

Numerical Solution of Ordinary Differential Equations

“Numerical Integration”

CONTENTS:

Introduction

A. Initial Value Problems

B. Boundary Value Problems

C. Solutions of Homogeneous Linear Difference Equations

These notes were produced by Les Jennings for the third year Numerical Analysis course for Mathematics students at The University of Western Australia. Further details are given in references cited in the text. Printed on 8 September 2004

Numerical Solution of Ordinary Differential Equations

“Numerical Integration”

Consider the problem of finding the solution $y(x)$ to the ordinary differential equation

$$\frac{dy}{dx} = y'(x) = f(x, y), \quad y(a) = y_0, \quad (1)$$

in an interval $a \leq x \leq b$ (or $b \leq x \leq a$). The more general problem can be written in vector form

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}), \quad \mathbf{y}(a) = \mathbf{y}_0 \quad (2)$$

where \mathbf{y} is a vector of functions (y_1, y_2, \dots, y_n) to be determined, and \mathbf{f} is a function of possibly $n + 1$ variables,

$$\begin{aligned} y_1' &= f_1(x, y_1, \dots, y_n), & y_1(a) &= y_{1,0}, \\ y_2' &= f_2(x, y_1, \dots, y_n), & y_2(a) &= y_{2,0}, \\ &\vdots & & \\ y_n' &= f_n(x, y_1, \dots, y_n), & y_n(a) &= y_{n,0}. \end{aligned}$$

Both forms (1) and (2) are first order systems, linear in the first derivative. They are initial value problems in that data is given only at one point in the interval where a solution is required.

Higher order equations which are initial value problems may be put into the above form by renaming the derivatives of $y(x)$. For example,

$$y'' + p(x)y'(x) + q(x)y^2(x) = h(x), \quad y(a) = y_0, \quad y'(a) = y_0'.$$

Let $y_1 = y$ and $y_2 = y'$, then $y'' = y_2'$, and the equation may be written as the first order system

$$\begin{aligned} y_1'(x) &= y_2(x), & y_1(a) &= y_0, \\ y_2'(x) &= -p(x)y_2(x) - q(x)y_1^2 + h(x), & y_2(a) &= y_0'. \end{aligned}$$

Higher order equations (or systems) with conditions given at both ends of some interval are called boundary value problems. There are different techniques of solution for the class of initial value problems and the class of boundary value problems. Boundary value problems may also have more problems with existence of a solution than initial value problems. Remember also that some differential equations have unbounded solutions on some interval. Most computer software for differential equations cannot compute these solutions close to these vertical asymptotes.

The approximate solution

Most numerical techniques generate a set of points $\{(x_i, Y_i), i = 0, 1, \dots, N\}$, where Y_i is an approximation to $y(x_i) = y_i$. In many cases the x_i are equally spaced and we will assume this, so that $x_i = a + ih$ and $h = (b - a)/N$. We are interested in the error $y_i - Y_i$, at each point x_i . This error does not have a straight forward form like interpolation, numerical differentiation or quadrature procedures. We will treat ordinary differential equations with a single unknown function $y(x)$, as most methods generalize easily to n unknown functions.

A. Initial Value Problems

Consider

$$\frac{dy}{dx} = y'(x) = f(x, y), \quad y(a) = y_0.$$

The existence of a solution on $[a, b]$ and its uniqueness is implied if

- (i) $f(x, y)$ is defined and is continuous on $a \leq x \leq b$, where a and b are finite and $-\infty < y < \infty$,

(ii) f is Lipschitz in y , that is, there exists an L such that for all $x \in [a, b]$ and all pairs y and y^* ,

$$|f(x, y) - f(x, y^*)| < L|y - y^*|.$$

Some of these conditions can be relaxed in special cases.

There are two major classes of methods for “integrating” an ordinary differential equation. Note that the fundamental theorem of differential and integral calculus implies on integrating (1),

$$y(x) = y_0 + \int_{x_0}^x y'(t) dt = y_0 + \int_{x_0}^x f(t, y(t)) dt.$$

These are (i), finite difference methods or multistep methods, and (ii), Runge–Kutta methods.

Of less importance is the Taylor series method. We have

$$y(x_0 + h) = y(x_0) + hy'(x_0) + \frac{h^2}{2}y''(x_0) + \cdots + \frac{h^n}{n!}y^{(n)}(x_0) + R_n$$

where $y'(x_0) = f(x_0, y_0)$, which is known, and,

$$y''(x_0) = \frac{\partial f}{\partial x}(x_0, y_0) + \frac{\partial f}{\partial y}(x_0, y_0) \cdot y'(x_0)$$

which is also known, and so on. The higher derivatives are rather tedious to calculate but usually straight forward (and modern CA packages could do them for you). A value for Y_1 is found by ignoring the remainder term of the Taylor series, which is the truncation error $y_1 - Y_1$. The next step is to attempt to calculate Y_2 and as we only have Y_1 and not y_1 we have to modify the above formula to

$$Y_2 = Y_1 + hY'(x_1) + \frac{h^2}{2}Y''(x_1) + \cdots + \frac{h^n}{n!}Y^{(n)}(x_1)$$

where

$$Y^{(m)}(x_1) = \frac{d^{m-1}f}{dx^{m-1}}(x_1, Y_1).$$

This has now complicated the truncation error which now consists of a local truncation error and error propagated from step one. We call the sum of these errors the global error. The other method to calculate Y_2 is to use the first formula with h replaced by $2h$. For the same number of terms in the Taylor series this produces a larger error in y_2 .

The diagram demonstrates this geometrically for the simplest Taylor series method, the Euler Method. The top value of Y_2 comes from taking a step of $2h$ from the initial point, while the lower value of Y_2 takes two Euler steps, the second from $x_1 = h$.

$$Y(x_0 + h) = Y_0 + hY'(x_0) = y_0 + hf(x_0, Y_0)$$

$$y(x_0 + h) = y_0 + hy'(x_0) + \frac{h^2}{2}y''(\xi), \quad x_0 < \xi < x_0 + h.$$

Hence the error $y(x_1) - Y_1 = \frac{h^2}{2}y''(\xi)$, and is the local truncation error, the error in one step forward assuming that there is no error in y_0 . From the diagram we can see that generally taking single steps is better than successively larger steps from the initial point.

A1. Finite difference methods or multi-step methods

To explain these methods we assume we have computed up to the n -th step, that is we know Y_0, Y_1, \dots, Y_n .

Formulae of open type or explicit methods.

We have

$$y(x_n + h) = y(x_n) + \int_{x_n}^{x_n+h} y'(x) dx,$$

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} y'(x) dx.$$

We approximate the integration in the above equation. The discussion is in terms of unrealizable history up to the point x_n . Suppose Y_0, \dots, Y_n are known and equal to y_0, \dots, y_n , and hence Y'_0, \dots, Y'_n are known (and equal to y'_0, \dots, y'_n) as $Y'_i = f(x_i, Y_i)$. We can fit a polynomial to some of the points $\{(x_i, Y'_i), i = 0, \dots, n\}$, using an appropriate number of the later points. The easiest approach is Newton's Backward Difference formulae for interpolation.

$$y'(x_n + sh) = y'_{n+sh} = E^s y'_n = (1 - \nabla)^{-s} y'_n$$

$$= y'_n - (-s)\nabla y'_n + \frac{-s(-s-1)}{2}\nabla^2 y'_n + \dots + (-1)^N \binom{-s}{N} \nabla^N y'_n + \dots$$

where $x = x_n + sh$. Changing variable from x to s in the integral form gives

$$y(x_n + h) = y(x_n) + h \int_{s=0}^{s=1} y'(x_n + sh) ds$$

$$y_{n+1} = y_n + h \int_0^1 y'_{n+sh} ds$$

$$Y_{n+1} = Y_n + h \int_0^1 \sum_{i=0}^N (-1)^i \binom{-s}{i} \nabla^i Y'_n ds$$

$$= Y_n + h \sum_{i=0}^N \left\{ \int_0^1 (-1)^i \binom{-s}{i} ds \right\} \nabla^i Y'_n$$

$$= Y_n + h \sum_{i=0}^N \bar{a}_i \nabla^i Y'_n, \quad \bar{a}_i = \int_0^1 (-1)^i \binom{-s}{i} ds$$

$$= Y_n + h \sum_{i=0}^N a_i Y'_{n-i}, \quad \text{on expanding differences.}$$

The local truncation error (assuming all past values are exact, which is unrealistic) is $|y_{n+1} - Y_{n+1}|$ and is

$$\left| (-1)^{N+1} h^{N+2} \int_0^1 \binom{-s}{N+1} y^{(N+2)}(\xi(s)) ds \right|$$

and can be estimated as $\bar{a}_{N+1} \nabla^{N+1} y'_n$, that is, the first term left out. Note that we do not have $y_i = Y_i$ for $i = 0, \dots, n$, and hence the above is not the actual error at x_{n+1} in practice. Other integration formulae can be obtained by writing

$$y_{n+1} = y_{n-p} + h \int_{-p}^1 y'(x_n + sh) ds$$

$$Y_{n+1} = Y_{n-p} + h \sum_{i=0}^N \bar{a}_i^p \nabla^i Y'_n, \quad \bar{a}_i^p = \int_{-p}^1 (-1)^i \binom{-s}{i} ds$$

$$= Y_{n-p} + h \sum_{i=0}^N a_i^p Y'_{n-i}.$$

These formulae are called open or explicit because the next value, Y_{n+1} is given explicitly in terms of past values. If $N = p$ the formulae correspond to Newton-Cotes open quadrature formulae. Infinitely many other formulae can be invented by taking combinations of the above formulae for different p .

Formulae of closed type or implicit formulae

In these formulae, Newton's backward difference formula is expanded about the point x_{n+1} .

$$\begin{aligned} y'(x_n + sh) &= E^{s-1} y'_{n+1} = (1 - \nabla)^{-s+1} y'_{n+1} \\ &= y'_{n+1} + (s-1)\nabla y'_{n+1} + \frac{(-s+1)(-s)}{2} \nabla^2 y'_{n+1} + \cdots + (-1)^N \binom{-s+1}{N} \nabla^N y'_{n+1} + \cdots \end{aligned}$$

where as before $x = x_n + sh$. Changing variable from x to s in the integral above gives

$$\begin{aligned} y(x_n + h) &= y(x_{n-p}) + h \int_{s=-p}^{s=1} y'(x_n + sh) ds \\ y_{n+1} &= y_{n-p} + h \int_{-p}^1 y'_{n+s} ds \\ Y_{n+1} &= Y_{n-p} + h \int_{-p}^1 \sum_{i=0}^N (-1)^i \binom{-s+1}{i} \nabla^i Y'_{n+1} ds \\ &= Y_{n-p} + h \sum_{i=0}^N \left\{ \int_{-p}^1 (-1)^i \binom{-s+1}{i} ds \right\} \nabla^i Y'_{n+1} \\ &= Y_{n-p} + h \sum_{i=0}^N \bar{b}_i^p \nabla^i Y'_{n+1}, \quad \bar{b}_i^p = \int_{-p}^1 (-1)^i \binom{-s+1}{i} ds \\ &= Y_{n-p} + h \sum_{i=0}^N b_i^p Y'_{n+1-i}, \quad \text{on expanding differences.} \\ &= Y_{n-p} + h \sum_{i=0}^N b_i^p f(x_{n+1-i}, Y_{n+1-i}) \end{aligned}$$

Note that Y_{n+1} is involved on both sides of the equation now. If f is linear in y we can take it across to the left and hence get an explicit equation for Y_{n+1} . If f is nonlinear in y the last equation is a nonlinear function in Y_{n+1} and is solved iteratively for Y_{n+1} . We have a good guess of Y_{n+1} namely Y_n . In fact one iteration for this equation is to simply keep substituting the new value of Y_{n+1} into the right hand side. If h is small enough this converges very quickly. The local truncation errors for these formulae are

$$\left| h^{N+2} \int_{-p}^1 \binom{-s+1}{N} y^{(N+2)}(\xi(s)) ds \right| \approx |\bar{b}_n^p \nabla^{N+1} y'_{n+1}|.$$

If p is odd and N an appropriate value the Newton-Cotes closed formulae are produced. The error coefficients for the closed formulae are smaller in general as we are implicitly interpolating for the closed formulae but extrapolating for the open formulae.

Predictor-Corrector Methods.

We can use an open type formula to 'predict' a value for Y_{n+1} and substitute this value into the right hand side of a closed formulae to 'correct' the value of Y_{n+1} . The value of Y_{n+1} may be corrected iteratively as before but usually there is no need. One particular advantage of this method is that the local truncation error is approximated very well as a scalar multiple of the difference between the predicted and corrected value of Y_{n+1} , so a difference table of Y'_i values does not have to be computed. An example of a predictor-corrector pair is the Euler and modified Euler formulae.

$$P : \quad Y_{n+1}^P = Y_n + hY'_n$$

$$C : \quad Y_{n+1}^C = Y_n + \frac{h}{2}(Y'_n + Y'_{n+1}), \quad Y'_{n+1} = f(x_{n+1}, Y_{n+1}^P)$$

By expanding about x_n using Taylor series we can show that $y_{n+1} - Y_{n+1}^C = \mathcal{O}(h^3)$, if we assume $Y_n = y_n$.

Suppose we have a predictor–corrector pair which have the same order of accuracy, and assume that $Y_i = y_i$, $i = 0, \dots, n$. Consider the local truncation errors in Y_{n+1}^P and Y_{n+1}^C ,

$$y_{n+1} - Y_{n+1}^P = ay_n^{(q)}h^p + \mathcal{O}(h^{p+1})$$

$$y_{n+1} - Y_{n+1}^C = by_n^{(q)}h^p + \mathcal{O}(h^{p+1})$$

Then

$$Y_{n+1}^C - Y_{n+1}^P = (a - b)y_n^{(q)}h^p + \mathcal{O}(h^{p+1}),$$

and hence on substituting for

$$y_n^{(q)}h^p$$

in the error term for the corrector we get

$$y_{n+1} - Y_{n+1}^C = \frac{b}{a - b}(Y_{n+1}^C - Y_{n+1}^P) + \mathcal{O}(h^{p+1})$$

We can use this simple multiple of the difference between the predictor and corrector values to monitor the local truncation error. We usually do not bother to add the correction to the corrected value of Y_{n+1} (and hence increase the order of the local truncation error). Some formulae for predictor-corrector with errors $\mathcal{O}(h^5)$ are:

1. Modified Adams Method. (Moulton (1926))

$$P : \quad Y_{n+1}^P = Y_n + \frac{h}{24}(55Y'_n - 59Y'_{n-1} + 37Y'_{n-2} - 9Y'_{n-3})$$

$$C : \quad Y_{n+1}^C = Y_n + \frac{h}{24}(9Y'_{n+1} + 19Y'_n - 5Y'_{n-1} + Y'_{n-2})$$

$$y_{n+1} - Y_{n+1}^C \approx \frac{-19}{270}(Y_{n+1}^C - Y_{n+1}^P) = \mathcal{O}(h^5).$$

2. Milne's method, (not as good).

$$P : \quad Y_{n+1}^P = Y_{n-3} + \frac{4h}{3}(2Y'_n - Y'_{n-1} + 2Y'_{n-2})$$

$$C : \quad Y_{n+1}^C = Y_{n-1} + \frac{h}{3}(Y'_{n+1} + 4Y'_n + Y'_{n-1})$$

$$y_{n+1} - Y_{n+1}^C \approx \frac{-1}{29}(Y_{n+1}^C - Y_{n+1}^P) = \mathcal{O}(h^5).$$

3. Hamming's method.

$$P : \quad Y_{n+1}^P = Y_{n-3} + \frac{4h}{3}(2Y'_n - Y'_{n-1} + 2Y'_{n-2})$$

$$C : \quad Y_{n+1}^C = \frac{1}{8}(9Y_n - Y_{n-1}) + \frac{3h}{8}(Y'_{n+1} + 2Y'_n - Y'_{n-1})$$

$$y_{n+1} - Y_{n+1}^C \approx \frac{-9}{121}(Y_{n+1}^C - Y_{n+1}^P) = \mathcal{O}(h^5).$$

Each of the methods in this class, open, closed, predictor-corrector (and others) are used in the following manner. Firstly some starting values are found (for example using Taylor series or Runge-Kutta formulae) to the same accuracy as the multistep method to be used. Once enough points are known the multistep formulae integrate forward using one or two new function values per step. The error is monitored either using differences or the above in the case of predictor-corrector methods. If the error becomes too large the step h is halved, new intermediate points are computed using interpolation or

Runge-Kutta, and the procedure continued at the smaller value of h . If the error becomes much smaller than required the step can be doubled for efficiency. Usually the user requires values at pre-arranged points, so interpolation (to the same accuracy) is used to approximate y at these points.

Consistency, Stability and Convergence for Multi-step Methods

Write the ordinary differential equation as $y' = f(x, y)$, with given initial condition. An explicit multi-step numerical scheme for integrating the ordinary differential equation can usually be put in the form

$$y_{n+1} = \sum_{i=0}^q a_i y_{n-i} + h \sum_{i=0}^p b_i y'_{n-i} + h\tau_n(h),$$

where $h\tau(h)$ is the local truncation error.

A numerical integration scheme is **consistent** if $\tau_n(h) \rightarrow 0$, as $h \rightarrow 0$, for all n . Rearranging the above equation as

$$\frac{y_{n+1} - \sum_{i=0}^q a_i y_{n-i}}{h} = \sum_{i=0}^p b_i y'_{n-i} + \tau_n(h),$$

we see that consistency is for the derivative of y , that is, on the left is an approximation to y' , while on the right is a weighted sum of y' . As $\tau_n(h)$ will have a term like $y^{(k)}(\xi)$, the local truncation error will be zero for ordinary differential equations of the form

$$y' = x^j, \quad 0 \leq j < k.$$

A numerical integration scheme is **stable** if all homogeneous solutions of the difference equation

$$y_{n+1} - \sum_{i=0}^q a_i y_{n-i} = 0,$$

are bounded as $n \rightarrow \infty$. All roots of the auxiliary equation are either < 1 in magnitude and if equal to 1 in magnitude occur with multiplicity one. This is a first approximation to the idea of stability. Note that this idea of stability is equivalent to making sure the solution of the ode $y' = 0$, $y(0) = 1$ say, has a numerical value which approaches $y(x) = 1$ as $h \rightarrow 0$, over all perturbations of computed y_i values of order h .

By **convergence** we mean that as $h \rightarrow 0$ the global error $Y(x) - y(x)$ tends to zero for all x in an appropriate domain.

Theorem: *Consistency and stability imply convergence.* ■

For implicit multistep methods the above Theorem also holds as the only difference is in the sum of derivative values ($\sum_{i=-1}^p$). Better results can be obtained by considering the first order Taylor expansion of $y'_{n-i} = f(x_{n-i}, y_{n-i})$ and including the subsequent linear term in Y_{n-i} in the difference equation when considering stability. This brings h into the stability definition giving a more accurate stability picture. Typical results produced using these techniques are that a method is convergent provided $h < h_{max}$ for some h_{max} related to $|\partial f / \partial y|$. (See Isaacson and Keller.)

A2. Runge-Kutta Methods.

Basically, using the ordinary differential equation $y'(x) = f(x, y)$ the Taylor series

$$y_{n+1} = y_n + h y'_n + \frac{h^2}{2} y''_n + \dots + \frac{h^p}{p!} y_n^{(p)} + \dots$$

is used to find the values of α_i , $i = 0, \dots, p$, μ_i , ν_i , $i = 1, \dots, p$, in the formula

$$Y_{n+1} = Y_n + h \left[\alpha_0 f(x_n, y_n) + \sum_{i=1}^p \alpha_i f(x_n + \mu_i h, Y_n + \nu_i h) \right].$$

This involves the algebraic computation of the formal p -th derivatives of an arbitrary function f . There are no solutions for some values of p , usually many solutions if there are any, and the truncation error is

found by taking the expansions one derivative further. It is possible to have both explicit and implicit formulae. Once the formulae are found it is a simple matter to use them as they compute the next value of y using $p + 1$ function evaluations.

Comparing these methods to the multi-step methods we see that the Runge-Kutta has the advantage of being self starting, and of being able to change step length at any time. The error needs to be monitored by a difference table, or other more involved methods similar to the predictor-corrector method of monitoring error. The Runge-Kutta are better than the Taylor series technique as no derivatives are required. But multistep methods once started are much more efficient requiring only one or two function evaluations per step, and the predictor-corrector methods have a simple error monitoring computation.

Explicit Runge-Kutta methods

These explicit methods are directly computable as each new function value is given explicitly in terms of the previous ones. We have

$$Y_{n+1} = Y_n + \alpha_0 k_0 + \alpha_1 k_1 + \cdots + \alpha_p k_p,$$

where

$$\begin{aligned} k_0 &= hf(x_n, Y_n) \\ k_1 &= hf(x_n + \mu_1 h, Y_n + \lambda_{1,0} k_0) \\ k_2 &= hf(x_n + \mu_2 h, Y_n + \lambda_{2,0} k_0 + \lambda_{2,1} k_1) \\ &\vdots \\ k_p &= hf(x_n + \mu_p h, Y_n + \lambda_{p,0} k_0 + \cdots + \lambda_{p,p-1} k_{p-1}) \end{aligned}$$

where $\{\alpha_i, i = 0, \dots, p\}$, $\{\mu_i, i = 1, \dots, p\}$ and, $\{\lambda_{i,j}, j = 0, \dots, i-1, i = 1, \dots, p\}$ are to be found by agreeing with the Taylor series expansion to a certain order. This is a total of $\frac{1}{2}(p^2 + 5p + 2)$ coefficients to determine.

Huen's method

$$\begin{aligned} Y_{n+1} &= Y_n + \frac{1}{2}(k_0 + k_1) \\ k_0 &= hf(x_n, Y_n), \quad k_1 = hf(x_n + h, Y_n + k_0). \\ y_{n+1} - Y_{n+1} &= \mathcal{O}(h^3). \end{aligned}$$

Kutta's method

$$\begin{aligned} Y_{n+1} &= Y_n + \frac{1}{6}(k_0 + 4k_1 + k_2) \\ k_0 &= hf(x_n, Y_n), \quad k_1 = hf(x_n + \frac{1}{2}h, Y_n + \frac{1}{2}k_0) \\ k_2 &= hf(x_n + h, Y_n + 2k_1 - k_0) \\ y_{n+1} - Y_{n+1} &= \mathcal{O}(h^4) \end{aligned}$$

An example of the algebraic calculation for an explicit method follows.

The Taylor series for $f(x + h)$ about x , where all derivatives on the right are evaluated at x , is

$$\begin{aligned} y(x + h) &= y + hf + \frac{h^2}{2}(f_x + f_y f) + \frac{h^3}{6}(f_{xx} + 2f_{xy} f + f_y^2 f + f_y f_x + f_{yy} f^2) \\ &\quad + \frac{h^4}{24}(\cdots) + \cdots \end{aligned}$$

The explicit formula above is expanded about x ,

$$\begin{aligned} Y(x + h) &= y + \alpha_0 hf \\ &\quad + \alpha_1 h \left[f + \mu_1 h f_x + \lambda_{10} h f_y f + h^2 \left(\frac{\mu_1^2}{2} f_{xx} + \mu_1 \lambda_{10} f_{xy} f + \frac{\lambda_{10}^2}{2} f_{yy} f^2 \right) + h^3(\cdots) \right] \\ &\quad + \alpha_2 h \left[f + \mu_2 h f_x + \lambda_{20} h f_y f + h^2 \left(\frac{\mu_2^2}{2} f_{xx} + \mu_2 \lambda_{20} f_{xy} f + \frac{\lambda_{20}^2}{2} f_{yy} f^2 \right) + h^3(\cdots) \right. \\ &\quad \left. + \lambda_{21} h^2 (f + \mu_1 h f_x + \cdots) \right] \\ &\quad + \alpha_3 h \left[\cdots \right] \end{aligned}$$

Collecting terms for $p = 1$ to $\mathcal{O}(h^2)$ we have for the coefficients of terms hf , h^2f_x and h^2f_yf respectively,

$$\begin{aligned}\alpha_0 + \alpha_1 &= 1, \\ \alpha_1\mu_1 &= \frac{1}{2}, \\ \alpha_1\lambda_{10} &= \frac{1}{2}.\end{aligned}$$

These three nonlinear equations in four unknowns have an infinity of solutions in say parameter α_1 , where $\alpha_0 = 1 - \alpha_1$ and $\mu_1 = \lambda_{10} = \frac{1}{2}\alpha_1$. Choosing $\alpha_1 = \frac{1}{2}$ gives Huen's formula (Modified Euler). Stability criteria determine the best value of the parameter α_1 . Stability for Runge-Kutta methods are not considered in these notes. See for example Isaacson and Keller.

A3. Stiff Differential Equations.

Some differential equations present difficulties when numerically integrated, usually caused by the functions being computed having the property of being of size 1 say, but the derivative being very large. Even consistent and stable formulae fail to compute the solution accurately for modest values of h . These equations are called *stiff*. Note that solution functions which have a singularity at a point in the domain will also appear stiff. An example of stiffness occurs when a simple linear constant coefficient ode has two eigenvalues with real part very different in magnitude. Consider the exaggerated example

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -10^6 - 1 & 10^6 \\ 10^6 & -10^6 - 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

One eigenvalue is $-2 \cdot 10^6 - 1$, the other is -1 . The solution is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_1 e^{-x} \\ c_2 e^{(-2 \cdot 10^6 - 1)x} \end{bmatrix} = \begin{bmatrix} e^{-x} + e^{(-2 \cdot 10^6 - 1)x} \\ e^{-x} - e^{(-2 \cdot 10^6 - 1)x} \end{bmatrix}, \quad \text{if } \mathbf{x}(0) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

Note that one component dies away to zero very rapidly compared to the other. Hence a very small value of h is needed to follow this component. Try the Euler method on this example for h successively taking values 10^{-4} , 10^{-6} and 10^{-8} . This structure is not the only way an ode can be stiff, for example, ode's with solution $\sqrt{1-x}$, will be stiff as $x \rightarrow 1$.

B. Boundary Value Problems.

There are in general three major classes of methods for solving boundary value problems. These are (i) finite difference methods, (ii) shooting methods of various sophistication, and, (iii) finite element methods (FEM).

B1. Finite Difference Methods

Basically we approximate derivatives in the ordinary differential equation by finite differences, all to the same order preferably, and then solve the resulting equations in the unknowns which are the function values $\{Y_i, i = 1, \dots, N - 1\}$. If the ordinary differential equation is linear in all derivatives and the function y then the equations are linear, otherwise not.

Example: Consider the linear equation

$$y''(x) + p(x)y'(x) + q(x)y(x) = r(x), \quad y(a) = y_0, \quad y(b) = y_N.$$

Let $x_i = a + (i - 1)h$, $h = (b - a)/N$ and $y_i = y(x_i)$. Approximate the derivatives as follows,

$$y''(x_i) = \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + \mathcal{O}(h^2),$$

$$y'(x_i) = \frac{y_{i+1} - y_{i-1}}{2h} + \mathcal{O}(h^2),$$

and substitute into the ordinary differential equation evaluated at each point x_i . This gives

$$y_{i-1} - 2y_i + y_{i+1} + \frac{hp_i}{2}(y_{i+1} - y_{i-1}) + h^2q_iy_i = h^2r_i + \mathcal{O}(h^4)$$

for $i = 1, \dots, N-1$ and where $p_i = p(x_i)$, $q_i = q(x_i)$ and $r_i = r(x_i)$. Collecting terms gives the three term recurrence relation (difference equation)

$$(1 - \frac{hp_i}{2})y_{i-1} + (-2 + h^2q_i)y_i + (1 + \frac{hp_i}{2})y_{i+1} = h^2r_i + \mathcal{O}(h^4)$$

where y_0 and y_N are known. Hence the $(N-1) \times (N-1)$ tridiagonal system

$$\mathbf{A}\mathbf{y} = \mathbf{b} + \mathcal{O}(h^4),$$

where $A_{i,i} = -2 + h^2q_i$, $A_{i,i-1} = 1 - hp_i/2$, $A_{i,i+1} = 1 + hp_i/2$, and, $b_1 = h^2r_1 - (1 - hp_1/2)y_0$, $b_{N-1} = h^2r_{N-1} - (1 + hp_{N-1}/2)y_N$, $b_i = h^2r_i$, for $i = 2, \dots, N-2$. The method of solution is to ignore the truncation error, so that the vector \mathbf{Y} which approximates \mathbf{y} is given by

$$\mathbf{A}\mathbf{Y} = \mathbf{b}, \quad \text{or} \quad \mathbf{Y} = \mathbf{A}^{-1}\mathbf{b}.$$

The error $\|\mathbf{Y} - \mathbf{y}\| = \|\mathbf{A}^{-1}\mathcal{O}(h^4)\|$, and in fact as $h \rightarrow 0$, $\mathbf{A}^{-1} \rightarrow \mathcal{O}(h^{-2})$ so that

$$\|\mathbf{Y} - \mathbf{y}\| = \mathcal{O}(h^2).$$

This means that as $h \rightarrow 0$ we have convergence of the numerical solution to the exact solution.

Methods to improve the accuracy.

1. Use a smaller value of h , a larger value of N . If N is doubled again and again Richardson's extrapolation can be used to get very accurate values at the original N points. (A vector Richardson's extrapolation.)
2. Calculate the $\mathcal{O}(h^4)$ terms as $\mathbf{a}h^4 + \mathcal{O}(h^6)$ by finite differences on the computed \mathbf{Y} values, then add the $\mathbf{a}h^4$ to \mathbf{b} and resolve for \mathbf{Y} . This process can be extended to calculate the higher order terms. It has the advantage of keeping the size of the matrix the same.
3. We can use more accurate formulae for the initial approximation to the derivatives, which involves more points. The derivatives at the ends are approximated by lopsided formulae. This leads to matrix fill-in as more bands are added to \mathbf{A} .

Derivative boundary conditions are approximated by finite differences. For example if $y'(a)$ is given, then y_0 becomes an unknown in the equations and we have to evaluate the ordinary differential equation at $x_0 = a$. This involves y_{-1} which is eliminated using the derivative boundary value

$$y'(a) = \frac{y_1 - y_{-1}}{2h} + \mathcal{O}(h^2) \quad \text{or} \quad y_{-1} = y_1 - 2hy'(a).$$

If the ode is not linear, then instead of computing $\mathbf{A}^{-1}\mathbf{b}$ above, a set of nonlinear equations in \mathbf{Y} has to be computed.

B2. Shooting Methods

Suppose the equation has the very general form $y'' = f(x, y, y')$ with boundary values $y(a) = y_0$, and $y(b) = y_N$. The solution function y has a slope $y'(a)$ at the left boundary. If we could guess this value we could use initial value software to integrate the equation forward. Initial value software is usually much more accurate and less complicated than the nonlinear equation solving which has to go with the finite difference method for boundary value problems. So the most basic shooting method is equivalent to firing artillery. We guess the slope of the gun barrel in order to hit the target at b . If α is the guessed slope and $Y_N(\alpha)$ the value of y at b after integrating using initial value software then we carry out a zero finding algorithm on $Y_N(\alpha) - y(b)$, where we can only evaluate function values. Much more sophisticated methods are employed in modern software, where existence of a solution also has to be detected. See for example the references given in the Nag software library.

B3. Finite Element Methods

The unknown function is approximated by a linear combination of basis functions,

$$y(x) \approx \sum_{j=1}^N a_j \phi_j(x).$$

The basis functions are usually chosen to have the property of *finite support*. That is, the function $\phi_j(x)$ is non-zero in a small subinterval of $[a, b]$ and zero on most of $[a, b]$. For example, the B-spline basis functions. The linear combination of basis functions is substituted into the BVP which is then evaluated at a set of points (this is called collocation), for example,

$$\sum_{j=1}^N a_j (\phi_j''(x_i) + p(x_i)\phi_j'(x_i) + q(x_i)\phi_j(x_i)) = r(x_i), \quad i = 1, \dots, N.$$

The set of points at which the BVP is evaluated is related to the set of points defining the finite support of the basis functions. Alternatively, the residual of the ode is minimised over the space of basis functions, to give the following equations,

$$\int_a^b \phi_i(x) \left(\sum_{j=1}^N a_j \phi_j''(x) + p(x) \sum_{j=1}^N a_j \phi_j'(x) + q(x) \sum_{j=1}^N a_j \phi_j(x) - r(x) \right) dx = 0, \quad i = 1, \dots, N.$$

This can be simplified to

$$\sum_{j=1}^N a_j C_{ij} = \int_a^b \phi_i(x) r(x) dx, \quad i = 1, \dots, N.$$

where

$$C_{ij} = \phi_i(b)\phi_j'(b) - \phi_i(a)\phi_j'(a) + \int_a^b -\phi_i'(x)\phi_j'(x) + \phi_i(x)(p(x)\phi_j'(x) + q(x)\phi_j(x)) dx.$$

Either method gives a set of equations for the unknown coefficients a_j , $j = 1, \dots, N$. Because of the finite support property these equations are usually banded, making their computation able to be done in order N operations.

The FEM becomes particularly important for partial differential equations which use basis functions with finite support in 2 or 3 dimensions.

B4. Eigenvalue problems (Sturm-Liouville)

Suppose we have the equation

$$y''(x) + p(x)y'(x) + q(x)y(x) = \alpha y(x),$$

with homogeneous boundary values on the function and or the derivative. We want to find a solution being an eigenvalue α and the corresponding eigen-function y . Discretizing as before we get the matrix eigenvalue problem in approximate 'eigen-function' \mathbf{Y} ,

$$\mathbf{A}\mathbf{Y} = \lambda\mathbf{Y}, \quad \lambda = h^2\alpha.$$

The FEM can also be used to set up an eigenvalue/vector equation in the coefficients of the linear combination of basis functions. The eigenvalues and vectors can be computed using LAPACK (or EISPACK).

C. Solutions of Homogeneous Linear Difference Equations

First order, constant coefficient

Consider the difference equation

$$y_{n+1} = ay_n, \quad y_0 \text{ given.}$$

This has solution $y_n = y_0 a^n$. So as $n \rightarrow \infty$, $y_n \rightarrow 0$ if $|a| < 1$, or $y_n = y_0$, if $a = 1$, or, y_n alternates between y_0 and $-y_0$ if $a = -1$, or, $|y_n| \rightarrow \infty$ if $|a| > 1$.

Second order, constant coefficient

Consider the difference equation

$$y_{n+1} = ay_n + by_{n-1}, \quad y_0, \text{ and } y_1 \text{ given.}$$

This is solved in a similar fashion to the second order constant coefficient ode's. Try $y_n = r^n$, substitute into the difference equation to get

$$r^{n+1} - ar^n - br^{n-1} = 0.$$

This gives a quadratic in r , (as $r = 0$ gives the zero solution), $r^2 - ar - b = 0$, the auxiliary equation, which is solved to give,

$$\begin{aligned} r_1 &= a/2 + \sqrt{a^2/4 + b}, \\ r_2 &= a/2 - \sqrt{a^2/4 + b}. \end{aligned}$$

The solution is, in terms of two arbitrary constants c_1 and c_2 ,

$$y_n = c_1 r_1^n + c_2 r_2^n,$$

but the usual three cases have to be considered. Two unequal real roots give the solution as above. Two equal roots mean that the solution can be written as

$$y_n = (c_1 + nc_2)r_1^n.$$

In the case of two complex conjugate roots, when $a^2/4 + b < 0$, the solution can be written as

$$y_n = r^n (c_1 \cos(n\theta) + c_2 \sin(n\theta)),$$

where $r = \sqrt{-b}$ and $\theta = \arctan(\sqrt{-b - a^2/4}, a/2)$.

In terms of whether $y_n \rightarrow 0$ or $y_n \rightarrow \infty$ or otherwise both roots of the auxiliary equation have to be considered as the initial points y_0 and y_1 are assumed arbitrary. So solutions are bounded as $n \rightarrow \infty$ if both roots of the auxiliary equation have modulus less than or equal one, and if both roots have modulus one then they must be different roots. The solution is unbounded if any root is outside the unit circle in the complex plane, or if the two roots are equal and on the unit circle of the complex plane.

Higher orders

The solutions are determined by the roots of the auxiliary equation of the difference equation. Similar statements as in the last paragraph apply as to whether the solutions remain bounded or unbounded. For stability of differential equation multi-step methods, all roots of the auxiliary equation must lie within or on the unit circle of the complex plane, with no repeated roots on the unit circle. Note that one of the roots of the auxiliary equation is likely to be one, so that the ode $y' = 0$ has a convergent solution.

D. Software for Ordinary Differential Equations

Freely available software for ODE's is available from NETLIB,
<http://www.netlib.org/index.html>

a repository of free software. Most is in FORTRAN, but it is possible to use the f2c translator on it to produce C code. Most of the computation systems, MATLAB, Maple, Mathematica, have inbuilt integrators of various sophistication. It is most important that you use an integrator which monitors the error as it integrates. Something like Euler's method or a simple 4-th order Runge-Kutta, which are simple to program with a fixed step, can quite easily pass through a singularity without you knowing it. The results close to and past the singularity are probably rubbish. Likewise stiffness of ODE's cannot be found unless the error is being monitored.