MATH97110: Numerical Methods for Finance

Topic 3: Introduction to Finite Difference Methods

Imperial College London

Spring 2022

Overview

1. Connection between probabilistic expectation and partial differential equation
2. Finite difference methods for differentiation
3. Finite difference methods for PDE
   - Explicit scheme
   - Implicit scheme
   - θ-scheme
4. Convergence of finite difference methods
   - Local truncation errors
   - Stability analysis

The Feynman-Kac formula

- There is a deep connection between partial differential equation (PDE) and expectation of stochastic process driven by Brownian motion $B = (B_t)_{t \geq 0}$

Proposition 3.1 (Feynman-Kac)

Suppose $V = V(t,x)$ is the solution to the PDE

$$\begin{cases} \frac{\partial V}{\partial t} + b(t,x)\frac{\partial V}{\partial x} + \frac{1}{2}\sigma^2(t,x)\frac{\partial^2 V}{\partial x^2} = r(t,x)V, & t < T; \\ V(T,x) = g(x), & t = T. \end{cases} \tag{3.1}$$

Then subject to some suitable regularity conditions on V, r, b and σ, we have

$$V(t,x) = \mathbb{E}^{(t,x)}\left[e^{-\int_t^T r(u,X_u)du} g(X_T)\right] \tag{3.2}$$

where $X = (X_s)_{s \in [t,T]}$ is the solution to the SDE

$$dX_s = b(s,X_s)ds + \sigma(s,X_s)dB_s, \quad X_t = x.$$

Remark: $\mathbb{E}^{(t,x)}[\cdot]$ denotes the conditional expectation operator given the information up to time t with $X_t = x$.

Sketch of proof of Prop 3.1

Consider t as fixed. For $s \in [t, T]$, write $D_s := e^{-\int_t^s r(u,X_u)du}$ and apply Ito's lemma to the process $M_s := e^{-\int_t^s r(u,X_u)du} V(s, X_s)$:

$$\begin{aligned} dM_s &= -r(s, X_s)e^{-\int_t^s r(u,X_u)du} V ds \\ &\quad + e^{-\int_t^s r(u,X_u)du} \left(\dot{V} ds + V_x dX_s + \frac{1}{2} V_{xx} (dX_s)^2 \right) \\ &= D_s \left(\dot{V} + b(s, X_s) V_x + \frac{1}{2} \sigma^2(s, X_s) V_{xx} - r(s, X_s) V \right) ds \\ &\quad + D_s V_x \sigma(s, X_s) dB_s \\ &= D_s V_x \sigma(s, X_s) dB_s \end{aligned}$$

and thus $M_T = M_t + \int_t^T D_s V_x \sigma(s, X_s) dB_s$. With some regularity conditions on σ and V_x, M is a true martingale such that

$$\begin{aligned} V(t, x) = M_t = \mathbb{E}^{(t,x)}[M_T] &= \mathbb{E}^{(t,x)} \left[e^{-\int_t^T r(u,X_u)du} V(T, X_T) \right] \\ &= \mathbb{E}^{(t,x)} \left[e^{-\int_t^T r(u,X_u)du} g(X_T) \right]. \end{aligned}$$

The usefulness of Feynman-Kac formula

- We can use probabilistic method to solve a PDE in form of (3.1). For example, Monte Carlo simulation can be used to estimate the expectation on the RHS of (3.2)

The usefulness of Feynman-Kac formula

- We can use probabilistic method to solve a PDE in form of (3.1). For example, Monte Carlo simulation can be used to estimate the expectation on the RHS of (3.2)
- Conversely, we can compute the probabilistic expectation on the RHS of (3.2) by solving the PDE in (3.2), using numerical schemes like finite difference methods

The usefulness of Feynman-Kac formula

- We can use probabilistic method to solve a PDE in form of (3.1). For example, Monte Carlo simulation can be used to estimate the expectation on the RHS of (3.2)
- Conversely, we can compute the probabilistic expectation on the RHS of (3.2) by solving the PDE in (3.2), using numerical schemes like finite difference methods
- This essentially highlights the two mainstream approaches to solve an option pricing problem in continous time

The Black-Scholes PDE

- Under the Black-Scholes model, the stock price process has SDE of

$$dS_t = rS_t dt + \sigma S_t dB_t$$

under the risk neutral measure $\mathbb{Q}$

The Black-Scholes PDE

- Under the Black-Scholes model, the stock price process has SDE of

$$dS_t = rS_t dt + \sigma S_t dB_t$$

under the risk neutral measure $\mathbb{Q}$

- Consider an option which pays $g(S_T)$ at its maturity date T, its time-t fair price is given by

$$V(t,s) := \mathbb{E}_{\mathbb{Q}}^{(t,s)}\left[e^{-r(T-t)}g(S_T)\right]$$

The Black-Scholes PDE

- Under the Black-Scholes model, the stock price process has SDE of

$$dS_t = rS_t dt + \sigma S_t dB_t$$

under the risk neutral measure $\mathbb{Q}$

- Consider an option which pays $g(S_T)$ at its maturity date T, its time-t fair price is given by

$$V(t,s) := \mathbb{E}_{\mathbb{Q}}^{(t,s)}\left[e^{-r(T-t)}g(S_T)\right]$$

- By the Feynman-Kac formula, $V(t,s)$ can be characterised by the solution to the PDE

$$\begin{cases} \frac{\partial V}{\partial t} + rs\frac{\partial V}{\partial s} + \frac{\sigma^2 s^2}{2}\frac{\partial^2 V}{\partial s^2} - rV = 0, & t < T; \\ V(t,s) = g(s), & t = T. \end{cases} \tag{3.3}$$

The above is known as the Black-Scholes PDE

Transformation of the Black-Scholes PDE

- (3.3) is stated as a PDE with a terminal condition but it is more common to work with a PDE propagating forward in time. If we perform a change of coordinate $\tau := T - t$ then the PDE becomes

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{\sigma^2 s^2}{2} \frac{\partial^2 V}{\partial s^2} + rs \frac{\partial V}{\partial s} - rV, & \tau > 0; \\ V(\tau, s) = g(s), & \tau = 0. \end{cases} \tag{3.4}$$

  The terminal condition now becomes an initial condition

Transformation of the Black-Scholes PDE

- (3.3) is stated as a PDE with a terminal condition but it is more common to work with a PDE propagating forward in time. If we perform a change of coordinate $\tau := T - t$ then the PDE becomes

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{\sigma^2 s^2}{2} \frac{\partial^2 V}{\partial s^2} + rs \frac{\partial V}{\partial s} - rV, & \tau > 0; \\ V(\tau, s) = g(s), & \tau = 0. \end{cases} \tag{3.4}$$

  The terminal condition now becomes an initial condition

- If we perform another coordinate transformation via $x := \ln s$, then it can be shown that (exercise)

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{\sigma^2}{2} \frac{\partial^2 V}{\partial x^2} + \left(r - \frac{\sigma^2}{2} \right) \frac{\partial V}{\partial x} - rV, & \tau > 0; \\ V(\tau, x) = g(e^x), & \tau = 0. \end{cases} \tag{3.5}$$

  The coefficients of the partial derivatives are now constants

Transformation of the Black-Scholes PDE

- (3.3) is stated as a PDE with a terminal condition but it is more common to work with a PDE propagating forward in time. If we perform a change of coordinate $\tau := T - t$ then the PDE becomes

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{\sigma^2 s^2}{2} \frac{\partial^2 V}{\partial s^2} + rs \frac{\partial V}{\partial s} - rV, & \tau > 0; \\ V(\tau, s) = g(s), & \tau = 0. \end{cases} \tag{3.4}$$

  The terminal condition now becomes an initial condition

- If we perform another coordinate transformation via $x := \ln s$, then it can be shown that (exercise)

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{\sigma^2}{2} \frac{\partial^2 V}{\partial x^2} + \left(r - \frac{\sigma^2}{2} \right) \frac{\partial V}{\partial x} - rV, & \tau > 0; \\ V(\tau, x) = g(e^x), & \tau = 0. \end{cases} \tag{3.5}$$

  The coefficients of the partial derivatives are now constants

- Under a further transformation, we can show that (exercise again) the PDE can be reduced to a heat equation in form of

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

  with a suitable initial condition

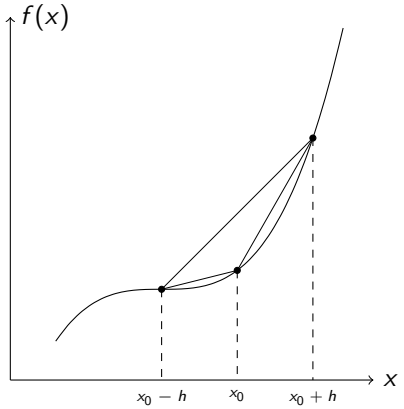
First order derivative approximation via finite difference

- Suppose we want to estimate the value of $f'(x_0)$ given the value of the function at $x_0, x_0 + h$ and $x_0 - h$
- A few possible approaches:

Forward difference: $\dfrac{f(x_0 + h) - f(x_0)}{h}$

Backward difference: $\dfrac{f(x_0) - f(x_0 - h)}{h}$

Central difference: $\dfrac{f(x_0 + h) - f(x_0 - h)}{2h}$



Order of accuracy of finite difference methods

Proposition 3.2

The forward, backward and central difference approximation of $f'(x_0)$ have the following order of accuracy:

Forward difference: $$\frac{f(x_0+h)-f(x_0)}{h} - f'(x_0) = O(h)$$

Backward difference: $$\frac{f(x_0)-f(x_0-h)}{h} - f'(x_0) = O(h)$$

Central difference: $$\frac{f(x_0+h)-f(x_0-h)}{2h} - f'(x_0) = O(h^2)$$

Order of accuracy of finite difference methods

Proposition 3.2

The forward, backward and central difference approximation of $f'(x_0)$ have the following order of accuracy:

$$\text{Forward difference:} \quad \frac{f(x_0+h)-f(x_0)}{h} - f'(x_0) = O(h)$$

$$\text{Backward difference:} \quad \frac{f(x_0)-f(x_0-h)}{h} - f'(x_0) = O(h)$$

$$\text{Central difference:} \quad \frac{f(x_0+h)-f(x_0-h)}{2h} - f'(x_0) = O(h^2)$$

Proof. By Taylor expansion, we have

$$f(x_0+h) = f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2}h^2 + \frac{f'''(x_0)}{6}h^3 + \cdots \tag{3.6}$$

$$f(x_0-h) = f(x_0) - f'(x_0)h + \frac{f''(x_0)}{2}h^2 - \frac{f'''(x_0)}{6}h^3 + \cdots \tag{3.7}$$

which easily establishes the $O(h)$ order of accuracy of the forward and backward difference approximation.

Order of accuracy of finite difference methods

Proposition 3.2

The forward, backward and central difference approximation of $f'(x_0)$ have the following order of accuracy:

$$\text{Forward difference:} \quad \frac{f(x_0+h)-f(x_0)}{h} - f'(x_0) = O(h)$$

$$\text{Backward difference:} \quad \frac{f(x_0)-f(x_0-h)}{h} - f'(x_0) = O(h)$$

$$\text{Central difference:} \quad \frac{f(x_0+h)-f(x_0-h)}{2h} - f'(x_0) = O(h^2)$$

Proof. By Taylor expansion, we have

$$f(x_0+h) = f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2}h^2 + \frac{f'''(x_0)}{6}h^3 + \cdots \tag{3.6}$$

$$f(x_0-h) = f(x_0) - f'(x_0)h + \frac{f''(x_0)}{2}h^2 - \frac{f'''(x_0)}{6}h^3 + \cdots \tag{3.7}$$

which easily establishes the $O(h)$ order of accuracy of the forward and backward difference approximation. Finally, on subtracting (3.6) by (3.7) we get

$$\frac{f(x_0+h)-f(x_0-h)}{2h} - f'(x_0) = \frac{f'''(x_0)}{6}h^2 + \cdots = O(h^2).$$

Second order derivative approximation via finite difference

- Suppose now we want to estimate the value of $f''(x_0)$

Second order derivative approximation via finite difference

- Suppose now we want to estimate the value of $f''(x_0)$
- We apply central difference using the points $x_0 - h/2$ and $x_0 + h/2$ to get

$$f''(x_0) \approx \frac{f'(x_0 + h/2) - f'(x_0 - h/2)}{h} \tag{3.8}$$

Second order derivative approximation via finite difference

- Suppose now we want to estimate the value of $f''(x_0)$
- We apply central difference using the points $x_0 - h/2$ and $x_0 + h/2$ to get

$$f''(x_0) \approx \frac{f'(x_0 + h/2) - f'(x_0 - h/2)}{h} \tag{3.8}$$

- If we apply central difference again to get an estimate of $f'(x_0 + h/2)$ and $f'(x_0 - h/2)$, then

$$f'(x_0 + h/2) \approx \frac{f(x_0 + h) - f(x_0)}{h}, \qquad f'(x_0 - h/2) \approx \frac{f(x_0) - f(x_0 - h)}{h}$$

such that substitution back to (3.8) gives

$$f''(x_0) \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} \tag{3.9}$$

Second order derivative approximation via finite difference

- Suppose now we want to estimate the value of $f''(x_0)$
- We apply central difference using the points $x_0 - h/2$ and $x_0 + h/2$ to get

$$f''(x_0) \approx \frac{f'(x_0 + h/2) - f'(x_0 - h/2)}{h} \tag{3.8}$$

- If we apply central difference again to get an estimate of $f'(x_0 + h/2)$ and $f'(x_0 - h/2)$, then

$$f'(x_0 + h/2) \approx \frac{f(x_0 + h) - f(x_0)}{h}, \qquad f'(x_0 - h/2) \approx \frac{f(x_0) - f(x_0 - h)}{h}$$

such that substitution back to (3.8) gives

$$f''(x_0) \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} \tag{3.9}$$

Proposition 3.3

The approximation in (3.9) *is second-order accurate. i.e.*

$$\frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} - f''(x_0) = O(h^2).$$

Second order derivative approximation via finite difference

- Suppose now we want to estimate the value of $f''(x_0)$
- We apply central difference using the points $x_0 - h/2$ and $x_0 + h/2$ to get

$$f''(x_0) \approx \frac{f'(x_0 + h/2) - f'(x_0 - h/2)}{h} \tag{3.8}$$

- If we apply central difference again to get an estimate of $f'(x_0 + h/2)$ and $f'(x_0 - h/2)$, then

$$f'(x_0 + h/2) \approx \frac{f(x_0 + h) - f(x_0)}{h}, \qquad f'(x_0 - h/2) \approx \frac{f(x_0) - f(x_0 - h)}{h}$$

such that substitution back to (3.8) gives

$$f''(x_0) \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} \tag{3.9}$$

Proposition 3.3

The approximation in (3.9) *is second-order accurate. i.e.*

$$\frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} - f''(x_0) = O(h^2).$$

Proof. On summing up (3.6) and (3.7) followed by a slight rearrangement, we can obtain

$$\frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} - f''(x_0) = \frac{f^{(4)}(x_0)}{12}h^2 + \cdots = O(h^2).$$

Finite difference methods for solving PDE

- Our goal is to obtain numerical solution to a second order linear PDE of form

$$\frac{\partial V}{\partial \tau} = a(\tau, x)\frac{\partial^2 V}{\partial x^2} + b(\tau, x)\frac{\partial V}{\partial x} - c(\tau, x)V, \qquad \tau < T$$

$$V(0, x) = g(x), \qquad \tau = 0$$

over a domain $(\tau, x) \in D := [0, T] \times \mathcal{X}$

Finite difference methods for solving PDE

- Our goal is to obtain numerical solution to a second order linear PDE of form

$$\frac{\partial V}{\partial \tau} = a(\tau, x)\frac{\partial^2 V}{\partial x^2} + b(\tau, x)\frac{\partial V}{\partial x} - c(\tau, x)V, \qquad \tau < T$$
$$V(0, x) = g(x), \qquad \tau = 0$$

  over a domain $(\tau, x) \in D := [0, T] \times \mathcal{X}$

- There are two cases:
  - If $\mathcal{X}$ is bounded in form of $[x_{min}, x_{max}]$, it is essential to impose boundary conditions on the end points by identifying the functions ℓ and u such that

$$V(\tau, x_{min}) = \ell(\tau, x_{min}), \qquad V(\tau, x_{max}) = u(\tau, x_{max}) \qquad \text{for all } \tau < T \qquad (3.10)$$

Finite difference methods for solving PDE

- Our goal is to obtain numerical solution to a second order linear PDE of form

$$\frac{\partial V}{\partial \tau} = a(\tau, x)\frac{\partial^2 V}{\partial x^2} + b(\tau, x)\frac{\partial V}{\partial x} - c(\tau, x)V, \qquad \tau < T$$
$$V(0, x) = g(x), \qquad \tau = 0$$

  over a domain $(\tau, x) \in D := [0, T] \times \mathcal{X}$

- There are two cases:
  - If $\mathcal{X}$ is bounded in form of $[x_{min}, x_{max}]$, it is essential to impose boundary conditions on the end points by identifying the functions ℓ and u such that

$$V(\tau, x_{min}) = \ell(\tau, x_{min}), \qquad V(\tau, x_{max}) = u(\tau, x_{max}) \qquad \text{for all } \tau < T \qquad (3.10)$$

  - If $\mathcal{X}$ is unbounded (e.g. $\mathcal{X} = (-\infty, \infty)$), we can approximate the open domain by a truncated interval $[x_{min}, x_{max}]$ for some very small and large values of x_{min} and x_{max}. Then we need to supply a pair of suitable boundary conditions as in (3.10)

Finite difference methods for solving PDE

- Our goal is to obtain numerical solution to a second order linear PDE of form

$$\frac{\partial V}{\partial \tau} = a(\tau, x)\frac{\partial^2 V}{\partial x^2} + b(\tau, x)\frac{\partial V}{\partial x} - c(\tau, x)V, \qquad \tau < T$$
$$V(0, x) = g(x), \qquad \tau = 0$$

  over a domain $(\tau, x) \in D := [0, T] \times \mathcal{X}$

- There are two cases:
  - If $\mathcal{X}$ is bounded in form of $[x_{min}, x_{max}]$, it is essential to impose boundary conditions on the end points by identifying the functions ℓ and u such that

$$V(\tau, x_{min}) = \ell(\tau, x_{min}), \qquad V(\tau, x_{max}) = u(\tau, x_{max}) \qquad \text{for all } \tau < T \qquad (3.10)$$

  - If $\mathcal{X}$ is unbounded (e.g. $\mathcal{X} = (-\infty, \infty)$), we can approximate the open domain by a truncated interval $[x_{min}, x_{max}]$ for some very small and large values of x_{min} and x_{max}. Then we need to supply a pair of suitable boundary conditions as in (3.10)

- In both cases, construct a uniform grid over D with $N + 1$ points along the time dimension and $M + 1$ points along the space dimension. Let

$$\triangle\tau := \frac{T}{N}, \qquad \triangle x := \frac{x_{max} - x_{min}}{M}.$$

  Then the values of the grid points are given by

$$\tau_n := n\triangle\tau, \qquad n = 0, 1, ..., N$$
$$x_k := x_{min} + k\triangle x, \qquad k = 0, 1, ..., M$$

Specifying the boundary conditions

- The appropriate boundary conditions to be imposed are usually inferred from the nature of the product to be priced (through payoff)

Specifying the boundary conditions

- The appropriate boundary conditions to be imposed are usually inferred from the nature of the product to be priced (through payoff)
- Suppose we want to price an option using the Black-Scholes PDE (3.4) where the natural stock price variable s lives on the domain $[0, \infty)$

Specifying the boundary conditions

- The appropriate boundary conditions to be imposed are usually inferred from the nature of the product to be priced (through payoff)
- Suppose we want to price an option using the Black-Scholes PDE (3.4) where the natural stock price variable s lives on the domain $[0, \infty)$
- In this case $s_{min} = 0$. We also need to choose some large s_{max} and impose boundary conditions for $V(\tau, 0)$ and $V(\tau, S_{max})$ based on the product types

Specifying the boundary conditions

- The appropriate boundary conditions to be imposed are usually inferred from the nature of the product to be priced (through payoff)
- Suppose we want to price an option using the Black-Scholes PDE (3.4) where the natural stock price variable s lives on the domain $[0, \infty)$
- In this case $s_{min} = 0$. We also need to choose some large s_{max} and impose boundary conditions for $V(\tau, 0)$ and $V(\tau, S_{max})$ based on the product types
- For call option with initial condition $V(0, s) = (s - K)^+$:
  1. Call option becomes worthless at zero stock price so $V(\tau, 0) = 0$
  2. For very large initial stock price, the option is very likely to be exercised where the option holder will receive the stock and pay K at time T. Thus $V(\tau, s_{max}) \approx s_{max} - Ke^{-r\tau}$

Specifying the boundary conditions

- The appropriate boundary conditions to be imposed are usually inferred from the nature of the product to be priced (through payoff)
- Suppose we want to price an option using the Black-Scholes PDE (3.4) where the natural stock price variable s lives on the domain $[0, \infty)$
- In this case $s_{min} = 0$. We also need to choose some large s_{max} and impose boundary conditions for $V(\tau, 0)$ and $V(\tau, S_{max})$ based on the product types
- For call option with initial condition $V(0, s) = (s - K)^+$:
  1. Call option becomes worthless at zero stock price so $V(\tau, 0) = 0$
  2. For very large initial stock price, the option is very likely to be exercised where the option holder will receive the stock and pay K at time T. Thus $V(\tau, s_{max}) \approx s_{max} - Ke^{-r\tau}$
- For put option with initial condition $V(0, s) = (K - s)^+$:
  1. Put option behaves like a fixed cash payment of K when stock price is zero, then the option fair value is $V(\tau, 0) = Ke^{-r\tau}$
  2. For very large initial stock price, the put is very likely to end up above the strike price and expire worthless, then $V(\tau, s_{max}) \approx 0$

Specifying the boundary conditions

- The appropriate boundary conditions to be imposed are usually inferred from the nature of the product to be priced (through payoff)
- Suppose we want to price an option using the Black-Scholes PDE (3.4) where the natural stock price variable s lives on the domain $[0, \infty)$
- In this case $s_{min} = 0$. We also need to choose some large s_{max} and impose boundary conditions for $V(\tau, 0)$ and $V(\tau, S_{max})$ based on the product types
- For call option with initial condition $V(0, s) = (s - K)^+$:
  1. Call option becomes worthless at zero stock price so $V(\tau, 0) = 0$
  2. For very large initial stock price, the option is very likely to be exercised where the option holder will receive the stock and pay K at time T. Thus $V(\tau, s_{max}) \approx s_{max} - Ke^{-r\tau}$
- For put option with initial condition $V(0, s) = (K - s)^+$:
  1. Put option behaves like a fixed cash payment of K when stock price is zero, then the option fair value is $V(\tau, 0) = Ke^{-r\tau}$
  2. For very large initial stock price, the put is very likely to end up above the strike price and expire worthless, then $V(\tau, s_{max}) \approx 0$
- How should we choose s_{max}? This could for example be guided by the statistical distribution of the stock price under $\mathbb{Q}$ (e.g. the 99.9th percentile)

Pricing under Black-Scholes model: which PDE to work with?

- The Black-Scholes PDE can be stated in the natural price scale (3.4) or the log-price scale (3.5). Which one shall we work with?
- Advantages of log-price scale relative to natural price scale
  - The coefficient of each partial derivative in (3.5) is constant independent of the space and time variable. This makes the implementation of the finite difference scheme a bit easier and improves the stability of certain types of scheme
- Disadvantages of log-price scale relative to natural price scale
  - The semi-open domain $[0, \infty)$ of the natural price maps to an open domain $(-\infty, \infty)$ of the log-price, which needs to be truncated both from the below at some x_{min} and from the above at some x_{max}
  - A uniform grid in the x-coordinate maps to a very distorted grid in the s-coordinate (via $s_n = \exp(x_n)$)

Discretising a PDE by finite difference

- Let V_k^n be our numerical approximation of $V(\tau_n, x_k)$

Discretising a PDE by finite difference

- Let V_k^n be our numerical approximation of $V(\tau_n, x_k)$
- We estimate the spatial derivatives $\frac{\partial V}{\partial x}$ and $\frac{\partial^2 V}{\partial x^2}$ at (τ_n, x_k) by central difference:

$$\frac{\partial V}{\partial x} \approx \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} \longleftrightarrow \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x}$$

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} \longleftrightarrow \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2}$$

Discretising a PDE by finite difference

- Let V_k^n be our numerical approximation of $V(\tau_n, x_k)$
- We estimate the spatial derivatives $\frac{\partial V}{\partial x}$ and $\frac{\partial^2 V}{\partial x^2}$ at (τ_n, x_k) by central difference:

$$\frac{\partial V}{\partial x} \approx \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} \longleftrightarrow \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x}$$

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} \longleftrightarrow \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2}$$

- For the time derivative $\frac{\partial V}{\partial \tau}$, it can be approximated by forward or backward difference:

Forward difference: $$\frac{\partial V}{\partial \tau} \approx \frac{V(\tau_{n+1}, x_k) - V(\tau_n, x_k)}{\triangle \tau} \longleftrightarrow \frac{V_k^{n+1} - V_k^n}{\triangle t}$$

Backward difference: $$\frac{\partial V}{\partial \tau} \approx \frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau} \longleftrightarrow \frac{V_k^n - V_k^{n-1}}{\triangle \tau}$$

Discretising a PDE by finite difference

- Let V_k^n be our numerical approximation of $V(\tau_n, x_k)$
- We estimate the spatial derivatives $\frac{\partial V}{\partial x}$ and $\frac{\partial^2 V}{\partial x^2}$ at (τ_n, x_k) by central difference:

$$\frac{\partial V}{\partial x} \approx \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} \longleftrightarrow \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x}$$

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} \longleftrightarrow \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2}$$

- For the time derivative $\frac{\partial V}{\partial \tau}$, it can be approximated by forward or backward difference:

$$\text{Forward difference:} \quad \frac{\partial V}{\partial \tau} \approx \frac{V(\tau_{n+1}, x_k) - V(\tau_n, x_k)}{\triangle \tau} \longleftrightarrow \frac{V_k^{n+1} - V_k^n}{\triangle t}$$

$$\text{Backward difference:} \quad \frac{\partial V}{\partial \tau} \approx \frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau} \longleftrightarrow \frac{V_k^n - V_k^{n-1}}{\triangle \tau}$$

- A discretised system is obtained on replacing all the partial derivatives in the PDE by the finite difference expressions above

Explicit scheme

- Let $a_k^n := a(\tau_n, x_k)$, $b_k^n := b(\tau_n, x_k)$ and $c_k^n := c(\tau_n, x_k)$
- Suppose we use forward difference for the time derivative, then the discretised system is

$$\begin{aligned}
&\frac{V_k^{n+1} - V_k^n}{\triangle\tau} = a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x} - c_k^n V_k^n \\
\implies \quad & V_k^{n+1} = \left(\frac{\triangle\tau}{\triangle x^2} a_k^n - \frac{\triangle\tau}{2\triangle x} b_k^n\right) V_{k-1}^n + \left(1 - \frac{2\triangle\tau}{\triangle x^2} a_k^n - \triangle\tau c_k^n\right) V_k^n \\
&\qquad + \left(\frac{\triangle\tau}{\triangle x^2} a_k^n + \frac{\triangle\tau}{2\triangle x} b_k^n\right) V_{k+1}^n \\
&\qquad = A_k^n V_{k-1}^n + (1 + B_k^n) V_k^n + C_k^n V_{k+1}^n
\end{aligned}$$

on defining

$$A_k^n := \frac{\triangle\tau}{\triangle x^2} a_k^n - \frac{\triangle\tau}{2\triangle x} b_k^n, \quad B_k^n := -\frac{2\triangle\tau}{\triangle x^2} a_k^n - \triangle\tau c_k^n, \quad C_k^n := \frac{\triangle\tau}{\triangle x^2} a_k^n + \frac{\triangle\tau}{2\triangle x} b_k^n \quad (3.11)$$

Matrix representation of the explicit scheme

- The recursive equation $V_k^{n+1} = A_k^n V_{k-1}^n + (1 + B_k^n) V_k^n + C_k^n V_{k+1}^n$ holds for $k = 1, 2, ..., M-1$

Matrix representation of the explicit scheme

- The recursive equation $V_k^{n+1} = A_k^n V_{k-1}^n + (1 + B_k^n) V_k^n + C_k^n V_{k+1}^n$ holds for $k = 1, 2, ..., M-1$
- Thus the equation doesn't give us the value of V_0^{n+1} and V_M^{n+1}. But it is fine because we know their values from the boundary conditions

Matrix representation of the explicit scheme

- The recursive equation $V_k^{n+1} = A_k^n V_{k-1}^n + (1 + B_k^n) V_k^n + C_k^n V_{k+1}^n$ holds for $k = 1, 2, ..., M-1$
- Thus the equation doesn't give us the value of V_0^{n+1} and V_M^{n+1}. But it is fine because we know their values from the boundary conditions
- The $M-1$ recursive equations can be summarised using matrix notation

$$\underbrace{\begin{bmatrix} V_0^{n+1} \\ V_1^{n+1} \\ V_2^{n+1} \\ \vdots \\ V_{M-1}^{n+1} \\ V_M^{n+1} \end{bmatrix}}_{=:V^{n+1}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & & & & 0 & 0 \\ A_1^n & 1+B_1^n & C_1^n & 0 & \cdots & & & 0 & 0 \\ 0 & A_2^n & 1+B_2^n & C_2^n & 0 & \cdots & & & \\ & & & \ddots & & & & & \\ \vdots & & & & & & & & \\ & & & & A_{M-2}^n & 1+B_{M-2}^n & C_{M-2}^n & 0 & \\ & & & & 0 & A_{M-1}^n & 1+B_{M-1}^n & C_{M-1}^n & \\ 0 & 0 & \cdots & & 0 & 0 & 0 & 1 & \end{bmatrix}}_{=:\mathbb{I}+L^n} \underbrace{\begin{bmatrix} V_0^n \\ V_1^n \\ V_2^n \\ \vdots \\ V_{M-1}^n \\ V_M^n \end{bmatrix}}_{=:V^n}$$

Here $\mathbb{I}$ is an $(M+1) \times (M+1)$ identity matrix and L^n is an $(M+1) \times (M+1)$ matrix in form of

$$L^n := \begin{bmatrix} 0 & 0 & 0 & \cdots & & & & 0 & 0 \\ A_1^n & B_1^n & C_1^n & 0 & \cdots & & & 0 & 0 \\ 0 & A_2^n & B_2^n & C_2^n & 0 & \cdots & & & \\ & & & & \ddots & & & & \\ \vdots & & & & & & & & \\ & & & & & A_{M-2}^n & B_{M-2}^n & C_{M-2}^n & 0 \\ & & & & & 0 & A_{M-1}^n & B_{M-1}^n & C_{M-1}^n \\ 0 & 0 & \cdots & & & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.12}$$

Matrix representation of the explicit scheme (cont')

- Note that the system $V^{n+1} = (\mathbb{I} + L^n)V^n$ doesn't give us the correct value of V_0^{n+1} and V_M^{n+1}. We need to manually overwrite the first and last entry of the vector to incorporate the boundary conditions at $x = x_0 = x_{min}$ and $x = x_M = x_{max}$

Matrix representation of the explicit scheme (cont')

- Note that the system $V^{n+1} = (\mathbb{I} + L^n)V^n$ doesn't give us the correct value of V_0^{n+1} and V_M^{n+1}. We need to manually overwrite the first and last entry of the vector to incorporate the boundary conditions at $x = x_0 = x_{min}$ and $x = x_M = x_{max}$

- The complete recursive algorithm is:

$$V^{n+1} = B^{n+1}((\mathbb{I} + L^n)V^n) \tag{3.13}$$

where $B^{n+1}(\cdot)$ is an operator which overwrites the first and last entry of the input vector to the values of the boundary conditions at $x = x_0 = x_{min}$ and $x = x_M = x_{max}$

Matrix representation of the explicit scheme (cont')

- Note that the system $V^{n+1} = (\mathbb{I} + L^n)V^n$ doesn't give us the correct value of V_0^{n+1} and V_M^{n+1}. We need to manually overwrite the first and last entry of the vector to incorporate the boundary conditions at $x = x_0 = x_{min}$ and $x = x_M = x_{max}$

- The complete recursive algorithm is:

$$V^{n+1} = B^{n+1}((\mathbb{I} + L^n)V^n) \tag{3.13}$$

  where $B^{n+1}(\cdot)$ is an operator which overwrites the first and last entry of the input vector to the values of the boundary conditions at $x = x_0 = x_{min}$ and $x = x_M = x_{max}$

- The scheme is called explicit because given the approximated function value at the previous time point $n-1$, the approximation at the current time point n can be explicitly computed

Explicit scheme - recap of the algorithm

1. Set up the grid over the computation domain:
   - Fix the number of sub-intervals M and N along the spatial and time direction
   - Define the grid points

$$\tau_n := n\triangle\tau, \qquad x_k := x_{min} + k\triangle x \quad \text{for } n = 0, 1, ..., N \text{ and } k = 0, 1, ..., M$$

   where $\triangle\tau := \frac{T}{N}$ and $\triangle x := \frac{x_{max} - x_{min}}{M}$

Explicit scheme - recap of the algorithm

1. Set up the grid over the computation domain:
   - Fix the number of sub-intervals M and N along the spatial and time direction
   - Define the grid points

$$\tau_n := n\triangle\tau, \qquad x_k := x_{min} + k\triangle x \quad \text{for } n = 0, 1, ..., N \text{ and } k = 0, 1, ..., M$$

   where $\triangle\tau := \frac{T}{N}$ and $\triangle x := \frac{x_{max} - x_{min}}{M}$

2. Initiate the vector V^0 by the initial condition:
   - For $k = 0, 1, ..., M$, set:

$$V_k^0 = g(x_k)$$

Explicit scheme - recap of the algorithm

1. Set up the grid over the computation domain:
   - Fix the number of sub-intervals M and N along the spatial and time direction
   - Define the grid points

$$\tau_n := n\triangle\tau, \qquad x_k := x_{min} + k\triangle x \quad \text{for } n = 0, 1, ..., N \text{ and } k = 0, 1, ..., M$$

   where $\triangle\tau := \frac{T}{N}$ and $\triangle x := \frac{x_{max} - x_{min}}{M}$

2. Initiate the vector V^0 by the initial condition:
   - For $k = 0, 1, ..., M$, set:

$$V_k^0 = g(x_k)$$

3. Loop forward in time to obtain the approximation vector at subsequent time points:
   - For $n = 0, 1, ..., N-1$, compute the following vector/matrix multiplication

$$V^{n+1} = B^{n+1}((\mathbb{I} + L^n)V^n)$$

   where $B^{n+1}(\cdot)$ is the boundary operator which modifies the first and the last entry of a vector based on the given boundary conditions at time $n+1$, L^{n+1} is the matrix defined in (3.12) and $\mathbb{I}$ is the identity matrix

Explicit scheme - recap of the algorithm

1. Set up the grid over the computation domain:
   - Fix the number of sub-intervals M and N along the spatial and time direction
   - Define the grid points

$$\tau_n := n\triangle\tau, \qquad x_k := x_{min} + k\triangle x \quad \text{for } n = 0, 1, ..., N \text{ and } k = 0, 1, ..., M$$

   where $\triangle\tau := \frac{T}{N}$ and $\triangle x := \frac{x_{max} - x_{min}}{M}$

2. Initiate the vector V^0 by the initial condition:
   - For $k = 0, 1, ..., M$, set:

$$V_k^0 = g(x_k)$$

3. Loop forward in time to obtain the approximation vector at subsequent time points:
   - For $n = 0, 1, ..., N-1$, compute the following vector/matrix multiplication

$$V^{n+1} = B^{n+1}((\mathbb{I} + L^n)V^n)$$

   where $B^{n+1}(\cdot)$ is the boundary operator which modifies the first and the last entry of a vector based on the given boundary conditions at time $n+1$, L^{n+1} is the matrix defined in (3.12) and $\mathbb{I}$ is the identity matrix

4. The numerical approximation of $V(\tau_n, x_k)$ is then given by the value of V_k^n for $n = 0, 1, ..., N$ and $k = 0, 1, ..., M$

Explicit scheme - recap of the algorithm

1. Set up the grid over the computation domain:
   - Fix the number of sub-intervals M and N along the spatial and time direction
   - Define the grid points

$$\tau_n := n\triangle\tau, \qquad x_k := x_{min} + k\triangle x \quad \text{for } n = 0, 1, ..., N \text{ and } k = 0, 1, ..., M$$

   where $\triangle\tau := \frac{T}{N}$ and $\triangle x := \frac{x_{max} - x_{min}}{M}$

2. Initiate the vector V^0 by the initial condition:
   - For $k = 0, 1, ..., M$, set:

$$V_k^0 = g(x_k)$$

3. Loop forward in time to obtain the approximation vector at subsequent time points:
   - For $n = 0, 1, ..., N-1$, compute the following vector/matrix multiplication

$$V^{n+1} = B^{n+1}((\mathbb{I} + L^n)V^n)$$

   where $B^{n+1}(\cdot)$ is the boundary operator which modifies the first and the last entry of a vector based on the given boundary conditions at time $n+1$, L^{n+1} is the matrix defined in (3.12) and $\mathbb{I}$ is the identity matrix

4. The numerical approximation of $V(\tau_n, x_k)$ is then given by the value of V_k^n for $n = 0, 1, ..., N$ and $k = 0, 1, ..., M$
5. (Use interpolation to obtain estimate of $V(t, x)$ for arbitrary (t, x))

Implicit scheme

- An implicit scheme is obtained if we estimate the time derivative $\frac{\partial V}{\partial \tau}$ at (τ_n, x_k) via backward difference instead:

$$\frac{\partial V}{\partial \tau} \approx \frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau} \longleftrightarrow \frac{V_k^n - V_k^{n-1}}{\triangle \tau}$$

Implicit scheme

- An implicit scheme is obtained if we estimate the time derivative $\frac{\partial V}{\partial \tau}$ at (τ_n, x_k) via backward difference instead:

$$\frac{\partial V}{\partial \tau} \approx \frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau} \longleftrightarrow \frac{V_k^n - V_k^{n-1}}{\triangle \tau}$$

- The discretised PDE is then given by

$$\begin{aligned}
\frac{V_k^n - V_k^{n-1}}{\triangle \tau} &= a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x} - c_k^n V_k^n \\
V_k^{n-1} &= -\left(\frac{\triangle \tau}{\triangle x^2} a_k^n - \frac{\triangle \tau}{2\triangle x} b_k^n\right) V_{k-1}^n + \left(\frac{2\triangle \tau}{\triangle x^2} a_k^n + \triangle \tau c_k^n + 1\right) V_k^n \\
&\quad - \left(\frac{\triangle \tau}{\triangle x^2} a_k^n + \frac{\triangle \tau}{2\triangle x} b_k^n\right) V_{k+1}^n \\
&= -A_k^n V_{k-1}^n + (1 - B_k^n) V_k^n - C_k^n V_{k+1}^n
\end{aligned}$$

on recalling the definition that

$$A_k^n := \frac{\triangle \tau}{\triangle x^2} a_k^n - \frac{\triangle \tau}{2\triangle x} b_k^n, \quad B_k^n := -\frac{2\triangle \tau}{\triangle x^2} a_k^n - \triangle \tau c_k^n, \quad C_k^n := \frac{\triangle \tau}{\triangle x^2} a_k^n + \frac{\triangle \tau}{2\triangle x} b_k^n$$

Matrix representation of the implicit scheme

- The recursive equation $V_k^{n-1} = -A_k^n V_{k-1}^n + (1 - B_k^n) V_k^n - C_k^n V_{k+1}^n$ holds for $k = 1, 2, ..., M-1$
- The recursive equations can be summarised using matrix notation

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & & & & 0 & 0 \\ -A_1^n & 1-B_1^n & -C_1^n & 0 & \cdots & & & 0 & 0 \\ 0 & -A_2^n & 1-B_2^n & -C_2^n & 0 & \cdots & & & \\ & & & & \ddots & & & & \\ \vdots & & & & & -A_k^n & 1-B_k^n & -C_k^n & 0 \\ & & & & & 0 & -A_{M-1}^n & 1-B_{M-1}^n & -C_{M-1}^n \\ 0 & 0 & \cdots & & & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_0^n \\ V_1^n \\ V_2^n \\ \vdots \\ \vdots \\ \vdots \\ V_{M-1}^n \\ V_M^n \end{bmatrix} = \begin{bmatrix} \ell_0^n \\ V_1^{n-1} \\ V_2^{n-1} \\ \vdots \\ \vdots \\ \vdots \\ V_{M-1}^{n-1} \\ u_M^n \end{bmatrix}$$

which can be written as

$$(\mathbb{I} - L^n) V^n = B^n(V^{n-1}) \tag{3.14}$$

where L^n has been defined previously in (3.12) and $B^n(\cdot)$ again is an operator which modifies the first and last entry of a vector to the appropriate BCs.

- The scheme is called implicit because given the approximation vector at a previous time point $n-1$, the vector at the next time point n needs to be obtained by solving a system of equations

Inverting a tridiagional matrix

- To obtain V^n given V^{n-1}, one has to solve a system of equations which in general involves inverting the matrix $(\mathbb{I} - L^n)$

Inverting a tridiagional matrix

- To obtain V^n given V^{n-1}, one has to solve a system of equations which in general involves inverting the matrix $(\mathbb{I} - L^n)$
- Inversion of a large matrix is computationally costly. But since $(\mathbb{I} - L^n)$ is a tridiagional matrix, efficient algorithm exists to solve the system of equations

Inverting a tridiagional matrix

- To obtain V^n given V^{n-1}, one has to solve a system of equations which in general involves inverting the matrix $(\mathbb{I} - L^n)$
- Inversion of a large matrix is computationally costly. But since $(\mathbb{I} - L^n)$ is a tridiagional matrix, efficient algorithm exists to solve the system of equations
- Consider a system of equation $Ax = d$ which is represented by an augmented matrix

$$\left[\begin{array}{cccccccccc|c}
\beta_0 & \gamma_0 & 0 & \cdots & & & & 0 & 0 & d_0 \\
\alpha_1 & \beta_1 & \gamma_1 & 0 & \cdots & & & 0 & 0 & d_1 \\
0 & \alpha_2 & \beta_2 & \gamma_2 & 0 & \cdots & & & & d_2 \\
 & & & & \ddots & & & & & \vdots \\
\vdots & & & & & \alpha_{M-2} & \beta_{M-2} & \gamma_{M-2} & 0 & d_{M-2} \\
 & & & & & 0 & \alpha_{M-1} & \beta_{M-1} & \gamma_{M-1} & d_{M-1} \\
0 & 0 & \cdots & & & 0 & 0 & \alpha_M & \beta_M & d_M
\end{array}\right]$$

Inverting a tridiagional matrix

- To obtain V^n given V^{n-1}, one has to solve a system of equations which in general involves inverting the matrix $(\mathbb{I} - L^n)$
- Inversion of a large matrix is computationally costly. But since $(\mathbb{I} - L^n)$ is a tridiagional matrix, efficient algorithm exists to solve the system of equations
- Consider a system of equation $Ax = d$ which is represented by an augmented matrix

$$\left[\begin{array}{cccccccc|c} \beta_0 & \gamma_0 & 0 & \cdots & & & 0 & 0 & d_0 \\ \alpha_1 & \beta_1 & \gamma_1 & 0 & \cdots & & 0 & 0 & d_1 \\ 0 & \alpha_2 & \beta_2 & \gamma_2 & 0 & \cdots & & & d_2 \\ & & & & \ddots & & & & \vdots \\ \vdots & & & & & \alpha_{M-2} & \beta_{M-2} & \gamma_{M-2} & 0 & d_{M-2} \\ & & & & & 0 & \alpha_{M-1} & \beta_{M-1} & \gamma_{M-1} & d_{M-1} \\ 0 & 0 & \cdots & & & 0 & 0 & \alpha_M & \beta_M & d_M \end{array}\right]$$

- The idea is to first use row operations to eliminate all the α_k's, and then use backward substitution to obtain the x_k's. This procedure is known as the *Thomas algorithm*

Thomas algorithm for solving a tridiagional system

1. For each $k = 1, 2, ..., M$, perform the row operation $(k) - \frac{\alpha_k}{\beta_{k-1}} \times (k-1) \to (k)$. i.e.:
   - For $k = 1, 2, ..., M$,

$$\text{Set:} \qquad \beta_k = \beta_k - \frac{\alpha_k}{\beta_{k-1}}\gamma_{k-1}, \qquad d_k = d_k - \frac{\alpha_k}{\beta_{k-1}} d_{k-1}$$

   Then the system becomes

$$\left[\begin{array}{cccccccc|c} \beta_0 & \gamma_0 & 0 & \cdots & & & 0 & 0 & d_0 \\ 0 & \beta_1 & \gamma_1 & 0 & \cdots & & 0 & 0 & d_1 \\ 0 & 0 & \beta_2 & \gamma_2 & 0 & \cdots & & & d_2 \\ & & & & \ddots & & & & \vdots \\ \vdots & & & & & 0 & \beta_{M-2} & \gamma_{M-2} & 0 & d_{M-2} \\ & & & & & 0 & 0 & \beta_{M-1} & \gamma_{M-1} & d_{M-1} \\ 0 & 0 & \cdots & & & 0 & 0 & 0 & \beta_M & d_M \end{array}\right]$$

2. Backward substitution: starting with $x_M = \frac{d_M}{\beta_M}$. For $k = M-1, M-2, ..., 0$, compute

$$x_k = \frac{d_k - \gamma_k x_{k+1}}{\beta_k}$$

Remark: The complexity of Thomas algorithm is $O(M)$, versus $O(M^3)$ under a standard Gaussian elimination algorithm.

Implicit scheme - recap of the algorithm

1. Set up the grid for computation (same as the procedure in the explicit scheme)
2. Initiate the vector V^0 by the initial condition:
   - For $k = 0, 1, ..., M$, set:

$$V_k^0 = g(x_k)$$

3. Loop forward in time to update the approximation vector:
   - For $n = 1, 2, ..., N$, compute V^n by solving the following system of equations (eg using Thomas algorithm)

$$(\mathbb{I} - L^n)V^n = B^n(V^{n-1})$$

4. The numerical approximation of $V(\tau_n, x_k)$ is then given by the value of V_k^n for $n = 0, 1, ..., N$ and $k = 0, 1, ..., M$
5. (Use interpolation to obtain estimate of $V(\tau, x)$ for arbitrary (τ, x))

Weighted average of explicit and implicit scheme

- The recursive equations for the explicit and implicit scheme are respectively given by

$$\text{Explicit:} \quad \frac{V_k^n - V_k^{n-1}}{\triangle\tau} = a_k^{n-1}\frac{V_{k+1}^{n-1} - 2V_k^{n-1} + V_{k-1}^{n-1}}{\triangle x^2} + b_k^{n-1}\frac{V_{k+1}^{n-1} - V_{k-1}^{n-1}}{2\triangle x} - c_k^{n-1}V_k^{n-1}$$

$$\text{Implicit:} \quad \frac{V_k^n - V_k^{n-1}}{\triangle\tau} = a_k^{n}\frac{V_{k+1}^{n} - 2V_k^{n} + V_{k-1}^{n}}{\triangle x^2} + b_k^{n}\frac{V_{k+1}^{n} - V_{k-1}^{n}}{2\triangle x} - c_k^{n}V_k^{n}$$

Weighted average of explicit and implicit scheme

- The recursive equations for the explicit and implicit scheme are respectively given by

$$\text{Explicit:} \quad \frac{V_k^n - V_k^{n-1}}{\triangle\tau} = a_k^{n-1}\frac{V_{k+1}^{n-1} - 2V_k^{n-1} + V_{k-1}^{n-1}}{\triangle x^2} + b_k^{n-1}\frac{V_{k+1}^{n-1} - V_{k-1}^{n-1}}{2\triangle x} - c_k^{n-1}V_k^{n-1}$$

$$\text{Implicit:} \quad \frac{V_k^n - V_k^{n-1}}{\triangle\tau} = a_k^{n}\frac{V_{k+1}^{n} - 2V_k^{n} + V_{k-1}^{n}}{\triangle x^2} + b_k^{n}\frac{V_{k+1}^{n} - V_{k-1}^{n}}{2\triangle x} - c_k^{n}V_k^{n}$$

- A θ-scheme involves taking weighted average of the RHS of the above equations:

$$\frac{V_k^n - V_k^{n-1}}{\triangle\tau} = (1-\theta)\left\{a_k^{n-1}\frac{V_{k+1}^{n-1} - 2V_k^{n-1} + V_{k-1}^{n-1}}{\triangle x^2} + b_k^{n-1}\frac{V_{k+1}^{n-1} - V_{k-1}^{n-1}}{2\triangle x} - c_k^{n-1}V_k^{n-1}\right\}$$
$$+\theta\left\{a_k^{n}\frac{V_{k+1}^{n} - 2V_k^{n} + V_{k-1}^{n}}{\triangle x^2} + b_k^{n}\frac{V_{k+1}^{n} - V_{k-1}^{n}}{2\triangle x} - c_k^{n}V_k^{n}\right\}$$

Weighted average of explicit and implicit scheme

- The recursive equations for the explicit and implicit scheme are respectively given by

$$\text{Explicit:} \quad \frac{V_k^n - V_k^{n-1}}{\triangle\tau} = a_k^{n-1}\frac{V_{k+1}^{n-1} - 2V_k^{n-1} + V_{k-1}^{n-1}}{\triangle x^2} + b_k^{n-1}\frac{V_{k+1}^{n-1} - V_{k-1}^{n-1}}{2\triangle x} - c_k^{n-1}V_k^{n-1}$$

$$\text{Implicit:} \quad \frac{V_k^n - V_k^{n-1}}{\triangle\tau} = a_k^{n}\frac{V_{k+1}^{n} - 2V_k^{n} + V_{k-1}^{n}}{\triangle x^2} + b_k^{n}\frac{V_{k+1}^{n} - V_{k-1}^{n}}{2\triangle x} - c_k^{n}V_k^{n}$$

- A θ-scheme involves taking weighted average of the RHS of the above equations:

$$\frac{V_k^n - V_k^{n-1}}{\triangle\tau} = (1-\theta)\left\{a_k^{n-1}\frac{V_{k+1}^{n-1} - 2V_k^{n-1} + V_{k-1}^{n-1}}{\triangle x^2} + b_k^{n-1}\frac{V_{k+1}^{n-1} - V_{k-1}^{n-1}}{2\triangle x} - c_k^{n-1}V_k^{n-1}\right\}$$
$$+\theta\left\{a_k^{n}\frac{V_{k+1}^{n} - 2V_k^{n} + V_{k-1}^{n}}{\triangle x^2} + b_k^{n}\frac{V_{k+1}^{n} - V_{k-1}^{n}}{2\triangle x} - c_k^{n}V_k^{n}\right\}$$

- On rearrangement we obtain

$$-\theta A_k^n V_{k-1}^n + (1-\theta B_k^n)V_k^n - \theta C_k^n V_{k+1}^n = (1-\theta)A_k^{n-1}V_{k-1}^{n-1} + [1+(1-\theta)B_k^{n-1}]V_k^{n-1} + (1-\theta)C_k^{n-1}V_{k+1}^{n-1}$$

where the coefficients A, B and C are defined in (3.11)

Matrix representation of the θ-scheme

- Based on the same idea as in the analysis of the explicit/implicit scheme, the recursive equations and the boundary conditions can be summarised by the following matrix notation

$$(\mathbb{I} - \theta L^n)V^n = B^n[(\mathbb{I} + (1-\theta)L^{n-1})V^{n-1}]$$

where the matrix L^n is defined in (3.12) and $B^n(\cdot)$ is the boundary operator

Matrix representation of the θ-scheme

- Based on the same idea as in the analysis of the explicit/implicit scheme, the recursive equations and the boundary conditions can be summarised by the following matrix notation

$$(\mathbb{I} - \theta L^n)V^n = B^n[(\mathbb{I} + (1-\theta)L^{n-1})V^{n-1}]$$

  where the matrix L^n is defined in (3.12) and $B^n(\cdot)$ is the boundary operator

- If $\theta = 0$, it recovers the explicit scheme (3.13)

Matrix representation of the θ-scheme

- Based on the same idea as in the analysis of the explicit/implicit scheme, the recursive equations and the boundary conditions can be summarised by the following matrix notation

$$(\mathbb{I} - \theta L^n)V^n = B^n[(\mathbb{I} + (1-\theta)L^{n-1})V^{n-1}]$$

  where the matrix L^n is defined in (3.12) and $B^n(\cdot)$ is the boundary operator

- If $\theta = 0$, it recovers the explicit scheme (3.13)
- If $\theta = 1$, it recovers the implicit scheme (3.14)

Matrix representation of the θ-scheme

- Based on the same idea as in the analysis of the explicit/implicit scheme, the recursive equations and the boundary conditions can be summarised by the following matrix notation

$$(\mathbb{I} - \theta L^n)V^n = B^n[(\mathbb{I} + (1-\theta)L^{n-1})V^{n-1}]$$

  where the matrix L^n is defined in (3.12) and $B^n(\cdot)$ is the boundary operator

- If $\theta = 0$, it recovers the explicit scheme (3.13)
- If $\theta = 1$, it recovers the implicit scheme (3.14)
- The choice of $\theta = \frac{1}{2}$ is known as the Crank-Nicolson scheme

θ-scheme - recap of the algorithm

1. Set up the grid for computation (same as the procedure in the explicit scheme)
2. Initiate the vector V^0 by the initial condition:
   - For $k = 0, 1, ..., M$, set:
   $$V_k^0 = g(x_k)$$
3. Loop forward in time to update the approximation vector:
   - For $n = 1, 2, ..., N$:
     - ⋆ Obtain V^n by solving the following system of equations (eg using Thomas algorithm)
     $$(\mathbb{I} - \theta L^n)V^n = B^n[(\mathbb{I} + (1-\theta)L^{n-1})V^{n-1}]$$
4. The numerical approximation of $V(\tau_n, x_k)$ is then given by the value of V_k^n for $n = 0, 1, ..., N$ and $k = 0, 1, ..., M$
5. (Use interpolation to obtain estimate of $V(\tau, x)$ for arbitrary (τ, x))

Local truncation error and consistency

- We hope V_k^n can serve as a good approximation to $V(\tau_n, x_k)$ where $V(\cdot, \cdot)$ is the true but unknown solution to the PDE

Local truncation error and consistency

- We hope V_k^n can serve as a good approximation to $V(\tau_n, x_k)$ where $V(\cdot, \cdot)$ is the true but unknown solution to the PDE
- We cannot easily measure how well the V_k^n's satisfies the original PDE since they are only defined at discrete points

Local truncation error and consistency

- We hope V_k^n can serve as a good approximation to $V(\tau_n, x_k)$ where $V(\cdot, \cdot)$ is the true but unknown solution to the PDE
- We cannot easily measure how well the V_k^n's satisfies the original PDE since they are only defined at discrete points
- But we can measure how well the (unknown) solution to the PDE, evaluated at the grid points, satisfies the finite difference scheme

Local truncation error and consistency

- We hope V_k^n can serve as a good approximation to $V(\tau_n, x_k)$ where $V(\cdot, \cdot)$ is the true but unknown solution to the PDE
- We cannot easily measure how well the V_k^n's satisfies the original PDE since they are only defined at discrete points
- But we can measure how well the (unknown) solution to the PDE, evaluated at the grid points, satisfies the finite difference scheme
- The local truncation error T_k^n at a particular grid point (τ_n, x_k) is the error by which the exact solution of a differential equation does not satisfy the difference equation at that point

Local truncation error and consistency

- We hope V_k^n can serve as a good approximation to $V(\tau_n, x_k)$ where $V(\cdot, \cdot)$ is the true but unknown solution to the PDE
- We cannot easily measure how well the V_k^n's satisfies the original PDE since they are only defined at discrete points
- But we can measure how well the (unknown) solution to the PDE, evaluated at the grid points, satisfies the finite difference scheme
- The local truncation error T_k^n at a particular grid point (τ_n, x_k) is the error by which the exact solution of a differential equation does not satisfy the difference equation at that point

Definition 3.4

A finite difference scheme is said to be consistent if the local truncation error T_k^n at each grid point (τ_n, x_k) converges to zero as $\triangle x, \triangle \tau \to 0$.

Local truncation error and consistency

- We hope V_k^n can serve as a good approximation to $V(\tau_n, x_k)$ where $V(\cdot, \cdot)$ is the true but unknown solution to the PDE
- We cannot easily measure how well the V_k^n's satisfies the original PDE since they are only defined at discrete points
- But we can measure how well the (unknown) solution to the PDE, evaluated at the grid points, satisfies the finite difference scheme
- The local truncation error T_k^n at a particular grid point (τ_n, x_k) is the error by which the exact solution of a differential equation does not satisfy the difference equation at that point

Definition 3.4

A finite difference scheme is said to be consistent if the local truncation error T_k^n at each grid point (τ_n, x_k) converges to zero as $\triangle x, \triangle \tau \to 0$.

Proposition 3.5

The explicit, implicit and Crank-Nicolson scheme are all consistent. Moreover,

$$T_k^n = \begin{cases} O(\triangle\tau) + O(\triangle x^2) & \text{for the explicit and implicit scheme;} \\ O(\triangle\tau^2) + O(\triangle x^2) & \text{for the Crank-Nicolson scheme.} \end{cases}$$

Local truncation error of explicit scheme

- The differencing equation for an explicit scheme is

$$\frac{V_k^{n+1} - V_k^n}{\triangle\tau} = a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x} - c_k^n V_k^n$$

Local truncation error of explicit scheme

- The differencing equation for an explicit scheme is

$$\frac{V_k^{n+1} - V_k^n}{\triangle\tau} = a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x} - c_k^n V_k^n$$

- The local truncation error of the explicit scheme is defined as

$$T_k^n := \frac{V(\tau_{n+1}, x_k) - V(\tau_n, x_k)}{\triangle\tau} - \Big[a_k^n \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} + b_k^n \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} - c_k^n V(\tau_n, x_k)\Big] \tag{3.15}$$

where $V(\cdot, \cdot)$ is the true solution to the PDE

Local truncation error of explicit scheme

- The differencing equation for an explicit scheme is

$$\frac{V_k^{n+1} - V_k^n}{\triangle \tau} = a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x} - c_k^n V_k^n$$

- The local truncation error of the explicit scheme is defined as

$$T_k^n := \frac{V(\tau_{n+1}, x_k) - V(\tau_n, x_k)}{\triangle \tau} - \Big[a_k^n \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} + b_k^n \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} - c_k^n V(\tau_n, x_k) \Big] \tag{3.15}$$

  where $V(\cdot, \cdot)$ is the true solution to the PDE

- Since we know

$$\left.\frac{\partial V}{\partial \tau}\right|_{(\tau_n, x_k)} = \frac{V(\tau_{n+1}, x_k) - V(\tau_n, x_k)}{\triangle \tau} + O(\triangle \tau),$$

$$\left.\frac{\partial V}{\partial x}\right|_{(\tau_n, x_k)} = \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} + O(\triangle x^2),$$

$$\left.\frac{\partial^2 V}{\partial x^2}\right|_{(\tau_n, x_k)} = \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} + O(\triangle x^2),$$

  We have,

$$T_k^n = \underbrace{\frac{\partial V}{\partial \tau} - a(\tau_n, x_k)\frac{\partial^2 V}{\partial x^2} - b(\tau_n, x_k)\frac{\partial V}{\partial x} + c(\tau_n, x_k)V(\tau_n, x_n)}_{=0 \text{ since } V \text{ solves the PDE}} + O(\triangle \tau) + O(\triangle x^2)$$

Local truncation error of Crank-Nicolson scheme

- Similarly, the local truncation error of the Crank-Nicolson scheme is defined as

$$
\begin{aligned}
T_k^n := & \frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau} - \frac{1}{2}\Big\{ a_k^{n-1} \frac{V(\tau_{n-1}, x_{k+1}) - 2V(\tau_{n-1}, x_k) + V(\tau_{n-1}, x_{k-1})}{\triangle x^2} \\
& + b_k^{n-1} \frac{V(\tau_{n-1}, x_{k+1}) - V(\tau_{n-1}, x_{k-1})}{2\triangle x} - c_k^{n-1} V(\tau_{n-1}, x_k) \Big\} \\
& - \frac{1}{2}\Big\{ a_k^{n} \frac{V(\tau_{n}, x_{k+1}) - 2V(\tau_{n}, x_k) + V(\tau_{n}, x_{k-1})}{\triangle x^2} \\
& + b_k^{n} \frac{V(\tau_{n}, x_{k+1}) - V(\tau_{n}, x_{k-1})}{2\triangle x} - c_k^{n} V(\tau_{n}, x_k) \Big\} \\
= & \frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau} - \frac{1}{2}\frac{\partial V}{\partial \tau}\Big|_{(\tau_{n-1}, x_k)} - \frac{1}{2}\frac{\partial V}{\partial \tau}\Big|_{(\tau_{n}, x_k)} + O(\triangle x^2) \\
= & \frac{\partial V}{\partial \tau}\Big|_{(\tau_{n-1/2}, x_k)} - \frac{1}{2}\frac{\partial V}{\partial \tau}\Big|_{(\tau_{n-1}, x_k)} - \frac{1}{2}\frac{\partial V}{\partial \tau}\Big|_{(\tau_{n}, x_k)} + O(\triangle \tau^2) + O(\triangle x^2)
\end{aligned}
$$

if we interpret $\frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau}$ as a central difference around a fictitious point $(\tau_{n-1/2}, x_k)$

Local truncation error of Crank-Nicolson scheme

- Similarly, the local truncation error of the Crank-Nicolson scheme is defined as

$$
\begin{aligned}
T_k^n := & \frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau} - \frac{1}{2}\Big\{ a_k^{n-1} \frac{V(\tau_{n-1}, x_{k+1}) - 2V(\tau_{n-1}, x_k) + V(\tau_{n-1}, x_{k-1})}{\triangle x^2} \\
& + b_k^{n-1} \frac{V(\tau_{n-1}, x_{k+1}) - V(\tau_{n-1}, x_{k-1})}{2\triangle x} - c_k^{n-1} V(\tau_{n-1}, x_k) \Big\} \\
& - \frac{1}{2}\Big\{ a_k^{n} \frac{V(\tau_{n}, x_{k+1}) - 2V(\tau_{n}, x_k) + V(\tau_{n}, x_{k-1})}{\triangle x^2} \\
& + b_k^{n} \frac{V(\tau_{n}, x_{k+1}) - V(\tau_{n}, x_{k-1})}{2\triangle x} - c_k^{n} V(\tau_{n}, x_k) \Big\} \\
= & \frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau} - \frac{1}{2}\frac{\partial V}{\partial \tau}\Big|_{(\tau_{n-1}, x_k)} - \frac{1}{2}\frac{\partial V}{\partial \tau}\Big|_{(\tau_{n}, x_k)} + O(\triangle x^2) \\
= & \frac{\partial V}{\partial \tau}\Big|_{(\tau_{n-1/2}, x_k)} - \frac{1}{2}\frac{\partial V}{\partial \tau}\Big|_{(\tau_{n-1}, x_k)} - \frac{1}{2}\frac{\partial V}{\partial \tau}\Big|_{(\tau_{n}, x_k)} + O(\triangle \tau^2) + O(\triangle x^2)
\end{aligned}
$$

  if we interpret $\frac{V(\tau_n, x_k) - V(\tau_{n-1}, x_k)}{\triangle \tau}$ as a central difference around a fictitious point $(\tau_{n-1/2}, x_k)$

- The first three terms in the above have the form

$$
f(\tau) - \frac{1}{2}[f(\tau + \triangle\tau/2) + f(\tau - \triangle\tau/2)]
$$

  where we can use Taylor expansion to show that this is of order $O(\triangle \tau^2)$. Hence we conclude $T_k^n = O(\triangle \tau^2) + O(\triangle x^2)$

Stability of a finite difference scheme

- Ultimately, we want to see under what conditions the estimates V_k^n converge to $V(\tau_n, x_k)$ when $\triangle\tau, \triangle x \to 0$

Stability of a finite difference scheme

- Ultimately, we want to see under what conditions the estimates V_k^n converge to $V(\tau_n, x_k)$ when $\triangle\tau, \triangle x \to 0$
- Consistency is only a necessary but not a sufficient condition for convergence. It is because the small local errors introduced in each time step of calculation might accumulate during the recursive computation. This is known as the issue of stability

Stability of a finite difference scheme

- Ultimately, we want to see under what conditions the estimates V_k^n converge to $V(\tau_n, x_k)$ when $\triangle\tau, \triangle x \to 0$
- Consistency is only a necessary but not a sufficient condition for convergence. It is because the small local errors introduced in each time step of calculation might accumulate during the recursive computation. This is known as the issue of stability
- Fix some notation as follows:
  - We write a finite difference scheme as

$$V^n = \mathcal{L} V^{n-1},$$

where $\mathcal{L}$ is an operator applied to a vector. Further, let

$$\mathcal{L}^{(n)} := \underbrace{\mathcal{L} \circ \mathcal{L} \circ \cdots \circ \mathcal{L}}_{n \text{ times}}$$

be the n-times convolution of the operator such that $V^n = \mathcal{L}^{(n)} V^0$

Stability of finite difference scheme: a motivating example

- Suppose we are working with an explicit scheme in form of

$$\frac{V_k^{n+1} - V_k^n}{\triangle \tau} = a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^{n-1}}{2\triangle x} - c_k^n V_k^n \tag{3.16}$$

Stability of finite difference scheme: a motivating example

- Suppose we are working with an explicit scheme in form of

$$\frac{V_k^{n+1} - V_k^n}{\triangle\tau} = a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^{n-1}}{2\triangle x} - c_k^n V_k^n \tag{3.16}$$

- Using (3.15) in the local truncation error analysis, we have

$$\begin{aligned}\frac{V(\tau_{n+1}, x_k) - V(\tau_n, x_k)}{\triangle\tau} = \Big[& a_k^n \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} \\ & + b_k^n \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} - c_k^n V(\tau_n, x_k) \Big] + T_k^n \end{aligned} \tag{3.17}$$

Stability of finite difference scheme: a motivating example

- Suppose we are working with an explicit scheme in form of

$$\frac{V_k^{n+1} - V_k^n}{\triangle \tau} = a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^{n-1}}{2\triangle x} - c_k^n V_k^n \tag{3.16}$$

- Using (3.15) in the local truncation error analysis, we have

$$\begin{aligned}\frac{V(\tau_{n+1}, x_k) - V(\tau_n, x_k)}{\triangle \tau} = \Big[& a_k^n \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} \\ & + b_k^n \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} - c_k^n V(\tau_n, x_k) \Big] + T_k^n \end{aligned} \tag{3.17}$$

- If we define $E_k^n := V(\tau_n, x_k) - V_k^n$ to be the error of the numerical estimate, then

$$\frac{E_k^{n+1} - E_k^n}{\triangle \tau} = a_k^n \frac{E_{k+1}^n - 2E_k^n + E_{k-1}^n}{\triangle x^2} + b_k^n \frac{E_{k+1}^n - E_{k-1}^n}{2\triangle x} - c_k^n E_k^n + T_k^n$$

$$\implies \qquad E_k^{n+1} = A_k^n E_{k-1}^n + (1 + B_k^n) E_k^n + C_k^n E_{k+1}^n + \triangle \tau T_k^n \tag{3.18}$$

on subtracting (3.17) by (3.16). Hence the error vector follows exactly the same differencing equation for V_k^n, plus the local truncation error

Stability of finite difference scheme: a motivating example

- Suppose we are working with an explicit scheme in form of

$$\frac{V_k^{n+1} - V_k^n}{\triangle\tau} = a_k^n \frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + b_k^n \frac{V_{k+1}^n - V_{k-1}^{n-1}}{2\triangle x} - c_k^n V_k^n \tag{3.16}$$

- Using (3.15) in the local truncation error analysis, we have

$$\begin{aligned}\frac{V(\tau_{n+1}, x_k) - V(\tau_n, x_k)}{\triangle\tau} = \Big[a_k^n \frac{V(\tau_n, x_{k+1}) - 2V(\tau_n, x_k) + V(\tau_n, x_{k-1})}{\triangle x^2} \\ + b_k^n \frac{V(\tau_n, x_{k+1}) - V(\tau_n, x_{k-1})}{2\triangle x} - c_k^n V(\tau_n, x_k)\Big] + T_k^n \end{aligned} \tag{3.17}$$

- If we define $E_k^n := V(\tau_n, x_k) - V_k^n$ to be the error of the numerical estimate, then

$$\begin{aligned} & \frac{E_k^{n+1} - E_k^n}{\triangle\tau} = a_k^n \frac{E_{k+1}^n - 2E_k^n + E_{k-1}^n}{\triangle x^2} + b_k^n \frac{E_{k+1}^n - E_{k-1}^n}{2\triangle x} - c_k^n E_k^n + T_k^n \\ \Longrightarrow \quad & E_k^{n+1} = A_k^n E_{k-1}^n + (1 + B_k^n) E_k^n + C_k^n E_{k+1}^n + \triangle\tau T_k^n \end{aligned} \tag{3.18}$$

  on subtracting (3.17) by (3.16). Hence the error vector follows exactly the same differencing equation for V_k^n, plus the local truncation error

- (3.18) can be expressed using the vector notation $E^{n+1} = \mathcal{L}E^n + \triangle\tau T^n$. Then

$$\begin{aligned} ||E^n|| &\le ||\mathcal{L}E^{n-1}|| + \triangle\tau||T^{n-1}|| \\ &\le ||\mathcal{L}^{(2)}E^{n-2}|| + \triangle\tau\left[||\mathcal{L}T^{n-2}|| + ||T^{n-1}||\right] \\ &\le \underbrace{||\mathcal{L}^{(n)}E^0||}_{=0} + \triangle\tau \underbrace{\left[||\mathcal{L}^{(n-1)}T^0|| + ||\mathcal{L}^{n-2}T^1|| + \cdots + ||\mathcal{L}T^{n-2}|| + ||T^{n-1}||\right]}_{n \text{ terms}} \end{aligned} \tag{3.19}$$

Convergence of a finite difference scheme

- The following definition of "stability" captures the idea that the local error vector should not grow too fast under repeated iterations

Definition 3.6

A finite difference scheme in form of $V^{n+1} = \mathcal{L}V^n$ *is stable if*

$$||\mathcal{L}^{(n)}U|| \leq C||U||$$

for all n *with* $n\triangle t$ *fixed. Here* C *is a constant independent of* n.

Convergence of a finite difference scheme

- The following definition of "stability" captures the idea that the local error vector should not grow too fast under repeated iterations

Definition 3.6

A finite difference scheme in form of $V^{n+1} = \mathcal{L}V^n$ is stable if

$$||\mathcal{L}^{(n)}U|| \leq C||U||$$

for all n with $n\triangle t$ fixed. Here C is a constant independent of n.

Theorem 3.7 (Lax Equivalence Theorem)

A consistent finite difference scheme is convergent if and only if it is stable.

Sketch of proof (of the 'if' part). (3.19) indeed holds for any finite difference scheme. Then:

$$\begin{aligned}
||E^n|| &\leq \triangle\tau \left[||\mathcal{L}^{n-1}T^0|| + ||\mathcal{L}^{n-2}T^1|| + \cdots + ||\mathcal{L}T^{n-2}|| + ||T^{n-1}||\right] \\
&\leq C\triangle\tau \left[||T^0|| + ||T^1|| + \cdots + ||T^{n-1}||\right] \qquad \text{(stability of the scheme)} \\
&\leq C\triangle\tau n \max_{i=0,\ldots,n-1} ||T^i|| \\
&= CT \max_{i=0,\ldots,n-1} ||T^i|| \to 0
\end{aligned}$$

as $\triangle\tau, \triangle x \to 0$ since $\max_{i=0,\ldots,n-1} ||T^i|| \to 0$ by the consistency of the scheme.

Von Neumann stability analysis

- Fourier series is a classical tool to analyse the solution to the heat equation

Von Neumann stability analysis

- Fourier series is a classical tool to analyse the solution to the heat equation
- This relies on linear constant coefficient structure of the PDE to achieve a separation of variables (in time and space) and the solution is constructed as superposition of some simple exponential/trigonometric functions

Von Neumann stability analysis

- Fourier series is a classical tool to analyse the solution to the heat equation
- This relies on linear constant coefficient structure of the PDE to achieve a separation of variables (in time and space) and the solution is constructed as superposition of some simple exponential/trigonometric functions
- We assume that the numerical scheme admits a solution in form of

$$V_k^n = R^n e^{ik\omega\triangle x}$$

where $i := \sqrt{-1}$ and $R = R(\triangle\tau, \triangle x, \omega)$ is the amplification factor of the Fourier component; ω is the wave-number

Von Neumann stability analysis

- Fourier series is a classical tool to analyse the solution to the heat equation
- This relies on linear constant coefficient structure of the PDE to achieve a separation of variables (in time and space) and the solution is constructed as superposition of some simple exponential/trigonometric functions
- We assume that the numerical scheme admits a solution in form of

$$V_k^n = R^n e^{ik\omega\triangle x}$$

where $i := \sqrt{-1}$ and $R = R(\triangle\tau, \triangle x, \omega)$ is the amplification factor of the Fourier component; ω is the wave-number

- Based on Definition 3.6, we expect the scheme is stable if

$$|R(\triangle\tau, \triangle x, \omega)|^n \leq C \tag{3.20}$$

for all ω and n where C is a constant independent of n with $n\triangle\tau = T$ fixed. A simple sufficient condition of (3.20) is that $|R(\triangle\tau, \triangle x, \omega)| \leq 1$

Stability of explicit scheme for heat equation

- Consider an explicit scheme applied to a heat equation in form of

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

such that the differencing equation is

$$\begin{aligned}
& \frac{V_k^{n+1} - V_k^n}{\triangle \tau} = \frac{V_{k-1}^n - 2V_k^n + V_{k+1}^n}{\triangle x^2} \\
\Longrightarrow \quad & V_k^{n+1} = V_k^n + \lambda[V_{k-1}^n - 2V_k^n + V_{k+1}^n] \qquad \text{with } \lambda := \frac{\triangle \tau}{\triangle x^2} \\
\Longrightarrow \quad & R^{n+1} e^{i\omega k \triangle x} = R^n e^{i\omega k \triangle x} + \lambda[R^n e^{i(k-1)\omega \triangle x} - 2R^n e^{i\omega k \triangle x} + R^n e^{i(k+1)\omega \triangle x}] \\
\Longrightarrow \quad & R = 1 + \lambda[e^{-i\omega \triangle x} - 2 + e^{i\omega \triangle x}] \\
& \quad = 1 + 2\lambda[\cos \omega \triangle x - 1] \\
& \quad = 1 - 4\lambda \sin^2(\omega \triangle x/2)
\end{aligned}$$

- To ensure $|R| \leq 1$ we need $1 - 4\lambda \geq -1$ which gives $\lambda \leq 1/2 \iff \frac{\triangle \tau}{\triangle x^2} \leq 1/2$
- The explicit scheme for solving the heat equation is conditionally stable provided that $\triangle \tau \leq \frac{\triangle x^2}{2}$

Stability of fully implicit scheme for heat equation

- Suppose now an implicit scheme is applied to solve the heat equation such that the differencing equation is

$$\frac{V_k^n - V_k^{n-1}}{\triangle\tau} = \frac{V_{k-1}^n - 2V_k^n + V_{k+1}^n}{\triangle x^2}$$

$$\implies \quad V_k^n = V_k^{n-1} + \lambda[V_{k-1}^n - 2V_k^n + V_{k+1}^n] \qquad \text{with } \lambda := \frac{\triangle\tau}{\triangle x^2}$$

$$\implies \quad R^n e^{i\omega k\triangle x} = R^{n-1} e^{i\omega k\triangle x} + \lambda[R^n e^{i(k-1)\omega\triangle x} - 2R^n e^{i\omega k\triangle x} + R^n e^{i(k+1)\omega\triangle x}]$$

$$\implies \quad R = 1 + R\lambda[e^{-i\omega\triangle x} - 2 + e^{i\omega\triangle x}]$$

$$\implies \quad R = \frac{1}{1 - 2\lambda(\cos\omega\triangle x - 1)}$$

$$= \frac{1}{1 + 4\lambda\sin^2(\omega\triangle x/2)}$$

  and as such $|R| \leq 1$ for any value of ω and λ

- We say that the implicit scheme for solving the heat equation is unconditionally stable

Summary

- With Feynman-Kac formula, it is easy to formulate an option pricing problem under various stochastic models via the PDE method
- Finite difference method returns a vector of option prices at multiple stock price levels and multiple time points (useful for risk or "greeks" computation). In contrast, lattice method only returns option prices at a small number of nodes
- While an explicit scheme is much easier to be implemented, stability of the scheme is not guaranteed in general
- An implicit scheme has better stability, but some hard work has to be done around matrix inversion. Moreover, it is much more subtle to apply an implicit scheme to price an American option (to be explored in the next lecture)

Optional reading

- Wilmott, P., Howson, S., Howison, S., & Dewynne, J. (1995). The Mathematics of Financial Derivatives: A Student Introduction. Chapter 3, 4 & 8.
- Morton, K. W., & Mayers, D. F. (2005). Numerical Solution of Partial Differential Equations: an Introduction. Chapter 2.