Metode iterative de rezolvare a sistemelor de ecuatii liniare

Dimitriu Gabriel

February 4, 2005

Contents

Chapter 1

Metoda Jacobi

1.1 Considerente generale

In R^m vom considera sistemul de ecuatii liniare

$$(I - B)x = b \tag{1.1}$$

in care $B = (b_{ij})_{i,j=\overline{1,m}}$ si $b = (b_1, ..., b_m)$ cu acestea sistemul se poate scrie pe componente astfel

$$x_i - \sum_{j=1}^{m} b_{ij} x_j = b_i, i = \overline{1,m} \tag{1.2}$$

Theorem 1 *(Jacobi)*

Afirmatiile urmatoare sunt echivalente:
1) $\lim_{n\to\infty} B^n = 0$
2)$\forall b \in R^m$ $\exists!$ z ai $(I - B)z = b$ si $\forall x^{(0)} \in R^m$ sirul $(x^{(n)})_{n\in N}$ definit prin

$$x_i^{(n+1)} = \sum_{j=1}^{m} b_{ij} x_j^{(n)} + b_i, i = \overline{1,m} \tag{1.3}$$

converge catre z
3)$\rho(B) < 1$
Demonstratie:
Se stie ca 1)$\Leftrightarrow$3) deci nu mai trebuie sa aratam decit ca 1)$\Leftrightarrow$2).
Pentru 1)$\Rightarrow$2) vom presupune ca $\lim_{n\to\infty} B^m = 0$ si vom arta ca $I - B$ este injectiva.
Deoarece $I - B$ este operator liniar este suficient sa demonstram ca $\ker(I - B) = \{0\}$.
Daca $(I - B)x = 0$ $\Leftrightarrow$iterind avem $x = Bx = BBx = B^2x = ...$ deci $x = B^n x, \forall n$.
Din relatia de mai sus si din ipoteza $\lim_{n\to\infty} B^n x = 0$ avem $x = 0$. Deci $I - B$ este injectiva.
Deoarece $\dim(R^m) = m$ finit atunci $I - B$ este surjectiva.
Deoarece $I - B$ este injectiva si surjectiva cum am artat mai sus atunci $I - B$ este bijectiva.
Pentru demostrarea convergentei sirului pornim de la $x^{(n+1)} - z = Bx^{(n)} + b - z$ dar din ipoteza $z - Bz = b$ deci

$$x^{(n+1)} - z = B(x^{(n)} - z)$$

Iterind o data formula de mai sus devine

$$x^{(n+1)} - z = B^2(x^{(n-1)} - z)$$

Iterind de n ori avem

$$x^{(n-1)} - z = B^{n+1}(x^{(0)} - z)$$

Trecind la limita si tinind cont de ipoteza $\lim_{n\to\infty} B^n = 0$ avem

$$\lim_{n\to\infty} x^{(n)} = z$$

Reciproc pentru 2)⇒1)
Fie $b = 0$. Solutia ecuatiei 1.1 este atunci $z = 0$.
Formula de recurenta a sirului $x^{(n+1)} = Bx^{(n)}$ o iteram de n ori si avem

$$x^{(n+1)} = B^{n+1}x^{(0)}$$

trecind la limita si tinind cont de ipoteza ca sirul $(x^{(n)})_{n\in N}$ converge la 0 pentru $\forall x^{(0)} \in R^m$ avem $\lim_{n\to\infty} B^n x^{(0)} = 0, \forall x^{(0)} \in R^m$ deci

$$\lim_{n\to\infty} B^n = 0$$

Theorem 2

Peresupunind ca $\|B\| \le q < 1$. Atunci avem formula de aproximare a erorii

$$\left\|x^{(n)} - z\right\| \le \frac{q}{(1-q)} \left\|x^{(n)} - x^{(n-1)}\right\| \le \frac{q^n}{(1-q)} \left\|x^{(1)} - x^{(0)}\right\|$$

Demonstratie:
Aplicind ipotezei $\|B\| \le q < 1$ proprietatile normei avem: $0 \le \|B^n\| \le \|B\|^n \le q^n$. Aplicind limita acestei relatii si tinind cont ca $q \in [0, 1)$ avem $\lim_{n\to\infty} \|B^n\| = 0$ sau $\lim_{n\to\infty} B = 0$. Datorita ultimei relatii putem aplica Teorema 1 (Jacobi).
Deci $\exists! z$ solutie pentru ecuatia (1.1) si este valabil sirul de aproximari

$$x^{(n+1)} - z^{(n)} = Bx^{(n)} + b - z^{(n)} = Bx^{(n)} + z - Bz - x^{(n)} = (I - B)(z - x^{(n)})$$

Deci $(I - B)(z - x^{(n)}) = x^{(n+1)} - x^{(n)}$ dar $I - B$ este inversabila deci

$$z - x^{(n)} = (I - B)^{-1}B(x^{(n)} - x^{(n-1)})$$

Relatiei anterioare aplicam norma si avem

$$\left\|z - x^{(n)}\right\| = \left\|(I - B)^{-1}B(x^{(n)} - x^{(n-1)}\right\|$$

aplicind relatiei anterioare $\|xy\| \le \|x\| \|y\|$ si $\|a - x\| \le a - \|x\|$ avem:

$$\left\|z - x^{(n)}\right\| \le \left\|(I - B)^{-1}\right\| \|B\| \left\|x^{(n)} - x^{(n-1)}\right\| \le \frac{1}{1 - \|B\|} \|B\| \left\|x^{(n)} - x^{(n-1)}\right\|$$

Aplicind ipoteza $\|B\| \le q$ relatiei anterioare avem

$$\left\| z - x^{(n)} \right\| \leq \frac{q}{1-q} \left\| x^{(n)} - x^{(n-1)} \right\|$$

Tinind cont ca $x^{(n)} - x^{(n-1)} = B^{n-1}(x^{(1)} - x^{(0)})$ avem

$$\left\| z - x^{(n)} \right\| \leq \left\| B^{n-1} \right\| \left\| x^{(1)} - x^{(0)} \right\|$$

Deci in final aplicind ipoteza avem

$$\left\| z - x^{(n)} \right\| \leq \frac{q}{1-q} \left\| x^{(n)} - x^{(n-1)} \right\| \leq \frac{q^n}{1-q} \left\| x^{(1)} - x^{(0)} \right\|$$

QED.

1.2 Metoda lui Jacobi pentru matrici diagonal dominante pe linii

1.2.1 Prezentarea teoretica a metodei

Fie sistemul

$$Ax = a \tag{1.4}$$

unde $A = (a_{ij})_{i,j=\overline{1,m}}, a = (a_1, ..., a_m)$ care se poate scrie pe componente

$$\sum_{j=1}^{m} a_{ij} x_j = a_i, \forall i = \overline{1,m} \tag{1.5}$$

Presupunind ca $a_{ii} \neq 0, \forall i = \overline{1,m}$ avem notatiile

$$D = \begin{pmatrix} a_{11} & 0 \\ 0 & a_{mm} \end{pmatrix} \Rightarrow \exists D^{-1} = \begin{pmatrix} \frac{1}{a_{11}} & 0 \\ 0 & \frac{1}{a_{mm}} \end{pmatrix}$$

Atunci daca inmultim cu D^{-1} sistemul (1.4) el se poate scrie $D^{-1}Ax = D^{-1}a$ care dupa citeva prelucari se poate aduce la forma

$$(I - (I - D^{-1}A))x = D^{-1}a$$

iar daca notam: $B = I - D^{-1}A$ si $b = D^{-1}a$ avem sistemul echivalent pentru (1.4)

$$(I - B)x = b \tag{1.6}$$

Theorem 3 *(Teorema Jacobi pentru matrici dominante pe linii)*

Daca

$$|a_{ii}| > \sum_{j=1, j\neq i}^{m} |a_{ij}|, \forall i = \overline{1,m} \tag{1.7}$$

atunci sistemul (1.6) care este chivalent cu sistemul (1.4) are solutia unica z si $\forall x^{(0)} \in R^m$ sirul $(x^{(n)})_{n\in N}$, $x^{(n+1)} = Bx^{(n)} + b$ converge catre z si au loc relatiile de evaluare a erorii

$$\left\|x^{(n)}-z\right\|_{\infty} \leq \frac{q}{1-q}\left\|x^{(n)}-x^{(n-1)}\right\|_{\infty} \leq \frac{q^n}{1-q}\left\|x^{(1)}-x^{(0)}\right\|_{\infty} \tag{1.8}$$

unde

$$q=\max_{1\leq i\leq m}\sum_{j=1,i\neq j}^{m}\left|\frac{a_{ij}}{a_{ii}}\right|$$

Demonstratie:
Daca facem calculele in relatia $B=I-D^{-1}A$ aceasta devine scrisa pe componente

$$B=\begin{pmatrix} 0 & -\frac{a_{ij}}{a_{ii}} \\ -\frac{a_{ij}}{a_{ii}} & 0 \end{pmatrix}, i=\overline{1,m}, j=\overline{1,m}$$

Aplicind norma infinit asupra matricii $B\in M_{mm}$ avem

$$\|B\|_{\infty}=\max_{1\leq i\leq m}\sum_{j=1,i\neq j}^{m}\left|\frac{a_{ij}}{a_{ii}}\right|$$

Aplicind ipoteza (1.7) avem $\|B\|_{\infty}<1$ sau altfel spus

$$\|B\|_{\infty}=q=\max_{1\leq i\leq m}\sum_{j=1,i\neq j}^{m}\left|\frac{a_{ij}}{a_{ii}}\right|<1$$

Deoarece avem $\|B\|_{\infty}<1$ putem aplica Teorema Jacobi (Theorem 1).
Din Teorema Jacobi avem $\exists! z$ solutie pentru sistemul (1.6)$\Leftrightarrow$(1.4).
Tot din Teorema Jacobi avem $\forall x^{(n)}$sirul $(x^{(n)})_{n\in N}$, definit prin $x^{(n+1)}=Bx^{(n)}+b$ converge catre z si au loc formulele de evaluare ale erorii (1.8).
Sirul se poate scrie pe componente

$$x_i^{(n+1)}=-\sum_{j=1,j\neq i}^{m}\frac{a_{ij}}{a_{ij}}x_j^{(n)}+\frac{a_i}{a_{ii}}, \forall i=\overline{1,m} \tag{1.9}$$

formula care se va utiliza pentru implementare impreuna cu formula de evaluare a erorii (1.8). QED.

1.2.2 Prezentare implementarii in C++

Functia care realizeaza rezolvare sistemului de ecuatii este:

*int jacobi_row(double **mat,double *va,double *xn,double err,long N,int type)*
*/**
returneaza 0 in caz de succes si -1 in caz de insucces.
mat este matricea A, va este vectorul termenilor liber, xn este solutia
err este eroarea cu care dorim sa calculam solutia sistemului
N este dimensiunea sistemului
type:
0 daca se doreste doar rezultatul
1 daca se doreste rezultatul si pasii intermediari scosi in fisierul jacobi_row.dat
2 daca se doreste rezultatul si pasii intermediari scosi in fisierul jacobi_row.dat si pe ecran

```
*/
{
    double *xn_1;
    double max,sum,q;
    long i,j,crt;
    double count;
    for(i=0;i<N;i++)
    {
        sum=0.0;
        for(j=0;j<N;j++) if(j!=i) sum+=fabs(mat[i][j]);
        if(fabs(mat[i][i])<sum)
        {
            cout<<"Sistemul nu poate fi rezolvat deoarece nu este dominant diagonal pe linii\n";
            return -1;
        }
    }
    xn_1=new double[N];
    ofstream file;
    if(type==1 || type==2) file.open("jacobi_row.dat");
    //calculam q
    q=0.0;
    for(j=1;j<N;j++) q+=fabs(mat[0][j]/mat[0][0]);
    for(i=1;i<N;i++)
    {
        sum=0.0;
        for(j=0;j<N;j++) if(j!=i) sum+=fabs(mat[i][j]/mat[i][i]);
        if(q<sum) q=sum;
    }
    max=fabs(va[0]/mat[0][0]);
    for(i=1;i<N;i++)
        if(max<fabs(va[i]/mat[i][i])) max=fabs(va[i]/mat[i][i]);
    count=q*max/(1-q);
    for(i=0;i<N;i++) xn[i]=va[i]/mat[i][i];
    cout<<"q="<<q<<endl;
    cout<<"max="<<max<<endl<<"count="<<fabs(count)<<endl;
    if(type==1 || type==2)
    {
        file<<"q="<<q<<endl;
        file<<"pas=0 err="<<count<<endl;
        if(type==2) cout<<"pas=0 err="<<count<<endl;
        for(i=0;i<N;i++)
        {
            file<<"x["<<i<<"]="<<xn[i]<<endl;
            if(type==2) cout<<"x["<<i<<"]="<<xn[i]<<endl;
        }
    }
    crt=1;
    while(fabs(count)>err)
```

```
    {
        for(i=0;i<N;i++) xn_1[i]=xn[i];
        for(i=0;i<N;i++)
        {
            xn[i]=va[i]/mat[i][i];
            for(j=0;j<N;j++) if(i!=j) xn[i]-=mat[i][j]/mat[i][i]*xn_1[j];
        }
        max=fabs(xn[0]-xn_1[0]);
        for(i=1;i<N;i++)
            if(max<fabs(xn[i]-xn_1[i])) max=fabs(xn[i]-xn_1[i]);
        count=q*max/(1-q);
        if(type==1 || type==2)
        {
            file<<"pas="<<crt<<" err="<<count<<endl;
            if(type==2) cout<<"pas="<<crt<<" err="<<count<<endl;
            for(i=0;i<N;i++)
            {
                file<<"x["<<i<<"]="<<xn[i]<<endl;
                if(type==2) cout<<"x["<<i<<"]="<<xn[i]<<endl;
            }
        }
        crt++;
    }
    if(type==1 || type==2) file.close();
    //Afisez nr pasii
    cout<<"Dupa "<<crt<<" pasi avem solutia"<<endl;
    delete []xn_1;
    return 0;
}
```

Metoda a fost testata cu urmatorul sistem de ecuatii cu eroarea de 0.000001:

$$A = \begin{pmatrix} 10 & 1 & 2 \\ -1 & 7 & 4 \\ -2 & -2 & 10 \end{pmatrix} \quad a = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

obtinind urmatorul rezultat prezentat in tabelul urmator

q=0.714286

pas	err	X[0]	X[1]	X[2]
0	0.357143	0.1	0.142857	0.1
1	0.121429	0.0657143	0.1	0.148571
2	0.0816327	0.0602857	0.0673469	0.133143
3	0.020102	0.0666367	0.0753878	0.125527
4	0.0131487	0.0673559	0.0806472	0.128405
5	0.0038551	0.0662543	0.0791052	0.129601
6	0.00210162	0.0661694	0.0782645	0.129072
7	0.000724995	0.0663592	0.0785545	0.128887
8	0.000332245	0.0663672	0.0786874	0.128983
9	0.000134223	0.0663347	0.0786337	0.129011
10	5.18641e-005	0.0663344	0.078613	0.128994
11	2.45247e-005	0.06634	0.0786228	0.128989
12	7.97582e-006	0.0663398	0.078626	0.128993
13	4.43052e-006	0.0663389	0.0786242	0.128993
14	1.35225e-006	0.0663389	0.0786237	0.128993
15	7.92436e-007	0.0663391	0.0786241	0.128993

OBS:
Se observa ca eroarea este calculata dupa formula

$$\left\| x^{(n)} - z \right\|_{\infty} \leq \frac{q}{1-q} \left\| x^{(n)} - x^{(n-1)} \right\|_{\infty}$$

deoarece aceasta este solutia care converge cel mai repede.

1.3 Metoda Jacobi pentru matrici diagonal dominante pe coloane

1.3.1 Prezentarea teoretica a metodei

Fie sistemul

$$Ax = a \tag{1.10}$$

in care $A = (a_{ij})_{i,j}$ si prespunem ca au loc relatiile

$$|a_{jj}| > \sum_{i=1, i\neq j}^{m} |a_{ij}|, \forall j = \overline{1,m} \tag{1.11}$$

si presupunind ca $a_{ii} \neq 0, \forall i = \overline{1,m}$ avem notatiile

$$D = \begin{pmatrix} a_{11} & 0 \\ 0 & a_{mm} \end{pmatrix} \Rightarrow \exists D^{-1} = \begin{pmatrix} \frac{1}{a_{11}} & 0 \\ 0 & \frac{1}{a_{mm}} \end{pmatrix}$$

Atunci sistemul (1.10) devine

$$AD^{-1}Dx = a \Leftrightarrow (I - (I - AD^{-1}))Dx = a$$

sau

$$(I - C)Dx = a \tag{1.12}$$

unde $C = I - AD^{-1}$ si $I = DD^{-1}$ este matricea unitate de dimensiune m.

Fie sistemul

$$(I - C)y = a \tag{1.13}$$

Theorem 4

Daca au loc ipotezele (1.11) atunci exista si este unic w astfel incit $(I - C)w = a$ si metoda Jacobi pentru (1.13) este convergenta.

Demonstratie:

Vom arata ca $\|C\|_1 < 1$.

Intr-adevar

$$C = I - AD^{-1} = \begin{pmatrix} 1 & ... & 0 \\ .. & 1 & ... \\ 0 & .. & 1 \end{pmatrix} - \begin{pmatrix} 1 & ... & \frac{a_{1m}}{a_{11}} \\ ... & 1 & ... \\ \frac{a_{m1}}{a_{mm}} & ... & 1 \end{pmatrix}$$

deci

$$C = \begin{pmatrix} 0 & ... & -\frac{a_{1m}}{a_{11}} \\ ... & 0 & ... \\ -\frac{a_{m1}}{a_{mm}} & ... & 0 \end{pmatrix}$$

Aplicind norma avem

$$\|C\|_1 = \max_{j \in \overline{1,m}} \sum_{i=1, i \neq j}^{m} \left| \frac{a_{ij}}{a_{jj}} \right| \overset{not}{=} q \overset{(1.11)}{<} 1$$

Atunci metoda Jacobi pentru sistemul (1.12) este convergenta deci

$$\exists!\, w \in R^m \; a.i. \; (I - C)w = a \tag{1.14}$$

si $\forall y^{(0)} \in R^m$ sirul $(y^{(n)})_{n \in N}$ dat de $y^{(n+1)} = Cy^{(n)} + a$ converge catre w si are loc formula de evaluare a erorii:

$$\left\| y^{(n)} - w \right\|_1 \leq \frac{q}{1-q} \left\| y^{(n)} - y^{(n-1)} \right\|_1 \leq \frac{q^n}{1-q} \left\| y^{(1)} - y^{(0)} \right\|_1 \tag{1.15}$$

Fie $z \in R^m$ astfel incit $Dz = w$ deci inlocuind in (1.14) avem $(I - C)Dz = a$ deci $Az = a$ asadar $z = D^{-1}w$.

Din theorema Jacobi avem $y^{(n)} \to w$ deci aplicind continuitatea si liniaritatea lui D^{-1} avem $D^{-1}y^{(n)} \to D^{-1}w = z$.

Dar noi am notat $D^{-1}y^{(n)} = x^{(n)}$ deci $x^{(n)} \to z$.

Pentru evaluarea erorii avem norma:

$$\left\| x^{(n)} - z \right\|_1 = \left\| D^{-1}y^{(n)} - D^{-1}w \right\|_1 = \left\| D^{-1}(y^{(n)} - w) \right\|_1 \leq \left\| D^{-1} \right\|_1 \left\| y^{(n)} - w \right\|_1$$

Aplicind definitia normei 1 avem

$$\left\|x^{(n)} - z\right\|_1 \leq \frac{1}{\min_{j\in\overline{1,m}} |a_{jj}|} \left\|y^{(n)} - w\right\|_1 \overset{(1.15)}{\leq} \frac{q}{\min_{j\in\overline{1,m}} |a_{jj}|} \frac{\left\|y^{(n)} - y^{(n-1)}\right\|_1}{1-q}$$

sau

$$\left\|x^{(n)} - z\right\|_1 \leq \frac{1}{\min_{j\in\overline{1,m}} |a_{jj}|} \frac{q^n}{1-q} \left\|y^{(1)} - y^{(0)}\right\|_1$$

QED.

1.3.2 Prezentare implementarii in C++

Functia care realizeaza rezolvare sistemului de ecuatii este:

```
int jacobi_collumn(double **mat,double *va,double *xn,double err,long N,int type)
/*
    returneaza 0 in caz de succes si -1 in caz de insucces.
    mat este matricea A, va este vectorul termenilor liber, xn este solutia
    err este eroarea cu care dorim sa calculam solutia sistemului
    N este dimensiunea sistemului
    type:
    0 daca se doreste doar rezultatul
    1 daca se doreste rezultatul si pasii intermediari scosi in fisierul jacobi_col.dat
    2 daca se doreste rezultatul si pasii intermediari scosi in fisierul jacobi_col.dat si pe ecran
*/
{
    double *xn_1;
    double *yn,*yn_1;
    double max,sum,q;
    long i,j,crt;
    double count;
    for(i=0;i<N;i++)
    {
        sum=0.0;
        for(j=0;j<N;j++) if(j!=i) sum+=fabs(mat[j][i]);
        if(fabs(mat[i][i])<sum)
        {
            cout<<"Sistemul nu poate fi rezolvat deoarece nu este dominant diagonal pe coloane";
            cout<<endl;
            return -1;
        }
    }
    xn_1=new double[N];
    yn=new double[N];
    yn_1=new double[N];
    ofstream file;
    if(type==1 || type==2) file.open("jacobi_col.dat");
    //calculeaza q
    q=0.0;
```

```
for(i=1;i<N;i++) q+=fabs(mat[i][0]/mat[i][i]);
for(i=1;i<N;i++)
{
    sum=0.0;
    for(j=0;j<N;j++) if(i!=j) sum+=fabs(mat[j][i]/mat[j][j]);
    if(q<sum) q=sum;
}
max=fabs(mat[0][0]);
for(i=1;i<N;i++) if(max>fabs(mat[i][i])) max=fabs(mat[i][i]);
count=q/(max*(1-q));
for(i=0;i<N;i++) yn[i]=va[i];
if(type==1 || type==2)
{
    file<<"q="<<q<<endl;
    file<<"pas=0 err="<<count<<endl;
    if(type==2) cout<<"pas=0 err="<<count<<endl;
    for(i=0;i<N;i++)
    {
        file<<"x["<<i<<"]="<<yn[i]/mat[i][i]<<endl;
        if(type==2) cout<<"x["<<i<<"]="<<yn[i]/mat[i][i]<<endl;
    }
}
crt=1;
sum=0.0;
for(i=0;i<N;i++) sum+=fabs(yn[i]);
count=count*sum;
while(fabs(count)>err)
{
    for(i=0;i<N;i++) yn_1[i]=yn[i];
    for(i=0;i<N;i++)
    {
        yn[i]=va[i];
        for(j=0;j<N;j++) if(i!=j) yn[i]-=mat[i][j]/mat[j][j]*yn_1[j];
        xn[i]=yn[i]/mat[i][i];
    }
    sum=0.0;
    for(i=0;i<N;i++) sum+=fabs(yn[i]-yn_1[i]);
    count=q*sum/(max*(1-q));
    if(type==1 || type==2)
    {
        file<<"pas="<<crt<<" err="<<count<<endl;
        if(type==2) cout<<"pas="<<crt<<" err="<<count<<endl;
        for(i=0;i<N;i++)
        {
            file<<"x["<<i<<"]="<<xn[i]<<endl;
            if(type==2) cout<<"x["<<i<<"]="<<xn[i]<<endl;
        }
    }
```

```
        crt++;
    }
    if(type==1 || type==2) file.close();
    delete []xn_1;
    delete []yn;
    delete []yn_1;
    //afisez numarul de pasi
    cout<<"Dupa "<<crt<<" pasi avem solutia"<<endl;
    return 0;
}
```

Metoda a fost testata cu urmatorul sistem de ecuatii cu eroarea de 0.000001:

$$A = \begin{pmatrix} 10 & 1 & 2 \\ -1 & 7 & 4 \\ -2 & -2 & 10 \end{pmatrix} \quad a = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

obtinind urmatorul rezultat prezentat in tabelul urmator
q=0.771429

pas	err	X[0]	X[1]	X[2]
0	0.482143	0.1	0.142857	0.1
1	0.544133	0.0657143	0.1	0.148571
2	0.210765	0.0602857	0.0673469	0.133143
3	0.0944803	0.0666367	0.0753878	0.125527
4	0.0350961	0.0673559	0.0806472	0.128405
5	0.0162809	0.0662543	0.0791052	0.129601
6	0.00579598	0.0661694	0.0782645	0.129072
7	0.00278644	0.0663592	0.0785545	0.128887
8	0.00094989	0.0663672	0.0786874	0.128983
9	0.0004737	0.0663347	0.0786337	0.129011
10	0.000154402	0.0663344	0.078613	0.128994
11	7.99925e-005	0.06634	0.0786228	0.128989
12	2.62282e-005	0.0663398	0.078626	0.128993
13	1.34171e-005	0.0663389	0.0786242	0.128993
14	4.50005e-006	0.0663389	0.0786237	0.128993
15	2.23492e-006	0.0663391	0.0786241	0.128993
16	7.66892e-007	0.0663391	0.0786241	0.128993

Chapter 2

Metoda Gauss-Seidel

2.1 Prezentarea teoretica a metodei

Fie sistemul

$$(I - B)x = b \tag{2.1}$$

Fie $B = (b_{ij})_{i,j=\overline{1,m}} = L + R$ unde

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ b_{21} & 0 & 0 & 0 & 0 \\ b_{31} & b_{32} & 0 & 0 & 0 \\ .. & .. & .. & .. & .. \\ b_{m1} & b_{m2} & b_{m3} & .. & 0 \end{pmatrix} \quad R = \begin{pmatrix} b_{11} & .. & b_{1m} \\ 0 & .. & .. \\ 0 & 0 & b_{mm} \end{pmatrix}$$

Atunci sistemul (2.1) devine $(I - L - R)x = b \Leftrightarrow (I - (I - L)^{-1}R)x = (I - L)^{-1}b$ si notind cu $C \overset{not}{=} (I - L)^{-1}R$ si cu $c \overset{not}{=} (I - L)^{-1}b$ sistemul se rescrie

$$(I - C)x = c \tag{2.2}$$

Metoda Gauss-Siedel este Metoda Jacobi pentru sistemul (2.2).
Sa consideram sirul $(x^{(n)})_{n \in N}$ definit prin

$$x^{(n+1)} = Cx^{(n)} + c$$

care, revenind la notatiile facute mai inainte, este echivalent cu

$$x^{(n+1)} = (I - L)^{-1}Rx^{(n)} + (I - L)^{-1}b$$

Daca aplicam la stinga $(I - L)$ avem

$$(I - L)x^{(n+1)} = Rx^{(n)} + b$$

Desfacind parantezele si rearanjind avem relatia de recurenta scrisa matricial si pe componente

$$\begin{aligned} x^{(n+1)} &= Lx^{(n+1)} + Rx^{(n)} + b \\ x_i^{(n+1)} &= \sum_{j=1}^{i-1} b_{ij} x_j^{(n+1)} + \sum_{j=i}^{m} b_{ij} x_j^{(n)} + b_i, i = \overline{2,m} \\ x_1^{(n+1)} &= \sum_{j=1}^{m} b_{1j} x_j^{(n)} + b_1 \end{aligned} \tag{2.3}$$

Fie

$$\begin{aligned} q_1 &= \sum_{j=1}^{m} |b_{ij}| \\ q_i &= \sum_{j=1}^{i-1} |b_{ij}|\, q_j + \sum_{j=i}^{m} |b_{ij}|\, , i = \overline{2,m} \end{aligned} \tag{2.4}$$

Fie

$$q = \max_{i=\overline{1,m}} q_i$$

Theorem 5

Daca q<1 atunci sistemul (2.1) care este echivalent cu (2.2) are solutie unica z si $\forall x^{(0)} \in R^m$ sirul definit prin (2.3) converge catre z. Au loc in acelasi timp relatiile de evaluare a erorii:

$$\left\| x^{(n)} - z \right\|_\infty \le \frac{q}{1-q} \left\| x^{(n)} - x^{(n-1)} \right\|_\infty \le \frac{q^n}{1-q} \left\| x^{(1)} - x^{(0)} \right\|_\infty \tag{2.5}$$

Demonstratie:
Vom arata ca $\|C\|_\infty \le q$.
Intr-adevar

$$\|C\|_\infty \stackrel{def}{=} \sup_{\|x\|_\infty \le 1} \|Cx\|_\infty$$

Fie $x = (x_1, ..., x_m); Cx \stackrel{not}{=} y; y = (y_1, ..., y_m)$.
Atunci $Rx = (I - L)y \Leftrightarrow (I - L)^{-1} Rx = y$.
Daca inmultim la stinga cu $(I - L)$ avem

$$y = Ly + Rx \tag{2.6}$$

Fie $y_1 = \sum_{j=1}^{m} b_{1j} x_j$ aplicind modulul si proprietatile lui avem

$$|y_1| \le \sum_{j=1}^{m} |b_{1j}|\, |x_j| \le \left(\sum_{j=1}^{m} |b_{1j}| \right) \|x\|_\infty$$

Aplicind relatiile (2.4) avem

$$|y_1| \le q_1 \|x\|_\infty$$

Presupunem prin inductie ca urmatoarele relatii sunt adevarate:

$$|y_k| \le q_k \|x\|_\infty, 1 \le k \le i-1 \tag{2.7}$$

Din relatia (2.6) avem

$$y_i = \sum_{j=1}^{i-1} b_{ij} y_j + \sum_{j=i}^{m} b_{ij} x_j$$

Aplicind modulul si proprietatile acestuia avem

$$|y_i| \le \sum_{j=1}^{i-1} |b_{ij}| |y_j| + \sum_{j=i}^{m} |b_{ij}| |x_j| \overset{(2.7)}{\le} \left(\sum_{j=1}^{i-1} |b_{ij}| q_j \right) \|x\|_\infty + \left(\sum_{j=i}^{m} |b_{ij}| \right) \|x\|_\infty$$

Deci

$$|y_i| \le \left(\sum_{j=1}^{i-1} |b_{ij}| q_j + \sum_{j=i}^{m} |b_{ij}| \right) \|x\|_\infty \le q_i \|x\|_\infty$$

Deci prin inductie avem $|y_i| \le q_i \|x\|_\infty, i = \overline{1,m}$.
Aplicind maximum relatiei precedente avem $\|y\|_\infty \le q \|x\|_\infty$.
Deci $\|Cx\|_\infty \le q$. Aplicind superior avem $\|C\|_\infty \le q$.
Daca q<1 relatia precedenta devine $\|C\|_\infty \le q < 1$ deci se poate aplica Teorema Jacobi, asadar sistemul (2.1)⇔(2.2) are solutie unica z si $\forall x^{(0)} \in R^m$ sirul definit prin relatia (2.3) converge catre z si au loc relatiile de evaluare a erorii (2.4). QED.

Theorem 6

Daca urmatoarele afirmatii au loc

$$\sum_{j=1}^{m} |b_{ij}| \le 1, \forall i = \overline{1,m} \tag{2.8}$$

si

$$\sum_{j=i}^{m} |b_{ij}| < 1, \forall i = \overline{1,m} \tag{2.9}$$

atunci metoda Gauss-Siedel este convergenta.
Demonstratie:
Vom arata ca are loc teorema (5) adica q<1.
Daca in (2.9) facem pe i=1 avem $\sum_{j=1}^{m} |b_{ij}| < 1$ deci $q_1 < 1$.
Vom arata prin inductie ca $q_k < 1, \forall k = \overline{1,m}$.
Presupunem ca $q_k < 1, \forall k = \overline{1,i-1}$ si aratam prim inductie ca $q_i < 1$, unde

$$q_i = \sum_{j=1}^{i-1} |b_{ij}| q_j + \sum_{j=i}^{m} |b_{ij}|$$

I) Daca $\sum_{j=1}^{i-1} |b_{ij}| q_j = 0$ atunci $q_i = \sum_{j=i}^{m} |b_{ij}| \overset{(2.9)}{<} 1$.
II) Daca $\sum_{j=1}^{i-1} |b_{ij}| q_j \neq 0$ atunci $\exists j_0$ astfel incit $|b_{ij_0}| q_{j_0} \neq 0$, dar

$$|b_{ij_0}| q_{j_0} \overset{ip\,ind}{<} |b_{ij_0}| \tag{2.10}$$

Atunci

$$q_i = \sum_{j=1}^{i-1} |b_{ij}| q_j + \sum_{j=i}^{m} |b_{ij}| \overset{(2.10)}{<} \sum_{j=1}^{i-1} |b_{ij}| + \sum_{j=i}^{m} |b_{ij}| = \sum_{j=1}^{m} |b_{ij}| \overset{(2.8)}{\leq} 1$$

Deci $q_i < 1$.
Conform principiului inductiei matematice atunci

$$q_i < 1 \, \forall i = \overline{1,m}$$

si deci $q < 1$.
Asadar se poate aplica teorema (5). QED.
Fie sistemul de ecuatii

$$Ax = a \tag{2.11}$$

cu $A = (a_{ij})_{i,j=\overline{1,m}}$.
Daca $\exists D^{-1}$ ($a_{ii} \neq 0 \, \forall i = \overline{1,m}$) atunci

$$Ax = a \Leftrightarrow (I - (\underbrace{I - D^{-1}A}_{B}))x = \underbrace{D^{-1}a}_{b}$$

Atunci sistemul (2.11) se poate scrie

$$(I - B)x = b$$

Daca facem calculele in relatia $B = I - D^{-1}A$ aceasta devine scrisa pe componente

$$B = \begin{pmatrix} 0 & -\frac{a_{ij}}{a_{ii}} \\ -\frac{a_{ij}}{a_{ii}} & 0 \end{pmatrix}, i = \overline{1,m}, j = \overline{1,m}$$

deci se poate aplica Gauss-Siedel iar conditiile (2.8) devine

$$\sum_{j=1, j\neq i}^{m} \left| \frac{a_{ij}}{a_{ii}} \right| \leq 1, \forall i = \overline{1,m} \tag{2.12}$$

si (2.9) devine

$$\sum_{j=i+1}^{m} \left| \frac{a_{ij}}{a_{ii}} \right| < 1, \forall i = \overline{1,m} \tag{2.13}$$

Aceste conditii revin la a spune ca matricea A trebuie sa fie diagonal dominanta.

2.2 Prezentarea implementarii in C++

Functia care implementeaza metoda Gauss-Siedel este

```
int gauss_siedel(double **mat,double *va,double *xn,double err,long N,int type)
/*
    returneaza 0 in caz de succes si -1 in caz de insucces.
    mat este matricea A, va este vectorul termenilor liber, xn este solutia
    err este eroarea cu care dorim sa calculam solutia sistemului
    N este dimensiunea sistemului
    type:
    0 daca se doreste doar rezultatul
    1 daca se doreste rezultatul si pasii intermediari scosi in fisier
    2 daca se doreste rezultatul si pasii intermediari scosi in fisier si pe ecran
    fisierul este gauss_siedel.dat
*/
{
    double sum1,sum2,*qi,q,max,count;
    double *xn_1;
    int i,j,crt;
    //verificam daca conditiile de convergenta sunt indeplinite
    for(i=0;i<N;i++)
    {
        sum1=0.0;
        sum2=0.0;
        for(j=0;j<N;j++)
            if(i!=j) sum1+=fabs(mat[i][j]/mat[i][i]);
        for(j=i+1;j<N;j++)
            sum2+=fabs(mat[i][j]/mat[i][i]);
        if(!(sum1<=1 && sum2<1))
        {
            cout<<"Sistemul nu poate fi rezolvat cu metoda Gauss-Siedel\n";
            return -1;
        }
    }
    xn_1=new double[N];
    ofstream file;
    if(type==1 || type==2) file.open("gauss_siedel.dat");
    //calculam q-urile
    qi=new double[N];
    for(i=0;i<N;i++) qi[i]=0.0;
    for(i=0;i<N;i++)
    {
        sum1=0.0;
        for(j=i+1;j<N;j++)
        sum1+=fabs(mat[i][j]/mat[i][i]);
        for(j=0;j<i;j++)
            sum1+=fabs(mat[i][j]/mat[i][i])*qi[j];
        qi[i]=sum1;
```

```
}
//calculam maximul (adica q real)
q=qi[0];
for(i=1;i<N;i++) if(q<qi[i]) q=qi[i];
delete[] qi;
if(q>=1)
{
    cout<<"Sistemul nu poate fi rezolvat cu metodat Gauss-Siedel";
    cout<<" deoarece q="<<q<<">=1\n";
    return -1;
}
//calculam primul pas
for(i=0;i<N;i++) xn_1[i]=0.0;
for(i=0;i<N;i++)
{
    xn[i]=va[i]/mat[i][i];
    for(j=i+1;j<N;j++) xn[i]-=mat[i][j]/mat[i][i]*xn_1[j];
    for(j=0;j<i;j++) xn[i]-=mat[i][j]/mat[i][i]*xn[j];
}
max=fabs(xn[0]-xn_1[0]);
for(i=1;i<N;i++)
    if(max<fabs(xn[i]-xn_1[i])) max=fabs(xn[i]-xn_1[i]);
count=q*max/(1-q);
cout<<"q="<<q<<endl<<"max="<<max<<endl;
cout<<"count="<<fabs(count)<<endl;
if(type==1 || type==2)
{
    file<<"q="<<q<<endl;
    file<<"pas=0 err="<<count<<endl;
    if(type==2) cout<<"pas=0 err="<<count<<endl;
    for(i=0;i<N;i++)
    {
        file<<"x["<<i<<"]="<<xn[i]<<endl;
        if(type==2) cout<<"x["<<i<<"]="<<xn[i]<<endl;
    }
}
crt=1;
while(fabs(count)>err)
{
    for(i=0;i<N;i++) xn_1[i]=xn[i];
    for(i=0;i<N;i++)
    {
        xn[i]=va[i]/mat[i][i];
        for(j=i+1;j<N;j++) xn[i]-=mat[i][j]/mat[i][i]*xn_1[j];
        for(j=0;j<i;j++) xn[i]-=mat[i][j]/mat[i][i]*xn[j];
    }
    max=fabs(xn[0]-xn_1[0]);
    for(i=1;i<N;i++)
```

```
            if(max<fabs(xn[i]-xn_1[i])) max=fabs(xn[i]-xn_1[i]);
        count=q*max/(1-q);
        if(type==1 || type==2)
        {
            file<<"pas="<<crt<<" err="<<count<<endl;
            if(type==2) cout<<"pas="<<crt<<" err="<<count<<endl;
            for(i=0;i<N;i++)
            {
                file<<"x["<<i<<"]="<<xn[i]<<endl;
                if(type==2) cout<<"x["<<i<<"]="<<xn[i]<<endl;
            }
        }
        crt++;
    }
    if(type==1 || type==2) file.close();
    //Afisez nr pasii
    cout<<"Dupa "<<crt<<" pasi avem solutia"<<endl;
    delete []xn_1;
    return 0;
}
```

Metoda a fost testata cu urmatorul sistem de ecuatii cu eroarea de 0.000001:

$$A = \begin{pmatrix} 10 & 1 & 2 \\ -1 & 7 & 4 \\ -2 & -2 & 10 \end{pmatrix} \quad a = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

obtinind urmatorul rezultat prezentat in tabelul urmator
q=0.614286

pas	err	X[0]	X[1]	X[2]
0	0.250265	0.1	0.157143	0.151429
1	0.148274	0.054	0.0640408	0.123608
2	0.0287021	0.0688743	0.0820631	0.130187
3	0.00669693	0.0657562	0.077858	0.128723
4	0.0014952	0.0664696	0.0787969	0.129053
5	0.000337129	0.0663097	0.0785852	0.128979
6	7.58448e-005	0.0663457	0.0786328	0.128996
7	1.70716e-005	0.0663376	0.0786221	0.128992
8	3.84213e-006	0.0663394	0.0786245	0.128993
9	8.64732e-007	0.066339	0.078624	0.128993

Chapter 3

Metode de relaxare

3.1 Prezentarea teoretica a metodei

Pe spatiul R^m vom defini produsul scalar dintre doi vectori astfel
$< x, y >= \sum_{i=1}^{m} x_i y_i$ unde $x = (x_1, .., x_m)$ si $y = (y_1, .., y_m)$.
Fie o matrice $A = (a_{ij})_{i,j=\overline{1,m}}$:

- A este o matrice simetrica daca si numai daca $< Ax, y >=< x, Ay >$,
  $\forall x, y \in R^m$
- daca A este simetrica atunci $S(A) = \{\lambda \in C | \det(A - \lambda I) = 0\} \subset R$
- daca A este simetrica atunci $\lambda \in S(A) \Leftrightarrow \exists x \in R^m, x \neq 0 \, a.i. \, Ax = \lambda x$ λ se numeste numar propriu
- daca A este pozitiv definita atunci $< Ax, x >> 0 \, \forall x \in R^m, x \neq 0$

Proposition 7

Daca A este simetrica si pozitiv definita atunci $S(A) \subset (0, \infty)$.
Demonstratie:
$\lambda \in S(A) \overset{def}{\Rightarrow} \exists x \in R^m, x \neq 0$ $a.i.$ $Ax = \lambda x$ daca aplicam produsul scalar cu x avem $< Ax, x >=< \lambda x, x >$

Deoarece $< Ax, x >> 0$ si $\|x\|_2^2 > 0$ realtia anterioara devine $0 < \lambda < x, x >= \lambda > 0$ atunci $\lambda > 0$.

Reciproc A este simetrica si $S(A) \subset (0, \infty)$ atunci A este pozitiv definita. QED.

Fie B o matrice de dimensiune mxm $B = (b_{ij})_{i,j=\overline{1,m}}$ iar $B^* = (b_{ji})_{i,j=\overline{1,m}}$ atunci avem $< B^* x, y >=< x, By >, \forall x, y \in R^m$ iar $\|B\|_2 = \sup_{\|x\|_2 \leq 1} \|Bx\|_2$ iar $\|x\|_2 = \sqrt{\sum_{i=1}^{m} x_i^2}$ de aici se poate arata ca

$$\|B\|_2 = \sqrt{\rho(BB^*)} \tag{3.1}$$

Fie $A = (a_{ij})_{i,j=\overline{1,m}}$ o matrice simetrica si pozitiv definita, deci $a_{ij} > 0$ in acest caz vom nota

$$D = \begin{pmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ ... & ... & ... & ... \\ 0 & 0 & 0 & a_{mm} \end{pmatrix}$$

Sistemul

$$Ax = a \tag{3.2}$$

este echivalent cu $D^{-1}Ax = D^{-1}a$ iar daca inmultim cu $\alpha > 0$ avem $\alpha D^{-1}Ax = \alpha D^{-1}a$ care este echivalent cu

$$(I - \underbrace{(I - \alpha D^{-1}A)}_{B_\alpha})x = \underbrace{\alpha D^{-1}a}_{b_\alpha}$$

Sistemul (3.2) este echivalent cu sistemul

$$(I - B_\alpha)x = b_\alpha \tag{3.3}$$

unde $B_\alpha = I - \alpha D^{-1}A$ iar $b_\alpha = \alpha D^{-1}a$.

Metoda relaxarii simultane este metoda Jacobi pentru sistemul (3.3).

OBS: Daca A este simetrica si pozitiv definita atuncti A este inversabila sau mai bine zis A inversabila daca si numai daca A este injectiva.

Demonstratie:

Daca $Ax = 0$ atunci $< Ax, x >= 0$ deci x=0 asadar A este injectiva.

Asadar (3.2) are solutie unica z si deci (3.3) are solutie unica z.

Fie $\lambda \in S(D^{-1}A) \Rightarrow \lambda \in R\ \ si\ \ \exists x \neq 0, x \in R^m\ \ avem\ D^{-1}Ax = \lambda x$ de aici avem $Ax = D\lambda x$ sau $Ax = \lambda Dx \Leftrightarrow < Ax, x >= \lambda < Dx, x >= \lambda \sum_{i=1}^{m} a_{ii}x_i^2$. Dar $Ax > 0$ si $a_{ii} > 0$ deci $\lambda > 0$.

Pentru $S(D^{-1}A) = (\lambda_1, \lambda_2, ..., \lambda_m)$ vom presupune ca $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_m$.

Proposition 8

$$S(I - \alpha D^{-1}A) = (1 - \alpha\lambda_1, 1 - \alpha\lambda_2, ..., 1 - \alpha\lambda_m)$$

Demonstratie:

Fie $\lambda \in S(I - \alpha D^{-1}A) \Rightarrow \lambda \in R\ \ si\ \ \exists x \neq 0, x \in R^m\ \ avem\ (I - \alpha D^{-1}A)x = \lambda x$ sau $x - \alpha(D^{-1}A)x = \lambda x$ care cu presupunerea anterioara este $x -\ \alpha\lambda^* x = \lambda x$ sau $(1 - \alpha\lambda^*)x = \lambda x$ rezultind $\lambda = (1 - \alpha\lambda^*)$.

Unde $\lambda^* = (\lambda_1, \lambda_2, ..., \lambda_m)$ din presupunerea anterioara.

Inlocuind avem $S(I - \alpha D^{-1}A) = (1 - \alpha(\lambda_1, \lambda_2, ..., \lambda_m)) = (1 - \alpha\lambda_1, 1 - \alpha\lambda_2, ..., 1 - \alpha\lambda_m)$. QED.

Notam $\langle x, y\rangle_D \stackrel{def}{=} \langle Dx, y\rangle$ unde $\langle , \rangle_D$ este produs scalar deci

$$\|x\|_D = \sqrt{\langle x, x\rangle_D} = \sqrt{\sum_{i=1}^{m} a_{ii}x_i^2}$$

Atunci pentru $B \in M^{mxm}$, $\|B\|_D = \sup_{\|x\|_D \leq 1} \|Bx\|_D$.

Theorem 9 *(Metoda Relaxarii Simultane)*

Fie A simetrica si pozitiv definita. Fie $(x^{(n)})_{n \in N}$ sirul definit prin

$$x^{(n)} = B_\alpha x^{(n)} + b_\alpha \tag{3.4}$$

Fie z solutia ecuatiei (3.2). Sunt echivalente urmatoarele afirmatii:

i) $\forall x^{(0)} \in R^m$ sirul (3.4) converge catre solutia z
ii) $0 < \alpha < 2/\lambda_m$.
Avem atunci urmatorele formale de evaluare a erorii:

$$\left\|x^{(n)} - z\right\|_D \leq \frac{q}{1-q}\left\|x^{(n)} - x^{(n-1)}\right\|_D \leq \frac{q^n}{1-q}\left\|x^{(1)} - x^{(0)}\right\|_D \tag{3.5}$$

unde $q = \max\limits_{1 \leq i \leq m} |1 - \alpha\lambda_i|$.
Demonstratie:
Direct (i)=>(ii):
$(i) \overset{Th\ Jacobi}{\Leftrightarrow} \rho(B_\alpha) < 1$.
Stiim ca $S(B_\alpha) \overset{(8)}{=} (1 - \alpha\lambda_1, 1 - \alpha\lambda_2, ..., 1 - \alpha\lambda_m)$ si ca $\rho(B_\alpha) = \max\limits_{i=\overline{1,m}} |1 - \alpha\lambda_i|$.
Din $\rho(B_\alpha) < 1 \Leftrightarrow |1 - \alpha\lambda_i| < 1\ \forall i = \overline{1,m}$ deci $-1 < 1 - \alpha\lambda_i < 1, \forall i = \overline{1,m}$.
Din $1 - \alpha\lambda_i < 1$ avem $\alpha\lambda_i > 0\, \forall i = \overline{1,m}$ dar $\lambda_i > 0, \forall i = \overline{1,m}$ asadar $\alpha > 0$.
Din $-1 < 1 - \alpha\lambda_i$ avem $\alpha\lambda i < 2$ dar din presupunea ca $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_m$ avem $\alpha\lambda_m < 2 \Leftrightarrow \alpha < 2/\lambda_m$.
Asadar (i)=>(ii).
Reciproc (ii)=>(i) :
Prelucram (i).
Din $\|B_\alpha\|_D = \sqrt{\rho(B_\alpha B_\alpha^*)}$, unde B_α^* este adjunctul lui B_α in raport cu $\langle , \rangle_D$, deci $\langle B_\alpha^* x, y\rangle_D = \langle B_\alpha x, y\rangle_D$ deci $B_\alpha^* = B_\alpha$.
Inlocuind in norma avem: $\|B_\alpha\|_D = \sqrt{\rho(B_\alpha^2)} = \sqrt{\rho^2(B_\alpha)} = \rho(B_\alpha)$.
Asadar $\|B_\alpha\|_D = \rho(B_\alpha) = \max\limits_{1 \leq i \leq m} |1 - \alpha\lambda_i|$.
(ii) este echivalent cu $\rho(B_\alpha) < 1 \Leftrightarrow \|B_\alpha\|_D < 1$.
Evaluarile (3.5) provin din Teorema Jacobi prezentata in (1).
Determinarea sirului de iteratii:
Din relatia (3.3) aplicind Teorema Jacobi (1) avem

$$x^{(n+1)} = B_\alpha x^{(n)} + b_\alpha$$

Inlocuind avem

$$x^{(n+1)} = x^{(n)} - \alpha D^{-1} A x^{(n)} + \alpha D^{-1} a$$

Stiind ca $x^{(n)} = (x_1^{(n)}, x_2^{(n)}, ..., x_m^{(n)})$ avem scrierea relatiei anterioare pe componente:

$$x_i^{(n+1)} = (1 - \alpha)x_i^{(n)} - \alpha \sum_{j=1, j \neq i}^{m} \frac{a_{ij}}{a_{ii}} x_j^{(n)} + \alpha \frac{a_{ii}}{a_{ij}}$$

cu $A = (a_{ij})_{i,j=\overline{1,m}}$ *si* $a = (a_1, a_2, ..., a_m)$. QED.
Se observa ca $q = q(\alpha)$ noi am dori o valoare α a.i. sa minimizeze $q(\alpha)$ fie aceasta valoarea $q_0 = \min\limits_{0<\alpha<2/\lambda_m} q(\alpha)$.
Cu notatiile si ipotezele precedente avem:

$$\min_{0<\alpha<2/\lambda_m} q(\alpha) = q\left(\frac{2}{\lambda_1 + \lambda_m}\right) = \frac{\lambda_m - \lambda_1}{\lambda_m + \lambda_1} \tag{3.6}$$

Demonstratie:

Stiim din ipotezele si notatiile anterioare

$$q(\alpha) = \max_{i=\overline{1,m}} |1 - \alpha\lambda_i| \quad cu \;\; \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_m$$

$$\alpha > 0, \forall i = \overline{1,m} \quad \alpha\lambda_i - 1 \leq \alpha\lambda_m - 1$$

$$\alpha > 0, \forall i = \overline{1,m} \quad 1 - \alpha\lambda_i \leq 1 - \alpha\lambda_1$$

din acestea avem

$\alpha\lambda_i - 1 \leq \max\{\alpha\lambda_m - 1, 1 - \alpha\lambda_1\}$ si $1 - \alpha\lambda_i \leq \max\{\alpha\lambda_m - 1, 1 - \alpha\lambda_1\}$ aceste doua relatii sunt de fapt definitia modului deci vom avea

$$|1 - \alpha\lambda_i| \leq \max\{\alpha\lambda_m - 1, 1 - \alpha\lambda_1\} \, \forall i = \overline{1,m}$$

Aplicind maximum dupa i avem

$$\max_{i=\overline{1,m}} |1 - \alpha\lambda_i| \leq \max\{\alpha\lambda_m - 1, 1 - \alpha\lambda_1\}$$

Vom lua numai relatia de egalitate si avem

$$\max_{i=\overline{1,m}} |1 - \alpha\lambda_i| = \max\{\alpha\lambda_m - 1, 1 - \alpha\lambda_1\}$$

Ceea ce inseamna cu notatiile anterioare

$$q(\alpha) = \max\{\alpha\lambda_m - 1, 1 - \alpha\lambda_1\}$$

Determinam punctul de intersectie al caracteristicilor maximului si avem $\alpha\lambda_m - 1 = 1 - \alpha\lambda_1 \Rightarrow \alpha(\lambda_m + \lambda_1) = 2$ deci

$$\alpha = \frac{2}{\lambda_m + \lambda_1}$$

Evident avem urmatoarele relatii

$$\begin{aligned} \frac{2}{\lambda_m + \lambda_1} &> 0 \\ \frac{2}{\lambda_m + \lambda_1} &< \frac{2}{\lambda_m} \end{aligned}$$

Din aceasta determina valoarea optima a parametrului de relaxare

$$q\left(\frac{2}{\lambda_1 + \lambda_m}\right) = 1 - \frac{2}{\lambda_1 + \lambda_m}\lambda_1 = \frac{\lambda_m - \lambda_1}{\lambda_m + \lambda_1}$$

3.2 Prezentarea implementarii in C++

Programul pentru rezolvarea cu ajutorul metodei relaxarii succesive este

```
#include<iostream.h>
#include<fstream.h>
#include<math.h>
#include<stdlib.h>
int main(int argc, char* argv[])
{
    double **mat,*xn,*va;
    double *temp;
    long i,j,N;
    double err;
    int type;
    cout<<"Introduceti N=";cout.flush();cin>>N;
    cout<<"Introduceti eroarea, err=";cout.flush();cin>>err;
    cout<<"Doriti rulare simpla=0\n";
    cout<<"Doriti rulare cu scoatere in fisier a pasilor intermediari=1\n";
    cout<<"Doriti rulare cu scoatere in fisier si la ecran a pasilor ";
    cout<<intermediari=2\n";
    cin>>type;
    /* aloc memorie */
    mat=(double **)calloc(N,sizeof(double *));
    temp=(double *)calloc(N*N,sizeof(double));
    for(i=0;i<N;i++)
    {
        mat[i]=temp;
        temp+=N;
    }
    xn=new double[N];
    va=new double[N];
    cout<<"Introduceti matricea sistemului\n";
    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            cin>>mat[i][j];
    cout<<"Introduceti vectorul termenilor liberi\n";
    for(i=0;i<N;i++) cin>>va[i];
    double lmin,lmax;
    cout<<"Introduceti valorile minime si maxime ale parametrilor lambda\n";
    cout<<"Lambda minim=";cin>>lmin;
    cout<<"Lambda maxim=";cin>>lmax;
    double q=(lmax-lmin)/(lmax+lmin);
    double alpha=2/(lmin+lmax);
    ofstream file;
    if(type==1 || type==2) file.open("relaxare.dat");
    int crt=0;
    double count=q/(1-q);
    double *xn_1;
```

```
xn_1=new double[N];
for(i=0;i<N;i++) xn_1[i]=0;
//calculam primul pas
for(i=0;i<N;i++)
{
    xn[i]=alpha*va[i]/mat[i][i]+(1-alpha)*xn_1[i];
    for(j=0;j<N;j++)
        if(i!=j) xn[i]-=alpha*mat[i][j]/mat[i][i]*xn_1[j];
}
//calculam eroarea
double sum;
sum=0.0;
for(i=0;i<N;i++) sum+=mat[i][i]*(xn[i]-xn_1[i]);
sum=sqrt(sum);
sum=sum*count;
crt++;
cout<<"q="<<q<<endl;
if(type==1 || type==2)
{
    file<<"q="<<q<<endl;
    file<<"pas=0 err="<<sum<<endl;
    if(type==2) cout<<"pas=0 err="<<sum<<endl;
    for(i=0;i<N;i++)
    {
        file<<"x["<<i<<"]="<<xn[i]<<endl;
        if(type==2) cout<<"x["<<i<<"]="<<xn[i]<<endl;
    }
}
while(sum>err)
{
    for(i=0;i<N;i++) xn_1[i]=xn[i];
    for(i=0;i<N;i++)
    {
        xn[i]=alpha*va[i]/mat[i][i]+(1-alpha)*xn_1[i];
        for(j=0;j<N;j++)
            if(i!=j) xn[i]-=alpha*mat[i][j]/mat[i][i]*xn_1[j];
    }
    //calculam eroarea
    sum=0.0;
    for(i=0;i<N;i++) sum+=mat[i][i]*(xn[i]-xn_1[i])*(xn[i]-xn_1[i]);
    sum=sqrt(sum);
    sum=sum*count;
    if(type==1 || type==2)
    {
        file<<"pas="<<crt<<" err="<<sum<<endl;
        if(type==2) cout<<"pas="<<crt<<" err="<<sum<<endl;
        for(i=0;i<N;i++)
        {
```

```
                file<<"x["<<i<<"]="<<xn[i]<<endl;
                if(type==2) cout<<"x["<<i<<"]="<<xn[i]<<endl;
            }
        }
        crt++;
    }
    if(type==1 || type==2) file.close();
    //Afisez nr pasii
    cout<<"Dupa "<<crt<<" pasi avem solutia"<<endl;
    for(i=0;i<N;i++)
        cout<<"X["<<i<<"]="<<xn[i]<<endl;
    //eliberez memorie
    delete[] xn;
    delete[] va;
    delete[] xn_1;
    free(*mat);
    free(mat);
    return 0;
}
```

Programul a fost verificat cu urmatorul sistem de ecuatii cu eroarea de 0.00001
Metoda a fost testata cu urmatorul sistem de ecuatii cu eroarea de 0.000001:

$$A = \begin{pmatrix} 13 & -1 & 1 \\ -1 & 13 & -1 \\ 1 & -1 & 13 \end{pmatrix} \quad a = \begin{pmatrix} 18 \\ -6 \\ 66 \end{pmatrix}$$

Calculam

$$D^{-1}A = \begin{pmatrix} 1 & -1/13 & 1/13 \\ -1/13 & 1 & -1/13 \\ 1/13 & -1/13 & 1 \end{pmatrix}$$

Calculam raza spectrala

$$S(D^{-1}A) = \{\lambda | \det(D^{-1}A - \lambda I) = 0\}$$

care este echivalent cu

$$\det \begin{pmatrix} 1-\lambda & -1/13 & 1/13 \\ -1/13 & 1-\lambda & -1/13 \\ 1/13 & -1/13 & 1-\lambda \end{pmatrix} = 0$$

Rezolvind avem $S(D^{-1}A) = \{0.923, 0.923, 1.1538\}$ deci avem $\lambda_{\min} = 0.923$ si $\lambda_{\max} = 1.1538$.
Rulind programul obtinem urmatorul rezultat prezentat in tabelul urmator
q=0.111133

pas	err	X[0]	X[1]	X[2]
0	1.0836	1.33341	-0.444471	4.88918
1	0.254842	0.987613	5.26811e-005	4.93828
2	0.0283197	1.00412	-0.00548795	4.99864
3	0.00314707	0.999847	1.30092e-006	4.99924
4	0.000349723	1.00005	-6.77606e-005	4.99998
5	3.88635e-005	0.999998	2.4094e-008	4.99999
6	4.31876e-006	1	-8.36653e-007	5

Chapter 4

Programul principal pentru Jacobi si Gauss-Siedel

```
#include <iostream.h>
    #include <fstream.h>
    #include <math.h>
    #include <stdlib.h>
    int main(int argc, char* argv[])
    {
        double **mat,*xn,*va;
        double *temp;
        long i,j,N;
        double sum,err;
        char test;
        int type;
        cout<<"Introduceti N=";cout.flush();cin>>N;
        cout<<"Introduceti eroarea, err=";cout.flush();cin>>err;
        cout<<"Doriti rulare simpla=0\n";
        cout<<"Doriti rulare cu scoatere in fisier a pasilor intermediari=1\n";
        cout<<"Doriti rulare cu scoatere in fisier si la ecran a pasilor ";
        cout<<intermediari=2\n";
        cin>>type;
        /* aloc memorie */
        mat=(double **)calloc(N,sizeof(double *));
        temp=(double *)calloc(N*N,sizeof(double));
        for(i=0;i<N;i++)
        {
            mat[i]=temp;
            temp+=N;
        }
        xn=new double[N];
        va=new double[N];
        cout<<"Introduceti matricea sistemului\n";
```

```
    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            cin>>mat[i][j];
    cout<<"Introduceti vectorul termenilor liberi\n";
    for(i=0;i<N;i++) cin>>va[i];
    int solutie;
    solutie=jacobi_collumn(mat,va,xn,err,N,type);
    if(solutie==-1)
    {
        cout<<"Sistemul nu se poate rezolva cu metoda Jacobi pe coloane\n";
        solutie=jacobi_row(mat,va,xn,err,N,type);
        if(solutie==-1)
        {
            cout<<"Sistemul nu se poate rezolva cu metoda Jacobi pe ";
            cout<<"rinduri vom incerca Gauss-Siedel\n";
        }
        solutie=gauss_siedel(mat,va,xn,err,N,type);
        if(solutie==-1)
        {
            cout<<"Sistemul nu se poate rezolva ";
            cout<<"prin metoda Gauss-Siedel\n";
            //eliberez memoria
            free(*mat);
            free(mat);
            delete []xn;
            delete []va;
            return 1;
        }
    }
    for(i=0;i<N;i++)
            cout<<"X["<<i<<"]="<<xn[i]<<endl;
    /* eliberez memoria */
    free(*mat);
    free(mat);
    delete []xn;
    delete []va;
    return 0;
}
```