FINITE DIFFERENCE - CRANK NICOLSON

Dr P. V. Johnson

School of Mathematics

2013

Outline

1. Review
   - Last time...
   - Today's lecture

Outline

1. Review
   - Last time...
   - Today's lecture
2. Improved Finite Difference Methods
   - The Crank-Nicolson Method
   - SOR method

Outline

1. Review
   - Last time...
   - Today's lecture
2. Improved Finite Difference Methods
   - The Crank-Nicolson Method
   - SOR method
3. Exotic options
   - American options
   - Convergence and accuracy

Outline

1. Review
    - Last time...
    - Today's lecture
2. Improved Finite Difference Methods
    - The Crank-Nicolson Method
    - SOR method
3. Exotic options
    - American options
    - Convergence and accuracy
4. Summary
    - Overview

MANCHESTER 1824

- Introduced the finite-difference method to solve PDEs
- Discetise the original PDE to obtain a linear system of equations to solve.
- This scheme was explained for the Black Scholes PDE and in particular we derived the explicit finite difference scheme to solve the European call and put option problems.

- Introduced the finite-difference method to solve PDEs
- Discetise the original PDE to obtain a linear system of equations to solve.
- This scheme was explained for the Black Scholes PDE and in particular we derived the explicit finite difference scheme to solve the European call and put option problems.
- The convergence of the method is similar to the binomial tree and, in fact, the method can be considered a trinomial tree.
- Explicit method can be unstable - constraints on our grid size.

- Here we will introduce the Crank-Nicolson method
- The method has two advantages over the explicit method:
  - stability;
  - improved convergence.
- Here we will need to solve a matrix equation.

- Here we will introduce the Crank-Nicolson method
- The method has two advantages over the explicit method:
  - stability;
  - improved convergence.
- Here we will need to solve a matrix equation.
- In addition we will discuss how to price American options
- and how to remove nonlinearity error in a variety of cases.

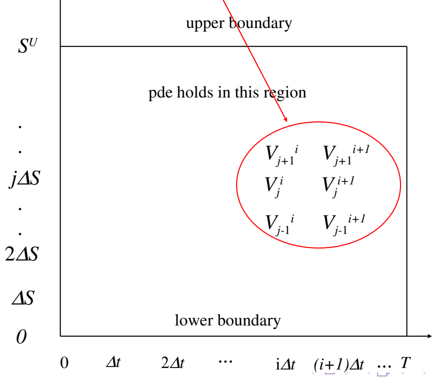
Crank Nicolson Method

- The Crank-Nicolson scheme works by evaluating the derivatives at $V(S, t + \Delta t/2)$.
- The main advantages of this are:
  - error in the time now $(\Delta t)^2$
  - no stability constraints
- Crank-Nicolson method is implicit, we will need to use three option values in the future $(t + \Delta t)$
- to calculate three option values at (t).

Crank-Nicolson grid

Focus attention on i, j-th value V_j^i, and a little

piece of the grid around that point



Approximating at the half step

- Now take approximations to the derivatives at the half step $t + 1/2\Delta t$
- They are in terms of V_j^i, as follows:

$$\frac{\partial V}{\partial t} \approx \frac{V_j^{i+1} - V_j^i}{\Delta t}$$

$$\frac{\partial V}{\partial S} \approx \frac{1}{4\Delta S}(V_{j+1}^i - V_{j-1}^i + V_{j+1}^{i+1} - V_{j-1}^{i+1})$$

$$\frac{\partial^2 V}{\partial S^2} \approx \frac{1}{2\Delta S^2}(V_{j+1}^i - 2V_j^i + V_{j-1}^i + V_{j+1}^{i+1} - 2V_j^{i+1} + V_{j-1}^{i+1})$$

Deriving the equation

- Here the V^i values are all unknown, so...
  - rearrange our equations to have the known values on one side
  - the unknown values on the other.

$$\tfrac{1}{4}(\sigma^2 j^2 - rj)V^i_{j-1} + \left(-\frac{\sigma^2 j^2}{2} - \frac{r}{2} - \frac{1}{\Delta t}\right)V^i_j + \tfrac{1}{4}(\sigma^2 j^2 + rj)V^i_{j+1} =$$
$$-\tfrac{1}{4}(\sigma^2 j^2 - rj)V^{i+1}_{j-1} - \left(-\frac{\sigma^2 j^2}{2} - \frac{r}{2} + \frac{1}{\Delta t}\right)V^{i+1}_j - \frac{1}{4}(\sigma^2 j^2 + rj)V^{i+1}_{j+1}$$

- There is one of these equations for each point in the grid

MATRIX EQUATIONS

- We can rewrite the valuation problem in terms of a matrix as follows:

$$
\begin{pmatrix}
b_0 & c_0 & 0 & 0 & . & . & . & . & 0 \\
a_1 & b_1 & c_1 & 0 & . & . & . & . & . \\
0 & a_2 & b_2 & c_2 & 0 & . & . & . & . \\
. & 0 & a_3 & b_3 & c_3 & 0 & . & . & . \\
. & . & . & . & . & . & . & . & . \\
. & . & . & 0 & a_j & b_j & c_j & 0 & . \\
. & . & . & . & . & . & . & . & . \\
0 & . & . & . & . & . & 0 & a_{jmax} & b_{jmax}
\end{pmatrix}
\begin{pmatrix}
V_0^i \\ V_1^i \\ V_2^i \\ V_3^i \\ . \\ . \\ V_{jmax-1}^i \\ V_{jmax}^i
\end{pmatrix}
=
\begin{pmatrix}
d_0^i \\ d_1^i \\ d_2^i \\ d_3^i \\ . \\ . \\ d_{jmax-1}^i \\ d_{jmax}^i
\end{pmatrix}
$$

Matrix equations

where:

$$
\begin{aligned}
a_j &= \tfrac{1}{4}(\sigma^2 j^2 - rj) \\
b_j &= -\frac{\sigma^2 j^2}{2} - \frac{r}{2} - \frac{1}{\Delta t} \\
c_j &= \tfrac{1}{4}(\sigma^2 j^2 + rj) \\
d_j &= -\tfrac{1}{4}(\sigma^2 j^2 - rj)V_{j-1}^{i+1} - \left(-\frac{\sigma^2 j^2}{2} - \frac{r}{2} + \frac{1}{\Delta t}\right)V_j^{i+1} \\
&\quad -\tfrac{1}{4}(\sigma^2 j^2 + rj)V_{j+1}^{i+1}
\end{aligned}
$$

What to do on the boundaries?

- Boundary conditions are an important part of solving any PDE
- For most PDEs we know the boundary conditions for large and small S
- For call options $a_{jmax} = 0$, $b_{jmax} = 1$, $d_{jmax} = S^u e^{-\delta(T-i\Delta t)} - Xe^{-r(T-i\Delta t)}$, $b_0 = 1$, $c_0 = 0$, $d_0 = 0$
- For put options $b_0 = 1$, $c_0 = 0$, $d_0 = Xe^{-r(T-i\Delta t)}$, $a_{jmax} = 0$, $b_{jmax} = 1$, $d_{jmax} = 0$
- In general we can always determine the values of b_0, c_0, d_0, a_{jmax}, b_{jmax} and d_{jmax} from our boundary conditions.

THE CRANK-NICOLSON METHOD

- At each point in time we need to solve the matrix equation in order to calculate the V_j^i values.
- There are two approaches to doing this,
  - solve the matrix equation directly (LU decomposition),
  - solve the matrix equation via an iterative method (SOR).
- If possible, the LU approach is the preferred approach as it gives you an exact value for V_j^i and is much faster.
- However, not possible to use LU approach with American options.
- The SOR (Successive Over Relaxation) can be easily adapted to value American options

Matrix Equations

- The SOR method is a simpler approach but can take a little longer as it relies upon iteration.
- If we consider each of the individual equations from $\mathbf{AV} = \mathbf{d}$ we have that

$$
\begin{aligned}
a_1 V_0^i + b_1 V_1^i + c_1 V_2^i &= d_1^i \\
a_2 V_1^i + b_2 V_2^i + c_2 V_3^i &= d_1^i \\
\dots\dots\dots\dots &= \dots \\
a_j V_{j-1}^i + b_j V_j^i + c_j V_{j+1}^i &= d_j^i \\
\dots\dots\dots\dots &= \dots \\
a_{jmax-1} V_{jmax-2}^i + b_{jmax-1} V_{jmax-1}^i + c_{jmax-1} V_{jmax}^i &= d_{jmax-1}^i
\end{aligned}
$$

Jacobi Iteration

- Rearrange these equations to get:

$$V_j^i = \frac{1}{b_j}(d_j^i - a_j V_{j-1}^i - c_j V_{j+1}^i)$$

- The Jacobi method is an iterative one that relies upon the previous equation.
  - Taking an initial guess for V_j^i, denoted as $V_j^{i,0}$
  - iterate using the formula below for the $(k+1)$th iteration:

$$V_j^{i,k+1} = \frac{1}{b_j}(d_j^i - a_j V_{j-1}^{i,k} - c_j V_{j+1}^{i,k})$$

  - carry on until the difference between $V_j^{i,k}$ and $V_j^{i,k+1}$ is sufficiently small for all j.

Jacobi Iteration

- Rearrange these equations to get:

$$V_j^i = \frac{1}{b_j}(d_j^i - a_j V_{j-1}^i - c_j V_{j+1}^i)$$

- The Jacobi method is an iterative one that relies upon the previous equation.
  - Taking an initial guess for V_j^i, denoted as $V_j^{i,0}$
  - iterate using the formula below for the $(k+1)$th iteration:

$$V_j^{i,k+1} = \frac{1}{b_j}(d_j^i - a_j V_{j-1}^{i,k} - c_j V_{j+1}^{i,k})$$

  - carry on until the difference between $V_j^{i,k}$ and $V_j^{i,k+1}$ is sufficiently small for all j.
- For Gauss-Seidel use the most up-to-date guess where possible:

SOR

- The SOR method is another slight adjustment. It starts from the trivial observation that

$$V_j^{i,k+1} = V_j^{i,k} + (V_j^{i,k+1} - V_j^{i,k})$$

- and so $(V_j^{i,k+1} - V_j^{i,k})$ is a correction term.
- Now try to *over* correct value, should work faster.
- This is true if $V_j^{i,k} \to V_j^i$ monotonically in k.

SOR

- The SOR method is another slight adjustment. It starts from the trivial observation that

$$V_j^{i,k+1} = V_j^{i,k} + (V_j^{i,k+1} - V_j^{i,k})$$

- and so $(V_j^{i,k+1} - V_j^{i,k})$ is a correction term.
- Now try to *over* correct value, should work faster.
- This is true if $V_j^{i,k} \to V_j^i$ monotonically in k.
- So the SOR algorithm says that

$$y_j^{i,k+1} = \frac{1}{b_j}(d_j^i - a_j V_{j-1}^{i,k+1} - c_j V_{j+1}^{i,k})$$

$$V_j^{i,k+1} = V_j^{i,k} + \omega(y_j^{i,k+1} - V_j^{i,k})$$

where $1 < \omega < 2$ is called the over-relaxation parameter.

AMERICAN OPTIONS: EXPLICIT

- American option pricing problem requires an optimal early exercise strategy.
- To generate one, compare the continuation value with the early exercise value - take the larger.
- With the explicit finite difference method is pretty straightforward
  - calculate the continuation value CoV_j^i

$$CoV_j^i = \frac{1}{1 + r\Delta t}\left(AV_{j+1}^{i+1} + BV_j^{i+1} + CV_{j-1}^{i+1}\right)$$

  - then compare this to the early exercise payoff.

American options: Explicit

- American option pricing problem requires an optimal early exercise strategy.
- To generate one, compare the continuation value with the early exercise value - take the larger.
- With the explicit finite difference method is pretty straightforward
  - calculate the continuation value CoV_j^i

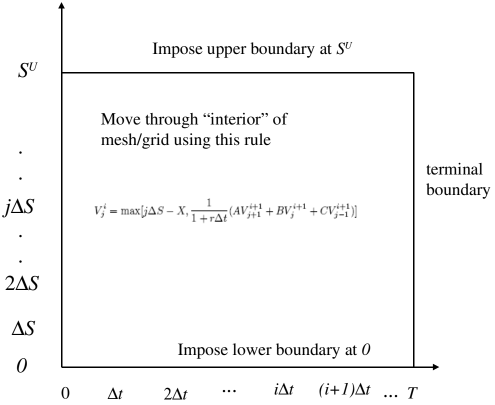
$$CoV_j^i = \frac{1}{1+r\Delta t}(AV_{j+1}^{i+1} + BV_j^{i+1} + CV_{j-1}^{i+1})$$

  - then compare this to the early exercise payoff.
- Thus for a put:

$$V_j^i = \max[X - j\Delta S, \frac{1}{1+r\Delta t}(AV_{j+1}^{i+1} + BV_j^{i+1} + CV_{j-1}^{i+1})]$$

- This is similar to using the binomial tree

American put option: Explicit



American put option: C-N

- The American option pricing problem is slightly more complex for the Crank-Nicolson method.
- Consider the process of calculating V_j^i...

American put option: C-N

- The American option pricing problem is slightly more complex for the Crank-Nicolson method.
- Consider the process of calculating V_j^i...
- The value of the option V_j^i, for all values of j, depends also upon the value of V_{j-1}^i and V_{j+1}^i.
- Optimally deciding when to early exercise requires that we already know these values.
- If we early exercise at some point this could change V_j^i for **all** j.

PSOR

- A simple solution to this problem is to project our SOR method (Projected SOR)
- In order to project, check whether or not it would be optimal to exercise at each iteration.

PSOR

- A simple solution to this problem is to project our SOR method (Projected SOR)
- In order to project, check whether or not it would be optimal to exercise at each iteration.
- This changes

$$y_j^{i,k+1} = \frac{1}{b_j}(d_j^i - a_j V_{j-1}^{i,k+1} - c_j V_{j+1}^{i,k})$$

$$V_j^{i,k+1} = V_j^{i,k} + \omega(y_j^{i,k+1} - V_j^{i,k})$$

to (in the case of the American put option)

$$y_j^{i,k+1} = \frac{1}{b_j}(d_j^i - a_j V_{j-1}^{i,k+1} - c_j V_{j+1}^{i,k})$$

$$V_j^{i,k+1} = \max(V_j^{i,k} + \omega(y_j^{i,k+1} - V_j^{i,k}), X - j\Delta S)$$

Convergence

- If the option price and the derivatives are well behaved then the errors of the
  - Explicit method are $O(\Delta t, (\Delta S)^2)$
  - Crank-Nicolson method are $O((\Delta t)^2, (\Delta S)^2)$.
- These can be considered similar to the distribution error for the binomial tree.
- If convergence is smooth we can use extrapolation.

Convergence

- If the option price and the derivatives are well behaved then the errors of the
  - Explicit method are $O(\Delta t, (\Delta S)^2)$
  - Crank-Nicolson method are $O((\Delta t)^2, (\Delta S)^2)$.
- These can be considered similar to the distribution error for the binomial tree.
- If convergence is smooth we can use extrapolation.
- Finite-difference methods can suffer from non-linearity error if the grid is not correctly aligned with respect to any discontinuities
  - in the option value,
  - or in the derivatives of the option value.

Non-linearity error

- Now have the freedom to construct the grid as desired.
- Makes it is simple to construct the grid so that you have a grid point upon any discontinuities.
- For example, if we consider an European call or put option then the only source of non-linearity error is at $S = X$ at expiry.

Non-linearity error

- Now have the freedom to construct the grid as desired.
- Makes it is simple to construct the grid so that you have a grid point upon any discontinuities.
- For example, if we consider an European call or put option then the only source of non-linearity error is at $S = X$ at expiry.
- Always choose ΔS so that $X = j\Delta S$ for some integer value of j.
- So if in this case $S_0 = 100$ and $X = 95$, you need a suitably large S^U and a ΔS which is a divisor of 95.

Barrier options

- When pricing barrier options, there is a large amount of non-linearity error that comes from not having the nodes in the tree aligned with the position of the barrier.
- Thus with barrier options we have two sources of non-linearity error
  - the error from the barrier
  - the error from the discontinuous payoff.

Barrier options

- When pricing barrier options, there is a large amount of non-linearity error that comes from not having the nodes in the tree aligned with the position of the barrier.
- Thus with barrier options we have two sources of non-linearity error
  - the error from the barrier
  - the error from the discontinuous payoff.
- Simply match the grid to the barrier and the payoff.
- For a down and out barrier option choose S^L (the lower value of S) to be on the barrier and then, as in the previous example, choose ΔS so that the exercise price is also on a node.

Overview

- We have introduced the Crank-Nicolson finite difference method.
- It is:
  - slightly harder to program;
  - has faster convergence;
  - better stability properties.

Overview

- We have introduced the Crank-Nicolson finite difference method.
- It is:
  - slightly harder to program;
  - has faster convergence;
  - better stability properties.
- Applying the method to American options requires the use of PSOR
- more complex than the method for valuing American options using the explicit method.
- Can choose the dimensions of the grid so as to remove the nonlinearity error.