Kako rešiti sistem linearnih jednačina sa ekstremno mnogo nepoznatih

Aleksa Srdanov, Radiša Stefanović
Visoka tehnička škola strukovnih studija
Požarevac, Srbija
aleksa.srdanov@vts-pozarevac.edu.rs, radisastefanovic@yahoo.com

Sažetak— **Rešavanje linearnog sistema jednačina n x n može biti problem i za računar, čak i kada je broj jednačina i nepoznatih relativno mali (par stotina). Sve postojeće metode su opterećene barem jednim od sledećih problema: 1. složenošću računanja izraženim kroz broj potrebnih operacija koje je potrebno izvršiti da bi se došlo do rešenja; 2. neograničenim rastom veličina međurezultata što uzrokuje overflow i underflow probleme; 3. promenom vrednosti nekih koeficijenata u polaznom sistemu što uzrokuje nestabilnost rešenja; 4. zahtevaju određene uslove za konvergenciju itd. U ovom radu se prezentuje približna metoda za rešavanje sistema linearnih jednačina sa proizvoljnim brojem jednačina i nepoznatih. Predloženom metodom se mogu izbeći svi pomenuti problemi.**

Ključne riječi-ekstremno mnogo nepoznatih;

I. UVOD

Pretpostavimo da je dat linearni sistem osam jednačina sa osam nepoznatih koji želimo rešiti Kramerovom metodom uz pomoć papira i olovke. Za nalaženje osam nepoznatih vrednosti potrebno je prethodno sračunati vrednosti devet determinatni reda osam. Sračunavanje jedne determinante reda osam zahteva 322560 binarnih operacija. Ceo postupak zahteva tek nešto manje od tri miliona operacija. Ako bismo svaku operaciju izvršavali u jednoj sekundi onda bi za rešavanje celog sistema bilo potrebno oko tri miliona sekundi. Prevedeno u dane, to je oko 57 dana, pri čemu se u svakoj sekundi mora odraditi jedna operacija, bez pauze za bilo šta drugo. Tako nešto je nemoguće i zbog toga što samo zapis velikih brojeva zahteva više sekundi. Osim navedenog vremenskog ograničenja nije realno očekivati da čovek, bez pomoći računara, taj posao sam odradi ne napravivši ni jednu jedinu grešku.

Primena matričnih metoda je po zahtevnosti jednako zahtevna kao i determinanti. Osim toga, obe pomenute metode i za pedesetak nepoznatih, čak i za današnje računare predstavljaju neprihvatljive postupke za primenu, zbog neočekivano velikog broja operacija koje su potrebne za primenu.

Gausovom metodom čovek bi sistem 8 x 8 mogao rešiti i uz pomoć papira i olovke. Za rešavanje takvog sistema broj operacija koje je potrebno izvršiti približno je hiljadu puta manji od onog koji je potreban za Kramerovu metodu. Ako broj nepoznatih poraste na pedeset i primena Gausove metode nije čoveku praktično moguća. Broj potrebnih operacija za primenu Gausove metode je značajno manji u odnosu na spomenute ali primena ove metode krije niz drugih neočekivanih problema. Jedan od njih je taj što čak i za relativno mali broj jednačina i nepoznatih međurezultati mogu postati neočekivano veliki. Drugi je, što Gausova metoda menja vrednosti početno datih koeficijenata što je apsolutno nedopustivo s obzirom na izrazitu nestabilnost rešenja sitema linearnih jednačina (male promene vrednosti koeficijenata mogu rezultovati velikim promenama vrednosti rešenja).

Ako matematiku shvatamo kao nauku usmerenu pre svega ka čoveku kao subjektu njene primene onda postavljeni problem u naslovu nije matematički. On je u svojoj suštini isključivo računarski jer i sama postavka sistema je apsolutno nemoguća bez pomoći računara. Samo za zapis hiljadu jednačina sa hiljadu nepoznatih čoveku bi bio potreban ceo život.

Zato bi od velike koristi bilo ako bismo mogli definisati metodu koja: 1) Zahteva što je moguće manji broj potrebnih operacija za izvršenje; 2) Ne proizvodi neočekivane overflow i underflow efekte; 3) Ne menja koeficijente polaznog sistema; 4) Primenljiva je za proizvoljano veliki sistem jednačina i 5) Bezuslovno i relativno brzo konvergira iz proizvoljne početne tačke.

II. POSTAVKA PROBLEMA

Pretpostavimo da je u memoriji računara smešten sistem od n jednačina sa n nepoznatih (n mnogo veće od 100). Neka je oblika:

$$\begin{aligned} &a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1 \\ &a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = b_2 \\ &\quad\quad . \quad . \quad . \\ &a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = b_n \end{aligned} \tag{1}$$

Na primer, za $n = 10000$, sistem (1) može isključivo računar formirati unutar svoje memorije. (Ako bismo pretpostavili da čovek svake sekunde ispisuje samo po jedan koeficijent uz odgovarajuću nepoznatu, za zapis prve jednačine, bilo bi potrebno deset hiljada sekundi, što je nešto malo manje od tri minuta. Ceo sistem bi samo za zapis, pod datim pretpostavkama, zahtevao nešto više od tri godine!) Moguće je definisati postupak za rešavanje posmatranog sistema koji ima minimalna, skoro trivajalna računanja, a potom, ako rešenje postoji, obavezno konvergira ka njemu.

Za sistem (1) jedina pretpostavka je da ima rešenje i da prilikom provere tog rešenja u bilo kojoj njegovoj jednačini ne dolazi do prekoračenja u međurezultatima, jer ako bi se to dešavalo onda je sistem nemoguće rešiti u takvom okruženju.

Za ovakav problem autorima nije poznato da li matematika ima postupak kojim bi se on rešio. Za ovaj rad je bitno da kibernetički pristup ovakav problem rešava čak elementarno jednostavno. Kibernetski pristup ne zahteva rešavanje sistema jednačina da bi se došlo do rešenja.

III. OPIS POSTUPKA

Svaka jednačina u zadatom sistemu nosi veliki broj informacija. Jedna od njih je da je svaka jednačina, geometrijski gledano, jedna hiperravan unutar prostora čija je dimenzija barem koliko ima i nepoznatih. Neke od tih informacija je moguće iskoristiti u postupku nalaženja rešenja. Postupak se slikovito može opisati na sledeći način: Ka rešenju možemo poći od bilo koje tačke, najpogodnije je da to bude (0,0, …,0). Posmatrajmo samo prvu jednačinu odvojenu od ostalih. Ako rešenje sistema postoji onda je rešenje neka od tačaka iz te hiperravni. Ne može se znati koja je to tačka, zato za prvu aproksimaciju možemo uzeti bilo koju tačku, a pogodno je da to bude podnožje normale iz polazne tačke u prvu hiperravan. Pređimo sada na drugu jednačinu. Ako rešenje sistema postoji ono je u preseku te dve hiperravni. Međutim, nije neophodno računati taj presek. Dovoljno je iz dosegnute tačke ponovo se normalom spustiti na sledeću hiperravan u njeno podnožje. Opšte poznata činjenica je: „Kateta je kraća od hipotenuze“. Zato se spuštanjem duž normale približavamo ka rešenju, pod uslovom da ono postoji. Taj postupak treba nastaviti i sa preostalim jednačinama (hiperravnima) na analogan način, svaki put se iz dosegnutog podnožja treba spustiti normalom u sledeću hiperravan. Svaka jednačina (hiperravan) u sebi nosi tačnu informaciju o vektoru svoje normale. Za zadati sistem, koordinate vektora normala svih hiperavni su dati koeficijentima sistema (1): $n_i = \left(a_{i,1}, a_{i,2}, ..., a_{i,n}\right) \;\; i = 1, 2, ..., n$.

Postupak nalaženja preseka normale iz neke tačke sa nekom hiperravni sadrži minimalno računanje i dato je na sledeći način. Neka niz tačaka $\left(y_{1,j}^k, y_{2,j}^k, ..., y_{n,j}^k\right) \quad j = 1, 2, ..., n,$ predstavlja međukorake istog iteracionog koraka $k = 0, 1, 2, ...$ Kada napravimo pun krug od 1 do n sa indeksom j počinje sledeća iteracija po indeksu k. Izračunavanje koordinata sledeće tačke unutar iste iteracije izvodi se na sledeći način. Neka se u k–toj iteraciji stiglo u j-ti međukorak u tačku $\left(y_{1,j}^k,\ y_{2,j}^k, ...,\ y_{n,j}^k\right)$ Za prelazak u sledeću tačku $\left(y_{1,j+1}^k,\ y_{2,j+1}^k,\ ...,\ y_{n,j+1}^k\right)$, $0 \le j \le n-1$, potrebno je prvo izračunati broj $t = \dfrac{b_j - \left(a_{j,1} \cdot y_{1,j}^k + a_{j,2} \cdot y_{2,j}^k ... + a_{j,n} \cdot y_{n,j}^k\right)}{\left|n_j\right|^2}$.

Potom se sve koordinate sledećeg međukoraka računaju formulama: $y_{m,j+1}^k = a_{m,j} \cdot t + y_{m,j}^k$, za sve $m = 1, 2, ..., n$.

Kada se završi prolazak kroz sve jednačine završava se jedan iterativni korak. Sledeći iterativni korak započinje ponovnim spuštanjem normalom iz dosegnute tačke poslednje hiperravni u prvu. Posle punog kruga i kompletno završene jedne iteracije proverava se da li je dosegnuta tražena tačnost. To se može postići tako što se proverava da li je na svakoj koordinati rastojanje dve uzastopne iteracije manje od zadate tačnosti ili zbirno sva rastojanja manja od date tačnosti, ako nije postupak se nastavlja.

IV. NEMOGUĆ I NEODREĐEN SISTEM

Ako zadati sistem nema rešenje (nemoguć) opisani postupak to može detektovati. U tu svrhu je potrebno pamtiti kompletne korake u dve uzastopne iteracije. Geometrijski gledano, rešenje ne postoji ako su neke dve hiperravni paralelne. U predloženom postupku to se može zaključiti ako se na nekoj poziciji na dvema susednim poluiteracijama u dve uzastopne iteracije pojavljuju neke dve koordinate kao jednako udaljene (to je slučaj kada postoje dve susedne paralelne hiperravni). Tu se nije izvršilo približavanje i nikada neće, jer je u tom slučaju kateta jednaka hipotenuzi. S obzirom na postupak rešavanja čim se tako nešto utvrdi postupak treba prekinuti. I kada paralene hiperravni nisu susedne može se utvrditi da je sistem nemoguć na sledeći način. Izračuna se odstojanje od prve aproksimacije do svih ostalih hiperravni, a potom se to ponovi kada se punim ciklusom stigne u tačku prve hiperravni. Potom se uporede svaka dva odgovarajuća odstojanja. Kada se ustanovi jednakost na nekoj poziciji to je zato što su upravo te dve hiperravni međusobno paralelne. Za velike sisteme ovakva provera nije postupak koji bi se mogao specificirati kao jednostavan. Za njegovu implementaciju unutar opšteg algoritma značajno bi se povećao broj operacija koje treba primeniti. Takav postupak može da se primeni samo u slučaju kada se posle određenog broja iteracija nije postigla očekivana tačnost, pa se želi ustanoviti razlog.

Ako je sistem neodređen onda postoji gradacija neodređenosti sa stepenima slobode. U posmatranom primeru sistem može biti stepena slobode od 1 do n-1. Ako je stepena slobode jedan, sve hiperravni imaju zajedničku pravu; Ako je stepen slobode dva, sve hiperravni imaju zajedničku ravan; Ako je stepen slobode tri, sve hiperravni imaju zajednički trodimenzionalni prostor unutar n-dimenzionog posmatranog prostora itd; Ako je stepen slobode n-1 onda sve jednačine sistema predstavljaju jednu istu hiperravan unutar prostora dimenzije n.

U predloženom algoritmu najjednostavnije je samo konstatovati da je sistem neodređen jer bi više od toga zahtevalo mnogo više provera, što za ovaj rad nije od velikog značaja. Da bi se konstatovalo da je sistem neodređen, stepena slobode n-1, dovoljno je da se posle prve interacije utvrdi da se ostalo u istoj polaznoj tački prve hiperravni. Da bi se konstatovalo da je sistem neodređen, nekog manjeg stepena slobode, neophodno je ceo opisan postupak ponavljati iz različitih početnih tačaka. Kada je sistem neodređen, polaskom iz različitih tačaka, dobijaće se rešenja ali ona tada neće biti jednaka.

U opštem slučaju, potrebno je pamtiti barem tri uzastopne iteracije. Kada se utvrdi da se u tri susedne iteracije u odgovarajućim dvama susednim iteracijama pojavljuju barem dve odgovarajuće poluiteracije sa vrednostima (na istim koordinatama) koje se ponašaju homotetijski, tada postupak treba prekinuti jer se neće doći do rešenja čak i da se zadovolji tačnost, jer će se u tom slučaju dobiti samo jedno od

beskonačno mnogo mogućih rešenja. To treba proveriti za svaku koordinatu ali i za par, trojku, ... odgovarajućih vrednosti koordinata u svake tri uzastopne iteracije. Kada se utvrdi da postoji homotetija unutar nekih *m* koordinata tada možemo konstatovati da je sistem neodređen stepena *m*.

Ako sve hiperravni imaju zajedničku pravu, tada se iteracijama spiralno spuštamo ka rešenju. To se može primetiti u tri uzastopne iteracije ako se sve odgovarajuće koordinate dobijenih tačaka homotetijski razlikuju za istu veličinu. Ako sistem ima stepen slobode dva tada postoje dve homotetije sa istom osobinom koja se može uočiti u četiri različite uzastopne iteracije itd.

V. KONVERGENCIJA I BRZINA

Strog matematički dokaz bi zahtevao puno prostora, a sam po sebi je jedan kompletan rad. U ovom radu predstavićemo skicu takvog dokaza koji i pored toga zahteva uvodjenje potpuno novih pojmova koje treba posebno definisati. Jedan od njih je uopštenje pojma ugla između dva trodimenziona prostora unutar istog četvorodimenzionog prostora. To se može učiniti na potpuno analogan način kako se trodimezionalnom prostoru definiše ugao između dve ravni, kao ugao diedra. Dakle, ugao između dva razna trodimeziona prostora se vidi u diedru trećeg trodimenzionog prostora (diedar čine odgovarajući preseci tog trećeg trodimenzionog prostora sa početna dva) koji je upravan na oba data u istom četvorodimenzionom prostoru. Taj postupak se nastavlja i dalje kada se želi definisati ugao između dva četvorodimenzionalna prostora unutar nekog petodimenzionog prostora itd.

Neka je dat sitem linearnih jednačina $n \times n,\ \ n \in N$, koji ima rešenje $X\left(y_1^*, y_2^*, ..., y_n^*\right)$. Označimo sa $P_1^i\left(y_{1,1}^0, y_{2,1}^0, ..., y_{n,1}^0\right)$, $i = 2, 3, ..., n,$ tačke prve iteracije dobijene predloženom metodom. Sada, ako sa d_j^1 , $j = 1,2,...,n$ označimo rastojanja između rešenja i prolaznih tačaka prve iteracije onda posle prvog koraka prve iteracije važi: $d_1^1 = d \cdot cos\left(\angle P_1^1 X P_2^1\right) < d$. Normala ulazi uvek u „oštar ugao“ koji može biti nula samo ako sistem nema rešenja (paralelne hiperravni). Tako dobijamo niz: $d_n^1 < d_{n-1}^1 < ... < d_1^1 < d$. Kako se postupak u sledećim iteracijamo produžava na analogan način to zaključujemo da metoda uvek konvergira pod uslovom da sistem ima rešenje.

Brzina nije konstantna i zavisi od „uglova“ koji čine dve uzastopne tačke sa tačnim rešenjem kao temenom. Što su ti „uglovi“ bliži pravom uglu brzina konvergencije je veća. Ako su svi uglovi veoma mali, oštri uglovi, brzina konvergencije je u tom slučaju sporija. Ako su svi uglovi pravi do tačnog rešenja se dolazi posle tačno jedne jedine kompletne iteracije.

Broj potrebnih operacija. Jednom izračunate dužine vektora normala nije potrebno ponovo računati, a za njih je potrebno izvršiti $2n^2$ operacija. Za računanje svake pojedinačne tačke unutar neke iteracije potrebno je n^2 operacija, a za celu jednu iteraciju ukupno n^3 operacija. Do rešenja se stiže posle m koraka i kao zaključak može se reći da je broj potrebnih operacija u ovoj metodi reda n^3.

VI. PRIMER

Neka je dat sledeći sistem linearnih jednačina:

$43x_1-11x_2+13x_3-17x_4+19x_5-23x_6+29x_7-31x_8+37x_9-41x_{10} = -496$
$41x_1-43x_2+11x_3-13x_4+17x_5-19x_6+23x_7-29x_8+31x_9-37x_{10} = -1008$
$37x_1-41x_2+43x_3-11x_4+13x_5-17x_6+19x_7-23x_8+29x_9-31x_{10} = -204$
$31x_1-37x_2+41x_3-43x_4+11x_5-13x_6+17x_7-19x_8+23x_9-29x_{10} = -864$
$29x_1-31x_2+37x_3-41x_4+43x_5-11x_6+13x_7-17x_8+19x_9-23x_{10} = 0$
$23x_1-29x_2+31x_3-37x_4+41x_5-43x_6+11x_7-13x_8+17x_9-19x_{10}=-864$
$19x_1-23x_2+29x_3-31x_4+37x_5-41x_6+43x_7-11x_8+13x_9-17x_{10}=204$
$17x_1-19x_2+23x_3-29x_4+31x_5-37x_6+41x_7-43x_8+11x_9-13x_{10}=-1008$
$13x_1-17x_2+19x_3-23x_4+29x_5-31x_6+37x_7-41x_8+43x_9-11x_{10}=496$
$11x_1-13x_2+17x_3-19x_4+23x_5-29x_6+31x_7-37x_8+41x_9-43x_{10}=-1008$

Tačno rešenje ovog sistema je: $x_1 = 11$; $x_2 = 13$; $x_3 = 17$; $x_4 = 19$; $x_5 = 23$; $x_6 = 29$; $x_7 = 31$; $x_8 = 37$; $x_9 = 41$; $x_{10} = 43$. Da bi se napisao program po opisanom algoritmu nije potrebno uključivati ništa dodatno u smislu kompleksnosti jer metoda ne menja koeficijente jednačine i ne računa ništa osim vrednosti svake jednačine u pojedinačnim tačkama.

Posle startovanja vrlo jednostavnog programa koji je napisan po opisanom algoritmu i može imati sledeći izgled:

```
#include <iostream>
#include <cmath>
int koja;
void UnesiMatricu(double a[][10]);
void SracunajIteraciju(double a[][10], double B[], double
X[][10], double X0[]);
void PrikaziTacku(double X1[][10]);
using namespace std;
int main() {
   double a[10][10], B[10], X1[10][10], X0[10], Eps, Tacnost
= 1.0;
   int i, j, k = 0;
   cout << "Unesi koeficijente matrice sistema \n";
   UnesiMatricu(a);
   cout << “Unesi desnu stranu sistema”;
   for(i = 0; i < 5; i++)  cin >> B[i];
   cout << "\n Unesi tacnost \n";
   cin >> Eps;
   for(i = 0; i < 10; i++)  X0[i] = 0;
   while (Tacnost > Eps) {
     SracunajIteraciju(a, B, X1, X0);
     Tacnost = 0.0;
     koja++;
     for(i = 0; i < 10; i++)
       Tacnost += (X1[9][i] - X1[8][i])*(X1[9][i] - X1[8][i]);
     Tacnost = sqrt(Tacnost);
   }
   cout << " Resenje se doseze u "<< koja << "iterativnih
koraka";
   for(k = 0; k < 10; k++) cout << X1[9][k]<< "  ";
   cout << "\n";
   system("pause");
}
```

Dobijamo sledeći izveštaj:

Unesi koeficijente matrice sistema

…

Unesi tacnost: Eps=0.00001

Resenje se doseze u 205 iterativnih koraka.

x_1 =10.9999 x_2 =12.9998 x_3 =16.9998 x_4 =18.9999 x_5 =23
x_6 =29.0001 x_7 =31.0002 x_8 =37.0002 x_9 =41.0001 x_{10} =43.
Press any key to continue . . .

Predloženi algoritam treba značajno proširiti i unaprediti i po drugim mogućnostima datih predloženom metodom. Za postavljeni primer brzina i tačnost su u ovom radu od veće važnosti od ostalih problema koji su navedeni. S toga je program napisan u maksimalno pojednostavljenom algoritmu koji sa svoje strane može biti jezgro i mnogo ozbiljnijeg algoritma.

VII. ZAKLJUČAK

Predložena metoda nudi krajnje jednostavan postupak, sa računarske tačke gledišta. Ovakav problem je primer zadatka kojem je dorastao jedino računar. Osim toga, ova metoda: 1. Zahteva broj operacija koji je moguće odraditi u realnom vremenu sa stanovišta moćnijih računara; 2. Ne proizvodi nekontrolisani rast međurezultata. Osim, ako sama veličina rešenja to ne uzrokuje. Tada je to nemoguće izbeći bilo kojom metodom; 3. Ne menja koeficijente početno zadatog sistema i s toga ne može uzrokovati nestabilnost rešenja; 4. Ako rešenje postoji uvek konvergira. Pri tome, postupak može biti započet iz bilo koje tačke prostora kome rešenje pripada. 5. Veoma je jednostavna za sastavljanje algoritma i dozvoljava utvrđivanje i slučajeva kada je sistem nemoguć i neodređen. Zbog svega navedenog trebalo bi joj dati veliki prioritet prilikom izbora metoda za rešavanje sistema jednačina sa ekstremno velikim brojem jednačina i nepoznatih.

Kako je ovo predlog približne metode postavlja se opravdano pitanje: „Da li je moguće kibernetski formulisati metodu koja nije računski zahtevna, a može dovesti do tačnog rešenja?“

LITERATURA

[1] J. Stoer, R. Bulirsch: Introduction to Numerical Analysis, Texts in Applied Mathematics 12, Springer 2002.
[2] N. Higham: Accuracy and Stability of Numerical Algorithms 2000.
[3] Zvnimir Boht. Numerične metode. Drzavna založba Slovenije. Ljubljana 1978.

ABSTRACT

Solving linear systems of equations n x n can be a problem for your computer, even when the number of equations and unknown relatively small (a few hundred). All existing methods are overloaded with: computation complexity, the number of necessary operations that lead to solutions, disproportionate growth in the size of intermediate results, changing the values of some coefficients in the initial system instability resulting solutions etc. This paper presents an approximate method for solving systems of equations with an arbitrary number of equations and unknown. The proposed method avoids all the above-mentioned problems. All the advantages of the proposed method can best be observed on systems that have an extremely large number of equations and unknown.

How to solve a system of linear equations with extremely many unknown

Aleksa Srdanov, Radiša Stefanović
Visoka tehnička škola strukvnih studija
Požarevac, Srbija