6. Finite Difference Methods: Dealing with American Option
References

1. Chapters 5 and 9, Brandimarte’s
2. Chapters 6, 7, 20, and 21, “Option Pricing”
6. Finite Difference Methods: Dealing with American Option

6.1 American call options
Constraints on American Options

An American option valuation problem is uniquely defined by a set of constraints:

- Lower bound: \( V(S,t) \geq \max (K - S, 0) \)
- Black-Scholes equation is replaced by an inequality
- \( V(S_f, t) \) is continuous at \( S = S_f \)
- The option delta (its slope) must be continuous at \( S = S_f \)

Let’s consider American put options as an example
American put options

• To avoid arbitrage, American put options must satisfy
  \[ P(S,t) \geq \max (K - S, 0) \]

• It is optimal to exercise American put options early if \( S \) is sufficiently small

• When the option is exercised early, \( P(S, t) = K-S \) and the B-S inequality holds; otherwise, \( P(S, t) > K-S \) and the B-S equality holds.

• Again, there is a unknown exercise boundary \( S_f(t) \), where option should be exercised if \( S < S_f(t) \) and held otherwise
For each time \( t \), we must divide the \( S \) axis into two regions:

i) \( 0 \leq S < S_f \) where early exercise is optimal:

\[
P = K - S, \quad \frac{\partial P}{\partial t} + \frac{\sigma^2 S^2}{2} \frac{\partial^2 P}{\partial S^2} + rS \frac{\partial P}{\partial S} - rP < 0
\]

ii) \( S_f \leq S < +\infty \) where retaining the option is optimal:

\[
P > K - S, \quad \frac{\partial P}{\partial t} + \frac{\sigma^2 S^2}{2} \frac{\partial^2 P}{\partial S^2} + rS \frac{\partial P}{\partial S} - rP = 0
\]

with boundary conditions at \( S = S_f(t) \).
Black-Scholes Inequality

For American put $P(S,t)$:

$$\frac{\partial P}{\partial t} + \frac{\sigma^2 S^2}{2} \frac{\partial^2 P}{\partial S^2} + rS \frac{\partial P}{\partial S} - rP \leq 0$$

In other words,

the return from the portfolio $\leq$ the return from a bank deposit
Taking consideration of early exercise

For a vanilla American option, we can check the possibility of early exercise easily in an explicit scheme:

\[ f_{ij} = \max \left( f_{ij}, K - j\Delta S \right) \]

But this is difficult to do in an implicit scheme as computing \( f_{ij} \) requires knowing the other \( f_{ij} \)'s.

To get around this difficulty, we can use iterative method to solve the linear system. Here we consider SOR method.
\[ M_1 f_i = r_i \equiv M_2 f_{i+1} + b \]
where \( b = \alpha_1 \left[ f_{i,0} + f_{i+1,0}, 0, \ldots, 0 \right]^T \). Recall that both \( M_1 \) and \( M_2 \) are tri-diagonal matrices.

Set
\[ g_j = K - j \Delta S \]
be the intrinsic value when \( S = j \Delta S \) for \( j = 1, \ldots, M - 1 \).
Crank-Nicolson Methods + Projected SOR to Price an American Put Option

For each time layer $i$, we have the iterative scheme

\[
\begin{align*}
    f_{i,1}^{(k+1)} &= \max \left\{ g_1, f_{i,1}^{(k)} + \frac{\omega}{1 - \beta_1} \left[ r_1 - (1 - \beta_1) f_{i,1}^{(k)} + \gamma_1 f_{i,2}^{(k)} \right] \right\} \\
    f_{i,2}^{(k+1)} &= \max \left\{ g_2, f_{i,2}^{(k)} + \frac{\omega}{1 - \beta_2} \left[ r_2 + \alpha_2 f_{i,1}^{(k+1)} - (1 - \beta_2) f_{i,2}^{(k)} + \gamma_2 f_{i,3}^{(k)} \right] \right\} \\
    &\vdots \\
    f_{i,M-1}^{(k+1)} &= \max \left\{ g_{M-1}, f_{i,M-1}^{(k)} + \frac{\omega}{1 - \beta_{M-1}} \left[ r_{M-1} + \alpha_{M-1} f_{i,M-2}^{(k+1)} - (1 - \beta_{M-1}) f_{i,M-1}^{(k)} \right] \right\}
\end{align*}
\]
Note on Implementation

When passing from a time layer to the next one, it may be reasonable to initialize the iteration with a guess to the values of the previous time layer.
Example

We compare Crank-Nicolson methods + Projected SOR for an American put, where $T = 5/12$ yr, $S_0 = $50, $K = $50, $\sigma = 40\%$, $r = 10\%$. ($\omega = 1.2$, tol = 0.001)

- CK Method with $S_{\text{max}} = $100, $\Delta S = 1$, $\Delta t = 1/600$: $4.2800$
- CK Method with $S_{\text{max}} = $100, $\Delta S = 1$, $\Delta t = 1/1200$: $4.2828$
Example (Stability)

We compare Crank-Nicolson methods + Projected SOR for an American put, where $T = 5/12 \text{ yr}$, $S_0 = $50, $K = $50, $\sigma = 40\%$, $r = 10\%$. ($\omega = 1.2$, $\text{tol} = 0.001$)

CK Method with $S_{\text{max}} = $100, $\Delta S=1$, $\Delta t=1/600$: $4.2800$
CK Method with $S_{\text{max}} = $100, $\Delta S=1$, $\Delta t=1/100$: $4.2778$
American call option with dividend-paying as a free boundary problem

For each time $t$, we must divide the $S$ axis into two regions:

i) $0 \leq S < S_f$ where retaining option is optimal:

$$C > S - K, \quad \frac{\partial C}{\partial t} + \frac{\sigma^2 S^2}{2} \frac{\partial^2 C}{\partial S^2} + (r - D_0) S \frac{\partial C}{\partial S} - rC = 0$$

ii) $S_f \leq S < +\infty$ where early exercise is optimal:

$$C = S - K, \quad \frac{\partial C}{\partial t} + \frac{\sigma^2 S^2}{2} \frac{\partial^2 C}{\partial S^2} + (r - D_0) S \frac{\partial C}{\partial S} - rC < 0$$

with boundary conditions at $S = S_f(t)$. 
6. Finite Difference Methods: Dealing with American Option

6.2 Iterative Methods of Solving a Linear System
Motivation: Direct vs Iterative Methods

**Direct Methods:**
- determine exact solution subject only to round-off error and involves factorization of matrix $A$
- pick up an appropriate method and adapt it to exploit $A$’s sparsity
- impractical if $A$ is large and sparse

**Iterative Methods:**
- generate a sequence of approximate solutions and involve matrix-vector multiplications
- whether it converges to the exact solution?
- how fast does it converge to the exact solution?
Basic ideas of iterative methods

A possible approach to generate a sequence of approximations to the solution to $x = G(x)$ is the iteration scheme

$$x^{(k)} = G(x^{(k-1)})$$

Starting from initial approximation $x^{(0)}$.

Similarly, we can rewritten $Ax=b$ as

$$x = (A+I) x - b = \hat{A} x - b$$

where $G(x) = \hat{A} x - b$. Using the previous iteration scheme, we have

$$x^{(k)} = \hat{A} x^{(k-1)} - b$$
Starting from initial approximation $x^{(0)}$, we have

$$x^{(1)} = \hat{A} x^{(0)} - b$$
$$x^{(2)} = \hat{A} x^{(1)} - b = \hat{A}^2 x^{(1)} - \hat{A} b - b$$

... 

This iteration scheme diverge if some elements of $\hat{A}^n$ grow without bound as $n \to \infty$

It converges only if $\rho(\hat{A}) < 1$

However, arbitrary systems of equations may often not satisfy this condition
Basic Ideas of Iterative Methods

A slightly different approach:
Transforming the system $Ax = b$ to an equivalent system

$$Mx = -Nx + b$$

where $A = M + N$, called a splitting of $A$.
The corresponding iteration scheme is

$$Mx^{(k)} = -Nx^{(k-1)} + b$$

i.e.  $x^{(k)} = -M^{-1}Nx^{(k-1)} + M^{-1}b$

The flexibility in choosing $M$ may be exploited to improve convergence
Convergence:

Let \( B = -M^{-1}N = I - M^{-1}A \). Let check the error

\[
e^{(k)} = x - x^{(k)} = B(x - x^{(k-1)}) = Be^{(k-1)}
\]

\[
\therefore \lim_{k \to \infty} e^{(k)} = \lim_{k \to \infty} B^k e^{(0)}
\]

It can be proved that \( \lim_{k \to \infty} B^k = 0 \) iff \( \rho(B) < 1 \).

To avoid computing eigenvalues, we may require

\[
\rho(B) \leq \|B\| < 1
\]

instead.
Implementation: check for convergence

Usually one or a combination of the four common tests is used to check convergence:

- Absolute difference:
  \[ \| x^{(k)} - x^{(k-1)} \| < \varepsilon_1 \]

- Relative difference:
  \[ \frac{\| x^{(k)} - x^{(k-1)} \|}{\| x^{(k-1)} \|} < \varepsilon_2 \]

- Absolute residual:
  \[ \| r(x^{(k)}) \| < \varepsilon_3 \quad \text{where} \quad r(x^{(k)}) = b - Ax^{(k)} \]

- Relative residual:
  \[ \frac{\| r(x^{(k)}) \|}{\| b \|} < \varepsilon_4 \]
Therefore, various iterative methods are developed along the following lines:

• A splitting $A = M+N$ is proposed where linear system of the form $Mz = d$ are easy to solve

• Classes of matrices are identifies for which the iteration matrix $B = (-M^{-1}N)$ satisfies $\rho(B) < 1$

• Further effort are studies to make $\rho(B)$ smaller than 1 as possible so that the error $e^{(k)}$ tends to zero faster.
Jacobi methods

Jacobi, Gauss-Seidel, Successive Over-Relaxation (SOR) methods are commonly used and can be described in following forms.

Let \( A = D + L + U \) (as shown in the figure)

**Jacobi:**

\[
Dx^{(k)} = -(L + U)x^{(k-1)} + b
\]

\[ A = \]

\[ U \]

\[ L \]
Example

Consider the 3-by-3 example:

\[
\begin{bmatrix}
3 & -1 & 0 \\
-1 & 3 & -1 \\
0 & -1 & 3
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}
\]

**Jacobi Method:**

Initial guess: \( \mathbf{x}^{(0)} = [0,0,0]^T \). For \( k = 1, 2, \ldots \)

\[
\begin{bmatrix}
3 & 0 & 0 \\
0 & 3 & 0 \\
0 & 0 & 3
\end{bmatrix}
\begin{bmatrix}
x_1^{(k)} \\
x_2^{(k)} \\
x_3^{(k)}
\end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x_1^{(k-1)} \\
x_2^{(k-1)} \\
x_3^{(k-1)}
\end{bmatrix} +
\begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}
\]

until there is little change for all nodes
Implementation

In general, consider $\mathbf{Ax} = \mathbf{b} \iff \sum_{j} a_{ij} x_j = b_i$ for $i = 1, \ldots, n$

**Jacobi Method:**

give an initial guess: $\mathbf{x}^{(0)} = (x_i^{(0)})$

for $k = 1, \ldots, \text{maxiter}$

for $i = 1, \ldots, n$,

$$b_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)}$$

$$x_i^{(k)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)}}{a_{ii}}$$

end

check for convergence

end
function [x, error, nIter] = Jacobi(A,b,x0,maxIter)

x = zeros(size(x0));
n = length(b); error = 1;
epi = 1E-6;
nIter = 0;

while ((error > epi)& (nIter < maxIter)),
    nIter = nIter + 1;
    for i = 1:n,
        x(i) = b(i) - A(i, 1:i-1) * x0(1:i-1)-A(i, i+1:n)*x0(i+1:n);
        if (abs(A(i, i)) > 1E-10),
            x(i) = x(i)/A(i, i);
        else
            'Error: diagonal is close to zero'
        end
    end
    error = norm(x - x0, inf); x0 = x;
end
Numerical example

\[ A = \begin{bmatrix} 3 & -1 \\ -1 & \ddots & \ddots \\ & \ddots & \ddots & -1 \\ -1 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ 40 \end{bmatrix}, \quad \text{and } b = Ax \]

Initial guess: \( x^{(0)} = [0, \cdots, 0]^T \); Convergence test: \( \| x^{(k)} - x^{(k-1)} \| < 10^{-6} \)

<table>
<thead>
<tr>
<th>Method</th>
<th>No of Iterations</th>
<th>Infinity norm of abs error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi</td>
<td>41</td>
<td>8.24E-7</td>
</tr>
</tbody>
</table>
Gauss-Seidel methods

Jacobi, Gauss-Seidel, Successive Over-Relaxation (SOR) methods are commonly used and can be described in following forms.

Let $A = D + L + U$ (as shown in the figure)

- **Jacobi:**
  \[ D x^{(k)} = -(L + U)x^{(k-1)} + b \]

- **Gauss-Seidel:**
  \[ (D + L)x^{(k)} = -(U)x^{(k-1)} + b \]
Consider the 3-by-3 example:

\[
\begin{bmatrix}
3 & -1 & 0 \\
-1 & 3 & -1 \\
0 & -1 & 3
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
= 
\begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}
\]

**Gauss-Seidel Method:**

Initial guess: \( x^{(0)} = [0,0,0]^T \). For \( k = 1, 2, \ldots \)

\[
\begin{bmatrix}
3 & 0 & 0 \\
-1 & 3 & 0 \\
0 & -1 & 3
\end{bmatrix}
\begin{bmatrix}
x_1^{(k)} \\
x_2^{(k)} \\
x_3^{(k)}
\end{bmatrix}
= 
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_1^{(k-1)} \\
x_2^{(k-1)} \\
x_3^{(k-1)}
\end{bmatrix}
+ 
\begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}
\]

until there is little change for all the unknowns.
Implementation

In general, consider \( \mathbf{Ax} = \mathbf{b} \iff \sum_j a_{ij} x_j = b_i \) for \( i = 1, \ldots, n \)

**Gauss-Seidel Method:**

give an initial guess: \( \mathbf{x}^{(0)} = (x_i^{(0)}) \)

for \( k = 1, \ldots, \text{maxiter} \)

for \( i = 1, \ldots, n, \)

\[
x_i^{(k)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)}}{a_{ii}}
\]

end

check for convergence

end
function [x, error, nIter] = GaussSeidel(A,b,x0,maxIter)

x = zeros(size(x0));
n = length(b);
error = 1;
epi = 1E-6;
nIter = 0;

while ((error > epi)& (nIter < maxIter)),
    nIter = nIter + 1;
    for i = 1:n,
        x(i) = b(i) - A(i, 1:i-1) * x(1:i-1) - A(i, i+1:n)*x0(i+1:n);
        if (abs(A(i, i)) > 1E-10),
            x(i) = x(i)/A(i, i);
        else
            'Error: diagonal is close to zero'
        end
    end
    error = norm(x - x0, inf); x0 = x;
end
Numerical example

\[ A = \begin{bmatrix} 3 & -1 \\ -1 & \ddots & \ddots \\ \ddots & \ddots & -1 \\ -1 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ 40 \end{bmatrix}, \quad \text{and } b = Ax \]

Initial guess: \( x^{(0)} = [0, \cdots, 0]^T \); Convergence test: \( \|x^{(k)} - x^{(k-1)}\| < 10^{-6} \)

<table>
<thead>
<tr>
<th>Method</th>
<th>No of Iterations</th>
<th>Infinity norm of abs error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi</td>
<td>41</td>
<td>8.24e-7</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>25</td>
<td>9.69e-7</td>
</tr>
</tbody>
</table>
Definition

Let $A = (a_{ij})$.

Matrix $A$ is strictly diagonally dominant if and only if for all $i$

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

Example: which ones are strictly diagonally dominant and why?

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 \end{bmatrix} \quad \begin{bmatrix} 2 & 1 \\ -1 & 3 & 2 \\ 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 2 & 1 \\ 1 & 3 & 1 \\ 1 & 2 \end{bmatrix}$$
Assessment: Existence

**Theorem**
Consider $Ax = b$.

If $A$ is strictly diagonally dominant, then there is a unique solution.
Assessment: Convergence

**Theorem**
Consider $Ax = b$.

If $A$ is strictly diagonally dominant, then for any initial guess $x^0$, both Jacobi and the Guess - Seidel algorithms converge to the exact solution.

Prove that $\|M^{-1}N\| < 1$
More on Gauss-Seidel methods

Let $A = D + L + U$ (as shown in the figure)

Gauss-Seidel (Forward):

$$ (D + L)x^{(k)} = (-U)x^{(k-1)} + b $$

where the correction order is $x_1, x_2, \ldots, x_n$

Gauss-Seidel (Backward):

$$ (D + U)x^{(k)} = (-L)x^{(k-1)} + b $$

where the correction order is $x_n, x_{n-1}, \ldots, x_1$

Gauss-Seidel (Symmetric):

It consists of a forward sweep followed by a backward sweep
Define the residual vector \( \mathbf{r} \) of an approximate \( \tilde{x} \) with respect to the system \( \mathbf{A}x = b \) as the following:

\[ \mathbf{r} = b - \mathbf{A}\tilde{x} \]

Denote \( \mathbf{r}_i^{(k)} = [r_{1i}^{(k)}, r_{2i}^{(k)}, \ldots, r_{ni}^{(k)}]^T \) as the residual vector for the Gauss-Seidel method corresponding to the approximation \( \mathbf{x}_i^{(k)} = [x_1^{(k)}, x_2^{(k)}, \ldots, x_{i-1}^{(k)}, x_i^{(k-1)}, \ldots, x_n^{(k-1)}]^T \).

Then Gauss-Seidel method can be characterized as choosing \( x_i^{(k)} \) to satisfy

\[ x_i^{(k)} = x_i^{(k-1)} + \frac{r_i^{(k)}}{a_{ii}} \]

or choosing \( x_{i+1}^{(k)} \) in such a way that \( r_{i,i+1}^{(k)} = 0 \).
More on Gauss-Seidel methods

Choosing $x_{i+1}^{(k)}$ in such a way that the $i$th component of $r_{i,i+1}^{(k)} = 0$, however, is not the most efficient way to reduce the norm of the vector $r_{i+1}^{(k)}$. If we modify the Gauss-Seidel procedure to

$$x_i^{(k)} = x_i^{(k-1)} + \omega \frac{r_{ii}^{(k)}}{a_{ii}}$$

for certain choices of positive $\omega$, then we can reduce the norm of the residual vector and obtain significantly faster convergence.
Relaxation methods

Methods involving choosing proper values of $\omega$ for

$$x_i^{(k)} = x_i^{(k-1)} + \omega \frac{r_{ii}^{(k)}}{a_{ii}} \quad (5.5.1)$$

to reduce the norm of the residual vector and speed up convergence are called "Relaxation Methods"

1. For $0 < \omega < 1$, it is called "under-relaxation methods" and can be used to obtain convergence of systems failed by Gauss-Seidel method.
2. For $1 < \omega$, it is called "over-relaxation methods" and can be used to accelerate the convergence for systems that are convergent by Gauss-Seidel method.

These methods are called called "Successive Over-Relaxation" (SOR).
We reformulate Eq. (5.5.1):

\[ x_i^{(k)} = (1 - \omega) x_i^{(k-1)} + \frac{\omega}{a_{ii}} \left[ b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right] \]  

(5.5.2)

Therefore, SOR methods can be characterized as a linear combination of old and Gauss-Seidel approximations:

\[ x^{(k)} = (1 - \omega) x^{(k-1)} + \omega x_{GS}^{(k)} \]

for \(0 < \omega < 2\)
Matrix version of SOR method

To determine the matrix version of SOR, we rewrite Eq. (5.5.2):

\[ a_{ii}x_i^{(k)} + \omega \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} = (1 - \omega) a_{ii}x_i^{(k-1)} - \omega \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} + \omega b_i \quad (5.5.3) \]

so that in the matrix and vector form, we have

\[ \text{SOR : } (D + \omega L)x^{(k)} = ( (1 - \omega)D - \omega U)x^{(k-1)} + \omega b \]

When \( \omega = 1 \), SOR methods reduces to the Gauss-Seidel methods

\[ \text{Gauss - Seidel : } (D + L)x^{(k)} = -Ux^{(k-1)} + b \]
Consider the 3-by-3 example:

\[
\begin{bmatrix}
  3 & -1 & 0 \\
  -1 & 3 & -1 \\
  0 & -1 & 3 \\
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
\end{bmatrix} =
\begin{bmatrix}
  1 \\
  1 \\
  1 \\
\end{bmatrix}
\]

**SOR Method** (Leave as an exercise)
Implementation

Successive Over-Relaxation Method:

give an initial guess: \( \mathbf{x}^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \end{pmatrix} \)

for \( k = 1, \ldots, \text{maxiter} \)

for \( i = 1, \ldots, n \),

\[
x_i^{(k)} = \omega \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)}}{a_{ii}} + (1 - \omega) x_i^{(k-1)}
\]

end

check for convergence

end
function [x, error, nIter] = SOR(A,b,x0,maxIter,omega)

    x = zeros(size(x0));
    n = length(b);
    error = 1;
    epi = 1E-6;
    nIter = 0;

    while ((error > epi) & (nIter < maxIter)),
        nIter = nIter + 1;
        for i = 1:n,
            x(i) = b(i) - A(i, 1:i-1) * x(1:i-1) - A(i, i+1:n) * x0(i+1:n);
            if (abs(A(i, i)) > 1E-10),
                x(i) = x(i)/A(i, i);
                x(i) = omega * x(i) + (1-omega)*x0(i);
            else
                'Error: diagonal is close to zero'
            end
        end
        error = norm(x - x0);
        x0 = x;
    end
Comparison

\[ A = \begin{bmatrix}
3 & -1.4 \\
-1.4 & \ddots & \ddots \\
& \ddots & -1.4 \\
-1.4 & 3 \\
\end{bmatrix}, \quad x = \begin{bmatrix}
1 \\
2 \\
\vdots \\
40 \\
\end{bmatrix}, \quad \text{and } b = Ax \\
\]

Initial guess: \( x^{(0)} = [0, \ldots, 0]^T \); Convergence test: \( \| x^{(k)} - x^{(k-1)} \| < 10^{-6} \)

<table>
<thead>
<tr>
<th>Method</th>
<th>No of Iterations</th>
<th>Infinity norm of abs error</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi</td>
<td>151</td>
<td>8.65e-6</td>
<td></td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>99</td>
<td>9.40e-7</td>
<td></td>
</tr>
<tr>
<td>SOR</td>
<td>42</td>
<td>5.72e-7</td>
<td>( \omega = 1.5 )</td>
</tr>
</tbody>
</table>
SOR Method: choose an appropriate $\omega$

\[
A = \begin{bmatrix}
3 & -1.4 \\
-1.4 & \ddots & \ddots \\
& \ddots & -1.4 \\
& & -1.4 & 3
\end{bmatrix}, \quad x = \begin{bmatrix}
1 \\
2 \\
\vdots \\
40
\end{bmatrix}, \quad \text{and } b = Ax
\]

Initial guess: $x^{(0)} = [0, \ldots, 0]^T$; Convergence test: $\left\| x^{(k)} - x^{(k-1)} \right\| < 10^{-6}$

<table>
<thead>
<tr>
<th>$\omega$ values</th>
<th>No of Iterations</th>
<th>Infinity norm of abs error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>109</td>
<td>9.72e-7</td>
</tr>
<tr>
<td>1.25</td>
<td>67</td>
<td>8.68e-7</td>
</tr>
<tr>
<td>1.5</td>
<td>42</td>
<td>9.72e-7</td>
</tr>
<tr>
<td>1.75</td>
<td>66</td>
<td>8.80e-7</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>11.84</td>
</tr>
</tbody>
</table>
Assessment: How to choose $\omega$

**Theorem 7.25**
(Numerical Analysis, Burden & Faires, Chap. 7, page 449) If $A$ is positive definite matrix and $0 < \omega < 2$, then the SOR method converges for any choice of $x^{(0)}$.

**Theorem 7.26**
(Numerical Analysis, Burden & Faires, Chap. 7, page 449) If $A$ is positive definite tridiagonal matrix then the optimal choice of $\omega$ for the SOR method is

$$
\omega = \frac{2}{1 + \sqrt{1 - \left[ \rho \left( T_j \right) \right]^2}}
$$

where $T_j = D^{-1} (L + U)$. 
Summary

Jacobi, Gauss-Seidel, Successive Over-Relaxation (SOR) methods are commonly used and can be described in following forms.

Let $A = D + L + U$ (as shown in the figure)

**Jacobi:**
$$D x^{(k)} = -(L + U)x^{(k-1)} + b$$

**Gauss-Seidel (SOR, $\omega = 1$):**
$$(D + L)x^{(k)} = -(U)x^{(k-1)} + b$$

**SOR:**
$$(D + \omega L)x^{(k)} = ((1 - \omega)D - \omega U)x^{(k-1)} + \omega b$$
More references


