6. Finite Difference Methods: Dealing with American Option

YORK UNIVERSITÉ UNIVERSITY

redefine THE POSSIBLE.

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

redefine THE POSSIBLE.

Constraints on American Options

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

- Lower bound: $V(S,t) \geqslant \max(K - S, 0)$
- 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

$$P(S,t) \geqslant \max(K - S, 0)$$

- 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

American put option as a free boundary problem

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

i) $0 \le S < S_f$ where early exercise is optimal:

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

ii) $S_f \le S < +\infty$ where retaining the option is optimal:

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

with boundary conditions at $S = S_f(t)$.

Black-Scholes Inequality

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

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

In other words,

the return from the portfolio $\le$ 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.

Crank-Nicolson Methods+ Projected SOR to Price an American Put Option

$$\mathbf{M}_1\mathbf{f}_i = \mathbf{r}_i \equiv \mathbf{M}_2\mathbf{f}_{i+1} + \mathbf{b}$$

where $\mathbf{b} = \alpha_1\left[f_{i,0} + f_{i+1,0}, 0, \ldots, 0\right]^T$. Recall that both $\mathbf{M}_1$ and $\mathbf{M}_2$ 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

$$f_{i,1}^{(k+1)} = max\left\{ g_1, f_{i,1}^{(k)} + \frac{\omega}{1-\beta_1}\left[r_1 - \left(1-\beta_1\right) f_{i,1}^{(k)} + \gamma_1 f_{i,2}^{(k)} \right] \right\}$$

$$f_{i,2}^{(k+1)} = max\left\{ g_2, f_{i,2}^{(k)} + \frac{\omega}{1-\beta_2}\left[r_2 + \alpha_2 f_{i,1}^{(k+1)} - \left(1-\beta_2\right) f_{i,2}^{(k)} + \gamma_2 f_{i,3}^{(k)} \right] \right\}$$

$$\vdots$$

$$f_{i,M-1}^{(k+1)} = max\left\{ g_{M-1}, f_{i,M-1}^{(k)} + \frac{\omega}{1-\beta_{M-1}}\left[r_{M-1} + \alpha_{M-1} f_{i,M-2}^{(k+1)} - \left(1-\beta_{M-1}\right) f_{i,M-1}^{(k)} \right] \right\}$$

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, σ=40%, r = 10%. (ω=1.2, tol = 0.001)

CK Method with S_{max}=\$100, ΔS=1, Δt=1/600: \$4.2800

CK Method with S_{max}=\$100, ΔS=1, Δt=1/1200: \$4.2828

Example (Stability)

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

CK Method with S_{max}=\$100, ΔS=1, Δt=1/600: \$4.2800

CK Method with S_{max}=\$100, ΔS=1, Δt=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) $0 \le S < S_f$ where retaining option is optimal:

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

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

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

with boundary conditions at $S = S_f\left(t\right)$.

6. Finite Difference Methods: Dealing with American Option

6.2 Iterative Methods of Solving a Linear System

YORK
UNIVERSITÉ
UNIVERSITY

redefine THE POSSIBLE.

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 $\mathbf{x} = G(\mathbf{x})$ is the iteration scheme

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

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

Similarly, we can rewritten $\mathbf{Ax}=\mathbf{b}$ as

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

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

$$\mathbf{x}^{(k)} = \hat{\mathbf{A}}\,\mathbf{x}^{(k-1)} - \mathbf{b}$$

Basic ideas of iterative methods

Starting from initial approximation $\mathbf{x}^{(0)}$, we have

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

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

$$\ldots$$

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

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

However, arbitrary systems of equations may often not satisfy this condition

YORK UNIVERSITÉ UNIVERSITY

Basic Ideas of Iterative Methods

A slightly different approach :

Transforming the sytem $\mathbf{Ax} = \mathbf{b}$ to an equivalent system

$$\mathbf{Mx} = -\mathbf{Nx} + \mathbf{b}$$

where $\mathbf{A} = \mathbf{M} + \mathbf{N}$, called a splitting of $\mathbf{A}$.

The corresponding iteration scheme is

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

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

The flexibility in choosing $\mathbf{M}$ may be exploited to improve convergence

Basic Ideas of Iterative Methods

Convergence:

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

$$\mathbf{e}^{(\mathbf{k})} = \mathbf{x} - \mathbf{x}^{(\mathbf{k})} = \mathbf{B}\left(\mathbf{x} - \mathbf{x}^{(\mathbf{k}-1)}\right) = \mathbf{B}\mathbf{e}^{(\mathbf{k}-1)}$$

$$\therefore \lim_{k\to\infty} \mathbf{e}^{(\mathbf{k})} = \lim_{k\to\infty} \mathbf{B}^k \mathbf{e}^{(0)}$$

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

To avoid computing eigenvalues, we may require

$$\rho(\mathbf{B}) \le \|\mathbf{B}\| < 1$$

instead.

Implementation: check for convergence

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

$$\text{absolute difference}: \left\| \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} \right\| < \varepsilon_1$$

$$\text{relative difference}: \frac{\left\| \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} \right\|}{\left\| \mathbf{x}^{(k-1)} \right\|} < \varepsilon_2$$

$$\text{absolute residual}: \left\| \mathbf{r}\left(\mathbf{x}^{(k)}\right) \right\| < \varepsilon_3 \quad \text{where } \mathbf{r}\left(\mathbf{x}^{(k)}\right) = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$$

$$\text{relative residual}: \frac{\left\| \mathbf{r}\left(\mathbf{x}^{(k)}\right) \right\|}{\left\| \mathbf{b} \right\|} < \varepsilon_4$$

Further discussion on basic ideas of iterative methods

Therefore, various iterative methods are developed along the following lines:

- A splitting $\mathbf{A} = \mathbf{M}+\mathbf{N}$ is proposed where linear system of the form $\mathbf{Mz} = \mathbf{d}$ are easy to solve

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

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

Jacobi methods

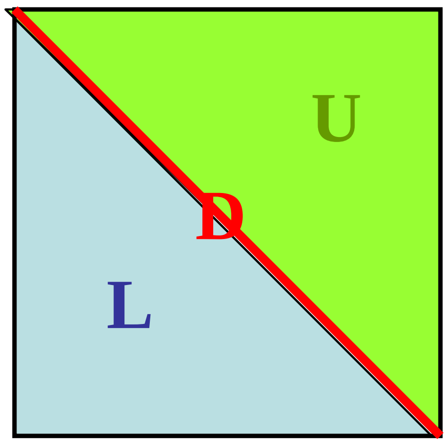
Jacobi, Gauss-Seidel, Successive Over-Relaxation (SOR) methods are commonly used and can be described in following forms.

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

Jacobi:

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

A =



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 i

Implementation

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

Jacobi Method:

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

for $k = 1, \ldots,$ maxiter

for i = 1, …, n,

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

end

check for convergence

end

MATLAB Implementation

```
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

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

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

Method	No of Iterations	Infinity norm of abs error
Jacobi	41	8.24E-7

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)

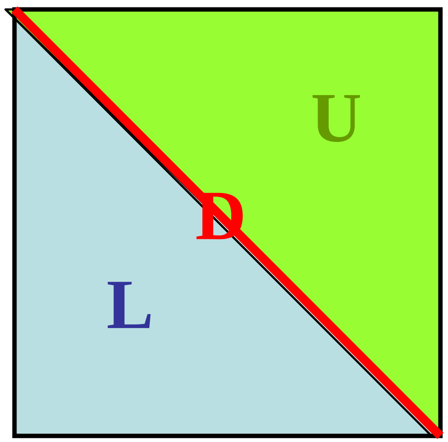
Jacobi:

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

Gauss-Seidel:

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

A =



Example

Consider the 3-by-3 example:

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

Gauss-Seidel Method:

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

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

until there is little change for all the unknowns

Implementation

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

Gauss-Seidel Method:

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

for $k = 1, \ldots,$ maxiter

for $\mathrm{i} = 1, \ldots, \mathrm{n}$,

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

end

check for convergence

end

MATLAB Implementation

```
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

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

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

Method	No of Iterations	Infinity norm of abs error
Jacobi	41	8.24e-7
Gauss-Seidel	25	9.69e-7

Assessment: Existence & Convergence

Definition

Let $\mathbf{A} = \left(a_{ij}\right)$.

Matrix $\mathbf{A}$ is strictly diagonally dominant if and only if for all i

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

Example *:* which ones are strictly diagonally dominant and why?

$$\begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 2 & 1 & \\ -1 & 3 & 2 \\ & 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 2 & 1 & \\ 1 & 3 & 1 \\ & 1 & 2 \end{bmatrix}$$

Assessment: Existence

Theorem

Consider $\mathbf{Ax} = \mathbf{b}$.

If $\mathbf{A}$ is strictly diagonally dominant, then there is a unique solution.

Assessment: Convergence

Theorem

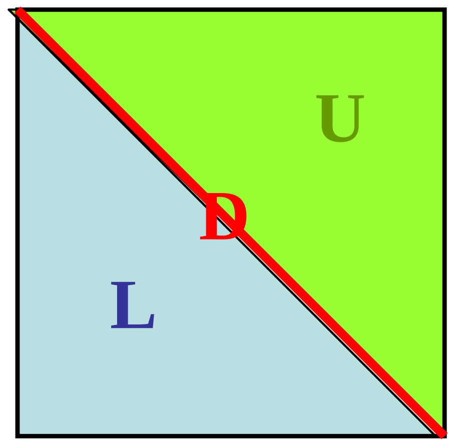
Consider $\mathbf{Ax} = \mathbf{b}$.

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

Prove that $\left\| \mathbf{M}^{-1}\mathbf{N} \right\| < 1$

More on Gauss-Seidel methods

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

A = 

Gauss-Seidel (Forward):

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

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

Gauss-Seidel (Backward):

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

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

Gauss-Seidel (Symmetric):

It consists of a forward sweep followed by a backward sweep

More on Gauss-Seidel methods (Numerical Analysis, Burden & Faires, Chap. 7)

Define the residual vector $\mathbf{r}$ of an approximate $\tilde{\mathbf{x}}$ with respect to the system $\mathbf{Ax} = \mathbf{b}$ as the following:

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$$

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

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

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

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 $\mathbf{r}_{i+1}^{(k)}$. 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 ω, then we can reduce the norm of the residual vector and obtain significantly faster convergence.

Relaxation methods

Methods involving choosing proper values of ω for

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

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 accelebrate the convergence for systems that are convergent by Gauss-Seidel method.

These methods are called called "**Successive Over - Relaxation**" (**SOR**).

SOR method

We reformulate Eq. (5.5.1):

$$x_i^{(k)} = (1-\omega)\, x_i^{(k-1)} + \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k-1)} \right] \quad (5.5.2)$$

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

$$\mathbf{x}^{(k)} = (1-\omega)\,\mathbf{x}^{(k-1)} + \omega \mathbf{x}_{GS}^{(k)}$$

for $0<\omega<2$

Matrix version of SOR method

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

$$a_{ii}x_i^{(k)} + \omega\sum_{j=1}^{i-1} a_{ij}\, x_j^{(k)} = (1-\omega)\, a_{ii}x_i^{(k-1)} - \omega\sum_{j=i+1}^{n} a_{ij}\, x_j^{(k-1)} + \omega b_i \qquad (5.5.3)$$

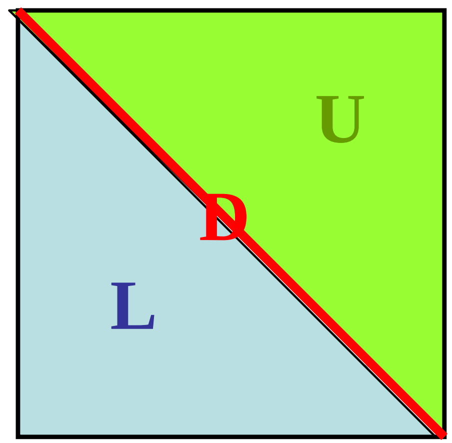
so that in the matrix and vector form, we have

$$\textbf{SOR:} \qquad (\mathbf{D}+\omega\mathbf{L})\mathbf{x}^{(\mathbf{k})} = \left((1-\omega)\mathbf{D}-\omega\mathbf{U}\right)\mathbf{x}^{(\mathbf{k}-1)} + \omega\mathbf{b}$$

When ω=1, SOR methods reduces to the Gauss-Seidel methods

$$\textbf{Gauss-Seidel:} \qquad (\mathbf{D}+\mathbf{L})\mathbf{x}^{(\mathbf{k})} = -\mathbf{U}\mathbf{x}^{(\mathbf{k}-1)} + \mathbf{b}$$

$\mathbf{A} =$



An Example

Consider the 3-by-3 example:

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

SOR Method (Leave as an exercise)

Implementation

Successive Over-Relaxation Method:

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

for k = 1, …, maxiter

for i = 1, …, 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

MATLAB Implementation

```
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

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

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

Method	No of Iterations	Infinity norm of abs error	Notes
Jacobi	151	8.65e-6	
Gauss-Seidel	99	9.40e-7	
SOR	42	5.72e-7	ω=1.5

SOR Method: choose an appropriate ω

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

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

ω values	No of Iterations	Infinity norm of abs error
1	109	9.72e-7
1.25	67	8.68e-7
1.5	42	9.72e-7
1.75	66	8.80e-7
2	300	11.84

Assessment: How to choose ω

Theorem 7.25

(Numerical Analysis, Burden &Faires, Chap. 7, page 449)

If $\mathbf{A}$ is positive definite matrix and $0 < \omega < 2$, then the SOR method converges for any choice of $\mathbf{x}^{(0)}$

Theorem 7.26

(Numerical Analysis, Burden &Faires, Chap. 7, page 449)

If $\mathbf{A}$ is positive definite tridiagonal matrix then the optimal choice of ω for the SOR method is

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

where $\mathbf{T}_j = \mathbf{D}^{-1}\left(\mathbf{L}+\mathbf{U}\right)$.

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:

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

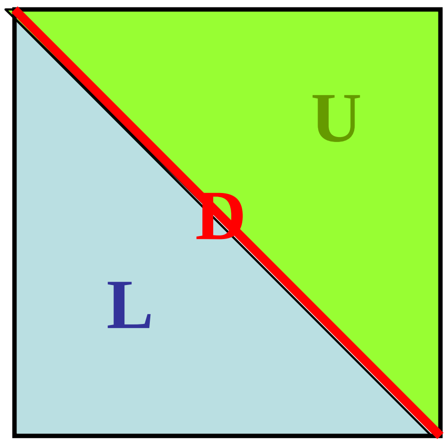
Gauss-Seidel (SOR, ω =1):

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

SOR:

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

A =



More references

1. D. Tavella and C. Randall, “Pricing financial instruments: The finite difference methods”, Wiley, New York, 2000
2. Y.-I. Zhu, X. Wu and I.-L. Chern, “Derivative securities and difference methods”, Springer, New York, 2004
3. R. Seydel, "Tools for computational finance", Springer-Verlag, Berlin, 2002
4. J. Topper, "Financial engineering with finite elements", Wiley, New York, 2005