Tema 2 - Metode Numerice
Factorizări. Valori. Cutremure

Responsabili tema :

Mihaela Vasile (mihaela.a.vasile@gmail.com)
Emil Racec (emil.racec@gmail.com)

Dată publicare :

02-04-2013

Dată predare :

19-04-2013

Data ultimei actualizări:

13-04-2013

Task1 - Factorizare ortogonală (10p)

Să se rezolve sistemul de ecuaţii liniare A*x=B prin metoda factorizării ortogonale Gram-Schmidt modificat. Se vor implementa funcţiile:

$$function\ [\mathrm{Q},\ \mathrm{R}] = gramSchmidtMod(A)$$

Semnificaţia parametrilor:

A – matricea de descompus, $A \epsilon R^{nxn}$,
Q - matrice ortogonală $A \epsilon Q^{nxn}$,
R - matrice superior triunghiulară $R \epsilon R^{nxn}$

$$function\ \mathrm{x} = solve(A, b)$$

care rezolvă un sistem de ecuaţii folosind factorizarea QR.

Task2 - Valori şi vectori proprii (10p)

Să se implementeze metoda QR pentru determinarea valorilor şi vectorilor proprii pentru o matrice A (simetrică şi pozitiv definită). Se va implementa funcţia:

$$function\ [\mathrm{l},\ \mathrm{X}] = qrEig(A)$$

Semnificaţia parametrilor:

A – matricea dată, $A \epsilon R^{nxn}$,
l - vectorul valorilor proprii $l \epsilon R^{1xn}$,
X - matricea ale cărei coloane sunt vectori proprii pentru A $X \epsilon R^{nxn}$

Task3 - Valori proprii şi cutremure (30p)

Oamenii de ştiinţă şi inginerii caută diferite modele pentru studiul structurilor aflate sub influenţa unor turbulenţe, de exemplu a cutremurelor.
Să considerăm următorul exemplu pentru o clădire cu trei etaje simulată de un model masă-resort. Masa fiecărui etaj este reprezentată de m_i, iar elasticitatea de $k_i, i \epsilon 1, 2, 3$

$m_3 = 8,000kg$

$k_3 = 1,800kN/m$

$m_2 = 10,000kg$

$k_2 = 2,400kN/m$

$m_1 = 12,000kg$

$k_1 = 3,000kN/m$

Folosind bilanţul forţelor pentru fiecare etaj, se obţine următorul sistem de ecuaţii:

$$\begin{gathered}(\frac{k_1+k_2}{m_1}-\omega_n^2)X_1-\frac{k_2}{m_1}X_2=0\\ -\frac{k_2}{m_2}X_1+(\frac{k_2+k_3}{m_2}-\omega_n^2)X_2-\frac{k_3}{m_2}X_3=0\\ -\frac{k_3}{m_3}X_2+(\frac{k_3}{m_3}-\omega_n^2)X_3=0\end{gathered}$$

unde X_i reprezintă mişcarea pe orizontală iar ω_n reprezintă frecvenţa de rezonanţă(radiani/s). Pentru a exprima frecvenţa de rezonanţă în Hertz se împarte ω_n la 2π.

- Pentru a determina frecvenţele de rezonanţă se vor calcula valorile proprii pentru matricea:

$$[\ \frac{k_1+k_2}{m_1}, -\frac{k_2}{m_1}, 0;\ -\frac{k_2}{m_2}, \frac{k_2+k_3}{m_2}, -\frac{k_3}{m_2};\ 0, -\frac{k_3}{m_3}, \frac{k_3}{m_3}\ ]\ (1)$$

- Pentru a determina amplitudinea mişcării pe orizontală pentru fiecare etaj, se vor calcula vectorii proprii normalizaţi, astfel încat amplitudinea utlimului etaj să fie egală cu 1, pentru matricea de mai sus, corespunzători fiecărei valori proprii.

În exemplul de mai sus valorile proprii sunt :

698.598, 339.478, and 56.924

iar $\omega_n = \frac{\sqrt{\lambda}}{2\pi}$

4.2066 2.9324 1.2008 (Hertz);

iar vectorii proprii normalizaţi :

$[1.69340; -2.10488; 1.00000], [-0.92070; -0.50879; 1.00000], [0.38008; 0.74700; 1.00000].$

Cerinţă

Se citeşte fişierul config.txt care are următorul format :

n - numărul de etaje
$m_{Etaj_1}\ k_{Etaj_1}$
$m_{Etaj_2}\ k_{Etaj_2}$
.....
$m_{Etaj_n}\ k_{Etaj_n}$

n este natural şi ceilalţi parametrii sunt de tip float.

- Să se construiască matricea (1):
  function A = readFile(filename) (10p);

- Să se determine valorile şi vectorii proprii pentru matricea construită, să se transforme frecvenţa în Hertz şi să se normalizeze vectorii proprii: function [l, X] = eigenValuesVectors(A) (10p)
  l - vectorul frecvenţelor exprimate în Hertz
  X - matrice care are drept coloane vectorii proprii normalizaţi.
  Atenţie! Pentru un calcul cat mai exact al rezultatului, doar la acest task este permisă (şi se recomandă) folosirea funcţiei eig din octave pentru determinarea vectorilor şi valorilor proprii unei matrice A oarecare.

- Să se scrie rezultatele obţinute la punctul anterior într-un fişier primit ca parametru:
  function writeResults(l, X, fileName) (5p);
  Formatul fişierului va fi:

n - numărul de etaje
ω_{n1}
vectorul propriu corespunzător valorii proprii λ_1
ω_{n2}
vectorul propriu corespunzător valorii proprii λ_2
.....
ω_{nn}
vectorul propriu corespunzător valorii proprii λ_n

- Să se reprezinte grafic vectorii proprii normalizaţi pentru fiecare valoare proprie:
  function myPlot(X) (5p);
  Hint! Funcţie ajutătoare: subplot

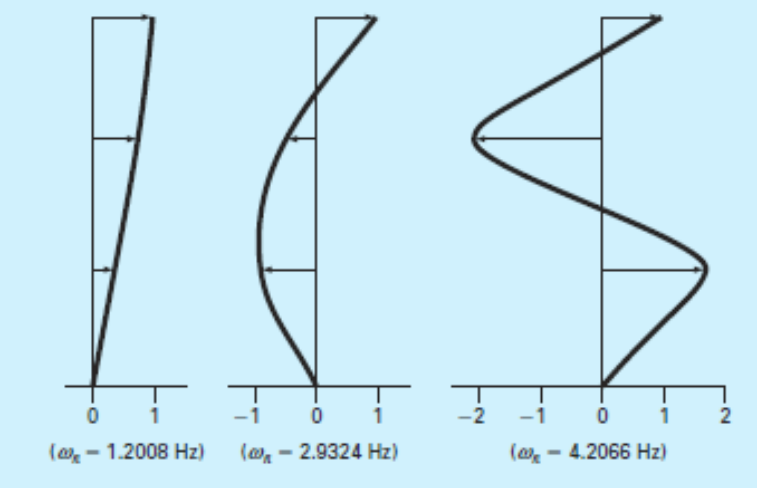

Figure 1: Graficul pentru exemplul de mai sus

Task4 – Predicţie liniară (40p + 10p bonus)

Fie X={ $x_1, x_2, x_3, ...$} un semnal (o secvenţă de valori măsurabile reale). Se doreşte estimarea valorii semnalului la un moment de timp pe baza unei combinaţii liniare a p eşantioane (sample-uri) consecutive anterioare.

$$\hat{x}(n) = - \sum_{i=1}^{p} a_i x(n-i)$$

unde $\hat{x}$(n) este valoarea prezisă, x(n-i) sample-uri anterioare, iar a_i coeficienţii ce trebuie găsiţi. Pentru a estima coeficienţii a_i se pune condiţia ca eroarea pătratică medie să fie minimă.

$$E_p = \sum_{n=1}^{\infty} e(n)^2 = \sum_{n=1}^{\infty} [\, x(n) - \hat{x}(n)]^2 = \sum_{n=1}^{\infty} [\, x(n) - \sum_{i=1}^{p} -a_i x(n-i)]^2$$

Derivarea ultimei formule în raport cu $a_i, i = 1 : p$ conduce la setul de restricţii:

$$\sum_{i=1}^{p} a_i R(i-k) = -R(k),\ k = 1{:}p \quad (1)$$

(mai multe detalii în [1] , paginile 4-6)

unde R este funcţia de autocorelare a lui X, definită astfel:

$$R(k) = \sum_{i=k+1}^{n} x(i)x(i-k),\ n = \text{numărul de sample-uri}\ R(-k) = R(k)$$

Restricţiile (1) sunt echivalente cu sistemul de mai jos care are ca necunoscută vectorul a.

Algoritmul de predicţie constă în rezolvarea unor sisteme de această formă.

$$\begin{bmatrix} R(0) & R(1) & R(2) & \cdots & R(p-1) \\ R(1) & R(0) & R(1) & \cdots & R(p-2) \\ R(2) & R(1) & R(0) & \cdots & R(p-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R(p-1) & R(p-2) & R(p-3) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ \vdots \\ a(p) \end{bmatrix} = - \begin{bmatrix} R(1) \\ R(2) \\ R(3) \\ \vdots \\ R(p) \end{bmatrix}$$

(R*a = -r)

Matricea R este simetrică şi Toeplitz (diagonalele paralele cu diagonala principală conţin elemente identice). Ţinând cont de aceste particularităţi sistemul se poate rezolva mai eficient cu algoritmi specifici cum ar fi algoritmul Levinson-Durbin.

Predicţia liniară este folosită în domeniul semnalelor audio emise de vocea umană. Scopul principal este să reducă rata de biţi (bitrate-ul) a unui fişier audio, deci dimensiunea acestuia. Fie o înregistrare audio obţinută la o frecvenţă de 8 kHz (frecvenţa cu care se achiziţionează sample-urile audio), deci 8000 de eşantioane pe secundă. Pentru vocea umană o astfel de frecvenţă este suficientă pentru că aceasta este limitată la aproximativ 4 kHz.
Putem aplica algoritmul de predicţie liniară pe segmente de 20 ms, aceasta fiind, de regulă, frecvenţa cu care se succed silabele într-un act de vorbire. În decursul pronunţiei unei silabe semnalele emise nu variază foarte mult, acesta fiind motivul pentru care putem estima valorile sample-urilor ca o combinaţie liniară a p eşantioane anterioare. Valoarea lui p este de ordinul zecilor (10-30). Coeficienţii calculaţi de algoritm sunt o reprezentare comprimată a segmentului original şi pot fi salvaţi sau transmişi în locul sample-urilor originale.
La o frecvenţă de 8 kHz există 160 de sample-uri în 20 ms. Pentru p = 20, trebuie stocaţi 20 de coeficienţi plus 20 de valori iniţiale, deci 40 de valori în loc de 160. Am obţinut o compresie de 400%. Algoritmul trebuie aplicat la fiecare 20 ms.

Curba albastră este semnalul original, iar cea roşie semnalul prezis. Deşi pare că diferenţa este semnificativă, la ascultare nu este sesizabil.

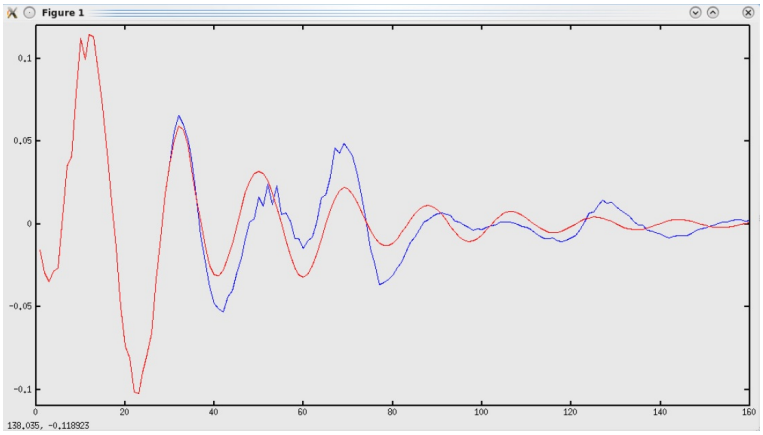

Figure 2: Exemplu predicţie vs. semnal real

Cerinţă

Să se aplice algoritmul de predicţie liniară pe fişierul sample.wav. Funcţia compress va scrie fişierul sample.compressed. Funcţia decompress va primi fişierul comprimat şi va scrie fişierul sample2.wav. Pentru lucrul cu fişiere wav exista funcţiile wavread, wavwrite, iar pentru lucrul cu fişiere binare puteţi folosi funcţiile fopen, fwrite, fread, fclose.
Scopul este să obţineţi o compresie cât mai bună fără a deteriora prea mult calitatea sunetului. Încercaţi mai multe valori ale lui p şi ale dimensiunii intervalului pe care aplicaţi algoritmul de predicţie. Motivaţi în README alegerile făcute şi menţionaţi observaţiile pe care le consideraţi interesante. Pentru a rezolva sistemul de ecuaţii folosiţi metoda implementată pentru taskul 1.

Se acordă bonus pentru rezolvarea sistemului folosind metoda Levinson-Durbin.

Observaţii

- Signaturile funcţiilor sunt :

  compress(original_file_name)- primeşte fişierul original şi crează fişierul comprimat (acelaşi nume, extensia compressed)

  decompress(compressed_file_name) – primeşte fişierul comprimat şi crează unul decomprimat (acelaşi nume concatenat cu ‘2‘, extensia wav)

- Este la libertatea fiecăruia să aleagă formatul de reprezentare a fişierului comprimat. Atâta timp cât funcţia de decompresie foloseşte aceleaşi

convenţii puteţi folosi orice format.

- Fişierul wav creat din fişierul comprimat trebuie să aibă aceeaşi frecvenţă cu fişierul original.
- Pentru autoverificare puteţi compara rezultatele cu cele oferite de funcţiile OCTAVE xcorr şi levinson.
- Menţionaţi în README parametrii aleşi, formatul de reprezentare şi ce metodă aţi folosit pentru rezolvarea sistemului.

Resurse:

`http://www.cs.tut.fi/kurssit/SGN-4010/LP_en.pdf`
`http://en.wikipedia.org/wiki/Linear_prediction`
`http://en.wikipedia.org/wiki/Levinson_recursion`

Precizări generale:

- Tema va fi trimisă pe vmchecker;
- Arhiva trebuie să conţină direct în rădăcină fişierele .m ;
- Arhiva trebuie să aibă formatul .zip
- Este disponibilă o arhiva de testare pentru task-urile 1, 2, 3 şi 4. Pentru testare:

  se copiază fişierele .m în folderul obţinut prin dezarhivarea checkerului

  se rulează *./checker.sh* pentru testarea întregii teme, sau din octave se apelează *check_taski*(), i = 1,4;
- 10p : coding style , README, folosire vectorizări în implementarea temei