

# Further Finite Difference Methods

Numerical Analysis  
Burden and Faires

In this lecture

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

} Implicit schemes

- Douglas Schemes  $\theta$ -method

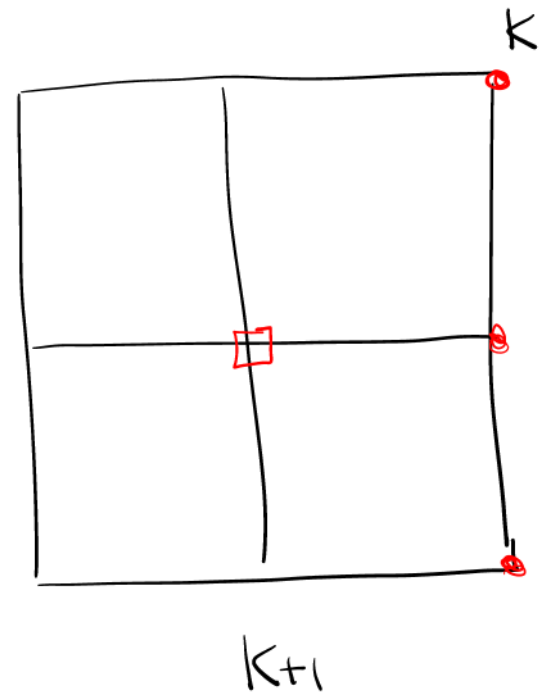
By the end of this lecture you will

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

$$V_i^{k+1} = F \left( \underbrace{V_{i-1}^k, V_i^k, V_{i+1}^k}_{\text{all known}} \right)$$

unknown

$y = f(x)$



## Introduction

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

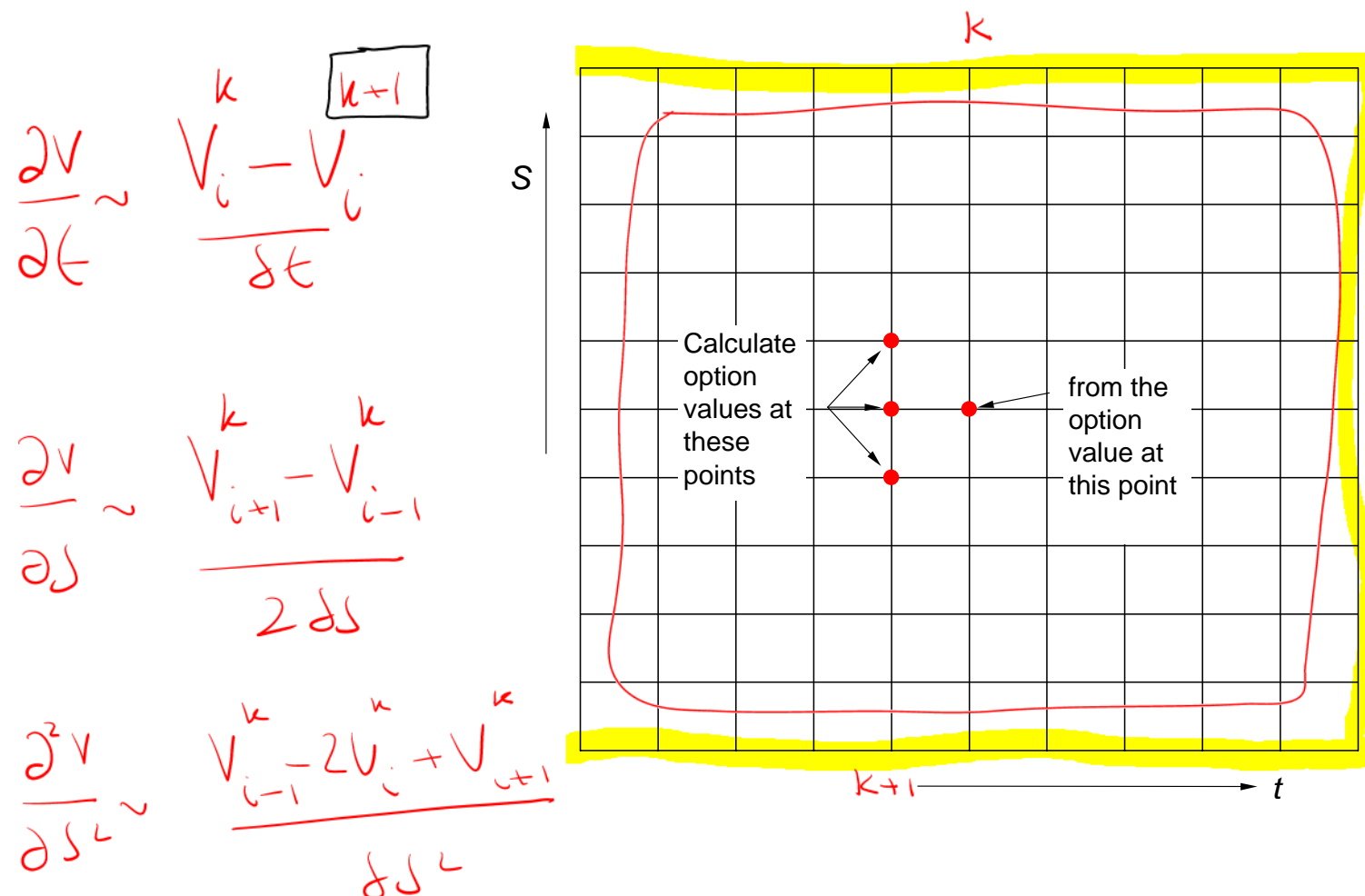
The more advanced methods are usually more complicated to program but have advantages in terms of stability and speed.

↙  
unconditional  
No constraint linking  
 $\delta t, \delta x$

↓  
Computational  
efficiency

# Full 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

single known option price

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

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

Unconditionally stable  $\forall \delta t > 0$

Use same Fourier stability analysis as last week.

$$G(V_{i-1}^{k+1}, V_i^{k+1}, V_{i+1}^{k+1}) = V_i^k \quad \therefore \text{implicit}$$

This can be written as

At each  $k$   
 $i=0$  B.C at  $S=0$  known option values known

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

where

$i=I$  B.C at  $S=S_\infty$

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

$$B_i^{k+1} = 2\nu_1 a_i^{k+1} - \delta t c_i^{k+1} \quad V_i^{k+1} = \alpha_i V_{i-1}^k + \beta_i V_i^k + \gamma_i V_{i+1}^k$$

and

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

where

$$A \underline{x} = \underline{b}$$

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

$$\textcircled{1} = (\sigma^2 S^2 \dots) \delta t$$

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

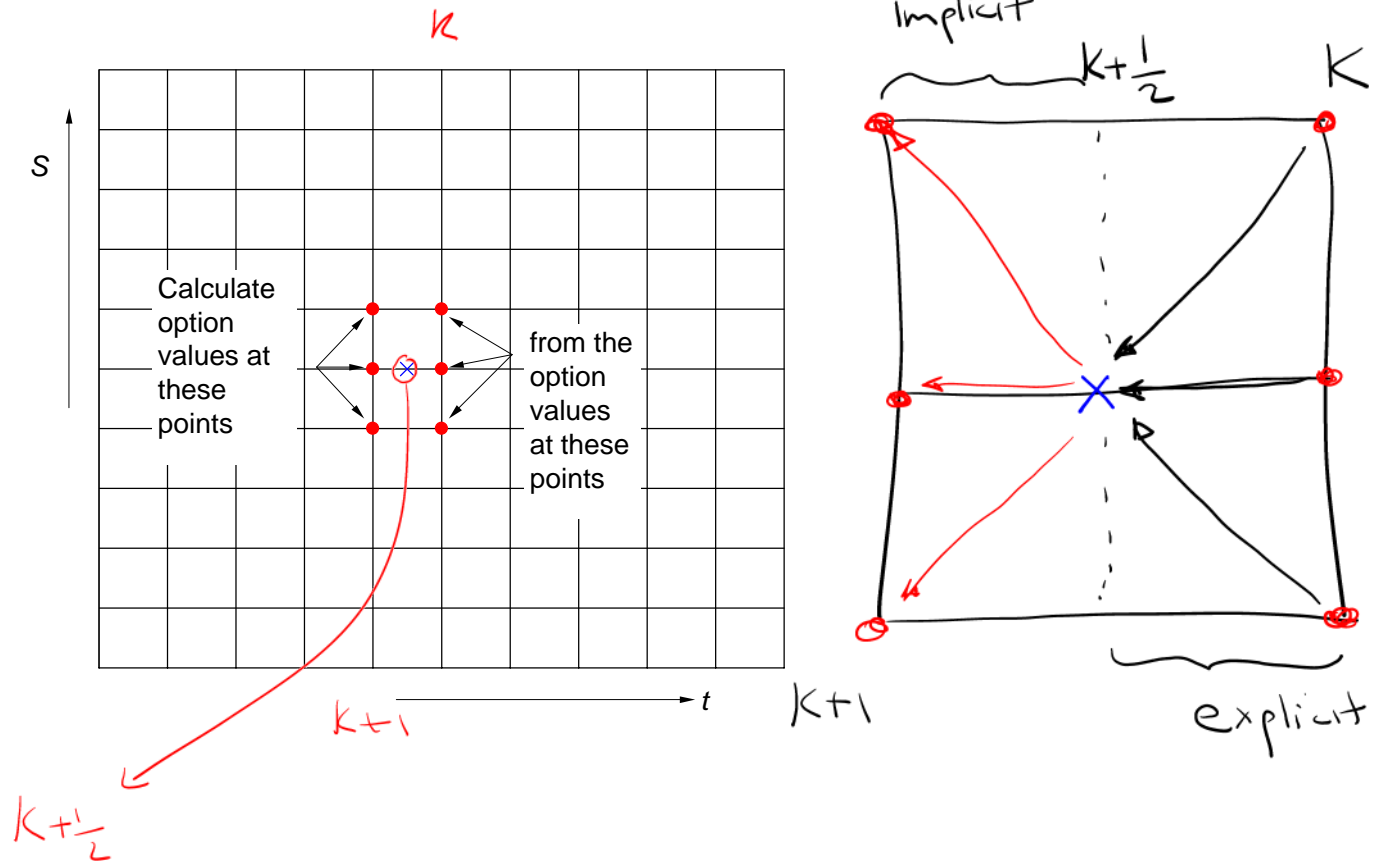
(1) is a system of linear equations, i.e. linear system

## 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.

$$\frac{[K + (K+1)] \delta t}{2}$$

$$(K + \frac{1}{2}) \delta t$$



The Crank–Nicolson scheme is

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

$\uparrow$  implicit  $\checkmark$        $\uparrow$  explicit  $\checkmark$       but  $O(\delta t^2 \delta S^2)$  accurate

$\delta t$        $\delta t^2$



This can be written as

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

where

Again another linear system

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

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

$$\underline{M} \underline{V}^{k+1} = \underline{\bar{M}} \underline{V}^k \rightarrow \text{another vector}$$

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

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

Note: B.C.s have not been incorporated

Tridiagonal



bandwidth = 2

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

$$\begin{pmatrix} \mathbf{M}_L^{k+1} \end{pmatrix}^{-1}$$

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

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.

$$(I+1) \times (I+1)$$

$$0, 1, \dots, I-1, I$$

**Example:** *Repeated from 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

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

as

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

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

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

## The matrix equation

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

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

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

We will tackle  $\underline{A}\underline{x}=\underline{b}$  directly  
i.e.  $\underline{x} = \underline{A}^{-1}\underline{b}$

---

We never calculate  $\underline{A}^{-1}$ .

# Numerical Linear Algebra

There are two much better ways for solving this system, called **LU decomposition** and **successive over-relaxation**. 2012

direct

iteration

For errors and efficiency refer to

Burden and Faires, (very comprehensive)

Direct

(done once)

vs.

Indirect

(iteration)

Method I

## 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.

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

Diagram illustrating the LU decomposition of matrix  $\mathbf{M}$  into lower triangular matrix  $\mathbf{L}$  and upper triangular matrix  $\mathbf{U}$ .

$\mathbf{L}$  is shown as a lower triangular matrix with a zero in the top-right corner.

$\mathbf{U}$  is shown as an upper triangular matrix with a zero in the bottom-left corner.

The process is labeled "row reduce / Gaussian elim<sup>n</sup>".



*M* *given*

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

*Sub-diagonal* →

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

*L* *U* *super*

*i.e.*  $l_i, d_i, u_i$

$$M = LU$$

We assume  $L$  and  $U$  of the given form above  
 (Doolittle's Method) i.e.  $L_{ii} = 1$ .  
 Crout's method  $U_{ii} = 1$

$$\begin{aligned}
 1 \times d_1 &= 1 + B_1 \\
 \Rightarrow d_1 &= 1 + B_1
 \end{aligned}$$

$$u_1 = C_1$$

It is not difficult to show that

$$\rightarrow d_1 = 1 + B_1$$

$u_1 = C_1$

and then

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

algorithm for obtaining all unknown terms in  $L, U$

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

Matrix inversion problem is  $\mathbf{M}\mathbf{v} = \mathbf{q}$

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

We can write


$$\mathbf{L}(\mathbf{U}\mathbf{v}) = \mathbf{q}. \quad (2)$$

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

Let  $\boxed{\mathbf{U}\mathbf{v} = \mathbf{w}}$

$$\mathbf{L}\mathbf{U}\mathbf{v} = \mathbf{q}$$

$\rightarrow \mathbf{L}\mathbf{w} = \mathbf{q}$  ① solve first.

$$\begin{pmatrix} \triangle & \bigcirc \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} \quad \text{to obtain } \underline{\mathbf{w}}$$

and then  $\mathbf{v}$  such that

$$\begin{pmatrix} \triangle & \bigcirc \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} \rightarrow \begin{matrix} \text{fwd.} \\ \text{subst.} \end{matrix}$$

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

$$\begin{pmatrix} \triangle & \bigcirc \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} \quad \begin{matrix} \text{back.} \\ \text{subst.} \end{matrix}$$

And then we are done.

$\rightarrow (v_1, \dots, v_n)$  original vector of unknowns

The first step gives

$$Lw = \underline{q}$$

$$w_1 = q_1$$

forward substitution

and

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

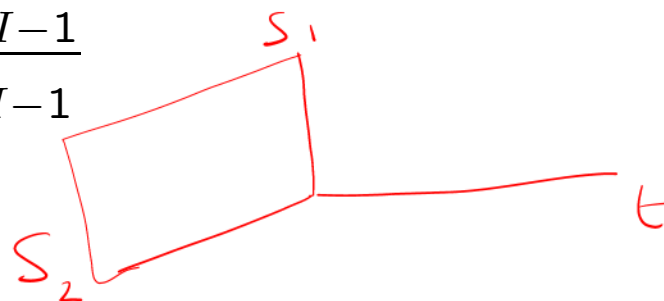
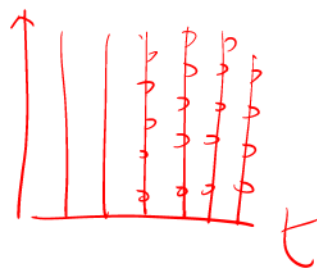
Finally

$$Uv = \underline{w}$$

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

backward substitution

and



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

## Method II

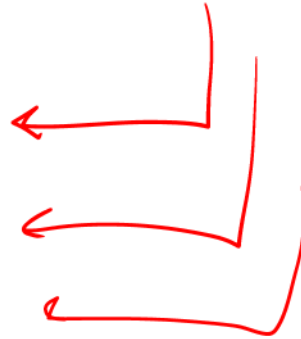
### Successive over-relaxation, SOR

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

In this method we solve the equations iteratively.

Root finding  
Newton-Raphson

Starting solution is a guess



Suppose that the matrix  $\mathbf{M}$  in the matrix equation

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

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

Component form

$$\begin{aligned} E1: & \quad M_{11}v_1 + M_{12}v_2 + \cdots + M_{1N}v_N = q_1 \\ E2: & \quad M_{21}v_1 + M_{22}v_2 + \cdots + M_{2N}v_N = q_2 \\ E3: & \quad \cdots \\ EN: & \quad M_{N1}v_1 + M_{N2}v_2 + \cdots + M_{NN}v_N = q_N \end{aligned}$$

$v_1 = \cdots$

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

Rewrite this as

$$\begin{aligned} \cancel{M_{11}}v_1 &= q_1 - (M_{12}v_2 + \cdots + M_{1N}v_N) \frac{1}{\cancel{M_{11}}} \\ \cancel{M_{22}}v_2 &= q_2 - (M_{21}v_1 + \cdots + M_{2N}v_N) \frac{1}{\cancel{M_{22}}} \\ &\quad \dots \end{aligned}$$

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

$$V_1^{(n+1)} = \frac{-1}{M_{11}} \left( M_{12} \overset{\downarrow(n)}{v_2} + \cdots + M_{1N} \overset{\downarrow(n)}{v_N} \right) + \frac{q_1}{M_{11}}$$

$$\underline{V}^{(0)} = \left( \overset{(0)}{v_1}, \overset{(0)}{v_2}, \overset{(0)}{v_3}, \cdots, \overset{(0)}{v_N} \right) = \underline{0} \quad \text{or all ones}$$



This system is easily solved *iteratively* using

*n - iteration step*

$$\begin{aligned}
 n=0 \quad & v_1^{(1)}, v_2^{(1)}, \dots \\
 n=1 \quad & v_1^{(2)}, v_2^{(2)}, \dots \\
 & v_1^{(n+1)} = \frac{1}{M_{11}} \left( q_1 - (M_{12}v_2^{(n)} + \dots + M_{1N}v_N^{(n)}) \right) \leftarrow \\
 & v_2^{(n+1)} = \frac{1}{M_{22}} \left( q_2 - (M_{21}v_1^{(n+1)} + \dots + M_{2N}v_N^{(n)}) \right) \\
 & v_3^{(n+1)} = \frac{1}{M_{33}} \left( q_3 - (M_{31}v_1^{(n+1)} + M_{32}v_2^{(n+1)} + \dots + M_{3N}v_N^{(n)}) \right) \\
 & v_N^{(n+1)} = \frac{1}{M_{NN}} \left( q_N - (M_{N1}v_1^{(n+1)} + \dots + M_{NN}v_N^{(n)}) \right)
 \end{aligned}$$

*Keep repeating until the solution settles down*

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

This iteration is started from some initial guess  $\mathbf{v}^0$ , e.g.  $\underline{0}$  or  $(1, \dots, 1)$

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

*i<sup>th</sup> iteration*                      *i+1<sup>th</sup>*

# VECTOR NORMS

Let  $\underline{x} \in \mathbb{R}^N$ . Let  $\|\cdot\|_p: \mathbb{R}^N \rightarrow \mathbb{R}$

$$\|\underline{x}\|_p = \left( \sum_{i=1}^N |x_i|^p \right)^{1/p}$$

Generalisation of concept of length in  $N$ -dim<sup>s</sup>

$p=1$   $\|\underline{x}\|_1$  Manhattan norm

$p=2$   $\|\underline{x}\|_2$  Euclidean norm  $\|\underline{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$   
 $l_2$  norm

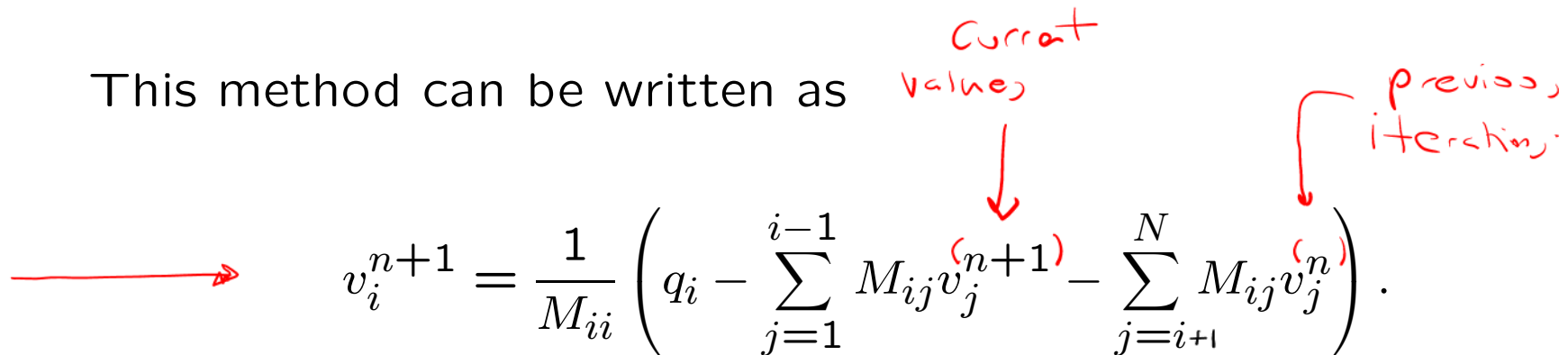
$p=\infty$   $\|\underline{x}\|_\infty$  infinity norm  $\|\underline{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$   $l_\infty$  norm

$$\frac{\|\underline{v}^{(n+1)} - \underline{v}^{(n)}\|_\infty}{\|\underline{v}^{(n)}\|_\infty} < \varepsilon \quad \text{for convergence} \quad \text{user chooses } \varepsilon$$

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


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

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.

S.D.D.  $\begin{pmatrix} \alpha & \beta & \gamma \\ a & b & c \\ A & B & C \end{pmatrix}$   $| \alpha | > | \beta | + | \gamma |$  ✓  
 $| b | > | a | + | c |$  ✓  
 $| C | > | A | + | B |$  ✓

SDD  $\Rightarrow$  inverse exists

If not SDD then either  
 i/ inverse exists  
 ii/ may not exist.

The method can be written as

① non-cgt sol<sup>n</sup> under relaxed  $0 < \omega < 1$

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

② slow to cge over relaxed  $1 < \omega < 2$

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

"spectral radius"

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

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

$A\underline{x} = \underline{b}$  if  $\underline{x}$  is an exact sol<sup>n</sup>  $\underline{b} - A\underline{x} = \underline{0}$   
 if  $\underline{x}$  obtained from numerical scheme,  $\underline{b} - A\underline{x} = \underline{r}$   
 s.o.r  $\lim_{n \rightarrow \infty} \underline{r} \rightarrow \underline{0}$

## Other methods

The questions that arise in any method are

- What is the error in the method in terms of  $\delta t$  and  $\delta S$ ?  
*accuracy of method*
- 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 coefficients, boundary conditions etc.?

## 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.

Weighted Average of exp and imp. schemes.


Recall C-N scheme was equally weighted  $\frac{1}{2}(k + (k+1))\delta t$

Now introduce a weighting  $0 < \theta < 1$

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


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

*exp.*

and the fully implicit is similarly


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

*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...

Exercise

On page 8 rewrite C-N in terms of the  $\theta$  method.

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

$\theta \times \text{Imp.} + (1-\theta) \times \text{Exp.}$

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

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

$$\left[ O\left(\frac{1}{2}\delta t + \frac{1}{12}\delta S^2 - \theta\delta t, \delta S^4, \delta t^2\right) \right] \text{Burden \& Fairweather}$$

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

$$\left. \begin{array}{ll} \theta = 0 & \text{Exp.} \\ \theta = \frac{1}{2} & \text{Imp. C-N} \\ \theta = 1 & \text{F. I} \end{array} \right\}$$

$$\text{Global Error} = \text{Exact} - \text{Approx}^2$$

$$\text{L.T.E} = \text{step} \rightarrow \text{step error.}$$

## 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.

Fwd marching  $(V^k, V^{k+1})$ . Bwd marching  $(V^k, V^{k-1})$

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

If we write  $-2V_i^k = -V_i^{k+1} - V_i^{k-1}$  then the above scheme

---

$$\Rightarrow \text{stable } \forall \delta t$$

## Two-factor models

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

$V(t, S, r)$

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

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

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

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

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

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

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

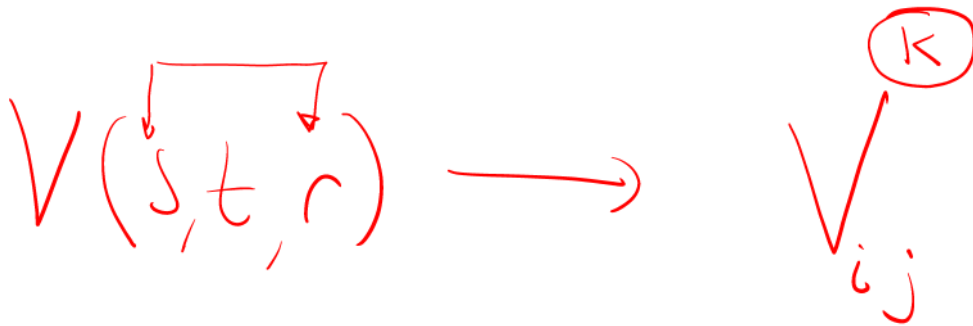
That is, we solve on a three-dimensional grid with

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

$j = 0, \dots, J$

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

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



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.

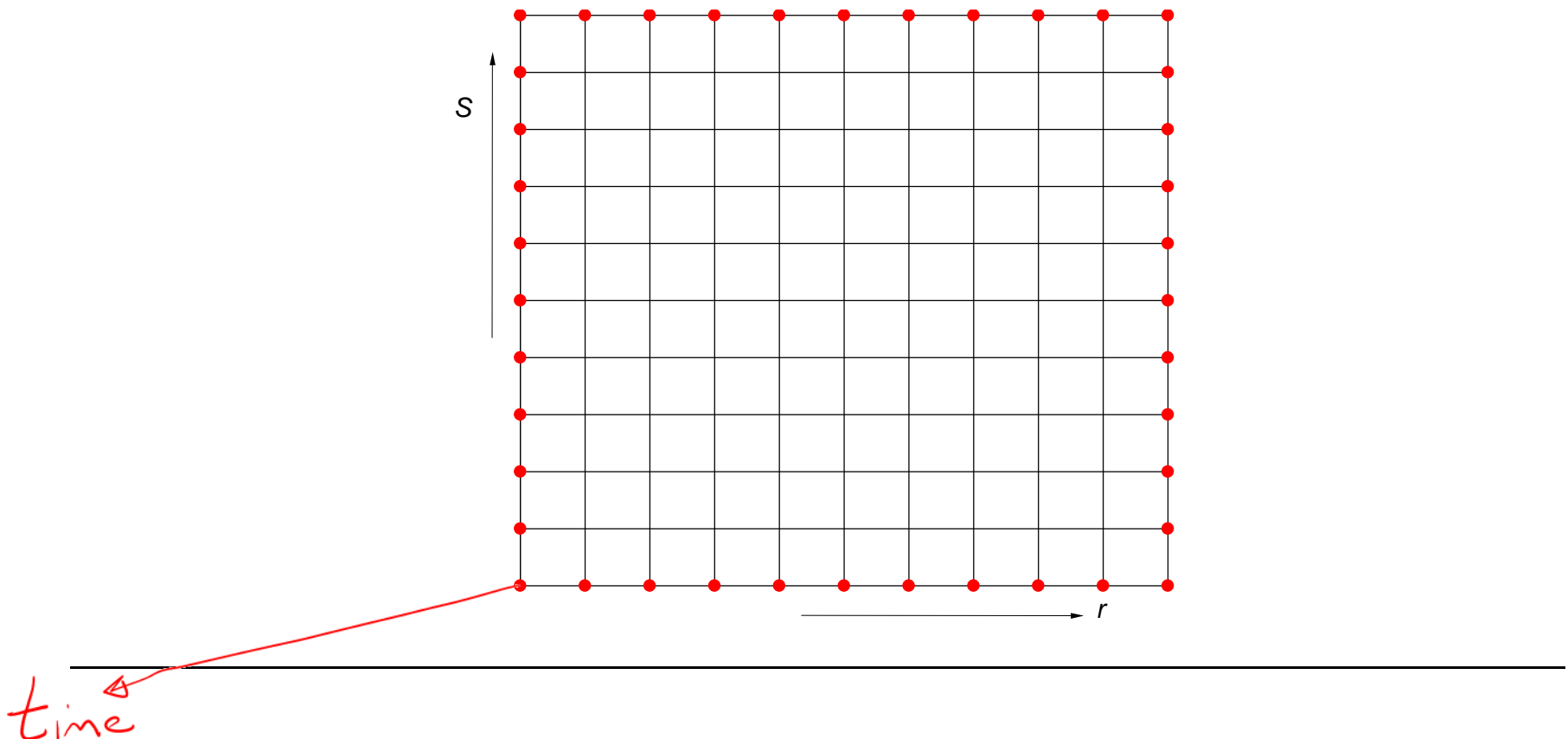
The final condition for this problem is then

The equation  $V(S, r, T) = V_{ij}^0 = \max(S - E, 0)$  is shown. A red box is drawn around the term  $V_{ij}^0$ . A red arrow points from the text "Fwd marching  $\Rightarrow$   $k=0$  is exp'g" to the boxed term  $V_{ij}^0$ .

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

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  $r$ ,

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

We can approximate this by

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

But

FD form

$$\frac{\partial V}{\partial r}(S + \delta S, r, t) \approx \frac{V_{i+1, j+1}^k - V_{i+1, j-1}^k}{2 \delta r} \quad \text{centered in } r$$

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


---

This suggests that a suitable discretization might be

$$\frac{\partial}{\partial S} \left( \frac{\partial V}{\partial r} \right) = \frac{\frac{V_{i+1,j+1}^k - V_{i+1,j-1}^k}{2\delta r} - \frac{V_{i-1,j+1}^k - V_{i-1,j-1}^k}{2\delta r}}{2\delta S}$$

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

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

Exercise: Write in FD form

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

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

The resulting explicit difference scheme is

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

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

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

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

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

zero with accuracy

The single  
unknown  
option value

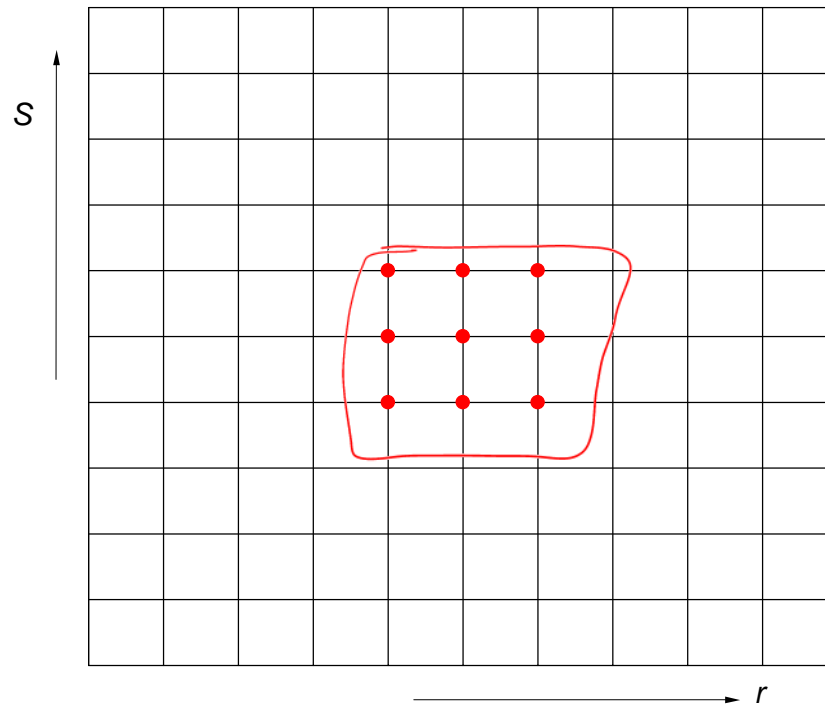
$\frac{\partial^2 V}{\partial S \partial r}$

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.

9 option values  
needed for  
each new one.



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.

If  $\rho = 0 \Rightarrow$   
equation simplifies

Please take away the following important ideas

- There are many more kinds of finite-difference method

than explicit

MC 60%  
FDM 30%  
Tree & Trajectory 10%

- Each method has its pros and cons

- Finite difference can be used for higher dimensional

problems

On Friday

① MC integration

② stability for  
fully imp method.