2. Gausova eliminacija

Dat je sistem jednačina:

$$5x_1 + 8x_2 + 5x_3 = 48$$
$$4x_1 + 2x_2 + 7x_3 = 43$$
$$8x_1 + 5x_2 + 8x_3 = 69$$

matrični oblik:

$$\begin{bmatrix} 5 & 8 & 5 \\ 4 & 2 & 7 \\ 8 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 48 \\ 43 \\ 69 \end{bmatrix}$$

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 = [
     5      8      5                            48
     4      2      7                            43
     8      5      8]                           69
```

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

```
x = A\b
```

Rezultat:

```
x =
     5
     1
     3
```

3. Proveriti tačnost jednakosti:

```
A*x
```

Rezultat:

```
ans =
    48
    43
    69
```

Zadatak 1. Definisati funkciju *upperTriangular.m* koja proizvoljni sistem jednačina svodi na gornji trougaoni oblik. Pokušati prvo ručno svođenje sistema na gornji trougaoni oblik.

```
A =
     5     8     5
     4     2     7
     8     5     8
```

Potrebno je eliminisati sve elemente ispod glavne dijagonale.

1. Odabrati 1. element 2. vrste, podeliti ga 1. elementom 1. vrste, a zatim dobijenim množiocem pomnožiti 1. vrstu i oduzeti je od 2. vrste:

```
A(2,:) = A(2,:) - A(1,:)*A(2,1)/A(1,1)
```

Rezultat:

```
A =
    5.0000    8.0000    5.0000
         0   -4.4000    3.0000
    8.0000    5.0000    8.0000
```

Svi elementi ispod glavne dijagonale u 2. vrsti su sada eliminisani.

2. Odabrati 1. element 3. vrste, podeliti ga 1. elementom 1. vrste, a zatim dobijenim množiocem pomnožiti 1. vrstu i oduzeti je od 3. vrste:

```
A(3,:) = A(3,:) - A(1,:)*A(3,1)/A(1,1)
```

Rezultat:

```
A =
    5.0000    8.0000    5.0000
         0   -4.4000    3.0000
         0   -7.8000         0
```

1. kolona u 3. vrsti je sada eliminisana. Preostaje još 2. kolona.

3. Odabrati 2. element 3. vrste, podeliti ga 2. elementom 2. vrste, a zatim dobijenim množiocem pomnožiti 2. vrstu i oduzeti je od 3. vrste:

```
A(3,:) = A(3,:) - A(2,:)*A(3,2)/A(2,2)
```

Rezultat:

```
A =
    5.0000    8.0000    5.0000
         0   -4.4000    3.0000
         0         0   -5.3182
```

Svi elementi ispod glavne dijagonale u 3. vrsti su sada eliminisani. Matrica je svedena na gornji trougaoni oblik.

Potrebno je definisati algoritam koji će ovo uraditi za proizvoljnu kvadratnu matricu.

1. Definisati praznu funkciju *upperTriangular.m*:

```
function A = upperTriangular(A)

end
```

Uporediti korake:

- za 1. vrstu:
  ```
  A(2,:) = A(2,:) - A(1,:)*A(2,1)/A(1,1)
  A(3,:) = A(3,:) - A(1,:)*A(3,1)/A(1,1)
  ```
- za 2. vrstu:
  ```
  A(3,:) = A(3,:) - A(2,:)*A(3,2)/A(2,2)
  ```
- za 3. vrstu nema elemenata ispod glavne dijagonale

Odabrati 1. vrstu i pogledati šta je fiksno, a šta promenljivo. Primetiti da promenljivi indeksi rastu od indeksa vrste + 1 do dimenzije matrice. Ovo se može zapisati jednom *for* petljom, pri čemu promenljivi indeksi zavise od indeksa *for* petlje:

```
for it2 = 1 + 1:rows
    A(it2,:) = A(it2,:) - A(1,:)*A(it2,1)/A(1,1)
end
```

Posmatrati ovu *for* petlju za sve vrste:

```
for it2 = 1 + 1:rows
    A(it2,:) = A(it2,:) - A(1,:)*A(it2,1)/A(1,1)
end

for it2 = 2 + 1:rows
    A(it2,:) = A(it2,:) - A(2,:)*A(it2,2)/A(2,2)
end
```

Pogledati šta je fiksno, a šta promenljivo. Primetiti da promenljivi indeksi rastu od 1, do dimenzije matrice - 1.

2. Prethodna *for* petlja se može ugnjezditi u novu *for* petlju pri čemu promenljivi indeksi zavise od indeksa nove for petlje. U funkciji definisati par *for* petlji:

```
for it1 = 1:rows - 1
    for it2 = it1 + 1:rows
        A(it2,:) = A(it2,:) - A(it1,:)*A(it2,it1)/A(it1,it1);
    end
end
```

3. **Pre for petlji** odrediti dimenziju matrice:

```
rows = size(A, 1);
```

4. Testirati funkciju *upperTriangular.m* na primeru:

```
A = [
     5     8     5
     4     2     7
     8     5     8];
upperTriangular(A)
```

Rezultat:

```
ans =
    5.0000    8.0000    5.0000
         0   -4.4000    3.0000
         0         0   -5.3182
```

Zadatak 2. Dopuniti funkciju *upperTriangular.m* tako da transformiše i vektor kolone slobodnih članova b, da bi sistem jednačina ostao ekvivalentan:

1. Modifikovati zaglavlje funkcije *upperTriangular.m*, tako da prihvata i vraća modifikovani vektor kolone slobodnih članova b:

```
function [A, b] = upperTriangular(A, b)
    .
    .
    .
end
```

2. Da bi sistem ostao ekvivalentan, svaka transformacija svake vrste matrice A mora se odraziti na odgovarajuću vrstu vektora b, uz napomenu da vektor b ima samo vrste, ali ne i kolone:

```
A(it2,:) = A(it2,:) - A(it1,:)*A(it2,it1)/A(it1,it1);
b(it2) = b(it2) - b(it1)*A(it2,it1)/A(it1,it1);
```

3. Primetiti da se množilac izračunava 2 puta. Postupak je moguće optimizovati:

```
m = A(it2,it1)/A(it1,it1);
A(it2,:) = A(it2,:) - A(it1,:)*m;
b(it2) = b(it2) - b(it1)*m;
```

4. Testirati funkciju *upperTriangular.m* na primeru:

```
A = [                                         b = [
     5     8     5                                 48
     4     2     7                                 43
     8     5     8];                               69];
[A, b] = upperTriangular(A, b)
```

Rezultat:

```
A =                                                   b =
    5.0000    8.0000    5.0000                          48.0000
         0   -4.4000    3.0000                           4.6000
         0         0   -5.3182                         -15.9545
```

Zadatak 3. Definisati funkciju *solveUpperTriangular.m* koja za sistem jednačina sveden na gornji trougaoni oblik (zadan kvadratnom matricom A i vektorom kolone b) nalazi vektor kolone rešenja x:

Pokušati prvo ručno izračunavanje vektora x.

1. Inicijalizovati vektor kolone rešenja x:

```
x = [
     0
     0
     0]
```

Ako je:

```
A =                                                   b =
    5.0000    8.0000    5.0000                          48.0000
         0   -4.4000    3.0000                           4.6000
         0         0   -5.3182                         -15.9545
```

$$
\begin{aligned}
5.0000x_1 + 8.0000x_2 + 5.0000x_3 &= 48.0000 \\
\cancel{0.0000x_1} - 4.4000x_2 + 3.0000x_3 &= 4.6000 \\
\cancel{0.0000x_1 + 0.0000x_2} - 5.3182x_3 &= -15.9545
\end{aligned}
$$

2. x_3, a zatim x_2 i na kraju x_1 se mogu izračunati direktno:

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

Rezultat:

```
x =
    5.0000
    1.0000
    3.0000
```

Potrebno je definisati algoritam koji će ovo uraditi za proizvoljnu kvadratnu matricu.

1. Definisati praznu funkciju *solveUpperTriangular.m*:

```
function x = solveUpperTriangular(A, b)

end
```

Uporediti korake:

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

Pretvoriti u vektorski zapis:

```
x(3) = (b(3) - [                  ]*[          ])/A(3, 3)
x(2) = (b(2) - [         A(2, 3)]*[      x(3)])/A(2, 2)
x(1) = (b(1) - [A(1, 2) A(1, 3)]*[x(2) x(3)])/A(1, 1)
```

Zapisati vektore kao izraze:

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

Proširiti indekse:

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

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

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

3. **Pre for petlje** odrediti dimenziju matrice i inicijalizovati vektor x:

```
rows = size(A, 1);
x = zeros(rows, 1);
```

4. Testirati funkciju *solveUpperTriangular.m* na primeru:

```
A = [                                          b = [
     5     8     5                                  48
     4     2     7                                  43
     8     5     8];                                69];
[A, b] = upperTriangular(A, b);
x = solveUpperTriangular(A, b)
```

Rezultat:

```
x =
    5.0000
    1.0000
    3.0000
```

Zadatak 4. Objediniti pozive funkcija *upperTriangular.m* i *solveUpperTriangular.m* u jedan poziv funkcije *gauss.m*.

1. Definisati funkciju *gauss.m*:

```
function x = gauss(A, b)
   [A, b] = upperTriangular(A, b);
   x = solveUpperTriangular(A, b);
end
```

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

```
A = [                                                  b = [
     5      8      5                                       48
     4      2      7                                       43
     8      5      8];                                     69];
x = gauss(A, b)
xt = A\b
abs(xt - x)
```

Rezultat:

```
x =
    5.0000
    1.0000
    3.0000

xt =
     5
     1
     3

ans =
   1.0e-14 *

    0.1776
    0.1110
         0
```

Zadatak 5. Definisati funkciju *gauss_PP.m* koja rešava sistem jednačina Gausovom eliminacijom, uvodeći parcijalni *pivoting*.

Pokušati prvo ručno pronalaženje i zamenu pivotske vrste.

```
A = [                                                  b = [
     5      8      5                                       48
     4      2      7                                       43
     8      5      8]                                      69]
```

Za eliminaciju elemenata u 1. koloni ispod 1. vrste najpogodniji je pivotski element u 3. vrsti jer ima najveću apsolutnu vrednost.

1. Izolovati 1. kolonu matrice:

```
A(1:3, 1)
```

Rezultat:

```
ans =
     5
     4
     8
```

2. Naći apsolutne vrednosti 1. kolone matrice:

```
abs(A(1:3, 1))
```

Rezultat:

```
ans =
     5
     4
     8
```

3. Naći po apsoulutnoj vrednosti maksimalni element 1. kolone matrice i njegov indeks:

```
[m, mi] = max(abs(A(1:3, 1)))
```

Rezultat:

```
m =
     8

mi =
     3
```

4. Ignorisati po apsolutnoj vrednosti maksimalni element 1. kolone matrice, a naći njegov indeks:

```
[~, mi] = max(abs(A(1:3, 1)))
```

Rezultat:

```
mi =
     3

A =
     5     8     5
     4     2     7
     8     5     8
```

Za eliminaciju elemenata u 2. koloni ispod 2. vrste najpogodniji je pivotski element u 3. vrsti jer ima najveću apsolutnu vrednost.

5. Ignorisati po apsolutnoj vrednosti maksimalni element 2. kolone matrice ispod 2. vrste matrice, a naći njegov indeks:

```
[~, mi] = max(abs(A(2:3, 2)))
```

Rezultat:

```
mi =
     2
```

Indeks 2 jeste indeks po apsolutnoj vrednosti najvećeg elementa 2. kolone **ako bi se indeksi brojali od 2. vrste**, tj. ako bi 2. vrsta imala indeks 1 (to se dešava jer je u izrazu kolona efektivno odsečena do 2. vrste).

6. Korigovati indeks po apsolutnoj vrednosti najvećeg elementa 2. kolone dodajući mu indeks 2. vrste umanjen za 1, tako da bude validan za celu kolonu:

```
[~, mi] = max(abs(A(2:3, 2)));
mi = mi + 2 - 1
```

Rezultat:

```
mi =
     3
```

7. Primeniti korigovani izraz za ponovno pronalaženje indeksa pivotskog elementa u 1. koloni, a zatim zameniti 1. vrstu sa vrstom pronađenog pivotskog elementa matrice A i primeniti istu transformaciju na vektor b da bi sistem ostao ekvivalentan:

```
[~, mi] = max(abs(A(1:3, 1)));
mi = mi + 1 - 1;
A([1,mi],:) = A([mi,1],:)
b([1,mi]) = b([mi,1])
```

Rezultat:

```
A =
     8     5     8
     4     2     7
     5     8     5

b =
    69
    43
    48
```

Uporediti korake:

- za 1. vrstu:
```
[~, mi] = max(abs(A(1:3, 1)));
mi = mi + 1 - 1;
A([1,mi],:) = A([mi,1],:)
b([1,mi]) = b([mi,1])
```
- za 2. vrstu:
```
[~, mi] = max(abs(A(2:3, 2)));
mi = mi + 2 - 1;
A([2,mi],:) = A([mi,2],:)
b([2,mi]) = b([mi,2])
```
- za 3. vrstu nema elemenata ispod glavne dijagonale

Primetiti šta je promenljivo. Promenljivi indeksi rastu od 1 do dimenzije matrice - 1.

1. Napraviti kopiju funkcije *upperTriangular.m* i nazvati je *upperTriangular_PP.m*. Uklopiti blok za odabir i zamenu pivotske vrste u spoljašnjoj *for* petlji funkcije *upperTriangular_PP.m*, pre eliminacije, tako da promenljivi indeksi zavise od indeksa spoljašnje for petlje:

```
function [A, b] = upperTriangular_PP(A, b)
        .
        .
        .
    for it1 = 1:rows - 1
        [~, mi] = max(abs(A(it1:rows, it1)));
        mi = mi + it1 - 1;
        A([it1,mi],:) = A([mi,it1],:);
        b([it1,mi]) = b([mi,it1]);
        for it2 = it1 + 1:rows
                .
                .
                .
```

2. Napraviti kopiju funkcije *gauss.m* i nazvati je *gauss_PP.m*. U njoj umesto funkcije *upperTriangular.m* pozvati funkciju *upperTriangular_PP.m*:

```
function x = gauss_PP(A, b)
    [A, b] = upperTriangular_PP(A, b);
    x = solveUpperTriangular(A, b);
end
```

3. Testirati funkciju *gauss_PP.m* na primer, izračunati tačnu vrednost upotrebom operatora \ i izračunati apsolutnu grešku:

```
A = [                                    b = [
     5     8     5                            48
     4     2     7                            43
     8     5     8];                          69];
x = gauss_PP(A, b)
xt = A\b
abs(xt - x)
```

Rezultat:

```
x =
     5
     1
     3

xt =
     5
     1
     3

ans =
     0
     0
     0
```

* **Zadatak 6**. Definisati funkciju *gauss_CP.m* koja rešava sistem jednačina Gausovom eliminacijom, uvodeći kompletan *pivoting*.

* **Zadatak 7**. Definisati funkciju *gauss_LT.m* koja rešava sistem jednačina Gausovom eliminacijom, svodeći sistem na donji trougaoni oblik.