Laborator 2

Citiți din fișierul pdf *Laborator1.pdf* materialele referitoare la reprezentări grafice, fișiere script și fișiere funcție și instrucțiuni de control în Matlab (de la pagina 13 până la pagina 20).

După ce sunteți familiarizați cu noțiunile descrise şi implementare lor în Matlab rezolvați exercițiile de mai jos.

Exerciţiul 1. Scrieţi codul Matlab (realizând fie o funcție, fie un script, fie scriind codul în linia de comandă) care trasează graficul funcţiei următoare:

$$f: [-10, 10] \rightarrow R \qquad f(x) = \begin{cases} 2x + 8, \text{ dacă } x \leq 2 \\ 3x^2, \text{ dacă } x > 2 \end{cases}$$

Coloraţi diferit cele două ”ramuri” are funcției modificând și alte proprietăți ale graficelor (markeri, grosime, etc.).

Exerciţiul 2. Scrieți o funcție în Matlab cu numele *genereazaMatrice.m* care primește ca argument un număr natural nenul *n* și întoarce o matrice A, cu n linii şi n+1 coloane ale cărei elemente sunt:

$$A(i,j) = \begin{cases} 2, \text{ dacă } i = j; \\ -1, \text{ dacă } |i - j| = 1; \text{ (în Matlab funcţia modul este } abs) \\ 0, \text{ în rest} \end{cases}$$

Indicație: puteți genera matricea A fie pe baza instrucțiunilor repetitive (dar neeficient în Matlab) sau folosind programarea matriceală implementată efficient în Matlab (generați A setând valorile elementelor pe baza indicilor sau scriind A ca sumă de matrice care se pot genera ușor cu funcția *eye*).

Exerciţiul 3

a. Scrieți o funcție în Matlab cu numele *adunaElemente.m* folosind instrucțiuni de control (for, while, break, etc.) care primește ca argument o matrice și returnează suma tuturor elementelor matricei, parcurse linie cu linie, până la întâlnirea celui mai mic element al matricei (acesta nu este luat în calculul sumei).

Exemplu: pentru matricea A de forma

```
      1 2 3 4
A =   5 6 7 8        suma returnată va fi 1+2+…+9 = 45.
      9 0 1 2
```

b. Generați cu ajutorul funcției *rand* o matrice A de dimensiuni *n x n* pentru numere aleatoare în (0,1). Calculați timpul de calcul necesar rulării funcției scrise folosind funcțiile *tic* și *toc* pentru valori ale lui *n* egale cu 10, 100, 1000, 10000.

c. Puteți scrie o implementare mai bună din punct de vedere al timpului de rulare pentru functia *adunaElemente.m*? (folosiți proprietățile de calcul matriceal implementate eficient în Matlab).

d. Plotați timpul de rulare al funcției pentru cele două implementări (cu instrucțiuni de control vs programare matriceală) pentru diferite valori ale lui *n*.

Exercițiul 4. Implementați înmulțirea a două matrice A și B de dimensiuni *n x n* folosind instrucțiunile de control și comparați din punct de vedere al timpului de rulare cu implementarea eficientă din Matlab (C = A * B).

Exerciţiul 5. Scrieți funcția *calculeazaDistante.m* ce calculează distanța euclidiană între toate perechile de linii a două matrice X și Y. Presupunem că matricea X are dimensiunile n × m, matricea Y are dimensiunile p × m. Matricea rezultantă Z va avea dimensiunile n × p, unde Z(i,j) va reprezenta distanța euclidiană dintre linia i a matricei X și linia j a matricei Y.

Indicație: implementați eficient în Matlab calculul distanței folosind programarea matriceală. Puteți avea în vedere relația funcțiile *repmat, sum, sqrt* sau relația

$$||\mathbf{a} - \mathbf{b}||^2 = ||\mathbf{a}||^2 + ||\mathbf{b}||^2 - 2\mathbf{a}^T\mathbf{b},$$

unde **a** și **b** sunt vectori (spre exemplu **a** e o linie din X și **b** e o linie din Y).