Finite Difference Methods

Goals

- To understand Black Scholes PDE 1o1 and its implementation in C++
- Understand ‘big picture’ from BS -> PDE -> FDM -> C++
- Extend and debug existing (possibly undocumented) application code
- Get an overview of popular FD methods as used in computational finance

Scope Problem

- One-factor plain option (exercise at $t = T$)
- Barrier options
- Introduce early exercise by checking constraint at each time level
- Different levels of how flexible the solution will be

Black Scholes PDE

- Describes the behaviour in time and space (S, t) of an option
- Time-dependent convection-diffusion equation
- Need extra boundary and initial conditions
- Care to be taken with truncation/transformation of the S domain

BS PDE

- $-\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$
- Add boundary conditions at S = 0 and S = Smax
- Initial condition is the option payoff
- These result in complete description of the PDE problem

5

Boundary Conditions

- Put

$$P(0,t) = Ke^{-rt}, \ \lim_{S\to\infty} P(S,t) = 0$$

- Call

$$C(0,t) = 0, \ \lim_{S\to\infty} C(S,t) = S$$

6

Finite Differencing

- Approximate PDE problem on a continuous space by finite differences on discrete space
- Approximate 2nd and 1st order derivatives in S by centred differences
- Approximation in time can be Backward, Forward or Centred in time
- (!! Stability and accuracy of finite difference schemes)

Explicit Euler

- Express the solution at level n+1 in terms of solution at time level n
- Take boundary conditions into account

$$-\frac{V_j^{n+1}-V_j^n}{k}+\frac{1}{2}\,\sigma^2\,S_j^2\left(\frac{V_{j+1}^n-2V_j^n+V_{j-1}^n}{k^2}\right)$$
$$+r\,S_j\left(\frac{V_{j+1}^n-V_{j-1}^n}{h^2}\right)-r\,V_j^n=0$$

$$V_j^{n+1}=\alpha_j\,V_{j-1}^n+\beta_j\,V_j^n+\gamma_j\,V_{j+1}^n$$

- No matrix inversion needed

Implicit Euler

$$-\frac{V_j^{n+1}-V_j^n}{k}+rj\Delta S\left(\frac{V_{j+1}^{n+1}-V_{j-1}^{n+1}}{2\Delta S}\right)$$
$$+\frac{1}{2}\sigma^2 j^2\Delta S^2\left(\frac{V_{j+1}^{n+1}-2V_j^{n+1}+V_{j-1}^{n+1}}{\Delta S^2}\right)=rV_j^{n+1}$$

$$a_j^{n+1}\,V_{j-1}^{n+1}+b_j^{n+1}\,V_j^{n+1}+c_j^{n+1}\,V_j^{n+1}=F_j^{n+1}$$

- Solve as a matrix system

9

Crank Nicolson

- Average of implicit and explicit Euler

$$-\frac{V_j^{n+1}-V_j^n}{k}+rj\Delta S\left(\frac{V_{j+1}^{n+\frac{1}{2}}-V_{j-1}^{n+\frac{1}{2}}}{2\Delta S}\right)$$
$$+\frac{1}{2}\sigma^2 j^2\Delta S^2\left(\frac{V_{j+1}^{n+\frac{1}{2}}-2V_j^{n+\frac{1}{2}}+V_{j-1}^{n+\frac{1}{2}}}{\Delta S^2}\right)=\tau V_j^{n+\frac{1}{2}}$$

$$\left(V_j^{n+\frac{1}{2}}\equiv\frac{1}{2}\left(V_j^{n+1}+V_j^n\right)\right)$$

10

Sanity Check

- Can use put-call parity to check put and call from FDM
- We also have explicit solutions for plain options

$$C(t) - P() = S(t) - KB(t,T)$$

$$B(t,T) = e^{-r(T-t)} \text{ (bond maturing at T)}$$

General PDE Formulation

$$Lu \equiv -\frac{\partial u}{\partial t} + \sigma(x,t)\frac{\partial^2 u}{\partial x^2} + \mu(x,t)\frac{\partial u}{\partial x} + b(x,t)u = f(x,t) \text{ in } 0.1$$

$$u(x,0) = \varphi(x),\ x \in \Omega$$

$$u(A,t) = g_0(t),\ u(B,t) = g_1(t),\ t \in (0,T)$$

$$D = (0, X\max) \times (0,T)$$

$$\Omega = (0, X\max)$$

Programming Tips

- Make sure you have worked out your algorithm properly
- Make sure C++ code ‘mirrors’ the algorithm
- Take care with data structures and indexing
- Take it step-by-step

13