3

Basic Concepts and Numerical Methods in Kinematics

Kinematics, which is the study of the motion of rigid bodies, is useful in two important ways. First, it is frequently necessary to generate, transmit, or control motion by the use of cams, gears, and linkages. An analysis of the displacement, velocity and acceleration of the system is necessary to determine the design geometry of the mechanical parts. Furthermore, as a result of the generated motion, forces are frequently developed that must be accounted for in the design of parts. Second, it is often necessary to determine the motion of a system of rigid bodies that results from applied forces. In both types of problems, one must first have command of the principles of rigid-body kinematics.

Kinematics analysis requires, in general, solution of nonlinear algebraic equations. For small problems with only a few variables and a few equations, it might be possible to write and solve these equations by hand. However, for large problems with many variables and even for accurate analysis of smaller problems, hand calculation, if not impossible, is tedious and unlikely to succeed. Therefore, numerical methods and computer programs are the obvious choice for fast and accurate solution of kinematic equations.

This chapter presents some of the definitions used in kinematics. The general forms of the kinematic equations are presented. Although systematic methods of deriving these equations are not discussed until Chaps. 4, 5, 6, and 7, numerical methods for solving such equations are discussed in this chapter. Several efficient methods for solving linear algebraic equations and nonlinear algebraic equations are reviewed. Algorithms and listings of computer programs for some of these methods are also presented.

3.1 DEFINITIONS

A *rigid body* is defined as a system of particles for which distances between particles remain unchanged. If a particle on such a body is located by a position vector fixed to

the body, the vector never changes its position relative to the body, even when the body is in motion. In reality, all solid materials change shape to some extent when forces are applied to them. Nevertheless, if movement associated with the changes in shape is small compared with the overall movement of the body, then the concept of rigidity is acceptable. For example, displacements due to elastic vibration of the connecting rod of an engine may be of no consequence in the description of engine dynamics as a whole, so the rigid-body assumption is clearly in order. On the other hand, if the problem is one of describing stress in the connecting rod due to vibration, then the deformation of the connecting rod becomes of prime importance and cannot be neglected.

In this text, essentially all analysis is based on the assumption of rigidity. A *mechanism* is a set of rigid elements that are arranged to produce a specified motion. This definition of a mechanism includes classical linkages, as well as interconnected bodies that make up a vehicle, a vending machine, aircraft landing gear, an engine, and many other mechanical systems. While one can study the kinematics of a *deformable body* by defining the position of every point in the body in its deformed state, this introduces considerable complexity that is not needed in many applications. This text is concerned with *rigid* (nondeforming) *bodies*. The term *bodies*, therefore, will be used instead of *rigid bodies*.

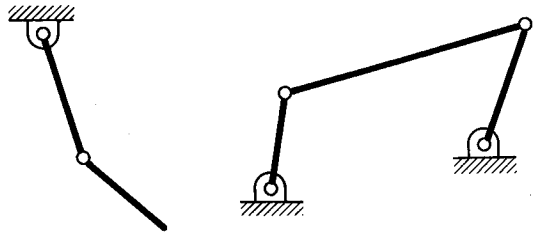
Kinematics is the study of motion, quite apart from the forces that produce the motion. More particularly, kinematics is the study of position, velocity, and acceleration in a system of bodies that make up a mechanism.

Kinematic synthesis is the process of finding the geometry of a mechanism that will yield a desired set of motion characteristics. *Kinematic analysis*, on the other hand, is the process of predicting position, velocity, and acceleration once a mechanism is specified. The processes of kinematic synthesis and kinematic analysis are normally intertwined. In order to do a synthesis, the engineer needs to be able to do an analysis to evaluate designs under consideration. Thus, kinematic analysis may be viewed as a tool that is needed to support the kinematic synthesis process.

The individual bodies that collectively form a mechanism are said to be *links*. The combination of two links in contact constitutes a *kinematic pair*, or *joint*. An assemblage of interconnected links is called a *kinematic chain*. A mechanism is formed when at least one of the links of the kinematic chain is held fixed and any of its other links can move. The fixed link(s) is (are) called the *ground* or *frame*.

If all the links of a mechanism move in a plane or in parallel planes, the mechanism is called a *planar mechanism*. If some links undergo motion in three-dimensional space, the mechanism is called a *spatial mechanism*.

A mechanism that is formed from a collection of links or bodies that are kinematically connected to one another but for which it is not possible to move to successive links across kinematic joints and return to the starting link is called an *open-loop* or *open-chain mechanism*. An open-loop mechanism may contain links with single joints. An example of this kind of mechanism is the double pendulum shown in Fig. 3.1(a). A *closed-loop mechanism* is formed from a closed chain, wherein each link is connected to at least two other links of the mechanism and it is possible to traverse a closed loop. Figure 3.1(b) shows a four-bar linkage, which is a closed-loop mechanism. Kinematic analysis considers systems containing only closed loops.



(a) (b)

Figure 3.1 (a) Open-loop mechanism—double pendulum, (b) Closed-loop mechanism—four-bar linkage.

A closed-loop mechanism may contain one or more loops (or closed paths) in its kinematic structure. If the number of loops in a closed-loop mechanism is 1, then the mechanism is called a *single-loop mechanism*. If the closed-loop mechanism contains more than one loop, then the mechanism is called a *multiloop mechanism*. Figure 3.2(a) is an example of a single-loop mechanism, and Fig. 3.2(b) shows a multiloop mechanism.

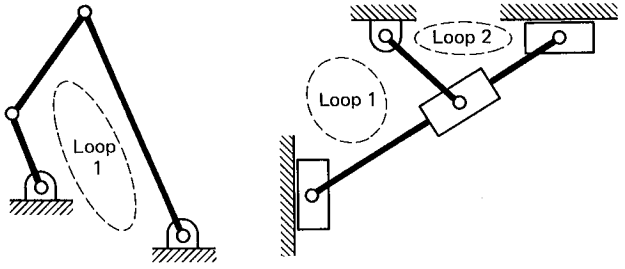
3.1.1 Classification of Kinematic Pairs

Mechanisms and kinematic pairs can be classified in a number of different ways. One method is purely descriptive; e.g., gear pairs, cams, sliding pairs, and so on. Such a division is convenient, and some of these pairs will be studied in Chaps. 4 and 7 under such headings. However, a broader view of the grouping of kinematic pairs is presented here.

Kinematic pairs may be classified generally into two groups.[17] Joints with surface contact are referred to as *lower pairs*, and those with point or line contact are referred to as *higher pairs*. Figure 3.3 gives a number of examples of kinematic pairs. The pairs (a), (b), (e), and (f) in Fig. 3.3 are examples of lower-pair joints, and pairs (c) and (d) are examples of higher-pair joints.

The constraint formulation for most lower-pair joints is generally simpler to derive than that for higher-pair joints.

Relative motion between two bodies of a kinematic pair may be planar or spatial. For example, pairs (a), (b), (c), and (d) in Fig. 3.3 display relative motion between bodies in a manner that can be considered either for planar or spatial kinematic analysis. In contrast, pairs (e) and (f) can be studied only in a spatial kinematic sense.



(a) (b)

Figure 3.2 (a) Single-loop mechanism, (b) Multiloop mechanism.

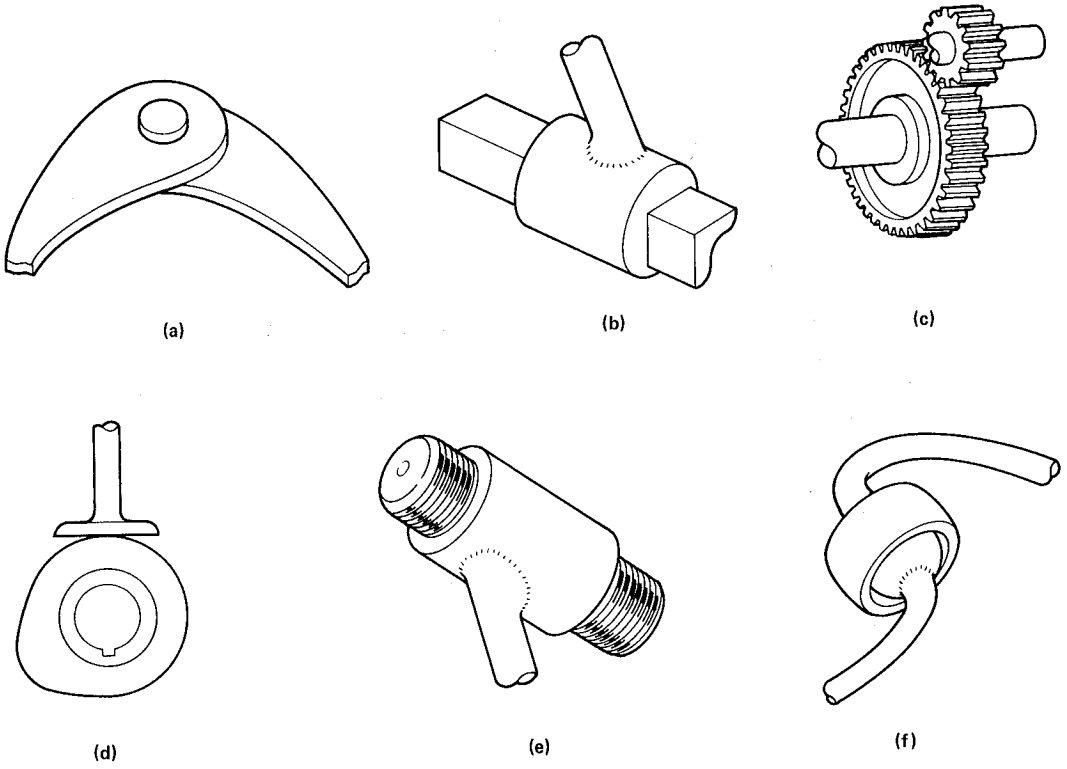


Figure 3.3 Examples of kinematic pairs: (a) revolute joint, (b) translational joint, (c) gear set, (d) cam follower, (e) screw joint, (f) spherical (ball) joint.

3.1.2 Vector of Coordinates

Any set of parameters that uniquely specifies the *position* (configuration) of all bodies of a mechanism is called a set of *coordinates*. For systems in motion, these parameters vary with time. The term *coordinates* can refer to any of the commonly used coordinate systems, but it can also refer to any of an infinite variety of other sets of parameters that serve to specify the configuration of a system. Vectors of coordinates are designated in this text by column vectors $\mathbf{q} \equiv [q_1, q_2, \ldots, q_n]^T$, where n is the total number of coordinates used in describing the system. Examples of commonly used coordinates are *Lagrangian* and *Cartesian* coordinates. In this text, *Cartesian* coordinates are used almost exclusively. The general distinction between the Lagrangian and Cartesian coordinate systems is that the former allows definition of the position of a body relative to a moving coordinate system, whereas the latter normally requires that the position of each body in space be defined relative to a fixed global coordinate system. Thus the Cartesian coordinate system requires that a large number of coordinates be defined to specify the position of each body of the system.

In order to specify the configuration of a planar system, a body-fixed $\xi\eta$ coordinate system is embedded in each body of the system, as shown in Fig. 3.4(a). Body i (i is an identifying number assigned to each body) can be located by specifying the global

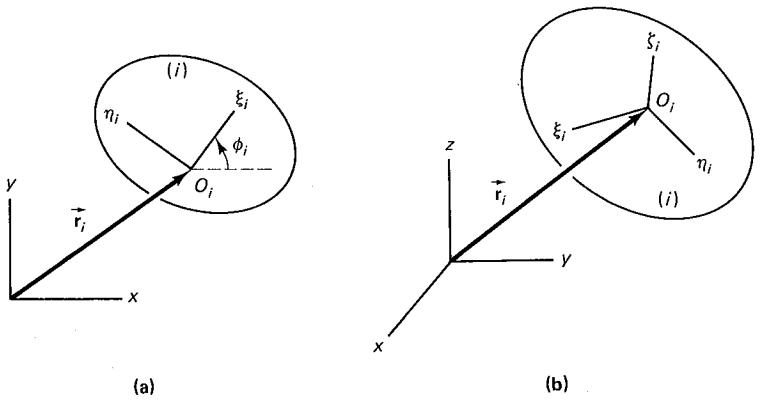


Figure 3.4 Global and body-fixed coordinate systems: (a) planar motion, (b) spatial motion.

translational coordinates $\mathbf{r}_i = [x, y]_i^T$ of the origin of the body-fixed $\xi_i\eta_i$ reference system and the angle ϕ_i of rotation of this system relative to the global xy axes. The column vector $\mathbf{q}_i \equiv [x, y, \phi]_i^T$ is the vector of coordinates for body i in a plane.†

For spatial systems, six coordinates are required to define the configuration of each body; e.g., body i shown in Fig. 3.4(b). The three components of the vector $\vec{r}_i$ — i.e., the global translational coordinates $\mathbf{r}_i \equiv [x, y, z]_i^T$ — locate the origin of the body-fixed $\xi_i\eta_i\zeta_i$ reference system relative to the global xyz axes, and the three rotational coordinates ϕ_{1i}, ϕ_{2i}, and ϕ_{3i} specify the angular orientation of the body. Therefore, column vector $\mathbf{q}_i \equiv [x, y, z, \phi_1, \phi_2, \phi_3]_i^T$ is the vector of coordinates for body i in three-dimensional space.

The concept of angular orientation of a body in three-dimensional space is discussed in detail in Chap. 6. It will be shown that instead of three rotational coordinates, four rotational coordinates with one equation relating these four coordinates can be used to avoid singularity problems. In this case, the coordinates for body i become $\mathbf{q}_i \equiv [x, y, z, e_0, e_1, e_2, e_3]_i^T$. The advantage of presenting the angular orientation of a body with four coordinates instead of three is also discussed in Chap. 6.

If a mechanism with b bodies is considered, the number of coordinates required is $n = 3 \times b$ if the system is planar, and $n = 6 \times b$ (or $7 \times b$) if the system is spatial. The overall vector of coordinates for the system is denoted by $\mathbf{q} = [\mathbf{q}_1^T, \mathbf{q}_2^T, \ldots, \mathbf{q}_b^T]^T$. Since bodies making up a mechanism are interconnected by joints, all of the coordinates are not independent — there are equations of constraint relating the coordinates.

Lagrangian coordinates, unlike Cartesian coordinates, do not necessarily assign the same number of coordinates to each body of the system. Some of the coordinates may be measured relative to a global coordinate system while others are measured relative to moving coordinate systems. An example of a set of Lagrangian coordinates is shown in Fig. 3.5. The variables ϕ_1, ϕ_2, and d define the configuration of the slider-crank mechanism at every instant. The vector of coordinates for the system can thus be

†For notational simplification, the body index is moved outside the bracket; e.g., $\mathbf{q}_i = [x, y, \phi]_i^T$ is used instead of $\mathbf{q}_i = [x_i, y_i, \phi_i]^T$.

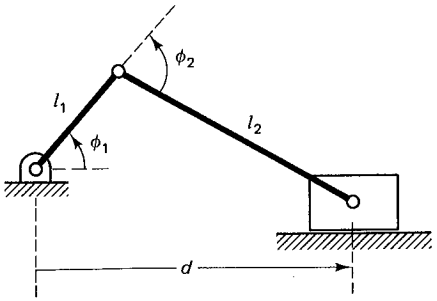


Figure 3.5 Slider-crank mechanism with Lagrangian coordinates.

defined as $\mathbf{q} \equiv [\phi_1, \phi_2, d]^T$. It must also be noted that these coordinates are not independent. If the lengths of links 1 and 2 are given as l_1 and l_2, once ϕ_1 is specified, ϕ_2 and d may be defined by simple trigonometry.

When a system is in motion, some or all of the coordinates describing the configuration of the system vary with time. In this text t denotes time and is considered to be an independent variable. In kinematics and dynamics, the motion of a body or a mechanism is analyzed for an interval of time from t^0 (initial time) to t^e (final time). In most problems, it is convenient to let $t^0 = 0$.

3.1.3 Degrees of Freedom

The minimum number of coordinates required to fully describe the configuration of a system is called the number of *degrees of freedom* (DOF) of the system. Consider the triple pendulum shown in Fig. 3.6. Here, no fewer than three angles, ϕ_1, ϕ_2, and ϕ_3, can uniquely determine the configuration of the system. Therefore, the triple pendulum has 3 degrees of freedom. Similarly, for the four-bar mechanism shown in Fig. 3.7, three variables, ϕ_1, ϕ_2, and ϕ_3, define the configuration of the system. However, the angles are not independent. There exist two algebraic constraint equations,

$$\begin{aligned} l_1 \cos \phi_1 + l_2 \cos \phi_2 - l_3 \cos \phi_3 - d_1 &= 0 \\ l_1 \sin \phi_1 + l_2 \sin \phi_2 - l_3 \sin \phi_3 - d_2 &= 0 \end{aligned} \tag{3.1}$$

which define loop closure of the mechanism. The two equations can be solved for ϕ_2 and ϕ_3 as a function of ϕ_1. Therefore, ϕ_1 is the only variable needed to define the configuration of the system, and so there is only 1 degree of freedom for the four-bar mechanism.

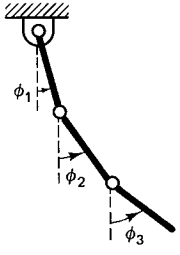


Figure 3.6
Triple pendulum.

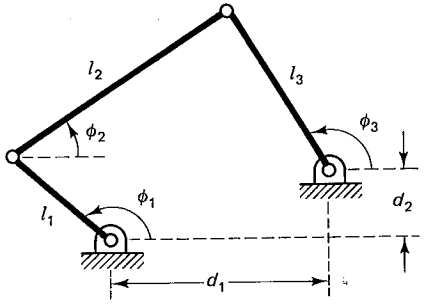


Figure 3.7 Four-bar mechanism.

In a mechanical system, if k is the number of degrees of freedom of the system, then k independent coordinates are required to completely describe the system. These k quantities need not all have the dimensions of length. Depending on the problem at hand, it may be convenient to choose some coordinates with dimensions of length and some that are dimensionless, such as angles or direction cosines. Any set of coordinates that are independent and are equal in number to the number of degrees of freedom of the system is called a set of *independent coordinates*. Any remaining coordinates, which may be determined as a function of the independent coordinates, are called *dependent coordinates*.

3.1.4 Constraint Equations

A kinematic pair imposes certain conditions on the relative motion between the two bodies it comprises. When these conditions are expressed in analytical form, they are called *equations of constraint*. Since in a kinematic pair the motion of one body fully or partially defines the motion of the other, it becomes clear that the number of degrees of freedom of a kinematic pair is less than the total number of degrees of freedom of two free rigid bodies. Therefore, a constraint is any condition that reduces the number of degrees of freedom in a system.

A constraint equation describing a condition on the vector of coordinates of a system can be expressed as follows:

$$\Phi \equiv \Phi(\mathbf{q}) = 0 \tag{3.2}$$

In some constraint equations, the variable time may appear explicitly:

$$\Phi \equiv \Phi(\mathbf{q}, t) = 0 \tag{3.3}$$

For example, Eq. 3.1 describes two constraint equations for the four-bar mechanism of Fig. 3.7, which has a vector of coordinates $\mathbf{q} = [\phi_1, \phi_2, \phi_3]^T$. These equations are of the form stated by Eq. 3.2.

Algebraic equality constraints in terms of the coordinates, and perhaps time, are said to be *holonomic* constraints. In general, if constraint equations contain *inequalities* or relations between velocity components that are *not integrable* in closed form, they are said to be *nonholonomic* constraints. In this text, for brevity, the term *constraint* will refer to a holonomic constraint, unless specified otherwise.

3.1.5 Redundant Constraints

A brief study of a mechanism is essential prior to actual kinematic or dynamic analysis. Knowledge of the number of degrees of freedom of the mechanism can be useful when constraint equations are being formulated. Often, the pictorial description of a mechanism can be misleading. Several joints may restrict the same degree of freedom and may therefore be equivalent or redundant.

As an example, consider the double parallel-crank mechanism shown in Fig. 3.8(a). This system has 1 degree of freedom. If this system is modeled for kinematic analysis as four moving bodies and six revolute joints, the set of constraint equations will contain redundant equations. The reason for redundancy becomes clear when one of the coupler links is removed to obtain the mechanism shown in Fig. 3.8(b). The two mechanisms are kinematically equivalent.

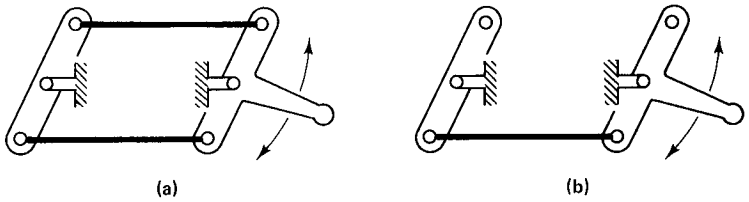


Figure 3.8 (a) A double parallel-crank mechanism, and (b) its kinematically equivalent system.

For a system having m *independent* constraint equations and n coordinates, the number of degrees of freedom is determined as follows:

$$k = n - m \tag{3.4}$$

In planar motion, a moving body can have three coordinates, and a revolute joint introduces two constraint equations. For the mechanism of Fig. 3.8(b), there are three moving bodies ($n = 3 \times 3 = 9$) and four revolute joints ($m = 4 \times 2 = 8$). Therefore, $k = 9 - 8 = 1$ DOF. However, for the mechanism of Fig 3.8(a), $n = 4 \times 3 = 12$ and $m = 4 \times 3 = 12$, which yields $k = 12 - 12 = 0$ DOF, which is obviously incorrect. Therefore, Eq. 3.4 yields a correct answer only when the m constraint equations are independent.

Example 3.1

Five coordinates $\mathbf{q} = [l_1, \phi_1, l_2, \phi_2, l_3]^T$ are used to describe the configuration of bodies in a mechanism. Determine the number of degrees of freedom of the system if the coordinates are dependent according to the following six constraint equations:

$$\Phi_1 = 6 \cos \phi_1 - l_2 = 0$$
$$\Phi_2 = 6 \sin \phi_1 - l_3 = 0$$
$$\Phi_3 = l_1 \cos \phi_1 - 2 \cos \phi_2 = 0$$
$$\Phi_4 = l_1 \sin \phi_1 - 2 \sin \phi_2 - l_3 + 3 = 0$$
$$\Phi_5 = 2 \cos \phi_2 + (6 - l_1) \cos \phi_1 - l_2 = 0$$
$$\Phi_6 = 2 \sin \phi_2 + (6 - l_1) \sin \phi_1 - 3 = 0$$

Solution An investigation of the six equations reveals that $\Phi_2 = \Phi_4 + \Phi_6$ and $\Phi_1 = \Phi_3 + \Phi_5$.[†] Therefore, two of the equations are redundant, and hence, $m = 6 - 2 = 4$. Since $n = 5$, then $k = 5 - 4 = 1$ DOF.

3.2 KINEMATIC ANALYSIS

Kinematics is the study of the position, velocity, and acceleration of mechanisms. In kinematic analysis, only constraint equations are considered. The first and second time derivatives of the constraint equations yield the kinematic velocity and acceleration equations.

[†]It will be shown in the forthcoming sections how redundant equations can be found by such techniques as Gaussian elimination or L-U factorization.

For position analysis, at any given instant, the value of k coordinates must be known (where k is equal to the number of degrees of freedom). Hence, the constraint equations can be solved for the other $m = n - k$ coordinates. Similarly, for velocity and acceleration analysis, the value of k velocities and k accelerations must be known in order to solve the kinematic velocity and acceleration equations for the other, unknown velocities and accelerations.

The process of kinematic analysis is presented in two slightly different forms in the next two sections. Each method has a computational advantage and disadvantage in relation to the other.

3.2.1 Coordinate Partitioning Method

The fundamentals of kinematic analysis with the coordinate partitioning method can be best understood by following the process in a simple example.

Example 3.2

The four-bar mechanism of Fig. 3.7 is considered for kinematic analysis and is shown again in Fig. 3.9. All of the lengths are known, and it is given that the crank is rotating counterclockwise (CCW) with a constant angular velocity of 2π rad/s from the initial orientation of $\phi_1^0 = 2.36$ rad. The constraint equations of Eq. 3.1 are written as follows:

$$\begin{aligned} 0.2\cos\phi_1 + 0.4\cos\phi_2 - 0.3\cos\phi_3 - 0.35 &= 0 \\ 0.2\sin\phi_1 + 0.4\sin\phi_2 - 0.3\sin\phi_3 - 0.1 &= 0 \end{aligned} \tag{1}$$

For position analysis the substitution $\phi_1 = 2.36$ is made in Eq. 1 to get

$$\begin{aligned} 0.4\cos\phi_2 - 0.3\cos\phi_3 &= 0.49 \\ 0.4\sin\phi_2 - 0.3\sin\phi_3 &= -0.04 \end{aligned} \tag{2}$$

These equations are solved to find $\phi_2 = 0.57$ rad and $\phi_3 = 2.11$ rad. For velocity analysis, the first time derivative of Eq. 1 is written as

$$\begin{aligned} -0.2\sin\phi_1\dot{\phi}_1 - 0.4\sin\phi_2\dot{\phi}_2 + 0.3\sin\phi_3\dot{\phi}_3 &= 0 \\ 0.2\cos\phi_1\dot{\phi}_1 + 0.4\cos\phi_2\dot{\phi}_2 - 0.3\cos\phi_3\dot{\phi}_3 &= 0 \end{aligned} \tag{3}$$

For $\phi_1 = 2.36$, $\phi_2 = 0.57$, $\phi_3 = 2.11$, and known angular velocity of the crank, i.e., $\dot{\phi}_1 = 6.28$ rad/s, Eq. 3 becomes

$$\begin{aligned} -0.22\dot{\phi}_2 + 0.26\dot{\phi}_3 &= 0.89 \\ 0.34\dot{\phi}_2 + 0.16\dot{\phi}_3 &= 0.89 \end{aligned} \tag{4}$$

The solution of Eq. 4 yields $\dot{\phi}_2 = 0.76$ rad/s and $\dot{\phi}_3 = 4.09$ rad/s. Similarly, for acceleration analysis, the time derivatives of Eq. 3 is

$$\begin{aligned} -0.2\sin\phi_1\ddot{\phi}_1 - 0.2\cos\phi_1\dot{\phi}_1^2 - 0.4\sin\phi_2\ddot{\phi}_2 - 0.4\cos\phi_2\dot{\phi}_2^2 + \\ 0.3\sin\phi_3\ddot{\phi}_3 + 0.3\cos\phi_3\dot{\phi}_3^2 = 0 \end{aligned} \tag{5}$$

$$\begin{aligned} 0.2\cos\phi_1\ddot{\phi}_1 - 0.2\sin\phi_1\dot{\phi}_1^2 + 0.4\cos\phi_2\ddot{\phi}_2 - 0.4\sin\phi_2\dot{\phi}_2^2 - \\ 0.3\cos\phi_3\ddot{\phi}_3 + 0.3\sin\phi_3\dot{\phi}_3^2 = 0 \end{aligned}$$

Since ϕ_1, ϕ_2, ϕ_3, $\dot{\phi}_1$, $\dot{\phi}_2$, $\dot{\phi}_3$ are known and $\ddot{\phi}_1 = 0$ (indicating constant angular velocity), Eq. 5 becomes

$$\begin{aligned} -0.22\ddot{\phi}_2 + 0.26\ddot{\phi}_3 &= -2.86 \\ 0.34\ddot{\phi}_2 + 0.16\ddot{\phi}_3 &= 1.39 \end{aligned} \tag{6}$$

This yields $\ddot{\phi}_2 = 6.62$ rad/s^2 and $\ddot{\phi}_3 = -5.39$ rad/s^2.

The process of position, velocity, and acceleration analysis can be repeated for different positions of the crank. If ϕ_1 is varied from its initial value through a complete revolution of the crank, then at every step the position, velocity, and acceleration analysis yield the results shown in the following table:

ϕ_1	ϕ_2	ϕ_3	$\dot{\phi}_2$	$\dot{\phi}_3$	$\ddot{\phi}_2$	$\ddot{\phi}_3$
2.36	0.57	2.11	0.76	4.09	6.62	−5.39
2.52	0.59	2.21	0.94	3.93	7.21	−7.17
2.67	0.62	2.31	1.13	3.73	7.91	−8.97
2.83	0.65	2.40	1.33	3.48	8.66	−10.74
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
8.49	0.55	2.01	0.60	4.20	6.21	−3.61
8.64	0.57	2.11	0.76	4.09	6.62	−5.39

$\dot{\phi}_1 = 6.28$ and $\ddot{\phi}_1 = 0$.

The result of the position analysis for one complete revolution of the crank is shown in Fig. 3.9.

In the preceding example, the angle ϕ_1, which has a known value at every instant, is called the *independent coordinate* or the *driving coordinate*. The remaining coordinates, such as ϕ_2 and ϕ_3, are called the *dependent coordinates* or the *driven coordinates*. The number of independent coordinates is equal to the number of degrees of freedom of the system; therefore the number of dependent coordinates is equal to the number of independent constraint equations in the system.

Kinematic analysis with coordinate partitioning considers the partitioned form of the coordinate vector $\mathbf{q} = [\mathbf{u}^T, \mathbf{v}^T]^T$, where $\mathbf{u}$ and $\mathbf{v}$ are the dependent and independent coordinates, respectively. The m constraint equations

$$\boldsymbol{\Phi} \equiv \boldsymbol{\Phi}(\mathbf{q}) = \mathbf{0} \tag{3.5}$$

may be expressed as

$$\boldsymbol{\Phi} \equiv \boldsymbol{\Phi}(\mathbf{u}, \mathbf{v}) = \mathbf{0} \tag{3.6}$$

Constraint equations as presented by Eq. 3.6 are, in general, nonlinear. For position analysis, iterative numerical methods may be used to solve the set of nonlinear algebraic equations. One such method is discussed in Section 3.4. The k independent coordinates $\mathbf{v}$ are specified at each instant of time t. Then, Eq. 3.6 becomes a set of m equations in

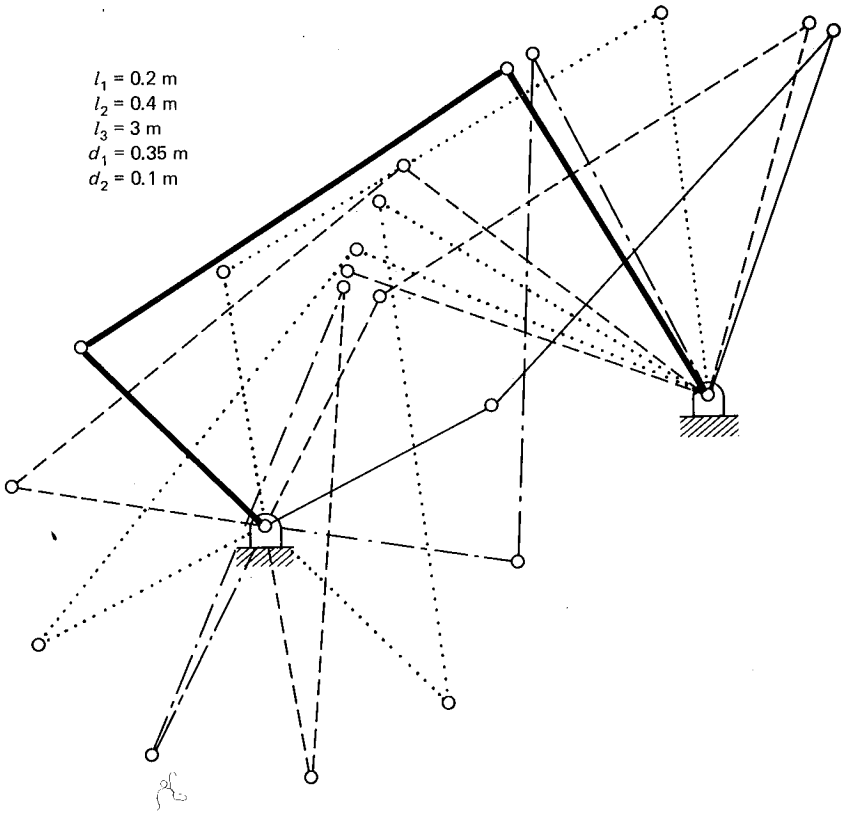


Figure 3.9 The result of the position analysis for the four-bar mechanism of Figure 3.7.

m unknowns, which may be solved for the m dependent coordinates $\mathbf{u}$. If the constraints of Eq. 3.5 are independent, then the existence of a solution to $\mathbf{u}$ for a given $\mathbf{v}$ is asserted by the *implicit function theorem*† of calculus.

Differentiation of Eq. 3.5 yields velocity equations

$$\boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}} = \mathbf{0} \tag{3.7}$$

where $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n]^T$ is the *vector of velocities*. The matrix $\boldsymbol{\Phi}_{\mathbf{q}} \equiv [\partial\boldsymbol{\Phi}/\partial\mathbf{q}]$, which contains partial derivatives of the constraint equations with respect to the coordinates, is called the *constraint Jacobian* matrix. Let $\dot{\mathbf{v}} = [\dot{v}_1, \dot{v}_2, \ldots, \dot{v}_k]^T$ represent the *independent velocities* with known values, and let $\dot{\mathbf{u}} = [\dot{u}_1, \dot{u}_2, \ldots, \dot{u}_m]^T$ represent the m *dependent velocities*. Equation 3.7 may be rewritten in partitioned form as

$$\boldsymbol{\Phi}_{\mathbf{u}}\dot{\mathbf{u}} = -\boldsymbol{\Phi}_{\mathbf{v}}\dot{\mathbf{v}} \tag{3.8}$$

†**Implicit Function Theorem:**[3] Consider a point $\mathbf{q}^i = [\mathbf{u}^{i^T}, \mathbf{v}^{i^T}]^T$ at time t^i for which the constraint equations are satisfied; i.e., for which

$$\boldsymbol{\Phi}(\mathbf{u}^i, v^i) = \mathbf{0} \tag{a}$$

If the partitioning of $\mathbf{q}$ into $\mathbf{u}$ and $\mathbf{v}$ has been selected so that the matrix $\boldsymbol{\Phi}_{\mathbf{u}} \equiv [\partial\boldsymbol{\Phi}/\partial\mathbf{u}]$ at $(\mathbf{u}^i, \mathbf{v}^i)$ is nonsingular, then in some neighborhood of $\mathbf{v}^i$ Eq. a has a unique solution $\mathbf{u} = \boldsymbol{\Psi}(\mathbf{v})$; i.e., $\boldsymbol{\Phi}(\boldsymbol{\Psi}(\mathbf{v}), \mathbf{v}) = \mathbf{0}$. Furthermore, if $\boldsymbol{\Phi}(\mathbf{u}, \mathbf{v})$ is j times continuously differentiable in its arguments, so is $\boldsymbol{\Psi}(\mathbf{v})$.

where $\mathbf{\Phi}_u$ and $\mathbf{\Phi}_v$ are two submatrices of $\mathbf{\Phi}_q$ that contain the columns of $\mathbf{\Phi}_q$ associated with $\mathbf{u}$ and $\mathbf{v}$, respectively. The term on the right side of Eq. 3.8 is denoted by

$$\boldsymbol{\nu} = -\mathbf{\Phi}_v \dot{\mathbf{v}} \tag{3.9}$$

Since the constraint equations of Eq. 3.5 are assumed to be independent, then $\mathbf{\Phi}_u$ is an $m \times m$ nonsingular matrix, and so Eq. 3.8 may be solved directly for $\dot{\mathbf{u}}$, once $\dot{\mathbf{v}}$ is given.

Differentiating the velocity equations of Eq. 3.7 yields the *acceleration equations*

$$\mathbf{\Phi}_q \ddot{\mathbf{q}} + (\mathbf{\Phi}_q \dot{\mathbf{q}})_q \dot{\mathbf{q}} = \mathbf{0} \tag{3.10}$$

where $\ddot{\mathbf{q}} = [\ddot{q}_1, \ddot{q}_2, \ldots, \ddot{q}_n]^T$ is the *vector of accelerations*. Let $\ddot{\mathbf{v}} = [\ddot{v}_1, \ddot{v}_2, \ldots, \ddot{v}_k]^T$ represent the *independent accelerations*, and $\ddot{\mathbf{u}} = [\ddot{u}_1, \ddot{u}_2, \ldots, \ddot{u}_m]^T$ represent the *dependent accelerations*. Equation 3.10 can be written in partitioned form as

$$\mathbf{\Phi}_u \ddot{\mathbf{u}} = -\mathbf{\Phi}_v \ddot{\mathbf{v}} - (\mathbf{\Phi}_q \dot{\mathbf{q}})_q \dot{\mathbf{q}} \tag{3.11}$$

Since $\mathbf{\Phi}_u$ is nonsingular, Eq. 3.11 can be solved for $\ddot{\mathbf{u}}$, once $\ddot{\mathbf{v}}$ is given. Note that the velocity and acceleration equations of Eqs. 3.8 and 3.11 are sets of *linear* algebraic equations in $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, respectively, whereas the constraint equations of Eq. 3.6 are *nonlinear* algebraic equations.

The general procedure for kinematic analysis, using the coordinate partitioning method, may be summarized in the following algorithm:

ALGORITHM K-I

(a) Set a time step counter i to $i = 0$ and initialize $t^i = t^0$ (initial time).
(b) Partition $\mathbf{q}$ into dependent and independent sets $\mathbf{u}$ and $\mathbf{v}$.
(c) Specify independent coordinates $\mathbf{v}^i$ and solve Eq. 3.6 iteratively for $\mathbf{u}^i$.
(d) Specify independent velocities $\dot{\mathbf{v}}^i$ and solve Eq. 3.8 for $\dot{\mathbf{u}}^i$.
(e) Specify independent accelerations $\ddot{\mathbf{v}}^i$ and solve Eq. 3.11 for $\ddot{\mathbf{u}}^i$.
(f) If the final time has been reached, then terminate; otherwise increment t^i to a new time t^{i+1}, let $i \rightarrow i + 1$, and go to (c).

The simple form of the constraint equations of Example 3.2 may give the impression that the constraint equations can always be explicitly partitioned into terms containing the independent coordinates and terms containing the dependent coordinates. In general, this partitioning is not possible for highly nonlinear equations. However, regardless of the order of nonlinearity of the constraint equations, the velocity and acceleration equations can be partitioned according to Eqs. 3.8 and 3.11, since they are linear in terms of $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, respectively.

Example 3.3

Derive the velocity and acceleration equations for the constraint equations

$$\begin{aligned} 2\cos\phi_1 + 3\cos(\phi_1 - \phi_2) - 2\cos(\phi_3 + \phi_4) - 4\cos\phi_4 - 5 &= 0 \\ 2\sin\phi_1 + 3\sin(\phi_1 - \phi_2) - 2\sin(\phi_3 + \phi_4) - 4\sin\phi_4 - 1 &= 0 \end{aligned} \tag{1}$$

Then express these equations in partitioned form if ϕ_1 and ϕ_3 are assumed to be the independent coordinates.

Solution The vector of coordinates is $\mathbf{q} = [\phi_1, \phi_2, \phi_3, \phi_4]^T$; thus $\mathbf{v} = [\phi_1, \phi_3]^T$ and $\mathbf{u} = [\phi_2, \phi_4]^T$. Since the constraint equations are nonlinear, they cannot be

partitioned explicitly in terms of **u** and **v**, and therefore they are left as they are in Eq. 1.

The kinematic velocity equations can be found either by direct differentiation of Eq. 1 or by using Eq. 3.7, as follows:

$$\begin{bmatrix} -2\sin\phi_1 - 3\sin(\phi_1-\phi_2) & 3\sin(\phi_1-\phi_2) \\ 2\cos\phi_1 + 3\cos(\phi_1-\phi_2) & -3\cos(\phi_1-\phi_2) \\ 2\sin(\phi_3+\phi_4) & 2\sin(\phi_3+\phi_4)+4\sin\phi_4 \\ -2\cos(\phi_3+\phi_4) & -2\cos(\phi_3+\phi_4)-4\cos\phi_4 \end{bmatrix} \begin{bmatrix} \dot\phi_1 \\ \dot\phi_2 \\ \dot\phi_3 \\ \dot\phi_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2}$$

where the 2×4 matrix at the left of Eq. 2 is the Jacobian matrix. Partitioning of Eq. 2 yields the velocity equations in the form of Eq. 3.8:

$$\begin{bmatrix} 3\sin(\phi_1-\phi_2) & 2\sin(\phi_3+\phi_4)+4\sin\phi_4 \\ -3\cos(\phi_1-\phi_2) & -2\cos(\phi_3+\phi_4)-4\cos\phi_4 \end{bmatrix} \begin{bmatrix} \dot\phi_2 \\ \dot\phi_4 \end{bmatrix} = -\begin{bmatrix} -2\sin\phi_1 - 3\sin(\phi_1-\phi_2) & 2\sin(\phi_3+\phi_4) \\ 2\cos\phi_1 + \cos(\phi_1-\phi_2) & -2\cos(\phi_3+\phi_4) \end{bmatrix} \begin{bmatrix} \dot\phi_1 \\ \dot\phi_3 \end{bmatrix} \tag{3}$$

The kinematic acceleration equations can be found either by direct differentiation of Eq. 2 or by using Eq. 3.10, as follows:

$$\begin{bmatrix} -2\sin\phi_1 - 3\sin(\phi_1-\phi_2) & 3\sin(\phi_1-\phi_2) \\ 2\cos\phi_1 + 3\cos(\phi_1-\phi_2) & -3\cos(\phi_1-\phi_2) \\ 2\sin(\phi_3+\phi_4) & 2\sin(\phi_3+\phi_4)+4\sin\phi_4 \\ -2\cos(\phi_3+\phi_4) & -2\cos(\phi_3+\phi_4)-4\cos\phi_4 \end{bmatrix} \begin{bmatrix} \ddot\phi_1 \\ \ddot\phi_2 \\ \ddot\phi_3 \\ \ddot\phi_4 \end{bmatrix} +$$

$$\begin{bmatrix} -2\cos\phi_1\dot\phi_1^2 - 3\cos(\phi_1-\phi_2)(\dot\phi_1-\dot\phi_2)^2 + \\ -2\sin\phi_1\dot\phi_1^2 - 3\sin(\phi_1-\phi_2)(\dot\phi_1-\dot\phi_2)^2 + \\ 2\cos(\phi_3+\phi_4)(\dot\phi_3+\dot\phi_4)^2 + 4\cos\phi_4\dot\phi_4^2 \\ 2\sin(\phi_3+\phi_4)(\dot\phi_3+\dot\phi_4)^2 + 4\sin\phi_4\dot\phi_4^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4}$$

Partitioning of Eq. 4 yields the acceleration equations, in the form of Eq. 3.11:

$$\begin{bmatrix} 3\sin(\phi_1-\phi_2) & 2\sin(\phi_3+\phi_4)+4\sin\phi_4 \\ -3\cos(\phi_1-\phi_2) & -2\cos(\phi_3+\phi_4)-4\cos\phi_4 \end{bmatrix} \begin{bmatrix} \ddot\phi_2 \\ \ddot\phi_4 \end{bmatrix} =$$

$$-\begin{bmatrix} -2\sin\phi_1 - 3\sin(\phi_1-\phi_2) & 2\sin(\phi_3+\phi_4) \\ 2\cos\phi_1 + \cos(\phi_1-\phi_2) & -2\cos(\phi_3+\phi_4) \end{bmatrix} \begin{bmatrix} \ddot\phi_1 \\ \ddot\phi_3 \end{bmatrix} -$$

$$\begin{bmatrix} -2\cos\phi_1\dot\phi_1^2 - 3\cos(\phi_1-\phi_2)(\dot\phi_1-\dot\phi_2)^2 + \\ -2\sin\phi_1\dot\phi_1^2 - 3\sin(\phi_1-\phi_2)(\dot\phi_1-\dot\phi_2)^2 + \\ 2\cos(\phi_3+\phi_4)(\dot\phi_3+\dot\phi_4)^2 + 4\cos\phi_4\dot\phi_4^2 \\ 2\sin(\phi_3+\phi_4)(\dot\phi_3+\dot\phi_4)^2 + 4\sin\phi_4\dot\phi_4^2 \end{bmatrix} \tag{5}$$

3.2.2 Method of Appended Driving Constraints

This method, unlike the coordinate partitioning method, does not partition the coordinates into independent and dependent sets. Additional constraint equations, called the *driving constraints,* equal in number to the number of degrees of freedom of the system, are appended to the original kinematic constraints. The driving constraints are equations representing each independent coordinate as a function of time. This method is best illustrated by an example.

Example 3.4

The four-bar mechanism of Example 3.2 is considered here again. The kinematic constraints are

$$\begin{aligned} \Phi_1 &\equiv 0.2 \cos \phi_1 + 0.4 \cos \phi_2 - 0.3 \cos \phi_3 - 0.35 = 0 \\ \Phi_2 &\equiv 0.2 \sin \phi_1 + 0.4 \sin \phi_2 - 0.3 \sin \phi_3 - 0.1 = 0 \end{aligned} \tag{1}$$

Since the crank angle ϕ_1 is the independent variable, a driving constraint can be written in terms of ϕ_1. The initial value at $t = 0$ for ϕ_1 is $\phi_1^0 = 2.36$ rad, and the constant angular velocity of the crank is 2π rad/s. Therefore $\phi_1 = 2.36 + 6.28t$ can be used to represent ϕ_1 as a function of time. This driving equation can be rewritten as

$$\Phi^{(d)} \equiv \phi_1 - 2.36 - 6.28t = 0 \tag{2}$$

At any instant of time, i.e., known t, Eqs. 1 and 2 represent three equations in three unknowns ϕ_1, ϕ_2, and ϕ_3. If these equations are solved for $t = 0$, it is found that $\phi_1 = 2.36$, $\phi_2 = 0.57$, and $\phi_3 = 2.11$.

For velocity analysis, the first time derivatives of Eqs. 1 and 2 are found and written as

$$\begin{bmatrix} -0.2 \sin \phi_1 & -0.4 \sin \phi_2 & 0.3 \sin \phi_3 \\ 0.2 \cos \phi_1 & 0.4 \cos \phi_2 & -0.3 \cos \phi_3 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 6.28 \end{bmatrix} \tag{3}$$

For known values of ϕ_1, ϕ_2, and ϕ_3, Eq. 3 yields $\dot{\phi}_1 = 6.28$, $\dot{\phi}_2 = 0.76$, and $\dot{\phi}_3 = 4.09$.

For acceleration analysis, the time derivative of Eq. 3 is found to be

$$\begin{bmatrix} -0.2 \sin \phi_1 & -0.4 \sin \phi_2 & 0.3 \sin \phi_3 \\ 0.2 \cos \phi_1 & 0.4 \cos \phi_2 & -0.3 \cos \phi_3 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{\phi}_1 \\ \ddot{\phi}_2 \\ \ddot{\phi}_3 \end{bmatrix} =$$

$$\begin{bmatrix} 0.2 \cos \phi_1 \dot{\phi}_1^2 + 0.4 \cos \phi_2 \dot{\phi}_2^2 - 0.3 \cos \phi_3 \dot{\phi}_3^2 \\ 0.2 \sin \phi_1 \dot{\phi}_1^2 + 0.4 \sin \phi_2 \dot{\phi}_2^2 - 0.3 \sin \phi_3 \dot{\phi}_3^2 \\ 0 \end{bmatrix} \tag{4}$$

Since ϕ_1, ϕ_2, ϕ_3, $\dot{\phi}_1$, $\dot{\phi}_2$, and $\dot{\phi}_3$ are known, Eq. 4 yields $\ddot{\phi}_1 = 0$, $\ddot{\phi}_2 = 6.62$, and $\ddot{\phi}_3 = 5.39$.

This process can be repeated at different instants of time. If t is incremented by $\Delta t = 0.025$ s, the same table as shown in Example 3.2 is obtained.

The method of appended driving constraints can now be stated in its most general form. If there are m kinematic constraints, then k driving constraints must be appended to the kinematic constraints to obtain $n = m + k$ equations:

$$\begin{aligned} \boldsymbol{\Phi} &\equiv \boldsymbol{\Phi}(\mathbf{q}) = \mathbf{0} \\ \boldsymbol{\Phi}^{(d)} &\equiv \boldsymbol{\Phi}(\mathbf{q}, t) = \mathbf{0} \end{aligned} \tag{3.12}$$

where superscript (d) denotes the driving constraints. Equation 3.12 represents n equations in n unknowns $\mathbf{q}$ which can be solved at any specified time t.

The velocity equations are obtained by taking the time derivative of Eq. 3.12:

$$\begin{aligned} \boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}} &= \mathbf{0} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{(d)}\dot{\mathbf{q}} + \boldsymbol{\Phi}_{t}^{(d)} &= \mathbf{0} \end{aligned} \tag{3.13}$$

or

$$\begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{q}} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{(d)} \end{bmatrix} \dot{\mathbf{q}} = \begin{bmatrix} \mathbf{0} \\ -\boldsymbol{\Phi}_{t}^{(d)} \end{bmatrix} \tag{3.14}$$

which represents n algebraic equations, linear in terms of $\dot{\mathbf{q}}$.

Similarly, the time derivative of Eq. 3.13 yields the acceleration equations:

$$\begin{aligned} \boldsymbol{\Phi}_{\mathbf{q}}\ddot{\mathbf{q}} + (\boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}})_{\mathbf{q}}\dot{q} &= \mathbf{0} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{(d)}\ddot{\mathbf{q}} + (\boldsymbol{\Phi}_{\mathbf{q}}^{(d)}\dot{\mathbf{q}})_{\mathbf{q}}\dot{q} + 2\boldsymbol{\Phi}_{\mathbf{q}t}^{(d)}\dot{\mathbf{q}} + \boldsymbol{\Phi}_{tt}^{(d)} &= \mathbf{0} \end{aligned} \tag{3.15}$$

or

$$\begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{q}} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{(d)} \end{bmatrix} \ddot{\mathbf{q}} = \begin{bmatrix} -(\boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}} \\ -(\boldsymbol{\Phi}_{\mathbf{q}}^{(d)}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}} - 2\boldsymbol{\Phi}_{\mathbf{q}t}^{(d)}\dot{\mathbf{q}} - \boldsymbol{\Phi}_{tt}^{(d)} \end{bmatrix} \tag{3.16}$$

which represents n algebraic equations linear in terms of $\ddot{\mathbf{q}}$. The term $-(\boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}}$ in Eq. 3.16 is referred to as the *right side of the kinematic acceleration equations,* and is represented as

$$\boldsymbol{\gamma} = -(\boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}} \tag{3.17}$$

In the above formulations, the driving constraints are assumed to have the general form $\boldsymbol{\Phi}(\mathbf{q}, t) = \mathbf{0}$. However, as shown in Example 3.4, the driving constraint can have a very simple form, such as $v_j - c(t) = 0$, where v_j is the jth independent variable and $c(t)$ is a known function of time. If there are k independent variables in the system and the k driving constraints have the form

$$\boldsymbol{\Phi}^{(d)} \equiv \mathbf{v} - \mathbf{c}(t) = \mathbf{0} \tag{3.18}$$

then Eqs. 3.12 through 3.16 can be simplified. In this case the Jacobian matrix becomes

$$\begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{q}} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{(d)} \end{bmatrix} \equiv \begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{q}} \\ \hat{\mathbf{I}} \end{bmatrix} \tag{3.19}$$

where $\hat{\mathbf{I}}$ is a permuted nonsquare identity matrix (*permuted* means the columns are reordered). The 1s of $\hat{\mathbf{I}}$ are in the columns associated with the independent variables $\mathbf{v}$. Therefore, Eq. 3.19 can be expressed as

$$\begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{q}} \\ \hat{\mathbf{I}} \end{bmatrix} \equiv \begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{u}} & \boldsymbol{\Phi}_{\mathbf{v}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.20}$$

Example 3.5

Four coordinates $\mathbf{q} = [x_1, x_2, x_3, x_4]^T$ are subject to the kinematic constraints

$$x_1^2 + 2x_2x_3 - x_1x_4 = 0$$
$$3x_1x_2 - x_2^2 + x_3x_4 - x_3 = 0$$

The coordinates x_2 and x_4 are expressed in terms of t as follows:

$$x_2 - 0.2t = 0$$
$$x_4 + 0.5t - 0.03t^2 = 0$$

The system Jacobian is

$$\begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{q}} \\ \hat{\mathbf{I}} \end{bmatrix} = \begin{bmatrix} 2x_1 - x_4 & 2x_3 & 2x_2 & -x_1 \\ 3x_2 & 3x_1 - 2x_2 & x_4 - 1 & x_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In terms of $\boldsymbol{\Phi}_{\mathbf{u}}$ and $\boldsymbol{\Phi}_{\mathbf{v}}$ the Jacobian is permuted as follows:

$$\begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{u}} & \boldsymbol{\Phi}_{\mathbf{v}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} 2x_1 - x_4 & 2x_2 & 2x_3 & -x_1 \\ 3x_2 & x_4 - 1 & 3x_1 - 2x_2 & x_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The general procedure for kinematic analysis using the method of appended driving constraints may be summarized in the following algorithm:

ALGORITHM K-II

(a) Set a time step counter i to $i = 0$ and initialize $t^i = t^0$ (initial time).
(b) Append k driving equations to the constraint equations.
(c) Solve Eq. 3.12 iteratively to obtain $\mathbf{q}^i$.
(d) Solve Eq. 3.14 to obtain $\dot{\mathbf{q}}^i$.
(e) Solve Eq. 3.16 to obtain $\ddot{\mathbf{q}}^i$.
(f) If final time is reached, then terminate; otherwise increment t^i to t^{i+1}, let $i \rightarrow i + 1$, and go to (c).

Kinematic analysis with the method of appended driving constraints usually requires the solution of a larger set of equations than with the coordinate partitioning method. However, this is not a major drawback since the programming effort for computer implementation of algorithm K-II is much less than of algorithm K-I. A computer program for kinematic analysis of mechanical systems based on this method is described in Chap. 5.

3.3 LINEAR ALGEBRAIC EQUATIONS

It was shown in Secs. 3.21 and 3.22 that the kinematic velocity and acceleration analysis of mechanical systems requires the solution of a set of linear algebraic equations. Al-

though position analysis involves the solution of a set of nonlinear algebraic equations, it will be seen that most numerical methods solve these by iteratively solving a set of linear equations. Therefore, for almost every step of kinematic analysis—position, velocity, or acceleration—sets of linear algebraic equations must be solved.

Consider a system of n linear algebraic equations with real constant coefficients,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= c_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= c_2 \\ &\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= c_n \end{aligned} \tag{3.21}$$

which can be written in matrix form as

$$\mathbf{Ax} = \mathbf{c} \tag{3.22}$$

There are many methods for solving this set of equations. Cramer's rule offers one of the best-known methods, but also the most inefficient. Among the more efficient methods are the Gaussian elimination, Gauss-Jordan reduction, and L-U factorization methods.

3.3.1 Gaussian Methods

The *Gaussian elimination method* for solving linear equations is based on the elementary idea of eliminating variables one at a time. This method consists of two major steps: (1) a forward elimination, which converts the matrix **A** into an upper-triangular matrix, and (2) a back substitution, which solves for the unknown **x**. There are a variety of Gaussian elimination algorithms that are similar in principle, but slightly different in approach. The method presented here converts the matrix **A** to an upper-triangular matrix with 1s on the diagonal. The process is best illustrated by an example that can be followed easily.

Example 3.6

Solve the set of equations

$$\begin{bmatrix} 3 & 1 & -1 \\ -1 & 2 & 1 \\ 2 & -3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ -1 \end{bmatrix}$$

to find x_1, x_2, and x_3.

Solution

FORWARD ELIMINATION

1. Multiply the first equation by $\frac{1}{3}$, to put a 1 in the a_{11} position, as follows:

$$\begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ -1 & 2 & 1 \\ 2 & -3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ 6 \\ -1 \end{bmatrix}$$

and add the first equation to the second and then add -2 times the first equation to the third, to put zeros in the first column below the diagonal:

$$\begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{7}{3} & \frac{2}{3} \\ 0 & -\frac{11}{3} & \frac{5}{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ \frac{20}{3} \\ -\frac{7}{3} \end{bmatrix}$$

2. Multiply the second equation by $\frac{3}{7}$, to put a 1 in the a_{22} position, as shown:

$$\begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & 1 & \frac{2}{7} \\ 0 & -\frac{11}{3} & \frac{5}{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ \frac{20}{7} \\ -\frac{7}{3} \end{bmatrix}$$

and add $\frac{11}{3}$ times the second equation to the third, to put a zero below the diagonal:

$$\begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & 1 & \frac{2}{7} \\ 0 & 0 & \frac{19}{7} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ \frac{20}{7} \\ \frac{57}{7} \end{bmatrix}$$

3. Multiply the third equation by $\frac{7}{19}$, to put a 1 in the a_{33} position:

$$\begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & 1 & \frac{2}{7} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ \frac{20}{7} \\ 3 \end{bmatrix}$$

BACK SUBSTITUTION

1. The third equation yields $x_3 = 3$.
2. The second equation then yields $x_2 + \frac{2}{7}(3) = \frac{20}{7}$, so $x_2 = 2$.
3. The first equation yields $x_1 + \frac{1}{3}(2) - \frac{1}{3}(3) = \frac{2}{3}$, so $x_1 = 1$.

Note that the forward-elimination step requires division by a_{jj} at the jth step. The preceding operation is valid only if $a_{jj} \neq 0$.

The *Gauss-Jordan reduction method* combines the forward-elimination and back-substitution steps of the Gaussian elimination method into one step. Matrix **A** is converted to a diagonal unit matrix, using elementary arithmetic. This method may also be illustrated with a simple example.

Example 3.7

Solve the set of linear algebraic equations

$$\begin{bmatrix} 2 & -1 & 1 \\ 1 & 3 & -2 \\ -1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -3 \\ 3 \\ 5 \end{bmatrix}$$

using the Gauss-Jordan reduction method.

Solution

1. Multiply the first equation by $\frac{1}{2}$, to put a 1 in the a_{11} position, as shown:

$$\begin{bmatrix} 1 & -\frac{1}{2} & \frac{1}{2} \\ 1 & 3 & -2 \\ -1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\frac{3}{2} \\ 3 \\ 5 \end{bmatrix}$$

and add -1 times the first equation to the second and then add the first equation to the third to put zeros below the diagonal:

$$\begin{bmatrix} 1 & -\frac{1}{2} & \frac{1}{2} \\ 0 & \frac{7}{2} & -\frac{5}{2} \\ 0 & \frac{1}{2} & \frac{5}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\frac{3}{2} \\ \frac{9}{2} \\ \frac{7}{2} \end{bmatrix}$$

2. Multiply the second equation by $\frac{2}{7}$ to put a 1 in the a_{22} position, as shown:

$$\begin{bmatrix} 1 & -\frac{1}{2} & \frac{1}{2} \\ 0 & 1 & -\frac{5}{7} \\ 0 & \frac{1}{2} & \frac{5}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\frac{3}{2} \\ \frac{9}{7} \\ \frac{7}{2} \end{bmatrix}$$

and add $\frac{1}{2}$ times the second equation to the first and then add $-\frac{1}{2}$ times the second equation to the third, to put zeros above and below the a_{22} position:

$$\begin{bmatrix} 1 & 0 & \frac{1}{7} \\ 0 & 1 & -\frac{5}{7} \\ 0 & 0 & \frac{20}{7} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\frac{6}{7} \\ \frac{9}{7} \\ \frac{20}{7} \end{bmatrix}$$

3. Multiply the third equation by $\frac{7}{20}$, to put a 1 in the a_{33} position:

$$\begin{bmatrix} 1 & 0 & \frac{1}{7} \\ 0 & 1 & -\frac{5}{7} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\frac{6}{7} \\ \frac{9}{7} \\ 1 \end{bmatrix}$$

and add $-\frac{1}{7}$ times the third equation to the first and then add $\frac{5}{7}$ times the third equation to the second, to put zeros above the a_{33} position:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$

Since the coefficient matrix has been converted to the identity matrix it is clear that $x_1 = -1$, $x_2 = 2$, and $x_3 = 1$ is the solution.

3.3.2 Pivoting

In the forward-elimination step of the Gaussian methods, the algorithm fails if at the jth step a_{jj} is zero. Also, when the pivot element a_{jj} becomes too small, numerical error

may occur. Therefore, the order in which the equations are treated during the forward-elimination step significantly affects the accuracy of the algorithm. To circumvent this difficulty, the order in which the equations are used is determined by the algorithm; i.e., the order may not necessarily be the original order $1, 2, \ldots, n$. The algorithm reorders the equations depending on the actual system being solved. This process is called pivoting. Two types of pivoting, *partial pivoting* and *full pivoting*, are discussed here.

In *partial pivoting*, during the jth forward-elimination step of the Gaussian algorithm, the equation with the largest coefficient (in absolute value) of x_j on or below the diagonal is chosen for pivoting. During the elimination step, *the rows of the matrix and also the elements of vector* $\mathbf{c}$ *are interchanged.* The following example illustrates this procedure.

Example 3.8

Perform Gaussian elimination with partial (row) pivoting on the following set of equations:

$$\begin{bmatrix} 4 & -3 & 5 & 2 \\ -3 & 1 & 1 & -6 \\ 5 & -5 & 10 & 0 \\ 2 & -3 & 9 & -7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -1.5 \\ 9 \\ 2.5 \\ 13.5 \end{bmatrix}$$

Solution

FORWARD ELIMINATION WITH PARTIAL PIVOTING

1. The largest coefficient in column 1 of the matrix is 5. Therefore interchange the third and first equations, to obtain

$$\begin{bmatrix} 5 & -5 & 10 & 0 \\ -3 & 1 & 1 & -6 \\ 4 & -3 & 5 & 2 \\ 2 & -3 & 9 & -7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 9 \\ -1.5 \\ 13.5 \end{bmatrix}$$

Then perform forward elimination, as in Example 3.6, to obtain

$$\begin{bmatrix} 1 & -1 & 2 & 0 \\ 0 & -2 & 7 & -6 \\ 0 & 1 & -3 & 2 \\ 0 & -1 & 5 & -7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 10.5 \\ -3.5 \\ 12.5 \end{bmatrix}$$

2. The largest coefficient in column 2 on or below the diagonal is -2; no interchange is necessary. Forward elimination yields

$$\begin{bmatrix} 1 & -1 & 2 & 0 \\ 0 & 1 & -3.5 & 3 \\ 0 & 0 & 0.5 & -1 \\ 0 & 0 & 1.5 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -5.25 \\ 1.75 \\ 7.25 \end{bmatrix}$$

3. The largest coefficient in column 3 on or below the diagonal is 1.5. Interchange the fourth and third equations to obtain

$$\begin{bmatrix} 1 & -1 & 2 & 0 \\ 0 & 1 & -3.5 & 3 \\ 0 & 0 & 1.5 & -4 \\ 0 & 0 & 0.5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -5.25 \\ 7.25 \\ 1.75 \end{bmatrix}$$

Then perform forward elimination to obtain

$$\begin{bmatrix} 1 & -1 & 2 & 0 \\ 0 & 1 & -3.5 & 3 \\ 0 & 0 & 1 & -2.66 \\ 0 & 0 & 0 & 0.33 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -5.25 \\ 4.83 \\ -0.66 \end{bmatrix}$$

4. Multiply the fourth equation by $\frac{1}{0.33}$ to obtain

$$\begin{bmatrix} 1 & -1 & 2 & 0 \\ & 1 & 3.5 & 3 \\ & & 1 & -2.66 \\ & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -5.25 \\ 4.83 \\ -2 \end{bmatrix}$$

Back substitution now yields $\mathbf{x} = [0.5, -1, -0.5, -2]^T$.

The preceding pivoting method is referred to as partial pivoting with *row interchange,* since the rows of the matrix are interchanged. The method can be modified for partial pivoting with *column interchange.*

Full or *complete pivoting* is the selection of the largest of all the coefficients (in absolute value) on the diagonal and to its right and below as the basis for the next stage of elimination, which operates on the corresponding variable. In full pivoting, both row interchange and column interchange are required. Note that *when two columns of the coefficient matrix are interchanged, their corresponding variables in the vector* **x** *are interchanged.*

Example 3.9

Apply the Gaussian elimination method with full pivoting to the following set of equations:

$$\begin{bmatrix} 2 & -1 & 1 \\ -1 & 0 & 2 \\ 1 & 4 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \\ -5 \end{bmatrix}$$

Solution

1. The largest coefficient in the matrix is 4. Interchanging columns 2 and 1, we get

$$\begin{bmatrix} -1 & 2 & 1 \\ 0 & -1 & 2 \\ 4 & 1 & -2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \\ -5 \end{bmatrix}$$

Interchanging rows 3 and 1 yields

$$\begin{bmatrix} 4 & 1 & -2 \\ 0 & -1 & 2 \\ -1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} -5 \\ 5 \\ 0 \end{bmatrix}$$

The elimination step gives

$$\begin{bmatrix} 1 & 0.25 & -0.5 \\ 0 & -1 & 2 \\ 0 & 2.25 & 0.5 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1.25 \\ 5 \\ -1.25 \end{bmatrix}$$

2. The largest coefficient in the 2×2 lower right submatrix is 2.25. Interchanging rows 3 and 2, we have

$$\begin{bmatrix} 1 & 0.25 & -0.5 \\ 0 & 2.25 & 0.5 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1.25 \\ -1.25 \\ 5 \end{bmatrix}$$

The elimination step gives

$$\begin{bmatrix} 1 & 0.25 & -0.5 \\ 0 & 1 & 0.22 \\ 0 & 0 & 2.22 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1.25 \\ -0.55 \\ 4.44 \end{bmatrix}$$

3. Multiplying the last equation by $\frac{1}{2.22}$, we get

$$\begin{bmatrix} 1 & 0.25 & -0.5 \\ 0 & 1 & 0.22 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1.25 \\ -0.55 \\ 2 \end{bmatrix}$$

Back substitution now yields $x_3 = 2$, $x_1 = -1$, and $x_2 = 0$.

In most computer programs, partial or full pivoting is carried out simply by interchanging the row (column) indices of the two rows (columns) to be interchanged. Two integer arrays hold the indices for column and row numbers of the matrix.

3.3.3 L-U Factorization

The *L-U factorization method* is a compact form of the Gaussian elimination method of operating on a matrix **A**. After the operation is completed, the set of linear equations $\mathbf{Ax} = \mathbf{c}$ is efficiently solved for any given **c** vector.

For any nonsingular matrix **A**, it can be proved that there exists an upper triangular matrix **U** with nonzero diagonal elements and a lower triangular matrix **L** with unit diagonal elements, such that

$$\mathbf{A} = \mathbf{LU} \tag{3.23}$$

The process of factoring **A** into the product **LU** is called L-U factorization. Once the L-U factorization is obtained, by whatever method, the equation

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{c} \tag{3.24}$$

is solved by transforming Eq. 3.24 into

$$\mathbf{Ly} = \mathbf{c} \tag{3.25}$$

and

$$\mathbf{Ux} = \mathbf{y} \tag{3.26}$$

Equation 3.25 is first solved for **y** and Eq. 3.26 is then solved for **x**. Since Eqs. 3.25 and 3.26 are both triangular systems of equations, the solutions are easily obtained by forward and backward substitution.

Crout's method calculates the elements of **L** and **U** recursively, without overwriting previous results.[2] To illustrate how Crout's method generates the elements of **L** and **U**, consider a matrix **A** of rank 4, requiring no row or column interchanges, i.e., no pivoting. The matrix **A** can be written as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{3.27}$$

Now an auxiliary matrix **B** can be defined, consisting of elements of **L** and **U**, such that

$$\mathbf{B} \equiv \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & u_{22} & u_{23} & u_{24} \\ l_{31} & l_{32} & u_{33} & u_{34} \\ l_{41} & l_{42} & l_{43} & u_{44} \end{bmatrix} \tag{3.28}$$

Elements of **B** are to be calculated one by one, in the order indicated as follows:

$$\begin{bmatrix} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} \\ \textcircled{5} & \textcircled{8} & \textcircled{9} & \textcircled{10} \\ \textcircled{6} & \textcircled{11} & \textcircled{13} & \textcircled{14} \\ \textcircled{7} & \textcircled{12} & \textcircled{15} & \textcircled{16} \end{bmatrix} \tag{3.29}$$

where $\textcircled{k}$ indicates the kth element to be calculated. The elements of **L** and **U** are calculated simply by equating a_{jk} successively, according to the order shown in Eq. 3.29, with the product of the jth row of **L** and the kth column of **U**. The Crout process for the

$n \times n$ matrix $\mathbf{A}$ is performed in $n - 1$ iterations. After $i - 1$ iterations matrix $\mathbf{A}$ finds the form

$$\begin{bmatrix} u_{11} & u_{12} & \cdot & \cdot & \cdot & u_{1n} \\ l_{21} & u_{22} & \cdot & \cdot & \cdot & u_{2n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \begin{bmatrix} & \mathbf{D}_i & \end{bmatrix} & & \\ l_{n1} & l_{n2} & \cdot & & & \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} i - 1 \text{ rows}$$

$$\underbrace{\qquad\qquad}_{i - 1 \text{ columns}} \tag{3.30}$$

where the conversion process of the first $i - 1$ rows and $i - 1$ columns has been completed. The $(n - i + 1) \times (n - i + 1)$ matrix in the lower right corner is denoted by $\mathbf{D}_i$. The elements of $\mathbf{D}_i$ are not the same as the elements of the original matrix $\mathbf{A}$. In the ith step, the Crout process converts matrix $\mathbf{D}_i$ to a new form:

$$\mathbf{D}_i \equiv \begin{bmatrix} d_{ii} & \mathbf{r}_i^T \\ \mathbf{s}_i & \mathbf{H}_{i+1} \end{bmatrix} \text{——Crout's process——} \longrightarrow \begin{bmatrix} u_{ii} & \mathbf{u}_i^T \\ l_i & \mathbf{D}_{i+1} \end{bmatrix} \tag{3.31}$$

where matrix $\mathbf{D}_{i+1}$ is one row and one column smaller than matrix $\mathbf{D}_i$. Crout's process can be stated as follows:

ALGORITHM LU-I

(a) Initially set an iteration counter $i = 1$. In the first step, matrix $\mathbf{D}_1 = \mathbf{A}$.

(b) Refer to the conversion formula of Eq. 3.31 and let

$$u_{ii} = d_{ii} \tag{3.32}$$

$$\mathbf{u}_i^T = \mathbf{r}_i^T \tag{3.33}$$

$$l_i = \frac{1}{u_{ii}} \mathbf{s}_i \tag{3.34}$$

$$\mathbf{D}_{i+1} = \mathbf{H}_{i+1} - l_i \mathbf{u}_i^T \tag{3.35}$$

(c) Increment i to $i + 1$. If $i = n$, L-U factorization is completed. Otherwise go to (b).

Note that calculation of element $\textcircled{k}$ of the auxiliary matrix $\mathbf{B}$, which is an element of either $\mathbf{L}$ or $\mathbf{U}$, involves only the element of $\mathbf{A}$ (or $\mathbf{D}_i$) in the same position and some elements of $\mathbf{B}$ that have already been calculated. As element $\textcircled{k}$ is obtained, it is recorded in the $\mathbf{B}$ matrix. In fact it may be recorded in the corresponding position in the $\mathbf{A}$ matrix, if there is no need to keep matrix $\mathbf{A}$. This calculated result need never be written over, since it is already one of the elements of $\mathbf{L}$ or $\mathbf{U}$.

Example 3.10

Apply L-U factorization to matrix $\mathbf{A}$, and then solve the set of algebraic equations $\mathbf{Ax} = \mathbf{c}$ for the unknown $\mathbf{x}$:

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 3 & -2 \\ -1 & 0 & -2 & -1 \\ 0 & 1 & 2 & 3 \\ 4 & 2 & 0 & -1 \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 0 \\ -1 \end{bmatrix}$$

Solution Following the LU-I algorithm, we have:

$i = 1$ $\mathbf{D}_1 = \mathbf{A}$ results in

$$d_{11} = 2 \qquad \mathbf{r}_1^T = [1 \quad 3 \quad -2]$$

$$\mathbf{s}_1 = \begin{bmatrix} -1 \\ 0 \\ 4 \end{bmatrix} \qquad \mathbf{H}_2 = \begin{bmatrix} 0 & -2 & -1 \\ 1 & 2 & 3 \\ 2 & 0 & -1 \end{bmatrix}$$

Then,

$$u_{11} = 2 \qquad \mathbf{u}_1^T = [1 \quad 3 \quad -2]$$

$$\boldsymbol{l}_1 = \begin{bmatrix} -0.5 \\ 0 \\ 2 \end{bmatrix}$$

$$\mathbf{D}_2 = \begin{bmatrix} 0 & -2 & -1 \\ 1 & 2 & 3 \\ 2 & 0 & -1 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 0 \\ 2 \end{bmatrix} [1 \quad 3 \quad -2] = \begin{bmatrix} 0.5 & -0.5 & -2 \\ 1 & 2 & 3 \\ 0 & -6 & 3 \end{bmatrix}$$

After the first iteration, matrix **A** becomes,

$$\begin{bmatrix} 2 & 1 & 3 & -2 \\ -0.5 & 0.5 & -0.5 & -2 \\ 0 & 1 & 2 & 3 \\ 2 & 0 & -6 & 3 \end{bmatrix}$$

$i = 2$ $\mathbf{D}_2 = \begin{bmatrix} 0.5 & -0.5 & -2 \\ 1 & 2 & 3 \\ 0 & -6 & 3 \end{bmatrix}$

results in

$$d_{22} = 0.5 \qquad \mathbf{r}_2^T = [-0.5 \quad -2]$$

$$\mathbf{s}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mathbf{H}_3 = \begin{bmatrix} 2 & 3 \\ -6 & 3 \end{bmatrix}$$

Then,

$$u_{22} = 0.5 \qquad \mathbf{u}_2^T = [-0.5 \quad -2]$$

$$\boldsymbol{l}_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \qquad \mathbf{D}_3 = \begin{bmatrix} 2 & 3 \\ -6 & 3 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix} [-0.5 \quad -2] = \begin{bmatrix} 3 & 7 \\ -6 & 3 \end{bmatrix}$$

After the second iteration, matrix **A** becomes,

$$\begin{bmatrix} 2 & 1 & 3 & -2 \\ -0.5 & 0.5 & -0.5 & -2 \\ 0 & 2 & 3 & 7 \\ 2 & 0 & -6 & 3 \end{bmatrix}$$

$i = 3 \qquad \mathbf{D}_3 = \begin{bmatrix} 3 & 7 \\ -6 & 3 \end{bmatrix}$

results in,

$$d_{33} = 3 \qquad \mathbf{r}_3^T = [7]$$
$$\mathbf{s}_3^T = [-6] \qquad \mathbf{H}_4 = [3]$$

Then,

$$u_{33} = 3 \qquad \mathbf{u}_3^T = [7]$$
$$l_3 = [-2] \qquad \mathbf{D}_4 = [3] - [-2][7] = [17]$$

Hence, matrix **A** becomes

$$\begin{bmatrix} 2 & 1 & 3 & -2 \\ -0.5 & 0.5 & -0.5 & -2 \\ 0 & 2 & 3 & 7 \\ 2 & 0 & -2 & 17 \end{bmatrix}$$

$i = 4$ Since $i = 4 = n$, the L-U factorization is completed. The **L** and **U** matrices are:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.5 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 2 & 0 & -2 & 1 \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} 2 & 1 & 3 & -2 \\ 0 & 0.5 & -0.5 & -2 \\ 0 & 0 & 3 & 7 \\ 0 & 0 & 0 & 17 \end{bmatrix}$$

The solution to $\mathbf{Ax} = \mathbf{c}$ is obtained by first solving $\mathbf{Ly} = \mathbf{c}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.5 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 2 & 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 0 \\ -1 \end{bmatrix} \longrightarrow \mathbf{y} = \begin{bmatrix} 1 \\ 3.5 \\ -7 \\ -17 \end{bmatrix}$$

Then, solving $\mathbf{Ux} = \mathbf{y}$ gives

$$\begin{bmatrix} 2 & 1 & 3 & -2 \\ 0 & 0.5 & -0.5 & -2 \\ 0 & 0 & 3 & 7 \\ 0 & 0 & 0 & 17 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 3.5 \\ -7 \\ -17 \end{bmatrix} \longrightarrow \mathbf{x} = \begin{bmatrix} -2 \\ 3 \\ 0 \\ -1 \end{bmatrix}$$

3.3.4 L-U Factorization with Pivoting

In the preceding subsection, the situation in which $u_{ii} = 0$, where Eq. 3.34 requires division by zero, is not discussed. In this case, partial or full pivoting must be applied. Since pivoting may change the order of the rows or columns of the matrix, this interchange information must be recorded in two additional permutation vectors.

Example 3.11

Apply L-U factorization with full pivoting to matrix **A**, and then solve the set of equations $\mathbf{Ax} = \mathbf{c}$:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & -1 & -3 \\ 4 & 1 & -1 & 1 \\ 2 & -3 & 0 & -1 \\ -1 & 0 & 5 & 2 \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} 0 \\ 4 \\ 11 \\ 1 \end{bmatrix}$$

Solution Two index vectors record the permutation of columns and rows during pivoting. These vectors are initialized to [1, 2, 3, 4] and $[1, 2, 3, 4]^T$. The pivot element (the largest element in absolute value) is moved to position d_{ii} at each step. The initial matrix is

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 2 & -1 & -3 \\ 4 & 1 & -1 & 1 \\ 2 & -3 & 0 & -1 \\ -1 & 0 & 5 & 2 \end{bmatrix} \end{array}$$

$i = 1$ The largest element (in absolute value) is moved to d_{11}:

$$\begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} -1 & 0 & 5 & 2 \\ 4 & 1 & -1 & 1 \\ 2 & -3 & 0 & -1 \\ 1 & 2 & -1 & -3 \end{bmatrix} \end{array} \rightarrow \begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 3 & 2 & 1 & 4 \end{array} \\ \begin{bmatrix} 5 & 0 & -1 & 2 \\ -1 & 1 & 4 & 1 \\ 0 & -3 & 2 & -1 \\ -1 & 2 & 1 & -3 \end{bmatrix} \end{array}$$

Crout's algorithm then yields

$$\begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 3 & 2 & 1 & 4 \end{array} \\ \begin{bmatrix} 5 & 0 & -1 & 2 \\ -\frac{1}{5} & 1 & \frac{19}{5} & \frac{7}{5} \\ 0 & -3 & 2 & -1 \\ -\frac{1}{5} & 2 & \frac{4}{5} & -\frac{13}{5} \end{bmatrix} \end{array}$$

$i = 2$ The largest element (in absolute value) in the 3×3 submatrix to the lower right is moved to d_{22}, to obtain

$$\begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 3 & 1 & 2 & 4 \end{array} \\ \begin{bmatrix} 5 & -1 & 0 & 2 \\ -\frac{1}{5} & \frac{19}{5} & 1 & \frac{7}{5} \\ 0 & 2 & -3 & -1 \\ -\frac{1}{5} & \frac{4}{5} & 2 & -\frac{13}{5} \end{bmatrix} \end{array}$$

Crout's algorithm then yields

$$\begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 3 & 1 & 2 & 4 \end{array} \\ \begin{bmatrix} 5 & -1 & 0 & 2 \\ -\frac{1}{5} & \frac{19}{5} & 1 & \frac{7}{5} \\ 0 & \frac{10}{19} & -\frac{67}{19} & -\frac{33}{19} \\ -\frac{1}{5} & \frac{4}{19} & \frac{34}{19} & -\frac{55}{19} \end{bmatrix} \end{array}$$

$i = 3$ The largest element (in absolute value) in the 2×2 matrix to the lower right is in d_{33}, so no interchange is needed. Crout's algorithm then yields

$$\begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 3 & 1 & 2 & 4 \end{array} \\ \begin{bmatrix} 5 & -1 & 0 & 2 \\ -\frac{1}{5} & \frac{19}{5} & 1 & \frac{7}{5} \\ 0 & \frac{10}{19} & -\frac{67}{19} & -\frac{33}{19} \\ -\frac{1}{5} & \frac{4}{19} & -\frac{34}{67} & -\frac{253}{67} \end{bmatrix} \end{array}$$

$i = 4$ Since $i = n$, the L-U factorization is completed.

Now, to solve $\mathbf{Ax} = \mathbf{c}$, first solve $\mathbf{Ly} = \mathbf{c}$:

$$\begin{bmatrix} 1 & & & \\ -\frac{1}{5} & 1 & & \\ 0 & \frac{10}{19} & 1 & \\ -\frac{1}{5} & \frac{4}{19} & -\frac{34}{67} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 11 \\ 0 \end{bmatrix}$$

The elements of vector $\mathbf{c}$ are interchanged according to the elements of the row index. Forward substitution yields $\mathbf{y} = [1, \frac{21}{5}, \frac{167}{19}, \frac{253}{67}]^T$. Then, solve $\mathbf{Ux} = \mathbf{y}$:

$$\begin{bmatrix} 5 & -1 & 0 & 2 \\ & \frac{19}{5} & 1 & \frac{7}{5} \\ & & -\frac{67}{19} & -\frac{33}{19} \\ & & & -\frac{253}{67} \end{bmatrix} \begin{bmatrix} x_3 \\ x_1 \\ x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{21}{5} \\ \frac{167}{19} \\ \frac{253}{67} \end{bmatrix}$$

The elements of vector $\mathbf{x}$ are interchanged according to the column index vector. A back substitution yields $\mathbf{x} = [2, -2, 1, -1]^T$.

3.3.5 Subroutines for Linear Algebraic Equations

Two FORTRAN subroutines for solving sets of linear algebraic equations, based on L-U factorization with partial pivoting, are presented in this section. The subroutines listed here are not unique. Other coding may be more compact, more general, or more efficient, but the subroutines show exactly what must be done. An overall understanding of these subroutines and their use makes it easier to understand other subroutines that follow in this text.

An explanation of the subroutines and a description of the variables appearing in the argument list of the subroutines are given here. To eliminate any redundancy of comments, these descriptions are not repeated in the FORTRAN listing of the subroutines.

Subroutine LU. This subroutine performs L-U factorization with partial pivoting on square matrices. The argument parameters in this subroutine are as follows:

A	The given N × N matrix. On return A contains **L** and **U** matrices.
ICOL	Integer N-vector containing the column indices.
N	Number of rows (columns) of matrix **A**.
EPS	Test value for deviation from zero due to round-off error.

The subroutine employs Crout's method with column pivoting on matrix A. Therefore, the columns of the matrix are generally interchanged at each elimination step to bring the largest element (in absolute value) to the pivot position. The interchange information is recorded in an integer permutation vector ICOL. The Kth column of the interchange matrix corresponds to the ICOL(K)th column in the original matrix A. Initially, the subroutine sets ICOL(K) = K, K = 1, . . . , N. During each pivot search, when the largest element (in absolute value) is found, it will be compared with a parameter EPS. This parameter, which must be assigned by the user, is used by the subroutine as the smallest (in magnitude) nonzero number in the computation. If the selected pivot element is smaller (in absolute value) than EPS, then the pivot element is considered to be zero; i.e., the matrix, within the specified error level, is singular. Therefore, the routine terminates the L-U factorization with an error message. For most practical problems on standard computers, a default value of 0.0001 is adequate for EPS. When L-U factorization is successfully completed, the matrix A will contain matrices **L** and **U**. A FORTRAN program for such a subroutine is as follows:

```
      SUBROUTINE LU (A,ICOL,N,EPS)
      DIMENSION A(N,N),ICOL(N)
      DO 10 K=1,N
10       ICOL(K)=K
      NM1=N-1
      DO 50 I=1,NM1
         PIV=ABS(A(I,I))
         IPIV=I
         IP1=I+1
         DO 20 K=IP1,N
            TEMP=ABS(A(I,K))
            IF (TEMP.LE.PIV) GO TO 20
            PIV=TEMP
            IPIV=K
```

```
 20     CONTINUE
        IF (PIV.LT.EPS) GO TO 60
        IF (IPIV.EQ.I) GO TO 40
        II=ICOL(I)
        ICOL(I)=ICOL(IPIV)
        ICOL(IPIV)=II
        DO 30 J=1,N
           TEMP=A(J,I)
           A(J,I)=A(J,IPIV)
 30        A(J,IPIV)=TEMP
 40     DO 50 J=IP1,N
           A(J,I)=A(J,I)/A(I,I)
           DO 50 K=IP1,N
              A(J,K)=A(J,K)-A(J,I)*A(I,K)
 50        CONTINUE
    RETURN
 60 WRITE(1,200)
    STOP
200 FORMAT(5X,'***THE MATRIX IS SINGULAR***')
    END
```

Subroutine LINEAR. This subroutine solves a set of linear equations in the form $\mathbf{Ax} = \mathbf{c}$ by calling subroutine LU to factorize matrix **A** into **L** and **U** matrices. The argument parameters in this subroutine are as follows:

A	The given N × N matrix, either as the original **A** matrix (ILU = 1), or in the form of LU (ILU = 0). In either case, on return, A contains the **L** and **U** matrices.
C	An N-vector containing the right side of the known quantities **c**, which upon return will contain the solution vector **x**.
W	An N-vector for work space.
ICOL	Integer N-vector, which upon return will contain the column indices.
N	Number of rows (columns) of matrix **A**.
ILU	An index that must be set to 0 or 1 by the calling program. If EQ. 0, **L** and **U** matrices are already available in A. If EQ. 1, **L** and **U** matrices are not available; i.e., this subroutine must call subroutine LU.
EPS	Test value for deviation from zero due to round-off error.

If L-U factorization must be employed on a matrix, then ILU must be set to 1 by the calling program. In this case, this subroutine will call subroutine LU to determine the **L** and **U** matrices and also the ICOL vector. However, if the **L** and **U** matrices and ICOL vector are already available, then ILU can be set to 0 by the calling program. This subroutine performs the steps of Eqs. 3.25 and 3.26 and stores the solution vector in the vector C. The dimensions of matrix A and vectors C, W, and ICOL must be set properly by the calling program. A FORTRAN program for such a subroutine is as follows:

```
      SUBROUTINE LINEAR (A,C,W,ICOL,N,ILU,EPS)
      DIMENSION A(N,N),C(N),W(N),ICOL(N)
      IF (ILU.GT.0) CALL LU (A,ICOL,N,EPS)
      DO 10 J=1,N
 10     W(J)=C(J)
```

```
C.....Perform forward elimination step. LY=C
      DO 30 J=2,N
         SUM=W(J)
         JM1=J-1
         DO 20 K=1,JM1
20          SUM=SUM-A(J,K)*W(K)
30       W(J)=SUM
C.....Perform back substitution step. UX=Y
      W(N)=W(N)/A(N,N)
      NP1=N+1
      DO 50 J=2,N
         I=NP1-J
         SUM=W(I)
         IP1=I+1
         DO 40 K=IP1,N
40          SUM=SUM-A(I,K)*W(K)
50       W(I)=SUM/A(I,I)
C.....Permute the solution vector to its original form
      DO 60 J=1,N
60       C(ICOL(J))=W(J)
      RETURN
      END
```

Example 3.12

Write a computer program, making use of subroutines LINEAR and LU, to solve a set of linear algebraic equations.

```
C*****      EXAMPLE 3.12     *****
      DIMENSION B(120),I(10)
      DATA EPS/0.0001/
C.....Read no. of rows (columns)
      WRITE(1,200)
      READ (1,*  ) N
C.....Pointers for subarrays
      N1=1
      N2=N1+N*N
      N3=N2+N
      NUSED=N3+N-1
C.....Perform L-U factorization: ILU=1
      ILU=1
      CALL SOLVE (B(N1),B(N2),B(N3),I,N,ILU,EPS)
      STOP
  200 FORMAT(5X,'ENTER N')
      END


      SUBROUTINE SOLVE (A,C,W,ICOL,N,ILU,EPS)
      DIMENSION A(N,N),C(N),W(N),ICOL(N)
C.....Read the matrix A row by row
      DO 10 J=1,N
         WRITE(1,200) J
   10    READ (1,*  ) (A(J,K),K=1,N)
C.....Read the right-hand-side vector C
   20 WRITE(1,210)
      READ (1,*  ) (C(J),J=1,N)
C.....Solve AX = C
      CALL LINEAR (A,C,W,ICOL,N,ILU,EPS)
C.....Report the solution vector
      WRITE(1,220) (C(J),J=1,N)
```

```
C.....Check for another vector C
      WRITE(1,230)
      READ (1,*  ) IC
      IF (IC.EQ.0) RETURN
      ILU=0
      GOTO 20
  200    FORMAT(5X,'ENTER ROW',I5)
  210 FORMAT(5X,'ENTER VECTOR C')
  220 FORMAT(5X,'THE SOLUTION IS',/,10F10.5)
  230 FORMAT(5X,'IF ANOTHER VECTOR C IS GOING TO BE GIVEN',/,
     +       7X,'THEN ENTER 1, OTHERWISE ENTER 0')
      END
```

Solution This program is written in a general form. It can accept up to 10 equations in 10 unknowns. If a set of more than 10 equations is to be solved, then the dimensions of the B and I arrays must be changed to 12N and N, respectively, where N is the number of equations.

Note how the B array is split into smaller subarrays. This technique will be used frequently in all of the programs in this text. For a set of equations $\mathbf{Ax} = \mathbf{c}$, the matrix **A** and vector **c** are entered. After the solution is obtained, the user may enter a different vector **c** to obtain a new solution. This process may be repeated as many times as needed.

3.4 NONLINEAR ALGEBRAIC EQUATIONS

One of the most frequently occurring problems in scientific work is to find the roots of one or a set of nonlinear algebraic equations of the form

$$\mathbf{\Phi(x)} = \mathbf{0} \tag{3.36}$$

i.e., zeros of the functions $\mathbf{\Phi(x)}$. The functions $\mathbf{\Phi(x)}$ may be given explicitly or as transcendental functions. Kinematic analysis of mechanical systems is one example for which solution of constraint equations of the form of Eq. 3.36 is required. In this case, the explicit form of the constraint equations is available.

Numerous methods are available to find the zeros of Eq. 3.36. However, depending on the application, some methods may have better convergence properties than others, and some may be more efficient. In either case, the methods are, in general, iterative. The most common and frequently used method is known as the Newton-Raphson method.

3.4.1 Newton-Raphson Method for One Equation in One Unknown

Consider the equation

$$\Phi(x) = 0 \tag{3.37}$$

to be nonlinear in the unknown x. The Newton-Raphson iteration is stated as

$$x^{j+1} = x^j - \frac{1}{\Phi_x(x^j)}\Phi(x^j) \tag{3.38}$$

where

$$\Phi_x(x^j) \equiv \frac{d\Phi(x)}{dx} \qquad \text{at} \qquad x = x^j \tag{3.39}$$

and the superscripts j and $j + 1$ are the iteration numbers. The Newton-Raphson algorithm produces a sequence of values, as follows:

$$x^1 = x^0 - \frac{\Phi(x^0)}{\Phi_x(x^0)}$$

$$x^2 = x^1 - \frac{\Phi(x^1)}{\Phi_x(x^1)}$$

$$\vdots$$

where x^0 is the initial estimate of the solution of Eq. 3.37. The sequence of values, in many problems, will approach a root of $\Phi(x)$.

The geometry of Newton-Raphson iteration is shown in Fig. 3.10. The Newton-Raphson method, when it works, is very efficient, but restrictions on the method are seldom discussed. The sketches in Figs. 3.11 through 3.13 show how the method may diverge or may converge to an unwanted solution. Since the Newton-Raphson method will not always converge, it is essential to terminate the process after a finite number of iterations.

3.4.2 Newton-Raphson Method for *n* Equations in *n* Unknowns

Consider n nonlinear algebraic equations in n unknowns,

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{0} \tag{3.40}$$

where a solution vector $\mathbf{x}$ is to be found. The Newton-Raphson algorithm for n equations is stated as

$$\mathbf{x}^{j+1} = \mathbf{x}^j - \boldsymbol{\Phi}_{\mathbf{x}}^{-1}(\mathbf{x}^j)\boldsymbol{\Phi}(\mathbf{x}^j) \tag{3.41}$$

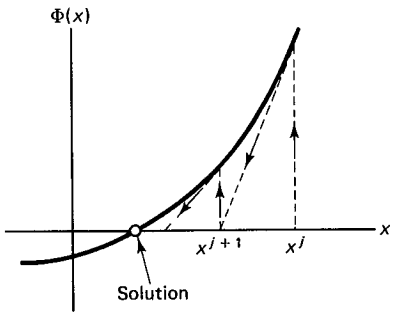


Figure 3.10 Graphic representation of Newton-Raphson method.

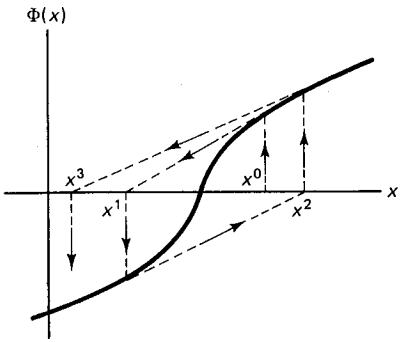


Figure 3.11 Root at infection point.

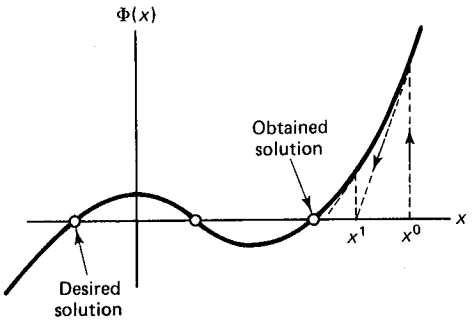


Figure 3.12 Multiple roots.

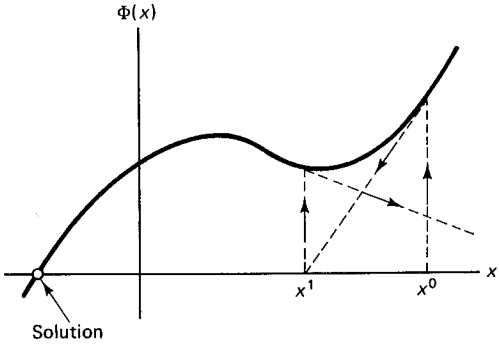


Figure 3.13 Divergence near a local minimum or maximum.

where $\mathbf{\Phi}_{\mathbf{x}}^{-1}(\mathbf{x}^j)$ is the *inverse* of the Jacobian matrix evaluated at $\mathbf{x} = \mathbf{x}^j$. Equation 3.41 can be identified as the n-dimensional version of Eq. 3.38. The term $\mathbf{\Phi}(\mathbf{x}^j)$ on the right side of Eq. 3.41 is known as the vector of *residuals*, which corresponds to the *violation* in the equations.

Equation 3.41 may be restated as a two-step operation:

$$\mathbf{\Phi}_{\mathbf{x}}(\mathbf{x}^j)\,\Delta\mathbf{x}^j = -\mathbf{\Phi}(\mathbf{x}^j) \tag{3.42}$$

$$\mathbf{x}^{j+1} = \mathbf{x}^j + \Delta\mathbf{x}^j \tag{3.43}$$

where Eq. 3.42, which is a set of n linear equations, is solved for $\Delta\mathbf{x}^j$. Then, $\mathbf{x}^{j+1}$ is evaluated from Eq. 3.43. Gaussian elimination or L-U factorization methods are frequently employed to solve Eq. 3.42. The term $\Delta\mathbf{x}^j = \mathbf{x}^{j+1} - \mathbf{x}^j$, known as the *Newton difference*, shows the amount of correction to the approximated solution in the jth iteration. The computational procedure is stated as follows:

ALGORITHM NR-I

(a) Set the iteration counter $j = 0$.

(b) An initial estimate $\mathbf{x}^0$ is made for the desired solution.

(c) The functions $\mathbf{\Phi}(\mathbf{x}^j)$ are evaluated. If the magnitudes of all of the residuals $\Phi_i(\mathbf{x}^j)$, $i = 1, \ldots, n$, are less than a specified tolerance ε, i.e., if $|\Phi_i| < \varepsilon$, $i = 1, \ldots, n$, then $\mathbf{x}^j$ is the desired solution; therefore terminate. Otherwise, go to (d).

(d) Evaluate the Jacobian matrix $\mathbf{\Phi}_{\mathbf{x}}(\mathbf{x}^j)$ and solve Eqs. 3.42 and 3.43 for $\mathbf{x}^{j+1}$.

(e) Increment j; i.e., set j to $j + 1$. If j is greater than a specified allowed number of iterations, then stop. Otherwise go to (c).

Algorithm NR-I is stated for the Newton-Raphson method in its simplest form. There are numerous techniques that can be included in the algorithm to improve its convergence. These techniques are not discussed in this text. Interested readers are referred to textbooks on numerical analysis.

Example 3.13

Figure 3.14 shows a disk that is pressed against a plane surface that passes through point A. Apply the Newton-Raphson method to find Φ_2 and d when $\Phi_1 = 30°$.

Solution Two constraint equations may be written, as follows:

$$\begin{aligned} \Phi_1 &\equiv b \cos \phi_1 + a \cos \phi_2 - d = 0 \\ \Phi_2 &\equiv b \sin \phi_1 + a \sin \phi_2 - r = 0 \end{aligned} \tag{1}$$

In order to analyze this system using the coordinate partitioning method of Sec. 3.2.1, the dependent and independent coordinates are taken to be $\mathbf{u} \equiv [\Phi_2, d]^T$, and $\mathbf{v} \equiv [\Phi_1]$. Hence, Eq. 1 may be rewritten as

$$\begin{aligned} \Phi_1 &= 2 \cos \phi_2 - d + 10 \cos \phi_1 = 0 \\ \Phi_2 &= 2 \sin \phi_2 - 4 + 10 \sin \phi_1 = 0 \end{aligned} \tag{2}$$

The Jacobian matrix for this system is

$$\mathbf{\Phi}_\mathbf{q} \equiv \begin{bmatrix} -2 \sin \phi_2 & -1 & -10 \sin \phi_1 \\ 2 \cos \phi_2 & 0 & 10 \cos \phi_1 \end{bmatrix}$$

where $\mathbf{q} = [\phi_2, d, \phi_1]^T$, or

$$\mathbf{\Phi}_\mathbf{u} \equiv \begin{bmatrix} -2 \sin \phi_2 & -1 \\ 2 \cos \phi_2 & 0 \end{bmatrix} \tag{3}$$

and

$$\mathbf{\Phi}_\mathbf{v} \equiv \begin{bmatrix} -10 \sin \phi_1 \\ 10 \cos \phi_1 \end{bmatrix} \tag{4}$$

For the Newton-Raphson algorithm, $\Delta\mathbf{u}$ is evaluated by solving

$$\begin{bmatrix} -2 \sin \phi_2 & -1 \\ 2 \cos \phi_2 & 0 \end{bmatrix} \begin{bmatrix} \Delta\phi_2 \\ \Delta d \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ -\Phi_2 \end{bmatrix} \tag{5}$$

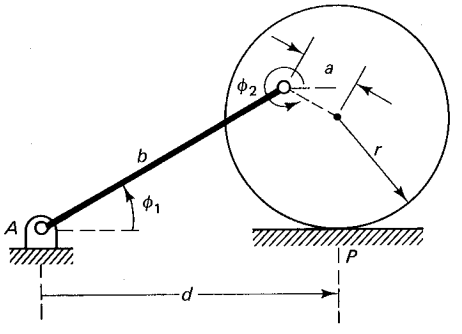


Figure 3.14 Roll with slip (Example 3.13).

Since Eq. 5 is simple to solve, there is no need to apply a numerical technique. It is found that

$$\Delta\phi_2 = \frac{-\Phi_2}{2\cos\phi_2}$$
$$\Delta d = \Phi_1 + \Phi_2 \tan\phi_2 \tag{6}$$

Iterative results using this formula are presented in the accompanying table, where $\phi_1 = 30°$. After three iterations, $\phi_2 = 5.76$ rad (330°) and $d = 10.3924$, and the residuals are $\Phi_1 = -0.0003$ and $\Phi_2 = 0.00007$, which are small enough to terminate the process. Note that, since $\Delta\phi_2$ is found in radians from Eq. 6, then either ϕ_2 must be converted to radians (from degrees), or $\Delta\phi_2$ must be converted to degrees (from radians).

Iteration number	$\phi_2^{\dagger}$	d	Φ_1	Φ_2	$\Delta\phi_2$	Δd
1	5.59 (326°)	10.0	0.19	−0.29	0.19	0.43
2	5.77	10.43	0.026	0.0206	−0.0118	−0.03759
3	5.76	10.3924	−0.0003	0.00007		

[†]All angles should be in radians.

3.4.3 A Subroutine for Nonlinear Algebraic Equations

The Newton-Raphson algorithm of Section 3.4.2 is represented here in the form of a subroutine that can be embedded in programs to solve sets of nonlinear algebraic equations. This subroutine makes use of subroutines LINEAR and LU to solve Eqs. 3.42 and 3.43 iteratively. This subroutine is written in the simplest possible form, which can be modified easily.

Subroutine NEWTON. The argument parameters in this subroutine are as follows:

A, C, W, ICOL, N, EPS	Refer to subroutine LINEAR
NRMAX	Maximum number of iterations allowed
FEPS	Error tolerance on the Newton differences
X	Vector of dependent coordinates (unknowns)

This subroutine allows a maximum of NRMAX Newton iterations for finding the solution vector X to the set of nonlinear algebraic equations. If the solution is not found in NRMAX iterations, the subroutine terminates the process with a CONVERGENCE FAILED message. In each iteration, a call to a *user-supplied* subroutine FUNCT is made. This subroutine must provide the Jacobian matrix $\mathbf{\Phi}_x$ and the constraint violations

Φ (refer to Eq. 3.41), for the particular problem at hand, in matrix A and array C, respectively. Equation 3.42 is then solved by a call to subroutine LINEAR. The Newton differences $\Delta\mathbf{x}$ are returned in array C, which is employed to correct the vector of unknown X. Note that Eqs. 3.42 and 3.43 can be written as

$$\Phi_{\mathbf{x}}(\mathbf{x}^j)\,\Delta\mathbf{x}^j = \Phi(\mathbf{x}^j) \tag{3.44}$$

and

$$\mathbf{x}^{j+1} = \mathbf{x}^j - \Delta\mathbf{x}^j \tag{3.45}$$

which are employed in this subroutine. Finally, if all of the Newton differences (in absolute value) that are stored in C are less than the specified tolerance FEPS, the subroutine successfully terminates the process by returning to the calling routines. A FORTRAN program for such a subroutine is as follows:

```
      SUBROUTINE NEWTON (A,C,W,ICOL,N,EPS,NRMAX,FEPS,X)
      DIMENSION A(N,N),C(N),W(N),ICOL(N),X(N)
      DO 20 I=1,NRMAX
         CALL FUNCT (A,C,N)
         CALL LINEAR (A,C,W,ICOL,N,1,EPS)
         ICONVR=0
         DO 10 J=1,N
            IF (ABS(C(J)).GT.FEPS) ICONVR=1
   10       X(J)=X(J)-C(J)
         IF (ICONVR) 30,30,20
   20 CONTINUE
      WRITE(1,200)
      STOP
   30 RETURN
  200 FORMAT(5X,'***CONVERGENCE FAILED***')
      END
```

Example 3.14

Write a computer program, making use of subroutine NEWTON, to solve the problem of Example 3.13.

Solution The main routine presented here for this example is written in a general form. It can handle problems with up to 10 independent variables **v** and 10 dependent variables **u**, without the need to increase the dimension of the arrays. The program asks the user for the following information:

Number of independent variables
Number of dependent variables
Known values for the independent variables
Initial estimates for the dependent variables

The main routine calls subroutine NEWTON, which in turn calls subroutines LINEAR (and LU) and FUNCT.

The constraint equations of Eq. 1 and the Jacobian matrix $\Phi_{\mathbf{u}}$ of Eq. 3 are formulated in subroutine FUNCT. In this subroutine, the array F, which has two elements, contains the constraint violations, and the 2 × 2 array (matrix) A contains the Jacobian entries.

```
C*****      EXAMPLE 3.14      *****
      COMMON /EXAMPL/ U(10),V(10)
      DIMENSION B(120),I(10)
      DATA EPS/0.0001/, FEPS/0.001/, NRMAX/25/
C.....Initialize the variables
      WRITE(1,200)
      READ (1,*  ) NV,N
      IF (NV.EQ.0) GOTO 10
      WRITE(1,210)
      READ (1,*  ) (V(J),J=1,NV)
   10 WRITE(1,220)
      READ (1,*  ) (U(J),J=1,N)
C.....Pointers for the subarrays
      N1=1
      N2=N1+N*N
      N3=N2+N
      NUSED=N3+N-1
C.....Perform Newton-Raphson iteration
      CALL NEWTON (B(N1),B(N2),B(N3),I,N,EPS,NRMAX,FEPS,U)
      WRITE(1,230) (U(J),J=1,N)
      STOP
  200 FORMAT(5X,'ENTER NO. OF INDEPENDENT VARIABLES V',/,
     +       8X,'AND NO. OF DEPENDENT VARIABLES U')
  210 FORMAT(5X,'ENTER VALUES FOR THE INDEPENDENT VARIABLES')
  220 FORMAT(5X,'ENTER ESTIMATES FOR THE DEPENDENT VARIABLES')
  230 FORMAT('THE SOLUTION TO DEPENDENT VARIABLES IS:',/,10F10.5)
      END

      SUBROUTINE FUNCT (A,F,N)
      COMMON /EXAMPL/ U(10),V(10)
      DIMENSION A(N,N),F(N)
C*****      EXAMPLE 3.13      *****
C.....Constraint equations
        F(1) = 2.0*COS(U(1))-U(2)+10.0*COS(V(1))
        F(2) = 2.0*SIN(U(1))-4.0 +10.0*SIN(V(1))
C.....Jacobian matrix
        A(1,1) =-2.0*SIN(U(1))
        A(1,2) =-1.0
        A(2,1) = 2.0*COS(U(1))
        A(2,2) = 0.0
      RETURN
      END
```

PROBLEMS

3.1 For each of the planar mechanical systems shown in Fig. P.3.1, answer the following questions:

(a) Is the system an open loop or a closed loop?

(b) If the system contains any closed loops, identify all of the closed loops.

(c) Determine the number of degrees of freedom of the system.

(d) Identify all of the kinematic joints.

Note: For mechanisms (b) and (i) consider two cases: where the wheel(s) do(es) not slip, or where the wheel(s) slip(s).

3.2 Determine which of the following constraint equations are nonholonomic:

(a) $2 \cos \phi_1 + 3.6 d_2 \cos \phi_3 - 3.1 = 0$

(b) $x_4 - 3 \cos \phi_4 - x_6 + 2.5 \sin \phi_6 = 0$

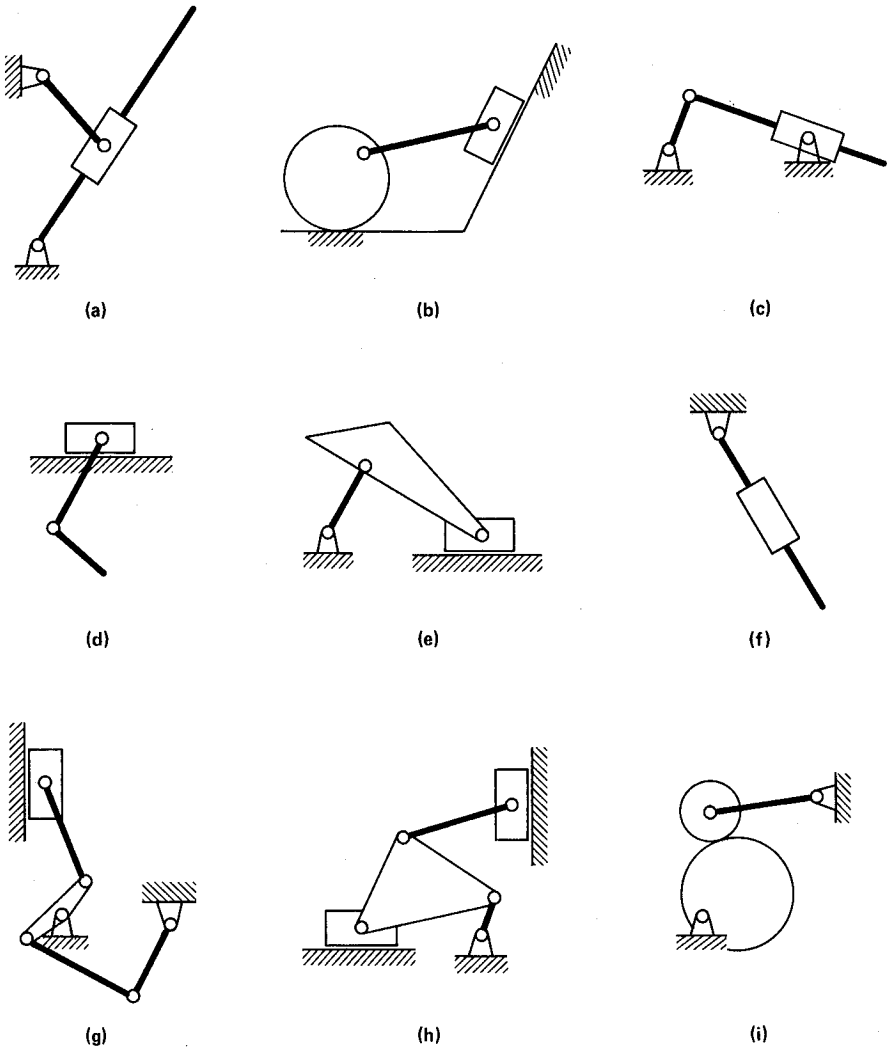


Figure P.3.1

(c) $\dot{x}_2 + 0.6t + 0.1 = 0$

(d) $x_2\dot{y}_3 + y_3\dot{x}_2 = 0$

(e) $(x_1 - x_3)^2 + (y_1 - y_3)^2 - d^2 > 0$

(f) $-e_1\dot{e}_0 + e_0\dot{e}_1 - e_3\dot{e}_2 + e_2\dot{e}_3 = 0$

3.3 For the slider-crank mechanism shown in Fig. P.3.3, assume $l_1 = 1.2$ and $l_2 = 2.6$.

(a) Write the constraint equations in terms of the coordinates ϕ_1, ϕ_2, and d. From these equations, derive the velocity and acceleration equations.

(b) If ϕ_1 is the driving coordinate, then for $\phi_1 = 0.8$ rad, $\dot{\phi}_1 = 0.1$ rad/s, and $\ddot{\phi}_1 = 0$, find the remaining coordinates, velocities, and accelerations.

(c) If the slider is the driving link, then for $d = 2.5$, $\dot{d} = -0.2$, and $\ddot{d} = -0.06$, find the remaining coordinates, velocities, and accelerations.

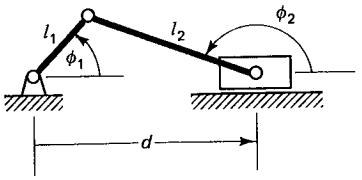


Figure P.3.3

3.4 Points P and Q are defined on the coupler of a four-bar mechanism as shown in Fig. P.3.4. Assume $l_1 = 0.5$, $l_2 = 1.2$, $l_3 = 0.8$, $l_4 = 0.7$, $l_5 = 0.15$, and $l_6 = 0.2$.

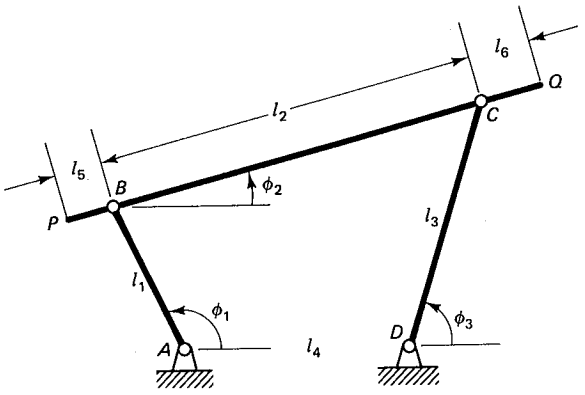


Figure P.3.4

(a) Derive the constraint, velocity, and acceleration equations (loop $ABCD$).

(b) Write expressions for x^P, y^P, x^Q, and y^Q in terms of the coordinates.

(c) From (b), derive expressions for the velocity and acceleration of P and Q.

(d) For $\phi_1 = \pi/4$, $\dot{\phi}_1 = -0.1\pi$, and $\ddot{\phi}_1 = 0$, solve the equations obtained in (a) to find the remaining coordinates, velocities, and accelerations.

(e) Use the results of (d) and substitute in the expressions of (b) and (c) to obtain the coordinates, velocities, and accelerations of P and Q.

3.5 Select some of the mechanical systems of Prob. 3.1 having only closed loops, define a proper set of coordinates, and then derive the constraint, velocity, and acceleration equations.

3.6 Assume that y is the independent coordinate in the constraint equations

$$x^2 + xy + yz - 2x + 3 = 0$$
$$y^2 - 3z^2 + xz + 2y - 1 = 0$$

Apply two different approaches to find the velocity and acceleration equations for the coordinate partitioning method:

(a) Take the first and the second time derivatives of the constraints.

(b) Employ Eqs. 3.8 and 3.11 directly.

3.7 Write the constraint, velocity, and acceleration equations for Prob. 3.6 in the form of the appended driving constraint method. Assume that the independent variable is defined as $y = c_1$, $\dot{y} = c_2$, and $\ddot{y} = c_3$. Express these equations in the form of Eqs. 3.12, 3.14, and 3.16.

3.8 A driver constraint equation for the slider-crank mechanism in Prob. 3.3 is stated as

$$\phi_1 - 0.8 - 0.1t = 0$$

Write the constraint, velocity, and acceleration equations in the form of the appended driving constraint formulation. Solve these equations for $t = 0$ and compare the result with that obtained in Prob. 3.3(b).

3.9 Solve the system of equations

$$\begin{bmatrix} 2 & -1 & 0 & 1 \\ -2 & 2 & 0 & -3 \\ 4 & 1 & 3 & -1 \\ 0 & -1 & -2 & 5 \end{bmatrix} \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \\ 1 \\ 2 \end{bmatrix}$$

by the following methods:

(a) Gaussian elimination

(b) Gauss-Jordan reduction

(c) L-U factorization

3.10 Solve the system of equations

$$\begin{bmatrix} 3 & 1 & -1 & 2 \\ -6 & -2 & 4 & 3 \\ 0 & 3 & 2 & -2 \\ 1 & 1 & -5 & 6 \end{bmatrix} \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 1 \\ -10 \end{bmatrix}$$

by the Gaussian elimination method

(a) Without pivoting

(b) With row pivoting when necessary

(c) With column pivoting when necessary

3.11 Solve the system of equations in Prob. 3.9 by L-U factorization and

(a) No pivoting

(b) Partial pivoting with row interchange

(c) Partial pivoting with column interchange

(d) Full pivoting

3.12 For the system shown in Fig. P.3.12, assume no slipping between the wheel and the contacting surface. If $OA = 1.5$, $AB = 2$, $BC = 1$, $r = 1.2$, and $OE = 0.2$, write the constraint equations for this system in terms of ϕ_1, ϕ_2, ϕ_3, and d. Note that two equations can be written for the loop closure and one equation can be written for the no-slip condition. Furthermore, it is known that the system was initially assembled for $d = 3$ and $\phi_3 = 120°$. For the crank angle $\phi_1 = 60°$, apply the Newton-Raphson algorithm to solve the constraint equations for the unknown coordinates. Show the result of each iteration in a table.

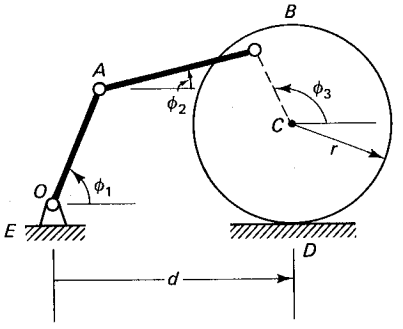


Figure P.3.12

3.13 For the quick-return mechanism shown in Fig. P.3.13, derive the constraint, velocity, and acceleration equations. Solve these equations by writing a computer program using subroutines NEWTON and LINEAR. Assume $OB = 0.7$, $CD = 2.1$, $OE = 0.9$, and the configuration is for $\phi_1 = \pi/6$, $\dot{\phi}_1 = -0.2$, and $\ddot{\phi}_1 = 0$.

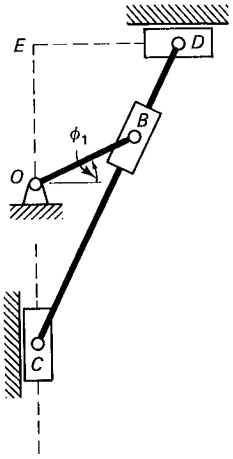


Figure P.3.13

3.14 For the mechanism of Prob. 3.13 assume a driver constraint as

$$\phi_1 - \frac{\pi}{6} + 0.2t = 0$$

Start from $t = 0$ and increment t gradually to simulate the motion of the system for a complete cycle.

3.15 Modify the computer program of Prob. 3.13 to solve Prob. 3.12. Assume that the crank rotates with a constant angular velocity of 0.2 rad/s.

3.16 The four-bar mechanism shown in Fig. P.3.16 is used to advance a film strip inside a movie projector.[17]

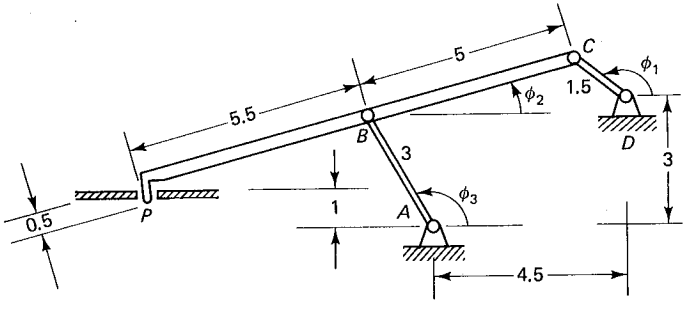


Figure P.3.16

(a) Write the constraint equations for the four-bar linkage *ABCD*.

(b) Write expressions for the coordinates of point P in terms of ϕ_1, ϕ_2, and ϕ_3.

(c) Develop a computer program to solve the constraint equations for a complete revolution of the crank and compute the coordinates of P.

(d) Plot the path of P and show where point P becomes engaged and disengaged with the film strip.