Calcul Scientifique

Ecole Hassania des Travaux Publics

Salem Nafiri

Méthode itérative

Une **méthode itérative** est un procédé algorithmique utilisé pour résoudre un problème, par exemple la recherche d'une solution d'un système d'équations ou d'un problème d'optimisation. En débutant par le choix d'*un point initial* considéré comme *une première ébauche* de solution, la méthode procède par itérations au cours desquelles elle détermine une succession de *solutions approximatives* raffinées qui se rapprochent graduellement de la solution cherchée. Les points générés sont appelés des **itérés**.

Introduction

- Pourquoi utiliser les méthodes itératives pour résoudre Ax=b ?

- Si les systèmes d'équations linéaires sont très grands, l'effort de calcul avec les méthodes directes est prohibitif et couteux.

- **Les solutions itératives ont la caractéristique qu'elles convergent vers une solution à partir d'une estimation initiale [une ébauche initiale].**

- **Les solutions itératives sont efficaces pour les problèmes de grande taille, par ex. Un pbm d'ordre 105x105 ou plus particulièrement si A est une matrice creuse.**

- Nous devons connaître les types de problèmes pour lesquels ces méthodes itératives sont applicables et pourquoi.

- Nous devons comprendre les conditions pour lesquelles ces méthodes itératives convergent et pourquoi.

- Théoriquement, les méthodes telles que l'élimination de Gauss Jordan, la décomposition LU [Doolittle, Crout], décomposition de Cholesky peuvent nous donner des solutions exactes.

- Si le système est trop grand, nous aurons des problèmes avec
    - les erreurs d'arrondi
    - La capacité de stockage de l'ordinateur.
    - l'effort de calcul des méthodes directes est prohibitif

- Comment le résoudre ?

- Itération: répéter un processus encore et encore jusqu'à ce qu'une approximation de la solution soit atteinte.

- Il est utile pour résoudre certains types de problèmes

- Lorsque le nombre d'inconnues est très important mais que la matrice de coefficients est **creuse**, l'élimination de Gauss devient inefficace et parfois inapplicable.

$$
\begin{bmatrix}
4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4
\end{bmatrix}
\cdot
\begin{bmatrix}
u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \\ u_{16}
\end{bmatrix}
=
\begin{bmatrix}
0.08 \\ 0.16 \\ 0.36 \\ 1.64 \\ 0.16 \\ 0.0 \\ 0.0 \\ 1.0 \\ 0.36 \\ 0 \\ 0 \\ 1.0 \\ 1.64 \\ 1.0 \\ 1.0 \\ 2.0
\end{bmatrix}
$$

Les avantages supplémentaires des méthodes itératives incluent

- (1) leur programmation est simple

- (2) il est facilement applicable lorsque les coefficients sont non linéaires.

Although there are many versions of iterative schemes, we introduce three iterative methods:

- **Jacobi iterative,**
- **Gauss-Seidel, et**
- **Successive-over-relaxation (SOR)**
  - SOR est une méthode utilisée pour accélérer la convergence
  - L'itération de Gauss-Seidel est un cas particulier de la méthode SOR

Les méthodes itératives peuvent s'appliquer au système jusqu'à 100 000 variables.

Des exemples de ces grands systèmes apparaissent dans **la résolution d'équations différentielles partielles**.

Désavantage: Les méthodes itératives ne peuvent pas s'appliquer à tous les systèmes

Principe des méthodes itératives

Ecrire le système Ax=b sous forme équivalente

$\quad$ **x=Ex+f** $\quad$ (comme x=g(x) pour les itérations du point fixe)

Démarrant avec **x⁰**, générer une suite d'approximations {**xᵏ**} itérée par

$\quad$ **xᵏ⁺¹=Exᵏ+f**

La représentation de **E** et de **f** dépendent du type de méthode utilisée

Mais pour chaque méthode, **E** et **f** sont obtenues à partir de **A** et **b**, mais d'une manière différente.

Jacobi Method by examples

- Consider the two-by-two system

$$2x + y = 6$$
$$x + 2y = 6$$

$$\implies \quad x = -\frac{1}{2}y + 3$$
$$y = -\frac{1}{2}x + 3$$

- Start with $\quad x^{(1)} = y^{(1)} = 1/2$

- *Simultaneous updating*
  - New values of the variables are not used until a new iteration step is begun

$$x^{(2)} = -\frac{1}{2}y^{(1)} + 3 = -\frac{1}{4} + 3 = \frac{11}{4}$$

$$y^{(2)} = -\frac{1}{2}x^{(1)} + 3 = -\frac{1}{4} + 3 = \frac{11}{4}$$
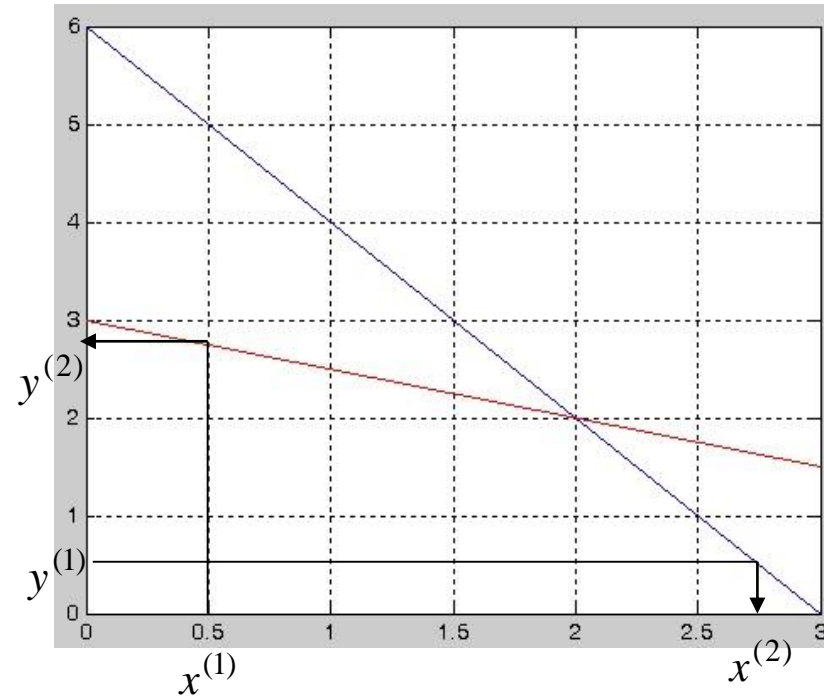
Jacobi Method

- Con't

$$x^{(3)} = -\frac{1}{2} y^{(2)} + 3 = -\frac{11}{8} + 3 = \frac{13}{8}$$

$$y^{(3)} = -\frac{1}{2} x^{(2)} + 3 = -\frac{11}{8} + 3 = \frac{13}{8}$$

```
>> Jacobi_f(A,b,x0,0.001,50)
    1.0000    2.7500    2.7500

    2.0000    1.6250    1.6250

    3.0000    2.1875    2.1875

    4.0000    1.9063    1.9063

    5.0000    2.0469    2.0469

    6.0000    1.9766    1.9766

    7.0000    2.0117    2.0117

    8.0000    1.9941    1.9941

    9.0000    2.0029    2.0029

   10.0000    1.9985    1.9985

   11.0000    2.0007    2.0007

   12.0000    1.9996    1.9996

Jacobi method converged
   13.0000    2.0002    2.0002
```

Jacobi Method

- Consider the three-by-three system

$$2x_1 - x_2 + x_3 = -1$$
$$x_1 + 2x_2 - x_3 = 6$$
$$x_1 - x_2 + 2x_3 = -3$$

$$x_1 = +0.5x_2 - 0.5x_3 - 0.5$$
$$x_2 = -0.5x_1 + 0.5x_3 + 3.0$$
$$x_3 = -0.5x_1 + 0.5x_2 - 1.5$$

$$
\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{bmatrix}
=
\begin{bmatrix} 0.0 & 0.5 & -0.5 \\ -0.5 & 0.0 & 0.5 \\ -0.5 & 0.5 & 0.0 \end{bmatrix}
\begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ x_3^{(k-1)} \end{bmatrix}
+
\begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix}
$$

- Start with $x^{(0)} = (0,0,0)$

```
>> Jacobi_f(A,b,x0,0.001,50)
 1.0000  -0.5000   3.0000  -1.5000

 2.0000   1.7500   2.5000   0.2500

 3.0000   0.6250   2.2500  -1.1250

 4.0000   1.1875   2.1250  -0.6875

 5.0000   0.9063   2.0625  -1.0313

 6.0000   1.0469   2.0313  -0.9219

 7.0000   0.9766   2.0156  -1.0078

 8.0000   1.0117   2.0078  -0.9805

 9.0000   0.9941   2.0039  -1.0020

10.0000   1.0029   2.0020  -0.9951

11.0000   0.9985   2.0010  -1.0005

12.0000   1.0007   2.0005  -0.9988

13.0000   0.9996   2.0002  -1.0001

Jacobi method converged
14.0000   1.0002   2.0001  -0.9997
```

Jacobi iteration

$$a_{11}\boxed{x_1} + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}\boxed{x_2} + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}\boxed{x_n} = b_n$$

$$x^0 = \begin{bmatrix} x_1^0 \\ x_2^0 \\ \vdots \\ x_n^0 \end{bmatrix}$$

$$x_1^1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2^0 - \cdots - a_{1n}x_n^0)$$

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

$$x_2^1 = \frac{1}{a_{22}}(b_2 - a_{21}x_1^0 - a_{23}x_3^0 - \cdots - a_{2n}x_n^0)$$

$$x_n^1 = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^0 - a_{n2}x_2^0 - \cdots - a_{nn-1}x_{n-1}^0)$$

Jacobi Method

$$\begin{cases} x_1{}^{new} = (b_1 - a_{12}x_2{}^{old} - a_{13}x_3{}^{old} - a_{14}x_4{}^{old})/a_{11} \\ x_2{}^{new} = (b_2 - a_{21}x_1{}^{old} - a_{23}x_3{}^{old} - a_{24}x_4{}^{old})/a_{22} \\ x_3{}^{new} = (b_3 - a_{31}x_1{}^{old} - a_{32}x_2{}^{old} - a_{34}x_4{}^{old})/a_{33} \\ x_4{}^{new} = (b_4 - a_{41}x_1{}^{old} - a_{42}x_2{}^{old} - a_{43}x_3{}^{old})/a_{44} \end{cases}$$

$x^{k+1}=Ex^k+f$ iteration for Jacobi method

A can be written as A=L+D+U (*not decomposition*)

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

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

$$x_i^{k+1} = \frac{1}{a_{ii}} \left[b_i - \underbrace{\sum_{j=1}^{i-1} a_{ij}x_j^k}_{Lx^k} - \underbrace{\sum_{j=i+1}^{n} a_{ij}x_j^k}_{Ux^k} \right]$$

Dx^{k+1}

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

$E=-D^{-1}(L+U)$

$f=D^{-1}b$

Gauss-Seidel (GS) iteration

Use the latest update

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

$$a_{21}\!\!\bigcirc\!\!x_1 + a_{22}\boxed{x_2} + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{n1}\!\!\bigcirc\!\!x_1 + a_{n2}\!\!\bigcirc\!\!x_2 + \cdots + a_{nn}\boxed{x_n} = b_n$$

$$x^0 = \begin{bmatrix} x_1^0 \\ x_2^0 \\ \vdots \\ x_n^0 \end{bmatrix}$$

$$x_1^1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2^0 - \cdots - a_{1n}x_n^0)$$

$$x_2^1 = \frac{1}{a_{22}}(b_2 - a_{21}x_1^1 - a_{23}x_3^0 - \cdots - a_{2n}x_n^0)$$

$$x_n^1 = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^1 - a_{n2}x_2^1 - \cdots - a_{nn-1}x_{n-1}^1)$$

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

Gauss-Seidel Method

Differ from Jacobi method by **sequential updating:**
use new x_i immediately as they become available

$$
\begin{cases}
x_1^{new} = (b_1 - a_{12}x_2^{old} - a_{13}x_3^{old} - a_{14}x_4^{old})/a_{11} \\
x_2^{new} = (b_2 - a_{21}x_1^{new} - a_{23}x_3^{old} - a_{24}x_4^{old})/a_{22} \\
x_3^{new} = (b_3 - a_{31}x_1^{new} - a_{32}x_2^{new} - a_{34}x_4^{old})/a_{33} \\
x_4^{new} = (b_4 - a_{41}x_1^{new} - a_{42}x_2^{new} - a_{43}x_3^{new})/a_{44}
\end{cases}
$$

(a) Gauss-Seidel Method   (b) Jacobi Method

First iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

Second iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

(a)   (b)
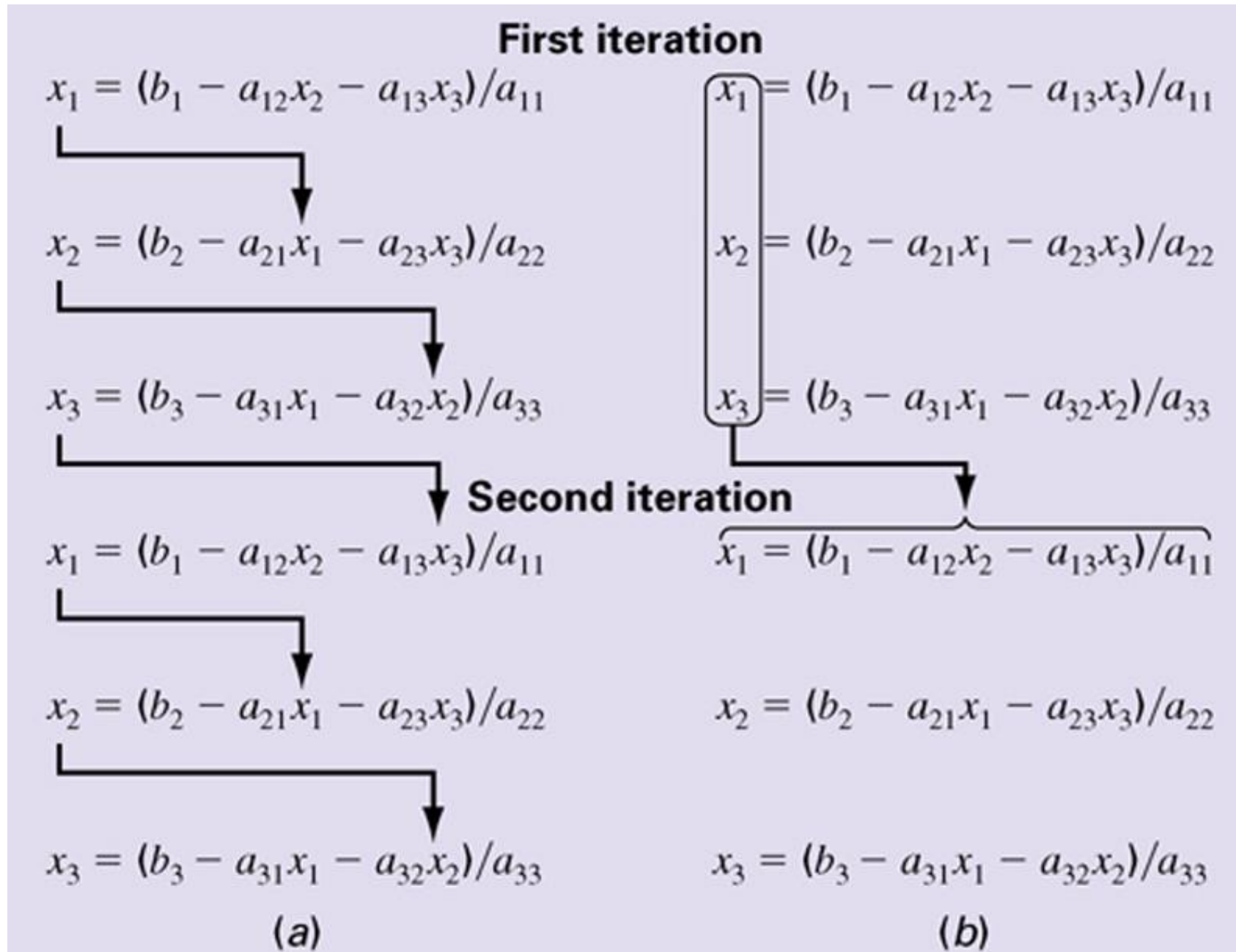
$x^{(k+1)}=Ex^{(k)}+f$ iteration for Gauss-Seidel

$Ax=b \Rightarrow (L+D+U)x=b$

$$x_i^{k+1} = \frac{1}{a_{ii}}\left[b_i - \underbrace{\sum_{j=1}^{i-1} a_{ij}x_j^{k+1}}_{Lx^{k+1}} - \underbrace{\sum_{j=i+1}^{n} a_{ij}x_j^{k}}_{Ux^{k}} \right]$$

Dx^{k+1}

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

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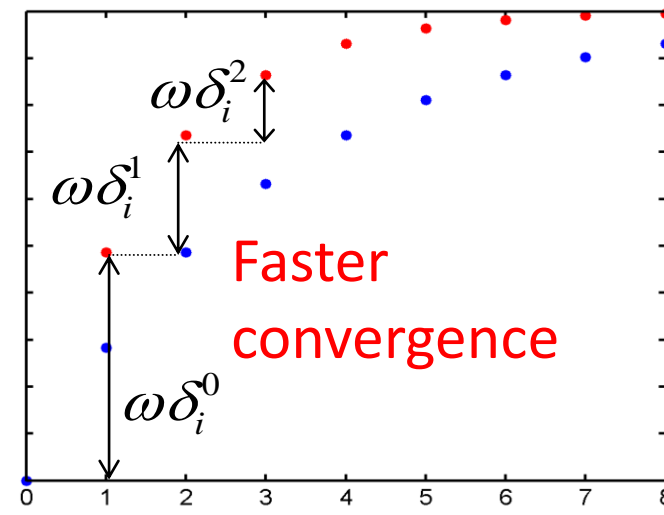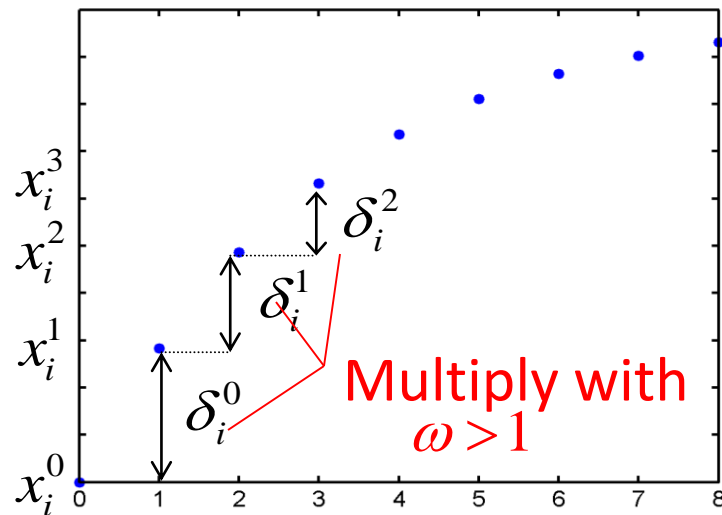
$E=-(D+L)^{-1}U$

$f=-(D+L)^{-1}b$

Comparison

- Gauss-Seidel iteration converges more rapidly than the Jacobi iteration since it uses the latest updates

- But there are some cases that *Jacobi* iteration *does converge* but *Gauss-Seidel* does *not*

- To accelerate the Gauss-Seidel method even further, successive over relaxation [SOR] method can be used

Successive Over Relaxation Method

- GS iteration can be also written as follows

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

$$x_i^{k+1} = x_i^k + \delta_i^k \quad \longrightarrow \quad \text{Correction term}$$

Multiply with $\omega > 1$

Faster convergence

SOR

$$x_i^{k+1} = x_i^k + \omega \delta_i^k$$

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

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

$1<\omega<2$ over relaxation (faster convergence)
$0<\omega<1$ under relaxation (slower convergence)
There is an optimum value for ω
Find it by trial and error (usually around 1.6)

$x^{(k+1)}=Ex^{(k)}+f$ iteration for SOR

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

$Dx^{k+1}=(1-\omega)Dx^k+\omega b-\omega Lx^{k+1}-\omega Ux^k$

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

$E=(D+\omega L)^{-1}[(1-\omega)D-\omega U]$

$f=\omega(D+\omega L)^{-1}b$

The Conjugate Gradient Method

$$d_0 = r_0 = b - Ax_0$$

$$\alpha_i = \frac{r_i^T r_i}{d_i^T A d_i}$$

$$x_{i+1} = x_i + \alpha_i A d_i$$

$$\beta_{i+1} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$$

$$d_{i+1} = r_{i+1} + \beta_{i+1} d_i$$

- Converges if A is a symmetric positive definite matrix
- Convergence is faster

Convergence of Iterative Methods

Define the solution vector as $\hat{x}$

Define an error vector as e^k

$$x^k = e^k + \hat{x}$$
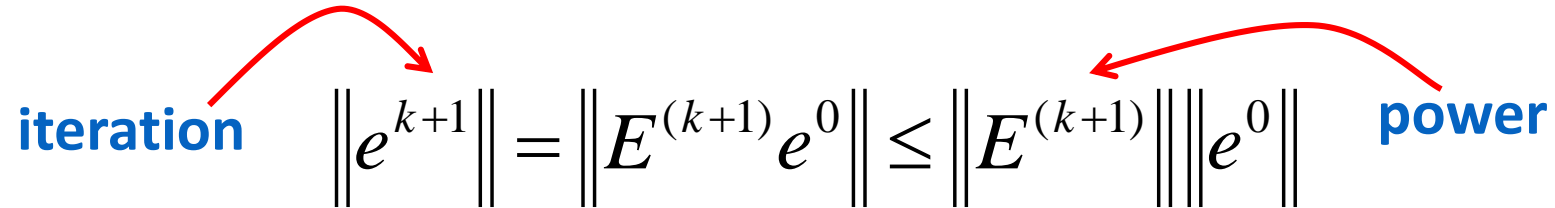
Substitute this into $x^{k+1} = Ex^k + f$

$$e^{k+1} + \hat{x} = E(e^k + \hat{x}) + f = E\hat{x} + f + Ee^k$$

$$e^{k+1} = Ee^k = EEe^{k-1} = EEEe^{k-2} = E^{(k+1)}e^0$$

Convergence of Iterative Methods

iteration   **power**

$$\left\| e^{k+1} \right\| = \left\| E^{(k+1)} e^0 \right\| \leq \left\| E^{(k+1)} \right\| \left\| e^0 \right\|$$

The iterative method will converge for any initial
iteration vector if the following condition is satisfied

Convergence condition

$$\operatorname*{Lim}_{k \to \infty} \left\| e^{k+1} \right\| \to 0 \quad \text{if} \quad \operatorname*{Lim}_{k \to \infty} \left\| E^{(k+1)} \right\| \to 0$$

Norm of a vector

A vector norm should satisfy these conditions

$$\|x\| \geq 0 \quad \text{for every nonzero vector } x$$

$$\|x\| = 0 \quad \text{iff } x \text{ is a zero vector}$$

$$\|\alpha x\| = |\alpha| \|x\| \quad \text{for scalar } \alpha$$

$$\|x + y\| \leq \|x\| + \|y\|$$

Vector norms can be defined in different forms as long as the norm definition satisfies these conditions

Commonly used vector norms

Sum norm or ℓ_1 norm

$$\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

Euclidean norm or ℓ_2 norm

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Maximum norm or ℓ_∞ norm

$$\|x\|_\infty = \max_i |x_i|$$

Norm of a matrix

A matrix norm should satisfy these conditions

$$\|A\| \geq 0$$

$$\|A\| = 0 \quad \text{iff} \quad A \text{ is a zero matrix}$$

$$\|\alpha A\| = |\alpha|\|A\| \quad \text{for scalar } \alpha$$

$$\|A + B\| \leq \|A\| + \|B\|$$

Important identitiy

$$\|Ax\| \leq \|A\|\|x\| \quad x \text{ is a vector}$$

Commonly used matrix norms

Maximum column-sum norm or ℓ_1 norm

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|$$

Spectral norm or ℓ_2 norm

$$\|A\|_2 = \sqrt{\text{maximum eigenvalue of } A^T A}$$

Maximum row-sum norm or ℓ_∞ norm

$$\|A\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$$

Example

- Compute the ℓ_1 and ℓ_∞ norms of the matrix

$$\begin{bmatrix} 3 & 9 & 5 \\ 7 & 2 & 4 \\ 6 & 8 & 1 \end{bmatrix}$$

17    $= \|A\|_\infty$

13

15

16   19   10

$= \|A\|_1$

Convergence condition

$$\lim_{k \to \infty} \left\| e^{k+1} \right\| \to 0 \quad \text{if} \quad \lim_{k \to \infty} \left\| E^{(k+1)} \right\| \to 0$$

Express E in terms of modal matrix P and Λ

Λ:Diagonal matrix with eigenvalues of E on the diagonal

$$E = P\Lambda P^{-1}$$

$$E^{(k+1)} = P\Lambda P^{-1} \ P\Lambda P^{-1} \ldots P\Lambda P^{-1}$$

$$E^{(k+1)} = P\Lambda^{(k+1)} P^{-1}$$

$$\Lambda^{k+1} = \begin{bmatrix} \lambda_1^{k+1} & & & \\ & \lambda_2^{k+1} & & \\ & & \ddots & \\ & & & \lambda_n^{k+1} \end{bmatrix}$$

$$\lim_{k \to \infty} \left\| E^{(k+1)} \right\| \to 0 \Rightarrow \lim_{k \to \infty} \left\| P\Lambda^{(k+1)} P^{-1} \right\| \to 0 \Rightarrow \lim_{k \to \infty} \left\| \Lambda^{(k+1)} \right\| \to 0$$

$$\Rightarrow \lim_{k \to \infty} \left| \lambda_i^{k+1} \right| \to 0 \Rightarrow \left| \lambda_i \right| < 1 \ \text{for} \ i = 1,2,...,n$$

33

Sufficient condition for convergence

If the magnitude of all eigenvalues of iteration matrix E is less than 1 than the iteration is convergent

It is easier to compute the norm of a matrix than to compute its eigenvalues

$$Ex = \lambda x$$

$$\left.\begin{array}{l} \|Ex\| = |\lambda|\|x\| \\ \|Ex\| \leq \|E\|\|x\| \end{array}\right\} \Rightarrow |\lambda|\|x\| \leq \|E\|\|x\| \Rightarrow |\lambda| \leq \|E\|$$

$\|E\| < 1$ is a sufficient condition for convergence

Convergence of Jacobi iteration

E=-D^{-1}(L+U)

$$E = \begin{bmatrix} 0 & -\dfrac{a_{12}}{a_{11}} & \cdots & \cdots & -\dfrac{a_{1n}}{a_{11}} \\[2ex] -\dfrac{a_{21}}{a_{22}} & 0 & -\dfrac{a_{23}}{a_{22}} & \cdots & -\dfrac{a_{2n}}{a_{22}} \\[2ex] \vdots & \ddots & \ddots & \ddots & \vdots \\[2ex] & & \ddots & \ddots & -\dfrac{a_{n-1n}}{a_{n-1n-1}} \\[2ex] -\dfrac{a_{n1}}{a_{nn}} & \cdots & \cdots & -\dfrac{a_{nn-1}}{a_{nn}} & 0 \end{bmatrix}$$

Convergence of Jacobi iteration

Evaluate the infinity(maximum row sum) norm of E

$$\|E\|_\infty < 1 \Rightarrow \sum_{\substack{j=1 \\ i \neq j}}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1 \quad \text{for i}=1,2,\ldots,\text{n}$$

$$\Rightarrow |a_{ii}| > \sum_{\substack{j=1 \\ i \neq j}}^{n} |a_{ij}| \qquad \text{Diagonally dominant matrix}$$

If A is a diagonally dominant matrix, then Jacobi iteration converges for any initial vector

Stopping Criteria

- Ax=b

- At any iteration k, the residual term is

  $r^k = b - Ax^k$

- Check the norm of the residual term

  $||b - Ax^k||$

- If it is less than a threshold value stop

Example 1 (Jacobi Iteration)

$$\begin{bmatrix} 4 & -1 & 1 \\ 4 & -8 & 1 \\ -2 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ -21 \\ 15 \end{bmatrix} \qquad x^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \left\| b - Ax^0 \right\|_2 = 26.7395$$

Diagonally dominant matrix

$$x_1^1 = \frac{7 + x_2^0 - x_3^0}{4} = \frac{7}{4} = 1.75$$

$$x_2^1 = \frac{21 + 4x_1^0 + x_3^0}{8} = \frac{21}{8} = 2.625 \qquad \left\| b - Ax^1 \right\|_2 = 10.0452$$

$$x_3^1 = \frac{15 + 2x_1^0 - x_2^0}{5} = \frac{15}{5} = 3.0$$

Example 1 continued…

$$x_1^2 = \frac{7 + x_2^1 - x_3^1}{4} = \frac{7 + 2.625 - 3}{4} = 1.65625$$

$$x_2^2 = \frac{21 + 4x_1^1 + x_3^1}{8} = \frac{21 + 4 \times 1.75 + 3}{8} = 3.875$$

$$x_3^2 = \frac{15 + 2x_1^1 - x_2^1}{5} = \frac{15 + 2 \times 1.75 - 2.625}{5} = 4.225$$

$$\left\| b - Ax^2 \right\|_2 = 6.7413$$

$$x_1^3 = \frac{7 + 3.875 - 4.225}{4} = 1.6625$$

$$x_2^3 = \frac{21 + 4 \times 1.65625 + 4.225}{8} = 3.98125$$

$$x_3^3 = \frac{15 + 2 \times 1.65625 - 3.875}{5} = 2.8875$$

$$\left\| b - Ax^2 \right\|_2 = 1.9534$$

Matrix is diagonally dominant, Jacobi iterations are *converging*

Example 2

$$\begin{bmatrix} -2 & 1 & 5 \\ 4 & -8 & 1 \\ 4 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 15 \\ -21 \\ 7 \end{bmatrix} \qquad x^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \left\| b - Ax^0 \right\|_2 = 26.7395$$

The matrix is not diagonally dominant

$$x_1^1 = \frac{-15 + x_2^0 + 5x_3^0}{2} = \frac{-15}{2} = -7.5$$

$$x_2^1 = \frac{21 + 4x_1^0 + x_3^0}{8} = \frac{21}{8} = 2.625 \qquad \left\| b - Ax^1 \right\|_2 = 54.8546$$

$$x_3^1 = 7 - 4x_1^0 + x_2^0 = 7.0$$

Example 2 continued...

$$x_1^1 = \frac{-15 + 2.625 + 5 \times 7}{2} = 11.3125$$

$$x_2^1 = \frac{21 - 4 \times 7.5 + 7}{8} = -0.25$$

$$\left\| b - Ax^2 \right\|_2 = 208.3761$$

$$x_3^1 = 7 + 4 \times 7.5 + 2.625 = 39.625$$

The residual term is increasing at each iteration, so the iterations are *diverging*.

Note that the matrix is <u>not diagonally dominant</u>

Convergence of Gauss-Seidel iteration

- GS iteration converges for any initial vector if A is a *diagonally dominant matrix*

- GS iteration converges for any initial vector if A is a *symmetric* and *positive definite matrix*

- Matrix A is positive definite if

  $x^TAx>0$ for every nonzero x vector

Positive Definite Matrices

- A matrix is positive definite if all its eigenvalues are positive

- A symmetric diagonally dominant matrix with positive diagonal entries is positive definite

- If a matrix is positive definite
  - All the diagonal entries are positive
  - The largest (in magnitude) element of the whole matrix must lie on the diagonal

Positive Definitiness Check

$$\begin{bmatrix} 20 & 12 & 25 \\ 12 & 15 & 2 \\ 25 & 2 & 5 \end{bmatrix}$$

Not positive definite
Largest element is not on the diagonal

$$\begin{bmatrix} 20 & 12 & 5 \\ 12 & -15 & 2 \\ 5 & 2 & 25 \end{bmatrix}$$

Not positive definite
All diagonal entries are not positive

$$\begin{bmatrix} 20 & 12 & 5 \\ 12 & 15 & 2 \\ 5 & 2 & 25 \end{bmatrix}$$

Positive definite
Symmetric, diagonally dominant, all diagonal entries are positive

Positive Definitiness Check

$$\begin{bmatrix} 20 & 12 & 5 \\ 12 & 15 & 2 \\ 8 & 2 & 25 \end{bmatrix}$$

A decision can not be made just by investigating the matrix.

The matrix is diagonally dominant and all diagonal entries are positive but it is *not symmetric*.

To decide, check if all the eigenvalues are positive

Example (Gauss-Seidel Iteration)

$$\begin{bmatrix} 4 & -1 & 1 \\ 4 & -8 & 1 \\ -2 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ -21 \\ 15 \end{bmatrix} \qquad x^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \left\| b - Ax^0 \right\|_2 = 26.7395$$

Diagonally dominant matrix

$$x_1^1 = \frac{7 + x_2^0 - x_3^0}{4} = \frac{7}{4} = 1.75 \qquad\qquad \left\| b - Ax^1 \right\|_2 = 3.0414$$

$$x_2^1 = \frac{21 + 4x_1^1 + x_3^0}{8} = \frac{21 + 4 \times 1.75}{8} = 3.5$$

$$\left\| b - Ax^1 \right\|_2 = 10.0452$$

$$x_3^1 = \frac{15 + 2x_1^1 - x_2^1}{5} = \frac{15 + 2 \times 1.75 - 3.5}{5} = 3.0$$

Jacobi iteration

Example 1 continued...

$$x_1^2 = \frac{7 + x_2^1 - x_3^1}{4} = \frac{7 + 3.5 - 3}{4} = 1.875$$

$$x_2^2 = \frac{21 + 4x_1^2 + x_3^1}{8} = \frac{21 + 4 \times 1.875 + 3}{8} = 3.9375$$

$$x_3^2 = \frac{15 + 2x_1^2 - x_2^2}{5} = \frac{15 + 2 \times 1.875 - 3.9375}{5} = 2.9625$$

$$\left\| b - Ax^2 \right\|_2 = 0.4765$$

$$\left\| b - Ax^2 \right\|_2 = 6.7413$$

Jacobi iteration

When both Jacobi and Gauss-Seidel iterations converge, Gauss-Seidel converges faster

Gauss-Seidel Method

The Gauss-Seidel Method can still be used

The coefficient matrix is not diagonally dominant

$$[A] = \begin{bmatrix} 3 & 7 & 13 \\ 1 & 5 & 3 \\ 12 & 3 & -5 \end{bmatrix}$$

But this is the same set of equations used in example #2, which did converge.

$$[A] = \begin{bmatrix} 12 & 3 & -5 \\ 1 & 5 & 3 \\ 3 & 7 & 13 \end{bmatrix}$$

If a system of linear equations is not diagonally dominant, check to see if rearranging the equations can form a diagonally dominant matrix.

Gauss-Seidel Method

Not every system of equations can be rearranged to have a diagonally dominant coefficient matrix.

Observe the set of equations

$$x_1 + x_2 + x_3 = 3$$

$$2x_1 + 3x_2 + 4x_3 = 9$$

$$x_1 + 7x_2 + x_3 = 9$$

Which equation(s) prevents this set of equation from having a diagonally dominant coefficient matrix?

Convergence of SOR method

- If $0<\omega<2$, SOR method converges for any initial vector if A matrix is symmetric and positive definite

- If $\omega>2$, SOR method diverges

- If $0<\omega<1$, SOR method converges but the convergence rate is slower (deceleration) than the Gauss-Seidel method.

Operation count

- The operation count for Gaussian Elimination or LU Decomposition was $O(n^3)$, order of n^3.

- For iterative methods, the number of scalar multiplications is $O(n^2)$ at each iteration.

- If the total number of iterations required for convergence is much less than n, then iterative methods are more efficient than direct methods.

- Also iterative methods are well suited for sparse matrices

TP avec Scilab

Example Use the Jacobi iteration to approximate the solution $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ of

$$\begin{bmatrix} 8 & 2 & 4 \\ 3 & 5 & 1 \\ 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -16 \\ 4 \\ -12 \end{bmatrix}.$$ Use the initial guess $X^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$

Solution

In this case $D = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{bmatrix}$ and $L + U = \begin{bmatrix} 0 & 2 & 4 \\ 3 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$.

First iteration.
The first iteration is $DX^{(1)} = -(L+U)X^{(0)} + B$, or in full

$$\begin{bmatrix} 8 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0 & -2 & -4 \\ -3 & 0 & -1 \\ -2 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix} + \begin{bmatrix} -16 \\ 4 \\ -12 \end{bmatrix} = \begin{bmatrix} -16 \\ 4 \\ -12 \end{bmatrix},$$

since the initial guess was $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$.

Taking this information row by row we see that

$$8x_1^{(1)} = -16 \quad \therefore \boxed{x_1^{(1)} = -2}$$

$$5x_2^{(1)} = 4 \quad \therefore \boxed{x_2^{(1)} = 0.8}$$

$$4x_3^{(1)} = -12 \quad \therefore \boxed{x_3^{(1)} = -3}$$

Thus the first Jacobi iteration gives us $X^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} -2 \\ 0.8 \\ -3 \end{bmatrix}$ as an approximation to X.

Second iteration.

The second iteration is $DX^{(2)} = -(L + U)X^{(1)} + B$, or in full

$$
\begin{bmatrix} 8 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} = \begin{bmatrix} 0 & -2 & -4 \\ -3 & 0 & -1 \\ -2 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} + \begin{bmatrix} -16 \\ 4 \\ -12 \end{bmatrix}.
$$

Taking this information row by row we see that

$$8x_1^{(2)} = -2x_2^{(1)} - 4x_3^{(1)} - 16 = -2(0.8) - 4(-3) - 16 = -5.6 \quad \therefore \boxed{x_1^{(2)} = -0.7}$$

$$5x_2^{(2)} = -3x_1^{(1)} - x_3^{(1)} + 4 = -3(-2) - (-3) + 4 = 13 \quad \therefore \boxed{x_2^{(2)} = 2.6}$$

$$4x_3^{(2)} = -2x_1^{(1)} - x_2^{(1)} - 12 = -2(-2) - 0.8 - 12 = -8.8 \quad \therefore \boxed{x_3^{(2)} = -2.2}$$

Therefore the second iterate approximating X is $X^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} = \begin{bmatrix} -0.7 \\ 2.6 \\ -2.2 \end{bmatrix}.$

Third iteration.

The third iteration is $DX^{(3)} = -(L+U)X^{(2)} + B$, or in full

$$\begin{bmatrix} 8 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1^{(3)} \\ x_2^{(3)} \\ x_3^{(3)} \end{bmatrix} = \begin{bmatrix} 0 & -2 & -4 \\ -3 & 0 & -1 \\ -2 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} + \begin{bmatrix} -16 \\ 4 \\ -12 \end{bmatrix}$$

Taking this information row by row we see that

$$8x_1^{(3)} = -2x_2^{(2)} - 4x_3^{(2)} - 16 = -2(2.6) - 4(-2.2) - 16 = -12.4 \quad \therefore \quad \boxed{x_1^{(3)} = -1.55}$$

$$5x_2^{(3)} = -3x_1^{(2)} - x_3^{(2)} + 4 = -3(-0.7) - (2.2) + 4 = 8.3 \quad \therefore \quad \boxed{x_2^{(3)} = 1.66}$$

$$4x_3^{(3)} = -2x_1^{(2)} - x_2^{(2)} - 12 = -2(-0.7) - 2.6 - 12 = -13.2 \quad \therefore \quad \boxed{x_3^{(3)} = -3.3}$$

Therefore the third iterate approximating X is $X^{(3)} = \begin{bmatrix} x_1^{(3)} \\ x_2^{(3)} \\ x_3^{(3)} \end{bmatrix} = \begin{bmatrix} -1.55 \\ 1.66 \\ -3.3 \end{bmatrix}$.

More iterations.

Three iterations is plenty when doing these calculations by hand! But the repetitive nature of the process is ideally suited to its implementation on a computer. It turns out that the next few iterates are

$$X^{(4)} = \begin{bmatrix} -0.765 \\ 2.39 \\ -2.64 \end{bmatrix}, \quad X^{(5)} = \begin{bmatrix} -1.2775 \\ 1.787 \\ -3.215 \end{bmatrix}, \quad X^{(6)} = \begin{bmatrix} -0.83925 \\ 2.2095 \\ -2.808 \end{bmatrix},$$

to 4 d.p.. Carrying on even further $X^{(20)} = \begin{bmatrix} x_1^{(20)} \\ x_2^{(20)} \\ x_3^{(20)} \end{bmatrix} = \begin{bmatrix} -0.9959 \\ 2.0043 \\ -2.9959 \end{bmatrix}$, to 4 decimal places.

After about 40 iterations successive iterates are equal to 4 decimal places. Continuing the iteration even further causes the iterates to agree to more and more decimal places. The method converges to the exact answer $X = \begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix}$.

Jacobi Method

- Consider the three-by-three system

$$2x_1 - x_2 + x_3 = -1$$

$$x_1 + 2x_2 - x_3 = 6$$

$$x_1 - x_2 + 2x_3 = -3$$

$$x_1 = \qquad\quad +0.5x_2 - 0.5x_3 - 0.5$$

$$x_2 = -0.5x_1 \qquad\quad +0.5x_3 + 3.0$$

$$x_3 = -0.5x_1 + 0.5x_2 \qquad\quad -1.5$$

$$\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.5 & -0.5 \\ -0.5 & 0.0 & 0.5 \\ -0.5 & 0.5 & 0.0 \end{bmatrix} \begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ x_3^{(k-1)} \end{bmatrix} + \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix}$$

- Start with $x^{(0)} = (0,0,0)$

```
>> Jacobi_f(A,b,x0,0.001,50)
   1.0000  -0.5000   3.0000  -1.5000

   2.0000   1.7500   2.5000   0.2500

   3.0000   0.6250   2.2500  -1.1250

   4.0000   1.1875   2.1250  -0.6875

   5.0000   0.9063   2.0625  -1.0313

   6.0000   1.0469   2.0313  -0.9219

   7.0000   0.9766   2.0156  -1.0078

   8.0000   1.0117   2.0078  -0.9805

   9.0000   0.9941   2.0039  -1.0020

  10.0000   1.0029   2.0020  -0.9951

  11.0000   0.9985   2.0010  -1.0005

  12.0000   1.0007   2.0005  -0.9988

  13.0000   0.9996   2.0002  -1.0001

Jacobi method converged
  14.0000   1.0002   2.0001  -0.9997
```

Jacobi Code

- Matlab function for Jacobi method

```
function x = Jacobi_f(A, b, x0, tol, max)
%  Inputs :
%     A      coefficient matrix (n-by-n)
%     b      right-hand side (n-by-1)
%     x0     initial solution (n-by-1)
%     tol    stop if norm of change in x < tol
%     max    maximum number of iterations
%  Outputs :
%     x      solution vector (n-by-1)
[n m] = size(A);   xold = x0;   C = -A;
for i=1:n
   C(i,i) = 0;
end
for i=1:n
   C(i,:)=C(i,:)/A(i,i);
end
for i=1:n
   d(i,1) = b(i)/A(i,i);
end
```

```
i = 1;
while (i <= max)
   xnew = C * xold + d;
   if norm(xnew-xold) <= tol
      x = xnew;
      disp('Jacobi method converged');
      disp ([i   xnew']);
      return;
   else
      xold=xnew;
   end
   disp ([i   xnew']);
   i=i+1;
end
disp('Jacobi method did not converge');
disp('results after maximum number of iterations');
x = xnew;
```

Jacobi Iterative Technique

Consider the following set of equations.

$$10x_1 \quad - x_2 \quad + 2x_3 \qquad\qquad = 6$$

$$- x_1 \quad + 11x_2 \quad - x_3 \quad + 3x_4 = 25$$

$$2x_1 \quad - x_2 \quad + 10x_3 \quad - x_4 = -11$$

$$3x_2 \quad - x_3 \quad + 8x_4 = 15$$

Convert the set Ax = b in the form of x = Tx + c.

$$x_1 = \quad\quad\quad \frac{1}{10}x_2 \quad -\frac{1}{5}x_3 \quad\quad\quad +\frac{3}{5}$$

$$x_2 = \quad \frac{1}{11}x_1 \quad\quad\quad +\frac{1}{11}x_3 \quad -\frac{3}{11}x_4 \quad +\frac{25}{11}$$

$$x_3 = \quad -\frac{1}{5}x_1 \quad +\frac{1}{10}x_2 \quad\quad\quad +\frac{1}{10}x_4 \quad -\frac{11}{10}$$

$$x_4 = \quad\quad\quad -\frac{3}{8}x_2 \quad +\frac{1}{8}x_3 \quad\quad\quad +\frac{15}{8}$$

Start with an initial approximation of:

$$x_1^{(0)} = 0, \; x_2^{(0)} = 0, \; x_3^{(0)} = 0 \text{ and } x_4^{(0)} = 0.$$

$$x_1^{(1)} = \frac{1}{10}x_2^{(0)} - \frac{1}{5}x_3^{(0)} + \frac{3}{5}$$

$$x_2^{(1)} = \frac{1}{11}x_1^{(0)} + \frac{1}{11}x_3^{(0)} - \frac{3}{11}x_4^{(0)} + \frac{25}{11}$$

$$x_3^{(1)} = -\frac{1}{5}x_1^{(0)} + \frac{1}{10}x_2^{(0)} + \frac{1}{10}x_4^{(0)} - \frac{11}{10}$$

$$x_4^{(1)} = -\frac{3}{8}x_2^{(0)} + \frac{1}{8}x_3^{(0)} + \frac{15}{8}$$

$$x_1^{(1)} = \frac{1}{10}(0) \quad -\frac{1}{5}(0) \qquad\qquad +\frac{3}{5}$$

$$x_2^{(1)} = \frac{1}{11}(0) \qquad\qquad +\frac{1}{11}(0) \quad -\frac{3}{11}(0) \quad +\frac{25}{11}$$

$$x_3^{(1)} = -\frac{1}{5}(0) \quad +\frac{1}{10}(0) \qquad\qquad +\frac{1}{10}(0) \quad -\frac{11}{10}$$

$$x_4^{(1)} = -\frac{3}{8}(0) \quad +\frac{1}{8}(0) \qquad\qquad +\frac{15}{8}$$

$$x_1^{(1)} = 0.6000, \, x_2^{(1)} = 2.2727,$$

$$x_3^{(1)} = -1.1000 \textbf{ and } x_4^{(1)} = 1.8750$$

$$x_1^{(2)} = \frac{1}{10} x_2^{(1)} - \frac{1}{5} x_3^{(1)} + \frac{3}{5}$$

$$x_2^{(2)} = \frac{1}{11} x_1^{(1)} + \frac{1}{11} x_3^{(1)} - \frac{3}{11} x_4^{(1)} + \frac{25}{11}$$

$$x_3^{(2)} = -\frac{1}{5} x_1^{(1)} + \frac{1}{10} x_2^{(1)} + \frac{1}{10} x_4^{(1)} - \frac{11}{10}$$

$$x_4^{(2)} = -\frac{3}{8} x_2^{(1)} + \frac{1}{8} x_3^{(1)} + \frac{15}{8}$$

$$x_1^{(k)} = \frac{1}{10}x_2^{(k-1)} - \frac{1}{5}x_3^{(k-1)} + \frac{3}{5}$$

$$x_2^{(k)} = \frac{1}{11}x_1^{(k-1)} + \frac{1}{11}x_3^{(k-1)} - \frac{3}{11}x_4^{(k-1)} + \frac{25}{11}$$

$$x_3^{(k)} = -\frac{1}{5}x_1^{(k-1)} + \frac{1}{10}x_2^{(k-1)} + \frac{1}{10}x_4^{(k-1)} - \frac{11}{10}$$

$$x_4^{(k)} = -\frac{3}{8}x_2^{(k-1)} + \frac{1}{8}x_3^{(k-1)} + \frac{15}{8}$$

69

Results of Jacobi Iteration:

k	0	1	2	3
$x_1^{(k)}$	0.0000	0.6000	1.0473	0.9326
$x_2^{(k)}$	0.0000	2.2727	1.7159	2.0530
$x_3^{(k)}$	0.0000	-1.1000	-0.8052	-1.0493
$x_4^{(k)}$	0.0000	1.8750	0.8852	1.1309

Gauss-Seidel Iterative Technique

Consider the following set of equations.

$$10x_1 \quad - x_2 \quad + 2x_3 \qquad\qquad = 6$$

$$-x_1 \quad + 11x_2 \quad - x_3 \quad + 3x_4 = 25$$

$$2x_1 \quad - x_2 \quad + 10x_3 \quad - x_4 = -11$$

$$3x_2 \quad - x_3 \quad + 8x_4 = 15$$

$$x_1^{(k)} = \frac{1}{10}x_2^{(k-1)} - \frac{1}{5}x_3^{(k-1)} + \frac{3}{5}$$

$$x_2^{(k)} = \frac{1}{11}x_1^{(k-1)} + \frac{1}{11}x_3^{(k-1)} - \frac{3}{11}x_4^{(k-1)} + \frac{25}{11}$$

$$x_3^{(k)} = -\frac{1}{5}x_1^{(k-1)} + \frac{1}{10}x_2^{(k-1)} + \frac{1}{10}x_4^{(k-1)} - \frac{11}{10}$$

$$x_4^{(k)} = -\frac{3}{8}x_2^{(k-1)} + \frac{1}{8}x_3^{(k-1)} + \frac{15}{8}$$

72

$$x_1^{(k)} = \frac{1}{10}x_2^{(k-1)} - \frac{1}{5}x_3^{(k-1)} + \frac{3}{5}$$

$$x_2^{(k)} = \frac{1}{11}x_1^{(k)} + \frac{1}{11}x_3^{(k-1)} - \frac{3}{11}x_4^{(k-1)} + \frac{25}{11}$$

$$x_3^{(k)} = -\frac{1}{5}x_1^{(k)} + \frac{1}{10}x_2^{(k)} + \frac{1}{10}x_4^{(k-1)} - \frac{11}{10}$$

$$x_4^{(k)} = -\frac{3}{8}x_2^{(k)} + \frac{1}{8}x_3^{(k)} + \frac{15}{8}$$

73

Results of Gauss-Seidel Iteration:
(Blue numbers are for Jacobi iterations.)

k	0	1	2	3
$x_1^{(k)}$	0.0000	0.6000	1.0300	1.0065
		0.6000	1.0473	0.9326
$x_2^{(k)}$	0.0000	2.3272	2.0370	2.0036
		2.2727	1.7159	2.0530
$x_3^{(k)}$	0.0000	-0.9873	-1.0140	-1.0025
		-1.1000	-0.8052	-1.0493
$x_4^{(k)}$	0.0000	0.8789	0.9844	0.9983
		1.8750	0.8852	1.1309

The solution is: $x_1 = 1$, $x_2 = 2$, $x_3 = -1$, $x_4 = 1$

It required 15 iterations for Jacobi method and 7 iterations for Gauss-Seidel method to arrive at the solution with a tolerance of 0.00001.

While Jacobi would usually be the slowest of the iterative methods, it is well suited to illustrate an algorithm that is well suited for parallel processing!!!

EXAMPLE Gauss-Seidel Method

Problem Statement. Use the Gauss-Seidel method to obtain the solution for

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85$$
$$0.1x_1 + 7x_2 - 0.3x_3 = -19.3$$
$$0.3x_1 - 0.2x_2 + 10x_3 = 71.4$$

Note that the solution is

$$[x]^T = \lfloor 3 \quad -2.5 \quad 7 \rfloor$$

Solution. First, solve each of the equations for its unknown on the diagonal:

$$x_1 = \frac{7.85 + 0.1x_2 + 0.2x_3}{3} \qquad\qquad \text{(E11.1.1)}$$

$$x_2 = \frac{-19.3 - 0.1x_1 + 0.3x_3}{7} \qquad\qquad \text{(E11.1.2)}$$

$$x_3 = \frac{71.4 - 0.3x_1 + 0.2x_2}{10} \qquad\qquad \text{(E11.1.3)}$$

By assuming that x_2 and x_3 are zero

$$x_1 = \frac{7.85 + 0.1(0) + 0.2(0)}{3} = 2.616667$$

This value, along with the assumed value of $x_3 = 0$, can be substituted into Eq.(E11.1.2) to calculate

$$x_2 = \frac{-19.3 - 0.1(2.616667) + 0.3(0)}{7} = -2.794524$$

The first iteration is completed by substituting the calculated values for x_1 and x_2 into Eq.(E11.1.3) to yield

$$x_3 = \frac{71.4 - 0.3(2.616667) + 0.2(-2.794524)}{10} = 7.005610$$

For the second iteration, the same process is repeated to compute

$$x_1 = \frac{7.85 + 0.1(-2.794524) + 0.2(7.005610)}{3} = 2.990557$$

$$x_2 = \frac{-19.3 - 0.1(2.990557) + 0.3(7.005610)}{7} = -2.499625$$

$$x_3 = \frac{71.4 - 0.3(2.990557) + 0.2(-2.499625)}{10} = 7.000291$$

Example

$$-5x_1 \qquad\qquad +12x_3 \quad = 80$$
$$4x_1 \quad -x_2 \qquad -x_3 \quad = -2$$
$$6x_1 \quad +8x_2 \qquad\qquad = 45$$

$$\begin{bmatrix} -5 & 0 & 12 \\ 4 & -1 & -1 \\ 6 & 8 & 0 \end{bmatrix}$$

Not diagonally dominant !!

Order of the equation can be important

Rearrange the equations to ensure convergence

$$4x_1 \quad -x_2 \qquad -x_3 \quad = -2$$
$$6x_1 \quad +8x_2 \qquad\qquad = 45$$
$$-5x_1 \qquad\qquad +12x_3 \quad = 80$$

$$\begin{bmatrix} 4 & -1 & -1 \\ 6 & 8 & 0 \\ -5 & 0 & 12 \end{bmatrix}$$

Gauss-Seidel Iteration

Rearrange

$$\begin{cases} x_1 = (x_2 + x_3 - 2) / 4 \\ x_2 = (45 - 6 x_1) / 8 \\ x_3 = (80 + 5 x_1) / 12 \end{cases}$$

Assume $x_1 = x_2 = x_3 = 0$

First iteration

$$\begin{cases} x_1 = (0 + 0 - 2) / 4 = -0.5 \\ x_2 = [45 - 6 \times (-0.5)] / 8 = 6.0 \\ x_3 = [80 + 5 \times (-0.5)] / 12 = 6.4583 \end{cases}$$

Gauss-Seidel Method

Second iteration
$$\begin{cases} x_1 = (-2 + 6 + 6.4583)/4 = 2.6146 \\ x_2 = (45 - 6(2.6146))/8 = 3.6641 \\ x_3 = (80 + 5(2.6146))/12 = 7.7561 \end{cases}$$

Third iteration
$$\begin{cases} x_1 = (-2 + 3.6641 + 7.7561)/4 = 2.3550 \\ x_2 = (45 - 6(2.3550))/8 = 3.8587 \\ x_3 = (80 + 5(2.3550))/12 = 7.6479 \end{cases}$$

Fourth iteration
$$\begin{cases} x_1 = (-2 + 3.8587 + 7.6479)/4 = 2.3767 \\ x_2 = (45 - 6(2.3767))/8 = 3.8425 \\ x_3 = (80 + 5(2.3767))/12 = 7.6569 \end{cases}$$

$5th: \quad x_1 = 2.3749, \quad x_2 = 3.8439, \quad x_3 = 7.6562$

$6th: \quad x_1 = 2.3750, \quad x_2 = 3.8437, \quad x_3 = 7.6563$

$7th: \quad x_1 = 2.3750, \quad x_2 = 3.8438, \quad x_3 = 7.6562$

MATLAB M-File for Gauss-Seidel method

```matlab
function x = GaussSeidel(A, b, es, maxit)
% GaussSeidel(A,b): Gauss-Seidel method.
% input:
%    A = Coefficient Matrix
%    b = right hand side vector
%    es = (optional) stop criterion (%) (default = 0.00001)
%    maxit = (optional) maximum iterations (default = 50)
% Output
%    x = solution vector
if nargin < 4, maxit = 50; end
if nargin < 3, es = 0.00001; end
[m,n] = size(A);
if m ~= n, error('Matrix A must be square'); end
C = A;
for i = 1 : n
    C(i,i) = 0;
    x(i) = 0;
end
x = x';
for i = 1 : n
    C(i,1:n) = C(i,1:n) / A(i,i);
end
for i = 1: n
    d(i) = b(i) / A(i,i);
end
```

MATLAB M-File for Gauss-Seidel method

```matlab
disp('         i         x1         x2         x3         x4   ...');
while (1)
    xold = x;
    for i = 1 : n
        x(i) = d(i) - C(i,:) * x;
        if x(i) ~= 0
            ea(i) = abs((x(i) - xold(i)) / x(i)) * 100;
        end
    end
    iter = iter + 1;
    disp([iter  x'])
    if max(ea) <= es | iter >= maxit, break, end
end
if iter >= maxit
disp('Gauss Seidel method did not converge');
disp('results after maximum number of iterations');
else
    disp('Gauss Seidel method has converged');
end
x;
```

Gauss-Seidel Iteration

```
» A = [4 -1 -1; 6 8 0; -5 0 12];
» b = [-2 45 80];
» x=Seidel(A,b,x0,tol,100);
      i           x1          x2          x3          x4 ....
   1.0000     -0.5000      6.0000      6.4583
   2.0000      2.6146      3.6641      7.7561
   3.0000      2.3550      3.8587      7.6479
   4.0000      2.3767      3.8425      7.6569
   5.0000      2.3749      3.8439      7.6562
   6.0000      2.3750      3.8437      7.6563
   7.0000      2.3750      3.8438      7.6562
   8.0000      2.3750      3.8437      7.6563
Gauss-Seidel method converged
```

Converges faster than the Jacobi method shown in next page

Jacobi Iteration

```
» A = [4 -1 -1; 6 8 0; -5 0 12];
» b = [-2 45 80];
» x=Jacobi(A,b,0.0001,100);
       i          x1          x2          x3          x4 ....
  1.0000     -0.5000      5.6250      6.6667
  2.0000      2.5729      6.0000      6.4583
  3.0000      2.6146      3.6953      7.7387
  4.0000      2.3585      3.6641      7.7561
  5.0000      2.3550      3.8561      7.6494
  6.0000      2.3764      3.8587      7.6479
  7.0000      2.3767      3.8427      7.6568
  8.0000      2.3749      3.8425      7.6569
  9.0000      2.3749      3.8438      7.6562
 10.0000      2.3750      3.8439      7.6562
 11.0000      2.3750      3.8437      7.6563
 12.0000      2.3750      3.8437      7.6563
 13.0000      2.3750      3.8438      7.6562
 14.0000      2.3750      3.8438      7.6562

Jacobi method converged
```

(a) Do two iterations with Gauss Jacobi to the system:

$$\begin{pmatrix} 2 & 0 \\ -2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Note that the second iterate is equal to the exact solution.

(b) Is the following claim correct?

The Gauss Jacobi method converges in mostly n iterations if A is a lower triangular matrix

Exercice:

1) Modifier le programme précédent et faites un code pour la méthode de relaxation.
2) Montrer suivant le paramètre omega que la méthode de relaxation est plus rapide que la méthode de GS. Tester votre code sur les exemples précédents.