CHAPTER I - NUMERICAL METHODS FOR THE SOLUTION OF DIFFERENTIAL EQUATIONS

JOSEPH G. CONLON

1. Ordinary Differential Equations (ODE)

A differential equation in n variables $y = [y_1, .., y_n] \in \mathbf{R}^n$ is of the form

$$\frac{dy(t)}{dt} \;=\; f(y(t), t)\ , \tag{1.1}$$

where $f(y,t) = [f_1(y,t), .., f_n(y,t)] \in \mathbf{R}^n$ is a vector valued function commonly referred to as a *vector field*. Thus for every $y \in \mathbf{R}^n$ it yields a time dependent vector $f(y,t)$ in $\mathbf{R}^n$. The fundamental theorem of ODE tells us that the equation (1.1) has a unique solution both forward and backward in time when the value of $y(t)$ is specified at a certain time say t_0, so that $y(t_0) = y^0$ is given. In most applications in applied mathematics we wish to solve (1.1) *forward in time* i.e. for $t \geq t_0$. In that case we say the condition $y(t_0) = y^0$ is the *initial condition* for the differential equation (1.1). An example of this occurs in fluid flow. When we place a light particle such as a leaf in a stream it will be carried along by the fluid. In that case $f(y,t)$ is the fluid velocity at the position y at time t, and y^0 is the position we put the leaf at the initial time t_0. We shall also be interested in solving differential equations *backward in time* i.e. for $t \leq t_0$. In that case the condition $y(t_0) = y^0$ is called the *terminal condition* for the equation (1.1). Such problems come up in control theory, a subject of which mathematical finance is a part. A familiar example of such a problem for an *infinite dimensional* ODE is the problem of finding the value of an option such as a call option. The value of the option as a function of the stock price S is known at the expiration date $T > 0$ of the option, and the bank wishes to compute the value of the option today which is time $t = 0 < T$ say. In the Black-Scholes (BS) theory one finds the value of the option by solving an infinite dimensional linear ODE like (1.1) backward in time until time $t = 0$ with the terminal condition given by the option price at the expiration date T.

It is sometimes possible to solve the differential equation (1.1) explicitly. The most common situation where that occurs is when the vector field $f(y,t)$ is *linear* so that

$$f(y,t) \;=\; A(t)y + B(t)\ , \quad A(t), B(t) \text{ time dependent } n \times n \text{ matrices.} \tag{1.2}$$

Example 1. *Consider the* 2 *dimensional system*

$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = 3y_2 - 2y_1\ . \tag{1.3}$$

We wish to find the unique solution of (1.3) with the given initial condition at time $t = 0$*,*

$$y_1(0) = 1, \quad y_2(0) = 0. \tag{1.4}$$

We can find an explicit solution by rewriting the system (1.3) as the scalar ODE

$$\frac{d^2y_1}{dt^2} = 3\frac{dy_1}{dt} - 2y_1 \ . \tag{1.5}$$

To find a basis of solutions for (1.5) we substitute $y_1(t) = e^{rt}$ into (1.5). This function is a solution to (1.5) provided r is a solution to the quadratic equation

$$r^2 - 3r + 2 = 0 \quad \text{which implies } r = 1 \text{ or } r = 2. \tag{1.6}$$

The general solution to (1.5) is therefore

$$y_1(t) \ = \ c_1e^t + c_2e^{2t} \quad \text{where } c_1, c_2 \text{ are arbitrary constants.} \tag{1.7}$$

The initial condition (1.4) determines c_1, c_2 uniquely by the simultaneous equations

$$c_1 + c_2 = 1, \quad c_1 + 2c_2 = 0 \quad \text{implies } c_1 = 2, \ c_2 = -1. \tag{1.8}$$

We conclude that the unique solution to the initial value problem (1.3), (1.4) is given by

$$y_1(t) = 2e^t - e^{2t} \ , \quad y_2(t) = 2e^t - 2e^{2t} \ . \tag{1.9}$$

In general one cannot produce explicit formulas for the solutions to (1.1). Hence we need to resort to *numerical methods*. Thus we discretize the time variable into multiples of some small basic time increment Δt, whence time now takes the discrete values $0, \Delta t, 2\Delta t, 3\Delta t, ...$ We replace the derivative in (1.1) by a finite difference which approximates the derivative for small Δt. There are several ways to carry this out:

$$\text{Forward difference :} \quad \frac{dy(t)}{dt} \simeq \frac{y(t+\Delta t) - y(t)}{\Delta t} \ , \tag{1.10}$$

$$\text{Backward difference :} \quad \frac{dy(t)}{dt} \simeq \frac{y(t) - y(t-\Delta t)}{\Delta t} \ , \tag{1.11}$$

$$\text{Central difference :} \quad \frac{dy(t)}{dt} \simeq \frac{y(t+\Delta t) - y(t-\Delta t)}{2\Delta t} \ . \tag{1.12}$$

Note that the central difference is the average of the forward and backward difference. The error made in these approximations goes to 0 as $\Delta t \to 0$. It goes faster however to 0 in the central difference approximation (1.12) than in the forward or backward difference. We can see this by doing a Taylor expansion

$$y(t+\Delta t) = y(t) + y'(t)[\Delta t] + y''(t)[\Delta t]^2/2 + y'''(t)[\Delta t]^3/6 + \cdots \tag{1.13}$$

It follows that if we denote the error by $\varepsilon(\Delta t)$ so that

$$\varepsilon(\Delta t) \ = \ \text{difference approximation} - dy(t)/dt \ , \tag{1.14}$$

then $\varepsilon(\Delta t) = O(\Delta t)$ for the forward and backward differences, but $\varepsilon(\Delta t) = O([\Delta t]^2)$ for the central difference approximation. Thus we say that the forward and backward differences are *first order accurate* approximations to the derivative, whereas the central difference is a *second order accurate* approximation.

We can use the difference approximations (1.10), (1.11) to obtain a numerical scheme for solving the ODE (1.1). Suppose first we wish to solve (1.1) forward in time $t > 0$ with initial condition $y(0) = y^0$ given. If we use the forward difference (1.10) to approximate the derivative, then we obtain the *explicit Euler scheme*

$$y(t+\Delta t) \ = \ y(t) + \Delta t f(y(t), t) \ . \tag{1.15}$$

On setting $y^m \sim y(m\Delta t)$, then (1.15) yields the recurrence relation

$$y^{m+1} = y^m + \Delta t f(y^m, m\Delta t) \ , \quad m = 0, 1, 2, ... \tag{1.16}$$

The numerical algorithm is explicit in the sense that once we have computed y^m then y^{m+1} is easily computed by evaluating the RHS of (1.16). Alternatively we may wish to solve (1.1) backward in time for $t < T$ with terminal condition $y(T) = y^M$ specified at $t = T$, where M is an integer satisfying $M\Delta t = T$. In this case we use the backward difference (1.11) to implement the explicit Euler method, whence we have

$$y(t - \Delta t) \ = \ y(t) - \Delta t f(y(t), t) \ . \tag{1.17}$$

Again on setting $y^m \sim y(m\Delta t)$, then (1.17) yields the recurrence relation

$$y^{m-1} \ = \ y^m - \Delta t f(y^m, m\Delta t) \ , \quad m = M, M-1, M-2, ... \tag{1.18}$$

Example 2. *We consider the linear system*

$$y'(t) \ = \ ry(t) \ , \quad y(0) = 1, \tag{1.19}$$

which we wish to solve numerically for $t > 0$. Evidently the exact value of the solution at time $T > 0$ is $y(T) = e^{rT}$. Implementing the explicit Euler method (1.16) we see that

$$y^{m+1} \ = \ [1 + r\Delta t]y^m \quad m = 0, 1, 2, .., \ \ y^0 = 1. \tag{1.20}$$

On iterating this recurrence M times where $M\Delta t = T$, we obtain the approximate value for $y(T)$,

$$y(T) \sim y^M = [1 + r\Delta t]^M \ = \ [1 + r\Delta t]^{T/\Delta t} \ . \tag{1.21}$$

Since we know from calculus that

$$\lim_{\Delta t \to 0} [1 + r\Delta t]^{T/\Delta t} \ = \ e^{rT} \ , \tag{1.22}$$

we conclude that the numerical solution of the Euler scheme converges as $\Delta t \to 0$ to the exact solution of the ODE.

A numerical scheme for the solution of the differential equation (1.1) which converges as the time discretization $\Delta t \to 0$ to the solution of the ODE is called a *convergent numerical scheme*. Evidently we should always want to use only convergent schemes. Another related notion is also important but is considerably more subtle. In numerical analysis we are not only interested in coming up with convergent schemes, but also in the issue of how accurate these schemes are for a given number of computations. Thus we introduce the notion of *stability* of a numerical scheme. We say the scheme has an interval of stability $\Delta t < \delta$ if for Δt in this interval the numerical solution and the exact solution of the ODE are "reasonably close". The larger we can choose δ the more stable the scheme is. We have not of course defined what "reasonably close" is, but in our applications it will mean that the numerical solution *diverges* outside the region of stability.

We consider again the problem of solving the ODE (1.1) for $t > 0$ with initial data $y(0) = y^0$. If instead of the forward difference as in (1.15), (1.16) we use the backward difference to approximate the derivative we now get (1.17), (1.18) but we wish to solve this *forward* in time, and so the scheme is *implicit*. To compute y^m

from the already computed value for y^{m-1} we need to solve the equation (1.18) for y^m. In Example 2 it is easy to do this since the equation is

$$y^{m-1} \ = \ [1 - r\Delta t]y^m \ , m = 0, 1, ... \tag{1.23}$$

Thus for this *implicit Euler method* we obtain instead of (1.21) the formula

$$y(T) \sim y^M = [1 - r\Delta t]^{-M} \ = \ [1 - r\Delta t]^{-T/\Delta t} \ . \tag{1.24}$$

It is clear from (1.22), (1.24) that the implicit Euler method is also convergent. Its stability properties however can be much better than the explicit Euler method in the case when $r << 0$.

To see why this is the case we consider a system of N equations

$$\frac{dy_1(t)}{dt} = -\lambda_1 y_1(t), \cdots \frac{dy_N(t)}{dt} = -\lambda_N y_N(t) \ , \tag{1.25}$$

which we wish to solve forward in time $t > 0$ with initial data

$$y_1(0) = y_1^0, \cdots y_N(0) = y_N^0 \ . \tag{1.26}$$

We assume that N is large and that $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_N$, where λ_N is an increasing function of N. Since the system (1.25) is decoupled we readily see as in Example 2 that the explicit Euler method yields the numerical value for the solution $y(T) = [y_1(T), ..., y_N(T)]$ of (1.25), (1.26) at time $T > 0$ to be given by

$$y_j(T) \sim [1 - \lambda_j \Delta t]^M y_j^0, \quad j = 1, .., N, \quad M = T/\Delta t \ . \tag{1.27}$$

The exact solution to (1.25), (1.26) is of course

$$y_j(T) = e^{-\lambda_j T} y_j^0, \quad j = 1, .., N. \tag{1.28}$$

Note that if $\lambda_N \Delta t > 2$ then the numerical value (1.27) for $y_N(T)$ is growing exponentially in M since in that case $1 - \lambda_N \Delta t < -1$. It also oscillates between large positive and large negative values. In contrast the exact solution $y_N(T)$ is exponentially smaller in T than the initial data y_N^0. We conclude that if $\lambda_N \Delta t > 2$ the numerical solution of (1.27), (1.28) bears no resemblance to the exact solution. Hence the stability interval for the explicit Euler method is $\Delta t < 2/\lambda_N$. If we implement the implicit Euler method for the problem the numerical value for the solution of (1.25), (1.26) is given as in (1.24) by

$$y_j(T) \sim [1 + \lambda_j \Delta t]^{-M} y_j^0, \quad j = 1, .., N, \quad M = T/\Delta t \ . \tag{1.29}$$

One immediately sees that (1.29) always gives a reasonable approximation to the exact solution (1.28) no matter how large Δt. The interval of stability for the implicit scheme is therefore $\Delta t < \infty$. Implicit schemes can be much more stable than explicit schemes if there are *many time scales* in the problem. Here we think of $1/\lambda_j$ as a time scale since $y_j(t)$ decreases in size by 50% in the time interval $0 < t < 1/\lambda_j$.

In our numerical scheme for solving the Black-Scholes equation we shall need to numerically solve a large number N of *coupled* ODE's. These come about because the BS equation is a partial differential equation (PDE) similar to the *heat equation*

$$\frac{\partial u(x,t)}{\partial t} \ = \ \frac{\partial^2 u(x,t)}{\partial x^2} \ . \tag{1.30}$$

A typical problem is to solve (1.30) for $t > 0$ in the interval $0 < x < 1$ with specified initial condition at $t = 0$ and *boundary conditions* at $x = 0, 1$. If we think in terms of heat flow then the interval $0 < x < 1$ represents a bar of length 1 with initial

temperature distribution given by some known function $u(x,0),\ 0<x<1$. The temperature at the ends of the bar $x=0,1$ are kept fixed in time at temperatures u_0, u_1 respectively, so that $u(0,t)=u_0,\ u(1,t)=u_1$ for $t>0$. We can find the temperature distribution $u(x,t),\ 0<x<1$, within the bar at a later time t by solving (1.30) with the given initial and boundary conditions. To numerically solve this problem we introduce a space discretization Δx such that $N\Delta x=1$ for some large integer N. In that case we can approximate the PDE (1.30) by a system of $N-1$ ODEs in $u_n(t)\sim u(n\Delta x,t),\ n=1,..N-1$. This ODE system is like (1.25) with $\lambda_n=\pi^2n^2,\ n=1,..,N-1$, whence the stability condition $\Delta t<2/\lambda_N$ is essentially $\Delta t<2/\pi^2N^2$. Since $N\Delta x=1$ we conclude that the stability condition can be written as $\Delta t/(\Delta x)^2<2/\pi^2$. Hence a stable explicit Euler method for solving the heat equation requires that the time discretization is *proportional to the square of the space discretization*. We shall see this condition come up many times during the course.

The explicit and implicit Euler methods for solving ODEs are first order accurate in time. We introduce now a method which is *second order accurate*. It is called the trapezoid rule for it is a generalization of the trapezoid rule for numerically estimating integrals. Suppose we wish to solve (1.1) forward in time with given initial condition $y(0)=y^0$. Setting $y^m\sim y(m\Delta t)$, the numerical algorithm for the generalized trapezoid rule is

$$y^{m+1}=y^m+\frac{\Delta t}{2}\left[f(y^m,m\Delta t)+f(y^{m+1},(m+1)\Delta t)\right]\ . \tag{1.31}$$

Consider now applying the explicit Euler method (1.16) and the trapezoid rule (1.31) when the function $f(y,t)$ in (1.1) depends only on t so $f(y,t)=f(t)$. If $M\Delta t=T$ for some integer M then we have that

$$y^M\sim y^0+\int_0^T f(t)\ dt\ , \tag{1.32}$$

so the numerical algorithm for solving the ODE (1.1) yields a numerical algorithm for integrating the function $f(\cdot)$. Evidently the explicit Euler method (1.16) gives the rectangle rule for integration which is first order accurate, and (1.31) the trapezoid rule which is second order accurate. The algorithm (1.31) continues to be second order accurate in the general case when $f(y,t)$ depends explicitly on y as well as t. The algorithm is also implicit, and because of that it has better stability properties than the explicit Euler method for the system (1.25). Thus we have from (1.31) that

$$y_j^{m+1}=y_j^m-\frac{\Delta t}{2}\left[\lambda_j y_j^m+\lambda_j y_j^{m+1}\right]\ ,\quad j=1,..,N, \tag{1.33}$$

whence we conclude that

$$y_j^m\ =\ \left(\frac{1-\lambda_j\Delta t/2}{1+\lambda_j\Delta t/2}\right)^m\ y_j^0\ ,\quad j=1,..,N. \tag{1.34}$$

Since $\lambda_j>0$ it follows that $|y_j^m|\le|y_j^0|$ and so the trapezoid algorithm is stable for $\Delta t<\infty$. Just as we can apply the explicit Euler algorithm to solve an initial-boundary value problem for the heat equation (1.30), we can also apply the trapezoid algorithm to this problem. When applied to numerically solving parabolic PDE such as (1.30) the trapezoid algorithm is known as the Crank-Nicolson method. In the PDE case the problem of solving the implicit equation (1.31) for

the N dimensional vector y^{m+1} is now by no means trivial as it was for (1.33). We need to solve a linear coupled system of N equations where N is large. Standard methods like Gauss elimination are much too inefficient to be used for this so one needs to make use of other better methods.

We conclude our discussion of numerical methods for solving ODEs by considering how to estimate the order of accuracy of a given method. Let us consider the forward in time explicit Euler method (1.16). If $y(t),\ t \geq 0$, is the exact solution to (1.1) then $y(m\Delta t),\ m = 0, 1, 2, ..,$ is an approximate solution to (1.16) in the sense that

$$y((m+1)\Delta t) = y(m\Delta t) + \Delta t f(y(m\Delta t), m\Delta t) + O[(\Delta t)^2] \ , \quad m = 0, 1, 2, ..., \tag{1.35}$$

by using the Taylor expansion

$$y(t + \Delta t) \ = \ y(t) + y'(t)\Delta t + O[(\Delta t)^2] \ . \tag{1.36}$$

For any $T > 0$ we let $\varepsilon_T(\Delta t) = y^M - y(T)$ where $y^m,\ m = 0, 1, 2, ..,$ is the solution to (1.16) and $M\Delta t = T$. Thus $\varepsilon_T(\Delta t)$ is the error at time T between the numerical solution computed by explicit Euler and the exact solution of the ODE (1.1) with the same initial data at $t = 0$. Comparing (1.16) and (1.35) we see that the error between the numerical and exact solutions grows by $O[(\Delta t)^2]$ at each time step. Hence the error at time T should be

$$\varepsilon_T(\Delta t) \ \sim \ MO[(\Delta t)^2] \ = \ O[\Delta t] \quad \text{since } M = T/\Delta t \ , \tag{1.37}$$

whence we conclude that the Euler method is first order accurate. We refer to the error in (1.35) where we regard the *exact solution* of the ODE (1.1) as an *approximate solution* of the Euler scheme (1.15) as the *truncation error* for the scheme. The error $\varepsilon_T(\Delta t)$ between the numerically computed solution and exact solution at time T is referred to as the *cumulative error*. Then (1.37) is a consequence of the general fact that

(1.38)

cumulative error $=$ one order of magnitude less than the truncation error.

2. Numerical Algorithms for solving the Black-Scholes Equation

We recall the BS theory for the pricing of options. Consider a simple option on a stock with expiration date $T > 0$, where $t = 0$ represents today. In the case of a call option with strike price K the payoff on the option at the expiration date is $\max[S - K, 0]$, where S is the stock price. To find the value of the option we solve the *terminal value* problem for the partial differential equation (PDE)

$$\frac{\partial V(S,t)}{\partial t} + \frac{\sigma^2}{2}S^2\frac{\partial^2 V(S,t)}{\partial S^2} + rS\frac{\partial V(S,t)}{\partial S} - rV(S,t) \ = 0 \quad \text{for } S > 0,\ t < T, \tag{2.1}$$

$$V(S,T) \ = \ \max[S - K, 0] \quad \text{for } S > 0. \tag{2.2}$$

In (2.1) the stock volatility is $\sigma > 0$ and $r > 0$ is the risk-free rate of interest. If the price of the stock today is S_0 then the value of the option today is $V(S_0, 0)$, and the dynamic hedging strategy which achieves this price without risk is given by the hedging portfolio:

(2.3)

$$\text{Hedging Portfolio when stockprice} = S, \text{ time} = t: \text{ one option} - \frac{\partial V(S,t)}{\partial S} \text{ of stock.}$$

Observe that the Black Scholes PDE (2.1) has coefficients which depend on S. We can transform the equation into a *constant coefficient* PDE by making the change of variable $S = e^x$. Hence if we write $V(S,t) = u(x,t)$ then the problem (2.1), (2.2) becomes the terminal value problem

(2.4)
$$\frac{\partial u(x,t)}{\partial t} + \frac{\sigma^2}{2}\frac{\partial^2 u(x,t)}{\partial x^2} + \left\{ r - \frac{\sigma^2}{2} \right\} \frac{\partial u(x,t)}{\partial x} - ru(x,t) \;=\; 0 \quad \text{for } -\infty < x < \infty,\; t < T,$$

$$u(x,T) \;=\; \max[e^x - K, 0] \quad \text{for } -\infty < x < \infty. \tag{2.5}$$

The BS equation (2.1) is a consequence of the assumption that the random evolution of the stock price $S(t)$ in the *risk neutral measure* is governed by the stochastic differential equation (SDE)

$$dS(t) \;=\; S(t)[rdt + \sigma dB(t)], \tag{2.6}$$

where $B(\cdot)$ is *Brownian motion* (BM). If we set $S(t) = \exp[X(t)]$ then it follows from (2.6) via the *Ito calculus* that

$$dX(t) \;=\; (r - \sigma^2/2)dt + \sigma dB(t)\ . \tag{2.7}$$

We can easily solve (2.7) to obtain

$$X(t) \;=\; X(0) + (r - \sigma^2/2)t + \sigma B(t)\ , \tag{2.8}$$

so the process $X(\cdot)$ is a translation of Brownian motion. Since $S(\cdot)$ is the exponential of $X(\cdot)$ we refer to $S(\cdot)$ as *geometric Brownian motion* (GBM). The connection between GBM and solutions to (2.1), (2.2) is that the solution of the terminal value problem can be written as an *expectation value*

$$V(S,0) \;=\; e^{-rT} E\left[V(S(T),T) \mid S(0) = S \right]\ . \tag{2.9}$$

In the case of the call option we have from (2.2) that

$$V(S(T),T) \;=\; \max[S(T) - K, 0]\ . \tag{2.10}$$

Similarly the solution to (2.4), (2.5) can be written as

$$u(x,0) \;=\; e^{-rT} E\left[u(X(T),T) \mid X(0) = x \right]\ , \tag{2.11}$$

where (2.5) implies that

$$u(X(T),T) \;=\; \max[e^{X(T)} - K, 0]\ . \tag{2.12}$$

To numerically solve the BS terminal value problem (2.1), (2.2) it will be sufficient for us to numerically solve the constant coefficient problem (2.4), (2.5). In order to do this we need to introduce a "space" discretization Δx for the logarithm x of the stock price as well as a time discretization Δt, which we have already done for numerically solving ODEs. We also need to reduce the *infinite interval* $-\infty < x < \infty$ to a *finite interval* $a < x < b$. First note that the finite interval should certainly include *at the money options* so we need to have $a < \log K < b$. Next we choose a sufficiently small so that the option with initial stock price $S_0 = e^a$ is so far out of the money that we can set $V(e^a,t) = u(a,t) = 0$ for $0 \le t \le T$. To decide on a good value for a we consider the representation (2.11). From (2.8) we see that the random variable $X(T)$ conditioned on $X(0) = x$ is Gaussian with

$$\text{mean} \;=\; x + (r - \sigma^2/2)T, \quad \text{variance} \;=\; \sigma^2 T\ . \tag{2.13}$$

The probability of the Gaussian variable $X(T)$ being larger than 3 standard deviations above its mean is 0.0014 so about 1/10 of 1 percent. Thus if we take

$a = \log K - 3\sigma\sqrt{T}$ then the probability that the option is in the money at expiration is very small, and hence we are justified in setting $u(a,t) = 0$ for $0 \leq t \leq T$. Note here than we have ignored the term $(r-\sigma^2/2)T$ in (2.13) since it will normally be insignificant compared to $3\sigma\sqrt{T}$.

To find an appropriate value for b and the value we should set for $u(b,t)$ we use *put-call parity*. Put-call parity can be seen as a consequence of the fact that the function

$$V(S,t) = S - Ke^{-r(T-t)}, \quad S > 0, t < T \tag{2.14}$$

is a solution of the BS PDE (2.1). It is clear that (2.14) is the value of a portfolio consisting of one stock minus K in cash at time T. Since we have that

$$1 \text{ stock } - K \text{ cash at time } T \ = 1 \text{ call option } - \ 1 \text{ put option }, \tag{2.15}$$

the call-put parity equation is obtained by setting the value of the LHS of (2.15) to be the RHS of (2.14). Observe now that if we set $b = \log K + 3\sigma\sqrt{T}$ then the put option with strike K and stock price $S = e^b$ is far out of the money so we can take its value to be 0. We conclude then from (2.14), (2.15) that for the call option terminal data (2.10), (2.12) we should set

$$u(b,t) \ = \ e^b - Ke^{-r(T-t)} \quad \text{for } 0 \leq t \leq T \quad \text{if } b = \log K + 3\sigma\sqrt{T}. \tag{2.16}$$

We can now estimate the value of the call option by solving (2.4) for $a < x < b$, $t < T$, with terminal data (2.5) and Dirichlet boundary conditions $u(a,t) = 0$, $t < T$, and $u(b,t)$, $t < T$, given by (2.16). To construct a numerical scheme we need to approximate the first two derivatives of $u(x,t)$ with respect to x by finite differences as well as $\partial u(x,t)/\partial t$. We shall use the central difference (1.12) to approximate the first derivative with respect to x so we set

$$\frac{\partial u(x,t)}{\partial x} \ \sim \ \frac{u(x+\Delta x,t) - u(x-\Delta x,t)}{2\Delta x} \ . \tag{2.17}$$

We have already observed that the central difference approximation is second order accurate, so the error in (2.17) is $O[(\Delta x)^2]$. We can also easily obtain a second order accurate approximation to the second derivative, which is

$$\frac{\partial^2 u(x,t)}{\partial x^2} \ \sim \ \frac{u(x+\Delta x,t) + u(x-\Delta x,t) - 2u(x,t)}{(\Delta x)^2} \ . \tag{2.18}$$

We can see why (2.18) is second order accurate by doing a Taylor expansion as in (1.13) but with more terms,

$$u(x+\Delta x) = u(x)+u'(x)[\Delta x]+u''(x)[\Delta x]^2/2+u'''(x)[\Delta x]^3/6+u''''(x)[\Delta x]^4/24+O[(\Delta x)^5]. \tag{2.19}$$

Hence we have that

$$u(x+\Delta x) + u(x-\Delta x) - 2u(x) = (\Delta x)^2\{u''(x) + u''''(x)(\Delta x)^2/12 + O[(\Delta x)^3]\} \ , \tag{2.20}$$

and so the difference between the LHS and RHS of (2.18) is $O[(\Delta x)^2]$. The finite difference approximation for the PDE (2.4) is completed now by taking the backward difference (1.11) as an approximation to the time derivative

$$\frac{\partial u(x,t)}{\partial t} \ \sim \ \frac{u(x,t) - u(x,t-\Delta t)}{\Delta t} \ . \tag{2.21}$$

The explicit backwards in time Euler method for the terminal value problem (2.4), (2.5) is then

$$u(x,t-\Delta t) \;=\; (p-r\Delta t)u(x,t)+p^+u(x+\Delta x,t)+p^-u(x-\Delta x,t)\;, \tag{2.22}$$

where p, p^+, p^- are given by the formulas

$$p=1-\sigma^2\Delta t/(\Delta x)^2,\quad p^+=\sigma^2\Delta t/2(\Delta x)^2+(r-\sigma^2/2)\Delta t/2\Delta x\;, \tag{2.23}$$
$$p^-=\sigma^2\Delta t/2(\Delta x)^2-(r-\sigma^2/2)\Delta t/2\Delta x\;.$$

To implement the explicit Euler method we choose Δx and Δt so that $M\Delta t=T$ for some integer M and $N\Delta x=b-a$ for some integer N. We set $u_n^m\simeq u(a+n\Delta x, m\Delta t)$ for $n=0,..,N$, and $m=0,..,M$. The terminal condition (2.5) then yields

$$u_n^M=\max[e^{a+n\Delta x}-K,0]\quad \text{for } n=0,..N. \tag{2.24}$$

The boundary conditions $u(a,t)=0,\ 0\le t\le T$, and (2.16) yield

$$u_0^m=0,\quad u_N^m=e^b-Ke^{-r(M-m)\Delta t}\quad \text{for } m=0,..,M. \tag{2.25}$$

The recurrence (2.22) is then given by

$$\begin{array}{rcl}\text{for } m & = & M:-1:1\\ \text{for } n & = & 1:N-1\\ u_n^{m-1} & = & (p-r\Delta t)u_n^m+p^+u_{n+1}^m+p^-u_{n-1}^m.\end{array} \tag{2.26}$$

We already discussed in §1 the issue of stability when using the explicit Euler method to numerically solve parabolic PDE such as (1.30), (2.4). There we observed that it is necessary to take $\Delta t/(\Delta x)^2=O(1)$ to ensure stability. We can see more clearly here why this is the case since the parameters p, p^+, p^- of (2.23) satisfy $p+p^++p^-=1$. If all three parameters are positive then it follows from (2.22) that

$$\sup_{a<x<b}|u(x,t-\Delta t)|\;\le\;[1-r\Delta t]\sup_{a\le x\le b}|u(x,t)|, \tag{2.27}$$

and since $M\Delta t=T$ we conclude that

$$\sup_{a<x<b}|u(x,0)|\;\le\;[1-r\Delta t]^M\sup_{a\le x\le b}|u(x,T)|\le e^{-rT}\sup_{a\le x\le b}|u(x,T)|\;, \tag{2.28}$$

where we are ignoring the effect of the boundary conditions. If one of the parameters p, p^+, p^- is negative then $|p|+|p^+|+|p^-|=1+\delta$ for some $\delta>0$. In that case we could have

$$\sup_{a<x<b}|u(x,0)|\;\simeq[1+\delta]^M\;=\;[1+\delta]^{T/\Delta t}\;, \tag{2.29}$$

which diverges as $\Delta t\to 0$. In fact this instability typically occurs if $|p|+|p^+|+|p^-|=1+\delta$ with $\delta>0$. The solution not only grows exponentially in $T/\Delta t$ but also oscillates. We cannot really go into an adequate explanation here except to note that the terminal data (2.5) contains high frequency oscillations. Although these oscillations are small the numerical algorithm amplifies them exponentially by $[1+\delta]^{T/\Delta t}$ in computing $u(x,0),\ a<x<b$. Evidently a necessary and sufficient condition for $p>0$ is that

$$\Delta t/(\Delta x)^2\;<\;1/\sigma^2\;. \tag{2.30}$$

Now the Euler method (2.22) is first order accurate in Δt and second order accurate in Δx. Hence the cumulative error is given by

$$u(a + n\Delta x, 0) - u_n^0 \;=\; O[\Delta t] + O[(\Delta x)^2] \quad \text{for } 0 \leq n \leq N. \tag{2.31}$$

It follows from (2.31) that for given Δx we should take Δt as large as possible consistent with (2.30) to minimize the number of computations without substantially increasing the cumulative error. In that case we see from (2.23) that p^+, p^- are also positive. Thus to implement explicit Euler in an optimal way we should choose Δx so that $O[(\Delta x)]^2]$ is an acceptable error. Then we choose Δt such that there is almost equality in (2.30).

If S_0 is today's stock price then the BS value of the call option is $u(x_0, 0)$ where $S_0 = e^{x_0}$. It may happen that x_0 is not a grid point for the numerical method, so for example there is a non-negative integer $n_0 < N$ such that $a + n_0\Delta x = x_0^- < x_0 < a + (n_0 + 1)\Delta x = x_0^+$. In that case we approximate the value of $u(x_0, 0)$ by *interpolation* from the numerically computed values of the function on the grid points. If we use *linear interpolation* then we set

$$u(x_0, 0) \;\simeq\; \{[x_0^+ - x_0]u(x_0^-, 0) + [x_0 - x_0^-]u(x_0^+, 0)\}/\Delta x \; . \tag{2.32}$$

Linear interpolation has the advantage that it is completely *local* in the sense that the value taken for $u(x_0, 0)$ is just a weighted average of the values of the function $u(\cdot, 0)$ on the two grid points closest to x_0. There is also a disadvantage in that it is only a first order accurate interpolation. This means that if $u(x, 0),\ a < x < b$, is a differentiable function then the difference between the LHS and RHS of (2.32) is $O(\Delta x)$. In our situation $u(x, 0)$ is computed at grid points x correct to second order, so the error at a grid point between the numerically computed solution and the solution to the continuous problem is $O[(\Delta x)^2]$. Evidently we wish to use an interpolation scheme which preserves the second order accuracy away from grid points. Linear interpolation cannot achieve this but *spline interpolation* will. It does this by approximating the function between grid points by a polynomial -say of degree 3 as in the case of *cubic splines*. Because approximation by polynomials give some extra degrees of freedom we can make the approximating function differentiable across grid points. Note that in the case of linear interpolation there are no extra degrees of freedom and so in general the approximating function has a jump in its derivative across a grid point. There is a cost to this extra accuracy of spline interpolation in that spline interpolation is *non-local* in the sense that the interpolated value for $u(x_0, 0)$ depends now on the values of $u(\cdot, 0)$ at *all* the grid points, not just at the two neighboring grid points of x_0. The spline interpolation is however *almost local* in the sense that the dependence of $u(x_0, 0)$ on a grid point far from x_0 is very small.

3. American Options

Options can have early exercise features, which means that one can exercise the option at any time up to the expiration date. We shall consider here the simple American put option with strike price K and expiration date T. Letting $V(S, t)$ be the value of the option at time $t \leq T$ when the stock price is S, then as in (2.2) we have that

$$V(S, T) \;=\; \max[K - S, 0] \quad \text{for } S > 0. \tag{3.1}$$

Since the value of the option cannot drop below the early exercise price we also have that

$$V(S,t) \;\geq\; \max[K-S,0] \quad \text{for } S>0,\; t\leq T. \tag{3.2}$$

Complementary to (3.2) is the equation

(3.3) The BS equation (2.1) for $V(S,t)$ holds when $V(S,t) > \max[K-S,0]$.

Evidently the value of the American option is at least the value of the corresponding European option, but it generally has only slightly greater value. In fact by a simple no arbitrage argument one can see that if the rate of interest $r=0$ then the value of the American option equals the value of the European. For $r>0$ it follows from (3.2), (3.3) that there is a curve $t \to S_{\text{exer}}(t),\; 0\leq t\leq T$, with the property that $0 < S_{\text{exer}}(t) \leq K$ for $0\leq t\leq T$, such that

(3.4) $V(S,t)$ satisfies (2.1) for $S > S_{\text{exer}}(t)$; $\quad V(S,t) = K-S$ for $S < S_{\text{exer}}(t)$.

The graph $\{(t,S) \,:\, S = S_{\text{exer}}(t),\; 0\leq t\leq T\}$ is known as the *early exercise boundary* since it divides the half infinite region $\{(t,S) : 0\leq t\leq T, S>0\}$ into two subregions. In the lower subregion it is optimal always to immediately exercise the option and in the upper subregion to wait until either the expiration date of the option or until the stock price hits the early exercise boundary. Note from Figure 1 that $S_{\text{exer}}(t)$ is an increasing and convex function of t with $\lim_{t\to T} S_{\text{exer}}(t) = K$. It is intuitively clear that $S_{\text{exer}}(t)$ is increasing and $\lim_{t\to T} S_{\text{exer}}(t) = K$, but the convexity of the graph is a subtle property. In fact even the increasing property cannot be proved in a straightforward way although the intuition behind it is simple i.e. that for a given stock price $S<K$ the longer the time to expiration, the more it makes sense to wait rather than immediately exercise the option.

We consider some properties of the value $V(S,t)$ of the option on the early exercise boundary. In particular we have that

$$V(S,t) \;=\; K-S, \quad \frac{\partial V(S,t)}{\partial S} \;=\; -1, \qquad \text{for } S = S_{\text{exer}}(t)\ . \tag{3.5}$$

The first identity of (3.5) states that the value of the option is continuous across the early exercise boundary, while the second identity states that the hedging portfolio is continuous across the boundary. The first identity is then just a consequence of the fact that the value of the option is a continuous function of (S,t). We can give an intuitive argument for the second identity based on the *no arbitrage* assumption underlying the BS theory. To show the second identity we first assume that

$$\lim_{S\to S_{\text{exer}}(t)^+} \frac{\partial V(S,t)}{\partial S} \;<\; -1\ . \tag{3.6}$$

Hence for $S > S_{\text{exer}}(t)$ with $S - S_{\text{exer}}(t)$ small then $V(S,t) < K-S$, which contradicts the inequality (3.2). Alternatively let us suppose that

$$\lim_{S\to S_{\text{exer}}(t)^+} \frac{\partial V(S,t)}{\partial S} \;>\; -1\ . \tag{3.7}$$

Now above the early exercise boundary the BS theory applies so for $S > S_{\text{exer}}(t)$ we have that

$$\text{one put option} \;-\; \frac{\partial V(S,t)}{\partial S} \text{ of stock} \;=\; \text{risk free portfolio.} \tag{3.8}$$

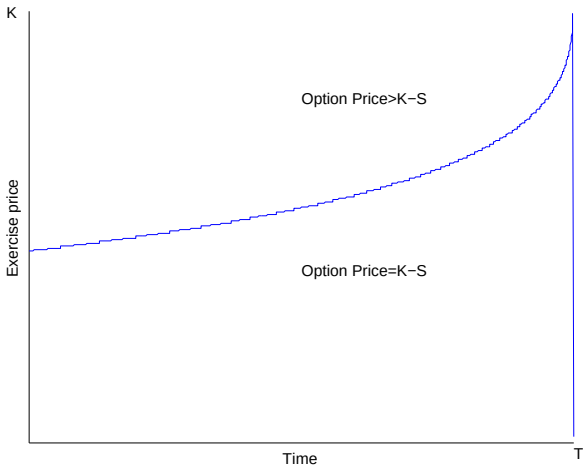

FIGURE 1. Graph of Early Exercise Boundary

Consider now what happens when S decreases to $S_{\text{exer}}(t)$ and (3.7) holds. Then the portfolio consists of

$$\text{one put option} + \alpha(\text{stock}) \quad \text{with } \alpha < 1. \tag{3.9}$$

If S continues to decrease below $S_{\text{exer}}(t)$ then it is clear that the portfolio (3.9) yields a profit. If S instead increases above $S_{\text{exer}}(t)$ then we are in the BS regime

where the value of the portfolio increases at the risk free rate. We conclude that if (3.7) holds then the portfolio (3.8) is an arbitrage since it can make a profit but cannot make a loss. This is again a contradiction to the no arbitrage assumption. We have therefore shown that neither of (3.6), (3.7) can hold, which implies the second identity of (3.5).

The mathematical analysis of (3.2), (3.3) is complicated since the early exercise boundary is a so called *free boundary*. This means it is not known *a priori* but comes as part of the solution to the problem. It is then somewhat surprising that the numerical algorithm for solving the American option problem is virtually the same as the algorithm for solving the European option problem. In fact all we do is to modify the European algorithm to insure that (3.2) holds. Thus we have instead of (2.22),

(3.10)
$$u(x,t-\Delta t) \;=\; \max\{K-e^x,\; (p-r\Delta t)u(x,t)+p^+u(x+\Delta x,t)+p^-u(x-\Delta x,t)\}\;,$$

where again $V(S,t)=u(x,t)$ with $S=e^x$. The terminal and boundary conditions are given by

$$u(x,T) \;=\; K-e^x, \quad a<x<b, \tag{3.11}$$

$$u(a,t) \;=\; K-e^a, \quad u(b,t) \;=0, \qquad 0\le t<T, \tag{3.12}$$

where a,b are as for the European option. The boundary condition (3.12) at a comes from the fact that a lies below the early exercise boundary for all $0\le t\le T$. Note that cash K is *not discounted* in the boundary condition (3.12), whereas in the European case (2.16) it is discounted.

Finally we note how to compute the early exercise boundary once we have obtained the function $V(S,t),\; S>0, 0\le t\le T$. If $u_n^m \simeq u(a+n\Delta x, m\Delta t)$ then for each fixed m we find the minimum $n=\mathcal{N}(m)$ such that $u_n^m > K-e^{a+n\Delta x}$. We obtain the early exercise boundary as in Figure 1 by graphing the function $(m\Delta t, e^{a+\mathcal{N}(m)\Delta x}),\; m=0\cdots M$.

4. Path Dependent Options

A path dependent option is one in which the payoff depends not only on the stock price at the expiration date of the option, but also on the values of the stock price during the lifetime of the option. For a *lookback option* the payoff depends on the maximum or minimum of the stock price over the life of the option. In the case of an Asian option the payoff depends on an average of the stock price during the lifetime of the option. We shall consider here just the Asian option. For the continuously sampled Asian option the payoff at expiration is given by the formula

$$\text{payoff} \;=\; \left[S(T)-\frac{1}{T}\int_0^T S(t)\;dt\right]^+ . \tag{4.1}$$

We can price this using the BS formalism by introducing a new variable I in addition to S,t by the formula

$$I(t) \;=\; \int_0^t S(t')\;dt'\;, \tag{4.2}$$

and consider the value of the option as being a function $V(S,I,t)$ of 3 variables S,I,t now. The value of the option today $t=0$ when the stock price is S_0 is

then $V(S_0, 0, 0)$. We can derive a PDE for $V(S, I, t)$ since (2.6), (4.2) imply that $[S(t), I(t)]$ are a solution to the system of SDEs

$$dS(t) \ = \ S(t)[rdt + \sigma dB(t)], \qquad dI(t) \ = \ S(t) \ dt \ . \tag{4.3}$$

To price the option we construct a portfolio as in (2.3), but observe now that the partial derivative $\partial V(S, I, t)/\partial S$ implies that we compute the *delta* of the option with both t and I fixed. From (4.3) if $\pi(t)$ is the value of the portfolio at time t then the Ito calculus implies that

$$d\pi(t) \ = \ \left[\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + S\frac{\partial V}{\partial I}\right] \ dt \ . \tag{4.4}$$

Following the standard BS argument we see that (4.4) implies the portfolio is risk free, whence from the no arbitrage assumption we conclude that $d\pi(t) = r\pi(t) \ dt$. Hence the function $V(S, I, t)$ satisfies the PDE

$$\frac{\partial V(S,I,t)}{\partial t} + \frac{\sigma^2}{2} S^2 \frac{\partial^2 V(S,I,t)}{\partial S^2} + S\frac{\partial V(S,I,t)}{\partial I} + rS\frac{\partial V(S,I,t)}{\partial S} - rV(S,I,t) \ = 0 \ . \tag{4.5}$$

The equation (4.5) holds for $S, I > 0$ and $t < T$. From (4.1) we see that the terminal condition for the problem is

$$V(S, I, T) \ = \ \max[S - I/T, 0] \quad \text{for } S, I > 0. \tag{4.6}$$

Just as for (2.1), (2.2) we also need boundary conditions to uniquely determine the solution. One boundary condition is obvious, namely that

$$\lim_{S \to 0} V(S, I, t) \ = \ 0, \tag{4.7}$$

since this merely states that if the price of the stock is small then not much can be gained by betting on the difference between average stock price and the terminal stock price. We also need to impose three other boundary conditions corresponding to $S \to \infty$ and also $I \to 0, \ I \to \infty$. It is not so easy to see what these boundary conditions should be.

We can reduce the number of variables (S, I, t) to 2 variables by observing that the function $V(S, I, t)$ is homogeneous. To see this let $\lambda > 0$ and consider the function $V_\lambda(S, I, t) = \lambda^{-1} V(\lambda S, \lambda I, t)$. It is easy to see that $V_\lambda(S, I, t)$ satisfies (4.5),(4.6) and so we conclude from general uniqueness considerations for the terminal value problem (4.5),(4.6) that V_λ is independent of $\lambda > 0$. Choosing $\lambda = S^{-1}$ we conclude that

$$V(S, I, t) \ = \ Sw(\xi, t), \quad \xi = I/S \ , \qquad \text{for } I, S > 0, \ t < T, \tag{4.8}$$

where $w(\xi, t)$ is a function of two variables and

$$w(\xi, T) \ = \ \max[1 - \xi/T, 0] \quad \xi > 0. \tag{4.9}$$

We can obtain from (4.5) a PDE which the function $w(\xi, t)$ satisfies. Using the chain rule we have that

$$\begin{aligned} \frac{\partial V(S,I,t)}{\partial t} \ &= \ S\frac{\partial w(\xi,t)}{\partial t} \ , \quad \frac{\partial V(S,I,t)}{\partial S} \ = \ w(\xi,t) - \xi\frac{\partial w(\xi,t)}{\partial \xi} \ , \\ \frac{\partial V(S,I,t)}{\partial I} \ &= \ \frac{\partial w(\xi,t)}{\partial \xi} \ , \quad S\frac{\partial^2 V(S,I,t)}{\partial S^2} \ = \ \xi^2 \frac{\partial^2 w(\xi,t)}{\partial \xi^2} \ . \end{aligned} \tag{4.10}$$

Hence (4.5) implies that $w(\xi,t)$ is a solution to the PDE

$$\frac{\partial w(\xi,t)}{\partial t} + \frac{1}{2}\sigma^2\xi^2\frac{\partial^2 w(\xi,t)}{\partial \xi^2} + (1-r\xi)\frac{\partial w(\xi,t)}{\partial \xi} \;=\; 0\ . \tag{4.11}$$

The boundary condition (4.7) turns into the boundary condition

$$\lim_{\xi\to\infty} w(\xi,t) \;=\; 0, \quad \text{for } t<T. \tag{4.12}$$

Now (4.7) and (4.12) are not equivalent since (4.7) actually only implies that $\lim_{\xi\to\infty} w(\xi,t)/\xi = 0$, so (4.12) is a stronger assumption than (4.7). The assumption (4.12) however makes sense if we think of $w(\xi,t)$ as being the value of a type of put option. Comparing (4.11), (4.9) to (2.1), (3.1) respectively, we see that $w(\xi,t)$ is like the value of a put option with strike price T. With that analogy then (4.12) states that the value of the option is roughly zero if it is far out of the money.

In order to have a unique solution we need to specify a boundary condition for $w(\xi,t)$ as $\xi\to 0$. Note that the value of the Asian option today is $V(S_0,0,0) = S_0 w(0,0)$ so the boundary condition at $\xi=0$ cannot be *explicit* as in the case of the standard European put option. We can obtain an *implicit* boundary condition by setting $\xi=0$ in the PDE (4.11). Thus we have that

$$\frac{\partial w(\xi,t)}{\partial t} + \frac{\partial w(\xi,t)}{\partial \xi} \;=\; 0 \quad \text{for } \xi=0,\ t<T. \tag{4.13}$$

We can understand the condition (4.13) better if we write the function $w(\xi,t)$ as an expectation value similar to (2.9). Thus letting $\xi(t),\ t>0$, be the solution to the SDE

$$d\xi(t) \;=\; [1-r\xi(t)]\,dt + \sigma\xi(t)dB(t)\ , \tag{4.14}$$

then we have that

$$w(\xi,t) \;=\; E\left[\ w(\xi(T),T)\ \middle|\ \xi(t)=\xi\ \right] \qquad \text{for } t<T. \tag{4.15}$$

More generally we have that

$$w(\xi,t) \;=\; E\left[\ w(\xi(t+\Delta t),t+\Delta t)\ \middle|\ \xi(t)=\xi\ \right] \qquad \text{for } \Delta t>0, \tag{4.16}$$

so the representation (4.15) follows upon setting $\Delta t = T-t$. Now if ξ is small and Δt is also small the SDE (4.14) becomes approximately $d\xi(s) = ds$ for $t<s<t+\Delta t$, whence $\xi(t+\Delta t)\simeq \xi+\Delta t$. Note that $\xi(t+\Delta t)>\xi$, which is a consequence of the fact that the drift term $1-r\xi$ in (4.14) pulls a path $\xi(t)$ away from 0, so a solution to (4.14) never actually hits 0. Now from (4.16) we have approximately

$$w(\xi,t) \;\simeq\; w(\xi+\Delta t,t+\Delta t) \quad \text{if } \xi,\Delta t \text{ are small}, \tag{4.17}$$

so we conclude that

$$\lim_{\Delta t\to 0}\frac{w(\Delta t,t+\Delta t)-w(0,t)}{\Delta t} \;=\; 0\ , \tag{4.18}$$

and this yields (4.13).

To numerically solve the terminal value problem (4.9), (4.11) with the boundary conditions (4.12), (4.13) we need to first replace the infinite interval $\{0<\xi<\infty\}$ by a finite interval $\{0<\xi<\xi_{\max}\}$. We choose $\xi_{\max}$ depending on volatility similarly to the way we did for pricing the European option. We already mentioned that $w(\xi,t)$ is like the price of a put option with strike price T. Thus we wish to choose $\xi_{\max}$ sufficiently large so that the option is far out of the money at expiration, whence we may set $w(\xi_{\max},t)=0$ for $0\le t<T$. Evidently we need to have $\xi_{\max}>T$ and

how much larger depends on σ. Comparing the SDEs (2.6) and (4.14) we see that if $\xi(0) = \xi_{\max}$ then $\log \xi(T)$ is approximately Gaussian with

$$E[\ \log \xi(T)\ ] \ = \ \log \xi_{\max}, \qquad \mathrm{Var}[\ \log \xi(T)\ ] \ = \ \sigma^2 T\ . \tag{4.19}$$

Thus we want $\xi_{\max}$ to satisfy the inequality

$$\xi_{\max} e^{-3\sigma\sqrt{T}} \ \geq \ T, \quad \text{so we take } \xi_{\max} = T e^{3\sigma\sqrt{T}}\ . \tag{4.20}$$

In the above argument we have ignored the drift term $1 - r\xi$ in (4.14), but this term actually helps since it is positive (assuming r small) and so pushes the path $\xi(t)$ away from the threshold T for the option to be in the money.

We use the explicit Euler method to numerically solve the problem. Thus we have similarly to (2.22) the recurrence equation

$$w(\xi, t - \Delta t) \ = \ p(\xi) w(\xi, t) + p^+(\xi) w(\xi + \Delta\xi, t) + p^-(\xi) w(\xi - \Delta\xi, t)\ , \tag{4.21}$$

where $p(\xi), p^+(\xi), p^-(\xi)$ are given by the formulas

$$p(\xi) = 1 - \sigma^2\xi^2\Delta t/(\Delta\xi)^2, \quad p^+(\xi) = \sigma^2\xi^2\Delta t/2(\Delta\xi)^2 + (1 - r\xi)\Delta t/2\Delta\xi\ , \tag{4.22}$$
$$p^-(\xi) = \sigma^2\xi^2\Delta t/2(\Delta\xi)^2 - (1 - r\xi)\Delta t/2\Delta\xi\ .$$

Note that $p(\xi) + p^+(\xi) + p^-(\xi) = 1$ for $0 < \xi < \xi_{\max}$. To implement the boundary condition (4.13) we approximate the derivative $\partial w(\xi, t)/\partial\xi$ at $\xi = 0$ by a *forward difference* so (4.13) becomes

$$\frac{w(0,t) - w(0, t - \Delta t)}{\Delta t} + \frac{w(\Delta\xi, t) - w(0,t)}{\Delta\xi} \ = \ 0, \tag{4.23}$$

which implies that

$$w(0, t - \Delta t) \ = \ [1 - \Delta t/\Delta\xi] w(0,t) + (\Delta t/\Delta\xi) w(\Delta\xi, t)\ . \tag{4.24}$$

Similarly to (2.30) we need to impose the stability condition $p(\xi) > 0$ for $0 < \xi < \xi_{\max}$, which implies

$$\Delta t/(\Delta\xi)^2 \ < \ 1/\sigma^2\xi_{\max}^2\ . \tag{4.25}$$

Recall that in (2.17) we used the symmetric difference to approximate the first space derivative of the value function. The reason for doing this was to preserve the second order accuracy of the numerical scheme in the space difference. In (4.21), (4.22) we have used the symmetric difference so the error should be $O[(\Delta\xi)^2]$. In contrast to this we are forced to use the forward difference to implement the boundary condition (4.13) as in (4.24). Since forward difference is just first order accurate, this will tend to make the scheme only first order accurate in the space difference, so the error would be $O(\Delta\xi)$ and not $O[(\Delta\xi)^2]$. In addition the scheme (4.21), (4.22) is not stable for all ξ in the interval $[0, \xi_{\max}]$ even assuming (4.25). We can see that $p^-(\xi) < 0$ if $\xi < \sqrt{\Delta\xi}/\sigma$, and so we have in fact a *boundary layer* at $\xi = 0$ of width $\sqrt{\Delta\xi}/\sigma$. This is analogous to boundary layers in fluid flows where the flow can change radically over a small length scale, such as in the case of air close to the wing of an aircraft.

We can have a numerical scheme which is stable for all ξ in the interval $[0, \xi_{\max}]$ by simply approximating the derivative $\partial w(\xi, t)/\partial\xi$ in (4.11) by a forward difference

$$\frac{\partial w(\xi, t)}{\partial \xi} \ \sim \ \frac{w(\xi + \Delta\xi, t) - w(\xi, t)}{\Delta\xi}\ . \tag{4.26}$$

In that case the numerical scheme becomes (4.21) with $p(\xi), p^+(\xi), p^-(\xi)$ given by the formulas

(4.27)
$$p(\xi) = 1-\sigma^2\xi^2\Delta t/(\Delta\xi)^2-(1-r\xi)\Delta t/\Delta\xi, \quad p^+(\xi) = \sigma^2\xi^2\Delta t/2(\Delta\xi)^2+(1-r\xi)\Delta t/\Delta\xi\ ,$$
$$p^-(\xi) = \sigma^2\xi^2\Delta t/2(\Delta\xi)^2\ .$$

We still have $p(\xi) + p^+(\xi) + p^-(\xi) = 1$ but now all of $p(\xi), p^+(\xi), p^-(\xi)$ remain positive for $0 < \xi < \xi_{\max}$ provided we are away from the threshold in (4.25). The error in the scheme (4.21), (4.27) is now $O(\Delta t) + O(\Delta\xi) = O[(\Delta\xi)^2] + O(\Delta\xi) = O(\Delta\xi)$, so we have lost second order accuracy in order to maintain stability. As a compromise in order to try to retain second order accuracy, we can use the scheme (4.21), (4.22) for $\sqrt{\Delta\xi}/\sigma < \xi < \xi_{\max}$, and (4.21), (4.27) for $0 \le \xi \le \sqrt{\Delta\xi}/\sigma$. This scheme is also stable and preserves the second order accurate property outside the boundary layer. Note however that the value of the option today is $S_0 w(0,0)$ so the boundary layer really does have an effect on the option price.

In the discretely averaged Asian option the payoff depends on the average of the option over a discrete set of times $t_0, .., t_L$, where $0 \le t_0 < t_1 < \cdots t_L \le T$ so that

$$\text{payoff} \ = \ \left[S(T) - \frac{1}{L+1}\sum_{i=0}^{L} S(t_i) \right]^+ \ . \tag{4.28}$$

To price this option we again introduce a new variable I where

$$I(t) \ = \ \sum_{t_j \le t} S(t_j)\ , \quad 0 \le t \le T\ . \tag{4.29}$$

The value $V(S, I, t)$ of the option now has terminal condition

$$V(S, I, T) \ = \ \max[S - I/(L+1), 0] \quad \text{for } S, I > 0. \tag{4.30}$$

The function $V(S, I, t)$ is also a solution to the Black-Scholes PDE (2.1) provided t is not equal to one of the $t_j,\ 0 \le j \le L$. At the times $t_j,\ 0 \le j \le L$, there is a continuity condition

$$\lim_{t\to t_j^-} V(S, I, t) \ = \ \lim_{t\to t_j^+} V(S, I + S, t)\ . \tag{4.31}$$

The identity (4.31) is a consequence of the fact that if the stock price is S at time t_j then I increases to $I + S$ at time t_j, which we can see from (4.29). For t in the time interval $t_j < t < t_{j+1}$ then I remains constant and so the no arbitrage assumption implies that $V(S, I, t)$ satisfies the PDE (2.1). Note that in the discrete case $[S(t), I(t)]$ also satisfy a system of SDEs similar to (4.3),

$$dS(t) \ = \ S(t)[r dt + \sigma dB(t)], \qquad dI(t) \ = \ b(S(t), t)\ dt\ , \tag{4.32}$$

where the function $b(S, t)$ is defined by

$$b(S, t) \ = \ S\sum_{j=0}^{L} \delta(t - t_j)\ , \quad \delta(\cdot) \ = \ \text{Dirac delta function.} \tag{4.33}$$

We now of course still have the problem of specifying the boundary conditions for $V(S, I, t)$ other than the obvious one (4.7), and it is not at all clear how to do this. In the next chapter we shall show how to find the value of the discrete Asian option using the Monte-Carlo method, which obviates the need to find boundary conditions.

5. Numerical Linear Algebra

To implement implicit methods for solving PDE such as the trapezoid rule one needs to be able to efficiently solve large *sparse* systems of linear equations. In this section we shall give a brief survey of how to go about this. The basic problem we are interested in is solving the linear system

(5.1) $Au = b$, where u, b are n dimensional vectors and A is a $n \times n$ matrix.

Thus the matrix A and vector b are known, and we wish to find the solution u to (5.1). The standard way to solve this problem is by Gauss elimination, which requires $O(n^3)$ computations for large n. In the cases we will be interested in, where A is derived from the discretization of a PDE, the matrix A is sparse. That means it has $O(n)$ non-zero entries. Evidently a diagonal matrix has at most n non-zero entries and a tri-diagonal matrix has at most $3n$ non-zero entries. Matrices such as these are therefore sparse. If we consider implementing the Gauss elimination method for solving (5.1) when A is sparse, we observe that most of the computations involved are trivial since we are simply adding zeros. Hence these computations- which we could have predicted the result of before any actual computing- yield no new *information*. A basic principle in determining whether a given numerical method is *efficient* is that each new computation gives more information about the quantity which we wish to estimate.

The starting point for efficient methods of solving (5.1) when A is sparse is to use *iterative techniques*. Thus we rewrite (5.1) as a *fixed point* equation

(5.2) $u = Bu + c$ where the matrix B and vector c are determined by A, b.

The matrix B is assumed to be sparse if A is sparse. We make an initial guess u^0 for the solution to (5.2) and then define a recurrence $u^k,\ k = 1, 2, ...,$ by $u^{k+1} = Bu^k + c$. If the sequence $u^k,\ k = 1, 2, ...,$ converges $\lim_{k\to\infty} u^k = u^\infty$, then $u = u^\infty$ is the solution to (5.1). Note that the evaluation of each u^k requires just $O(n)$ computations since B is sparse. There are two aspects to the implementation of this method:

(a) **Convergence:** We need to show that the matrix B has properties which imply that sequences $u^k,\ k = 1, 2, ...,$ converge.
(b) **Rate of Convergence:** We need to know how large k needs to be so that the error $u^\infty - u^k$ is small.

The number of computations required for the evaluation of u^k is $\simeq kn$, so (b) is very important in understanding how efficient the method really is. Obviously if we need to take $k = n^2$ to get a good approximation to the solution of (5.2), we might as well have used the straightforward Gauss elimination method instead.

We give three examples of matrices B for which (5.1), (5.2) are equivalent. The first example is known as the *Jacobi method*. We write $A = D - E - F$, where D is a diagonal matrix consisting of just the diagonal entries of the matrix A. The matrix $-E$ consists of the *lower triangular matrix* with entries equal to the entries of A *below* the diagonal. The matrix $-F$ consists of the *upper triangular matrix* with entries equal to the entries of A *above* the diagonal. Hence (5.1) is equivalent to the equation $Du = (E + F)u + b$. Assuming D is invertible we see that (5.2) holds with $B = D^{-1}(E + F)$ and $c = D^{-1}b$, whence the matrix B is the Jacobi iteration matrix. The corresponding iteration sequence $u^k,\ k = 1, 2, ...,$ therefore

satisfies the recurrence

$$Du^{k+1} \;=\; Eu^k + Fu^k + b, \quad k = 1, 2, ..., \tag{5.3}$$

In the Gauss-Seidel (GS) method we partially update the RHS of (5.3) by making the recurrence equation

$$Du^{k+1} \;=\; Eu^{k+1} + Fu^k + b, \quad k = 1, 2, ... \tag{5.4}$$

Intuitively the iterates of (5.4) should converge faster than the iterates of (5.3) because of the updating. Actually in practice there is not much difference between the rates of convergence of (5.3) and (5.4). The iteration matrix B for (5.4) is evidently $B = (D-E)^{-1}F$ and $c = (D-E)^{-1}b$. Since $D-E$ is a lower triangular sparse matrix the evaluation of u^{k+1} in (5.4) from u^k requires $O(n)$ computations. The final method is the so called SOR (successive over-relaxation) method. In this method we modify the GS method by introducing a parameter ω. Observe that (5.1) is equivalent to the equation

$$(D - \omega E)u \;= [(1-\omega)D + \omega F]u + \omega b\ , \tag{5.5}$$

where $\omega = 1$ corresponds to the fixed point equation for the GS method. The SOR recurrence equation is then given by

$$(D - \omega E)u^{k+1} \;= [(1-\omega)D + \omega F]u^k + \omega b, \quad k = 1, 2, ... \tag{5.6}$$

Again since $D - \omega E$ is lower triangular and sparse, the evaluation of u^{k+1} in (5.6) from u^k requires $O(n)$ computations. The term "over-relaxation" refers to the fact that one takes ω in (5.6) to satisfy $1 < \omega < 2$. It turns out that for some problems associated with solving differential equations, by choosing $\omega < 2$ but close to 2 (depending on the discretization of the differential equation) then one gets significant acceleration in the convergence of the iterates for (5.6) over (5.4) or (5.3).

We consider the boundary value problem

$$-\alpha \frac{d^2u(x)}{dx^2} + u(x) \;=\; f(x),\ 0 < x < 1, \quad u(0) = a,\ u(1) = b. \tag{5.7}$$

where $f : [0,1] \to \mathbf{R}$ is a given function. If $\alpha > 0$ then the problem (5.7) has a unique solution. We can find the solution numerically by introducing a space discretization Δx satisfying $N\Delta x = 1$ for some integer N, and writing $u^n \sim u(n\Delta x), n = 0, .., N$. Evidently $u_0 = a,\ u_N = b$, and then $u_1, .., u_{N-1}$ are determined by a linear system of $N-1$ equations obtained from the differential equation (5.7). Approximating the second derivative by the difference (2.18) we have from (5.7) that

$$\alpha \frac{2u_n - u_{n-1} - u_{n+1}}{(\Delta x)^2} + u_n \;=\; f(n\Delta x) = f_n, \quad \text{for } n = 1, .., N. \tag{5.8}$$

We have already observed the difference approximation is a second order accurate approximation to the second derivative, whence we conclude that the solution $u_n,\ n = 1, .., N-1$ of the linear $N-1$ dimensional system (5.8) satisfies $u(n\Delta x) = u_n + O[(\Delta x)^2]$.

The Jacobi iteration (5.3) for (5.8) is given by the formula

$$u_n^{k+1} \;=\; \frac{1}{2\alpha + (\Delta x)^2}\{\alpha(u_{n-1}^k + u_{n+1}^k) + (\Delta x)^2 f_n\ \} \quad n = 1, .., N-1. \tag{5.9}$$

The GS iteration is given by

$$u_n^{k+1} \;=\; \frac{1}{2\alpha+(\Delta x)^2}\{\alpha(u_{n-1}^{k+1}+u_{n+1}^k)+(\Delta x)^2 f_n\ \}\quad n=1,..,N-1. \tag{5.10}$$

Finally the SOR iteration can be given by a two step formula

$$\begin{aligned} y_n^{k+1} \;&=\; \frac{1}{2\alpha+(\Delta x)^2}\{\alpha(u_{n-1}^{k+1}+u_{n+1}^k)+(\Delta x)^2 f_n\ \}\ , \\ u_n^{k+1} \;&=\; u_n^k+\omega[y_n^{k+1}-u_n^k]\quad \text{for } n=1,..,N-1. \end{aligned} \tag{5.11}$$

Note that in the first step of (5.11) where we compute y_n^{k+1}, we are using the fact that we have already computed u_{n-1}^{k+1}.

We compare the implementation of the Jacobi iteration versus the GS iteration. For the Jacobi iteration we need two $N+1$ dimensional vectors $u, \tilde{u}$ where we set $u_0=\tilde{u}_0=a,\ u_N=\tilde{u}_N=b$. We set u initially by a suitable "guess". Then to get the $2m$th Jacobi iteration we write

$$\begin{array}{lcl} \text{for } j & = & 1:m \\ \text{for } n & = & 1:N-1 \\ \tilde{u}_n & = & \frac{1}{2\alpha+(\Delta x)^2}\{\alpha(u_{n-1}+u_{n+1})+(\Delta x)^2 f_n\ \} \\ \text{end} & & \\ \text{for } n & = & 1:N-1 \\ u_n & = & \frac{1}{2\alpha+(\Delta x)^2}\{\alpha(\tilde{u}_{n-1}+\tilde{u}_{n+1})+(\Delta x)^2 f_n\ \} \\ \text{end} & & \\ \text{end} & & \end{array} \tag{5.12}$$

The $2m$th iteration is given by the vector u. The GS algorithm is in fact simpler since we only need *one* $N+1$ dimensional vector u in which we set $u_0=a,\ u_N=b$. Again we set u initially by a suitable guess and obtain the mth GS iteration by:

$$\begin{array}{lcl} \text{for } j & = & 1:m \\ \text{for } n & = & 1:N-1 \\ u_n & = & \frac{1}{2\alpha+(\Delta x)^2}\{\alpha(u_{n-1}+u_{n+1})+(\Delta x)^2 f_n\ \} \\ \text{end} & & \\ \text{end} & & \end{array} \tag{5.13}$$

Note that if we are computing the $j=(k+1)$st iteration in (5.13) then $u_n=u_n^{k+1}$ on the LHS of the equation and on the RHS $u_{n+1}=u_{n+1}^k$. However because the algorithm continually overwrites the vector u we have that $u_{n-1}=u_{n-1}^{k+1}$ on the RHS of (5.13) since we have already computed u_{n-1}^{k+1} in the loop. Hence (5.13) yields the GS iteration (5.10).

Finally we wish to find properties of the matrix B in (5.2) which guarantees convergence of the sequence $u^k,\ k=1,2,..$, to the solution of (5.2). Let $u=u^\infty$ be the solution to (5.2) and $\varepsilon^k=u^k-u^\infty$ be the error of the kth iteration. Since $u^\infty=Bu^\infty+c$ and $u^{k+1}=Bu^k+c$ it follows that

$$\varepsilon^{k+1} \;=\; B\varepsilon^k,\quad \text{which implies } \varepsilon^k \;=\; B^k\varepsilon^0\ . \tag{5.14}$$

We measure the size of ε^k by introducing a *norm* $\|\cdot\|$ on the space $\mathbf{R}^n$ of n dimensional vectors. A norm must have three properties:

(a) Positivity: $\|v\| \geq 0$ for all $v \in \mathbf{R}^n$ and $\|v\| = 0$ only if $v = 0$.
(b) Scalar multiplication: $\|\lambda v\| = |\lambda|\|v\|$ for all $v \in \mathbf{R}^n$ and $\lambda \in \mathbf{R}$.
(c) Triangle inequality: $\|v + w\| \leq \|v\| + \|w\|$ for all $v, w \in \mathbf{R}^n$.

The most familiar norm is the *Euclidean distance* norm $\|\cdot\|_2$ defined by

$$\|v\|_2 \;=\; \left\{ \sum_{j=1}^{n} v_j^2 \right\}^{1/2} \qquad v = [v_1, .., v_n] \in \mathbf{R}^n \;. \tag{5.15}$$

The Euclidean distance is sometimes called the $2-$norm to distinguish it from the more general $p-$norm where $1 \leq p \leq \infty$. The norm we shall be interested in is the $\infty-$norm defined by

$$\|v\|_\infty \;=\; \max_{1 \leq j \leq n} |v_j| \;, \quad v = [v_1, .., v_n] \in \mathbf{R}^n \;. \tag{5.16}$$

Observe that the set of $n \times n$ matrices is an n^2 dimensional linear space so can be identified with $\mathbf{R}^{n^2}$, and hence we can define norms of matrices as well as norms of vectors. A norm $\|\cdot\|$ on $\mathbf{R}^n$ induces a special norm on the linear space of $n \times n$ matrices as follows:

$$\|B\| \;=\; \max_{\|v\|=1} \|Bv\| \;. \tag{5.17}$$

It follows from the definition (5.17) that $\|Bv\| \leq \|B\|\|v\|$ for all $v \in \mathbf{R}^n$. Hence if B_1, B_2 are two $n \times n$ matrices then

$$\|B_1 B_2 v\| \;\leq\; \|B_1\|\|B_2 v\| \;\leq\; \|B_1\|\|B_2\|\|v\| \quad \text{for } v \in \mathbf{R}^n. \tag{5.18}$$

In particular we have that $\|B^k\| \leq \|B\|^k,\ k = 1, 2, ..$, whence it follows that if $\|B\| < 1$ then $\lim_{k\to\infty} \|B^k\| = 0$. Comparing with (5.14), we have therefore shown that if $\|B\| < 1$ for some induced matrix norm then the sequence $u^k,\ k = 1, 2, ..$, converges to the solution of (5.2).

A convenient way of measuring the rate of convergence is to ask how many iterations k_0 are required to guarantee that the initial error is reduced by 50%. The reason for this is that once we know the number k_0 then the error decreases exponentially in m upon doing mk_0 iterations. From the previous paragraph we see that it is possible to to take k_0 such that $\|B\|^{k_0} = 1/2$. Consider now the Jacobi iteration matrix B associated with the boundary value problem (5.7), (5.8). From (5.9) we see that B is a tri-diagonal matrix with non-negative entries and the sum of the entries in a row is $2\alpha/[2\alpha + (\Delta x)^2] = 1 - (\Delta x)^2/[2\alpha + (\Delta x)^2 = 1 - O[(\Delta x)^2]$. Letting $\|B\|_\infty$ be the matrix norm on $n \times n$ matrices $B = \{b_{i,j}\}$ induced from the ∞ norm on $\mathbf{R}^n$, it is clear that

$$\|B\|_\infty \;=\; \max_{1 \leq i \leq n} \sum_{j=1}^{n} |b_{i,j}| \;. \tag{5.19}$$

We conclude that the matrix B in (5.9) satisfies $0 < 1 - \|B\|_\infty = O[(\Delta x)^2]$. Thus $\|B\|_\infty^{k_0} = 1/2$ implies that $\{1 - O[(\Delta x)^2]\}^{k_0} = 1/2$, whence $k_0 \simeq 1/(\Delta x)^2$. The number of iterations necessary to reduce error by 50% therefore grows as the discretization size decreases. Furthermore since each iteration requires $\simeq 1/(\Delta x)$ computations, the total number of computations required to reduce initial error by 50% is $\simeq 1/(\Delta x)^3$. Since this number is $O(n^3)$ with $n = N - 1 \simeq 1/(\Delta x)$, it appears that our iteration method is rather worse than the simple Gauss elimination

method. It is true that the pure Jacobi or GS method does not do better than Gauss elimination, but SOR will do better if we choose ω in (5.11) close to 2, depending on Δx. There exist now quite sophisticated iteration algorithms (*multi-grid algorithms*) based on the simple ones we have been considering, which reduce error by 50% in $\simeq \log(1/\Delta x)$ iterations. These algorithms are closely linked to *fast Fourier transform* algorithms which enable one to compute the discrete Fourier transform of an n dimensional vector in $\simeq n \log n$ computations.

6. The Crank-Nicolson Algorithm

We already introduced the trapezoid rule (1.31) in §1, which is an implicit second order accurate method for solving ODEs and mentioned that its extension to solving linear parabolic PDE is known as the Crank-Nicolson (CN) method. We shall use the CN method here to obtain an algorithm for pricing a *barrier* call option. In the barrier option the value of the option becomes zero if the price of the underlying stock falls below the barrier price before expiration. Assuming e^a is the barrier stock price then the value of the barrier option is given by solving (2.1), (2.2) in the region $S > e^a,\ t < T$, with terminal condition (2.2) and boundary condition $V(e^a, t) = 0,\ t < T$. Defining $u(x,t)$ as in §2 we see that we need to solve the PDE (2.4) with the terminal condition (2.5) and boundary conditions $u(a,t) = 0,\ t < T$, and (2.16). From (1.31),(2.17),(2.18),(2.21) we see that the CN algorithm for (2.4) is given by the equation

$$\begin{aligned}(6.1)\quad &\frac{u(x,t)-u(x,t-\Delta t)}{\Delta t}+\frac{\sigma^2}{4}\frac{u(x+\Delta x,t)+u(x-\Delta x,t)-2u(x,t)}{(\Delta x)^2}+\\ &\frac{\sigma^2}{4}\frac{u(x+\Delta x,t-\Delta t)+u(x-\Delta x,t-\Delta t)-2u(x,t-\Delta t)}{(\Delta x)^2}+\\ &\frac{1}{2}\left\{r-\frac{\sigma^2}{4}\right\}\left[\frac{u(x+\Delta x,t)-u(x-\Delta x,t)}{2\Delta x}+\frac{u(x+\Delta x,t-\Delta t)-u(x-\Delta x,t-\Delta t)}{2\Delta x}\right]\\ &-\frac{r}{2}[u(x,t)+u(x,t-\Delta t)] \;=\; 0.\end{aligned}$$

We write (6.1) as

$$(6.2)\quad (p+r\Delta t/2)u(x,t-\Delta t)-p^+u(x+\Delta x,t-\Delta t)-p^-u(x-\Delta x,t-\Delta t) \;=\; f(x,t)\,,$$

where p, p^+, p^- are given by the formulas

$$(6.3)\quad p = 1+\sigma^2\Delta t/2(\Delta x)^2,\quad p^+ = \sigma^2\Delta t/4(\Delta x)^2+(r-\sigma^2/2)\Delta t/4\Delta x\,,$$
$$p^- = \sigma^2\Delta t/4(\Delta x)^2-(r-\sigma^2/2)\Delta t/4\Delta x\,.$$

The RHS $f(x,t),\ a < x < b$, of (6.2) is given in terms of the already computed function $u(x,t),\ a < x < b$. We shall solve (6.2) by using the GS algorithm with initial guess $u(x,t),\ a < x < b$, since we expect that this function is a good approximation to the function $u(x, t-\Delta t),\ a < x < b$. Observe that the algorithm converges since $p^+ + p^- < p$.

Assuming we are able to solve the linear system of equations (6.2) exactly, then the error in the CN method is $O[(\Delta t)^2] + O[(\Delta x)^2]$. Thus in the implementation of CN we take $\Delta t \simeq \Delta x$. In the one dimensional case we are considering here we can actually solve (6.2) by using LU factorization in $\simeq 1/\Delta x$ computations. Thus the total number of computations required to get a solution with error $O[(\Delta x)^2]$

is $\simeq 1/(\Delta x)^2$ since there are $\simeq 1/\Delta t$ time steps and $\Delta t \simeq \Delta x$. In the explicit Euler method since $\Delta t \simeq (\Delta x)^2$ the total number of computations required to get a solution with error $O[(\Delta x)^2]$ is $\simeq 1/(\Delta x)^3$. Hence for a given accuracy the CN method is significantly more efficient than explicit Euler provided we have an efficient linear equation solver. We have already pointed out in the previous section that the simple iteration methods we introduced there are not efficient. We can estimate as we did in §5 how many computations using the Jacobi iteration are required to yield a solution which is $O[(\Delta x)^2]$ accurate. From (6.3) we see that $0 < 1 - (p^+ + p^-)/p \simeq (\Delta x)^2/\Delta t \simeq \Delta x$. Thus we need to do $\simeq 1/\Delta x$ iterations to reduce the initial error in the solution to the linear system (6.2) by 50%. Thus the number of computations at each time step is $\simeq 1/(\Delta x)^2$. Since there are $\simeq 1/\Delta t \simeq 1/\Delta x$ time steps we conclude the total number of computations to obtain an $O[(\Delta x)^2]$ accurate solution is $\simeq 1/(\Delta x)^3$, which is the same as in the explicit Euler method. This result is perhaps not so surprising given the remarks at the end of §5.

University of Michigan, Department of Mathematics, Ann Arbor, MI 48109-1109
E-mail address: conlon@umich.edu