3. SLAJ, iterativne metode

Dat je sistem jednačina:

$$9x_1 + 3x_2 + 1x_3 = 33$$
$$7x_1 + 8x_2 + 9x_3 = 54$$
$$4x_1 + x_2 + 9x_3 = 13$$

matrični oblik:

$$\begin{bmatrix} 9 & 3 & 1 \\ 7 & 8 & 9 \\ 4 & 1 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 33 \\ 54 \\ 13 \end{bmatrix}$$

ili:

$$Ax = b$$

, gde je A matrica množilaca rešenja sistema x, a b vektor slobodnih članova.

1. Definisati matricu A i vektor slobodnih članova b:

```
A = [                                    b = [
     9      3      1                          33
     7      8      9                          54
     4      1      9]                         13]
```

2. Do vektora x se može doći upotrebom operatora `\`:

```
x = A\b
```

Rezultat:

```
x =
     2.0000
     5.0000
     0.0000
```

3. Proveriti tačnost jednakosti:

```
A*x
```

Rezultat:

```
ans =
    33
    54
    13
```

1. *Jacobi* metoda

Zadatak 1. Napisati *Jacobi* iterativnu metodu da za rešavanje sistema linearnih algebarskih jednačina. Pokušati prvo ručno jednu *Jacobi* iteraciju vrstu po vrstu:

```
A = [                          b = [                          x0 = [
    9     3     1                  33                             0
    7     8     9                  54                             0
    4     1     9];                13];                           0];
```

Postaviti tekuće $x^{(1)}$ na početno $x^{(0)}$:

```
x = x0
```

Rezultat:

```
 x =                  A =                   x0 =                  b =
     0                    9     3     1          0                    33
     0                    7     8     9          0                    54
     0                    4     1     9          0                    13
```

Transformisati 1. vrstu iz oblika $Ax = b$ u $x = Tx + c$:

$$
\begin{aligned}
x_1 a_{11} + x_2 a_{12} + x_3 a_{13} &= b_1 \\
x_1 a_{11} &= b_1 - x_2 a_{12} - x_3 a_{13} \\
x_1 &= \frac{1}{a_{11}}(b_1 - x_2 a_{12} - x_3 a_{13}) \\
x_1 &= \frac{1}{a_{11}}\left(b_1 - \begin{bmatrix} a_{12} & a_{13} \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}\right) \\
{x_1}^{(k)} &= \frac{1}{a_{11}}\left(b_1 - \begin{bmatrix} a_{12} & a_{13} \end{bmatrix} \begin{bmatrix} {x_2}^{(k-1)} \\ {x_3}^{(k-1)} \end{bmatrix}\right)
\end{aligned}
$$

```
x(1) = 1/A(1,1)*(b(1) - A(1,2:end)*x0(2:end))
```

Rezultat:

```
 x =                  A =                   x0 =                  b =
    3.6667                9     3     1          0                    33
         0                7     8     9          0                    54
         0                4     1     9          0                    13
```

Transformisati 2. vrstu iz oblika $Ax = b$ u $x = Tx + c$:

$$
\begin{aligned}
x_1 a_{21} + x_2 a_{22} + x_3 a_{23} &= b_2 \\
x_2 a_{22} &= b_2 - x_1 a_{21} - x_3 a_{23} \\
x_2 &= \frac{1}{a_{22}}(b_2 - x_1 a_{21} - x_3 a_{23}) \\
{x_2}^{(k)} &= \frac{1}{a_{22}}(b_2 - {x_1}^{(k-1)} a_{21} - {x_3}^{(k-1)} a_{23})
\end{aligned}
$$

```
x(2) = 1/A(2,2)*(b(2) - A(2,1:1)*x0(1:1) - A(2,3:end)*x0(3:end))
```

Rezultat:

```
 x =                         A =                         x0 =                         b =
     3.6667                      9     3     1                0                          33
     6.7500                      7     8     9                0                          54
          0                      4     1     9                0                          13
```

$$x_1a_{31} + x_2a_{32} + x_3a_{33} = b_3$$
$$x_3a_{33} = b_3 - x_1a_{31} - x_2a_{32}$$
$$x_3 = \frac{1}{a_{33}}(b_3 - x_1a_{31} - x_2a_{32})$$
$$x_3 = \frac{1}{a_{33}}\left(b_3 - \begin{bmatrix} a_{31} & a_{32} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$$
$$x_3{}^{(k)} = \frac{1}{a_{33}}\left(b_3 - \begin{bmatrix} a_{31} & a_{32} \end{bmatrix}\begin{bmatrix} x_1{}^{(k-1)} \\ x_2{}^{(k-1)} \end{bmatrix}\right)$$

```
x(3) = 1/A(3,3)*(b(3) - A(3,1:2)*x0(1:2)
```

Rezultat:

```
 x =                         A =                         x0 =                         b =
     3.6667                      9     3     1                0                          33
     6.7500                      7     8     9                0                          54
     1.4444                      4     1     9                0                          13
```

Pripremiti sledeću iteraciju. Postaviti sledeće $x^{(2)}$ na tekuće $x^{(1)}$:

```
x0 = x
```

Rezultat:

```
 x =                         A =                         x0 =                         b =
     3.6667                      9     3     1           3.6667                          33
     6.7500                      7     8     9           6.7500                          54
     1.4444                      4     1     9           1.4444                          13
```

Uporediti korake:

```
x(1) = 1/A(1,1)*(b(1)                      - A(1,2:end)*x0(2:end))
x(2) = 1/A(2,2)*(b(2) - A(2,1:1)*x0(1:1) - A(2,3:end)*x0(3:end))
x(3) = 1/A(3,3)*(b(3) - A(3,1:2)*x0(1:2))
x0 = x
```

Dopuniti nedostajuće elemente:

```
x(1) = 1/A(1,1)*(b(1) - A(1,1:0)*x0(1:0) - A(1,2:end)*x0(2:end))
x(2) = 1/A(2,2)*(b(2) - A(2,1:1)*x0(1:1) - A(2,3:end)*x0(3:end))
x(3) = 1/A(3,3)*(b(3) - A(3,1:2)*x0(1:2) - A(3,4:end)*x0(4:end))
x0 = x
```

Proširiti indekse:

```
x(1) = 1/A(1,1)*(b(1) - A(1,1:1 - 1)*x0(1:1 - 1) - A(1,1 + 1:end)*x0(1 + 1:end))
x(2) = 1/A(2,2)*(b(2) - A(2,1:2 - 1)*x0(1:2 - 1) - A(2,2 + 1:end)*x0(2 + 1:end))
x(3) = 1/A(3,3)*(b(3) - A(3,1:3 - 1)*x0(1:3 - 1) - A(3,3 + 1:end)*x0(3 + 1:end))
x0 = x
```

1. Pogledati šta je fiksno, a šta promenljivo. Primetiti da promenljivi indeksi rastu od 1, do dimenzije matrice. Ovo se može zapisati jednom *for* petljom, pri čemu promenljivi indeksi zavise od indeksa *for* petlje.

```
for row = 1:rows
    x(row) = 1/A(row,row)*(b(row) - A(row,1:row - 1)*x0(1:row - 1) - A(row,row + 1:end)*x0(row + 1:end));
end
x0 = x;
```

Ako se prethodni postupak ponavlja:

nakon 1. ponavljanja:

```
x =                                         x0 =
    3.6667                                      3.6667
    6.7500                                      6.7500
    1.4444                                      1.4444
```

nakon 2. ponavljanja:

```
x =                                         x0 =
    1.2562                                      1.2562
    1.9167                                      1.9167
   -0.9352                                     -0.9352
```

.

.

.

nakon 53. ponavljanja:

```
x =                                         x0 =
    2.0001                                      2.0001
    5.0001                                      5.0001
    0.0001                                      0.0001
```

nakon 54. ponavljanja:

```
x =                                         x0 =
    1.9999                                      1.9999
    4.9999                                      4.9999
   -0.0000                                     -0.0000
```

nakon 55. ponavljanja:

```
x =                                         x0 =
    2.0000                                      2.0000
    5.0001                                      5.0001
    0.0000                                      0.0000
```

nakon 56. ponavljanja:

```
x =                                         x0 =
    2.0000                                      2.0000
    4.9999                                      4.9999
   -0.0000                                     -0.0000
```

nakon 57. ponavljanja:

```
x =                                         x0 =
    2.0000                                      2.0000
    5.0001                                      5.0001
    0.0000                                      0.0000
```

nakon 58. ponavljanja:

```
x =                                          x0 =
    2.0000                                       2.0000
    4.9999                                       4.9999
   -0.0000                                      -0.0000
```

nakon 59. ponavljanja:

```
x =                                          x0 =
    2.0000                                       2.0000
    5.0000                                       5.0000
    0.0000                                       0.0000
```

nakon 60. ponavljanja:

```
x =
    2.0000
    5.0000
   -0.0000
```

... tekuće i prethodno rešenje će postati blisko ili jednako.

2. Prethodna *for* petlja se može ugnjezditi u beskonačnu *while* petlju:

```
x = x0;
while true
    for row = 1:rows
        x(row) = 1/A(row,row)*(b(row) - A(row,1:row - 1)*x0(1:row - 1) - A(row,row + 1:end)*x0(row + 1:end));
    end

    x0 = x;
end
```

3. Potrebno je definisati uslov za prekid iteracije u zavisnosti od tražene preciznosti:

```
if abs(x - x0) < errMax
    return
end
```

4. Za slučaj da metod **divergira** ili da je iz drugih razloga potrebno ograničiti broj iteracija, potrebno je definisati njihov maksimalni broj. Umesto *while* petljom, to se može iskazati *for* petljom:

```
x = x0;
for it = 1:itMax
    for row = 1:rows
    .
    .
    .
end
```

5. Sada je moguće definisati funkciju koja sadrži prethodni algoritam:

```
function [x, it] = jacobi(A, b, x0, itMax, errMax)
    rows = size(A, 1);
    .
    .
    .
end
```

6. Testirati funkciju *jacobi.m* na primeru, izračunati tačnu vrednost upotrebom operatora \ i izračunati apsolutnu grešku:

```
A = [                           b = [                         x0 = [
     9     3     1                   33                            0
     7     8     9                   54                            0
     4     1     9];                 13];                          0];
[x, it] = jacobi(A, b, x0, 100, 10^-5)
xt = A\b
abs(xt - x)
```

Rezultat:

```
x =
    2.0000
    5.0000
    0.0000

it =
    71

xt =
    2.0000
    5.0000
    0.0000

ans =
   1.0e-05 *

    0.1938
    0.4259
    0.1617
```

2. *Gauss-Seidel* metoda

Zadatak 2. Konvergencija *Jacobi* metode se može **ubrzati**. Napisati *Gauss-Seidel* iterativnu metodu da za rešavanje sistema linearnih algebarskih jednačina.

1. *Gauss-Seidel* za izračunavanje rešenja $x_i^{(k)}$ koristi već izračunate vrednosti ostalih rešenja $x_{j<i}^{(k)}$ iz tekuće iteracije:

```
function [x, it] = gs(A, b, x0, itMax, errMax)
    .
    .
    .
        x(row) = 1/A(row,row)*(b(row) - A(row,1:row - 1)*x(1:row - 1) - A(row,row + 1:end)*x0(row + 1:end));
    .
    .
    .
end
```

2. Testirati funkciju *gs.m* na primeru uporediti rezultat sa metodom *jacobi.m*, izračunati tačnu vrednost upotrebom operatora \ i izračunati apsolutnu grešku:

```
A = [                              b = [                              x0 = [
     9     3     1                      33                                 0
     7     8     9                      54                                 0
     4     1     9];                    13];                               0];
[xJacobi, itJacobi] = jacobi(A, b, x0, 100, 10^-5)
[xGS, itGS] = gs(A, b, x0, 100, 10^-5)
xt = A\b
errJacobi = abs(xt - xJacobi)
errGS = abs(xt - xGS)
```

Rezultat:

```
xJacobi =
    2.0000
    5.0000
    0.0000

itJacobi =
    71

xGS =
    2.0000
    5.0000
   -0.0000

itGS =
    16

xt =
    2.0000
    5.0000
    0.0000

errJacobi =
   1.0e-05 *

    0.1938
    0.4259
    0.1617

errGS =
   1.0e-05 *

    0.0651
    0.2953
    0.0039
```

Zadatak 3. Isprobati *Gauss-Seidel* metodu na sledećem primeru:

```
A = [                               b = [                               x0 = [
     2     5     6                      69                                  0
     5     4     9                     100                                  0
     6     2     3];                    67];                                0];
[x, it] = gs(A, b, x0, 100, 10^-5)
```

Rezultat:

```
x =
   1.0e+81 *

    0.8944
   -0.6344
   -1.3659

it =
   100
```

Metoda nije uspela da pronađe rešenje u ograničenom broju iteracija.

Postoje sistemi kod kojih **nije zagarantovana konvergencija** iterativnih metoda. Odabir drugačijeg početnog rešenja može da reši problem, ali ne uvek.

* **Zadatak 4**. Napisati *Successive Over-Relaxation* metodu za rešavanje sistema linearnih algebarskih jednačina.