5

Numerical Solution of Differential and Integral Equations

• • •

The aspect of the calculus of Newton and Leibnitz that allowed the mathematical description of the physical world is the ability to incorporate derivatives and integrals into equations that relate various properties of the world to one another. Thus, much of the theory that describes the world in which we live is contained in what are known as differential and integral equations. Such equations appear not only in the physical sciences, but in biology, sociology, and all scientific disciplines that attempt to understand the world in which we live. Innumerable books and entire courses of study are devoted to the study of the solution of such equations and most college majors in science and engineering require at least one such course of their students. These courses generally cover the analytic closed form solution of such equations. But many of the equations that govern the physical world have no solution in closed form. Therefore, to find the answer to questions about the world in which we live, we must resort to solving these equations numerically. Again, the literature on this subject is voluminous, so we can only hope to provide a brief introduction to some of the basic methods widely employed in finding these solutions. Also, the subject is by no means closed so the student should be on the lookout for new techniques that prove increasingly efficient and accurate.

5.1 The Numerical Integration of Differential Equations

When we speak of a differential equation, we simply mean any equation where the dependent variable appears as well as one or more of its derivatives. The highest derivative that is present determines the *order* of the differential equation while the highest power of the dependent variable or its derivative appearing in the equation sets its *degree*. Theories which employ differential equations usually will not be limited to single equations, but may include sets of simultaneous equations representing the phenomena they describe. Thus, we must say something about the solutions of sets of such equations. Indeed, changing a high order differential equation into a system of first order differential equations is a standard approach to finding the solution to such equations. Basically, one simply replaces the higher order terms with new variables and includes the equations that define the new variables to form a set of first order simultaneous differential equations that replace the original equation. Thus a third order differential equation that had the form

$$f'''(x) + \alpha f''(x) + \beta f'(x) + \gamma f(x) = g(x) \quad , \tag{5.1.1}$$

could be replaced with a system of first order differential equations that looked like

$$\left.\begin{aligned} y'(x) + \alpha z'(x) + \beta f'(x) + \gamma f(x) &= g(x) \\ z'(x) &= y(x) \\ f'(x) &= z(x) \end{aligned}\right\} \quad . \tag{5.1.2}$$

This simplification means that we can limit our discussion to the solution of sets of first order differential equations with no loss of generality.

One remembers from beginning calculus that the derivative of a constant is zero. This means that it is always possible to add a constant to the general solution of a first order differential equation unless some additional constraint is imposed on the problem. These are generally called the *constants of integration*. These constants will be present even if the equations are inhomogeneous and in this respect differential equations differ significantly from functional algebraic equations. Thus, for a problem involving differential equations to be fully specified, the constants corresponding to the derivative present must be given in advance. The nature of the constants (i.e. the fact that their derivatives are zero) implies that there is some value of the independent variable for which the dependent variable has the value of the constant. Thus, constants of integration not only have a value, but they have a "place" where the solution has that value. If all the constants of integration are specified at the same place, they are called *initial values* and the problem of finding a solution is called an *initial value problem*. In addition, to find a numerical solution, the range of the independent variable for which the solution is desired must also be specified. This range must contain the initial value of the independent variable (i.e. that value of the independent variable corresponding to the location where the constants of integration are specified). On occasion, the constants of integration are specified at different locations. Such problems are known as boundary value problems and, as we shall see, these require a special approach. So let us begin our discussion of the numerical solution of ordinary differential equations by considering the solution of first order initial value differential equations.

The general approach to finding a solution to a differential equation (or a set of differential equations) is to begin the solution at the value of the independent variable for which the solution is equal to the initial values. One then proceeds in a step by step manner to change the independent variable and move

across the required range. Most methods for doing this rely on the local polynomial approximation of the solution and all the stability problems that were a concern for interpolation will be a concern for the numerical solution of differential equations. However, unlike interpolation, we are not limited in our choice of the values of the independent variable to where we can evaluate the dependent variable and its derivatives. Thus, the spacing between solution points will be a free parameter. We shall use this variable to control the process of finding the solution and estimating this error.

Since the solution is to be locally approximated by a polynomial, we will have constrained the solution and the values of the coefficients of the approximating polynomial. This would seem to imply that before we can take a new step in finding the solution, we must have prior information about the solution in order to provide those constraints. This "chicken or egg" aspect to solving differential equations would be removed if we could find a method that only depended on the solution at the previous step. Then we could start with the initial value(s) and generate the solution at as many additional values of the independent variable as we needed. Therefore let us begin by considering one-step methods.

a. One Step Methods of the Numerical Solution of Differential Equations

Probably the most conceptually simple method of numerically integrating differential equations is *Picard's method*. Consider the first order differential equation

$$y'(x) = g(x, y). \tag{5.1.3}$$

Let us directly integrate this over the small but finite range h so that

$$\int_{y_0}^{y} dy = \int_{x_0}^{x_0+h} g(x, y)\, dx , \tag{5.1.4}$$

which becomes

$$y(x) = y_0 + \int_{x_0}^{x_0+h} g(x, y)\, dx , \tag{5.1.5}$$

Now to evaluate the integral and obtain the solution, one must know the answer to evaluate g(x,y). This can be done iteratively by turning eq (5.1.5) into a fixed-point iteration formula so that

$$\left.\begin{aligned} y^{(k)}(x_0 + h) &= y_0 + \int_{x_0}^{x_0+h} g[x, y^{(k-1)}(x)]\, dx \\ y^{(k-1)}(x) &= y^{(k-1)}(x_0 + h) \end{aligned}\right\} . \tag{5.1.6}$$

A more inspired choice of the iterative value for $y^{(k-1)}(x)$ might be

$$y^{(k-1)}(x) = \tfrac{1}{2}[y_0 + y^{(k-1)}(x_0 + h)]. \tag{5.1.7}$$

However, an even better approach would be to admit that the best polynomial fit to the solution that can be achieved for two points is a straight line, which can be written as

$$y(x) = y_0 + a(x - x_0) = \{[y^{(k-1)}(x_0 + h)](x - x_0) + [y_0(x_0)](x_0 + h - x)]\} / h . \tag{5.1.8}$$

While the right hand side of equation (5.1.8) can be used as the basis for a fixed point iteration scheme, the iteration process can be completely avoided by taking advantage of the functional form of g(x,y). The linear

form of y can be substituted directly into g(x,y) to find the best value of a. The equation that constrains a is then simply

$$ah = \int_{x_0}^{x_0+h} g[x,(ax + y_0)]dx \quad . \tag{5.1.9}$$

This value of a may then be substituted directly into the center term of equation (5.1.8) which in turn is evaluated at $x = x_0+h$. Even should it be impossible to evaluate the right hand side of equation (5.1.9) in closed form any of the quadrature formulae of chapter 4 can be used to directly obtain a value for a. However, one should use a formula with a degree of precision consistent with the linear approximation of y.

To see how these various forms of Picard's method actually work, consider the differential equation

$$y'(x) = xy \quad , \tag{5.1.10}$$

subject to the initial conditions

$$y(0) = 1 \quad . \tag{5.1.11}$$

Direct integration yields the closed form solution

$$y = e^{x^2/2} \quad . \tag{5.1.12}$$

The rapidly varying nature of this solution will provide a formidable test of any integration scheme particularly if the step size is large. But this is exactly what we want if we are to test the relative accuracy of different methods.

In general, we can cast Picard's method as

$$y(x) = 1 + \int_0^z zy(z)\,dz \quad , \tag{5.1.13}$$

where equations (5.1.6) - (5.1.8) represent various methods of specifying the behavior of y(z) for purposes of evaluating the integrand. For purposes of demonstration, let us choose h = 1 which we know is unreasonably large. However, such a large choice will serve to demonstrate the relative accuracy of our various choices quite clearly. Further, let us obtain the solution at x = 1, and 2. The naive choice of equation (5.1.6) yields an iteration formula of the form

$$y(x_0 + h) = 1 + \int_{x_0}^{x_0+h} zy^{(k-1)}(x_0 + h)\,dz + 1 + [h(x_0 + h)/2]\,y^{(k-1)}(x_0 + h) \quad . \tag{5.1.14}$$

This may be iterated directly to yield the results in column (a) of table 5.1, but the fixed point can be found directly by simply solving equation (5.1.14) for $y^{(\infty)}(x_0+h)$ to get

$$y^{(\infty)}(x_0 + h) = (1 - hx_0 - h^2/2)^{-1} \quad . \tag{5.1.15}$$

For the first step when $x_0 = 0$, the limiting value for the solution is 2. However, as the solution proceeds, the iteration scheme clearly becomes unstable.

Table 5.1

Results for Picard's Method

	(A)	(B)	(C)	(D)
i	**y(1)**	**y(1)**	**y(1)**	**y_c(1)**
0	1.0	1.0		
1	1.5	1.5		
2	1.75	1.625		
3	1.875	1.6563		
4	1.938	1.6641		
5	1.969	1.6660		
∞	2.000	5/3	7/4	1.6487
i	**y(2)**	**y(2)**	**y(2)**	**y_c(2)**
0	4.0	1.6666		
1	7.0	3.0000		
2	11.5	4.5000		
3	18.25	5.6250		
4	28.375	6.4688		
5	43.56	7.1015		
∞	∞	9.0000	17.5	7.3891

Estimating the appropriate value of y(x) by averaging the values at the limits of the integral as indicated by equation (5.1.7) tends to stabilize the procedure yielding the iteration formula

$$y^{(k)}(x_0+h) = 1+\tfrac{1}{2}\int_{x_0}^{x_0+h} z[y(x_0)+y^{(k-1)}(x_0+h)\,dz = 1+[h(x_0+h)/2][y(x_0)+y^{(k-1)}(x_0+h)]/2\,, \tag{5.1.16}$$

the application of which is contained in column (b) of Table 5.1. The limiting value of this iteration formula can also be found analytically to be

$$y^{(\infty)}(x_0+h) = \frac{1 + [h(x_0+h/2)y(x_0)]/2}{[1 - h(x_0+h/2)/2]}\,, \tag{5.1.17}$$

which clearly demonstrates the stabilizing influence of the averaging process for this rapidly increasing solution.

Finally, we can investigate the impact of a linear approximation for y(x) as given by equation (5.1.8). Let us assume that the solution behaves linearly as suggested by the center term of equation (5.1.8).

This can be substituted directly into the explicit form for the solution given by equation (5.1.13) and the value for the slope, a, obtained as in equation (5.1.9). This process yields

$$a = y(x_0)(x_0+h/2)/[1-(x_0h/2)-(h^2/3)] \ , \qquad (5.1.18)$$

which with the linear form for the solution gives the solution without iteration. The results are listed in table 5.1 in column (c). It is tempting to think that a combination of the right hand side of equation (5.1.7) integrated in closed form in equation (5.1.13) would give a more exact answer than that obtained with the help of equation (5.1.18), but such is not the case. An iteration formula developed in such a manner can be iterated analytically as was done with equations (5.1.15) and (5.1.17) to yield exactly the results in column (c) of table 5.1. Thus the best one can hope for with a linear Picard's method is given by equation (5.1.8) with the slope, a, specified by equation (5.1.9).

However, there is another approach to finding one-step methods. The differential equation (5.1.3) has a full family of solutions depending on the initial value (i.e. the solution at the beginning of the step). That family of solutions is restricted by the nature of g(x,y). The behavior of that family in the neighborhood of $x = x_0+h$ can shed some light on the nature of the solution at $x = x_0+h$. This is the fundamental basis for one of the more successful and widely used one-step methods known as the *Runge-Kutta method*. The Runge-Kutta method is also one of the few methods in numerical analysis that does not rely directly on polynomial approximation for, while it is certainly correct for polynomials, the basic method assumes that the solution can be represented by a Taylor series.

So let us begin our discussion of Runge-Kutta formulae by assuming that the solution can be represented by a finite taylor series of the form

$$y_{n+1} = y_n + hy'_n + (h^2/2!)y''_n + \cdots + (h^k/k!)y_n^{(k)} \ . \qquad (5.1.19)$$

Now assume that the solution can also be represented by a function of the form

$$y_{n+1} = y_n + h\{\alpha_0 g(x_n,y_n)+\alpha_1 g[(x_n+\mu_1 h),(y_n+b_1 h)] +\alpha_2 g[(x_n+\mu_2 h),(y_n+b_2 h)]+ \cdots +\alpha_k g[(x_n+\mu_k h),(y_n+b_k h)]\} \ . \qquad (5.1.20)$$

This rather convoluted expression, while appearing to depend only on the value of y at the initial step (i.e. y_n) involves evaluating the function g(x,y) all about the solution point x_n, y_n (see Figure 5.1).

By setting equations (5.1.19) and (5.1.20) equal to each other, we see that we can write the solution in the from

$$y_{n+1} = y_n + \alpha_0 t_0 + \alpha_1 t_1 + \cdots + \alpha_k t_k \ , \qquad (5.1.21)$$

where the t_is can be expressed recursively by

$$\left.\begin{aligned} t_0 &= hg(x_n, y_n) \\ t_1 &= hg[(x_n + \mu_1 h),(y_n + \lambda_{1,0} t_0)] \\ t_2 &= hg[(x_n + \mu_2 h),(y_n + \lambda_{2,0} t_0 + \lambda_{2,1} t_1)] \\ &\vdots \qquad\qquad \vdots \\ t_k &= hg[(x_n + \mu_k h),(y_n + \lambda_{k,0} t_0 + \lambda_{k,1} t_1 + \cdots + \lambda_{k,k-1} t_{k-1})] \end{aligned}\right\} \ . \qquad (5.1.22)$$

Now we must determine k+1 values of α, k values of μ and $k\times(k+1)/2$ values of $\lambda_{i,j}$. But we only have k+1

terms of the Taylor series to act as constraints. Thus, the problem is hopelessly under-determined. Thus indeterminency will give rise to entire families of Runge-Kutta formulae for any order k. In addition, the algebra to eliminate as many of the unknowns as possible is quite formidable and not unique due to the undetermined nature of the problem. Thus we will content ourselves with dealing only with low order formulae which demonstrate the basic approach and nature of the problem. Let us consider the lowest order that provides some insight into the general aspects of the Runge-Kutta method. That is k=1. With k=1 equations (5.1.21) and (5.1.22) become

$$\left.\begin{aligned} y_{n+1} &= y_n + \alpha_0 t_0 + \alpha_1 t_1 \\ t_0 &= hg(x_n y_n) \\ t_1 &= hg[(x_n + \mu h), (y_n + \lambda t_0)] \end{aligned}\right\} . \qquad (5.1.23)$$

Here we have dropped the subscript on λ as there will only be one of them. However, there are still four free parameters and we really only have three equations of constraint.

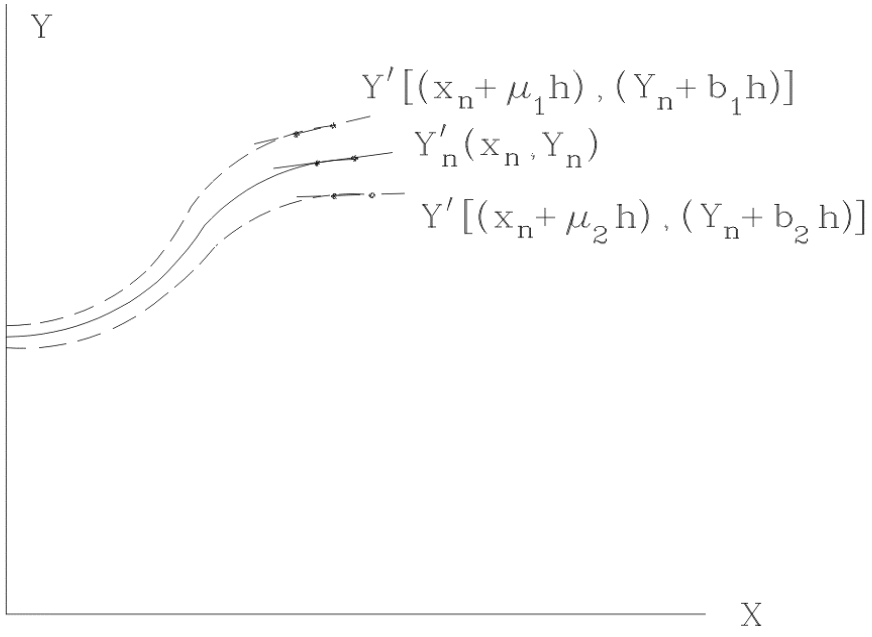


Figure 5.1 show the solution space for the differential equation y' = g(x,y). *Since the initial value is different for different solutions, the space surrounding the solution of choice can be viewed as being full of alternate solutions. The two dimensional Taylor expansion of the Runge-Kutta method explores this solution space to obtain a higher order value for the specific solution in just one step.*

If we expand g(x,y) about x_n, y_n, in a two dimensional taylor series, we can write

$$g[(x_n+\mu h),(y_n+\lambda t_0)] = g(x_n,y_n)+\mu h\frac{\partial g(x_n,y_n)}{\partial x}+\lambda t_0\frac{\partial g(x_n,y_n)}{\partial y}+\tfrac{1}{2}\mu^2 h^2\frac{\partial g(x_n,y_n)}{\partial x^2}$$
$$+\tfrac{1}{2}\lambda^2 t_0^2\frac{\partial^2 g(x_n,y_n)}{\partial y^2}+\mu\lambda t_0\frac{\partial^2 g(x_n,y_n)}{\partial x\partial y}+\cdots+ \quad . \qquad (5.1.24)$$

Making use of the third of equations (5.1.23), we can explicitly write t_1 as

$$t_1 = hg(x_n,y_n)+h^2\left[\mu\frac{\partial g(x_n,y_n)}{\partial x}+\lambda g(x_n,y_n)\frac{\partial g(x_n,y_n)}{\partial y}\right]$$
$$\tfrac{1}{2}h^3\left[\mu^2\frac{\partial^2 g(x_n,y_n)}{\partial x^2}+\lambda^2 g^2(x_n,y_n)\frac{\partial^2 g(x_n,y_n)}{\partial y^2}+2\mu\lambda g(x_n,y_n)\frac{\partial^2 g(x_n,y_n)}{\partial x\partial y}\right] \quad . \qquad (5.1.25)$$

Direct substitution into the first of equations (5.1.23) gives

$$y_{n+1} = y_n+h(\alpha_0+\alpha_1)g(x_n,y_n)+h^2\left[\mu\frac{\partial g(x_n,y_n)}{\partial x}+\lambda g(x_n,y_n)\frac{\partial g(x_n,y_n)}{\partial y}\right]$$
$$\tfrac{1}{2}h^3\alpha_1\left[\mu^2\frac{\partial^2 g(x_n,y_n)}{\partial x^2}+\lambda^2 g^2(x_n,y_n)\frac{\partial^2 g(x_n,y_n)}{\partial y^2}+2\mu\lambda g(x_n,y_n)\frac{\partial^2 g(x_n,y_n)}{\partial x\partial y}\right] \quad . \qquad (5.1.26)$$

We can also expand y' in a two dimensional taylor series making use of the original differential equation (5.1.3) to get

$$\left.\begin{aligned} y' &= g(x,y)\\ y'' &= \frac{\partial g(x,y)}{\partial x}+y'\frac{\partial g(x,y)}{\partial y}=\frac{\partial g(x,y)}{\partial x}+g(x,y)\frac{\partial g(x,y)}{\partial y}\\ y''' &= \frac{\partial y''}{\partial x}+y'\frac{\partial y''}{\partial y}=\frac{\partial^2 g(x,y)}{\partial x^2}+\frac{\partial g(x,y)}{\partial x}\bullet\frac{\partial g(x,y)}{\partial y}+g(x,y)\frac{\partial^2 g(x,y)}{\partial x\partial y}\\ &\quad +g(x,y)\frac{\partial^2 g(x,y)}{\partial y\partial x}+g(x,y)\left[\frac{\partial g(x,y)}{\partial y}\right]^2+g(x,y)\frac{\partial^2 g(x,y)}{\partial y^2} \end{aligned}\right\} . \qquad (5.1.27)$$

Substituting this into the standard form of the Taylor series as given by equation (5.1.19) yields

$$\left.\begin{aligned} y_{n+1} = y_n &+ hg(x,y)+h^2\left[\frac{\partial g(x,y)}{\partial x}+\lambda g(x,y)\frac{\partial g(x,y)}{\partial y}\right]+\frac{h^3}{6}\left(\frac{\partial^2 g(x,y)}{\partial x^2}+g^2(x,y)\frac{\partial^2 g(x,y)}{\partial y^2}\right.\\ &\left.+2g(x,y)\frac{\partial^2 g(x,y)}{\partial x\partial y}+\frac{\partial g(x,y)}{\partial y}\left[\frac{\partial g(x,y)}{\partial x}+g(x,y)\frac{\partial g(x,y)}{\partial y}\right]\right) \end{aligned}\right\} . \qquad (5.1.28)$$

Now by comparing this term by term with the expansion shown in equation (5.1.26) we can conclude that the free parameters α_0, α_1, μ, and λ must be constrained by

$$\left.\begin{aligned}(\alpha_0+\alpha_1)&=1\\ \alpha_1\mu&=\tfrac{1}{2}\\ \alpha_1\lambda&=\tfrac{1}{2}\end{aligned}\right\}. \tag{5.1.29}$$

As we suggested earlier, the formula is under-determined by one constraint. However, we may use the constraint equations as represented by equation (5.1.29) to express the free parameters in terms of a single constant c. Thus the parameters are

$$\left.\begin{aligned}\alpha_0&=1-c\\ \alpha_1&=c\\ \mu=\lambda&=\tfrac{1}{2}c\end{aligned}\right\}. \tag{5.1.30}$$

and the approximation formula becomes

$$\left.\begin{aligned}y_{n+1}=y_n+hg(x,y)+h^2\left[\frac{\partial g(x,y)}{\partial x}+\lambda g(x,y)\frac{\partial g(x,y)}{\partial y}\right]+\frac{h^3}{8c}\left[\frac{\partial^2 g(x,y)}{\partial x^2}+g^2(x,y)\frac{\partial^2 g(x,y)}{\partial y^2}\right.\\ \left.+2g(x,y)\frac{\partial^2 g(x,y)}{\partial x\partial y}\right]\end{aligned}\right\}. \tag{5.1.31}$$

We can match the first two terms of the Taylor series with any choice of c. The error term will than be of order $O(h^3)$ and specifically has the form

$$R_{n+1}=-\frac{h^3}{24c}\left([3-4c]y_n'''-3\frac{\partial g(x_n,y_n)}{\partial y}y_n''\right). \tag{5.1.32}$$

Clearly the most effective choice of c will depend on the solution so that there is no general "best" choice. However, a number of authors recommend c = ½ as a general purpose value.

If we increase the number of terms in the series, the under-determination of the constants gets rapidly worse. More and more parameters must be chosen arbitrarily. When these formulae are given, the arbitrariness has often been removed by *fiat*. Thus one may find various Runge-Kutta formulae of the same order. For example, a common such fourth order formula is

$$\left.\begin{aligned}y_{n+1}&=y_n+(t_0+2t_1+2t_2+t_3)/6\\ t_0&=hg(x_n,y_n)\\ t_1&=hg[(x_n+\tfrac{1}{2}h),(y_n+\tfrac{1}{2}t_0)]\\ t_2&=hg[(x_n+\tfrac{1}{2}h),(y_n+\tfrac{1}{2}t_1)]\\ t_3&=hg[(x_n+h),(y_n+t_2)]\end{aligned}\right\}. \tag{5.1.33}$$

Here the "best" choice for the under-determined parameters has already been made largely on the basis of experience.

If we apply these formulae to our test differential equation (5.1.10), we need first specify which Runge-Kutta formula we plan to use. Let us try the second order (i.e. exact for quadratic polynomials) formula given by equation (5.1.23) with the choice of constants given by equation (5.1.29) when c = ½. The formula then becomes

$$\left.\begin{aligned} y_{n+1} &= y_n + \tfrac{1}{2}t_0 + \tfrac{1}{2}t_1 \\ t_0 &= hg(x_n, y_n) \\ t_1 &= hg[(x_n + h), (y_n + t_0)] \end{aligned}\right\} . \qquad (5.1.34)$$

So that we may readily compare to the first order Picard formula, we will take h = 1 and y(0) = 1. Then taking g(x,y) from equation (5.1.10) we get for the first step that

$$\left.\begin{aligned} t_0 &= hx_0y_0 = (1)(0)(1) = 0 \\ t_1 &= h(x_0 + h)(y_0 + t_0) = (1)(0+1)(1+0) = 1 \\ y(x_0 + h) &= y_1 = (1) + (½)(0) + (½)(1) = \tfrac{3}{2} \end{aligned}\right\} . \qquad (5.1.35)$$

The second step yields

$$\left.\begin{aligned} t_0 &= hx_1y_1 = (1)(1)(3⁄2) = \tfrac{3}{2} \\ t_1 &= h(x_1 + h)(y_1 + t_0) = (1)(1+1)(1+3⁄2) = 5 \\ y(x_1 + h) &= y_2 = (3⁄2) + (½)(3⁄2) + (½)(5) = \tfrac{19}{4} \end{aligned}\right\} . \qquad (5.1.36)$$

Table 5.2
Sample Runge-Kutta Solutions

Second Order Solution **Fourth Order Solution**

Step 1

	h=1	h=1/2	y_c	h=1
i	t_i	t_i		t_i
0	0.0	[0 , 9/32]		0.00000
1	1.0	[1/4 , 45/64]		0.50000
2	----------	----------		0.62500
3	----------	----------		1.62500
y_1	1.5	1.6172	1.64587	1.65583

δy_1 0.1172

h'_1 0.8532*

Step 2

i	t_i	t_i		t_i
0	1.5	[0.8086 , 2.1984]		1.64583
1	5.0	[1.8193 , 5.1296]		3.70313
2	----------	----------		5.24609
3	----------	----------		13.78384
y_2	4.75	6.5951	7.38906	7.20051

δy_2 1.8451

h'_2 0.0635

* This value assumes that $\delta y_0 = 0.1$

The Runge-Kutta formula tends to under-estimate the solution in a systematic fashion. If we reduce the step size to h = ½ the agreement is much better as the error term in this formula is of $O(h^3)$. The results for h = ½ are given in table 5.2 along with the results for h = 1. In addition we have tabulated the results for the fourth order formula given by equation (5.1.33). For our example, the first step would require that equation (5.1.33) take the form

$$\left.\begin{aligned} t_0 &= hx_0y_0 = (1)(0)(1) = 0 \\ t_1 &= h(x_0 + \tfrac{1}{2}h)(y_0 + \tfrac{1}{2}t_0) = (1)(0 + ½)(1 + 0) = \tfrac{1}{2} \\ t_2 &= h(x_0 + \tfrac{1}{2}h)(y_0 + \tfrac{1}{2}t_1) = (1)(0 + ½)[1 + (½)(½)] = \tfrac{5}{8} \\ t_3 &= h(x_0 + \tfrac{1}{2}h)(y_0 + \tfrac{1}{2}t_2) = (1)(0 + 1)[1 + (½)(⅝)] = \tfrac{13}{8} \\ y(x_0 + h) = y_1 &= (1) + [(0) + 2(½) + 2(⅝) + (\tfrac{13}{8})]/6) = \tfrac{79}{48} \end{aligned}\right\} . \qquad (5.1.37)$$

The error term for this formula is of $O(h^5)$ so we would expect it to be superior to the second order formula for h = ½ and indeed it is. These results demonstrate that usually it is preferable to increase the accuracy of a solution by increasing the accuracy of the integration formula rather than decreasing the step size. The calculations leading to Table 5.2 were largely carried out using fractional arithmetic so as to eliminate the round-off error. The effects of round-off error are usually such that they are more serious for a diminished step size than for an integration formula yielding suitably increased accuracy to match the decreased step size. This simply accentuates the necessity to improve solution accuracy by improving the approximation accuracy of the integration formula.

The Runge-Kutta type schemes enjoy great popularity as their application is quite straight forward and they tend to be quite stable. Their greatest appeal comes from the fact that they are one-step methods. Only the information about the function at the previous step is necessary to predict the solution at the next step. Thus they are extremely useful in initiating a solution starting with the initial value at the boundary of the range. The greatest drawback of the methods is their relative efficiency. For example, the forth order scheme requires four evaluations of the function at each step. We shall see that there are other methods that require far fewer evaluations of the function at each step and yet have a higher order.

b. Error Estimate and Step Size Control

A numerical solution to a differential equation is of little use if there is no estimate of its accuracy. However, as is clear from equation (5.1.32), the formal estimate of the truncation error is often more difficult than finding the solution. Unfortunately, the truncation error for most problems involving differential equations tends to mimic the solution. That is, should the solution be monotonically increasing, then the absolute truncation error will also increase. Even monotonically decreasing solutions will tend to have truncation errors that keep the same sign and accumulate as the solution progresses. The common effect of truncation errors on oscillatory solutions is to introduce a "phase shift" in the solution. Since the effect of truncation error tends to be systematic, there must be some method for estimating its magnitude.

Although the formal expression of the truncation error [say equation (5.1.32)] is usually rather formidable, such expressions always depend on the step size. Thus we may use the step size h itself to

estimate the magnitude of the error. We can then use this estimate and an *a priori* value of the largest acceptable error to adjust the step size. Virtually all general algorithms for the solution of differential equations contain a section for the estimate of the truncation error and the subsequent adjustment of the step size h so that predetermined tolerances can be met. Unfortunately, these methods of error estimate will rely on the variation of the step size at each step. This will generally triple the amount of time required to effect the solution. However, the increase in time spent making a single step may be offset by being able to use much larger steps resulting in an over all savings in time. The general accuracy cannot be arbitrarily increased by decreasing the step size. While this will reduce the truncation error, it will increase the effects of round-off error due to the increased amount of calculation required to cover the same range. Thus one does not want to set the *a priori* error tolerance to low or the round-off error may destroy the validity of the solution. Ideally, then, we would like our solution to proceed with rather large step sizes (i.e. values of h) when the solution is slowly varying and automatically decrease the step size when the solution begins to change rapidly. With this in mind, let us see how we may control the step size from tolerances set on the truncation error.

Given either the one step methods discussed above or the multi-step methods that follow, assume that we have determined the solution y_n at some point x_n. We are about to take the next step in the solution to x_{n+1} by an amount h and wish to estimate the truncation error in y_{n+1}. Calculate this value of the solution two ways. First, arriving at x_{n+1} by taking a single step h, then repeat the calculation taking two steps of (h/2). Let us call the first solution $y_{1,n+1}$ and the second $y_{2,n+1}$. Now the exact solution (neglecting earlier accumulated error) at x_{n+1} could be written in each case as

$$\left.\begin{aligned} y_e &= y_{1,n+1} + \alpha h^{k+1} + \cdots + \\ y_e &= y_{2,n+1} + 2\alpha(\tfrac{1}{2} h^{k+1}) + \cdots + \end{aligned}\right\} , \tag{5.1.38}$$

where k is the order of the approximation scheme. Now α can be regarded as a constant throughout the interval h since it is just the coefficient of the Taylor series fit for the (k+1)th term. Now let us define δ as a measure of the error so that

$$\delta(y_{n+1}) y_{2,n+1} - y_{1,n+1} = \alpha h^{k+1} / (1 - 2^k)\ \delta) \ . \tag{5.1.39}$$

Clearly,

$$\delta(y_{n+1}) \approx h^{k+1} , \tag{5.1.40}$$

so that the step size h can be adjusted at each step in order that the truncation error remains uniform by

$$h_{n+1} = h_n \left| \delta(y_n) / \delta(y_{n+1}) \right|^{k+1} . \tag{5.1.41}$$

Initially, one must set the tolerance at some pre-assigned level ε so that

$$\left| \delta y_0 \right| \le \varepsilon \ . \tag{5.1.42}$$

If we use this procedure to investigate the step sizes used in our test of the Runge-Kutta method, we see that we certainly chose the step size to be too large. We can verify this with the second order solution for we carried out the calculation for step sizes of h=1 and h=½. Following the prescription of equation (5.1.39) and (5.1.41) we have, that for the results specified in Table 5.2,

$$\left.\begin{aligned} \delta y_1 &= y_{2,2} - y_{1,1} = 1.6172 - 1.500 = 0.1172 \\ h_1 &= h_0 \left| \delta y_0 \big/ \delta y_1 \right| = (1)(0.1/0.1172) = 0.8532 \end{aligned}\right\} . \tag{5.1.43}$$

Here we have tacitly assumed an initial tolerance of $\delta y_0 = 0.1$. While this is arbitrary and rather large for a tolerance on a solution, it is illustrative and consistent with the spirit of the solution. We see that to maintain the accuracy of the solution within $\left| 0.1 \right|$ we should decrease the step size slightly for the initial step. The error at the end of the first step is 0.16 for h = 1, while it is only about 0.04 for h = ½. By comparing the numerical answers with the analytic answer, y_c, we see that factor of two change in the step size reduces the error by about a factor of four. Our stated tolerance of 0.1 requires only a reduction in the error of about 33% which implies a reduction of about 16% in the step size or a new step size $h_1' = 0.84h_1$. This is amazingly close to the recommended change, which was determined without knowledge of the analytic solution.

The amount of the step size adjustment at the second step is made to maintain the accuracy that exists at the end of the first step. Thus,

$$\left.\begin{aligned} \delta y_2 &= y_{2,2} - y_{1,2} = 6.5951 - 4.7500 = 1.8451 \\ h_2 &= h_1 \left| \delta y_1 \big/ \delta y_2 \right| = (1)(0.1172/1.8451) = 0.0635 \end{aligned}\right\} . \tag{5.1.44}$$

Normally these adjustments would be made cumulatively in order to maintain the initial tolerance. However, the convenient values for the step sizes were useful for the earlier comparisons of integration methods. The rapid increase of the solution after x = 1 causes the Runge-Kutta method to have an increasingly difficult time maintaining accuracy. This is abundantly clear in the drastic reduction in the step size suggested at the end of the second step. At the end of the first step, the relative errors where 9% and 2% for the h=1 and h=½ step size solutions respectively. At the end of the second step those errors, resulting from comparison with the analytic solution, had jumped to 55% and 12% respectively (see table 5.2). While a factor of two-change in the step size still produces about a factor of four change in the solution, to arrive at a relative error of 9%, we will need more like a factor of 6 change in the solution. This would suggest a change in the step size of a about a factor of three, but the recommended change is more like a factor of 16. This difference can be understood by noticing that equation (5.1.42) attempts to maintain the absolute error less than δy_n. For our problem this is about 0.11 at the end of step one. To keep the error within those tolerances, the accuracy at step two would have to be within about 1.5% of the correct answer. To get there from 55% means a reduction in the error of a *factor* of 36, which corresponds to a reduction in the step size of a factor of about 18, is close to that given by the estimate.

Thus we see that the equation (5.1.42) is designed to maintain an absolute accuracy in the solution by adjusting the step size. Should one wish to adjust the step size so as to maintain a relative or percentage accuracy, then one could adjust the step size according to

$$h_{n+1} = h_n \left| \{ [\delta(y_n) y_{n+1}] / [\delta(y_{n+1}) y_n] \right|^{(k+1)} . \tag{5.1.45}$$

While these procedures vary the step size so as to maintain constant truncation error, a significant price in the amount of computing must be paid at each step. However, the amount of extra effort need not be used only to estimate the error and thereby control it. One can solve equations (5.1.38) (neglecting terms of order greater than k) to provide an improved estimate of y_{n+1}. Specifically

$$y_e \cong y_{2,n+1} + \delta(y_{n+1}) \big/ (2^k - 1) \quad . \tag{5.1.46}$$

However, since one cannot simultaneously include this improvement directly in the error estimate, it is advisable that it be regarded as a "safety factor" and proceeds with the *error estimate* as if the improvement had not been made. While this may seem unduly conservative, in the numerical solution of differential equations conservatism is a virtue.

c. Multi-Step and Predictor-Corrector Methods

The high order one step methods achieve their accuracy by exploring the solution space in the neighborhood of the specific solution. In principle, we could use prior information about the solution to constrain our extrapolation to the next step. Since this information is the direct result of prior calculation, far greater levels of efficiency can be achieved than by methods such as Runge-Kutta that explore the solution space in the vicinity of the required solution. By using the solution at n points we could, in principle, fit an (n-1) degree polynomial to the solution at those points and use it to obtain the solution at the (n+1)st point. Such methods are called *multi-step methods*. However, one should remember the caveats at the end of chapter 3 where we pointed out that polynomial extrapolation is extremely unstable. Thus such a procedure by itself will generally not provide a suitable method for the solution of differential equations. But when combined with algorithms that compensate for the instability such schemes can provide very stable solution algorithms. Algorithms of this type are called *predictor-corrector* methods and there are numerous forms of them. So rather than attempt to cover them all, we shall say a few things about the general theory of such schemes and give some examples.

A predictor-corrector algorithm, as the name implies, consists of basically two parts. The predictor extrapolates the solution over some finite range h based on the information at prior points and is inherently unstable. The corrector allows for this local instability and makes a correction to the solution at the end of the interval also based on prior information as well as the extrapolated solution. Conceptually, the notion of a predictor is quite simple. In its simplest form, such a scheme is the one-step predictor where

$$y_{n+1} = y_n + hy'_n \, . \tag{5.1.47}$$

By using the value of the derivative at x_n the scheme will systematically under estimate the proper value required for extrapolation of any monotonically increasing solution (see figure 5.2). The error will build up cumulatively and hence it is unstable. A better strategy would be to use the value of the derivative midway between the two solution points, or alternatively to use the information from the prior two points to predict y_{n+1}. Thus a two point predictor could take the form

$$y_{n+1} = y_n + 2hy'_n \quad . \tag{5.1.48}$$

Although this is a two-point scheme, the extrapolating polynomial is still a straight line. We could have used the value of y_n directly to fit a parabola through the two points, but we didn't due to the instabilities to be associated with a higher degree polynomial extrapolation. This deliberate rejection of the some of the informational constraints in favor of increased stability is what makes predictor-corrector schemes non-trivial and effective. In the general case, we have great freedom to use the information we have regarding y_i and y'_i. If we were to include all the available information, a general predictor would have the

form

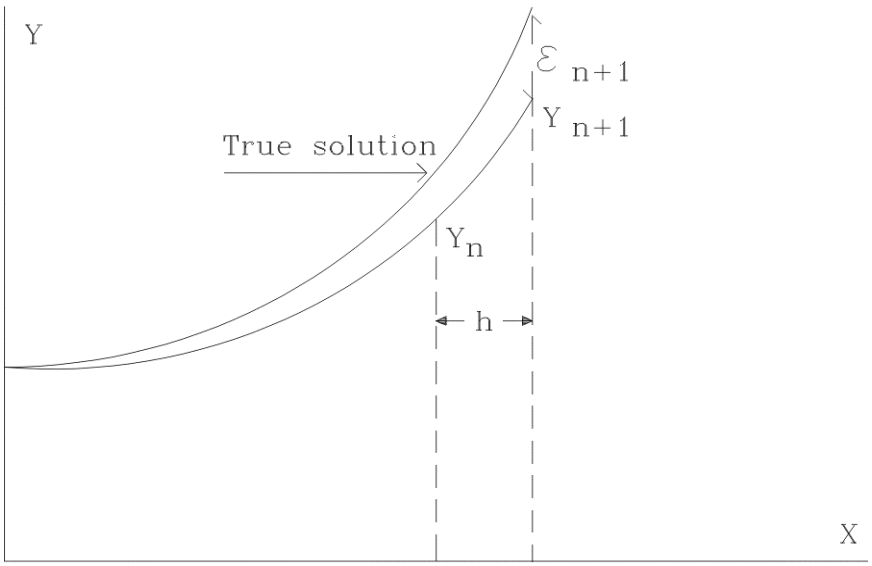
$$y_{n+1} = \sum_{i=0}^{n} a_i y_i + h\sum_{i=0}^{n} b_i y'_i + R \,, \tag{5.1.49}$$

where the a_i s and b_i s are chosen by imposing the appropriate constraints at the points x_i and R is an error term.

When we have decided on the form of the predictor, we must implement some sort of corrector scheme to reduce the truncation error introduced by the predictor. As with the predictor, let us take a simple case of a corrector as an example. Having produced a solution at x_{n+1} we can calculate the value of the derivative y'_{n+1} at x_{n+1}. This represents new information and can be used to modify the results of the prediction. For example, we could write a corrector as

$$y^{(k)}_{n+1} = y_n + \tfrac{1}{2} h[y'^{(k-1)}_{n+1} + y'_n] \,. \tag{5.1.50}$$

Therefore, if we were to write a general expression for a corrector based on the available information we



would get

Figure 5.2 shows the instability of a simple predictor scheme that systematically underestimates the solution leading to a cumulative build up of truncation error.

$$y^{(k)}_{n+1} = \sum_{i=0}^{n} \alpha_i y_i + h\sum_{i=0}^{n} \beta_i y'_i + h\beta_{n+1} y'^{(k+1)}_{n+1} \,. \tag{5.1.51}$$

Equations (5.1.50) and (5.1.51) both are written in the form of iteration formulae, but it is not at all clear that

the fixed-point for these formulae is any better representation of the solution than single iteration. So in order to minimize the computational demands of the method, correctors are generally applied only once. Let us now consider certain specific types of predictor corrector schemes that have been found to be successful.

Hamming[1] gives a number of popular predictor-corrector schemes, the best known of which is the Adams-Bashforth-Moulton Predictor-Corrector. Predictor schemes of the Adams-Bashforth type emphasize the information contained in prior values of the derivative as opposed to the function itself. This is presumably because the derivative is usually a more slowly varying function than the solution and so can be more accurately extrapolated. This philosophy is carried over to the Adams-Moulton Corrector. A classical fourth-order formula of this type is

$$\left.\begin{aligned} y^{(1)}_{n+1} &= y_n + h(55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3})/24 \quad + O(h^5) \\ y_{n+1} &= y_n + h(9y'_{n+1} + 19y'_n - 5y'_{n-1})/24 \quad + O(h^5) \end{aligned}\right\} . \tag{5.1.52}$$

Lengthy study of predictor-corrector schemes has evolved some special forms such as this one

$$\left.\begin{aligned} z_{n+1} &= (2y_{n-1} + y_{n-2})/3 \quad + h(191y'_n - 107y'_{n-1} + 109y'_{n-2} - 25y'_{n-3})/75 \\ u_{n+1} &= z_{n+1} - 707(z_n - c_n)/750 \\ c_{n+1} &= (2y_{n-1} + y_{n-2})/3 \quad + h(25u'_{n+1} + 91y'_n + 43y'_{n-1} + 9y'_{n-2})/72 \\ y_{n+1} &= c_{n+1} + 43(z_{n+1} - c_{n+1})/750 \quad + O(h^6) \end{aligned}\right\} . \tag{5.1.53}$$

where the extrapolation formula has been expressed in terms of some recursive parameters u_i and c_i. The derivative of these intermediate parameters are obtained by using the original differential equation so that

$$u' = g(x, u) . \tag{5.1.54}$$

By good chance, this formula [equation (5.1.53)] has an error term that varies as $O(h^6)$ and so is a fifth-order formula. Finally a classical predictor-corrector scheme which combines Adams-Bashforth and Milne predictors and is quite stable is parametrically (i.e. Hamming p206)

$$\left.\begin{aligned} z_{n+1} &= \tfrac{1}{2}(y_n + y_{n-1}) \quad + h(119y'_n - 99y'_{n-1} + 69y'_{n-2} - 17y'_{n-3})/48 \\ u_{n+1} &= z_{n+1} - 161(z_n - c_n)/170 \\ c_{n+1} &= \tfrac{1}{2}(y_n + y_{n-1}) \quad + h(17u'_{n+1} + 51y'_n + 3y'_{n-1} + y'_{n-2})/48 \\ y_{n+1} &= c_{n+1} + 9(z_{n+1} - c_{n+1})/170 \quad + O(h^6) \end{aligned}\right\} . \tag{5.1.55}$$

Press et al[2] are of the opinion that predictor-corrector schemes have seen their day and are made obsolete by the Bulirsch-Stoer method which they discuss at some length[3]. They quite properly point out that the predictor-corrector schemes are somewhat inflexible when it comes to varying the step size. The step size can be reduced by interpolating the necessary missing information from earlier steps and it can be expanded in integral multiples by skipping earlier points and taking the required information from even earlier in the solution. However, the Bulirsch-Stoer method, as described by Press et. al. utilizes a predictor scheme with some special properties. It may be parameterized as

$$\left.\begin{aligned} z_0 &= y(x_0) \\ z_1 &= z_0 + hz'_0 \\ z_{k+1} &= z_{k-1} + hz'_k \quad k = 1,2,3,\cdots,n-1 \\ y_n^{(1)} &= \tfrac{1}{2}(z_n + z_{n-1} + hz'_n) \quad + O(h^5) \\ z' &= g(z,x) \end{aligned}\right\} . \tag{5.1.56}$$

It is an odd characteristic of the third of equations (5.1.56) that the error term only contains even powers of the step size. Thus, we may use the same trick that was used in equation (5.1.46) of utilizing the information generated in estimating the error term to improve the approximation order. But since only even powers of h appear in the error term, this single step will gain us two powers of h resulting in a predictor of order seven.

$$y_{nh} = \{4y_n^{(1)}(x+nh) - y_{n/2}^{(1)}[x+(n/2)(2h)]\}/3 \quad + \quad O(h^7) . \tag{5.1.57}$$

This yields a predictor that requires something on the order of 1½ evaluations of the function per step compared to four for a Runge-Kutta formula of inferior order.

Now we come to the aspect of the Bulirsch-Stoer method that begins to differentiate it from classical predictor-correctors. A predictor that operates over some *finite* interval can use a successively increasing number of steps in order to make its prediction. Presumably the prediction will get better and better as the step size decreases so that the number of steps to make the one prediction increases. Of course practical aspects of the problem such as roundoff error and finite computing resources prevent us from using arbitrarily small step sizes, but we can approximate what would happen in an ideal world without round-off error and utilizing unlimited computers. Simply consider the prediction at the end of the finite interval H where

$$H = \alpha h . \tag{5.1.58}$$

Thus $y_\alpha(x+H)$ can be taken to be a function of the step size h so that,

$$y_\alpha(x+H) = y(x+\alpha h) = f(h) . \tag{5.1.59}$$

Now we can phrase our problem to estimate the value of that function in the limit

$$\lim_{\substack{h\to 0 \\ \alpha\to\infty}} f(h) = Y_\infty(x+H) . \tag{5.1.60}$$

We can accomplish this by carrying out the calculation for successively smaller and smaller values of h and, on the basis of these values, extrapolating the result to h=0. In spite of the admonitions raised in chapter 3 regarding extrapolation, the range here is small. But to produce a truly powerful numerical integration algorithm, Bulirsch and Stoer carry out the extrapolation using rational functions in the manner described in section 3.2 [equation (3.2.65)]. The superiority of rational functions to polynomials in representing most analytic functions means that the step size can be quite large indeed and the conventional meaning of the 'order' of the approximation is irrelevant in describing the accuracy of the method.

In any case, remember that accuracy and order are not synonymous! Should the solution be described by a slowly varying function and the numerical integration scheme operate by fitting high order polynomials to prior information for the purposes of extrapolation, the high-order formula can give very inaccurate results. This simply says that the integration scheme can be unstable even for well behaved solutions.

Press et. al.[4] suggest that all one needs to solve ordinary differential equations is either a Runge-Kutta or Bulirsch-Stoer method and it would seem that for most problems that may well be the case. However, there are a large number of commercial differential equation solving algorithms and the majority of them utilize predictor-corrector schemes. These schemes are generally very fast and the more sophisticated ones carry out very involved error checking algorithms. They are generally quite stable and can involve a very high order when required. In any event, the user should know how they work and be wary of the results. It is far too easy to simply take the results of such programs at face value without ever questioning the accuracy of the results. Certainly one should always ask the question "Are these results reasonable?" at the end of a numerical integration. If one is genuinely skeptical, it is not a bad idea to take the final value of the calculation as an initial value and integrate back over the range. Should one recover the original initial value within the acceptable tolerances, one can be reasonably confident that the results are accurate. If not, the difference between the beginning initial value and what is calculated by the reverse integration over the range can be used to place limits on the accuracy of the initial integration.

d. Systems of Differential Equations and Boundary Value Problems

All the methods we have developed for the solution of single first order differential equations may be applied to the case where we have a coupled system of differential equations. We saw earlier that such systems arose whenever we dealt with ordinary differential equations of order greater than one. However, there are many scientific problems which are intrinsically described by coupled systems of differential equations and so we should say something about their solution. The simplest way to see the applicability of the single equation algorithms to a system of differential equations is to write a system like

$$\left.\begin{aligned} y'_1 &= g_1(x, y_1, y_2, \cdots y_n) \\ y'_2 &= g_2(x, y_1, y_2, \cdots y_n) \\ \vdots &\qquad\qquad \vdots \\ y'_n &= g_n(x, y_1, y_2, \cdots y_n) \end{aligned}\right\} , \tag{5.1.61}$$

as a vector where each element represents one of the dependent variables or unknowns of the system. Then the system becomes

$$\vec{y}' = \vec{g}(x, \vec{y}) \quad , \tag{5.1.62}$$

which looks just like equation (5.1.3) so that everything applicable to that equation will apply to the system of equations. Of course some care must be taken with the terminology. For example, equation (5.1.4) would have to be understood as standing for an entire system of equations involving far more complicated integrals, but in principle, the ideas carry over. Some care must also be extended to the error analysis in that the error

term is also a vector $\vec{R}$ (x). In general, one should worry about the magnitude of the error vector, but in practice, it is usually the largest element that is taken as characterizing the accuracy of the solution.

To generate a numerical integration method for a specific algorithm, one simply applies it to each of the equations that make up the system. By way of a specific example, let's consider a forth order Runge-Kutta algorithm as given by equation (5.1.33) and apply it to a system of two equations. We get

$$\left.\begin{aligned}
y_{1,n+1} &= y_{1,n} + (t_0 + 2t_1 + 2t_2 + t_3)/6 \\
y_{2,n+1} &= y_{2,n} + (u_0 + 2u_1 + 2u_2 + u_3)/6 \\
t_0 &= hg_1(x_n, y_{1,n}, y_{2,n}) \\
t_1 &= hg_1[(x_n + \tfrac{1}{2}h), (y_{1,n} + \tfrac{1}{2}t_0), (y_{2,n} + \tfrac{1}{2}u_0)] \\
t_2 &= hg_1[(x_n + \tfrac{1}{2}h), (y_{1,n} + \tfrac{1}{2}t_1), (y_{2,n} + \tfrac{1}{2}u_1)] \\
t_3 &= hg_1[(x_n + h), (y_{1,n} + t_t), (y_{2,n} + u_2)] \\
u_0 &= hg_2(x_n, y_{1,n}, y_{2,n}) \\
u_1 &= hg_2[(x_n + \tfrac{1}{2}h), (y_{1,n} + \tfrac{1}{2}t_0), (y_{2,n} + \tfrac{1}{2}u_0)] \\
u_2 &= hg_2[(x_n + \tfrac{1}{2}h), (y_{1,n} + \tfrac{1}{2}t_1), (y_{2,n} + \tfrac{1}{2}u_1)] \\
u_1 &= hg_2[(x_n + h), (y_{1,n} + t_t), (y_{2,n} + u_2)]
\end{aligned}\right\}. \qquad (5.1.63)$$

We can generalize equation (5.1.63) to an arbitrary system of equations by writing it in vector form as

$$\vec{y}_{n+1} = \vec{A}(\vec{y}_n). \qquad (5.1.64)$$

The vector $\vec{A}(\vec{y}_n)$ consists of elements which are functions of dependent variables $y_{i,n}$ and x_n, but which all have the same general form varying only with $g_i(x, \vec{y})$. Since an nth order differential equation can always be reduced to a system of n first order differential equations, an expression of the form of equation (5.1.63) could be used to solve a second order differential equation.

The existence of coupled systems of differential equations admits the interesting possibility that the constants of integration required to uniquely specify a solution are not all given at the same location. Thus we do not have a full compliment of $y_{i,0}$'s with which to begin the integration. Such problems are called *boundary value problems*. A comprehensive discussion of boundary value problems is well beyond the scope of this book, but we will examine the simpler problem of *linear two point boundary value problems*. This subclass of boundary value problems is quite common in science and extremely well studied. It consists of a system of linear differential equations (i.e. differential equations of the first degree only) where part of the integration constants are specified at one location x_0 and the remainder are specified at some other value of the independent variable x_n. These points are known as the boundaries of the problem and we seek a solution to the problem within these boundaries. Clearly the solution can be extended beyond the boundaries as the solution at the boundaries can serve as initial values for a standard numerical integration.

The general approach to such problems is to take advantage of the linearity of the equations, which

guarantees that any solution to the system can be expressed as a linear combination of a set of basis solutions. A set of basis solutions is simply a set of solutions, which are linearly independent. Let us consider a set of m linear first order differential equations where k values of the dependent variables are specified at x_0 and (m-k) values corresponding to the remaining dependent variables are specified at x_n. We could solve (m-k) initial value problems starting at x_0 and specifying (m-k) independent, sets of missing initial values so that the initial value problems are uniquely determined. Let us denote the missing set of initial values at x_0 by $\vec{y}^{(0)}(x_0)$ which we know can be determined from initial sets of linearly independent trial initial values ${}^{j}\vec{y}^{(t)}(x_0)$ by

$$\vec{y}^{(0)}(x_0) = \mathbf{A}\mathbf{y}^{(t)}(x_0), \tag{5.1.65}$$

The columns of $\mathbf{y}^{(t)}(x_0)$ are just the individual vectors ${}^{j}\vec{y}^{(t)}(x_0)$. Clearly the matrix **A** will have to be diagonal to always produce $\vec{y}^{(0)}(x_0)$. Since the trial initial values are arbitrary, we will choose the elements of the (m-k) sets to be

$${}^{j}y_i(x_0) = \delta_{ij}, \tag{5.1.66}$$

so that the missing initial values will be

$$\vec{y}^{(0)}(x_0)\,\mathbf{1} = \mathbf{1A} = \mathbf{A}\ . \tag{5.1.67}$$

Integrating across the interval with these initial values will yield (m-k) solution ${}^{j}\vec{y}^{(t)}(x_n)$ at the other boundary. Since the equations are linear each trial solution will be related to the known boundary values ${}^{j}\vec{y}^{(t)}(x_n)$ by

$$\vec{y}^{(0)}(x_n) = \mathbf{A}[{}^{j}\vec{y}^{(t)}(x_n)]\ , \tag{5.1.68}$$

so that for the complete set of trial solutions we may write

$$\vec{y}^{(0)}(x_n)\,\mathbf{1} = \mathbf{A}\mathbf{y}^{(t)}(x_n)\ , \tag{5.1.69}$$

where by analogy to equation (5.1.65), the column vectors of $\mathbf{y}^{(t)}(x_n)$ are ${}^{j}\vec{y}^{(t)}(x_n)$. We may solve these equations for the unknown transformation matrix **A** so that the missing initial values are

$$\vec{y}^{(0)}(x_n)\,\mathbf{1} = \mathbf{A} = \mathbf{y}^{-1}\ \vec{y}^{(0)}(x_n). \tag{5.1.70}$$

If one employs a one step method such as Runge-Kutta, it is possible to collapse this entire operation to the point where one can represent the complete boundary conditions at one boundary in terms of the values at the other boundary $\vec{y}_n$ a system of linear algebraic equations such as

$$\vec{y}(x_0) = \mathbf{B}\vec{y}(x_n). \tag{5.1.71}$$

The matrix **B** will depend only on the details of the integration scheme and the functional form of the equations themselves, not on the boundary values. Therefore it may be calculated for any set of boundary values and used repeatedly for problems differing only in the values at the boundary (see Day and Collins[5]).

To demonstrate methods of solution for systems of differential equations or boundary value problems, we shall need more than the first order equation (5.1.10) that we used for earlier examples. However, that equation was quite illustrative as it had a rapidly increasing solution that emphasized the shortcomings of the various numerical methods. Thus we shall keep the solution, but change the equation. Simply differentiate equation (5.1.10) so that

$$Y'' = 2(1 + 2x^2)e^{x^2} = 2(1 + x^2)y \,. \tag{5.1.72}$$

Let us keep the same initial condition given by equation (5.1.11) and add a condition of the derivative at x = 1 so that

$$\left.\begin{aligned} y(0) &= 1 \\ y'(1) &= 2e = 5.43656 \end{aligned}\right\} \,. \tag{5.1.73}$$

This insures that the closed form solution is the same as equation (5.1.12) so that we will be able to compare the results of solving this problem with earlier methods. We should not expect the solution to be as accurate for we have made the problem more difficult by increasing the order of the differential equation in addition to separating the location of the constants of integration. This is no longer an initial value problem since the solution value is given at x = 0, while the other constraint on the derivative is specified at x = 1. This is typical of the classical two-point boundary value problem.

We may also use this example to indicate the method for solving higher order differential equations given at the start of this chapter by equations (5.1.1) and (5.1.2). With those equations in mind, let us replace equation (5.1.72) by system of first order equations

$$\left.\begin{aligned} y'_1(x) &= y_2(x) \\ y'_2(x) &= 2(1 + 2x^2)y_1(x) \end{aligned}\right\} \,, \tag{5.1.74}$$

which we can write in vector form as

$$\vec{y}' = \mathbf{A}(x)\vec{y} \,, \tag{5.1.75}$$

where

$$\mathbf{A}(x) = \begin{pmatrix} 0 & 1 \\ 2(1 + x^2) & 0 \end{pmatrix} \,. \tag{5.1.76}$$

The components of the solution vector $\vec{y}$ are just the solution we seek (i.e.) and its derivative. However, the form of equation (5.1.75) emphasizes its linear form and were it a scalar equation, we should know how to proceed.

For purposes of illustration, let us apply the fourth order Runge-Kutta scheme given by equation (5.1.63). Here we can take specific advantage of the linear nature of our problem and the fact that the dependence on the independent variable factors out of the right hand side. To illustrate the utility of this fact, let

$$g(x, y) = [f(x)]y \,, \tag{5.1.77}$$

in equation (5.1.63).

Then we can write the fourth order Runge-Kutta parameters as

$$\left.\begin{aligned} t_0 &= hf_0 y_n \\ t_1 &= hf_1(y_n + \tfrac{1}{2}t_0) = hf_1(y_n + \tfrac{1}{2}hf_0 y_n = (hf_1 + \tfrac{1}{2}h^2 f_1 f_0)y_n \\ t_2 &= hf_1(y_n + \tfrac{1}{2}t_1) = (hf_1 + \tfrac{1}{2}h^2 f_1^2 + \tfrac{1}{4}h^3 f_1^2 f_0)y_n \\ t_3 &= hf_2(y_n + t_2) = (hf_2 + h^2 f_2 f_1 + \tfrac{1}{2}h^3 f_1^2 f_2 + \tfrac{1}{4}h^4 f_2 f_1^2 f_0)y_n \end{aligned}\right\} . \qquad (5.1.78)$$

where

$$\left.\begin{aligned} f_0 &= f(x_n) \\ f_1 &= f(x_n + \tfrac{1}{2}h) \\ f_2 &= f(x_n + h) \end{aligned}\right\} , \qquad (5.1.79)$$

so that the formula becomes

$$\begin{aligned} y_{n+1} &= y_n + (t_0 + 2t_1 2t_2 + t_3) \\ &= \left[1 + \frac{h}{6}(f_0 + 4f_1 + f_2) + \frac{h^2}{6}(f_1 f_0 + f_1^2 + f_2 f_1) + \frac{h^3}{12}(f_1^2 f_0 + f_2 f_1^2) + \frac{h^4}{24} f_2 f_1^2 f_0)\right] y_n \end{aligned} . \qquad (5.1.80)$$

Here we see that the linearity of the differential equation allows the solution at step n to be factored out of the formula so that the solution at step n appears explicitly in the formula. Indeed, equation (5.1.80) represents a power series in h for the solution at step (n+1) in terms of the solution at step n. Since we have been careful about the order in which the functions f_i multiplied each other, we may apply equation (5.1.80) directly to equation (5.1.75) and obtain a similar formula for systems of linear first order differential equations that has the form

$$\vec{y}_{n+1} = \left[1 + \frac{h}{6}(\mathbf{A}_0 + 4\mathbf{A}_1 + \mathbf{A}_2) + \frac{h^2}{6}(\mathbf{A}_0\mathbf{A}_1 + 4\mathbf{A}_1^2 + \mathbf{A}_2\mathbf{A}_1) + \frac{h^3}{12}(\mathbf{A}_1^2\mathbf{A}_0 + \mathbf{A}_2\mathbf{A}_1^2) + \frac{h^4}{24}\mathbf{A}_2\mathbf{A}_1^2\mathbf{A}_0\right]\vec{y}_n . \qquad (5.1.81)$$

Here the meaning of $\mathbf{A}_i$ is the same as f_i in that the subscript indicates the value of the independent variable x for which the matrix is to be evaluated. If we take h = 1, the matrices for our specific problem become

$$\left.\begin{aligned} \mathbf{A}_0 &= \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix} \\ \mathbf{A}_1 &= \begin{pmatrix} 0 & 1 \\ 3 & 0 \end{pmatrix} \\ \mathbf{A}_2 &= \begin{pmatrix} 0 & 1 \\ 4 & 0 \end{pmatrix} \end{aligned}\right\} . \qquad (5.1.82)$$

Keeping in mind that the order of matrix multiplication is important, the products appearing in the second order term are

$$\left.\begin{aligned} \mathbf{A}_1\mathbf{A}_0 &= \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \\ \mathbf{A}_1^2 &= \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \\ \mathbf{A}_2\mathbf{A}_1 &= \begin{pmatrix} 3 & 0 \\ 0 & 6 \end{pmatrix} \end{aligned}\right\} . \tag{5.1.83}$$

The two products appearing in the third order term can be easily generated from equations (5.1.82) and (5.1.83) and are

$$\left.\begin{aligned} \mathbf{A}_1^2\mathbf{A}_0 &= \begin{pmatrix} 0 & 3 \\ 9 & 0 \end{pmatrix} \\ \mathbf{A}_2\mathbf{A}_1^2 &= \begin{pmatrix} 0 & 3 \\ 18 & 0 \end{pmatrix} \end{aligned}\right\} . \tag{5.1.84}$$

Finally the single matrix of the first order term can be obtain by successive multiplication using equations(5.1.82) and (5.1.84) yielding

$$\left.\mathbf{A}_2\mathbf{A}_1^2\mathbf{A}_0 = \begin{pmatrix} 9 & 0 \\ 0 & 18 \end{pmatrix} \right\} . \tag{5.1.85}$$

Like equation (5.1.80), we can regard equation (5.1.81) as a series solution in h that yields a system of linear equations for the solution at step n+1 in terms of the solution at step n. It is worth noting that the coefficients of the various terms of order h^k are similar to those developed for equal interval quadrature formulae in chapter 4. For example the lead term being the unit matrix generates the coefficients of the trapezoid rule while the h(+1, +4, +1)/6 coefficients of the second term are the familiar progression characteristic of Simpson's rule. The higher order terms in the formula are less recognizable since they depend on the parameters chosen in the under-determined Runge-Kutta formula.

If we define a matrix $\mathbf{P}(h^k)$ so that

$$\vec{y}_{n+1} = \mathbf{P}(h^k) \equiv {}^k\mathbf{P}\,\vec{y}_n , \tag{5.1.86}$$

the series nature of equation (5.1.81) can be explicitly represented in terms of the various values of ${}^k\mathbf{P}$.

For our problem they are:

$$\left.\begin{aligned}
{}^0\mathbf{P} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
{}^1\mathbf{P} &= \begin{pmatrix} 1 & 1 \\ \frac{11}{6} & 0 \end{pmatrix} \\
{}^2\mathbf{P} &= \begin{pmatrix} \frac{7}{3} & 1 \\ \frac{11}{6} & 0 \end{pmatrix} \\
{}^3\mathbf{P} &= \begin{pmatrix} \frac{7}{3} & \frac{3}{2} \\ \frac{49}{12} & 3 \end{pmatrix} \\
{}^4\mathbf{P} &= \begin{pmatrix} \frac{65}{24} & \frac{3}{2} \\ \frac{49}{12} & \frac{15}{4} \end{pmatrix}
\end{aligned}\right\} . \qquad (5.1.87)$$

The boundary value problem now is reduced to solving the linear system of equations specified by equation (5.1.86) where the known values at the respective boundaries are specified. Using the values given in equation (5.1.73) the linear equations for the missing boundary values become

$$\left.\begin{aligned}
1 &= {}^k\mathbf{P}_{11}\, y_1(0) + {}^k\mathbf{P}_{12}(5.43656) \\
y_2(0) &= {}^k\mathbf{P}_{21}\, y_1(1) + {}^k\mathbf{P}_{22}(5.43656)
\end{aligned}\right\} . \qquad (5.1.88)$$

The first of these yields the missing solution value at x = 0 [i.e. $y_2(0)$]. With that value the remaining value can be obtained from the second equation. The results of these solutions including additional terms of order h^k are given in table 5.3. We have taken h to be unity, which is unreasonably large, but it serves to demonstrate the relative accuracy of including higher order terms and simplifies the arithmetic. The results for the missing values $y_2(0)$ and $y_1(1)$ (i.e. the center two rows) converge slowly, and not uniformly, toward their analytic values given in the column labeled $k = \infty$.

Had we chosen the step size h to be smaller so that a number of steps were required to cross the interval, then each step would have produced a matrix ${}^k_i\mathbf{P}$ and the solution at each step would have been related to the solution at the following step by equation (5.1.86). Repeated application of that equation would yield the solution at one boundary in terms of the solution at the other so that

$$\vec{y}_n = ({}^{k}_{n-1}\mathbf{P}\ {}^{k}_{n-2}\mathbf{P}\ {}^{k}_{n-3}\mathbf{P}\cdots{}^{0}_{0}\mathbf{P})\vec{y}_0 = {}^{k}\mathbf{Q}\,\vec{y}_0 \quad . \tag{5.1.89}$$

Table 5.3

Solutions of a Sample Boundary Value Problem for Various Orders of Approximation

\ K	0	1	2	3	4	∞
$y_1(0)$	1.0	1.0	1.0	1.0	1.0	1.0
$y_2(0)$	5.437	3.60	1.200	0.4510	0.3609	0.0
$y_1(1)$	1.0	4.60	3.53	3.01	3.25	2.71828
$y_2(1)$	5.437	5.437	5.437	5.437	5.437	2e

Thus one arrives at a similar set of linear equations to those implied by equation (5.1.86) and explicitly given in equation (5.1.88) relating the solution at one boundary in terms of the solution at the other boundary. These can be solved for the missing boundary values in the same manner as our example. Clearly the decrease in the step size will improve the accuracy as dramatically as increasing the order k of the approximation formula. Indeed the step size can be variable at each step allowing for the use of the error correcting procedures described in section 5.1b.

Table 5.4

Solutions of a Sample Boundary Value Problem

\ K	0	1	2	3	4	∞
$y_1(0)$	1.0	1.0	1.0	1.0	1.0	1.0
$y_2(0)$	0.0	0.0	0.0	0.0	0.0	0.0
$y_1(1)$	1.0	1.0	2.33	2.33	2.708	2.718
$y_2(1)$	0.0	1.83	1.83	4.08	4.08	5.437

Any set of boundary values could have been used with equations (5.1.81) to yield the solution elsewhere. Thus, we could treat our sample problem as an initial value problem for comparison. If we take the analytic values for $y_1(0)$ and $y_2(0)$ and solve the resulting linear equations, we get the results given in Table 5.4. Here the final solution is more accurate and exhibits a convergence sequence more like we would expect from Runge-Kutta. Namely, the solution systematically lies below the rapidly increasing analytic solution. For the boundary value problem, the reverse was true and the final result less accurate. This is not an uncommon result for two-point boundary value problems since the error of the approximation scheme is directly reflected in the determination of the missing boundary values. In an initial value problem, there is assumed to be no error in the initial values.

This simple example is not meant to provide a definitive discussion of even the restricted subset of linear two-point boundary value problems, but simply to indicate a way to proceed with their solution. Anyone wishing to pursue the subject of two-point boundary value problems further should begin with the venerable text by Fox[6].

e. Partial Differential Equations

The subject of partial differential equations has a literature at least as large as that for ordinary differential equations. It is beyond the scope of this book to provide a discussion of partial differential equations even at the level chosen for ordinary differential equations. Indeed, many introductory books on numerical analysis do not treat them at all. Thus we will only sketch a general approach to problems involving such equations.

Partial differential equations form the basis for so many problems in science, that to limit the choice of examples. Most of the fundamental laws of physical science are written in terms of partial differential equations. Thus one finds them present in computer modeling from the hydrodynamic calculations needed for airplane design, weather forecasting, and the flow of fluids in the human body to the dynamical interactions of the elements that make up a model economy.

A partial derivative simply refers to the rate of change of a function of many variables, with respect to just one of those variables. In terms of the familiar limiting process for defining differentials we would write

$$\frac{\partial F(x_1, x_2, \cdots, x_n)}{\partial x_j} = \lim_{\Delta x_j \to 0}\left[\frac{F(x_1, x_2, \cdots, x_j, \cdots x_n) - F(x_1, x_2, \cdots, x_j + \Delta x_j, \cdots x_n)}{\Delta x_j}\right]. \quad (5.1.90)$$

Partial differential equations usually relate derivatives of some function with respect to one variable to derivatives of the same function with respect to another. The notion of order and degree are the same as with ordinary differential equations.

Although a partial differential equation may be expressed in multiple dimensions, the smallest number for illustration is two, one of which may be time. Many of these equations, which describe so many aspects of the physical world, have the form

$$a(x,y)\frac{\partial^2 z(x,y)}{\partial x^2} + 2b(x,y)\frac{\partial^2 z(x,y)}{\partial x \partial y} + c(x,y)\frac{\partial^2 z(x,y)}{\partial y^2} = F\left[x, y, z\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right]. \quad (5.1.91)$$

and as such can be classified into three distinct groups by the discriminate so that

$$\left.\begin{array}{ll} [b^2(x,y) - a(x,y)c(x,y)] < 0 & \text{Elliptic} \\ [b^2(x,y) - a(x,y)c(x,y)] = 0 & \text{Parabolic} \\ [b^2(x,y) - a(x,y)c(x,y)] > 0 & \text{Hyperbolic} \end{array}\right\}. \quad (5.1.92)$$

Should the equation of interest fall into one of these three categories, one should search for solution algorithms designed to be effective for that class. Some methods that will be effective at solving equations of one class will fail miserably for another.

While there are many different techniques for dealing with partial differential equations, the most wide-spread method is to replace the differential operator by a finite difference operator thereby turning the differential equation into a finite difference equation in at least two variables. Just as a numerical integration scheme finds the solution to a differential equation at discrete points x_i along the real line, so a two dimensional integration scheme will specify the solution at a set of discrete points x_i, y_j. These points can be viewed as the intersections on a grid. Thus the solution in the x-y space is represented by the solution on a finite grid. The location of the grid points will be specified by the finite difference operators for the two coordinates. Unlike problems involving ordinary differential equations, the initial values for partial differential equations are not simply constants. Specifying the partial derivative of a function at some particular value of one of the independent variables still allows it to be a function of the remaining independent variables of the problem. Thus the functional behavior of the solution is often specified at some boundary and the solution proceeds from there. Usually the finite difference scheme will take advantage of any symmetry that may result for the choice of the boundary. For example, as was pointed out in section 1.3 there are thirteen orthogonal coordinate systems in which Laplace's equation is separable. Should the boundaries of a problem match one of those coordinate systems, then the finite difference scheme would be totally separable in the independent variables greatly simplifying the numerical solution. In general, one picks a coordinate system that will match the local boundaries and that will determine the geometry of the grid. The solution can then proceed from the initial values at a particular boundary and move across the grid until the entire space has been covered. Of course the solution should be independent of the path taken in filling the grid and that can be used to estimate the accuracy of the finite difference scheme that is being used. The details of setting up various types of schemes are beyond the scope of this book and could serve as the subject of a book by themselves. For a further introduction to the solution of partial differential equations the reader is referred to Sokolnikoff and Redheffer[7] and for the numerical implementation of some methods the student should consult Press et.al.[8]. Let us now turn to the numerical solution of integral equations.

5.2 The Numerical Solution of Integral Equations

For reasons that I have never fully understood, the mathematical literature is crowded with books, articles, and papers on the subject of differential equations. Most universities have several courses of study in the subject, but little attention is paid to the subject of integral equations. The differential operator is linear and so is the integral operator. Indeed, one is just the inverse of the other. Equations can be written where the dependent variable appears under an integral as well as alone. Such equations are the analogue of the differential equations and are called *integral equations*. It is often possible to turn a differential equation into an integral equation which may make the problem easier to numerically solve. Indeed many physical phenomena lend themselves to description by integral equations. So one would think that they might form as large an area for analysis are do the differential equations. Such is not the case. Indeed, we will not be able to devote as much time to the discussion of these interesting equations as we should, but we shall spend enough time so that the student is at least familiar with some of their basic properties. Of necessity, we will restrict our discussion to those integral equations where the unknown appears linearly. Such linear equations are more tractable and yet describe much that is of interest in science.

a. Types of Linear Integral Equations

We will follow the standard classification scheme for integral equations which, while not exhaustive, does include most of the common types. There are basically two main classes known as Fredholm and Volterra after the mathematicians who first studied them in detail. Fredholm equations involve definite integrals, while Volterra equations have the independent variable as one of the limits. Each of these categories can be further subdivided as to whether or not the dependent variable appears outside the integral sign as well as under it. Thus the two types of Fredholm equations for the unknown φ are

$$\left.\begin{aligned} F(x) &= \int_a^b K(x,t)\phi(t)\,dt \qquad \text{Fredholm} \quad \text{Type I} \\ \phi(x) &= F(x) + \lambda\int_a^b K(x,t)\phi(t)\,dt \quad \text{Fredholm} \quad \text{Type II} \end{aligned}\right\}, \tag{5.2.1}$$

while the corresponding two types of Volterra equations for φ take the form

$$\left.\begin{aligned} F(x) &= \int_a^x K(x,t)\phi(t)\,dt \qquad \text{Volterra} \quad \text{Type I} \\ \phi(x) &= F(x) + \lambda\int_a^x K(x,t)\phi(t)\,dt \quad \text{Volterra} \quad \text{Type II} \end{aligned}\right\}. \tag{5.2.2}$$

The parameter K(x,t) appearing in the integrand is known as the *kernel* of the integral equation. Its form is crucial in determining the nature of the solution. Certainly one can have homogeneous or inhomogeneous integral equations depending on whether or not F(x) is zero. Of the two classes, the Fredholm are generally easier to solve.

b. The Numerical Solution of Fredholm Equations

Integral equations are often easier to solve than a corresponding differential equation. One of the reasons is that the truncation errors of the solution tend to be averaged out by the process of quadrature while they tend to accumulate during the process of numerical integration employed in the solution of differential equations. The most straight-forward approach is to simply replace the integral with a quadrature sum. In the case of Fredholm equations of type one, this results in a functional equation for the unknown φ(x) at a discrete set of points t_j used by the quadrature scheme. Specifically

$$F(x) = \sum_{j=0}^{n} K(x,t_j)\varphi(t_j)W_j + R_n(x) \quad . \tag{5.2.3}$$

Since equation (5.2.3) must hold for all values of x, it must hold for values of x equal to those chosen for the quadrature points so that

$$x_j = t_j \qquad j = 0, 1, 2, \cdots, n \quad . \tag{5.2.4}$$

By picking those particular points we can generate a linear system of equations from the functional equation (5.2.3) and, neglecting the quadrature error term, they are

$$F(x_i) = \sum_{j=0}^{n} K(x_i,t_j)\varphi(t_j)W_j = \sum_{j=0}^{n} A_{ij}\varphi(x_j) \qquad i = 0, 1, 2, \cdots, n \ , \tag{5.2.5}$$

which can be solved by any of the methods discussed in Chapter 2 yielding

$$\phi(x_j) = \sum_{k=0}^{n} A_{jk}^{-1} F(x_k) \qquad j = 0,1,2,\cdots,n\,. \tag{5.2.6}$$

The solution will be obtained at the quadrature points x_j so that one might wish to be careful in the selection of a quadrature scheme and pick one that contained the points of interest. However, one can use the solution set $\varphi(x_j)$ to interpolate for missing points and maintain the same degree of precession that generated the solution set. For Fredholm equations of type 2, one can perform the same trick of replacing the integral with a quadrature scheme. Thus

$$\phi(x) = F(x) + \lambda \sum_{j=0}^{n} K(x,t_j)\phi(t_j)W_j \quad + \quad R_n(x)\,. \tag{5.2.7}$$

Here we must be a little careful as the unknown $\phi(x)$ appears outside the integral. Thus equation (5.2.7) is a functional equation for $\phi(x)$ itself. However, by evaluating this functional equation as we did for Fredholm equations of type 1 we get

$$\phi(x_i) = F(x_i) + \lambda \sum_{j=0}^{n} K(x_i,t_j)\phi(t_j)W_j \qquad , \tag{5.2.8}$$

which, after a little algebra, can be put in the standard form for linear equations

$$F(x_i) = \sum_{j=0}^{n} [\delta_{ij} - \lambda K(x_i,t_j)W_j]\phi(t_j) = \sum_{j=0}^{n} B_{ij}\phi(x_j) \qquad i = 0,1,2,\cdots,n \quad , \tag{5.2.9}$$

that have a solution

$$\phi(x_j) = \sum_{k=0}^{n} B_{jk}^{-1} F(x_k) \qquad j = 0,1,2,\cdots,n \quad . \tag{5.2.10}$$

Here the solution set $\phi(x_j)$ can be substituted into equation (5.2.7) to directly obtain an interpolation formula for $\phi(x)$ which will have the same degree of precision as the quadrature scheme and is valid for all values of x. Such equations can be solved efficiently by using the appropriate Gaussian quadrature scheme that is required by the limits. In addition, the form of the kernel K(x,t) may influence the choice of the quadrature scheme and it is useful to include as much of the behavior of the kernel in the quadrature weight functions as possible. We could also choose to break the interval $a \rightarrow b$ in several pieces depending on the nature of the kernel and what can be guessed about the solution itself. The subsequent quadrature schemes for the sub-intervals will not then depend on the continuity of polynomials from one sub-interval to another and may allow for more accurate approximation in the sub-interval.

For a specific example of the solution to Fredholm equations, let us consider a simple equation of the second type namely

$$y(x) = 1 + x\int_0^1 ty\, dt\,. \tag{5.2.11}$$

Comparing this to equation (5.2.7), we see that F(x) = 1, and that the kernel is separable which leads us immediately to an analytic solution. Since the integral is a definite integral, it may be regarded as some constant α and the solution will be linear of the form

$$y(x) = 1 + \alpha x\int_0^1 t(1+\alpha t)\, dt = 1 + x(\tfrac{1}{2} + \tfrac{\alpha}{3}) \quad . \tag{5.2.12}$$

This leads to a value for α of

$$\alpha = 3/4 \ . \tag{5.2.13}$$

However, had the equation required a numerical solution, then we would have proceeded by replacing the integral by a quadrature sum and evaluating the resulting functional equation at the points of the quadrature. Knowing that the solution is linear, let us choose the quadrature to be Simpson's rule which has a degree of precision high enough to provide an exact answer. The linear equations for the solution become

$$\left.\begin{aligned} y(0) &= 1 + (0)[(0)y(0) + 4(\tfrac{1}{2})y(\tfrac{1}{2}) + y(1)]/6 = 1 \\ y(\tfrac{1}{2}) &= 1 + (\tfrac{1}{2})[(0)y(0) + 4(\tfrac{1}{2})y(\tfrac{1}{2}) + y(1)]/6 = 1 + y(\tfrac{1}{2})/6 + y(1)/12 \\ y(1) &= 1 + (1)[(0)y(0) + 4(\tfrac{1}{2})y(\tfrac{1}{2}) + y(1)]/6 = 1 + y(\tfrac{1}{2})/3 + y(1)/6 \end{aligned}\right\} , \tag{5.2.14}$$

which have the immediate solution

$$\left.\begin{aligned} y(0) &= 1 \\ y(\tfrac{1}{2}) &= \frac{11}{8} \\ y(1) &= \frac{7}{4} \end{aligned}\right\} . \tag{5.2.15}$$

Clearly this solution is in exact agreement with the analytic form corresponding to α=3/4,

$$y(x) = 1 + 3x/4 \ . \tag{5.2.16}$$

While there are variations on a theme for the solution of these type of equations, the basic approach is nicely illustrated by this approach. Now let us turn to the generally more formidable Volterra equations.

c. The Numerical Solution of Volterra Equations

We may approach Volterra equations in much the same way as we did Fredholm equations, but there is the problem that the upper limit of the integral is the independent variable of the equation. Thus we must choose a quadrature scheme that utilizes the endpoints of the interval; otherwise we will not be able to evaluate the functional equation at the relevant quadrature points. One could adopt the view that Volterra equations are, in general, just special cases of Fredholm equations where the kernel is

$$K(x,t) = 0, \ \ t > x \ . \tag{5.2.17}$$

but this would usually require the kernel to be non-analytic However, if we choose such a quadrature formula then, for Volterra equations of type 1, we can write

$$\left.\begin{aligned} F(x_i) &= \sum_{j=0}^{i} K(x_i, x_j)\phi(x_j)W_j \qquad i = 0,1,2,\cdots, n \\ x_k &= a + kh \end{aligned}\right\} . \tag{5.2.18}$$

Not only must the quadrature scheme involve the endpoints, it must be an equal interval formula so that

successive evaluations of the functional equation involve the points where the function has been previously evaluated. However, by doing that we obtain a system of n linear equations in (n+1) unknowns. The value of $\phi(a)$ is not clearly specified by the equation and must be obtained from the functional behavior of F(x). One constraint that supplies the missing value of $\phi(x)$ is

$$\phi(a) - K(a,a) = \left.\frac{dF(x)}{dx}\right|_{x=a} . \tag{5.2.19}$$

The value of $\phi(a)$ reduces equations (5.2.18) to a triangular system that can be solved quickly by successive substitution (see section 2.2). The same method can be used for Volterra equations of type 2 yielding

$$\left.\begin{aligned} F(x_i) &= \phi(x_i) + \sum_{j=0}^{i} K(x_i, x_j)\phi(x_j)W_j \qquad i = 0,1,2,\cdots,n \\ x_k &= a + kh \end{aligned}\right\} . \tag{5.2.20}$$

Here the difficulty with $\phi(a)$ is removed since in the limit as $x \to a$

$$\phi(a) = F(a) . \tag{5.2.21}$$

Thus it would appear that type 2 equations are more well behaved that type 1 equations. To the extent that this is true, we may replace any type 1 equation with a type 2 equation of the form

$$F'(x) = K(x,x)\phi(x) + \int_a^x \frac{\partial K(x,t)}{\partial x}\phi(t)\,dt . \tag{5.2.22}$$

Unfortunately we must still obtain F'(x) which may have to be accomplished numerically.

Consider how these direct solution methods can be applied in practice. Let us choose equation (5.1.10), which served so well as a test case for differential equations. In setting that equation up for Picard's method, we turned it into a type 2 Volterra integral equation of the form

$$y(x) = 1 + x\int_0^x ty\,dt . \tag{5.2.23}$$

If we put this in the form suggested by equation (5.2.17) where the kernel vanishes for t > x, we could write

$$1 = y(x) - x\int_0^x ty\,dt = y(x_i) - x_i\sum_{j=0}^{n} t_j y(t_j)W_j , \quad W_j = 0,\ j > i . \tag{5.2.24}$$

Here we have insured that the kernel vanishes for t>x by choosing the quadrature weights to be zero when that condition is satisfied. The resulting linear equations for the solution become

$$\left.\begin{aligned} 1 &= y(0) - [(0)y(0) + 4(0)y(\tfrac{1}{2}) + (0)y(1)]/6 = y(0), \quad i = 1 \\ 1 &= y(\tfrac{1}{2}) - [(0)y(0) + 4(\tfrac{1}{2})y(\tfrac{1}{2}) + (0)y(1)]/6 = 2y(\tfrac{1}{2})/3, \quad i = 2 \\ 1 &= y(1) - [(0)y(0) + 4(\tfrac{1}{2})y(\tfrac{1}{2}) + y(1)]/6 = -y(\tfrac{1}{2})/3 + 5y(1)/6, \quad i = 3 \end{aligned}\right\} . \tag{5.2.25}$$

The method of using equal interval quadrature formulae of varying degrees of precision as x increases is expresses by equation (5.2.18), which for our example takes the form

$$1 = y(x) - x\int_0^x ty\,dt = y(x_i) - \sum_{j=0}^{i} t_j y(t_j)W_j . \tag{5.2.26}$$

This results in linear equations for the solution that are

$$\left.\begin{aligned} 1 &= y(0) - (0) \\ 1 &= y(½) - [(0)y(0) + (½)y(½)]/2 = 3y(½)/4, \\ 1 &= y(1) - [(0)y(0) + 4(½)y(½) + y(1)]/6 = -y(½)/3 + 5y(1)/6 \end{aligned}\right\} . \qquad (5.2.27)$$

The solutions to the two sets of linear equations (5.2.25) and (5.2.27) that represent these two different approaches are given in table 5.5

Table 5.5

Sample Solutions for a Type 2 Volterra Equation

	Fredholm Soln.	Triangular Soln.	Analytic Soln.
y(0)	1.0	1.0	1.0
% Error	0.0%	0.0%	---------
y(½)	1.5	1.333	1.284
% Error	16.8%	3.8%	---------
y(1)	1.8	1.733	2.718
% Error	-33.8%	-36.2%	---------

As with the other examples, we have taken a large step size so as to emphasize the relative accuracy. With the step size again being unity, we get a rather poor result for the rapidly increasing solution. While both method give answers that are slightly larger than the correct answer at x = ½, they rapidly fall behind the exponentially increasing solution by x = 1. As was suggested, the triangular solution is over all slightly better that then Fredholm solution with the discontinuous kernel.

When applying quadrature schemes directly to Volterra equations, we generate a solution with variable accuracy. The quadrature scheme can initially have a degree of precision no greater than one. While this improves as one crosses the interval the truncation error incurred in the first several points accumulates in the solution. This was not a problem with Fredholm equations as the truncation error was spread across the interval perhaps weighted to some degree by the behavior of the kernel. In addition, there is no opportunity to use the highly efficient Gaussian schemes directly as the points of the quadrature must be equally spaced. Thus we will consider an indirect application of quadrature schemes to the solution of both types of integral equations.

By using a quadrature scheme, we are tacitly assuming that the integrand is well approximated by a polynomial. Let us instead assume that the solution itself can be approximated by a polynomial of the form

$$\phi(x_i) = \sum_{j=0}^{n} \alpha_j \xi_j(x) \ . \qquad (5.2.28)$$

Substitution of this polynomial into the integral of either Fredholm or Volterra equations yields

$$\int K(x,t)\phi(t)\,dt = \sum_{j=0}^{n} \alpha_j \int K(x,t)\xi_j(t)\,dt + R = \sum_{j=0}^{n} \alpha_j H_j(x) + R \ . \qquad 5.2.29)$$

Now the entire integrand of the integral is known and may be evaluated to generate the functions $H_j(x)$. It should be noted that the function $H_j(x)$ will depend on the limits for both classes of equations, but its evaluation poses a separate problem from the solution of the integral equation. In some cases it may be evaluated analytically and in others it will have to be computed numerically for any chosen value of x. However, once that is done, type one equations of both classes can be written as

$$F(x_i) = \sum_{j=0}^{n} \alpha_j H_j(x_i) \quad i=0,1,2, \cdots, n \quad , \tag{5.2.30}$$

which constitute a linear system of (n+1) algebraic equations in the (n+1) unknowns α_j. These, and equation (5.2.28) supply the desired solution $\phi(x)$. Solution for the type 2 equations is only slightly more complicated as equation (5.2.28) must be directly inserted into the integral equation an evaluated at $x=x_i$. However, the resulting linear equations can still be put into standard form so that the α_js can be solved for to generate the solution $\phi(x)$.

We have said nothing about the functions $\xi_j(x)$ that appear in the approximation equation (5.2.28). For nominal polynomial approximation these might be x^j, but for large n such a choice tends to develop instabilities. Thus the same sort of care that was used in developing interpolation formulae should be employed here. One might even wish to employ a rational function approach to approximating $\phi(x)$ as was done in section 3.2. Such care is justified as we have introduced an additional source of truncation error with this approach. Not only will there be truncation error resulting from the quadrature approximation for the entire integral, but there will be truncation error from the approximation of the solution itself [i.e. equation (5.2.28)]. While each of these truncation errors is separately subject to control, their combined effect is less predictable.

Finally, we should consider the feasibility of iterative approaches in conjunction with quadrature schemes for finding solutions to these equations. The type 2 equations immediately suggest an iterative function of the form

$$\phi^{(k)}(x) = F(x) + \lambda \int_a^b K(x,t)\phi^{(k-1)}(t)\,dt \quad . \tag{5.2.31}$$

Remembering that it is $\phi(x)$ that we are after, we can use equation (2.3.20) and the linearity of the integral equations with respect to $\phi(x)$ to establish that the iterative function will converge to a fixed point as long as

$$\left| \lambda \int_a^b K(x,t)\,dt < 1 \right| \quad . \tag{5.2.32}$$

Equation (5.2.17) shows us that a Volterra equation is more likely to converge by iteration than a Fredholm equation with a similar kernel. If λ is small, then not only is the iterative sequence likely to converge, but an initial guess of

$$\phi^{(0)}(x) = F(x) \quad . \tag{5.2.33}$$

suggests itself. In all cases integration required for the iteration can be accomplished by any desireable quadrature scheme as the preliminary value for the solution $\phi^{(k-1)}(x)$ is known.

d. The Influence of the Kernel on the Solution

Although the linearity of the integral operator and its inverse relationship to the differential operator tends to make one think that integral equations are no more difficult than differential equations, there are some subtle differences. For example, one would never attempt a numerical solution of a differential equation that could be shown to have no solution, but that can happen with integral equations if one is not careful. The presence of the kernel under the operator makes the behavior of these equations less transparent than differential equations. Consider the apparently benign kernel

$$K(x,t) = \cos(x)\cos(t) \quad , \tag{5.2.34}$$

and an associated Fredholm equation of the first type

$$F(x) = \int_{-a}^{+a} \cos(x)\cos(t)\phi(t)dt = \cos(x)Z(a) \quad . \tag{5.2.35}$$

Clearly this equation has a solution if and only if F(x) has the form given by the right hand side. Indeed, any kernel that is separable in the independent variables so as to have the form

$$K(x,t) = P(x)Q(t) \quad , \tag{5.2.36}$$

places constraints on the form of F(x) for which the equation has a solution. Nevertheless, it is conceivable that someone could try to solve equation (5.2.35) for functional forms of F(x) other than then those which allow for a value of $\phi(x)$ to exist. Undoubtedly the numerical method would provide some sort of answer. This probably prompted Baker[9], as reported by Craig and Brown[10], to remark 'without care we may well find ourselves computing approximate solutions to problems that have no true solutions'. Clearly the form of the kernel is crucial to nature of the solution, indeed, to its very existence. Should even the conditions imposed on F(x) by equation (5.2.35) be met, any solution of the form

$$\phi(x) = \phi_0(x) + \zeta(x) \quad , \tag{5.2.37}$$

where $\phi_0(x)$ is the initial solution and $\zeta(x)$ is any anti-symmetric function will also satisfy the equation. Not only are we not guaranteed existence, we are not even guaranteed uniqueness when existence can be shown. Fortunately, these are often just mathematical concerns and equations that arise from scientific arguments will generally have unique solutions if they are properly formulated. However, there is always the risk that the formulation will insert the problem in a class with many solutions only one of which is physical. The investigator is then faced with the added problem of finding all the solutions and deciding which ones are physical. That may prove to be more difficult than the numerical problem of finding the solutions.

There are other ways in which the kernel can influence the solution. Craig and Brown[11] devote most of their book to investigating the solution of a class of integral equations which represent inversion problems in astronomy. They show repeatedly that the presence of an inappropriate kernel can cause the numerical methods for the solution to become wildly unstable. Most of their attention is directed to the effects of random error in F(x) on the subsequent solution. However, the truncation error in equation (5.2.3) can combine with F(x) to simulate such errors. The implications are devastating. In Fredholm equations of Type 2, if λ is large and the kernel a weak function of t, then the solution is liable to be extremely unstable. The reason for this can be seen in the role of the kernel in determining the solution $\phi(x)$. K(x,t) behaves like a

filter on the contribution of the solution at all points to the local value of the solution. If K(x,t) is large only for $\tilde{x}t$ then the contribution of the rest of the integral is reduced and $\phi(x)$ is largely determined by the local value of x [i.e. F(x)]. If the Kernel is broad then distant values of $\phi(t)$ play a major role in determining the local value of $\phi(x)$. If λ is large, then the role of F(x) is reduced and the equation becomes more nearly homogeneous. Under these conditions $\phi(x)$ will be poorly determined and the effect of the truncation error on F(x) will be disproportionately large. Thus one should hope for non-separable Kernels that are strongly peaked at x = t.

What happens at the other extreme when the kernel is so strongly peaked at x=t that it exhibits a singularity. Under many conditions this can be accommodated within the quadrature approaches we have already developed. Consider the ultimately peaked kernel

$$K(x,t) = \delta(x-t) \quad , \tag{5.2.38}$$

where $\delta(x)$ is the Dirac delta function. This reduces all of the integral equations discussed here to have solutions

$$\left.\begin{aligned} \phi(x) &= F(x) && \text{type 1} \\ \phi(x) &= F(x)(1-\lambda)^{-1} && \text{type 2} \end{aligned}\right\} \quad . \tag{5.2.39}$$

Thus, even though the Dirac delta function is "undefined" for zero argument, the integrals are well defined and the subsequent solutions simple. For kernels that have singularities at x = t, but are defined elsewhere we can remove the singularity by the simple expedient of adding and subtracting the answer from the integrand so that

$$\phi^{(k)}(x) = F(x) + \lambda\int_a^b K(x,t)[\phi(t) - \phi(x)]\,dt + \lambda\phi(x)\int_a^b K(x,t)\,dt \quad . \tag{5.2.40}$$

We may use the standard quadrature techniques on this equation if the following conditions are met:

$$\left.\begin{aligned} &\left|\int_a^b K(x,t)\,dt\right| < \infty \,, \quad \forall x \\ &\lim_{t\to x}\{K(x,t)[\phi(t) - \phi(x)]\} = 0 \end{aligned}\right\} \quad . \tag{5.2.41}$$

The first of these is a reasonable constraint of the kernel. If that is not met it is unlikely that the solution can be finite. The second condition will be met if the kernel does not approach the singularity faster than linearly and the solution satisfies a Lipshitz condition. Since this is true of all continuous functions, it is likely to be true for any equation that arises from modeling the real world. If this condition is met then the contribution to the quadrature sum from the terms where (i = j) can be omitted (or assigned weight zero). With that slight modification all the previously described schemes can be utilized to solve the resulting equation. Although some additional algebra is required, the resulting linear algebraic equations can be put into standard form and solved using the formalisms from Chapter 2.

In this chapter we have considered the solution of differential and integral equations that arise so often in the world of science. What we have done is but a brief survey. One could devote his or her life to the study of these subjects. However, these techniques will serve the student of science who wishes simply to use them as tools to arrive at an answer. As problems become more difficult, algorithms may need to become more sophisticated, but these fundamentals always provide a good beginning.

Chapter 5 Exercises

1. Find the solution to the following differential equation

   $y' = 3y$,

   in the range $0 \rightarrow 3$. Let the initial value be

   $y(0) = 1$.

   Use the following methods for your solution:

   a. a second order Runge-Kutta
   b. a 3-point predictor-corrector.
   c. Picard's method with 10 steps.
   d. Compare your answer to the analytic solution.

2. Find the solution for the differential equation

   $x^2y'' + xy' + (x^2-6)y = 0$,

   in the range $0 \rightarrow 10$ with initial values of $y'(0)=y(0)=0$. Use any method you like,but explain why it was chosen.

3. Find the numerical solution to the integral equation

   $y(x) = 2 + \int_0^1 y(t)(x^2t+xt^3+5t^5)dt$, $0 \le x \le 2$.

   Comment on the accuracy of your solution and the reason for using the numerical method you chose.

4. Find a closed form solution to the equation in problem 3 of the form

   $y(x) = ax^2 + bx + c$,

   and specifically obtain the values for a,b, and c.

5. How would you have numerically obtained the values for a, b, and c of problem 4 had you only known the numerical solution to problem 3? How would the compare to the values obtained from the closed form solution?

6. We wish to find an approximate solution to the following integral equation:

$$y(x) = 1 + x + 2\int_0^1 t\,x^2\,y(t)\,dt \ .$$

   a. First assume we shall use a quadrature formula with a degree of precision of two where the points of evaluation are specified to be x_1=0.25, x_2=0.5, and x_3=0.75. Use Lagrange interpolation to find the weights for the quadrature formula and use the results to find a system of linear algebraic equations that represent the solution at the quadrature points.

   b. Solve the resulting linear equations by means of Gauss-Jordan elimination and use the results to find a interpolative solution for the integral equation. Comment on the accuracy of the resulting solution over the range $0 \rightarrow \infty$.

7. Solve the following integral equation:

$$B(x) = 1/2\int_0^\infty B(t)E_1\left|t-x\right|dt \ ,$$

where

$$E_1(x) = \int_0^\infty e^{-xt}dt/t \ .$$

   a. First solve the equation by treating the integral as a Gaussian sum. Note that

$$\lim_{x\to 0} E_1\left|x\right| = \infty \ ,$$

   b. Solve the equation by expanding B(t) in a Taylor series about x and thereby changing the integral equation into an nth order linear differential equation. Convert this equation into a system of n first order linear differential equations and solve the system subject to the boundary conditions

$$B(0) = B_0, \quad B'(\infty) = B''(\infty) = \cdots B^{(n)}(\infty) \ .$$

Note that the integral equation is a homogeneous equation. Discuss how that influenced your approach to the problem.

Chapter 5 References and Supplemental Reading

1. Hamming, R.W., "Numerical Methods for Scientists and Engineers" (1962) McGraw-Hill Book Co., Inc., New York, San Francisco, Toronto, London, pp. 204-207.

2. Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., "Numerical Recipies the Art of Scientific Computing" (1986), Cambridge University Press, Cambridge, pp. 569.

3. Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., "Numerical Recipies the Art of Scientific Computing" (1986), Cambridge University Press, Cambridge, pp. 563-568.

4. Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., "Numerical Recipies the Art of Scientific Computing" (1986), Cambridge University Press, Cambridge, pp. 563.

5. Day, J.T., and Collins, G.W.,II, "On the Numerical Solution of Boundary Value Problems for Linear Ordinary Differential Equations", (1964), Comm. A.C.M. 7, pp 22-23.

6. Fox, L., "The Numerical Solution of Two-point Boundary Value Problems in Ordinary Differential Equations", (1957), Oxford University Press, Oxford.

7. Sokolnikoff, I.S., and Redheffer, R.M., "Mathematics of Physics and Modern Engineering" (1958) McGraw-Hill Book Co., Inc. New York, Toronto, London, pp. 425-521.

8. Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., "Numerical Recipies the art of scientific computing" (1986), Cambridge University Press, Cambridge, pp. 615-657.

9. Baker, C.T.N., "The Numerical Treatment of Integral Equations", (1977), Oxford University Press, Oxford.

10. Craig, I.J.D., and Brown, J.C., (1986), "Inverse Problems in Astronomy -A Guide to Inversion Strategies for Remotely Sensed Data", Adam Hilger Ltd. Bristol and Boston, pp. 51.

11 Craig, I.J.D., and Brown, J.C., (1986), "Inverse Problems in Astronomy -A Guide to Inversion Strategies for Remotely Sensed Data", Adam Hilger Ltd. Bristol and Boston.