Proiect PCLP3
Simulator de Octave

Realizat de: Botez Luca
Croitoru Constantin-Bogdan

Pentru acest proiect am extins tema 2 de la PCLP1, in care am avut sa creem un simulator de Octave.

In implementarea initiala a acestei teme, datele au fost retinute ca un vector de matrici, lucru care nu este eficient din punct de vedere al memoriei deoarece trebuie sa existe mereu o zona contigua de memorie care sa cuprinda toate datele. Astfel eu si colegul meu am eficientizat aceasta tema si am schimbat implementarea. In loc sa folosim un vector de matrici , am folosit listele inlantuite, lucru care a eficientizat codul din punct de vedere al memoriei si nu a mai fost nevoie sa dam realloc vectorului de fiecare data cand adaugam sau stergem o matrice.

In codul proiectului exista urmatoarele functii pe care le-am modificat pentru a se potrivi cu implemetarea cu liste:

1)incarcarea in memorie a unei matrice (aceasta functie retine o matrice in memorie)

Mod de apelare: L dim1 dim2

2)determinarea dimensiunilor unei matrice (aceasta functie afiseaza dimensiunea unei matrici, indicele matricei se citeste de la tastatura)

Mod de apelare: D index

3)afisarea unei matrice (se afiseaza o anumita matrice din lista)

Mod de apelare: P index

4)redimensionarea unei matrice (aceasta functie redimensioneaza o matrice, iar rezultatul este retinut in locul matricei initiale)

Mod de apelare: C index

5)inmultirea a doua matrice (inmulteste 2 matrice, iar rezultatul este adaugat in lista, ca ultimul nod)

Mod de apelare: M index1 index2

7)sortarea matricelor (aceasta functie ordoneaza matricile in functie de suma elementele)

Mod de apelare: O

8)transpunerea unei matrice (aceasta functie transpune o matrice , iar rezultatul este retinut in locul matricei initiale)

Mod de apelare: T index

9)ridicarea unei matrice la o putere in timp logaritmic

Mod de apelare: R index1 putere

10)eliberarea memoriei unei matrice(se elimina din memorie o matrice)

Mod de apelare: F index

11)eliberarea tuturor resurselor (se elibereaza toate memoria)

Mod de apelare: Q

Pe langa operatiile pe care le-am modificat pentru a se potrivi in implementarea cu lista, am adaugat urmatoarele functii:

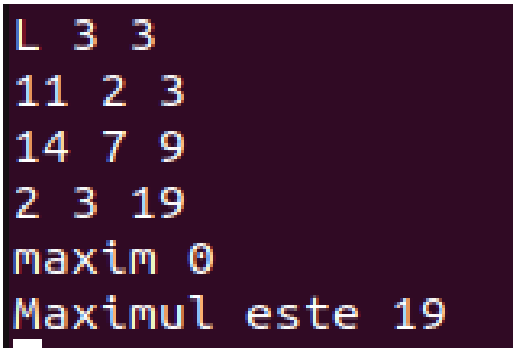
Functiile realizate de Croitoru Constantin-Bogdan:

1.MAXIMUL

- aceasta functie calculeaza maximul dintr-o matrice, iar rezultatul este afisat

Mod de apelare: maxim index

Exemplu de rulare:

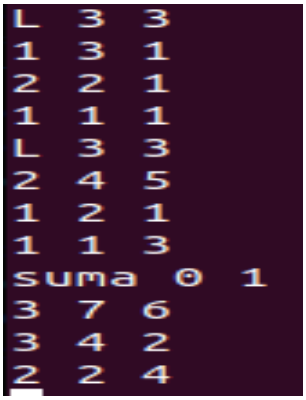


2.SUMA

– aceasta realizeaza adunarea dintre doua matrici si afiseaza rezultatul

Mod de apelare : suma index1 index2

Exemplu de rulare:

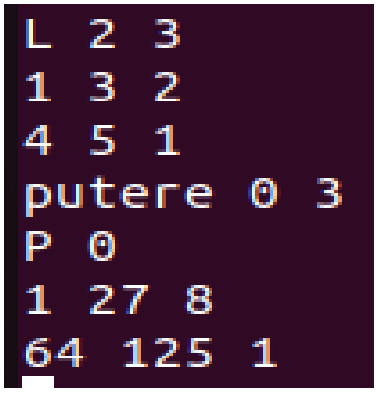


3.PUTERE

-aceasta functie ridica fiecare element la o putere si este echivalenta cu functie .^ din octave

Mod de apelare : putere index put

Exemplu de rulare:

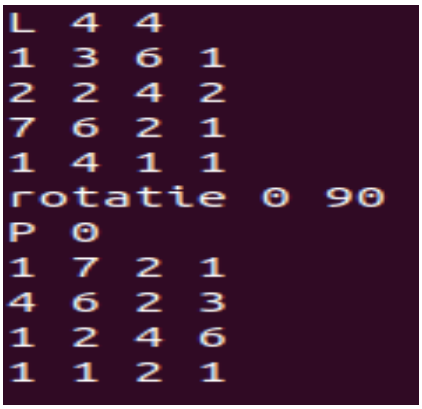


4.Rotire

-aceasta functie roteste o matrice patratica la 90 180 270 de grade

Mod de apelare : rotatie index grade

Exemplu de rulare:



5.Determinant

-aceasta primeste o matrice, verifica daca este patratica si afiseaza determinantul acesteia

Mod de apelare : determinant index

Exemplu de rulare:

```
L 3 3
1 2 3
2 5 1
3 2 6
determinant 0
Determinantul matricei este -23
```

$$\begin{vmatrix} 1 & 2 & 3 \\ 2 & 5 & 1 \\ 3 & 2 & 6 \end{vmatrix} = -23$$

6.RANG

-aceasta functie calculeaza rangul unei matrici

Mod de apelare : rang index

Exemplu de rulare:

```
L 3 3
3 2 1
4 7 1
1 2 5
rang 0
Rangul este 3
```

$$\operatorname{rank}\begin{pmatrix} 3 & 2 & 1 \\ 4 & 7 & 1 \\ 1 & 2 & 1 \end{pmatrix} = 3$$

7.Strassen

- functia inmulteste doua matrici prin metoda strassen

Mod de apelare : strassen index1 index2

Exemplu de rulare:

```
L 4 4
1 2 3 4
2 3 1 1
2 2 1 1
3 2 1 1

L 4 4
1 7 2 1
2 4 2 1
2 4 5 1
2 2 2 2
strassen 0 1
19 35 29 14
12 32 17 8
10 28 15 7
11 35 17 8
```

8.Doolittle

-aceasta functie descompunere matricea in produs de 2 matrici L si U

Mod de apelare : doolittle index

Exemplu de rulare:

```
L 3 3
1 3 2
2 1 1
2 3 1
doolittle 0
L este:
1.000000 0.000000 0.000000
2.000000 1.000000 0.000000
2.000000 0.600000 1.000000
U este:
1.000000 3.000000 2.000000
0.000000 -5.000000 -3.000000
0.000000 0.000000 -1.200000
```

$$\begin{pmatrix} 1 & 3 & 2 \\ 2 & 1 & 1 \\ 2 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & \frac{3}{5} & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 3 & 2 \\ 0 & -5 & -3 \\ 0 & 0 & \frac{-6}{5} \end{pmatrix}$$

Functiile realizate de Botez Luca:

Inversa unei matrici

Descriere: Programul verifica daca matricea data este nonsingulara, calculeaza si afiseaza inversa.

Mod de apelare `inverse idx`

Exemplu de rulare:

```
load 3 3
4 2 1
3 8 9
7 5 2

inverse 0
The inverse is :
0.674419 -0.023256 -0.232558
-1.325581 -0.023256 0.767442
0.953488 0.139535 -0.604651
```

$$\begin{pmatrix} 4 & 2 & 1 \\ 3 & 8 & 9 \\ 7 & 5 & 2 \end{pmatrix}^{(-1)} = \begin{pmatrix} 0.67442 & -0.02326 & -0.23256 \\ -1.32558 & -0.02326 & 0.76744 \\ 0.95349 & 0.13953 & -0.60465 \end{pmatrix}$$

Valoarea proprie dominanta

Descriere: Programul calculeaza valoarea proprie dominanta cu ajutorul metodei puterii directe (MPD), pornind de la o aproximatie a solutiei. Rezultatul nu va fi unul exact, ci o aproximare cu o eroare ce poate fi modificata.

Mod de apelare: `eig idx`

Exemplu de rulare:

```
load 2 2
5 4
1 2

eig 0
6.000040
```

$$A=\begin{pmatrix} 5 & 4 \\ 1 & 2 \end{pmatrix}$$

Eigenvectors for the matrix A:

- $v=\begin{pmatrix} -1 \\ 1 \end{pmatrix}$, eigenvalue $\lambda_1=1$
- $v=\begin{pmatrix} 4 \\ 1 \end{pmatrix}$, eigenvalue $\lambda_2=6$

Aducerea la forma superior triunghiulara

Descriere: Programul calculeaza si afiseaza matricea superior triunghiulara cu ajutorul eliminarii gaussiene.

Mod de apelare: `upper idx`

Exemplu de rulare:

```
load 3 3
4 2 1
3 8 9
7 5 2

upper 0
4.00000 2.00000 1.00000
0.00000 6.50000 8.25000
0.00000 0.00000 -1.65385
```

$$\begin{pmatrix} 4 & 2 & 1 \\ 3 & 8 & 9 \\ 7 & 5 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0.75 & 1 & 0 \\ 1.75 & 0.23077 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & 2 & 1 \\ 0 & 6.5 & 8.25 \\ 0 & 0 & -1.65385 \end{pmatrix}$$

Elementul minim

Descriere: Programul determina si afiseaza elementul minim al unei matrici.

Mod de apelare: `min idx`

Exemplu de rulare:

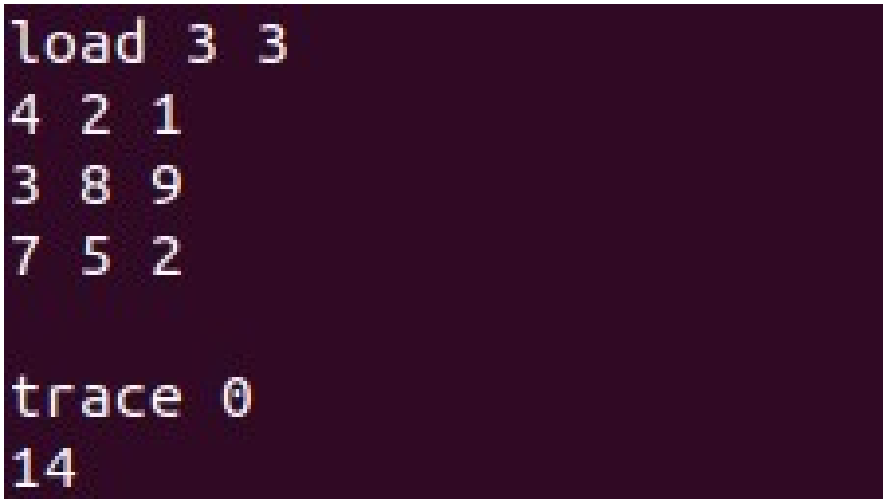
```
load 3 3
4 2 1
3 8 9
7 5 2

min 0
1
```

Urma unei matrici

Descriere: Programul calculeaza si afiseaza suma elementelor de pe diagonala principala ale unei matrici (urma).

Mod de apelare: `trace idx`

Exemplu de rulare:



Produsul Hadamard a doua matrice

Descriere: Programul calculeaza si afiseaza produsul Hadamard dintre doua matrice (daca acestea au dimensiunile egale).

Mod de apelare: `hadamard idx1 idx2`

Exemplu de rulare:

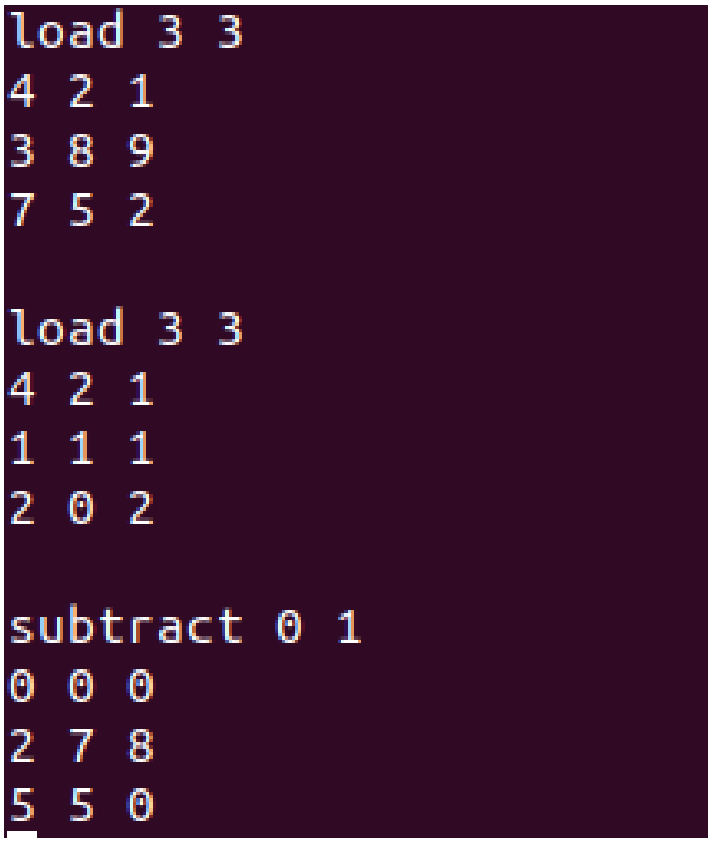
```
load 3 3
4 2 1
3 8 9
7 5 2

load 3 3
2 2 2
2 2 2
2 2 2

hadamard 0 1
8 4 2
6 16 18
14 10 4
```

Scaderea a doua matrice

Descriere: Programul calculeaza si afiseaza diferenta dintre doua matrice (daca acestea au dimensiunile egale).

Mod de apelare: `subtract idx1 idx2`

Exemplu de rulare:



Concepte acoperite:

In realizarea proiectului am lucrat cu liste, lucru pe care l-am facut la curs si laborator, am creat un makefile pentru proiect, am folosit alocarea dinamica a datelor, am impartit codul pe headere si am creat un define cu ajutorul caruia verificam alocarea dinamica.

Nu am avut dificultati majore in realizarea acestui proiect, singurele dificultati intalnite au fost eliberarea corecta a memoriei si fara erori, si documentarea cum putem sa realizam anumite operatii.

Acest proiect ne-a ajutat sa ne imbunatatim lucrul in echipa, sa invatam sa folosim git-ul , sa utilizam valgrind-ul si gdb-ul, facand debug asupra codului.

Acest proiect se poate folosi ca un octave, dar la capacitate mult mai mica deoarece permite doar operatii asupra matricilor.

Link catre repository-ul proiectului:

https://github.com/lucabotez/Advanced-Matrix-Calculator