CS 774 Assignment 2

Fall 2009

Instructors:
Peter Forsyth, *paforsyt@uwaterloo.ca* Office: DC3631
Office Hours: Tues: 4:00-5:00; Thurs: 11:00-12:00
Lectures:MWF 3:30-4:20 MC2036

CS 774 Web Site: www.student.cs.uwaterloo.ca/∼cs774/

Assignments 1 and 2 are not to be handed in, but are meant to give you a head start on the course project. If you have any problems, be sure to come and see me.

Numerical PDE Option Pricing

Basic Tasks
The objective of this assignment is to use Matlab to familiarize yourselves with Numerical PDE option pricing.

Suppose we want to solve the BS equation, for an American option. Formally, we can state the problem as a linear complementarity problem:

$$-\frac{\partial V}{\partial \tau} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV \leq 0 \tag{1}$$

$$V^* \leq V \tag{2}$$

where one of (1-2) holds with equality. V^* is typically the payoff condition. Boundary conditions are typically:

$$\begin{aligned} -\frac{\partial V}{\partial \tau} - r(t)V &\leq 0 \\ V^* &\leq V \end{aligned} \tag{3}$$

as $S \to 0$, and, boundary conditions for large S are

$$\begin{aligned} V &\simeq 0 \quad S \to \infty \quad ; \quad \text{for a put} \\ V &\simeq S \quad S \to \infty \quad ; \quad \text{for a call} \end{aligned} \tag{4}$$

Usual payoff conditions are

$$\begin{aligned} V(S, \tau = 0) &= \max(K - S, 0) \quad \text{for a put} \\ V(S, \tau = 0) &= \max(S - K, 0) \quad \text{for a call} \end{aligned} \tag{5}$$

where K is the strike price.

Let

$$\mathcal{L}V \equiv \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV \tag{6}$$

and the discrete form of $\mathcal{L}V$ at node (S_i, τ^n) be denoted by $(\mathcal{L}V)_i^n$. Let $V_i^n = V(S_i, \tau^n)$ be the discrete values of the option.

Then, we can write the C-N timestepping as

$$\left[\frac{V_i^{n+1} - V_i^n}{\Delta \tau}\right] = (1-\theta)(\mathcal{L}V)_i^{n+1} + \theta(\mathcal{L}V)_i^n + q_i^{n+1}, \tag{7}$$

where the penalty term q_i^{n+1} is defined as

$$q_i^{n+1} = P(V_i^{n+1}) \frac{1}{\Delta\tau}(V_i^* - V_i^{n+1}) \tag{8}$$

with

$$\begin{aligned} P(V_i^{n+1}) &= Large \quad \text{if } V_i^{n+1} < V_i^* \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{9}$$

and where *Large* is a large number. Note that $\theta = 0$ is fully implicit, $\theta = 1/2$ is Crank-Nicolson. The factor *Large* in equation (9) should be selected so that

$$Large \simeq \frac{1}{tol} \tag{10}$$

where *tol* is the convergence tolerance for the iteration.

Defining

$$V^{n+1} = \begin{bmatrix} V_0^{n+1} \\ V_1^{n+1} \\ | \\ V_m^{n+1} \end{bmatrix} \quad ; \quad V^n = \begin{bmatrix} V_0^n \\ V_1^n \\ | \\ V_m^n \end{bmatrix} \quad ; \quad V^* = \begin{bmatrix} V_0^* \\ V_1^* \\ | \\ V_m^* \end{bmatrix}, \tag{11}$$

and

$$\left[\hat{M} V^n\right]_{row\ i} = -\Delta\tau \alpha_i V_{i-1}^n + \Delta\tau(\alpha_i + \beta_i + r)V_i^n - \Delta\tau\beta_i V_{i+1}^n \tag{12}$$

where α_i, β_i are defined as in the course notes. Note that the first and last rows of $\hat{M}$ will have to be altered to take into account the boundary conditions.

Let the diagonal matrix $\bar{P}$ is given by

$$\begin{aligned} \bar{P}(V^{n+1})_{ii} &= Large \quad \text{if } V_i^{n+1} < V_i^* \\ &= 0 \quad \text{otherwise} \\ \bar{P}(V^{n+1})_{ij} &= 0 \quad \text{if } i \neq j \end{aligned} \tag{13}$$

then we can write equation (7) as

$$\begin{aligned} \left[I + (1-\theta)\hat{M} + \bar{P}(V^{n+1})\right] V^{n+1} &= \left[I - \theta\hat{M}\right] V^n + \left[\bar{P}(V^{n+1})\right] V^* \\ & (V^*)_i = V_i^*. \end{aligned} \tag{14}$$

An iterative method for solving the nonlinear equations (14) is described in the course notes. The theory behind the penalty method is discussed in the paper *Quadratic convergence for a Penalty method for American options*, by Forsyth and Vetzal, Siam J. Sci. Comp. (2002) vol 23, pages 2096-2123.

Basic Tasks

Using the data in Table 1, code up the above PDE method in Matlab. You will have to solve a tridiagonal matrix. DO NOT use the full matrix objects in Matlab. Use the sparse matrix objects. Use the Matlab sparse matrix function *spalloc* to allocate a sparse matrix of the correct size, and then insert nonzero elements into the matrix and right hand side to correctly represent the system of equations.

First, solve for a European Put option, using the data given in the Table, (with CN timestepping) and compare with the exact solution using *blsprice*. Experiment with different grid spacings and timestep sizes. You might find that an unequally spaced grid, with finer spacing near the exercise price is useful (assuming you are interested in the price near the the exercise price). Do you observe second order convergence? An example of a simple unequally spaced grid is given by (Matlab notation)

```
S = [0:0.1*K:0.4*K,...
        0.45*K:0.05*K:0.8*K,...
        0.82*K:0.02*K:0.9*K,...
        0.91*K:0.01*K:1.1*K,...
        1.12*K:0.02*K:1.2*K,...
        1.25*K:.05*K:1.6*K,...
        1.7*K:0.1*K:2*K,...
        2.2*K, 2.4*K, 2.8*K,...
        3.6*K, 5*K, 7.5*K, 10*K];
```

where K is the strike, and we are interested in the solution near $S = K$.

Plot delta V_S and gamma V_{SS}, as well as the price (near the exercise price). Carry out a convergence study, using different mesh sizes and timesteps (a financial year has 250 days). A reasonable number of timesteps to use on a coarse grid would be about $T/25$. You may find that if you are using Crank-Nicolson timestepping, that you may have to take two fully implicit steps at the start, and then C-N thereafter. This is discussed in the course notes.

Note: to carry out a convergence study, you should solve the pricing problem on a sequence of problems. Each problem has a twice as many nodes as the previous problem (new nodes inserted halfway between the coarse grid nodes) and the timestep size is halved. Assuming that

$$Error \;=\; O((\Delta t)^2, (\Delta S)^2) \;;\; \Delta S = \max_i (S_{i+1} - S_i) \tag{15}$$

then if we let

$$\begin{aligned} h &= C_1 \Delta S \\ h &= C_2 \Delta t \end{aligned}$$

then, if we can label each problem in the above sequence by a set of h values, so that the solution on each grid (at a given point) has the form

$$\begin{aligned} V(h) &= V_{exact} + Ah^2 \\ V(h/2) &= V_{exact} + A(h/2)^2 \\ V(h/4) &= V_{exact} + A(h/4)^2 \end{aligned} \tag{16}$$

where we have assumed that the mesh size and timestep are small enough that the A in equation (16) is approximately constant. Now, equation (16) implies that

$$\frac{V(h) - V(h/2)}{V(h/2) - V(h/4)} \simeq 4 \tag{17}$$

Check the theory by examining the rate of convergence of your pricer.

Note the delta and gamma can be computed by

$$\begin{aligned} \Delta_i^n &\simeq \frac{V_{i+1}^n - V_{i-1}^n}{S_{i+1} - S_{i-1}} \\ \Gamma_i^{n+1} &\simeq \frac{1}{(\Delta S_{i+1/2} + \Delta S_{i-1/2})/2} \left[\left(\frac{V_{i+1}^{n+1} - V_i^{n+1}}{\Delta S_{i+1/2}} \right) + \left(\frac{V_{i-1}^{n+1} - V_i^{n+1}}{\Delta S_{i-1/2}} \right) \right] \end{aligned} \tag{18}$$

Now, price an American put, with the same data. I get about \$3.07 for the price at $S = 100$. (There is no exact solution in this case). Experiment with different convergence tolerances for the iteration within each timestep.

As another check, for the case of $T = 1yr$, $\sigma = .40$, $r = .06$, $S = 100$, (American put), I get 13.2957.

Compare with applying the American constraint explicitly.

Some additional tasks

Table 1: A typical test case

σ	.2
r	.10
Time to expiry	0.25 years
Strike Price	\$100
Initial asset price S^0	\$100

1. Experiment with a European digital payoff, i.e. the payoff is

$$\begin{aligned} payoff &= 1 \quad \text{if} \quad S > K \\ &= 0 \quad \text{if} \quad S \leq K \end{aligned}$$

   Try fully implicit and CN timestepping, plot delta and gamma. You may see some oscillations if CN is used. These can be cured by the following trick: take two initial steps of fully implicit, followed by CN. You should also average the payoff at the discontinuity. Carry out a convergence study, using smaller timesteps and a finer mesh.

2. Use a second order BDF timestepping method (see course notes) to price American options. Carry out a convergence study.

3. Try using the boundary condition $V_{SS} = 0, S \to \infty$ (see the section (Sec 22) in the course notes on this). Again, this should not require many changes to your code. An advantage of this method is that we do not have to guess the form of the solution for large values of S. The boundary condition is imposed automatically.

4. You can get a head start on the project by writing the following function. Solve the pricing PDE and store V_S, V_{SS} at all grid points and timesteps (S_i, τ^n) (use a couple of arrays). Then, write a function which takes as input

   - The arrays of stored information, delta and gamma at each grid point and timestep.
   - A discrete time τ^* (which may not coincide with $\{\tau^n\}$), and an array of S values, which may not coincide with the grid points.

   The output of this function should be

   - Arrays of V_S, V_{SS}, corresponding to the input data. You will have to linearly interpolate the input data arrays to produce the output.

   Try to vectorize this as much as possible.

   You will need this function when you simulate a hedging strategy. Each simulation path will require determination of V_S, V_{SS} at each discrete hedging time.

5. You will also need to keep track of the exercise boundary. You will need to generate an array where you store the exercise boundary $S^*(t_i)$ at each timestep. You will need this to determine if the option is going to be exercised during your hedging simulations.