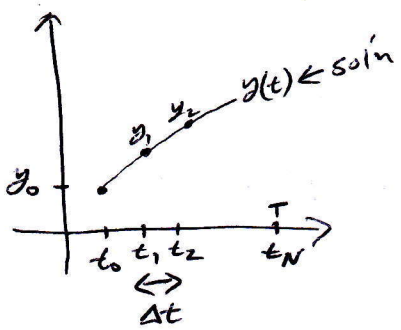
Numerical Methods For Mathematical Finance

ODE

$$\frac{dy}{dt} = f(t,y) \quad \leftarrow \text{ODE}$$

$$y(t_0) = y_0 \quad \leftarrow \text{initial Condition}$$



$$\Delta t = \frac{t_N - t_0}{N} = \frac{T - t_0}{N}$$

Taylor Expansion

Approximating any fn (that doesn't have singularity) with a polynomial expanded as power series about point x_0.

$$f(x) = f(x_0) + \sum_{n=1}^{N} \frac{(x-x_0)^n}{n!} \frac{d^n f}{dx^n}\Big|_{x=x_0} + O(x-x_0)^{N+1}$$

Black-scholes Eqn (option Price)

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

Where, we solve for $V(t,S)$ i.e. value of assets at time t.

σ^2 = Volatility, r = Interest rate

B-S eqn can be solved analytically for only simple cases like that of European call/Put options. Other cases we must rely on Numerical Methods.

Forward Euler Method

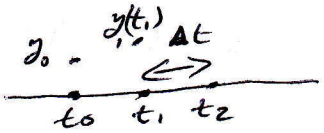
$$\frac{dy}{dt} = f(t,y) \; ; \; y(t_0) = y_0$$

Applying Taylor expansion at t to get value at $(t+\Delta t)$.

$$y(t+\Delta t) = y(t) + \frac{dy}{dt}\Big|_t \Delta t = y(t) + f(t,y)\Delta t$$

Defining mesh as:

$$t_i = t_0 + i\Delta t, \; i = 1, \ldots, N$$



Substituting $t \equiv t_i$, $t + \Delta t = t_{i+1}$

- $$y(t_{i+1}) = y(t_i) + f(t_i, y(t_i))\Delta t + O(\Delta t^2)$$ [Explicit Method]

Back Ward Euler Method

$$\frac{dy}{dt} = f(t,y) \; ; \; y(t_0) = y_0$$

Applying Taylor expansion at t to get value at $t-\Delta t$,

$$y(t-\Delta t) = y(t) - \left.\frac{dy}{dt}\right|_t \Delta t = y(t) - f(t,y)\Delta t$$

Subst. $t \equiv t_{i+1}$, $t-\Delta t = t_i$

$$y_i = y_{i+1} - f(t_{i+1}, y_{i+1})\Delta t$$

$$\boxed{y_{i+1} = y_i + f(t_{i+1}, y_{i+1})\Delta t}$$ Implicit Method

Analytical Soln of 1st order ODE

- Separation of Variable Method

Need to find $y(t)$ that satisfies ODE:

$$\boxed{\frac{dy}{dt} = f(t,y) \; ; \; y(t_0) = y_0 \quad (IC)}$$

If $f(t,y) = G(t)\,H(y)$ then,

$$\frac{dy}{dt} = G(t)\,H(y) \; ; \; y(t_0) = y_0$$

$$\int_{y_0}^{y} \frac{1}{H(y)}\,dy = \int_{t_0}^{t} G(t)\,dt$$ Solving this gives $y(t)$.

Mid-Point Method for ODE

$$\frac{dy}{dt} = f(t,y) \; ; \; y(t_0) = y_0$$

Applying Taylor expansion for Forward & backward to get

(a) $y(t+\Delta t) = y_{i+1}$ & (b) $y(t_i - \Delta t) = y_{i-1}$ & substracting a-

$$y_{i+1} = y_i + \Delta t f(t_i, y_i) + \frac{\Delta t^2}{2} f'(t_i, y_i) + O(\Delta t^3)$$

$$y_{i-1} = y_i - \Delta t f(t_i, y_i) + \frac{\Delta t^2}{2} f'(t_i, y_i) + O(\Delta t^3)$$

(-)

$$y_{i+1} - y_{i-1} = 2\Delta t f(t_i, y_i) + O(\Delta t^3)$$

Note: $f' = \frac{df}{dt} = \frac{d^2y}{dt^2}$

$$\boxed{y_{i+1} = y_{i-1} + 2 f(t_i, y_i)\Delta t + O(\Delta t^3)}$$ ← Mid-pt Method (Explicit & more accurate)

Note: for $i=0$,

$$y_1 = y_{-1} + 2f(t_0, y_0)\Delta t$$

What is y_{-1}? This method requires initialisation to generate additional values from previous time step. So, we need to apply Backward Euler for initialisation step:

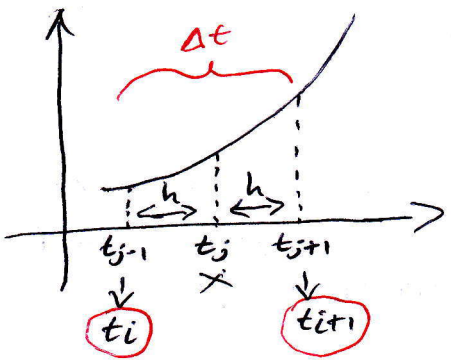
$$y_{-1} = y_0 - \Delta t f(t_0, y_0)$$

Second-Order Runge-Kutta Method (Trapezoidal)

Second-Order Runge-Kutta Method is the modification of Mid-point method by doubling the time-step. i.e. $\Delta t = 2h$.

Mid-Point Method

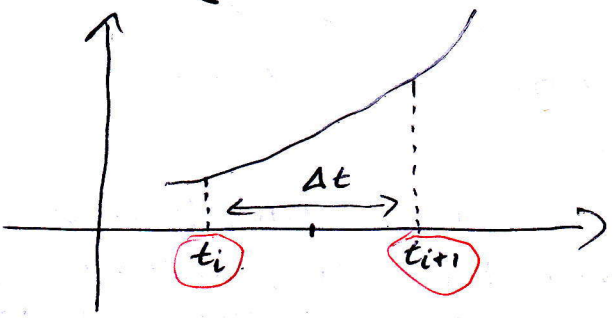
$$y_{j+1} = y_{j-1} + 2f(t_j, y_j)h$$



$\Delta t = 2h$ →

2nd order R-K Method

$$y_{i+1} = y_i + f\left(t_i + \frac{\Delta t}{2}, y\left(t_i + \frac{\Delta t}{2}\right)\right)\Delta t$$



(T.S)

$$y\left(t_i + \frac{\Delta t}{2}\right) \overset{\text{T.S}}{=} y_i + \frac{\Delta t}{2} f(t_i, y_i)$$

So, the 2nd-order R-K method can be written as

$$\boxed{y_{i+1} = y_i + K_2 + O(\Delta t^3)}$$

Where $K_2 = \Delta t f\left(t_i + \frac{\Delta t}{2}, y_i + \frac{K_1}{2}\right)$

$$K_1 = \Delta t f(t_i, y_i)$$

This is an explicit method & the initialisation problem in the mid-pt Method is eliminated by using the Forward Euler explicit Extrapolation for the intermediate step.

Fourth-Order Runge-Kutta Method

Derived in similar way to that of 2nd order R-K Method but using 2 more Taylor Expansion, this method is explicit & more accurate.

$$y_{i+1} = y_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) + O(\Delta t^5)$$

where, $K_1 = \Delta t\, f(t_i, y_i)$

$K_2 = \Delta t\, f\left(t_i + \frac{\Delta t}{2}, y_i + \frac{K_1}{2}\right)$

$K_3 = \Delta t\, f\left(t_i + \frac{\Delta t}{2}, y_i + \frac{K_2}{2}\right)$

$K_4 = \Delta t\, f(t_i + \Delta t, y_i + K_3)$

Errors & Stability

↦ Round-off → error coming from finite precision arithmetic from the computer.

↦ Truncation → Errors coming from the method.

↦ Stability

You should never fully trust the result of your Numerical Sol'n as the effect of time step, Truncation error, Round-off error can change the result that might look exactly like real effects. You should always do some checking & see if result changes:

(a) Change time step.
(b) Use different method.
(c) Use double precision arithmetic.

Stability of Forward & Backward Euler Method

- eg: find/check Stability of Forward Euler (Explicit) & Backward Euler (Implicit) for the following ODE:

$$\frac{dy}{dt} = -y \;;\; y(0) = 1$$

$$\downarrow$$

$$f(t,y)$$

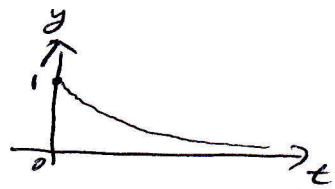
Analytic/Exact soln:

$$\frac{dy}{dt} = -y$$

$$\int_1^y \frac{1}{y}\,dy = \int_0^t -dt$$

$$\ln y - \ln 1 = -t$$

$$y = e^{-t}$$



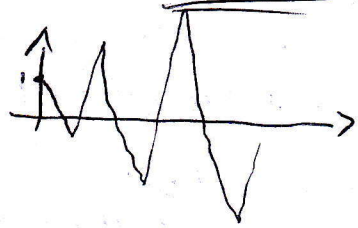
Foward Euler (stability)

$$y_{i+1} = y_i + \Delta t f(t_i, y_i) = y_i + \Delta t(-y_i) = y_i - y_i \Delta t$$

$$y_{i+1} = (1-\Delta t) y_i$$

If $\Delta t > 0 \Rightarrow |y_{i+1}| > |y_i|$

As $i \to \infty \Rightarrow |y| \to \infty$ So Unstable.



Backward Euler (stability)

$$y_{i+1} = y_i + \Delta t f(t_{i+1}, y_{i+1}) = y_i - \Delta t y_{i+1}$$

$$(1+\Delta t) y_{i+1} = y_i$$

$$y_{i+1} = \frac{y_i}{1+\Delta t} \Rightarrow |y_{i+1}| < |y_i| \quad \forall \Delta t$$

Stable.

Analytic Soln of 2nd-Order Linear ODE with Constant Coefficients

$$P(t) y'' + Q(t) y' + S(t) y = h(t); \quad y(t_0) = y_0, \; y'(t_0) = y'_0$$

$P(t), Q(t), S(t)$ & $h(t)$ are constants.

$$Py'' + Qy' + Sy = h; \quad y(t_0) = y_0, \; y'(t_0) = y'_0$$

Sub. e^{rt} into the homogeneous form of above ODE:

$$Py'' + Qy' + Sy = 0$$

$$Pr^2 e^{rt} + Qre^{rt} + Se^{rt} = 0$$

$$Pr^2 + Qr + S = 0 \leftarrow \text{characteristic eqn}$$

r_1 & r_2 are the roots of characteristic eqn.

If (a) $r_1 \neq r_2$ then $y(t) = C_1 e^{r_1 t} + C_2 e^{r_2 t} + \frac{h}{S}$

(b) $r_1 = r_2$ then $y(t) = (C_1 t + C_2) e^{rt} + \frac{h}{S}$

C_1 & C_2 are found by applying initial conditions.

Numerical Soln for System of 1st order ODE's

If we have system of m 1st order ODE's:

$$\frac{dy_1}{dt} = f_1(t, y_1, y_2, \ldots, y_m); \; y_1(t_0) = y_1^0$$

$$\frac{dy_2}{dt} = f_2(t, y_1, y_2, \ldots, y_m); \; y_2(t_0) = y_2^0$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$\frac{dy_m}{dt} = f_m(t, y_1, y_2, \ldots, y_m); \; y_m(t_0) = y_m^0$$

The solution involves applying numerical method to all eqns. We can add a new sub-index j that goes from 1 to m covering all m eqns.
For eg, we can apply forward Euler method:

$$y_{j,i+1} = y_{j,i} + f_j(t, y_{1,i}, y_{2,i}, \ldots, y_{m,i})\,\Delta t$$

Numerical Soln for N^{th} Order Linear ODE's

N^{th} order LODE can be substituted by a system of coupled 1st order differential eqns.
For eg, for case of 2nd order ODE:

$$P(t)\,y'' + Q(t)\,y' + r(t)\,y = h(t); \quad y(t_0) = y_0, \quad y'(t_0) = y_0'$$

Now, substituting:

$$y_1(t) = y(t) \; ; \; y_2(t) = y'(t)$$ we get:

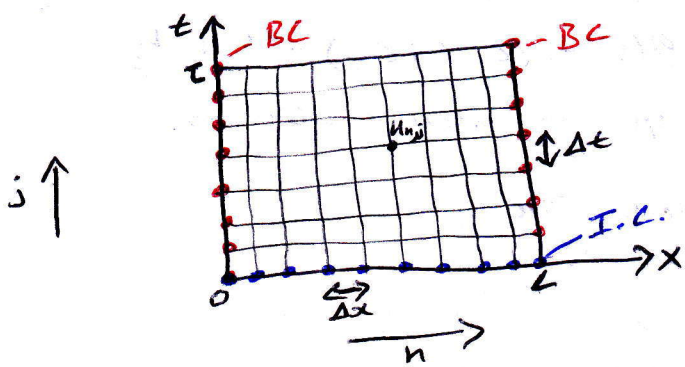
$$\left\{ \begin{aligned} \frac{dy_1}{dt} &= y_2 \\ \frac{dy_2}{dt} &= \frac{h(t)}{P(t)} - \frac{Q(t)}{P(t)}\,y_2 - \frac{r(t)}{P(t)}\,y_1 \end{aligned} \right. ; \quad \begin{aligned} y_1(t_0) &= y_0 \\ y_2(t_0) &= y_0' \end{aligned}$$

↳ We can solve this coupled 1st order ODE's using the Numerical method. This works for any $P(t)$, $Q(t)$, $r(t)$ & $h(t)$.
Note if the value of P, Q, r & t are not constant, its very difficult to solve it analytically & need numerical method as above.

Mesh for F-D representation of PDE's

The soln of PDE is to find the f'n $U(X,t)$. We need to discretize both X & t. ∴ a grid (mesh) is defined in (X,t) space so that each point is characterised by:

$X_n = X_0 + n\Delta X$ where $n = 0, 1, \ldots, N_x$
$t_j = t_0 + j\Delta t$ where $j = 0, 1, \ldots, N_t$
N_x = No. of Points in X-direction.
N_t = No. of Points in t-direction.
time step $= \Delta t = \frac{T}{N_t}$; Space step $= \Delta x = \frac{L}{N_x}$



Defn:
$U(X_n, t_j) \equiv U_{n,j}$

Classification of 2nd-order linear PDE's

Almost all PDE's encountered in Finance are linear & 2nd order. A linear 2nd-order PDE satisfied by a f'n $u(x,t)$ depending on just two variables (space & time) is:

$$\left[A\frac{\partial^2 u}{\partial t^2} + 2B\frac{\partial^2 u}{\partial x \partial t} + C\frac{\partial^2 u}{\partial x^2} = D\left(x, t, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial x}\right) \right]$$

↓

linear in $\frac{\partial u}{\partial t}$ & $\frac{\partial u}{\partial x}$

This can be classified into 3 catagories:

(1) $B^2 - AC > 0 \rightarrow$ Hyperbolic

Wave or Advection eq^n: $\frac{\partial^2 u}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}$

(2) $B^2 - AC = 0 \rightarrow$ Parabolic

Heat or Diffusion eq^n: $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$

(Black-Scholes eq^n)

Since, Black-Scholes eq^n is a Parabolic PDE & can have some component of Advection term we will analyse (1) & (2) mostly for Finance.

(3) $B^2 - AC < 0 \rightarrow$ Elliptic (Not related to Finance)

Laplace Eq^n: $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$

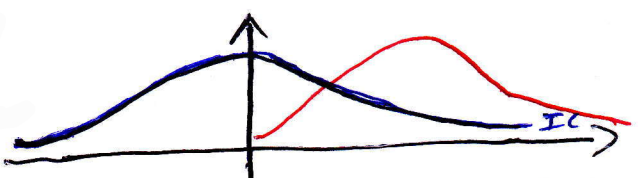
The solution for each type of eq^n differs drastically. Lets look at an example with same initial condition $u(x,0) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ (a pulse) with free boundaries (the derivative of u is zero at Boundary), and look at Sol'n for different eq^{ns} (Wave vs Diffusion eq^n).

The simplest version of both wave & diffusion eq^n can be solved analytically. Lets look at the solutions:

Wave Eq^n

$$\left[u(x,t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-ct)^2}{2}} \right]$$

↑ Sol'n

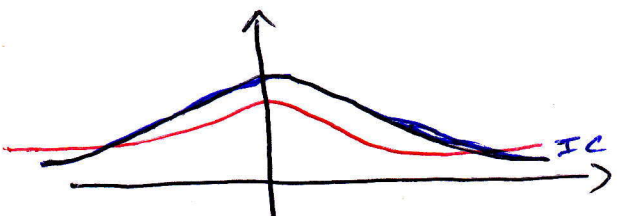


The Sol'n is same as IC but just displaced by c (wave velocity). The form of wave/curve is same.

Diffusion Eq^n

$$\left[u(x,t) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} \frac{1}{\sqrt{4\pi D t}} e^{-\frac{(x-z)^2}{4Dt}} dz \right]$$

Sol'n



The Sol'n has no displacement & maximum decreases & becomes more smooth by diffusion. The Sol'n gets more diffused as time grows.

The Advection / Wave Eqn

Based on conservation of particles inside a volume. The change in the no. of particles flowing inside a finite volume is equal to the flux of the current associated to these moving particles.

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0$$

← 2nd order linear hyperbolic eqn. (linear in both x & t).

Since, fluid is incompressible the no. of particles going inside volume is same as no. of particles going out.

can be seperated into

$$\left(\frac{\partial}{\partial t} + c\frac{\partial}{\partial x}\right)\left(\frac{\partial u}{\partial t} - c\frac{\partial u}{\partial x}\right) = 0$$

$= 0$ or $= 0$

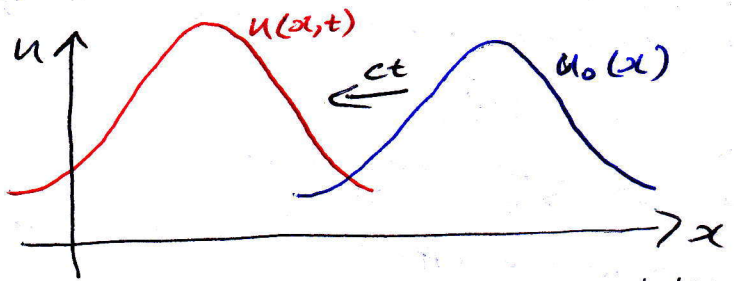
The simplest version where c is constant has an analytical soln.

$$\frac{\partial u}{\partial t} - c\frac{\partial u}{\partial x} = 0$$

$$\frac{\partial u}{\partial t} = c\frac{\partial u}{\partial x} \quad ; \quad u_0(x) = \text{I.C.}$$

The soln is $u(x,t) = u_0(x+ct)$.

This means that the form of the at any time will be same as I.C. but displaced to the left by ct.



When c is not constant, then the analytic evaluation becomes complicated. We use this simple case to check our Numerical method with the exact soln.

Sources of Errors & Stability

(1) **Consistency**: A numerical method is called consistent if the finite difference approximation converges to PDE as space & time step tends to zero. When spatial & time discretisations are kept separate (as in general case) consistency doesn't matter but has to be checked when both discretisation are mixed.

(2) **Stability**: A stable method is when the difference between Numerical soln & exact soln remains bounded as no. of steps tends to ∞.

(3) **Convergence**: A scheme is said to converge if the difference between the numerical soln at a fixed pt. in the domain tends to zero uniformly as the space & time discretisations tends to zero (not necessarily independently from each other).

- **Lax Equivalence Thm**
  Given a properly posed linear initial value problem & a consistent finite difference scheme, stability is the only requirement for convergence.
  So, we will allocate much effort on analysing & understanding stability.

Stability Analysis: Fourier Approach (Von Neumann)

- The stability problem arises because we are using finite precision computer arithmetic to solve the difference eqns, which introduces rounding errors into the numerical soln. The system is said to be stable if these rounding errors are not magnified at each iteration. We must ask the question: if a small error ε is introduced into the soln, is it magnified by numerical method or does it decay away. If it decays then its stable.
  The Fourier method is based on decomposing the numerical soln into Fourier harmonics on the spatial grid. Although this method does not capture the influence of Boundary conditions, it is quite easy to formulate & usually accurate enough to provide practical stability criteria.

- We can decompose the soln into Fourier modes on the mesh:

$$U_{n,j} = \sum_m \underbrace{u_m(t_j)}_{[\xi(k_m)]^j} e^{ik_m x_n}$$

where, $k_m = \frac{m\pi}{L}$

L = total length in X-direction

$x_n = n\Delta x$

- **Stability Analysis:**

$$u_{n,j} = \xi_{(k)}^{j} e^{ikn\Delta x}$$

Substitute this into finite-difference eqⁿ. Now,
If $|\xi_{(k)}| > 1$, it is Unstable.

Explicit Forward time centred Space (FTCS) Method (Advection Eqⁿ)

$$\frac{\partial u}{\partial t} = c \frac{\partial u}{\partial x} \quad \text{——— (i)}$$

- **Spatial discretisation:**
Taylor expanding up to second-order in spatial direction.

$$u(x_n + \Delta x, t_j) = u(x_n, t_j) + \frac{\partial u}{\partial x}\Big|_{n,j} \Delta x + \frac{1}{2} \frac{\partial^2 u}{\partial x^2}\Big|_{n,j} \Delta x^2 + O(\Delta x^3)$$

$$u(x_n - \Delta x, t_j) = u(x_n, t_j) - \frac{\partial u}{\partial x}\Big|_{n,j} \Delta x + \frac{1}{2} \frac{\partial^2 u}{\partial x^2}\Big|_{n,j} \Delta x^2 + O(\Delta x^3)$$

(−)

$$u_{n+1,j} - u_{n-1,j} = 2 \frac{\partial u}{\partial x}\Big|_{n,j} \Delta x + O(\Delta x^3)$$

$$\Rightarrow \left[\frac{\partial u}{\partial x}\Big|_{n,j} = \frac{u_{n+1,j} - u_{n-1,j}}{2\Delta x} + O(\Delta x^2) \right]$$

- **Time discretisation:**
We can just use forward Euler for time derivative.

$$u_{n,j+1} = u_{n,j} + \frac{\partial u}{\partial t}\Big|_{n,j} \Delta t + O(\Delta t^2)$$

$$\frac{\partial u}{\partial t}\Big|_{n,j} = \frac{u_{n,j+1} - u_{n,j}}{\Delta t} + O(\Delta t)$$

Applying the discretisation to (i) yields:

$$\frac{u_{n,j+1} - u_{n,j}}{\Delta t} = c \frac{u_{n+1,j} - u_{n-1,j}}{2\Delta x}$$

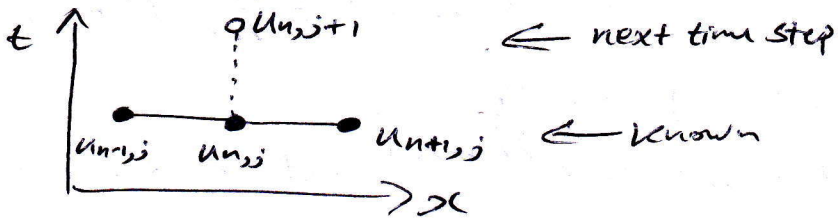
Since we want to find values at next time step, we rearrange to get

$$\boxed{u_{n,j+1} = u_{n,j} + \frac{c\Delta t}{2\Delta x} (u_{n+1,j} - u_{n-1,j})}$$

This is the FD representation & holds for $n = 1, \ldots, N-1$
For $n = 0$ & $n = N$, we apply Boundary Condition i.e.

$u_{0,j}$ & $u_{N,j}$

The FTCS is an explicit scheme: $U_{n,j+1}$ for each n can be calculated explicitly from the quantities that are already known. (The implicit on other hand require us to solve implicit eqns coupling the $U_{n,j+1}$ for various n.) FTCS is an eg. of single-level schemes, as it only requires values at time level j to find values at time level $j+1$.



FD: Stencil for FTCS

Stability Analysis of Forward-time Central Space Method

$$U_{n,j+1} = U_{n,j} + \frac{c\Delta t}{2\Delta x}(U_{n+1,j} - U_{n-1,j}) \quad \leftarrow \text{FTCS for Advection Eqn}$$

Let $E = \frac{c\Delta t}{\Delta x}$ for simplicity so,

$$U_{n,j+1} = U_{n,j} + \frac{E}{2}(U_{n+1,j} - U_{n-1,j}) \quad \text{---} \quad (*)$$

Now, Apply Fourier Approach i.e. substitute

$$U_{n,j} = \xi^j(k)\, e^{ikn\Delta x} \text{ into } (*), \text{ to get}$$

$$\xi^{j+1} e^{ikn\Delta x} = \xi^j e^{ikn\Delta x} + \frac{E}{2}\left[\xi^j e^{ik(n+1)\Delta x} - \xi^j e^{ik(n-1)\Delta x}\right]$$

Simplifying $\rightarrow$

$$\xi = 1 + \frac{E}{2}\left(e^{ik\Delta x} - e^{-ik\Delta x}\right)$$

$\downarrow$ Applying $e^{i\theta} = \cos\theta + i\sin\theta$

$$\xi = 1 + E\, i \sin k\Delta x$$

We need to calculate $|\xi|$ (modulus of ξ) & Since ξ is a complex No, Recall, modulus of complex no: $a+ib$ is $\sqrt{a^2+b^2}$ so,

$$|\xi| = \sqrt{1 + E^2 \sin^2 k\Delta x}$$

As $(E^2 \sin^2 k\Delta x)$ will always be ≥ 0, we have

$|\xi| \geq 1 \Rightarrow$ The solution explodes (oscillatory) & becomes Unstable for any Δt & Δx.

The Lax Method (Advection Eqn)

Starting from the explicit Forward-time Central Space method (FTCS)

$$U_{n,j+1} = U_{n,j} + \frac{c}{2}\frac{\Delta t}{\Delta x}(U_{n+1,j} - U_{n-1,j})$$

The Lax Method simply replaces the term $U_{n,j}$ in the time

derivative by it's average:

$$u_{n,j} \longrightarrow \frac{1}{2}(u_{n+1,j} + u_{n-1,j})$$

This yields Lax-Method:

$$\boxed{u_{n,j+1} = \frac{1}{2}(u_{n+1,j} + u_{n-1,j}) + \frac{c\Delta t}{2\Delta x}(u_{n+1,j} - u_{n-1,j})}$$

- Stability-Analysis of Lax-Method:

Substituting $u_{n,j} = \xi^j e^{ikn\Delta x}$ into Lax-Method & simplifying

$$\xi^{j+1} e^{ikn\Delta x} = \frac{1}{2}\left[\xi^j e^{ik(n+1)\Delta x} + \xi^j e^{ik(n-1)\Delta x}\right] + \frac{c\Delta t}{2\Delta x}\left[\xi^j e^{ik(n+1)\Delta x} - \xi^j e^{ik(n-1)\Delta x}\right]$$

$$\xi = \frac{1}{2}\underbrace{(e^{ik\Delta x} + e^{-ik\Delta x})}_{2\cos k\Delta x} + \frac{c\Delta t}{2\Delta x}\underbrace{(e^{ik\Delta x} - e^{-ik\Delta x})}_{2i\sin k\Delta x}$$

// $\beta = \frac{c\Delta t}{\Delta x}$

$$\xi = \cos k\Delta x + i\beta \sin k\Delta x$$

$$|\xi| = \sqrt{\cos^2 k\Delta x + \beta^2 \sin^2 k\Delta x}$$

// $\cos^2 k\Delta x = 1 - \sin^2 k\Delta x$

$$|\xi| = \sqrt{1 + (\beta^2 - 1)\sin^2 k\Delta x}$$

The stability condition requires $|\xi(k)| \leq 1$. This leads to

$$\sqrt{1 + (\beta^2 - 1)\sin^2 k\Delta x} \leq 1$$

$$\beta^2 - 1 \leq 0 \qquad \text{as max of } \sin\theta = 1.$$

$$\beta \leq 1$$

$$\Rightarrow \boxed{\beta = \frac{c\Delta t}{\Delta x} \leq 1} \quad \text{(Courant-Friedrichs-Lewy Condition)}$$

So, Lax-Method is Conditionally Stable.

Matrix Representation of the Explicit Forward time Central Space Method for Advection Eq^n.

$$\boxed{u_{n,j+1} = \beta u_{n-1,j} + u_{n,j} - \beta u_{n+1,j}} \text{ with } \beta = \frac{c\Delta t}{2\Delta x} \text{ for } n = 1, \ldots, N_x - 1$$

Matrix Representation:

$$\begin{pmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N-1,j+1} \end{pmatrix} = \begin{pmatrix} \beta & 1 & -\beta & 0 & \cdots & 0 \\ 0 & \beta & 1 & -\beta & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & \beta & 1 & -\beta \end{pmatrix} \begin{pmatrix} u_{0,j} \\ u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N-1,j} \\ u_{N,j} \end{pmatrix}$$

Size: $(N-1, 1)$ $\quad (N-1, N+1)$ $\quad (N+1, 1)$

Problems With the Lax Method

$$\beta = \frac{c\,\Delta t}{\Delta x}$$

When,

- $\beta > 1$ ⟶ The method is unstable.
- $\beta < 1$ ⟶ The method gets diffusive like heat eqⁿ (it gets worse to get smaller time steps)
- $\beta = 1$ ⟶ The method converges to exact result.

When we changed FTCS approximation of Advection eqⁿ to get Lax Method, the Lax Method was the FTCS representation of following equation with the introduction of diffusive term.

$$\frac{\partial u}{\partial t} = c\frac{\partial u}{\partial x} + \frac{\Delta x^2}{2\Delta t}\frac{\partial^2 u}{\partial x^2} \qquad \text{——— (*)}$$

(advection eqⁿ + plus extra term + diffusive term)

Lets prove this was true.

So, Approximating $\frac{\partial u}{\partial t}$ by $\frac{u_{n,j+1} - u_{n,j}}{\Delta t}$ — (i)

$\frac{\partial u}{\partial x}$ by $\frac{u_{n+1,j} - u_{n-1,j}}{2\Delta x}$ — (ii)

To approximate $\frac{\partial^2 u}{\partial x^2}$ we do Taylor expansion on x variable up to 2nd order:

$$u(x+\Delta x, t) = u(x,t) + \Delta x\frac{\partial u}{\partial x}(x,t) + \frac{1}{2}\Delta x^2\frac{\partial^2 u}{\partial x^2}(x,t)$$

(+) $$u(x-\Delta x, t) = u(x,t) - \Delta x\frac{\partial u}{\partial x}(x,t) + \frac{1}{2}\Delta x^2\frac{\partial^2 u}{\partial x^2}(x,t)$$

$$u(x+\Delta x,t) + u(x-\Delta x,t) = 2u(x,t) + \Delta x^2\frac{\partial^2 u}{\partial x^2}(x,t)$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{\Delta x^2}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j} = \frac{u_{n+1,j} - 2u_{n,j} + u_{n-1,j}}{\Delta x^2} \qquad \text{— (iii)}$$

Substituting, (i), (ii) & (iii) above in (*) We obtain

$$u_{n,j+1} = \frac{1}{2}(u_{n+1,j} + u_{n-1,j}) + \frac{c\,\Delta t}{2\Delta x}(u_{n+1,j} + u_{n-1,j})$$

which is the Lax Method. ∴ The Lax FD method is not consistent method.

So, be careful when making small changes to a method as it may change the governing eqⁿ.

Staggered Leapfrog Method

As Lax Method is expensive (& dangerous) computationally, we need another method.

A staggered leapfrog method is 2nd order accurate in both space & time & pushes right to their stability limit with smaller computational times:

So, the advection eqn is descritised using staggered leapfrog as

$$\frac{u_{n,j+1} - u_{n,j-1}}{\Delta t} = c\,\frac{u_{n+1,j} - u_{n-1,j}}{\Delta x} \quad \text{---} \circledast$$

- **Stability:** Applying Von Neumann stability analysis (Fourier method) by substituting $u_{n,j} = \xi^j e^{ikn\Delta x}$ into above eq^n $\circledast$ we get:

$$\frac{\xi^{j+1}e^{ikn\Delta x} - \xi^{j-1}e^{ikn\Delta x}}{\Delta t} = c\,\frac{\xi^j e^{ik(n+1)\Delta x} - \xi^j e^{ik(n-1)\Delta x}}{\Delta x}$$

$$\xi - \xi^{-1} = \frac{c\Delta t}{\Delta x}\underbrace{\left(e^{ik\Delta x} - e^{-ik\Delta x}\right)}_{2i\sin k\Delta x}$$

$$\xi^2 - 1 = \xi 2\frac{c\Delta t}{\Delta x} i\sin k\Delta x$$

$$\xi^2 - 2\frac{c\Delta t}{\Delta x} i\sin(k\Delta x)\,\xi - 1 = 0$$

$$\Rightarrow \xi = i\frac{c\Delta t}{\Delta x}\sin k\Delta x \pm \sqrt{1 - \left(\frac{c\Delta t}{\Delta x}\sin k\Delta x\right)^2}$$

if $\left(\frac{c\Delta t}{\Delta x}\sin k\Delta x > 1\right)$ then the square root part is also imaginary & so $|\xi| > 1$.

if $\left(\frac{c\Delta t}{\Delta x}\sin k\Delta x < 1\right)$ then Sq. root part is a Real No so,

$$|\xi|^2 = \left(\frac{c\Delta t}{\Delta x}\sin k\Delta x\right)^2 + \left[1 - \left(\frac{c\Delta t}{\Delta x}\sin k\Delta x\right)^2\right]$$

$$|\xi|^2 = 1$$

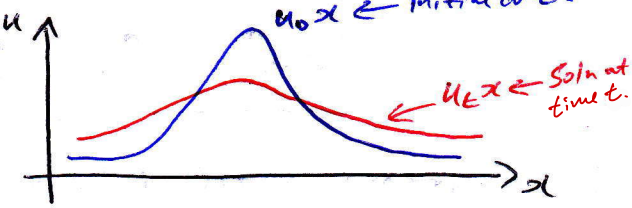
$$|\xi| = 1 \quad \text{if } \frac{c\Delta t}{\Delta x} \le 1$$

Hence, the Courant Condition is again required for stability, in fact $|\xi| = 1$ (no diffusion as method is consistent) for any $c\Delta t \le \Delta x$.

The Heat/Diffusion Eqn

The conservation of Heat-energy inside a Volume is applied: the change in the energy (so that, in the temperature) inside a volume equals the flux of heat, which within the Fourier Law is proportional to the gradient of the temperature (conductivity, D).

$$\begin{cases} \dfrac{\partial u}{\partial t} = D\dfrac{\partial^2 u}{\partial x^2} \\ u(x,0) = u_0(x) \quad (IC) \end{cases}$$



The general Soln for the diffusion eqn with constant coefficient D ~~does~~ (for B.C. i.e. gradient = 0 at both $x=\infty$ & $x=-\infty$) is:

$$u(x,t) = \frac{1}{2\sqrt{\pi D t}} \int_{-\infty}^{\infty} u_0(s)\, e^{-\frac{(x-s)^2}{4Dt}}\, ds$$ ← (Analytic Soln)

Explicit Forward time Central Space Method for Heat/Diffusion Eqn

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \quad \text{— (*)}$$

- time Discretisation:

Taylor expanding up to 1st order in time,

$$u(x,t+\Delta t) = u(x,t) + \frac{\partial u}{\partial t}\Delta t + O(\Delta t^2)$$

$$\frac{\partial u}{\partial t} = \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} + O(\Delta t)$$

$$\left.\frac{\partial u}{\partial t}\right|_{n,j} = \frac{u_{n,j+1} - u_{n,j}}{\Delta t} + O(\Delta t)$$

- Space discretisation:

Taylor expanding up to 2nd order in x, for $x+\Delta x$ & $x-\Delta x$,

$$u(x+\Delta x,t) = u(x,t) + \Delta x \frac{\partial u}{\partial x}(x,t) + \frac{1}{2}\Delta x^2 \frac{\partial^2 u}{\partial x^2}(x,t)$$

$$u(x-\Delta x,t) = u(x,t) - \Delta x \frac{\partial u}{\partial x}(x,t) + \frac{1}{2}\Delta x^2 \frac{\partial^2 u}{\partial x^2}(x,t)$$

(+)

$$u(x+\Delta x,t) + u(x-\Delta x,t) = 2u(x,t) + \Delta x^2 \frac{\partial^2 u}{\partial x^2}(x,t) + O(\Delta x^4)$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{\Delta x^2} + O(x^2)$$

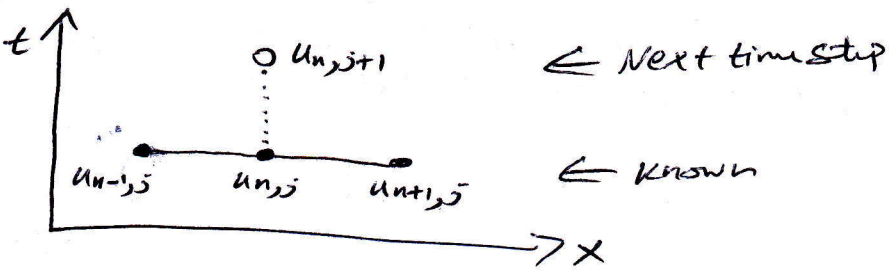
$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j} = \frac{u_{n+1,j} - 2u_{n,j} + u_{n-1,j}}{\Delta x^2} + O(x^2)$$

Applying the time & Space discretisation to heat eqn (*) above yields:

$$\frac{u_{n,j+1} - u_{n,j}}{\Delta t} = D\,\frac{u_{n+1,j} - 2u_{n,j} + u_{n-1,j}}{\Delta x^2} + O(\Delta t, \Delta x^2)$$

The order of error for this method is Δt & Δx^2.

- Stencil:



So, Rearranging the FTCS method for Heat eq^n is:

$$\boxed{u_{n,j+1} = \alpha u_{n+1,j} + (1-2\alpha) u_{n,j} + \alpha u_{n-1,j}} \quad \text{---(*)}$$

With $\alpha = \frac{D\Delta t}{\Delta x^2}$ & $n = 1, \ldots, N_x - 1$

for $n = 0$ & $n = N_x$ we use B.C.

Stability Analysis of Explicit FTCS method for Heat/Diffusion eq^n

Applying Von Neumann Substitution (Fourier method) to (*) above

$$u_{n,j} = \xi^j e^{ikn\Delta x}$$

We get

$$\xi^{j+1} e^{ikn\Delta x} = \alpha\left[\xi^j e^{ik(n+1)\Delta x} - 2\xi^j e^{ikn\Delta x} + \xi^j e^{ik(n-1)\Delta x}\right] + \xi^j e^{ikn\Delta x}$$

$$\xi = 1 + \alpha\left[\underbrace{e^{ik\Delta x} + e^{-ik\Delta x}}_{2\cos k\Delta x} - 2\right]$$

$$\xi = 1 + 2\alpha\left[\cos k\Delta x - 1\right]$$

$$\xi = 1 + 2\frac{D\Delta t}{\Delta x^2}\left[\cos k\Delta x - 1\right]$$

If $\cos k\Delta x = 1 \Rightarrow \xi = 1$ So satisfies Stability Criteria $|\xi| \leq 1$.

For other $\cos k\Delta x$ values, the requirement $|\xi(k)| \leq 1$ leads to the following Stability Criterion:

$$\boxed{\frac{2D\Delta t}{\Delta x^2} \leq 1}$$

This condition puts huge limitations on the calculation procedure. For eg, if we want to analyse with great detail in space, taking very small step size in x-direction ($\Delta x \ll 1$) implies that $\Delta t \leq \frac{\Delta x^2}{2D}$ so $\Delta t \ll 1$, so a huge no. of steps in time will be required until something interesting happens. The computational requirement may be enormous & therefore even though the method is stable it's not efficient & we require new method.

Fully Implicit Method for the Diffusion Eqn

This is like the FTCS method except that the spatial derivatives on the RHS of Heat eqn ($\frac{\partial^2 u}{\partial x^2}$) are evaluated at time step $j+1$:

$$\frac{u_{n,j+1} - u_{n,j}}{\Delta t} = D\,\frac{u_{n+1,j+1} - 2u_{n,j+1} + u_{n-1,j+1}}{\Delta x^2} + O(\Delta t, \Delta x^2)$$

↑ Accuracy / Error of method

Stencil:

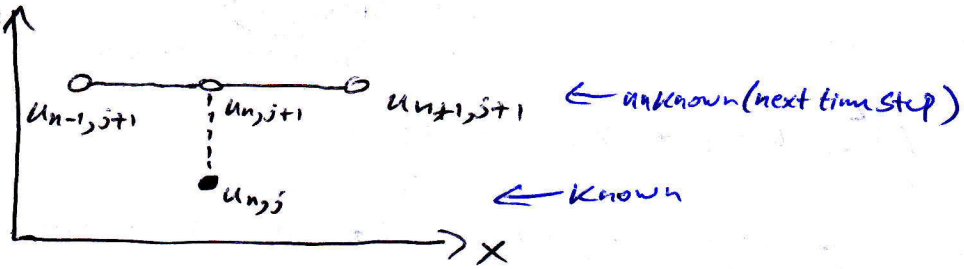


Fig: Implicit

Rewriting,

$$u_{n,j} = -\alpha u_{n-1,j+1} + (1+2\alpha) u_{n,j+1} - \alpha u_{n+1,j+1}$$

with $n = 1, \ldots, N_x - 1$ & $\alpha = \frac{D\Delta t}{\Delta x^2}$

Supplemented by Boundary conditions at $n=0$ & $n = N_x$.

We need Matrix representation to solve this as there are 3 unknowns at time $= j+1$.

Matrix Representation of the fully Implicit Method for the Diffusion Eqn

$$u_{n,j} = -\alpha u_{n-1,j+1} + (1+2\alpha) u_{n,j+1} - \alpha u_{n+1,j+1}$$

with $n = 1, \ldots, N_x - 1$

$\alpha = \frac{D\Delta t}{\Delta x^2}$

Matrix Representation:

$$\begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N_x-1,j} \end{pmatrix} = \begin{bmatrix} -\alpha & (1+2\alpha) & -\alpha & 0 & \cdots & 0 \\ 0 & -\alpha & (1+2\alpha) & -\alpha & 0 \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -\alpha & (1+2\alpha) & -\alpha \end{bmatrix} \begin{pmatrix} u_{0,j+1} \\ u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N_x-1,j+1} \\ u_{N_x,j+1} \end{pmatrix}$$

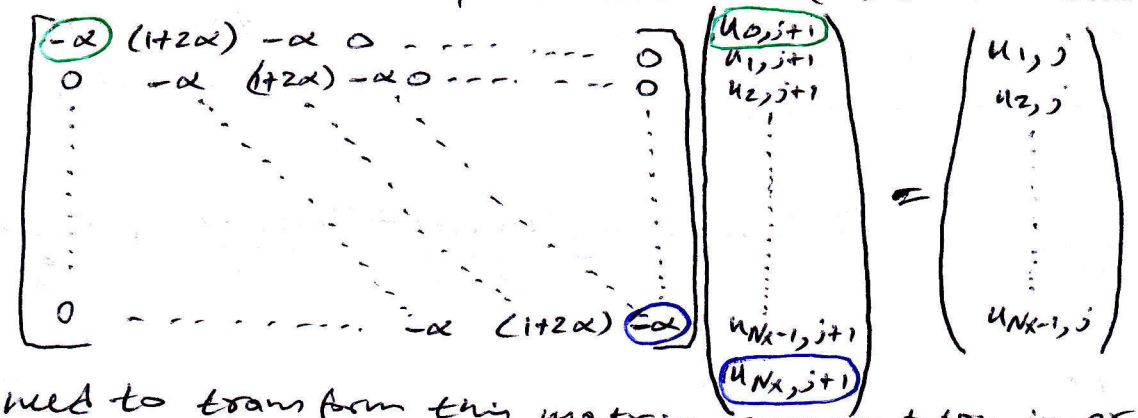
size: $(N_x - 1, 1)$ $\quad$ $(N_x - 1, N_x + 1)$ $\quad$ $(N_x + 1, 1)$

If the matrix were square matrix, we could use inverse of matrix & find the values of u for next step $(j+1)$. For eg if $\underline{a} = M\,\underline{b}$

$$\underline{b} = M^{-1}\underline{a}$$

However, for the problem above the matrix has $N_x - 1$ rows and $N_x + 1$ columns (Not Sq. Matrix), which is the representation of the $N_x - 1$ eqns and $N_x + 1$ unknowns. The missing eqns for $u_{0,j+1}$ & $u_{N_x,j+1}$ comes from the Boundary Conditions. Using these conditions, we are going to convert this system of equations into a system of eqns involving a Sq. Matrix.

Lets rewrite the Matrix representation with (j+1) terms on LHS & j on RHS.



We need to transform this matrix representation in order to get:

$$M^{L}_{j+1}\, u_{j+1} + Y^{L}_{j+1} = u_j$$

- M^{L}_{j+1}: Square Matrix (L = LHS)
- u_{j+1}: Some vector evaluated at time $(j+1)$
- Y^{L}_{j+1}: Some vector where B.C. are incorporated
- u_j: same u vector as above (RHS)

We will consider how to do this transformation for two types of Problem

(i) Dirichlet BC

(ii) Neumann BC

Matrix Representation of Fully Implicit Method for Diffusion eqn with Dirichlet BC

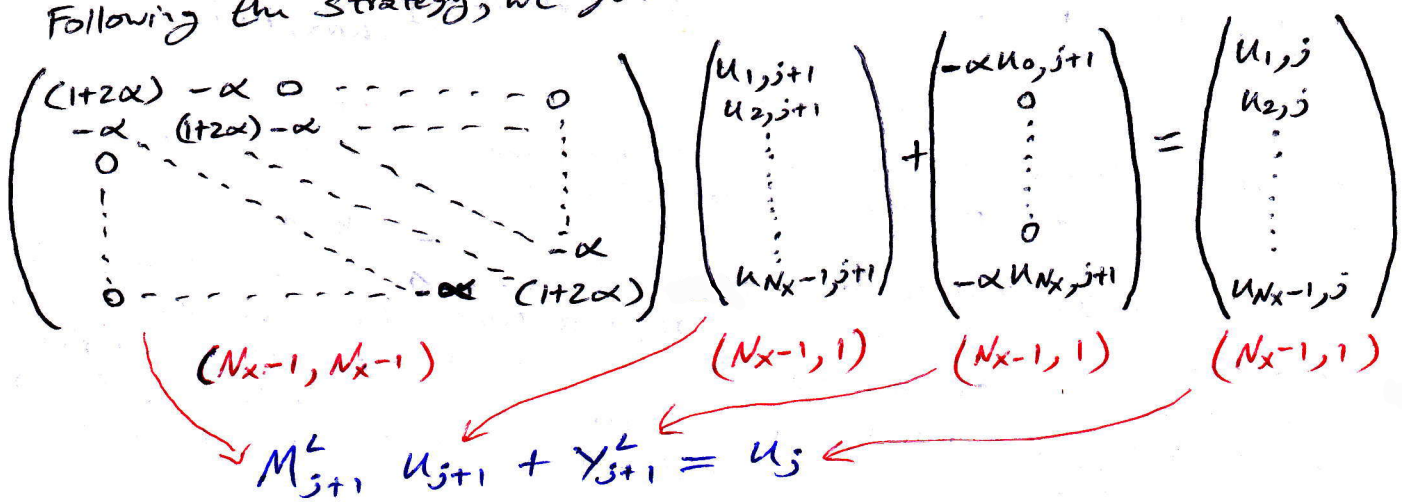
Dirichlet BC means we know the value of u at on the Boundary $u_{0,j}$ & $u_{N_x,j}$.

The Strategy is as follows:

(i) We are going to remove 1st & last column of Matrix so that it is Square Matrix. (N_x-1, N_x-1).

(ii) The 1st & last value of vector on LHS will be removed as they are known through BC i.e. $u_{0,j+1}$, $u_{N_x,j+1}$. This does have an effect so we add another vector Y_{j+1} which incorporates this information i.e. the product of $-\alpha \times u_{0,j+1}$ & $-\alpha \times u_{N_x,j+1}$ shown by green & blue circle in above diagram respectively.

Following the strategy, we get the transformed Matrix Representation as



(N_x-1, N_x-1) $(N_x-1, 1)$ $(N_x-1, 1)$ $(N_x-1, 1)$

$$M^{L}_{j+1}\, u_{j+1} + Y^{L}_{j+1} = u_j$$

With the transformed system we can get u_{j+1} :

$$\left[u_{j+1} = \left(M_{j+1}^{-1}\right)^{L} \left(u_j - Y_{j+1}^{L}\right) \right]$$

Matrix Representation of fully Implicit method for diffusion eq^n with Neumann BC

Neumann BC involves spatial derivative of f'n at the Boundary.

Consider a particular case:

$$\text{B.C.} \begin{cases} \frac{\partial u}{\partial x}(0,t) = u(0,t) \\ \frac{\partial u}{\partial x}(L,t) = -u(L,t) \end{cases} \forall t$$

(Left Boundary; Right Boundary)

We need to make finite-difference representation of these BC's.

Taking Foward differences:

$$\frac{\partial u}{\partial x}(0,t) = u(0,t) \rightarrow \frac{u_{1,j} - u_{0,j}}{\Delta x} = u_{0,j} \Rightarrow u_{0,j} = \frac{u_{1,j}}{1+\Delta x}$$

$$\frac{\partial u}{\partial x}(L,t) = -u(L,t) \rightarrow \frac{u_{Nx,j} - u_{Nx-1,j}}{\Delta x} = -u_{Nx,j} \Rightarrow u_{Nx,j} = \frac{u_{Nx-1,j}}{1+\Delta x}$$

Writing this way, we see Boundary nodes $u_{0,j}$ & $u_{Nx,j}$ are connected to nodes $u_{1,j}$ & $u_{Nx-1,j}$ respectively. Note we don't have the actual node value like in Dirichlet BC but we have to use the connection between nodes to transform into Sq. matrix representation.

Transform:

$$u_{0,j+1} = \frac{u_{1,j+1}}{1+\Delta x} \quad (BC)$$

$$\begin{bmatrix} -\alpha & (1+2\alpha) & -\alpha & 0 & \cdots & 0 \\ 0 & -\alpha & (1+2\alpha) & -\alpha & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & & -\alpha & (1+2\alpha) & -\alpha \end{bmatrix} \begin{bmatrix} u_{0,j+1} \\ u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{Nx-1,j+1} \\ u_{Nx,j+1} \end{bmatrix} = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{Nx-1,j} \end{bmatrix}$$

$(1+2\alpha) \rightarrow 1+2\alpha - \frac{\alpha}{1+\Delta x}$ (top-left)

$(1+2\alpha) \rightarrow 1+2\alpha - \frac{\alpha}{1+\Delta x}$ (bottom-right)

$$u_{Nx,j+1} = \frac{u_{Nx-1,j+1}}{1+\Delta x} \quad (B.C.)$$

Omiting 1st & last column of matrix & 1st & last elmnt of j+1 vector on L.H.S, we achieve the transformation but we must include the contribution associated with $(-\alpha \times u_{0,j+1})$ & $(-\alpha \times u_{Nx,j+1})$. Since, $u_{0,j+1}$ is connected with $u_{1,j+1}$ by B.C. the effect of this multiplication of $-\alpha$ with $u_{0,j+1}$ is like adding this (contribution that) multiplication by u_1 in following way

- the $M_{0,0}$ elmnt of Sq. Matrix becomes $(1+2\alpha) \rightarrow 1+2\alpha - \frac{\alpha}{1+\Delta x}$

Similarly, - the $M_{Nx-1, Nx-1}$ elmnt of Sq. matrix becomes $(1+2\alpha) \rightarrow 1+2\alpha - \frac{\alpha}{1+\Delta x}$

The transformation looks like:

$$\begin{bmatrix} \left(1+2\alpha-\frac{\alpha}{1+\Delta x}\right) & -\alpha & 0 & \cdots & 0 \\ -\alpha & (1+2\alpha) & -\alpha & 0 & \cdots \\ 0 & \ddots & \ddots & \ddots & -\alpha \\ \vdots & & & & \\ 0 & \cdots & -\alpha & & \left(1+2\alpha-\frac{\alpha}{1+\Delta x}\right) \end{bmatrix} \begin{pmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N_x-1,j+1} \end{pmatrix} = \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N_x-1,j} \end{pmatrix}$$

M^L_{j+1} $\quad u_{j+1}$ $\quad u_j$

Size: (N_x-1, N_x-1) $\quad (N_x-1, 1)$ $\quad (N_x-1, 1)$

Hence, u_{j+1} can be written as:

$$\left[u_{j+1} = (M^L_{j+1})^{-1} u_j \right]$$

Note, this time $Y^L_{j+1} = 0$.

Stability of fully implicit method for Diffusion eqn

$$u_{n,j} = -\alpha u_{n-1,j+1} + (1+2\alpha) u_{n,j+1} - \alpha u_{n+1,j+1} \quad \text{with } \alpha = \frac{D\Delta t}{\Delta x^2}$$

Applying Von Neumann Substitution ($u_{n,j} = \xi^j e^{ikn\Delta x}$)

$$\xi^j e^{ikn\Delta x} = -\alpha \xi^{j+1} e^{ik(n-1)\Delta x} + (1+2\alpha)\xi^{j+1} e^{ikn\Delta x} - \alpha \xi^{j+1} e^{ik(n+1)\Delta x}$$

$$1 = -\alpha \xi e^{-ik\Delta x} + (1+2\alpha)\xi - \alpha \xi e^{ik\Delta x}$$

$$\xi = \frac{1}{1+2\alpha(1-\cos k\Delta x)}$$

Since $2\alpha(1-\cos k\Delta x) \geq 0$

$|\xi(k)| \leq 1$, for any $\Delta t, \Delta x$.

So, the scheme is <u>unconditionally stable</u>.

Crank–Nicholson Method for Diffusion Eqn

This combines the stability of an Implicit method with the accuracy of a method that is 2nd order in both space & time. (Note: The previous fully implicit method was only 1st order in time). We will take the average of the explicit & implicit FTCS schemes (left & right hand sides are both centred at time step $j+\frac{1}{2}$).

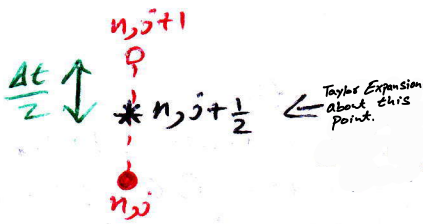
Taylor expanding about $u_{n,j+\frac{1}{2}}$ forward & backward in time gives:

$$u_{n,j+1} = u_{n,j+\frac{1}{2}} + \left.\frac{\partial u}{\partial t}\right|_{n,j+\frac{1}{2}} \frac{\Delta t}{2} + \frac{1}{2}\left.\frac{\partial^2 u}{\partial t^2}\right|_{n,j+\frac{1}{2}} \left(\frac{\Delta t}{2}\right)^2 + O(\Delta t^3)$$

$$u_{n,j} = u_{n,j+\frac{1}{2}} - \left.\frac{\partial u}{\partial t}\right|_{n,j+\frac{1}{2}} \frac{\Delta t}{2} + \frac{1}{2}\left.\frac{\partial^2 u}{\partial t^2}\right|_{n,j+\frac{1}{2}} \left(\frac{\Delta t}{2}\right)^2 + O(\Delta t^3)$$

(−)

$$u_{n,j+1} - u_{n,j} = 2\left.\frac{\partial u}{\partial t}\right|_{n,j+\frac{1}{2}} \frac{\Delta t}{2} + O(\Delta t^3)$$



$$(*)\quad \left.\frac{\partial u}{\partial t}\right|_{n,j+\frac{1}{2}} = \frac{u_{n,j+1} - u_{n,j}}{\Delta t} + O(\Delta t^2)$$

We are also going to Taylor expand f'n $\frac{\partial^2 u}{\partial x^2}$ about pt. $n,j+\frac{1}{2}$ going only in time direction (considering only time evolution)

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j+1} = \left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j+\frac{1}{2}} + \frac{\partial}{\partial t}\left(\left.\frac{\partial^2 u}{\partial x^2}\right)\right|_{n,j+\frac{1}{2}} \frac{\Delta t}{2} + O(\Delta t^2)$$

$$(+)\quad \left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j} = \left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j+\frac{1}{2}} - \frac{\partial}{\partial t}\left(\left.\frac{\partial^2 u}{\partial x^2}\right)\right|_{n,j+\frac{1}{2}} \frac{\Delta t}{2} + O(\Delta t^2)$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j+1} + \left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j} = 2\left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j+\frac{1}{2}} + O(\Delta t^2)$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j+\frac{1}{2}} = \frac{1}{2}\left[\left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j+1} + \left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j}\right]$$

Now, introducing space evolution:

$$(*)\quad \left.\frac{\partial^2 u}{\partial x^2}\right|_{n,j+\frac{1}{2}} = \frac{1}{2}\left[\frac{u_{n+1,j+1} - 2u_{n,j+1} + u_{n-1,j+1}}{\Delta x^2} + \frac{u_{n+1,j} - 2u_{n,j} + u_{n-1,j}}{\Delta x^2}\right]$$

Sub (*)'s into diffusion eqn yields:

$$\Rightarrow \frac{u_{n,j+1} - u_{n,j}}{\Delta t} = \frac{D}{2}\left[\frac{(u_{n+1,j+1} - 2u_{n,j+1} + u_{n-1,j+1}) + (u_{n+1,j} - 2u_{n,j} + u_{n-1,j})}{\Delta x^2}\right] + O(\Delta t^2, \Delta x^2)$$

Rewriting $j+1$ in RHS & j term in LHS gives:

$$-\alpha u_{n-1,j+1} + (2+2\alpha) u_{n,j+1} - \alpha u_{n+1,j+1} = \alpha u_{n-1,j} + (2-2\alpha) u_{n,j} + \alpha u_{n+1,j}$$

with $n = 1, \dots, N_x - 1$ & $\alpha = \frac{D\Delta t}{\Delta x^2}$

Supplemented by B.C. at $n = 0$ & $n = N_x$.

Matrix Representation of Crank-Nicholson method for diffusion eqn

The matrix representation this time will have 2 Matrices on both sides of eqn.

$$\begin{bmatrix} -\alpha & (2+2\alpha) & -\alpha & 0 & \cdots & 0 \\ 0 & -\alpha & (2+2\alpha) & -\alpha & & \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \cdots & & -\alpha & (2+2\alpha) & -\alpha \end{bmatrix} \begin{bmatrix} u_{0,j+1} \\ u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N_x-1,j+1} \\ u_{N_x,j+1} \end{bmatrix} = \begin{bmatrix} \alpha & (2-2\alpha) & \alpha & 0 & \cdots & 0 \\ 0 & \alpha & (2-2\alpha) & \alpha & & \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \cdots & & \alpha & (2-2\alpha) & \alpha \end{bmatrix} \begin{bmatrix} u_{0,j} \\ u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N_x-1,j} \\ u_{N_x,j} \end{bmatrix}$$

$(N_x - 1, N_x + 1)$ $(N_x + 1, 1)$ $(N_x - 1, N_x + 1)$ $(N_x + 1, 1)$

Note, the 2 Matrices have $N_x - 1$ rows & $N_x + 1$ columns which are the representation of $N_x - 1$ eqns & $N_x + 1$ unknowns. The two missing eqns comes from the BC. Using these conditions we are going to convert this system of eqns into a system of eqns involving Sq. matrices. The aim is to write a system of eqns in the form:

$$M^L_{j+1} u_{j+1} + Y^L_{j+1} = M^R_j u_j + Z^R_j \qquad (L = \text{LHS eqn},\ R = \text{RHS eqn})$$

So that,

$$u_{j+1} = \left(M^L_{j+1}\right)^{-1} \left(M^R_j u_j + Z^R_j \bullet - Y^L_{j+1} \right)$$

Sq. Matrix · u vector · Some vector incorporting BC

Dirichlet BC in Matrix Representation of Crank Nicholson Method for Diffusion Eqn

Dirichlet BC $\Rightarrow$ $u_{0,j}$ & $u_{N_x,j}$ is given for all j.

We follow same strategy of fully implicit case but this time we have to do for both matrices on left & Right hand sides of the eqn. So, we omit 1st & last column of matrix. Since, $u_{0,j+1}$ & $u_{N_x,j+1}$ are known by BC we omit that too from u vector. But since there are contribution of product of $-\alpha \times u_{0,j+1}$ & $-\alpha \times u_{N_x,j+1}$, this needs to be included in additional vector Y. Same procedure is done for RHS as well.

$$\begin{bmatrix} (2+2\alpha) & -\alpha & 0 & \cdots & 0 \\ -\alpha & (2+2\alpha) & -\alpha & & \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & & -\alpha & (2+2\alpha) \end{bmatrix} \begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N_x-1,j+1} \end{bmatrix} + \begin{bmatrix} -\alpha u_{0,j+1} \\ 0 \\ \vdots \\ -\alpha u_{N_x,j+1} \end{bmatrix} \leftarrow M^L_{j+1} u_{j+1} + Y^L_{j+1}$$

Transformed Square Matrix representation of LHS, incorporting BC.

$$= \begin{bmatrix} (2-2\alpha) & \alpha & 0 & \cdots & 0 \\ \alpha & (2-2\alpha) & \alpha & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & \alpha \\ \vdots & \ddots & \ddots & \ddots & (2-2\alpha) \end{bmatrix} \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N_x-1,j} \end{pmatrix} + \begin{bmatrix} \alpha u_{0,j} \\ \vdots \\ \alpha u_{N_x,j} \end{bmatrix} \leftarrow M_j^R u_j + z_j^R$$

Transformed Square matrix representation of RHS incorporating B.C.

So, $M_{j+1}^L u_{j+1} + y_{j+1}^L = M_j^R u_j + z_j^R$

$$\left[u_{j+1} = (M_{j+1}^L)^{-1} \left[M_j^R u_j + z_j^R - y_{j+1}^L \right] \right] //$$

Neumann BC in the Matrix Representation of Crank-Nicholson Method for Diffusion Eq^n

Neumann BC $\Rightarrow$ Spatial derivative of u f'n at $x=0$ & $x=L$ is known for all time values.

BC:

$$\frac{\partial u}{\partial x}(0,t) = u(0,t) \rightarrow \frac{u_{1,j} - u_{0,j}}{\Delta x} = u_{0,j} \rightarrow u_{0,j} = \frac{u_{1,j}}{1+\Delta x}$$

$$\frac{\partial u}{\partial x}(L,t) = u(L,t) \rightarrow \frac{u_{N_x,j} - u_{N_x-1,j}}{\Delta x} = -u_{N_x,j} \rightarrow u_{N_x,j} = \frac{u_{N_x-1,j}}{1+\Delta x}$$

We apply the same strategy used in fully implicit method but this time we have to do this for both matrices on LHS & RHS of eq^n. Since, $u_{0,j}$ & $u_{1,j}$ and also $u_{N_x,j}$ & $u_{N_x-1,j}$ are related using BC we omit 1st & last column of the matrices. The contribution of

$-\alpha \times u_{0,j+1}$ will be added to 1st elmt of New Sq. Matrix

$-\alpha \times u_{N_x,j+1}$ will be added to last elmt of New Sq. Matrix

Now, we omit 1st & last element of u vector.
We allow same strategy to RHS. This yields,

$$\begin{bmatrix} \left(2+2\alpha-\frac{\alpha}{1+\Delta x}\right) & -\alpha & 0 & \cdots & 0 \\ -\alpha & (2+2\alpha) & -\alpha & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & -\alpha \\ \vdots & \ddots & \ddots & \ddots & \left(2+2\alpha-\frac{\alpha}{1+\Delta x}\right) \end{bmatrix} \begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N_x-1,j+1} \end{bmatrix} =$$

M_{j+1}^L (Sq. Matrix)

u_{j+1} (u vector)

Note: $y_{j+1}^L = 0$

$$= \begin{bmatrix} \left(2-2\alpha+\frac{\alpha}{1+\Delta x}\right) & \alpha & 0 & \cdots & 0 \\ \alpha & (2-2\alpha) & \alpha & & \\ 0 & \ddots & \ddots & \ddots & \alpha \\ \vdots & & \ddots & \ddots & \alpha \\ 0 & \cdots & \cdots & \alpha & \left(2-2\alpha+\frac{\alpha}{1+\Delta x}\right) \end{bmatrix} \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N_x-1,j} \end{pmatrix}$$

M_j^R (Source Matrix) $\qquad$ u_j (u vector)

(Here: $Z_j^R = 0$)

So, $M_{j+1}^L u_{j+1} = M_j^R u_j$

$$u_{j+1} = (M_{j+1}^L)^{-1} [M_j^R u_j^R]$$

Stability Analysis of Crank-Nicholson Method for Diffusion eqn

$-\alpha u_{n-1,j+1} + (2+2\alpha) u_{n,j+1} - \alpha u_{n+1,j+1} = \alpha u_{n-1,j} + (2-2\alpha) u_{n,j} + \alpha u_{n+1,j}$, $\alpha = \frac{D\Delta t}{\Delta x^2}$

Applying Von Neumann Substitution ($u_{n,j} = \xi_{(k)}^j e^{ikn\Delta x}$) to the above method & going through calculation gives:

$$\xi = \frac{1-\alpha(1-\cos k\Delta x)}{1+\alpha(1-\cos k\Delta x)}$$

The denominator is always higher than numerator so, $|\xi_{(k)}| \leq 1$ for any $\Delta t, \Delta x$. This scheme is unconditionally stable.

As Matrix Inversion is incredibly inefficient, we will explore Iterative methods.

The Jacobi Method

This is an indirect method (has to be solved iteratively) to solve matrix eqn $M\underline{u} = \underline{q}$ and although the resulting soln will never be exact, we can find it to whatever accuracy we want.

Method: If we have $M\underline{u} = \underline{q}$

$$\begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1N} \\ M_{21} & M_{22} & \cdots & M_{2N} \\ \vdots & & & \vdots \\ M_{N1} & M_{N2} & & M_{NN} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix} \longleftrightarrow$$

$$M_{11}u_1 + M_{12}u_2 + \cdots + M_{1N}u_N = q_1$$
$$M_{21}u_1 + M_{22}u_2 + \cdots + M_{2N}u_N = q_2$$
$$\vdots$$
$$M_{N1}u_1 + M_{N2}u_2 + \cdots + M_{NN}u_N = q_N$$

Now, we write this eqn so that we have eqn in terms of u_i's (the diagonal elements above circled)

$$u_1 = \frac{1}{M_{11}}\left[q_1 - (M_{12}u_2 + \dots + M_{1N}u_N)\right]$$
$$u_2 = \frac{1}{M_{12}}\left[q_2 - (M_{21}u_1 + \dots + M_{2N}u_N)\right]$$
$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$
$$u_N = \frac{1}{M_{NN}}\left[q_N - (M_{N1}u_1 + \dots + M_{NN-1}u_{N-1})\right]$$

— (*)

This system is solved iteratively (Jacobi Method) by starting with some initial guess

$$\underline{u}^0 = (u_1^0, u_2^0, \dots, u_N^0) \leftarrow \text{here } i=0 \text{ (iteration level)}$$

We plug values of $\underline{u}^0$ (initial guess) into RHS of above (*) to find u values of next iteration. We keep iterating until convergence.

$$u_1^{i+1} = \frac{1}{M_{11}}\left[q_1 - (M_{12}u_2^i + \dots + M_{1N}u_N^i)\right]$$
$$u_2^{i+1} = \frac{1}{M_{12}}\left[q_2 - (M_{21}u_1^i + \dots + M_{2N}u_N^i)\right]$$
$$\vdots \qquad\qquad \vdots$$
$$u_N^{i+1} = \frac{1}{M_{NN}}\left[q_N - (M_{N1}u_1^i + \dots + M_{NN-1}u_{N-1}^i)\right]$$

The superscript i & $i+1$ denotes the level of iteration. We stop the iteration based on some error tolerance level ε. The convergence criteria can be $|\underline{u}^{i+1} - \underline{u}^i| < \varepsilon$ or absolute relative difference i.e. $\dfrac{|\underline{u}^{i+1} - \underline{u}^i|}{|\underline{u}^{i+1}|} < \varepsilon$.

The jacobi Method can be written succently as:

$$u_n^{i+1} = \frac{1}{M_{nn}}\left[q_n - \sum_{j=1}^{n-1} M_{nj}u_j^i - \sum_{j=n+1}^{N} M_{nj}u_j^i\right]$$

for $n = 1, \dots, N$

i = level of iteration

The Gauss–Seidel Method

The Jacobi Method is inefficient as some of the values u_n^{i+1} are evaluated before others and not used in the current iterations. The Gauss–Seidel method is the modification of Jacobi method where the updated values are used as soon as they are evaluated. So,

$$u_1^{i+1} = \frac{1}{M_{11}}\left[q_1 - (M_{12}u_2^i + \dots + M_{1N}u_N^i)\right]$$
$$u_2^{i+1} = \frac{1}{M_{12}}\left[q_2 - (M_{21}u_1^{i+1} + \dots + M_{2N}u_N^i)\right]$$
$$\vdots \qquad\qquad \vdots$$
$$u_N^{i+1} = \frac{1}{M_{NN}}\left[q_N - (M_{N1}u_1^{i+1} + \dots + M_{NN-1}u_{N-1}^{i+1})\right]$$

Before, writing the formula for Gauss-Seidel Method, lets define our Matrix in a different way so that both Jacobi & Gauss-Seidel method can be written more elegantly.
Lets write the matrix M as the sum of a diagonal matrix D, an upper triangular matrix T & a lower triangular matrix L.

$$M = D + T + L$$

$$\begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1N} \\ M_{21} & M_{22} & \cdots & M_{2N} \\ \vdots & & & \vdots \\ M_{N1} & M_{N2} & \cdots & M_{NN} \end{bmatrix} = \begin{bmatrix} M_{11} & 0 & \cdots & 0 \\ 0 & M_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & M_{NN} \end{bmatrix} + \begin{bmatrix} 0 & M_{12} & \cdots & M_{1N} \\ 0 & 0 & & M_{2N} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & \cdots & 0 \\ M_{21} & 0 & & 0 \\ \vdots & & \ddots & \vdots \\ M_{N1} & M_{N2} & \cdots & 0 \end{bmatrix}$$

$$M \qquad\qquad D \qquad\qquad T \qquad\qquad L$$

Using this representation of M, we can rewrite Jacobi method as:

$$u_n^{i+1} = \frac{1}{M_{nn}} \left[q_n - \sum_{j=1}^{n-1} M_{nj} u_j^i - \sum_{j=n+1}^{N} M_{nj} u_j^i \right]$$

$$\Downarrow$$

$$u^{i+1} = D^{-1} \left[q - (T+L) u^i \right] \leftarrow \text{Jacobi Method.}$$

Now, the Gauss-Seidel Method can be written as:

$$u_n^{i+1} = \frac{1}{M_{nn}} \left[q_n - \sum_{j=1}^{n-1} M_{nj} u_j^{i+1} - \sum_{j=n+1}^{N} M_{nj} u_j^i \right]$$

$$\Downarrow$$

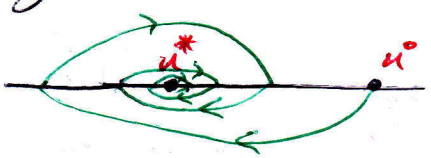
$$u^{i+1} = D^{-1} \left[q - L u^{i+1} - T u^i \right] \leftarrow \text{Gauss-Seidel Method.}$$

SOR (Successive Over-Relaxation) Method

SOR method is slight modification/improvement based on Gauss-Seidel method. Generally, iterative methods usually converge to the correct soln from one side, that is, the correction $u_n^{i+1} - u^i$ stays on the same side of the sign as no. of iterations i increases.

u^* → exact solution
u° → Initial guess value of iteration
↷ → At each iteration, we move close to soln u^* but we remain on same side.

SOR allows the method to go to other side back & forth to speed up the convergence.



↷ → Allows method to go to other side at each iteration to speed up convergence to u^*.

SOR method is the weighted of the u value obtained at previous iteration plus the Gauss-Seidel method. The parameter w is the Acceleration/over relaxation parameter which is $1 \leq w \leq 2$. w allows the method to go from 1-side to the other.

The SOR method can be written as:

$$u_n^{i+1} = (1-\omega)\,u_n^i + \omega \times (\text{Gauss-Seidel})$$

↑ previous iteration value

$$u_n^{i+1} = (1-\omega)\,u_n^i + \frac{\omega}{M_{nn}}\left[q_n - \sum_{j=1}^{n-1} M_{nj}\,u_j^{i+1} - \sum_{j=n+1}^{N} M_{nj}\,u_j^i\right]$$ ← SOR Method

This can be rewritten in D, T, L matrix as:

$$u^{i+1} = (1-\omega)\,u^i + \omega D^{-1}[q - Lu^{i+1} - Tu^i]$$

$$u^{i+1} = (1-\omega)\,u^i + \omega D^{-1}q - \omega D^{-1}Lu^{i+1} - \omega D^{-1}Tu^i$$

$$u^{i+1} + \omega D^{-1}Lu^{i+1} = (1-\omega)\,u^i - \omega D^{-1}Tu^i + \omega D^{-1}q$$

$$u^{i+1}[I + \omega D^{-1}L) = ((1-\omega)I - \omega D^{-1}T)u^i + \omega D^{-1}q$$

SOR → $$u^{i+1} = (I + \omega D^{-1}L)^{-1}\left[((1-\omega)I - \omega D^{-1}T)u^i + \omega D^{-1}q\right]$$

- **Optimal Value of ω**

The error $e^i = u^i - u^*$, where u^* is the exact solution, satisfies

$$e^{i+1} = \underbrace{(I+\omega D^{-1}L)^{-1}[(1-\omega)I - \omega D^{-1}T]}_{A}\,e^i$$

$$e^{i+1} = A\,e^i$$

The SOR method will converge provided that the largest of the moduli of the eigenvalues (called Spectral radius) of the matrix A is less than 1. There is a theoretical optimum value for ω, that is when the spectral radius is minimum. In practice it is very simple to iterate on ω to find the optimum value by calculating spectral radius for different values of ω.

- **Recap (Spectral Radius of a Matrix)**
  - For a $n \times n$ matrix A,

$(A - \lambda I)$ ⟶ characteristic matrix.

$|A - \lambda I|$ —(Taking determinant)→ characteristic polynomial.

$|A - \lambda I| = 0$ ⟶ characteristic eqn.

The zeros/root/soln of characteristic eqn is called eigenvalues of A.

  - For a $n \times n$ matrix A, it will have n eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$.

→ Spectrum of A = set of all eigenvalues of A = $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ no repetation of λ_i

→ Spectral radius of A = $\max\{|\lambda_1|, |\lambda_2|, \ldots, |\lambda_n|\}$ ← max. of modulus of eigenvalues of A.

eg: If $A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ -4 & 4 & 3 \end{pmatrix}$ then,

$$|A - \lambda I| = 0$$

$$\begin{vmatrix} 1-\lambda & 1 & 1 \\ 0 & 2-\lambda & 1 \\ -4 & 4 & 3-\lambda \end{vmatrix} = 0$$

$$-\lambda^3 + 6\lambda^2 - 11\lambda + 6 = 0$$

So, $\lambda = 1, 2, 3$ are eigen values of A.

The spectral radius of $A = \max\{|1|, |2|, |3|\}$

$= 3$ //

Derivation of the Black-Scholes Equation

The option value can be written interms of two independant variables S (Asset Price) & t (time).

$$V(S, t; \sigma, \mu; E, T; r) = V(S, t)$$

- σ = annual Volatility of asset (constant)
- μ = average rate of growth of asset
- E = Strike Price
- T = Maturity
- r = Interest rate (constant)

Parameters

We create a portfolio Π which is long position on option and A short position in some quantity Δ of underlying S (asset).

$$\Pi = V(S, t) - \Delta S$$

Assuming S follows a log Normal Random Walk:

$$dS = \mu S\, dt + \sigma S\, dX$$

Weiner Process

$dX = N(0,1)\sqrt{dt}$

$\overline{dX^2} = dt$

$\sigma_{dt} = \sigma\sqrt{dt}$

↓ SD in time interval dt

↓ Volatility

$$ds^2 = \mu^2 s^2 dt^2 + \sigma^2 s^2 dX^2 + 2\mu s\, dt\, \sigma s\, dX$$

$$ds^2 = \sigma^2 s^2 dt \quad (dX^2 \to dt)$$

The change in the value of the portfolio from t to t+dt is:

$$d\Pi = dV - \Delta\, dS$$

Now, Taking Taylor expansion of dV in both S & t up to 2nd order:

$$dV = ds\frac{\partial V}{\partial s} + dt\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial s^2}ds^2 + \frac{1}{2}\frac{\partial^2 V}{\partial t^2}dt^2 + \frac{\partial^2 V}{\partial s \partial t}ds\,dt$$

We need to go up to 2nd order as $ds^2 = \sigma^2 s^2 dt$. Ignoring higher order in dt i.e. dt^2 & $ds\,dt$ we get:

$$dV = \frac{\partial V}{\partial s}ds + \frac{\partial V}{\partial t}dt + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 V}{ds^2}dt \quad \text{(Ito's lemma)}$$

i.e. keeping only the linear terms & ignoring higher order terms.

Therefore,

$$d\Pi = \frac{\partial V}{\partial t}dt + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}dt + \frac{\partial V}{\partial S}dS - \Delta dS$$

Now, choosing $\Delta = \frac{\partial V}{\partial S}$, the randomness (risk) associated with the stochasticity of the variable dS is reduced to zero. This is called <u>Delta Hedging</u> as we eliminate the dS term.

So,

$$d\Pi = \frac{\partial V}{\partial t}dt + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}dt$$

As this change is completely riskless, it must be the same as the growth we would have got if we put the equivalent amount of cash in a risk-free interest-bearing bank account

$$d\Pi = r\Pi dt$$

Assuming No Arbitrage,

So,

$$r\Pi dt = \frac{\partial V}{\partial t}dt + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}dt$$

$$\boxed{\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0}$$ ← Black Scholes Equation

There is no effect of μ on the eqn. Since, the eqn doesn't specify the kind of option, all option follow B-S eqn. Different kind of option information will be included in the Initial & Boundary Condition. The B-S eqn is a 2-dimensional linear Parabolic PDE. linear implies if we have 2 soln's for the eqn then their sum is also a solution. Parabolic implies that if we start with a discontinuity in the final data, due to a discontinuity in the payoff, this immediately gets smoothed out, due to the diffusive nature of the eqn.

- <u>Meaning of terms of BS eqn</u>

$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}$ → Basic diffusion contribution, so that any discontinuity in the payoff would be instantly diffused away. The diffusion coefficient is a fn of S.

$rS\frac{\partial V}{\partial S}$ → Represents the advection term moving V in a preferred direction (like breeze blows the smoke)

rV → This is a reaction term. Balancing this term & the time derivative would give a model for decay of a radioactive body.

Boundary and Initial/Final Conditions in Black-Scholes Eqn

Black-Scholes eqn knows nothing about what kind of option we are valuing, whether it is a call or put nor what is the strike price and the expiry, which are dealt with the final condition.

To uniquely specify the problems we must prescribe Boundary Conditions (how the soln must behave for all time at certain values of the asset, usually at $S=0$ & $S\to\infty$) and Initial $t=0$, or Final Conditions $t=T$ (Payoff at Maturity). For option pricing we usually consider Final Condition than initial.

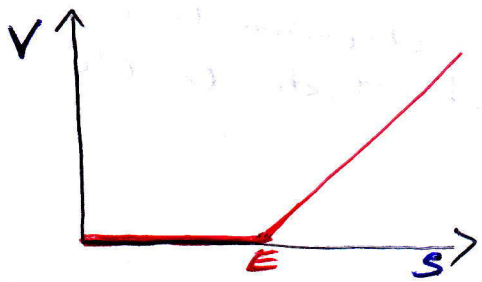
The Black-Scholes eqn is a Backward Eqn (the signs of the t derivative & the 2nd S derivative in the equation are the same when written on the same side which is different to the diffusion eqn), thus, a final condition usually a payoff at maturity $V(S,T)$ has to be imposed.

When we solve for diffusion eqn we need initial condition. In B-S eqn we are given Final Condition, so we need a change of variable from t to τ with the relationship $\tau = T-t$, so that the final condition for t becomes initial condition for the variable τ.

Initial/Final Conditions based on Payoffs at Maturity depending on τ / t

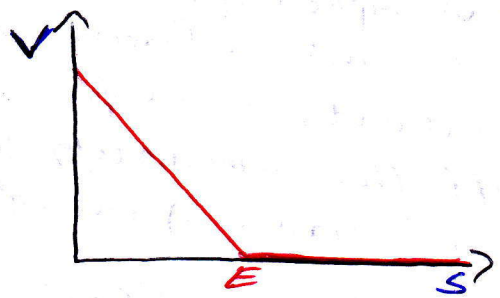
- European Call

  $V(S,T) = \max(S-E,0)$



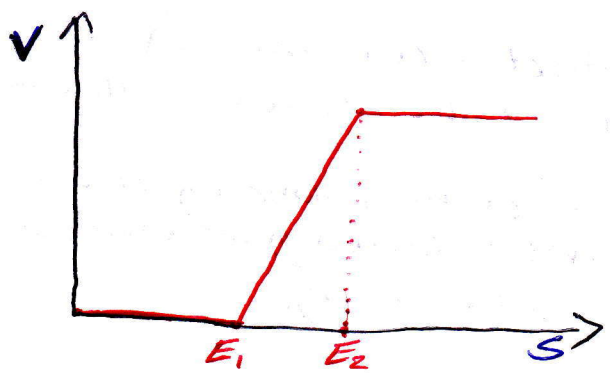
- European Put

  $V(S,T) = \max(E-S,0)$



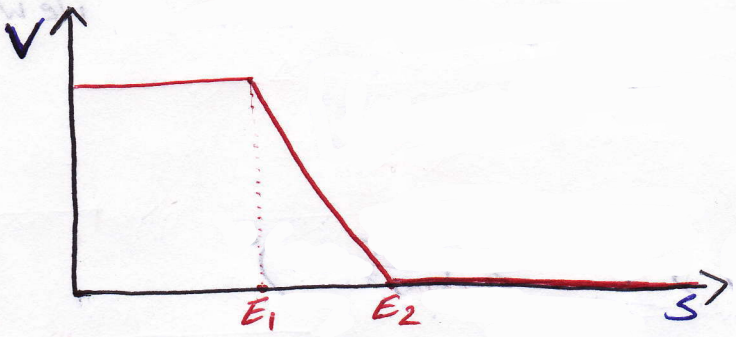
- Bull Spread

  $V(S,T) = \max(S-E_1,0) - \max(S-E_2,0)$

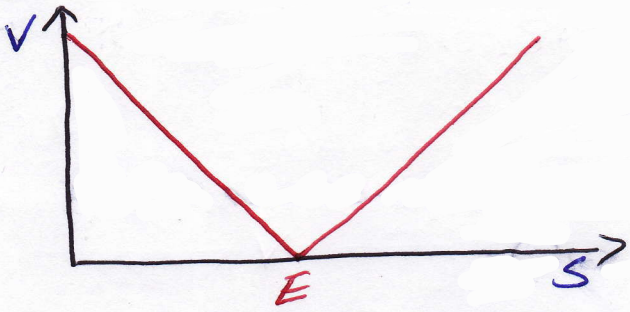


- **Bear Spread**

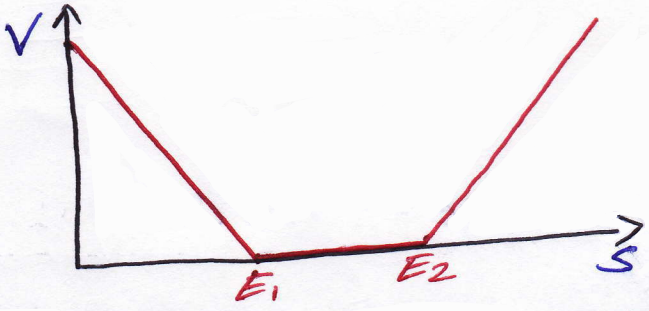
$V(S,T) = \max(E_1 - S, 0) - \max(E_2 - S, 0)$



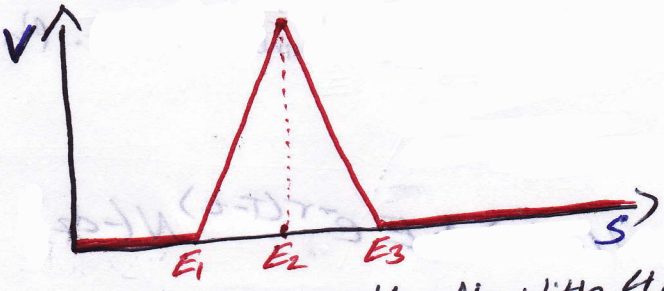
- **Straddle: A call & a Put with same strike Price.**



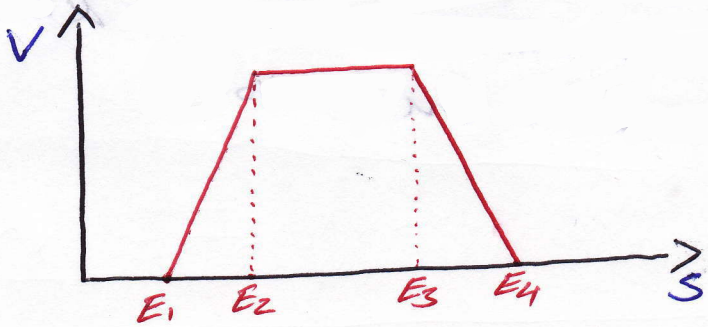
- **Strangle: A call & Put with different strike Price**



- **Butterfly: Involves purchase & sale of Options with 3 different expiries/strike prices.**



- **Condor: Like a Butterfly with 4 different strikes/expiries.**



Transformation of Black-Scholes eqn to Diffusion Eqn & Analytic Soln

We can transform the basic Black-Scholes eqn (σ, r being constant) into something simpler by change of variables. If we write:

$$V(S,t) = e^{\alpha x + \beta \tau} u(x,\tau)$$

where,

$$\alpha = -\frac{1}{2}\left(\frac{2r}{\sigma^2} - 1\right) \; ; \; \beta = -\frac{1}{4}\left(\frac{2r}{\sigma^2} + 1\right)^2$$

$$S = e^x \quad ; \quad t = T - \frac{2\tau}{\sigma^2}$$

Then, $u(x,\tau)$ satisfies the basic diffusion eqn:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}$$

We thus have a soln for basic Black-Scholes eqn for European Options (Analytic soln):

$$V(S,t) = \frac{e^{-r(T-t)}}{\sqrt{2\pi(T-t)}\sigma} \int_0^\infty e^{-\frac{(\log(S/S') + (r - 1/2\sigma^2)(T-t))^2}{2\sigma^2(T-t)}} \text{Payoff}(S') \frac{dS'}{S'}$$

For example,

- A European call option value is:

$$V_c(S,t) = SN(d_1) - Ee^{-r(T-t)}N(d_2)$$

where,

$$d_1 = \frac{\log(S/E) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \; ; \; d_2 = \frac{\log(S/E) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}$$

& $N(d) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{d} e^{-\frac{1}{2}x^2} dx$ is the cumulative dist. fn for Normal dist.

- A European Put option value is:

$$V_p(S,t) = -SN(-d_1) + Ee^{-r(T-t)}N(-d_2)$$

[Note: If σ & r are not constant or if we add other features to B-S Eqn, then analytic soln will be too complex & we require Numerical Method to calculate the solution].

Analysis of the Greeks

Sensitivity of option to different variables & parameters. These are given by the greeks.

(i) **Delta**: $\Delta = \frac{\partial V}{\partial S}$. This is the partial derivative of option wrt asset. Measures how sensitive is option wrt the asset price. This appears in **Delta Hedging** which means holding one of the option in long position & shorting a quantity Δ of the underlyings. Δ varies as S & t varies (dynamic hedging). If Δ is too big the Delta hedging World collapse.

(ii) **Gamma**: $\Gamma = \frac{\partial^2 V}{\partial S^2}$. Since, Γ is the sensitivity of the Δ to the underlying it is a measure of how often a position must be rehedged in order to maintain a delta neutral position. The hedging requirement is decreased by a **Γ-neutral strategy**.

(iii) **Theta**: $\Theta = \frac{\partial V}{\partial t}$. Appears in B-S eqn that contributes to ensure that a Δ hedged position earns the **risk-free rate**.

(iv) **Vega**: $\text{vega} = \frac{\partial V}{\partial \sigma}$. This shows the sensitivity to Volatility. This is different (not even a Greek letter) since it is the derivative wrt a parameter (which is not known accurately) & not a variable. We can also **Vega hedge** to reduce sensitivity to Volalility which might be the major step to reduce model risk as volatility arises in the log normal hypothesis describing the asset.

(V) **Rho**: $\rho = \frac{\partial V}{\partial r}$. This shows the sensitivity to the interest rate. One often uses a whole term structure of interest rate.

Extensions of Black-Scholes Eqn

(1) Options on Dividend-paying equities:

Assuming the asset receives a constant dividend yield D, that is in a time dt, each asset receives an amount $D S dt$.

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-D) S \frac{\partial V}{\partial S} - rV = 0$$

(2) Currency Options:

In holding the foreign currency, we receive interest at the foreign rate of interest r_f.

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-r_f) S \frac{\partial V}{\partial S} - rV = 0$$

(3) Commodity Options:

One key feature of commodities is that they have a cost of carry. Let's call q the fraction of the value of the commodity that goes to pay the cost of carry. This is just a -ve dividend.

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r+q) S \frac{\partial V}{\partial S} - rV = 0$$

(4) Options on Futures:

The future price F, of a non-dividend paying equity is related to the spot price by $F = e^{r(T-s)} S$.

We can easily change variables $V(S,t) = U(F,t)$ to get:

$$\frac{\partial U}{\partial t} + \frac{1}{2}\sigma^2 F^2 \frac{\partial^2 U}{\partial F^2} - rU = 0$$

(5) Options on time-dependent Parameters:

The parameters are known fn of time.

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma(t)^2 S^2 \frac{\partial^2 V}{\partial S^2} + [r(t) - D(t)] S \frac{\partial V}{\partial S} - r(t) V = 0$$

(6) Power Options:

An Option with a payoff that depends on the asset price at expiry raised to some power α. That is if $P = S^\alpha$, we can write:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\alpha^2 \sigma^2 P^2 \frac{\partial^2 V}{\partial P^2} + \alpha\left[\frac{1}{2}\sigma^2(\alpha-1) + r\right] P \frac{\partial V}{\partial P} - rV = 0$$

⑦ Two-factor Options:

When interest rates are also stochastic, $dr = u(r,t)dt + w(r,t)dX$, but not correlated to the asset, the value of the option is determined by:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + \frac{1}{2} w^2 \frac{\partial^2 V}{\partial r^2} + rS\frac{\partial V}{\partial S} + (u - \lambda w)\frac{\partial V}{\partial r} - r(t)V = 0$$

Here, $V(S,r,t)$ is f'n of 3 variables.

⑧ Early-Exercise Options:

These options can be exercised early, prior to expiry like American options, Bermuda etc.

And many other derivatives.

Finite Difference Representations for Black-Scholes Eq^n

Closed-form Soln's for values of options are very rare, most problem require numerical approach. Even though implicit FD method comes with extra complexity compared to Explicit FD method, it has superior stability properties.

- Boundary Conditions: As B-S needs to be solved in $0 \le S < \infty$, $S_{max} = N_S \Delta S$ will be our approximation to ∞. The two Boundary condition will be specified at $S=0$ & $S = S_{max}$ $\forall t$.

$$S_n = n\Delta S \text{ with } n = 0, 1, \ldots, N_S.$$

- Final → Initial Condition: Since, we are given Final Condition i.e. Payoff at Maturity, We need a change of variable from t to τ such that we change the direction of time & solve B-S backwards in time going from maturity to initial time of contract.

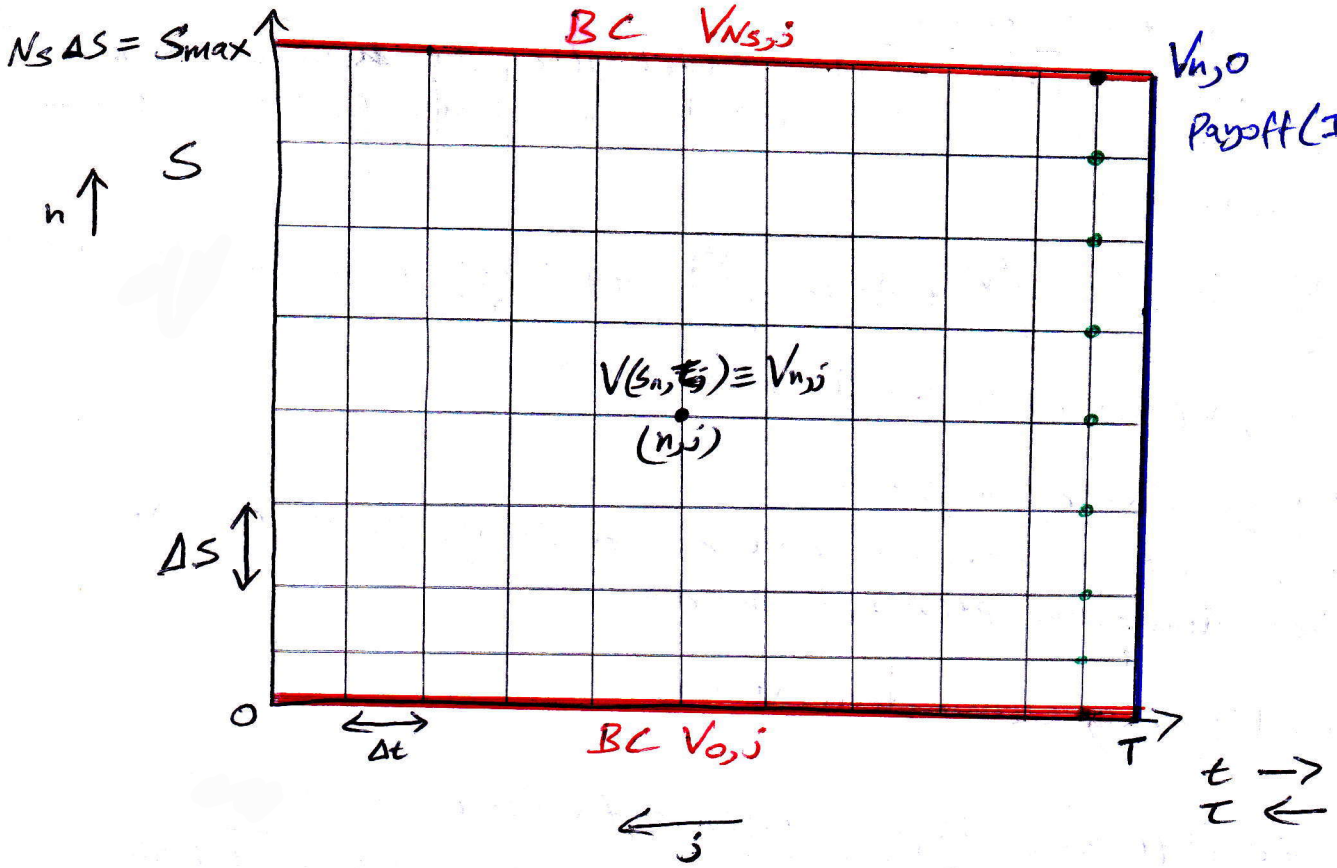
$$\tau = T - t$$

$$\tau_j = j\Delta t, \; j = 0, 1, \ldots, N_t \; ; \; \Delta t = \frac{T}{N_t}$$

In the τ variable, we have changed the direction of time, as j increases time decreases.

With this variable change from t to τ, the $\frac{\partial V}{\partial t} = -\frac{\partial V}{\partial \tau}$ & the Original B-S eq^n: $\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$

becomes: $\boxed{\frac{\partial V}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV}$



We start at $V_{n,0}$ (IC) extreme right & calculate backwards the first column to be calculated is the green dots.

Explicit FTCS method for Black-Scholes Eqn

We are going to use our experience from solving Advection & diffusion eqn & apply it to B-S eqn. Lets rewrite the transformed B-S eqn in τ variable & find the finite difference approximations

$$\left[\frac{\partial V}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV\right] \quad -(*)$$

$\frac{1}{2}\sigma^2 S^2 = a(S,\tau)$, $rS = b(S,\tau)$, $-r = c(S,\tau)$

$\frac{\partial V}{\partial \tau}$ → same as that for Advection/diffusion eqn explicit method → $\frac{V_{n,j+1} - V_{n,j}}{\Delta t} + O(\Delta t)$

$\frac{\partial^2 V}{\partial S^2}$ → same as explicit method for Diffusion eqn → $\frac{V_{n+1,j} - 2V_{n,j} + V_{n-1,j}}{\Delta S^2} + O(\Delta S^2)$

$\frac{\partial V}{\partial S}$ → same as explicit Method for Advection eqn → $\frac{V_{n+1,j} - V_{n-1,j}}{2\Delta S} + O(\Delta S^2)$

$$a(S,\tau) = \frac{1}{2}\sigma^2 S^2$$

$$b(S,\tau) = rS$$

$$c(S,\tau) = -r$$

Substituting the Finite difference approximation for the derivative into the B-S eqn (*), we get:

$$\frac{V_{n,j+1} - V_{n,j}}{\Delta t} = a_{n,j}\frac{V_{n+1,j} - 2V_{n,j} + V_{n-1,j}}{\Delta S^2} + b_{n,j}\frac{V_{n+1,j} - V_{n-1,j}}{2\Delta S} + c_{n,j}V_{n,j} + O(\Delta t, \Delta S^2)$$

With $a_{n,j} = a(S_n, \tau_j)$
$b_{n,j} = b(S_n, \tau_j)$
$c_{n,j} = c(S_n, \tau_j)$

Making $V_{n,j+1}$ the subject,

$$V_{n,j+1} = (\lambda_1 a_{n,j} - \tfrac{1}{2}\lambda_2 b_{n,j})V_{n-1,j} + (1 - 2\lambda_1 a_{n,j} + \Delta t c_{n,j})V_{n,j} + (\lambda_1 a_{n,j} + \tfrac{1}{2}\lambda_2 b_{n,j})V_{n+1,j} + O(\Delta t^2, \Delta t \Delta S^2)$$

With $\lambda_1 = \frac{\Delta t}{\Delta S^2}$; $\lambda_2 = \frac{\Delta t}{\Delta S}$

Writing $S_{n,j} = n\Delta S$ & substituting $a_{n,j}$, $b_{n,j}$ & $c_{n,j}$ we get,

$$V_{n,j+1} = \tfrac{1}{2}(\sigma^2 n^2 - rn)\Delta t\, V_{n-1,j} + [1 - (\sigma^2 n^2 + r)\Delta t]V_{n,j} + \tfrac{1}{2}(\sigma^2 n^2 + rn)\Delta t\, V_{n+1,j} + O(\Delta t^2, \Delta t \Delta S^2)$$

This equation holds for $n = 1, \ldots, N_S - 1$, since, $V_{-1,j}$ and $V_{N_S+1,j}$ are not defined. Thus, there are $N_S - 1$ eqns for $N_S + 1$ unknowns. The remaining two eqn's comes from the two Boundary conditions at $n = 0$ & $n = N_S$, which are treated separately.

Examples of Boundary Conditions for Black Scholes Eqn

- Boundary Conditions for a Call Option

  → At $S = 0$, we know the option value is 0 (worthless).
  Therefore, $V_{0,j} = 0 \quad \forall j$ ← BC

  → For large S i.e. S_{max}, the call option value goes to $S_{max} - Ee^{-r(T-t)}$ (T = τ_j). Thus our upper BC would be

$$V_{N_S,j} = \underbrace{N_S \Delta S}_{S_{max}} - Ee^{-r j \Delta t}$$ ← BC

with $\tau_j = j\Delta t$.

- **Boundary Conditions for a Put Option**

→ At $S=0$, we have the condition that $V(0,t) = Ee^{-r(T-t)}$ ($T-t = \tau_j$)

Which becomes $$\boxed{V_{0,j} = Ee^{-r\tau_j \Delta t}} \leftarrow BC$$

↑ discounted strike price

→ The Put option becomes worthless for large S (S_{max}).

So, $$\boxed{V_{N_S,j} = 0 \quad \forall j} \leftarrow BC$$

- **Boundary Conditions at $S=0$ for most Contracts**

→ A useful Boundary condition to apply at $S=0$ for most contracts (including Calls & Puts) is that the diffusion and drift terms switch off i.e.:

$$\frac{\partial V}{\partial \tau} = \underbrace{\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}}_{=0} + \underbrace{rS\frac{\partial V}{\partial S}}_{=0} - rV$$

This means that on $S=0$, the payoff is guaranteed, resulting in the condition:

$$\frac{\partial V}{\partial \tau}(0,\tau) = -rV(0,\tau)$$

Applying Finite difference approximation to this gives:

$$\frac{V_{0,j+1} - V_{0,j}}{\Delta t} = -rV_{0,j}$$

$$\boxed{V_{0,j+1} = (1 - r\Delta t)V_{0,j}} \leftarrow BC$$

- **Boundary Condition at S_{max} for Most Contracts**

→ This BC can be applied to most options with linear behaviours. eg. for call option

When the option has a payoff that is almost linear in the underlying for large values of S, then you can use the following upper BC: [Note: 2nd derivative measures curvature]

$$\frac{\partial^2 V}{\partial S^2}(S,\tau) = 0 \text{ as } S \to \infty \implies \frac{V_{N_S,j} - 2V_{N_S-1,j} + V_{N_S-2,j}}{\Delta S^2} = 0$$

$$\boxed{V_{N_S,j} = 2V_{N_S-1,j} - V_{N_S-2,j}} \leftarrow BC$$

Fully Implicit Method for Black Scholes Eqn

$$\frac{\partial V}{\partial \tau} = \underbrace{\frac{1}{2}\sigma^2 S^2}_{a(S,\tau)} \frac{\partial^2 V}{\partial S^2} + \underbrace{rS}_{b(S,\tau)} \frac{\partial V}{\partial S} - \underbrace{rV}_{c(S,\tau)}$$

$a(S,\tau) = \frac{1}{2}\sigma^2 S^2$
$b(S,\tau) = rS$
$c(S,\tau) = -r$

This is similar to Explicit method except everything on RHS of above eqn will be evaluated at time $j+1$.

$$\frac{\partial V}{\partial \tau} \longrightarrow \frac{V_{n,j+1} - V_{n,j}}{\Delta t} + O(\Delta t)$$

$$\frac{\partial^2 V}{\partial S^2} \longrightarrow \frac{V_{n+1,j+1} - 2V_{n,j+1} + V_{n-1,j+1}}{\Delta S^2} + O(\Delta S^2)$$

$$\frac{\partial V}{\partial S} \longrightarrow \frac{V_{n+1,j+1} - V_{n-1,j+1}}{2\Delta S} + O(\Delta S^2)$$

$a_{n,j+1} \longrightarrow a(S_n, \tau_{j+1})$
$b_{n,j+1} \longrightarrow b(S_n, \tau_{j+1})$
$c_{n,j+1} \longrightarrow c(S_n, \tau_{j+1})$

Substituting them into the above equation gives the Finite Difference approximation as:

$$\frac{V_{n,j+1} - V_{n,j}}{\Delta t} = a_{n,j+1} \frac{V_{n+1,j+1} - 2V_{n,j+1} + V_{n-1,j+1}}{\Delta S^2} + b_{n,j+1} \frac{V_{n+1,j+1} - V_{n-1,j+1}}{2\Delta S} + c_{n,j+1} V_{n,j+1} + O(\Delta t, \Delta S^2)$$

$$V_{n,j} = \left(\frac{1}{2}\lambda_2 b_{n,j+1} - \lambda_1 a_{n,j+1}\right) V_{n-1,j+1} + \left(1 + 2\lambda_1 a_{n,j+1} - \Delta t\, c_{n,j+1}\right) V_{n,j+1} - \left(\lambda_1 a_{n,j+1} + \frac{1}{2}\lambda_2 b_{n,j+1}\right) V_{n+1,j+1} + O(\Delta t^2, \Delta t \Delta S^2)$$

where $\lambda_1 = \frac{\Delta t}{\Delta S^2}$ & $\lambda_2 = \frac{\Delta t}{\Delta S}$

Writing $S_{n,j} = n\Delta S$ & substituting $\begin{cases} a_{n,j+1} = \frac{1}{2}\sigma^2 n^2 \Delta S^2 \\ b_{n,j+1} = rn\Delta S \\ c_{n,j+1} = -r \end{cases}$ we get:

$$\frac{1}{2}(rn - \sigma^2 n^2)\Delta t\, V_{n-1,j+1} + \left[1 + (\sigma^2 n^2 + r)\Delta t\right] V_{n,j+1} - \frac{1}{2}(\sigma^2 n^2 + rn)\Delta t\, V_{n+1,j+1} = V_{n,j}$$

This equation holds for $n = 1, \ldots, N_S - 1$ since $V_{-1,j+1}$ & $V_{N_S+1,j+1}$ are not defined. Thus, there are $N_S - 1$ equations for $N_S + 1$ unknowns. But, as we already know, there is a huge difference between explicit and implicit methods. Stability is highly improved for implicit method while the solution procedure is no longer straightforward.

Matrix Representation of the Fully Implicit Method for B-S eqn.

Writing $A_{n,j} = \frac{1}{2}(rn - \sigma^2 n^2)\Delta t = \frac{1}{2}\lambda_2 b_{n,j} - \lambda_1 a_{n,j}$

$B_{n,j} = (\sigma^2 n^2 + r)\Delta t = 2\lambda_1 a_{n,j} - \Delta t\, c_{n,j}$

$C_{n,j} = -\frac{1}{2}(\sigma^2 n^2 + rn)\Delta t = -\frac{1}{2}\lambda_2 b_{n,j} - \lambda_1 a_{n,j}$

The implicit method can be rewritten as:

$$A_{n,j+1} V_{n-1,j+1} + (1 + B_{n,j+1}) V_{n,j+1} + C_{n,j+1} V_{n+1,j+1} = V_{n,j}$$

for $n = 1, \dots, N_S - 1$.

The matrix representation for this can be written as:

$$\begin{bmatrix} A_{1,j+1} & (1+B_{1,j+1}) & C_{1,j+1} & 0 & \cdots & 0 \\ 0 & A_{2,j+1} & (1+B_{2,j+1}) & C_{2,j+1} & 0 & \cdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & & A_{N_S-1,j+1} & (1+B_{N_S-1,j+1}) & C_{N_S-1,j+1} \end{bmatrix} \begin{bmatrix} V_{0,j+1} \\ V_{1,j+1} \\ V_{2,j+1} \\ \vdots \\ V_{N_S-1,j+1} \\ V_{N_S,j+1} \end{bmatrix} = \begin{bmatrix} V_{1,j} \\ V_{2,j} \\ \vdots \\ V_{N_S-1,j} \end{bmatrix}$$

$(N_S - 1, N_S + 1)$ $\quad (N_S + 1, 1)$ $\quad (N_S - 1, 1)$

The matrix has $N_S - 1$ rows & $N_S + 1$ columns, which is a representation of the $N_S - 1$ eqns & $N_S + 1$ unknowns. The two eqns that we are missing come from the Boundary conditions. Using these conditions, we are going to convert this system of eqns into a system of eqns involving a square Matrix. The aim is to write a system of eqns in the form:

$$M^{\angle}_{j+1} V_{j+1} + Y^{\angle}_{j+1} = V_j$$

$$V_{j+1} = (M^{\angle}_{j+1})^{-1}(V_j - Y^{\angle}_{j+1})$$

For known square matrix $M^{\angle}_{j+1}$ & known vector $Y^{\angle}_{j+1}$, where the details of the Boundary conditions have been fully incorporated.

Implementation of Boundary Conditions in the Matrix Representation of Fully Implicit Method for B-S eqn (Dirichlet BC)

In Dirichlet BC, we are given $V_{0,j}$ & $V_{N_S,j}$ $\forall j$. Sometimes we know that the option has a particular value on the boundary, $n = 0$ & $n = N_S$. For example, if we have an European call, we know that for $S = 0$, we have $V(0,t) = 0$ & for $S = S_{max}$ we have $V(S_{max}, t) = S_{max} - Ee^{-r(T-t)}$. So, the two BC's can be written as

$$V_{0,j} = 0 \quad \& \quad V_{N_S,j} = N_S \Delta S - E e^{-rj\Delta t}$$

We need to incorporate the Boundary condition into the matrix representation & convert into square matrix form.
The strategy is same as before, we will omit 1st row & last column and omit 1st & last element of the $j+1$ vector as values of $V_{0,j+1}$ & $V_{N_S,j+1}$ are known through boundary condition. One thing we must include is an additional $Y^{\leftarrow}_{j+1}$ vector that would include the the contribution of $(A_{1,j+1}) \times (V_{0,j+1})$ & $(C_{N_S-1,j+1}) \times (V_{N_S,j+1})$.
The Matrix representation now becomes:

$$\begin{bmatrix} (1+B_{1,j+1}) & C_{1,j+1} & 0 & \cdots & 0 \\ A_{2,j+1} & (1+B_{2,j+1}) & C_{2,j+1} & & \\ 0 & \ddots & \ddots & \ddots & \\ & & A_{N_S-2,j+1} & (1+B_{N_S-2,j+1}) & C_{N_S-2,j+1} \\ 0 & \cdots & 0 & A_{N_S-1,j+1} & (1+B_{N_S-1,j+1}) \end{bmatrix} \begin{bmatrix} V_{1,j+1} \\ V_{2,j+1} \\ \vdots \\ V_{N_S-1,j+1} \end{bmatrix} + \begin{bmatrix} A_{1,j+1}V_{0,j+1} \\ 0 \\ \vdots \\ C_{N_S-1,j+1}V_{N_S,j+1} \end{bmatrix} = \begin{bmatrix} V_{1,j} \\ V_{2,j} \\ \vdots \\ V_{N_S-1,j} \end{bmatrix}$$

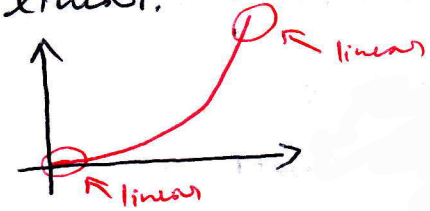
(N_S-1, N_S-1) $(N_S-1, 1)$ $(N_S-1, 1)$ $(N_S-1, 1)$

$M^{\leftarrow}_{j+1}$ V_{j+1} + $Y^{\leftarrow}_{j+1}$ = V_j

$$\Rightarrow V_{j+1} = (M^{\leftarrow}_{j+1})^{-1}(V_j - Y^{\leftarrow}_{j+1})$$

Implementation of the Boundary conditions in the Matrix Representation of Fully Implicit method for the B-S eqn (Neumann type)

This is similar to Neumann type except we consider 2nd derivative instead of first. In this BC, we are given $\frac{\partial^2 V}{\partial S^2} = 0 \ \forall t$ at $S=0$ & $S = S_{max}$, That is, for small & large S values the curve is almost linear.



This condition is particularly useful since it's independant of the type of the contract, as long as the contract has a payoff that is almost linear in the underlying. This condition can be approximated using finite difference (central difference):

- $S=0$

$$\frac{\partial^2 V}{\partial S^2} = 0 \longrightarrow \frac{V_{2,j} - 2V_{1,j} + V_{0,j}}{\Delta S^2} = 0 \Rightarrow \boxed{V_{0,j} = 2V_{1,j} - V_{2,j}}$$

- $S = S_{max}$

$$\frac{\partial^2 V}{\partial S^2} = 0 \longrightarrow \frac{V_{N_S,j} - 2V_{N_S-1,j} + V_{N_S-2,j}}{\Delta S^2} = 0$$

$$\Rightarrow \boxed{V_{N_S,j} = 2V_{N_S-1,j} - V_{N_S-2,j}}$$

We will now incorporate the BC into the matrix representation & change the matrix into Square Matrix. We will use same strategy as before for Neumann BC for Diffusion eqn. We will omit 1st & last column of the matrix so that it is a Sq. Matrix & since we know $V_{0,j+1}$ & $V_{N_S,j+1}$, we will omit them from the vector on LHS.

Since, $V_{0,j+1}$ depends on $V_{1,j+1}$ & $V_{2,j+1}$ and $V_{N_S,j}$ depends on $V_{N_S-1,j+1}$ & $V_{N_S-2,j+1}$ from the BC's, we need to be careful on the contribution of product of $(A_{1,j+1}) \times (V_{0,j+1})$ and $(C_{N_S-1,j+1}) \times (V_{N_S,j+1})$.

$$(A_{1,j+1}) \times (V_{0,j+1}) = A_{1,j+1}(2V_{1,j+1} - V_{2,j+1}) = 2A_{1,j+1} V_{1,j+1} - A_{1,j+1} V_{2,j+1}$$

$$(C_{N_S-1,j+1}) \times (V_{N_S,j+1}) = C_{N_S-1,j+1}(2V_{N_S-1,j+1} - V_{N_S-2,j+1})$$

$$= 2C_{N_S-1,j+1} V_{N_S-1,j+1} - C_{N_S-1,j+1} V_{N_S-2,j+1}$$

We thus need to make 4 changes to the new matrix elements:

$$M_{11} = 1 + B_{1,j+1} + 2A_{1,j+1}$$

$$M_{12} = C_{1,j+1} - A_{1,j+1}$$

$$M_{N_S-1,N_S-1} = 1 + B_{N_S-1,j+1} + 2C_{N_S-1,j+1}$$

$$M_{N_S-1,N_S-2} = A_{N_S-1,j+1} - C_{N_S-1,j+1}$$

The Matrix Representation now becomes:

$$\begin{bmatrix} (1+B_{1,j+1}+2A_{1,j+1}) & (C_{1,j+1}-A_{1,j+1}) & 0 & \cdots & 0 \\ A_{2,j+1} & (1+B_{2,j+1}) & C_{2,j+1} & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & A_{N_S-2,j+1} & (1+B_{N_S-2,j+1}) & C_{N_S-2,j+1} \\ 0 & \cdots & 0 & (A_{N_S-1,j+1}-C_{N_S-1,j+1}) & (1+B_{N_S-1,j+1}+2C_{N_S-1,j+1}) \end{bmatrix} \begin{bmatrix} V_{1,j+1} \\ V_{2,j+1} \\ \vdots \\ V_{N_S-1,j+1} \end{bmatrix} = \begin{bmatrix} V_{1,j} \\ V_{2,j} \\ \vdots \\ V_{N_S-1,j} \end{bmatrix}$$

M_{j+1}^{L} $\quad V_{j+1}$ $\quad V_j$

(N_S-1, N_S-1) $\quad (N_S-1, 1)$ $\quad (N_S-1, 1)$

$$M_{j+1}^{L} \; V_{j+1} = V_j$$

Therefore, $\boxed{V_{j+1} = (M_{j+1}^{L})^{-1} V_j}$

Note: $Y_{j+1}^{L} = 0$

The Crank-Nicholson Method for B-S Eqn

$$\left[\frac{\partial V}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV\right]$$

We apply same strategy as that of diffusion eqn. This is an average of the fully Implicit ($i+1$ step) & the explicit (i step) methods. This uses 6 points. So, the above eqn is approximated as:

$$\frac{V_{n,j+1} - V_{n,j}}{\Delta t} = \frac{1}{2}\left[a_{n,j+1}\frac{V_{n+1,j+1} - 2V_{n,j+1} + V_{n-1,j+1}}{\Delta S^2} + b_{n,j+1}\frac{V_{n+1,j+1} - V_{n-1,j+1}}{2\Delta S} + c_{n,j+1}V_{n,j+1}\right] + \frac{1}{2}\left[a_{n,j}\frac{V_{n+1,j} - 2V_{n,j} + V_{n-1,j}}{\Delta S^2} + b_{n,j}\frac{V_{n+1,j} - V_{n-1,j}}{2\Delta S} + c_{n,j}V_{n,j}\right] + O(\Delta t, \Delta S^2)$$

(Implicit: $a_{n,j+1}$, $b_{n,j+1}$, $c_{n,j+1}$; Explicit: $a_{n,j}$, $b_{n,j}$, $c_{n,j}$)

Where, $a_{n,j+1} = \frac{1}{2}\sigma^2 S_{n,j+1}$; $b_{n,j+1} = rS_{N,j+1}$; $c_{n,j+1} = -r$

$a_{n,j} = \frac{1}{2}\sigma^2 S_{n,j}$; $b_{n,j} = rS_{n,j}$; $c_{n,j} = -r$

Rewriting everything that depends on $j+1$ on RHS & everything that depends on j on LHS:

$$\left[A_{n,j+1}V_{n-1,j+1} + (1+B_{n,j+1})V_{n,j+1} + C_{n,j+1}V_{n-1,j+1} = -A_{n,j}V_{n-1,j} + (1-B_{n,j})V_{n,j} - C_{n,j}V_{n-1,j}\right] \quad -(*)$$

With, $A_{n,j} = \frac{1}{4}\lambda_2 b_{n,j} - \frac{1}{2}\lambda_1 a_{n,j}$

$B_{n,j} = \lambda_1 a_{n,j} - \frac{1}{2}\Delta t\, c_{n,j}$

$C_{n,j} = -\frac{1}{2}\lambda_1 a_{n,j} - \frac{1}{4}\lambda_2 b_{n,j}$

Where $\lambda_1 = \frac{\Delta t}{\Delta S^2}$; $\lambda_2 = \frac{\Delta t}{\Delta S}$

These equations goes from $1 \le n \le N_S - 1$ & the BC's again supply the two missing eqn's.
Note, the Crank-Nicholson Method is a implicit method, so we required a matrix representation to solve it.

Matrix Representation of the Crank-Nicholson Method for B-S Eqn

We can represent the above Finite difference equation $(*)$ in a Matrix equation/form. Note, this time we will have two matrices, one on LHS & one on RHS of the equation.

$$A\,V_{j+1} = B\,V_j$$

(A: Non-Sq Matrix; B: Non-Square Matrix)

$$\begin{bmatrix} A_{1,j+1}(1+B_{1,j+1}) & C_{1,j+1} & 0 & \cdots & 0 \\ 0 & A_{2,j+1} & (1+B_{2,j+1}) & C_{2,j+1} & \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ & & A_{N_S-2,j+1} & (1+B_{N_S-2,j+1}) & C_{N_S-2,j+1} \\ 0 & \cdots & & A_{N_S-1,j+1} & (1+B_{N_S-1,j+1}) \end{bmatrix} \begin{bmatrix} V_{0,j+1} \\ V_{1,j+1} \\ V_{2,j+1} \\ \\ \\ V_{N_S-1,j+1} \\ V_{N_S,j+1} \end{bmatrix}$$

$$= \begin{bmatrix} -A_{1,j} & (1-B_{1,j}) & -C_{1,j} & 0 & \cdots & 0 \\ 0 & -A_{2,j} & (1-B_{2,j}) & -C_{2,j} & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & -A_{N_S-2,j} & (1-B_{N_S-2,j}) & -C_{N_S-2,j} \\ 0 & \cdots & & 0 & -A_{N_S-1,j} & (1-B_{N_S-1,j}) \end{bmatrix} \begin{bmatrix} V_{0,j} \\ V_{1,j} \\ V_{2,j} \\ \vdots \\ \\ V_{N_S-1,j} \\ V_{N_S,j} \end{bmatrix}$$

This shows the Matrix representation of the Crank-Nicholson Scheme for BS Eqn. The matrices have (N_S-1) rows & (N_S+1) columns, which is a representation of the (N_S-1) eqns & (N_S+1) unknowns. The two missing eqns come from the BC's.
Using the BC's, we are going to convert this system of eqns into a system of eqns involving square matrices. The aim is to write a system of eqns in the form:

$$M^L_{j+1} V_{j+1} + Y^L_{j+1} = M^R_j V_j + Z^R_j$$

Where the matrices are square matrices & vectors Y^L_{j+1} & Z^R_j incorporate the details of Boundary conditions. We then solve this matrix eqn for V_{j+1}.

$$V_{j+1} = (M^L_{j+1})^{-1} \left[M^R_j V_j + Z^R_j - Y^L_{j+1} \right]$$

The implementation of BC are same as that for fully implicit Method.

Free Boundary Problems (American and Bermudean Options)

- American Option

The main difficulty in pricing an American option is the existence of a free Boundary due to the possibility of early exercise right. There are no exact formulas available even for simple case. American option is like European option except the option can be exercised at any time. So, the payoff for American call option for example is:

$$\text{Payoff} = \max(S-E, 0) \quad \forall t.$$

So, to price an American option we require Numerical methods.

The key thing when pricing American option numerically is to avoid Arbitrage, so this means the option value at each point cannot be less than the immediate payoff if the option is exercised. Therefore,

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

- Bermudean Option

In this option, early exercise is only allowed on certain dates. So, the Payoff function is also time dependent. The Payoff f'n of this option is zero except on the special dates when it is some prescribed f'n on the underlying. For eg. A Payoff of Bermudean call option will be:

$$\text{Payoff} = \begin{cases} 0 & \text{when } t = t_0, t_2, t_4 \dots \\ \max(S-E, 0) & \text{when } t = t_1, t_3, t_5 \dots \end{cases}$$

So, the general Payoff f'n depends both on S & t.

$$V(S,t) \geq \text{Payoff}(S,t) \quad \text{for No Arbitrage.}$$

Implementing Finite-Difference Representation for BS eqns with Free Boundaries

For Free Boundary Problems, we need to satisfy

$$V(S,t) \geq \text{Payoff}(S,t) \quad — (*)$$

for No arbitrage.

Suppose that we have found $V_{n,j}$ for all n at time step j & we want to proceed to find the option value at time $j+1$ by using a Finite-Difference scheme. Therefore, in order to implement the possibility of Early Exercise we do as follows:

- Don't worry about whether or not you have violated Option Constraint (*) above until you have found the option values $V_{n,j+1}$ for all n.
- Now, check whether the new option values are greater or equal to the Payoff (i.e. the option Constraint (*)).
- If they are less than the payoff then we have an arbitrage. We cannot allow for arbitrage, so, at every value of n for which the option value has an arbitrage, replace the option value calculated with the payoff at that asset value (Payoff(S,t)).

[Note: For American Option, we need to check the option Constraint for any time. For Bermudean Option, we need to do the check for only allowed Exercise dates.]