ES/AM 346-0 Notes
Numerical Methods for Ordinary Differential Equations

Prof. David Chopp

Winter 2000

Contents

1 Preliminary Information

The basic problem to be solved is the *ordinary differential equation*

$$\frac{dy}{dx} = F(x, y), \quad y(x_0) = y_0. \tag{1}$$

We wish to solve for the function $y(x)$ on the interval $x_0 \le x \le x_f$. Here y may be a vector of length $n \ge 1$, and F must also be of dimension n. If $n > 1$, we say we have a *system of differential equations*.

1.1 Lipschitz condition

A *Lipschitz condition* for a function $F(x, y)$ is that for any u, v,

$$\|F(x, u) - F(x, v)\| \le L\|u - v\|$$

for some constant L.

Example 1.1:

If F is a scalar differentiable function, then let

$$L = \max_y \left| \frac{\partial F}{\partial y}(x, y) \right|.$$

Then we have

$$\begin{aligned} \frac{|F(x, u) - F(x, v)|}{|u - v|} &= \frac{|\frac{\partial F}{\partial y}(x, w)(u - v)|}{|u - v|}, \quad \text{for some } w \text{ between } u \text{ and } v \\ &\le \max_y \left| \frac{\partial F}{\partial y}(x, y) \right| = L \end{aligned}$$

Note that the Lipschitz condition does not require differentiability.

Example 1.2:

Consider $F(x, y) = |y|$ which is not differentiable, then

$$|F(x, u) - F(x, v)| = |(|u| - |v|)| = \begin{cases} |u - v| & \text{if } uv \ge 0 \\ |u + v| & \text{if } uv < 0 \end{cases}$$

If $uv < 0$, then $|u + v| \le |u - v|$, and therefore,

$$|F(x, u) - F(x, v)| \le |u - v|$$

Therefore, $F(x, y) = |y|$ is satisfies the Lipschitz condition with Lipschitz constant $L = 1$.

In the case of systems, this translates into the largest eigenvalue of the Jacobian matrix when the function F is differentiable.

Theorem 1 The equation

$$\frac{dy}{dx} = F(x, y), \quad y(x_0) = x_0$$

has a unique solution in any domain where $F(x, y)$ is a Lipschitz continuous function of x and y.

The Lipschitz condition bounds the difference between two solutions with different initial conditions. We will assume, unless otherwise noted, that F is always Lipschitz continuous.

1.2 Higher order ordinary differential equations

Note that we are assuming only one derivative in our discussion above. This is normal because any higher order equation can be reduced to a system of first order equations.

Example 1.3:

Consider the second order ordinary differential equation

$$y'' = F(x, y, y')$$

Let $u = y'$, then $u' = y'' = F(x, y, u)$ and we have the system

$$\begin{aligned} y' &= u \\ u' &= F(x, y, u) \end{aligned}$$

---

2 Basic Numerical Methods

2.1 Basic Strategy

The basic strategy for solving equation (1) is to give approximate values for the solution of equation (1) at discrete nodes on the interval $x_0 \le x \le x_f$. We will label the nodes by $x_0,\ldots,x_j,\ldots,x_N = x_f$. We define the *step size* to be $\Delta x_j = x_{j+1} - x_j$ and note that $x_f - x_0 = \sum_{j=0^{N-1}} \Delta x_j$. A fixed step size means that $\Delta x = \Delta x_j$ for all j and $x_f - x_0 = N\Delta x$. whe discussing convergence and stability, the measure of the step size will be determined by $\max_j \Delta x_j$.

Our goal is to compute values y_j such that $y(x_j) \approx y_j$.

For the following discussion, we will assume the step size is fixed unless otherwise noted. Most of the theory for fixed step sizes carries over to variable step sizes, but the proofs are just messier.

2.2 One-Step Methods

2.2.1 Taylor Series Methods

Our first class of numerical methods are derived from the Taylor polynomial approximation of the solution. First consider a scalar equation

$$y' = f(x, y), \quad y(x_0) = y_0$$

Suppose $y_j = y(x_j)$, then we want to approximate $y(x_{j+1})$ by y_{j+1}. The Taylor polynomial expansion gives

$$y_{j+1} \approx y(x_{j+1}) = y(x_j) + (x_{j+1} - x_j)y'(x_j) + \frac{1}{2}(x_{j+1} - x_j)^2 y''(x_j) + \cdots$$

Now, we know $y'(x_j) = F(x_j, y(x_j))$ from the differential equation, so we have

$$y(x_{j+1}) = y_j + \Delta x F(x_j, y_j) + \frac{1}{2}\Delta x^2 y''(x_j) + \cdots$$

If we cut off the expansion at the first order, then we get the famous Euler's method

$$y_{j+1} = y_j + \Delta x F(x_j, y_j)$$

To get more terms, one must compute $y''(x_j)$, $y'''(x_j)$ and so on. Thus,

$$\begin{aligned} y''(x) &= \frac{d}{dx}y'(x) \\ &= \frac{d}{dx}F(x,y) \\ &= F_x(x,y) + F_y(x,y)y'(x) \\ &= F_x(x,y) + F_y(x,y)F(x,y) \end{aligned}$$

This leads to the second order Taylor series method

$$y_{j+1} = y_j + \Delta x F(x_j, y_j) + \frac{1}{2}\Delta x^2 (F_x(x_j, y_j) + F_y(x_j, y_j)F(x_j, y_j))$$

Notice that we called this method second order. To determine the *order* of a method, we must compute the *local truncation error* of the method. To compute it, plug the exact solution into the numerical method and compute the error.

Example 2.1:

For Euler's method, the local truncation error τ is computed by taking Taylor expansions around the point $(x_j, y(x_j))$ to get

$$\begin{aligned} \tau &= y(x_{j+1}) - y(x_j) - \Delta x F(x_j, y(x_j)) \\ &= y(x_j) + \Delta x y'(x_j) + \frac{1}{2}\Delta x^2 y''(x_j) + O(\Delta x^3) - y(x_j) - \Delta x F(x_j, y(x_j)) \\ &= \frac{1}{2}\Delta x^2 y''(x_j) + O(\Delta x^3) \end{aligned}$$

This shows that $\tau = O(\Delta x^2)$. We will see later that one power of Δx is lost over the interval $x_0 \le x \le x_f$, so Euler's method is a first order method. This makes sense because it comes from the first order Taylor expansion.

Example 2.2:

For the second method,

$$\begin{aligned} \tau &= y(x_{j+1}) - y(x_j) - \Delta x F(x_j, y(x_j)) - \frac{1}{2}\Delta x^2 [F_x(x_j, y(x_j)) + F_y(x_j, y(x_j))F(x_j, y(x_j))] \\ &= y(x_j) + \Delta x y'(x_j) + \frac{1}{2}\Delta x^2 y''(x_j) + \frac{1}{6}\Delta x^3 y'''(x_j) + O(\Delta x^4) - y(x_j) - \Delta x F(x_j, y(x_j)) \\ &\quad - \frac{1}{2}\Delta x^2 [F_x(x_j, y(x_j)) + F_y(x_j, y(x_j))F(x_j, y(x_j))] \\ &= \frac{1}{6}\Delta x^3 y'''(x_j) + O(\Delta x^4) \end{aligned}$$

Thus, the local truncation error is $\tau = O(\Delta x^3)$ and the method is second order.

These are examples of *one-step methods* because y_{j+1} depends only on the value of y_j and not on any of the previous $y_0, \ldots, y_{j-1}$. It is an *explicit method* because y_{j+1} does not appear on the right hand side.

To generate an *implicit method*, we need only change the point around which the original Taylor expansion is taken.

$$y(x_j) = y(x_{j+1}) + (x_j - x_{j+1})y'(x_{j+1}) + \frac{1}{2}(x_j - x_{j+1})^2 y''(x_{j+1}) + O(\Delta x^3)$$

This leads to the second order implicit Taylor series method

$$y_j = y_{j+1} - \Delta x F(x_{j+1}, y_{j+1}) + \frac{1}{2}\Delta x^2 (F_x(x_{j+1}, y_{j+1}) + F_y(x_{j+1}, y_{j+1}) F(x_{j+1}, y_{j+1}))$$

The first order implicit Taylor series method is given by

$$y_{j+1} = y_j + \Delta x F(x_{j+1}, y_{j+1})$$

which is otherwise known as Backward Euler.

Implementing an implicit method such as this is difficult because this may be a non-linear system of equations. The advantage of these methods is their improved stability properties.

2.2.2 Methods based on quadrature rules

One problem with Taylor series methods is that they require the evaluation of one or more derivatives of the function F where $y' = F$. We will now look for methods that only use F itself.

We can derive a new set of methods by studying numerical quadrature methods. The relationship between quadratures and the differential equation can be seen below:

$$y'(x) = F(x, y(x)) \tag{2}$$

$$\int_{x_0}^{x_1} y'(x)\,dx = \int_{x_0}^{x_1} F(x, y(x))\,dx$$

$$y(x_1) - y(x_0) = \int_{x_0}^{x_1} F(x, y(x))\,dx$$

$$y(x_1) = y(x_0) + \int_{x_0}^{x_1} F(x, y(x))\,dx \tag{3}$$

We must approximate the last integral in order to derive a new numerical method. As a first approximation, suppose we take on the interval $x_0 \le x \le x_1$, the approximation $F(x, y(x)) \approx F(x_0, y(x_0))$. Then the integral becomes

$$\begin{aligned}\int_{x_0}^{x_1} F(x, y(x))\,dx &\approx \int_{x_0}^{x_1} F(x_0, y(x_0))\,dx \\ &= (x_1 - x_0) F(x_0, y(x_0))\end{aligned}$$

Plugging this approximation into equation (3) gives

$$\begin{aligned}y(x_1) &= y(x_0) + \int_{x_0}^{x_1} F(x, y(x))\,dx \\ &\approx y(x_0) + (x_1 - x_0) F(x_0, y(x_0))\end{aligned}$$

and we see we recover Euler's method.

We can improve our method by taking a better approximation for the integral in equation (3). The trapezoidal rule approximation for an integral has the formula

$$\int_{x_0}^{x_1} F(x, y(x))\,dx \approx \frac{1}{2}(x_1 - x_0)(F(x_0, y(x_0)) - F(x_1, y(x_1)))$$

Using this approximation, we get the method

$$y_1 = y_0 + \frac{\Delta x}{2}(F(x_0, y_0) + F(x_1, y_1))$$

where we have used the notation $y_j \approx y(x_j)$. Notice that this is an implicit method because of the presence of y_1 on the right hand side of the formula. We can convert this method to an explicit method by giving an approximation for y_1 on the right hand side. To do this, we can use Euler's method so that

$$y_1^* = y_0 + \Delta x F(x_0, y_0)$$

where $y_1^* \approx y_1$. Putting this into the quadrature method gives

$$\begin{aligned} y_1 &= y_0 + \frac{\Delta x}{2}(F(x_0, y_0) + F(x_1, y_1^*)) \\ &= y_0 + \frac{\Delta x}{2}(F(x_0, y_0) + F(x_1, y_0) + \Delta x F(x_0, y_0)) \end{aligned}$$

To reduce the number of function evaluations, this method is commonly reorganized into the form

$$\begin{aligned} k_1 &= F(x_0, y_0) \\ k_2 &= F(x_1, y_0 + \Delta x k_1) \\ y_1 &= y_0 + \frac{\Delta x}{2}(k_1 + k_2) \end{aligned}$$

This method is called the Improved Euler method.

Is this method really better? We should verify the order of convergence for this method by computing the local truncation error. To do this, first note that

$$\begin{aligned} F(x_1, y(x_0) + \Delta x F(x_0, y(x_0))) &= F(x_0 + \Delta x, y(x_0) + \Delta x F(x_0, y(x_0))) \\ &= F(x_0, y(x_0)) + \Delta x F_x(x_0, y(x_0)) + \Delta x F(x_0, y(x_0)) F_y(x_0, y(x_0)) + O(\Delta x^2) \end{aligned}$$

We therefore get

$$\begin{aligned} \tau &= y(x_0) + \Delta x y'(x_0) + \frac{\Delta x^2}{2} y''(x_0) + \frac{\Delta x^3}{6} y'''(x_0) + O(\Delta x^4) - y(x_0) - \frac{\Delta x}{2}(F(x_0, y(x_0)) + F(x_1 + y(x_0) + \Delta x F(x_0, y(x_0))) \\ &= \Delta x y'(x_0) + \frac{\Delta x^2}{2} y''(x_0) + \frac{\Delta x^3}{6} y'''(x_0) + O(\Delta x^4) - \frac{\Delta x}{2}(F + F + \Delta x F_x + \Delta x F F_y + O(\Delta x^2)) \\ &= \frac{\Delta x^3}{6} y'''(x_0) + O(\Delta x^3) \\ &= O(\Delta x^3) \end{aligned}$$

Therefore, the method is second order accurate.

A shorthand way to quickly check the order of accuracy of a method is to verify its accuracy for the differential equation $F(x, y) = \lambda y$. The exact solution for this equation is $y(x) = y_0 e^{\lambda(x - x_0)}$. Using this differential equation in the Improved Euler method gives

$$\begin{aligned} y_1 &= y_0 + \frac{\Delta x}{2}(\lambda y_0 + \lambda(y_0 + \Delta x \lambda y_0)) \\ &= y_0 + \frac{\Delta x}{2}(2\lambda y_0 + \Delta x \lambda^2 y_0) \\ &= y_0 + \lambda \Delta x y_0 + \frac{1}{2}\Delta x^2 \lambda^2 y_0 \\ &= \left(1 + \lambda \Delta x + \frac{1}{2}(\lambda \Delta x)^2\right) y_0 \\ &= e^{\lambda \Delta x} y_0 + O(\Delta x^3) \end{aligned}$$

Note that this is a necessary condition for the convergence rate, but not sufficient. It is possible to construct differential equations for which this shorthand method will claim a higher order method than it really is,

but it does give a maximum size on the actual order of the given method. In this example, Improved Euler is at best a second order method (which in fact it is).

The next best integration method is Simpson's Rule. The quadrature rule is given by

$$\int_{x_0}^{x_1} F(x, y(x))\,dx \approx \frac{1}{6}(x_1 - x_0)(F(x_0, y(x_0)) + 4F(x_{1/2}, y(x_{1/2})) + F(x_1, y(x_1)))$$

A first attempt at using Simpson's Rule for a time integration could go something like this:

$$\begin{aligned} y^*_{1/2} &= y_0 + \frac{\Delta x}{2} F(x_0, y_0) \\ y^*_1 &= y_0 + \Delta x F(x_{1/2}, y^*_{1/2}) \\ y_1 &= y_0 + \frac{\Delta x}{6}(F(x_0, y_0) + 4F(x_{1/2}, y^*_{1/2}) + F(x_1, y^*_1)) \end{aligned}$$

Let us use the shorthand method to check for the accuracy of this method. Assume $y' = \lambda y$, then

$$\begin{aligned} y_1 &= y_0 + \frac{\Delta x}{6}(\lambda y_0 + 4\lambda y^*_{1/2} + \lambda y^*_1) \\ &= y_0 + \frac{\Delta x}{6}\left(\lambda y_0 + 4\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right) + \lambda(y_0 + \Delta x \lambda y^*_{1/2})\right) \\ &= y_0 + \frac{\Delta x}{6}\left(\lambda y_0 + 4\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right) + \lambda\left(y_0 + \Delta x\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right)\right)\right) \\ &= y_0 + \lambda\Delta x y_0 + \frac{1}{2}(\lambda\Delta x)^2 y_0 + \frac{1}{12}(\lambda\Delta x)^3 y_0 \\ &= \left(1 + (\lambda\Delta x) + \frac{1}{2}(\lambda\Delta x)^2 + \frac{1}{12}(\lambda\Delta x)^3\right) y_0 \\ &= (e^{\lambda\Delta x} + O(\Delta x^3)) y_0 \end{aligned}$$

This shows that the method so constructed is no better than second order accurate.

Considering that we replaced a lower order quadrature with a higher order quadrature, we should expect better. One way to improve the calculation would be to take a better approximation for the intermediate value $y_{1/2}$. Instead of using Euler's method to get $y_{1/2}*$, perhaps we should use Improved Euler. The method then becomes

$$\begin{aligned} y^*_{1/2} &= y_0 + \frac{\Delta x}{2} F(x_0, y_0) \\ y^{**}_{1/2} &= y_0 + \frac{\Delta x}{2} F(x_{1/2}, y^*_{1/2}) \\ y^*_1 &= y_0 + \Delta x F(x_{1/2}, y^{**}_{1/2}) \\ y_1 &= y_0 + \frac{\Delta x}{6}(F(x_0, y_0) + 4F(x_1/2, y^{**}_{1/2}) + F(x_1, y^*_1)) \end{aligned}$$

If we use this algorithm, the shorthand accuracy check gives the result

$$y_1 = \left(1 + (\lambda\Delta x) + \frac{1}{2}(\lambda\Delta x)^2 + \frac{1}{4}(\lambda\Delta x)^3\right) y_0$$

Notice that in the previous case, the cubic term had a coefficient of $\frac{1}{12}$ which is too small, and this time it is $\frac{1}{4}$ which is too big (of course what we really want is the coefficient $\frac{1}{6}$).

In light of this discrepancy, let us take a different approach and take a linear combination of the two methods to try to get the right coefficient for the cubic term. The linear combination of the two methods amounts to taking

$$y_1 = y_0 + \frac{\Delta x}{6}(F(x_0, y_0) + \alpha F(x_{1/2}, y^*_{1/2}) + \beta F(x_{1/2}, y^{**}_{1/2}) + F(x_1, y^*_1))$$

Let's again look at the shorthand accuracy check.

$$\begin{aligned}
y_1 &= y_0 + \frac{\Delta x}{6}(\lambda y_0 + \alpha\lambda y^*_{1/2} + \beta\lambda y^{**}_{1/2} + \lambda y^*_1) \\
&= y_0 + \frac{\Delta x}{6}\left(\lambda y_0 + \alpha\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right) + \beta\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y^*_{1/2}\right) + \lambda\left(y_0 + \Delta x\lambda y^{**}_{1/2}\right)\right) \\
&= y_0 + \frac{\Delta x}{6}\left(\lambda y_0 + \alpha\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right) + \beta\lambda\left(y_0 + \frac{\Delta x}{2}\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right)\right)\right. \\
&\qquad\qquad \left. + \lambda\left(y_0 + \Delta x\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y^*_{1/2}\right)\right)\right) \\
&= y_0 + \frac{\Delta x}{6}\left(\lambda y_0 + \alpha\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right) + \beta\lambda\left(y_0 + \frac{\Delta x}{2}\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right)\right)\right. \\
&\qquad\qquad \left. + \lambda\left(y_0 + \Delta x\lambda\left(y_0 + \frac{\Delta x}{2}\lambda\left(y_0 + \frac{\Delta x}{2}\lambda y_0\right)\right)\right)\right) \\
&= \left(1 + \frac{1}{6}\lambda\Delta x(2 + \alpha + \beta) + \frac{1}{6}(\lambda\Delta x)^2\left(\frac{1}{2}\alpha + \frac{1}{2}\beta + 1\right) + \frac{1}{6}(\lambda\Delta x)^3\left(\frac{1}{4}\beta + \frac{1}{2}\right) + \frac{1}{4}(\lambda\Delta x)^4\right) y_0
\end{aligned}$$

To get the best approximation of $e^{\lambda\Delta x}$, we must have solve the following equations

$$\begin{aligned}
\frac{1}{6}(2 + \alpha + \beta) &= 1 \\
\frac{1}{6}\left(\frac{1}{2}\alpha + \frac{1}{2}\beta + 1\right) &= \frac{1}{2} \\
\frac{1}{6}\left(\frac{1}{4}\beta + \frac{1}{2}\right) &= \frac{1}{6}
\end{aligned}$$

The first two equations turn out to be the same, and the third requires $\beta = 2$ so that the solution is $\alpha = \beta = 2$. This gives the equation

$$y_1 = \left(1 + (\lambda\Delta x) + \frac{1}{2}(\lambda\Delta x)^2 + \frac{1}{6}(\lambda\Delta x)^3 + \frac{1}{24}(\lambda\Delta x)^4\right) y_0 = (e^{\lambda\Delta x} + O(\Delta x^5))y_0$$

which indicates the method is at best fourth order.

Reorganizing this new method into a more compact form, we get

$$\begin{aligned}
k_1 &= F(x_0, y_0) \\
k_2 &= F\left(x_0 + \frac{\Delta x}{2}, y_0 + \frac{\Delta x}{2}k_1\right) \\
k_3 &= F\left(x_0 + \frac{\Delta x}{2}, y_0 + \frac{\Delta x}{2}k_2\right) \\
k_4 &= F(x_0 + \Delta x, y_0 + \Delta x k_3) \\
y_1 &= y_0 + \frac{\Delta x}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}$$

This method is called Runge-Kutta 4 because it is a fourth order accurate method of the Runge-Kutta class which we will describe next.

2.2.3 Runge-Kutta methods

The Runge-Kutta class of numerical methods are one-step methods which can be constructed of any order accuracy. The general description is as follows. To take one step from x_j to x_{j+1}, let

$$X_0 = x_j, \quad Y_0 = y_j, \quad F_0 = F(X_0, Y_0)$$

Then for $k = 1,\ldots,s$,

$$\begin{aligned} X_k &= x_j + \alpha_k \Delta x \\ Y_k &= y_j + \Delta x \sum_{m=0}^{s} \beta_{k,m} F_m \\ F_k &= F(X_k, Y_k) \\ y_{j+1} &= y_j + \Delta x \sum_{k=0}^{s} \gamma_k F_k \end{aligned}$$

Example 2.3:

Euler's method is a first order Runge-Kutta method where $s = 0$, $\gamma_0 = 1$.

Example 2.4:

The fourth-order Runge-Kutta method described earlier uses the parameters $s = 3$, $\alpha_1 = 1/2$, $\alpha_2 = 1/2$, $\alpha_3 = 1$, $\beta_{1,0} = 1/2$, $\beta_{2,1} = 1/2$, $\beta_{3,2} = 1$, $\gamma_0 = 1/6$, $\gamma_1 = \gamma_2 = 1/3$, $\gamma_3 = 1/6$.

We can see from the last example that writing down all the parameters for a specific Runge-Kutta method can be cumbersome. To organize the coefficients we can use a Butcher array where the coefficients are arranged in a table Each row of the array is a single *stage* of the Runge-Kutta method

$$\begin{array}{r|ccccc} 0 = \alpha_0 & & & & & \\ \alpha_1 & \beta_{1,0} & & & & \\ \alpha_2 & \beta_{2,0} & \beta_{2,1} & & & \\ \vdots & \vdots & & \ddots & & \\ \alpha_s & \beta_{s,0} & \cdots & \cdots & \beta_{s,s-1} & \\ \hline & \gamma_0 & \cdots & \cdots & \gamma_{s-1} & \gamma_s \end{array}$$

Example 2.5:

The fourth order Runge-Kutta method in the earlier example can be written as a four stage Butcher array shown in Table 1

$$\begin{array}{r|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Table 1: Butcher array for a Fourth order Runge-Kutta method

Note that the objective in building a Runge-Kutta method is to get the most accuracy with the fewest uses of the function F. To get an optimal method, we must choose the α's, β's, and γ's so that the local truncation error has the highest possible exponent (i.e. is maximally accurate). Consider the Runge-Kutta method To get the optimal method, we want to choose α_1, $\beta_{1,0}$, γ_0, and γ_1 so that the local truncation error

$$X_0 = x_j, \quad Y_0 = y_j, \quad F_0 = F(X_0, Y_0)$$
$$X_1 = x_j + \alpha_1 \Delta x, \quad Y_1 = y_j + \Delta x \beta_{1,0} F_0, \quad F_1 = F(X_1, Y_1)$$
$$y_{j+1} = y_j + \Delta x[\gamma_0 F_0 + \gamma_1 F_1]$$

$0 = \alpha_0$		
α_1	$\beta_{1,0}$	
	γ_0	γ_1

has the highest possible order. Computing the local truncation error, we get

$$\begin{aligned}
\tau &= y(x_j) + \Delta x y'(x_j) + \frac{1}{2}\Delta x^2 y''(x_j) + \frac{1}{6}\Delta x^3 y'''(x_j) + O(\Delta x^4) \\
&\quad - y(x_j) - \Delta x[\gamma_0 F(x_j, y_j) + \gamma_1 F(x_j + \alpha_1 \Delta x, y_j + \Delta x \beta_{1,0} F(x_j, y_j)] \\
&= \Delta x F(x_j, y_j) + \frac{\Delta x^2}{2}(F_x + F_y F) + \frac{\Delta x^3}{6}(F_{xx} + F_{xy}F + F_{xy}F + F_{yy}F^2 + F_y(F_x + F_y F)) \\
&\quad - \Delta x \left(\gamma_0 F + \gamma_1 \left(F + F_x \alpha_1 \Delta x + F_y \Delta x \beta_{1,0} F + \frac{1}{2} F_{xx} \alpha_1^2 \Delta x^2 + F_{xy} \alpha_1 \beta_{1,0} \Delta x^2 F + \frac{1}{2} F_{yy} \Delta x^2 \beta_{1,0}^2 F^2 \right) \right) \\
&\quad + O(\Delta x^4)
\end{aligned}$$

Now look at the coefficients of the Δx and we get

Δx^1: $F - (\gamma_0 + \gamma_1)F = 0$

Δx^2: $\frac{1}{2}(F_x + F_y F) - \gamma_1 \alpha_1 F_x - \gamma_1 \beta_{1,0} F_y F = 0$

Δx^3: $\frac{1}{6}(F_{xx} + 2F_{xy}F + F_{yy}F^2 + f_x F_y + F_y^2 F) - \frac{1}{2}\gamma_1 \alpha_1^2 F_{xx} - \alpha_1 \gamma_1 \beta_{1,0} F_{xy} F - \frac{1}{2}\gamma_1 \beta_{1,0}^2 F_{yy} F^2 = 0$

Knocking off these terms, we get the equations

Δx^1: $\gamma_0 + \gamma_1 = 0$

Δx^2: $\frac{1}{2} - \gamma_1 \alpha_1 = 0$, $\frac{1}{2} - \gamma_1 \beta_{1,0} = 0$

Δx^3: $\frac{1}{6} - \frac{1}{2}\gamma_1 \alpha_1^2 = 0$, $\frac{1}{3} - \gamma_1 \alpha_1 \beta_{1,0} = 0$, $\frac{1}{6} - \frac{1}{2}\gamma_1 \beta_{1,0}^2 = 0$, ???

Since the Δx^3 term cannot be eliminated with the parameters given, then the method will be at most second order provided the first two sets of equations are satisfied. The equations for Δx and for Δx^2 can be solved to get

$$\gamma_0 + \gamma_1 = 1, \quad \gamma_1 = \frac{1}{2\alpha_1}, \quad \alpha_1 = \beta_{1,0} \tag{4}$$

Clearly, the improved Euler method has $\gamma_0 = \gamma_1 = 1/2$, $\alpha_1 = 1$, and $\beta_{1,0} = 1$. Other solutions also exist such as $\alpha_1 = 3/4$, $\beta_{1,0} = 3/4$, $\gamma_1 = 2/3$, $\gamma_0 = 1/3$. All methods which solve equations (4) are optimal in the sense that the highest order method possible is achieved with the given number of function evaluations.

Note that to satisfy the equations (4), it must be that $\gamma_1 \neq 0$. If we let $\gamma_1 = \theta$, then all the parameters can be written as a function of θ.

$$\begin{aligned}
\gamma_1 &= \theta \\
\gamma_0 &= 1 - \gamma_1 = 1 - \theta \\
\alpha_1 &= \frac{1}{2\gamma_1} = \frac{1}{2\theta} \\
\beta_{1,0} &= \alpha_1 = \frac{1}{2\theta}
\end{aligned}$$

This is now a one-parameter family of second order Runge-Kutta methods. Which value of θ is best? Look at the local truncation error and we see that

$$\tau = \Delta x^3 \left\{ \frac{1}{6}[F_{xx} + 2F_{xy}F + F_{yy}F^2 + F_xF_y + F_y^2F] - \frac{1}{2}\frac{1}{4\theta}F_{xx} - \frac{1}{4\theta}F_{xy}F - \frac{1}{2}\frac{1}{4\theta}F_{yy}F^2 \right\}$$

We want to choose θ so as to reduce the error as much as possible. For example, take θ so that $\frac{1}{6} - \frac{1}{8\theta} = 0$ implies that $\theta = \frac{3}{4}$.

In general, for higher order Runge-Kutta methods, the number of parameters can be very large. Determining the optimal parameters depends on the function F, so there is not in general a unique choice for the parameters that is optimal for all problems.

2.3 Multistep Methods

2.3.1 Adams-Bashforth and Adams-Moulton methods

The Runge-Kutta and Taylor series classes of methods all advance one time step by only requiring knowledge of one previous time step, i.e. y_{j+1} depends only on y_j. Recall that the Runge-Kutta class was motivated by trying to use numerical quadrature rules to generate methods for ordinary differential equations. We can instead use quadrature over two or more previous steps to also get a higher order method.

The most common explicit methods of this type are the Adams-Bashforth class. Suppose we are given n previous time steps, y_j, y_{j-1}, ..., y_{j-n+1}. Define $P_n(x)$ to be the interpolating polynomial such that

$$P_n(x_q) = F(x_q, y_q) \text{ for } q = j, \ldots, j-n+1.$$

We can write a formula for $P_n(x)$ directly using Lagrange polynomials.

$$P_n(x) = \sum_{k=1}^{n} F(x_{j+1-k}, y_{j+1-k}) L_k(x)$$

$$L_k(x) = \prod_{\substack{m=1 \\ m \neq k}}^{n} \frac{x - x_{j+1-m}}{x_{j+1-k} - x_{j+1-m}}$$

Note that $L_k(x)$ is a polynomial such that

$$L_k(x_q) = \begin{cases} 1 & \text{if } x_q = x_{j+1-k} \\ 0 & \text{if } x_q \neq x_{j+1-k} \end{cases}$$

It is this property that makes constructing $P_n(x)$ straightforward.

We now can construct a method based on the quadrature of $P_n(x)$.

$$\begin{aligned} y_{j+1} &= y_j + \int_{x_j}^{x_j+\Delta x} P_n(x)\, dx \\ &= y_j + \int_{x_j}^{x_j+\Delta x} \sum_{k=1}^{n} F(x_{j+1-k}, y_{j+1-k}) L_k(x)\, dx \\ &= y_j + \sum_{k=1}^{n} F(x_{j+1-k}, y_{j+1-k}) \int_{x_j}^{x_j+\Delta x} L_k(x)\, dx \end{aligned}$$

This formula can be simplified if we assume a fixed step size, $x_q = x_0 + q\Delta x$, and hence

$$L_k(x) = \prod_{\substack{m=1 \\ m \neq k}}^{n} \frac{x - x_{j+1-m}}{(m-k)\Delta x}$$

and the integral becomes

$$\int_{x_j}^{x_j+\Delta x} L_k(x)\,dx = \Delta x \int_0^1 \prod_{\substack{m=1\\m\neq k}}^{n} \frac{\Delta x(t+m-1)}{(m-k)\Delta x}\,dt$$
$$= \Delta x \int_0^1 \prod_{\substack{m=1\\m\neq k}}^{n} \frac{t+m-1}{m-k}\,dt$$

We can now derive the coefficients for the Adams-Bashforth class of methods.

Example 2.6:

Adams-Bashforth-1. In this case, we only go back to x_j:

$$\begin{aligned} P_1(x) &= F(x_j, y_j) \\ y_{j+1} &= y_j + \int_{x_j}^{x_j+\Delta x} P_1(x)\,dx \\ &= y_j + \int_{x_j}^{x_j+\Delta x} F(x_j, y_j)\,dx \\ &= y_j + \Delta x F(x_j, y_j) \end{aligned}$$

and we see that we again recover Euler's method.

Example 2.7:

Adams-Bashforth-2.

$$\begin{aligned} P_2(x) &= \left(\frac{x-x_{j-1}}{x_j-x_{j-1}}\right) F(x_j, y_j) + \left(\frac{x-x_j}{x_{j-1}-x_j}\right) F(x_{j-1}, y_{j-1}) \\ &= \frac{1}{\Delta x}(x-x_{j-1})F(x_j, y_j) - \frac{1}{\Delta x}(x-x_j)F(x_{j-1}, y_{j-1}) \\ y_{j+1} &= y_j + \int_{x_j}^{x_j+\Delta x} \frac{1}{\Delta x}(x-x_{j-1})F_j - \frac{1}{\Delta x}(x-x_j)F_{j-1}\,dx \\ &= y_j + \left[\frac{1}{2}\frac{1}{\Delta x}(x-x_{j-1})^2 F_j - \frac{1}{2}\frac{1}{\Delta x}(x-x_j)^2 F_{j-1}\right]_{x_j}^{x_j+\Delta x} \\ &= y_j + \left[\frac{1}{2}\frac{1}{\Delta x}4\Delta x^2 F_j - \frac{1}{2}\frac{1}{\Delta x}\Delta x^2 F_{j-1} - \frac{1}{2}\frac{1}{\Delta x}\Delta x^2 F_j + 0\right] \\ y_{j+1} &= y_j + \frac{3}{2}\Delta x F_j - \frac{1}{2}\Delta x F_{j-1} \end{aligned}$$

To compute Adams-Bashforth-3, let's use the simplified formulas for the integrals:

$$\begin{aligned}\int_{x_j}^{x_j+\Delta x} L_1(x)\,dx &= \Delta x \int_0^1 \left(\frac{t+1}{2-1}\right)\left(\frac{t+2}{3-1}\right)\,dt \\ &= \Delta x \int_0^1 \frac{1}{2}(t^2+3t+2)\,dt \\ &= \frac{\Delta x}{2}\left(\frac{1}{3}t^3+\frac{3}{2}t^2+2t\right)_0^1 \\ &= \frac{\Delta x}{2}\frac{23}{6} = \frac{23}{12}\,dx\end{aligned}$$

$$\begin{aligned}\int_{x_j}^{x_j+\Delta x} L_2(x)\,dx &= \Delta x \int_0^1 \left(\frac{t}{1-2}\right)\left(\frac{t+2}{3-2}\right)\,dt \\ &= -\Delta x \int_0^1 t^2+2t\,dt \\ &= -\Delta x\left(\frac{1}{3}t^3+t^2\right)_0^1 \\ &= -\frac{4}{3}\Delta x\end{aligned}$$

$$\begin{aligned}\int_{x_j}^{x_j+\Delta x} L_3(x)\,dx &= \Delta x \int_0^1 \left(\frac{t}{1-3}\right)\left(\frac{t+1}{2-3}\right)\,dt \\ &= \frac{\Delta x}{2}\left(\frac{1}{3}t^3+\frac{1}{2}t^2\right)_0^1 \\ &= \frac{5}{12}\Delta x\end{aligned}$$

and therefore

$$y_{j+1} = y_j + \frac{23}{12}\Delta x F_j - \frac{4}{3}\Delta x F_{j-1} + \frac{5}{12}\Delta x F_{j-2}$$

Let us now check the truncation error for this method.

$$\begin{aligned}\tau &= y + \Delta x y' + \frac{1}{2}\Delta x^2 y'' + \frac{1}{6}\Delta x^3 y''' + O(\Delta x^4) \\ &\quad - y - \frac{23}{12}\Delta x F + \frac{4}{3}\Delta x F(x_{j-1},y_{j-1}) - \frac{5}{12}\Delta x F(x_{j-2},y_{j-2}) \\ &= \frac{1}{2}\Delta x^2 y'' + \frac{1}{6}\Delta x^3 y''' + O(\Delta x^4) - \frac{11}{12}\Delta x F \\ &\quad + \frac{4}{3}\left(F - \Delta x F_x + (y_{j-1}-y_j)F_y + \frac{1}{2}\Delta x^2 F_{xx} - \Delta x(y_{j-1}-y_j)F_{xy} + \frac{1}{2}(y_{j-1}-y_j)^2 F_{yy} + O(\Delta x^3)\right) \\ &\quad - \frac{5}{12}\Delta x\left(F - 2\Delta x F_x + (y_{j-2}-y_j)F_y + \frac{1}{2}(2\Delta x)^2 F_{xx} - 2\Delta x(y_{j-2}-y_j)F_{xy} + \frac{1}{2}(y_{j-2}-y_j)^2 F_{yy} + O(\Delta x^3)\right)\end{aligned}$$

Now note that

$$\begin{aligned}y_{y-1} - y_j &= -\Delta x y' + \frac{1}{2}\Delta x^2 y'' + O(\Delta x^3) \\ y_{j-2} - y_j &= -2\Delta x y' + \frac{1}{2}(2\Delta x)^2 y'' + O(\Delta x^3)\end{aligned}$$

Thus,

$$\begin{aligned}
\tau &= \frac{1}{2}\Delta x^2 y'' + \frac{1}{6}\Delta x^3 y''' + \frac{4}{3}\Delta x \left[-\Delta x F_x + \left(-\Delta x y' + \frac{1}{2}\Delta x^2 y'' \right) F_y + \frac{1}{2}\Delta x^2 F_{xx} \right. \\
&\qquad \left. - \Delta x \left(-\Delta x y' + \frac{1}{2}\Delta x^2 y'' \right) F_{xy} + \frac{1}{2}\left(-\Delta x y' + \frac{1}{2}\Delta x^2 y'' \right)^2 F_{yy} \right] \\
&\quad - \frac{5}{12}\Delta x \left[-2\Delta x F_x + (-2\Delta x y' + 2\Delta x^2 y'') F_y + 2\Delta x^2 F_{xx} \right. \\
&\qquad \left. - 2\Delta x(-2\Delta x y' + 2\Delta x^2 y'') F_{xy} + \frac{1}{2}(-2\Delta x y' + 2\Delta x^2 y'')^2 F_{yy} \right] \\
&= \Delta x^2 \left[\frac{1}{2}y'' - \frac{4}{3}F_x - \frac{4}{3}FF_y + \frac{5}{6}F_x + \frac{5}{6}FF_y \right] \\
&\quad + \Delta x^3 \left[\frac{1}{6}y''' + \frac{2}{3}y''F_y + \frac{2}{3}F_{xx} + \frac{4}{3}y'F_{xy} + \frac{2}{3}y'^2 F_{yy} - \frac{5}{6}y''F_y - \frac{5}{6}F_{xx} - \frac{5}{3}y'F_{xy} - \frac{5}{6}y'^2 F_{yy} \right] + O(\Delta x^4) \\
&= \Delta x^2 \left[\frac{1}{2}y'' - \frac{1}{2}y'' \right] + \frac{1}{6}\Delta x^3 \left[y''' - (F_x + FF_y)F_y - F_{xx} - 2FF_{xy} - F^2 F_{yy} \right] \\
&= O(\Delta x^4)
\end{aligned}$$

The Adams-Moulton family of methods are implicit methods, but they are constructed in a similar way. This time, the polynomial is such that

$$P_n(x_{j+1-k}) = F_{j+1-k}, \text{ for } k = 0,\ldots,n-1.$$

We again integrate this polynomial to get

$$y_{j+1} = y_j + \int_{x_j}^{x_j+\Delta x} P_n(x)\,dx$$

Again, we use the Lagrange polynomials so that

$$P_n(x) = \sum_{k=0}^{n-1} F_{j+1-k} L_k(x)$$

where

$$L_k(x) = \prod_{\substack{m=0 \\ m\neq k}}^{n-1} \frac{x - x_{j+1-m}}{x_{j+1-k} - x_{j+1-m}}$$

We then get

$$y_{j+1} = y_j + \sum_{k=0}^{n-1} F_{j+1-k} \int_{x_j}^{x_j+\Delta x} L_k(x)\,dx$$

Example 2.8:

Adams-Moulton 1: We have

$$\begin{aligned}
P_1(x) &= F_{j+1} \\
y_{j+1} &= y_j + \int_{x_j}^{x_j+\Delta x} F_{j+1}\,dx \\
&= y_j + \Delta x F_(x_{j+1}, y_{j+1})
\end{aligned}$$

and we see that we recover the backward Euler method.

2.3.2 Backward Difference Formulas

Note that the Euler methods can also be derived by looking at forward and backward finite differences:

$$\frac{y(x_{j+1}) - y(x_j)}{x_{j+1} - x_j} \approx y'(x_j) = F(x_j, y_j)$$

Solving for y_{j+1} we get

$$y_{j+1} = y_j + \Delta x F_j$$

which is again Euler's method. This approximation could equally be applied to give an approximation for $y'(x_{j+1})$ so that

$$\frac{y(x_{j+1}) - y(x_j)}{x_{j+1} - x_j} \approx y'(x_{j+1}) = F(x_{j+1}, y_{j+1})$$

Thus, we get the method

$$y_{j+1} = y_j + \Delta x F_{j+1}$$

which is backward Euler.

Now suppose we take a better approximation for the derivative. Let $P_n(x)$ be a function which interpolates the values of y (not F as for the Adams methods). Then

$$P_n(x_{j+1-k}) = y_{j+1-k},, \text{ for } k = 0,\ldots,n.$$

and we require $P_n'(x_{j+1}) = F(x_{j+1}, P_n(x_{j+1})) = F(x_{j+1}, y_{j+1})$. Now,

$$L_k(x) = \prod_{\substack{m=0 \\ m\neq k}}^{n} \frac{x - x_{j+1-m}}{x_{j+1-k} - x_{j+1-m}}, \text{ for } k = 0,\ldots,n$$

and hence

$$P_n(x) = \sum_{k=0}^{n} y_{j+1-k} L_k(x)$$

$$P_n'(x_{j+1}) = \sum_{k=0}^{n} y_{j+1-k} L_k'(x_{j+1}) = F(x_{j+1}, y_{j+1})$$

Recall that

$$\begin{aligned}
L_k(x) &= \prod_{\substack{m=0 \\ m\neq k}}^{n} \frac{x - (x_j + (1-m)\Delta x)}{x_j + (1-k)\Delta x - (x_j + (1-m)\Delta x)} \\
&= \prod_{\substack{m=0 \\ m\neq k}}^{n} \frac{x - x_j + (m-1)\Delta x}{(m-k)\Delta x)} \\
\frac{d}{dx} L_k(x)\Big|_{x_{j+1}} &= \sum_{\substack{\ell=0 \\ \ell\neq k}}^{n} \frac{1}{(\ell-k)\Delta x} \prod_{\substack{m=0 \\ m\neq k,\ell}}^{n} \frac{x - x_j + (m-1)\Delta x}{(m-k)\Delta x}\Bigg|_{x_{j+1}} \\
&= \sum_{\substack{\ell=0 \\ \ell\neq k}}^{n} \frac{1}{(\ell-k)\Delta x} \prod_{\substack{m=0 \\ m\neq k,\ell}}^{n} \frac{m}{(m-k)}\Bigg|_{x_{j+1}} \\
&= \frac{1}{\Delta x}\alpha_k
\end{aligned}$$

Now, we get

$$\alpha_0 y_{j+1} + \alpha_1 y_1 + \cdots + \alpha_n y_{j+1-n} = \Delta x F(x_{j+1}, y_{j+1})$$

For $n = 1$, we again get backward Euler. For $n = 2$,

$$\alpha_0 = \sum_{\substack{\ell=0 \\ \ell\neq 0}}^{2} \frac{1}{(\ell - 0)} \prod_{\substack{m=0 \\ m\neq 0,\ell}}^{2} \frac{m}{m-0} = \sum_{\ell=1,2} \frac{1}{\ell} = \frac{3}{2}$$

$$\alpha_1 = \sum_{\substack{\ell=0 \\ \ell\neq 1}}^{2} \frac{1}{(\ell - 1)} \prod_{\substack{m=0 \\ m\neq 1,\ell}}^{2} \frac{m}{m-1} = -2$$

$$\alpha_2 = \sum_{\substack{\ell=0 \\ \ell\neq 2}}^{2} \frac{1}{(\ell - 2)} \prod_{\substack{m=0 \\ m\neq 2,\ell}}^{2} \frac{m}{m-2} = \frac{1}{2}$$

Therefore, the method is

$$\frac{3}{2} y_{j+1} - 2y_j + \frac{1}{2} y_{j-1} = \Delta x F(x_{j+1}, y_{j+1})$$

Backward difference formula methods will appear again when we discuss stiff equations.

2.3.3 General Linear Multistep Methods

We have seen three different families of linear multistep methodswhich are derived in different ways. All these methods can be encapsulated in the general form

$$\sum_{i=0}^{k} \alpha_i y_{j+1-i} = \Delta x \sum_{i=0}^{k} \beta_i F(x_{j+1-i}, y_{j+1-i}) = \Delta x \sum_{i=0}^{k} \beta_i F_{j+1-i}$$

We require $\alpha_0 \neq 0$ so that we can advance the method. The method is explicit if $\beta_0 = 0$ and implicit if $\beta_0 \neq 0$.

The characteristic polynomial for this method is the polynomial with the α_i's as coefficients:

$$\rho(\theta) = \alpha_0 \theta^k + \alpha_1 \theta^{k-1} + \cdots + \alpha_k \theta^0 = \sum_{i=0}^{k} \alpha_i \theta^{k-i}$$

Note that the choices of the α's and β's are not unique. To normalize them, we assume $\rho'(1) = 1$.

Example 2.9:

All Adams-Bashforth methods have the form

$$y_{j+1} = y_j + \sum_{i=0}^{k} \beta_i F_{j-i}$$

hence $\alpha_0 = 1$, $\alpha_1 = -1$. Therefore,

$$\rho(\theta) = \theta^k - \theta^{k-1}$$
$$\rho'(\theta) = k\theta^{k-1} - (k-1)\theta^{k-2}$$
$$\rho'(1) = k - (k-1) = 1$$

so the Adams-Bashforth methods are normalized.

Example 2.10:

A second order backward difference formula method is

$$\frac{3}{2}y_{j+1} - 2y_j + \frac{1}{2}y_{j-1} = \Delta x F_{j+1}$$

In this case,

$$\begin{aligned}\rho(\theta) &= \frac{3}{2}\theta^2 - 2\theta + \frac{1}{2}\\ \rho'(\theta) &= 3\theta - 2\\ \rho'(1) &= 1\end{aligned}$$

so this method is normalized.

2.3.4 Predictor-Corrector Methods

We have seen both explicit and implicit linear multistep methods. Explicit methods are easier to program and implicit methods allow for learger time steps. Can we find a middle ground? A predictor-correcteor pair is a combination of an explicit method (predictor) and an implicit method (corrector).

Example 2.11:

A predictor-corrector pair can be made from Adams-Bashforth 1 and Adams-Moulton 2:

$$y^*_{j+1} = y_j + \Delta x F(x_j, y_j) \tag{AB1}$$

$$y_{j+1} = y_j + \frac{\Delta x}{2}(F(x_j, y_j) + F(x_{j+1}, y^*_{j+1})) \tag{AM2}$$

In fact, this method we have already seen as Runge-Kutta 2, aka improved Euler.

A popular combination is to use Adams-Bashforth methods as predictor and Adams-Moulton as corrector where the Adams-Moulton corrector is one higher order. This is because the two methods use the same number of earlier time steps.

Example 2.12:

Notice that if we combine Adams-Bashforth-2 with Adams-Moulton-3, the two methods use only two past values of F, namely F_j and F_{j-1}.

$$y^*_{j+1} = y_j + \frac{3}{2}F_j - \frac{1}{2}F_{j-1} \tag{AB2}$$

$$y_{j+1} = y_j + \frac{5}{12}F^*_{j+1} + \frac{8}{12}F_j - \frac{1}{12}F_{j-1} \tag{AM3}$$

Next, we consider the order of a predictor-corrector pair. Suppose the two methods are given below:

$$\text{Predictor:} \quad \sum_{i=0}^{k} \alpha_i^* y_{j+i-i} = \Delta x \sum_{i=1}^{k} \beta_i^* F_{j+1-i}$$

$$\text{Corrector:} \quad \sum_{i=0}^{k} \alpha_i y_{j+i-i} = \Delta x \sum_{i=1}^{k} \beta_i F_{j+1-i}$$

Suppose we have the local truncation errors for each method given by

$$\tau_P = y(x_{j+1}) + \sum_{i=1}^{k} \left[-\frac{\alpha_i^*}{\alpha_0^*} y(x_{j+1-i}) + \Delta x \frac{\beta_i^*}{\alpha_0^*} F(x_{j+1-i}, y(x_{j+1-i})) \right]$$

$$\tau_C = y(x_{j+1}) + \sum_{i=1}^{k} \left[-\frac{\alpha_i}{\alpha_0} y(x_{j+1-i}) + \Delta x \frac{\beta_i}{\alpha_0} F(x_{j+1-i}, y_{j+1-i}) \right] + \Delta x \frac{\beta_0}{\alpha_0} F(x_{j+1}, y(x_{j+1}))$$

We now take this information to compute the truncation error of the combination method.

$$\begin{aligned}
\tau &= \tau_C + \Delta x \frac{\beta_0}{\alpha_0} F(x_{j+1}, y(x_{j+1}) - \tau_P) - \Delta x \frac{\beta_0}{\alpha_0} F(x_{j+1}, y(x_{j+1})) \\
&= \tau_C + \Delta x \frac{\beta_0}{\alpha_0} [F(x_{j+1}, y(x_{j+1})) - \tau_P F_y(x_{j+1}, y(x_{j+1}))] - \Delta x \frac{\beta_0}{\alpha_0} F(x_{j+1}, y(x_{j+1})) \\
&= \tau_C - \Delta x \frac{\beta_0}{\alpha_0} F_y(x_{j+1}, y(x_{j+1})) \tau_P \\
&= O(\tau_C) + O(\Delta x \tau_P)
\end{aligned}$$

Therefore, we get error of order equal to the corrector provided the predictor is at least one less order than the corrector.

2.4 Implementation of Implicit Methods

Consider the linear multistep method

$$y_{j+1} = \Delta x \frac{\beta_0}{\alpha_0} F(x_{j+1}, y_{j+1}) + \sum_{i=1}^{k} \left[\Delta x \frac{\beta_i}{\alpha_0} F_{j+1-i} - \frac{\alpha_i}{\alpha_0} y_{j+1-i} \right]$$

When $\beta_0 \neq 0$, this method is implicit and we must solve this equation for y_{j+1} at each time step. To focus on that solution, we can rewrite the method in the form

$$v = \Delta x \gamma F(v) + \Delta x \delta + \mu \tag{5}$$

Our objective is now to solve this equation for v. Let us assume that F is Lipschitz continuous in y so that

$$||F(v) - F(w)|| \leq L||v - w||$$

for some L. This assumption gaurantees equation (5) has a unique solution for a sufficiently small Δx.

An iterative method for solving equation (5) is to set $u_0 = y_j$, then iterate by

$$u_{m+1} = \Delta x \gamma F(u_m) + \Delta x \delta + \mu$$

If V is the exact solution, then we have

$$\begin{aligned} ||v - u_{m+1}|| &= ||\Delta x\gamma F(v) + \Delta x\delta + \mu - \Delta x\gamma F(u_m) - \Delta x\delta - \mu|| \\ &= \Delta x|\gamma|\, ||F(v) - F(u_m)|| \\ &\le \Delta x|\gamma|L||v - u_m|| \le \cdots \le (\Delta x|\gamma|L)^{m+1}||v - u_0|| \end{aligned}$$

Therefore, this converges provided $\Delta x|\gamma|L < 1$, i.e. $\Delta x < 1/|\gamma|L$.

This shows that even though an implicit method can in theory take arbitrarily large time steps, solving the system of equations imposes its own time step restriction which can be even more severe than for an explicit method.

3 Convergence and Stability

We will now establish the general heuristic theorem:

Theorem 2 Consistency plus stability implies convergence.

3.1 One-step Methods

Any one-step method can be written in the form

$$y_{j+1} = y_j + \Delta x\Phi(x_j, y_j, \Delta x)$$

where Φ is a function which also depends on F. We wish to show that a method constructed in this way converges to the exact solution as $\Delta x \to 0$.

Note that

$$\frac{y_{j+1} - y_j}{\Delta x} = \Phi(x_j, y_j, \Delta x)$$

so if $\Delta x \to 0$, the left side approaches $y'(x_j) = F(x_j, y_j)$. Therefore, if we are to get convergence, we must require

$$\lim_{\Delta x \to 0} \Phi(x_j, y_j, \Delta x) = F(x_j, y_j)$$

This is called *consistency*.

A method is stable if $\Phi(x, y, \Delta x)$ is Lipschitz continuous as a function of y. For Runge-Kutta methods, this condition is met if and only if F is Lipschitz continuous.

Now we will prove convergence.

Lemma 1 Suppose $\{w_j\}$ is a sequence which solves

$$w_{j+1} = (1 + \Delta xL)w_j + \Delta x\delta_j, \quad j = 0, 1, \ldots$$

then $w_j \le e^{Lj\Delta x}w_0 + \sum_{\ell=0}^{j-1} e^{L(\Delta x(j-\ell-1))}\Delta x\delta_\ell$.

Proof 1
We proceed by induction. Clearly, for $j = 0$, we have $w_0 \le w_0$. For $j = 1$, we have

$$w_1 = (1 + \Delta xL)w_0 + \Delta x\delta_0 \le e^{L\Delta x}w_0 + \Delta x\delta_0$$

In general,

$$\begin{aligned}
w_{j+1} &= (1+\Delta x L)w_j + \Delta x \delta_j \\
&\le (1+\Delta x L)\left[e^{L\Delta x j}w_0 + \sum_{\ell=0}^{j-1} e^{L\Delta x(j-\ell-1)}\Delta x \delta_\ell\right] + \Delta x\delta_j \\
&\le e^{L\Delta x}\left[e^{L\Delta x j}w_0 + \sum_{\ell=0}^{j-1} e^{L\Delta x(j-\ell-1)}\Delta x \delta_\ell\right] + \Delta x\delta_j \\
&= e^{L\Delta x(j+1)}w_0 + \sum_{\ell=0}^{j-1} e^{L\Delta x(j-\ell)}\Delta x\delta_\ell + \Delta x\delta_j \\
&= e^{L\Delta x(j+1)}w_0 + \sum_{\ell=0}^{j} e^{L\Delta x(j-\ell)}\Delta x\delta_\ell
\end{aligned}$$

which proves the inductive step.

Now define the error of the method by

$$w_j = ||y_j - y(x_j)||$$

and let $\delta_j = \frac{1}{\Delta x}||\tau_j||$ where τ_j is the local truncation error at step j. Finally, define $\delta_{\max} = \max|\delta_j|$. Then,

$$\begin{aligned}
w_{j+1} &= ||y_{j+1} - y(x_{j+1})|| \\
&= ||(y_j + \Delta x\Phi(x_j, y_j, \Delta x)) - (y(x_j) + \Delta x\Phi(x_j, y(x_j), \Delta x) + \tau_j)|| \\
&= ||(y_j - y(x_j)) + \Delta x(\Phi(x_j, y_j, \Delta x) - \Phi(x_j, y(x_j), \Delta x)) - \tau_j|| \\
&\le ||y_j - y(x_j)|| + \Delta x L||y_j - y(x_j)|| + ||\tau_j|| \\
&= (1+\Delta x L)w_j + \Delta x\delta_j
\end{aligned}$$

We can now apply the Lemma to conclude

$$w_j \le e^{L\Delta x j}w_0 + \Delta x\sum_{\ell=0}^{j-1} e^{L\Delta x(j-\ell-1)}\delta_\ell$$

Assume we are solving the differential equation on a fixed interval $[a, b]$, then $\Delta x j \le b - a$, and hence

$$\begin{aligned}
w_j &\le e^{L(b-a)}w_0 + \Delta x\sum_{\ell=0}^{j-1} e^{L(b-a)}\delta_\ell \\
&\le e^{L(b-a)}(w_0 + (b-a)\delta_{\max})
\end{aligned}$$

Therefore, $w_j \to 0$ as $\Delta x \to 0$ because $\delta_{\max} = O(\Delta x^p)$ where p is the order of the method and $w_0 \to 0$. Note that w_0 is normally not zero, but of order machine epsilon

3.2 Convergence of Adams-Bashforth methods

Recall the general Adams-Bashforth method:

$$y_{j+1} = y_j + \Delta x\sum_{i=1}^{n} F_{j+1-i}\beta_i$$

where

$$\beta_i = \frac{1}{\Delta x}\int_{x_j}^{x_j+\Delta x}\prod_{\substack{m=0\\m\neq i}}^{n}\frac{x-x_{j+1-m}}{x_{j+1-i}-x_{j+1-m}}\,dx$$

Let $e_j = y(x_j) - y_j$, then

$$y(x_{j+1}) - y_{j+1} = y(x_j) + \Delta x\sum_{i=1}^{n}F(x_{j+1-i}, y(x_{j+1-i})) + \tau_j - y_j - \Delta x\sum_{i=1}^{n}F(x_{j+1-i}, y_{j+1-i})$$

Taking the norm of each side, we get

$$||e_{j+1}|| \leq ||e_j|| + \Delta x L\sum_{i=1}^{n}||e_{j+1-i}||\,|\beta_i| + ||\tau_j||$$

Next, define $w_0 = \max_{0\leq k<n}||e_k||$, $\beta = \sum_{i=1}^{n}|\beta_i|$, then $w_{j+1} = (1+\Delta x L\beta)w_j + \Delta x||\tau_j/\Delta x||$. Then we have

$$||e_{j+1}|| \leq (1+\Delta x L\beta)w_j + \Delta x\delta_j = w_{j+1}$$

If $w_j \geq \max_{0\leq k\leq j}||e_k||$, then $w_{j+1} \geq \max(w_j, ||e_{j+1}||) \geq \max_{0\leq k\leq j+1}||e_k||$. Finally, we use the lemma as before to get

$$w_j \leq e^{L\beta(b-a)}(w_0 + (b-a)\delta_{\max}$$

3.3 Characteristic Polynomials of Linear Multistep Methods

Recall the general form of a linear multistep method

$$\sum_{i=0}^{k}\alpha_i y_{j+1-i} = \Delta x\sum_{i=0}^{k}\beta_i F_{j+1-i}$$

where $\alpha_0 \neq 0$. Recall we defined the first charateristic polynomial

$$\rho(\theta) = \sum_{i=0}^{k}\alpha_i\theta^{k-i}$$

It is normalized if $\rho'(1) = 1$. The second characteristic polynomial is formed in a similar way using the β's

$$\sigma(\theta) = \sum_{i=0}^{k}\beta_i\theta^{k-i}$$

If the method is to be at least order 1, then

$$\begin{aligned}
\tau &= \sum_{i=0}^{k}\alpha_i y(x_{j+1-i}) - \Delta x\sum_{i=0}^{k}\beta_i F(x_{j+1-i}, y(x_{j+1-i}))\\
&= \sum_{i=0}^{k}\alpha_i\left(y(x_{j+1}) - i\Delta x y'(x_{j+1}) + \frac{(i\Delta x)^2}{2}y''(x_{j+1})\right)\\
&\quad - \Delta x\sum_{i=0}^{k}\beta_i(F(x_{j+1}, y(x_{j+1})) - i\Delta x F_x(x_{j+1}, y(x_{j+1})) - i\Delta x y'(x_{j+1})F_y(x_{j+1}, y(x_{j+1}))) + O(\Delta x^3)\\
&= y(x_{j+1})\left[\sum_{i=0}^{k}\alpha_i\right] - \Delta x y'(x_{j+1})\left[\sum_{i=0}^{k}i\alpha_i + \beta_i\right] + \frac{\Delta x^2}{2}y''(x_{j+1})\left[\sum_{i=0}^{k}i^2\alpha_i + 2i\beta_i\right]
\end{aligned}$$

Thus, a zero order method would require that the first sum be zero, a first order method would also require the second sum to be zero, and a second order method would require all three sums to be zero. Let's look at these in order:

$$0 = \sum_{i=0}^{k} \alpha_i = \rho(1) \tag{6}$$

$$0 = \sum_{i=0}^{k} i\alpha_i + \beta_i = \sum_{i=0}^{k} k\alpha_i - (k-i)\alpha_i + \sum_{i=0}^{k} \beta_i = k\rho(1) - \rho'(1) + \sigma(1) \tag{7}$$

$$\begin{aligned} 0 = \sum_{i=0}^{k} i^2\alpha_i + 2i\beta_i &= \sum_{i=0}^{k} (k^2 - 2ki + i^2 - k + i)\alpha_i - (k^2 - k)\alpha_i - (-2k+1)i\alpha_i + 2\sum_{i=0}^{k} k\beta_i - (k-i)\beta_i \\ &= \rho''(1) - (k^2-k)\rho(1) - (-2k+1)(k\rho(1) - \rho'(1)) + 2k\sigma(1) - 2\sigma'(1) \end{aligned} \tag{8}$$

Therefore, we have the conditions for the order of the method to be

First Order

$$0 = \rho(1)$$

Second Order

$$0 = k\rho(1) - \rho'(1) + \sigma(1) = 0 - 1 + \sigma(1) = \sigma(1) - 1$$

Third Order

$$\begin{aligned} 0 &= \rho''(1) - (k^2-k)\rho(1) + (2k-1)(k\rho(1) - \rho'(1)) + 2k\sigma(1) - 2\sigma'(1) \\ &= \rho''(1) - (2k-1) + 2k - 2\sigma'(1) \\ &= \rho''(1) - 2\sigma'(1) + 1 \end{aligned}$$

For stability, consider the simple problem

$$y' = 0, \quad y(0) = 0$$

and suppose the starting data for the linear multistep method has errors so that $y_j = \delta_j$ where δ_j is small but not zero for $0 \le j < k$. Then the numerical method becomes

$$\sum_{i=0}^{k} \alpha_i y_{j+1-i} = 0$$

This is a constant coefficient difference equation. Equations of this form are solved by guessing a solution of the form

$$y_j = \gamma z^j$$

Inserting this into the equation, we get

$$0 = \sum_{i=0}^{k} \alpha_i y_{j+1-i} = \sum_{i=0}^{k} \alpha_i \gamma z^{j+1-i}$$

Throwing away the trivial solution $z = 0$, we are left with

$$0 = \sum_{i=0}^{k} \alpha_i z^{k-i} = \rho(z)$$

So if z solves $\rho(z) = 0$, then $y_j = \gamma z^j$ solves the difference equation.

Now, suppose there is a root z of $\rho(z) = 0$, where $|z| > 1$, then suppose $\delta_j = \gamma z^j$, then

$$|y_j| = |\gamma|\,|z|^j \to \infty \text{ as } j \to \infty$$

Clearly, this is not a stable method if it sends $y' = 0$ off to infinity. Therefore, we have a necessary condition for stability that all roots of $\rho(z) = 0$ must have magnitude $|z| \leq 1$. A linear multistep method which satisfies this requirement is called *zero stable*. Note that we know at least one root is $z = 1$ because we know $\rho(1) = 0$ is required for consistency.

Example 3.1:

The Adams-Bashforth methods are stable because $\rho(z) = z - 1$ which has only the root $z = 1$.

Example 3.2:

Consider the linear multistep method

$$\frac{1}{6}(y_{j+1} + 4y_j - 5y_{j-1}) = \frac{\Delta x}{6}[4F_j + 2F_{j-1}]$$

For this method, $\rho(z) = \frac{1}{6}z^2 + \frac{2}{3}z - \frac{5}{6}$ and $\sigma(z) = \frac{2}{3}z + \frac{1}{3}$. Note that $\rho'(1) = 1$, so the method is normalized. Also, $\rho(1) = 0$, $\sigma(1) = 1$, and $\rho''(1) - 2\sigma'(1) = -1$ so this method is at least second order. However, this method is not zero-stable because $rho(z) = 0$ has roots $z = 1, -5$. To see why this is a problem, suppose small errors in the initial data are made when solving $y' = 0$, say $y_0 = \epsilon$, $y_1 = -5\epsilon$, then

$$\begin{aligned} y_2 &= -4y_1 + 5y_0 = -4(-5\epsilon) + 5\epsilon = 25\epsilon \\ y_3 &= -4y_2 + 5y_1 = -4(25\epsilon) + 5(-5\epsilon) = -125\epsilon \\ &\vdots \\ y_n &= (-5)^n\epsilon \end{aligned}$$

Thus, the solution rapidly grows, and hence the method is not stable.

It should also be noted that if a root z of $\rho(z) = 0$ is such that $|z| = 1$, then a cannot be a repreated root of $\rho(z) = 0$. To see why, suppose z is a repeated root of $\rho(z) = 0$, then $\rho'(z) = 0$ as well. In this case, it can be shown the difference equation has a solution

$$y_j = \gamma(j + 1)z^j$$

which will also grow if $|z| = 1$ and the method is not stable.

Therefore, we have the *root condition*: If a linear multistep method is stable, then all roots of $\rho(z) = 0$ must be $|z| \leq 1$ with roots $|z| = 1$ not repeated. The *strong root condition* requires that all roots of $\rho(z) = 0$ be $|z| < 1$ except for the required $z = 1$ root.

3.4 Multistep Convergence Theorems

We state, without proof, the following results regarding convergence of multistep methods using fixed step sizes.

Theorem 3 A backward difference formula method of order n is not stable if $n > 6$.

Theorem 4 A backward difference formula method of order $n \leq 6$ is convergent of order $\min(p, q)$ where q is the order of accuracy of the starting values.

Theorem 5 A linear multistep method of order p is convergent of order $min(p, q)$ where the accuracy of the starting values is order q if it satisfies the root condition.

Theorem 6 A predictor-corrector pair of order (p, q) converges of order $min(p+1, q, r)$ where the accuracy of the starting values is order r if both steps satisfy the root condition.

3.5 Computational Errors

One source of error that we have conveniently ignored is roundoff error. For the majority of computations, roundoff is not a difficulty, but it can arise.

To fit in the theory so far, we can lump roundoff error into the truncation error of a method. So for our proof of convergence of a stable one-step method, we use the fact that $\tau = O(\Delta x^p)$ for some $p > 1$. The truth is, $\tau = O(\Delta x^p) + \epsilon$ where ϵ is an error due to roundoff. Returning to the conclusion of the proof of convergence, we see we really have

$$\begin{aligned} w_j &\leq e^{L(b-a)}(w_0 + (b-a)\delta_{\max}) \\ &= e^{L(b-a)}(w_0 + (b-a)\left(O(\Delta x^{p-1}) + \frac{\epsilon}{\Delta x}\right) \end{aligned}$$

Thus, as $\Delta x \to 0$, $w_j \not\to 0$ because the last term increases as Δx decreases. This only becomes a factor when the theoretical truncation error approaches ϵ which is typically about machine ϵ.

Example 3.3:

Consider Euler's method applied to $y' = y^2$, $y(0) = 1$, then

$$\tau_j = \frac{1}{2}\Delta x^2 y''(x_j) + O(\Delta x^3)$$

Now, $y'' = 2yy' = 2y^3$, and $y(x) = 1/(1-x)$ so that

$$\tau = \frac{1}{2}\Delta x^2 2(1-x_j)^{-3} + O(\Delta x^3)$$

At $x_j = 0.9$, $y(x_j) = 10$, so that for double precision, $\epsilon \approx 10^{-16} \cdot 10 = 10^{-15}$. So, we must be cautious if

$$\frac{1}{2}\Delta x^2 \cdot 2 \cdot 10^3 \approx 10^{-15}$$

i.e. $\Delta x \approx 10^{-9}$.

Example 3.4:

Suppose we try Improved Euler on $y' = y^2$, then

$$\tau_j = \frac{1}{6}\Delta x^3 y'''(x_j) - \frac{1}{4}\Delta x^3(F_{xx} + 2FF_{xy} + F^2F_{yy}) + O(\Delta x^4) + \epsilon$$

where $y' = F(x, y)$. Now, $y''' = -6(1-x)^{-4}$, $F_{xx} = F_{xy} = 0$, $F_{yy} = 2$, $F^2 = y^4$, so

$$\begin{aligned} \tau_j &= \Delta x^3\left[\frac{1}{6}(-6)(1-x_j)^{-4} - \frac{1}{4}2y_j^4\right] + \epsilon + O(\Delta x^4) \\ &= -\frac{3}{2}\Delta x^3(1-x_j)^{-4} + \epsilon + O(\Delta x^4) \end{aligned}$$

Thus, at $x_j = 0.9$, we must be cautious if $\Delta x^3 10^4 \approx 10^{-15}$, i.e. $\Delta x \approx 10^{-19/3} \approx 10^{-6.3}$.

Note that both these examples would look much worse if we used single precision arithmetic. Then machine epsilon is $\epsilon \approx 10^{-8}$ and Euler's method breaks down for $\Delta x \approx 10^{-5}$ while improved Euler's method breaks down for $\Delta x \approx 10^{-3.6}$. This analysis is supported by looking at the table of values 1 where $y' = y^2$ is solved using single precision. Note the decay in the accuracy for $\Delta x \approx 10^{-3}$. This agrees with our predicted

Δx	10^{-1}	10^{-2}	10^{-3}	10^{-4}
$E(\Delta x)$	2.3×10^{-3}	3.2×10^{-5}	5.7×10^{-5}	2.9×10^{-4}
$E(\Delta x)/\Delta x^2$	2.3×10^{-1}	3.2×10^{-1}	5.7×10^{1}	2.9×10^{4}

Figure 1: Error data for Improved Euler applied to $y' = y^2$, $y(0) = 1$

breakdown of $\Delta x \approx 10^{3.6}$.

This analysis is particularly important when considering implicit methods and determining the stopping criterion for the iterative solver of y_{j+1}. In this case, the error in doing a finite number of iterations can be interpreted as a computational error ϵ. It could be reduce to approximately machine epsilon, but at a high cost. It is better to iterate until the relative error is an order of magnitude smaller than the local truncation error so that it does not affect the theoretical convergence rate of the method.

3.6 Absolute Stability

When we discussed stability in the context of convergence, it was a requirement for the convergence of the method and not necessarily one useful in practice. A more practical interpretation of stability is to understand how large of a step size can be taken without destroying stability. To study this, we consider the linear system of ordinary differential equations

$$y' = Ay + B.$$

Suppose that u, v are both solutions of this equation, then

$$u' - v' = A(u - v).$$

If $w = u - v$, then of course $w' = Aw$. Absolute stability is where a method has the property that

$$|u_{j+1} - v_{j+1}| \leq |u_0 - v_0|$$
$$|w_{j+1}| \leq |w_0|$$

for all j.

As an example, consider Euler's method, and let w_j be given by

$$w_{j+1} = w_j + \Delta x A w_j \tag{9}$$

Assume A is diagonalizable, then $A = T\Lambda T^{-1}$ where Λ is a diagonal matrix with the eigenvalues of A down the diagonal. Equation (9) now becomes

$$TT^{-1}w_{j+1} = TT^{-1}w_j + \Delta x T\Lambda T^{-1}w_j$$

Let $\tilde{w}_j = T^{-1}w_j$, then $\tilde{w}_j$ solves

$$\tilde{w}_{j+1} = \tilde{w}_j + \Delta x \Lambda \tilde{w}_j$$

This is now a decoupled system, so we can look at each of the scalar equations

$$\tilde{w}^\ell_{j+1} = \tilde{w}^\ell_j + \Delta x \lambda_\ell \tilde{w}^\ell_j$$

where λ_ℓ is the ℓ^{th} eigenvalue of A.

The *region of absolute stability* is the set of values of $\Delta x\lambda$ in the complex plane for which the magnitude of $\tilde{w}$ does not increase. It can be calculated by calculating the region for each of the separate scalar equations and the final region is the intersection for each of the different scalar equations. Suppose we are looking at the ℓ^{th} equation (dropping the ℓ for convenience) we have $\tilde{w}_{j+1} = \tilde{w}_j + \Delta x\lambda\tilde{w}_j$. This is a difference equation which we can solve by assuming $\tilde{w}_j = \alpha z^j$ to get

$$\begin{aligned} \alpha z^{j+1} &= \alpha z^j + \alpha\Delta x\lambda z^j \\ z &= 1 + \Delta x\lambda = 1 + \xi \end{aligned}$$

where $\xi = \Delta x\lambda$. Now what we need is the values of all ξ such that $|z| \leq 1$.

$$1 \geq |z|^2 = (1+\xi)(1+\bar{\xi}) = 1 + \xi + \bar{\xi} + \xi\bar{\xi}$$

Now let $\xi = x + iy$, then this becomes

$$1 \geq 1 + (x+iy) + (x-iy) + (x+iy)(x-iy) = 1 + 2x + x^2 + y^2 = (1+x)^2 + y^2$$

Thus, the region of absolute stability for Euler's method is a circle of radius 1 centered at $\xi = -1$ in the complex plane. This same strategy can be employed for other methods, but rapidly becomes more complicated.

Compare this to the region for backward Euler. For backward Euler, we have

$$\begin{aligned} \tilde{w}_{j+1} &= \tilde{w}_j + \Delta x\lambda\tilde{w}_{j+1} \\ \alpha z^{j+1} &= \alpha z^j + \alpha\Delta x\lambda z^{j+1} \\ z &= 1 + \xi z \\ (1-\xi)z &= 1 \\ z &= \frac{1}{1-\xi} \end{aligned}$$

Therefore,

$$\begin{aligned} 1 &\geq |z\bar{z}| \\ &= \left(\frac{1}{1-\xi}\right)\left(\frac{\bar{1}}{1-\xi}\right) \\ &= \frac{1}{(1-\xi)(1-\bar{\xi})} \\ &= \frac{1}{1-2x+x^2+y^2} \\ &= \frac{1}{(1-x)^2+y^2} \end{aligned}$$

Thus, the region of absolute stability is everywhere outside the disk in the complex plane centered at $\xi = 1$. The difference between these is that forward Euler has a bounded region of absolute stability and backward Euler has an unbounded region of absolute stability.

Theorem 7 All zero-stable explicit Runge-Kutta methods, explicit linear multistep methods, and explicit predictor-corrector pairs have finite regions of absolute stability.

In general, calculating regions of absolute stability is not easy by hand but can be carried out by computer. Comparison between regions of absolute stability helps determine the optimal method for a given problem by looking at the values of λ that are present in a given ordinary differential equation.

4 Error Estimation and Control

When computing numerical solutions, we would like to be able to know at least a bound on the error of the computed solution. So far, we have seen that so long as Δx is "small enough", the solution will converge to the exact solution, but what is "small enough?" Ideally, we would like to specify an error tolerance as an input parameter to a method. In order to accomplish this, we must be able to (1) estimate the error and (2) use the estimate to dynamically control the error.

4.1 Global Error Estimates

Up to now, we have focused on the order of the local truncation error, but have not done much else with it. In fact, what we can do is make an assumption which can be shown to be valid, that the error of a method can be written in the form

$$y_j = y(x_j) + \Delta x^p e(x_j) + O(\Delta x^{p+1})$$

for some smooth function $e(x_j)$. We can use this representation to estimate the error as we have already seen in the homework. Suppose $x_N = b$ is the final time of a calculation, then we use the same method twice with two different step sizes to get an estimate for the error.

$$\begin{aligned} \Delta x: &\qquad y_N = y(b) + \Delta x^p e(b) + O(\Delta x^{p+1}) \\ \frac{\Delta x}{2}: &\qquad y^*_{2N} = y(b) + \left(\frac{\Delta x}{2}\right)^p e(b) + O(\Delta x^{p+1}) \end{aligned}$$

Taking the difference, we get

$$\begin{aligned} y_N - y^*_{2N} &= \Delta x^p e(b)(1 - 2^{-p}) + O(\Delta x^{p+1}) \\ \text{Error } &= \Delta x^p e(b) = \frac{y_N - y^*_{2N}}{1 - 2^{-p}} + O(\Delta x^{p+1}) \end{aligned}$$

This is the same result in a slightly different form that was derived in the first homework.

4.2 Local Error Estimation

While the formula for the global error is useful for estimates of the error for a fixed step method, it is not an efficient means for controlling the error of a computation. Ideally, one wishes to give the gloval error as a parameter for a solver which then produces a solution within the accepted tolerance.

There are two components necessary for accomplishing this, step size control and error estimators. We gebin with error estimators. We will focus on one-step methods because they are much easier to use for adaptive time stepping.

We assume the method is of the form

$$y_{j+1} = y_j + \Delta x \Phi(x_j, y_j, \Delta x)$$

The *principal error function* of this method is a function $\psi(x, y)$ which is a smooth function that is derived from the leading order error term in the local truncation error for the method. Thus,

$$\tau_{j+1} = u(x_{j+1}) - y_{j+1} = \Delta x^{p+1} \psi(x_j, y_j) + O(\Delta x^{p+2})$$

where $u(x)$ is the exact solution with given initial condition $u(x_j) = y_j$.

4.2.1 Richardson Extrapolation

Consider a one-step p^{th} order method, then we know

$$u(x_{j+1}) - y_{j+1} = \Delta x^{p+1}\psi(x_j, y_j) + O(\Delta x^{p+2})$$

where ψ is the principal error function. Now suppose we cover the same step in two half steps:

$$u_1(x_{j+1/2}) - y^*_{j+1/2} = \left(\frac{\Delta x}{2}\right)^{p+1} \psi(x_j, y_j) + O(\Delta x^{p+2})$$
$$u_2(x_{j+1}) - y^*_{j+1} = \left(\frac{\Delta x}{2}\right)^{p+1} \psi(x_{j+1/2}, y^*_{j+1/2}) + O(\Delta x^{p+2})$$

Now,

$$(u(x_{j+1}) - y_{j+1}) - (u_2(x_{j+1}) - y^*_{j+1}) = (1 - 2^{-(p+1)})\Delta x^{p+1}\psi(x_j, y_j) + O(\Delta x^{p+2})$$
$$y^*_{j+1} - y_{j+1} + (u(x_{j+1}) - u_2(x_{j+1})) = (1 - 2^{-(p+1)})\Delta x^{p+1}\psi(x_j, y_j) + O(\Delta x^{p+2})$$

And then we have

$$u(x_{j+1}) - u_2(x_{j+1}) = u(x_{j+1/2}) - u_2(x_{j+1/2}) + O\left(\frac{\Delta x}{2}\|u(x_{j+1/2}) - u_2(x_{j+1/2})\|\right)$$
$$u(x_{j+1/2}) - u_2(x_{j+1/2}) = u(x_{j+1/2}) - y^*_{j+1/2} = \left(\frac{\Delta x}{2}\right)^{p+1} \psi(x_j, y_j)$$
$$u(x_{j+1}) - u_2(x_{j+1}) = \left(\frac{\Delta x}{2}\right)^{p+1} \psi(x_j, y_j) + O(\Delta x^{p+2})$$

Therefore,

$$\begin{aligned} y^*_{j+1} - y_{j+1} &= (1 - 2^{-(p+1)} - 2^{-(p+1)})\Delta x^{p+1}\psi(x_j, y_j) + O(\Delta x^{p+2}) \\ &= \frac{2^p - 1}{2^p}\Delta x^{p+1}\psi(x_j, y_j) \end{aligned}$$

Thus, the local error estimate is

$$\text{Local error } = \Delta x^{p+1}\psi(x_j, y_j) = \frac{2^p}{2^p - 1}(y^*_{j+1} - y_{j+1})$$

Note that this method can also be used to improve the order of accuracy of the method by always taking the higher order approximation and adding the error back in. However, adding the error back in can adversely affect the stability of the method, so is not always done.

4.2.2 Solution Pair Method

Suppose you have two methods, one of order p, and the other of order $p+1$. If we look at each of the errors we get

$$\begin{aligned} p: &\qquad u(x_{j+1}) - y_{j+1} = \Delta x^{p+1}\psi(x_j, y_j) + O(\Delta x^{p+2}) \\ p+1: &\qquad u(x_{j+1}) - y^*_{j+1} = O(\Delta x^{p+2}) \end{aligned}$$

Taking the difference, we see that

$$\begin{aligned} \text{Local error } &= \Delta x^{p+1}\psi(x_j, y_j) \\ &= (u(x_{j+1}) - y_{j+1}) - (u(x_{j+1}) - y^*_{j+1}) \\ &= y^*_{j+1} - y_{j+1} \end{aligned}$$

4.2.3 Forward-Backward Method

Given a single p^{th} order method, take a forward step to get

$$u(x_{j+1}) - y_{j+1} = \Delta x^{p+1}\psi(x_j, y_j) + O(\Delta x^{p+2})$$

Next, take a backward step of the same size

$$\begin{aligned} v(x_j) - y_j^* &= (-\Delta x)^{p+1}\psi(x_{j+1}, y_{j+1}) + O(\Delta x^{p+2}) \\ &= (-dx)^{p+1}\psi(x_j, y_j) + O(\Delta x^{p+2}) \end{aligned}$$

If we assume p is odd, then add together to get

$$\begin{aligned} 2\Delta x^{p+1}\psi(x_j, y_j) &= u(x_{j+1}) - y_{j+1} + v(x_j) - y_j^* \\ &= u(x_{j+1}) - v(x_{j+1}) + v(x_j) - u(x_j) + y_j - y_j^* \end{aligned}$$

Now, as we did for Richardson extrapolation, we have

$$\begin{aligned} v(x_j) - u(x_j) &= v(x_{j+1}) - u(x_{j+1}) + O(\Delta x||v(x_{j+1}) - u(x_{j+1})||) \\ &= v(x_{j+1}) - u(x_{j+1}) + O(\Delta x||y_{j+1} - u(x_{j+1})||) \\ &= v(x_{j+1}) - u(x_{j+1}) + O(\Delta x^{p+2}) \end{aligned}$$

Plugging this back in gives

$$\begin{aligned} y_j - y_j^* &= 2\Delta x^{p+2}\psi(x_j, y_j) + O(\Delta x^{p+2}) \\ \Delta x^{p+1}\psi(x_j, y_j) &= \frac{1}{2}(y_j - y_j^*) + O(\Delta x^{p+2}) \end{aligned}$$

Note that when we advance one step and use these error estimates, we could use these estimates to improve the accuracy of our method. This is called *local extrapolation*. For soe methods it is possible to use local extrapolation, but the stability of these methods is difficult to establish and can lead to problems for stiff equations. Also, the error estimate is no longer technically valid.

4.3 Step Size Control

Now that we have error estimates, we will use these estimates to control the error. Suppose we are solving an equation on an interval of length L and have a requirement that the error be bounded by some tolerance ϵ. Certainly if we bound the error of step j by $e_j \leq \frac{\epsilon}{L}\Delta x_j$ where Δx_j is the length of step j, then

$$\text{Total error } = \sum_j e_j \leq \sum_j \frac{\epsilon}{L}\Delta x_j = \frac{\epsilon}{L}L = \epsilon$$

So the objective is to ensure $e_j \leq \frac{\epsilon}{L}\Delta x_j$.

Recall $e_j \approx \Delta x_j^{p+1}\psi(x_j, y_j)$, then

$$\begin{aligned} \Delta x_j^{p+1}\psi(x_j, y_j) &\leq \frac{\epsilon}{L}\Delta x_j \\ \Delta x_j^{p} &\leq \frac{\epsilon}{L}\frac{1}{\psi(x_j, y_j)} \end{aligned}$$

Since we are free to choose any size for Δx_j, then we can always find a Δx_j that will satisfy the one-step error bound. We therefore have a crterion for an acceptable time step, namely

$$e_j = \Delta x_j^{p+1}\psi(x_j, y_j) \leq \frac{\epsilon}{L}\Delta x_j$$

Using this estimate, if we take a step of size Δx_j^*, instead, then the estimated error would be

$$\text{predicted error } (\Delta x_j^*) \ = \Delta x_j^{*p+1} \psi(x_j, y_j) = \left(\frac{\Delta x_j^*}{\Delta x_j}\right)^{p+1} e_j$$

Thus, the optimal step size is predicted to be

$$\left(\frac{\Delta x_j^*}{\Delta x_j}\right)^{p+1} e_j \leq \frac{\epsilon}{L} \Delta x_j^*$$

$$(\Delta x_j^*)^p = \frac{\epsilon}{L} \frac{\Delta x_j^{p+1}}{e_j}$$

$$\Delta x_j^* \approx \left(\frac{\epsilon}{L} \frac{\Delta x_j^{p+1}}{e_j}\right)^{1/p}$$

This gives a predicted optimal step size. The actual step size which satisfies the local error criterion must be verified. Since the optimal step size is approximate, people often take step sizes to be about $0.9\Delta x_j^*$.

One must be careful about rapidly varying step sizes, particularly when they grow for stability reasons. Thus, it is common to allow the step size to shrink quickly, but the step size should not increase by more than a factor of 5 from one time step to the next.

Let us now describe the variable step algorithm for a method M of order p on the interval $a \leq x \leq b$.

1. Given Δx_j (usually the previous step size), compute error estimate e_j.
2. Use e_j to compute the optimal step size $\Delta x_j^* \approx \left(\frac{\epsilon}{L} \frac{\Delta x_j^{p+1}}{e_j}\right)^{1/p}$
3. Set $\Delta x_j' = \min(0.9\Delta x_j^*, 2\Delta x_j, b - x_j)$
4. Compute the error estimate e_j' using $\Delta x_j'$
5. If $e_j' > \frac{\epsilon}{L} \Delta x_j'$, then go to step 4.
6. Apply the method with step size $\Delta x_j = \Delta x_j'$.
7. Go to step 1

5 Stiff Equations

Recall from our convegence proofs for one-step methods we found that the computed error was bounded by

$$w_j \leq e^{L(b-a)}(w_0 + (b-a)\delta_{\max})$$

where $\delta_{\max}$ was given by the local truncation error, and L is the Lipschitz conatant of F. When discussing practical step size limitations due to fixed point iterations for implicit methods, we saw that

$$\Delta x |\gamma| L \leq 1$$

where $\gamma = O(1)$, and L is the Lipschitz constant. In fact, the quantity $L(b-a)$ is one measure of the *stiffness* of a problem. If $L(b-a)$ is large, it shows that we must take much smaller timesteps to bound the error and smaller time steps to gaurantee convergence for fixed point iteration. The stiffer the problem, the more difficult it is to solve numerically.

Consider the scale ordinary differential equation

$$y' = J(y - p(x)) + p'(x), \quad y(0) = A$$

where J is a constant. The exact solution is

$$y(x) = (A - p(0))e^{Jx} + p(x)$$

If $J < 0$, then this problem is stable and if J is large, then the solution curves converge quickly to the curve $y = p(x)$ during the initial transition or boundary layer.

In the boundary layer, the truncation error is

$$\tau = \frac{\Delta x^2}{2} y''(\xi) = \frac{\Delta x^2}{2}[(a - p(0))J^2 e^(J\xi) + p''(\xi)]$$

If J is large, then the first term will dominate in size and a small time step is required. In this regime $(b-a)$ is small, so $L(b-a)$ is manageable.

Outside the boundary layer,

$$\tau = \frac{\Delta x^2}{2} y''(\xi) \approx \frac{\Delta x^2}{2} p''(\xi)$$

and hence an accurate solution is independent of J, so larger time steps would be OK. But the stability requirement of methods still apply, so that stable solutions are only possible for small values of Δx. Since the stability restriction seems to be the problem, then we can turn to the implicit method backward Euler which has almost the same truncation error. Ordinarily, this will require fixed point iteration which has a convergence requirement of $\Delta x L \leq 1$ which is even more restrictive than Forward Euler!

It is therefore clear we need the unconditional stability properties of implicit methods while skirting around the fixed point iteration procedure. In practice, backward difference formula methodsare the primary tool for stiff equations so we will focus on them. Consider the second order backward difference formula method

$$y_{j+1} = \frac{1}{3}(4y_j - y_{j-1}) + \frac{2}{3}\Delta x F(x_{j+1}, y_{j+1})$$

At each step we must solve for y_{j+1} and x_{j+1}, y_j, y_{j-1} are all fixed. Thus, this equation can be wrtten in the form

$$y = \Delta x \gamma F(y) + \psi$$

If we apply fixed point iteration, we get

$$y^{m+1} = \Delta x \gamma F(y^m) + \psi$$

If y^* is the exact solution, then $y^* = \Delta x \gamma F(y^*) + \psi$ and we get

$$y^{m+1} - y^* = \Delta x \gamma (F(y^m) - F(y^*))$$

Suppose F is a linear function of y, say $F(y) = Jy + B$, then

$$y^{m+1} - y^* = \Delta x \gamma J (y^m - y^*)$$

Let $\lambda_{\max}$ be the largest magnitude eigenvalue of J, then

$$|y^{m+1} - y^*| \leq \Delta x \gamma_{\max} |y^m - y^*| \leq (\Delta x \gamma \lambda_{\max})^m |y^0 - y^*|$$

and hence we see that backward difference formula methods have the same practical time step limitation as other implicit methods.

Suppose we linearize F about the current iterate y^m. Then,

$$F(y) \approx F(y^m) + J(y - y^m)$$

where J is an approximation to $\dfrac{\partial F}{\partial y}(y^m)$. If we make this replacement, we get

$$\begin{aligned} y^{m+1} &= \Delta x\gamma F(y^{m+1}) + \psi \\ &= \Delta x\gamma[F(y^m) - J(y^{m+1} - y^m)] + \psi \\ (I - \Delta x\gamma J)y^{m+1} &= \Delta x\gamma[F(y^m) - Jy^m] + \psi \\ y^{m+1} &= (I - \Delta x\gamma J)^{-1}[\Delta x\gamma(F(y^m) - Jy^m) + \psi] \end{aligned}$$

If $J = \frac{\partial F}{\partial y}(y^m)$, then this amounts to Newton's method. There are two difficulties with simply using Newton's method. First, it requires a matrix inversion every iteration because J depends on m, and second, the matrix J can be very large for some systems so that storage becomes an issue.

Suppose J is held constant where $J \approx \frac{\partial F}{\partial y}(y^*)$, then the procedure goes as follows:

$$\begin{aligned} y^{m+1} &= \psi + \Delta x\gamma F(y^m) + \Delta x\gamma J(y^{m+1} - y^m) \\ y^* &= \psi + \Delta x\gamma F(y^*) + \Delta x\gamma J(y^* - y^*) \\ y^{m+1} - y^* &= \Delta x\gamma(F(y^m) - F(y^*)) + \Delta x\gamma J(y^{m+1} - y^*) - \Delta x\gamma J(y^m - y^*) \\ &= \Delta x\gamma\mathcal{J}(y^m - y^*) + \Delta x\gamma J(y^{m+1} - y^*) - \Delta x\gamma J(y^m - y^*) \end{aligned}$$

where $\mathcal{J}$ is the evaluation of $\frac{\partial F}{\partial y}$ at some point between y^m and y^* via the mean value theorem

$$F(u) - F(v) = \frac{\partial F}{\partial y}(w)(u - v)$$

for some w between u and v. Now we have

$$y^{m+1} - y^* = (I - \Delta x\gamma J)^{-1}\Delta x\gamma(\mathcal{J} - J)(y^m - y^*)$$

Thus, convergence of the iteration is dependent upon

$$||(I - \Delta x\gamma J)^{-1}\Delta x\gamma(\mathcal{J} - J)|| < 1$$

This shows that faster convergence is achieved by either taking Δx smaller or making J a better approximation of $\mathcal{J}$.

The method for getting the next iterate is actually slightly different. Subtract y^m from each side to get

$$y^{m+1} - y^m = \psi + \Delta x\gamma F(y^m) - y^m + \Delta x\gamma J(y^{m+1} - y^m)$$

Now define the residual $r^m = \psi + \Delta x\gamma F(y^m) - y^m$, then

$$\begin{aligned} y^{m+1} - y^m &= r^m + \Delta x\gamma J(y^{m+1} - y^m) \\ (I - \Delta x\gamma J)(y^{m+1} - y^m) &= r^m \\ y^{m+1} - y^m &= (I - \Delta x\gamma J)^{-1}r^m \\ y^{m+1} &= y^m + (I - \Delta x\gamma J)^{-1}r^m \end{aligned}$$

Using this type of iteration larger times steps can be taken provided J is a good approximation of $\mathcal{J}$. For that reason we must study approximations of $\mathcal{J}$.

At present, we have a couple choices. We can supply an explicit expression of the Jacobian, or we can compute a numerical approximation of the Jacobian. The first method is preferrable, but is not always possible or practical for large complex systems. We will concentrate on numerical approximation.

Suppose $y' = F(x, y)$ is a system of n equations so that the Jacobian is an $n \times n$ matrix. Let e_k be the k^{th} coordinate vector, and let δ be a small scalar. The k^{th} column of J is then given by

$$J_{\cdot k} = \frac{[F(x, y + \delta e_k) - F(x, y)]}{\delta} \approx \frac{\partial F}{\partial y_k}(x, y) + O(\delta^2)$$

Thus, it requires $N+1$ evaluations of F to compute J. If F is complicated, this could be expensive to be used every time step. For nono-stiff equation, this many function evaluations is not very good, so often Jacobians are kept and reused for multiple steps.

We can now describe the algorithm for stiff equations

1. Choose a backward difference formula method of desired order and write in the form

$$\sum_{i=0}^{k} \alpha_i y_{n+1-i} = \gamma F(x_{n+1}, y_{n+1})$$

   Define the function $\tilde{F}$ to be this method rewritten in the form

$$y_{n+1} = \gamma \tilde{F}(y_{n+1}) + \psi$$

   where we note again that all past values y_n, y_{n-1}, etc. and x_{n+1} are all constants in the solution of this equation.

2. Determine an approximation of the Jacobian J either by an explicit description or by numerical differentiation.

3. Given $y_1,\ldots,y_n$, define $r^0 = \psi + \Delta x \gamma \tilde{F}(y_n) - y_n$, and $y^0 = y_n$. Then iterate

$$y^{m+1} = y^m + (I - \Delta x \gamma J)^{-1} r^m$$

   Terminate when $y^{m+1} - y^m$ is below tolerance.

4. $y_{n+1} = y^{m+1}$.

6 Application to Partial differential equations

We can use these methods we have described to solve partial differential equations. Consider the equation

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial u}{\partial x}, \quad a \le x \le b, \quad 0 \le t \le T$$

Suppose we have an approximation of $\frac{\partial u}{\partial x}$ at any time t, then we could think of this as an infinite set of ordinary differential equations where an ordinary differential equation corresponds to each point x in the domain $a \le x \le b$. The different ordinary differential equations are coupled together through the presence of the spatial derivative term $\frac{\partial u}{\partial x}$.

Suppose now we choose a finite subset of these ordinary differential equations, say regularly space points x_j and assume $u(x,t)$ is approximated by an interpolating function at the x_j. Assume for now there is a matrix D such that

$$\begin{bmatrix} \frac{\partial u}{\partial x}(x_0) \\ \vdots \\ \frac{\partial u}{\partial x}(x_N) \end{bmatrix} \approx D \begin{bmatrix} u(x_0) \\ \vdots \\ u(x_N) \end{bmatrix}$$

Then we would have a system of ordinary differential equations given by

$$\begin{bmatrix} \frac{\partial u}{\partial t}(x_0) \\ \vdots \\ \frac{\partial u}{\partial t}(x_N) \end{bmatrix} = \alpha D \begin{bmatrix} u(x_0) \\ \vdots \\ u(x_N) \end{bmatrix}$$

Let $U(t) = \begin{bmatrix} u(x_0,t) \\ \vdots \\ u(x_N,t) \end{bmatrix}$, then we have $\frac{dU}{dt} = \alpha D U$.

Now the only question that remains is what the matrix D looks like. At x_j, we want to approximate $\frac{\partial u}{\partial x}(x_j, t)$, so let's assume u is approximated by a quadratic polynomial that interpolates u at x_{j-1}, x_j, x_{j+1}. Using Lagrange polynomials, the interpolating polynomial would be

$$p_j(x) = u(x_{j-1})\frac{(x-x_j)(x-x_{j+1})}{(x_{j-1}-x_j)(x_{j-1}-x_{j+1})} + u(x_j)\frac{(x-x_{j-1})(x-x_{j+1})}{(x_j-x_{j-1})(x_j-x_{j+1})} + u(x_{j+1})\frac{(x-x_{j-1})(x-x_j)}{(x_{j+1}-x_{j-1})(x_{j+1}-x_j)}$$

Assume $x_{j+1} - x_j = x_j - x_{j-1} = \Delta x$, then

$$p_j(x) = \frac{u(x_{j-1})}{2\Delta x^2}(x-x_j)(x-x_{j+1}) - \frac{u(x_j)}{\Delta x^2}(x-x_{j-1})(x-x_{j+1}) + \frac{u(x_{j+1})}{2\Delta x^2}(x-x_{j-1})(x-x_j)$$

Now, we have

$$\begin{aligned}
\frac{\partial u}{\partial x}(x_j) &\approx p_j'(x_j) \\
&= \left\{ \frac{u(x_{j-1})}{2\Delta x^2}[(x-x_j)+(x-x_{j+1})] - \frac{u(x_j)}{\Delta x^2}[(x-x_{j-1})+(x-x_{j+1})] \right. \\
&\quad \left. + \frac{u(x_{j+1})}{2\Delta x^2}[(x-x_{j-1})+(x-x_j)] \right\}_{x=x_j} \\
&= \frac{u(x_{j-1})}{2\Delta x^2}(-\Delta x) - \frac{u(x_j)}{\Delta x^2}(\Delta x - \Delta x) + \frac{u(x_{j+1})}{2\Delta x^2}(\Delta x) \\
&= \frac{1}{2\Delta x}(u(x_{j+1}) - u(x_{j-1}))
\end{aligned}$$

The matrix D can now be seen to be a matrix of the form

$$D = \begin{bmatrix}
\ddots & \ddots & & & & & \\
\ddots & \ddots & \ddots & & & & \\
 & \frac{-1}{2\Delta x} & 0 & \frac{1}{2\Delta x} & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \frac{-1}{2\Delta x} & 0 & \frac{1}{2\Delta x} & \\
 & & & & \ddots & \ddots & \ddots \\
 & & & & & \ddots & \ddots
\end{bmatrix}$$

The only part that remains unsolved is what to do at the endpoints. If we take periodic boundary conditions, then D becomes

$$D = \frac{1}{2\Delta x}\begin{bmatrix}
0 & 1 & 0 & \cdots & 0 & -1 \\
-1 & \ddots & \ddots & \ddots & & 0 \\
0 & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & 0 \\
0 & & \ddots & \ddots & \ddots & 1 \\
1 & 0 & \cdots & 0 & -1 & 0
\end{bmatrix}$$

Of course this only scratches the surface of the analysis required for solving partial differential equations, but it at least shows the connection between methods for ordinary differential equations and those for partial differential equations.