NAD-Numerička analiza
Predispitni rad

Stefan Kangrga-Mikulić, 2017/526
Fizička elektronika 13E082NUM, Elektrotehnički fakultet, Beograd

16.09.2021.godine

Članovi tima: Stefan Kangrga-Mikulić, 2017/526

Zadatak 1.5

Izlazna struja solarne ćelije zavisi od napona. Napon V_{mp} pri kome je izlazna struja maksimalna je dat jednačinom:

$$e^{\frac{q \cdot V_{mp}}{k_B \cdot T}} \cdot (1 + \frac{q \cdot V_{mp}}{k_B \cdot T}) = \frac{q \cdot V_{OC}}{k_B \cdot T}$$

gde je V_{OC} napon otvorenog kola, T temperatura u kelvinima, $q = 1,6022 \cdot 10^{-19} C$ naelektrisanje elektrona, $k_B = 1.3806 \cdot 10^{-23} J/k$ Bolcmanova konstanta. Neka je $V_{OC} = 0.5V$ i $T = 297K$. Razviti program koji izračunava napon V_{mp} za koji je izlazna struja iz solarne ćelije maksimalna. Primeniti metodu proste iteracije sa početnom iteracijom $V_{mp}^0 = 0.5V$ sa tačnošću 10^{-3}.

1 Opis algoritma

Na početku mora da se preformuliše polazna jednačina na oblik:

$$g(x) = x$$

za neku funkciju g, ali tako da za svako $x \in [a, b]$, gde je interval $[a, b]$ gde se nalazi približna vrednost, važi da je $a \leq g(x) \leq b$ i da je $\mid g'(x) \mid \leq k$ pri čemu je $0 < k < 1$. Od polazne jednačine se dobija:

$$V_{mp} = \frac{k_B \cdot T}{2 \cdot q} \cdot (\frac{q \cdot V_{OC}}{k_B \cdot T} - \ln \frac{q \cdot V_{mp}}{k_B \cdot T})$$

tj.

$$g(x) = \frac{k_B \cdot T}{2 \cdot q} \cdot (\frac{q \cdot V_{OC}}{k_B \cdot T} - \ln \frac{q \cdot x}{k_B \cdot T})$$

Iterativna metoda glasi:

$$x_{n+1} = \frac{k_B \cdot T}{2 \cdot q} \cdot (\frac{q \cdot V_{OC}}{k_B \cdot T} - \ln \frac{q \cdot x_n}{k_B \cdot T})$$

Deo koda koji vrši iteraciju :

```
while (radi) {
    prethodni_rezultat = pomocna_promenljiva;
    pomocna_promenljiva = (kB * T) / (2 * q) *((q * Voc) / (
   kB * T) - log((q * pomocna_promenljiva) / (kB * T)));
    broj_iteracije++;
    cout << broj_iteracije << "    " << pomocna_promenljiva <<
    endl;
    if (broj_iteracije == 10 || radi == uporedi(
   pomocna_promenljiva, prethodni_rezultat)) { break; }
  }
```

Za $x \in [0, 0.5]$ važi da je $\mid g'(x) \mid = \mid \frac{k_B \cdot T}{2 \cdot q \cdot x} \mid \leq 0.0256 < 1$. Sada su ispunjeni svi uslovi da se primeni ova iterativna metoda sa $x_0 = V_{mp}^0 = 0.5$.

Za kriterijum zaustavljanja nalazimo:

$$\frac{0.1^n}{1 - 0.1} \cdot \mid 0.211966 - 0.5 \mid \leq 10^{-3}$$

da je za $n = 3$ ispunjena ocena.

n	$x_n (= g(x_{n-1}))$
0	0.5
1	0.211966
2	0.222947
3	0.222301
4	0.222338
5	0.222336
6	0.222336

2 Komentari

Imajući u vidu na zahtevanu tačnost, poklapanja vrednosti na 4 decimale je došlo već u 3. i 4.iteraciji,pa je program tako napisan da ipak radi dalje dok ne dodje do poklapanja na 6 decimale.

3 Dodatak

Main.cpp

```cpp
#include <cmath>
#include <iostream>
using namespace std;

bool uporedi(float trenutni_rezultat, float prethodni_rezultat) {
  if (fabs(trenutni_rezultat - prethodni_rezultat) < 0.000001f) {
    return true;
  }
  return false;
}

int main() {
  float q = 0.00000000000000000016022;
  float kB = 0.00000000000000000000013806;
  float Voc = 0.5;
  float T = 297;
  float Vmp_pocetno = 0.5;
  float pomocna_promenljiva = 0, prethodni_rezultat = 0;
  int broj_iteracije = 0;
  bool radi = true;

  cout << "Pocetna jednacina se moze tranformisati u: Vmp=\n" <<  (kB * T)/(2*q) << " * ( " << (q * Voc) / (kB * T) << " - ln( " << q / (kB * T) << "*Vmp))" << endl << endl;
  pomocna_promenljiva = Vmp_pocetno;

  cout << broj_iteracije << "    " << pomocna_promenljiva << endl;
  while (radi) {
    prethodni_rezultat = pomocna_promenljiva;
    pomocna_promenljiva = (kB * T) / (2 * q) *((q * Voc) / (kB * T) - log((q * pomocna_promenljiva) / (kB * T)));
    broj_iteracije++;
    cout << broj_iteracije << "    " << pomocna_promenljiva << endl;
    if (broj_iteracije == 10 || radi == uporedi(pomocna_promenljiva, prethodni_rezultat)) { break; }
    //radi = uporedi(pomocna_promenljiva, prethodni_rezultat);
  }

}
```

Zadatak 2.5

Struje i_1, i_2, i_3, i_4, i_5 u kolu prikazanom na slici 3, mogu se odrediti rešavanjem sistema jednacina:

$$\begin{aligned} 9.5i1 - 2.5i2 - 2i4 &= 12 \\ -2.5i1 + 11i2 - 3.5i3 - 5i5 &= -16 \\ -3.5i2 + 15.5i3 - 4i5 &= 14 \\ -2i1 + 7i4 - 3i5 &= 10 \\ -5i2 - 4i3 - 3i4 + 12i5 &= -30 \end{aligned}$$

Razviti program koji izračunava struje primenom Gausove metode eliminacije sa izborom pivota i LU dekompozicije. Razviti program koji primenjuje Gaus-Zajdelovu metodu i ispitati da li ova metoda konvergira za početnu iteraciju koja je jednaka nula-vektoru. Komentarisati dobijene rezultate.

4 Opis algoritma

Program, koji simulira rešavanje zadatka pomoću Gausove metode eliminacije i Gaus-Zajdelove metode, radjen je na programskom jeziku C++. Program se sastoji iz nekoliko fajlova(Jednacina.h, Jednacina.cpp, SistemJednacina.h, SistemJednacina.cpp i Main.cpp). Klasa "Jednačina" ima promenljivu "koeficijenti_" što predstavlja vektor elemenata čiji članovi ce biti koeficijente date jednačine. Klasa "SistemJednacina" ima promenljivu "jednacine_"(predstavlja vektor čiji članovi ce biti objekti klase "Jednačina" i predstavljaće odredjenu vrstu u matrici sistema jednačine), "multiplikatori_"
(predstavlja vektor tipa "double" što će predstavljati multiplikatore i ovaj vektor će se prazniti pri svakom novom koraku pri odredjivanju nove matrice), "resenje_Gaus_", "resenje_Zajdel_"(vektore gde čuvamo rezultate programa).

4.1 Gausova metoda eliminacije

Formiramo proširenu matricu sistema jednacine:

$$\big[A, b\big] = \left[\begin{array}{ccccc|c} 9.5 & -2.5 & 0 & -2 & 0 & 12 \\ -2.5 & 11 & -3.5 & 0 & -5 & -16 \\ 0 & -3.5 & 15.5 & 0 & -4 & 14 \\ -2 & 0 & 0 & 7 & -3 & 10 \\ 0 & -5 & -4 & -3 & 12 & -30 \end{array}\right]$$

Za prvu kolonu biramo pivot tako što nalazimo po apsolutnoj vrednosti najveći element u prvoj koloni. Kako je najveći pivot u 1.vrsti, ne menjamo mesta vrstama. Deo programa koji radi pivotiranje vrste jednačina se zasniva na tome sto zamenjuje mesta promenljive tipa "Jednačina"u vektoru "jednačine"*itakosimulirame*

```
void SistemJednacina::pivotiranje(int broj_kolone){
  int maks = broj_kolone;
  for (int i = maks; i+1< 5; i++) {
  if (abs(jednacine_[maks]->koeficijenti_[broj_kolone]) < abs
   (jednacine_[i + 1]->koeficijenti_[broj_kolone])) {
    maks = i + 1;
  }
  }
  if (maks != broj_kolone) {
  Jednacina* jednacina = new Jednacina();
  jednacina->koeficijenti_.push_back(jednacine_[maks]->
   koeficijenti_[0]);
  jednacina->koeficijenti_.push_back(jednacine_[maks]->
   koeficijenti_[1]);
  jednacina->koeficijenti_.push_back(jednacine_[maks]->
   koeficijenti_[2]);
  jednacina->koeficijenti_.push_back(jednacine_[maks]->
   koeficijenti_[3]);
  jednacina->koeficijenti_.push_back(jednacine_[maks]->
   koeficijenti_[4]);
  jednacina->koeficijenti_.push_back(jednacine_[maks]->
   koeficijenti_[5]);

  jednacine_[maks]->koeficijenti_[0] = jednacine_[broj_kolone
   ]->koeficijenti_[0];
  jednacine_[maks]->koeficijenti_[1] = jednacine_[broj_kolone
   ]->koeficijenti_[1];
  jednacine_[maks]->koeficijenti_[2] = jednacine_[broj_kolone
   ]->koeficijenti_[2];
  jednacine_[maks]->koeficijenti_[3] = jednacine_[broj_kolone
   ]->koeficijenti_[3];
  jednacine_[maks]->koeficijenti_[4] = jednacine_[broj_kolone
   ]->koeficijenti_[4];
  jednacine_[maks]->koeficijenti_[5] = jednacine_[broj_kolone
   ]->koeficijenti_[5];

  jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
   koeficijenti_[0];
  jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
   koeficijenti_[1];
  jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
```

```
    koeficijenti_[2];
27  jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
    koeficijenti_[3];
28  jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
    koeficijenti_[4];
29  jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
    koeficijenti_[5];
30
31  }
32 }
```

Sledeće se nalaze modifikatori za k-tu vrstu (k=2,3,4,5):

$$m_{21} = a_{21}/a_{11} = -2.5/9.5 = -0.263158$$
$$m_{31} = a_{31}/a_{11} = 0/9.5 = 0$$
$$m_{41} = a_{41}/a_{11} = -2/9.5 = -0.210526$$
$$m_{51} = a_{51}/a_{11} = 0/9.5 = 0$$

Deo programa koji nalazi nove multiplikatore tako što u vektor tipa *double* dodaju elementi koje predstavljaju nove multiplikatore koji se dobijaju deljenjem odgovarajućih članova vektora koeficijenata(predstavlja kolonu) sa odgovarajućim članovima vektora jednacina(predstavlja vrstu):

```
1 void SistemJednacina::noviMultiplikatori(int broj_vrste){
2   int pocetni_red = broj_vrste - 1;
3   for (int i = broj_vrste; i < 5; i++) {
4     multiplikatori_.push_back(jednacine_[i]->koeficijenti_[
    broj_vrste-1]/ jednacine_[pocetni_red]->koeficijenti_[
    broj_vrste-1]);
5   }
6 }
```

Sada odgovarajući elementi odgovarajućih vrsta se oduzimaju sa odgovarajućim elementima prve vrste koja je prethodno pomnožena odgovarajućim multiplikatorom:

$$a_{ij}^2 = a_{ij} - m_{i1} \cdot a_{1j}$$
$$b_i^2 = b_i - m_{i1} \cdot b_1$$

Deo programa koji sredjuje matricu tako što menja članove vektora promenljive "koeficijenti_" za date clanove vektora promenljive "jednacine_":

```
1 void SistemJednacina::novaMatrica(int broj_vrste){
2   int pocetni_red = broj_vrste-1, k=0;
3   for (int i = broj_vrste; i < 5; i++) {
4     for (int j = broj_vrste-1; j < 6; j++) {
5       jednacine_[i]->koeficijenti_[j] = jednacine_[i]->
    koeficijenti_[j] - multiplikatori_[k] * jednacine_[
    pocetni_red]->koeficijenti_[j];
```

```
    }
    k++;
  }
}
```

Ponavljamo iste postupke(biranje pivota, zamena mesta vrste matrica, nalaženje multiplikatora i oduzimanje vrsta) na sistemu od 4 jednačine što je za jedan manje nego prethodni korak, i tako sve dok se ne formira gornja trougaona matrica čije se konačno rešenje dobija postupkom rešavanjem unazad:

$$x_n = \frac{b_n}{a_{nn}}$$

$$x_i = \frac{1}{a_{ii}}(b_i - \sum_{i+1<k<n} a_{ik} \cdot x_k)$$

Deo programa koji nalaze od poslednje date matrice konačna rešenja:

```
void SistemJednacina::nalazenjeResenja(){
  double sum = 0;

  for (int i = 4; i >= 0; i--) {
    sum = 0;
    if (i == 4) {
      resenje_Gaus_.push_back(jednacine_[i]->koeficijenti_[i
   +1] / jednacine_[i]->koeficijenti_[i]);
    }
    else {
      for (int j = i+1, k=1; j<5 ; j++,k++ ) {
        sum += (jednacine_[i]->koeficijenti_[j]*
   resenje_Gaus_[(int)(resenje_Gaus_.size())-k]);
      }
      resenje_Gaus_.push_back(1/(jednacine_[i]->koeficijenti_
   [i])*(jednacine_[i]->koeficijenti_[5] -sum));
    }
  }
}
}
```

$$\left[A^2, b^2\right] = \left[\begin{array}{ccccc|c} 9.5 & -2.5 & 0 & -2 & 0 & 12 \\ 0 & 10.3421 & -3.5 & -0.526316 & -5 & -12.8421 \\ 0 & -3.5 & 15.5 & 0 & -4 & 14 \\ 0 & -0.526316 & 0 & 6.57895 & -3 & 12.5263 \\ 0 & -5 & -4 & -3 & 12 & -30 \end{array}\right]$$

$$m_{32} = a_{32}/a_{22} = -0.338422$$

$$m_{42} = a_{42}/a_{22} = -0.0508906$$

$$m_{52} = a_{52}/a_{22} = -0.483461$$

$$\left[A^3, b^3\right] = \left[\begin{array}{ccccc|c} 9.5 & -2.5 & 0 & -2 & 0 & 12 \\ 0 & 10.3421 & -3.5 & -0.526316 & -5 & -12.8421 \\ 0 & 0 & 14.3155 & -0.178117 & -5.69211 & 9.65394 \\ 0 & 0 & -0.178117 & 6.55216 & -3.25445 & 11.8728 \\ 0 & 0 & -5.69211 & -3.25445 & 9.5827 & -36.2087 \end{array}\right]$$

$$m_{43} = a_{43}/a_{33} = -0.0124422$$

$$m_{53} = a_{53}/a_{33} = -0.397618$$

$$\left[A^4, b^4\right] = \left[\begin{array}{ccccc|c} 9.5 & -2.5 & 0 & -2 & 0 & 12 \\ 0 & 10.3421 & -3.5 & -0.526316 & -5 & -12.8421 \\ 0 & 0 & 14.3155 & -0.178117 & -5.69211 & 9.65394 \\ 0 & 0 & 0 & 6.54995 & -3.32528 & 11.9929 \\ 0 & 0 & 0 & -3.32528 & 7.31941 & -32.3701 \end{array}\right]$$

$$m_{54} = a_{54}/a_{44} = -0.50768$$

$$\left[A^5, b^5\right] = \left[\begin{array}{ccccc|c} 9.5 & -2.5 & 0 & -2 & 0 & 12 \\ 0 & 10.3421 & -3.5 & -0.526316 & -5 & -12.8421 \\ 0 & 0 & 14.3155 & -0.178117 & -5.69211 & 9.65394 \\ 0 & 0 & 0 & 6.54995 & -3.32528 & 11.9929 \\ 0 & 0 & 0 & 0 & 5.63123 & -26.2815 \end{array}\right]$$

$$x_5 = b_5/a_{55} = -2.5/9.5 = -4.6671$$

$$x_4 = \frac{1}{a_{44}}(b_4 - a_{45} \cdot x_5) = -0.5384$$

$$x_3 = \frac{1}{a_{33}}(b_3 - a_{34} \cdot x_4 - a_{35} \cdot x_5) = -1.18805$$

$$x_2 = \frac{1}{a_{22}}(b_2 - a_{23} \cdot x_3 - a_{24} \cdot x_4 - a_{25} \cdot x_5) = -3.92755$$

$$x_1 = \frac{1}{a_{11}}(b_1 - a_{12} \cdot x_2 - a_{13} \cdot x_3 - a_{14} \cdot x_4 - a_{15} \cdot x_5) = 0.116244$$

4.2 Gaus-Zajdelova metoda

Od polaznih jednačina:

$$\begin{aligned} 9.5i1 - 2.5i2 - 2i4 &= 12 \\ -2.5i1 + 11i2 - 3.5i3 - 5i5 &= -16 \\ -3.5i2 + 15.5i3 - 4i5 &= 14 \\ -2i1 + 7i4 - 3i5 &= 10 \\ -5i2 - 4i3 - 3i4 + 12i5 &= -30, \end{aligned}$$

potrebno je iz i-te jednačine izraziti promenljivu x_i i formiramo Gaus-Zajdelov iterativni proces:

$$x_1^{(k+1)} = \frac{2.5}{9.5} \cdot x_2^{(k)} + \frac{2}{9.5} \cdot x_4^{(k)} + \frac{12}{9.5}$$

$$x_2^{(k+1)} = \frac{2.5}{11} \cdot x_1^{(k+1)} + \frac{3.5}{11} \cdot x_3^{(k)} + \frac{5}{11} \cdot x_5^{(k)} - \frac{16}{11}$$

$$x_3^{(k+1)} = \frac{3.5}{15.5} \cdot x_2^{(k+1)} + \frac{4}{15.5} \cdot x_5^{(k)} + \frac{14}{9.5}$$

$$x_4^{(k+1)} = \frac{2}{7} \cdot x_1^{(k+1)} + \frac{3}{7} \cdot x_5^{(k)} + \frac{10}{7}$$

$$x_5^{(k+1)} = \frac{5}{12} \cdot x_2^{(k+1)} + \frac{4}{12} \cdot x_3^{(k+1)} + \frac{3}{12} \cdot x_4^{(k+1)} - \frac{30}{12}$$

Početna vrednost je $x^0 = [0, 0, 0, 0]^T$ i stajemo sa iteracijom kada se dve susedne iteracije poklope sa 4 decimala.

Za ovu metodu je bilo potrebno u programu pripremiti sistem jednačina kako bi se primenile iteracije. Deo programa koji priprema vektore tipa "Jednačina" :

```
1  void SistemJednacina::noviSistemJednacina(){
2    for (int i = 0; i < 5; i++) {
3      for (int j = 0; j < 6; j++) {
4        if (i != j) {
5          if (j == 5) {
6            jednacine_[i]->koeficijenti_[j] = jednacine_[i]->
     koeficijenti_[j] / jednacine_[i]->koeficijenti_[i];
7          }
8          else {
9            jednacine_[i]->koeficijenti_[j] = (jednacine_[i]->
     koeficijenti_[j] / jednacine_[i]->koeficijenti_[i])*(-1);
10         }
11       }
12     }
```

```
    jednacine_[i]->koeficijenti_[i] = 0;
  }
}
```

Deo programa koji nalazi rešenja i puni vektor "resenje_Zajdel_":

```
void SistemJednacina::nalazenjeResenja2(){
  bool radi = true;
  double x1, x2, x3, x4, x5;

  while (radi) {
    for (double R : resenje_Zajdel_) {
      cout << R << "  ";
    }
    cout << endl;
    x1 = resenje_Zajdel_[1];
    x2 = resenje_Zajdel_[2];
    x3 = resenje_Zajdel_[3];
    x4 = resenje_Zajdel_[4];
    x5 = resenje_Zajdel_[5];

    for (int i = 1; i <= 5; i++) {
      resenje_Zajdel_[i] = jednacine_[i-1]->koeficijenti_[0]
   * resenje_Zajdel_[1] + jednacine_[i - 1]->koeficijenti_[1]
    * resenje_Zajdel_[2] + jednacine_[i - 1]->koeficijenti_
   [2] * resenje_Zajdel_[3] + jednacine_[i - 1]->
   koeficijenti_[3] * resenje_Zajdel_[4] + jednacine_[i -
   1]->koeficijenti_[4] * resenje_Zajdel_[5] + jednacine_[i -
    1]->koeficijenti_[5];
    }
    //if (x1 == resenje_Zajdel_[1] && x2 == resenje_Zajdel_
   [2] && x3 == resenje_Zajdel_[3] && x4 == resenje_Zajdel_
   [4] && x5 == resenje_Zajdel_[5]) {
    //  radi = false;
    //}

    if (uporedi(resenje_Zajdel_[1], x1) && uporedi(
   resenje_Zajdel_[2], x2) && uporedi(resenje_Zajdel_[3], x3)
    && uporedi(resenje_Zajdel_[4], x4) && uporedi(
   resenje_Zajdel_[5], x5)) {
      radi = false;
    }

    resenje_Zajdel_[0]++;
    //for (double R : resenje_Zajdel_) {
    //  cout << R << "  ";
    //}
    //cout << endl;

  }
```

```
}

bool SistemJednacina::uporedi(double trenutni_rezultat,
    double prethodni_rezultat, double razlika){
  if (fabs(trenutni_rezultat - prethodni_rezultat) < razlika)
    {
    return true;
  }
  return false;
}
```

Tabela Iteracije:

k	x_1	x_2	x_3	x_4	x_5
0	0	0	0	0	0
1	1.26316	-1.16746	0.639605	1.78947	-2.32587
2	0.33266	-2.00537	-0.149826	0.812529	-3.18238
3	0.906487	-2.74273	-0.537361	0.32369	-3.741
4	0.609531	-3.18745	-0.781941	-0.000564235	-4.08889
5	0.424236	-3.46552	-0.934508	-0.202601	-4.30612
6	0.308527	-3.6391	-1.02976	-0.328757	-4.44173
7	0.236289	-3.74746	-1.08923	-0.407517	-4.5264
8	0.19119	-3.81512	-1.12636	-0.456688	-4.57926
9	0.163034	-3.85736	-1.14953	-0.487368	-4.61226
10	0.145456	-3.88373	-1.16401	-0.506552	-4.63286
11	0.134481	-3.90019	-1.17304	-0.518517	-4.64572
12	0.127630	-3.91047	-1.17868	-0.525987	-4.65375
13	0.123353	-3.91689	-1.1822	-0.53065	-4.65877
14	0.120682	-3.92089	-1.1844	-0.533562	-4.6619
15	0.119015	-3.92339	-1.18577	-0.53538	-4.66385
16	0.117974	-3.92496	-1.18663	-0.536515	-4.66507
17	0.117324	-3.92593	-1.18716	-0.537223	-4.66583
18	0.116919	-3.92654	-1.1875	-0.537665	-4.66631
19	0.116665	-3.92692	-1.18771	-0.537941	-4.6666
20	0.116507	-3.92716	-1.18784	-0.538114	-4.66679
21	0.116408	-3.9273	-1.18792	-0.538222	-4.6669
22	0.116347	-3.9274	-1.18797	-0.538289	-4.66698
23	0.116308	-3.92745	-1.188	-0.538331	-4.66702
24	0.116284	-3.92749	-1.18802	-0.538357	-4.66705
25	0.116269	-3.92751	-1.18803	-0.538373	-4.66707
26	0.11626	-3.92753	-1.18804	-0.538383	-4.66708

$$x_1 = 0.1162$$
$$x_2 = -3.9275$$
$$x_3 = -1.1880$$
$$x_4 = -0.5383$$
$$x_5 = -4.6670$$

5 Komentari

Koeficijenti sistema ne ispunjavaju uslov stroge dijagonalne dominatnosti:

$$\mid a_{ii} \mid > \sum_{1<j<n, j\neq i} \mid a_{ij} \mid,$$

tačnije u poslednjoj vrsti matrice sistema dijagonalni element jednak je zbiru preostalih elemenata te vrste ($\mid 12 \mid = \mid -5 \mid + \mid -4 \mid + \mid -3 \mid$). Medjutim, uslov strogo dijagonalne dominantnosti nije potreban uslov da bi Gaus-Zajdelova metoda konvergirala. Uslov strogo dijagonalne dominantnosti obezbedjuje pre svega da pri preformulaciji jednačina dijagonalni elementi kojima delimo budu različiti od 0 što u ovom slučaju ne moramo da se brinemo tj. nema potrebe za elementarnim transformacija sistema jednačina. Broj iterativnih koraka će biti manje ukoliko su elementi na dijagonali izrazito dominatniji tj. konvergencija će biti brža(u ovom slučaju je spora konvergencija jer dijagonalni elementi nisu čak ni striktno dominantni).

6 Dodatak

JEDNAČINA.H

```
#ifndef JEDNACINA_H
#define JEDNACINA_H

#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Jednacina {
```

```
public:
  Jednacina(const string& ime = "jednacina");

  void citajJednacinu(string unos);
  void pisi() const;
  vector<double> koeficijenti_;

private:
  string ime_;
};

#endif
#pragma once
```

JEDNAČINA.CPP

```
#include "Jednacina.h"

Jednacina::Jednacina(const string& ime) : ime_(ime) {}

void Jednacina::citajJednacinu(string unos){
  int poz = 0;
  int promenljiva = 1;
  string broj = "";
  float i;

  while (poz < unos.length()) {
    while (((unos[poz] >= '0' && unos[poz] <= '9') || unos[poz] == '.' || unos[poz] == '-') && poz < unos.length()) {
      broj += unos[poz];
      poz++;
    }
    if (poz >= unos.length()) {
      while (promenljiva <= 5) {
        koeficijenti_.push_back(0);
        promenljiva++;
      }
      koeficijenti_.push_back(stod(broj));
      break;
    }
    if (unos[poz+1] == promenljiva) {
      koeficijenti_.push_back(stod(broj));
      promenljiva++;
    }
    else {
      while ((int)(unos[poz + 1]-'0') > promenljiva) {
        koeficijenti_.push_back(0);
        promenljiva++;
      }
      koeficijenti_.push_back(stod(broj));
```

```
      promenljiva++;
    }
    poz += 3;
    broj = "";
    if(unos[poz-1] == '-'){ broj += '-'; }
  }
}

void Jednacina::pisi() const{
  for (float koeficijent : koeficijenti_) {
    cout << koeficijent << "    ";
  }
  cout << endl;
  cout << endl;
  cout << endl;
}
```

SISTEMJEDNAČINA.H

```
#ifndef SISTEMJEDNACINA_H
#define SISTEMJEDNACINA_H

#include "Jednacina.h"

#include <cmath>
#include <vector>
using namespace std;

class SistemJednacina {
public:
  SistemJednacina(const string& ime = "sistem jednacina");

  void citajJednacinu(string unos);
  void pisi() const;

  void pivotiranje(int);
  void noviMultiplikatori(int);
  void gausMetoda();
  void novaMatrica(int);
  void nalazenjeResenja();

  void gausZajdelovaMetoda();
  void noviSistemJednacina();
  void nalazenjeResenja2();
  bool uporedi(double trenutni_rezultat, double
   prethodni_rezultat, double razlika=0.00001f);

private:
  vector<Jednacina*> jednacine_;
  string ime_;
```

```
  vector<double> multiplikatori_;
  vector<double> resenje_Gaus_;
  vector<double> resenje_Zajdel_;
};

#endif
```

SISTEMJEDNAČINA.CPP

```
#include "SistemJednacina.h"

SistemJednacina::SistemJednacina(const string& ime) : ime_(ime) {
  for (int i = 0; i < 6; i++) {
    resenje_Zajdel_.push_back(0);
  }
}

void SistemJednacina::citajJednacinu(string unos){
  Jednacina* jednacina = new Jednacina();

  jednacina->citajJednacinu(unos);
  jednacine_.push_back(jednacina);

}

void SistemJednacina::pisi() const{
  for (Jednacina* jednacina : jednacine_) {
    jednacina->pisi();
  }
}

void SistemJednacina::pivotiranje(int broj_kolone){
  int maks = broj_kolone;
  for (int i = maks; i+1< 5; i++) {
    if (abs(jednacine_[maks]->koeficijenti_[broj_kolone]) < abs(jednacine_[i + 1]->koeficijenti_[broj_kolone])) {
      maks = i + 1;
    }
  }
  if (maks != broj_kolone) {
    Jednacina* jednacina = new Jednacina();
    jednacina->koeficijenti_.push_back(jednacine_[maks]->koeficijenti_[0]);
    jednacina->koeficijenti_.push_back(jednacine_[maks]->koeficijenti_[1]);
    jednacina->koeficijenti_.push_back(jednacine_[maks]->koeficijenti_[2]);
    jednacina->koeficijenti_.push_back(jednacine_[maks]->koeficijenti_[3]);
```

```
36      jednacina->koeficijenti_.push_back(jednacine_[maks]->
        koeficijenti_[4]);
37      jednacina->koeficijenti_.push_back(jednacine_[maks]->
        koeficijenti_[5]);
38
39      jednacine_[maks]->koeficijenti_[0] = jednacine_[
        broj_kolone]->koeficijenti_[0];
40      jednacine_[maks]->koeficijenti_[1] = jednacine_[
        broj_kolone]->koeficijenti_[1];
41      jednacine_[maks]->koeficijenti_[2] = jednacine_[
        broj_kolone]->koeficijenti_[2];
42      jednacine_[maks]->koeficijenti_[3] = jednacine_[
        broj_kolone]->koeficijenti_[3];
43      jednacine_[maks]->koeficijenti_[4] = jednacine_[
        broj_kolone]->koeficijenti_[4];
44      jednacine_[maks]->koeficijenti_[5] = jednacine_[
        broj_kolone]->koeficijenti_[5];
45
46      jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
        koeficijenti_[0];
47      jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
        koeficijenti_[1];
48      jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
        koeficijenti_[2];
49      jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
        koeficijenti_[3];
50      jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
        koeficijenti_[4];
51      jednacine_[broj_kolone]->koeficijenti_[0] = jednacina->
        koeficijenti_[5];
52
53    }
54  }
55
56  void SistemJednacina::noviMultiplikatori(int broj_vrste){
57    int pocetni_red = broj_vrste - 1;
58    for (int i = broj_vrste; i < 5; i++) {
59      multiplikatori_.push_back(jednacine_[i]->koeficijenti_[
        broj_vrste-1]/ jednacine_[pocetni_red]->koeficijenti_[
        broj_vrste-1]);
60    }
61  }
62
63  void SistemJednacina::gausMetoda(){
64    int i = 5;
65    /*pivotiranje(0);
66    noviMultiplikatori(1);
67    novaMatrica(1);
68    multiplikatori_.clear();
```

```
69    pisi();
70
71    pivotiranje(1);
72    noviMultiplikatori(2);
73    novaMatrica(2);
74    multiplikatori_.clear();
75    pisi();
76
77    pivotiranje(2);
78    noviMultiplikatori(3);
79    novaMatrica(3);
80    multiplikatori_.clear();
81    pisi();
82
83    pivotiranje(3);
84    noviMultiplikatori(4);
85    novaMatrica(4);
86    multiplikatori_.clear();
87    pisi();*/
88    for (int i = 0; i < 4; i++) {
89      pivotiranje(i);
90      noviMultiplikatori(i+1);
91      novaMatrica(i + 1);
92      cout << endl;
93      cout << i+1 << ".korak";
94      cout << "
       ========================================================" <<
       endl;
95      cout << "multiplikatori: ";
96      for (double mp : multiplikatori_) {
97        cout << mp << "   ";
98      }
99      cout << endl;
100     cout << "
       ========================================================" <<
       endl;
101     multiplikatori_.clear();
102     cout << "
       ========================================================" <<
       endl;
103     pisi();
104     cout << "
       ========================================================" <<
       endl;
105   }
106   nalazenjeResenja();
107
108   cout << "Resenje sistema jednacina su: ";
109   for (double R : resenje_Gaus_) {
```

```
110     cout << "x" << i-- << ":" << R << "   ";
111   }
112   cout << endl;
113 
114 }
115 
116 void SistemJednacina::gausZajdelovaMetoda(){
117   noviSistemJednacina();
118   nalazenjeResenja2();
119 
120 }
121 
122 void SistemJednacina::noviSistemJednacina(){
123   for (int i = 0; i < 5; i++) {
124     for (int j = 0; j < 6; j++) {
125       if (i != j) {
126         if (j == 5) {
127           jednacine_[i]->koeficijenti_[j] = jednacine_[i]->
    koeficijenti_[j] / jednacine_[i]->koeficijenti_[i];
128         }
129         else {
130           jednacine_[i]->koeficijenti_[j] = (jednacine_[i]->
    koeficijenti_[j] / jednacine_[i]->koeficijenti_[i])*(-1);
131         }
132       }
133     }
134     jednacine_[i]->koeficijenti_[i] = 0;
135   }
136 }
137 
138 void SistemJednacina::nalazenjeResenja2(){
139   bool radi = true;
140   double x1, x2, x3, x4, x5;
141 
142   while (radi) {
143     for (double R : resenje_Zajdel_) {
144       cout << R << "   ";
145     }
146     cout << endl;
147     x1 = resenje_Zajdel_[1];
148     x2 = resenje_Zajdel_[2];
149     x3 = resenje_Zajdel_[3];
150     x4 = resenje_Zajdel_[4];
151     x5 = resenje_Zajdel_[5];
152 
153     for (int i = 1; i <= 5; i++) {
154       resenje_Zajdel_[i] = jednacine_[i-1]->koeficijenti_[0]
    * resenje_Zajdel_[1] + jednacine_[i - 1]->koeficijenti_[1]
     * resenje_Zajdel_[2] + jednacine_[i - 1]->koeficijenti_
```

```
    [2] * resenje_Zajdel_[3] + jednacine_[i - 1]->
    koeficijenti_[3] * resenje_Zajdel_[4] + jednacine_[i -
    1]->koeficijenti_[4] * resenje_Zajdel_[5] + jednacine_[i -
     1]->koeficijenti_[5];
155     }
156     //if (x1 == resenje_Zajdel_[1] && x2 == resenje_Zajdel_
    [2] && x3 == resenje_Zajdel_[3] && x4 == resenje_Zajdel_
    [4] && x5 == resenje_Zajdel_[5]) {
157     //  radi = false;
158     //}
159
160     if (uporedi(resenje_Zajdel_[1], x1) && uporedi(
    resenje_Zajdel_[2], x2) && uporedi(resenje_Zajdel_[3], x3)
     && uporedi(resenje_Zajdel_[4], x4) && uporedi(
    resenje_Zajdel_[5], x5)) {
161       radi = false;
162     }
163
164     resenje_Zajdel_[0]++;
165     //for (double R : resenje_Zajdel_) {
166     //  cout << R << "  ";
167     //}
168     //cout << endl;
169
170   }
171 }
172
173 bool SistemJednacina::uporedi(double trenutni_rezultat,
     double prethodni_rezultat, double razlika){
174   if (fabs(trenutni_rezultat - prethodni_rezultat) < razlika)
      {
175     return true;
176   }
177   return false;
178 }
179
180 void SistemJednacina::novaMatrica(int broj_vrste){
181   int pocetni_red = broj_vrste-1, k=0;
182   for (int i = broj_vrste; i < 5; i++) {
183     for (int j = broj_vrste-1; j < 6; j++) {
184       jednacine_[i]->koeficijenti_[j] = jednacine_[i]->
    koeficijenti_[j] - multiplikatori_[k] * jednacine_[
    pocetni_red]->koeficijenti_[j];
185     }
186     k++;
187   }
188 }
189
190 void SistemJednacina::nalazenjeResenja(){
```

```
  double sum = 0;

  for (int i = 4; i >= 0; i--) {
    sum = 0;
    if (i == 4) {
      resenje_Gaus_.push_back(jednacine_[i]->koeficijenti_[i+1] / jednacine_[i]->koeficijenti_[i]);
    }
    else {
      for (int j = i+1, k=1; j<5 ; j++,k++ ) {
        sum += (jednacine_[i]->koeficijenti_[j]*resenje_Gaus_[(int)(resenje_Gaus_.size())-k]);
      }
      resenje_Gaus_.push_back(1/(jednacine_[i]->koeficijenti_[i])*(jednacine_[i]->koeficijenti_[5] -sum));
    }
  }
}
```

MAIN.CPP

```
#include "SistemJednacina.h"

#include "string"
using namespace std;

int main() {
  string nova_jednacina;
  int izbor;
  bool radi = true;

  SistemJednacina* sistem_jednacina = new SistemJednacina();

  while (radi) {
    cout << "Odaberite opciju:\n1. Unos nove jednacine\n2. Kraj unosa " << endl;
    cin >> izbor;

    try {
      switch (izbor) {
      case 1: {  //Ucitavanje nove jednacine
        string putanja;

        //cout << izbor << "\nNova jednacina => " << endl;
    ////////////////////////////
        //cin >> nova_jednacina;
    ////////////////////////////

        //sistem_jednacina->citajJednacinu(nova_jednacina);
    ;//////////////////////////////////////
```

```
26          sistem_jednacina->citajJednacinu("9.5i1-2.5i2-2i4=12"
    );
27          sistem_jednacina->citajJednacinu("-2.5i1+11i2-3.5i3-5
    i5=-16");
28          sistem_jednacina->citajJednacinu("-3.5i2+15.5i3-4i5
    =14");
29          sistem_jednacina->citajJednacinu("-2i1+7i4-3i5=10");
30          sistem_jednacina->citajJednacinu("-5i2-4i3-3i4+12i5
    =-30");
31          sistem_jednacina->pisi();
32          cout << "\nZavrsen unos " << endl;
33
34          break;
35        }
36        case 2: {  //Kraj unosa
37          radi = false;
38          break;
39        }
40        default:
41          cout << "Netacan unos\n\n";
42        }
43     }
44     catch (string izuzetak) {
45       //prihvatiGresku(izuzetak);
46     }
47   }
48
49   if (izbor) { radi = true; }
50
51   while (radi) {
52     cout << "
     ========================================================"
     << endl;
53     cout << "Molimo Vas, odaberite nacin resavanja zadatka:\
     n1. Gausova metoda eliminacije sa izborom pivota\n2. Gaus-
     Zajdenlova metoda\n3. Prikaz prosirene matrice sistema
     jednacine\n4. Kraj programa\n " << endl;
54     cout << "
     ========================================================"
     << endl << endl;
55     cin >> izbor;  //izbor opcija
56
57     switch (izbor) {
58
59     case 1: {   //Gausova metoda eliminacije sa izborom
     pivota
60       sistem_jednacina->gausMetoda();
61       //sistem_jednacina->pisi();
62       break;
```

```
63      }
64      case 2: {   //Gaus-Zajdenlova metoda
65        sistem_jednacina->gausZajdelovaMetoda();
66        break;
67      }
68      case 3: {   //Prikaz proserene matrice koeficijenata
69        sistem_jednacina->pisi();
70        break;
71      }
72      case 4: {   //Kraj programa
73        cout << "Kraj programa" << endl;
74        radi = false;
75        break;
76      }
77      default:
78        cout << "Netacan unos\n\n";
79      }
80    }
81
82    delete sistem_jednacina;
83
84    return 0;
85  }
```