

# Further Finite Difference Methods

## In this lecture

- Fully implicit method
- Crank-Nicolson method
- Numerical Linear Algebra

Further FDM  
Implicit schemes

\* 2 step methods (time)

- Douglas Schemes

• 2 factor model

By the end of this lecture you will

- know several more ways of solving parabolic partial differential equations numerically  $\Rightarrow$  more work.

## Introduction


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

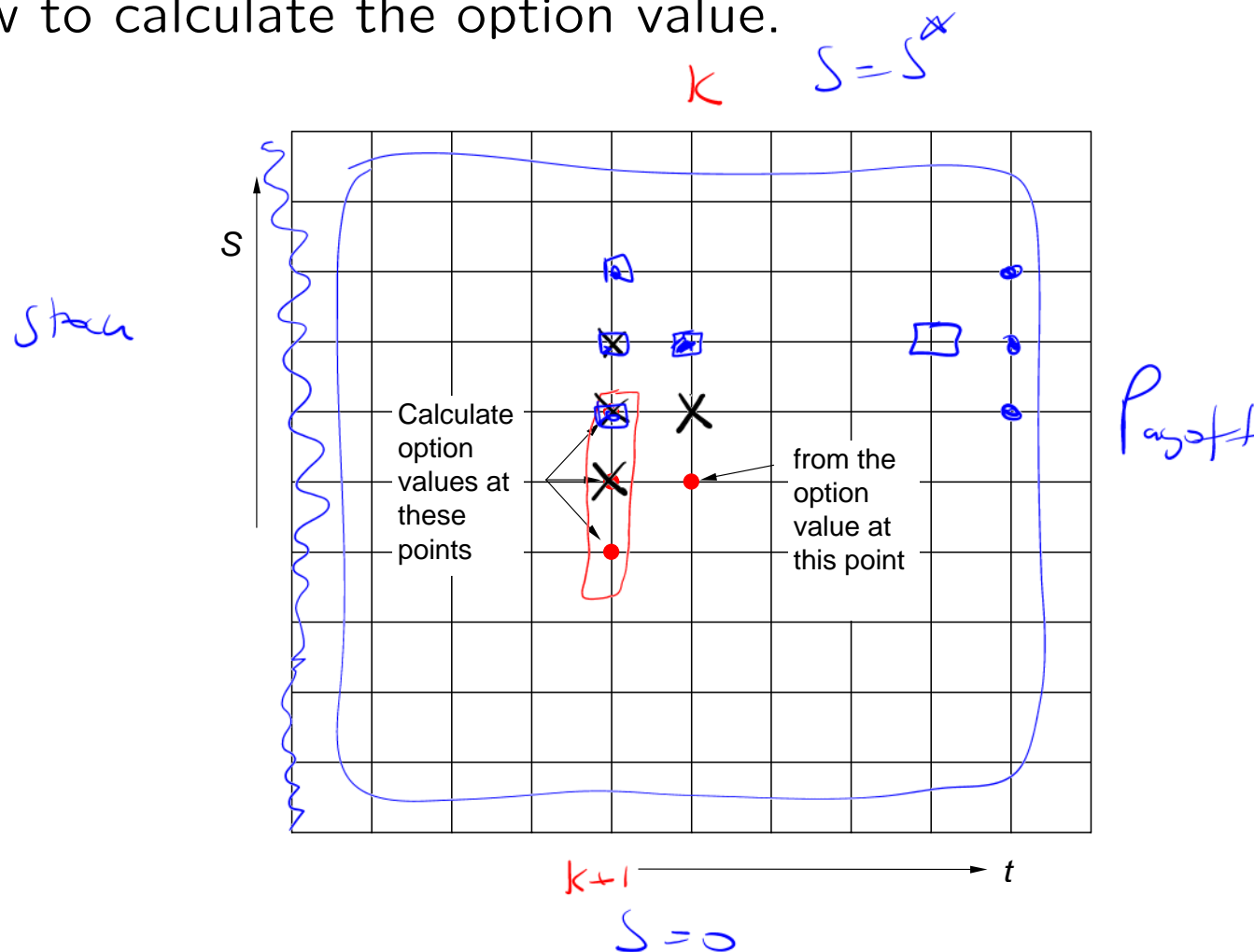
The more advanced methods are usually more complicated to program but have advantages in terms of stability and speed.

Explicit FDM

today we go further.

## 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

only known option price

$$\begin{aligned}
 & \frac{\partial V}{\partial t} \quad \frac{1}{2} \sigma^2 \frac{\partial^2 V}{\partial S^2} \quad \frac{\partial V}{\partial S} \quad O(\delta S^2) \\
 & \left( V_i^k - V_i^{k+1} \right) / \delta t + a_i^{k+1} \left( \frac{V_{i+1}^{k+1} - 2V_i^{k+1} + V_{i-1}^{k+1}}{\delta S^2} \right) \\
 & + b_i^{k+1} \left( \frac{V_{i+1}^{k+1} - V_{i-1}^{k+1}}{2\delta S} \right) + c_i^{k+1} V_i^{k+1} = 0.
 \end{aligned}$$

$(r-D) \delta S$   $\frac{\partial V}{\partial S} \quad O(\delta S)$   $V$

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

$$\begin{aligned}
 \frac{\partial V}{\partial t} & \approx \frac{V_i^k - V_i^{k+1}}{\delta t} \quad O(\delta t) \\
 \frac{\partial V}{\partial S} & \approx \left( \right) \underline{V_{i-1}^{k+1}} + \left( \right) \underline{V_i^{k+1}} + \left( \right) \underline{V_{i+1}^{k+1}} = \underline{V_i^k}
 \end{aligned}$$

$\omega_{eff}$   $\omega_{eff}$   $\omega_{eff}$

This can be written as

$i = 1, \dots, I-1$

3 unknown option price

known

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

where

In each eq<sup>n</sup>  $\exists$  3 unknowns

$$A_i^{k+1} = -\nu_1 a_i^{k+1} + \frac{1}{2} \nu_2 b_i^{k+1},$$

$$B_i^{k+1} = 2\nu_1 a_i^{k+1} - \delta t c_i^{k+1}$$

and

$$C_i^{k+1} = -\nu_1 a_i^{k+1} - \frac{1}{2} \nu_2 b_i^{k+1}$$

where

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

$i=0 \quad S=0 \quad B.C$

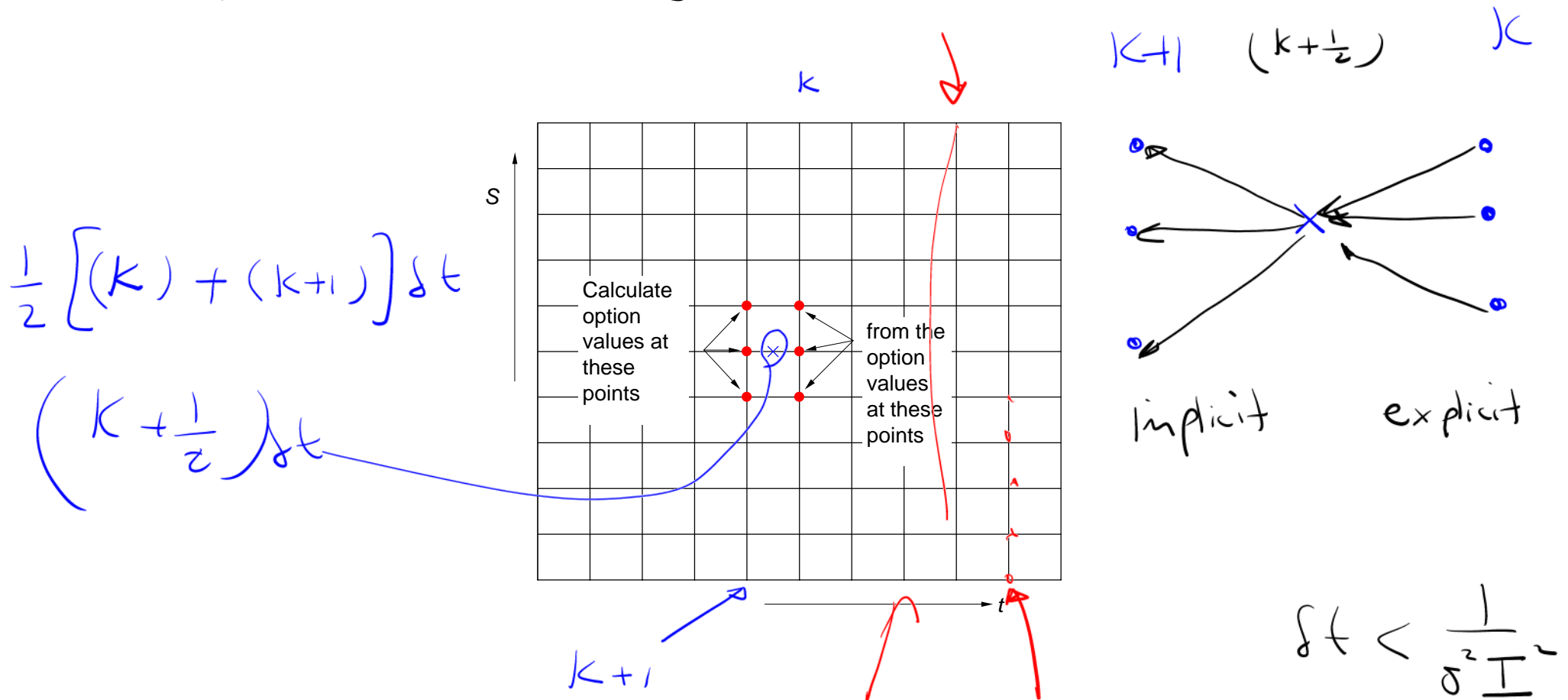
$i=I \quad S=S^* \quad B.C$

$$F(V_{i-1}^{k+1}, V_i^{k+1}, V_{i+1}^{k+1}) = V_i^k$$

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

$$\begin{aligned}
 & \frac{V_i^k - V_i^{k+1}}{\delta t} + \frac{a_i^{k+1}}{2} \left( \frac{V_{i+1}^{k+1} - 2V_i^{k+1} + V_{i-1}^{k+1}}{\delta S^2} \right) + \frac{a_i^k}{2} \left( \frac{V_{i+1}^k - 2V_i^k + V_{i-1}^k}{\delta S^2} \right) \\
 & + \frac{b_i^{k+1}}{2} \left( \frac{V_{i+1}^{k+1} - V_{i-1}^{k+1}}{2 \delta S} \right) + \frac{b_i^k}{2} \left( \frac{V_{i+1}^k - V_{i-1}^k}{2 \delta S} \right) \\
 & + \frac{1}{2} c_i^{k+1} V_i^{k+1} + \frac{1}{2} c_i^k V_i^k = 0 = O(\delta t^2, \delta S^2).
 \end{aligned}$$

Same as  
with  
Explicit &  
fully implicit

option  
price  
imp.

option  
price  
exp.

extra order  
of accuracy in

$$F(V_{i-1}^{k+1}, V_i^{k+1}, V_{i+1}^{k+1}) = G(V_{i-1}^k, V_i^k, V_{i+1}^k)$$



This can be written as

Another linear system

$$\begin{aligned}
 & F(V_{i-1}^{k+1}, V_i^{k+1}, V_{i+1}^{k+1}) \\
 & -A_i^{k+1} V_{i-1}^{k+1} + (1 - B_i^{k+1}) V_i^{k+1} - C_i^{k+1} V_{i+1}^{k+1} \\
 & = A_i^k V_{i-1}^k + (1 + B_i^k) V_i^k + C_i^k V_{i+1}^k,
 \end{aligned}$$

$G(V_{i-1}^k, V_i^k, V_{i+1}^k)$   
 known

S  
 T

where

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

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

$\swarrow$   
 L.T.E (error analysis)  $\pm \delta S$   
 $+ \delta t$

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

L.H.S

$S=0 \rightarrow$

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

$S=S^* \rightarrow$

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

Matrix \* vector  
↓  
vector

To

Complete the matrix, required:

i) Row 0  $S=0$  B.C. at bottom

ii) Row I  $S=S^*$  B.C. along top boundary

Matrix inversion problem:  $A \underline{V}^{k+1} = \underline{V}^k$

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

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

extra col<sup>l</sup> vector  
containing  
B.C.s

(X)

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.

Numerical Analysis

Burden and Faires

$\underline{\underline{\mathbf{V}}}^{k+1} =$

**Example:** *Seen last week*

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  $V_0^{k+1} = Ee^{-r(k+1)\delta t}$ .

We can write

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

as

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

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

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

## The matrix equation

$$\leftarrow \underline{A} \underline{x} = \underline{b} \quad \underline{x} = \underline{A}^{-1} \underline{b}$$

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

rearranging (\*) on slide 11

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

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

Never calculate the inverse of an  
arb. large matrix

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

Numerical Linear Algebra

Direct (once)

Gaussian  $\mathcal{O}(n^3)$  (not encouraged)

LU Decomp.

Indirect (iteration)

Jacobi }  
Gauss-Seidel } SOR

## LU decomposition / factorisation



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.

$$\mathbf{M} = \mathbf{L} \mathbf{U}$$



Note: **L** and **U**  
are not  
unique



$$\underline{x}^T A \underline{x} > 0$$

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

$d_i$  leading diag.  
 $u_i$  upper diag.

$l_i$  sub diag.  
T.B.D.  
index i.e.

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

$L$   $V=$   $U$   $d_i, u_i$  unknown

$$M = LU$$

Doolittle's Method

read off corresponding matrix entries

CROUT's Method

$L$  with a non-unit diag.  
 $U$  - unit diagonal

CROUT

$$\sum_2 d_i = A_i$$

It is not difficult to show that

$$d_1 = 1 + B_1$$

and then *the resulting algorithm is*

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

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

$$\mathbf{M}\mathbf{v} = \mathbf{q}$$

but  $\mathbf{M} = \mathbf{L} \times \mathbf{U}$

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

We can write

$\rightarrow$  call it  $\underline{w}$  Problem 1:  $\mathbf{U}\underline{v} = \underline{w}$   
 $\rightarrow$   $\mathbf{L}(\mathbf{U}\mathbf{v}) = \mathbf{q}$ . Problem 2:  $\mathbf{L}\underline{w} = \underline{q}$   
Solve Problem (2) (2)  
first  $\because \underline{q}$  is known  $\rightarrow \underline{w}$

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

Next solve problem 1:  $\mathbf{U}\underline{v} = \underline{w}$   
 $\because \underline{w}$  is known

and we obtain  $\underline{v}$   
 vector of option prices

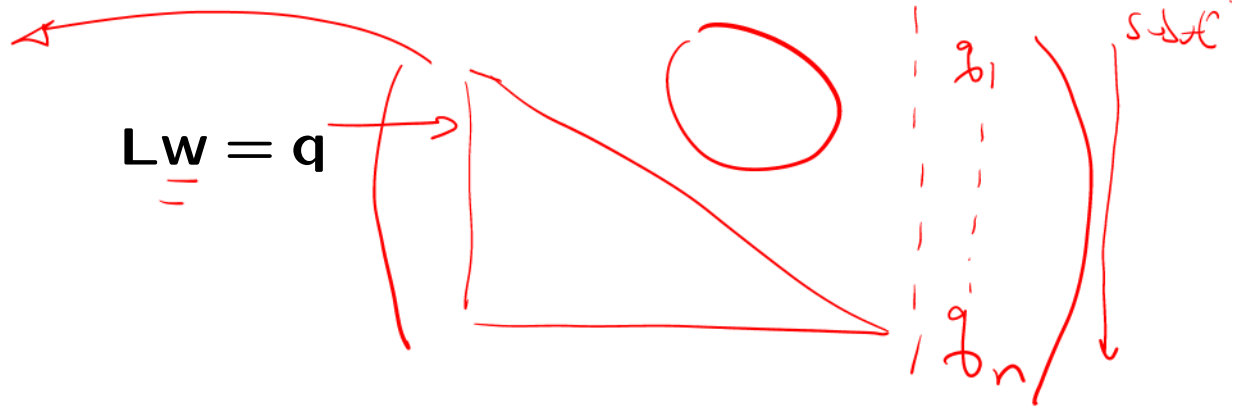
Solve Equation (2) in two steps. First find  $\mathbf{w}$  such that

$$L_1 w_1 = q_1$$

$$\rightarrow w_1$$

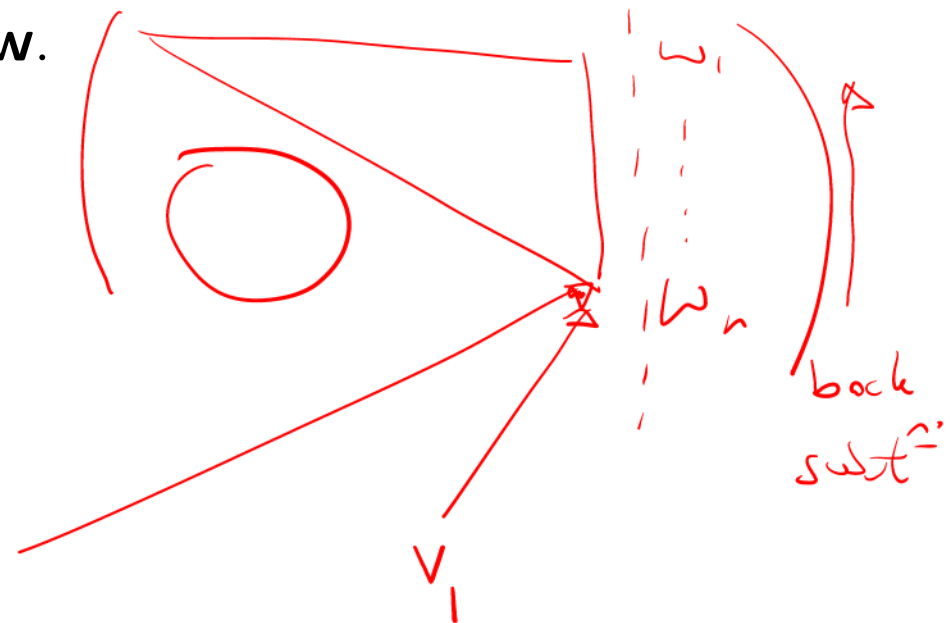
$$L_2$$

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



and then  $\mathbf{v}$  such that

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



And then we are done.

$$\alpha x = 5$$

$$x = \frac{5}{\alpha}$$

The first step gives

$$\underline{Lw = q}$$

$$\rightarrow w_1 = q_1$$

forward substitution

and

algorithm :

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

Finally

$$\underline{Uv = w}$$

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

backward substitution

and

algorithm

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

## Successive over-relaxation, SOR

Indirect Method

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

In this method we solve the equations iteratively.

Key: For iteration we need a starting  
"solution"

Convergence criteria / acceptable tolerance

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

$$\mathbf{M}\mathbf{v} = \mathbf{q}$$

Coeff. matrix  $\leftarrow$   $\rightarrow$  Known vector  
 vector of unknown option prices  $\leftarrow$   $\rightarrow$   $N$  eq's,  $N$  unknowns

has entries  $M_{ij}$  then the system of equations can be written as  
 (in component form) i.e. line by line

$$\begin{array}{l}
 E_1 \\
 E_2 \\
 \vdots \\
 E_N
 \end{array}
 \begin{array}{l}
 M_{11}v_1 + M_{12}v_2 + \cdots + M_{1N}v_N = q_1 \\
 M_{21}v_1 + M_{22}v_2 + \cdots + M_{2N}v_N = q_2 \\
 \vdots \\
 M_{N1}v_1 + M_{N2}v_2 + \cdots + M_{NN}v_N = q_N
 \end{array}$$

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

From eq<sup>n</sup>  $i$  rearrange to get  $v_i$  on its own

Rewrite this as

$$M_{11}v_1 = q_1 - (M_{12}v_2 + \cdots + M_{1N}v_N)$$

$$M_{22}v_2 = q_2 - (M_{21}v_1 + \cdots + M_{2N}v_N)$$

...

$$M_{NN}v_N = q_N - (M_{N1}v_1 + \cdots)$$

Need an  
initial guess

$$V_1 = \frac{q_1}{M_{11}} - \frac{1}{M_{11}} ( \begin{matrix} v_2, v_3, \dots, v_N \\ \text{---} \end{matrix} )$$

$$V_2 = \frac{q_2}{M_{22}} - \frac{1}{M_{22}} ( \begin{matrix} v_1, v_3, v_4, \dots, v_N \\ \text{---} \end{matrix} )$$

$$V_N = \frac{q_N}{M_{NN}} - \frac{1}{M_{NN}} ( \begin{matrix} v_1, v_2, \dots, v_{N-1} \\ \text{---} \end{matrix} )$$



This system is easily solved *iteratively* using

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \underline{v}^{(0)}$$

$$v_1^{n+1} = \frac{1}{M_{11}} \left( q_1 - (M_{12}v_2^n + \dots + M_{1N}v_N^n) \right)$$

$$v_2^{n+1} = \frac{1}{M_{22}} \left( q_2 - (M_{21}v_1^n + \dots + M_{2N}v_N^n) \right)$$

$$\vdots$$

$$v_N^{n+1} = \frac{1}{M_{NN}} \left( q_N - (M_{N1}v_1^n + \dots) \right)$$

*Handwritten notes:*   
 - "it. number" points to  $n$  in the superscript.   
 - "previous, self" points to  $v_i^n$  in the equations.   
 -  $n = 0, 1, \dots, \infty$    
 -  $\underline{v}^{(1)}, \underline{v}^{(2)}, \underline{v}^{(3)}, \dots, \underline{v}^{(n+1)}$    
 -  $v_1^{(n+1)}, v_2^{(n+1)}, \dots, v_N^{(n+1)}$

where the superscript  $n$  denotes the level of the iteration *and not the time step*.  $l_\infty : \max_{1 \leq i \leq N} |v_i|$

This iteration is started from some initial guess  $\underline{v}^0$ . *e.g.*  $\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$  or  $\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}^T$    
 $p=1, 2, \dots, \infty$   $p=2$  Euclidean norm

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

*Vector Norms*  $\underline{v} = (a, b, c) \in \mathbb{R}^3$   $|\underline{v}| = \sqrt{a^2 + b^2 + c^2}$

$\|\cdot\| : \mathbb{R}^N \rightarrow \mathbb{R}$   $\|\underline{v}\|_p = \left( \sum_{i=1}^N |v_i|^p \right)^{1/p}$   $p=2 : p\text{-th power}$   $l_2 \text{ norm}$

$\underline{v} \in \mathbb{R}^N$

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 *latest value, from current iteration, option price, for previous iteration*

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

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.

Convergence  
Criteria

$$\frac{\|x^{(k+1)} - x^{(k)}\|_{\infty}}{\|x^{(k)}\|_{\infty}} < \varepsilon$$

$\varepsilon$  - tolerance

a popular method  
to ascertain  
convergence.

The method can be written as  
acceleration factor

Indep.  $\rightarrow$  Spectral radius  
study

S.D.D

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

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

But now the factor  $\omega$ , 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 System does not cge (under-relaxed)  $0 < \omega < 1$

② System converges slowly (over-relaxed)  $1 < \omega < 2$

## Other methods : Numerical Method consideration

The questions that arise in any method are

- What is the error in the method in terms of  $\delta t$  and  $\delta S$ ?  
*accuracy*
- What are the restrictions on the time step and/or asset step?  
*\* stability \**
- Can we solve the resulting difference equations quickly?  
*computational efficiency.*
- Is the method flexible enough to cope with changes in coef-  
ficients, boundary conditions etc.?  
*flexibility*

→ Fully Implicit scheme unconditionally stable ✓

## Douglas schemes

$\Theta$ -Method

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

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

The explicit method applied to this equation is just

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

and the fully implicit is similarly

$$\frac{V_i^{k+1} - V_i^k}{\delta t} = \frac{V_{i+1}^{k+1} - 2V_i^{k+1} + V_{i-1}^{k+1}}{\delta S^2}. \quad \checkmark \text{ fully imp.}$$

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

This leads to the  $\theta$  method.

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



$$0 \leq \theta \leq 1$$

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

Imp. Exp.

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

For a general value of  $\theta$  the local truncation error is

$$O\left(\underbrace{\frac{1}{2}\delta t + \frac{1}{12}\delta S^2}_{\text{Exp.}} - \theta\delta t, \boxed{\delta S^4}, \boxed{\delta t^2}\right).$$

↖ ↗

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

↙   ↓   ↘  
Exp.   C-N   F. Implicit

## Three time-level methods

C.F.D

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

Upwinding

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.

Thus for  $k, k+1$ . Also have  $k, k-1$ . 2 time-level method

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

unstable  
write  
 $-2V_i^k = -V_i^{k+1} - V_i^{k-1}$

This is now stable  $\forall \delta t$ .

## Two-factor models

$$dr = (\gamma - \delta r)dt + \sqrt{\alpha r} dX_2$$

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

$\epsilon_X:$

$$a = \frac{1}{2} \sigma^2 S^2$$

$$b = (r - D)S$$

$$c = -r$$

$$d = \frac{1}{2} \alpha r$$

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

$$F = m a$$

$$F = \frac{1}{\gamma} m a$$

$$+ d(S, r, t) \frac{\partial^2 V}{\partial r^2} + e(S, r, t) \frac{\partial^2 V}{\partial S \partial r} + f(S, r, t) \frac{\partial V}{\partial r} = 0. \quad (3)$$

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

$$e = \sigma S \sqrt{\alpha r}$$

For the general two-factor problem (3) to be parabolic we need

$$f = (\gamma - \delta r)$$

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

C.I.R.

$$V(r, S, t)$$

GBM

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.

$$V(r, S, t) := V_{ij} \dots$$

*K Time is always kept on its own as a superscript.*

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

→  $V(S, r, t) = V_{ij}^k.$

$i \ j \ k \ \checkmark$

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

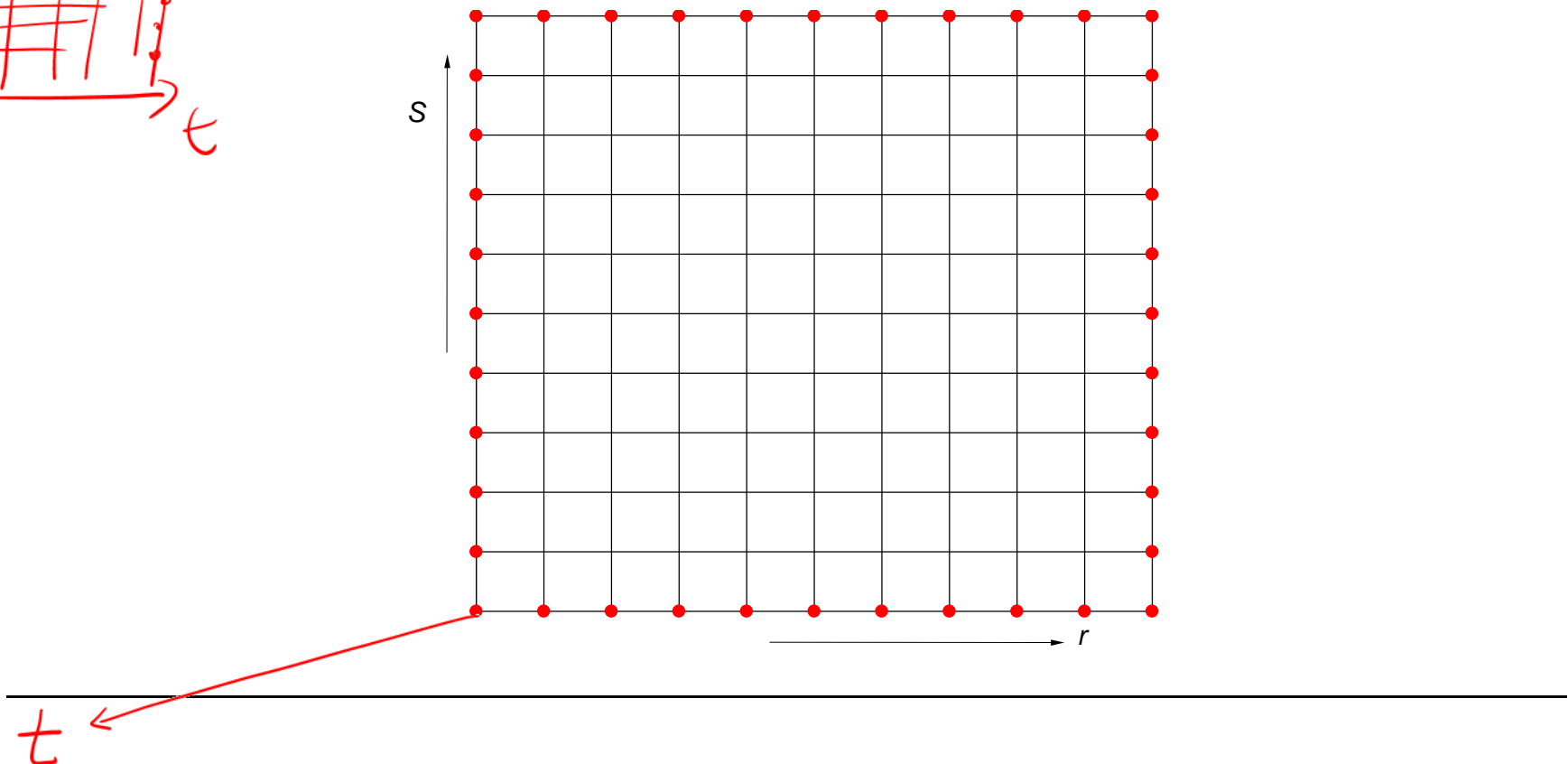
$$\tau = T - k \text{ ft}$$

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

$K=0 \Rightarrow \text{time} = \text{expiring}$

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  $r$ ,

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

normally this would be  
V written  
in f.d  
form

We can approximate this by

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

$i+1$                        $i-1$

But

centred in  
I.R.

$$\left\{ \begin{aligned} \frac{\partial V}{\partial r}(S + \delta S, r, t) &\approx \frac{V_{i+1, j+1}^k - V_{i+1, j-1}^k}{2 \delta r} \\ \text{ex: } \frac{\partial V}{\partial r}(S - \delta S, r, t) &\approx \end{aligned} \right.$$

This suggests that a suitable discretization might be

$$\begin{aligned}
 & \frac{\frac{V_{i+1,j+1}^k - V_{i+1,j-1}^k}{2\delta r} - \frac{V_{i-1,j+1}^k - V_{i-1,j-1}^k}{2\delta r}}{2\delta S} \\
 & \stackrel{\substack{\partial^2 V \\ \partial S \partial r}}{=} \frac{V_{i+1,j+1}^k - V_{i+1,j-1}^k - V_{i-1,j+1}^k + V_{i-1,j-1}^k}{4\delta S \delta r} \stackrel{\substack{\partial^2 V \\ \partial r \partial S}}{=}
 \end{aligned}$$

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

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

$$\frac{\partial}{\partial S} \left( \frac{\partial V}{\partial r} \right)$$

$$\frac{\partial}{\partial r} \left( \frac{\partial V}{\partial S} \right)$$

The resulting explicit difference scheme is

$$\begin{aligned}
 & \frac{\partial v}{\partial t} \frac{V_{ij}^k - V_{ij}^{k+1}}{\delta t} + a_{ij}^k \left( \frac{V_{i+1,j}^k - 2V_{ij}^k + V_{i-1,j}^k}{\delta S^2} \right) \frac{\partial^2 v}{\partial S^2} \\
 & + b_{ij}^k \left( \frac{V_{i+1,j}^k - V_{i-1,j}^k}{2\delta S} \right) \frac{\partial v}{\partial S} + c_{ij}^k V_{ij}^k - rV \\
 & + d_{ij}^k \left( \frac{V_{i,j+1}^k - 2V_{ij}^k + V_{i,j-1}^k}{\delta r^2} \right) \frac{\partial^2 v}{\partial r^2} \\
 & + e_{ij}^k \left( \frac{V_{i+1,j+1}^k - V_{i+1,j-1}^k - V_{i-1,j+1}^k + V_{i-1,j-1}^k}{4\delta S \delta r} \right) \frac{\partial^2 v}{\partial S \partial r} \\
 & + f_{ij}^k \left( \frac{V_{i,j+1}^k - V_{i,j-1}^k}{2\delta r} \right) \frac{\partial v}{\partial r} = O(\delta t, \delta S^2, \delta r^2).
 \end{aligned}$$

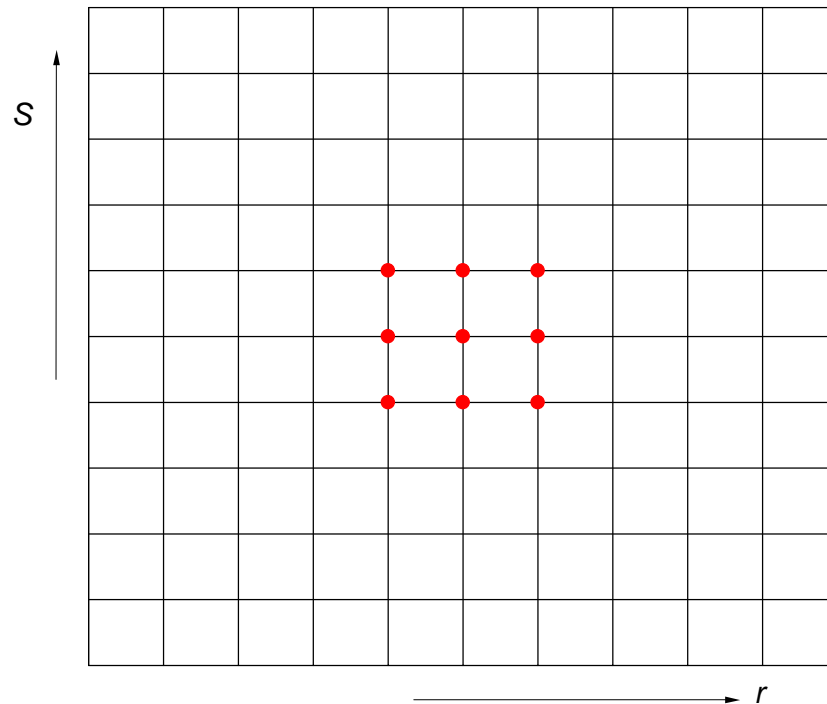
$\frac{\partial v}{\partial t}$  (red)  
 $\frac{\partial^2 v}{\partial S^2}$  (red)  
 $\frac{\partial v}{\partial S}$  (red)  
 $-rV$  (red)  
 $\frac{\partial^2 v}{\partial r^2}$  (red)  
 $\frac{\partial^2 v}{\partial S \partial r}$  (red)  
 $\frac{\partial v}{\partial r}$  (red)  
 $\frac{\partial v}{\partial t}$  (red)  
 $\frac{\partial v}{\partial S}$  (red)  
 $\frac{\partial v}{\partial r}$  (red)  
 $\frac{\partial^2 v}{\partial S^2}$  (red)  
 $\frac{\partial^2 v}{\partial r^2}$  (red)  
 $\frac{\partial^2 v}{\partial S \partial r}$  (red)

Sole unknown option price (red)  
 Time (red)  
 Stock (red)  
 r (red)

We could rewrite this in the form

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

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.

Please take away the following important ideas

- There are many more kinds of finite-difference method

than explicit

Implicit method + associated  
linear algebra techniques.

- Each method has its pros and cons → Linear Algebra

- Finite difference can be used for higher dimensional

problems (as in the last few slides)