Further Finite-difference Methods for One-factor Models

In this lecture. . .

- implicit finite-difference methods including Crank–Nicolson
- Douglas schemes
- Richardson extrapolation
- American-style exercise and exotic options

By the end of this lecture you will

- know several more ways of solving parabolic partial differential equations numerically

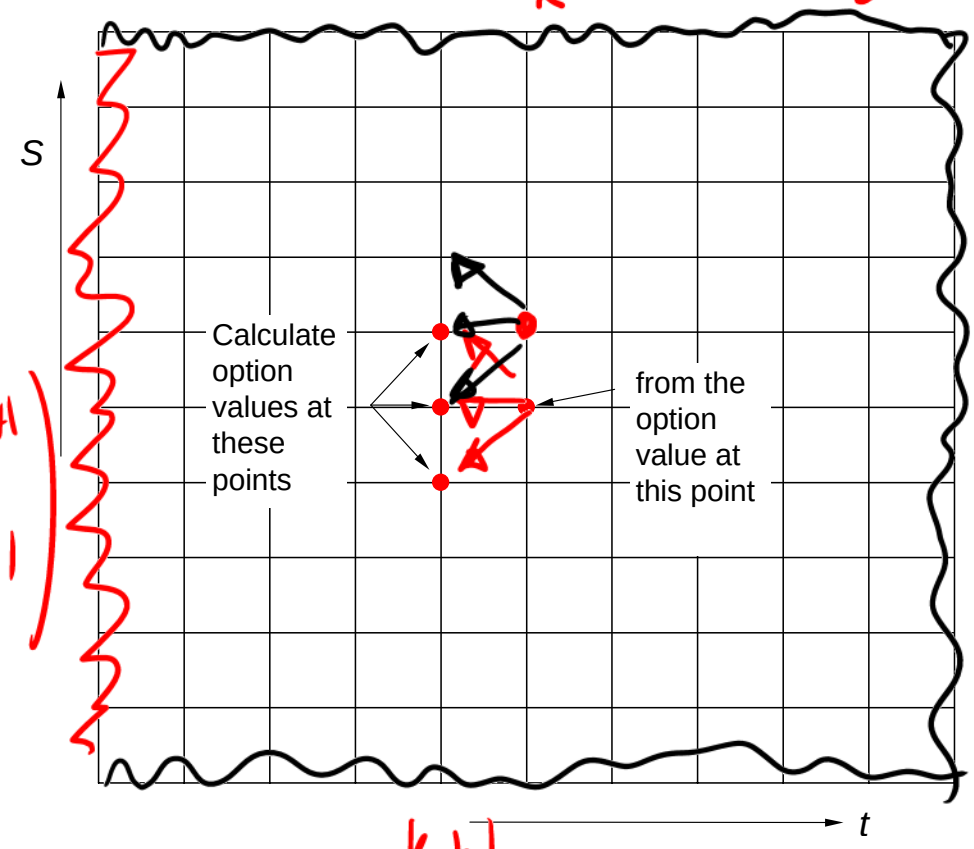
Introduction

There are many more ways of solving parabolic partial differential equations than the explicit method.

The more advanced methods are usually more complicated to program but have advantages in terms of stability and speed.

Implicit finite-difference methods

The **fully implicit method** uses the points shown in the figure below to calculate the option value.



The relationship between the option values on the mesh is simply

$$\frac{V_i^k - V_i^{k+1}}{\delta t} + a_i^{k+1}\left(\frac{V_{i+1}^{k+1} - 2V_i^{k+1} + V_{i-1}^{k+1}}{\delta S^2}\right)$$

$$+b_i^{k+1}\left(\frac{V_{i+1}^{k+1} - V_{i-1}^{k+1}}{2\delta S}\right) + c_i^{k+1}V_i^{k+1} = 0.$$

The method is accurate to $O(\delta t, \delta S^2)$.

This can be written as

$$A_i^{k+1} V_{i-1}^{k+1} + (1 + B_i^{k+1}) V_i^{k+1} + C_i^{k+1} V_{i+1}^{k+1} = V_i^k \qquad (1)$$

where

$$\begin{cases} A_i^{k+1} = -\nu_1 a_i^{k+1} + \frac{1}{2}\nu_2 b_i^{k+1}, \\ B_i^{k+1} = 2\nu_1 a_i^{k+1} - \delta t\, c_i^{k+1} \\ C_i^{k+1} = -\nu_1 a_i^{k+1} - \frac{1}{2}\nu_2 b_i^{k+1} \end{cases}$$

and

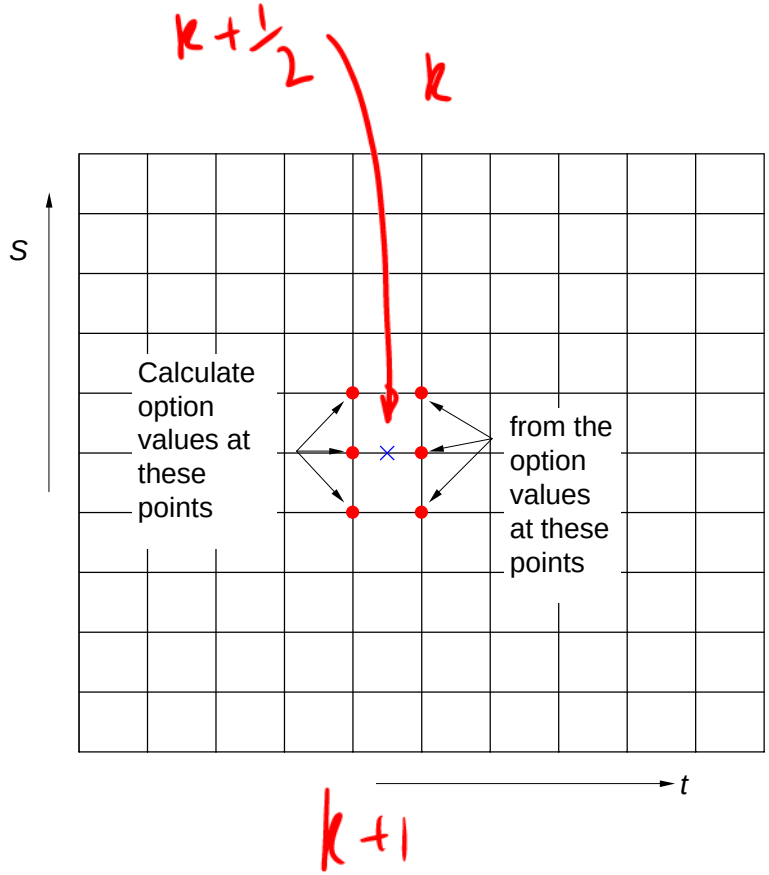
where

$$\nu_1 = \frac{\delta t}{\delta S^2} \quad \text{and} \quad \nu_2 = \frac{\delta t}{\delta S}.$$

Equation (1) does not hold for $i = 0$ or $i = I$, the boundary conditions supply the two remaining equations.

The Crank–Nicolson method

The **Crank–Nicolson method** can be thought of as an average of the explicit method and the fully implicit method. It uses the six points shown in the figure below.



The Crank–Nicolson scheme is

$$\frac{V_i^k - V_i^{k+1}}{\delta t} + \frac{a_i^{k+1}}{2}\left(\frac{V_{i+1}^{k+1} - 2V_i^{k+1} + V_{i-1}^{k+1}}{\delta S^2}\right) + \frac{a_i^k}{2}\left(\frac{V_{i+1}^k - 2V_i^k + V_{i-1}^k}{\delta S^2}\right)$$

$$+ \frac{b_i^{k+1}}{2}\left(\frac{V_{i+1}^{k+1} - V_{i-1}^{k+1}}{2\,\delta S}\right) + \frac{b_i^k}{2}\left(\frac{V_{i+1}^k - V_{i-1}^k}{2\,\delta S}\right)$$

$$+ \tfrac{1}{2}c_i^{k+1}V_i^{k+1} + \tfrac{1}{2}c_i^k V_i^k = O(\delta t^2, \delta S^2).$$

This can be written as

$$-A_i^{k+1}V_{i-1}^{k+1} + (1 - B_i^{k+1})V_i^{k+1} - C_i^{k+1}V_{i+1}^{k+1}$$
$$= A_i^k V_{i-1}^k + (1 + B_i^k)V_i^k + C_i^k V_{i+1}^k,$$

where

$$A_i^k = \tfrac{1}{2}\nu_1 a_i^k - \tfrac{1}{4}\nu_2 b_i^k, B_i^k = -\nu_1 a_i^k + \tfrac{1}{2}\delta t\, c_i^k, C_i^k = \tfrac{1}{2}\nu_1 a_i^k + \tfrac{1}{4}\nu_2 b_i^k.$$

The beauty of this method lies in its stability and accuracy. The error in the method is $O(\delta t^2, \delta S^2)$.

The Crank–Nicolson method can be written in the matrix form

$$\begin{pmatrix} -A_1^{k+1} & 1-B_1^{k+1} & -C_1^{k+1} & 0 & . & . & . \\ 0 & -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} & 0 \\ . & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} & -C_{I-1}^{k+1} \end{pmatrix} \begin{pmatrix} V_0^{k+1} \\ V_1^{k+1} \\ . \\ . \\ . \\ V_{I-1}^{k+1} \\ V_I^{k+1} \end{pmatrix}$$

$$= \begin{pmatrix} A_1^k & 1+B_1^k & C_1^k & 0 & . & . & . \\ 0 & A_2^k & 1+B_2^k & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1+B_{I-2}^k & C_{I-2}^k & 0 \\ . & . & . & 0 & A_{I-1}^k & 1+B_{I-1}^k & C_{I-1}^k \end{pmatrix} \begin{pmatrix} V_0^k \\ V_1^k \\ . \\ . \\ . \\ V_{I-1}^k \\ V_I^k \end{pmatrix}$$

The two matrices have $I-1$ rows and $I+1$ columns. This is a representation of $I-1$ equations in $I+1$ unknowns. Our aim is to write this as

$$\mathbf{M}_L^{k+1}\mathbf{v}^{k+1} + \mathbf{r}^k = \mathbf{M}_R^k\mathbf{v}^k,$$

for known *square* matrices $\mathbf{M}_L^{k+1}$ and $\mathbf{M}_R^k$, and a known vector $\mathbf{r}^k$ and where details of the boundary conditions have been fully incorporated.

Example:

Sometimes we know that our option has a particular value on the boundary $i = 0$, or on $i = I$.

For example, if we have a European put we know that $V(0, t) = Ee^{-r(T-t)}$.

This translates to knowing that $V_0^{k+1} = Ee^{-r(k+1)\delta t}$.

$S = 0 \qquad \frac{\partial V}{\partial t} - rV = 0$

We can write

$$\begin{pmatrix} -A_1^{k+1} & 1-B_1^{k+1} & -C_1^{k+1} & 0 & . & . & . \\ 0 & -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} & 0 \\ . & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} & -C_{I-1}^{k+1} \end{pmatrix} \begin{pmatrix} V_0^{k+1} \\ V_1^{k+1} \\ . \\ . \\ . \\ V_{I-1}^{k+1} \\ V_I^{k+1} \end{pmatrix}$$

as

$$\begin{pmatrix} 1-B_1^{k+1} & -C_1^{k+1} & 0 & . & . & \\ -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & \\ & 0 & . & . & . & 0 \\ & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} \\ & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} \end{pmatrix} \begin{pmatrix} V_1^{k+1} \\ . \\ . \\ . \\ \\ V_{I-1}^{k+1} \end{pmatrix} + \begin{pmatrix} A_1^{k+1}V_0^{k+1} \\ 0 \\ 0 \\ . \\ . \\ 0 \\ . \end{pmatrix}$$

$$= \mathbf{M}_L^{k+1}\mathbf{v}^{k+1} + \mathbf{r}^k.$$

The matrix $\mathbf{M}_L$ is square and of size $I-1$.

The matrix equation

Remembering that we know $\mathbf{v}^k$, the matrix multiplication and vector addition on the right-hand side is simple enough to do.

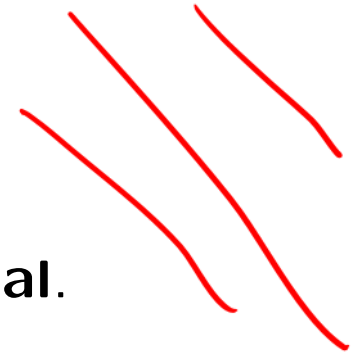
But how do we then find $\mathbf{v}^{k+1}$? In principle, the matrix $\mathbf{M}_L^{k+1}$ could be inverted to give

$$\mathbf{v}^{k+1} = (\mathbf{M}_L^{k+1})^{-1}(\mathbf{M}_R^k \mathbf{v}^k - \mathbf{r}^k),$$

except that matrix inversion is very time consuming, and from a computational point of view extremely inefficient.

There are two much better ways for solving this system, called **LU decomposition** and **successive over-relaxation**.

LU decomposition



The matrix $\mathbf{M}_L^{k+1}$ is special in that it is **tridiagonal**.

We can decompose the matrix into the product of two other matrices, one having non-zero elements along the diagonal and the subdiagonal and the other having non-zero elements along the diagonal and the superdiagonal.

Call these two matrices **L** and **U** respectively.

$$M$$

$$\begin{pmatrix} 1+B_1 & C_1 & 0 & . & . & . & 0 \\ A_2 & 1+B_2 & C_2 & 0 & . & . & . \\ 0 & A_3 & 1+B_3 & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1+B_{I-3} & C_{I-3} & 0 \\ . & . & . & 0 & A_{I-2} & 1+B_{I-2} & C_{I-2} \\ . & . & . & . & 0 & A_{I-1} & 1+B_{I-1} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & . & . & . & 0 \\ l_2 & 1 & 0 & 0 & . & . & . \\ 0 & l_3 & 1 & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1 & 0 & 0 \\ . & . & . & 0 & l_{I-2} & 1 & 0 \\ . & . & . & . & 0 & l_{I-1} & 1 \end{pmatrix} \begin{pmatrix} d_1 & u_1 & 0 & . & . & . & 0 \\ 0 & d_2 & u_2 & 0 & . & . & . \\ 0 & 0 & d_3 & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & d_{I-3} & u_{I-3} & 0 \\ . & . & . & 0 & 0 & d_{I-2} & u_{I-2} \\ . & . & . & . & 0 & 0 & d_{I-1} \end{pmatrix}$$

i.e.

$$L \qquad\qquad U$$

$$\mathbf{M = LU}$$

Doolittle's Method

Crout's Method

It is not difficult to show that

$$d_1 = 1 + B_1$$

and then

$$l_i d_{i-1} = A_i, \quad u_{i-1} = C_{i-1} \quad \text{and} \quad d_i = 1 + B_i - l_i u_{i-1} \quad \text{for} \quad 2 \leq i \leq I-1.$$

Now we exploit the decomposition to solve the original matrix equation. This equation is of the form

$$\mathbf{Mv} = \mathbf{q}$$

where we want to find $\mathbf{v}$.

We can write

$$\mathbf{LUv} = \mathbf{q}. \tag{2}$$

The vector $\mathbf{q}$ contains both the old option value array, at time step k, and details of the boundary conditions.

Solve Equation (2) in two steps. First find $\mathbf{w}$ such that

$$\mathbf{L}\mathbf{w} = \mathbf{q}$$

and then $\mathbf{v}$ such that

$$\mathbf{U}\mathbf{v} = \mathbf{w}.$$

And then we are done.

The first step gives

$$w_1 = q_1$$

and

$$w_i = q_i - l_i w_{i-1} \quad \text{for} \quad 2 \leq i \leq I-1.$$

Finally

$$v_{I-1} = \frac{w_{I-1}}{d_{I-1}}$$

and

$$v_i = \frac{w_i - u_i v_{i+1}}{d_i} \quad \text{for} \quad I-2 \geq i \geq 1.$$

The advantages of the LU decomposition method

- It is quick

- The decomposition need only be done once if the matrix **M** is independent of time

The disadvantages of the LU decomposition method

- It is not immediately applicable to American options

- The decomposition must be done each time step if the matrix **M** is time dependent

Iteration

Successive over-relaxation, SOR

We now come to an example of an 'indirect method.'

In this method we solve the equations iteratively.

Suppose that the matrix **M** in the matrix equation

$$\mathbf{Mv} = \mathbf{q}$$

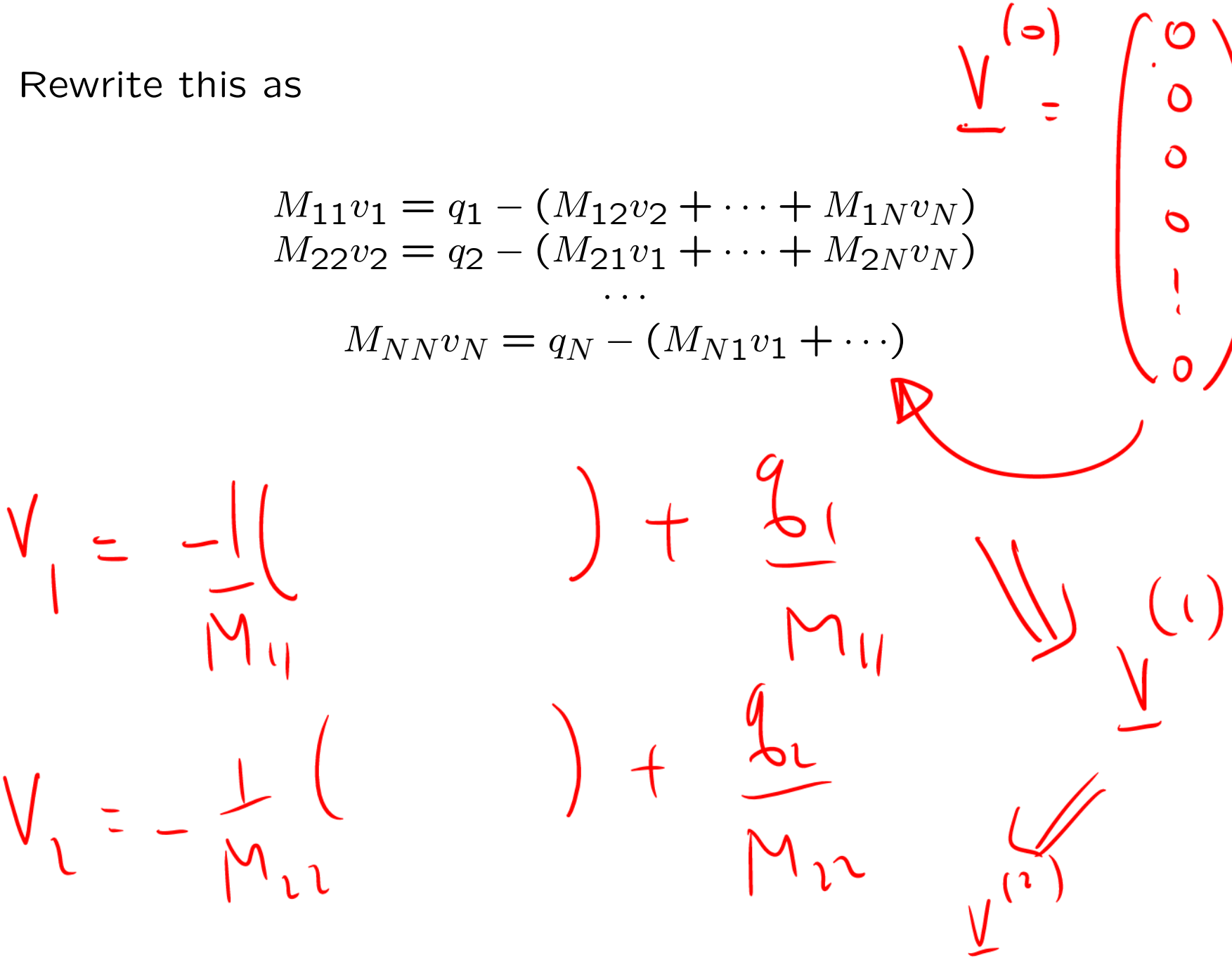
has entries M_{ij} then the system of equations can be written as

$$\begin{aligned} M_{11}v_1 + M_{12}v_2 + \cdots + M_{1N}v_N &= q_1 \\ M_{21}v_1 + M_{22}v_2 + \cdots + M_{2N}v_N &= q_2 \\ &\cdots \\ M_{N1}v_1 + M_{N2}v_2 + \cdots + M_{NN}v_N &= q_N \end{aligned}$$

where N is the number of equations, the size of the matrix.

Rewrite this as

$$
\begin{aligned}
M_{11}v_1 &= q_1 - (M_{12}v_2 + \cdots + M_{1N}v_N) \\
M_{22}v_2 &= q_2 - (M_{21}v_1 + \cdots + M_{2N}v_N) \\
&\cdots \\
M_{NN}v_N &= q_N - (M_{N1}v_1 + \cdots)
\end{aligned}
$$



This system is easily solved *iteratively* using

$$
\begin{aligned}
v_1^{n+1} &= \frac{1}{M_{11}}\left(q_1 - (M_{12}v_2^n + \cdots + M_{1N}v_N^n)\right)\\
v_2^{n+1} &= \frac{1}{M_{22}}\left(q_2 - (M_{21}v_1^n + \cdots + M_{2N}v_N^n)\right)\\
&\cdots\\
v_N^{n+1} &= \frac{1}{M_{NN}}\left(q_N - (M_{N1}v_1^n + \cdots)\right)
\end{aligned}
$$

where the superscript n denotes the level of the iteration *and not the time step*.

This iteration is started from some initial guess $\mathbf{v}^0$.

This iterative method is called the **Jacobi method**.

When we implement the Jacobi method in practice we find some of the values of v_i^{n+1} before others.

In the **Gauss–Seidel** method we use the updated values as soon as they are calculated.

This method can be written as

$$v_i^{n+1} = \frac{1}{M_{ii}} \left(q_i - \sum_{j=1}^{i-1} M_{ij} v_j^{n+1} - \sum_{j=i}^{N} M_{ij} v_j^n \right).$$

Observe that there are some terms on the right-hand side with the superscript $n+1$.

These are values of v that were calculated earlier but at the same level of iteration.

When the matrix **M** has come from a finite-difference discretization of a parabolic equation (and that includes almost all finance problems) the above iterative methods usually converge to the correct solution *from one side*.

This means that the corrections $v_i^{n+1} - v_i^n$ stay of the same sign as n increases.

This is exploited in the **successive over-relaxation** or **SOR** method to speed up convergence.

The method can be written as

$$v_i^{n+1} = v_i^n + \frac{\omega}{M_{ii}} \left(q_i - \sum_{j=1}^{i-1} M_{ij} v_j^{n+1} - \sum_{j=i}^{N} M_{ij} v_j^n \right).$$

Again, the new values for v_i are used as soon as they are obtained.

But now the factor ω, called the **acceleration** or **over-relaxation parameter** is included.

This parameter, which must lie between 1 and 2, speeds up the convergence to the true solution.

① Makes an otherwise non-convergent system, under relaxed $0 < \omega < 1$

② where cgce is too slow, we accelerate cgce. Over relaxed $1 < \omega < 2$

The advantages of the SOR method

- It is easier to program than LU decomposition
- It is easily applied to American options

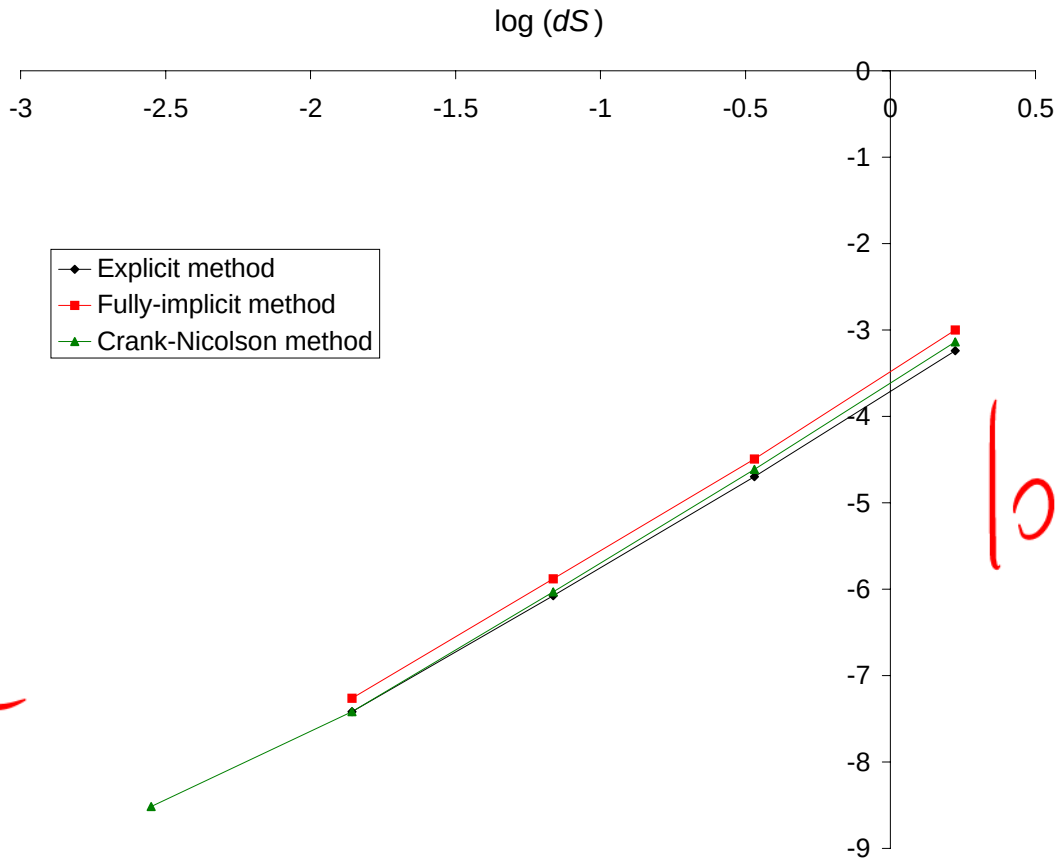
The disadvantage of the SOR method

- It is slightly slower than LU decomposition for European options

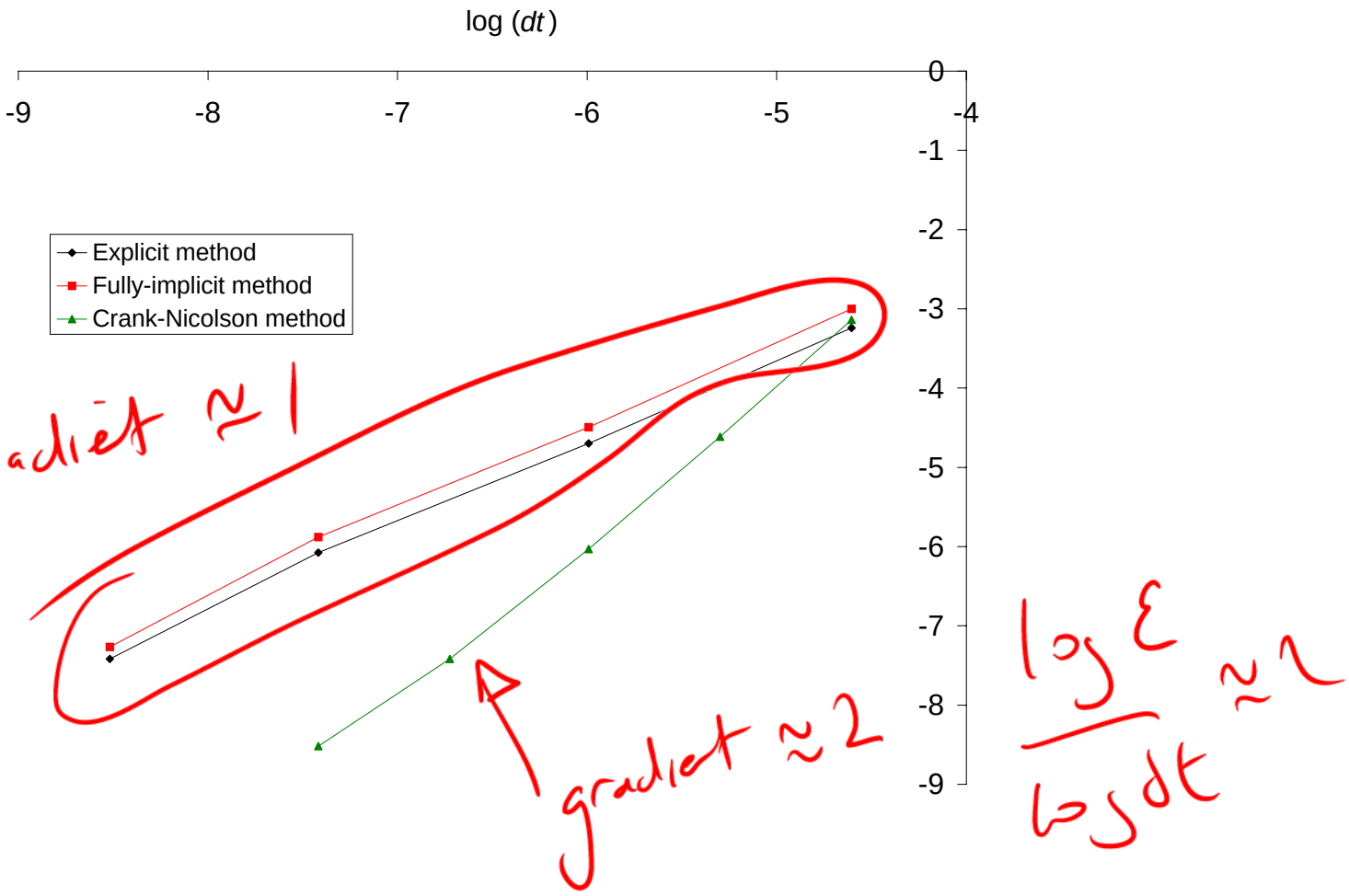
Comparison of finite-difference methods

Example: An at-the-money European call option, strike 20, three months to expiry, a volatility of 20% and an interest rate of 5%.

c.f. Black Scholes formula



The logarithm of the error against the logarithm of the asset step size.



The logarithm of the error against the logarithm of the time step.

$\Rightarrow \varepsilon \approx dt^2$

Other methods

The questions that arise in any method are

- What is the error in the method in terms of δt and δS? Accuracy
- What are the restrictions on the time step and/or asset step? Stability
- Can we solve the resulting difference equations quickly? Comp. efficiency
- Is the method flexible enough to cope with changes in coefficients, boundary conditions etc.? flexibility to price other contracts

Douglas schemes

This is a method that manages to have a local truncation error of $O(\delta S^4, \delta t^2)$ for the same computational effort as the Crank–Nicolson scheme.

The basic diffusion equation is

$$\frac{\partial V}{\partial t} + \frac{\partial^2 V}{\partial S^2} = 0.$$

The explicit method applied to this equation is just

$$\frac{V_i^{k+1} - V_i^k}{\delta t} = \frac{V_{i+1}^k - 2V_i^k + V_{i-1}^k}{\delta S^2}.$$

and the fully implicit is similarly

$$\frac{V_i^{k+1} - V_i^k}{\delta t} = \frac{V_{i+1}^{k+1} - 2V_i^{k+1} + V_{i-1}^{k+1}}{\delta S^2}.$$

The Crank–Nicolson scheme is an average of these two methods. What about a *weighted* average?

This leads to the θ method.

Take a weighted average of the explicit and implicit methods to get...

$$\frac{V_i^{k+1}-V_i^k}{\delta t}=\theta\left(\frac{V_{i+1}^{k+1}-2V_i^{k+1}+V_{i-1}^{k+1}}{\delta S^2}\right)+(1-\theta)\left(\frac{V_{i+1}^{k}-2V_i^{k}+V_{i-1}^{k}}{\delta S^2}\right).$$

When $\theta=\frac{1}{2}$ we are back to the Crank–Nicolson method.

For a general value of θ the local truncation error is

$$O\left(\tfrac{1}{2}\delta t+\tfrac{1}{12}\delta S^2-\theta\delta t,\delta S^4,\delta t^2\right).$$

When $\theta=0$, $\frac{1}{2}$ or 1 we get the results we have seen so far.

But if

$$\theta = \frac{1}{2} - \frac{\delta S^2}{12\,\delta t}$$

then the local truncation error is improved.

The implementation of the method is no harder than the Crank–Nicolson scheme.

Three time-level methods

Numerical schemes are not restricted to the use of just two time levels.

We can construct many algorithms using three or more time levels.

Again, we would do this if it gave us a better local truncation error or had better convergence properties.

Richardson extrapolation

In the explicit method the error is $O(\delta t, \delta S^2)$.

If we assume that the approach to the correct solution as the time step and asset step tend to zero is in a sense 'regular' then we could postulate that

$$\text{approximate solution} = \text{exact solution} + \epsilon_1 \delta t + \epsilon_2 \delta S^2 + \epsilon_3 \delta t^2 + \cdots$$

for some coefficients ϵ_i.

Suppose that we have *two* approximate solution V_1 and V_2 using different grid sizes, we can write

$$V_1 = \text{ exact solution } + \epsilon_1\,\delta t_1 + \epsilon_2\,\delta S_1^2 + \epsilon_3\,\delta t_1^2 + \cdots$$

$$= \text{ exact solution } + \delta S_1^2\left(\epsilon_1\frac{\delta t_1}{\delta S_1^2} + \epsilon_2\right) + \cdots$$

and

$$V_2 = \text{ exact solution } + \epsilon_1\,\delta t_2 + \epsilon_2\,\delta S_2^2 + \epsilon_3\,\delta t_2^2 + \cdots$$

$$= \text{ exact solution } + \delta S_2^2\left(\epsilon_1\frac{\delta t_2}{\delta S_2^2} + \epsilon_2\right) + \cdots.$$

If we choose

$$\frac{\delta t_1}{\delta S_1^2} = \frac{\delta t_2}{\delta S_2^2},$$

i.e. ν_1 constant, where the subscripts denote the step sizes used in finding the solutions V_1 and V_2, then we can find a *better* solution than both V_1 and V_2 by eliminating the leading-order error terms in the above two equations.

This better approximation is given by

$$\frac{\delta S_2^2 V_1 - \delta S_1^2 V_2}{\delta S_2^2 - \delta S_1^2}.$$

The accuracy of the method is now $O(\delta t^2, \delta S^3)$.

Example: Call option with a strike of 100, expiry of one year, underlying asset at 100 with a volatility of 20% and an interest rate of 10%. The exact Black–Scholes value is 13.269.

Solve using an explicit finite-difference scheme.

Method	**BS**	**Numerical**	**Error**	**Time taken**
20 asset steps	13.269	13.067	0.202	8,000
30 asset steps	13.269	13.183	0.086	27,000
Richardson	13.269	13.275	0.006	**35,000**
100 asset steps	13.269	13.276	0.007	**1,000,000**

Free boundary problems and American options

The value of American options must always be greater than the payoff, otherwise there will be an arbitrage opportunity

$$V(S,t) \geq \text{Payoff}(S).$$

American options are examples of 'free boundary problems' we must solve a partial differential equation with an unknown boundary, the position of which is determined by having one more boundary condition than if the boundary were prescribed.

In the American option problem we know that both the option value and its delta are continuous with the payoff function, the so called **smooth pasting condition**.

Early exercise and the explicit method

Suppose that we have found V_i^k for all i at the time step k, time step to find the option value at $k+1$ using the finite-difference scheme

$$V_i^{k+1} = A_i^k V_{i-1}^k + (1 + B_i^k) V_i^k + C_i^k V_{i+1}^k.$$

Don't worry about whether or not you have violated the American option constraint until you have found the option values V_i^{k+1} for all i.

Now let's check whether the new option values are greater or less than the payoff.

If they are less than the payoff then we have arbitrage.

We can't allow that to happen so at every value of i for which the option value has allowed arbitrage, replace that value by the payoff at that asset value.

Early exercise and Crank–Nicolson

The Crank–Nicolson method is implicit, and every value of the option at the $k+1$ time step is linked to every other value at that time step.

It is therefore not good enough to just replace the option value with the payoff after the values have all been calculated, the replacement must be done at the same time as the values are found:

$$v_i^{n+1} = \max\left(v_i^n + \frac{\omega}{M_{ii}}\left(q_i - \sum_{j=1}^{i-1} M_{ij} v_j^{n+1} - \sum_{j=i}^{N} M_{ij} v_j^n\right), \text{Payoff}\right).$$

The payoff is evaluated at i and $k+1$ in the obvious manner. This method is called **projected SOR**.

Two-factor models

We are going to refer to the general two-factor equation

$$\frac{\partial V}{\partial t} + a(S,r,t)\frac{\partial^2 V}{\partial S^2} + b(S,r,t)\frac{\partial V}{\partial S} + c(S,r,t)V$$

$$+d(S,r,t)\frac{\partial^2 V}{\partial r^2} + e(S,r,t)\frac{\partial^2 V}{\partial S \partial r} + f(S,r,t)\frac{\partial V}{\partial r} = 0. \qquad (1)$$

It will be quite helpful if we think of solving the two-factor convertible bond problem.

For the general two-factor problem (1) to be parabolic we need

classification

$$e(S,r,t)^2 < 4a(S,r,t)\, d(S,r,t).$$

As in the one-factor world, the variables must be discretized.

That is, we solve on a three-dimensional grid with

$$S = i\,\delta S, \quad r = j\,\delta r, \quad \text{and} \quad t = T - k\,\delta t.$$

Expiry is $t = T$ or $k = 0$.

The indices range from zero to I and J for i and j respectively.

$V(r, S, t)$

We will assume that the interest rate model is only specified on $r \geq 0$.

This may not be the case, some simple interest rate models such as Vasicek, are defined over negative r as well.

The contract value is written as

$$V(S, r, t) = V_{ij}^{k}.$$

Whatever the problem to be solved, we must impose certain conditions on the solution.

First of all, we must specify the final condition.

This is the payoff function, telling us the value of the contract at the expiration of the contract.

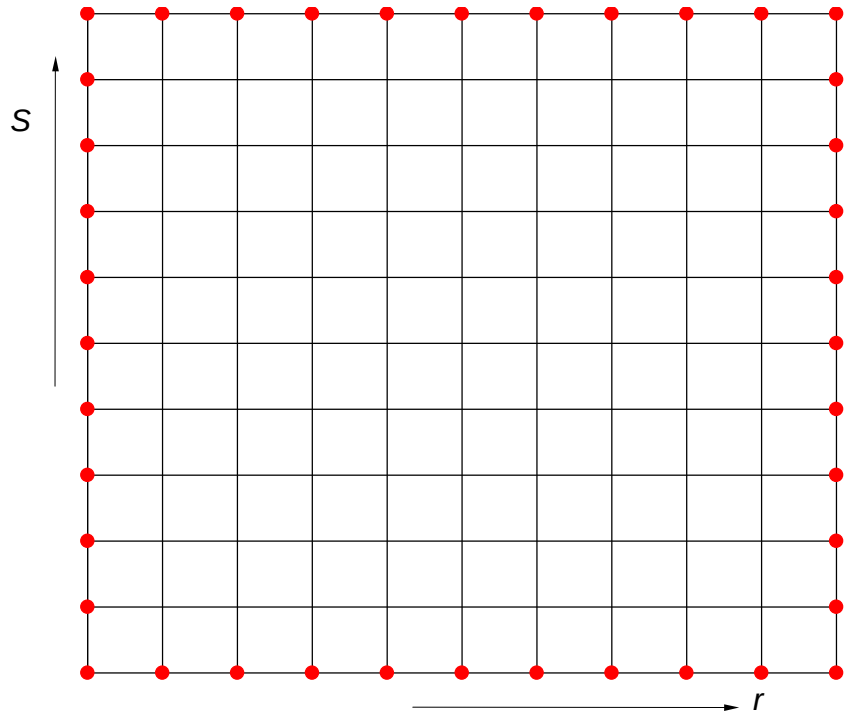
Suppose that we are pricing a long-dated warrant with a call payoff.

The final condition for this problem is then

$$V(S, r, T) = V_{ij}^0 = \max(S - E, 0).$$

Boundary conditions must be imposed at all the grid points marked with a dot. The boundary conditions will depend on the contract.

Remember that there is also a time axis coming out of the page, and not drawn in this figure.



The explicit method

The one-factor explicit method can be extended to two-factors with very little effort.

In fact, the ease of programming make it a very good method for those new to the subject.

We will use symmetric central differences for all derivatives in (1).

This is the best way to approximate the second derivatives but may not be the best for the first derivatives.

We have seen how to use central differences for all of the terms with the exception of the second derivative with respect to both S and r,

$$\frac{\partial}{\partial S}\left(\frac{\partial V}{\partial r}\right) = \frac{\partial^2 V}{\partial S \partial r} = \frac{\partial}{\partial r}\left(\frac{\partial V}{\partial S}\right).$$

We can approximate this by

$$\frac{\partial\left(\frac{\partial V}{\partial r}\right)}{\partial S} \approx \frac{\frac{\partial V}{\partial r}(S+\delta S, r, t) - \frac{\partial V}{\partial r}(S-\delta S, r, t)}{2\,\delta S}.$$

But

$$\frac{\partial V}{\partial r}(S+\delta S, r, t) \approx \frac{V^k_{i+1,j+1} - V^k_{i+1,j-1}}{2\,\delta r}.$$

This suggests that a suitable discretization might be

$$\frac{\frac{V^k_{i+1,j+1}-V^k_{i+1,j-1}}{2\,\delta r}-\frac{V^k_{i-1,j+1}-V^k_{i-1,j-1}}{2\,\delta r}}{2\,\delta S}$$

$$=\frac{V^k_{i+1,j+1}-V^k_{i+1,j-1}-V^k_{i-1,j+1}+V^k_{i-1,j-1}}{4\,\delta S\,\delta r}.$$

This is particularly good since, not only is the error of the same error as in the other derivative approximations but also it preserves the property that

$$\frac{\partial^2 V}{\partial S \partial r}=\frac{\partial^2 V}{\partial r \partial S}.$$

The resulting explicit difference scheme is

$$\frac{V_{ij}^k - V_{ij}^{k+1}}{\delta t} + a_{ij}^k \left(\frac{V_{i+1,j}^k - 2V_{ij}^k + V_{i-1,j}^k}{\delta S^2} \right)$$

$$+ b_{ij}^k \left(\frac{V_{i+1,j}^k - V_{i-1,j}^k}{2\,\delta S} \right) + c_{ij}^k V_{ij}^k$$

$$+ d_{ij}^k \left(\frac{V_{i,j+1}^k - 2V_{ij}^k + V_{i,j-1}^k}{\delta r^2} \right)$$

$$e_{ij}^k \left(\frac{V_{i+1,j+1}^k - V_{i+1,j-1}^k - V_{i-1,j+1}^k + V_{i-1,j-1}^k}{4\,\delta S\,\delta r} \right)$$

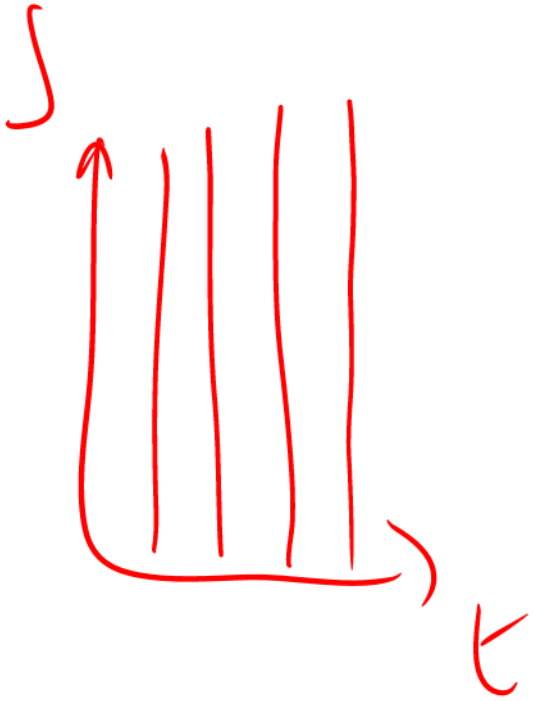
$$+ f_{ij}^k \left(\frac{V_{i,j+1}^k - V_{i,j-1}^k}{2\,\delta r} \right) = O(\delta t, \delta S^2, \delta r^2).$$

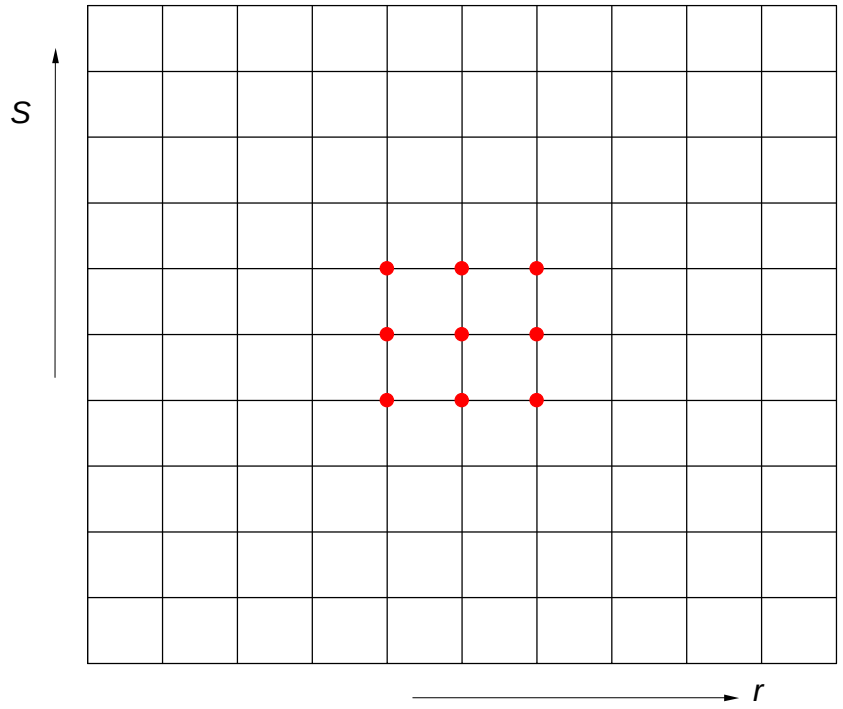
mixed partial deriv

We could rewrite this in the form

$$V_{ij}^{k+1} = \ldots,$$

where the right-hand side is a linear function of the nine option values shown schematically below.





The coefficients of these nine values at time step k are related to a, b etc.

It would not be very helpful to write the difference equation in this form, since the actual implementation is usually more transparent than this.

Note that in general all nine points (i, j), $(i \pm 1, j)$, $(i, j \pm 1)$, $(i \pm 1, j \pm 1)$ are used in the scheme.

If there is no cross derivative term then only the five points (i, j), $(i \pm 1, j \pm 1)$ are used.

This simplifies some of the methods.