13 Numerical Methods in Dynamics

In this chapter several algorithms for the numerical solution of the equations of motion are presented. These algorithms utilize the numerical methods given in Chap. 12 for solving ordinary differential equations. If a mechanical system does not have any kinematic joints, i.e., if it is an unconstrained system, then these algorithms can be employed directly. However, if a mechanical system contains kinematic joints, and if Cartesian coordinates are employed in deriving the equations of motion, then these numerical integration algorithms must be modified.

The techniques and algorithms that are discussed in this chapter can be applied to solve the equations of motion when they are derived either in Cartesian coordinates or in other coordinate systems, such as Lagrangian coordinates. If the Lagrangian coordinates describing the configuration of a system are the generalized coordinates (i.e., if the number of Lagrangian coordinates is equal to the number of degrees of freedom), then the equations of motion are ordinary differential equations with no algebraic constraints, regardless of the presence or the absence of any kinematic joints. If the number of Lagrangian coordinates is greater than the number of degrees of freedom, then the equations of motion are mixed algebraic-differential equations. This is the same type of equation as in the case of Cartesian coordinates for systems containing kinematic joints.

13.1 INTEGRATION ARRAYS

A numerical solution to the equations of motion may be obtained by utilizing any commonly used numerical integration algorithm. These algorithms are useful in solving first-order differential equations that take the form

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t) \tag{13.1}$$

If there are n second-order differential equations of motion, they can be converted to $2n$ first-order equations by defining the $\mathbf{y}$ and $\dot{\mathbf{y}}$ arrays as follows:

$$\mathbf{y} = \begin{bmatrix} \text{Position coordinates} \\ \text{Velocities} \end{bmatrix}, \qquad \dot{\mathbf{y}} = \begin{bmatrix} \text{Velocities} \\ \text{Accelerations} \end{bmatrix} \tag{13.2}$$

Although the arrangement of the elements in $\mathbf{y}$ and $\dot{\mathbf{y}}$ is quite arbitrary, the two arrays must follow a similar order. For example, if the jth element of $\mathbf{y}$ contains x_i, then the jth element of $\dot{\mathbf{y}}$ must contain $\dot{x}_i$.

The process of numerical integration at time $t = t^i$ can be interpreted by the following diagram:

$$\dot{\mathbf{y}}(t^i) \xrightarrow{\text{(integration)}} \mathbf{y}(t^i + \Delta t) \tag{13.3}$$

In other words, velocities and accelerations at $t = t^i$ yield coordinates and velocities at $t = t^i + \Delta t$.

13.2 KINEMATICALLY UNCONSTRAINED SYSTEMS

The equations of motion for b unconstrained bodies containing n coordinates are represented by

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{g} \tag{13.4}$$

If there are no mathematical constraints[†] on the coordinates, then the number of degrees of freedom is also n. A numerical solution to Eq. 13.4 can be found in the same manner as that shown in Example 12.2. In the following algorithm, called the direct integration algorithm (DI), arrays $\mathbf{y}$ and $\dot{\mathbf{y}}$ are defined as follows:

$$\mathbf{y} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} \qquad \dot{\mathbf{y}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix}$$

ALGORITHM DI-1

(a) Main routine

(a.1) Specify initial conditions for $\mathbf{q}$ and $\dot{\mathbf{q}}$.

(a.2) Transfer the contents of $\mathbf{q}$ and $\dot{\mathbf{q}}$ to vector $\mathbf{y} \equiv [\mathbf{q}^T, \dot{\mathbf{q}}^T]^T$.

(a.3) Enter the numerical integration routine (NI).

(b) Numerical integration routine

(This routine solves initial-value problems of the form $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$ from an initial time t^0 to a final time t^e)

•

•

•

[†]For an unconstrained system of bodies in spatial motion, there are no kinematic constraints; however, there is one *mathematical constraint* for each set of Euler parameters.

(b.1) In the process of numerical integration, $\mathbf{f}(\mathbf{y}, t)$ must be evaluated. For this purpose enter a DIFEQN routine with known $\mathbf{y}^i$ and t^i to determine $\mathbf{f}(\mathbf{y}^i, t^i)$.

.

.

.

(c) DIFEQN routine

- **(c.1)** Transfer the contents of $\mathbf{y}$ to $\mathbf{q}$ and $\dot{\mathbf{q}}$.
- **(c.2)** Evaluate $\mathbf{M}$ (since $\mathbf{M}$ is constant, it needs to be evaluated only once) and $\mathbf{g}$.
- **(c.3)** Solve Eq. 13.4 for $\ddot{\mathbf{q}}$.
- **(c.4)** Transfer the contents of $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ to $\dot{\mathbf{y}}$.
- **(c.5)** Return.

During an integration time step, the routine DIFEQN is called several times. The contents of $\mathbf{y}$ are changed automatically by the integration routine, according to the algorithm. For example, in the Runge-Kutta subroutine of Sec. 12.2.2, the subroutine DIFEQN is called four times in every integration time step. The arrays $\mathbf{y}$ and $\dot{\mathbf{y}}$ are named Y and F, respectively, in that subroutine.

13.2.1 Mathematical Constraints

A kinematically unconstrained system may be represented by a set of dependent coordinates. This situation exists when Euler parameters are employed as rotational coordinates.

The complete set of equations of motion is written, from Eq. 11.25, as

$$\mathbf{p}_i^T\mathbf{p}_i - 1 = 0 \qquad i = 1, \ldots, b \tag{13.5}$$

$$\dot{\mathbf{p}}_i^T\mathbf{p}_i = 0 \qquad i = 1, \ldots, b \tag{13.6}$$

$$\begin{bmatrix} \mathbf{M}^* & \mathbf{P}^T \\ \mathbf{P} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\sigma} \end{bmatrix} = \begin{bmatrix} \mathbf{g}^* - \mathbf{b}^* \\ -\mathbf{c} \end{bmatrix} \tag{13.7}$$

An algorithm for solving Eqs. 13.5 through 13.7 can be developed by a slight modification to algorithm DI-1:

ALGORITHM DI-2

(a) and **(b)** the same as for DI-1.

(c) DIFEQN routine

- **(c.1)** Transfer $\mathbf{y}$ to $\mathbf{q}$ and $\dot{\mathbf{q}}$.
- **(c.2)** Evaluate $\mathbf{M}^*$, $\mathbf{P}$, $\mathbf{g}^*$, $\mathbf{b}^*$, and $\mathbf{c}$.
- **(c.3)** Solve Eq. 13.7 for $\ddot{\mathbf{q}}$ and $\boldsymbol{\sigma}$.
- **(c.4)** Transfer $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ to $\dot{\mathbf{y}}$.
- **(c.5)** Return.

In this algorithm, the artificial Lagrange multipliers $\boldsymbol{\sigma}$ are evaluated as a by-product when Eq. 13.7 is solved. This algorithm requires correct initial conditions on $\mathbf{p}_i$

and $\dot{\mathbf{p}}_i$, $i = 1, \ldots, b$. Since a numerical integration algorithm yields only an approximate solution to the exact response, the computed values for $\mathbf{p}_i$ and $\dot{\mathbf{p}}_i$ may contain some numerical error after several time steps. Therefore, Eqs. 13.5 and 13.6 may no longer be satisfied. If the accumulation of the error is not corrected or controlled, erroneous results may be obtained. Two methods for correcting the numerical error are discussed in the following.

Method 1. The numerically integrated values for the Euler parameters of body i at any time step are denoted by $\mathbf{p}_i^*$, which may not satisfy Eq. 13.5; i.e., it may be that

$$\mathbf{p}_i^{*T}\mathbf{p}_i^* - 1 = \delta \tag{13.8}$$

where δ represents the violation in the constraint. In this case, the transformation matrix $\mathbf{A}_i^*$, calculated in terms of $\mathbf{p}_i^*$, will lose the orthogonality condition; i.e., the result will be that

$$\mathbf{A}_i^{*T}\mathbf{A}_i^* \neq \mathbf{I}$$

A correction in $\mathbf{p}_i^*$ by $\boldsymbol{\varepsilon}$ can be found to yield a corrected set of Euler parameters, as follows:

$$\mathbf{p}_i = \mathbf{p}_i^* + \boldsymbol{\varepsilon}$$

which will satisfy Eq. 13.5. An infinite number of $\boldsymbol{\varepsilon}$ vectors can be found for this purpose.

A popular method for evaluating the *best* set of $\boldsymbol{\varepsilon}$ vectors is to minimize the sum of squares of the elements of $\boldsymbol{\varepsilon}$ as follows:

$$\begin{aligned} &\text{Minimize } f_1 = \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} \\ &\quad \text{subject to the constraints of Eq. 13.5} \end{aligned} \tag{a}$$

or[†]

$$\text{Minimize } f_2 = \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} + (\mathbf{p}_i^T\mathbf{p}_i - 1)\lambda$$

This yields:

$$\left[\frac{\partial f_2}{\partial \mathbf{p}_i}\right]^T \equiv 2(\mathbf{p}_i - \mathbf{p}_i^*) + 2\mathbf{p}_i\lambda = 0$$

or

$$\mathbf{p}^* = (1 + \lambda)\mathbf{p}$$

Substitution of this equation in Eq. 13.8 gives $(1 + \lambda)^2 = 1 + \delta$ which results in

$$\mathbf{p} = \frac{1}{\sqrt{1 + \delta}}\mathbf{p}^* \tag{13.9}$$

This is the correction formula for the Euler parameters.[‡] All four parameters are normalized by the same quantity, and in such a way that only the angle of rotation ϕ is affected, not the direction $\vec{u}$ of the orientational axis of rotation.

[†]In constrained optimization techniques, the constraint equation(s) can be included in the objective function by the use of Lagrange multipliers.

[‡]The selection of Eq. a as the objective function for the optimization process is rather arbitrary. If other objective functions are selected, different correction formulas are obtained. Somewhat different formula can be found in Ref. 20.

The numerical integration error may also yield numerical values for the violation of Eq. 13.6 by the elements of $\dot{\mathbf{p}}_i$:

$$\dot{\mathbf{p}}_i^{*T}\mathbf{p}_i = \sigma \tag{13.10}$$

A process similar to the preceding minimization process gives a correction formula for the velocities:

$$\dot{\mathbf{p}}_i = \dot{\mathbf{p}}_i^* - \sigma\mathbf{p}_i \tag{13.11}$$

The correction formulas of Eqs. 13.9 and 13.11 can be included in the algorithm DI-2, step c.1.

Method II. The constraints of Eqs. 13.5 and 13.6 can be treated in much ti e same way as the kinematic constraint equations. This subject is discussed later in this chapter.

13.2.2 Using Angular Velocities

If the equations of motion are taken in the form given by Eq. 11.37, a considerable amount of computational efficiency can be gained. In this case, vectors $\mathbf{y}$ and $\dot{\mathbf{y}}$ are defined as follows:

$$\mathbf{y} = \begin{bmatrix}\mathbf{q}\\ \mathbf{h}\end{bmatrix} \qquad \dot{\mathbf{y}} = \begin{bmatrix}\dot{\mathbf{q}}\\ \dot{\mathbf{h}}\end{bmatrix}$$

The dimension of $\mathbf{y}$ or $\dot{\mathbf{y}}$ is $13 \times b$, so that each contains b fewer elements than the arrays of Sec. 13.2. The integration of the velocities and accelerations of a typical body i is performed according to the following diagram:

$$\begin{bmatrix}\dot{\mathbf{r}}_i\\ \dot{\mathbf{p}}_i\\ \ddot{\mathbf{r}}_i\\ \dot{\boldsymbol{\omega}}_i'\end{bmatrix} \xrightarrow{\text{(integration)}} \begin{bmatrix}\mathbf{r}_i\\ \mathbf{p}_i\\ \dot{\mathbf{r}}_i\\ \boldsymbol{\omega}_i'\end{bmatrix}$$

The computed values for $\mathbf{p}_i$ and $\boldsymbol{\omega}_i'$ are employed in Eq. 6.109 to find $\dot{\mathbf{p}}_i$.

An algorithm for dynamic analysis using Eq. 11.37 can be stated by a slight modification to algorithm DI-1:

ALGORITHM DI-3

(a) Main routine
- **(a.1)** Specify initial condition for $\mathbf{q}$ and $\mathbf{h}$.
- **(a.2)** Define vector $\mathbf{y}$ as $\mathbf{y} = [\mathbf{q}^T, \mathbf{h}^T]^T$.
- **(a.3)** Enter the numerical integration routine (NI).

(b) Numerical integration routine
(Same as DI-1)

(c) DIFEQN routine
- **(c.1)** Transfer the translational coordinates and velocities $\mathbf{r}_i$ and $\dot{\mathbf{r}}_i$, $i = 1, \ldots, b$, from $\mathbf{y}$ to $\mathbf{q}$ and $\dot{\mathbf{q}}$. Transfer $\mathbf{p}_i$, $i = 1, \ldots, b$, from $\mathbf{y}$ to $\mathbf{q}$ after correcting for the numerical error. Obtain $\boldsymbol{\omega}_i'$, $i = 1, \ldots, b$, from $\mathbf{y}$, use Eq. 6.109 to calculate $\dot{\mathbf{p}}_i$, and then transfer to $\dot{\mathbf{q}}$.

(c.2) Evaluate **M** (since **M** is constant in Eq. 11.37, it can be evaluated only once), **b**, and **g**.

(c.3) Solve Eq. 11.37 for $\dot{\mathbf{h}}$.

(c.4) Transfer $\dot{\mathbf{q}}$ and $\dot{\mathbf{h}}$ to $\dot{\mathbf{y}}$.

(c.5) Return.

13.3 KINEMATICALLY CONSTRAINED SYSTEMS

The complete set of equations of motion for a kinematically constrained mechanical system is given as

$$\mathbf{\Phi} \equiv \mathbf{\Phi}(\mathbf{q}) = \mathit{0} \tag{13.12}$$

$$\dot{\mathbf{\Phi}} \equiv \mathbf{\Phi}_{\mathbf{q}}\dot{\mathbf{q}} = \mathit{0} \tag{13.13}$$

$$\ddot{\mathbf{\Phi}} \equiv \mathbf{\Phi}_{\mathbf{q}}\ddot{\mathbf{q}} - \boldsymbol{\gamma} = \mathit{0} \tag{13.14}$$

$$\mathbf{M}\ddot{\mathbf{q}} - \mathbf{\Phi}_{\mathbf{q}}^T\boldsymbol{\lambda} = \mathbf{g} \tag{13.15}$$

These equations may represent the planar equations of motion given in Eq. 9.6, or the spatial equations of motion given in Eq. 11.42. In the case of spatial motion, it is assumed that the constraints of Eq. 13.12 contain both kinematic constraints and mathematical constraints. Therefore, the Jacobian matrix $\mathbf{\Phi}_{\mathbf{q}}$ in Eqs. 13.13 to 13.15 contains the **P** and **B** matrices of Eq. 11.42. If Eqs. 13.14 and 13.15 are appended together, a set of algebraic equations, linear in $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$, is obtained:

$$\begin{bmatrix} \mathbf{M} & \mathbf{\Phi}_{\mathbf{q}}^T \\ \mathbf{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \boldsymbol{\gamma} \end{bmatrix} \tag{13.16}$$

It should be clear that $\mathbf{g}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\gamma}$ in Eq. 13.16 represent $\mathbf{g}^* - \mathbf{b}^*$, $\boldsymbol{\sigma}$ and $\boldsymbol{\lambda}$, and $\mathbf{c}$ and $\boldsymbol{\gamma}$, respectively, in Eq. 11.42.

A simple but crude method for obtaining the dynamic response of a system represented by Eqs. 13.12 to 13.15 is to employ the direct integration algorithm DI-1 with some minor modifications:

ALGORITHM DI-4

(a) and **(b)** the same as in DI-1.

(c) DIFEQN routine

(c.1) Transfer **y** to **q** and $\dot{\mathbf{q}}$.

(c.2) Evaluate **M** (**M** is constant in Eq. 9.6, but a function of $\mathbf{p}_i$, $i = 1, \ldots, b$, in Eq. 11.42), $\mathbf{\Phi}_{\mathbf{q}}$, **g**, and $\boldsymbol{\gamma}$.

(c.3) Solve Eq. 13.16 for $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$.

(c.4) Transfer $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ to $\dot{\mathbf{y}}$.

(c.5) Return.

The initial conditions on **q** and $\dot{\mathbf{q}}$ must satisfy Eqs. 13.12 and 13.13. However, on account of the numerical integration error, these equations may be violated. In the preceding sections, several methods for circumventing this problem were presented.

13.3.1 Constraint Violation Stabilization Method

The constraint violation stabilization method[1] is an extension of feedback control theory applied to the dynamic analysis of mechanical systems. One of the goals in designing a feedback controller is to suppress the growth of error and achieve a stable response.

In control systems, it is well known that circuits described by second-order differential equations such as

$$\ddot{y} = 0 \tag{a}$$

are unstable, since outside disturbances such as noise (or numerical error, in the case of a numerical integration process) can be amplified. In contrast to Eq. *a*, which is said to be an open-loop system, a closed-loop system, such as

$$\ddot{y} + 2\alpha\dot{y} + \beta^2 y = 0$$

is stable if α and β are positive constants. The terms $2\alpha\dot{y}$ and $\beta^2 y$ are the feedback control terms that achieve stability for the differential equation.

The violations in the constraints of Eqs. 13.12 and 13.13 are denoted as

$$\mathbf{\Phi} \equiv \mathbf{\Phi}(\mathbf{q}^*) = \boldsymbol{\varepsilon} \tag{b}$$

and

$$\dot{\mathbf{\Phi}} \equiv \mathbf{\Phi}_{\mathbf{q}}\dot{\mathbf{q}}^* = \boldsymbol{\sigma} \tag{c}$$

where $\mathbf{q}^*$ and $\dot{\mathbf{q}}^*$ are the computed values of $\mathbf{q}$ and $\dot{\mathbf{q}}$. Knowing $\mathbf{q}^*$ and $\dot{\mathbf{q}}^*$, we can find the acceleration vector $\ddot{\mathbf{q}}$ from Eq. 13.16. For these computed vectors, Eq. 13.14 finds the form

$$\ddot{\mathbf{\Phi}} \equiv \mathbf{\Phi}_{\mathbf{q}}\ddot{\mathbf{q}}^* - \boldsymbol{\gamma}^* = \boldsymbol{0} \tag{13.17}$$

Vector $\ddot{\mathbf{q}}^*$ is different from the correct acceleration vector $\ddot{\mathbf{q}}$. The errors in the three vectors are

$$\mathbf{q}^* - \mathbf{q} = \Delta\mathbf{q}$$

$$\dot{\mathbf{q}}^* - \dot{\mathbf{q}} = \Delta\dot{\mathbf{q}}$$

$$\ddot{\mathbf{q}}^* - \ddot{\mathbf{q}} = \Delta\ddot{\mathbf{q}}$$

Since $\ddot{\mathbf{q}}^*$ is integrated to obtain $\dot{\mathbf{q}}^*$ in the next step, any error $\Delta\ddot{\mathbf{q}}$ subsequently adds to any existing error in the velocity vector. It is ideal to have $\Delta\ddot{\mathbf{q}} = \boldsymbol{0}$. But since this is an open-loop system, it can be replaced, for the integration process, by the closed-loop system

$$\Delta\ddot{\mathbf{q}} + 2\alpha\Delta\dot{\mathbf{q}} + \beta^2\Delta\mathbf{q} = \boldsymbol{0} \tag{d}$$

Equation *b* is expanded about $\mathbf{q}$ and the second- and higher-order terms are eliminated, to find

$$\mathbf{\Phi}_{\mathbf{q}}\Delta\mathbf{q} = \boldsymbol{\varepsilon}$$

From Eq. *c*, it is found that

$$\mathbf{\Phi}_{\mathbf{q}}\Delta\dot{\mathbf{q}} = \boldsymbol{\sigma}$$

Premultiplying Eq. *d* by $\mathbf{\Phi}_{\mathbf{q}}$ yields

$$\mathbf{\Phi}_{\mathbf{q}}(\ddot{\mathbf{q}}^* - \ddot{\mathbf{q}}) + 2\alpha\mathbf{\Phi}_{\mathbf{q}}\Delta\dot{\mathbf{q}} + \beta^2\mathbf{\Phi}_{\mathbf{q}}\Delta\mathbf{q} = \boldsymbol{0}$$

or

$$\boldsymbol{\Phi}_{\mathbf{q}}\ddot{\mathbf{q}}^* - \gamma + 2\alpha\boldsymbol{\sigma} + \beta^2\boldsymbol{\varepsilon} = \mathbf{0} \qquad (e)$$

If the constraint violations $\boldsymbol{\varepsilon}$ and $\boldsymbol{\sigma}$ are replaced by constraint symbols $\boldsymbol{\Phi}$ and $\dot{\boldsymbol{\Phi}}$ respectively, then Eq. 13.17 and Eq. e yield

$$\gamma^* = \gamma - 2\alpha\dot{\boldsymbol{\Phi}} - \beta^2\boldsymbol{\Phi}$$

Appending Eq. 13.17 to Eq. 13.15 yields the stabilized form of Eq. 13.16:

$$\begin{bmatrix} \mathbf{M} & \boldsymbol{\Phi}_{\mathbf{q}}^T \\ \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \boldsymbol{\gamma} - 2\alpha\dot{\boldsymbol{\Phi}} - \beta^2\boldsymbol{\Phi} \end{bmatrix} \qquad (13.18)$$

where $\ddot{\mathbf{q}}$ represents the computed accelerations. When there is no violation in the constraints, Eq. 13.18 becomes identical to Eq. 13.16.

An algorithm CS for the constraint stabilization method can be stated by a slight modification to the algorithm DI-4:

ALGORITHM CS-1

(a) and **(b)** are as in DI-1, but in **(a)** values are assigned to α and β.

(c) DIFEQN routine

- **(c.1)** Transfer $\mathbf{y}$ to $\mathbf{q}$ and $\dot{\mathbf{q}}$.
- **(c.2)** Evaluate $\mathbf{M}$, $\boldsymbol{\Phi}_{\mathbf{q}}$, $\mathbf{g}$, and $\boldsymbol{\gamma}$.
- **(c.3)** Evaluate $\boldsymbol{\Phi}$ and calulate $\dot{\boldsymbol{\Phi}} \equiv \boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}}$.
- **(c.4)** Solve Eq. 13.18 for $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$.
- **(c.5)** Transfer $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ to $\dot{\mathbf{y}}$.
- **(c.6)** Return.

The effect of introducing the feedback terms in Eq. 13.18 is illustrated in Fig. 13.1, with some exaggeration, for a typical response. When both α and β are given zero values, which is exactly the method of algorithm DI-4, the numerical result may diverge from the exact solution. For nonzero values of α and β, the solution oscillates about the exact solution. The amplitude and the frequency of the oscillation due to the stabilization terms depend upon the values of α and β. Experience has shown that for most practical problems, a range of values between 1 and 10 for α and β is adequate. When $\alpha = \beta$, critical damping is achieved, which usually stabilizes the response more quickly.[19]

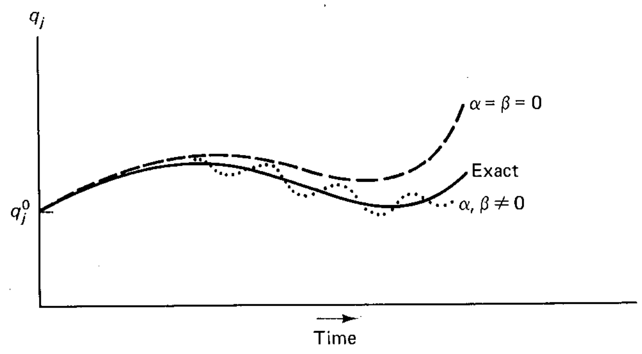

Figure 13.1 Schematic representation of the exact and numerical solutions to a typical dynamic response.

13.3.2 Coordinate Partitioning Method

The coordinate partitioning method[†] controls the accumulation of the numerical error quite differently from the constraint stabilization method. This method makes use of the fact that the n coordinates $\mathbf{q}$ are not independent. If the n coordinates are partitioned into m dependent coordinates $\mathbf{u}$ and k independent coordinates $\mathbf{v}$, then the velocity vector $\dot{\mathbf{q}}$ can be partitioned accordingly into $\dot{\mathbf{u}}$ and $\dot{\mathbf{v}}$. The integration arrays $\mathbf{y}$ and $\dot{\mathbf{y}}$ are defined in terms of the independent variables:

$$\mathbf{y} = \begin{bmatrix} \mathbf{v} \\ \dot{\mathbf{v}} \end{bmatrix}, \qquad \dot{\mathbf{y}} = \begin{bmatrix} \dot{\mathbf{v}} \\ \ddot{\mathbf{v}} \end{bmatrix}$$

where $\ddot{\mathbf{v}}$ is the vector of independent accelerations. The two arrays $\mathbf{y}$ and $\dot{\mathbf{y}}$ each have a dimension of $2k$.

The kinematic constraints and velocity equations of Eqs. 13.12 and 13.13 can be expressed as

$$\boldsymbol{\Phi}(\mathbf{u}, \mathbf{v}) = \mathbf{0} \tag{13.19}$$

and

$$\boldsymbol{\Phi}_{\mathbf{u}}\dot{\mathbf{u}} = -\boldsymbol{\Phi}_{\mathbf{v}}\dot{\mathbf{v}} \tag{13.20}$$

Equations 13.19 and 13.20 each represent m independent equations in terms of $\mathbf{u}$ and $\dot{\mathbf{u}}$ respectively. Having $\mathbf{v}$ and $\dot{\mathbf{v}}$ from $\mathbf{y}$, we can solve Eqs. 13.19 and 13.20 for $\mathbf{u}$ and $\dot{\mathbf{u}}$; then vectors $\mathbf{q}$ and $\dot{\mathbf{q}}$ are completely known. At this point Eq. 13.16 is solved for $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$.

An algorithm for the coordinate partitioning method (CP) can be stated, in its simplest form, as follows:

ALGORITHM CP-1

```
(a) Main routine
    (a.1) Specify initial conditions on $\mathbf{q}$ and $\dot{\mathbf{q}}$.
    (a.2) Specify the independent variables $\mathbf{v}$ (and $\dot{\mathbf{v}}$).
    (a.3) Define vector $\mathbf{y}$ as $\mathbf{y} = [\mathbf{v}^T, \dot{\mathbf{v}}^T]^T$.
    (a.4) Enter the numerical integration routine (NI).
(b) Numerical integration routine
    (same as for DI-1)
(c) DIFEQN routine
    (c.1) Obtain $\mathbf{v}$ and $\dot{\mathbf{v}}$ from $\mathbf{y}$.
    (c.2) Solve Eq. 13.19 for $\mathbf{u}$ using the Newton-Raphson method; $\mathbf{q}$ is found.
    (c.3) Solve Eq. 13.20 for $\dot{\mathbf{u}}$; $\dot{\mathbf{q}}$ is found.
    (c.4) Solve Eq. 13.16 for $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$.
    (c.5) Transfer $\dot{\mathbf{v}}$ and $\ddot{\mathbf{v}}$ (from $\ddot{\mathbf{q}}$) to $\dot{\mathbf{y}}$.
    (c.6) Return.
```

[†]The coordinate partitioning method was first developed in a planar-motion computer program called DADS-2D (dynamic analysis and design system) by Wehage and Haug in 1982.[18] The three-dimensional motion version of this program for DADS-3D was first developed by Nikravesh and Chung, 1982.[12]

The most troublesome part in this algorithm is step c.2. In this step, independent coordinates $\mathbf{v}^i$ are known and the constraint equations are solved for the dependent coordinates $\mathbf{u}^i$. Since the constraints are nonlinear algebraic equations, iterative methods must be employed. These require an estimate for $\mathbf{u}^i$ in every time step. The estimate cannot be too far from the correct solution, since if it is it may cause divergence. An estimate for $\mathbf{u}^i$, at time t^i, can be found by using the information from the previous time t^{i-1}:

$$\mathbf{u}^i \simeq \mathbf{u}^{i-1} + h\dot{\mathbf{u}}^{i-1} + 0.5h^2\ddot{\mathbf{u}}^{i-1}$$

where h is the time step from t^{i-1} to t^i.

Proper partitioning of the coordinates $\mathbf{q}$ into $\mathbf{u}$ and $\mathbf{v}$ is critical in controlling the accumulation of the numerical error. In order to keep this error under control, it might be necessary to switch from one set of independent coordinates to a different set during the integration process. For example, consider the single pendulum shown in Fig. 13.2. Since this is a 1-degree of freedom system, the dimension of $\mathbf{v}$ is 1. For the pendulum (the moving body), with coordinates $\mathbf{q} = [x, y, \phi]^T$, two equations can be written:

$$x = d \cos \phi$$
$$y = d \sin \phi$$

If the numerical error in the coordinates is denoted by δx, δy, and $\delta\phi$, then

$$\delta x = -d \sin \phi\, \delta\phi$$
$$\delta y = d \cos \phi\, \delta\phi$$

In the selection of the independent coordinates, three cases may arise:

1. $\mathbf{v} = [x]$, $\mathbf{u} = [y, \phi]^T$. An error δx causes errors in y and ϕ, as follows:

$$\delta\phi = -\frac{1}{d \sin \phi}\, \delta x$$

$$\delta y = -\frac{\cos \phi}{\sin \phi}\, \delta x$$

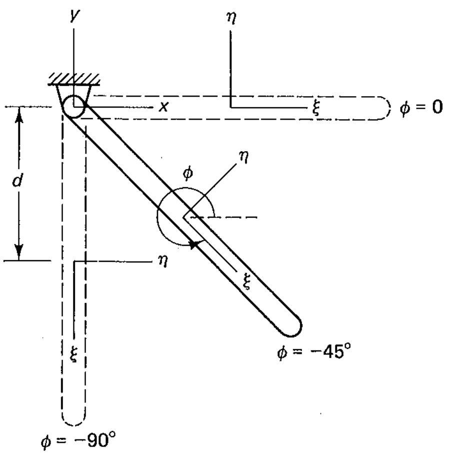

Figure 13.2 A single pendulum.

2. $\mathbf{v} = [y]$, $\mathbf{u} = [x, \phi]^T$. An error δy causes errors in x and ϕ, as follows:

$$\delta\phi = \frac{1}{d \cos\phi}\,\delta y$$

$$\delta x = -\frac{\sin\phi}{\cos\phi}\,\delta y$$

3. $\mathbf{v} = [\phi]$, $\mathbf{u} = [x, y]^T$. An error in $\delta\phi$ causes errors in x and y, as follows:

$$\delta x = -d \sin\phi\,\delta\phi$$

$$\delta y = d \cos\phi\,\delta\phi$$

A comparison of the three cases reveals that for $\phi = 0$ or $\phi = \pi$, case 1 yields large errors in ϕ and y for even a small error in x. However, the errors of the other two cases are bounded. Therefore, for these values of ϕ, or any value of ϕ in these neighborhoods, the selection of x as the independent coordinate is the worst case. Similarly, in the neighborhood of $\phi = \pm\pi/2$, the y coordinate is the worst choice for the independent coordinate. If the pendulum starts from the initial condition $\phi = 0$ and the y coordinate is selected as the independent coordinate, then around $\phi = \pm\pi/4$ the independent coordinate must be switched from y to x in order to keep the error under control. The third case shows that if ϕ is selected as the independent coordinate, the error remains bounded regardless of the orientation of the pendulum, and therefore there is no need to switch to another coordinate at any time.

An automatic technique for partitioning the coordinates into the dependent and independent sets is shown in Sec. 13.3.3. During the integration process, some criteria must be used to indicate whether the independent coordinates must be redefined. Such criteria can be based upon the following observations:

1. The number of iterations in the corrector step of a predictor-corrector integration algorithm keeps increasing from one time step to the next.
2. The number of iterations in the Newton-Raphson process of step c.2 keeps increasing from one time step to the next, i.e., the estimated values for **u** are getting too far from the solution.

A conservative but safe process is to automatically redefine vector **v** once every few time steps.

A modified version of algorithm CP-1 can be stated that allows for redefining the independent and dependent coordinates.

ALGORITHM CP-2

```
(a) and (b) are the same as in CP-1.
(c) DIFEQN routine
    (c.1) Obtain v and v̇ from y.
    (c.2) Solve Eq. 13.19 for u.
    (c.3) Is it necessary to redefine the independent coordinates?
              If yes, then return to step a.2.
              If no, then continue.
```

(c.4) Solve Eq. 13.20 for $\dot{\mathbf{u}}$.

(c.5) Solve Eq. 13.16 for $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$.

(c.6) Transfer $\dot{\mathbf{v}}$ and $\ddot{\mathbf{v}}$ to $\dot{\mathbf{y}}$.

(c.7) Return.

13.3.3 Automatic Partitioning of the Coordinates

In step a.2 of algorithm CP-2, an automatic process can be employed to partition the coordinates into dependent and independent sets. A matrix factorization technique, such as the Gaussian elimination with full or partial (column) pivoting, can be performed on the Jacobian matrix for this process. For a mechanical system with m constraints and n coordinates, the Jacobian is an $m \times n$ matrix. The order of the columns of the matrix corresponds to the order of the elements in vector $\mathbf{q}$. After pivoting, the order of the columns determines the reordering of the elements of $\mathbf{q}$. The first m elements of the reordered $\mathbf{q}$ can be used as the dependent coordinates $\mathbf{u}$, and the remaining k elements represent the independent coordinates $\mathbf{v}$.

Example 13.1

Consider the single pendulum with the oscillating mass shown in Fig. 13.3. Constraint equations for the ground and for the revolute and translational joints are written as

$$\begin{aligned}
&x_1 = 0\\
&y_1 = 0\\
&\phi_1 = 0\\
&x_1 - x_2 + 0.5 \sin \phi_2 = 0\\
&y_1 - y_2 - 0.5 \cos \phi_2 = 0\\
&\sin \phi_2(y_3 - y_2) + \cos \phi_2(x_3 - x_2) = 0\\
&\phi_2 - \phi_3 = 0
\end{aligned}$$

The first three equations are the ground constraints on body 1, the fourth and fifth equations are the revolute joint constraints from Eq. 4.9, the sixth and seventh equations are the translational joint constraints from Eq. 4.12. The first translational constraint is obtained by defining three points on the line of translation having local coordinates $\xi_2^P = 0$, $\eta_2^P = 0$, $\xi_2^Q = 0$, $\eta_2^Q = 1$, $\xi_3^P = 0$, $\eta_3^P = 0$.

If the vector of coordinates is defined as

$$\mathbf{q} = [x_1, y_1, \phi_1, x_2, y_2, \phi_2, x_3, y_3, \phi_3]^T$$

then the Jacobian matrix is written as

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & -1 & 0 & 0.5\cos\phi_2 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & -1 & 0.5\sin\phi_2 & 0 & 0 & 0\\
0 & 0 & 0 & -\cos\phi_2 & -\sin\phi_2 & \textcircled{1} & \cos\phi_2 & \sin\phi_2 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1
\end{bmatrix}$$

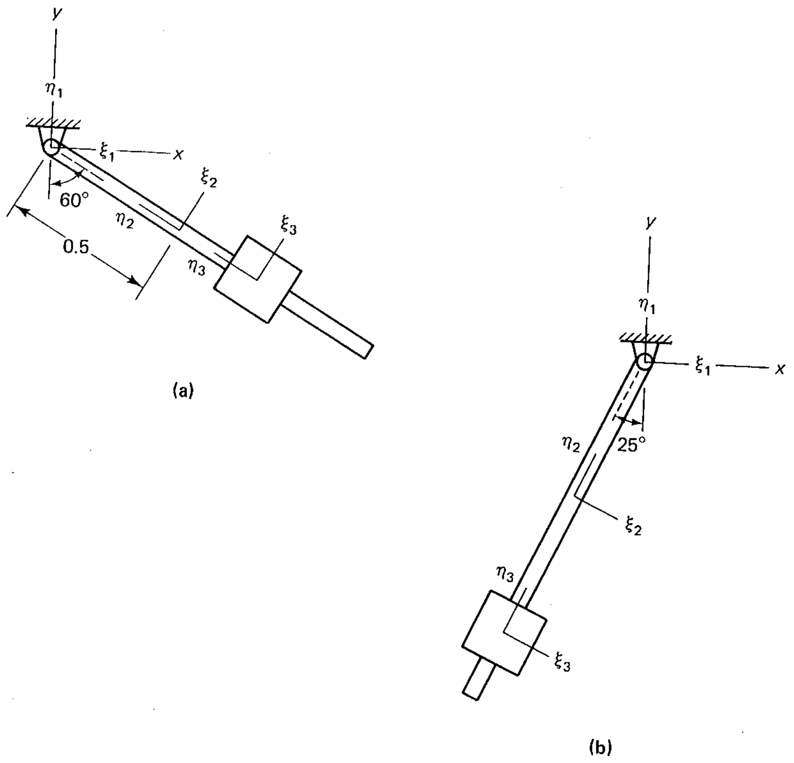

Figure 13.3 A single pendulum with an oscillating mass in two different orientations.

where

$$\textcircled{1} \equiv -\sin\phi_2(x_3 - x_2) + \cos\phi_2(y_3 - y_2)$$

For the configuration in Fig. 13.3(a), the coordinates of the moving bodies are:

$$\begin{array}{lll} x_2 = 0.43, & y_2 = -0.25, & \phi_2 = 60^\circ \\ x_3 = 0.69, & y_3 = -0.40, & \phi_3 = 60^\circ \end{array}$$

With these coordinates, the Jacobian matrix becomes

$$\begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \end{array}$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0.43 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & -0.87 & -0.3 & 0.5 & 0.87 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

The column indices corresponding to the order of elements in $\mathbf{q}$ are shown at the top of the matrix. A Gaussian elimination with partial (column) pivoting yields

$$\begin{array}{ccccccccc}
1 & 2 & 3 & 4 & 5 & 8 & 6 & 7 & 9 \\
\end{array}$$
$$\left[\begin{array}{ccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -0.25 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & -0.43 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -0.92 & 0.58 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1
\end{array}\right]$$

The order of the column indices shows the reordering of the elements of $\mathbf{q}$ as

$$\mathbf{q} \equiv [x_1, y_1, \phi_1, x_2, y_2, y_3, \phi_2, x_3, \phi_3]^T$$

Since $m = 7$ and $k = 2$, vectors $\mathbf{u}$ and $\mathbf{v}$ are defined as

$$\mathbf{u} \equiv [x_1, y_1, \phi_1, x_2, y_2, y_3, \phi_2]^T$$
$$\mathbf{v} \equiv [x_3, \phi_3]^T$$

For the configuration shown in Fig. 13.3(b), the moving bodies have the coordinates

$$x_2 = -0.21, \qquad y_2 = -0.45, \qquad \phi_2 = -25^\circ$$
$$x_3 = -0.42, \qquad y_3 = -0.91, \qquad \phi_3 = -25^\circ$$

With these coordinates, the Jacobian matrix becomes

$$\begin{array}{ccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}$$
$$\left[\begin{array}{ccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0.45 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & -0.21 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.91 & 0.42 & -0.5 & 0.91 & -0.42 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1
\end{array}\right]$$

A Gaussian elimination on this matrix yields

$$\begin{array}{ccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 9 & 8 & 7 \\
\end{array}$$
$$\left[\begin{array}{ccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -0.45 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0.22 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0.42 & -0.91 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.42 & -0.91
\end{array}\right]$$

In this configuration, the appropriate sets of dependent and independent coordinates are

$$\mathbf{u} \equiv [x_1, y_1, \phi_1, x_2, y_2, \phi_2, \phi_3]^T$$

$$\mathbf{v} \equiv [y_3, x_3]^T$$

The Gaussian elimination with pivoting suggests that for Fig. 13.3(a) the appropriate independent coordinates are x_3 and ϕ_3 but for Fig. 13.3(b) the appropriate independent coordinates are x_3 and y_3. Note that the coordinates of the non-moving body always become part of the set of dependent coordinates.

Several factors influence the selection of the independent coordinates. One such factor is the choice of the unit system. Since not all of the elements of the Jacobian matrix have the same physical dimension, their numerical values may form different ratios when different systems of unit are employed (e.g., SI units versus the U.S. customary FPS units). This in effect yields a different pivoting process, and hence a different set of dependent and independent coordinates.

The second factor is the type of pivoting. A partial (column) pivoting may yield a different result from that given by a full pivoting. Matrix factorization with full pivoting may have some advantage, in terms of the numerical error, over the partial pivoting. However, it cannot be said that the partitioning of the coordinates through a full pivoting process yields a better (in the physical sense) set of independent coordinates.

Possibly the most influential factor in an automatic partitioning of coordinates is the method of matrix factorization. If an L-U factorization process is employed on the constraint Jacobian matrix, instead of the standard Gaussian elimination, different sets of dependent and independent coordinates may be obtained. The original coordinate partitioning algorithm with L-U factorization was suggested by Wehage.[18] A brief discussion on coordinate partitioning with L-U factorization can be found in Appendix C. In recent years, several other matrix factorization techniques have been employed by other researchers, such as the singular-value decomposition, the QR decomposition, and the Gram-Schmidt process. These techniques offer some advantages over L-U factorization, although the main idea remains basically the same. [8, 10, 11]

13.3.4 Stiff Differential Equation Method

The method of solving a mixed system of algebraic and differential equations of motion presented in this section is completely different in principle from the methods discussed in the preceding sections. This method considers the algebraic constraint equations to be a special form of differential equation in which the time derivatives of the variables do not appear. This assumption has proved to cause the system equations to become numerically stiff. Therefore, a stiff numerical integration method must be applied to solve the equations.[†]

[†]This algorithm has served as a forerunner in the development of the numerical methods in the area of mechanical systems.[16] The algorithm has been formulated into a computer program for three-dimensional motion known as ADAMS.

In Sec. 13.1, it was stated that the standard numerical integration algorithms are designed to solve systems of differential equations of the form

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t) \tag{13.21}$$

The modified approach taken here allows for the simultaneous solution of *mixed algebraic and differential equations* of the form

$$\mathbf{g}(\mathbf{y}, \dot{\mathbf{y}}, t) = \mathbf{0} \tag{13.22}$$

where some components of $\dot{\mathbf{y}}$ may not appear in some of the equations. When none of the components of $\dot{\mathbf{y}}$ appear in an equation, that equation is an algebraic equation; otherwise it is a differential equation.

The kth order Gear algorithm and its corresponding corrector formula are given by Eqs. 12.23 and 12.24. In order to modify these formulas to solve a mixed system of algebraic and differential equations, Eq. 13.22 is written as

$$\mathbf{g}(\mathbf{z}, t) = \mathbf{0} \tag{13.23}$$

where $\mathbf{z} = [\mathbf{y}^T, \dot{\mathbf{y}}^T]^T$. The Newton-Raphson formula for this equation is

$$\mathbf{g}_{\mathbf{z}}^{(l)}\, \Delta\mathbf{z}^{(l)} = -\mathbf{g}^{(l)} \tag{13.24}$$

where l is the iteration number. When the substitution $\mathbf{z} = [\mathbf{y}^T, \dot{\mathbf{y}}^T]^T$ is made in Eq. 13.24, it is found that

$$\mathbf{g}_{\mathbf{y}}^{(l)}\, \Delta\mathbf{y}^{(l)} + \mathbf{g}_{\dot{\mathbf{y}}}^{(l)}\, \Delta\dot{\mathbf{y}}^{(l)} = -\mathbf{g}^{(l)} \tag{13.25}$$

For the lth and $l + 1$st Newton-Raphson iterations, Eq. 12.23 can be rewritten as

$$(\mathbf{y}^{i+1})^{(l)} - hb_{-1}(\dot{\mathbf{y}}^{i+1})^{(l)} - \sum_{j=0}^{k-1} (a_j \mathbf{y}^{i-j}) = \mathbf{0} \tag{13.26}$$

$$(\mathbf{y}^{i+1})^{(l+1)} - hb_{-1}(\dot{\mathbf{y}}^{i+1})^{(l+1)} - \sum_{j=0}^{k-1} (a_j \mathbf{y}^{i-j}) = \mathbf{0} \tag{13.27}$$

The summation terms in these equations are not a function of the iteration number—they are a function of the information from the ith and previous time steps, so they remain constant at each iteration. Subtracting Eq. 13.26 from Eq. 13.27 yields

$$(\mathbf{y}^{i+1})^{(l+1)} - (\mathbf{y}^{i+1})^{(l)} - hb_{-1}(\dot{\mathbf{y}}^{i+1})^{(l+1)} + hb_{-1}(\dot{\mathbf{y}}^{i+1})^{(l)} = \mathbf{0}$$

or

$$\Delta\dot{\mathbf{y}}^{(l)} = \frac{1}{hb_{-1}}\Delta\mathbf{y}^{(l)} \tag{13.28}$$

which holds true for the $i + 1$st or any other time step. Substitution of Eq. 13.28 into Eq. 13.25 results in the *corrector formula*

$$\left(\mathbf{g}_{\mathbf{y}}^{(l)} + \frac{1}{hb_{-1}}\mathbf{g}_{\dot{\mathbf{y}}}^{(l)}\right)\Delta\mathbf{y}^{(l)} = -\mathbf{g}^{(l)} \tag{13.29}$$

If Eq. 13.22 is of the form

$$\mathbf{g}(\mathbf{y}, \dot{\mathbf{y}}, t) = \mathbf{P}\dot{\mathbf{y}} + \mathbf{p}(\mathbf{y}, t) = \mathbf{0} \tag{13.30}$$

where $\mathbf{P}$ is a constant matrix or a time-dependent matrix, then Eq. 13.29 can be written in a simpler form as

$$\left(\mathbf{p}_{\mathbf{y}}^{(l)} + \frac{1}{hb_{-1}}\mathbf{P}\right)\Delta\mathbf{y}^{(l)} = -\mathbf{g}^{(l)} \tag{13.31}$$

At each time step, the iterative corrector process of Eq. 13.29 or Eq. 13.31 is continued until all of the Newton differences $\Delta\mathbf{y}^{(l)}$ are below a specified tolerance level. At each Newton-Raphson iteration, arrays $\mathbf{y}$ and $\dot{\mathbf{y}}$ are updated:

$$\begin{aligned}\mathbf{y}^{(l+1)} &= \mathbf{y}^{(l)} + \Delta\mathbf{y}^{(l)}\\ \dot{\mathbf{y}}^{(l+1)} &= \dot{\mathbf{y}}^{(l)} + \frac{1}{hb_{-1}}\Delta\mathbf{y}^{(l)}\end{aligned} \tag{13.32}$$

The total-system equations of motion of Eqs. 13.15 and 13.12 are written as

$$\mathbf{M}\dot{\mathbf{s}} - \mathbf{\Phi}_{\mathbf{q}}^{T}\boldsymbol{\lambda} = \mathbf{g} \tag{13.33}$$

$$\dot{\mathbf{q}} = \mathbf{s} \tag{13.34}$$

$$\mathbf{\Phi}(\mathbf{q}) = \mathit{0} \tag{13.35}$$

Equations 13.33 to 13.35 may be expressed in the form of Eq. 13.30, where

$$\mathbf{P} = \begin{bmatrix}\mathbf{M} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix}-\mathbf{\Phi}_{\mathbf{q}}^{T}\boldsymbol{\lambda} - \mathbf{g}\\ -\mathbf{s}\\ \mathbf{\Phi}\end{bmatrix}$$

$$\mathbf{y} = [\mathbf{s}^T, \mathbf{q}^T, \boldsymbol{\lambda}^T]^T$$

and

$$\dot{\mathbf{y}} = [\dot{\mathbf{s}}^T, \dot{\mathbf{q}}^T, \dot{\boldsymbol{\lambda}}^T]^T$$

The corrector formula of Eq. 13.31 can be employed to solve for the unknown $\mathbf{y}$ at every time step. In this case, Eq. 13.31 provides $2n + m$ equations in $2n + m$ unknowns.

It must be expected that the iterative solution of $2n + m$ equations, using the Newton-Raphson method, may not be successful for every problem. For large systems of equations, this method may not be considered efficient. One major drawback of this algorithm is the initial estimate on the variables at time $t = t^0$. At the starting time, the initial condition on $\mathbf{q}$ and $\dot{\mathbf{q}}$ might be available. However, at $t = t^0$, for almost every practical problem, no information on the Lagrange multipliers $\boldsymbol{\lambda}$ can be found. Therefore, starting the Newton-Raphson iteration at $t = t^0$ for an arbitrary estimate on $\boldsymbol{\lambda}$ may cause divergence.

It was noted at the beginning of this section that treating algebraic equations as special forms of differential equations yields numerically stiff systems. This causes artificially high-frequency components in the solution. The high-frequency components of the response do not represent the physical system—they are introduced into the solution numerically. Because of the presence of the high-frequency components in the response, the time increment h must be chosen relatively small. For small values of h, the term $1/h$ in the algorithm can become substantially large. Experience has shown that this algorithm cannot be implemented on machines with small word length (4 bytes or single precision). Either a machine with a word length of 8 bytes is needed, or double precision must be employed.

13.4 JOINT COORDINATE METHOD

In Table 1.1 of Sec. 1.4.1, a comparison was made among three different coordinate systems in terms of various aspects of formulating the equations of motion. The table showed, in a relative sense, that if a set of generalized coordinates (in which the coordinates are equal in number to the number of degrees of freedom) is employed, derivation of the equations of motion can be quite difficult. However, computational efficiency in solving these equations is gained. In contrast, employing Cartesian coordinates yields easy derivation of the equations of motion, but computational efficiency is lost. The joint coordinate method takes advantage of Cartesian coordinates for easy formulation, and Lagrangian coordinates for computational efficiency. This is done by *numerically* combining the two schemes of formulation. This method is based on the *velocity transformations* developed by Jerkovsky.[7]

Consider two bodies i and j connected by a revolute joint as shown in Fig. 13.4(a). If the relative angle between the two bodies about the joint axis is denoted by θ, then, for known coordinates of body i and known θ, the coordinates of body j can be found; i.e.,

$$\mathbf{q}_j = \mathbf{f}^{(r)}(\mathbf{q}_i, \theta)$$

If the two bodies are connected by a translational joint, as in Fig. 13.4(b), a similar formula can be found:

$$\mathbf{q}_j = \mathbf{f}^{(t)}(\mathbf{q}_i, \theta)$$

where θ indicates the relative distance between the two bodies. If there are 2 degrees of freedom between the two bodies, as there are for the universal joint shown in Fig. 13.4(c), then there are two relative angles, θ_1 and θ_2, denoted by $\boldsymbol{\theta}_{ij} = [\theta_1, \theta_2]^T$:

$$\mathbf{q}_j = \mathbf{f}^{(u)}(\mathbf{q}_i, \boldsymbol{\theta}_{ij})$$

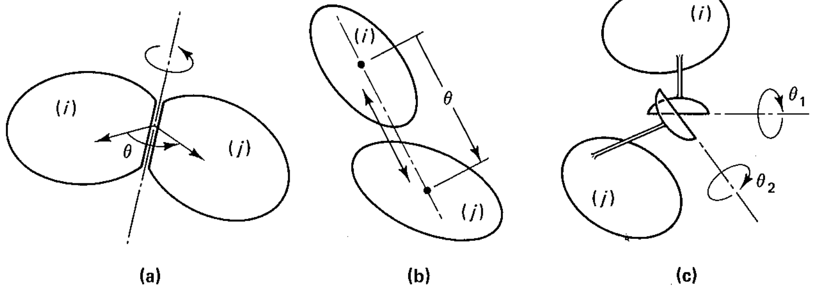

Figure 13.4 Two bodies connected by (a) a revolute joint, (b) a translational joint, and (c) a universal joint.

In general, if two bodies i and j are connected by a kinematic joint having relative (joint) coordinates $\boldsymbol{\theta}_{ij}$, then the coordinates $\mathbf{q}_j$ can be expressed as follows:

$$\mathbf{q}_j = \mathbf{f}^{(c)}(\mathbf{q}_i, \boldsymbol{\theta}_{ij}) \tag{13.36}$$

The number of relative coordinates in $\boldsymbol{\theta}_{ij}$ is equal to the number of relative degrees of freedom between the two bodies, which is dependent only on the kinematic joint connecting the bodies.

A similar expression can be stated for velocity calculation. If the relative velocity between bodies i and j is described as $\dot{\boldsymbol{\theta}}_{ij}$, then $\dot{\mathbf{q}}_j$ can be expressed as follows:

$$\dot{\mathbf{q}}_j = \mathbf{f}^{(v)}(\dot{\mathbf{q}}_i, \dot{\boldsymbol{\theta}}_{ij}) \tag{13.37}$$

A similar but inverse expression can be stated for the accelerations. If it is assumed that the absolute accelerations $\ddot{\mathbf{q}}_i$ and $\ddot{\mathbf{q}}_j$ are known, then the relative acceleration can be found as

$$\ddot{\boldsymbol{\theta}}_{ij} = \mathbf{f}^{(a)}(\ddot{\mathbf{q}}_i, \ddot{\mathbf{q}}_j) \tag{13.38}$$

Explicit formulas for coordinate, velocity, and acceleration computations (Eqs. 13.36 through 13.38) can be derived for a variety of kinematic joints.[9] This is left as an exercise to the interested reader.

13.4.1 Open-Chain Systems

Consider the open-chain system shown in Fig. 13.5(a), containing one branch and one grounded body (called the base body). Consecutive bodies are connected by kinematic joints. The system may contain force elements that are not shown in the figure. If the bodies are numbered from 1 to b, in any desired order, then relative coordinates $\boldsymbol{\theta}_{ij}$ are defined between every two adjacent bodies. In this system, the coordinates of the base body $\mathbf{q}_1$ are constants. A vector of relative coordinates $\boldsymbol{\theta}$ is defined as follows:

$$\boldsymbol{\theta} \equiv [\boldsymbol{\theta}_{12}^T, \boldsymbol{\theta}_{23}^T, \ldots, \boldsymbol{\theta}_{(b-1)b}^T]^T$$

For numerical integration, the two vectors $\mathbf{y}$ and $\dot{\mathbf{y}}$ are then defined:

$$\mathbf{y} = \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \end{bmatrix}, \qquad \dot{\mathbf{y}} = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} \tag{13.39}$$

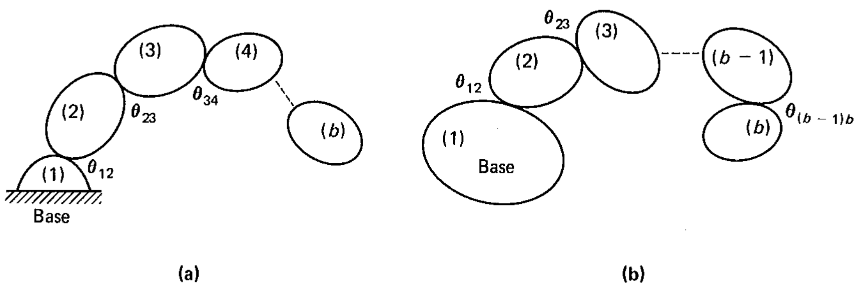

Figure 13.5 A single-branch open-chain system with (a) a fixed base body, and (b) a floating base body.

If, like the system shown in Fig. 13.5(b), the mechanical system does not have a fixed (grounded) base body, then a base body has to be chosen for the system; the resulting base body is called a *floating* base body. If, for example, body 1 is chosen to be the floating base body, then vectors $\mathbf{y}$ and $\dot{\mathbf{y}}$ are defined as follows:

$$\mathbf{y} = \begin{bmatrix} \mathbf{q}_1 \\ \boldsymbol{\theta} \\ \dot{\mathbf{q}}_1 \\ \dot{\boldsymbol{\theta}} \end{bmatrix} \qquad \dot{\mathbf{y}} = \begin{bmatrix} \dot{\mathbf{q}}_1 \\ \dot{\boldsymbol{\theta}} \\ \ddot{\mathbf{q}}_1 \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} \tag{13.40}$$

The dimension of $\mathbf{y}$ or $\dot{\mathbf{y}}$ in the form of either Eq. 13.39 or Eq. 13.40, is twice the number of degrees of freedom of the system.

For systems with no grounded body, there is at least one floating base body. Although the selection of the floating base body is not unique, one body may be a better candidate for the floating base body than another. If the mass of one body is substantially greater than the mass of any of the other bodies, then that body should be selected as the floating base body. It should be noted that the floating base body is not necessarily one of the end bodies in the chain. If no one body has a mass substantially greater than the others, a floating base body can be selected by employing a simple procedure. Each joint is given a number, called the *distance,* which represents the number of its relative degrees of freedom. For some commonly used three-dimensional kinematic joints, the following data can be found:

Joint	Symbol	Distance
Spherical (globular)	*G*	3
Revolute	*R*	1
Universal	*U*	2
Cylindrical	*C*	2
Translational (prismatic)	*P*	1
Screw	*S*	1

Each body in the system is treated momentarily as a candidate for the floating base. The sum of distances from branch to branch starting from the body under consideration is calculated and recorded. When the next neighboring body becomes the candidate, the sum of distances decreases in the direction of the move and increases in the other direction. For example, consider the system shown in Fig. 13.6. The following table can be found for the sums of distances for all the bodies:

Floating base candidate	Sum of distances to left	Sum of distances to right
1	0	11
2	1	10
3	2	9
4	3	8
5	5	6
6	8	3
7	11	0

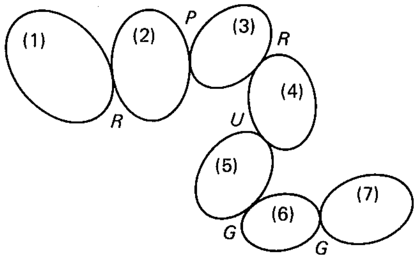

Figure 13.6 An open-chain system.

This table shows that the difference between the left and right distances is smallest for body 5, and therefore body 5 is the best candidate for the floating base body. This selection minimizes the propagation of numerical error in the computation. For example, if body 5 is selected as the floating base body, then the coordinates of body 1 contain the numerical error accumulated from five relative coordinates. But if body 6 is selected as the floating base body, then body 1 contains the numerical error from eight relative coordinates. Applying this observation to both left and right subbranches for each of the bodies shows how the total error can be minimized by this process.

Some systems may have multiple branches, such as the system shown in Fig. 13.7, which has two branches. For position and velocity computation, the process starts from the base body and moves toward the last body in each branch. For multibranch systems, after the process is completed for the first branch, the process can start on the second branch from the branching body (in Fig. 13.7, body 3).

The order of connectivity between the bodies of a system is called the *system topology*. The topology of a system either can be defined by inspection or can be done automatically through *graph theory*. The topology of the system can be set up in the for of a table showing the direction to move in calculating the coordinates (or velocities) of body j, once $\boldsymbol{\theta}_{ij}$ and coordinates (or velocities) of body i are known. For the system of Fig. 13.7, this table can have the following entries:

Body i	Body j	
1 (base)	2	branch 1
2	3	
3	4	
4	. . .	
. . .	j	
3	$j+1$	branch 2
$j+1$	$j+2$	
$j+2$	. . .	
. . .	b	

This table also serves as a directive for calculating $\ddot{\boldsymbol{\theta}}_{ij}$ from $\ddot{\mathbf{q}}_i$ and $\ddot{\mathbf{q}}_j$.

At this point, an algorithm can be stated for the joint coordinate method. At the beginning of each time step, the numerical values for $\mathbf{q}_{\text{base}}$, $\dot{\mathbf{q}}_{\text{base}}$, $\boldsymbol{\theta}$, and $\dot{\boldsymbol{\theta}}$ are known. Equations 13.36 and 13.37 yield $\mathbf{q}$ and $\dot{\mathbf{q}}$ for all of the bodies in the system. The coefficient matrix and the right-side vector of Eq. 13.16 can be evaluated, since they are functions of $\mathbf{q}$ and $\dot{\mathbf{q}}$. Then the solution of Eq. 13.16 yields $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$. Since $\ddot{\mathbf{q}}$ is known for all

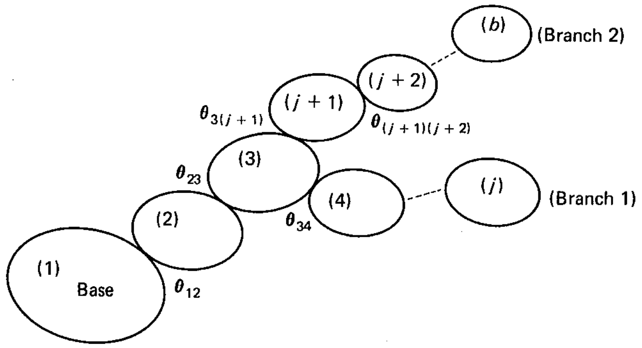

Figure 13.7 A multibranch open-chain system.

of the bodies, Eq. 13.38 yields $\ddot{\boldsymbol{\theta}}$. Then the numerical integration routine moves the process to the next time step.

ALGORITHM JC-1

```
(a) Main routine
    (a.1) Specify initial conditions for $\mathbf{q}$ and $\dot{\mathbf{q}}$.
    (a.2) Specify (or automatically determine) the topology of the system.
    (a.3) Compute initial conditions for $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$.
    (a.4) Transfer the initial values to $\mathbf{y}$ (Eq. 13.39 or 13.40).
(b) Numerical integration routing
    (Same as DI-1)
(c) DIFEQN routine
    (c.1) Transfer the contents of $\mathbf{y}$ to $\mathbf{q}_{\text{base}}$, $\dot{\mathbf{q}}_{\text{base}}$ (if there is a floating base body), $\boldsymbol{\theta}$,
          and $\dot{\boldsymbol{\theta}}$.
    (c.2) Compute $\mathbf{q}$ and $\dot{\mathbf{q}}$ for all of the bodies (Eqs. 13.36 and 13.37).
    (c.3) Evaluate $\mathbf{M}$, $\boldsymbol{\Phi}_{\mathbf{q}}$, $\mathbf{g}$, and $\boldsymbol{\gamma}$.
    (c.4) Solve Eq. 13.16 for $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$.
    (c.5) Compute $\ddot{\boldsymbol{\theta}}$ (Eq. 13.38).
    (c.6) Transfer $\dot{\mathbf{q}}_{\text{base}}$, $\ddot{\mathbf{q}}_{\text{base}}$ (if there is a floating base body), $\dot{\boldsymbol{\theta}}$, and $\ddot{\boldsymbol{\theta}}$ to $\dot{\mathbf{y}}$.
    (c.7) Return.
```

In this algorithm, since the coordinates of a body are found from the coordinates of the adjacent body and the interconnected joint coordinates, the constraint equations in terms of Cartesian coordinates are never violated. The same argument is also true for the velocity equations.

13.4.2 Closed-Loop Systems

A mechanical system may contain one or more independent closed kinematic loops. For example, the system shown in Fig. 13.8(a) contains one closed loop. If the closed loop is *cut* at one of its joints, as in Fig. 13.8(b), the system becomes equivalent to an open-chain system. If the topology of the equivalent open-chain system is defined, then $\mathbf{q}$ and

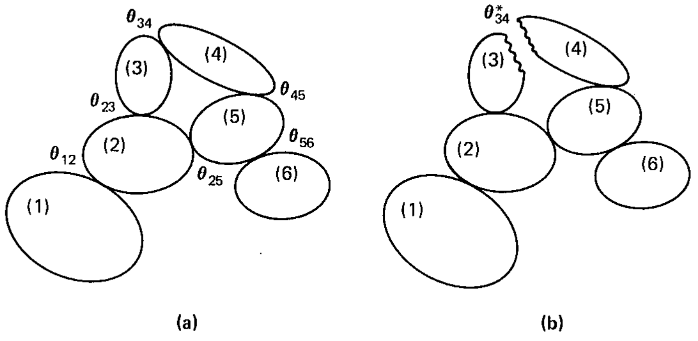

Figure 13.8 A system containing a closed loop.

$\dot{\mathbf{q}}$ for all of the bodies can be computed from $\mathbf{q}_{\text{base}}$, $\dot{\mathbf{q}}_{\text{base}}$, $\boldsymbol{\theta}$, and $\dot{\boldsymbol{\theta}}$. In this process the joint coordinates and velocities of the cut edge(s) are not needed, e.g., the joint coordinates between bodies 3 and 4 of Fig. 13.8(b) denoted by $\boldsymbol{\theta}_{34}^*$. If algorithm JC-1 is applied to this system, it is likely that the kinematic constraints describing the joint at the cut edge(s) will be violated. This may happen because the coordinates of $\mathbf{q}_4$ are not found from $\mathbf{q}_3$ and $\boldsymbol{\theta}_{34}$. In order to eliminate the constraint violation at the cut edge(s), feedback terms for constraint stabilization can be introduced in Eq. 13.16.

The constraint equations can be divided into those for uncut edges and for cut edges, as follows:

$$\boldsymbol{\Phi}(\mathbf{q}) = \boldsymbol{0}$$
$$\boldsymbol{\Phi}^*(\mathbf{q}) = \boldsymbol{0}$$

where the asterisk denotes the cut edges. Then, stabilization terms are included in Eq. 13.16, as follows:

$$\begin{bmatrix} \mathbf{M} & \boldsymbol{\Phi}_{\mathbf{q}}^T & \boldsymbol{\Phi}_{\mathbf{q}}^{*T} \\ \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}_{\mathbf{q}}^* & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \boldsymbol{\gamma} \\ \boldsymbol{\gamma}^* - 2\alpha\dot{\boldsymbol{\Phi}}^* - \beta^2\boldsymbol{\Phi}^* \end{bmatrix} \tag{13.41}$$

Graph theory provides schemes for determining which joints should be cut. It should be noted that the number of joint coordinates for a closed-loop system is larger than the number of degrees of freedom of that system. Therefore, for the equivalent open-chain representation of a closed-loop system, vector $\mathbf{y}$ contains more elements than twice the number of degrees of freedom of the actual system.

PROBLEMS

13.1 Use algorithm DI-1 in conjunction with subroutine RUNGK4. Assuming zero initial velocities, solve the equations of motion from the following problems:

(a) Prob. 9.6

(b) Prob. 9.7

13.2 For the unconstrained body shown in Fig. P. 13.2, the translational coordinate vector has components $\mathbf{r} = [0, 5, 4]^T$, and the local and the global coordinate systems are initially parallel. Let $a = 0.6$, $b = 0.4$, and $c = 0.2$, and assume a mass of 50. A force $\vec{f}$ with a constant magnitude of 12 acts at point A and remains perpendicular to plane ABC.

(a) Write the translation equations of motion (Eq. 11.8).

(b) Write the rotational equations of motion in terms of Euler parameters (Eq. 11.16).

(c) Use algorithm DI-2 in conjunction with subroutine RUNGK4 to solve these equations for a specified period of time. Assume initial velocities of zero.

(d) Monitor (plot) the constraint violations for $\mathbf{p}^T\mathbf{p} - 1 = 0$ and $\mathbf{p}^T\dot{\mathbf{p}} = 0$.

(e) Repeat the process for different values of Δt.

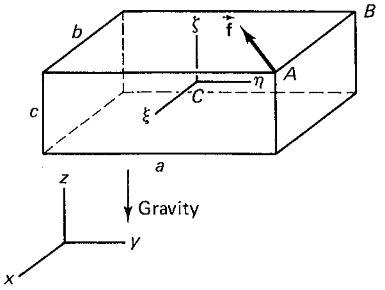

Figure P. 13.2

13.3 Repeat Prob. 13.2 and correct the computed values of $\mathbf{p}$ and $\dot{\mathbf{p}}$ according to Eqs. 13.9 and 13.11.

13.4 Repeat Prob. 13.2 and correct the Euler parameters for any constraint violations using the constraint violation stabilization technique.

13.5 Employ the coordinate partitioning method to Prob. 13.2. Assume $\mathbf{v} = [\mathbf{r}^T, \mathbf{e}^T]^T$ and $\mathbf{u} = [e_0]$. For values of e_0 close to zero, how would you determine the sign of e_0?

13.6 Repeat Prob. 13.2 and instead of Eq. 11.16, use Eq. 11.18. Then:

(a) use algorithm DI-3 to solve the equations.

(b) Monitor the constraint violation for $\mathbf{p}^T\mathbf{p} - 1 = 0$.

(c) Modify the program to eliminate or control any constraint violation.

13.7 Why is there no need to correct the computed values of $\dot{\mathbf{p}}$ in algorithm DI-3?

13.8 State the reasons why algorithm DI-3 is more efficient than algorithm DI-2.

13.9 Solve the constrained equations of motion from Prob. 9.8 by employing subroutine RUNGK4 in the following algorithms:

(a) Algorithm DI-4

(b) Algorithm CS-1

(c) Algorithm CP-1, assuming $\mathbf{v} = [x_1, y_1, \phi_1, \phi_2]^T$

(d) Algorithm JC-1, assuming $\theta = [x_1, y_1, \phi_1, \theta_{12}]^T$

13.10 In order to develop a planar dynamic analysis program using the joint coordinate method, the transformation formulas of Eqs. 13.36 through 13.38 must be derived explicitly for some of the standard kinematic joints. Derive these formulations for the three kinematic joints shown in Fig. P.13.10 and assume:

(a) $\theta_{ij} = \alpha$ for the revolute joint.

(b) $\theta_{ij} = d$ for the translational joint.

(c) $\theta_{ij} = [\alpha_1, \alpha_2]^T$ for the revolute-revolute joint.

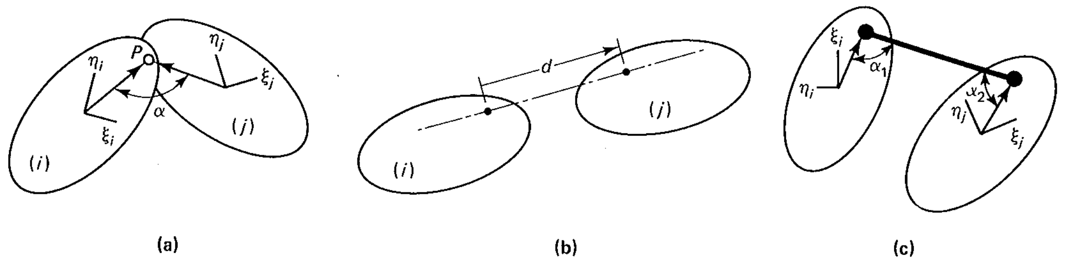

Figure P. 13.10

13.11 Modify the dynamic analysis program in Chap. 10 from the direct integration method (DI) and devise the following algorithms:

(a) Coordinate partitioning method

(b) Joint coordinate method

13.12 Derive the joint coordinate transformation formulas for a spatial revolute joint in a general case where the joint axis is not parallel to any of the local coordinate axes. *Hint:* Assume a second local coordinate system $\xi_{j'}\eta_{j'}\zeta_{j'}$ attached to body j and initially parallel to $\xi_i\eta_i\zeta_i$, as shown in Fig. P.13.12. The transformation matrix between $\xi_j\eta_j\zeta_j$ and $\xi_{j'}\eta_{j'}\zeta_{j'}$ is a constant matrix. The joint axis becomes the relative orientational and instantaneous axis of rotation between $\xi_i\eta_i\zeta_i$ and $\xi_{j'}\eta_{j'}\zeta_{j'}$. Therefore, $\mathbf{A}_{ij'}$ can be expressed as a function of $\theta_{ij'}$ and hence of θ_{ij}.

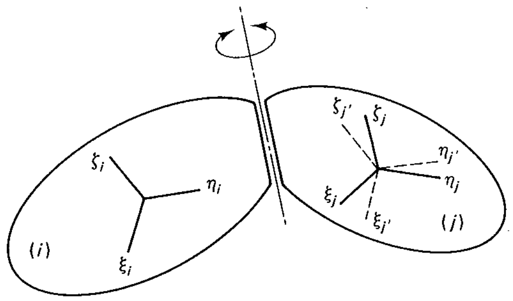

Figure P. 13.12

13.13 Repeat Prob. 13.12 for special cases where the joint axis is parallel to one of the coordinate axes of each body; e.g., where ξ_i is parallel to ξ_j. Estimate the percentage of computational efficiency that is gained in this formulation as compared with the general-case formulation.

13.14 Derive the joint coordinate transformation formulas for a spatial translational joint in a general case where the joint axis is not parallel to any of the coordinate axes.

13.15 Repeat Prob. 13.14 for special cases where the joint axis is parallel to one of the coordinate axes of each body; e.g., for ξ_i parallel to ξ_j. Estimate the percentage of computational efficiency that is gained in this formulation as compared with the general-case formulation.

13.16 Derive the joint coordinate transformation formulas for the following spatial kinematic joints (each joint allows two relative degrees of freedom):

(a) A universal joint

(b) A cylindrical joint

Derive the formulas for general and special cases. *Hint:* You may assume a third, fictitious body between the two bodies that has one relative DOF with each body.

13.17 Derive the joint coordinate transformation formulas for a spherical joint. Hint: Use a relative set of Euler parameters.

13.18 In order to develop a spatial dynamic analysis program, an approach similar to that of the planar program in Chap. 10 can be followed. For developing the first spatial analysis program, the following formulation and algorithm are suggested:

(a) Use the formulation of Eq. 11.49 for the equations of motion. Use the elements of Table 11.1 for the entries of the Jacobian matrix and vector $\gamma^{\#}$.

(b) Employ the direct integration algorithm DI-4. Do not be concerned initially with the constraint violation. After the initial version of the program is developed, an additional modification for constraint violation can be implemented,

(c) Employ a well-developed variable step/order predictor-corrector numerical integration package.

The constraint equations listed in Table 11.1 can be combined to model a variety of commonly used kinematic joints. The elements of this table can be easily programmed by employing elementary vector and matrix operations dealing with 3-vectors and 3×3 matrices. Rearrangement of the equations may yield an elementary operation on 3×4 matrices (such as **L** and **G** matrices). A careful organization of the program and the use of these elementary operations can easily yield a spatial dynamic analysis program.