Initial Value Problem

$$\text{Solve } \mathbf{y}' = \mathbf{F}(x, \mathbf{y}), \mathbf{y}(a) = \boldsymbol{\alpha}$$

The general form of a first-order differential equation is

$$y' = f(x, y) \tag{7.1a}$$

where $y' = dy/dx$ and $f(x, y)$ is a given function. The solution of this equation contains an arbitrary constant (the constant of integration). To find this constant, we must know a point on the solution curve; that is, y must be specified at some value of x, say, at $x = a$. We write this auxiliary condition as

$$y(a) = \alpha \tag{7.1b}$$

An ordinary differential equation of order n

$$y^{(n)} = f\left(x, y, y', \ldots, y^{(n-1)}\right) \tag{7.2}$$

can always be transformed into n first-order equations. Using the notation

$$y_1 = y \qquad y_2 = y' \qquad y_3 = y'' \qquad \ldots \qquad y_n = y^{(n-1)} \tag{7.3}$$

the equivalent first-order equations are

$$y_1' = y_2 \qquad y_2' = y_3 \qquad y_3' = y_4 \qquad \ldots \qquad y_n' = f(x, y_1, y_2, \ldots, y_n) \tag{7.4a}$$

The solution now requires the knowledge of n auxiliary conditions. If these conditions are specified at the same value of x, the problem is said to be an *initial value problem*. Then the auxiliary conditions, called *initial conditions*, have the form

$$y_1(a) = \alpha_1 \qquad y_2(a) = \alpha_2 \qquad y_3(a) = \alpha_3 \qquad \ldots \qquad y_n(a) = \alpha_n \tag{7.4b}$$

If y_i are specified at different values of x, the problem is called a *boundary value problem*.

For example,

$$y'' = -y \qquad y(0) = 1 \qquad y'(0) = 0$$

is an initial value problem since both auxiliary conditions imposed on the solution are given at $x = 0$. On the other hand,

$$y'' = -y \qquad y(0) = 1 \qquad y(\pi) = 0$$

is a boundary value problem because the two conditions are specified at different values of x.

use of vector notation, which allows us to manipulate sets of first-order equations in a concise form. For example, Eqs. (7.4) are written as

$$\mathbf{y}' = \mathbf{F}(x, \mathbf{y}) \qquad \mathbf{y}(a) = \boldsymbol{\alpha} \tag{7.5a}$$

where

$$\mathbf{F}(x, \mathbf{y}) = \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_n \\ f(x, \mathbf{y}) \end{bmatrix} \tag{7.5b}$$

A numerical solution of differential equations is essentially a table of x- and $\mathbf{y}$-values listed at discrete intervals of x.

Taylor Series Approach

The Taylor series method is conceptually simple and capable of high accuracy. Its basis is the truncated Taylor series for $\mathbf{y}$ about x:

$$\mathbf{y}(x+h) \approx \mathbf{y}(x) + \mathbf{y}'(x)h + \frac{1}{2!}\mathbf{y}''(x)h^2 + \frac{1}{3!}\mathbf{y}'''(x)h^3 + \cdots + \frac{1}{n!}\mathbf{y}^{(m)}(x)h^m \tag{7.6}$$

Because Eq. (7.6) predicts $\mathbf{y}$ at $x+h$ from the information available at x, it is also a formula for numerical integration. The last term kept in the series determines the *order of integration*. For the series in Eq. (7.6) the integration order is m.

The truncation error, due to the terms omitted from the series, is

$$\mathbf{E} = \frac{1}{(m+1)!}\mathbf{y}^{(m+1)}(\xi)h^{m+1}, \quad x < \xi < x+h$$

Using the finite difference approximation

$$\mathbf{y}^{(m+1)}(\xi) \approx \frac{\mathbf{y}^{(m)}(x+h) - \mathbf{y}^{(m)}(x)}{h}$$

we obtain the more usable form

$$\mathbf{E} \approx \frac{h^m}{(m+1)!}\left[\mathbf{y}^{(m)}(x+h) - \mathbf{y}^{(m)}(x)\right] \tag{7.7}$$

which could be incorporated in the algorithm to monitor the error in each integration step.

■ taylor

The function `taylor` implements the Taylor series method of integration order four. It can handle any number of first-order differential equations $y_i' = f_i(x, y_1, y_2, \ldots, y_n)$, $i = 1, 2, \ldots, n$. The user is required to supply the function `deriv` that returns the $4 \times n$ array

$$\mathbf{d} = \begin{bmatrix} (\mathbf{y}')^T \\ (\mathbf{y}'')^T \\ (\mathbf{y}''')^T \\ (\mathbf{y}^{(4)})^T \end{bmatrix} = \begin{bmatrix} y_1' & y_2' & \cdots & y_n' \\ y_1'' & y_2'' & \cdots & y_n'' \\ y_1''' & y_2''' & \cdots & y_n''' \\ y_1^{(4)} & y_2^{(4)} & \cdots & y_n^{(4)} \end{bmatrix}$$

The function returns the arrays `xSol` and `ySol` that contain the values of x and $\mathbf{y}$ at intervals h.

■ printSol

This function prints the results `xSol` and `ySol` in tabular form. The amount of data is controlled by the printout frequency `freq`. For example, if `freq = 5`, every fifth integration step would be displayed. If `freq = 0`, only the initial and final values will be shown.

EXAMPLE 7.1

Given that

$$y' + 4y = x^2 \qquad y(0) = 1$$

determine $y(0.2)$ with the fourth-order Taylor series method using a single integration step. Also compute the estimated error from Eq. (7.7) and compare it with the actual error. The analytical solution of the differential equation is

$$y = \frac{31}{32}e^{-4x} + \frac{1}{4}x^2 - \frac{1}{8}x + \frac{1}{32}$$

Solution The Taylor series up to and including the term with h^4 is

$$y(h) = y(0) + y'(0)h + \frac{1}{2!}y''(0)h^2 + \frac{1}{3!}y'''(0)h^3 + \frac{1}{4!}y^{(4)}(0)h^4$$

Differentiation of the differential equation yields

$$\begin{aligned} y' &= -4y + x^2 \\ y'' &= -4y' + 2x = 16y - 4x^2 + 2x \\ y''' &= 16y' - 8x + 2 = -64y + 16x^2 - 8x + 2 \\ y^{(4)} &= -64y' + 32x - 8 = 256y - 64x^2 + 32x - 8 \end{aligned}$$

Thus

$$\begin{aligned} y'(0) &= -4(1) = -4 \\ y''(0) &= 16(1) = 16 \\ y'''(0) &= -64(1) + 2 = -62 \\ y^{(4)}(0) &= 256(1) - 8 = 248 \end{aligned}$$

With $h = 0.2$, Eq. (a) becomes

$$\begin{aligned} y(0.2) &= 1 + (-4)(0.2) + \frac{1}{2!}(16)(0.2)^2 + \frac{1}{3!}(-62)(0.2)^3 + \frac{1}{4!}(248)(0.2)^4 \\ &= 0.4539 \end{aligned}$$

According to Eq. (7.7) the approximate truncation error is

$$E = \frac{h^4}{5!}\left[y^{(4)}(0.2) - y^{(4)}(0)\right]$$

where

$$y^{(4)}(0) = 248$$

$$y^{(4)}(0.2) = 256(0.4539) - 64(0.2)^2 + 32(0.2) - 8 = 112.04$$

Therefore,

$$E = \frac{(0.2)^4}{5!}(112.04 - 248) = -0.0018$$

The analytical solution yields

$$y(0.2) = \frac{31}{32}e^{-4(0.2)} + \frac{1}{4}(0.2)^2 - \frac{1}{8}(0.2) + \frac{1}{32} = 0.4515$$

so that the actual error is $0.4515 - 0.4539 = -0.0024$.

EXAMPLE 7.2

Solve

$$y'' = -0.1y' - x \qquad y(0) = 0 \qquad y'(0) = 1$$

from $x = 0$ to 2 with the Taylor series method of order four using $h = 0.25$.

Solution With $y_1 = y$ and $y_2 = y'$ the equivalent first-order equations and initial conditions are

$$\mathbf{y}' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_2 \\ -0.1y_2 - x \end{bmatrix} \qquad \mathbf{y}(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Repeated differentiation of the differential equations yields

$$\mathbf{y}'' = \begin{bmatrix} y_2' \\ -0.1y_2' - 1 \end{bmatrix} = \begin{bmatrix} -0.1y_2 - x \\ 0.01y_2 + 0.1x - 1 \end{bmatrix}$$

$$\mathbf{y}''' = \begin{bmatrix} -0.1y_2' - 1 \\ 0.01y_2' + 0.1 \end{bmatrix} = \begin{bmatrix} 0.01y_2 + 0.1x - 1 \\ -0.001y_2 - 0.01x + 0.1 \end{bmatrix}$$

$$\mathbf{y}^{(4)} = \begin{bmatrix} 0.01y_2' + 0.1 \\ -0.001y_2' - 0.01 \end{bmatrix} = \begin{bmatrix} -0.001y_2 - 0.01x + 0.1 \\ 0.0001y_2 + 0.001x - 0.01 \end{bmatrix}$$

Thus the derivative array required by `taylor` is

$$\mathbf{d} = \begin{bmatrix} y_2 & -0.1y_2 - x \\ -0.1y_2 - x & 0.01y_2 + 0.1x - 1 \\ 0.01y_2 + 0.1x - 1 & -0.001y_2 - 0.01x + 0.1 \\ -0.001y_2 - 0.01x + 0.1 & 0.0001y_2 + 0.001x - 0.01 \end{bmatrix}$$

which is computed by

```
function d = fex7_2(x,y)
% Derivatives used in Example 7.2

d = zeros(4,2);
d(1,1) = y(2);
d(1,2) = -0.1*y(2) - x;
d(2,1) = d(1,2);
d(2,2) = 0.01*y(2) + 0.1*x -1;
d(3,1) = d(2,2);
d(3,2) = -0.001*y(2) - 0.01*x + 0.1;
d(4,1) = d(3,2);
d(4,2) = 0.0001*y(2) + 0.001*x - 0.01;
```

```
>> [x,y] = taylor(@fex7_2, 0, [0 1], 2, 0.25);
>> printSol(x,y,1)
```

x	y1	y2
0.0000e+000	0.0000e+000	1.0000e+000
2.5000e-001	2.4431e-001	9.4432e-001
5.0000e-001	4.6713e-001	8.2829e-001
7.5000e-001	6.5355e-001	6.5339e-001
1.0000e+000	7.8904e-001	4.2110e-001
1.2500e+000	8.5943e-001	1.3281e-001
1.5000e+000	8.5090e-001	-2.1009e-001
1.7500e+000	7.4995e-001	-6.0625e-001
2.0000e+000	5.4345e-001	-1.0543e+000

The analytical solution of the problem is

$$y = 100x - 5x^2 + 990(e^{-0.1x} - 1)$$

from which we obtain $y(2) = 0.543\,45$ and $y'(2) = -1.0543$, which agree with the numerical solution.

Runge–Kutta Methods

The aim of Runge–Kutta methods is to eliminate the need for repeated differentiation of the differential equations. Since no such differentiation is involved in the first-order Taylor series integration formula

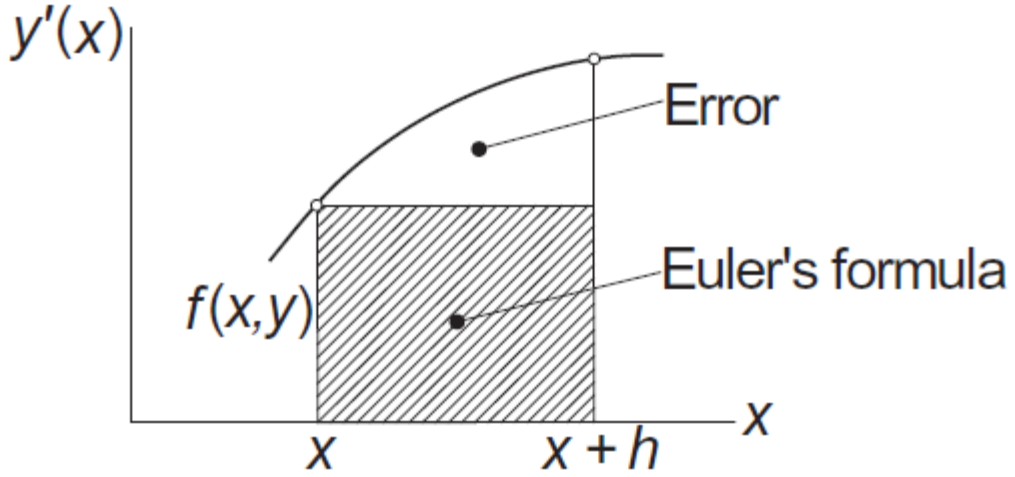
$$\mathbf{y}(x+h) = \mathbf{y}(x) + \mathbf{y}'(x)h = \mathbf{y}(x) + \mathbf{F}(x, \mathbf{y})h \tag{7.8}$$

it can be considered as the first-order Runge–Kutta method; it is also called *Euler's method*. Due to excessive truncation error, this method is rarely used in practice.

Let us now take a look at the graphical interpretation of Euler's equation. For the sake of simplicity, we assume that there is a single-dependent variable y, so that the differential equation is $y' = f(x, y)$. The change in the solution y between x and $x + h$ is

$$y(x+h) - y(h) = \int_x^{x+h} y'\,dx = \int_x^{x+h} f(x, y)dx$$

Truncation error is proportional to y''(x)



Graphical representation of error in euler integration

Second-Order Runge–Kutta Method

To arrive at the second-order method, we assume an integration formula of the form

$$\mathbf{y}(x+h) = \mathbf{y}(x) + c_0\mathbf{F}(x,\mathbf{y})h + c_1\mathbf{F}\left[x+ph, \mathbf{y}+qh\mathbf{F}(x,\mathbf{y})\right]h \tag{a}$$

and attempt to find the parameters c_0, c_1, p, and q by matching Eq. (a) to the Taylor series

$$\begin{aligned}\mathbf{y}(x+h) &= \mathbf{y}(x) + \mathbf{y}'(x)h + \frac{1}{2!}\mathbf{y}''(x)h^2 + \mathcal{O}(h^3)\\ &= \mathbf{y}(x) + \mathbf{F}(x,\mathbf{y})h + \frac{1}{2}\mathbf{F}'(x,\mathbf{y})h^2 + \mathcal{O}(h^3)\end{aligned} \tag{b}$$

Noting that

$$\mathbf{F}'(x,\mathbf{y}) = \frac{\partial \mathbf{F}}{\partial x} + \sum_{i=1}^{n}\frac{\partial \mathbf{F}}{\partial y_i}y_i' = \frac{\partial \mathbf{F}}{\partial x} + \sum_{i=1}^{n}\frac{\partial \mathbf{F}}{\partial y_i}F_i(x,\mathbf{y})$$

where n is the number of first-order equations, Eq.(b) can be written as

$$\mathbf{y}(x+h) = \mathbf{y}(x) + \mathbf{F}(x,\mathbf{y})h + \frac{1}{2}\left(\frac{\partial \mathbf{F}}{\partial x} + \sum_{i=1}^{n}\frac{\partial \mathbf{F}}{\partial y_i}F_i(x,\mathbf{y})\right)h^2 + \mathcal{O}(h^3) \tag{c}$$

Returning to Eq. (a), we can rewrite the last term by applying Taylor series in several variables:

$$\mathbf{F}\left[x+ph, \mathbf{y}+qh\mathbf{F}(x,\mathbf{y})\right] = \mathbf{F}(x,\mathbf{y}) + \frac{\partial \mathbf{F}}{\partial x}ph + qh\sum_{i=1}^{n}\frac{\partial \mathbf{F}}{\partial y_i}F_i(x,\mathbf{y}) + \mathcal{O}(h^2)$$

Returning to Eq. (a), we can rewrite the last term by applying Taylor series in several variables:

$$\mathbf{F}\left[x+ph,\mathbf{y}+qh\mathbf{F}(x,\mathbf{y})\right]=\mathbf{F}(x,\mathbf{y})+\frac{\partial\mathbf{F}}{\partial x}ph+qh\sum_{i=1}^{n}\frac{\partial\mathbf{F}}{\partial y_i}F_i(x,\mathbf{y})+\mathcal{O}(h^2)$$

so that Eq. (a) becomes

$$\mathbf{y}(x+h)=\mathbf{y}(x)+(c_0+c_1)\,\mathbf{F}(x,\mathbf{y})h+c_1\left[\frac{\partial\mathbf{F}}{\partial x}ph+qh\sum_{i=1}^{n}\frac{\partial\mathbf{F}}{\partial y_i}F_i(x,\mathbf{y})\right]h+\mathcal{O}(h^3) \quad \text{(d)}$$

Comparing Eqs. (c) and (d), we find that they are identical if

$$c_0+c_1=1 \qquad c_1p=\frac{1}{2} \qquad c_1q=\frac{1}{2} \quad \text{(e)}$$

$c_0 = 0$	$c_1 = 1$	$p = 1/2$	$q = 1/2$	Modified Euler's method
$c_0 = 1/2$	$c_1 = 1/2$	$p = 1$	$q = 1$	Heun's method
$c_0 = 1/3$	$c_1 = 2/3$	$p = 3/4$	$q = 3/4$	Ralston's method

All these formulas are classified as second-order Runge–Kutta methods, with no formula having a numerical superiority over the others. Choosing the *modified Euler's method*, substitution of the corresponding parameters into Eq. (a) yields

$$\mathbf{y}(x+h) = \mathbf{y}(x) + \mathbf{F}\left[x + \frac{h}{2}, \mathbf{y} + \frac{h}{2}\mathbf{F}(x, \mathbf{y})\right] h \tag{f}$$

This integration formula can be conveniently evaluated by the following sequence of operations

$$\begin{aligned} \mathbf{K}_1 &= h\mathbf{F}(x, \mathbf{y}) \\ \mathbf{K}_2 &= h\mathbf{F}\left(x + \frac{h}{2}, \mathbf{y} + \frac{1}{2}\mathbf{K}_1\right) \\ \mathbf{y}(x+h) &= \mathbf{y}(x) + \mathbf{K}_2 \end{aligned} \tag{7.9}$$

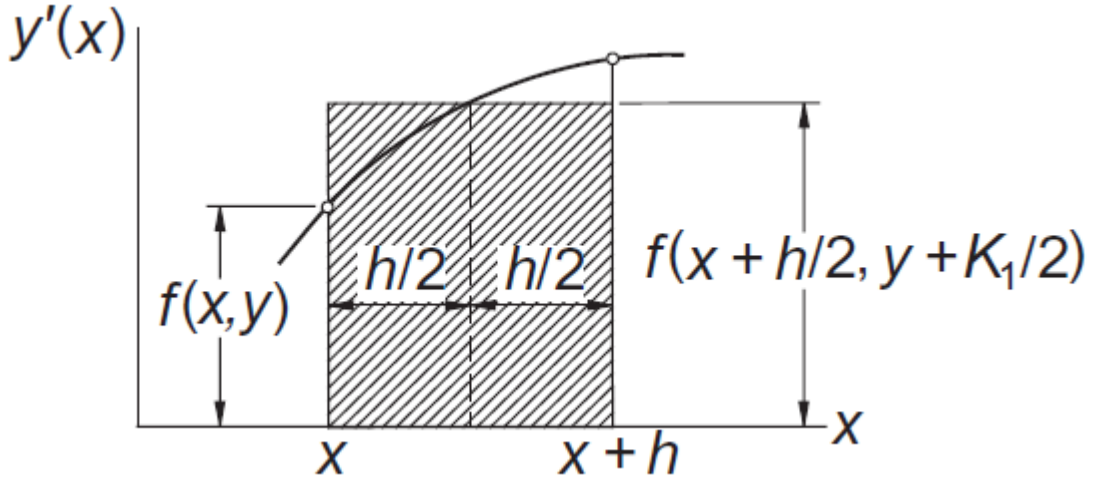
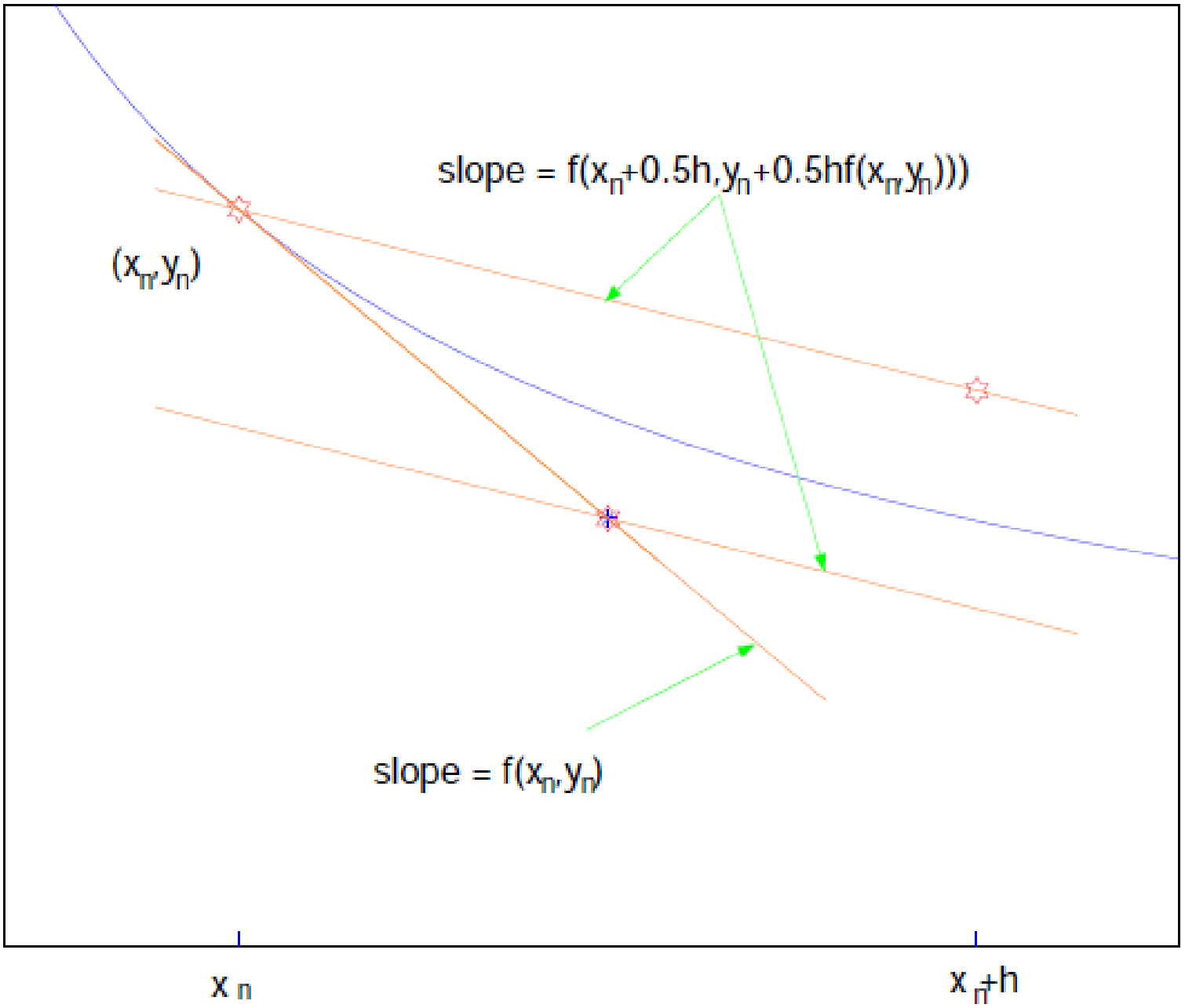


Figure 7.2 displays the graphical interpretation of modified Euler's formula for a single differential equation $y' = f(x, y)$. The first of Eqs. (7.9) yields an estimate of y at the midpoint of the panel by Euler's formula: $y(x + h/2) = y(x) + f(x, y)h/2 = y(x) + K_1/2$. The second equation then approximates the area of the panel by the area K_2 of the cross-hatched rectangle. The error here is proportional to the curvature y''' of the plot.


slope = f(x_n+0.5h,y_n+0.5hf(x_n,y_n)))
(x_n,y_n)
slope = f(x_n,y_n)
x_n
x_n+h

Fourth Order Runge Kutta Method

$$\mathbf{K}_1 = h\mathbf{F}(x, \mathbf{y})$$

$$\mathbf{K}_2 = h\mathbf{F}\left(x + \frac{h}{2}, \mathbf{y} + \frac{\mathbf{K}_1}{2}\right)$$

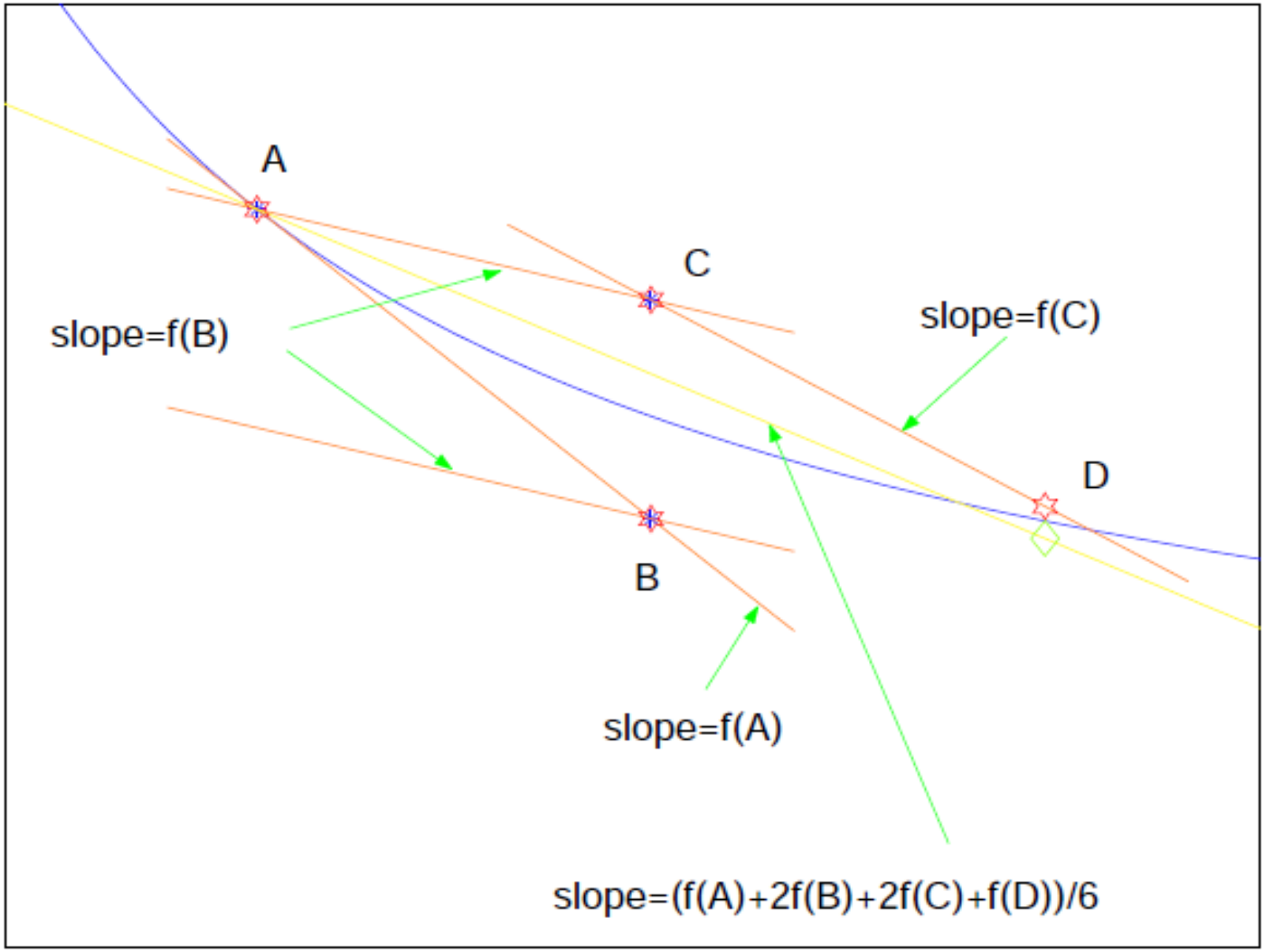
$$\mathbf{K}_3 = h\mathbf{F}\left(x + \frac{h}{2}, y + \frac{\mathbf{K}_2}{2}\right)$$

$$\mathbf{K}_4 = h\mathbf{F}(x + h, \mathbf{y} + \mathbf{K}_3)$$

$$\mathbf{y}(x + h) = \mathbf{y}(x) + \frac{1}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4)$$

- k_1 is the delta based on the slope at the beginning of the interval, using y_n, (Euler'
- k_2 is the delta based on the slope at the midpoint of the interval, using $y_n + 1/2\ k_1$
- k_3 is again the delta based on the slope at the midpoint, but now using $y_n + 1/2\ k_2$
- k_4 is the delta based on the slope at the end of the interval, using $y_n + k_3$.

The main drawback of this method is that is does not lend itself to an estimate of the truncation error. Therefore, we must guess the integration step size h, or determine it by trial and error. In contrast, the so-called *adaptive methods* can evaluate the truncation error in each integration step and adjust the value of h accordingly (but at a higher cost of computation). One such adaptive method is introduced in the next section.


A
C
slope=f(B)
slope=f(C)
D
B
slope=f(A)
slope=(f(A)+2f(B)+2f(C)+f(D))/6

■ runKut4

The function `runKut4` implements the Runge–Kutta method of order four. The user must provide `runKut4` with the function `dEqs` that defines the first-order differential equations $\mathbf{y}' = \mathbf{F}(x, \mathbf{y})$.

EXAMPLE 7.4

Solve

$$y'' = -0.1y' - x \qquad y(0) = 0 \qquad y'(0) = 1$$

from $x = 0$ to 2 in increments of $h = 0.25$ with the fourth-order Runge–Kutta method. (This problem was solved by the Taylor series method in Example 7.2.)

Solution Letting $y_1 = y$ and $y_2 = y'$, the equivalent first-order equations are

$$\mathbf{F}(x, \mathbf{y}) = \mathbf{y}' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_2 \\ -0.1y_2 - x \end{bmatrix}$$

which are coded in the following function:

```
function F = fex7_4(x,y)
% Differential. eqs. used in Example 7.4

F = zeros(1,2);
F(1) = y(2); F(2) = -0.1*y(2) - x;
```

```
>> [x,y] = runKut4(@fex7_4,0,[0 1],2,0.25);
>> printSol(x,y,1)
```

x	y1	y2
0.0000e+000	0.0000e+000	1.0000e+000
2.5000e-001	2.4431e-001	9.4432e-001
5.0000e-001	4.6713e-001	8.2829e-001
7.5000e-001	6.5355e-001	6.5339e-001
1.0000e+000	7.8904e-001	4.2110e-001
1.2500e+000	8.5943e-001	1.3281e-001
1.5000e+000	8.5090e-001	-2.1009e-001
1.7500e+000	7.4995e-001	-6.0625e-001
2.0000e+000	5.4345e-001	-1.0543e+000

$$y = Ce^{3x} + e^{-x}$$

which can be verified by substitution. The initial condition $y(0) = 1$ yields $C = 0$, so that the solution to the problem is indeed $y = e^{-x}$.

The cause of trouble in the numerical solution is the dormant term Ce^{3x}. Suppose that the initial condition contains a small error ε, so that we have $y(0) = 1 + \varepsilon$. This changes the analytical solution to

$$y = \varepsilon e^{3x} + e^{-x}$$

We now see that the term containing the error ε becomes dominant as x is increased. Since errors inherent in the numerical solution have the same effect as small changes in initial conditions, we conclude that our numerical solution is the victim of *numerical instability* due to sensitivity of the solution to initial conditions. The lesson here is: Do not always trust the results of numerical integration.

Stability of Euler's Method

As a simple illustration of stability, consider the problem

$$y' = -\lambda y \qquad y(0) = \beta \tag{7.11}$$

where λ is a positive constant. The exact solution of this problem is

$$y(x) = \beta e^{-\lambda x}$$

Let us now investigate what happens when we attempt to solve Eq. (7.11) numerically with Euler's formula

$$y(x + h) = y(x) + hy'(x) \tag{7.12}$$

Substituting $y'(x) = -\lambda y(x)$, we get

$$y(x+h) = (1-\lambda h)\, y(x)$$

If $|1-\lambda h| > 1$, the method is clearly unstable since $|y|$ increases in every integration step. Thus Euler's method is stable only if $|1-\lambda h| \leq 1$, or

$$h \leq 2/\lambda \tag{7.13}$$

The results can be extended to a system of n differential equations of the form

$$\mathbf{y}' = -\Lambda \mathbf{y} \tag{7.14}$$

where Λ is a constant matrix with the positive eigenvalues λ_i, $i = 1, 2, \ldots, n$. It can be shown that Euler's method of integration formula is stable if

$$h < 2/\lambda_{\max} \tag{7.15}$$

where $\lambda_{\max}$ is the largest eigenvalue of Λ.

Stiffness

An initial value problem is called *stiff* if some terms in the solution vector $\mathbf{y}(x)$ vary much more rapidly with x than others. Stiffness can be easily predicted for the differential equations $\mathbf{y}' = -\Lambda\mathbf{y}$ with constant coefficient matrix Λ. The solution of these equations is $\mathbf{y}(x) = \sum_i C_i\mathbf{v}_i \exp(-\lambda_i x)$, where λ_i are the eigenvalues of Λ and $\mathbf{v}_i$ are the corresponding eigenvectors. It is evident that the problem is stiff if there is a large disparity in the magnitudes of the positive eigenvalues.

Numerical integration of stiff equations requires special care. The step size h needed for stability is determined by the largest eigenvalue $\lambda_{\max}$, even if the terms $\exp(-\lambda_{\max} x)$ in the solution decay very rapidly and become insignificant as we move away from the origin.

For example, consider the differential equation[18]

$$y'' + 1001y' + 1000y = 0 \tag{7.16}$$

Using $y_1 = y$ and $y_2 = y'$, the equivalent first-order equations are

$$\mathbf{y}' = \begin{bmatrix} y_2 \\ -1000y_1 - 1001y_2 \end{bmatrix}$$

In this case

$$\Lambda = \begin{bmatrix} 0 & -1 \\ 1000 & 1001 \end{bmatrix}$$

The eigenvalues of Λ are the roots of

$$|\Lambda - \lambda \mathbf{I}| = \begin{vmatrix} -\lambda & -1 \\ 1000 & 1001 - \lambda \end{vmatrix} = 0$$

Expanding the determinant we get

$$-\lambda(1001 - \lambda) + 1000 = 0$$

which has the solutions $\lambda_1 = 1$ and $\lambda_2 = 1000$. These equations are clearly stiff. According to Eq. (7.15), we would need $h \leq 2/\lambda_2 = 0.002$ for Euler's method to be stable. The Runge–Kutta method would have approximately the same limitation on the step size.

EXAMPLE 7.7

(1) Show that the problem

$$y'' = -\frac{19}{4}y - 10y' \qquad y(0) = -9 \qquad y'(0) = 0$$

is moderately stiff and estimate $h_{\max}$, the largest value of h for which the Runge–Kutta method would be stable. (2) Confirm the estimate by computing $y(10)$ with $h \approx h_{\max}/2$ and $h \approx 2h_{\max}$.

Solution of Part (1) With the notation $y = y_1$ and $y' = y_2$ the equivalent first-order differential equations are

$$\mathbf{y}' = \begin{bmatrix} y_2 \\ -\frac{19}{4}y_1 - 10y_2 \end{bmatrix} = -\Lambda \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

where

$$\Lambda = \begin{bmatrix} 0 & -1 \\ \frac{19}{4} & 10 \end{bmatrix}$$

The eigenvalues of Λ are given by

$$|\Lambda - \lambda \mathbf{I}| = \begin{vmatrix} -\lambda & -1 \\ \frac{19}{4} & 10 - \lambda \end{vmatrix} = 0$$

Solution of Part (2) An estimate for the upper limit of the stable range of h can be obtained from Eq. (7.15):

$$h_{\max} = \frac{2}{\lambda_{\max}} = \frac{2}{19/2} = 0.2153$$

Although this formula is strictly valid for Euler's method, it is usually not too far off for higher-order integration formulas.

Here are the results from the Runge–Kutta method with $h = 0.1$ (by specifying `freq = 0` in `printSol`, only the initial and final values were printed):

```
>>        x              y1              y2
    0.0000e+000   -9.0000e+000    0.0000e+000
    1.0000e+001   -6.4011e-002    3.2005e-002
```

The analytical solution is

$$y(x) = -\frac{19}{2}e^{-x/2} + \frac{1}{2}e^{-19x/2}$$

```
>>        x              y1              y2
    0.0000e+000   -9.0000e+000    0.0000e+000
    1.0000e+001   -6.4011e-002    3.2005e-002
```

The analytical solution is

$$y(x) = -\frac{19}{2}e^{-x/2} + \frac{1}{2}e^{-19x/2}$$

yielding $y(10) = -0.0640\,11$, which agrees with the value obtained numerically.

With $h = 0.5$ we encountered instability, as expected:

```
>>        x              y1              y2
    0.0000e+000   -9.0000e+000    0.0000e+000
    1.0000e+001    2.7030e+020   -2.5678e+021
```

Adaptive Runge–Kutta Method

Determination of a suitable step size h can be a major headache in numerical integration. If h is too large, the truncation error may be unacceptable; if h is too small, we are squandering computational resources. Moreover, a constant step size may not be appropriate for the entire range of integration. For example, if the solution curve starts off with rapid changes before becoming smooth (as in a stiff problem), we should use a small h at the beginning and increase it as we reach the smooth region. This is where *adaptive methods* come in. They estimate the truncation error at each integration step and automatically adjust the step size to keep the error within prescribed limits.

The adaptive Runge–Kutta methods use so-called *embedded integration formulas*. These formulas come in pairs: one formula has the integration order m, the other one is of order $m+1$. The idea is to use both formulas to advance the solution from x to $x+h$. Denoting the results by $\mathbf{y}_m(x+h)$ and $\mathbf{y}_{m+1}(x+h)$, an estimate of the truncation error in the formula of order m is obtained from:

$$\mathbf{E}(h) = \mathbf{y}_{m+1}(x+h) - \mathbf{y}_m(x+h) \tag{7.17}$$

What makes the embedded formulas attractive is that they share the points where $\mathbf{F}(x, \mathbf{y})$ is evaluated. This means that once $\mathbf{y}_m(x+h)$ has been computed, relatively small additional effort is required to calculate $\mathbf{y}_{m+1}(x+h)$.

Here are the Runge–Kutta embedded formulas of orders five and four that were originally derived by Fehlberg; hence, they are known as *Runge–Kutta-Fehlberg formulas*:

$$\begin{aligned}\mathbf{K}_1 &= h\mathbf{F}(x,\, y)\\ \mathbf{K}_i &= h\mathbf{F}\left(x + A_i h, \mathbf{y} + \sum_{j=0}^{i-1} B_{ij}\mathbf{K}_j\right), \quad i = 2, 3, \ldots, 6\end{aligned} \tag{7.18}$$

$$\mathbf{y}_5(x+h) = \mathbf{y}(x) + \sum_{i=1}^{6} C_i\mathbf{K}_i \quad \text{(fifth-order formula)} \tag{7.19a}$$

$$\mathbf{y}_4(x+h) = \mathbf{y}(x) + \sum_{i=1}^{6} D_i\mathbf{K}_i \quad \text{(fourth-order formula)} \tag{7.19b}$$

i	A_i	B_{ij}					C_i	D_i
1	–	–	–	–	–	–	$\frac{37}{378}$	$\frac{2825}{27\,648}$
2	$\frac{1}{5}$	$\frac{1}{5}$	–	–	–	–	0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$	–	–	–	$\frac{250}{621}$	$\frac{18\,575}{48\,384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$	–	–	$\frac{125}{594}$	$\frac{13\,525}{55\,296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$	–	0	$\frac{277}{14\,336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$

Table 7.1. Cash–Karp coefficients for Runge–Kutta–Fehlberg formulas

The solution is advanced with the fifth-order formula in Eq. (7.19a). The fourth-order formula is used only implicitly in estimating the truncation error

$$\mathbf{E}(h) = \mathbf{y}_5(x+h) - \mathbf{y}_4(x+h) = \sum_{i=1}^{6} (C_i - D_i)\mathbf{K}_i \tag{7.20}$$

We could also control some gross measure of the error, such as the root-mean-square error defined by

$$\bar{E}(h) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} E_i^2(h)} \tag{7.22}$$

where n is the number of first-order equations. Then we would use

$$e(h) = \bar{E}(h) \tag{7.23}$$

Error control is achieved by adjusting the increment h so that the per-step error e is approximately equal to a prescribed tolerance ε. Noting that the truncation error in the fourth-order formula is $\mathcal{O}(h^5)$, we conclude that

$$\frac{e(h_1)}{e(h_2)} \approx \left(\frac{h_1}{h_2}\right)^5 \tag{a}$$

Let us now suppose that we performed an integration step with h_1 that resulted in the error $e(h_1)$. The step size h_2 that we should have used can now be obtained from Eq. (a) by setting $e(h_2) = \varepsilon$:

$$h_2 = h_1\left[\frac{e(h_1)}{\varepsilon}\right]^{1/5} \tag{b}$$

Factor of safety $\quad h_2 = 0.9h_1\left[\dfrac{e(h_1)}{\varepsilon}\right]^{1/5}$

EXAMPLE 7.8

The aerodynamic drag force acting on a certain object in free fall can be approximated by

$$F_D = av^2e^{-by}$$

where

v = velocity of the object in m/s

y = elevation of the object in meters

$a = 7.45$ kg/m

$b = 10.53 \times 10^{-5}$ m^{-1}

The exponential term accounts for the change of air density with elevation. The differential equation describing the fall is

$$m\ddot{y} = -mg + F_D$$

where $g = 9.80665$ m/s^2 and $m = 114$ kg is the mass of the object. If the object is released at an elevation of 9 km, determine its elevation and speed after a 10 s fall with the adaptive Runge–Kutta method.

Solution The differential equation and the initial conditions are

$$\ddot{y} = -g + \frac{a}{m}\dot{y}^2 \exp(-by)$$

$$= -9.80665 + \frac{7.45}{114}\dot{y}^2 \exp(-10.53 \times 10^{-5} y)$$

$$y(0) = 9000 \text{ m} \qquad \dot{y}(0) = 0$$

Letting $y_1 = y$ and $y_2 = \dot{y}$, the equivalent first-order equations and the initial conditions become

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -9.80665 + \left(65.351 \times 10^{-3}\right)(y_2)^2 \exp(-10.53 \times 10^{-5} y_1) \end{bmatrix}$$

$$\mathbf{y}(0) = \begin{bmatrix} 9000 \text{ m} \\ 0 \end{bmatrix}$$

The function describing the differential equations is

```
>> [x,y] = runKut5(@fex7_8,0,[9000 0],10,0.5,1.0e-2);
>> printSol(x,y,1
```

EXAMPLE 7.9

Integrate the moderately stiff problem

$$y'' = -\frac{19}{4}y - 10y' \qquad y(0) = -9 \qquad y'(0) = 0$$

from $x = 0$ to 10 with the adaptive Runge–Kutta method and plot the results (this problem also appeared in Example 7.7).

```
>> [x,y] = runKut5(@fex7_7,0,[-9 0],10,0.1);
>> printSol(x,y,4)
```

Bulrisch-Stoer Method

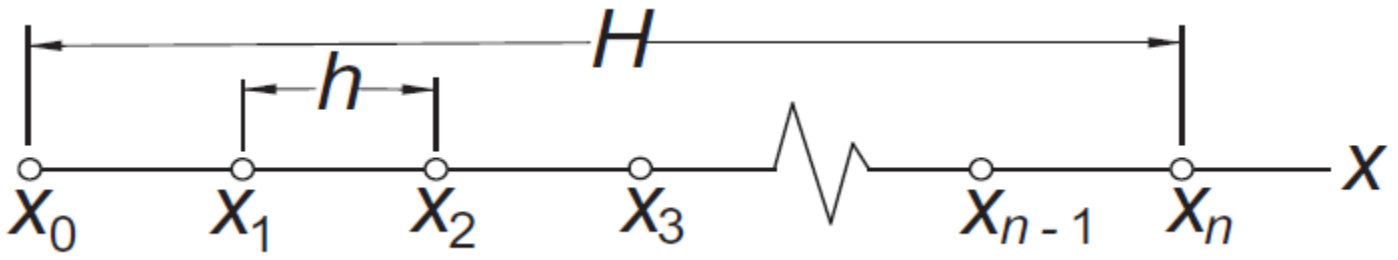
Midpoint Method

The midpoint formula of numerical integration of $\mathbf{y}' = \mathbf{F}(x, \mathbf{y})$ is

$$\mathbf{y}(x + h) = \mathbf{y}(x - h) + 2h\mathbf{F}\left[x, \mathbf{y}(x)\right]$$

Consider now advancing the solution of $\mathbf{y}'(x) = \mathbf{F}(x, \mathbf{y})$ from $x = x_0$ to $x_0 + H$ with the midpoint formula. We divide the interval of integration into n steps of length

$$h = H/n \text{ each}$$



$$\mathbf{y}_1 = \mathbf{y}_0 + h\mathbf{F}_0$$

$$\mathbf{y}_2 = \mathbf{y}_0 + 2h\mathbf{F}_1$$

$$\mathbf{y}_3 = \mathbf{y}_1 + 2h\mathbf{F}_2$$

$$\vdots$$

$$\mathbf{y}_n = \mathbf{y}_{n-2} + 2h\mathbf{F}_{n-1}$$

Here we used the notation $\mathbf{y}_i = \mathbf{y}(x_i)$ and $\mathbf{F}_i = \mathbf{F}(x_i, \mathbf{y}_i)$.

Taking the average of

$$\mathbf{y}_n \approx \mathbf{y}_{n-1} + h\mathbf{F}_n$$

And sum of previous terms: yn

To get

$$\mathbf{y}(x_0 + H) = \frac{1}{2}\left[\mathbf{y}_n + \left(\mathbf{y}_{n-1} + h\mathbf{F}_n\right)\right]$$

Error in this expression is

$$\mathbf{E} = \mathbf{c}_1 h^2 + \mathbf{c}_2 h^4 + \mathbf{c}_3 h^6 + \cdots$$

This is fantastic because we can now use Richardson's extrapolation

Herein lies the great utility of the midpoint method: we can eliminate as many of the leading error terms as we wish by Richarson's extrapolation. For example, we could compute $\mathbf{y}(x_0 + H)$ with a certain value of h and then repeat process with $h/2$. Denoting the corresponding results by $\mathbf{g}(h)$ and $\mathbf{g}(h/2)$, Richardson's extrapolation – see Eq. (5.9) – then yields the improved result

$$\mathbf{y}_{\text{better}}(x_0 + H) = \frac{4\mathbf{g}(h/2) - \mathbf{g}(h)}{3}$$

This is fourth order accurate

sequence $h/2,\ h/4,\ h/6,\ h/8,\ h/10, \cdots$, which has been found to be more economical.

■ midpoint

The function `integrate` in this module combines the midpoint method with Richardson extrapolation. The first application of the midpoint method uses two integration steps. The number of steps is increased by 2 in successive integrations, each integration being followed by Richardson extrapolation. The procedure is stopped when two successive solutions differ (in the root-mean-square sense) by less than a prescribed tolerance.

Bulirsch–Stoer Algorithm

When used on its own, the module `midpoint` has a major shortcoming: the solution at points between the initial and final values of x cannot be refined by Richardson extrapolation, so that $\mathbf{y}$ is usable only at the last point. This deficiency is rectified in the Bulirsch–Stoer method. The fundamental idea behind the method is simple: apply the midpoint method in a piecewise fashion. That is, advance the solution in stages of length H, using the midpoint method with Richardson extrapolation to perform the integration in each stage. The value of H can be quite large, since the precision of the result is determined mainly by the step length h in the midpoint method, not by H. However, if H is too large, the midpoint method may not converge. If this happens, try smaller value of H or larger error tolerance.

■ bulStoer

This function contains our greatly simplified algorithm for the Bulirsch–Stoer method.

Solution With $n = 2$, the step length is $h = 0.25$. The midpoint formulas, Eqs. (7.26) and (7.27) yield

$$
\begin{aligned}
y_1 &= y_0 + hf_0 = 1 + 0.25 \sin 1.0 = 1.210\,368 \\
y_2 &= y_0 + 2hf_1 = 1 + 2(0.25) \sin 1.210\,368 = 1.467\,87\,3 \\
y_h(0.5) &= \frac{1}{2}(y_1 + y_0 + hf_2) \\
&= \frac{1}{2}(1.210\,368 + 1.467\,87\,3 + 0.25 \sin 1.467\,87\,3) \\
&= 1.463\,459
\end{aligned}
$$

EXAMPLE 7.10

Compute the solution of the initial value problem

$$y' = \sin y \qquad y(0) = 1$$

at $x = 0.5$ with the midpoint formulas using $n = 2$ and $n = 4$, followed by Richardson extrapolation (this problem was solved with the second-order Runge–Kutta method in Example 7.3).

Using $n = 4$, we have $h = 0.125$ and the midpoint formulas become

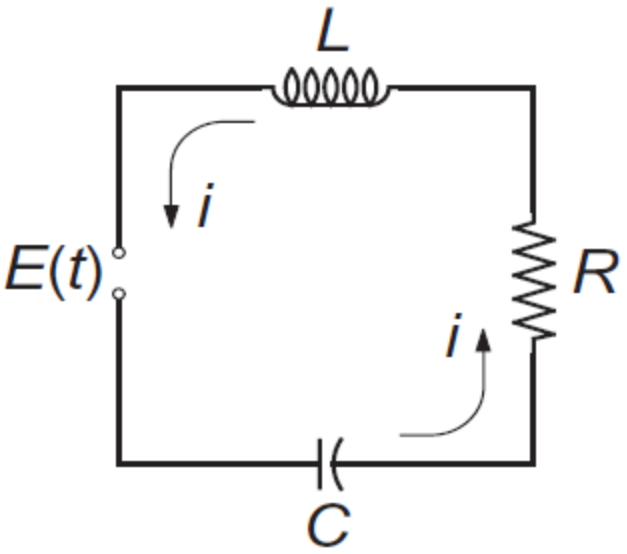
$$
\begin{aligned}
y_1 &= y_0 + hf_0 = 1 + 0.125 \sin 1.0 = 1.105\,184 \\
y_2 &= y_0 + 2hf_1 = 1 + 2(0.125) \sin 1.105\,184 = 1.223\,387 \\
y_3 &= y_1 + 2hf_2 = 1.105\,184 + 2(0.125) \sin 1.223\,387 = 1.340\,248 \\
y_4 &= y_2 + 2hf_3 = 1.223\,387 + 2(0.125) \sin 1.340\,248 = 1.466\,772 \\
y_{h/2}(0.5) &= \frac{1}{2}(y_4 + y_3 + hf_4) \\
&= \frac{1}{2}(1.466\,772 + 1.340\,248 + 0.125 \sin 1.466\,772) \\
&= 1.465\,672
\end{aligned}
$$

Richardson extrapolation results in

$$y(0.5) = \frac{4(1.465\,672) - 1.463\,459}{3} = 1.466\,410$$

which compares favorably with the "true" solution $y(0.5) = 1.466\,404$.

EXAMPLE 7.11



The differential equations governing the loop current i and the charge q on the capacitor of the electric circuit shown are

$$L\frac{di}{dt} + Ri + \frac{q}{C} = E(t) \qquad \frac{dq}{dt} = i$$

If the applied voltage E is suddenly increased from zero to 9 V, plot the resulting loop current during the first 10 s. Use $R = 1.0\,\Omega$, $L = 2\,\text{H}$, and $C = 0.45\,\text{F}$.

Solution Letting

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} q \\ i \end{bmatrix}$$

and substituting the given data, the differential equations become

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ (-Ry_1 - y_0/C + E)/L \end{bmatrix}$$

The initial conditions are

$$\mathbf{y}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

```
% Example 7.11 (Bulirsch-Stoer integration)
[xSol,ySol] = bulStoer(@fex7_11,0,[0 0],10,0.5);
plot(xSol,ySol(:,2),'k-o')
grid on
xlabel('Time (s)')
ylabel('Current (A)')
```

Numerical Methods in Structural Dynamics

- Implicit or explicit methods.
- Small time-step generally is good enough, but costly.
- Errors include erroneous phase-shift or artificial damping

Central Difference

At t = 0

$$m\,\ddot{v}_0 + c\,\dot{v}_0 + k\,v_0 = p_0$$

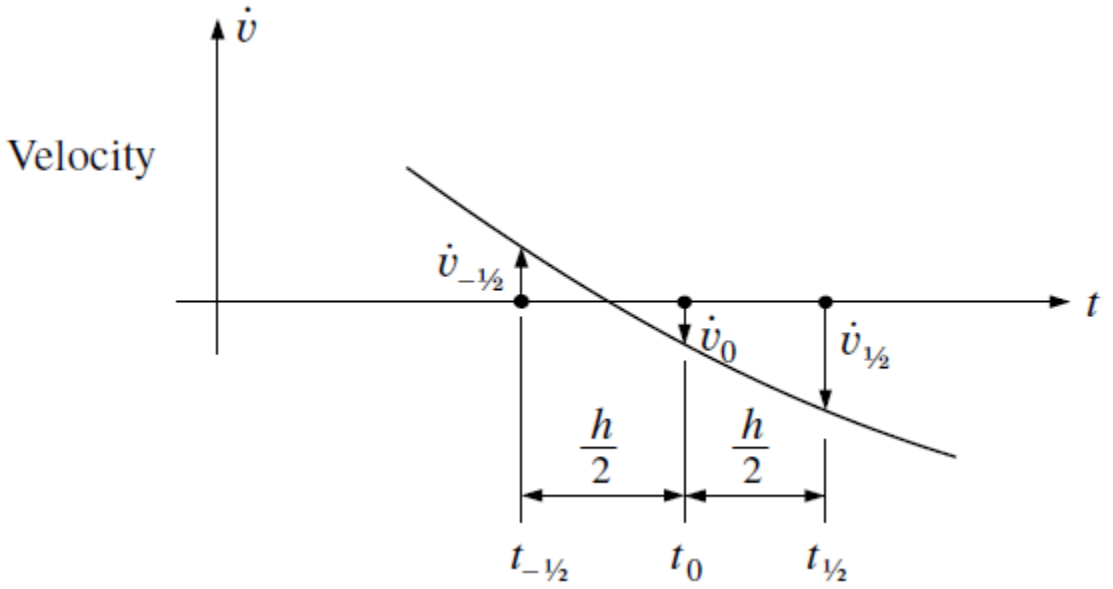
$$\ddot{v}_0 = \frac{1}{m}\left[p_0 - c\,\dot{v}_0 - k\,v_0\right]$$

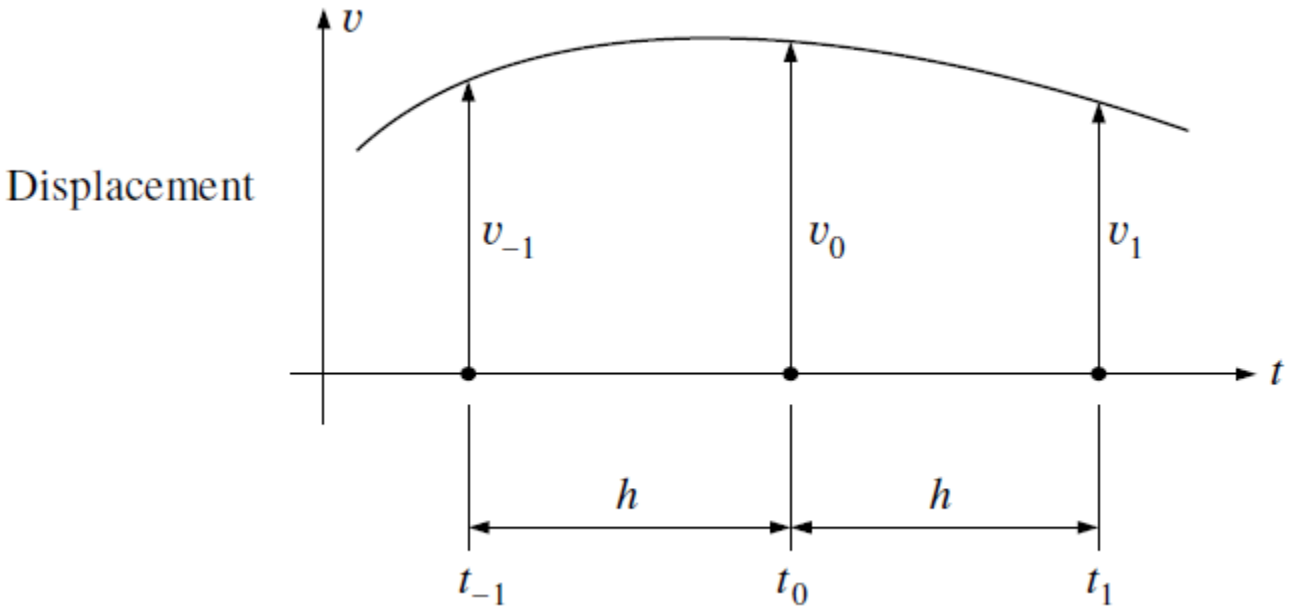
$$\dot{v}_{-1/2} \doteq \frac{v_0 - v_{-1}}{h} \qquad \dot{v}_{1/2} \doteq \frac{v_1 - v_0}{h}$$

$$\ddot{v}_0 \doteq \frac{\dot{v}_{1/2} - \dot{v}_{-1/2}}{h} \doteq \frac{1}{h^2}\left(v_1 - v_0\right) - \frac{1}{h^2}\left(v_0 - v_{-1}\right)$$

from which

$$\ddot{v}_0 = \frac{1}{h^2}\left(v_1 - 2\,v_0 + v_{-1}\right)$$


Velocity
$\dot{v}$
t
$\dot{v}_{-1/2}$
$\dot{v}_0$
$\dot{v}_{1/2}$
$\frac{h}{2}$
$\frac{h}{2}$
$t_{-1/2}$
t_0
$t_{1/2}$


Displacement
v
t
v_{-1}
v_0
v_1
h
h
t_{-1}
t_0
t_1

Substituting this expression into Eq. (7-7) then leads to

$$v_1 - 2\,v_0 + v_{-1} = \frac{h^2}{m}\,(p_0 - c\,\dot{v}_0 - k\,v_0)$$

and solving this for the displacement at the end of the time step results in

$$v_1 = \frac{h^2}{m}\,(p_0 - c\,\dot{v}_0 - k\,v_0) + 2\,v_0 - v_{-1}$$

$$\dot{v}_0 = \frac{v_1 - v_{-1}}{2h}$$

$$v_{-1} = v_1 - 2h\,\dot{v}_0$$

$$v_1 = v_0 + h\,\dot{v}_0 + \frac{h^2}{2m}\,(p_0 - c\,\dot{v}_0 - k\,v_0)$$

$$\frac{1}{2}\left(\dot{v}_0 + \dot{v}_1\right) = \frac{v_1 - v_0}{h}$$

$$\dot{v}_1 = \frac{2\left(v_1 - v_0\right)}{h} - \dot{v}_0$$

Conditionally stable, and will blow up if the following condition
Is not obeyed

$$\frac{h}{T} \leq \frac{1}{\pi} = 0.318$$

Newmark Beta Method

$$\dot{v}_1 = \dot{v}_0 + (1-\gamma)\, h\, \ddot{v}_0 + \gamma\, h\, \ddot{v}_1$$

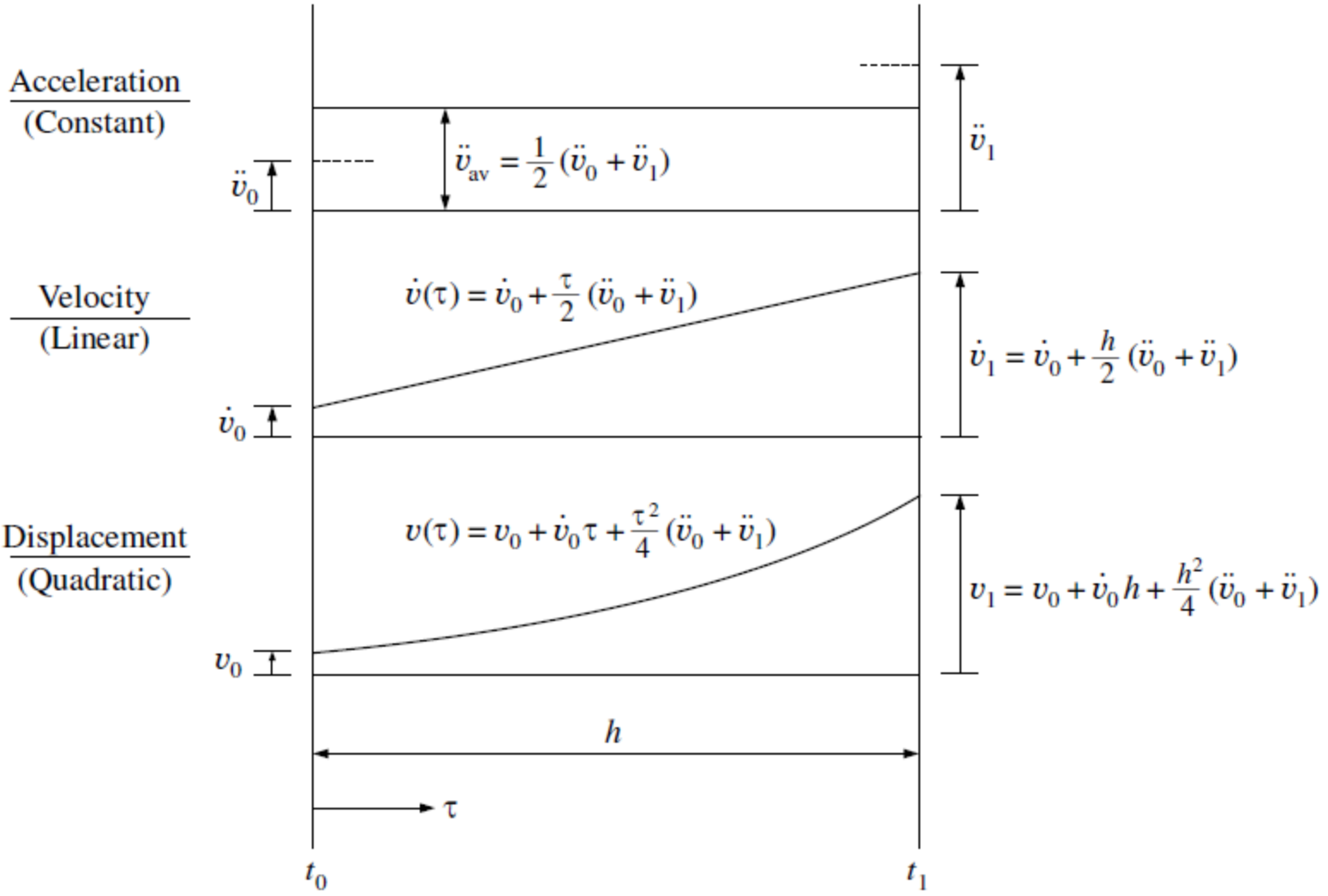
$$v_1 = v_0 + h\, \dot{v}_0 + (\frac{1}{2} - \beta)\, h^2\, \ddot{v}_0 + \beta\, h^2\, \ddot{v}_1$$

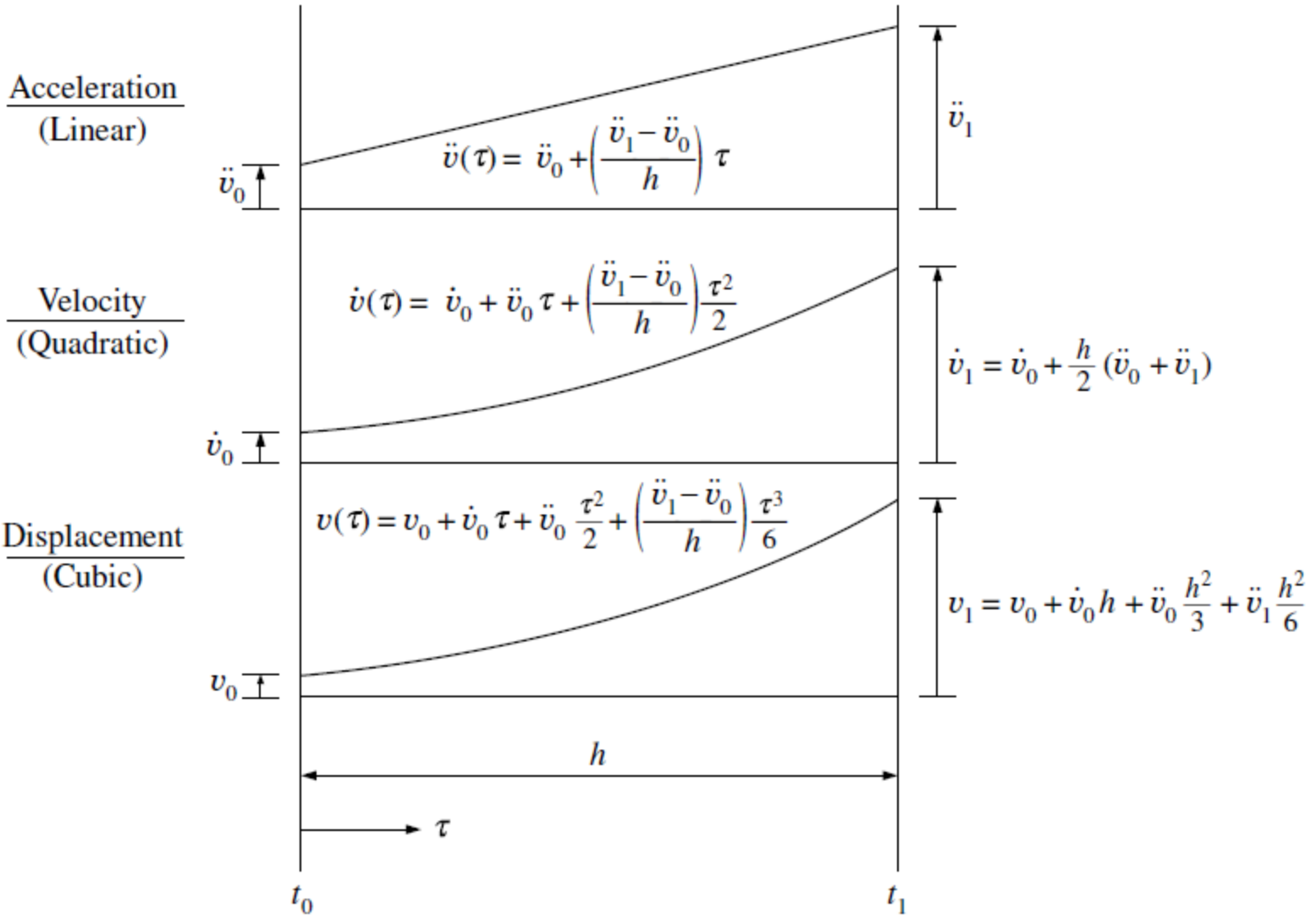
No artificial damping if γ = 1/2, so used for SDOF

Using γ = ½ and β = ¼ we reduce to the constant acceleration method (unconditonally stable)
Using γ = ½ and β = 1/6 we get linear acceleration method (conditionally stable: h/T < 0.55). Made unconditionally stable with Wilson's metho

$$\dot{v}_1 = \dot{v}_0 + \frac{h}{2}\,(\ddot{v}_0 + \ddot{v}_1)$$

$$v_1 = v_0 + \dot{v}_0\, h + \frac{h^2}{3}\,\ddot{v}_0 + \frac{h^2}{6}\,\ddot{v}_1$$


Acceleration
(Constant)
$\ddot{v}_{av} = \frac{1}{2}(\ddot{v}_0 + \ddot{v}_1)$
$\ddot{v}_0$
$\ddot{v}_1$
Velocity
(Linear)
$\dot{v}(\tau) = \dot{v}_0 + \frac{\tau}{2}(\ddot{v}_0 + \ddot{v}_1)$
$\dot{v}_1 = \dot{v}_0 + \frac{h}{2}(\ddot{v}_0 + \ddot{v}_1)$
$\dot{v}_0$
Displacement
(Quadratic)
$v(\tau) = v_0 + \dot{v}_0\tau + \frac{\tau^2}{4}(\ddot{v}_0 + \ddot{v}_1)$
$v_1 = v_0 + \dot{v}_0 h + \frac{h^2}{4}(\ddot{v}_0 + \ddot{v}_1)$
v_0
h
τ
t_0
t_1


Acceleration
(Linear)
$\ddot{v}(\tau) = \ddot{v}_0 + \left(\frac{\ddot{v}_1 - \ddot{v}_0}{h}\right)\tau$
$\ddot{v}_0$
$\ddot{v}_1$
Velocity
(Quadratic)
$\dot{v}(\tau) = \dot{v}_0 + \ddot{v}_0\,\tau + \left(\frac{\ddot{v}_1 - \ddot{v}_0}{h}\right)\frac{\tau^2}{2}$
$\dot{v}_1 = \dot{v}_0 + \frac{h}{2}(\ddot{v}_0 + \ddot{v}_1)$
$\dot{v}_0$
Displacement
(Cubic)
$v(\tau) = v_0 + \dot{v}_0\,\tau + \ddot{v}_0\,\frac{\tau^2}{2} + \left(\frac{\ddot{v}_1 - \ddot{v}_0}{h}\right)\frac{\tau^3}{6}$
$v_1 = v_0 + \dot{v}_0 h + \ddot{v}_0\frac{h^2}{3} + \ddot{v}_1\frac{h^2}{6}$
v_0
h
τ
t_0
t_1

Conversion to explicit form for γ = ½ and β = ¼

$$\ddot{v}_1 = \frac{4}{h^2}\,(v_1 - v_0) - \frac{4}{h}\,\dot{v}_0 - \ddot{v}_0$$

$$\dot{v}_1 = \frac{2}{h}\,(v_1 - v_0) - \dot{v}_0$$

$$m\,\ddot{v}_1 + c\,\dot{v}_1 + k\,v_1 = p_1$$

Writing equations in a special form

$$\tilde{k}_c\,v_1 = \tilde{p}_{1c} \qquad\qquad \tilde{k}_c = k + \frac{2c}{h} + \frac{4m}{h^2}$$

$$\tilde{p}_{1c} = p_1 + c\left(\frac{2v_0}{h} + \dot{v}_0\right) + m\left(\frac{4v_0}{h^2} + \frac{4}{h}\,\dot{v}_0 + \ddot{v}_0\right)$$

Finally we obtain the acceleration at the final step

$$\ddot{v}_1 = \frac{1}{m}\left(p_1 - c\,\dot{v}_1 - k\,v_1\right)$$

The algorithm proceeds further.
For a MDOF system we need to do one inversion
To solve matrix equation of the form
Kx = F