) CALL INDRVR (A(N5),IA(M5),ND)
C.....Read special points of interest data
      IF (NP.GT.0) CALL INPOIN (A(N7),IA(M7),NP)
C.....Read initial time, final time, and the time increment
      WRITE(1,230)
      READ(1,*   ) TO,TE,DT
C.....End of input data
      NRMAX=20
      FEPS =0.001
      EPSLU=0.001
C.....Start KINEMATIC ANALYSIS.....
      CALL KINEM (A,IA,MAXA,MAXIA)
      STOP
  200 FORMAT(5X,'ENTER NB,NR,NT,NG,NS,ND,NP')
  210 FORMAT(5X,'***INPUT ERROR*** N=',I3,'  M=',I3)
  220 FORMAT(5X,'***DIMENSION ON A AND/OR IA ARRAYS NOT SUFFICIENT***',
     +    /,10X,'MINIMUM DIMENSION ON A  MUST BE',I5,
     +    /,10X,'MINIMUM DIMENSION ON IA MUST BE',I5)
  230 FORMAT(5X,'ENTER STARTING TIME, FINAL TIME, AND DT')
      END
```

5.1.1 Model-Description Subroutines

The following subroutines are called by the main routine of KAP to read the description of the model.

Subroutine INBODY. This subroutine reads initial estimates of x_i, y_i and ϕ_i for each body with a prompt:

FOR BODY i ENTER INITIAL EST. ON X, Y, PHI

This information is saved in array Q, dimensioned as Q(3,NB). For example, for body i,

$$x_i \rightarrow \text{Q}(1,\text{I})$$
$$y_i \rightarrow \text{Q}(2,\text{I})$$
$$\phi_i \rightarrow \text{Q}(3,\text{I})$$

Subroutine INBODY is as follows:

```
      SUBROUTINE INBODY (Q,NB)
      DIMENSION Q(3,NB)
      DO 10 I=1,NB
         WRITE(1,200) I
         READ(1,*) (Q(J,I),J=1,3)
   10 CONTINUE
      RETURN
  200 FORMAT(5X,'FOR BODY',I4,' ENTER INITIAL EST. ON X, Y, PHI')
      END
```

Subroutine INRVLT. This subroutine is called if NR > 0, to read information describing the revolute joints in the system (refer to Sec. 4.2.1). The prompt given by this subroutine is

```
FOR REV. JOINT NO. k ENTER
         BODY NOS. I AND J, THEN
         XI-P-I, ETA-P-I, XI-P-J, ETA-P-J
```

This prompt is repeated for $k = 1, \ldots, \text{NR}$. Body numbers i and j that are connected by revolute joint k are stored in array IRJ, dimensioned as IRJ(NR,2).

The local components of vectors that locate the revolute joints on the bodies are stored in array RJ, dimensioned as RJ(NR,4). For example, for the kth revolute joint in a system, connecting bodies i and j, the entries for the IRJ and RJ arrays are

$$
\begin{array}{ll}
I \rightarrow \text{IRJ(K,1)} & J \rightarrow \text{IRJ(K,2)} \\
\xi_i^P \rightarrow \text{RJ(K,1)} & \xi_j^P \rightarrow \text{RJ(K,3)} \\
\eta_i^P \rightarrow \text{RJ(K,2)} & \eta_j^P \rightarrow \text{RJ(K,4)}
\end{array}
$$

Subroutine INRVLT is as follows:

```
      SUBROUTINE INRVLT (RJ,IRJ,NR)
      DIMENSION RJ(NR,4),IRJ(NR,2)
      DO 10 K=1,NR
         WRITE(1,200) K
         READ(1,*) (IRJ(K,L),L=1,2),(RJ(K,L),L=1,4)
   10 CONTINUE
      RETURN
  200 FORMAT(5X,'FOR REV. JOINT NO.',I3,' ENTER BODY NOS. I AND J',/,
     +       10X,'XI-P-I,ETA-P-I,XI-P-J,ETA-P-J')
      END
```

Subroutine INTRAN. The subroutine INTRAN is called if NT > 0, to read information describing the translational joints in the system (refer to Sec. 4.2.1). The prompt given by this subroutine is

```
FOR TRAN. JOINT NO. k ENTER
          BODY NOS. I AND J, THEN
          XI-P-I, ETA-P-I, XI-Q-I, ETA-Q-I
          XI-P-J, ETA-P-J
```

This prompt is repeated for $k = 1, \ldots, \text{NT}$. Body numbers i and j that are connected by the kth translational joint are stored in array ITJ, dimensioned as ITJ(NT,2). The local coordinates of points P_i and P_j are stored in array TJ, dimensioned as TJ(NT,7). The local coordinates of point Q_i are not saved directly. Instead,

$$\alpha_i = \xi_i^P - \xi_i^Q, \qquad \beta_i = \eta_i^P - \eta_i^Q$$

are stored in vector TJ. For the kth translational joint in the system, the elements of the ITJ and TJ arrays are:

$$\begin{array}{rl} I \rightarrow \text{ITJ(K,1)} & J \rightarrow \text{ITJ(K,2)} \\ \xi_i^P \rightarrow \text{TJ(K,1)} & \xi_j^P \rightarrow \text{TJ(K,5)} \\ \eta_i^P \rightarrow \text{TJ(K,2)} & \eta_j^P \rightarrow \text{TJ(K,6)} \\ \alpha_i \rightarrow \text{TJ(K,3)} & \phi_i^0 - \phi_j^0 \rightarrow \text{TJ(K,7)} \\ \beta_i \rightarrow \text{TJ(K,4)} & \end{array}$$

Subroutine INTRAN is as follows:

```
      SUBROUTINE INTRAN (TJ,ITJ,NT,Q,NB)
      DIMENSION TJ(NT,7),ITJ(NT,2),Q(3,NB)
      DO 10 K=1,NT
         WRITE(1,200) K
         READ(1,*) (ITJ(K,L),L=1,2),(TJ(K,L),L=1,6)
         TJ(K,3)=TJ(K,1)-TJ(K,3)
         TJ(K,4)=TJ(K,2)-TJ(K,4)
   10    TJ(K,7)=Q(3,ITJ(K,1))-Q(3,ITJ(K,2))
      RETURN
  200 FORMAT(5X,'FOR TRAN. JOINT NO.',I3,' ENTER BODY NOS. I AND J',/,
     +10X,'XI-P-I,ETA-P-I,XI-Q-I,ETA-Q-I,XI-P-J,ETA-P-J')
      END
```

Subroutine INGRND. The subroutine INGRND is called if NG > 0, to read the number(s) of a body or bodies constrained to the ground by giving the prompt

ENTER BODY NO. FOR NO. k GROUNDED BODY

This prompt is repeated NG times. This subroutine also transfers initial estimates of the coordinates for the grounded bodies from array Q to array GR, dimensioned as GR(3*NG). Each body that is constrained to the ground is treated as having three simple constraints on its x, y, and ϕ motion. This information is stored in an integer array IGR, dimensioned as IGR(3*NG,2). The way the data are stored in the IGR and GR arrays is similar to the storage of data in the SM and ISM arrays for simple constraints, as will be shown in the next subsection.

Subroutine INGRND is as follows:

```
      SUBROUTINE INGRND (GR,IGR,NG,Q,NG3,NB)
      DIMENSION GR(NG3),IGR(NG3,2),Q(3,NB)
      DO 10 K=1,NG
         WRITE(1,200) K
         READ(1,*) IB
         DO 10 J=1,3
         KK=(K-1)*3+J
         IGR(KK,1)=IB
         IGR(KK,2)=J
         GR(KK)=Q(J,IB)
   10 CONTINUE
      RETURN
  200 FORMAT(5X,'ENTER BODY NO. FOR NO.',I3,' GROUNDED BODY')
      END
```

Subroutine INSMPL. This subroutine is called if NS > 0, to read information on the bodies that have simple constraints (refer to Eqs. 4.31, 4.32, and 4.33). The prompt given by this subroutine is

```
FOR SIMPLE CONSTRAINT NO. k ENTER
          BODY NO. AND 1, 2, OR 3 FOR X, Y, OR PHI CONSTRAINT DIRECTION
```

This prompt is repeated NS times. The information is stored in array ISM, dimensioned as ISM(NS,2). In addition, this subroutine transfers the initial values of the coordinates from array *Q* to array SM, dimensioned as SM(NS). For example, if the *k*th simple constraint acts in the *y* direction on body *i*, then

$$\mathrm{I} \rightarrow \mathrm{ISM(K,1)},\ 2(\text{for } y) \rightarrow \mathrm{ISM(K,2)}\ ,\ \mathrm{Q(2,I)} \rightarrow \mathrm{SM(K)}$$

where Q(2,I) contains the initial value on the *y* coordinate of body *i*.

The entries of array SM will be used as the constant c_1, c_2, or c_3 in Eq. 4.31, 4.32, or 4.33.

Subroutine INSMPL is as follows:

```
      SUBROUTINE INSMPL (SM,ISM,NS,Q,NB)
      DIMENSION SM(NS),ISM(NS,2),Q(3,NB)
      DO 10 K=1,NS
         WRITE(1,200) K
         READ(1,*) (ISM(K,L),L=1,2)
         SM(K)=Q(ISM(K,2),ISM(K,1))
   10 CONTINUE
      RETURN
  200 FORMAT(5X,'FOR SIMPLE CONSTRAINT NO.',I3,' ENTER BODY NO.',/,
     + 10X,'AND 1,2,OR 3 FOR X,Y,OR PHI CONSTRAINT DIRECTION')
      END
```

Subroutine INDRVR. This subroutine is called if ND > 0, to read information on the driving link(s) (refer to Sec. 4.2.8). The prompt given by this subroutine is

```
FOR DRIVER NO. k ENTER
          BODY NO., 1, 2, OR 3 FOR X, Y, OR PHI, THEN
          INITIAL POSITION, VELOCITY, AND ACCELERATION
```

This prompt is repeated ND times. The body number(s) and indices indicating *x*, *y*, or ϕ are constrained and stored in an integer array IDR, dimensioned as IDR(ND,2). The initial value of *x*, *y*, or ϕ and its velocity and acceleration at $t = 0$ are stored in array DR, dimensioned as DR(ND,3). For example, if the *k*th driver constraint is acting on the rotation of body *i* as follows:

$$\phi_i - \phi_i^0 - \omega t - 0.5\alpha t^2 = 0$$

then

$$I \rightarrow \mathrm{IDR(K,1)}, \qquad 3\ (\text{for } \phi) \rightarrow \mathrm{IDR(K,2)}\,,$$
$$\phi_i^0 \rightarrow \mathrm{DR(K,1)}, \qquad \omega \rightarrow \mathrm{DR(K,2)}, \qquad \alpha \rightarrow \mathrm{DR(K,3)}$$

Subroutine INDRVR is as follows:

```
      SUBROUTINE INDRVR (DR,IDR,ND)
      DIMENSION DR(ND,3),IDR(ND,2)
      DO 10 K=1,ND
         WRITE(1,200) K
         READ(1,*) (IDR(K,L),L=1,2),(DR(K,L),L=1,3)
   10 CONTINUE
      RETURN
  200 FORMAT(5X,'FOR DRIVER NO.',I3,' ENTER BODY NO.',/,
     +       10X,'1, 2, OR 3 FOR X, Y, OR PHI',/,
     +       10X,'INITIAL POSITION, VELOCITY, AND ACCELERATION')
      END
```

Subroutine INPOIN. This subroutine is called if NP > 0, to read information on points of interest defined by the user. The user may specify some points of interest on one or more bodies (refer to Eqs. 4.1, 4.43, and 4.45). Hence, the program will report the global position, velocity, and acceleration for these points at every time step. Note that the program reports the coordinates of all the bodies in the system automatically, and therefore there is no need to specify the centroid of a body as a point of interest. The prompt from this subroutine is

```
FOR POINT OF INTEREST NO. k ENTER
          BODY NO., THEN XI-P AND ETA-P COORDINATES
```

This prompt is repeated NP times. The body numbers are stored in array IPI, dimensioned as IPI(NP). The ξ_i^P and η_i^P coordinates are stored in array PI, dimensioned as PI(NP,2).

Subroutine INPOIN is as follows:

```
      SUBROUTINE INPOIN (PI,IPI,NP)
      DIMENSION PI(NP,2),IPI(NP)
      DO 10 K=1,NP
         WRITE(1,200) K
         READ(1,*) IPI(K),(PI(K,L),L=1,2)
   10 CONTINUE
      RETURN
  200 FORMAT(5X,'FOR POINT OF INTEREST NO.',I3,' ENTER BODY NO.',/,
     + 10X,'XI-P AND ETA-P COORDINATES')
      END
```

5.1.2 Kinematic Analysis

Following the process of inputting data to describe a model, the main routine calls subroutine KINEM. This subroutine is organized according to Algorithm K-II of Sec. 3.2.2, with a few minor additional steps.

Subroutine KINEM. This subroutine increments the time variable T from the initial time T0 to final time TE by steps DT. At each time step, the subroutine performs position, velocity, and acceleration analysis, and reports the results by making calls to other subroutines. Two flags are set by this subroutine:

JACOB	A flag for Jacobian matrix evaluation.
	EQ.0: Jacobian matrix does not need to be evaluated.
	EQ.1: Jacobian matrix must be evaluated.

IFNCT	A flag for evaluation of constraint equations.
	EQ.0: No function needs to be evaluated.
	EQ.1: Constraint equations must be evaluated.
	EQ.2: The right side of the velocity equations must be evaluated.
	EQ.3: The right side of the acceleration equations must be evaluated.

Position Analysis. The flags JACOB and IFNCT are both set to 1 and a call is made to subroutine NUTON2 for position analysis.

Velocity Analysis. The flag JACOB is set to 1 and IFNCT is set to 2. A call to subroutine FUNCT evaluates the Jacobian matrix and the right side of the velocity equations. At this step, subroutine LINEAR (refer to Sec. 3.3.5) is used to solve for the velocities according to Eq. 3.14.

Acceleration Analysis. The flag JACOB is set to 0, since the Jacobian matrix and its corresponding **L** and **U** matrices are still valid from the velocity analysis step, and IFNCT is set to 3. A call to subroutine FUNCT evaluates the right side of acceleration equations. At this step, subroutine LINEAR is used to solve for the acceleration, according to Eq. 3.16.

The velocities and accelerations determined by subroutine LINEAR are originally stored in array F. The velocities and accelerations are then transferred from array F to arrays QD and QDD, respectively, by subroutine TRANSF. The contents of arrays QD and QDD are organized in a way similar to the arrangements of array Q.

Computationally, the evaluation of trigonometric functions is time-consuming. Each time new values for coordinates are calculated, the sine and cosine of the rotational coordinates are computed and stored in array RB by subroutine TRIG. Array RB is dimensioned RB(NB,2), and, for example, for body i,

$$\sin\phi_i \rightarrow \text{RB(I,1)}, \qquad \cos\phi_i \rightarrow \text{RB(I,2)}$$

The contents of array RB are used in several other subroutines.

Subroutines KINEM, TRANSF, and TRIG are as follows:

```
      SUBROUTINE KINEM (A,IA,MAXA,MAXIA)
      COMMON /ANALYS/ JACOB,IFNCT
      COMMON /CONST / NRMAX,FEPS,EPSLU
      COMMON /MPNTR / M1,M2,M3,M4,M5,M6,M7,M8,M9,M10
      COMMON /NELMNT/ NB,NR,NT,NG,NG3,NS,ND,NP
      COMMON /NPNTR / N1,N2,N3,N4,N5,N6,N7,N10,N11,N12,N14,N15,N16
      COMMON /ROWCOL/ IR,IC,M,N
      COMMON /TIME  / TO,TE,DT,T
      DIMENSION A(MAXA),IA(MAXIA)
      WRITE (1,200)
C.....Initialize time variable
      ISTEP=0
      T=TO
C.....Calculate sine and cosine of rotational coordinates
      CALL TRIG (A(N6),A(N10),NB)
C.....Position (coordinate) analysis......
   10 JACOB=1
      IFNCT=1
      CALL NUTON2 (A,IA,MAXA,MAXIA,A(N10),A(N15),NB,JACOB)
C.....Velocity analysis..................
      IFNCT=2
      CALL FUNCT (A,IA,MAXA,MAXIA,A(N10),A(N11),A(N14),A(N15),JACOB)
      CALL LINEAR (A(N14),A(N15),A(N16),IA(M10),M,JACOB,EPSLU)
```

```
C.....Transfer velocities from F array to QD array
      CALL TRANSF (A(N11),A(N15),N)
C.....Acceleration analysis...............
      JACOB=0
      IFNCT=3
      CALL FUNCT (A,IA,MAXA,MAXIA,(N10),A(N11),A(N14),A(N15),JACOB)
      CALL LINEAR (A(N14),A(N15),A(N16),IA(M10),M,JACOB,EPSLU)
C.....Transfer accelerations from F array to QDD array
      CALL TRANSF (A(N12),A(N15),N)
C.....Report the result for this time step
      CALL REPORT (A(N6),A(N7),A(N10),A(N11),A(N12),IA(M7),NB,NP,T)
C.....Increment the time variable
      ISTEP=ISTEP+1
      T=T0+DT*FLOAT(ISTEP)
      IF (T.GT.TE.OR.DT.EQ.0.0) STOP
      GO TO 10
  200 FORMAT(///,10X,'*****  KINEMATIC ANALYSIS   *****',//)
      END

      SUBROUTINE TRANSF (B,F,N)
      DIMENSION B(N),F(N)
      DO 10 I=1,N
   10    B(I)=F(I)
      RETURN
      END

      SUBROUTINE TRIG (RB,Q,NB)
      DIMENSION RB(NB,2),Q(3,NB)
      DO 10 I=1,NB
         RB(I,1)=SIN(Q(3,I))
         RB(I,2)=COS(Q(3,I))
   10 CONTINUE
      RETURN
      END
```

Subroutine NUTON2. Subroutine NUTON2 is similar to subroutine NEWTON of Sec. 3.4.3, with minor modifications. This subroutine is called by subroutine KINEM for position analysis when JACOB = 1 and IFNCT = 1. This subroutine calls subroutine FUNCT to evaluate the Jacobian matrix and constraint equations. If Newton-Raphson iteration fails to converge in NRMAX iterations, analysis will be terminated with the message

```
***CONVERGENCE FAILED***
```

This failure may be caused by either of the following:

1. Large time increments DT (numerical failure)
2. Nonexistence of a solution (physical impossibility)

Subroutine NUTON2 is as follows:

```
      SUBROUTINE NUTON2 (A,IA,MAXA,MAXIA,Q,F,NB,JACOB)
      COMMON /CONST / NRMAX,FEPS,EPSLU
      COMMON /MPNTR / M1,M2,M3,M4,M5,M6,M7,M8,M9,M10
      COMMON /NPNTR / N1,N2,N3,N4,N5,N6,N7,N10,N11,N12,N14,N15,N16
      COMMON /ROWCOL/ IR,IC,M,N
      DIMENSION Q(N),A(MAXA),IA(MAXIA),F(M)
      DO 20 I=1,NRMAX
         CALL FUNCT (A,IA,MAXA,MAXIA,A(N10),A(N11),A(N14),A(N15),JACOB)
         CALL LINEAR (A(N14),A(N15),A(N16),IA(M10),M,JACOB,EPSLU)
      ICONVR=0
      DO 10 J=1,N
         IF(ABS(F(J)).GT.FEPS) ICONVR=1
10       Q(J)=Q(J)-F(J)
         CALL TRIG (A(N6),Q,NB)
      IF (ICONVR) 30,30,20
20 CONTINUE
    WRITE(*,200)
    STOP
30 RETURN
200 FORMAT(5X,'***CONVERGENCE FAILED***')
    END
```

Subroutine REPORT. This subroutine reports the values of coordinates for all bodies in the system at the end of each time step. In addition, the global coordinates, velocities, and accelerations of the special points of interest are computed and reported.

Subroutine REPORT is as follows:

```
      SUBROUTINE REPORT (RB,PI,Q,QD,QDD,IPI,NB,NP,T)
      DIMENSION Q(3,NB),QD(3,NB),QDD(3,NB),PI(NP,2),IPI(NP),RB(NB,2)
      WRITE(1,200) T
      DO 10 I=1,NB
10      WRITE(1,210)I,(Q(J,I),J=1,3),(QD(J,I),J=1,3),(QDD(J,I),J=1,3)
15    IF (NP.EQ.0) RETURN
      WRITE(1,220)
      DO 20 K=1,NP
         I=IPI(K)
         XPMX=PI(K,1)*RB(I,2)-PI(K,2)*RB(I,1)
         YPMY=PI(K,1)*RB(I,1)+PI(K,2)*RB(I,2)
         XP  =Q(1,I)+XPMX
         YP  =Q(2,I)+YPMY
         XDP =QD(1,I)-YPMY*QD(3,I)
         YDP =QD(2,I)+XPMX*QD(3,I)
         XDDP=QDD(1,I)-XPMX*QD(3,I)**2-YPMY*QDD(3,I)
         YDDP=QDD(2,I)-YPMY*QD(3,I)**2+XPMX*QDD(3,I)
         WRITE(1,230) K,XP,YP,XDP,YDP,XDDP,YDDP
20    CONTINUE
      RETURN
200   FORMAT(/,'TIME =',F10.4,/,'---------------',/,
     +       ' BODY',5X,'X',7X,'Y',5X,'PHI',6X,'XD',6X,'YD',4X,'PHID',
     +       'XD',6X,'YD',6X,'XDD',6X,'YDD')
210   FORMAT(I3,6F8.3,3F9.3)
220   FORMAT('POINTS OF INTEREST',/,' NO.',5X,'X',7X,'Y',6X,
             'XD',6X,'YD',6X,'XDD',6X,'YDD')
230   FORMAT(I3,4F8.3,2F9.3)
      END
```

5.1.3 Function Evaluation

Subroutine FUNCT. The constraint equations, the right sides of the velocity and acceleration equations, and the Jacobian matrix are evaluated by calling subroutine FUNCT. This subroutine initializes a row number counter IR to zero. This counter gives the total number of functions or rows in the Jacobian matrix. The counter is incremented by subroutines RVLT, TRAN, SMPL, and DRVR. After all of the constraint equations have been considered, IR is equal to M (or N).

Prior to evaluation of the entries in the Jacobian matrix, the entries in the matrix FQ are initialized to zero. Subroutine FUNCT calls subroutines RVLT, TRAN, SMPL (twice), and DRVR to evaluate those equations and the Jacobian matrix corresponding to revolute joints, translational joints, ground, simple constraints, and driver links. The order of calling these subroutines has been written arbitrarily and does not have any significance.

Subroutine FUNCT is as follows:

```
      SUBROUTINE FUNCT (A,IA,MAXA,MAXIA,Q,QD,FQ,F,JACOB)
      COMMON /MPNTR / M1,M2,M3,M4,M5,M6,M7,M8,M9,M10
      COMMON /NELMNT/ NB,NR,NT,NG,NG3,NS,ND,NP
      COMMON /NPNTR / N1,N2,N3,N4,N5,N6,N7,N10,N11,N12,N14,N15,N16
      COMMON /ROWCOL/ IR,IC,M,N
      DIMENSION A(MAXA),IA(MAXIA),Q(N),QD(N),FQ(M,N),F(M)
      IR=0
      IF (JACOB.EQ.0) GOTO 20
      DO 10 I=1,M
         DO 10 J=1,N
10          FQ(I,J)=0.0
20    IF (NR.GT.0) CALL RVLT (A(N1),IA(M1),A(N6),Q,QD,FQ,F,NR,NB)
      IF (NT.GT.0) CALL TRAN (A(N2),IA(M2),A(N6),Q,QD,FQ,F,NT,NB)
      IF (NG.GT.0) CALL SMPL (A(N3),IA(M3),Q,FQ,F,NG3,NB)
      IF (NS.GT.0) CALL SMPL (A(N4),IA(M4),Q,FQ,F,NS,NB)
      IF (ND.GT.0) CALL DRVR (A(N5),IA(M5),Q,FQ,F,ND,NB)
      RETURN
      END
```

Subroutine RVLT. Subroutine RVLT is called by subroutine FUNCT when NR > 0. It evaluates the constraint equation violations, the right sides of the velocity and acceleration equations, and the entries of the Jacobian matrix corresponding to the revolute joints in the system (refer to Sec. 4.2.1). Array F is used to store the constraint equation violations and the right sides of the velocity and acceleration equations, depending on the value of the flag IFNCT. The nonzero entries of the Jacobian matrix are stored in matrix form in array FQ, when JACOB = 1.

Subroutine RVLT is as follows:

```
      SUBROUTINE RVLT (RJ,IRJ,RB,Q,QD,FQ,F,NR,NB)
      COMMON /ANALYS/ JACOB,IFNCT
      COMMON /ROWCOL/ IR,IC,M,N
      DIMENSION RJ(NR,4),IRJ(NR,2),RB(NB,2),Q(3,NB),QD(3,NB),
     +          FQ(M,N),F(M)
      DO 100 K=1,NR
         I=IRJ(K,1)
         J=IRJ(K,2)
         XPIMXI= RJ(K,1)*RB(I,2)-RJ(K,2)*RB(I,1)
         YPIMYI= RJ(K,1)*RB(I,1)+RJ(K,2)*RB(I,2)
         XPJMXJ= RJ(K,3)*RB(J,2)-RJ(K,4)*RB(J,1)
         YPJMYJ= RJ(K,3)*RB(J,1)+RJ(K,4)*RB(J,2)
         GOTO (10,20,30) ,IFNCT
C.....Constraint equations
   10    F(IR+1)= Q(1,I)+XPIMXI-Q(1,J)-XPJMXJ
         F(IR+2)= Q(2,I)+YPIMYI-Q(2,J)-YPJMYJ
         GOTO 40
C....Right-hand-side of velocity equations
   20    F(IR+1)= 0.0
         F(IR+2)= 0.0
         GOTO 40
C....Right-hand-side of acceleration equations
   30    F(IR+1)= XPIMXI*QD(3,I)**2-XPJMXJ*QD(3,J)**2
         F(IR+2)= YPIMYI*QD(3,I)**2-YPJMYJ*QD(3,J)**2
   40    IF (JACOB) 60,60,50
```

```
C.....Jacobian matrix nonzero entries
   50     ICI= 3*(I-1)
          ICJ= 3*(J-1)
          FQ(IR+1,ICI+1)= 1.0
          FQ(IR+1,ICI+3)=-YPIMYI
          FQ(IR+1,ICJ+1)=-1.0
          FQ(IR+1,ICJ+3)= YPJMYJ
          FQ(IR+2,ICI+2)= 1.0
          FQ(IR+2,ICI+3)= XPIMXI
          FQ(IR+2,ICJ+2)=-1.0
          FQ(IR+2,ICJ+3)=-XPJMXJ
   60     IR=IR+2
  100 CONTINUE
      RETURN
      END
```

Subroutine TRAN. This subroutine is called by subroutine FUNCT when NT > 0. It evaluates the constraint equation violations, the right sides of the velocity and acceleration equations, and the entries in the Jacobian matrix corresponding to the translational joints in the system (refer to Secs. 4.2.1 and 4.3.1). The organization of this subroutine is similar to that of subroutine RVLT.

Subroutine TRAN is as follows:

```
      SUBROUTINE TRAN (TJ,ITJ,RB,Q,QD,FQ,F,NT,NB)
      COMMON /ANALYS/ JACOB,IFNCT
      COMMON /ROWCOL/ IR,IC,M,N
      DIMENSION TJ(NT,7),ITJ(NT,2),RB(NB,2),Q(3,NB),QD(3,NB),
     +          FQ(M,N),F(M)
      DO 100 K=1,NT
         I=ITJ(K,1)
         J=ITJ(K,2)
         XPIMXI  = TJ(K,1)*RB(I,2)-TJ(K,2)*RB(I,1)
         YPIMYI  = TJ(K,1)*RB(I,1)+TJ(K,2)*RB(I,2)
         XPJMXJ  = TJ(K,5)*RB(J,2)-TJ(K,6)*RB(J,1)
         YPJMYJ  = TJ(K,5)*RB(J,1)+TJ(K,6)*RB(J,2)
         XPIMXQI= TJ(K,3)*RB(I,2)-TJ(K,4)*RB(I,1)
         YPIMYQI= TJ(K,3)*RB(I,1)+TJ(K,4)*RB(I,2)
         GOTO (10,20,30) ,IFNCT
C.....Constraint equations
   10    F(IR+1)= XPIMXQI*(Q(2,J)+YPJMYJ-Q(2,I)-YPIMYI)
     +           -YPIMYQI*(Q(1,J)+XPJMXJ-Q(1,I)-XPIMXI)
         F(IR+2)= Q(3,I)-Q(3,J)-TJ(K,7)
         GOTO 40
C.....Right-hand-side of velocity equations
   20    F(IR+1)= 0.0
         F(IR+2)= 0.0
         GOTO 40
C.....Right-hand-side of acceleration equations
   30   F(IR+1)=-QD(3,I)*(2.*(XPIMXQI*(QD(1,I)-QD(1,J))
     +                       +YPIMYQI*(QD(2,I)-QD(2,J)))
     +               +QD(3,I)*(XPIMXQI*(Q (2,I)-Q (2,J))
     +                       -YPIMYQI*(Q (1,I)-Q (1,J))))
        F(IR+2)= 0.0
   40    IF (JACOB) 60,60,50
C.....Jacobian matrix nonzero entries
   50    ICI= 3*(I-1)
         ICJ= 3*(J-1)
         FQ(IR+1,ICI+1)= YPIMYQI
         FQ(IR+1,ICI+2)=-XPIMXQI
         FQ(IR+1,ICI+3)=-(Q(1,J)+XPJMXJ-Q(1,I))*XPIMXQI
     +                  -(Q(2,J)+YPJMYJ-Q(2,I))*YPIMYQI
```

```
            FQ(IR+1,ICJ+1)=-YPIMYQI
            FQ(IR+1,ICJ+2)= XPIMXQI
            FQ(IR+1,ICJ+3)= XPJMXJ*XPIMXQI+YPJMYJ*YPIMYQI
            FQ(IR+2,ICI+3)= 1.0
            FQ(IR+2,ICJ+3)=-1.0
   60       IR=IR+2
  100   CONTINUE
        RETURN
        END
```

Subroutine SMPL. Subroutine SMPL evaluates the constraint equation violations, the right sides of the velocity and acceleration equations, and the entries in the Jacobian matrix for simple constraints (refer to Sec. 4.2.7). It is called by subroutine FUNCT when NG > 0 and NS > 0. Grounded bodies are treated as bodies with three simple constraints. The organization of this subroutine is similar to that of subroutine RVLT.

Subroutine SMPL is as follows:

```
        SUBROUTINE SMPL (SM,ISM,Q,FQ,F,NS,NB)
        COMMON /ANALYS/ JACOB,IFNCT
        COMMON /ROWCOL/ IR,IC,M,N
        DIMENSION SM(NS),ISM(NS,2),Q(3,NB),FQ(M,N),F(M)
        DO 100 K=1,NS
            I=ISM(K,1)
            L=ISM(K,2)
            GOTO (10,20,20) ,IFNCT
C.....Constraint equation
   10       F(IR+1)= Q(L,I)-SM(K)
            GOTO 40
C.....Right-hand-side of velocity/acceleration equation
   20       F(IR+1)= 0.0
   40       IF (JACOB) 60,60,50
C.....Jacobian matrix nonzero entry
   50       ICI= 3*(I-1)
            FQ(IR+1,ICI+L)= 1.0
   60       IR=IR+1
  100   CONTINUE
        RETURN
        END
```

Subroutine DRVR. This subroutine is called by subroutine FUNCT when ND > 0. It evaluates constraint equation violations, the right sides of the velocity and acceleration equations, and the entries in the Jacobian matrix for the driver links in the system (refer to Sec. 4.2.8). This is the only function evaluation subroutine in which the time variable T is used explicitly.

Subroutine DRVR is as follows:

```
        SUBROUTINE DRVR (DR,IDR,Q,FQ,F,ND,NB)
        COMMON /ANALYS/ JACOB,IFNCT
        COMMON /ROWCOL/ IR,IC,M,N
        COMMON /TIME   / TO,TE,DT,T
        DIMENSION DR(ND,3),IDR(ND,2),Q(3,NB),FQ(M,N),F(M)
        DO 100 K=1,ND
            I=IDR(K,1)
            L=IDR(K,2)
            GOTO (10,20,30) ,IFNCT
C.....Constraint equation
   10       F(IR+1)= Q(L,I)-DR(K,1)-T*(DR(K,2)+T*DR(K,3)/2.0)
            GOTO 40
```

```
C.....Right-hand-side of velocity equation
   20    F(IR+1)= DR(K,2)+T*DR(K,3)
         GOTO 40
C.....Righ-hand-side of acceleration equation
   30    F(IR+1)= DR(K,3)
   40    IF (JACOB) 60,60,50
C.....Jacobian matrix nonzero entry
   50    ICI= 3*(I-1)
         FQ(IR+1,ICI+L)= 1.0
   60    IR=IR+1
  100 CONTINUE
      RETURN
      END
```

5.1.4 Input Prompts

A list of all the prompts given by the program KAP is given here. The prompts are labeled for easy reference, from (a) through (i). In examples that follow, each prompt is referred to by its corresponding label. The parameter k in the prompts is problem-dependent. For example, in a model with two revolute joints, prompt (c) is repeated twice and k takes on values of 1 and 2.

The prompts given are as follows:

(a) ENTER NB, NR, NT, NG, NS, ND, NP

(b) FOR BODY k ENTER INITIAL EST. ON X, Y, PHI

(c) FOR REV. JOINT NO. k ENTER BODY NOS. I AND J
  XI-P-I, ETA-P-I, XI-P-J, ETA-P-J

(d) FOR TRAN. JOINT NO. k ENTER BODY NOS. I AND J
  XI-P-I, ETA-P-I, XI-Q-I, ETA-Q-I, XI-P-J, ETA-P-J

(e) ENTER BODY NO. FOR NO. k GROUNDED BODY

(f) FOR SIMPLE CONSTRAINT NO. k ENTER BODY NO.
  AND 1, 2, OR 3 FOR X, Y, OR PHI CONSTRAINT DIRECTION

(g) FOR DRIVER NO. k ENTER BODY NO.
  1, 2, OR 3 FOR X, Y, OR PHI
  INITIAL POSITION, VELOCITY AND ACCELERATION

(h) FOR POINT OF INTEREST NO. k ENTER BODY NO.
  XI-P AND ETA-P COORDINATES

(i) ENTER STARTING TIME, FINAL TIME, AND TIME INCREMENT

5.2 SIMPLE EXAMPLES

In the following sections, several simple examples are presented. The steps needed to set up a model for each mechanism are explained. Input data for the kinematic analysis program are listed for each example. Similar steps can be followed to analyze many other mechanisms using this program.

5.2.1 Four-Bar Linkage

The four-bar linkage shown in Fig. 5.2(a) is considered for kinematic analysis. This mechanism consists of four bodies, including ground; four revolute joints; one driver; and one point of interest (point P). The body numbers and their corresponding coordinate systems and the revolute joint numbers are shown in Fig. 5.2(b).

Initial estimates for the coordinates are as follows:

$$\begin{array}{lll} x_1 = 0.0, & y_1 = 0.0, & \phi_1 = 0.0 \\ x_2 = 0.5, & y_2 = 0.8, & \phi_2 = 60.0^\circ \\ x_3 = 2.6, & y_3 = 2.6, & \phi_3 = 30.0^\circ \\ x_4 = 3.5, & y_4 = 1.8, & \phi_4 = 60.0^\circ \end{array}$$

The local coordinates for the revolute joints are

$$\begin{array}{llll} \xi_1^A = 0.0, & \eta_1^A = 0.0, & \xi_2^A = -1.0, & \eta_2^A = 0.0 \\ \xi_2^B = 1.0, & \eta_2^B = 0.0, & \xi_3^B = -2.0, & \eta_3^B = 0.0 \\ \xi_3^C = 2.0, & \eta_3^C = 0.0, & \xi_4^C = 2.0, & \eta_4^C = 0.0 \\ \xi_4^D = -2.0, & \eta_4^D = 0.0, & \xi_1^D = 2.5, & \eta_1^D = 0.0 \end{array}$$

Body 2 is the driver link, and its corresponding driving variable is ϕ_2, with $\phi_2^0 = 1.0472$ rad (60°) and a constant angular velocity of $\omega = 6.2832$ rad/s. For one complete revolution, one second of simulation time is required. A time increment of 0.025 s results in increments of 9° in ϕ_2.

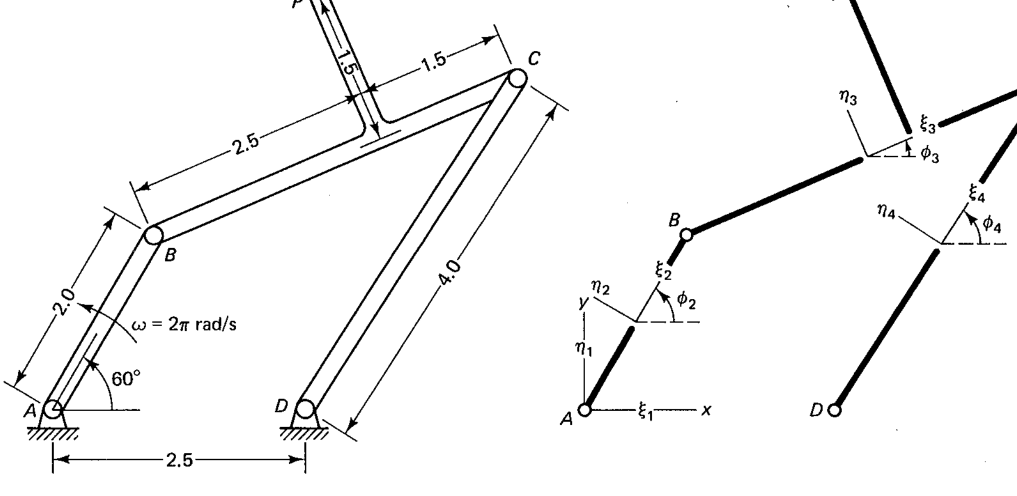


Figure 5.2 (a) A four-bar linkage, and (b) the corresponding joint numbers and body-fixed coordinate systems.

The point of interest (point P) is located on body 3. The local coordinates of this point are

$$\xi_3^P = 0.5, \qquad \eta_3^P = 1.5$$

When KAP is executed, the following sequence of prompts is given by the program (refer to Sec. 5.1.4), followed by inputs from the user to describe the model:

Prompt	k	Input
(a)		4,4,0,1,0,1,1
(b)	1	0.0,0.0,0.0
(b)	2	0.5,0.8,1.047
(b)	3	2.6,2.6,0.5
(b)	4	3.5,1.8,1
(c)	1	1,2,0.0,0.0,−1.0,0.0
(c)	2	2,3,1.0,0.0,−2.0,0.0
(c)	3	3,4,2.0,0.0,2.0,0.0
(c)	4	4,1,−2.0,0.0,2.5,0.0
(e)	1	1
(g)	1	2,3,1.0472,6.2832,0.0
(h)	1	3,0.5,1.5
(i)		0.0,1.0,0.025

Note that all of the angles are given in radians. The coordinates are given as estimates, except for ϕ_2 (since this is the independent variable) and the coordinates of the nonmoving body 1.

The output of this simulation for one second, with increments of 0.025 s, provides 41 time steps (including $t = 0$ s). A portion of the output for the first two time steps is as follows:

```
          *****  KINEMATIC ANALYSIS  *****


TIME =      .0000
----------------
 BODY     X       Y      PHI     XD      YD     PHID     XDD     YDD    PHIDD
  1     .000    .000    .000    .000   .000    .000     .000    .000    .000
  2     .500    .866   1.047  -5.441  3.142   6.283  -19.739 -34.190    .000
  3    2.824   2.553    .423 -11.085  6.732    .246  -52.441 -39.898  15.646
  4    3.574   1.687   1.004  -5.644  3.590   3.344  -32.702  -5.709  12.264
 POINTS OF INTEREST
  NO.     X       Y      XD      YD      XDD      YDD
  1    2.663   4.126 -11.472   6.692  -77.042  -42.500
```

```
TIME =      .0250
----------------
BODY     X        Y        PHI      XD       YD       PHID      XDD       YDD      PHIDD
 1     .000     .000     .000     .000     .000     .000      .000      .000      .000
 2     .358     .934    1.204   -5.866    2.252    6.283   -14.148   -36.856      .000
 3    2.531    2.708     .434  -12.220    5.558     .581   -38.613   -53.046    11.545
 4    3.423    1.774    1.091   -6.354    3.306    3.581   -24.465   -16.189     7.116
POINTS OF INTEREST
 NO.     X        Y        XD       YD       XDD       YDD
 1    2.355    4.279  -13.133    5.455  -56.693   -55.617
```

It can be observed that at $t = 0$, the program corrects the initial estimates on the coordinates. The value of ϕ_2 is kept constant according to the value given with the driving constraint. Also, the coordinates of body 1 remain unchanged, since body 1 is the ground. At each time step, the body numbers and the corresponding coordinates, velocities, and accelerations of their points of interest are reported. Figure 5.3 shows the path taken by point P for one revolution of the crank.

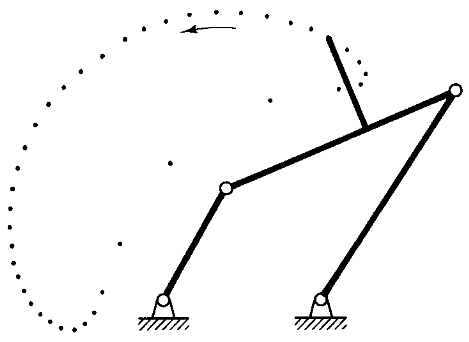


Figure 5.3 Path covered by point of interest P.

5.2.2 Slider-Crank Mechanism

The slider-crank mechanism of Sec. 4.4.1 is now considered for kinematic analysis. The mechanism is modeled in two slightly different forms.

Model 1. The first model considers the mechanism as discussed in Sec. 4.4.1. There are four bodies, three revolute joints, and one translational joint in this model. The flow of input data that describe this model is as follows:

Prompt	k	Input
ⓐ		4,3,1,1,0,1,0
ⓑ	1	0.0,0.0,0.0
ⓑ	2	−86.6,50.0,5.76
ⓑ	3	−467.0,40.0,0.2
ⓑ	4	−663.1,0.0,0.0
ⓒ	1	4,3,0.0,0.0,−200.0,0.0
ⓒ	2	3,2,300.0,0.0,−100.0,0.0

ⓒ	3	2,1,100.0,0.0,0.0,0.0
ⓓ	1	4,1,0.0,0.0,100.0,0.0,0.0,0.0
ⓔ	1	1
ⓖ	1	2,3,5.76,−1.2,0.0
ⓘ		0.0,5.3,0.1

Figure 5.4 shows acceleration of the slider for one complete revolution of the crank.

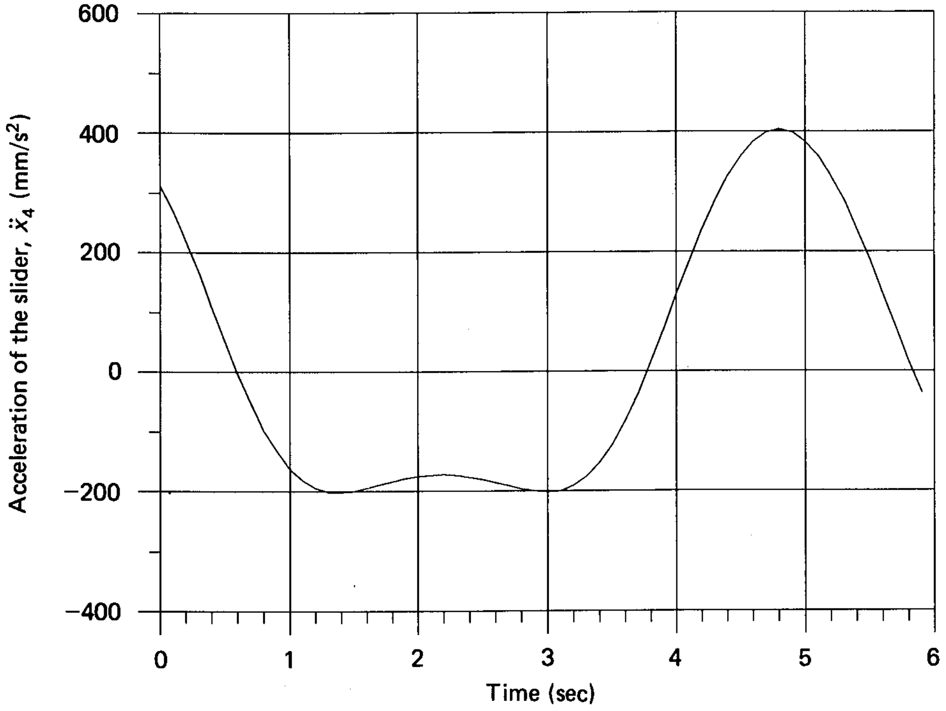


Figure 5.4 Acceleration of the slider versus time.

Model 2. The translational joint of the first model is replaced in model 2 by two simple constraints. The slider, body 4, is constrained in the y and ϕ directions. The flow of input data that describe this model is as follows:

Prompt	k	Input
ⓐ		4,3,0,1,2,1,0
ⓑ	1	0.0,0.0,0.0

ⓑ	2	−86.6,50.0,5.76
ⓑ	3	−467.0,40.0,0.2
ⓑ	4	−663.1,0.0,0.0
ⓒ	1	4,3,0.0,0.0,−200.0,0.0
ⓒ	2	3,2,300.0,0.0,−100.0,0.0
ⓒ	3	2,1,100.0,0.0,0.0,0.0
ⓔ	1	1
ⓕ	1	4,2
ⓕ	2	4,3
ⓖ	1	2,3,5.76,−1.2,0.0
ⓘ		0.0,5.3,0.1

5.2.3 Quick-Return Mechanism

Consider the quick-return mechanism of Sec. 4.4.2. Model 1 consists of six bodies, five revolute joints, and two translational joints. The flow of input data for this model is as follows:

Prompt	k	Input
ⓐ		6,5,2,1,0,1,0
ⓑ	1	0.0,−300.0,0.0
ⓑ	2	70.0,−60.0,1.1
ⓑ	3	50.0,40.0,0.5
ⓑ	4	90.0,60.0,1.1
ⓑ	5	80.0,180.0,6.0
ⓑ	6	0.0,200.0,0.0
ⓒ	1	3,4,50.0,0.0,0.0,0.0
ⓒ	2	2,5,250.0,0.0,60.0,0.0
ⓒ	3	1,2,0.0,0.0,−250.0,0.0
ⓒ	4	5,6,−60.0,0.0,0.0,0.0
ⓒ	5	1,3,0.0,300.0,−50.0,0.0
ⓓ	1	4,2,0.0,0.0,−10.0,0.0,0.0,0.0,
ⓓ	2	6,1,0.0,0.0,−10.0,0.0, 0.0,500.0
ⓔ	1	1
ⓖ	1	3,3,0.52,3.0,0.0
ⓘ		0.0,2.1,0.025

The velocity of body 6 (the slider) is plotted from the output in Fig. 5.5.

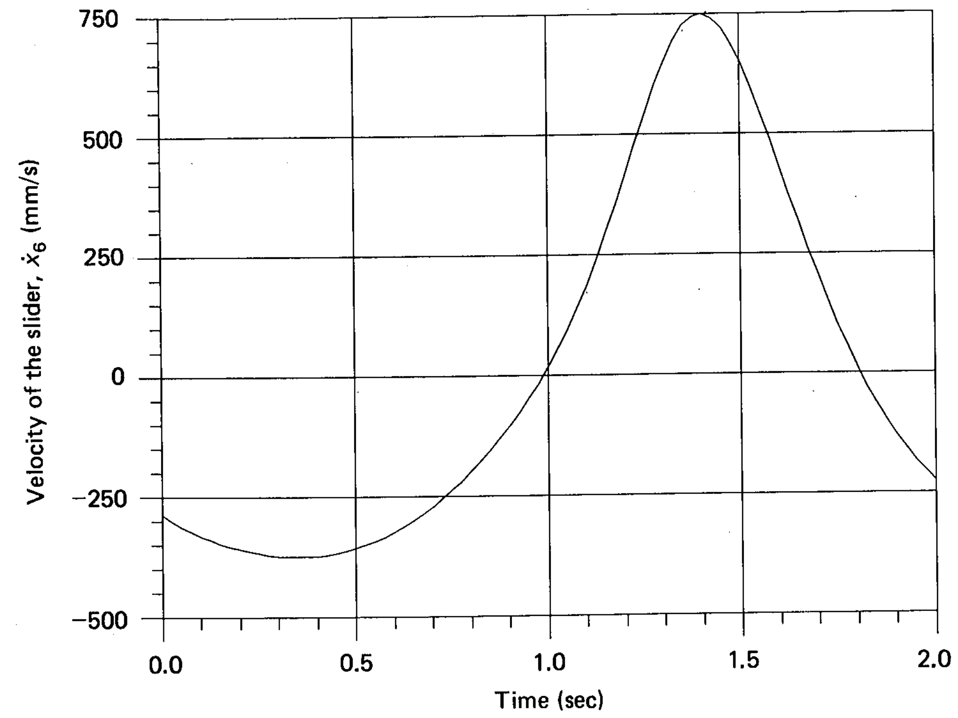


Figure 5.5 Slider velocity of a quick-return mechanism versus time.

5.3 PROGRAM EXPANSION

The computer program that is listed in the opening of Sec. 5.1 and in subsections 5.1.1 through 5.1.3 has been presented in its simplest form. This program may be expanded to accommodate, for example, other forms of input/output or other types of kinematic joints. These improvements to KAP are suggested at the end of this chapter, through a series of exercises (problems). These suggestions are only guidelines—the modifications can also take other forms. Actual use of the program, for modeling and analyzing a wide range of examples, will provide the user with additional ideas for expansion and modification.

PROBLEMS

The following problems provide examples that can be simulated by using a kinematic analysis program such as KAP. Many of the problems can be simulated on the existing listed version of KAP. Other problems may require some modifications or extensions to the program. Guidelines for improving the versatility and increasing the capability of KAP are included for some of those problems. The majority of the problems involve the analysis of four-bar mechanisms. The four-bar mechanism is one of the simplest closed-loop mechanisms, since it has only 1 DOF. A wide variety of motion can be generated by this mechanism. In the following four-bar mechanisms, a, b, c, and d denote, respectively, the length of the crank, coupler, follower, and frame. The four-bar mechanisms are classified under various types. The following nomenclature is used to describe each type of four-bar mechanism:[17]

$$s = \text{length of shortest link}$$
$$l = \text{length of longest link}$$
$$m, n = \text{lengths of the other two links}$$

5.1 A double-crank four-bar mechanism is shown in Fig. P.5.1. For this class, $s + l < m + n$ and the shortest link is the frame. Assume $a = 2.1$, $b = 1.6$, $c = 2.6$, and $d = 0.9$ (any unit). Drive link AB through 360° with a constant angular velocity. Monitor the path, velocity, and acceleration of points B and C.

5.2 A rocker-crank four-bar mechanism is shown in Fig. P.5.2. For this class, $s + l < m + n$ and the shortest link is one of the side links. Assume that $a = 0.8$, $b = 2.1$, $c = 1.6$, and $d = 2.5$. Drive link AB through 360° and find the forbidden zones for the rocker CD and the angle of oscillation α.

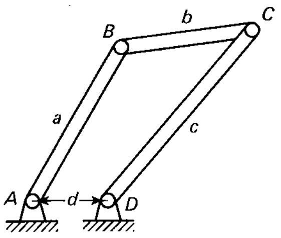


Figure P.5.1

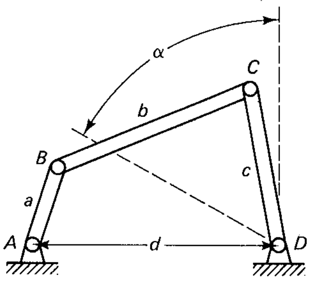


Figure P.5.2

5.3 A double-rocker four-bar mechanism is shown in Fig. P.5.3. For this class, $s + l < m + n$ and the shortest link is the coupler. Assume that $a = 2.1$, $b = 0.8$, $c = 2.5$, and $d = 1.6$. Drive the coupler BC through one revolution. Determine the forbidden zones and the range of the angles of oscillation for the two rockers AB and CD.

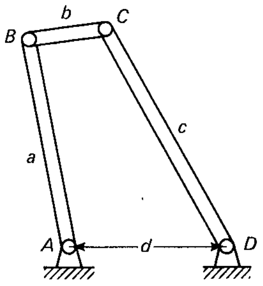


Figure P.5.3

5.4 Repeat Prob. 5.3 but instead of driving the coupler *BC*, drive one of the rockers:

(a) Drive the rocker *AB* counterclockwise.

(b) Drive the rocker *AB* clockwise.

(c) Determine the forbidden zones and the range of the angle of oscillation, and then compare the result with the result of Prob. 5.3.

5.5 Figure P.5.5 shows a change-point four-bar mechanism. For this class, $s + l = m + n$. Assume that $a = 1.2$, $b = 1.4$, $c = 1.6$, and $d = 1.0$. Drive link *AB* through 720°. Observe that after one revolution of *AB*, point *C* finds a new position at *C'*. After the second revolution of *AB*, point *C* returns to its initial position. Plot the rotational acceleration of *CD* versus its rotational angle.

5.6 The Galloway mechanism (Fig. P.5.6) is another example of a change-point four-bar mechanism. In this mechanism, $a = d$, $b = c$, and $a < b$. Assume that $a = 1.0$ and $b = 1.5$. Drive *AB* through 720° and observe that *CD* executes only 360°.

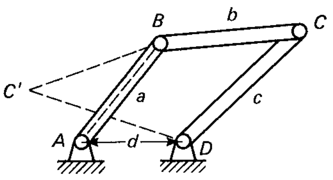


Figure P.5.5

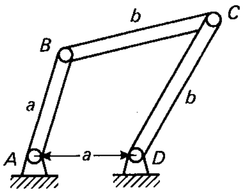


Figure P.5.6

5.7 Figure P.5.7 shows another example of a change-point four-bar mechanism. Configuration (a) is usually called *parallelogram linkage* and configuration (c) is called *antiparallel linkage*. Configuration (b) is the change-point state. A mechanism starting in configuration (a) and going through configuration (b) may end up in configuration (a) or (c). During position analysis, if the mechanism is near the change-point state, the Newton-Raphson iteration may converge to any of the two solutions. Drive *AB* through 360° and monitor the position of point *C*. Repeat for different values of time increment DT.

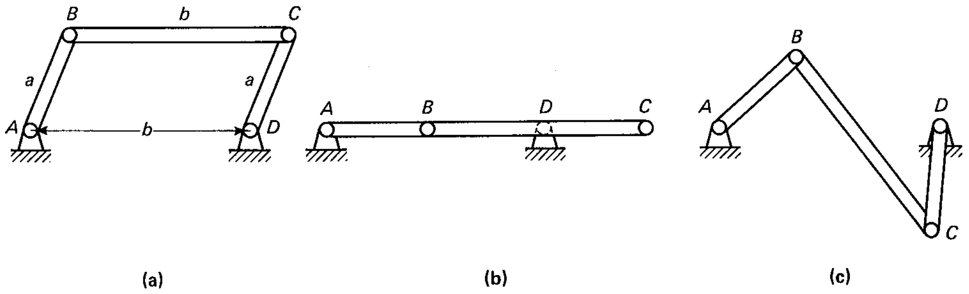


Figure P.5.7

Note: A mechanism at a change-point state becomes kinematically indeterminate. During numerical analysis, if the constraint equations are assembled in the exact change-point state, the Jacobian matrix loses rank; i.e., the number of degrees of freedom increases at the change-point state. Since the Jacobian matrix becomes singular, a solution cannot be obtained at such a state.

5.8 It would be useful to save the input data in a file, in the same sequence in which it is entered interactively. This input data file could then be used if further simulation of the same problem were needed.

- **(a)** Include any required statements in the program to save a copy of the interactively entered data in a disk file. The name of the file should be specified by the user.
- **(b)** The program should be able to accept data either interactively or from a file.
- **(c)** Include a WRITE statement after each READ statement to echo a copy of the input data to the output unit.

5.9 Provide the capability for the program to save the output response at every time step in an output file.

5.10 If the output is written into the terminal, the flow of data may be fast. In subroutine KINEM, after reporting the response and before incrementing T, include a WRITE and a READ statement making the program pause. The write statement should prompt the user to enter a C for continue or a T to terminate.

5.11 Modify the program to accept data on angles either in radians or in degrees. If the data are given in degrees, then the program must convert them to radians. Similarly, the program should provide the output in either radians or degrees.

5.12 For some mechanisms, the user may be interested in a complete kinematic analysis. However, sometimes position analysis or position and velocity analysis may be sufficient. It would be more efficient to avoid unwanted levels of analysis. Modify the program by introducing a flag as follows:

IANAL = 1: Perform position analysis.
IANAL = 2: Perform position and velocity analysis.
IANAL = 3: Perform position, velocity, and acceleration analysis.

5.13 Formulate additional kinematic joints in the program. Follow the form of the subroutines for the revolute joint that already exists in KAP. Include the following joints:

- **(a)** Revolute-revolute joint (Eq. 4.14.)
- **(b)** Revolute-translational joint (Eq. 4.16)
- **(c)** Spur gears (Eq. 4.18)

5.14 Include other types of driver constraints in the program. Some useful constraints are:

- **(a)** $\phi_i - \phi_j - c_1(t) = 0$
- **(b)** $x_i - x_j - c_2(t) = 0$
- **(c)** $y_i - y_j - c_3(t) = 0$
- **(d)** Eq. 4.39
- **(e)** Eq. 4.40

The time-dependent functions can be provided either as a table of data or as a closed-form function such as a sine or a cosine function.

5.15 In position analysis, the Newton-Raphson algorithm uses values of the coordinates from the previous time step as an estimate on $\mathbf{q}^i$ to start the iteration. These estimates can be improved as follows:

$$\mathbf{q}^i \simeq \mathbf{q}^{i-1} + \Delta t\dot{\mathbf{q}}^{i-1} + \tfrac{1}{2}\Delta t^2\ddot{\mathbf{q}}^{i-1}$$

Include this modification in subroutine KINEM to improve the efficiency of the program.

5.16 Employ a Gaussian elimination (or L-U factorization) algorithm with full or partial (row interchange) pivoting in KAP to identify any redundant constraint equation. This process can be performed once on the Jacobian matrix before the start of the kinematic analysis.

5.17 Simple but general algorithms and program listings for spline functions can be found in most numerical computation textbooks. Include such a program in KAP. These subroutines may be called for interpolation when a curve is defined in discrete form.

5.18 When nonstandard constraint equations are to be included in KAP to model a particular mechanism, it is not efficient to modify and compile the entire program. In this case a user-supplied subroutine can be written to implement the constraints. In subroutine FUNCT, include a call to a subroutine USRCON before the return statement, as follows:

```
CALL USRCON (A, IA, MAXA, MAXIA)
```

Then, include a dummy user-supplied subroutine:

```
SUBROUTINE USRCON (A, IA, MAX, MAXIA)
C.....Include all of the common blocks
DIMENSION A(MAXA), IA(MAXIA)
C.....Insert new constraints, define the Jacobian, etc.
RETURN
END
```

This subroutine will be called by subroutine FUNCT at every time step. If there is no nonstandard function in the model, this subroutine will not affect the simulation. However, if equations are inserted in this subroutine and it is linked to KAP, then each time this subroutine is called, the nonstandard constraints will be included in the model.

5.19 When constraint equations are formulated in terms of a set of Cartesian coordinates, the resultant Jacobian is a sparse matrix; i.e., most of the elements of the matrix are zero. There are matrix factorization algorithms designed specifically for sparse matrices:† These algorithms are much more efficient than the standard full-matrix algorithms, such as subroutine LU used in KAP. If you have access to such sparse-matrix algorithms, employ them in KAP to improve the efficiency of the program.

5.20 The linkage shown in Fig. P.5.20 is used to advance a film strip intermittently by the motion of point P, which periodically engages and disengages from the sprocket holes in the filmstrip as crank CD rotates clockwise at 1800 rpm.[17]

(a) Plot the path of point P.

(b) How far apart should the sprocket holes be placed on the film strip?

(c) What is the mean speed of the film strip?

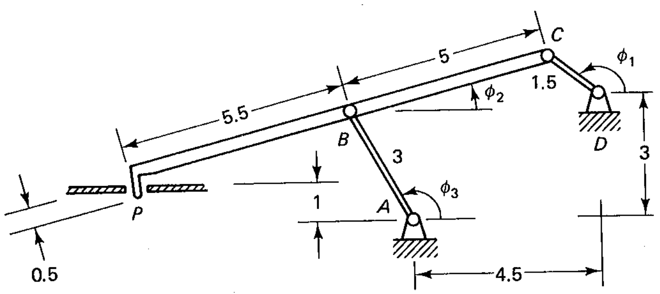


Figure P.5.20

5.21 Assign values to the lengths of the links of the four-bar mechanism shown in Fig. P.5.21. Rotate the crank and monitor the path of several points on the coupler, such as $P_1, P_2, \ldots$.

†The HARWELL Subroutine Library offers FORTRAN subroutines for sparse-matrix factorization. For more detail, write to HARWELL Subroutine Library, United Kingdom Atomic Energy Authority, Computer Science and System Division, AERE Harwell, Oxfordshire.

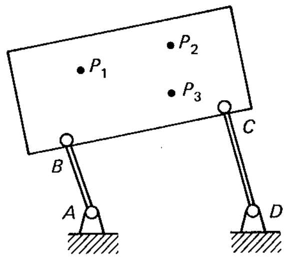


Figure P.5.21

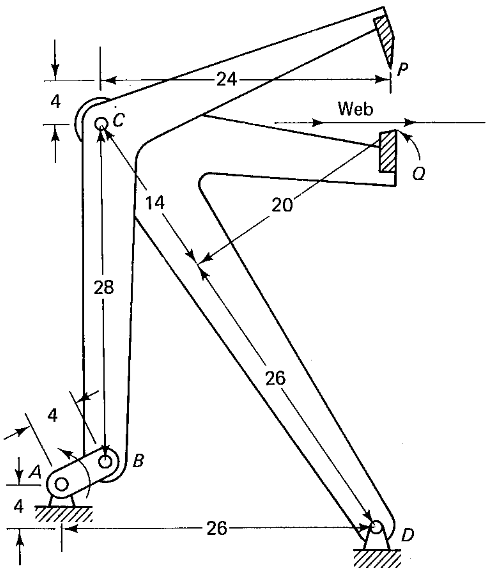


Figure P.5.22

5.22 The crank-rocker four-bar mechanism shown in Fig. P.5.22 is a web cutter.[17] Link AB rotates with a constant angular velocity of 100 rpm.

(a) Determine the angle of the crank at the time of cutting.

(b) What should be the velocity of the web at the time of cutting?

5.23 Figure P.5.23 shows an elliptic trammel. Any point P on the link KL traces out an ellipse. Define several points $P_1, P_2, \ldots$ as special points of interest on link KL, and then rotate this link through 360° and plot the paths of these points.

5.24 A dough-kneader mechanism is shown in Fig. P.5.24.[‡] The crank AB rotates through 360°. Note that in order to model this mechanism two revolute joints are needed at B. Plot the path of point P. Assume $AB = 6$, $BC = EF = 13$, $BE = CF = 6$, $DC = 15$, $AD = 18$, $BP = 26$.

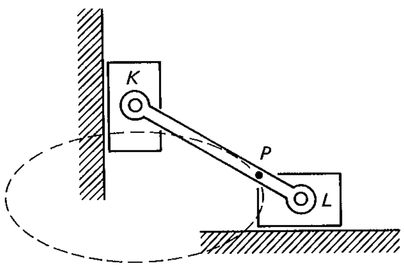


Figure P.5.23

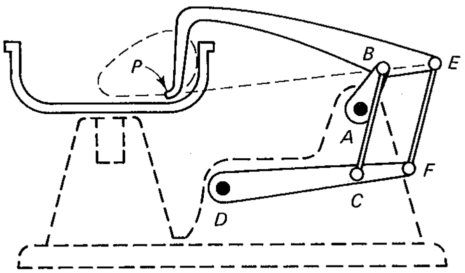


Figure P.5.24

[‡]These mechanisms have been adopted for kinematic simulation from the following reference: Artobolevsky, I. I., *Mechanisms in Modern Engineering Design*, Vol. I, MIR Publishers, Moscow, 1975. Many other simple to complicated mechanisms can be found in this and subsequent volumes of the same reference.

5.25 Figure P.5.25 shows a translator mechanism for a drafting table.[‡] The lengths of the links comply with the conditions $AB = DC$, $AD = BC$, $FG = EH$, and $EF = HG$. Verify that the system has 2 DOF. Also show that parallel lines can be drawn by link p. This can be done in several ways; for example, keep the angle between EF and EH constant (include the constraint $\phi_i - \phi_j - c = 0$), and then rotate link AB.

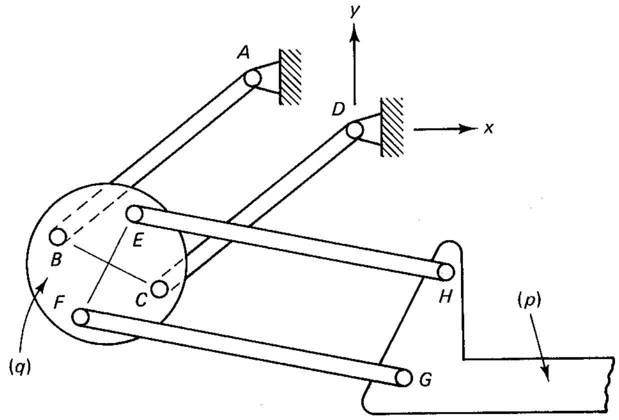


Figure P.5.25

5.26 Repeat Prob. 5.25 and assume that ring q consists of two rings q_1 and q_2 as shown in Fig. P.5.26. The two rings are connected by revolute joint I. Angle α between the two rings can be varied in order to change the angle of link p. While keeping α constant during any given simulation, perform several simulations for different values of α.

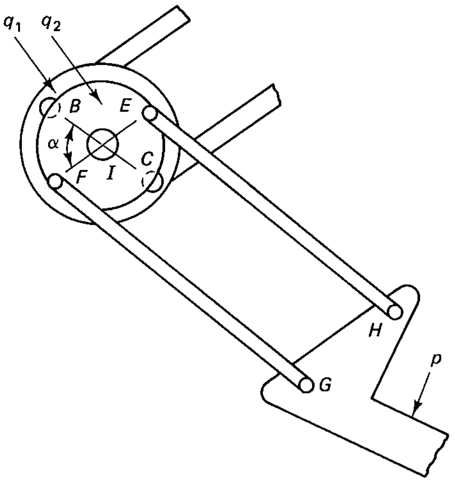


Figure P.5.26

5.27 Figure P.5.27 shows two pantograph mechanisms where T is the tracing point and D is the drawing point. Since O, D, and T are on a straight line, the paths traced by T and D are

[‡]See footnote to Prob. 5.24.

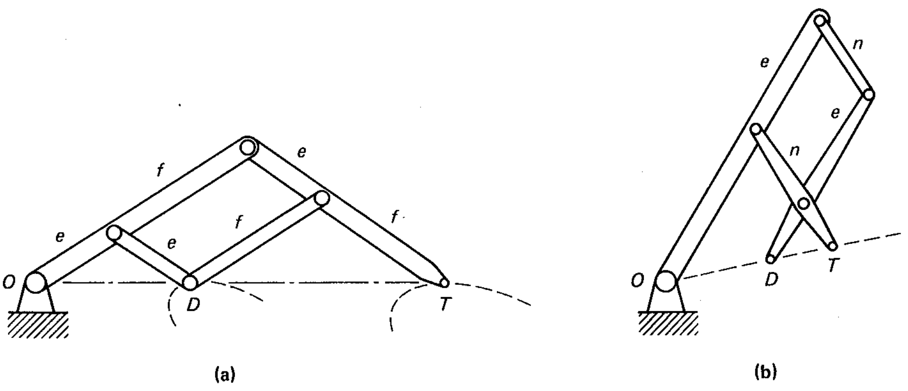


Figure P.5.27

geometrically similar. In order to set up a simulation model, the following tasks must be performed:

(a) Carry out Prob. 5.12 for position analysis only.

(b) Carry out Prob. 5.14(d) and (e).

Select an outline to be traced; for example, a known geometrical shape. Discretize the outline into several points, and then find the x and y coordinates of these points with respect to the global xy coordinate system. These coordinates provide driver constraints on x^T and y^T.

5.28 A pantograph mechanism can be modified easily to draw nonsimilar or warped traces. For example if point D is moved to position D_1 (see Fig. P.5.28), the shape of the drawn path will be different from the traced path. Similarly, if D is moved to position D_2, then the drawn path will be quite different from the traced path. By moving D to different positions, completely unfamiliar shapes can be generated. If the result is displayed on a graphics terminal, interesting paths may be observed. Make this problem a computer graphics oriented project.

5.29 The mechanism shown in Fig. P.5.29 is known as a Kostitsyn approximate straight-line system, where $b = 4a$, $c = 5a$, and $AD = b$.[‡] When link BF rotates, point E describes a path

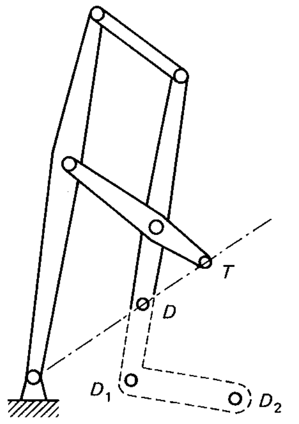


Figure P.5.28

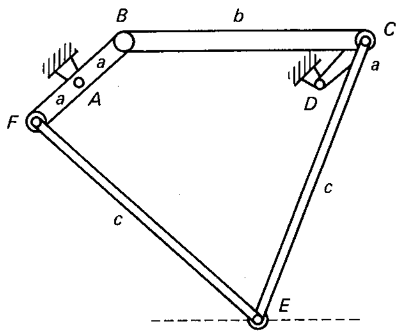


Figure P.5.29

[‡]See footnote to Prob. 5.24.

of which a portion is an approximate straight line. Verify this and note that the linkage *ABCD* is a change-point four-bar mechanism.

5.30 Figure P.5.30 shows the Peaucellier-Lipkin exact straight-line mechanism.[‡] Choose $e = 1.2, f = 3.0$, and $g = 1.4$. Rotate link *RE* and observe that point *S* describes a straight line. Determine the length of the straight line drawn by point *S*.

5.31 The Peaucellier-Lipkin circle inversion mechanism is shown in Fig. P.5.31.[‡] Assume that $a = 2.0$, $b = 1.6$, and $c = 1.4$, where $FE < c$. Rotate link *ER* and observe that point *S* describes a circle *c* centered at point *O*. Verify that $FO = FE(a^2 - b^2)/(c^2 - FE^2)$ and that the radius of the circle is $r = c(FO/FE)$ (for simulation assume $FE = 0.8$).

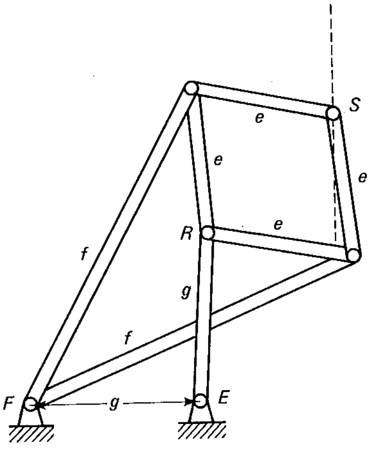


Figure P.5.30

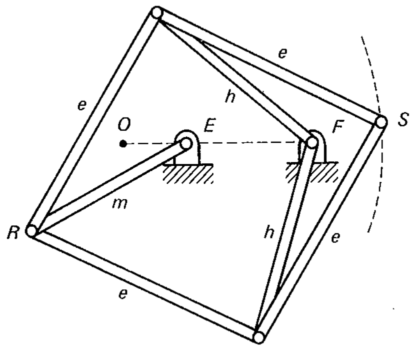


Figure P.5.31

5.32 The Hart mechanism for drawing ellipses is shown in Figure P.5.32. The lengths of the links comply with the conditions $n = 2m$ and $e = \sqrt{2}m$. Verify that when link *MR* rotates, point *S* describes a straight line and any intermediate point *P* on link *RS* describes an ellipse.

5.33 The Chebyshev six-bar reversing mechanism is shown in Fig. P.5.33. Verify that one revolution of crank *EF* corresponds to one revolution of link *JK* in the opposite direction. Assume $EF = 1.08$, $FG = GH = GI = 2$, $IJ = JK = 1.14$, $HK = 2.78$, and $EH = 2.66$.

5.34 The mechanism shown in Fig. P.5.34 is a simplified version of a thread-and-needle guiding system of a sewing machine.[‡] Rotate crank *c* and plot the path of point *H*. Assume $AB = 10$, $BE = 9$, $DE = DF = 6.5$, $EF = 5$, $FG = 17.5$, $a = f = 10$, $b = 7$, $d = 1$, and $e = 4$.

5.35 For the mechanism shown in Fig. P.5.35, verify that when the crank rotates, the output link dwells. Plot the angular velocity of the output link versus the crank angle. Assume that $OA = 0.8$, $AB = 3.6$, $BC = 2.3$, $CH = DE = EF = 3.5$, $OH = 1.5$, $AD = CG = 4$, $BD = 2$, and $FG = 1$.

5.36 For the mechanism shown in Fig. P.5.36, find that portion of the crank orientation for which the output link dwells. Plot the path of the motion of the slider. Assume $OA = 1$, $AB = 4$, $BC = OE = 3.5$, $CE = 5$, and $BD = OF = 2.5$.

5.37 A Geneva-wheel mechanism is shown in Fig. P.5.37. For a specific design, the following information is known: the radius of the crank ρ; the number of slots n; the half angle between two adjacent slots $\beta = \pi/n$; the center distance $d = \rho/\sin\beta$. It is also known that the crank axis is perpendicular to the slot axis at the moment of engagement or disengage-

[‡]See footnote to Prob. 5.24.

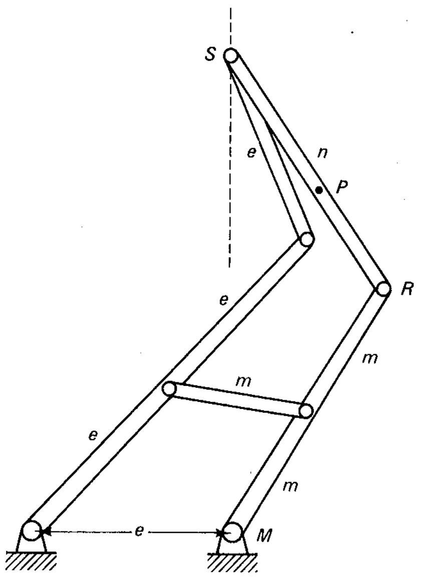


Figure P.5.32

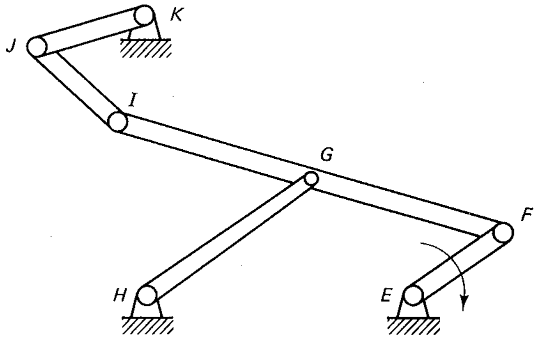


Figure P.5.33

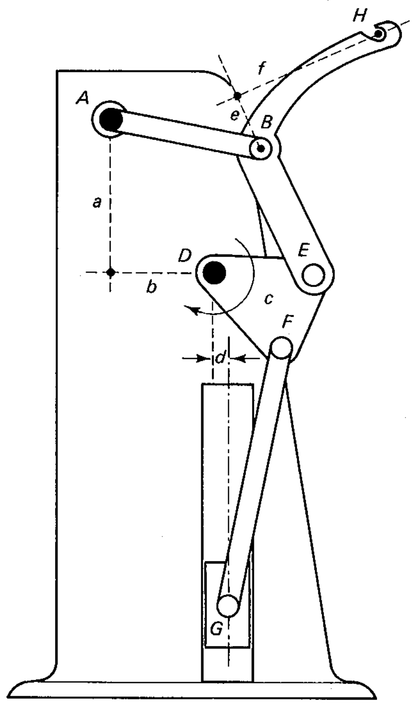


Figure P.5.34

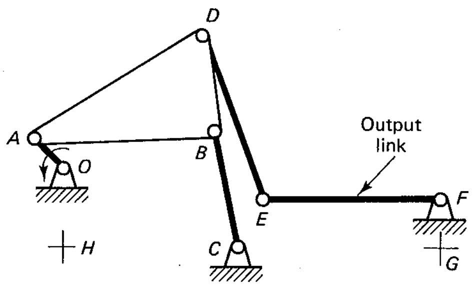


Figure P.5.35

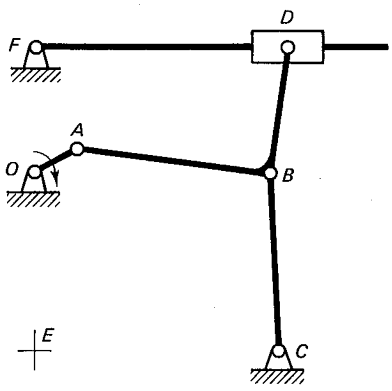


Figure P.5.36

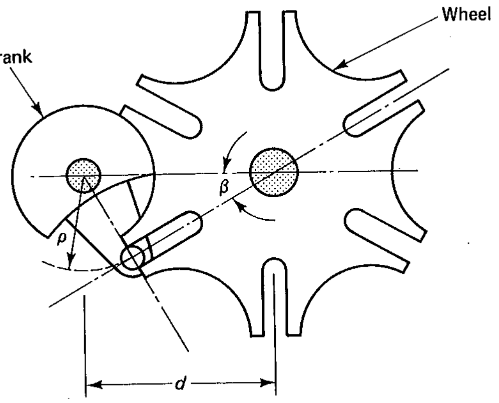


Figure P.5.37

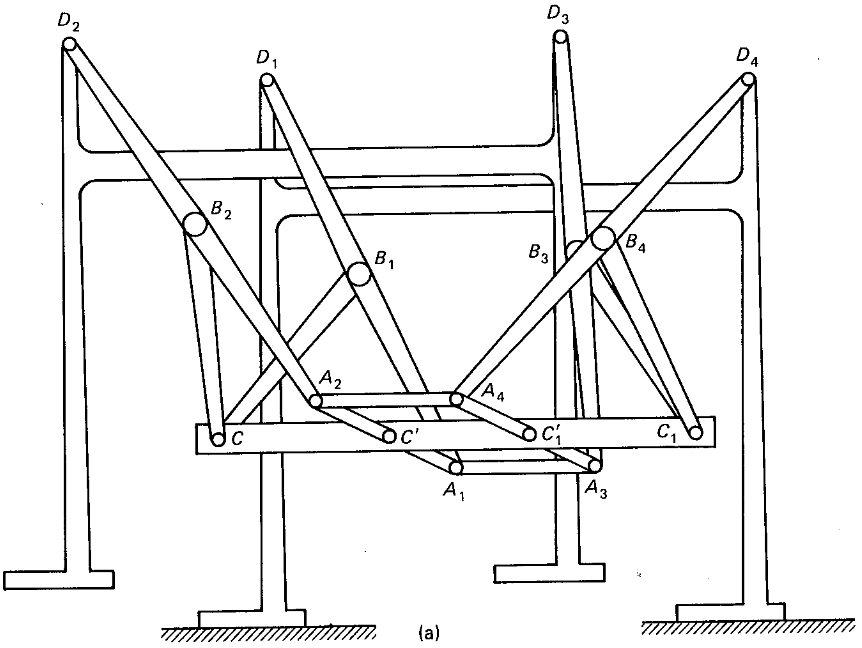
ment. In order to model this mechanism, two types of constraints must be employed interchangeably:

(a) When the pin is disengaged from the wheel, the wheel does not rotate.

(b) When the pin is engaged with a slot, the pin-slot combination can be modeled as a revolute-translational joint. It is clear that the axis of the translational joint will be different from one revolution to the next.

Include the constraint equations and the proper logic for switching between these equations in KAP.

5.38 A walking robot mechanism is shown in Fig. P.5.38(a).[‡] The mechanism consists of two identical Π-shaped legs, each having two feet. When one pair of feet is on the ground, the



(a)

[‡]See footnote to Prob. 5.24.

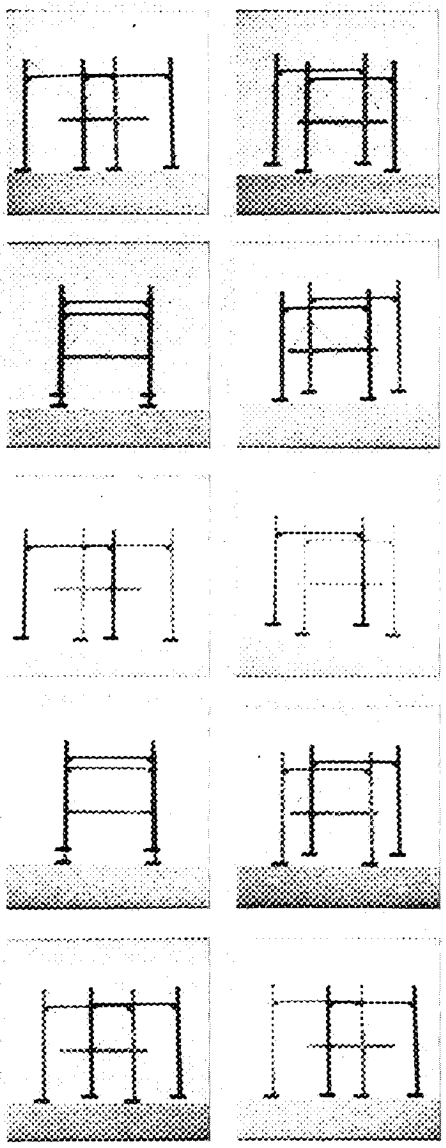

(b)

Figure P.5.38

other pair moves forward by the rotation of two parallel cranks $A_1C'A_2$ and $A_3C_1'A_4$. The lengths of the links comply with the conditions:

$$A_iB_i = B_iD_i = 1 \qquad i = 1, 2, 3, 4$$
$$B_1C = B_2C = B_3C_1 = B_4C_1 = 1$$
$$A_1C' = A_2C' = A_3C_1' = A_4C_1' = 0.355$$
$$A_2A_4 = A_1A_3 = C'C_1' = 0.634$$
$$CC' = C_1C_1' = 0.785$$

A kinematic analysis of this system requires the following:

(a) If all of the bodies and revolute joints that are shown in the figure are included in the model, redundant constraint equations will result. Therefore, the redundant equations must be eliminated.

(b) During one-half of the crank revolution, one pair of feet must be constrained to the ground, and during the other half of the crank revolution, the other pair of feet must be constrained to the ground. This process must be performed interchangeably in every revolution of the crank.

If the result of the simulation is viewed on a graphics terminal, a sequence of graphs, as shown in Fig. P.5.38(b), will be observed.

5.39 Some relatively complex mechanisms can become interesting simulation projects. For the more complex systems, graphics output can be a valuable analysis and design tool. Examples of such mechanisms can be found in the following:

(a) Recliner chairs

(b) Sofa-sleepers

(c) Crank mechanisms for automobile windows

(d) Convertible tops in some automobiles

Find data on these mechanisms and try to simulate their kinematics.