Finance 513 - Spring 2019 *Professor Johnson*
UIUC - Gies College of Business *tcj@illinois.edu*

Finite difference methods in one dimension

- This note describes a solution technique for the no-arbitrage PDE in one dimension that is based on the idea of working backwards on a discrete grid of values and approximating partial derivatives via discrete differences on the grid.
- Suppose we have an asset, C, whose payoff we know at time-T, and which we know obeys:

$$-C_t = \frac{1}{2}\sigma^2 S^2 C_{SS} + rSC_S - rC.$$

- If we approximate dt as Δt, then we can view $C_t = \frac{\partial C}{\partial t}$ as $[C(t) - C(t - \Delta t)]/\Delta t$ and view the PDE as specifying a recipe for moving backwards in time.
  - Conceptually, if we know the solution at T we could evaluate all the partial derivatives using that solution.
  - Then we could just iterate back:

$$C(t - \Delta t) = C(t) + \Delta t\,(\text{all the other terms evaluated at t})\,.$$

  - Then, just like in the binomial model, we'd repeat the process again for time $t - 2\Delta t$, and continue back to today.
- Finite difference methods are ways of making this practical by choosing a discrete grid of N values of S, and then, at any point in the grid, say S_i, approximating, e.g.,

$$\frac{\partial^2 C}{\partial S^2} \approx \frac{C(S_{i+1}) - 2C(S_i) + C(S_{i-1})}{(\Delta S)^2}$$

  where $\Delta S = S_{i+1} - S_i$ or perhaps $\frac{1}{2}(S_{i+1} - S_{i-1})$, etc.
- While this is all fine conceptually (as long as the grid spacing is small enough), it turns out this simple procedure is not very good numerically.
  - It doesn't keep the solutions smooth enough.
  - And little errors will cause it to blow up.
- To ensure that the solutions are stable, one usually has to evaluate some of the approximate partial derivatives using the grid values at the earlier time step.
  - These are called *implicit* methods, because you are solving for those earlier time step values at the same time that you are using them implicitly to approximate the derivatives.
- To illustrate, suppose we are at time t_{n-1}.
  - Then we can write a *linear system* of equations that solves for C for all the S_i values at once, as follows.

- Approximate the PDE now as

$$C(t-\Delta t) - \Delta t\,(\text{all the other terms evaluated at } t-\Delta t) = C(t).$$

  Specifically, at the ith stock grid point at the nth time step,

$$C_i - \Delta t\left(\frac{1}{2}\sigma^2 S_i^2 \frac{C_{i+1}-2C_i+C_{i-1}}{(\Delta S)^2} + rS_i\frac{C_{i+1}-C_{i-1}}{2\Delta S} - rC_i\right)$$
$$= C_i(t_{n+1}).$$

  where all the Cs on the left are the values at t_n that we have to solve for.

- This is a linear equation involving C_{i_1}, C_i, and $C_{i,j}$.
  * And we have one such equation for each i.
  * So if we stack all N equations, we have a complete system $W = K\ C$ that looks like:

$$\begin{bmatrix} \vdots \\ \vdots \\ W_i \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} & \ddots & & & & & \\ 0 & k_{i+1}^+ & k_{i+1}^0 & k_{i+1}^- & 0 & \cdots & \\ \cdots & 0 & k_i^+ & k_i^0 & k_i^- & 0 & \cdots \\ & \cdots & 0 & k_{i-1}^+ & k_{i-1}^0 & k_{i-1}^- & 0 \\ & & & & & \ddots & \end{bmatrix} \cdot \begin{bmatrix} \vdots \\ C_{i+1} \\ C_i \\ C_{i-1} \\ \vdots \end{bmatrix}$$

    where $W_i = C_i(t_{n+1})$ and
    $k_i^+ = -\sigma^2 S_i^2 \frac{\Delta t}{2\Delta S^2} - rS_i\frac{\Delta t}{2\Delta S}$ and
    $k_i^- = -\sigma^2 S_i^2 \frac{\Delta t}{2\Delta S^2} + rS_i\frac{\Delta t}{2\Delta S}$ and
    $k_i^0 = 1 + r\Delta t + \sigma^2 S_i^2 \frac{\Delta t}{\Delta S^2}$.
  * In fact we don't even have to invert the matrix K_j here because it is a tri-diagonal form that allows the system to be solved quickly by back-substitution. (Matlab does this automatically via their backslash matrix division operator.)

• Using this approach we can solve the linear system for the time t_{n-1} grid of solutions for $C(t_n)$.

  - And of course it's trivial to adapt the above scheme for the version of the PDE that includes continuous payouts to either the S or C or both.

• I skipped a step here however: we actually can't build the system as I described at the very top and very bottom points of our i-vector.

  - The $i = 1$ and $i = N$ nodes will involve C values that are off the grid.
  - So the PDE only actually yields $N-2$ equations.
  - We have to supply two more equations to close the system.
  - If we know the value exactly on one of the boundaries, that provides an additional equation.
  - Otherwise, another possibility is to make the grid large enough that we feel confident that the *curvature is going to zero* near the boundary. (Of course we need to test this assumption.)
    * Then we would have an additional equation that said, for example,

$$[1, \quad -2, \quad 1] \begin{bmatrix} C_N \\ C_{N-1} \\ C_{N-2} \end{bmatrix} = 0.$$

- Finite difference methods restore many of the benefits of binomial trees, because, by working backwards in time, it is easy to evaluate early-exercise type decisions.
- Also notice that we can easily solve the generalized version of the PDE that incorporates *state dependence* in any of the coefficients, e.g., $\sigma = \sigma(S, t)$ or $r = r(S, t)$.
  - Since we know the values of the arguments at each node in the grid, we can simply evaluate these coefficients and plug them straight into the approporiate coefficients in our linear system.