线性方程组的迭代法

- ◆迭代法的一般形式
  - ➢ Richardson迭代
  - ➢ Jacobi迭代
  - ➢ Gauss-Seidel迭代
  - ➢ 连续过松弛(SOR)迭代
- ◆迭代法的收敛性

线性方程组的迭代法

- 引例：求解方程组$\begin{cases} 3u + v = 5 \\ u + 2v = 5 \end{cases}$

迭代1：$\begin{cases} u_{k+1} = \frac{5 - v_k}{3} \\ v_{k+1} = \frac{5 - u_k}{2} \end{cases}$

迭代2：$\begin{cases} u_{k+1} = 5 - 2v_k \\ v_{k+1} = 5 - 3u_k \end{cases}$

- 需要回答的问题：

1. 为什么需要迭代法？
2. 如何设计通用的迭代法？
3. 迭代的收敛性？

迭代法的一般形式

- $Qx_{k+1} = (Q - A)x_k + b$，$Q$是分裂矩阵
- 不动点形式$x_{k+1} = (I - Q^{-1}A)x_k + Q^{-1}b$
  迭代矩阵$G = (I - Q^{-1}A)$
- 设计原则：易算、收敛
- （定义）赋范线性空间$(V, \|\cdot\|)$向量序列$(v_0, v_1, \dots, v_k, \dots)$收敛：$\lim_{k\to\infty} \|v_k - v\|=0$

Richardson迭代

- $Qx_{k+1} = (Q - A)x_k + b$， $Q = I, G = I - A$

- 例：$\begin{cases} 3u + v = 5 \\ u + 2v = 5 \end{cases}$

系数矩阵$A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$

Richardson迭代：$\begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} -2 & -1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} u_k \\ v_k \end{bmatrix} + \begin{bmatrix} 5 \\ 5 \end{bmatrix}$

Richardson迭代

迭代过程：$x_0 = [0,0]^T$

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} -2 & -1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} -2 & -1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 5 \\ 5 \end{bmatrix} + \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} -10 \\ -5 \end{bmatrix}$$

$$\begin{bmatrix} u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} -2 & -1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} -10 \\ -5 \end{bmatrix} + \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 30 \\ 20 \end{bmatrix}$$

k	u	v
0	0.00000	0.00000
1	5.00000	5.00000
2	-10.00000	-5.00000
3	30.00000	20.00000
4	-75.00000	-45.00000
5	200.00000	125.00000
6	-520.00000	-320.00000
7	1365.00000	845.00000
8	-3570.00000	-2205.00000
9	9350.00000	5780.00000
10	-24475.00000	-15125.00000

Jacobi迭代

- $Qx_{k+1} = (Q - A)x_k + b,\ Q = D,\ G = -D^{-1}(L + U)$

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \ddots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Jacobi迭代

- 例：$\begin{cases} 3u + v = 5 \\ u + 2v = 5 \end{cases}$

系数矩阵$A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$

Jacobi迭代： $Qx_{k+1} = (Q - A)x_k + b$, $Q = D$

$$\begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}^{-1}}_{D^{-1}} \underbrace{\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}}_{D-A} \begin{bmatrix} u_k \\ v_k \end{bmatrix} + \underbrace{\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}^{-1}}_{D^{-1}} \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 & -1/3 \\ -1/2 & 0 \end{bmatrix} \begin{bmatrix} u_k \\ v_k \end{bmatrix} + \begin{bmatrix} 5/3 \\ 5/2 \end{bmatrix}$$

$$= \begin{bmatrix} (5 - v_k)/3 \\ (5 - u_k)/2 \end{bmatrix}$$

Jacobi迭代

迭代过程：$x_0 = [0,0]^T$

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} (5 - v_0)/3 \\ (5 - u_0)/2 \end{bmatrix} = \begin{bmatrix} 5/3 \\ 5/2 \end{bmatrix}$$

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} (5 - v_1)/3 \\ (5 - u_1)/2 \end{bmatrix} = \begin{bmatrix} 5/6 \\ 5/3 \end{bmatrix}$$

$$\begin{bmatrix} u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} (5 - v_2)/3 \\ (5 - u_2)/2 \end{bmatrix} = \begin{bmatrix} 10/9 \\ 25/12 \end{bmatrix}$$

k	u	v
0	0.00000	0.00000
1	1.66667	2.50000
2	0.83333	1.66667
3	1.11111	2.08333
4	0.97222	1.94444
5	1.01852	2.01389
6	0.99537	1.99074
7	1.00309	2.00231
8	0.99923	1.99846
9	1.00051	2.00039
10	0.99987	1.99974

Jacobi迭代

- Jacobi迭代与Richardson迭代的关系

$$
\begin{tikzcd}
x_{k+1} = (I - A)x_k + b & Ax = b \arrow[l, "\text{Richardson}"'] \arrow[r, "\text{Jacobi}"] \arrow[d] & x_{k+1} = (I - D^{-1}A)x_k + D^{-1}b \\
& D^{-1}Ax = D^{-1}b \arrow[ur, "\text{Richardson}"] &
\end{tikzcd}
$$

Gauss-Seidel迭代

- $Qx_{k+1} = (Q - A)x_k + b,\ Q = D + L,\ G = -(L + D)^{-1}U$

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \ddots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Gauss-Seidel迭代

- 例：$\begin{cases} 3u + v = 5 \\ u + 2v = 5 \end{cases}$

系数矩阵$A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$

Gauss-Seidel迭代： $Qx_{k+1} = (Q - A)x_k + b,\ Q = L + D$

$$\underbrace{\begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}}_{L+D} \begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}}_{-U} \begin{bmatrix} u_k \\ v_k \end{bmatrix} + \begin{bmatrix} 5 \\ 5 \end{bmatrix} \Rightarrow \begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} \dfrac{5 - v_k}{3} \\ \dfrac{5 - u_{k+1}}{2} \end{bmatrix}$$

Gauss-Seidel迭代

- Gauss-Seidel迭代与Jacobi迭代的区别与联系

Jacobi迭代： $Dx_{k+1} = (D - A)x_k + b,\ \begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} \frac{5-v_k}{3} \\ \frac{5-\boxed{u_k}}{2} \end{bmatrix}$

Gauss-Seidel迭代： $(D + L)x_{k+1} = (D + L - A)x_k + b, \begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} \frac{5-v_k}{3} \\ \frac{5-\boxed{u_{k+1}}}{2} \end{bmatrix}$

Gauss-Seidel迭代

- 迭代过程：$x_0 = [0,0]^T$

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} \frac{5 - v_0}{3} \\ \frac{5 - u_1}{2} \end{bmatrix} = \begin{bmatrix} \frac{5}{3} \\ \frac{5}{3} \end{bmatrix}$$

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} \frac{5 - v_1}{3} \\ \frac{5 - u_2}{2} \end{bmatrix} = \begin{bmatrix} \frac{10}{9} \\ \frac{35}{18} \end{bmatrix}$$

k	u	v
0	0.00000	0.00000
1	1.66667	1.66667
2	1.11111	1.94444
3	1.01852	1.99074
4	1.00309	1.99846
5	1.00051	1.99974
6	1.00009	1.99996

Gauss-Seidel迭代通常收敛比Jacobi迭代快
Jacobi迭代更适合并行计算

连续超松弛（SOR）迭代

- 对于一般的线性迭代

$$x_{k+1} = Gx_k + c$$

采用外推（松弛）法得到新的迭代

$$x_{k+1} = \gamma(Gx_k + c) + (1 - \gamma)x_k, \gamma \neq 0$$

即：

$$x_{k+1} = G_\gamma x_k + \gamma c, G_\gamma = \gamma G + (1 - \gamma)I$$

连续超松弛（SOR）迭代

- SOR迭代： $x_{k+1} = (1-\omega)x_k + \omega \boxed{D^{-1}(b - Ux_k - Lx_{k+1})}$ Gauss-Seidel
  1. $\omega = 1$, Gauss-Seidel
  2. $\omega > 1$, 超松弛
  3. $\omega < 1$, 欠松弛
- 不动点形式： $Qx_{k+1} = (Q - A)x_k + b$

分裂矩阵： $Q = \frac{1}{\omega}(D + \omega L)$

迭代矩阵： $G = (D + \omega L)^{-1}[(1-\omega)D - \omega U]$

连续超松弛（SOR）迭代

- 例：$\begin{cases} 3u + v = 5 \\ u + 2v = 5 \end{cases}$

Gauss-Seidel迭代： $\begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} \frac{5-v_k}{3} \\ \frac{5-u_{k+1}}{2} \end{bmatrix}$

SOR迭代： $\begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = (1-\omega)\begin{bmatrix} u_k \\ v_k \end{bmatrix} + \omega \begin{bmatrix} \frac{5-v_k}{3} \\ \frac{5-u_{k+1}}{2} \end{bmatrix}$

连续超松弛（SOR）迭代

迭代过程

$\omega = 0.8$,欠松弛

k	u	v
0	0.00000	0.00000
1	1.33333	1.33333
2	1.24444	1.77778
3	1.10815	1.92593
4	1.04138	1.97531
5	1.01486	1.99177
6	1.00517	1.99726
7	1.00176	1.99909
8	1.00060	1.99970
9	1.00020	1.99990

$\omega = 1$,Gauss-Seidel

k	u	v
0	0.00000	0.00000
1	1.66667	1.66667
2	1.11111	1.94444
3	1.01852	1.99074
4	1.00309	1.99846
5	1.00051	1.99974
6	1.00009	1.99996

$\omega = 1.2$,超松弛

k	u	v
0	0.00000	0.00000
1	2.00000	2.00000
2	0.80000	2.00000
3	1.04000	2.00000
4	0.99200	2.00000
5	1.00160	2.00000
6	0.99968	2.00000
7	1.00006	2.00000

连续超松弛（SOR）迭代

ω取值对收敛步数N_{iter}的影响$(TOL = 10^{-5})$

ω	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
N_{iter}	15	12	10	9	7	6	5	7	9	11	14

对比不同迭代法

- 例：

$$\begin{bmatrix} 3 & -1 & 0 & 0 & 0 & \frac{1}{2} \\ -1 & 3 & -1 & 0 & \frac{1}{2} & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & 0 \\ 0 & \frac{1}{2} & 0 & -1 & 3 & -1 \\ \frac{1}{2} & 0 & 0 & 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} \frac{5}{2} \\ \frac{3}{2} \\ 1 \\ 1 \\ \frac{3}{2} \\ \frac{5}{2} \end{bmatrix}$$

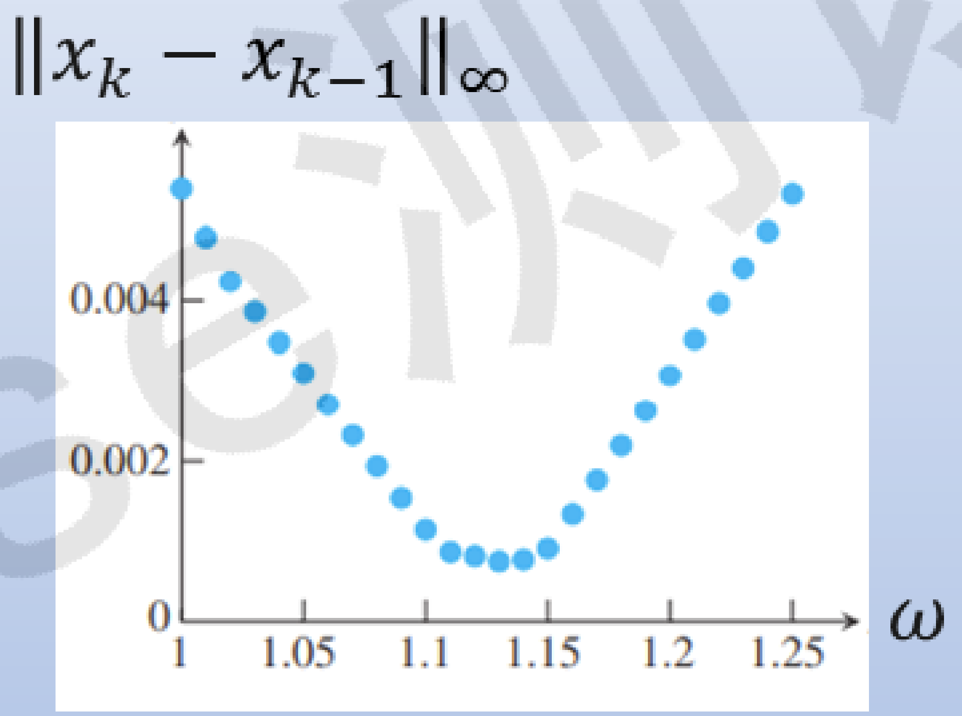
真解：$x = [1,1,1,1,1,1]^T$

对比不同迭代法

$x_0 = [0,0,0,0,0,0]^T$

	Jacobi	Gauss–Seidel	SOR
u_1	0.9879	0.9950	0.9989
u_2	0.9846	0.9946	0.9993
u_3	0.9674	0.9969	1.0004
u_4	0.9674	0.9996	1.0009
u_5	0.9846	1.0016	1.0009
u_6	0.9879	1.0013	1.0004

6步迭代(SOR取$\omega = 1.1$)



6步SOR迭代误差的∞范数

迭代法的代码

- 伪代码

```
integer k, kmax
real array (x^(0))_{1:n}, (b)_{1:n}, (c)_{1:n}, (x)_{1:n}, (y)_{1:n}, (A)_{1:n×1:n}, (Q)_{1:n×1:n}
x ← x^(0)
for k = 1 to kmax do
    y ← x
    c ← (Q − A)x + b
    solve Qx = c
    output k, x
    if ||x − y|| < ε then
        output "convergence"
        stop
    end if
end for
output "maximum iteration reached"
```

迭代法则$Qx_{k+1} = (Q - A)x_k + b$

收敛判别

迭代法的代码

Gauss-Seidel C Code示例

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


#define Ndim 2
#define TOL 1E-3

#define MAX(x,y) (((x)>(y))?(x):(y))
#define MIN(x,y) (((x)<(y))?(x):(y))

double A[Ndim][Ndim]={{3,1},{1,2}};
double b[Ndim]={5,5};

void VectorTransformation_Gauss_Seidel(double x[],double y[]);
void Solve_Linear_System_Lower_Tri(double x[],double c[]);
void VectorAdd(double x[],double y[],double z[]);
void VectorSubtraction(double x[],double y[],double z[]);
void VectorCopy(double x[],double y[]);
double VectorInfinityNorm(double x[]);

void VectorTransformation_Gauss_Seidel(double x[],double y[])//y=(D+L-A)x
{
    int i,j;
    for(i=0;i<Ndim;i++)
    {
        y[i]=0;
        for(j=i+1;j<Ndim;j++)
        {
            y[i]-=A[i][j]*x[j];
        }
    }
}
```

问题定义

计算$(D + L - A)x$

```c
void Solve_Linear_System_Lower_Tri(double x[],double c[])//solve (L+D)x=c
{
    int i,j;
    for(i=0;i<Ndim;i++)
    {
        x[i]=c[i];
        for(j=0;j<i;j++) x[i]-=A[i][j]*x[j];
        x[i]/=A[i][i];
    }
}
```

求解$(D + L)x = c$

```c
void VectorAdd(double x[],double y[],double z[])//z=x+y
{
    int i;
    for(i=0;i<Ndim;i++)
    {
        z[i]=x[i]+y[i];
    }
}
```

向量加法

```c
void VectorSubtraction(double x[],double y[],double z[])//z=x-y
{
    int i;
    for(i=0;i<Ndim;i++)
    {
        z[i]=x[i]-y[i];
    }
}
```

向量减法

```c
void VectorCopy(double x[],double y[])//y=x
{
    int i;
    for(i=0;i<Ndim;i++)
    {
        y[i]=x[i];
    }
}
```

向量复制

```c
double VectorInfinityNorm(double x[])
{
    int i;
    double norm=0;
    for(i=0;i<Ndim;i++)
    {
        norm=MAX(norm,fabs(x[i]));
    }
    return norm;
}
```

向量无穷范数

迭代法的代码

```c
int main()
{
    int i,Niter=0;
    double Error=1E40;
    double c[Ndim],y[Ndim],tem[Ndim];
    double x[Ndim]={0,0};
    FILE *fp;
    fp=fopen("result.txt","w+");
    fprintf(fp,"%d",Niter);
    for(i=0;i<Ndim;i++) fprintf(fp,"\t%.5f",x[i]);
    fprintf(fp,"\n");
    while(Error>TOL&&Niter<100)
    {
        VectorCopy(x,tem);
        VectorTransformation_Gauss_Seidel(x,y);//y=(D+L-A)x
        VectorAdd(y,b,c);//c=y+b=(D+L-A)x+b
        Solve_Linear_System_Lower_Tri(x,c);
        VectorSubtraction(x,tem,y);
        Error=VectorInfinityNorm(y);
        Niter++;
        fprintf(fp,"%d",Niter);
        for(i=0;i<Ndim;i++) fprintf(fp,"\t%.5f",x[i]);
        fprintf(fp,"\n");
    }
    if(Error<=TOL) printf("Convergence!\n");
    else printf("Maximum iteration reached!\n");
    fclose(fp);
    return 1;
}
```

主循环

$tem = x$

$y = (D + L - A)x$

$c = y + b = (D + L - A)x + b$

Solve $(D + L)x = c$

$y = x - tem$

$Error = \|y\|_\infty$

迭代法的收敛性

- 引理（诺伊曼级数定理）：若对某个算子范数$\|G\| < 1$, 则$I-G$可逆，且

$$(I-G)^{-1} = \sum_{k=0}^{\infty} G^k$$

- 迭代收敛定理：若迭代矩阵的某个算子范数$\|I-Q^{-1}A\| = \delta < 1$, 则$x_{k+1} = (I-Q^{-1}A)x_k + Q^{-1}b$收敛于$Ax=b$的解。并且

$$\|x-x_k\| \le \frac{\delta}{1-\delta}\|x_k - x_{k-1}\| \le \cdots \frac{\delta^k}{1-\delta}\|x_1 - x_0\|$$

迭代法的收敛性

- 推论1：如果A是单位严格行/列对角占优矩阵，即

$$a_{ii} = 1 > \sum_{j=1, j\neq i}^{n} |a_{ij}|, (1 \leq i \leq n)$$

或

$$a_{jj} = 1 > \sum_{i=1, i\neq j}^{n} |a_{ij}|, (1 \leq j \leq n)$$

则Richardson迭代$x_{k+1} = (I - A)x_k + b$收敛。

迭代法的收敛性

- 例：　求解$Ax=b$，Richardson迭代能否保证收敛？

$$A=\begin{bmatrix}3 & 1\\ 1 & 2\end{bmatrix}$$ ×

$$A=\begin{bmatrix}1 & 1/2 & 1/3\\ 1/2 & 1 & 1/3\\ 1/3 & 1/2 & 1\end{bmatrix}$$ √

$$A=\begin{bmatrix}1 & 1/2 & 1/2\\ 1/2 & 1 & 1/3\\ 1/3 & 1/3 & 1\end{bmatrix}$$ √

迭代法的收敛性

- 推论2：如果A是严格行对角占优矩阵，即

$$|a_{ii}| > \sum_{j=1,j\neq i}^{n} |a_{ij}|, (1 \le i \le n)$$

则Jacobi迭代$x_{k+1} = -D^{-1}(L+U)x_k + D^{-1}b$收敛。

迭代法的收敛性

- 例：求解$Ax=b$，Jacobi迭代能否保证收敛？

$$\begin{cases} 3u+v=5 \\ u+2v=5 \end{cases} \Rightarrow A=\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$$ √

$$\begin{cases} u-v+3w=b_1 \\ -5u+2v+2w=b_2 \\ 6u+8v+w=b_3 \end{cases} \Rightarrow A=\begin{bmatrix} 1 & -1 & 3 \\ -5 & 2 & 2 \\ 6 & 8 & 1 \end{bmatrix}$$ ×

$$\begin{cases} -5u+2v+2w=b_2 \\ 6u+8v+w=b_3 \\ u-v+3w=b_1 \end{cases} \Rightarrow A=\begin{bmatrix} -5 & 2 & 2 \\ 6 & 8 & 1 \\ 1 & -1 & 3 \end{bmatrix}$$ √

迭代法的收敛性

- Gauss-Seidel迭代收敛性？

$$Qx_{k+1} = (Q - A)x_k + b, \qquad Q = D + L, \qquad G = -(L + D)^{-1}U$$

- Schur酉三角化定理：每个方阵A酉相似于一个上三角阵T，即：

$$A = UTU^*, \qquad (UU^* = I, U^*\text{是}U\text{的共轭转置})$$

- Schur酉三角化定理的推论：每个方阵A相似于非对角元是任意小（可能为复数）的上三角阵。

迭代法的收敛性

- 谱半径定理：
  - 矩阵A的谱半径满足$\rho(A) \le \|A\|$, $\|\cdot\|$为 “任意算子范数"
  - 存在算子范数$\|\cdot\|^*$任意接近$\rho(A)$，即：

$$\|A\|^* \le \rho(A) + \epsilon, \epsilon\text{为任意正数}$$

迭代法的收敛性

- 迭代收敛定理2：迭代$x_{k+1} = Gx_k + c$（c为任意向量）对任意初始向量x_0收敛于$(I - G)^{-1}c$的充要条件是$\rho(G) < 1$.

- 定理：若A严格行对角占优，对任何初始向量Gauss-Seidel迭代$x_{k+1} = -(L + D)^{-1}Ux_k + (L + D)^{-1}b$收敛到$Ax = b$的解。

- 定理：若A的主对角元$a_{ii} \neq 0(i = 1,\ldots,n)$，则SOR迭代($G = (D + \omega L)^{-1}[(1 - \omega)D - \omega U]$)收敛的必要条件是$0 < \omega < 2$。

迭代法的收敛性

- 例：$\begin{cases} 3u + v = 5 \\ u + 2v = 5 \end{cases}$
- Richardson迭代:$G = I - A = \begin{bmatrix} -2 & -1 \\ -1 & -1 \end{bmatrix}, \rho(G) = \frac{3+\sqrt{5}}{2}$ ×
- Jacobi迭代：$G = I - D^{-1}A = \begin{bmatrix} 0 & -\frac{1}{3} \\ -\frac{1}{2} & 0 \end{bmatrix}, \rho(G) = \frac{1}{\sqrt{6}}$ √
- Gauss-Seidel迭代：$G = -(L + D)^{-1}U = \begin{bmatrix} 0 & -\frac{1}{3} \\ 0 & \frac{1}{6} \end{bmatrix}, \rho(G) = \frac{1}{6}$ √

为什么需要迭代方法？

- Gauss消元/LU分解复杂度$O(n^3)$，矩阵乘法$O(n^2)$，迭代法复杂度$O(kn^2)$，k为迭代步数。
- 对于稀疏矩阵，非零元个数为$O(n)$，迭代法优势明显：更少的内存更快的速度！

$$A=\begin{bmatrix}
3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
-1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\
0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 3 & -1 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -1 & 3 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 \\
\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3
\end{bmatrix}
\xrightarrow{\text{消元}}
\begin{bmatrix}
3 & -1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.500 \\
0 & 2.7 & -1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.500 & 0.165 \\
0 & 0 & 2.6 & -1.0 & 0 & 0 & 0 & 0 & 0 & 0.500 & 0.187 & 0.062 \\
0 & 0 & 0 & 2.6 & -1.000 & 0 & 0 & 0 & 0.500 & 0.191 & 0.071 & 0.024 \\
0 & 0 & 0 & 0 & 2.618 & -1.000 & 0 & 0.500 & 0.191 & 0.073 & 0.027 & 0.009 \\
0 & 0 & 0 & 0 & 0 & 2.618 & -1.000 & 0.191 & 0.073 & 0.028 & 0.010 & 0.004 \\
0 & 0 & 0 & 0 & 0 & 0 & 2.618 & -0.927 & 0.028 & 0.011 & 0.004 & 0.001 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.562 & -1.032 & -0.012 & -0.005 & -0.001 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.473 & -1.047 & -0.018 & -0.006 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.445 & -1.049 & -0.016 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.440 & -1.044 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.458
\end{bmatrix}$$

为什么需要迭代方法？

- 例：稀疏矩阵大规模计算：$n = 10^5$

1. Gauss消元：双精度内存$8 \times n^2 = 8 \times 10^{10}$=80G字节；操作数的量级$n^3 = 10^{15}$, CPU频率几个GHz（每秒$10^8$次浮点数操作），计算时间量级$10^7$秒（1年$3 \times 10^7$ 秒）
2. 迭代法：内存$O(n)$，每步迭代操作数$O(n)$，100次迭代的计算时间在秒量级

为什么需要迭代方法？

- 稀疏矩阵求解，精确解为$[0,1,2,\dots,n-1]$：Gauss消元与Gauss-Seidel对比

Gauss-Seidel迭代次数N_{iter}随矩阵维度n的变化($TOL=10^{-7}$)

n	10	10^2	10^3	10^4	10^5	10^6	10^7
N_{iter}	28	56	64	71	78	84	91

计算时间随矩阵维度n的变化(Gauss − Seidel取$TOL=10^{-7}$)

n	10	10^2	10^3	10^4	10^5	10^6	10^7
Gauss消元	0s	0.001s	1.02s	1483.46s	–	–	–
Gauss-Seidel迭代	0s	0s	0.002s	0.02s	0.20s	2.73s	22.65s

思考与练习

- 对方程重新组织，得到一个严格行对角占优系数矩阵，并推导相应的Jacobi迭代和Gauss-Seidel迭代公式

$$\begin{cases} u-8v-2w=1 \\ u+v+5w=4 \\ 3u-v+w=-2 \end{cases}$$

- 编程实现Jacobi迭代、Gauss-Seidel迭代和SOR迭代，并迭代求解以上方程组。所有迭代初始向量取0，误差限取10^{-6}，结果保留8位小数。

- （选做）编程实现Jacobi迭代求解课件中的稀疏系数矩阵的线性方程组，并与之前实现的Gauss消元进行比较。