Chapter 13
Integration Methods in Dynamic Analysis

Abstract This chapter describes the main integration algorithms utilized in the resolution of the dynamics equations of motion. Particular emphasis is paid to the Euler method, Runge-Kutta approach and Adams predictor-corrector method that allows for the use of variable time steps during the integration process. The material presented here, relative to numerical integration of ordinary differential equations, follows that of any undergraduate text on numerical analysis.

Keywords Euler method · Runge-Kutta method · Adams predictor-corrector method

In the previous paragraph, the equations of motion for multibody systems were derived from the Newton-Euler formulation together with the augmentation method. The Newton-Euler equations represent the translational and rotational motions of bodies, while the augmentation method is used to adjoin the constraint equations of the multibody systems. In other words, the augmentation formulation denotes the process where the algebraic kinematic constraint equations are augmented to the differential equations of motion, in order that the number of unknowns for which the system is being solved corresponds to the number of system equations (Nikravesh 1988). As a consequence, the equations of motion of multibody systems (12.1) are differential and algebraic equations (DAE) rather than ordinary differential equations (ODE) (Blajer 1999). Prior to integrate the system state variables, Eq. (12.1) is solved for $\dot{\mathbf{v}}$ and $\boldsymbol{\lambda}$.

In the present work, the DAE are converted to ODE because the most frequently used numerical integration algorithms are useful in solving ODE (Shampine and Gordon 1975). However, for a detailed discussion on DAE, the interested reader may consult the works by Petzold (1983) and Brenan et al. (1989). The material presented below, relative to numerical integration of ODE, follows that of any undergraduate text on numerical analysis such as those by Conte and Boor (1981) and Atkinson (1989).

The process of converting n second-order differential equations to $2n$ first-order equations can be expressed by (Shampine and Gordon 1975; Conte and Boor 1981; Atkinson 1989)

P. Flores, *Concepts and Formulations for Spatial Multibody Dynamics*,
SpringerBriefs in Applied Sciences and Technology,
DOI 10.1007/978-3-319-16190-7_13

$$\ddot{y}_1 = f(y_1, \dot{y}_1, t) \tag{13.1}$$

such that it can be written as the following system

$$\dot{y}_1 = y_2 \tag{13.2}$$

$$\dot{y}_2 = f(y_1,\ y_2,\ t) \tag{13.3}$$

The most popular and used numerical integration methods introduced in the vast thematic literature are Euler method, Rung-Kutta methods and Adams predictor-corrector methods. These methods have been known for many years, for instance, the Runge-Kutta methods have been known for more than an 100 years, but their potential was not fully realized until computers became available. These methods involve a step-by-step process in which a sequence of discrete points t^0, t^1, t^2, …, t^n is generated. The discrete points may have either constant or variable spacing defined as $h^i = t^{i+1} - t^i$, where h^i is the step size for any discrete point t^i. At each point t^i, the solution $y(t^i)$ is approximated by a number y^i. Since no numerical method is capable of finding $y(t^i)$ exactly, the quantity

$$\varepsilon_g^i = \left| y(t^i) - y^i \right| \tag{13.4}$$

represents the global or total error at $t = t^i$. The total error consists of two components, the truncation error and the round-off error. The truncation error depends on the nature of the numerical algorithm used in computing y^i. The round-off error is due to the finite word length in a computer.

The integration methods are called single step methods when they only require information on the current time step to advance to the next time step. Euler and Runge-Kutta methods are single step methods. When information of the previous steps is used, the algorithm methods are called multistep methods, as it is the case of Adams predictor-corrector schemes. The single step methods are self starting and they need a minimum amount of storage requirements. However, these methods require a larger number of function evaluations, for instance, four for the fourth-order Runge-Kutta method. Function evaluation is the name of the process by which, given t and y, the value of $\dot{y}$ is computed. The multistep methods require a small 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. Moreover, error estimates are easily provided and step size adjustments can be performed with no difficulties. The multistep methods are not self starting and require the help of a single step scheme to start the integration process (Atkinson 1989).

Regardless of the numerical method used, the numerical task deals with the integration of an initial-value problem that can be written as

$$\dot{y}_1 = f(y, t) \tag{13.5}$$

With the initial condition $y(t^0) = y^0$ and where y is the variable to be integrated and function $f(t, y)$ is defined by the computational sequence of the algorithm selected. Equation (13.5) has a solution $y(t)$. The initial value y^0 can be defined for any value of t^0, although it is often assumed that a transformation has been made so that $t^0 = 0$. This does not affect the solution or method used to approximate the solution.

It is known that the Euler integration method is one of the simplest integrators available. This approach may be sufficient in giving a very rough idea of the motion of multibody systems. This method solves differential equations in a single step as

$$y^{i+1} = y^i + hf(y^i, t) \tag{13.6}$$

where h is the integration step size $h = t^{i+1} - t^i$, for i a non-negative integer. This method implies that the next step of the state variable can be evaluated by using the current state variable.

The intuitive basis of the Euler method is illustrated in Fig. 13.1, in which the curve labeled $y = y(t)$ is the solution of the differential Eq. (13.5), which passes through point $P(t^0, y^0)$. It is desired to find the value of $y^1 = y^0 + \Delta y$ corresponding to $t = t^1$. In other words, the height RQ needs to be determined. Although the position of the curve at every point is not known, its slope is equal to $f(t, y)$, which is simply the geometric interpretation of the differential equation. Thus, the slope of the tangent at point P is $\dot{y}^0 = f(t^0, y^0)$, which can be computed since y^0 and t^0 are both known. If h is reasonable small, the tangent line PS should not deviate too much from the curve PQ, hence, the height RS (which by simple geometry is equal to $h\dot{y}^0$) should be an approximation to the required height RQ. Thus, a first

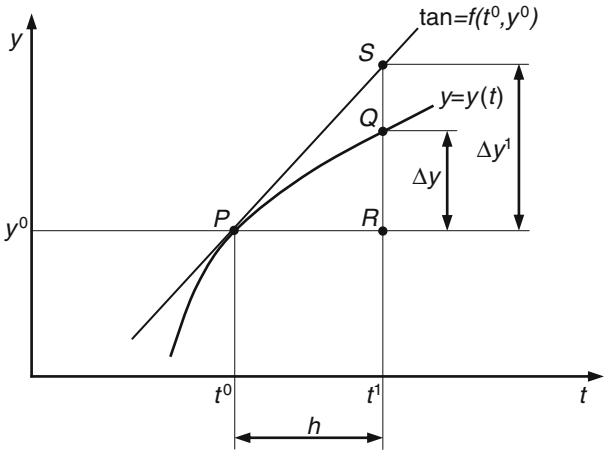

Fig. 13.1 Geometric interpretation of the Euler integration method

approximation to Δy is given by $\Delta y^1 = RS = hf\,(t^0, y^0)$. Assuming that the appropriate derivatives exist, then $y(t)$ can be expanded in a Taylor series about $t = t^i$ and the expression is evaluated at $t = t^{i+1}$, yielding

$$y(t^{i+1}) = y(t^i) + hf(t^i, y^i) + O(h^2) \tag{13.7}$$

From the analysis of Eq. (13.7), neglecting the higher-order terms, the discretization or local truncation error is given by

$$\varepsilon_l = O(h^2) \tag{13.8}$$

The order of a numerical integration method can be used to specify its accuracy and can be expressed using the local truncation error. Knowing that for a scalar equation of type

$$\varepsilon_l = O(h^{p+1}) \tag{13.9}$$

is said to be of pth order, then it is clear that the Euler integration method is of first order. Thus, for highly oscillatory motion there are rapid changes in the derivatives of the function and if h is too large, then inaccuracies in the computation of the state variables are made (Nikravesh 1988).

In turn, the global truncation error at t^i can be evaluated as the difference between the actual and computed solution, in the absence of round-off error by the end of the simulation, that is

$$\varepsilon_g^i = \left| y(t^i) - y^i \right| \tag{13.10}$$

A more accurate integration method is the second-order Runge-Kutta algorithm, which can be expressed as

$$y^{i+1} = y^i + \frac{h}{2}(f_1 + f_2) \tag{13.11}$$

where

$$f_1 = f(t^i, y^i) \tag{13.12}$$

$$f_2 = f(t^i + h, y^i + hf_1) \tag{13.13}$$

This approach is also known as the improved Euler method, modified trapezoidal method or the Heun method. It should be noted 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. Figure 13.2 shows the geometric interpretation of the second-order

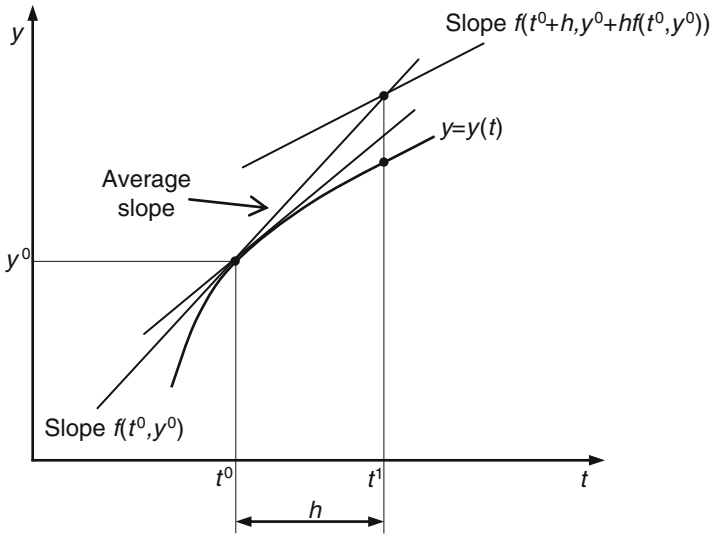

Fig. 13.2 Geometric interpretation of the second-order Runge-Kutta method

Runge-Kutta method. This method is explicit in the measure that f_1 does not depend on f_2 and neither one depends on y^{i+1} (Jalón and Bayo 1994).

The local error of the second-order Runge-Kutta method is of order h^3, whereas that of Euler method is h^2. Thus, it is expected to be able to use a larger time step with the second-order Runge-Kutta method. The price to pay for this is that it requires to evaluate the function $f(t, y)$ twice for each time step of the integration process.

For larger time steps and for greater accuracy, the fourth-order Runge-Kutta integration method is most popular and widely used. This method is stable and, as a computer program, occupy relatively small amount of core storage. The fourth-order Runge-Kutta integration algorithm can be expressed by Pina (1995)

$$y^{i+1} = y^i + hg \tag{13.14}$$

where

$$g = \frac{1}{6}(f_1 + 2f_2 + 2f_3 + f_4) \tag{13.15}$$

$$f_1 = f(t^i, y^i) \tag{13.16}$$

$$f_2 = f(t^i + \frac{h}{2},\ y^i + \frac{h}{2}f_1) \tag{13.17}$$

$$f_3 = f\left(t^i + \frac{h}{2},\ y^i + \frac{h}{2} f_2\right) \tag{13.18}$$

$$f_4 = f\left(t^i + h,\ y^i + h f_3\right) \tag{13.19}$$

This method is explicit because all f_i depend only on previous values already calculated. This algorithm is easy to implement in the measure that it only requires function evaluations, and it is self starting integrator scheme, which means that there is no need for any other algorithm or technique to start the integration process.

Figure 13.3 illustrates the geometric interpretation of the fourth-order Runge-Kutta integration method. In this method four tangents are determined, being their average weighted according to Eqs. 13.14–13.19.

The standard fourth-order Runge-Kutta method does not provide an estimate of the local error, so that the user does not have way of knowing whether the time step being used is adequate. The local error of this method is of order h^5, which is relatively small even for larger time steps. The major disadvantage of this method is that the function $f(t, y)$ needs to be evaluated four time at each time step.

For the Euler and Runge-Kutta methods the next step value y^{i+1} is computed by using solely the current value y^i and time t^i, over a time range of $h = t^{i+1} - t^i$. Multistep methods utilize information about the solution at more than one point. The objective of the multistep methods is to automatically select the proper order and the proper time step size, which will minimize the amount of work required to achieve the specified accuracy for a given problem. The multistep algorithms require only two function evaluation per step compared with four function evaluations per step with the fourth-order Runge-Kutta method, being, therefore,

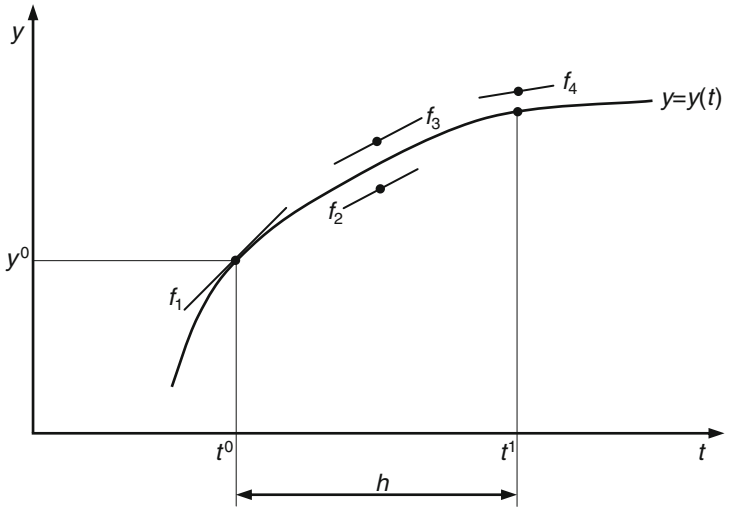

Fig. 13.3 Geometric interpretation of the fourth-order Runge-Kutta method

considerably faster and require less computation work. Predictor-corrector methods provide an automatic error estimate at each time step, thus allowing the algorithm to select an optimum value of h for a required accuracy. This type of approach is also better with respect to the propagation of error that it can use time steps more than twice as large.

In Adams predictor-corrector methods an explicit method is used to predict a value of y^{i+1}, while an implicit method corrects that value. The implicit corrects appear to be more stable and accurate than the explicit predictors and are both chosen to be of equal order. The Adams-Bashforth predictor algorithm of fourth-order can be written as

$$y^{i+1} = y^i + \frac{h}{24}(55f^i - 59f^{i-1} + 37f^{i-2} - 9f^{i-3}) \tag{13.20}$$

where

$$f^i = f(t^i, y^i) \tag{13.21}$$

$$f^{i-j} = f(t^{i-j}, y^{i-j}), \qquad (j = 1, 2, 3) \tag{13.22}$$

The corresponding Adams-Moulton corrector algorithm can be expressed by

$$y^{i+1} = y^i + \frac{h}{24}(9f^{i+1} + 19f^i - 5f^{i-1} + f^{i-2}) \tag{13.23}$$

where

$$f^i = f(t^i, y^i) \tag{13.24}$$

$$f^{i-j} = f(t^{i-j}, y^{i-j}), \qquad (j = 1, 2) \tag{13.25}$$

The major disadvantage of multistep methods is that they are not self starting. Thus, in the fourth-order Adams predictor-corrector method four successive values of function evaluation at equally spaced points before instant of time t^i must be known. These starting values must be obtained by some independent method, such as the Runge-Kutta method. On the other hand, Adams predictor-corrector algorithms are more complicated to program in the measure that they require special techniques for starting and for doubling and halving the time step, and they be subject to numerical instability (Conte and Boor 1981). In short, the Adams methods, when being carefully use, are more efficient than any other method. To achieve this efficiency it is necessary to vary the time step and the order that are used. Thus, it is necessary to estimate the errors that are incurred for various time steps and orders so as to make these decisions. Advanced codes also attempt to detect abnormal situations such as discontinuities or certain types of instabilities and to deal with them in a reasonable way. A detailed discussion on the Adams

predictor-corrector implementation can be found in the book by Shampine and Gordon (1975).

Gear (1971) developed a family of variable order stiffly-stable algorithms for the solution of stiff problems. A stiff system is referred to as any initial-value problem in which the complete solution consists of fast and slow components. The stiffness can be produced by physical characteristics of the multibody systems, such as components with large differences in their masses, stiffness and 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.

The Gear algorithm of fourth-order can be expressed as

$$y^{i+1} = \frac{1}{25}(48y^{i} - 36y^{i-1} + 16y^{i-2} - 3y^{i-3} + 12hf^{i+1}) \tag{13.26}$$

where

$$f^{i+1} = f(t^{i+1}, y^{i+1}) \tag{13.27}$$

Since the Gear algorithm is an implicit multistep scheme, it is necessary to solve an implicit equation in each time step (Nikravesh 1988).

References

Atkinson KA (1989) An introduction to numerical analysis, 2nd edn. Wiley, New York

Blajer W (1999) Elimination of constraint violation and accuracy improvement in numerical simulation of multibody systems. In: Ambrósio J, Schiehlen W (eds) Proceedings of EUROMECH colloquium 404, advances in computational multibody dynamics IDMEC/IST. Lisbon, 20–23 Sept, pp 769–787

Brenan KE, Campbell SL, Petzold LR (1989) Numerical solution of initial-value problems in differential-algebraic equations. Elsevier, New York

Conte SD, Boor C (1981) Elementary numerical analysis: an algorithmic approach, 3rd edn. McGraw-Hill, Singapore

Gear CW (1971) Numerical initial value problems in ordinary differential equations. Prentice-Hall, Englewood Cliffs

Jalón JG, Bayo E (1994) Kinematic and dynamic simulations of multibody systems: the real-time challenge. Springer, New York

Nikravesh PE (1988) Computer-aided analysis of mechanical systems. Prentice Hall, Englewood Cliffs

Petzold LR (1983) A description of DASSL: a differential/algebraic system solver. In: Stepleman R et al (ed) Scientific computing North-Holland Pub. Co. pp 65–68

Pina H (1995) Métodos numéricos. McGraw-Hill, Lisboa

Shampine L, Gordon M (1975) Computer solution of ordinary differential equations: the initial value problem. Freeman, San Francisco