NUMERIČKE METODE I MATEMATIČKO MODELIRANJE

6. PREDAVANJE

LINEARNA ALGEBRA U NUMERIČKIM METODAMA I MATEMATIČKOM MODELIRANJU

LINEARNA ALGEBRA - VEKTORI I MATRICE

- vektori se prikazuju pomoću jednodimenzionalnih polja

$$\mathbf{x} \in \mathbb{R}^n \iff \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

- tipična klasifikacija matrica:

$\mathbf{A} = \mathbf{A}^T$	simetrična	$a_{ij} = a_{ji}$
$\mathbf{A} = \left(\mathbf{A}^T\right)^{-1}$	realna ortogonalna	$\sum_k a_{ik}a_{jk} = \sum_k a_{ki}a_{kj} = \delta_{ij}$
$\mathbf{A} = \mathbf{A}^*$	realna	$a_{ij} = a^*_{ij}$
$\mathbf{A} = \mathbf{A}^\dagger$	hermitska	$a_{ij} = a^*_{ji}$
$\mathbf{A} = \left(\mathbf{A}^\dagger\right)^{-1}$	unitarna	$\sum_k a_{ik}a^*_{jk} = \sum_k a^*_{ki}a_{kj} = \delta_{ij}$

LINEARNA ALGEBRA - MATRICE

- česti tipovi matrica:

1) dijagonalna, kada je $a_{ij} = 0$,za $i \neq j$

2) gornja trokutna $a_{ij} = 0$ za $i > j$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{nn} \end{pmatrix}$$

3) donja trokutna $a_{ij} = 0$ za $i < j$

$$\begin{pmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

4) gornja Hessenbergova $a_{ij} = 0$ za $i > j + 1$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{pmatrix}$$

5) donja Hessenbergova $a_{ij} = 0$ za $i < j + 1$

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

6) trodijagonalna $a_{ij} = 0$ za $|i - j| > 1$

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{pmatrix}$$

LINEARNA ALGEBRA - MATRICE

Osnovne operacije sa matricama:

- zbrajanje i oduzimanje

$$\mathbf{A} = \mathbf{B} \pm \mathbf{C} \Longrightarrow a_{ij} = b_{ij} \pm c_{ij}$$

- množenje skalara i matrice

$$\mathbf{A} = \gamma \mathbf{B} \Longrightarrow a_{ij} = \gamma b_{ij}$$

- množenje vektora i matrice

$$\mathbf{y} = \mathbf{A}\mathbf{x} \Longrightarrow y_i = \sum_{j=1}^{n} a_{ij} x_j$$

- množenje dvije matrice

$$\mathbf{A} = \mathbf{B}\mathbf{C} \Longrightarrow a_{ij} = \sum_{k=1}^{n} b_{ik} c_{kj}$$

- transponiranje matrice

$$\mathbf{A} = \mathbf{B}^T \Longrightarrow a_{ij} = b_{ji}$$

- konjugiranje kompleksne matrice $\mathbf{A} \in \mathbb{C}^{n \times n}$

$$\mathbf{A} = \overline{\mathbf{B}}^T \Longrightarrow a_{ij} = \overline{b}_{ji}$$

$$z = x + \imath y \qquad \overline{z} = x - \imath y$$

Osnovne operacije sa vektorima:

- zbrajanje i oduzimanje vektora

$$\mathbf{x} = \mathbf{y} \pm \mathbf{z} \Longrightarrow x_i = y_i \pm z_i$$

- množenje skalara i vektora

$$\mathbf{x} = \gamma \mathbf{y} \Longrightarrow x_i = \gamma y_i$$

LINEARNA ALGEBRA - VEKTORI

- množenje dva vektora (Hadamardov produkt)

$$\mathbf{x} = \mathbf{yz} \Longrightarrow x_i = y_i z_i$$

- unutarnji (skalarni) produkt dva vektora

$$\mathbf{x} = \mathbf{y}^T \mathbf{z} \Longrightarrow x_i = \sum_{j=1}^{n} y_j z_j$$

- vanjski (vektorski) produkt dva vektora

$$\mathbf{A} = \mathbf{y}\mathbf{z}^T \Longrightarrow a_{ij} = y_i z_j$$

LINEARNA ALGEBRA - NORMA

- Norma vektora i matrica
- Klasa vektorskih normi: p-norme

$$||\mathbf{x}||_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}$$

$$||\mathbf{x}||_1 = |x_1| + |x_2| + \cdots + |x_n|$$

$$||\mathbf{x}||_2 = (|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2)^{\frac{1}{2}} = (\mathbf{x}^T\mathbf{x})^{\frac{1}{2}}$$

$$||\mathbf{x}||_\infty = \max |x_i| \qquad 1 \leq i \leq n$$

- Cauchy-Schwartz nejednakost: za svaki x i y (realni ili kompleksni prostor), ako su x i y linearno zavisni vrijedi nejednakost

$$|\mathbf{x}^T\mathbf{y}| \leq ||\mathbf{x}||_2||\mathbf{y}||_2$$

LINEARNA ALGEBRA - NORMA

- norma se može koristiti u proračunu relativne greške u reprezentaciji vektora x u računalu

- npr. ako je $fl(\mathbf{x}) \in \mathbb{R}^n$ reprezentacija vektora $\mathbf{x} \in \mathbb{R}^n$ u računalu, onda se za $\mathbf{x} \neq 0$ može definirati relativna greška

$$\epsilon = \frac{||fl(\mathbf{x}) - \mathbf{x}||}{||\mathbf{x}||}$$

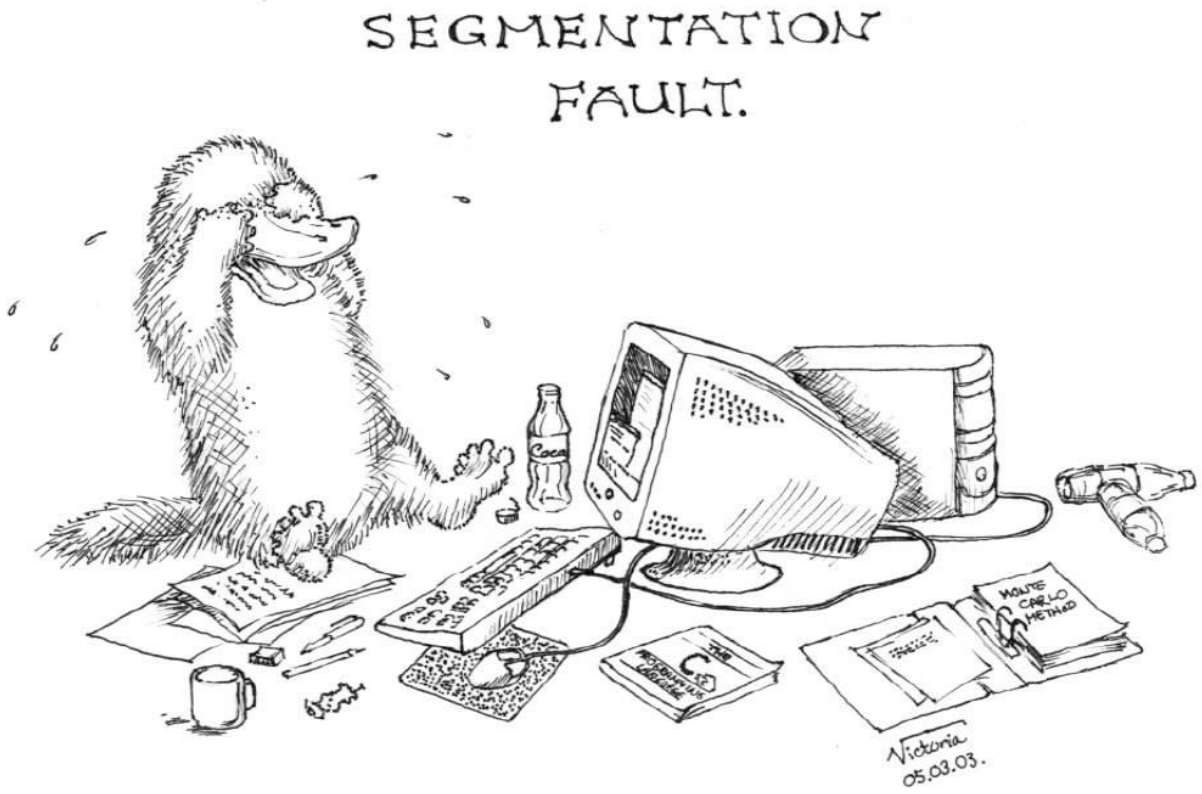
- matrična norma – npr. Frobeniusova norma

$$||\mathbf{A}||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}$$

- p-norma $$||\mathbf{A}||_p = \frac{||\mathbf{Ax}||_p}{||\mathbf{x}||_p} \qquad \mathbf{x} \neq 0$$

LINEARNA ALGEBRA - VEKTORI I MATRICE

- vektori i matrice u matematičkom modeliranju se kodiraju jednodimenzionalnim i dvodimenzionalnim poljima
- Potrebna je posebna pozornost u inicijalizaciji, pristupu, alociranju i brisanju polja



Victoria Popsueva (M. H. Jensen, Computation Physics)

MATEMATIČKO MODELIRANJE SA VEKTORIMA I MATRICAMA

- dva moguća smjera u radu sa poljima:

1) deklaracije polja gdje je dimenzija polja dana pri kompajliranju

2) deklaracije polja gdje je dimenzija polja određena prilikom samog izvođenja programa (dinamička alokacija memorije)

- deklaracije i baratanje jednodimenzionalnim poljima smo već savladali

- u slučaju matrica primjenjujemo analogni pristup, u 2) slučaju koriste se dvostruki pokazivači (pokazivači na pokazivače za neku varijablu) npr. int **matr;

DEFINIRANJE POLJA PRILIKOM KOMPAJLIRANJA

```
int main()
{
    int k,m, row = 3, col = 5;
    int vec[5];
    int matr[3][5];

    for(k = 0; k < col; k++) vec[k] = k;
    for(m = 0; m < row; m++) {
        for(k = 0; k < col ; k++) matr[m][k] = m + 10 * k;
    }
    printf("\n\nVector data in main():\n");
    for(k = 0; k < col; k++) printf("vector[%d]  = %d ",k, vec[k]);
     printf("\n\nMatrix data in main():");
    for(m = 0; m < row; m++) {
        printf("\n");
        for(k = 0; k < col; k++)
            printf("matr[%d][[%d]  = %d ",m,k,matr[m][k]);
        }
    }
    printf("\n");
    sub_1(row, col, vec, matr);
    return 0;
}   // End: function main()
```

Vektori i matrice su ovdje definirani unaprijed, polja su određena prilikom kompajliranja programa

DEFINIRANJE POLJA PRILIKOM KOMPAJLIRANJA

```
void sub_1(int row, int col, int vec[], int matr[][5])
{
    int k,m;

    printf("\n\nVector data in sub_1():\n");
    for(k = 0; k < col; k++) printf("vector[%d]  = %d ",k, vec[k]);
     printf("\n\nMatrix data in sub_1():");
    for(m = 0; m < row; m++) {
        printf("\n");
        for(k = 0; k < col; k++) {
            printf("matr[%d][[%d]  = %d ",m, k, matr[m][k]);
        }
    }
    printf("\n");
}   // End: function sub_1()
```

DINAMIČKA DEFINICIJA POLJA ZA VEKTORE I MATRICE

```
int main()
{
    int *vec;
    int **matr;
    int m, k, row, col, total = 0;

    printf("\n\nRead in number of rows = ");
    scanf("%d",&row);
    printf("\n\nRead in number of column = ");
    scanf("%d", &col);
    vec = new int [col];
    matr = (int **)matrix(row, col, sizeof(int));
    for(k = 0; k < col; k++) vec[k] = k;
    for(m = 0; m < row; m++) {
        for(k = 0; k < col; k++) matr[m][k] = m + 10 * k;
    }
    printf("\n\nVector data in main():\n");
    for(k = 0; k < col; k++) printf("vector[%d] = %d ",k,vec[k]);
```

DINAMIČKA DEFINICIJA POLJA ZA VEKTORE I MATRICE

```
    printf("\n\nArray data in main():");
    for(m = 0; m < row; m++) {
        printf("\n");
        for(k = 0; k < col; k++) {
            printf("matrix[%d][%d] = %d ",m, k, matr[m][k]);
        }
    }
    printf("\n");
    for(m = 0; m < row; m++) {
        for(k = 0; k < col; k++) total += matr[m][k];
    }
    printf("\n\nTotal = %d\n",total);
    sub_1(row, col, vec, matr);
    free_matrix((void **)matr);
    delete [] vec;
    return 0;
}   // End: function main()
```

DINAMIČKA DEFINICIJA POLJA ZA VEKTORE I MATRICE

```
void sub_1(int row, int col, int vec[], int **matr)                    // line h
{
    int k,m;

    printf("\n\nVector data in sub_1():\n");              // print vector data
    for(k = 0; k < col; k++) printf("vector[%d] = %d ",k, vec[k]);
    printf("\n\nMatrix data in sub_1():");
    for(m = 0; m < row; m++) {
        printf("\n");
        for(k = 0; k < col; k++) {
            printf("matrix[%d][[%d] = %d ",m,k,matr[m][k]);
        }
    }
    printf("\n");
}   // End: function sub_1()
```

DINAMIČKA DEFINICIJA POLJA ZA VEKTORE I MATRICE

- program poziva funkciju void **matrix(…) koja rezervira memoriju za dvodimenzionalnu matricu
- matrica je reprezentirana dvostrukim pokazivačem na segment memorije koji sadrži niz double* pokazivača
- svaki double* pokazivač pokazuje na jedan red matrice

DINAMIČKA DEFINICIJA POLJA ZA VEKTORE I MATRICE

- moguće je korištenje gotove biblioteke sa funkcijama za definiranje matrica i vektora, itd.

http://folk.uio.no/mhjensen/fys3150/2005/programs/cplusplusLibrary/lib.cpp

- potrebno je uključiti u program i odgovarajuću datoteku zaglavlja lib.h

- Navedena biblioteka se može preuzeti sa web stranice kolegija, preporuča se integriranje poziva unutar makefile-a

OPERACIJE S MATRICAMA

- zbrajanje dvije $n \times n$ matrice $\mathbf{A} = \mathbf{B} + \mathbf{C}$

```
for(i=0 ; i < n ; i++) {
    for(j=0 ; j < n ; j++) {
        a[i][j]=b[i][j]+c[i][j]
    }
}
```

- množenje dvije $n \times n$ matrice $\mathbf{A} = \mathbf{BC}$

```
for(i=0 ; i < n ; i++) {
    for(j=0 ; j < n ; j++) {
        for(k=0 ; k < n ; k++) {
            a[i][j]+=b[i][k]*c[k][j]
        }
    }
}
```

OPERACIJE S MATRICAMA

- prilikom korištenja matrica u modeliranju na računalu, treba voditi računa o raspoloživoj memoriji
- za pohranjivanje $n \times n$ matrice čiji elementi su dvostruke preciznosti, potrebno je 64 bita po matričnom elementu, tj. ukupno $n \times n \times 8$ byte-ova
- dakle, za pohranjivanje matrice dimenzije $n = 10000$ potrebno je približno 1GB memorije
- u pravilu se izbjegava fiksno definiranje dimenzija matrica pri kompajliranju, koristi se dinamičko alociranje memorije i oslobađanje memorije kada matrica više nije potrebna (koristiti new i delete)

ZADATAK 6

Napisati program u kojem se dinamički alociraju dvije nxn matrice A i B, u programu treba definirati vrijednosti matričnih elemenata.

Treba definirati dvije funkcije koje se pozivaju iz glavnog programa i izvode sljedeće:

1) funkcija koja računa matricu C, gdje je C = A+A*A - A*B

2) funkcija koja provjerava da li je matrica A dijagonalna, gornja trokutna ili donja trokutna

U svakoj od navedenih funkcija treba ispisati rezultat. Provjeriti na jednostavnom primjeru matrice malih dimenzija da program dobro radi. Preporuka je koristiti funkcije iz lib.cpp, odnosno lib.h.