# Seminar 6

# Probleme de examen

Acest seminar are rolul de a vă pregati pentru examen.

Examenul final este unul scris, la care trebuie să rezolvaţi un set de probleme personalizat. Nu uitaţi că puteţi folosi orice material scris doriţi. Puteţi de asemenea să folosiţi orice tip de calculator care nu este conectat la vreo sursă externă de informaţii (cum ar fi conexiunea wireless la INTERNET). Folosirea telefoanelor mobile şi orice fel de comunicare cu colegii nu este permisă. Pedeapsa pentru o acţiune interzisă este pierderea examenului.

Dăm în continuare câteva exemple exemple de seturi de probleme pentru examenul final.

## 6.1 Setul de probleme nr. 1

**1.** Matricea $A \in \mathbb{R}^{n\times n}$ are elemente nenule numai pe diagonală şi pe prima coloană. Scrieţi un algoritm eficient care să:

**a.** calculeze $\|A\|_1$, $\|A\|_\infty$ $\|A\|_F$;

**b.** rezolve $Ax = b$, unde $b \in \mathbb{R}^n$ este dat;

**c.** calculeze $A^{-1}$.

**2. a.** Adaptaţi algoritmul de factorizare Crout fară pivotare pentru o matrice nesingulară şi inferior Hessenberg $H \in \mathbb{R}^{n\times n}$ având toate submatricele lider principal nesingulare.

**b.** Folosind rezultatul algoritmului precedent, scrieţi un algoritm care să rezolve sistemul $Dx = H^{-1}b$, unde $D \in \mathbb{R}^{n\times n}$ este o matrice diagonală nesingulară iar $b \in \mathbb{R}^n$ este un vector dat. Atât matricea $D$ cât şi vectorul $b$ sunt date.

**3.** Fie $U_1 \in \mathbb{R}^{m\times m}$ un reflector elementar dat (de indice 1) definit de vectorul $u_1 \in \mathbb{R}^m$. Scrieţi un algoritm eficient care să

**a.** calculeze $U_1A$, unde $A \in \mathbb{R}^{m\times n}$ este dat;

**b.** rezolve sistemul $U_1x = b$, cu $b \in \mathbb{R}^m$ dat;

**4.** Fie $b \in \mathbb{R}^m$ dat. Scrieţi un algoritm care să calculeze $m-1$ rotaţii $G_1$, $G_2$, ..., $G_{m-1}$ astfel încât $G_{m-1}\dots G_2G_1b = \alpha e_m$, unde $\alpha = \|b\|_2$. (Amintiţi-vă că $e_m$ este vectorul unitate de indice $m$.)

### 6.1.1 Soluţia problemelor din setul nr. 1

*Observaţie* Programele MATLAB de implementare a algoritmilor elaboraţi sunt prezentaţi în Anexa

**Problema 1.** Matricea dată $A \in \mathbb{R}^{n\times n}$ are elemente nenule numai pe diagonală şi pe prima coloană, astfel $A$ este o matrice inferior triunghiulară cu următoarea structură.

$$A = \begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

**a.** Exploatând structura matricei avem:

$$\mu = \|A\|_1 = \max_{j=1:n}\left(\sum_{i=1:n}|a_{ij}|\right) = \max\left(\sum_{i=1:n}|a_{i1}|, \max_{j=2:n}(|a_{jj}|)\right),$$

$$\nu = \|A\|_\infty = \max_{i=1:n}\left(\sum_{j=1:n}|a_{ij}|\right) = \max\left(|a_{11}|, \max_{i=2:n}(|a_{i1}| + |a_{ii}|)\right),$$

$$\rho = \|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}^2} = \sqrt{a_{11}^2 + \sum_{i=2}^{n}(a_{i1}^2 + a_{ii}^2)}.$$

Astfel, un algoritm eficient care calculează cele trei norme ale matricei este:

1. $\mu = 0$
2. **pentru** $i = 1 : n$
    1. $\mu = \mu + |a_{i1}|$
3. **pentru** $j = 2 : n$
    1. **dacă** $|a_{jj}| > \mu$
        1. $\mu = |a_{jj}|$
4. $\nu = |a_{11}|$
5. **pentru** $i = 2 : n$
    1. $\sigma = |a_{i1}| + |a_{ii}|$
    2. **dacă** $\sigma > \nu$
        1. $\nu = \sigma$
6. $\rho = a_{11}^2$
7. **pentru** $i = 2 : n$
    1. $\rho = \rho + a_{i1}^2 + a_{ii}^2$
8. $\rho = \sqrt{\rho}$

**b.** Sistemul $Ax = b$, unde $b \in \mathbb{R}^n$ este dat, este inferior triunghiular şi are o soluţie unică datorită faptului că matricea $A$ este nesingulară, întrucât $a_{ii} \neq 0$ oricare ar fi $i = 1 : n$. Pentru a calcula $x$ vom adapta algoritmul de substituţie înainte la structura matricei $A$. Ecuaţia $i$ este $a_{i1}x_1 + a_{ii}x_i = b_i$, aşadar vom folosi formula $x_i = (b_i - a_{i1}x_1)/a_{ii}$. Algoritmul este

1. $x_1 = b_1/a_{11}$
2. **pentru** $i = 2 : n$
    1. $x_i = (b_i - a_{i1}x_1)/a_{ii}$

**c.** *Soluţia 1.* Pentru a calcula $X = A^{-1}$ vom rezolva sistemul liniar matriceal $AX = I_m$, care este echivalent cu rezolvarea sistemelor binare $Ax_j = e_j$, $j = 1 : n$, unde $x_j = X(:, j)$ este coloana $j$ a lui $X$ şi $e_j = I_m(:, j)$ este coloana $j$ a matricei unitate $I_m$. Observaţi mai întâi că matricea $X = A^{-1}$ este inferior triunghiulară deoarece $A$ este inferior triunghiulară. Astfel $x_j(1 : j-1) = 0$. La fel, ca în cazul algoritmului general de inversare a matricelor inferior triunghiulare (vezi cursul), vectorul $x_j(j : n)$ este soluţia sistemului liniar inferior triunghiular

$$A(j:n, j:n)x_j(j:n) = e_1, \quad \text{unde} \quad e_1 = e_j(j:n) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{n-j+1}.$$

Datorită structurii matricei $A$, pentru $j = 2 : n$ matricea $A(j : n, j : n)$ este una diagonală şi, astfel, soluţia sistemului de mai sus pentru $j = 2 : n$ este

$$x_j(j:n) = \begin{bmatrix} 1/a_{jj} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \text{astfel} \quad x_j = \frac{1}{a_{jj}}\, e_j \in \mathbb{R}^n.$$

Prima coloană a lui $X$ poate fi calculată de algoritmul de la punctul **b**) pentru $b = e_1$. În concluzie, matricea $X = A^{-1}$ are aceeaşi structură ca şi $A$ şi poate fi calculată eficient de următorul algoritm.

1. $x_{11} = 1/a_{11}$
2. **pentru** $i = 2 : n$
   1. $x_{i1} = -a_{i1}x_{11}/a_{ii}$
3. **pentru** $j = 2 : n$
   1. **pentru** $i = 1 : n$
      1. $x_{ij} = 0$
   2. $x_{jj} = 1/a_{jj}$

*Soluţia 2.* Fie $D = \text{diag}(A)$ matricea diagonală nesingulară

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

şi definim

$$M_1 = D^{-1}A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_{21}/a_{22} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}/a_{nn} & 0 & \cdots & 1 \end{bmatrix} = I_n - m_1 e_1^T, \quad \text{unde} \quad m_1 = \begin{bmatrix} 0 \\ -a_{21}/a_{22} \\ \vdots \\ -a_{n1}/a_{nn} \end{bmatrix}$$

care este o matrice inferior triunghiulară elementară de ordin $n$ şi indice 1. Deci (vezi cursul), inversa sa este $Y = M_1^{-1} = I_n + m_1 e_1^T$. Astfel,

$$X = A^{-1} = D^{-1}M_1^{-1} = \begin{bmatrix} 1/a_{11} & 0 & \cdots & 0 \\ 0 & 1/a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/a_{nn} \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -a_{21}/a_{22} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1}/a_{nn} & 0 & \cdots & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1/a_{11} & 0 & \cdots & 0 \\ -a_{21}/(a_{22}a_{11}) & 1/a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}/(a_{nn}a_{11}) & 0 & \cdots & 1/a_{nn} \end{bmatrix}$$

Algoritmul explicit este

1. $x_{11} = 1/a_{11}$
2. **pentru** $i = 2 : n$
   1. $x_{i1} = -a_{i1}/(a_{11}a_{ii})$
3. **pentru** $j = 2 : n$
   1. **pentru** $i = 1 : n$
      1. $x_{ij} = 0$
   2. $x_{jj} = 1/a_{jj}$

cu acelaşi număr de operaţii (flops) $N_{fl} \approx 3n$ cu solutia 1.

*Observaţie.* O soluţie e suficientă pentru a obţine punctajul maxim.

**Problema 2. a.** Algoritmul de factorizare Crout fără pivotare pentru o matrice inferior Hessenberg $H = L * U$ poate fi obţinut în aceeaşi manieră ca şi algoritmul general (vezi cursul), calculând, în ordine directă, o coloana a matricei inferior triunghiulare $L$ şi linia corespunzătoare a matricei superior triunghiulare unitate $U$. Utilizând identitatea

$$\begin{bmatrix} h_{11} & h_{12} & \cdots & 0 & 0 \\ h_{21} & h_{22} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ h_{n-1,1} & h_{n-1,2} & \cdots & h_{n-1,n-1} & h_{n-1,n} \\ h_{n1} & h_{n2} & \cdots & h_{n,n-1} & h_{nn} \end{bmatrix} =$$

$$= \begin{bmatrix} l_{11} & 0 & \cdots & 0 & 0 \\ l_{21} & l_{22} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ l_{n-1,1} & l_{n-1} & \cdots & l_{n-1,n-1} & 0 \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & l_{nn} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & \cdots & u_{1,n-1} & u_{1n} \\ 0 & 1 & \cdots & u_{2,n-1} & u_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & u_{n-1,n} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

putem continua după cum urmează.

*Pas 1.* Avem $h_{i1} = l_{i1} * 1$, astfel $l_{i1} = h_{i1}$, $i = 1 : n$. Atunci $h_{1,j} = l_{11} * u_{1j}$, astfel $u_{12} = h_{12}/l_{11}$, şi $u_{2j} = 0$ pentru $j = 3 : n$. Observaţi că matricea $U$ este bidiagonală pe prima linie.

*Pas k.* Presupunem că am calculat primele $k-1$ coloane ale matricei $L$ şi primele $k-1$ linii ale matricei $U$ şi că matricea $U$ este superior bidiagonală pe primele $k-1$ linii. Avem $h_{ik} = l_{i,k-1} * u_{k-1,k} + l_{ik} * 1$, astfel

$$l_{ik} = h_{ik} - l_{i,k-1} * u_{k-1,k}, \quad i = 1 : n.$$

Atunci, $h_{k,j} = l_{kk} * u_{kj}$ şi datorită faptului că $u_{kj} = 0$ pentru $j = k+2 : n$ avem

$$u_{k,k+1} = h_{k,k+1}/l_{kk} \quad \text{si} \quad u_{kj} = 0 \ \text{ pentru } \ j = k+1 : n.$$

Astfel, matricea $U$ este bidiagonală pe linia $k$. Prin inducţie, matricea $U$ va fi superior bidiagonală.

Expresiile de mai sus pentru elementele matricelor $L$ şi $U$ duc la următorul algoritm Crout adaptat pentru matrice inferioare Hessenberg.

1. **pentru** $i = 1 : n$
   1. **pentru** $j = 1 : n$
      1. $l_{ij} = 0$
      2. $u_{ij} = 0$
   2. $u_{ii} = 1$
2. **pentru** $i = 1 : n$
   1. $l_{i1} = h_{i1}$
3. $u_{12} = h_{12}/l_{11}$
4. **pentru** $k = 2 : n$
   1. **pentru** $i = k : n$
      1. $l_{ik} = h_{ik} - l_{i,k-1} * u_{k-1,k}$
   2. **dacă** $k = n$
      1. STOP
   3. $u_{k,k+1} = h_{k,k+1}/l_{kk}$

Evident, matricea Hessenberg $H$ poate fi suprascrisă de matricea $L$ şi de matricea $U$ (exceptând diagonala). Numărul de operaţii (flops) necesar algoritmului este $N_{fl} \approx 3n$.

**b.** Sistemul $Dx = H^{-1}b$ este echivalent cu $HDx = b$ şi folosind algoritmul de factorizare Crout de la punctul precedent sistemul se transformă în $LUDx = b$. Astfel, notând $Dx = y$ şi $Uy = z$ o variantă eficientă pentru a calcula $x$ este

1. Se rezolvă sistemul inferior triunghiular $Lz = b$
2. Se rezolvă sistemul superior bidiagonal $Uy = z$
3. Se rezolvă sistemul diagonal $Dx = y$

care este detaliată în următorul algoritm

1. $z = LTRIS(L, b)$
2. $y_n = z_n$
3. **pentru** $i = n - 1 : -1 : 1$
   1. $y_i = z_i - u_{i,i+1} * z_{i+1}$
4. **pentru** $i = 1 : n$
   1. $x_i = y_i/d_{ii}$

Efortul de calcul este $N_{fl} \approx n^2$ operaţii (flops).

**Problema 3.**

**a.** Pentru a calcula $B = U_1A$, unde $A \in \mathbb{R}^{m\times n}$ este dat, vom partiţiona $B$ pe coloane, şi vom calcula $b_j = U_1a_j$ pentru $j = 1 : n$, unde $b_j = B(:, j)$ şi $a_j = A(:, j)$. Datorită faptului că $U_1 = I_m - u_1u_1^T/\beta_1$ unde $\beta_1 = \|u_1\|^2/2$, , avem

$$b_j = U_1a_j = (I_m - u_1u_1^T/\beta_1)a_j = a_j - (u_1u_1^T/\beta_1)a_j = a_j - \tau u_1,$$

unde $\tau = (u_1^Ta_j)/\beta_1 = \left(\sum_{i=1}^m u_{i1}a_{ij}\right)/\beta_1$. Astfel, un algoritm eficient pentru calcularea matricei $B = U_1A$ este

1. $\beta_1 = \left(\sum_{i=1}^m u_{i1}^2\right)/2$
2. **pentru** $j = 1 : n$
   1. $\tau = \left(\sum_{i=1}^m u_{i1}a_{ij}\right)/\beta_1$
   2. **pentru** $i = 1 : m$
      1. $b_{ij} = a_{ij} - \tau u_{i1}$,

şi necesită $N_{fl} \approx 4mn$ operaţii (flops).

**b.** Pentru a rezolva sistemul $U_1x = b$, vom folosi faptul că reflectorii $U_1$ sunt matrice simetrice şi ortogonale. Astfel, $x = U_1^{-1}b = U_1^Tb = U_1b$ iar algoritmul este

1. $\beta_1 = \left(\sum_{i=1}^{m} u_{i1}^2\right)/2$
2. $\tau = \left(\sum_{i=1}^{m} u_{i1} b_i\right)/\beta_1$
3. **pentru** $i = 1 : m$
   1. $x_i = b_i - \tau u_{i1}$,

cerând $N_{fl} \approx 6m$ operaţii (flops).

**Problema 4.** (10p) Problema cere un algoritm care să calculeze $m-1$ rotaţii $G_1$, $G_2$, ..., $G_{m-1}$ astfel încât $G_{m-1} \dots G_2 G_1 b = \alpha e_m$, unde $\alpha = \|b\|_2$, deci avem nevoie să transformăm în zero *primele* elemente ale vectorului $b$. Pentru a face asta să calculăm matricea de rotaţie $P_{ij}$ pentru $i < j$ astfel încât $(P_{ij}b)(i) = 0$. Scalarii $c$ şi $s$ trebuiesc să fie o soluţie a sistemului

$$\begin{cases} c^2 + s^2 = 1 \\ cb_i + sb_j = 0 \end{cases}$$

cerinţă care este satisfacută, de exemplu, de

$$c = \frac{b_j}{\sqrt{b_i^2 + b_j^2}}, \qquad s = \frac{-b_i}{\sqrt{b_i^2 + b_j^2}}.$$

Folosind astfel de rotaţii porblema este rezolvată dacă luăm spre exemplu $G_k = P_{m-k,m}$, astfel încât secvenţa următoare

$$b \leftarrow P_{1,m} \dots P_{m-2,m} P_{m-1,m} b = \alpha e_m$$

reprezintă schema de calcul.

1. **pentru** $i = m-1 : -1 : 1$
   1. Se calculează $c_i$, $s_i$ adică rotaţia $P_{im}$, astfel încât $(P_{im}b)(i) = 0$
   2. $b \leftarrow P_{im} b$.

Algoritmul este

1. **pentru** $i = m-1 : -1 : 1$
   1. $\rho = \sqrt{b_i^2 + b_m^2}$
   2. $c_i = b_m/\rho$
   3. $s_i = -b_i/\rho$
   4. $b_i = 0$
   5. $b_m = \rho$

şi necesită $m-1$ extrageri de radical şi $5(m-1)$ operaţii (flops).

## 6.2 Setul de probleme nr. 2

**1.** Care sunt permutările de linii realizate când se aplică algoritmul $GPP$ matricei $A = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 4 & 3 \\ 1 & -3 & -1 \end{bmatrix}$.

**2.** Fie $A \in \mathbb{R}^{n \times n}$ o matrice inferior Hessenberg ($a_{ij} = 0$, pentru $j > i+1$) având toate submatricele $A(1:k, 1:k)$ nesingulare.

**a.** Scrieţi un algoritm eficient care să calculeze norma Frobenius a matricei $A$.

**b.** Adaptaţi algoritmul de eliminare gaussiană (fără pivotare) pentru matricea $A$.

**c.** Scrieţi un algoritm detaliat pentru rezolvarea sistemului $A^{-p}x = b$, unde $p \geq 2$ şi $b \in \mathbb{R}^n$ sunt de asemenea date.

**3.** Dacă matricea $A \in \mathbb{R}^{n\times n}$ este simetrică şi pozitiv definită, elaboraţi un algoritm detaliat pentru calculul factorizării Cholesky $A = UU^T$, unde $U$ este o matrice superior triunghiulară cu elemente diagonale pozitive

**4.** Calculaţi o forma Schur reală a matricei $A = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}$.

## 6.3 Setul de probleme nr. 3

**1.** Care sunt permutările care apar la aplicarea algoritmului $GPP$ matricei $A = \begin{bmatrix} 1 & -3 & -2 \\ -2 & 4 & -2 \\ 1 & 2 & 3 \end{bmatrix}$ ?
Aceeaşi întrebare pentru cazul în care se aplică $GPC$.

**2.** Se poate utiliza *în siguranţă* algoritmul de eliminare Gaussiană (fără pivotare) pentru a calcula determinantul unei matrice ? De ce ?

**3.** Fie $A \in \mathbb{R}^{n\times n}$ o matrice tridiagonală cu toate submatricele $A(1:j, 1:j)$ nesingulare.

**a.** Scrieţi un algoritm eficient care să calculeze eficient norma infinit a matricei $A$ ($\|A\|_\infty$).

**b.** Adaptaţi algoritmul Crout pentru $A$.

**4.** Fie $A \in \mathbb{R}^{n\times n}$ dat. Scrieţi detaliat un algoritm care să

**a.** calculeze reflectorul Householder $U_2$ astfel încât $(U_2A)_{i1} = 0$, $i = 3:n$;

**b.** calculeze $A \leftarrow U_2A$;

**c.** calculeze $A \leftarrow AU_2$.

**d.** Ce valori proprii are matricea calculată după paşii **a–c** ? De ce ?

## 6.4 Setul de probleme nr. 4

**1. a.** Calculaţi factorizarile Doolittle şi Crout ale matricei $A = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$.

**b.** Găsiţi o matrice $B$ $2 \times 2$ pentru care $\|B\|_1 \neq \|B\|_\infty$. Arataţi ca pentru toate matricele $B$ $2 \times 2$ avem $\frac{1}{2}\|B\|_\infty \leq \|B\|_1 \leq 2\|B\|_\infty$.

**2.** Fie $R \in \mathbb{R}^{n\times n}$ o matrice superior triunghiulară şi $b \in \mathbb{R}^n$ un vector. Scrieţi algoritmi detaliaţi şi eficienţi care să

**a.** calculeze $\|R\|_F$;

**b.** calculeze $b^T RR^T b$;

**c.** calculeze $b^T R^{-1} R^{-T} b$.

**d.** Care este numărul de operaţii necesar pentru a calcula $x_{12}$, unde $X = R^{-1}$ ? (celelalte elemente ale lui $X$ nu se calculează).

**3. a.** Adaptaţi algoritmul de triangularizare ortogonală pentru matricea $A = \begin{bmatrix} R \\ z^T \end{bmatrix}$, unde $R \in \mathbb{R}^{n\times n}$ este o matrice superior triunghiulară şi $z \in \mathbb{R}^n$ un vector, ambii daţi.

**b.** Scrieţi un algoritm eficient care să rezolve sistemul supradeterminat $Ax = b$ în sensul celor mai mici pătrate, unde $A$ este matricea de la punctul **a** şi $b \in \mathbb{R}^{n+1}$ este un vector dat.

## 6.5 Bibliografie

1. B. Jora, B. Dumitrescu, C. Oară, METODE NUMERICE, UPB, Bucuresti, 1995.
2. G.W. Stewart, INTRODUCTION TO MATRIX COMPUTATIONS, Academic Press, 1973.

**Observaţie.** Pentru a rezolva problemele de la examen referinţa [1] este suficientă. Referinţa [2] poate, de asemenea să, fie de mare ajutor.

## 6.6 Anexă

Prezentăm programele MATLAB pentru implementarea algoritmilor din acest seminar.

```
1:  function [n1,ni,nF]=norme2(A)
2:  %------------------------------------------------------------------------
3:  % Seminar 6 Problema 1 punctul a
4:  % Functia calculeaza norma unu, norma infinit si norma Frobenius a unei
5:  % matrice patratice A, care are elemente nenule numai pe diagonala si pe
6:  % prima coloana.  Cele trei norme sunt notate cu n1, ni respectiv nF.
7:  % Apelul:  [n1,ni,nF] = norme2(A)
8:  %
9:  % Valentin Buta, mai, 2006
10: %------------------------------------------------------------------------
11:
12: [n,m]=size(A);
13: if n~=m
14: error('Matricea nu este patratica!');
15: end
16: for j=2:n
17:     if A[1,j]~=0
18:         error('Matricea nu este inferior triunghiulara!');
19:     end
20: end
21: for i=2:n
22:     for j=i+1:n
23:         if A[i,j]~=0
24:             error('Matricea nu este inferior triunghiulara!');
25:         end
26:         if A[j,i]~=0
27:             error('Matricea are elemente nenule sub diagonala care nu se afla pe prima coloana!')
28:         end
29:     end
30: end
31:
32: n1=0;
33: for i=1:n
34:     n1=n1+abs(A(i,1));
35: end
36: for j=2:n
37:     if abs(A(j,j))>n1
38:         n1=abs(A(j,j));
39:     end
40: end
41: ni=abs(A(1,1));
42: for i=2:n
43:     sigma=abs(A(i,1))+abs(A(i,i));
44:     if sigma>ni
45:         ni=sigma;
46:     end
47: end
```

```
48:  nF=A(1,1)^2;
49:  for i=2:n
50:      nF=nF+A(i,1)^2+A(i,i)^2;
51:  end
52:  nF=sqrt(nF);
```

```
 1:  function [x]=rez_sis(A,b)
 2:  %-------------------------------------------------------------------------_
 3:  % Seminar 6 Problema 1 punctul b
 4:  % Functia calculeaza eficient solutia sistemului Ax=b unde A este o matrice
 5:  % patratica cu elemente nenule numai pe diagonala si pe prima coloana.
 6:  % Apelul:  [x] = rez_sis(A,b)
 7:  %
 8:  % Valentin Buta, mai, 2006
 9:  %-------------------------------------------------------------------------
10:
11:  [n,m]=size(A);
12:  if n~=m
13:      error('Matricea nu este patratica!');
14:  end
15:  if min(size(b))~=1
16:      error('b nu este vector!');
17:  end
18:  if n~=max(size(b))
19:      error('Vectorul b nu are dimensiunile corecte!');
20:  end
21:  for j=2:n
22:      if A(1,j)~=0
23:          error('Matricea nu este inferior triunghiulara!');
24:      end
25:  end
26:  for i=2:n
27:      for j=i+1:n
28:          if A(i,j)~=0
29:              error('Matricea nu este inferior triunghiulara!');
30:          end
31:          if A(j,i)~=0
32:              error('Matricea are elemente nenule sub diagonala care nu se afla pe prima coloana!');
33:          end
34:      end
35:  end
36:
37:  x(1)=b(1)/A(1,1);
38:  for i=2:n
39:      x(i)=(b(i)-A(i,1)*x(1))/A(i,i);
40:  end
```

```
 1:  function [X]=invers1(A)
```

```
2:  %------------------------------------------------------------------------
3:  % Seminar 6 Problema 1 punctul c.  Solutia 1.
4:  % Functia calculeaza eficient inversa matricei A unde A este o matrice
5:  % patratica cu elemente nenule numai pe diagonala si pe prima coloana.
6:  % Inversa matricei A va fi notata cu X.
7:  % Apelul:  [X] = invers(A)
8:  %
9:  % Valentin Buta, mai, 2006
10: %------------------------------------------------------------------------
11:
12: [n,m]=size(A);
13: if n~=m
14:     error('Matricea nu este patratica!');
15: end
16: for j=2:n
17:     if A(1,j)~=0
18:         error('Matricea nu este inferior triunghiulara!');
19:     end
20: end
21: for i=2:n
22:     for j=i+1:n
23:         if A(i,j)~=0
24:             error('Matricea nu este inferior triunghiulara!');
25:         end
26:         if A(j,i)~=0
27:             error('Matricea are elemente nenule sub diagonala care nu se afla pe prima coloana!')
28:         end
29:     end
30: end
31:
32: X(1,1)=1/A(1,1);
33: for i=2:n
34:     X(i,1)=-A(i,1)*X(1,1)/A(i,i);
35: end
36: for j=2:n
37:     for i=1:n
38:         X(i,j)=0;
39:     end
40:     X(j,j)=1/A(j,j);
41: end
```

```
1:  function [X]=invers2(A)
2:  %------------------------------------------------------------------------
3:  % Seminar 6 Problema 1 punctul c.  Solutia 2.
4:  % Functia calculeaza eficient inversa matricei A unde A este o matrice
5:  % patratica cu elemente nenule numai pe diagonala si pe prima coloana.
6:  % Inversa matricei A va fi notata cu X.
7:  % Apelul:  [X] = invers(A)
8:  %
9:  % Valentin Buta, mai, 2006
```

```
10:  %------------------------------------------------------------------------
11:
12:  [n,m]=size(A);
13:  if n~=m
14:      error('Matricea nu este patratica!');
15:  end
16:  for j=2:n
17:      if A(1,j)~=0
18:          error('Matricea nu este inferior triunghiulara!');
19:      end
20:  end
21:  for i=2:n
22:      for j=i+1:n
23:          if A(i,j)~=0
24:              error('Matricea nu este inferior triunghiulara!');
25:          end
26:          if A(j,i)~=0
27:              error('Matricea are elemente nenule sub diagonala care nu se afla pe prima coloana!');
28:          end
29:      end
30:  end
31:
32:  X(1,1)=1/A(1,1);
33:  for i=2:n
34:      X(i,1)=-A(i,1)/(A(1,1)*A(i,i));
35:  end
36:  for j=2:n
37:      for i=1:n
38:          X(i,j)=0;
39:      end
40:      X(j,j)=1/A(j,j);
41:  end
```

```
1:   function [L,U]=Crout_MH(H)
2:   %------------------------------------------------------------------------
3:   % Seminar 6 Problema 2 punctul a.
4:   % Algoritmul calculeaza factorizarea Crout fara pivotare in cazul unei
5:   % matrice patratice inferior Hessenberg cu toate submatricele lider principal
6:   % nesingulare.
7:   % Apelul:  [U,L] = Crout_MH(H)
8:   %
9:   % Valentin Buta, mai, 2006
10:  %------------------------------------------------------------------------
11:
12:  [n,m]=size(H);
13:  if n~=m
14:      error('Matricea nu este patratica!');
15:  end
16:  for i=1:n-2
17:      for j=i+2:n
```

```
18:           if H(i,j)~=0
19:               error('Matricea nu este inferior Hessenberg!');
20:           end
21:       end
22:   end
23:   for i=1:n
24:       for j=1:n
25:           L(i,j)=0;
26:           U(i,j)=0;
27:       end
28:       U(i,i)=1;
29:   end
30:   for i=1:n
31:       L(i,1)=H(i,1);
32:   end
33:   U(1,2)=H(2,1)/L(1,1);
34:   for k=2:n
35:       for i=k:n
36:           L(i,k)=H(i,k)-L(i,k-1)*U(k-1,k);
37:       end
38:       if k~=n
39:           U(k,k+1)=H(k,k+1)/L(k,k);
40:       end
41:   end
```

```
1:    function [B]=U1xA(U1,A)
2:    %-------------------------------------------------------------------------
3:    % Seminar 6 Problema 3 punctul a.
4:    % Algoritmul calculeaza eficient produsul a doua matrice U1 si A, unde U1
5:    % este un de reflector elementar definit de vectorul u1,
6:    % iar A este o matrice m*n oarecare.
7:    % Apelul:  [B]=U1xA(U1,A)
8:    %
9:    % Valentin Buta, mai, 2006
10:   %-------------------------------------------------------------------------
11:
12:   [m,n]=size(A);
13:   beta1=0;
14:   for i=1:m
15:       for j=1:n
16:           B(i,j)=0;
17:       end
18:   end
19:   for i=1:m
20:       beta1=beta1+U1(i,1)*U1(i,1);
21:   end
22:   beta1=beta1/2;
23:   for j=1:n
24:       tau=0;
25:       for i=1:m
```

```
26:           tau=tau+U1(i,1)*A(i,j);
27:       end
28:       tau=tau/beta1;
29:       for i=1:m
30:           B(i,j)=A(i,j)-tau*U1(i,1);
31:       end
32:   end
```

```
 1:   function [x]=Sis_U(U1,b)
 2:   %------------------------------------------------------------------------
 3:   % Seminar 6 Problema 3 punctul b.
 4:   % Algoritmul calculeaza eficient solutia sistemului U1 * x = b, unde U1
 5:   % este o matrice patratica m*m de reflectori elementari, iar b este un
 6:   % vector dat.
 7:   % Apelul:  [x]=Sis_U(U1,b)
 8:   %
 9:   % Valentin Buta, mai, 2006
10:   %------------------------------------------------------------------------
11:
12:   [m,n]=size(U1);
13:   if min(size(b))~=1
14:       error('b nu este vector!');
15:   end
16:   if m~=max(size(b))
17:       error('Vectorul b nu are dimensiunile corecte!');
18:   end
19:   beta1=0;
20:   tau=0;
21:   for i=1:m
22:       beta1=beta1+U1(i,1)^2;
23:       tau=tau+U1(i,1)*b(i);
24:   end
25:   beta1=beta1/2;
26:   tau=tau/beta1;
27:   for i=1:m
28:       x(i)=b(i)-tau*U1(i,1);
29:   end
```