CURS 6: APROXIMAREA FUNCTIILOR PRIN REGRESIE

1 Metoda celor mai mici pătrate

1.1 Formularea problemei. Notaţii

Metoda celor mai mici pătrate (ale erorii) este cea mai uzuală metodă de aproximare a unei dependenţe $y=y(x)$, date tabelar (ca în tabelul 1), printr-o funcţie analitică, care se numeşte ***funcţie*** (de obicei ***polinomială***) ***de regresie***. De aceea, metoda celor mai mici pătrate se mai numeşte şi *regresie*, cu largă aplicabilitate în estimare, statistică şi prelucrarea datelor.

x	x_1	x_2	...	x_n
y	y_1	y_2	...	y_n

Tabel 1 Dependenţa $y=y(x)$ dată prin tabel

Dependenţa $y=y(x)$ se poate reprezenta grafic în planul (x, y), obţinându-se aşa-numitul „*nor de puncte*". De cele mai multe ori, după alura norului de puncte se poate stabili forma exactă a funcţiei de regresie: dacă punctele se distribuie în jurul unei drepte, atunci se alege drept funcţie de regresie un polinom de gradul I (regresie liniară), dacă se distribuie în jurul unei parabole sau a unei parabole cubice, atunci trebuie aleasă o funcţie polinomială de gradul al II-lea, respectiv de gradul al III-lea etc. Dacă forma sugerată a norului de puncte nu corespunde unui caz simplu, atunci se procedează la regresia cu o funcţie polinomială de grad superior, ales arbitrar în primă fază.

Dacă se notează cu $\widehat{y}_i$, $i=\overline{1,n}$, valorile aproximate (estimate) şi cu f_{reg} funcţia generală de regresie, atunci este valabilă relaţia:

(1) $$\hat{y}_i = f_{reg}(x_i)$$

Funcţia f_{reg} depinde de un set de m parametri, fie aceştia a_i, $i=\overline{1,m}$ (în particular, aceştia sunt coeficienţii polinomului de regresie), care se determină din ***minimizarea erorii pătratice*** cumulate dintre valorile reale, y_i, şi valorile aproximate, $\widehat{y}_i$:

(2) $$E(a_1, a_2, ...a_m) \triangleq \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$$

Funcţia de eroare (2) este o expresie a ***dispersiei*** valorilor reale de la curba presupusă. Fiind o sumă de pătrate, această funcţie prezintă un minim pozitiv (în cazul ideal, acesta este 0, corespunzând aproximării exacte), care se obţine din anularea derivatelor sale parţiale în raport cu fiecare dintre parametrii a_i, $i=\overline{1,m}$. Rezultă astfel sistemul de m ecuaţii în necunoscutele a_i, $i=\overline{1,m}$, numit ***sistemul ecuaţiilor normale***:

(3) $$\begin{cases} \dfrac{\partial E(a_1, a_2, ...a_m)}{\partial a_i} = 0, \; i=\overline{1,m} \end{cases}$$

În cazul în care f_{reg} este funcţie polinomială, sistemul (3) este liniar şi compatibil determinat, soluţia lui admiţând o formă generală, care este prezentată în §1.5.

O aplicaţie imediată şi intuitivă a metodei celor mai mici pătrate este trasarea grafică a unei dependenţe de tipul celei din tabelul 1, presupunând că datele respective s-au obţinut în urma măsurării variaţiei variabilei dependente *y* în funcţie de variaţia variabilei independente *x*. Măsurătorile fiind în general afectate de erori, se aplică întâi o metodă de regresie pentru determinarea unei aproximări analitice a dependenţei respective. Prin reprezentarea grafică a norului de puncte de coordonate (x_i,y_i), i=1,2,...n, şi a funcţiei de regresie $\hat{y} = \hat{y}(x)$ în acelaşi sistem de coordonate, se observă că aproximarea prin regresie corespunde trasării graficului „***printre puncte***" (figura 1).

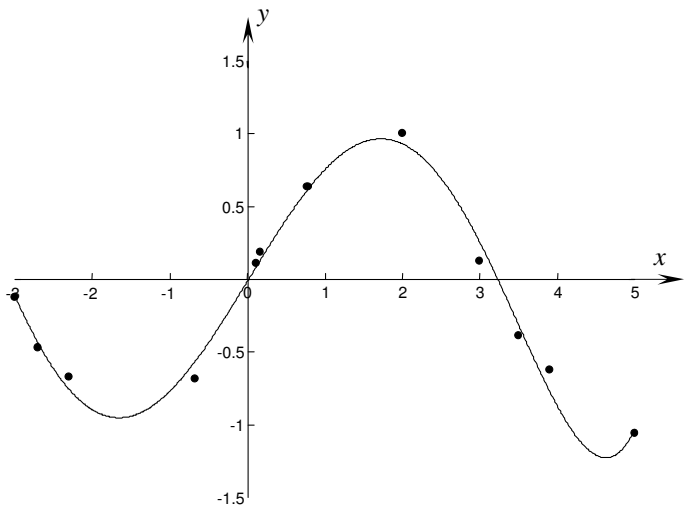

Fig. 1 Trasarea graficului de regresie printre punctele norului de puncte

Algoritmii de regresie sunt practic înşiruirea relaţiilor de calcul al coeficienţilor de regresie. De aceea se prezintă direct implementările Matlab ale celor mai uzuale cazuri de regresie – ***liniară***, ***parabolică*** şi ***exponenţială*** – şi a ***cazului general de regresie polinomială***.

1.2 Aproximarea prin regresie liniară

Regresia liniară constă în aproximarea unui nor de puncte printr-o *dreaptă*; ca atare, relaţia de calcul al valorilor aproximate (estimate), $\hat{y}_i$, $i = \overline{1,n}$, este:

(4) $$\hat{y}_i = a \cdot x_i + b$$

Coeficienţii *a* şi *b* sunt soluţiile sistemului (3), particularizat pentru *m*=2:

(5) $$\begin{cases} a = \dfrac{n \cdot S_{xy} - S_x \cdot S_y}{n \cdot S_{xx} - S_x^2}, \quad b = \dfrac{S_{xx} \cdot S_y - S_x \cdot S_{xy}}{n \cdot S_{xx} - S_x^2}, \end{cases}$$

unde s-au făcut notaţiile: $S_x = \sum_{i=1}^{n} x_i$, $S_{xx} = \sum_{i=1}^{n} x_i^2$, $S_y = \sum_{i=1}^{n} y_i$, $S_{xy} = \sum_{i=1}^{n} x_i y_i$. Relaţiile (5) rezultă din dezvoltarea determinanţilor implicaţi în sistemul (3).

Funcţia Matlab de implementare a metodei de regresie liniară trebuie să primească drept date de intrare cei doi vectori, *x* şi *y*, şi să returneze valorile celor doi coeficienţi de regresie,

a şi b. Funcţia Matlab `cmmp_lin` listată mai jos mai returnează şi şirul de valori aproximate, $\widehat{y_i}$, $i=\overline{1,n}$, în vectorul `y_aprox`, precum şi valoarea (minimizată) a erorii pătratice globale (conform relaţiei (2)) în variabila `E`.

```
function [a,b,y_aprox,E]=cmmp_lin(x,y)
    %REGRESIE LINIARĂ pe un set de perechi de date (x,y);
    %dreapta de regresie are ecuaţia y=a*x+b;
    %x şi y sunt vectori de aceeaşi dimensiune,
    %(recomandabil ca x să fie ordonat crescător/descrescător)

    n=length(x);
    %calculul sumelor
    Sx=sum(x);Sxx=sum(x.^2);Sy=sum(y);Syy=sum(x.*y);
    %calculul numitorului şi numărătorilor din formulele lui a şi b
    d=n*Sxx-Sx^2;da=n*Syy-Sx*Sy;db=Sxx*Sy-Sx*Syy;
    %calculul coeficienţilor de regresie, a şi b
    a=da/d;b=db/d;

    %valorile aproximate
    y_aprox=a*x+b;
    %eroarea globală
    E=sum((y-y_aprox).^2);
```

Ca o observaţie, se poate folosi şi funcţia `det` din biblioteca Matlab, care calculează determinantul unei matrice pătratice.

În exemplul de mai jos se arată o modalitate de verificare a funcţiei `cmmp_lin`.

Exemplul 1: Dându-se şirul de valori `x=[1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6]` (la cazul general, acestea nu trebuie să fie echidistante), se generează vectorul `y` după relaţia `y=2*x+7` şi se apelează funcţia `cmmp_lin`.

```
x=[1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6];y=2*x+7;
[a,b,y_aprox,E]=cmmp_lin(x,y)
```

Rezultatul este conform aşteptărilor:

```
a =
     2

b =
     7

y_aprox =
    9    10    11    12    13    14    15    16    17    18    19
E =
     0
```

Se perturbă apoi valorile obţinute cu formula `y=2*x+7` (de exemplu, cu maxim ±10%), rezultând `y1=[8.1 11 9.9 13.2 11.7 15.4 13.5 17.6 15.3 19.8 17.1]`. Apelul funcţiei `cmmp_lin` cu argumentele de intrare `x` şi `y1` (existente în spaţiul de lucru):

```
[a1,b1,y1_aprox,E1]=cmmp_lin(x,y1)
```

produce următoarele rezultate:

```
a1 =
    1.8757

b1 =
    7.3104

y1_aprox =
  Columns 1 through 7
    9.1860  10.1238  11.0617  11.9995  12.9373  13.8752  14.8130
  Columns 8 through 11
   15.7508  16.6887  17.6265  18.5643

E1 =
    4.5376
```

Evident, valorile coeficienţilor de regresie, *a* şi *b*, vor fi cu atât mai deviate de la valorile 2, respectiv 7, cu cât perturbaţia setului *y* de la dreapta exactă 2*x*+7 va fi mai mare. Rezultatele regresiei se vizualizează grafic prin reprezentarea norului de puncte şi a graficului dreptei de regresie, *ax*+*b* (pentru care se discretizează intervalul de variaţie al variabilei independente, *x*, cu un pas suficient de mic – rezultatul se depune în vectorul `xx` – şi se calculează variabila dependentă, *y*, în fiecare valoare astfel obţinută – rezultatul se depune în vectorul `yy`).

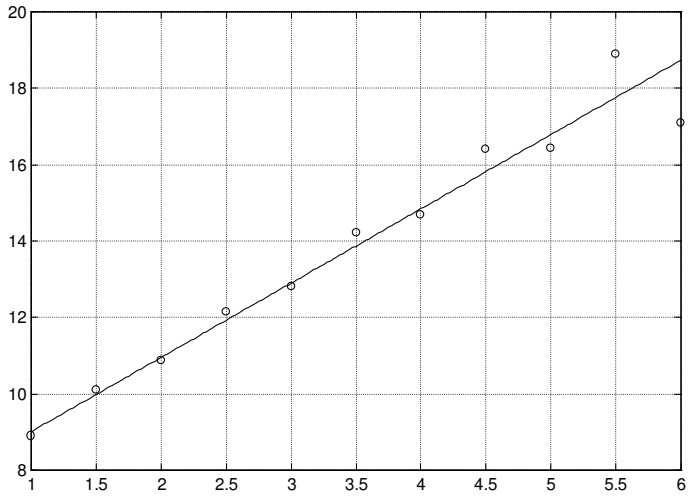

Fig. 2 Aproximarea unui nor de puncte prin regresie liniară

În urma comenzilor Matlab:

```
xx=min(x):(max(x)-min(x))/200:max(x);
yy=a*xx+b;
plot(x,y,'o',xx,yy,'-');grid;
```

se obţine graficul din figura 2.

1.3 Aproximarea prin regresie parabolică

Dacă norul de puncte are alură de *parabolă*, atunci se poate aplica regresia parabolică. Relaţia de calcul al valorilor aproximate, $\widehat{y}_i$, $i=\overline{1,n}$, este în acest caz:

(6) $$\hat{y}_i = a \cdot x_i^2 + b \cdot x_i + c$$

Coeficienţii *a*, *b* şi *c* sunt soluţiile sistemului (2.3), particularizat pentru *m*=3:

(7) $$\begin{cases} a = \dfrac{\Delta_a}{\Delta}, & b = \dfrac{\Delta_b}{\Delta}, & c = \dfrac{\Delta_c}{\Delta}, \end{cases}$$

unde Δ este determinantul sistemului (2.3), iar Δ_a, Δ_b şi Δ_c sunt determinanţii de calcul ai coeficienţilor. Formele lor compacte, precum şi formulele desfăşurate sunt date mai jos.

$$\Delta = \begin{vmatrix} S_{4x} & S_{3x} & S_{xx} \\ S_{3x} & S_{xx} & S_x \\ S_{xx} & S_x & n \end{vmatrix} = nS_{xx}S_{4x} - S_x^2 S_{4x} - nS_{3x}^2 + 2S_x S_{xx} S_{3x} - S_{xx}^3$$

$$\Delta_a = \begin{vmatrix} S_{2xy} & S_{3x} & S_{xx} \\ S_{xy} & S_{xx} & S_x \\ S_y & S_x & n \end{vmatrix} = nS_{xx}S_{2xy} - S_x^2 S_{2xy} - nS_{3x}S_{xy} + S_x S_{3x} S_y + S_x S_{xx} S_{xy} - S_{xx}^2 S_y$$

$$\Delta_b = \begin{vmatrix} S_{4x} & S_{2xy} & S_{xx} \\ S_{3x} & S_{xy} & S_x \\ S_{xx} & S_y & n \end{vmatrix} = nS_{4x}S_{xy} - S_x S_{4x} S_y - nS_{3x}S_{2xy} + S_x S_{xx} S_{2xy} + S_{xx} S_{3x} S_y - S_{xx}^2 S_{xy}$$

$$\Delta_c = \begin{vmatrix} S_{4x} & S_{3x} & S_{2xy} \\ S_{3x} & S_{xx} & S_{xy} \\ S_{xx} & S_x & S_y \end{vmatrix} = S_{xx}S_{4x}S_y - S_x S_{4x} S_{xy} - S_{3x}^2 S_y + S_{xx} S_{3x} S_{xy} + S_x S_{3x} S_{2xy} - S_{xx}^2 S_{2xy}$$

S-au făcut notaţiile: $S_x = \sum_{i=1}^{n} x_i$, $S_{xx} = \sum_{i=1}^{n} x_i^2$, $S_{3x} = \sum_{i=1}^{n} x_i^3$, $S_{4x} = \sum_{i=1}^{n} x_i^4$, $S_y = \sum_{i=1}^{n} y_i$, $S_{xy} = \sum_{i=1}^{n} x_i y_i$, $S_{2xy} = \sum_{i=1}^{n} x_i^2 y_i$.

Metoda de regresie parabolică este implementată printr-o funcţie numită `cmmp_par`, care primeşte aceleaşi argumente de intrare ca şi `cmmp_lin`, funcţia destinată regresiei liniare, şi anume seturile de date *x* şi *y*. Pe lângă coeficienţii de regresie, *a*, *b* şi *c*, şi eroarea globală, funcţia mai returnează şi perechea de vectori (`xx`,`yy`), calculaţi pentru reprezentarea grafică a curbei de regresie (în maniera arătată la regresia liniară, exemplul 2.1).

```
function [a,b,c,E,xx,yy]=cmmp_par(x,y)
    %REGRESIE PARABOLICĂ pe seturile de date (x,y);
    %                                    2
    %curba de regresie are ecuaţia y = a*x + b*x + c;
    %x şi y sunt vectori de aceeaşi dimensiune,
    %(recomandabil ca x să fie ordonat crescător/descrescător)

    n=length(x);
    %calculul sumelor
```

```
    Sx=sum(x);Sxx=sum(x.^2);S3x=sum(x.^3);S4x=sum(x.^4);
    Sy=sum(y);Sxy=sum(x.*y);S2xy=sum((x.^2).*y);
    %calculul determinanţilor
    d=n*Sxx*S4x-Sx^2*S4x-n*S3x^2+2*Sx*Sxx*S3x-Sxx^3;
    da=n*Sxx*S2xy-Sx^2*S2xy-n*S3x*Sxy+Sx*S3x*Sy+ ...
       Sx*Sxx*Sxy-Sxx^2*Sy;
    db=n*S4x*Sxy-Sx*S4x*Sy-n*S3x*S2xy+Sx*Sxx*S2xy+ ...
       Sxx*S3x*Sy-Sxx^2*Sxy;
    dc=Sxx*S4x*Sy-Sx*S4x*Sxy-S3x^2*Sy+Sxx*S3x*Sxy+ ...
       Sx*S3x*S2xy-Sxx^2*S2xy;
    %calculul coeficienţilor de regresie
    a=da/d;
    b=db/d;
    c=dc/d;

    %vectori calculaţi în vederea unei eventuale reprezentări grafice
    xx=min(x):(max(x)-min(x))/200:max(x);
    yy=a*xx.^2+b*xx+c;
    %eroarea globală
    E=sum((y-a*x.^2-b*x-c).^2);
```

A se observa modul de plasare pe mai multe rânduri a unei formule neobişnuit de lungi (terminarea fiecărui rând cu caracterele „...”).

Exemplul 2: Dându-se vectorul `x=[0 0.3 1 1.4 1.6 2 2.1 2.4 2.8 3 3.5 3.7 3.9 4]` şi vectorul `y=x.^2-4*x-10`, apelul funcţiei `cmmp_par` cu returnarea numai a primelor patru argumente de ieşire:

```
[a,b,c,E]=cmmp_par(x,y)
```

va avea drept ecou la ecran:

```
a =
    1.0000

b =
   -4.0000

c =
  -10.0000
E =
  4.3108e-024
```

Rezultatele confirmă aşteptările; câteva comentarii se impun. Se observă că valoarea erorii globale este foarte mică, practic 0; totuşi, ea nu a fost afişată ca 0, ceea ce înseamnă că *nu este exact* 0, aşa cum s-a obţinut în cazul regresiei liniare pe un set de date perfect „curat”, neperturbat (exemplul 2.1). Nici coeficienţii de regresie nu corespund *exact* valorilor curbei după care a fost generat vectorul *y*; aceasta se constată reafişând coeficienţii în formatul de afişare cu 15 zecimale:

```
format long
a
```

```
a =
   0.99999999999995

b

b =
  -3.99999999999985

c

c =
  -9.99999999999946
```

Rezultatele confirmă o intuiţie firească: erorile de calcul cresc cu cât ordinul regresiei creşte (cu cât numărul coeficienţilor de regresie este mai mare).

Exemplul 3: Regresia parabolică se aplică acum pe seturi de date legate printr-o lege liniară; de exemplu, fie cele din exemplul 2.1: `x=[1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6]` şi `y=2*x+7`.

Rezultatul aşteptat în urma apelului funcţiei `cmmp_par` este ca parabola de regresie să aibă coeficientul *a* aproximativ nul (parabola se reduce la o dreaptă). Într-adevăr, comanda:

```
[a,b,c,E]=cmmp_par(x,y)
```

are ca rezultat:

```
a =
     0

b =
     2

c =
     7

E =
     0
```

Rezultatul este de această dată exact, coeficienţii sunt determinaţi fără eroare.

Rămânând în continuare la exemplul 2.1 şi apelând funcţia `cmmp_par` pentru setul *y* perturbat, `y1=[8.1 11 9.9 13.2 11.7 15.4 13.5 17.6 15.3 19.8 17.1]`:

```
[a1,b1,c1,E1]=cmmp_par(x,y1)
```

se obţine:

```
a1 =
   -0.1305

b1 =
    2.8592

c1 =
    5.7909
```

```
E1 =
    21.4862
```

Eroarea globală obţinută prin regresie parabolică (21.4862) este mai mare decât cea prin regresie liniară (4.5376, din exemplul 2.1), ceea ce arată că respectivul nor de puncte *se aproximează mai bine printr-o dreaptă decât printr-o parabolă*.

De aici se desprinde o concluzie generală: în cazul aplicării regresiei polinomiale unui set dat de date, mai întâi trebuie determinat *ordinul* regresiei care asigură eroarea minimă. Se va reveni la această problemă când se va discuta cazul general de regresie polinomială (§2.1.5).

1.4 Aproximarea prin regresie exponenţială

Regresia exponenţială se aplică atunci când variabila dependentă are valori de semn constant (are fie numai valori pozitive, fie numai valori negative, şi atunci se consideră $|y|$ ca variabilă dependentă). Fără a reduce generalitatea se poate, deci, presupune că *y* conţine numai valori pozitive. Norul de puncte se aproximează printr-o *curbă exponenţială*; valorile estimate, $\hat{y}_i > 0,\ i = \overline{1,n}$, determinându-se cu:

(8) $$\hat{y}_i = C \cdot e^{\alpha \cdot x_i},\ C > 0$$

Prin logaritmarea relaţiei (2.8) şi introducerea schimbării de variabilă $\hat{z} \triangleq \ln(\hat{y}),\ \hat{z} > 0$, se obţine:

(9) $$\hat{z}_i = \underbrace{\alpha}_{A} \cdot x_i + \underbrace{\ln(C)}_{B},$$

care este un model de ***regresie liniară*** pentru seturile de date *x* şi $z \triangleq \ln(y)$, cu coeficienţii $A = \alpha$ şi $B = \ln(C) > 0$. Aceştia se pot, deci, determina adaptând formulele (2.5):

(10) $$\begin{cases} A = \dfrac{n \cdot S_{xz} - S_x \cdot S_z}{n \cdot S_{xx} - S_x^2}, \quad B = \dfrac{S_{xx} \cdot S_z - S_x \cdot S_{xz}}{n \cdot S_{xx} - S_x^2}, \end{cases}$$

unde s-au făcut notaţiile: $S_x = \sum_{i=1}^{n} x_i$, $S_{xx} = \sum_{i=1}^{n} x_i^2$, $S_z = \sum_{i=1}^{n} z_i = \sum_{i=1}^{n} \ln(y_i) = \ln\left(\prod_{i=1}^{n} y_i\right)$,

$S_{xz} = \sum_{i=1}^{n} x_i z_i = \sum_{i=1}^{n} x_i \ln(y_i)$. Relaţia (2.8) se poate rescrie sub forma:

$$\hat{y}_i = e^{A \cdot x_i + B},\ B > 0$$

Având în vedere reducerea de mai sus, funcţia Matlab pentru regresia exponenţială, numită `cmmp_exp`, constă în esenţă în apelarea funcţiei `cmmp_lin`, de regresie liniară. Semnificaţia argumentelor de ieşire ale funcţiei `cmmp_exp` este aceeaşi ca în cazul funcţiei `cmmp_par`, de regresie parabolică; ea este listată şi comentată mai jos.

```
function [A,B,E,xx,yy]=cmmp_exp(x,y)
    %REGRESIE după o CURBA EXPONENTIALĂ a unui set de date (x,y);
    %                                   A*x+B
    %curba de regresie are ecuaţia  y=e;
```

```
%x şi y sunt vectori de aceeaşi dimensiune;
%(recomandabil ca x să fie ordonat crescător/descrescător);
%y conţine valori pozitive

%regresia de acest tip se reduce la regresia liniară prin
%logaritmarea naturală a valorilor din y
z=log(y);
[A,B,E1,y_aprox1]=cmmp_lin(x,z);

%şiruri de valori calculate în vederea unei eventuale reprezentări grafice
xx=min(x):(max(x)-min(x))/200:max(x);
yy=exp(A*xx+B);
%eroarea globală
E=sum((y-exp(A*x+B)).^2);
```

Funcţia returnează coeficienţii *A* şi *B* (o modificare minoră, folosind relaţiile introduse în formula (2.9), este necesară pentru a întoarce valorile coeficienţilor *C* şi α).

Exemplul 4: Pornind de la seturile de date `x=[-3 -2 0 0.5 1.5 3 4]` şi `y =[0.2725 0.4493 1.2214 1.5683 2.5857 5.4739 9.0250]` (generat cu relaţia $y = e^{0.5 \cdot x + 0.2}$: `y=exp(0.5*x+0.2)`), apelul funcţiei `cmmp_exp`: cu primele trei argumente de ieşire:

```
[A,B,E]=cmmp_exp(x,y)
```

furnizează rezultatul aşteptat:

```
A =
    0.5000

B =
    0.2000

E =
  1.6424e-030
```

Eroarea globală nu este exact 0, din cauza erorilor introduse de logaritmarea iniţială.

Exemplul 5: Se perturbă acum uşor valorile lui `y` din exemplul 2.4 cu maxim ±25%, obţinându-se şirul `y1=[0.2998 0.4044 1.3435 1.4115 2.8443 4.9266 9.9275 8.9768]`.

Făcând abstracţie de modul de generare a datelor, alura norului de puncte poate corespunde la fel de bine şi unei curbe exponenţiale, dar şi unei parabole. Se vor încerca ambele tipuri de regresie – exponenţială şi parabolică – şi se vor compara erorile globale.

Se apelează întâi funcţia `cmmp_exp`, pentru regresia exponenţială:

```
[A,B,E1]=cmmp_exp(x,y1)
```

obţinându-se:

```
A =
    0.4641

B =
    0.1791
```

```
E1 =
   69.5620
```

Deci norul de puncte poate fi aproximat prin $\hat{y}_{exp} = e^{0.4641 \cdot x + 0.1791}$, cu eroarea globală de 69.562.

Apelul funcţiei `cmmp_par`, pentru regresie parabolică:

```
[a,b,c,E2]=cmmp_par(x,y1)
```

conduce la:

```
a =
    0.2457

b =
    0.9255

c =
    1.1475

E2 =
   66.1489
```

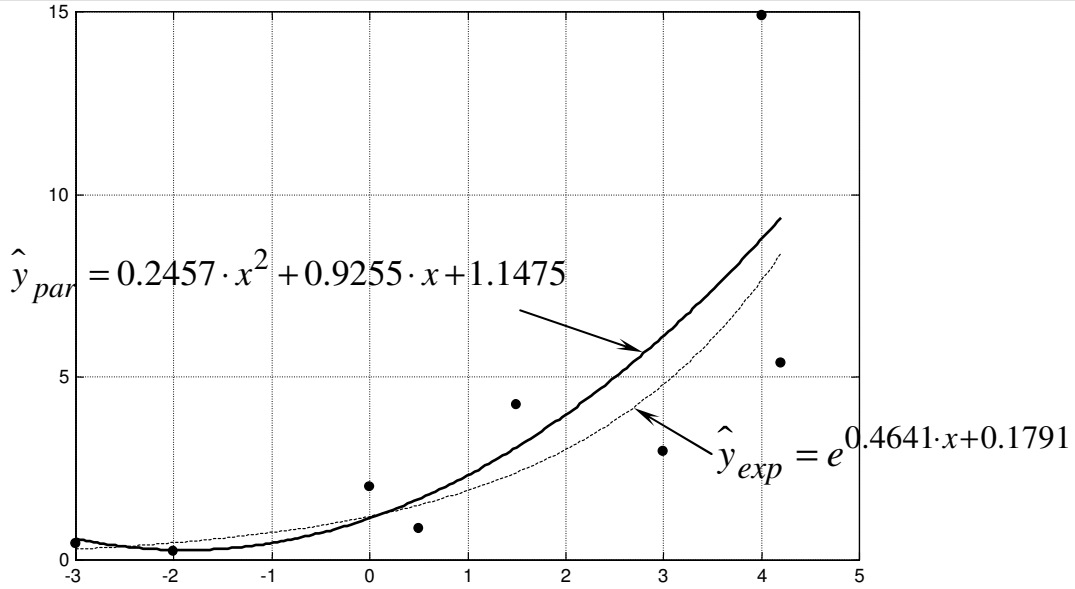

Fig. 3 Aproximarea prin regresie parabolică şi exponenţială a unui nor de puncte

Deci norul de puncte mai admite şi aproximarea $\hat{y}_{par} = 0.2457 \cdot x^2 + 0.9255 \cdot x + 1.1475$, cu eroarea globală de 66.1489.

Se observă că eroarea globală are valori mari în ambele cazuri, dar că este mai mică în cazul regresiei parabolice (66.1489) decât în cazul regresiei exponenţiale (69.562). Deci, deşi datele iniţiale, neperturbate, corespundeau unei dependenţe exponenţiale, perturbarea lor a fost suficient de semnificativă pentru a modifica *tipul* dependenţei: datele perturbate ale variabilei dependente, *y*, respectă mai degrabă o lege parabolică în raport cu variabila independentă, *x*.

Reprezentarea norului de puncte şi trasarea graficelor curbelor de regresie obţinute mai sus în acelaşi sistem de coordonate se realizează prin următorul bloc de comenzi:

```
[A,B,E1,xx1,yy1]=cmmp_exp(x,y1);
```

```
[a,b,c,E2,xx2,yy2]=cmmp_par(x,y1);
plot(x,y,'ko',xx1,yy1,'k--',xx2,yy2,'k-');grid;
```

care produce rezultatele din figura 2.3.

1.5 Cazul general de regresie polinomială

Dacă alura norului de puncte nu corespunde niciunuia din cazurile de mai sus, atunci se efectuează regresia după o curbă polinomială de grad cel puţin 3. Notând cu *m* numărul coeficienţilor de regresie – care este, deci, cel puţin 4 – valorile aproximate, $\widehat{y}_i$, $i=\overline{1,n}$, sunt date de relaţia generală:

(11) $$\hat{y}_i = a_m \cdot x_i^{m-1} + a_{m-1} \cdot x_i^{m-2} + ... + a_2 \cdot x_i + a_1$$

Să observăm că există o cerinţă generală: pentru a face regresie polinomială cu *m* parametri este necesar să se dispună de cel puţin *m* perechi de date (x_i,y_i). Deci trebuie îndeplinită condiţia $n \geq m$. Notând cu $\underline{c} = \left[a_i\right]_{i=\overline{1,m}}$ vectorul coeficienţilor de regresie, sistemul (2.3) este liniar pătratic; el se poate pune sub forma compactă:

(2.3') $$S \cdot \underline{c} = \underline{t}$$

unde $S = \left[s_{ji}\right]_{j=\overline{1,m},\ i=\overline{1,m}}$ este matricea sistemului, iar $\underline{t} = \left[t_j\right]_{j=\overline{1,m}}$ este vectorul termenilor liberi, ale căror elemente se calculează după relaţiile:

(12) $$s_{ji} = \sum_{k=1}^{n} x_k^{j+i-2},\ t_j = \sum_{k=1}^{n} y_k \cdot x_k^{j-1}$$

Matricea *S* fiind nesingulară, sistemul (2.3') este compatibil determinat şi are soluţia:

(13) $$\underline{c} = S^{-1} \cdot \underline{t}$$

Mai jos este prezentată funcţia Matlab `cmmp_gen`, care reprezintă o implementare posibilă a cazului general de regresie polinomială. Spre deosebire de funcţiile de regresie prezentate până acum, această funcţie mai primeşte încă un argument de intrare, şi anume pe `m`, numărul de parametri de regresie (pe lângă vectorii `x` şi `y`). Pentru simplitate, formula (2.13) s-a realizat prin apelul funcţiei `inv` din biblioteca Matlab (implementarea de algoritmi numerici pentru calculul inverselor matriciale va fi discutată pe larg mai târziu în această lucrare). Primul argument de ieşire al funcţiei de mai jos, `coef`, este vectorul coeficienţilor polinomului de regresie, în ordine *crescătoare* a puterilor variabilei. Celelalte argumente de ieşire au aceeaşi semnificaţie ca în cazul celorlalte funcţii.

```
function [coef,E,y_aprox,xx,yy]=cmmp_gen(x,y,m)
    %metoda CMMP de aproximare a dependenţei  y=f(x)
    %printr-un POLINOM DE GRADUL m-1 (altfel spus, regresie cu m parametri);
    %coeficienţii polinomului de regresie se returnează în  coef,
    %care este de dimensiune m, în ordine crescătoare a puterilor variabilei;
    %x şi y au aceeaşi dimensiune;
    %(recomandabil ca x să fie ordonat crescător/descrescător);
    %n>m (numărul de perechi de date trebuie sa fie mai mare
    %       decât numărul parametrilor de regresie)
```

```
n=length(x);
%calculul elementelor matricei şi al termenilor liberi
for j=1:m,
   t(j)=0;
   for i=1:m,
      s(j,i)=0;
      for k=1:n,
         s(j,i)=s(j,i)+x(k)^(j+i-2);
      end;
   end;
   for k=1:n,
      t(j)=t(j)+y(k)*(x(k)^(j-1));
   end;
end;

coef=inv(s)*t';
coef=coef';%operaţie de transpunere pentru ca vectorul coef să rezulte de tip linie

%calculul valorilor aproximate ale variabilei dependente
for k=1:n,
   y_aprox(k)=0;
   for i=1:m,
      y_aprox(k)=y_aprox(k)+coef(i)*(x(k)^(i-1));
   end;
end;
%eroarea globală
E=(y-y_aprox)*(y-y_aprox)';%echivalent cu  E=(y-y_aprox).^2

%generarea vectorilor de valori pentru o eventuală reprezentare grafică
xx=min(x):(max(x)-min(x))/200:max(x);
for k=1:length(xx),
   yy(k)=0;
   for i=1:m,
      yy(k)=yy(k)+coef(i)*(xx(k)^(i-1));
   end;
end;
```

Observaţie: Dacă se doreşte obţinerea coeficienţilor de regresie în ordine *descrescătoare* a puterilor variabilei (ceea ce corespunde manierei Matlab de returnare a rezultatelor de acest tip), atunci se poate folosi funcţia Matlab `fliplr` (engl. *flip left-right*, oglindire stânga-dreapta; alte funcţii din aceeaşi clasă sunt `flipud` – engl. *flip up-down* – şi `rot90`). Avantajul acestei variante este că `y_aprox` şi `yy` se pot calcula mai elegant, şi anume apelând o altă funcţie din bibliotecă, `polyval`, de calcul al valorii unui polinom dat prin vectorul coeficienţilor în ordine descrescătoare a puterilor variabilei într-o valoare dată (a se vedea şi §1.3.3.8, exerciţiul propus 4). Ultimele trei blocuri de comenzi din funcţia `cmmp_gen` se rescriu sub forma:

```
    coef=inv(s)*t';
    coef=fliplr(coef');

    %calculul valorilor aproximate ale variabilei dependente
    for k=1:n,
       y_aprox(k)=polyval(coef,x(k));
    end;
    %eroarea globală
    E=(y-y_aprox)*(y-y_aprox)';

    %generarea vectorilor de valori pentru o eventuală reprezentare grafică
    xx=min(x):(max(x)-min(x))/200:max(x);
    for k=1:length(xx),
       yy(k)=polyval(coef,xx(k));
    end;
```

O problemă anticipată în paragrafele anterioare constă în modul ***cum se stabileşte gradul polinomului de regresie***, altfel spus numărul parametrilor de regresie, *m*. Având în vedere condiţia *m*≤*n*, în primă fază se poate considera *m*=*n*, adică efectuarea unei regresii de *grad maxim* admis de seturile respective de date. Este posibil ca această regresie să nu fie necesară, caz în care unul sau mai mulţi coeficienţi polinomiali, începând cu cel de rang maxim, *m*-1, să fie aproape nuli sau, în orice caz, mult mai mici în valoare absolută decât ceilalţi. În acest caz, o a doua iteraţie de regresie se va face cu gradul egal cu rangul primului coeficient nenul (în ordine descrescătoare a puterilor variabilei).

Exemplul 6: Pentru seturile de date `x=[3 3.3 3.7 3.9 4.7 4.8 4.9 5 5.25]` şi `y=[3 5 10 14 43 49 56 63 84]` (de lungime *n*=9) se efectuează întâi regresia de gradul *n*-1, adică de gradul 8:

```
[coef,E]=cmmp_gen(x,y,length(x)-1)
```

Rezultatul apelului de mai sus constă în primul rând în afişarea unui mesaj de avertizare asupra apropierii de singularitate a matricei sistemului (a cărei inversă se calculează), numărul ei de condiţionare – calculat în `RCOND` – fiind mic (comparabil cu epsilon maşină). Interpretorul avertizează astfel că este posibil ca acurateţea rezultatelor să nu fie bună.

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 7.158432e-024
```

Setările implicite din mediul Matlab prevăd ca avertizările de acest gen să fie ignorate de către interpretor şi să nu oprească execuţia programelor; astfel, după mesajul de eroare se afişează rezultatele de mai jos:

```
coef =
  1.0e+004 *
  Columns 1 through 7
   -4.2489   6.7751  -4.4086   1.4564  -0.2303   0.0038   0.0044
  Columns 8 through 9
   -0.0006   0.0000

E =
  816.4324
```

Se observă că ultimele două elemente, de rang 7 şi 8, din vectorul `coef` sunt mult mai

mici decât restul coeficienţilor. Deci procedura de regresie se poate repeta cu gradul polinomului de regresie micşorat cu două unităţi (gradul 6):

```
[coef,E]=cmmp_gen(x,y,length(x)-3)
```

Rezultatele sunt din nou însoţite de acelaşi mesaj de eroare, dar eroarea globală are o valoare mult mai mică:

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.281806e-019

coef =
  1.0e+004 *
    0.8818  -1.2970  0.7868  -0.2519  0.0449  -0.0042  0.0002

E =
    0.2157
```

Dacă se scade în continuare gradul polinomului de regresie la 5:

```
[coef,E]=cmmp_gen(x,y,length(x)-4)
```

se observă că valoarea erorii globale creşte (rămânând de acelaşi ordin de mărime), dar nu se mai afişează mesajul de eroare:

```
coef =
   17.9262   -7.4289   -3.7338    3.0191   -0.7671    0.0888

E =
    0.3205
```

Dacă nivelul maxim admisibil al erorii de aproximare o permite, scăderea gradului regresiei poate continua în aceeşi manieră, ţinând cont că rangul maxim este ocupat de un coeficient cu un ordin de mărime mai mic în modul decât ceilalţi (0.0888).

Dacă nu se continuă scăderea gradului, se pune problema care dintre ultimele două rezultate se reţine drept rezultat final. În general, trebuie evitate rezultatele de încredere diminuată (însoţite de avertizări de genul celor de mai sus); în cazul de faţă, ultimul rezultat ar trebui ales, cu atât mai mult cu cât eroarea de aproximare faţă de cazul imediat superior nu este cu mult mai mare. Ca atare, seturile de date `x` şi `y` se găsesc aproximativ în dependenţa:

$$y \cong \hat{y} = 0.0888x^5 - 0.7671x^4 + 3.0191x^3 - 3.7338x^2 - 7.4289x + 17.9262$$

În biblioteca Matlab există o funcţie care implementează cazul general de regresie polinomială, ea se numeşte `polyfit`. Această funcţie primeşte ca date de intrare cele două seturi de date, precum şi *gradul* polinomului de regresie (funcţia `cmmp_gen`, scrisă mai sus, primeşte *numărul* coeficienţilor polinomiali) şi returnează coeficienţii polinomului în ordine *descrescătoare* a puterilor variabilei (în timp ce `cmmp_gen` îi returnează în ordine *crescătoare*).

De pildă, pentru aceleaşi seturi de date `x=[2 4 5 7 8]` şi `y=[1 1.4 2.3 1.2 0.4]`, pentru ca cele două funcţii să calculeze acelaşi polinom, de exemplu de gradul 3, ele se vor apela cu:

```
[coef1]=polyfit(x,y,4);
[coef2]=cmmp_gen(x,y,5);
```

şi vor furniza rezultatele `coef1=[0.0406 -0.8733 6.4739 -19.0567 19.5556]` şi `coef2=[19.5556 -19.0567 6.4739 -0.8733 0.0406]`, care exprimă acelaşi polinom de regresie, $0.0406 \cdot x^4 - 0.8733 \cdot x^3 + 6.4739 \cdot x^2 - 19.0567 \cdot x + 19.5556$.