Numeričke metode

Aleksandar Maksimović
IRB

Sustavi linearnih jednažbi

- Sustav n jednadžbi i n nepoznanica

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1j}x_j + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2j}x_j + \cdots + a_{2n}x_n &= b_2 \\
\vdots \qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \quad &\quad \vdots \\
a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ij}x_j + \cdots + a_{in}x_n &= b_i \\
\vdots \qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \quad &\quad \vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nj}x_j + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

Matrica $A = [\,a_{ij}\,]_{i,j=1}^{n} \in \mathbb{R}^{n\times n}$ je **matrica sustava**, a njeni elementi su **koeficijenti sustava**. Vektor $b = [\,b_i\,]_{i=1}^{n} \in \mathbb{R}^{n}$ je **vektor desne strane** sustava. Treba odrediti **vektor nepoznanica** $x = [\,x_i\,]_{i=1}^{n} \in \mathbb{R}^{n}$ tako da vrijedi $Ax = b$.

Primjer 1

Interpolacijski polinom

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = \sum_{j=0}^{n} a_j x^j$$

nepoznati su koeficijenti, određujemo ih poznatih vrijednosti y

$$p(x_i) = y_i,\ i = 0, \ldots, n,$$

$$\begin{array}{l} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_{n-1} x_0^{n-1} + a_n x_0^n = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_{n-1} x_1^{n-1} + a_n x_1^n = y_1 \\ \vdots \quad \vdots \quad \vdots \quad \cdots \quad \vdots \quad \vdots \quad \vdots \\ a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_{n-1} x_i^{n-1} + a_n x_i^n = y_i \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_{n-1} x_n^{n-1} + a_n x_n^n = y_n. \end{array}$$

Primjer 1

$$Va = y$$

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \cdots & x_0^{n-1} & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^{n-1} & x_1^n \\ & & \cdots & & \cdots & & \\ 1 & x_i & x_i^2 & x_i^3 & \cdots & x_i^{n-1} & x_i^n \\ & & \cdots & & \cdots & & \\ 1 & x_n & x_n^2 & x_n^3 & \cdots & x_n^{n-1} & x_n^n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}}_{y}.$$

Primjer 2

Promotrimo sljedeći rubni problem:

$$-\frac{d^2}{dx^2}u(x) = f(x), \quad 0 < x < 1,$$
$$u(0) = u(1) = 0.$$

$$h = \frac{1}{n+1}, \quad x_i = ih, \quad i = 0, \ldots, n+1.$$

$$u_i = u(x_i),\ i = 0, \ldots, n+1.$$

$$u_0 = u_{n+1} = 0$$

Primjer 2

$$u(x_i + h) = u(x_i) + u'(x_i)h + \frac{u''(x_i)}{2}h^2 + \frac{u'''(x_i)}{6}h^3 + \frac{u^{(4)}(x_i + \alpha_i)}{24}h^4$$

$$u(x_i - h) = u(x_i) - u'(x_i)h + \frac{u''(x_i)}{2}h^2 - \frac{u'''(x_i)}{6}h^3 + \frac{u^{(4)}(x_i + \zeta_i)}{24}h^4;$$

Kombinacijom ova dva Taylorova reda dobivamo drugu derivaciju

$$u_{i+1} + u_{i-1} = 2u_i + u''(x_i)h^2 + (u^{(4)}(x_i + \alpha_i) + u^{(4)}(x_i + \zeta_i))\frac{h^4}{24},$$

Druga derivacija metodom konačnih razlika

$$-u''(x_i) = \frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2}$$

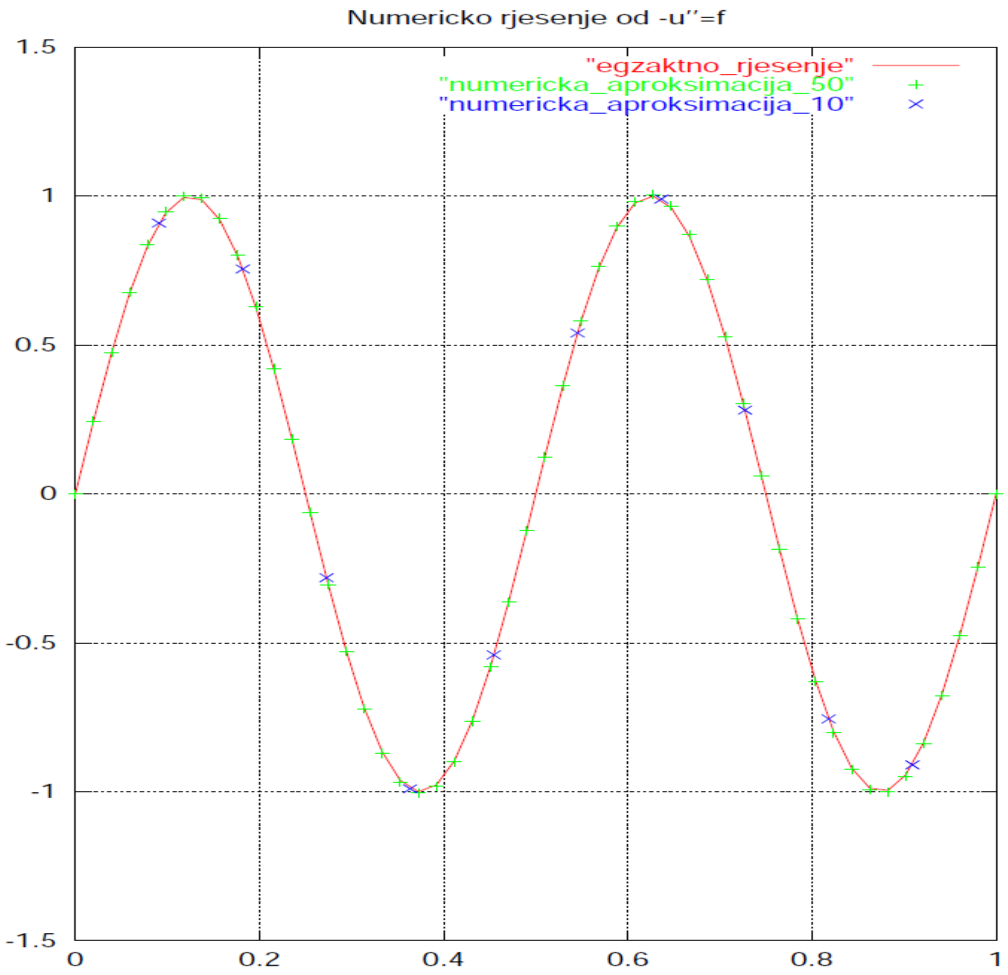
Primjer 2

$$\underbrace{\begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \cdots & \cdots & \cdots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix}}_{T_n} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-2} \\ u_{n-1} \\ u_n \end{bmatrix}}_{u} = h^2 \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{bmatrix}}_{f} .$$

Primjer 2

$$f(x) = 16\pi^2 \sin 4\pi x$$

$$n = 10 \; i \; n = 50$$



Linearni sustav 2x2

$$\begin{aligned} 2x_1 - \quad x_2 &= 1 \\ -x_1 + 2x_2 &= 1 \end{aligned}$$

$$x_1 = \frac{1}{2}(1 + x_2)$$

$$\underbrace{-\frac{1}{2}(1 + x_2)}_{x_1} + 2x_2 = 1, \quad \text{tj. } \frac{3}{2}x_2 = \frac{3}{2}, \quad \text{tj.} \quad x_2 = 1, \qquad x_1 = 1$$

Metoda supstitucije

$$\begin{aligned} 5x_1 + x_2 + 4x_3 &= 19 \\ 10x_1 + 4x_2 + 7x_3 &= 39 \\ -15x_1 + 5x_2 - 9x_3 &= -32 \end{aligned} \quad \equiv \quad \underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 10 & 4 & 7 \\ -15 & 5 & -9 \end{bmatrix}}_{A = [\,a_{ij}\,]_{i,j=1}^{3}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 19 \\ 39 \\ -32 \end{bmatrix}}_{b = [\,b_i\,]_{i=1}^{3}}.$$

Koristimo metodu supstitucija, odnosno eliminacija. Prvo iz prve jednadžbe izrazimo x_1 pomoću x_2 i x_3, te to uvrstimo u zadnje dvije jednadžbe, koje postaju dvije jednadžbe s dvije nepoznanice (x_2 i x_3). Dobivamo

$$x_1 = \frac{1}{5}\,(19 - x_2 - 4x_3),$$

Metoda supstitucije

pa druga jednadžba sada glasi

$$\frac{10}{5}(19 - x_2 - 4x_3) + 4x_2 + 7x_3 = 39,$$

tj.

$$-\frac{10}{5}(x_2 + 4x_3) + 4x_2 + 7x_3 = 39 + \left(-\frac{10}{5}19\right)$$

Dakle, efekt ove transformacije je ekvivalentno prikazan kao rezultat množenja prve jednadžbe s

$$-\frac{a_{21}}{a_{11}} = -\frac{10}{5} = -2$$

i zatim njenim dodavanjem (pribrajanjem) drugoj jednadžbi. Druga jednadžba sada glasi

$$2x_2 - x_3 = 1.$$

Metoda supstitucije

$$\underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 10 & 4 & 7 \\ -15 & 5 & -9 \end{bmatrix}}_{A} \mapsto \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{L^{(2,1)}} \underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 10 & 4 & 7 \\ -15 & 5 & -9 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 0 & 2 & -1 \\ -15 & 5 & -9 \end{bmatrix}}_{A^{(1)} = \left[a_{ij}^{(1)} \right]_{i,j=1}^{3}}$$

Nepoznanicu x_1 eliminiramo iz zadnje jednadžbe ako prvu pomnožimo s

$$\underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 0 & 2 & -1 \\ -15 & 5 & -9 \end{bmatrix}}_{A^{(1)}} \mapsto \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix}}_{L^{(3,1)}} \underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 0 & 2 & -1 \\ -15 & 5 & -9 \end{bmatrix}}_{A^{(1)}} = \underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 0 & 2 & -1 \\ 0 & 8 & 3 \end{bmatrix}}_{A^{(2)} = \left[a_{ij}^{(2)} \right]_{i,j=1}^{3}} .$$

Vektor desne strane je u ove dvije transformacije promijenjen u

Metoda supstitucije

Vektor desne strane je u ove dvije transformacije promijenjen u

$$\underbrace{\begin{bmatrix} 19 \\ 39 \\ -32 \end{bmatrix}}_{b} \mapsto \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{L^{(2,1)}} \begin{bmatrix} 19 \\ 39 \\ -32 \end{bmatrix} = \underbrace{\begin{bmatrix} 19 \\ 1 \\ -32 \end{bmatrix}}_{b^{(1)}}$$

$$\mapsto \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix}}_{L^{(3,1)}} \begin{bmatrix} 19 \\ 1 \\ -32 \end{bmatrix} = \underbrace{\begin{bmatrix} 19 \\ 1 \\ 25 \end{bmatrix}}_{b^{(2)}}.$$

Novi, ekvivalentni, sustav je $A^{(2)}x = b^{(2)}$, *tj.*

Metoda supstitucije

Novi, ekvivalentni, sustav je $A^{(2)}x = b^{(2)}$*, tj.*

$$\begin{aligned} 5x_1 + x_2 + 4x_3 &= 19 \\ 2x_2 - x_3 &= 1 \\ 8x_2 - 3x_3 &= 25, \end{aligned}$$

$$-\frac{a_{32}^{(2)}}{a_{22}^{(2)}} = -4$$

$$\begin{aligned} 5x_1 + x_2 + 4x_3 &= 19 \\ 2x_2 - x_3 &= 1 \\ 7x_3 &= 21. \end{aligned}$$

Metoda supstitucije

$$\underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 0 & 2 & -1 \\ 0 & 8 & 3 \end{bmatrix}}_{A^{(2)}} \mapsto \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{bmatrix}}_{L^{(3,2)}} \underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 0 & 2 & -1 \\ 0 & 8 & 3 \end{bmatrix}}_{A^{(2)}} = \underbrace{\begin{bmatrix} 5 & 1 & 4 \\ 0 & 2 & -1 \\ 0 & 0 & 7 \end{bmatrix}}_{A^{(3)}}$$

transformaciju vektora desne strane

$$\underbrace{\begin{bmatrix} 19 \\ 1 \\ 25 \end{bmatrix}}_{b^{(2)}} \mapsto \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{bmatrix}}_{L^{(3,2)}} \underbrace{\begin{bmatrix} 19 \\ 1 \\ 25 \end{bmatrix}}_{b^{(2)}} = \underbrace{\begin{bmatrix} 19 \\ 1 \\ 21 \end{bmatrix}}_{b^{(3)} = (b_i^{(3)})_{i=1}^{3}} .$$

Metoda supstitucije

1. *Iz treće jednadžbe je* $x_3 = \frac{21}{7} = 3.$
2. *Iz druge jednadžbe je* $x_2 = \frac{1}{2}(1 + x_3) = 2.$
3. *Iz prve jednadžbe je* $x_1 = \frac{1}{5}(19 - x_2 - 4x_3) = 1.$

Opći prikaz metode

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right].$$

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \quad i = 2, \ldots, n.$$

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right].$$

Opći prikaz metode

$$m_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}}, \quad i = 3, \ldots, n,$$

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ & & \ddots & \vdots & \vdots \\ & & & a_{nn}^{(n)} & b_n^{(n)} \end{array} \right].$$

Uz pretpostavku da je $a_{nn}^{(n)} \neq 0$, ovaj se linearni sustav lako rješava povratnom supstitucijom

pivotiranje

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}},$$

$$x_i = \frac{1}{a_{ii}^{(i)}} \left(b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j \right), \quad i = n-1, \ldots, 1.$$

Uobičajeno **parcijalno pivotiranje** kao pivotni element bira element koji je po apsolutnoj vrijednosti najveći u ostatku tog stupca — na glavnoj dijagonali ili ispod nje. Drugim riječima, ako je u k-tom koraku

$$|a_{rk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|,$$

onda ćemo zamijeniti r-ti i k-ti redak i početi korak eliminacije elemenata k-tog stupca.

pivotiranje

Osim parcijalnog pivotiranja, može se provoditi i **potpuno pivotiranje**. U k-tom koraku, bira se maksimalni element u cijelom "ostatku" matrice $A^{(k)}$, a ne samo u k-tom stupcu. Ako je u k-tom koraku

$$|a_{rs}^{(k)}| = \max_{k \leq i,j \leq n} |a_{ij}^{(k)}|,$$

parcijalno pivotiranje - zamjena redaka

potpuno pivotiranje - zamjena redaka i stupaca

algoritam

Algoritam 5.2.1. (Gaussove eliminacije s parcijalnim pivotiranjem)

```
    {Trokutasta redukcija}
for k := 1 to n − 1 do
    begin
      {Nađi maksimalni element u ostatku stupca}
    max_elt := 0.0;
    ind_max := k;
    for i := k to n do
      if abs(A[i, k]) > max_elt then
        begin
        max_elt := abs(A[i, k]);
```

algoritam

```
        ind_max := i;
        end;
if max_elt > 0.0 then
    begin
    if ind_max <> k then
        {Zamijeni k-ti i ind_max-ti r
        begin
        for j := k to n do
            begin
            temp := A[ind_max, j];
            A[ind_max, j] := A[k, j];
            A[k, j] := temp;
            end;
        temp := b[ind_max];
        b[ind_max] := b[k];
        b[k] := temp;
        end;
    for i := k + 1 to n do
        begin
        mult := A[i, k]/A[k, k];
        A[i, k] := 0.0;    {Ne treba, ne koristi se kasnije}
        for j := k + 1 to n do
            A[i, j] := A[i, j] - mult * A[k, j];
        b[i] := b[i] - mult * b[k];
        end;
    end
else
        {Matrica je singularna, stani s algoritmom}
    begin
    error := true;
    exit;
    end;
end;
```

algoritam

```
  {Povratna supstitucija, rješenje x ostavi u b}
b[n] := b[n]/A[n, n];
for i := n − 1 downto 1 do
  begin
  sum := b[i];
  for j := i + 1 to n do
    sum := sum − A[i, j] ∗ b[j];
  b[i] := sum/A[i, i];
  end;
error := false;
```

Zadatak 5.2.1. *Pokušajte samostalno napisati algoritam koji koristi potpuno pivotiranje. Posebnu pažnju obratite na efikasno pamćenje zamjena varijabli koje su posljedica zamjena stupaca. Može li se isti princip efikasno primijeniti i za pamćenje zamjena redaka, tako da se potpuno izbjegnu eksplicitne zamjene elemenata u matrici A i vektoru b?*

gaussj.c

```c
#include <math.h>
#define NRANSI
#include "nrutil.h"
#define SWAP(a,b) {temp=(a);(a)=(b);(b)=temp;}
void gaussj(float **a, int n, float **b, int m)
{
    int *indxc,*indxr,*ipiv;
    int i,icol,irow,j,k,l,ll;
    float big,dum,pivinv,temp;

    indxc=ivector(1,n);
    indxr=ivector(1,n);
    ipiv=ivector(1,n);
    for (j=1;j<=n;j++) ipiv[j]=0;
    for (i=1;i<=n;i++) {
        big=0.0;
        for (j=1;j<=n;j++)
            if (ipiv[j] != 1)
                for (k=1;k<=n;k++) {
                    if (ipiv[k] == 0) {
                        if (fabs(a[j][k]) >= big) {
                            big=fabs(a[j][k]);
                            irow=j;
                            icol=k;
                        }
                    }
                }
        ++(ipiv[icol]);
        if (irow != icol) {
            for (l=1;l<=n;l++) SWAP(a[irow][l],a[icol][l])
            for (l=1;l<=m;l++) SWAP(b[irow][l],b[icol][l])
        }
        indxr[i]=irow;
        indxc[i]=icol;
        if (a[icol][icol] == 0.0) nrerror("gaussj: Singular Matrix");
        pivinv=1.0/a[icol][icol];
        a[icol][icol]=1.0;
        for (l=1;l<=n;l++) a[icol][l] *= pivinv;
        for (l=1;l<=m;l++) b[icol][l] *= pivinv;
        for (ll=1;ll<=n;ll++)
            if (ll != icol) {
                dum=a[ll][icol];
                a[ll][icol]=0.0;
                for (l=1;l<=n;l++) a[ll][l] -= a[icol][l]*dum;
                for (l=1;l<=m;l++) b[ll][l] -= b[icol][l]*dum;
            }
    }
    for (l=n;l>=1;l--) {
        if (indxr[l] != indxc[l])
            for (k=1;k<=n;k++)
                SWAP(a[k][indxr[l]],a[k][indxc[l]]);
    }
    free_ivector(ipiv,1,n);
    free_ivector(indxr,1,n);
    free_ivector(indxc,1,n);
}
#undef SWAP
#undef NRANSI
```

LU faktorizacija

$$A^{(3)} = L^{(3,2)} L^{(3,1)} L^{(2,1)} A.$$

Matrica $A^{(3)}$ je gornjetrokutasta, a produkt $L^{(3,2)} L^{(3,1)} L^{(2,1)}$ je donjetrokutasta matrica. Dakle, polaznu matricu A smo množenjem slijeva donjetrokutastom matricom načinili gornjetrokutastom. To možemo pročitati i ovako:

$$A = LA^{(3)}, \quad L = (L^{(2,1)})^{-1}(L^{(3,1)})^{-1}(L^{(3,2)})^{-1},$$

gdje je L donjetrokutasta matrica. Lako se provjerava da je

$$L = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{(L^{(2,1)})^{-1}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}}_{(L^{(3,1)})^{-1}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{bmatrix}}_{(L^{(3,2)})^{-1}} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & 4 & 1 \end{bmatrix}.$$

LU faktorizacija

Matrica A je produkt gornjotrokutaste i donjotrokutaste matrice

$$A = LA^{(3)} \qquad U = A^{(3)} \qquad A = LU.$$

Imamo: A x =b, A=L U, L (U x) = b, U x=y,

L y = b

U x = y

Linearni sustav riješen je pomoću 3 koraka:

1. Matricu sustava A treba faktorizirati u obliku $A = LU$, gdje je L donjetrokutasta, a U gornjetrokutasta matrica.
2. Rješavanjem donjetrokutastog sustava $Ly = b$ treba odrediti vektor $y = L^{-1}b$.
3. Rješavanjem gornjetrokutastog sustava $Ux = y$ treba odrediti vektor $x = U^{-1}y = U^{-1}(L^{-1}b)$.

supstitucija unaprijed

$$\begin{bmatrix} \ell_{11} & 0 & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & \ell_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

linearni sustav s donjotrokutastom matricom

matrica je regularna $\ell_{ii} \neq 0$

algoritam:

$$x_1 = \frac{b_1}{\ell_{11}};$$

za $i = 2, \ldots, n$ {

$$x_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij} x_j \right) \Big/ \ell_{ii}; \}$$

$$x_1 = \frac{b_1}{\ell_{11}}$$

$$x_2 = \frac{1}{\ell_{22}} (b_2 - \ell_{21} x_1)$$

$$x_3 = \frac{1}{\ell_{33}} (b_3 - \ell_{31} x_1 - \ell_{32} x_2)$$

$$x_4 = \frac{1}{\ell_{44}} (b_4 - \ell_{41} x_1 - \ell_{42} x_2 - \ell_{43} x_3).$$

supstitucija unazad

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

linearni sustav s gornjotrokutastom matricom

algoritam

$$x_n = \frac{b_n}{u_{nn}};$$

za $i = n-1, \ldots, 1$ {

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \Big/ u_{ii}; \;\}$$

$$x_4 = \frac{b_4}{u_{44}}$$

$$x_3 = \frac{1}{u_{33}} (b_3 - u_{34} x_4)$$

$$x_2 = \frac{1}{u_{22}} (b_2 - u_{23} x_3 - u_{24} x_4)$$

$$x_1 = \frac{1}{u_{11}} (b_1 - u_{12} x_2 - u_{13} x_3 - u_{14} x_4).$$

gauss eliminacija

SUBROUTINE gaussj(a,n,np,b,m,mp)

Linear equation solution by Gauss-Jordan elimination, equation (2.1.1) above.

a(1:n,1:n) is an input matrix stored in an array of physical dimensions np by np.

b(1:n,1:m) is an input matrix containing the m right-hand side vectors, stored in an array of physical dimensions np by mp. On output, a(1:n,1:n) is replaced by its matrix inverse, and b(1:n,1:m) is replaced by the corresponding set of solution vectors.

6 argumenata

void gaussj(float **a, int n, float **b, int m)

4 argumenta, ne treba NP i MP dimenzije od matrice i vektora.

b argument kod C verzije je matrica.

NR primjer

- Fortran
  - Files: Xgaussj.for, gaussj.for, MATRX1.dat
    - Xgaussj.for je modificirani "kod" originala xgaussj.for, problem je bio s čitanjem podataka iz file-a. xgaussj.for nemodificirani kod.
    - f77 -o fgauss Xgaussj.for gaussj.for
- C
  - Files: xgaussj.c gaussj.c nrutils/nrutil.c nrutils/, MATRX1.dat
    - nrutils/ sadrži .h fileove, tj. nrutil.h i nr.h
    - nrutil/, tj. nrutil.c i nrutil.h dijelovi za alociranje matrice, vektora itd..
  - gcc -o xgauss xgaussj.c gaussj.c nrutils/nrutil.c -I nrutils/

F77 gauss

fortran primjer: tgauss.f

Kompajliranje: g77 -o tg1 tgauss.f gaussj.for lib1.f

```
 open (unit=8,file="linsys1.dat")

 call readmatrix(MA1,n1,m1,fp)
call readvektor(b,L1,fp)

call gaussj(TA1,n1,NP,x,1,MP)
```

```
print *, "Solution is: "
call displayvektor(x,L1)
write(*,*) 'Inverse of Matrix A : '
call displaymatrix(TA1,n1,m1)
call  multMatrixMatrix(TA1,MA1,XMM,n1,m1,n1,m1)
 write(*,*) 'Inverse of Matrix A * A: '
 call displaymatrix(XMM,n1,m1)
```

C gauss

C primjer: tgauss.c

Kompajliranje: gcc -g -o tg tgauss.c libnr.c gaussj.c nrutils/nrutil.c -I nrutils/

ako se nrutil.c i header fileovi nalaze u poddirektoriju nrutils

```
#include "nr.h"
#include "nrutil.h"
#include "libnr.h"
#define NP 20
-----------------------------------
 float *b,*x;
  /*  float b[10],x[10]; */
  float **MA1,**TA1,**XMM,**X;
```

```
 X=matrix(1,NP,1,NP);
 b=vector(1,NP);
readmatrix(MA1,n1,m1,fp);
 readvektor(b,L1,fp);

gaussj(TA1,n1,X,1);
 printf("Solution is:\n");
            displaymatrix(X,n1,1);
printf("Inverse of  A : \n"); displaymatrix(TA1,n1,m1);
```

LU metoda

SUBROUTINE ludcmp(a,n,np,indx,d)

void ludcmp(float **a, int n, int *indx, float *d)

Given a matrix a[1..n][1..n], this routine replaces it by the LU decomposition of a rowwise permutation of itself. a and n are input. a is output; indx[1..n] is an output vector that records the row permutation effected by the partial pivoting; d is output as ±1 depending on whether the number of row interchanges was even or odd, respectively. This routine is used in combination with lubksb to solve linear equations or invert a matrix.

NR PRIMJER:

```
 gcc -o xludcmpC xludcmp.c  ludcmp.c nrutils/nrutil.c -I nrutils/
g77 -o xludcmpF xludcmp.for  ludcmp.for
gcc -o xlubsub xlubksb.c  lubksb.c ludcmp.c nrutils/nrutil.c -I nrutils/
 g77 -o xlubsubF xlubksb.for  lubksb.for ludcmp.for
```

SUBROUTINE lubksb(a,n,np,indx,b)

void lubksb(float **a, int n, int *indx, float b[])

Solves the set of n linear equations A X = B. Here a is input, not as the matrix A but rather as its LU decomposition, determined by the routine ludcmp. indx is input as the permutation vector returned by ludcmp. b(1:n) is input as the right-hand side vector B, and returns with the solution vector X. a, n, np, and indx are not modified by this routine and can be left in place for successive calls with different right-hand sides b. This routine takes into account the possibility that b will begin with many zero elements, so it is effcient for use in matrix inversion.

```
 g77 -g -o tludecmp tludecmp.for ludcmp.for lib1.f
gcc -g  -o cludecmp tludecmp.c ludcmp.c nrutils/nrutil.c libnr.c -I nrutils/
gcc -g  -o cludecmp tludecmp.c ludcmp.c nrutils/nrutil.c libnr.c -I nrutils/
g77 -o tlubksub tlubksub.for lubksb.for ludcmp.for lib1.f
gcc -g  -o clubksb tlubksub.c ludcmp.c lubksb.c nrutils/nrutil.c libnr.c -I nrutils/
```

Osobine metoda

- Kvadratični linearni sustavi, broj nepoznanica= jednadžbi
- Pivotiranje je neophodno za stabilnost algoritma
- Gaussova eliminacija
  - Slabosti
    - Zahtjeva desnu stranu (vektor b), koju treba mjenjati u algoritmu
    - Sporost, naročito kada nam ne treba inverzna matrica
  - Prednost, jednostavna
- LU faktorizacija, manji broj operacije od Gauss metode

Zadaci za praktikum

1. Izvršite programe napravljene od xgaussj i tgaussj rutina koje koriste gaussovu eliminaciju.
2. Napravite program koji pročita linearni sustav iz datoteke LIN.DAT i riješite sustav gaussovom metodom.
3. Izvršite programe dobivene iz xludcmp , xlubksb, tludecmp, tlubksb koji koriste LU faktorizaciju.
4. Napravite program koji pročita linearni sustav iz datoteke LIN.DAT i riješite sustav LU faktorizacijom.
5. Dobiveno riješenje pošaljite kao poruku u mailu (jedno riješenje), a source programa stavite kao attachment. Mail je Aleksandar.Maksimovic@irb.hr

Literatura

- Online literatura:
  - Numerička matematika-osnovni udžbenik, PMF, projekt mzt.
  - Numerical Recipes in C
  - Numerical Recipes in Fortran