Numerical Methods for Financial Derivatives

Hwan C. Lin
Department of Economics
UNC Charlotte

Lecture 11: Direct Solvers for Systems of Difference Equations

Introduction

- Direct Solvers - Gaussian Elimination
    - LU Decomposition of a General Matrix
    - LU Decomposition of a Tridiagonal Matrix (The Thomas Algorithm)
- Iterative Solvers
    - Jacobi's method
    - Gauss-Seidel Method
    - SOR (Successive Overrelaxation)

Linear System

- Linear System: $A \cdot x = b$

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_x = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_x = b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_x = b_n$$

- Matrix Form:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{b}$$

- **Forward substitutttion:** Change $A \cdot x = b$ to $U \cdot x = \widehat{b}$:

$$\underbrace{\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} \widehat{b}_1 \\ \widehat{b}_2 \\ \vdots \\ \widehat{b}_n \end{bmatrix}}_{\widehat{b}}$$

where U is called *upper triangular* and $A = L \cdot U$.

- **Backward substitution:** $x_n = \widehat{b}_n / u_{nn}$
  - $u_{n-1,n-1} x_{n-1} + u_{n-1,n} x_n = \widehat{b}_{n-1} \Rightarrow x_{n-1} = \frac{\widehat{b}_{n-\mathbf{1}} - u_{n-\mathbf{1},n} x_n}{u_{n-\mathbf{1},n-\mathbf{1}}}$
  - $u_{n-2,n-2} x_{n-2} + u_{n-2,n-1} x_{n-1} + u_{n-2,n} x_n = \widehat{b}_{n-2}$
  - $\Rightarrow x_{n-2} = \frac{\widehat{b}_{n-\mathbf{2}} - u_{n-\mathbf{2},n-\mathbf{1}} x_{n-\mathbf{1}} - u_{n-\mathbf{2},n} x_n}{u_{n-\mathbf{2},n-\mathbf{2}}}$

Before forward substitution, we need to guard against "dividing by zero" or "ill-conditioned" matrix:

- Partial pivoting: place a coefficient of larger magnitude on the diagonal by row interchanges
- Scaling: scale the coefficients of equations by dividing each row by the largest coefficient

Example: Gaussian Elimination - LU Decomposition of A

- Example:

$$\begin{bmatrix} 4 & -2 & 1 & | & 15 \\ -3 & -1 & 4 & | & 8 \\ 1 & -1 & 3 & | & 13 \end{bmatrix}$$

$$\begin{matrix} \\ (3/4)R_1 + R_2 \rightarrow \\ -(1/4)R_1 + R_3 \rightarrow \end{matrix} \begin{bmatrix} 4 & -2 & 1 & | & 15 \\ 0 & -2.5 & 4.75 & | & 19.25 \\ 0 & -0.5 & 2.75 & | & 9.25 \end{bmatrix}$$

$$\begin{matrix} \\ \\ -(-0.5/-2.5)R_2 + R_3 \rightarrow \end{matrix} \begin{bmatrix} 4 & -2 & 1 & | & 15 \\ 0 & -2.5 & 4.75 & | & 19.25 \\ 0 & 0.0 & 1.80 & | & 5.40 \end{bmatrix}$$

$$\Rightarrow x_3 = 3, x_2 = -2, x_1 = 2$$

$$\begin{bmatrix} 4 & -2 & 1 \\ (-0.75) & -2.5 & 4.75 \\ (0.25) & (0.2) & 1.80 \end{bmatrix} \Rightarrow A = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -0.75 & 1 & 0 \\ 0.25 & 0.2 & 1 \end{bmatrix}}_{L} \cdot \underbrace{\begin{bmatrix} 4 & -2 & 1 \\ 0 & -2.5 & 4.75 \\ 0 & 0 & 1.80 \end{bmatrix}}_{U}$$

LU Decomposition of a Tridiagonal Matrix

- Tridiagonal Matrix

$$\begin{bmatrix} \alpha_1 & \beta_1 & & & 0 \\ \gamma_2 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-1} & \alpha_{n-1} & \beta_{n-1} \\ 0 & & & \gamma_n & \alpha_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

Solution

[Thomas Algorithm] $\widehat{\alpha}_1 = \alpha_1, \widehat{b}_1 = b_1$

(forward loop) for $i = 2, \ldots, n$:

$$\widehat{\alpha}_i = \alpha_i - \beta_{i-1}\left(\frac{\gamma_i}{\widehat{\alpha}_{i-1}}\right), \widehat{b}_i = b_i - \widehat{b}_{i-1}\left(\frac{\gamma_i}{\widehat{\alpha}_{i-1}}\right)$$

(backward loop) for $i = n\text{-}1, \ldots, 1$:

$$x_n = \widehat{b}_n / \widehat{\alpha}_n$$

$$x_i = (\widehat{b}_i - \beta_i x_{i+1}) / \widehat{\alpha}_i$$

The Heat Equation

Problem

$$\frac{\partial y(x,\tau)}{\partial \tau} = \frac{\partial^2 y(x,\tau)}{\partial x^2}$$

$$y(x,0) = \sin \pi x, \quad 0 < x < 1$$

$$y(0,\tau) = y(1,\tau) = 0, \quad \tau > 0$$

Solution

$$y(x,\tau) = e^{-\pi^2 \tau} \sin \pi x$$

How to use the Thomas Algorithm to solve the heat equation?

Explicit Scheme

$$
\begin{bmatrix} w_{1,i+1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ w_{N-1,i+1} \end{bmatrix}
=
\underbrace{\begin{bmatrix}
1-2\lambda & \lambda & 0 & \cdots & \cdots & 0 \\
\lambda & 1-2\lambda & \lambda & \ddots & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & \ddots & \lambda & 1-2\lambda & \lambda \\
0 & \cdots & \cdots & 0 & \lambda & 1-2\lambda
\end{bmatrix}}_{A_R,\ (N-1)\times(N-1)}
\begin{bmatrix} w_{1,i} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ w_{N-1,i} \end{bmatrix}
$$

Implicit Scheme

- Solve $A_L \cdot w^{(i+1)} = w^{(i)}$:

$$\underbrace{\begin{bmatrix} 1+2\lambda & -\lambda & 0 & \cdots & \cdots & 0 \\ -\lambda & 1+2\lambda & -\lambda & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -\lambda & 1+2\lambda & -\lambda \\ 0 & \cdots & \cdots & 0 & -\lambda & 1+2\lambda \end{bmatrix}}_{A_L} \underbrace{\begin{bmatrix} w_{1,i+1} \\ w_{2,i+1} \\ \vdots \\ \vdots \\ w_{N-2,i+1} \\ w_{N-1,i+1} \end{bmatrix}}_{w^{(i+\mathbf{1})}} = \underbrace{\begin{bmatrix} w_{1,i} \\ w_{2,i} \\ \vdots \\ \vdots \\ w_{N-2,i} \\ w_{N-1,i} \end{bmatrix}}_{w^{(i)}}$$

- How to apply the Thomas Algorithm?

Crank-Nicolson Scheme

- Solve $A_L \cdot w^{(i+1)} = A_R \cdot w^{(i)}$:

$$A_L = \begin{bmatrix} 1+\lambda & -\lambda/2 & 0 & \cdots & \cdots & 0 \\ -\lambda/2 & 1+\lambda & -\lambda/2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -\lambda/2 & 1+\lambda & -\lambda/2 \\ 0 & \cdots & \cdots & 0 & -\lambda/2 & 1+\lambda \end{bmatrix}_{(N-\mathbf{1})\times(N-\mathbf{1})}$$

$$A_R = \begin{bmatrix} 1-\lambda & \lambda/2 & 0 & \cdots & \cdots & 0 \\ \lambda/2 & 1-\lambda & \lambda/2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \lambda/2 & 1-\lambda & \lambda/2 \\ 0 & \cdots & \cdots & 0 & \lambda/2 & 1-\lambda \end{bmatrix}_{(N-\mathbf{1})\times(N-\mathbf{1})}$$

- How to apply the Thomas algorithm?

Nonlinear System

- How to solve nonlinear system,

$$f(x) = 0,\ x \in R^n$$

- Taylor series expansion (assuming $x \in \mathscr{R}$):

$$f(x_0 + \triangle x) = f(x_0) + J(x_0)\triangle x + O(\triangle x^2),\ \ x \in R^n$$

$$f(x_0 + \triangle x) = f(x_0) + f'(x_0)\triangle x + O(\triangle x^2),\ \ x \in R$$

$$\triangle x = -\frac{f(x_0)}{f'(x_0)} \quad \text{or} \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

- Newton-Ralphson's method: setting $f(x_0 + \triangle x) = 0$,

$$J(x^{(k)})\triangle x = -f(x^{(k)}),\ \ \triangle x = x^{(k+1)} - x^{(k)}$$

$$\triangle x = -\frac{f(x^{(k)})}{f'(x^{(k)})}, \quad \text{or} \quad x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})},\ \ k = 0, 1, \ldots, k_{max}$$

- Secant Method:

$$x^{(k+1)} = x^{(k)} - f(x^{(k)})\frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})},\ \ k = 0, 1, \ldots, k_{max}$$

Newton-Ralphson Algorithm

Newton-Ralphson Algorithm

1. Input: initial guess, $x^{(0)}$.
2. Evaluate $f(x^{(k)}),\ k = 0, 1, \ldots$.
3. Compute the Jacobian matrix $J(x^{(k)}) \equiv \left[f_{ij}\right]_{n\times n},\ f_{ij} = \frac{\partial f_i}{\partial x_j}$ [or compute $f'(x^{(k)})$ *if*]
4. Solve $J(x^{(k)})\triangle x = -f(x^{(k)})$ for $\triangle x$ [or compute $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$]
5. Update $x^{(k+1)} = x^{(x)} + \triangle x$, and repeat steps 2 - 5, until either $|\triangle x| <$ error tolerance or $|f(x^{(k)})| <$ error tolerance.

Note: In a muti-dimensional case, it is preferrable to let the computer calculate the partial derivatives using the finite difference approximation,

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(x + e_j h) - f_i(x)}{h}$$

where e_i is a unit vector in the direction of x_j.

Application to Implied Volatility

- Valuation formula of standard European options:

$$V_C^{eur} = v(S,\tau,K,r,\delta,\sigma) = S_t e^{-\delta(T-t)}F(d_1) - Ke^{-r(T-t)}F(d_2),$$
$$V_P^{eur} = v(S,\tau,K,r,\delta,\sigma) = -S_t e^{-\delta(T-t)}F(-d_1) + Ke^{-r(T-t)}F(-d_2)$$

- Model calibration: Suppose that actual market data V^{mar} of the prices are known. Then if one of the parameters is unknown, it can be fixed via the implicit equation,

$$V^{mar} - v(S,\tau,K,r,\delta,\sigma) = 0$$

- Implied volatility: If σ is the unknown parameter, then the zero of

$$f(\sigma) \equiv V^{mar} - v((S,\tau,K,r,\delta,\sigma)$$

is call "implied volatility."

Volatility Smile

- Problem: Use Newton-Ralphson's method to construct a sequence $x^{(k)} \to \sigma$ for the case of V_C^{eur} or V_P^{eur}:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

  where the derivative $f'(x^{(k)}$ can be approximated by the difference quotient

$$\frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

- For the resulting secant iterations, apply the stopping criterion that requires smallness of both $|f(x^{(k)}|$ and $|x^{(k)} - x^{(k-1)}|$.
- Calculate the implied volatilites for the data, $\tau = T - t = 0.211$, $S_0 = 5290.36$, $r = 0.0328$, $\delta = 0$, and the pairs of (K, V):

K	6000	6200	6300	6350	6400	6600	6800
V	80.2	47.1	35.9	31.3	27.7	16.6	11.4

- Plot a convex curve, called "volatility smile," by connecting the points of (K, σ).