7
Numerical Integration of the Equations of Motion

It was shown in Chapter 5 how the application of the laws of dynamics to constrained multibody systems leads to a set of differential algebraic equations (DAE). These can be transformed to second order ordinary differential equations (ODE) by proper differentiation of the kinematic constraint equations, by use of an independent set of coordinates, or by penalty formulations. A stable and accurate integration of both DAE and ODE is of great importance for the solution of the equations of motion. Although analytical solutions may be found for some simple cases, the number and complexity of the equations resulting from the majority of multibody systems requires numerical solutions. Because the theory of ordinary differential equations has been known for a long time, the stability, convergence, and accuracy of many methods have been studied in great detail. This has led to a wide use of these methods as compared to the differential algebraic equations, not so thoroughly known at this stage. As a consequence, many of the computer programs currently available for the computer-aided analysis and design of multibody systems rely on well-established methods for the solution of ODE.

In this chapter, we first present a brief introduction to the concepts involved in the solution of ODE with particular emphasis on the notion of stability, which is of fundamental importance for the integration of constrained multibody systems. This is followed by a description of the most widely-used methods from the viewpoint of their application to the numerical integration of the equations of motion of multibody systems. Finally, we also describe some of the methods used for the direct integration of DAE and draw some conclusions for real time analysis.

7.1 Integration of Ordinary Differential Equations

As shown in Chapter 5, the mixed differential algebraic equations of motion can be written after differentiation of the kinematic constraint equations as a set of n

second order ordinary differential equations. They can be expressed in a general way as

$$\ddot{\mathbf{q}} = \mathbf{F}(t, \mathbf{q}, \dot{\mathbf{q}}) \tag{7.1}$$

The solution $\mathbf{q}$ to this equation must satisfy also the initial conditions $\mathbf{q}(t_0)=\mathbf{q}_0$ and $\dot{\mathbf{q}}(t_0)=\dot{\mathbf{q}}_0$. The majority of general purpose computer programs for the solution of ODE integrate *first order* equations. For this purpose, the second order system of equation (7.1) may be written as a system of *2n* first order equations by defining the additional set of variables $\mathbf{s} = \dot{\mathbf{q}}$. Consequently (7.1) becomes:

$$\dot{\mathbf{s}} = \mathbf{F}(t, \mathbf{q}, \mathbf{s}) \tag{7.2}$$

$$\dot{\mathbf{q}} = \mathbf{s} \tag{7.3}$$

Introducing $\mathbf{y}^T \equiv \{\mathbf{q}^T, \mathbf{s}^T\}$ as a new vector variable that includes $\mathbf{q}$ and $\mathbf{s}$, one can write equation (7.2) in the form commonly used in textbooks dealing with ODE:

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}) \tag{7.4}$$

Function evaluation is the name of the process by which, given t and $\mathbf{y}$, the value of $\dot{\mathbf{y}}$ or $\mathbf{f}$ is computed. This obviously entitles the calculation of the accelerations as a function of the positions and velocities. Equation (5.67) may be used to perform the function evaluation with independent coordinates; equation (5.10) with Lagrange multipliers; and equation (5.37) for the penalty method.

7.1.1 General Background

Since in a general setting it is not possible to find closed form (analytic) solutions to the set of equations (7.4) describing the motion of a multibody system, we seek numerical methods (time-marching schemes) to approximate the solution at discrete times t_1, t_2, ... t_n. We will define the time step Δt as the difference $(t_{n+1}-t_n)$, and we will consider it constant during the integration process unless it is otherwise specified. If the function $\mathbf{f}$ is continuously differentiable with respect to t and $\mathbf{y}$ over the interval of interest, then there is a unique solution to (7.3) which also satisfies the initial conditions $\mathbf{y}_0 = \mathbf{y}(t_0)$. A proof of this theorem can be found in Ince (1956).

Taylor's Series Method. A first approach to the discrete approximate solution of (7.4), assuming that $\mathbf{f}$ is sufficiently differentiable with respect to t and $\mathbf{y}$, is to expand $\mathbf{y}$ in Taylor series as

$$\mathbf{y}(t) = \mathbf{y}_0 + \Delta t\, \mathbf{y}^{\mathrm{I}}(t_0) + \frac{\Delta t^2}{2!}\mathbf{y}^{\mathrm{II}}(t_0) + \ \ldots\ldots \tag{7.5}$$

where the total time derivatives of $\mathbf{y}$ (denoted by $(^{\mathrm{I}})$) can be found by differentiating $\mathbf{f}$ as:

$$\mathbf{y}^{\mathrm{I}} = \mathbf{f}(t, \mathbf{y}) \tag{7.6}$$

$$\mathbf{y}^{\mathrm{II}} = \mathbf{f}^{\mathrm{I}} = \mathbf{f}_t + \mathbf{f}_{\mathbf{y}}\,\mathbf{f} \tag{7.7}$$

Therefore the Taylor's expansion (7.4) can be written in terms of the function **f** and its derivatives. The difficulties involved in obtaining high order derivatives of **f** make this method unsuitable except for low order approximations resulting from the truncation of the higher order terms of equation (7.5). One such approximation is the well-known *Euler's method* which, given the solution at time step *n,* approximates the solution at step *n+1* as follows:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t\,\mathbf{f}(t_n, \mathbf{y}_n) \tag{7.8}$$

which is an *explicit* method, because the RHS does not depend on $\mathbf{y}_{n+1}$.

One can infer from Taylor's series expansion of (7.5) that the local truncation error resulting from Euler's method is $(\Delta t^2\mathbf{f}'(\xi)/2)$. It has been demonstrated (Gear (1971)), that Euler's method converges to the true solution as Δt decreases and that it is first order accurate. Although the local truncation error is proportional to the square of Δt , or $O(\Delta t^2)$, the global error consisting of error accumulated as the integration proceeds in time depends linearly on Δt, or $O(\Delta t)$. It is also demonstrated in Gear (1971) how an *r*th order method in which its global error is $O(\Delta t^r)$ has a local truncation error equal to $O(\Delta t^{r+1})$.

Accuracy is a very important aspect that needs to be considered when choosing a method to integrate the equations of motion. Euler´s method is extremely simple and easy to implement in a computer program, but it yields low accuracy and requires rather small time steps. In some cases, the time step Δt may need to be so small that the round-off errors become important and render the method useless. The search for higher order methods that will produce more accurate results as the time step is decreased would lead us to include more terms in Taylor's series (7.5) and approximate the solution at each time step with a more accurate polynomial interpolation. This is not a simple task since the high order derivatives of **f** are not readily available. One could use numerical or even symbolic differentiation; but the equations of motion of multibody systems are so involved that these tasks would not be of practical implementation in a general purpose computer program.

Stability. In addition to accuracy, another important aspect to be considered for the integration of ODE is *stability*. One can loosely define stability as the property of an integration method to keep the errors resulting in the integration process of a given equation bounded at subsequent time steps. An unstable method will make the integration errors increase exponentially, and an arithmetic overflow can be expected even after just a few time steps. Since stability depends not only on the given method but also on the type of problem, the test equation $y'=\lambda y$, where λ is a complex valued constant, is customarily used to characterize the stability properties of a given method. This characterization is performed by defining the set of values of λ and Δt for which the corresponding method is stable.

Mathematical literature has defined a large number of different kinds of stability. We only address three kinds of stability which are of use for the analysis of multibody systems and which will be referred to repeatedly later in the text.

1. Algorithms that are stable for some restricted range of values ($\lambda\Delta t$) are called *conditionally stable*. When using these methods, the time step should be chosen depending on the characteristics of the problem as defined by λ (or a set of λ). In the case of a nonlinear problem for which the value of λ changes with time, the algorithm may be stable for some part of the integration and unstable for another. Consequently, it is very important when using conditionally stable algorithms to know in advance the range of values ($\lambda\Delta t$) for which the method is stable and to compare it with the possible range of λ values of the given problem. For this purpose the *region of absolute stability* of a method is defined as that set of values ($\lambda\Delta t$) (area of the complex plane **C**) for which a perturbation in the solution y_n will produce a change in subsequent values which does not increase from step to step. The region of absolute stability is an intrinsic characteristic of the method which should be considered prior to the use of conditionally stable algorithms. As an example, Euler's method described above is conditionally stable and Δt must be less than $|\lambda|/2$ to assure stability.

2. Algorithms whose regions of absolute stability are $\mathrm{Re}(\lambda\Delta t) < -\eta$ and where η is a positive constant, are call *stiffly stable* methods. These methods are important when dealing with *stiff systems* of ordinary differential equations. The *stiff problems* contain a very large dispersion (several orders of magnitude) on the values of λ and arise either as a consequence of the type of formulation such as the penalty formulations, or simply because of the physical characteristics of the multibody system. The integration of these systems by conditionally stable algorithms should be avoided, because it would require such small time steps that the integration would become exceedingly expensive and even inaccurate due to round-off errors. Stiffly stable algorithms do not include the imaginary axis of the complex plane as part of their region of absolute stability. Multibody systems may have pure vibration modes whose λ may lie in the imaginary axis. For these cases the stiff stable methods are inadequate and a more stable family of algorithms, such as *A-stable* methods, should be required for the integration.

3. An algorithm is said to be *A-stable* if the solution to $y' = \lambda y$ tends to zero as $n \to \infty$ when the $\mathrm{Re}(\lambda) < 0$, which means that the numerical solution decays to zero whenever the corresponding exact solution decays to zero. An *A*-stable algorithm may be also defined as an algorithm whose region of absolute stability is the complete left half complex plane including the imaginary axis. The most important consequence of the *A*-stability property is that there is no limitation on the size of Δt for the stability of the integration process: A-stable algorithms have also been called *unconditionally stable* in the linear setting. It is apparent that this property is very important and generally desired

in the integration of multibody and other engineering systems, since the analyst would only have to be concerned with the step size for accuracy purposes and not for stability. The use of *A*-stable algorithms may become strictly necessary for stiff problems, where the imaginary axis must be part of the region of absolute stability. An important subclass of the *A*-stable methods is the *L-stable* ones. A method is said to be *L-stable* (also called *stiff A-stable*), if it is *A*-stable and when applied to the solution of $y'=\lambda y$ with $\mathrm{Re}(\lambda)<0$, it gives $y_{n+1}=A(\lambda\Delta t)y_n$ where $A(\lambda\Delta t)$ tends to zero as $\mathrm{Re}(\lambda)\to\infty$. The difference between *L*-stability and *A*-stability is that the former damps out the response of the stiff components (that is equations with high λ) very rapidly, almost in only one time step. This property is interesting and very applicable in those cases with spurious stiff equations that may arise as a consequence of the formulation or modeling process. However, caution should be practiced when using these methods since they tend to artificially damp out part of the response of interest corresponding to values of λ that are not that high.

7.1.2 Runge-Kutta Methods

Euler's method is not useful because of its low accuracy. Higher order Taylor's series are also impractical because of the difficulty involved in obtaining the derivatives of $\mathbf{f}(t, \mathbf{y})$. It is possible, however, to develop one step methods that match the accuracy of the higher order Taylor's series methods by sequentially computing the function $\mathbf{f}(t, \mathbf{y})$ at several points within the time interval. The task of computing higher order derivatives is replaced by function evaluations at a given number of points. Such methods are called the *Runge-Kutta* methods. One can refer to classical books in numerical analysis (Carnahan et al. (1969), Conte and Boor (1972)) for a detailed description and derivation of these methods.

One of these algorithms which is commonly used in engineering applications is the second order explicit Runge-Kutta method defined by:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}(\mathbf{k}_1 + \mathbf{k}_2) \tag{7.9}$$

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{y}_n) \tag{7.10}$$

$$\mathbf{k}_2 = \mathbf{f}(t_n + \Delta t, \mathbf{y}_n + \Delta t\,\mathbf{k}_1) \tag{7.11}$$

which is also called the *improved Euler's method*, *modified trapezoidal method*, or *Heun's method*. Note that two function evaluations are required per time step, which in the case of multibody systems implies the solution of the equations of motion to obtain the accelerations twice at the given time step. The method is also explicit because $\mathbf{k}_1$ does not depend on $\mathbf{k}_2$, and neither one depends on $\mathbf{y}_{n+1}$.

One of the most widely used Runge-Kutta methods is the fourth order method which requires four function evaluations per time step:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{6}\left(\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3 + \mathbf{k}_4\right) \tag{7.12}$$

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{y}_n) \tag{7.13}$$

$$\mathbf{k}_2 = \mathbf{f}(t_n + \frac{\Delta t}{2}, \mathbf{y}_n + \frac{\Delta t}{2}\mathbf{k}_1) \tag{7.14}$$

$$\mathbf{k}_3 = \mathbf{f}(t_n + \frac{\Delta t}{2}, \mathbf{y}_n + \frac{\Delta t}{2}\mathbf{k}_2) \tag{7.15}$$

$$\mathbf{k}_4 = \mathbf{f}(t_n + \Delta t, \mathbf{y}_n + \Delta t\,\mathbf{k}_3) \tag{7.16}$$

This method is *explicit*, because all the $\mathbf{k}_i$ depend on previous values already calculated. Otherwise, the method is said to be *implicit*. Solving for $\mathbf{y}_{n+1}$ at each time step requires an iterative process for the solution of a set of nonlinear equations. A general *r*-stage implicit Runge-Kutta method requires *r* function evaluations and has the following general form:

$$\mathbf{k}_i = \mathbf{f}(t_n + c_i\,\Delta t, \mathbf{y}_n + \Delta t \sum_{j=1}^{r} a_{ij}\,\mathbf{k}_j) \tag{7.17}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \sum_{i=1}^{r} b_i\,\mathbf{k}_i \tag{7.18}$$

If $a_{ij}=0$ for $j \geq i$, then the values $\mathbf{k}_i$ can be computed explicitly from the preceding values $\mathbf{k}_1, \mathbf{k}_2, \ldots \mathbf{k}_{i-1}$, and the method is *explicit*. If $a_{ij}=0$ for $j>i$ and some $a_{ii}\neq 0$, the method is called *semi-explicit*. If all $a_{ii}\neq 0$ the method is *diagonally implicit*. Butcher (1964) has shown that it is possible to achieve order $2r$ for an implicit *r*-stage method. The two-stage method with $a_{11}=a_{12}=0$, $a_{21}=a_{22}=1/2$, $c_1=0$, $c_2=1$, and $b_1=b_2=1/2$ corresponds to the *trapezoidal rule* which according to Dalhquist (1963) is the most accurate second order *A*-stable method. This method can also be expressed in finite difference form as

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}\left[\mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})\right] \tag{7.19}$$

Norsett (1974) proposed a diagonally implicit two-stage Runge-Kutta family of methods for which $a_{11}=a_{22}=\alpha$, $a_{21}=1-2\alpha, a_{12}=0$; $b_1=b_2=1/2$ and $c_1=\alpha$, and $c_2=1-\alpha$. For general values of α, the method is second order accurate. For $\alpha=1\pm\sqrt{2}/2$ the method is *L*-stable. For $\alpha=(3+\sqrt{6})/2$ the method is third order and *A*-stable and corresponds to the approximation used by Calahan (1968). Good results of the Norsett algorithms have been reported by Smith (1975) who used them for the solution of first as well as second order ODEs.

The explicit Runge-Kutta methods are easy to implement, because they only require function evaluations, and are self-starting, meaning that they do not need any other algorithm or technique to start the integration process. However, they are only conditionally stable. Figure 7.1 shows the area of absolute stability of the fourth order explicit Runge-Kutta method of equation (7.12). In addition,

Figure 7.1. Areas of absolute stability of some fourth order methods.

they require several function evaluations per time step. This turns to be computationally expensive when considering large systems of nonlinear equations, such as those arising in the analysis of multibody systems.

Implicit Runge-Kutta methods are more stable (the r-stage methods of order $2r$ are A-stable), and much more accurate than the explicit ones. However, except for the simplest trapezoidal rule, they are more difficult to implement and much more expensive to use. A set of nonlinear equations that involve repeated function evaluations needs to be solved at each time step. A disadvantage of both the implicit and explicit methods is that it is very difficult to evaluate bounds for the accumulated or propagated error. This makes error control and time-step adjustments rather involved with this kind of method. An algorithm that uses error control and time step adjustment for the Runge-Kutta method of order four is due to Felhberg (1970). The implementation is explained in detail by Burden et al. (1989).

7.1.3 Explicit and Implicit Multistep Methods

The Taylor's series and the Runge-Kutta Methods are called *single step* methods because they only require information on the time step $(n, n+1)$ to advance to the next step. When information of other previous steps is also used, the resulting method is called *multistep*. These methods are very simply derived by integrating

the differential equation $\mathbf{y}'=\mathbf{f}(t, \mathbf{y})$ from t_{n-p} to t_{n+1}, where p is a positive integer.

$$\mathbf{y}_{n+1} = \mathbf{y}_{n-p} + \int_{t_{n-p}}^{t_{n+1}} \mathbf{f}(t, \mathbf{y})\, dt \tag{7.20}$$

In order to carry out the integration, one can approximate the function $\mathbf{f}(t, \mathbf{y})$ for instance by using backward finite differences, with the values $\mathbf{f}(t, \mathbf{y})$ calculated at previous time steps. One is referred to Carnahan et al. (1969) and Conte and Boor (1972) for a detailed description of this process. The end result is a family of methods that depends on the order of approximation of $\mathbf{f}(t, \mathbf{y})$ and on the value of p. If the approximation for $\mathbf{f}(t, \mathbf{y})$ includes the value at $n+1$, the method is called implicit, and explicit otherwise. Implicit methods are much more accurate and stable than the explicit ones. However, they are also more difficult to use, since $\mathbf{y}_{n+1}$ can not be solved for explicitly and an iteration process is required.

The general form of the multistep methods is given by the following expression:

$$\sum_{i=0}^{p+1} \alpha_i\, \mathbf{y}_{n+1-i} + \sum_{i=0}^{k} \Delta t\, \beta_i\, \mathbf{f}(t_{n+1-i}, \mathbf{y}_{n+1-i}) = 0 \tag{7.21}$$

The α_i and β_i are parameters that define the method. If $\beta_0=0$ the term $\mathbf{f}(t_{n+1}, \mathbf{y}_{n-1})$ does not appear in the difference equation and the method is explicit. If $\beta_0 \neq 0$ the method is implicit. The methods resulting from $\beta_i=0$ for all $i \geq 1$ are called *backward-difference methods*.

One can readily see that the trapezoidal rule defined by (7.19) corresponds to $\alpha_0=-\alpha_1=1$ and $\beta_0=\beta_1=-1/2$. In addition, the more terms involved in (7.21) the better the approximation and the accuracy of the method will be. However, increasing the number of terms also leads to a larger amount of information from previous steps that needs to be stored. This can be a serious disadvantage in the analysis of multibody systems with a large number of equations, since it may lead to swapping of information from the in-core to out-of-core computer storage.

Three important conclusions known as Dahlquist's theorem (1963) may be

Table 7.1. Coefficients of the Adams-Bashforth Methods.

k	α_0	α_1	β_0	β_1	β_2	β_3	β_4	β_5
1	1	1	0	1				
2	2	2	0	–3	1			
3	12	12	0	–23	16	–5		
4	24	24	0	–55	59	–37	9	
5	720	720	0	–1901	2774	–2616	1274	–251

Table 7.2. Coefficients of the Adams-Moulton Methods.

k	α_0	α_1	β_0	β_1	β_2	β_3	β_4
1	1	1	–1				
2	2	2	–1	–1			
3	12	12	–5	–8	1		
4	24	24	–9	–19	5	–1	
5	720	720	–251	–646	264	–106	19

mentioned at this stage, which can be summarized as follows:

1. There is not an *A*-stable explicit linear multistep method.
2. The order of accuracy of a linear multistep *A*-stable method cannot exceed two.
3. The second order accurate *A*-stable linear multistep method with the smallest error constant is the trapezoidal rule.

The widely known Adams-Bashforth methods are explicit with $p=0$. Table 7.1 shows the parameters corresponding to different orders of accuracy. The most commonly used Adams-Bashforth method is the fourth order method that takes the following form:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{24}(55\,\mathbf{f}_n - 59\,\mathbf{f}_{n-1} + 37\,\mathbf{f}_{n-2} - 9\,\mathbf{f}_{n-3}) \tag{7.22}$$

with local truncation error $\mathbf{E} = \left(251/720\right)\Delta t^{\,5}\,\mathbf{f}^{\mathrm{IV}}\left(\xi\right)$. Being explicit, these methods are conditionally stable, and the area of absolute stability for the fourth order method is illustrated in Figure 7.1.

The family of implicit methods obtained for $p=0$ are called the *Adams-Moulton* methods. Table 7.2 depicts the coefficients of several of the methods. The widely used fourth order method is

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{24}(9\,\mathbf{f}_{n+1} + 19\,\mathbf{f}_n - 5\,\mathbf{f}_{n-1} + \mathbf{f}_{n-2}) \tag{7.23}$$

with the local truncation error $\mathbf{E} = -\left(19/720\right)\Delta t^5\,\mathbf{f}^{\mathrm{IV}}(\xi)$.

The use of (7.23) requires an iteration process that is usually initiated using (7.22) to obtain an estimate or prediction of the value of $\mathbf{f}_{n+1}$. It is worth paying this price, since the Adams-Moulton method is much more accurate than the Adams-Bashforth as seen by the respective constants of the error terms. Note also that the (7.22) is a fourth order method with only one function evaluation per time step as compared to the four evaluations required by the explicit fourth order Runge-Kutta Method. As a consequence of Dahlquist's theorem (1975), the

Adams-Moulton method is also conditionally stable. The area of absolute stability for the fourth order method is also illustrated in Figure 7.1.

Predictor-Corrector Iteration. The use of an implicit multistep method leads to much more accurate results than the explicit ones. Since $\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$ is nonlinear for the case of multibody systems, a set of nonlinear equations is required for the solution of $\mathbf{y}_{n+1}$. The necessary iteration process can be initiated by using an explicit method of the same order to obtain a predictor. For the case of the Adams-Moulton method of (7.23) a good predictor is the Adams-Bashforth method of equation (7.22). One can obtain a predicted value $\mathbf{y}^0_{n+1}$ with which $\mathbf{f}_{n+1}$ of (7.23) can be computed and from these a new approximation $\mathbf{y}^1_{n+1}$. This value can be reentered again in (7.23) to obtain a new estimate $\mathbf{y}^2_{n+1}$, and so on until the difference of two consecutive values is smaller than a prescribed tolerance. This process can be summarized in the following algorithm:

Algorithm 7-1

1. Use the explicit method (predictor) to obtain $\mathbf{y}^0_{n+1}$
2. Use the implicit method (corrector) for the successive $\mathbf{y}^k_{n+1}$, $k = 1, 2, \ldots$
3. Check $\dfrac{\mathbf{y}^k_{n+1} - \mathbf{y}^{k-1}_{n+1}}{\mathbf{y}^k_{n+1}} <$ tolerance
4. If 3 is true, go to the next step. Otherwise go to step 2.

The predictor-corrector algorithm outlined above is equivalent to performing a fixed point iteration for the solution of the nonlinear equations on $\mathbf{y}_{n+1}$. This iteration is rather slow, since the rate of convergence is only linear in the neighborhood of the solution. It can also be shown (Carnahan et al. (1969)) that the time step required for this iteration to converge has to satisfy the following condition:

$$\Delta t < \frac{C}{\partial \mathbf{f}/\partial \mathbf{y}} \tag{7.24}$$

where C is a constant that depends on the type of explicit method being used. One may see how Δt is limited not only by the stability criteria but also by the convergence of the iteration process that is required at each time step.

It is also possible to use a Newton-Raphson or quasi Newton-Raphson iteration by providing a derivative of the function $\mathbf{f}(t, \mathbf{y})$ with respect to $\mathbf{y}$. In this way the convergence rate increases from linear to quadratic. Some integrators have the option of computing $\mathbf{f}_\mathbf{y}$ by finite difference approximations or allowing the user to provide it. In this last case the calculation of $\mathbf{f}_\mathbf{y}$ is entirely equivalent to linearize the equations of motion as explained in Chapter 9.

Error Estimates. A very positive advantage of the use of predictor-corrector algorithms is the possibility of finding an estimate of the error at each time step. This can be accomplished using the local truncation errors of both predictor-cor-

rector methods. For example, if $\mathbf{y}(t_{n+1})$ represents the exact value of $\mathbf{y}$ at time t_{n+1} the errors produced by the predictor and corrector of (7.22) and (7.23) will respectively be:

$$\mathbf{y}(t_{n+1}) - \mathbf{y}^0_{n+1} = \frac{251}{720}\,\Delta t^5\,\mathbf{f}^{IV}(\xi_1) \tag{7.25}$$

$$\mathbf{y}(t_{n+1}) - \mathbf{y}^1_{n+1} = -\frac{19}{720}\,\Delta t^5\,\mathbf{f}^{IV}(\xi_2) \tag{7.26}$$

For a sufficiently small time step one can assume that $\mathbf{f}^{IV}$ remains constant and by subtracting (7.25) and (7.26), one can get the following estimate of the error after one iteration:

$$\mathbf{y}(t_{n+1}) - \mathbf{y}^1_{n+1} \cong \frac{1}{14}\,(\mathbf{y}^1_{n+1} - \mathbf{y}^0_{n+1}) \tag{7.27}$$

This error estimate allows for the possibility of general purpose integrators with time step adjustments. If the error is larger than a pre-specified tolerance, it may be more convenient to reduce the time step rather than continue the fixed point iteration. However, if the error is small, the time step can be enlarged to achieve a faster integration. The most expensive part of the integration process is the function evaluation $\mathbf{f}(t, \mathbf{y})$ which requires the solution of the dynamic equations. These function evaluations can be composed in the case of multibody systems of a large number of coupled nonlinear equations.

7.1.4 Comparison Between the Runge-Kutta and the Multistep Methods

The Runge-Kutta and multistep methods can be used for the solution of the equations of motion of multibody systems. The Runge-Kutta methods are single-step and therefore self-starting. In addition they need a minimum amount of storage requirements. However, they require a larger number of function evaluations (four for the fourth order method). Due to the difficulty of estimating their local truncation errors, the time step adjustment can only be performed by integrating with two different time steps. This is computationally expensive.

The multistep methods require a smaller amount of function evaluations, particularly if the time step is chosen so that the number of predictor-corrector iterations per step is kept below two or three. Error estimates are easily provided, and step-size adjustments can be performed with no difficulties. Being a multi-step method, they are not self-starting, requiring the help of a single-step method to start the integration. In the case of multibody system, the forcing terms may have jumps or discontinuities. While this does not affect a single-step method, a multistep method will need to be reinitialized. The necessary historical data pool is larger for these methods thus requiring a larger amount of storage.

Regardless of these difficulties and since the total cost of integration is mostly due to the number of function evaluations, the general purpose integrators that are based on predictor-corrector Adams-Moulton-Bashforth methods

Table 7.3. Parameter values of the Gear's Stiffly Stable Methods.

p	β_0	α_0	α_1	α_2	α_3	α_4
1	–1	1	–1			
2	–2	3	–4	1		
3	–6	11	–18	9	–2	
4	–12	25	–48	36	–16	3

(Gear (1971), and Shampine and Gordon (1975)) have been customarily used for the integration of the equations of motion of multibody systems. Both explicit Runge-Kutta methods and multistep methods suffer from being only conditionally stable. This poses serious limitations in the size of the time steps in those systems with a large dispersion on the values of λ, the so-called *stiff systems*, and also in those cases in which the forcing terms contain high frequency components.

Stiff Systems of Equations. A stiff system of differential equations is characterized by a large dispersion of the values λ of each of the individual equations. The stiffness can be produced by the physical characteristics of the multibody system (components with large differences in their masses, stiffness and/or damping). However in many other instances, stiffness is numerically induced due to either the discretization process, the large number of components and equations of motion, or sudden or accumulated violations in the constraint conditions. Gear (1971) developed a family of variable order *stiffly-stable* algorithms for the solution of stiff problems. Table 7.3 illustrates the parameter values of such methods. Each method is a backward-difference multistep algorithm. The second order method is *A*-stable (it actually is *L*-stable) but much less accurate than the trapezoidal rule which is the most accurate second order *A*-stable method. For orders larger than two, the regions of stability do not include the complete imaginary axis. This feature does not appear appropriate for typical multibody systems with oscillatory motion. In second order systems, the backward difference methods may introduce an excess of artificial damping in the interesting part of the response. They tend to yield better results in first order problems with exponentially decaying responses; such as those in heat conduction, mass transport, and ground water flow (Wood (1990)).

In the case of constrained multibody systems with stiff equations, the analyst may benefit by the choice of an *A*-stable method (unconditionally stable) and not have to worry about the stiff effects produced by either the physical characteristics of the multibody system, the forcing terms, or the large number of equations of motion. Unconditionally stable methods are widely used in the field of structural dynamics because of the high frequencies produced by the finite element discretization, even to such an extent that *A*-stability is considered as a necessary condition for any new method to gain acceptance within the structural engineer-

ing community. The next section reviews a few of the mostly used *A*-stable methods in structural dynamics. Some of them have also been successfully used in the integration of the equations of motion of multibody systems (See Bayo et al. (1991), and Chapter 8).

7.1.5 Newmark Method and Related Algorithms

Newmark (1959) proposed what has become one of the most popular family of algorithms for the solution of problems in structural dynamics. His method relies on the following interpolations that relate positions, velocities, and accelerations from step n to $n+1$:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t \left[(1-\gamma)\, \mathbf{a}_n + \gamma\, \mathbf{a}_{n+1} \right] \tag{7.28}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t\, \mathbf{v}_n + \frac{\Delta t^2}{2} \left[(1-2\beta)\, \mathbf{a}_n + 2\beta\, \mathbf{a}_{n+1} \right] \tag{7.29}$$

where $\mathbf{x}_n$, $\mathbf{v}_n$, and $\mathbf{a}_n$ are approximations to the position, velocity, and acceleration vectors at time step n; and β and γ are the parameters that define the method. The method is implicit, and *A*-stability is guaranteed for $2\beta \geq \gamma \geq 1/2$ (Bathe (1982)). The trapezoidal rule is a particular case of this family for which $\beta = 1/4$ and $\gamma = 1/2$. This case also corresponds to the assumption that the acceleration is constant over the time interval $[t_n, t_{n+1}]$ and equal to $(\mathbf{a}_n + \mathbf{a}_{n+1})/2$. This method is also known as the *average acceleration method*. For any other set of β and γ values within the range of *A*-stability, the degree of accuracy of the method degrades to order one. The linear acceleration method, in which a linear variation of the acceleration in the time interval $[t_n, t_{n+1}]$ is assumed, corresponds to the case $\beta = 1/6$ and $\gamma = 1/2$. This method is conditionally stable and has little practical importance; however, it was used as the basis for another important method known as the *Wilson-θ* method (Bathe (1982)).

Similar to the multistep methods, the implicit algorithm of equations (7.28) and (7.29) can be used in a predictor-corrector fashion with fixed point iteration. This is not the way it is customarily applied in structural dynamics. Rather, the interpolations of (7.28) and (7.29) are directly introduced into the equations of motion. This leads to a set of algebraic equations which can be linear or nonlinear depending on the type of problem, with $\mathbf{a}_{n+1}$ as the resulting unknowns. The algebraic equations can alternatively be solved with $\mathbf{x}_{n+1}$ as primary unknowns, by substituting $\mathbf{a}_{n+1}$ and $\mathbf{v}_{n+1}$ in terms of $\mathbf{x}_n$, $\mathbf{v}_n$, $\mathbf{a}_n$ and $\mathbf{x}_{n+1}$. Equations (7.28) and (7.29) can then be written as:

$$\mathbf{a}_{n+1} = \frac{1}{\beta\, \Delta t^2} (\mathbf{x}_{n+1} - \mathbf{x}_n) - \frac{1}{\beta\, \Delta t}\, \mathbf{v}_n - \left(1 - \frac{1}{2\beta}\right) \mathbf{a}_n \tag{7.30}$$

$$\mathbf{v}_{n+1} = \frac{\gamma}{\beta\, \Delta t} (\mathbf{x}_{n+1} - \mathbf{x}_n) - \left(\frac{\gamma}{\beta} - 1\right) \mathbf{v}_n - \left(\frac{\gamma}{2\,\beta} - 1\right) \Delta t\, \mathbf{a}_n \tag{7.31}$$

The procedure can be explained by applying these equations to a linear structural dynamics problem which has the following form (Bathe (1982) or Hughes (1987)):

$$\mathbf{M}\ \mathbf{a}_{n+1} + \mathbf{C}\ \mathbf{v}_{n+1} + \mathbf{K}\ \mathbf{x}_{n+1} = \mathbf{F}(t) \tag{7.32}$$

with **M**, **K**, and **C** as the mass, stiffness, and damping matrices, respectively; and **F** as the vector of externally applied forces. The substitution of equations (7.30) and (7.31) into (7.32) yields

$$\begin{aligned} &\left[\frac{1}{\beta\,\Delta t^{2}}\mathbf{M} + \frac{\gamma}{\beta\,\Delta t}\mathbf{C} + \mathbf{K}\right]\mathbf{x}_{n+1} = \\ &= \mathbf{F}(t) + \mathbf{M}\left[\frac{1}{\beta\,\Delta t^{2}}\mathbf{x}_n + \frac{1}{\gamma\,\Delta t}\mathbf{v}_n + (1-\frac{1}{2\,\beta})\,\mathbf{a}_n\right] + \\ &+ \mathbf{C}\left[\frac{\gamma}{\beta\,\Delta t}\mathbf{x}_n + (\frac{\gamma}{\beta}-1)\,\mathbf{v}_n + (\frac{\gamma}{2\,\beta}-1)\,\Delta t\,\mathbf{a}_n\right] \end{aligned} \tag{7.33}$$

Once the constant matrix that multiplies $\mathbf{x}_{n+1}$ on the left-hand side has been triangularized, the solution for the displacements at each time step only requires the formation of the right-hand side of (7.33) plus a forward reduction and a backwards substitution. This implementation is by far more efficient than the iterative process required with a fixed point iteration which would require a function evaluation per iteration with several iterations per time step. This substitution has also been carried out for nonlinear problems in structural dynamics. The resulting set of nonlinear algebraic equations in $\mathbf{x}_{n+1}$ is customarily solved by either Newton-Raphson iteration which has a quadratic convergence in the neighborhood of the solution, secant methods, or quasi-Newton methods (Bathe (1982)).

The same idea can be applied in principle to the equations of motion of multibody systems which take the following general form:

$$\mathbf{M}(\mathbf{q})\ \ddot{\mathbf{q}} + \mathbf{P}(\mathbf{q},\ \dot{\mathbf{q}}) = \mathbf{F}(t) \tag{7.34}$$

However, the substitution of (7.30) and (7.31) in (7.34) usually yields a highly nonlinear set of equations in the unknowns $\mathbf{q}_{n+1}$. The tangent or quasi-tangent matrices necessary for the solution of the resulting set of equations through Newton-Raphson iteration may be of such complexity, that one may end up being forced to use the simpler but less efficient predictor-corrector algorithms with the fixed point iteration outlined above. For those cases in which the formation of the tangent or quasi-tangent matrix is possible, this way of integrating the equations of motion tends to be much more efficient than the predictor-corrector schemes which only have linear convergence in the neighborhood of the solution. Such cases and special implementations will be dealt with in Chapter 8.

The most accurate *A*-stable algorithm of the Newmark family is the *trapezoidal rule* ($\beta=1/4$ and $\gamma=1/2$) which is energy preserving for linear systems. It does not damp out any of the frequency content of the system during the integra-

tion process (for a detailed analysis of the accuracy and stability of the Newmark method refer to Bathe (1982) and Hughes (1987)). For $\gamma > 1/2$, the *A*-stability is preserved and artificial damping is introduced. However, the accuracy is reduced to first order. This artificial damping is in some instances necessary because the mathematical model may contain spurious high frequency content that needs to be damped out.

There is another reason why artificial damping may be necessary. The trapezoidal rule is unconditionally stable for linear problems. However, this property is not maintained in the nonlinear regime. As examples of instabilities, Hughes (1976) reported pathological energy growth in structural dynamic problems with bilinear softening material. Cardona and Geradin (1989) and Bayo et al. (1991) also reported instability of the trapezoidal rule in the integration of constrained multibody systems. Gourlay (1970) considered the nonlinear scalar equation $\dot{y}+\lambda(y)y = 0$. He reported that when $\lambda_n > \lambda_{n+1}$, the trapezoidal rule becomes conditionally stable with critical time step $\Delta t \leq 4/(\lambda_n - \lambda_{n+1})$ (See example below). One way of circumventing this problem is to use the *midpoint rule* and related algorithms (Simo and Wong (1991)) that preserve unconditional stability in the nonlinear regime. The midpoint rule also uses the same interpolations defined by the equations (7.28) and (7.29), with the difference that equilibrium is computed at the middle of the time step rather than at the end. Accordingly equation (7.34) becomes

$$\mathbf{M}(\mathbf{q}_{n+1/2})\,\ddot{\mathbf{q}}_{n+1/2} + \mathbf{P}(\mathbf{q}_{n+1/2},\,\dot{\mathbf{q}}_{n+1/2}) = \mathbf{F}(t_{n+1/2}) \tag{7.35}$$

with $t_{n+1/2} = t_n + \Delta t/2$, $\mathbf{q}_{n+1/2} = (\mathbf{q}_{n+1}+\mathbf{q}_n)/2$, and the same for $\dot{\mathbf{q}}_{n+1/2}$ and for $\ddot{\mathbf{q}}_{n+1/2}$.

The instability of the trapezoidal rule in nonlinear problems can also be avoided by: first, introducing the energy constraints (Hughes (1983)) or the kinematic velocity constraints (Bayo et al. (1991) or Section 8.5); and secondly, by adding some artificial damping through the numerical integration scheme. We discuss in the next sections a couple of these methods.

Example 7.1

Demonstrate that the trapezoidal rule is conditionally stable in the nonlinear setting for $\lambda_n > \lambda_{n+1}$ and that the critical time step becomes $\Delta t \leq 4/(\lambda_n - \lambda_{n+1})$.

Consider the equation of motion $\dot{y}+\lambda(y)y = 0$. (i)

The trapezoidal rule is given by

$$y_{n+1} = y_n + \frac{\Delta t}{2}\left[\dot{y}_n + \dot{y}_{n+1}\right] \tag{ii}$$

Substituting (i) into (ii) and grouping terms together one obtains

$$y_{n+1}\,(2 + \lambda_{n+1}\,\Delta t) = y_n\,(2 - \lambda_n\,\Delta t) \tag{iii}$$

and the condition for stability becomes

$$\frac{y_{n+1}}{y_n} = \frac{|2 - \lambda_n \Delta t|}{|2 + \lambda_{n+1} \Delta t|} \leq 1 \qquad \text{(iv)}$$

which leads to the critical time step $\Delta t \leq 4/(\lambda_n - \lambda_{n+1})$.

α-Method (Hilber, Hughes and Taylor). Since numerical damping can not be introduced with the Newmark family unless the order of accuracy is degraded, Hilber, Hughes, and Taylor (1977) proposed the α-method which can still maintain second order accuracy and *A*-stability and in addition introduce variable damping depending on the value of the parameter α. This method uses the same finite difference expressions of the Newmark method (7.28) and (7.29), however the equations of motion of the type (7.34) are modified in the following form:

$$\mathbf{M}(\mathbf{q}_{n+1})\,\ddot{\mathbf{q}}_{n+1} + (1+\alpha)\,\mathbf{P}(\mathbf{q}_{n+1}, \dot{\mathbf{q}}_{n+1}) - \alpha\,\mathbf{P}(\mathbf{q}_n, \dot{\mathbf{q}}_n) = \mathbf{F}(t_{n+\alpha}) \qquad (7.36)$$

where $t_{n+\alpha}=(1+\alpha)t_{n+1}-\alpha t_n$. For $\alpha=0$ this method reduces to the Newmark family. The best choices for the parameter α lie in the interval [–1/3, 0]. One can reduce this three parameter family of methods to only one parameter by choosing $\gamma = (1-2\alpha)/2$ and $\beta = (1-\alpha)^2/4$. This choice of parameters results in a second order accurate, *A*-stable algorithm with variable artificial damping, depending on the value of α. The smaller the value of α the larger the damping is. When $\alpha=0$, there is no damping. The resulting method is the trapezoidal rule. A comparison of the accuracy of the Newmark method and the α-method as measured by the period elongation and algorithmic damping ratios can be seen in Hughes (1987) and Hilber (1976). Cardona and Geradin (1989) used the α-method successfully in the integration of constrained multibody systems.

Wilson-θ and Collocation Methods. In order to make the linear acceleration method (Newmark's method with $\beta=1/6$ and $\gamma=1/2$) unconditionally stable, Wilson (1968) proposed the idea of applying the equations of motion not at time $t+\Delta t$ but at $t+\theta\Delta t$, where $\theta>1.37$ for unconditional stability. This method, when applied to the type of equations of multibody systems (7.22), yields the following expressions:

$$\mathbf{M}(\mathbf{q}_{n+\theta})\,\ddot{\mathbf{q}}_{n+\theta} + \mathbf{P}(\mathbf{q}_{n+\theta}, \dot{\mathbf{q}}_{n+\theta}) = \mathbf{F}(t_{n+\theta}) \qquad (7.37)$$

$$\ddot{\mathbf{q}}_{n+\theta} = (1-\theta)\,\ddot{\mathbf{q}}_{n+\theta} + \theta\,\ddot{\mathbf{q}}_n \qquad (7.38)$$

$$\mathbf{F}_{n+\theta} = (1-\theta)\,\mathbf{F}_{n+\theta} + \theta\,\mathbf{F}_n \qquad (7.39)$$

with the displacement and velocity expressions given by the Newmark interpolation (7.18) with $\beta=1/6$, and $\gamma=1/2$.

The *collocation methods* were proposed by Hilber (1976) as a three-parameter family of methods resulting from the combination of the Newmark interpolation with the Wilson-θ method. The resulting equations are composed of equations

Figure 7.2. Double pendulum with rotational springs.

(7.37)-(7.39) plus the Newmark finite difference equations that now include the θ parameter (also called the collocation parameter), as follows:

$$\dot{\mathbf{q}}_{n+\theta} = \dot{\mathbf{q}}_n + \theta\,\Delta t\left[(1-\gamma)\,\ddot{\mathbf{q}}_n + \gamma\,\ddot{\mathbf{q}}_{n+\theta}\right] \tag{7.40}$$

$$\mathbf{q}_{n+\theta} = \mathbf{q}_n + \theta\,\Delta t\,\dot{\mathbf{q}}_n + \frac{(\theta\,\Delta t)^2}{2}\left[(1-2\,\beta)\,\ddot{\mathbf{q}}_n + 2\,\beta\,\ddot{\mathbf{q}}_{n+\theta}\right] \tag{7.41}$$

This family of methods now encompasses the Newmark method for which $\theta=1$ and the Wilson-θ for which $\beta=1/6$ and $\gamma=1/2$. Second order accuracy and unconditional stability are assured if the following conditions on γ, β, and θ are met:

$$\gamma = 1/2, \quad \theta \geq 1, \quad \frac{\theta}{2(\theta+1)} \geq \beta \geq \frac{2\,\theta^2 - 1}{4\,(2\theta^3 - 1)} \tag{7.42}$$

Similar to the α-method, the collocation method also allows for controllable algorithm damping. A good comparison on accuracy and damping characteristics of both methods is included in Hughes (1987) and Hilber (1976). However, as reported by Goudreau and Taylor (1973), the method suffers from a tendency to a spurious *overshoot* of the response in the first few steps of integration. Such disadvantage is not present in either the Newmark or the α-methods.

Example 7.2

Double pendulum with rotational springs. With this example, we show the stability and convergence properties of the trapezoidal rule when its difference equations are introduced in the equations of motion as explained in Section 7.1.4. The positions are the primary variables, and the resulting set of equations are solved by means of Newton-Raphson iteration. Figure 7.2 illustrates a double pendulum that has two elements of unit mass m = 1 and unit length with rotational springs at the joints of value k. The joints are subjected to equal initial velocities of value 100

rad/sec. The system is then analyzed using the two independent joint rotations φ_1 and φ_2 as coordinates for a total time of 20 seconds.

The kinetic energy of this system is

$$T = \frac{1}{2} \begin{Bmatrix} \dot{\varphi}_1 & \dot{\varphi}_2 \end{Bmatrix} \begin{bmatrix} 3 + 2\,cos\varphi_2 & 1 + cos\varphi_2 \\ 1 + cos\varphi_2 & 1 \end{bmatrix} \begin{Bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{Bmatrix}$$

and the potential energy,

$$V = \frac{1}{2} k\,(\varphi_1^2 + \varphi_2^2)$$

The application of the Lagrange's equations leads to the following equations of motion:

$$\begin{bmatrix} 3 + 2\,cos\varphi_2 & 1 + cos\varphi_2 \\ 1 + cos\varphi_2 & 1 \end{bmatrix} \begin{Bmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \end{Bmatrix} = - \begin{Bmatrix} k\,\varphi_1 \\ k\,\varphi_2 \end{Bmatrix} + sin\varphi_2 \begin{bmatrix} 2\,\dot{\varphi}_2 & \dot{\varphi}_2 \\ (\dot{\varphi}_2 - \dot{\varphi}_1) & -\dot{\varphi}_1 \end{bmatrix} \begin{Bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{Bmatrix}$$

Since independent joint coordinates are being used, the mass matrix is not constant, and there are velocity-dependent terms in the RHS of the equations of motion which make the integration more critical. Also, the terms coming from the springs constants k in the RHS of the equation add additional numerical stiffness to the above equations of motion. These are integrated for different values of the spring constants k: first, in the predictor-corrector fashion (Algorithm 7-1) by using the subroutine DE (Shampine and Gordon (1975)); and secondly, by substituting the difference equations corresponding to the trapezoidal rule (equations (7.28) and (7.29) with $\beta = 1/4$ and $\gamma = 1/2$) and solving the resulting set of nonlinear algebraic equations by means of the Newton-Raphson iteration. Although it is rather involved, the expression for the tangent matrix can be obtained for this particular case in closed form and its elements are:

$$t_{11} = \frac{4}{\Delta t^2}(3 + 2\,cos\varphi_2) + k - \frac{4}{\Delta t}\dot{\varphi}_2\,sin\varphi_2$$

$$t_{12} = \frac{4}{\Delta t^2}(1 + cos\varphi_2) - \frac{4}{\Delta t}(\dot{\varphi}_1 + \dot{\varphi}_2)sin\varphi_2 - (2\,\dot{\varphi}_1\,\dot{\varphi}_2 + \dot{\varphi}_2^2)cos\varphi_2 - (2\,\ddot{\varphi}_1 + \ddot{\varphi}_2)sin\varphi_2$$

$$t_{21} = \frac{4}{\Delta t^2}(1 + cos\varphi_2) - \frac{4}{\Delta t}(\dot{\varphi}_1 + \dot{\varphi}_2)sin\varphi_2$$

$$t_{22} = \frac{4}{\Delta t^2} + k + \dot{\varphi}_1^2\,cos\varphi_2 - \ddot{\varphi}_1\,sin\varphi_2$$

where the velocities and accelerations can be expressed in terms of the positions using the difference equations (7.28 and 7.29). Even in this simple case, the elements of the tangent matrix are quite involved. One can conclude that for a general case the formulation of the equations of motion in independent coordinates makes the task of obtaining the tangent matrix exceedingly complicated. On the other hand, the use of dependent coordinates and in particular the natural coordinates leads to a much simpler tangent matrix at the expense of increasing the number of equations and adding constraint conditions. This can be done as an exercise (See problem at the end of the chapter). These concepts are further developed in Chapter 8 (Section 8.5) in the context of real time analysis.

Another very important point is that the spring terms k appear in the tangent matrix and do not produce numerical instabilities. On the contrary, these terms help in the convergence process with increasing values of k; thus coping very nicely with the problem of numerical stiffness.

Table 7.4 contains a comparative study of both methods, by showing the resulting maximum energy errors along with the processing times for a time step of 0.001 seconds and different values of the spring constant k. The constant is changed with the idea of adding numerical stiffness to the problem. One can see how the integration with the trapezoidal rule, although not as accurate, compares favorably with the subroutine DE for small values of k (low numerical stiffness). When stiffness is present due to large values of k, the implementation of the trapezoidal rule with the positions as primary variables becomes much faster and accurate than DE which behaves very slowly in some cases and does not converge in others. With this implementation of the trapezoidal rule, the integration becomes faster as the system gets stiffer.

7.2 Integration of Differential-Algebraic Equations

7.2.1 Preliminaries

We present in this section a brief discussion on the direct integration of the differential algebraic equations (DAEs) with special emphasis in its application to the integration of the equations of motion of multibody systems. The intention is to give a basic notion of where the state of the art in this field currently is and of how these new methods are applied to the problems at hand. One is referred to specialized works in this area (Brenan et al. (1989), and Haug and Deyo (1990))

Table 7.4. Maximum errors and total CPU time in the integration of the double pendulum.

			Error		CPU time	
k_1	k_2	h	Trapez.	DE	Trapez.	DE
0	0	0.001	$4.6\ 10^{-4}$	$-3.4\ 10^{-5}$	14.6	11.2
100	100	0.001	$2.7\ 10^{-3}$	$7.3\ 10^{-3}$	17.2	14.2
10^4	10^4	0.001	$2.3\ 10^{-3}$	$3.8\ 10^{-3}$	14.7	18.0
10^6	10^6	0.001	$1.2\ 10^{-3}$	$-2.0\ 10^{-3}$	12.1	48.4
10^8	10^8	0.001	$7.7\ 10^{-5}$	$-4.9\ 10^{-3}$	9.2	342.6
0	100	0.001	$4.0\ 10^{-4}$	$-1.7\ 10^{-5}$	14.2	10.9
0	10^4	0.001	$6.0\ 10^{-5}$	$6.0\ 10^{-5}$	14.0	12.1
0	10^6	0.001	$1.0\ 10^{-3}$	$3.5\ 10^{-2}$	11.7	47.3
0	10^8	0.001	$3.7\ 10^{-3}$	No conv.	8.8	No conv.

for a more comprehensive study on these topics.

Nonlinear DAEs are classified into two major groups: *implicit* and *semi-explicit*. Implicit equations take the following form:

$$\mathbf{F}(t, \mathbf{y}, \mathbf{y}') = \mathbf{0} \tag{7.43}$$

with initial conditions $\mathbf{y}(t_o) = \mathbf{y}_o$, and where $\partial\mathbf{F}/\partial\mathbf{y}'$ may be singular. This type of equation arises in problems related to electrical circuits. Semi-explicit equations can be written as:

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{z}, \mathbf{y}) \tag{7.44}$$

$$\mathbf{0} = \mathbf{g}(t, \mathbf{z}, \mathbf{y}) \tag{7.45}$$

with initial conditions $\mathbf{y}(t_o) = \mathbf{y}_o$ and $\mathbf{z}(t_o) = \mathbf{z}_o$. This type of equation arises commonly in constrained multibody systems, optimal control, and trajectory prescribed path problems.

Whereas the theory of existence and uniqueness of ordinary differential equations (ODEs) is complete, that of the DAEs is still incomplete. It is also more difficult to establish than that of the ODEs. Solutions to DAEs may not always exist. If they do, they may not be unique. Other important issues that pertain to the integration of DAE are their theoretical as well as numerical solvability; that is, the identification of analytical as well as numerical solutions. Such topics are very important when considering the development of both special and general purpose DAE solvers. The amount of research dedicated to this important mathematical problem has been steadily increasing in recent years, as measured by the amount of related literature appearing in specialized journals and conferences. One may identify such published works in Brenan et al. (1989), and Haug and Deyo (1990).

Differential algebraic equations are classified according to their *differential index* or simply *index*, defined as the number of times that the DAE has to be differentiated to obtain a standard set of ODE. This ODE is also called the *underlying ODE*, and is satisfied by all the components of the solution. The differential index can also be defined as the number of differentiations necessary to solve for $\mathbf{y}'$ uniquely in terms of $\mathbf{y}$ and t. The higher the index the more complex the integration becomes. As an example, one can verify that the equations of constrained multibody systems of the form:

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} = \mathbf{Q}(t, \mathbf{q}, \dot{\mathbf{q}}) - \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}(\mathbf{q})\,\boldsymbol{\lambda} \tag{7.46}$$

$$\mathbf{\Phi}(t, \mathbf{q}) = \mathbf{0} \tag{7.47}$$

is semi-explicit of index three, with $\mathbf{q}$ as the positions, $\dot{\mathbf{q}}$ the velocities, $\ddot{\mathbf{q}}$ the accelerations, and $\mathbf{Q}$ as all the forcing terms. Equations (7.46) and (7.47) can be transformed to first order form by the following transformation $\dot{\mathbf{q}} = \mathbf{s}$, yielding:

$$\mathbf{M}(\mathbf{q})\,\dot{\mathbf{s}} = \mathbf{Q}(t, \mathbf{q}, \mathbf{s}) - \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}(\mathbf{q})\,\boldsymbol{\lambda} \tag{7.48}$$

$$\dot{\mathbf{q}} = \mathbf{s} \tag{7.49}$$

$$\mathbf{\Phi}(t, \mathbf{q}) = \mathbf{0} \tag{7.50}$$

Two families of methods used for the integration of ODEs have also been utilized for DAEs: the backward difference formulae (BDF), and the implicit Runge-Kutta methods (IRK). This does not necessarily mean that all DAE problems are solvable by ODE methods. In fact this does not hold true in all the cases. However, success has been reported when using these methods for certain classes of DAE. Special care must be exercised at the time of using these algorithms since their implementation is not as simple as in the case of ODEs.

The way of solving a differential algebraic equation in essence is to approximate $\mathbf{y}'$ in (7.43) or (7.44), and (7.45) by a finite difference formula or implicit Runge-Kutta method, and solve the resulting set of nonlinear algebraic equations by some iterative procedure for an approximation to $\mathbf{y}$. For example, the simple backward Euler method ($\mathbf{y}_{n+1}=\mathbf{y}_n+\Delta t\, \dot{\mathbf{y}}_{n+1}$) can be substituted into the implicit equation (7.28) to yield

$$\mathbf{F}\left[t_{n+1}, \mathbf{y}_{n+1}, \frac{1}{\Delta t}(\mathbf{y}_{n+1} - \mathbf{y}_n)\right] = 0 \tag{7.51}$$

The procedure can be applied in the same manner for the semi-explicit equations (7.44) and (7.45). Equation (7.51) constitutes a set of nonlinear equations with $\mathbf{y}_{n+1}$ as unknowns that can be solved by standard methods such as Newton-Raphson iteration. This way of proceeding is similar to that outlined above for the solution of the equations in structural dynamics and used in the numerical example of the double pendulum, in which the finite difference equations of the integrator (the Newmark family, collocation, or α-methods, etc.) that approximate the derivatives (accelerations and velocities) are directly introduced in the equations of motion. More recently, the implicit Runge-Kutta methods have also been used in a similar fashion, resulting in a set of nonlinear equations with $\mathbf{y}_{n+1}$ as unknowns.

7.2.2 Solutions by Backward Difference Formulae

The BDF methods were first successfully applied to the solution of DAEs. The theory of stability and convergence has been almost completed, and there is even public domain software currently available (Petzold (1982)). The method consists of directly applying the backward difference formulae to the system equations. This approach is one of the options used in commercial programs for the integration of the equations of motion of constrained mechanical systems (Chace (1984)). The procedure consists of substituting the backward difference approximations:

$$\dot{\mathbf{q}}_{n+1} = \frac{1}{\Delta t\ \beta_o}(\mathbf{q}_{n+1} - \sum_{i=0}^{p} \alpha_i\, \mathbf{q}_{n-i}) \tag{7.52}$$

$$\dot{\mathbf{s}}_{n+1} = \frac{1}{\Delta t\ \beta_o}(\mathbf{s}_{n+1} - \sum_{i=0}^{p} \alpha_i\, \mathbf{s}_{n-i}) \tag{7.53}$$

into the equations of motion (7.48)-(7.50) to form the following set of nonlinear algebraic equations with $\mathbf{q}_{n+1}$ and $\mathbf{s}_{n+1}$ as unknowns:

$$\mathbf{M}(\mathbf{q}_{n+1}) \frac{1}{\Delta t\ \beta_o} (\mathbf{s}_{n+1} - \sum_{i=0}^{p} \alpha_i\ \mathbf{s}_{n-i}) = \mathbf{Q}_{n+1} - \mathbf{\Phi}_{\mathbf{q}}^{T}(\mathbf{q}_{n+1})\ \boldsymbol{\lambda}_{n+1} \tag{7.54}$$

$$\frac{1}{\Delta t\ \beta_o} (\mathbf{q}_{n+1} - \sum_{i=0}^{p} \alpha_i\ \mathbf{q}_{n-i}) = \mathbf{s}_{n+1} \tag{7.55}$$

$$\mathbf{\Phi}(t, \mathbf{q}_{n+1}) = \mathbf{0} \tag{7.56}$$

The solution of this system of equations can be carried out by Newton-Raphson iteration provided a tangent matrix is available. Otherwise, a less accurate secant method or simplest fixed point iteration would have to be used. An approach that consists of substituting the backward difference equations (7.52) and (7.53) into the equations of motion written in the form of the generalized coordinate partitioning method has been proposed by Haug and Yen (1990).

It has been demonstrated by Gear and Petzold (1984) that for all index one DAEs, the *k*-step BDF with fixed step size are stable and convergent to order $O(\Delta t^k)$, if all initial values are sufficiently accurate. An extension of this result to variable step size is shown by Gear et al. (1985). General purpose DAE solvers designed to solve this kind of problems are not free of some implementation problems that are described below. For higher index systems, instability may be present. Encouraging results have been obtained for the particular case of semi-explicit systems which are those in which the multibody systems are classified. It has been demonstrated (Lotstedt and Petzold (1986)) that the *k*-step BDF with constant step-size are also convergent to order $O(\Delta t^k)$ for semi-explicit index two DAEs and even index three of the type of equations (7.46) and (7.47), when the initial conditions are defined within sufficient accuracy. This result has been generalized (Gear et al. (1985)) for variable step-size and index two problems.

While the BDF seem to achieve convergence and yield satisfactory results for a wide variety of DAEs, the actual implementation of the algorithms in general purpose solvers is not free from serious numerical difficulties. This becomes more acute for index three problems such as constrained multibody systems with equations (7.46) and (7.47). Such difficulties stem from the following points:

- It can be shown that for an index *m* DAE, the tangent or quasi-tangent matrix used in the Newton-Raphson iteration has a condition number of order $O(1/\Delta t^m)$. Consequently, the practical implementation of the method is bound to have large round-off errors for small sizes of the time step.
- Instabilities may result for sudden changes in the system variables and constraints, such as impacts or sudden appearances or disappearances of constraints. Any time there is a situation of discontinuity in the response, the multistep BDF tries to fit a polynomial through such discontinuity, and the step size must be severely reduced. This results in an ill-condition iteration

matrix, and the Newton-Raphson iteration may end up near a solution and yet be unable to converge. These problems can be circumvented as explained by Steigerwald (1990) at the expense of reinitializing the integration with consistent initial conditions which have to be obtained from the derivatives of the constraint equations. This reinitialization produces serious delays in the integration process.

- The multistep methods are not self-starting. A *k*-step method requires sufficiently accurate (*k–1*) starting values which have to be obtained by other methods which may render the method sensitive to the starting values. This problem also arises when either the time step or order of the BDF method is changed during the integration process.

7.2.3 Solutions by Implicit Runge-Kutta Methods

The IRK methods have been used as an alternative to the BDF methods for the integration of DAEs, offering some important advantages over them for the integration of multibody systems. Being single step, the IRK methods do not suffer from systems discontinuities and changes in the order or time step as the BDF do. The IRK are also self-starting which means that the only starting values required are the initial conditions. These methods have the disadvantage of leading after the time discretization to larger and more complex sets of nonlinear algebraic equations. In addition, the convergence and stability analysis is still in early stages and no public domain software is yet available.

The method consists of substituting the expression of the IRK (7.17) into the system equations. In the case of the implicit type of equations (7.43), this substitution leads to

$$\mathbf{F}(t_{\mathrm{n}} + c_i\,\Delta t,\ \mathbf{y}_{\mathrm{n}} + \Delta t \sum_{j=1}^{r} a_{ij}\,\mathbf{k}_j,\ \mathbf{k}_i) = 0\ , \qquad i = 1,2,...,r \tag{7.57}$$

Once the system (7.57) is solved for the values of the stage derivatives $\mathbf{k}_{\mathrm{i}}$, the solution at step *n+1* is given by equation (7.18). The application of the IRK to the solution of the equations of motion of constrained multibody systems of the form (7.48)-(7.50) becomes:

$$\mathbf{M}(\mathbf{q}_{\mathrm{n}} + \Delta t \sum_{j=1}^{\mathrm{r}} a_{ij}\,\mathbf{k}_j)\,\mathbf{L}_i = \mathbf{Q}_{\mathrm{n+1}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}(\mathbf{q}_{\mathrm{n}} + \Delta t \sum_{j=1}^{r} a_{ij}\,\mathbf{k}_j)\,\boldsymbol{\lambda}_{\mathrm{n+1}} \qquad i = 1,\ 2,\ ...,r \tag{7.58}$$

$$\mathbf{k}_i = \mathbf{s}_{\mathrm{n}} + \Delta t \sum_{\mathrm{j=1}}^{\mathrm{r}} a_{ij}\,\mathbf{L}_j \qquad \mathrm{i} = 1,\ 2,\ ...,\ r \tag{7.59}$$

$$\mathbf{\Phi}\ (t_{\mathrm{n}} + c_i\,\Delta t,\ \mathbf{q}_{\mathrm{n}} + \Delta t \sum_{\mathrm{j=1}}^{\mathrm{r}} a_{ij}\,\mathbf{k}_j) = 0 \tag{7.60}$$

where $\mathbf{k}_i$ and $\mathbf{L}_i$ are the stage derivatives for the approximations to $\mathbf{q}$ and $\mathbf{s}$, respectively. Equations (7.58) to (7.60) have to be solved for the unknowns $\mathbf{k}_i$, $\mathbf{L}_i$, and Lagrange multipliers $\boldsymbol{\lambda}$. The resulting set of equations can be seen becoming larger and more involved than that resulting from the application of the BDF. Obtaining the tangent matrix of equation (7.58) for a Newton-Raphson iteration becomes an exceedingly complicated task. The use of the IRK method still defies implementation in a general purpose DAE solver.

Stability conditions for index one and semi-explicit index two DAE are given by Burrage (1982), and Brenan and Petzold (1989).

7.3 Considerations for Real-Time Simulation

Real time simulation of multibody systems requires an analysis time (integration time plus time for graphical display) smaller than the physical time taken by the actual motion of the multibody system. This has an important influence on the type of method used for the integration of the equations of motion. Commonly used integration routines in multibody dynamics are based on a variable order, variable step size multistep methods with error control. The user specifies the maximum allowable error, and the routine adapts the order and the step to fulfill the error conditions. With this kind of integrators, it is not possible to predict the computer time necessary to integrate the equations on a determined period of time. Multistep methods adjust very poorly to force and/or system discontinuities such as those arising from sudden forces and appearances or disappearances of constraints, leading to very small time steps and ill-conditioning of the Jacobian matrices. Special formulas are required to restart the integration after system or force discontinuities. These restarting procedures do not fit well into the real time condition, since the requirement for a fixed time of integration per time step will not be met.

Explicit multistep methods can be inexpensive and accurate for real time analysis provided the time step is chosen sufficiently small. However, they do not present good stability conditions This poses a serious limiting factor for real time integration. Sometimes, the equations of motion may become stiff, whereby the solution has components whose time constants can differ in several orders of magnitude. It is convenient that the chosen integrator performs well under such conditions. This makes the implicit stiffly-stable or *A*-stable methods more suitable.

It seems that for real time integration, it would be more convenient to use an implicit single step integration formula with fixed time step and order that will have the same computational cost in each integration step (fixed number of iterations) and will be capable of adjusting to system discontinuities. Since the computational time needs to be smaller than the step size, the selected integration method must be computationally inexpensive, with few function evaluations and iterations in each step, and must allow for large step sizes without introducing excessive loss of accuracy and, most importantly, must not become unstable.

The stated preference towards implicit *A*-stable methods does not necessarily mean that the explicit methods cannot be used for real time integration. This would require, on the analyst's part, a previous knowledge of the range of the values $\lambda(t)$ of the given problem, so that the necessary Δt may be chosen. As soon as the problem has a certain degree of stiffness as most practical cases do, the necessary time step will be so small that the use of an implicit method will be more than justified.

The suitable methods for real time simulation of multibody systems should be implicit, simple step, and inexpensive in terms of computational power required, with good stability properties (*A*-stability, if possible), and sufficiently accurate. Accuracy contradicts other characteristics such as stability, economy and single step integration. For real time applications, accuracy is the feature that must be sacrificed in conflicts with other properties. It is better to obtain a solution with some small error than not be able to obtain it at all in the allowed time. Moreover, many real time applications incorporate a feedback control either embedded in the dynamic formulation or carried out visually as in the case of a teleoperator training system. Feedback control helps to compensate errors and disturbances, including integration errors.

According to these considerations and in view of the theory and methods explained above, it seems that the methods used in structural dynamics (Newmark family, α-method, collocation methods), the midpoint rule, and the implicit Runge-Kutta methods are suitable choices for the integration of the equations of motion in their ODE form. In particular, the α-method which adds artificial damping to the trapezoidal rule without losing second order accuracy, the generalized trapezoidal rule with energy or velocity constraints, and midpoint rule seem to be good choices and somehow preferable over the more expensive implicit Runge-Kutta methods. However, in order to make these methods computationally inexpensive, it is necessary to limit the number of iterations necessary to solve the resulting system of algebraic nonlinear equations. We will show in Chapter 8, how the trapezoidal rule with velocity constraints performs very satisfactorily in real time analysis when it is directly introduced in the equations of motion of multibody systems. We will also show how this integration scheme fits quite well in the framework of the natural Cartesian coordinates and the penalty formulation, because it allows for a very simple expression of the tangent matrix necessary for the Newton-Raphson iteration.

Regarding the integration of the equations of motion in their DAE form, it seems that currently available DAE solvers are not capable of competing against their ODE counterparts in speed of integration.

References

Bathe, K.-J., *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, (1982).

Bayo, E., García de Jalón, J., Avello, A., and Cuadrado, J., "An Efficient Computational Method for Real Time Multibody Dynamic Simulation in Fully Cartesian Coordinates", *Computer Methods in Applied Mechanics and Engineering*, Vol. 92, pp. 377-395, (1991).

Brenan, K.E, Campbell, S.L., and Petzold, L.R., *The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*, Elsevier, (1989).

Brenan, K.E. and Petzold, L.R., "The Numerical Solution of Higher Index Differential-Algebraic Equations by Implicit Runge-Kutta Methods", *SIAM Journal on Numerical Analysis,* Vol. 26, pp. 976-996, (1989).

Burden, R.L., Faires, J.D., and Reynolds, A.C., *Numerical Analysis,* 4th edition, Prindle, Weber and Schmidt, Boston, Massachusetts, (1989).

Burrage, K., "Efficiently Implementable Algebraically Stable Runge-Kutta Methods", *SIAM Journal on Numerical Analysis,* Vol. 19, pp. 245-258, (1982).

Butcher, J.C., "Implicit Runge-Kutta Processes", *Mathematical Computation*, Vol. 18, pp. 50-64, (1964).

Calahan, D.A., "A-stable Accurate Method of Numerical Integration for Nonlinear Systems", *Proceedings of IEEE*, Vol. 56, pp. 744-751, (1968).

Cardona, A. and Geradin, M., "Time Integration of the Equations of Motion in Mechanism Analysis", *Computer & Structures*, Vol. 33, pp. 801-820, (1989).

Carnahan, B., Luther, H.A., and Wilkes, J. O., *Applied Numerical Methods*, Wiley, (1969).

Chace, M.A, "Methods and Experience in Computer-Aided Design of Large-Displacement Mechanical Systems", *Computer-Aided Analysis and Optimization of Mechanical System Dynamics,* NATO ASI Series, Vol. F9, pp. 233-259, Springer-Verlag, (1984)

Conte, S.D and Boor, C., *Elementary Numerical Analysis*, 2nd edition, McGraw-Hill, (1972).

Dahlquist, G., "A Special Stability Problem for Linear Multistep Methods", *BIT*, Vol. 3, pp. 27-43, (1963).

Fehlberg, E., "Klassische Runge-Kutta Formeln Vierter und Niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Warmeleitungs-Probleme", *Computing*, Vol. 6, pp. 61-71, (1970).

Gear, C.W., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, (1971).

Gear, C.W. and Petzold, L.R., "ODE Methods for the Solution of Differential/Algebraic Systems", *SIAM Journal on Numerical Analysis,* Vol. 21, pp. 367-384, (1984).

Gear, C.W., Leimkuhler, B., and Gupta, G.K., "Automatic Integration of Euler-Lagrange Equations with Constraints", *Journal of Computational and Applied Mathematics*, Vol. 12, pp. 77-90, (1985).

Goudreau, G.L. and Taylor, R.L., "Evaluation of Numerical Methods in Elastodynamics", *Computer Methods in Applied Mechanics and Engineering*, Vol. 2, pp. 69-97, (1973).

Gourley, A.R., "A Note on Trapezoidal Methods for the Solution of Initial Value Problems", *Mathematics of Computation*, Vol. 24, pp. 629-633, (1970).

Haug, E.J. and Deyo, R.C. (editors), *Real-Time Integration Methods for Mechanical System Simulation*, NATO ASI Series, Vol. F 69, Springer-Verlag, (1990).

Haug, E.J and Yen, J., "Generalized Coordinate Partitioning Methods for Numerical Integration of Differential Algebraic Equations of Dynamics", in *Real-Time Integration Methods for Mechanical System Simulation*, NATO ASI Series, Vol. F 69, pp. 97-114, Springer-Verlag, (1990).

Hilber, H.M., "Analysis and Design of Numerical Integration Methods in Structural Dynamics", *EERC Report No. 76-29*, Earthquake Engineering Research Center, University of California, Berkeley, (1976).

Hilber, H.M., Hughes, T.J.R., and Taylor, R.L., "Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics", *Earthquake Engineering and Structural Dynamics*, Vol. 5, pp. 283-292, (1977).

Hughes, T.J.R., "Stability, Convergence, and Growth and Decay of Energy of the Average Acceleration Method in Nonlinear Structural Dynamics", *Computer and Structures*, Vol. 6, pp. 313-324, (1976).

Hughes, T.J.R., *Computational Methods for Transient Analysis* (eds. T. Belytschko and T.J.R. Hughes), Chapter 2, North-Holland, (1983).

Hughes, T.J.R., *The Finite Element Method: Linear Static and Dynamic Analysis*, Prentice-Hall, (1987).

Ince, E.L., *Ordinary Differential Equations*, Dover, New York, (1956).

Lodsted, P. and Petzold, L., "Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints: Convergence Results for Backward Differentiation Formulae", *Mathematics of Computation*, Vol. 46, pp. 491-516, (1986).

Newmark, N.M., "A Method of Computation for Structural Dynamics", *Journal of the Engineering Mechanics Division, ASCE*, pp. 67-94, (1959).

Norsett, S.P., "One Step Methods of Hermite Type for the Numerical Solution of Stiff Systems", *BIT*, Vol. 14, pp. 63-77, (1974).

Petzold, L.R., "A Description of DASSL: A Differential/Algebraic System Solver", *IMACS Transactions on Scientific Computation*. Vol. 1, R.S. Stepleman (ed.), (1982).

Shampine, L. and Gordon, M., *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, Freeman, San Francisco, (1975).

Simo, J.C. and Wong, S., "Unconditionally Stable Algorithms for Rigid Body Dynamics that Exactly Preserve Energy and Momentum", *International Journal for Numerical Methods in Engineering*, Vol. 31, pp. 19-52, (1991).

Smith, I.M., "Some Time Dependent Soil-Structure Interaction Problems", *In Finite Elements in Geomechanics* (ed. Gudehus), Chapter 8, Wiley, (1975).

Steigerwald, M.F., "BDF Methods for DAEs in Multibody Dynamics: Shortcomings and Improvements", in *Real-Time Integration Methods for Mechanical System Simulation*, NATO ASI Series, Vol. F 69, pp. 345-352, Springer-Verlag, (1990).

Wilson, E.L., "A Computer Program for the Dynamic Stress Analysis of Underground Structures," *SESM Report No. 68-1*, Division of Structural Engineering Structural Mechanics, University of California, Berkeley, (1968).

Wood, W.L., *Practical Time-Stepping Schemes*, Oxford University Press, Chapter 7, (1990).

Problems

7/1 Integrate the second order differential equation: $\ddot{y}+y=0$ with initial conditions $y(t=0)=1$ and $\dot{y}(t=0)=0$, during the time interval [0,10] with $\Delta t=0.1$, using the following methods:

a) The trapezoidal rule (Newmark with $\gamma=1/2$ and $\beta=1/4$).

b) The a-method with $a = -0.1$.

c) The backward difference method with p=1 of Table 7.3 equivalent to Newmark with $\gamma=1$ and $\beta=1/2$.

In all cases, introduce the difference equations of the corresponding algorithm into the equation of motion and solve the resulting set of linear equations for the unknown $\mathbf{y}_{n+1}$. Plot the results of each of the methods and the exact solution and compare the period elongation and amplitude decay. Draw some conclusions about the accuracy and artificial damping introduced by each of the methods.

7/2 Write a subroutine for the integration of the first order differential equations using the four order Adams-Bashforth-Moulton predictor-corrector algorithm (Eqs. 7.22 and 7.23). Solve Problem 7/1 by transforming the second order equation into a set of two first order ones and then using the new subroutine. Compare your results with those obtained in Problem 7/1.

7/3 Formulate the equations of motion of the double pendulum used in the numerical example of Section 7/1 including rotational dash pots and using the two independent joint coordinates. Then use the trapezoidal rule and the Adams-Bashforth-Moulton predictor-corrector algorithm developed in Problem 7/2 (or any other multistep method available in the software library of your computer). Write up a table with the energy errors and CPU times obtained by both methods for different values of the springs and dash pots.

7/4 Repeat Problem 7/3 using the Cartesian coordinates of the ends of the two links with additional constraints to introduce the joint angles. Then use the penalty formulation (Chapter 5) to form the equations of motion (take a penalty value equal to 10^7). Compare the results with those obtained in Problem 7/3.

7/5 Repeat Problem 7/4 with the same coordinates and the Lagrange multiplier formulation to form the equations of motion of the type of equation (7.46). Then use the backward difference formula with $p=1$ (Table 7.3) to solve the resulting set of differential and algebraic equations. Compare your results with those obtained in Problems 7/4 and 7/3.