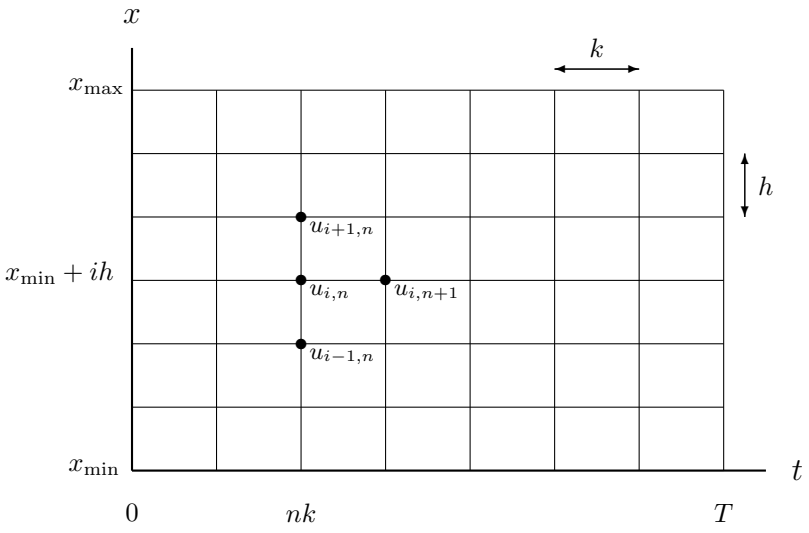
Figure 1: pde solution grid



3. Numerically Solving PDE's: Crank-Nicholson Algorithm

This note provides a brief introduction to finite difference methods for solving partial differential equations. We focus on the case of a pde in one state variable plus time. Suppose one wishes to find the function $u(x,t)$ satisfying the pde

$$au_{xx} + bu_x + cu - u_t = 0 \tag{12}$$

subject to the initial condition $u(x,0) = f(x)$ and other possible boundary conditions. This initial condition will correspond to a maturity or expiry date value condition in our applications and t will denote time left to maturity. Thus time will run backwards down to 0, explaining the negative u_t term in (1). The coefficients a, b, c may themselves be functions of x, t. Any parameters are presumed given, so that one can calculate a numerical value for these functions for any given x, t.

Obviously one cannot 'calculate' the entire function u. What we shall consider a solution is the numerical values that u takes on a grid of x, t values placed over some domain of interest. Once we have these u values, since u is assumed to be smooth almost everywhere, we can interpolate within this grid to get values for arbitrary x, t.

To this end, suppose the domain we will work on is rectangular with x ranging from $x_{\min}$ to $x_{\max}$ and t ranging from 0 to T. Divide $[0,T]$ into N equally spaced intervals at t values indexed by $n = 0, 1, \ldots, N$, and $[x_{\min}, x_{\max}]$ into I intervals at x values indexed by $i = 0, 1, \ldots, I$. The length of these intervals is k in the time direction and h in the x direction. We seek an approximation to the true values of u at the $(N+1) \times (I+1)$ gridpoints. Let $u_{i,n}$ denote our approximation at the gridpoint where $x = x_{\min} + ih$, $t = nk$.

The next step — and this is what makes this procedure a *finite difference* method — is to approximate the partial derivatives of u at each gridpoint by difference expressions in the as yet unknown $u_{i,n}$'s. We can calculate $u_{i,0}$ for each i directly from the initial value condition f. Thus it is natural to start from this boundary are work outward, calculating the $u_{i,n+1}$'s from $u_{i,n}$. Focussing on an arbitrary internal gridpoint in, one could approximate the partial derivatives at that point by the following:

$$\begin{aligned} u &= u_{i,n} \\ \partial u/\partial t &= \frac{u_{i,n+1} - u_{i,n}}{k} \\ \partial u/\partial x &= \frac{u_{i+1,n} - u_{i-1,n}}{2h} \\ \partial^2 u/\partial x^2 &= \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{h^2} \end{aligned} \tag{13}$$

The differences in the x, or state, direction have been centered around the point in to give 'second order' accuracy to the approximation. These expressions could then be substituted into the pde (1). Solving the resulting equation for $u_{i,n+1}$ gives the explicit solution

$$u_{i,n+1} = (\frac{k}{h^2}a + \frac{k}{2h}b)u_{i+1,n} + (1 + kc - \frac{2k}{h^2}a)u_{i,n} + (\frac{k}{h^2}a - \frac{k}{2h}b)u_{i-1,n} \tag{14}$$

One could then proceed to calculate all the $u_{i,n+1}$'s from the $u_{i,n}$'s and recursively obtain u for the entire grid. Since equation (3) applies only to the interior gridpoints, we at each time step would have to make use of some other boundary conditions (e.g., at $x_{\min}$ and $x_{\max}$) to identify all the u's. More will be said about this later.

The result of this is called an *explicit* finite difference solution for u. It is second order accurate in the x direction, though only first order accurate in the t direction, and easy to implement. Unfortunately the numerical solution is *unstable* unless the ratio k/h^2 is sufficiently small. By unstable is meant that small errors due either to arithmetic inaccuracies or to the approximate

nature of the derivative expressions will tend to accumulate and grow as one proceeds rather than dampen out. Loosely speaking, this will occur when the difference equation system described by (3) has eigenvalues greater than one in absolute value. See a numerical analysis book such as Vemuri and Karplus (1981) or Lapidus and Pinder (1982) for discussion of stability issues.

The instability problem can be handled by instead using and *implicit* finite difference scheme. The recommended method for most problems in the Crank-Nicholson algorithm, which has the virtues of being unconditionally stable (i.e., for all k/h^2) and also is second order accurate in both the x and t directions (i.e., one can get a given level of accuracy with a coarser grid in the time direction, and hence less computation cost). This is the algorithm implemented by the routines `CNSET` and `CNSTEP` handed out in class.

The algorithm entails proceeding as before, but using difference expressions for the partial derivatives which are centered around $t + k/2$ rather than around t. Thus the expressions for u, u_x and u_{xx} are averages of what we had in (3) for times n and $n + 1$:

$$\begin{aligned} u &= \frac{u_{i,n} + u_{i,n+1}}{2} \\ \partial u/\partial t &= \frac{u_{i,n+1} - u_{i,n}}{k} \\ \partial u/\partial x &= \frac{u_{i+1,n} - u_{i-1,n} + u_{i+1,n+1} - u_{i-1,n+1}}{4h} \\ \partial^2 u/\partial x^2 &= \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n} + u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1}}{2h^2} \end{aligned} \tag{15}$$

Substituting the above into the pde, multiplying through by $4h^2k$ to eliminate the denominators, and collecting all the terms involving the unknown $u_{\cdot,n+1}$'s on the left hand side results in

$$\begin{aligned} &\overbrace{-(2ka + khb)}^{A_i} u_{i+1,n+1} + \overbrace{(4h^2 + 4ka - 2h^2kc)}^{B_i} u_{i,n+1} \overbrace{-(2ka - khb)}^{C_i} u_{i-1,n+1} \\ &= \\ &\qquad \underbrace{(2ka + khb)u_{i+1,n} + (4h^2 - 4ka + 2h^2kc)u_{i,n} + (2ka - khb)u_{i-1,n}}_{D_i} \end{aligned} \tag{16}$$

for each $i = 1, \ldots, I - 1$. It is apparent that that the $u_{\cdot,n+1}$'s cannot individually be written as simple linear combinations of the $u_{\cdot,n+1}$'s, but are simultaneously determined as the solution to this system of linear equations. However this system has a very convenient structure. Written in

matrix form, (5) provides the interior equations of

$$\begin{bmatrix} B_0 & C_0 & 0 & 0 & \cdots & 0 \\ A_1 & B_1 & C_1 & 0 & \cdots & 0 \\ 0 & A_2 & B_2 & C_2 & 0 & \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & & 0 & A_I & B_I \end{bmatrix} \begin{pmatrix} u_{0,n+1} \\ u_{1,n+1} \\ u_{2,n+1} \\ \vdots \\ u_{I,n+1} \end{pmatrix} = \begin{pmatrix} D_0 \\ D_1 \\ D_2 \\ \vdots \\ D_I \end{pmatrix} \tag{17}$$

This sort of system is most efficiently solved by Gaussian elimination. It requires just $8I$ floating point operations to determine the unknown $u_{\cdot,n+1}$'s (e.g., see Press et al., 1986, p.40 or the listing of routine TRIDAG at the end of CNSTEP).

Boundary conditions

We almost have a procedure for recursively determining the entire grid of $u_{i,n}$'s starting from the given initial values. Substitution of the the difference expressions into the differential equation only gave us a linear equation for each interior point in the grid. That gives $I - 1$ equations at each time step, which is not sufficient to determine the $I + 1$ unknowns. The missing two equations must be provided by boundary conditions applied at each time step. It would be desirable for these to be representable in a form that preserves the tridiagonal form of the system and thus the efficiency of solution. Three possibilities are considered.

The most convenient to apply would be knowledge of the value of u on the boundaries $x_{\max}$ and $x_{\min}$. In (6), this could be embodied by setting the coefficients $B_0 = 1$, $C_0 = 0$, $A_I = 0$ and $B_I = 1$, then setting D_0 and D_I on the right hand side of (6) equal to the known values $u(x_{\min}, t)$ and $u(x_{\max}, t)$ respectively. This could be appropriate, for example, if u was the value of a call option on a stock whose current price was x, and the assumed stochastic process for x had $0 = x_{\min}$ as an absorbing state (if x ever hit 0, it never again moves away from it). The call option would then be worth nothing — i.e., $u(x_{\min}, t) = 0$.

Alternatively, from the nature of the security being valued we may have knowledge of the derivative u_x at one or the other boundary. This could be expressed in the form of (6) by setting $B_0 = -1$, $C_0 = 1$ and $D_0 = hu_x(x_{\min}, t)$. The upper boundary slope is handled analogously. This could be appropriate, for example, if a boundary was in the region where an option would rationally be exercised immediately, and if the derivative of the exercise value of the option with respect to x was known. The difference

approximation we are using the u_x is actually centered around a point $h/2$ inside the boundary. If u was known to be linear in the neighbourhood of the boundary this should be adequate. If not, then a more elaborate approximation to the slope at the boundary (e.g., that of a quadratic through the boundary gridpoint and the two interior ones) should be used. This would initially upset the tridiagonal form of (6) by introducing a non-zero term just to the right of C_0. However a manual elimination of this term by subtracting a appropriate multiple of the second equation from the first would restore the system to the desired form.

In each of the above cases, the boundary at which you know the value or slope of u might be at $x = \infty$. In that case you might consider transforming the domain for u to some finite interval. For example, if x ranges from 0 to ∞, then $y = x/(1 + x)$ ranges from 0 to 1. Transforming the pde into one in $v(y, t) \equiv u(x(y), t)$ and solving for v would let you use the known boundary information at $y = 1$. See Brennan and Schwartz (1979) for an example. One disadvantage of this approach is that a fairly large grid may be needed to get a desired level of fineness in the levls of x of interest. In many situations you will do better with the next approach.

The third possibility is that we have no simple information about u at the boundaries we have chosen, yet we feel that the behaviour at these boundaries should have little impact on the behaviour of u in that part of the grid in which we are interested. This could be the case if the probability of reaching the boundary from relevant starting levels of the state in time $t_{\max} - t_{\min}$ is small. This could occur if, for example, the state variable followed a mean-reverting process. A procedure that I have found often works in this case is to determine the boundary values of u by simple extrapolation of the immediately interior values. Focussing on the lower boundary, quadratic extrapolation of u_3, u_2, u_1 to u_0 implies

$$(x_3 - x_2) - (x_2 - x_1) = (x_2 - x_1) - (x_1 - x_0)$$

or, equivalently,

$$x_0 - 3x_1 + 3x_2 - x_3 = 0$$

Using this relation as the first equation in system (6) gives us

$$\begin{bmatrix} 1 & -3 & 3 & -1 & \cdots & 0 \\ A_1 & B_1 & C_1 & 0 & \cdots & 0 \\ 0 & A_2 & B_2 & C_2 & 0 & \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & & 0 & A_I & B_I \end{bmatrix} \begin{pmatrix} u_{0,n+1} \\ u_{1,n+1} \\ u_{2,n+1} \\ \vdots \\ u_{I,n+1} \end{pmatrix} = \begin{pmatrix} 0 \\ D_1 \\ D_2 \\ \vdots \\ D_I \end{pmatrix} \tag{18}$$

which is no longer tridiagonal. However the offending elements of the first row of our matrix can be eliminated as follows: Define

$$G \equiv \frac{C_1}{B_2 + 3C_2}$$

Multiply the first row by GC_2. Add G times the third row to the first, then subtract the second row from the first. A little algebra verifies that the third and fourth elements of the first row are now 0. The first two elements and corresponding right hand side element are now

$$\begin{aligned} B_0 &= GC_2 - A_1 \\ C_0 &= G(A_2 - 3C_2) - B_1 \\ D_0 &= GD_2 - D_1 \end{aligned} \tag{19}$$

Extrapolation at the $x_{\max}$ boundary could be similarly handled. Treating the boundaries in this fashion may not be stricly correct for a given problem. We are in effect imposing a boundary condition $u_{xxx} = 0$. This is consistent with u being truly linear or constant in the neighbourhood of the boundary. If used, one should check to see how much damage it may be causing by trying out a few differing levels of $x_{\min}$ and $x_{\max}$ to make sure the solution in the domain of interest is not sensitive to it.

In the routine CNSET this last procedure is the default, implemented if the flags ISMN or ISMX are set to 0. If either is set to 1 then it is presumed that a known boundary value is being provided via functions FMIN(T) or FMAX(T) respectively.

Interpretation

Viewed as a numerical procedure, what we have done is convert a partial differential equation that holds everywhere in some domain into a system of simultaneous linear equations to get an approximate solution. However what is going on can be given economic interpretation. This is most readily apparent in the explicit approximation.

Separating out the term involving c in equation (3) lets us write $u_{i,n+1}$ as

$$\begin{aligned} u_{i,n+1} = & \\ & ((\frac{k}{h^2}a + \frac{k}{2h}b)u_{i+1,n} + (1 - \frac{2k}{h^2}a)u_{i,n} + (\frac{k}{h^2}a - \frac{k}{2h}b)u_{i-1,n}) + kcu_{i,n} \end{aligned} \tag{20}$$

The coefficient c will usually equal $-r$ in valuing securities. The final term is thus $-rku_{i,n}$, the interest lost by holding u for an amount of time k. The

remaining coefficients on the right hand side add up to one. They can be viewed as the probabilites of the state x moving up by h, staying unchanged, and moving down by h respectively. Indeed, if these were the only moves possible over the interval $n+1$ to n, one may verify that the expected change in x is kb and its variance is $2ka$. The distribution of x one period ahead thus agrees with that of the risk adjusted process for x in terms of mean and variance.

Equation (9) calculates $u_{i,n+1}$ as the probability weighted average of the possible values u may take on next period, with allowance for the opportunity cost of holding the security. This is just the expected discounted value of the security at the next time point. The values at the time n had been calculated in turn as the discounted expectation of their possible values at time $n-1$, and so on down to the asset's maturity. This permits a more intuitive interpretation of the valuation pde that we started out to solve: It states that the equilibrium of an asset equals the expectation of the discounted present value of its cash flows to maturity, where the expectation is with respect to the risk adjusted process for the state variables. Cox, Ingersoll and Ross (1985) provide a formal demonstration of this interpretation.

One can also see the limits of the explicit difference. If k is too large relative to h, some of the alleged probabilities will be negative. This reflects the maximum volatility of the state variable that can be represented by simply moving up or down by h over an interval k. Negative probabilities in this interpretation correspond to instability of the difference scheme.

We can also interpret the implicit Crank-Nicholson scheme as a jump process approximation to the risk adjusted x process. The important difference is that that approximation permits jumps to *any* point in the x grid over the interval k, and the implicit probabilities are all positive. Thus an arbitrarily large volatility can always be represented and large k may be used. We leave it as an exercise to work out the details. Or you may consult Brennan and Schwartz (1978).