Teme de laborator

Norme de matrici.
Metode iterative pentru sisteme liniare.

1. Fie $A \in \mathcal{M}_{n,n}(\mathbb{C})$ inversabilă. Se numeşte *număr de condiţionare (conditioning number)* al matricii A relativ la norma operator $||\cdot||$, numărul

$$\mathcal{K}(A) := \mathcal{K}_{||\cdot||}(A) = ||A|| \left|\left|A^{-1}\right|\right|.$$

Dacă A nu este inversabilă vom scrie $\mathcal{K}(A) = \infty$.

a) Să se arate că pentru orice normă operator $\mathcal{K}(A) \geq 1$.

b) Fie $A \in \mathcal{M}_{n,n}(\mathbb{C})$ o matrice unitară, i.e., $A^*A = I$. Arătaţi că

$$\mathcal{K}_2(A) = 1,$$

unde $\mathcal{K}_2(A)$ este calculată cu norma $||\cdot||_2$.

c) **(Octave/MatLab)** Utilizaţi funcţia predefinită `hilb` pentru a genera câteva matrici Hilbert. Scrieţi un script Octave/MatLab care să calculeze numerele de condiţionare relative la normele 2, 1 şi ∞ pentru matrici Hilbert de dimensiuni $n = 3, ..., 7$. Cum sunt aceste matrici din punct de vedere al condiţionării?

2) **(Octave/MatLab)** Se consideră sistemul liniar:

$$\begin{pmatrix} 2.01 & 1.99 \\ 1.99 & 2.01 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$$

i) Se observă că $x^* = (1,1)^T$ este o soluţie a sistemului de mai sus. Notăm cu A matricea sistemului şi cu b membrul drept. Considerăm o perturbare $\widehat{b}$ a lui b,

$$\widehat{b} = \begin{pmatrix} 4.02 \\ 3.98 \end{pmatrix}.$$

Se observă că soluţia sistemului $Ax = \widehat{b}$ este $\widehat{x} = (2,0)^T$.

ii) Verificaţi soluţiile sistemelor de mai sus folosind următoarele comenzi:

```
>> A= [2.01 1.99;1.99 2.01];
>> b = [4;4];
>> bhat = [4.02;3.98];
>> xstar=A\b
>> xhat =A\bhat
```

iii) Calculaţi $\mathcal{K}_2(A)$, $\mathcal{K}_1(A)$ şi $\mathcal{K}_\infty(A)$. Explicaţi diferenţa între x^* şi $\widehat{x}$.

2. **(Octave/MatLab)** Să se scrie o funcţie cu antetul

```
function A = frank_matrix(n)
```

care să implementeze matricea Frank de dimensiune n. O matrice $A = (a_{ij})_{1\le i,j\le n}$ este o matrice Frank dacă

$$a_{ij} = \begin{cases} 0, & j < i-2, \\ n+1-i, & j = i-1, \\ n+1-j, & j \ge i. \end{cases}$$

Printaţi numerele de condiţionare pentru matricile Frank implementate folosind scriptul `test_frank_matrix.m`.

3. **(Octave/MatLab)** Să se scrie o funcţie ce generează o matrice diagonal dominantă pe baza unei matrici date, după următoarea schemă: dacă $A = (a_{ij})_{1\le i,j\le n}$ este matricea input, iar $B = (b_{ij})_{1\le i,j\le n}$ este matricea output, atunci

$$b_{ij} = \begin{cases} a_{ij}, & i \ne j, \\ 1 + \sum_{k\ne i} |a_{ik}|, & i = j. \end{cases}$$

Funcţia Octave/Matlab corespunzătoare va avea antetul

```
function B = create_dominant_matrix(A)
```

Scriptul de test pentru aceasta este `test_dominant_matrix.m`.

4. **(Octave/MatLab)** Folosind problema de mai sus să se genereze o matrice diagonal dominantă ce are elementele din afara diagonalei numere aleatoare, iar cele de pe diagonala principală să fie obţinute în acelaşi mod ca mai sus. Funcţia va avea antetul

```
function A = rand_dominant_matrix(n)
```

unde `n` este dimensiunea matricii. Folosiţi *built-in function* `randn` pentru a genera componentele aleatorii.

5. **(Octave/MatLab)** Implementaţi şi testaţi metoda lui Jacobi pentru rezolvarea sistemelor liniare. Folosiţi descrierea pe componente a metodei. Mai precis, scrieţi două funcţii Octave/MatLab, după cum urmează:

   - `function xk = iteratie_Jacobi(A,b,x0)`, unde `A` este matricea sistemului, `b` este vectorul (coloană) reprezentând membrul drept al sistemului liniar, `x0` vectorul de start al iteraţiei, iar `xk` vectorul obţinut după o **singură** iteraţie. Antetul funcţiei este precizat în porţiunea de cod de mai jos:

```
function xk = iteratie_Jacobi(A,b,x0)
% functie ce implementeaza o iteratie din metoda lui Jacobi
% Se foloseste forma metodei pe componente.
%
% Input:
%     - A,b:  matricea si membrul drept al sistemului Ax = b
%     - x0:   valoarea de start a iteratiei (vector coloana)
%
% Output:
%     - xk:   valoarea de final a iteratiei
%
% ISA 2020-2021
```

- `function [xk,iter,isTerminated] = metoda_Jacobi(A,b,x0,tol,maxIter)` unde `A,b` sunt matricea si membrul drept al sistemului liniar, `x0` este vectorul iniţial, `maxIter` este numărul maxim de iteraţii admis, `xk` este iteraţia finală, `iter` este numărul de iteraţii realizate, iar `isTerminated` o valoare din $\{0,1\}$ cu `isTerminated = 0` dacă `maxIter` a fost atins şi `isTerminated = 1` altfel. Parametrul de input, `tol` este folosit în criteriul de terminare al iteraţiilor, i.e., metoda este oprită dacă:
$$\left|\left|x^{(k+1)} - x^{(k)}\right|\right|_2 \leq tol.$$
Antetul funcţiei este precizat în porţiunea de cod de mai jos:

```
function [xk,iter,isTerminated] = metoda_Jacobi(A,b,x0,tol,maxIter)
% functie ce implementeaza metoda lui Jacobi
% Se foloseste forma metodei pe componente.
%
% Input:
%     - A,b:  matricea si membrul drept al sistemului Ax = b
%     - x0:   prima valoare in metoda iterativa
%     - tol:  valoare definita de utilizator pentru a fi folosita
%             pe post de criteriu de terminare al iteratiilor.
%     - maxIter: numarul maxim de iteratii
%
% Output:
%     - xk:   valoarea ultimei iteratii
%     - iter: numarul de iteratii pana la raspunsul final
%     - isTeminated: 0 daca s-a atins maxIter, 1 altfel
%
% ISA 2020-2021
```

În cazul în care matricea $B := D^{-1}B$ satisface

$$||B||_\infty < 1 \text{ sau } ||B||_1 < 1$$

folosiţi un criteriu de terminare alternativ care asigură utilizatorul că diferenţa între soluţia aproximativă şi cea exactă este mărginită de ceva cunoscut.

Folosiţi scriptul `print_tabel_Jacobi.m` pentru a testa metoda lui Jacobi. Explicaţi pe baza cunoştiinţelor teoretice (analitice) valorile ultimei rubrici din tabel.