Cap.12. Aspecte ale identificarii asistate de calculator a evolutiilor parametrice experimentale descrise numeric

12.1 Generalităţi.

În cercetarea experimentală asistată de calculator există situaţii frecvente în care se realizează prelevarea asistată de calculator a unor date experimentale (de exemplu sub forma evoluţiei în timp a unei mărimi măsurabile) şi care descriu funcţionarea unui sistem fizic. Adesea structura matematică (analitică) a legii de evoluţie a acestor date experimentale este cunoscută. Determinarea valorilor exacte ale constantelor ce intervin în structura legii de evoluţie a datelor experimentale (prin aşa numita identificare) permite caracterizarea sistemului fizic care a generat respectivele date. În literatura de limbă engleză acest proces de identificare este cunoscut sub denumirea de *curve fitting*.

Pentru simplificarea demersului să presupunem o evoluţie experimentală caracterizată prin n puncte de stare (x_i, y_i) şi că legea de evoluţie este de forma $y = f(p_1, p_2,....,p_k, x)$ în care x este parametrul relaţiei, de regulă timpul. Identificarea perfectă a evoluţiei experimentale presupune determinarea valorilor exacte a constantelor $p_1, p_2,....,p_k$ astfel încât toate punctele experimentale să se găsească fizic exact pe curba $y = f(p_1, p_2,....,p_k, x)$. Altfel spus se realizează condiţia matematică:

$$\sum_{i=1}^{n} \left| y_i - f(p_1, p_2, ...p_k, x_i) \right| = 0 \tag{12.1}$$

Sau suma modulelor diferenţelor dintre ordonatele experimentale y_i şi ordonatele rezultate prin aplicarea funcţiei pentru valorile absciselor x_i trebuie sa fie nulă. Relaţia (12.1) este o altfel de scriere a condiţiei:

$$y_i - f(p_1, p_2, ...,p_k, x_i) = 0 \qquad \forall i = 1 \div n \tag{12.2}$$

Criteriul de identificare ideală din relaţia (12.1) este evident aproape imposibil de realizat în practică, datele experimentale sunt adesea afectate de elemente de natură nedeterministă (zgomot, erori de măsurare, etc.). Din acest motiv identificarea practică a evoluţiei experimentale presupune determinarea valorilor constantelor $p_1, p_2,....,p_k$ astfel încât toate punctele experimentale să se găsească fizic <u>cât mai aproape posibil</u> de curba $y = f(p_1, p_2,....,p_k, x)$. În aceste condiţii relaţia (12.2) are membrul drept nenul, sub forma unei erori de identificare ε_i:

(12.3) $$y_i - f(p_1, p_2, ...p_k, x_i) = \varepsilon_i \qquad i = 1 \div n$$

Cea mai bună identificare a evoluţiei experimentale se realizează dacă -în termenii relaţiei (12.1)- eroarea cumulată este minimă, conform cu:

(12.4) $$\varepsilon = \sum_{i=1}^{n} |\varepsilon_i| = \sum_{i=1}^{n} \left| y_i - f(p_1, p_2, ...p_k, x_i) \right| \to min$$

În aceste condiţii identificarea se rezumă la determinarea valorilor constantelor $p_1, p_2, ..., p_k$ ai funcţiei f care asigură îndeplinirea condiţiei (12.4).

12.2 Aspecte ale determinării constantelor de identificare a evoluţiilor experimentale.

În relaţia (12.4) s-a definit deja criteriul de validare a identificării. Determinarea valorilor constantelor $p_1, p_2, ..., p_k$ ai funcţiei f se poate face formal urmând o tehnică foarte simplă: se calculează valorile erorii cumulate ε din (12.4) pentru toate seturile posibile de valori ale constantelor. Se reţine acel set de constante $p_1, p_2, ..., p_k$ pentru care ε are valoarea minimă. Setul respectiv de constante este rezultatul identificării.

Evident, această tehnică simplă, cu succes garantat, are un dezavantaj esenţial: precizia identificării depinde de mărimea numărului de seturi de valori ale constantelor. Fie Np_j numărul de valori posibile pentru constanta p_j. Numărul total de seturi posibile de valori ale constantelor $p_1, p_2, ..., p_k$ deci de valori ε ce trebuie calculate (pentru a se determina valoarea erorii cumulate minime) este dat de:

(12.5) $$N = \prod_{j=1}^{k} Np_j$$

Numărul maxim de valori N (deci o precizie maximă a identificării) se asigură dacă numerele Np_j sunt maxime. Evident, de aici decurge un impediment major: dificultatea implementării acestei tehnici de căutare nesistematică din cauza timpului de execuţie. Să presupunem că $k=4$ şi $Np_j = 1000 \ \forall j = 1 \div k$. Rezultă $N=1000^4$. Presupunând utilizarea unui sistem de calcul ce permite calculul a 10^7 valori ε pe secundă (evident mult exagerată), atunci timpul total necesar pentru identificare este $1000^4/10^7=10^5$ s adică 27,(77) ore, durată evident de neacceptat.

Din acest motiv propunem o metodă de căutare a setului de constante după cum urmează.

Fie pentru constanta p_j definit intervalul de căutare $p_{jmin} < p_j < p_{jmax}$, precum şi un număr rezonabil de mic de valori Np_j ale constantei din acest interval, rezultând un interval de

discretizare $Ip_j = (p_{jmax} - p_{jmin})/Np_j$. Cele Np_j valori ale constantei p_j vor fi: p_{jmin}, $p_{jmin}+Ip_j$, $p_{jmin}+2\cdot Ip_j$,.... $p_{jmin}+Np_j \cdot Ip_j = p_{jmax}$.

Se aplică tehnica de căutare prezentată mai sus (cu aplicarea criteriului (12.4)) rezultând o primă valoare p_j^1 rezultat al identificării. Este evident faptul că după această primă etapă cea mai bună aproximare posibilă a constantei p_j se găseşte undeva în intervalul $p_j^1 - Ip_j < p_j < p_j^1 + Ip_j$. La fel de evident este faptul că se poate acum aplica din nou tehnica de căutare după valoarea minimă a erorii cumulate pe intervalul de căutare $p_{jmin}^1 < p_j < p_{jmax}^1$ (cu $p_{jmin}^1 = p_j^1 - Ip_j$ şi $p_{jmax}^1 = p_j^1 + Ip_j$), cu utilizarea unui nou interval de discretizare $Ip_j^1 = (p_{jmax}^1 - p_{jmin}^1)/Np_j$. Rezultatul acestui nou demers este o nouă valoare (p_j^2) de aproximare (mai exactă decât precedenta) a constantei p_j.

Această tehnică de căutare se poate repeta de un număr finit de ori (fie acesta m), de fiecare dată intervalul de căutare respectiv cel de discretizare reducându-se corespunzător. Cea mai bună aproximare a constantei p_j fiind p_j^m, căutată în intervalul $p_{jmin}^{m-1} < p_j < p_{jmax}^{m-1}$, cu limitele $p_{jmin}^{m-1} = p_j^{m-1} - Ip_j^{m-2}$ respectiv $p_{jmax}^{m-1} = p_j^{m-1} + Ip_j^{m-2}$ şi intervalul de discretizare $Ip_j^{m-1} = (p_{jmax}^{m-1} - p_{jmin}^{m-1})/Np_j$.

Se realizează astfel o discretizare foarte fină a intervalului de căutare a valorii constantei dar progresiv, numai în jurul valorii celei mai probabile a acestuia.

Cu titlu de observaţie, cu excepţia primului pas, tehnica de căutare prezentată anterior poate folosi valori $Np_j=2$. Se obţine de fiecare dată înjumătăţirea intervalului de discretizare.

12. 3 Exemple de identificare a evoluţiilor simulate.

Cea mai facilă cale de a ilustra corectitudinea tehnicii de identificare prezentate este aplicarea acesteia mai întâi pe un set de date experimentale simulate.

Fie descrisă mai întâi identificarea unui set de date (x_i, y_i) provenite dintr-o evoluţie liniară. Se utilizează pentru aplicarea tehnicii de identificare un program Matlab cu listingul expus mai jos:

Program idenlin

```
clear all;close all;tic;
j=1;
for i=0:.01:10
    x(j)=i;y(j)=1.2345*i+5.4321;j=j+1;
end

dim=size(y);l=dim(2);
rafinare=20;
min=99999999999;k=0;eroare=0;
aimin=0;aimax=100;stepai=(aimax-aimin)/rafinare;mem=stepai;
nimin=-100;nimax=100;stepni=(nimax-nimin)/rafinare;
run iteratie1
```

```
for ncicli=1:50
min=99999999999;eroare=0;
aimin=c(1)-stepai;aimax=c(1)+stepai;stepai=stepai/2;
nimin=c(2)-stepni;nimax=c(2)+stepni;stepni=stepni/2;
run iteratie1
end
d(1)=c(1);d(2)=c(2);
for i=1:l;b(i)=d(1)*x(i)+d(2);end;
plot(x,y,'b','LineWidth',1.5);hold on;plot(x,b,'r','LineWidth',1.5);hold on;
%axis([0 1.1 0 500]);
title('EVOLUTIE SIMULATA SI IDENTIFICATA')
xlabel('Evolutie abscisa');ylabel('Evolutie ordonata')
toc;
d(1),d(2)
% close all;rez=b-y;plot(x,rez);
```

Programul apelează în mod repetat o rutină de calcul numită `iteratie1`, redată mai jos:

Program iteratie1

```
for ai=aimin:stepai:aimax;
      for ni=nimin:stepni:nimax;
               for i=1:l;
      b(i)=ai*x(i)+ni;
      eroare=eroare+abs(y(i)-b(i));k=k+1;
   end
   if eroare<min;c(1)=ai;c(2)=ni;min=eroare;
      else end;eroare=0;
end
end
```

Cu observaţia că toate programele prezentate aici vor fi accesibile pentru exerciţiu pe suport electronic, (de exemplu precedentele programe se regăsesc în folderul *interpolare liniara*) se face în continuare comentariul programului pe porţiuni.

Secvenţa de instrucţiuni:

```
clear all;close all;tic;
```

conţine trei instrucţiuni delimitate prin semnul `;`. În general toate instrucţiunile se încheie astfel, excepţie făcând cele la care execuţia este urmată de afişarea rezultatului execuţiei în fereastra de comandă. Prima instrucţiune şterge toate variabilele prealabil definite în mediul Matlab. A doua instrucţiune închide toate reprezentările grafice eventual generate anterior. A treia instrucţiune marchează momentul de timp în care începe rularea programului. Aceasta, împreună cu instrucţiunea `toc;` din finalul programului generează în fereastra de comandă Matlab durata efectivă de execuţie a programului în secunde.

Cu secvenţa:

```
j=1;
for i=0:.01:10
    x(j)=i;y(j)=1.2345*i+5.4321;j=j+1;
end
```

se generează un set de 1000 puncte (x_i, y_i) ce simulează o evoluţie liniară utilizând ecuaţia dreptei prin tăieturi ($y=p_1 \cdot x+p_2$) cu p_1 şi p_2 constantele descrierii. Valorile pantei p_1 şi ale tăieturii p_2 au fost aici alese arbitrar.
În practică secvenţa de mai sus se substitue cu încărcarea setului de puncte din experiment.

Rularea restului programului ar trebui să regăsească valorile celor două constante p_1 şi p_2 (sau valori apropiate acestora).

Fie următoarea secvenţă de program:

```
dim=size(y);l=dim(2);
rafinare=20;
eroare=0;min=99999999999;k=0;
aimin=0;aimax=100;stepai=(aimax-aimin)/rafinare;mem=stepai;
nimin=-100;nimax=100;stepni=(nimax-nimin)/rafinare;
```

în prima instrucţiune se detectează numărul de puncte (x_i, y_i) rezident în variabila `l`. În a doua instrucţiune se precizează numărul Np_j (variabila `rafinare`). În a treia instrucţiune se stabileşte valoarea iniţială a erorii cumulate ε, valoarea iniţială a unei variabile *min* (al cărei rol va fi explicat ulterior), şi a unei variabile contoar `k`.

Ultimele două instrucţiuni stabilesc limitele de căutare p_{jmin} , p_{jmax} pentru fiecare dintre cele două constante, notaţiile pentru limitele acestor constante fiind diferite. Deliberat, pentru a sublinia performanţele de identificare ale programului s-au stabilit limite suficient de largi. Pentru fiecare dintre cele două constante intervalul iniţial de discretizare este notat cu `stepai` repectiv `stepni` şi calculat conform celor deja prezentate anterior (cu formula $Ip_j =(p_{jmax} - p_{jmin})/Np_j$).
Următoarea instrucţiune din programul principal:

```
run iteratie1
```

apelează la execuţie necondiţionată rutina `iteraţie1`:

```
for ai=aimin:stepai:aimax;
      for ni=nimin:stepni:nimax;
               for i=1:l;
      b(i)=ai*x(i)+ni;
      eroare=eroare+abs(y(i)-b(i));k=k+1;
   end
   if eroare<min;c(1)=ai;c(2)=ni;min=eroare;
      else end;eroare=0;
end
end
```

Aceasta realizează cu ajutorul a trei cicluri for-end imbricate calculul erorii cumulate ε conform cu relaţia (12.4), detectează valoarea minimă a acesteia şi reţine în matricea *c* cele două valori identificate ale constantelor funcţiei (p_j^1, $j=1,2$) din aplicarea primului pas de interpolare. Aici, în linia marcată cu caractere îngroşate, se observă şi rolul variabilei `min` care

a fost iniţializată anterior. După execuţia ultimei linii din rutină, execuţia revine în programul apelant şi se continuă cu secvenţa:

```
for ncicli=1:50
min=99999999999;eroare=0;
aimin=c(1)-stepai;aimax=c(1)+stepai;stepai=stepai/2;
nimin=c(2)-stepni;nimax=c(2)+stepni;stepni=stepni/2;
run iteratie1
end
d(1)=c(1);d(2)=c(2);
```

care pregăteşte şi lansează în execuţie ceilalţi paşi de identificare care trebuie să conducă la determinarea valorilor p_j^m ale constantelor, urmând ceilalţi ($m = 50$) paşi de identificare (pentru m s-a folosit notaţia matlab `ncicli`). Se observă faptul că la fiecare nou pas de identificare se recalculează limitele de căutare precum şi intervalul de discretizare, considerând, aşa cum deja s-a anticipat, $Np_j = 2$. De asemeni, la fiecare pas de identificare se lansează din nou rutina `iteraţie 1`. Ultima instrucţiune copie în matricea `d` valorile identificate ale constantelor funcţiei (cea mai bună aproximare).

Ultimul set de instrucţiuni:

```
for i=1:l;b(i)=d(1)*x(i)+d(2);end;
plot(x,y,'b','LineWidth',1.5);hold on;plot(x,b,'r','LineWidth',1.5);
title('EVOLUTIE SIMULATA SI IDENTIFICATA')
xlabel('Evolutie abscisa');ylabel('Evolutie ordonata')
toc;
d(1),d(2)
%close all;rez=b-y;plot(x,rez);
```

realizează în prima linie calculul valorilor funcţiei $f(p_1, p_2, x_i)$ folosind valorile constantelor rezultate din identificare şi depuse în matricea `b`. În linia a doua se realizează reprezentarea fizică a curbei rezultat al interpolării prin punctele de coordonate x_i , $f(p_1, p_2, x_i)$ unite prin segmente de dreaptă cu instrucţiunea `plot(x,y,'b','LineWidth',1.5);`(evoluţia identificată) precum şi a curbei descrise de punctele de x_i, y_i (evoluţia simulată) cu instrucţiunea `(x,b,'r','LineWidth',1.5);`. Reprezentările (prima cu culoarea albastră, a doua cu culoarea roşie) realizează o imagine grafică asupra preciziei identificării, o identificare foarte bună ar trebui să producă suprapunerea celor două curbe (ceea ce este evident în figura 1).

Setul de instrucţiuni:

```
title('EVOLUTIE SIMULATA SI IDENTIFICATA')
xlabel('Evolutie abscisa');ylabel('Evolutie ordonata')
```

realizează generarea titlului, a etichetelor de abscisă şi de ordonată a reprezentării grafice.

Penultima instrucţiune afişează valorile parametrilor identificării, adică valorile pantei şi ale tăieturii.

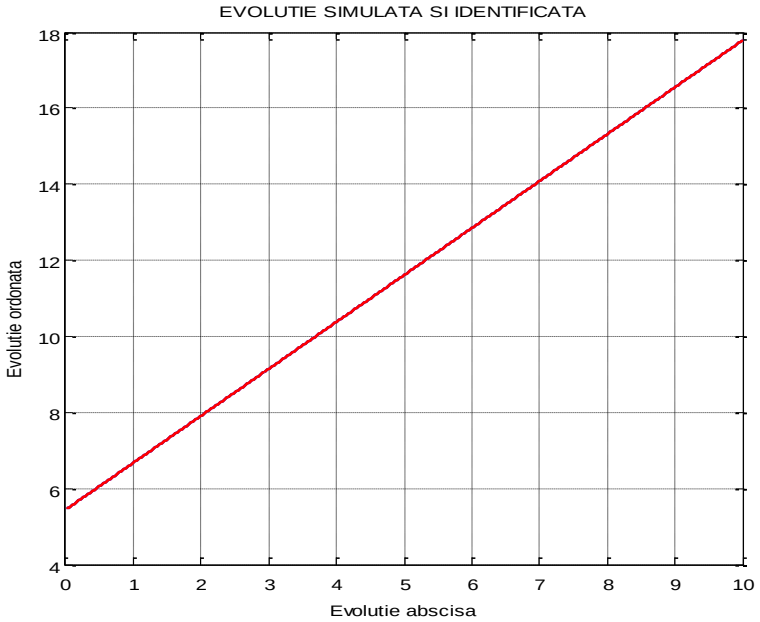

Figura 12.1 *Rezultatul grafic al identificării.*

Ultima instrucţiune: `%close all;rez=b-y;plot(x,rez);` nu se execută din cauza semnului `%`. Se poate importa ca text şi se poate plasa în fereastra de comandă Matlab. Cu comanda enter aceasta se execută şi are ca efect calculul şi reprezentarea reziduului identificării adică o succesiune de puncte de abscisă x_i şi ordonată $f(p_1,p_2,x_i)-y_i$. Reprezentarea reziduului oferă o imagine şi mai corectă asupra preciziei identificării. O identificare perfectă ar trebui să producă o reprezentare a reziduului sub forma unei drepte care se confundă cu abscisa,

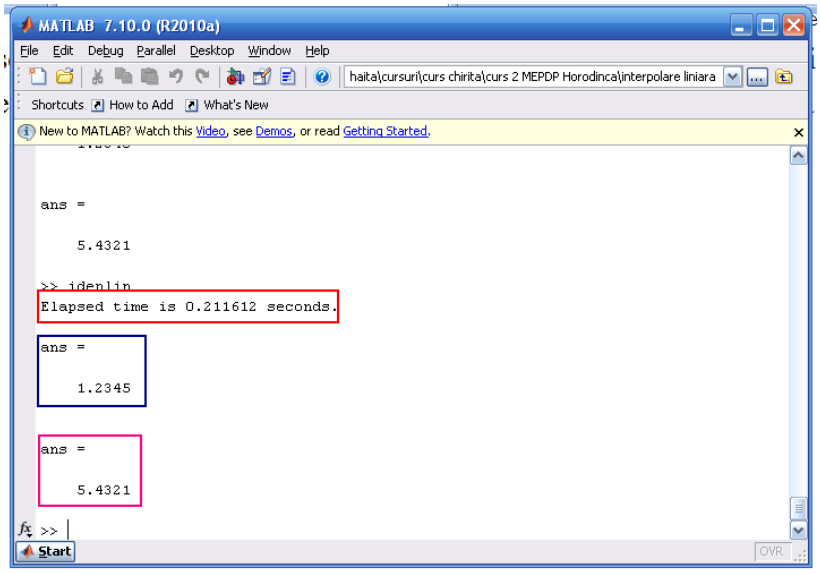

Figura 12.2 *Copie a fereastrei de comandă cu rezultatele rulării programului* `idenlin`.

indiferent de natura funcţiei de identificare.

În figura 12.1 se reproduce rezultatul grafic al rulării programului `idenlin`.

Se observă suprapunerea perfectă a celor două curbe. Calitatea identificării rezultă din datele afişate în fereastra de comandă aşa cum rezultă din copia ecran din figura 12.2.

S-a marcat aici cu primul chenar durata totală de execuţie a identificării (0,2111s), cu al doilea chenar valoarea pantei (prima constantă, p_1 = *1,2345*) respectiv cu al treilea chenar valoarea tăieturii (a doua constantă, p_2 = 5,4321). S-au obţinut exact valorile utilizate la simulare. Mai mult, asa cum s-a precizat deja, plasând în fereastra de comandă instrucţiunea: `close all;rez=b-y;plot(x,rez);` se poate realiza reprezentarea reziduului identificării conform figurii 12.3.

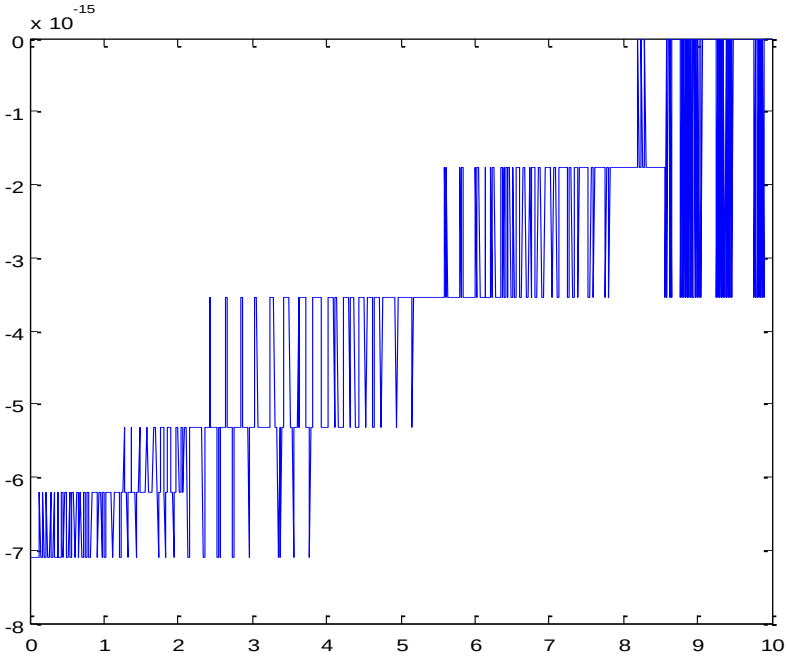

Figura 12.3 *Reprezentare grafică a reziduului identificării*

Rezultatul este mai mult decât spectaculos, reziduul total al identificării nu depăşeşte valoarea de $7x10^{-15}$.

Maniera aceasta de identificare poate fi aplicată evident şi evoluţiilor experimentale descriptibile cu funcţii cu mai mulţi parametri. De exemplu, programul `identparab` şi subrutina `iteratie` (depuse în folderul *interpolare parabola*) simulează un set de puncte experimentale x_i, y_i pe baza unei funcţii parabolice (cu trei constante) şi ulterior realizează identificarea acestei evoluţii simulate, cu determinarea valorii constantelor.

Program identparab

```
close all;clear all;tic;
m=-5.321;n=-3.234;p=1.11111;k=1;
for i=-10:.001:10;
   xr(k)=i;yr(k)=m*i^2+n*i+p;k=k+1;
end
rafinare=10;
mmin=-100;mmax=100;stepm=(mmax-mmin)/rafinare;
nmin=-100;nmax=100;stepn=(nmax-nmin)/rafinare;
pmin=-1000;pmax=1000;stepp=(pmax-pmin)/rafinare;
run iteratie
for l=1:20;
   mmin=m2-stepm;mmax=m2+stepm;stepm=stepm/2;
```

```
   nmin=n2-stepn;nmax=n2+stepn;stepn=stepn/2;
   pmin=p2-stepp;pmax=p2+stepp;stepp=stepp/2;
   run iteratie
end
toc
m2,n2,p2
plot(xr,yr,'b','LineWidth',1.5);hold on;plot(xr,yt,'r','LineWidth',1.5);grid
%reziduu=yr-yt;close all;plot(xr,reziduu)
```

Program iteraţie

```
min=100000000;
for m1=mmin:stepm:mmax;
   for n1=nmin:stepn:nmax;
      for p1=pmin:stepp:pmax;
      eroare=0;
      for i=1:k-1;
         yt(i)=m1*xr(i)^2+n1*xr(i)+p1;
         eroare=eroare+abs(yt(i)-yr(i));
      end
      if eroare<min;min=eroare;m2=m1;n2=n1;p2=p1;else end
   end
end
end
```

Structura programului şi a subrutinei respectă principiile deja enunţate anterior.

Temă 1:

Să se ruleze programul `identparab` şi să se obţină reprezentările grafice aferente, inclusiv a reziduului pentru diferite valori ale celor trei constante ale parabolei respectiv pentru diferite valori *m* ale numărului de paşi de interpolare (în program *m=20*).

Temă 2:

În folderul *interpolare raspuns liber* se găseşte un set de date experimentale ce descriu un răspuns liber periodic amortizat, efectiv prelevate de pe un sistem fizic real. Se va rula programul `identu` si se vor face observaţii legate de structura programului, structura funcţiei de interpolare (care utilizează patru constantei), rezultatele interpolării. Se vor reda pe suport fizic reprezentări grafice legate de rularea programului (evoluţia experimentală şi identificată, zoom pe reprezentare, reprezentarea reziduului). Se va determina valoarea frecvenţei semnalului.