CURS 7: DETERMINAREA FUNCTIILOR PRIN INTERPOLARE

1 Interpolarea funcţiilor

După cum s-a văzut, funcţia de regresie permite estimarea valorii variabilei dependente ***în orice punct din*** domeniul de variaţie al variabilei independente – operaţie ce se numeşte generic ***interpolare*** – aşa cum ea poate fi folosită şi la calculul de valori ***dinafara*** acestui domeniu – ceea ce se numeşte ***extrapolare***.

Rămânând în cadrul funcţiilor de o *singură variabilă reală*, interpolarea porneşte de la aceleaşi date (tabelate) de intrare, (x_i, y_i), i=1,2,...n, ca şi metoda celor mai mici pătrate şi are acelaşi scop: determinarea unei funcţii analitice care aproximează dependenţa exprimată prin tabelul de intrare. Deosebirea faţă de regresie este că, de această dată, funcţia de interpolare este constrânsă să treacă ***prin punctele*** (x_i, y_i) şi ***nu printre*** ele, aşa cum rezulta în cazul regresiei. Aceste puncte se numesc ***noduri*** (de interpolare).

În general se foloseşte interpolarea cu funcţii *polinomiale*. Acest paragraf este dedicat implementării în Matlab a celor mai uzuale metode de interpolare unidimensională, prin funcţii utilizator sau folosind funcţiile disponibile în biblioteca mediului.

1.1 Formula de interpolare a lui Newton

Dându-se n noduri de interpolare, (x_i, y_i), i=1,2,...n, atunci valoarea variabilei y într-un punct intermediar x este dată de ***polinomul de interpolare al lui Newton***, de gradul n-1, notat cu $N(x)$:

$$(1) \qquad y = y_1 + (x - x_1)\cdot \Delta y_1 + (x - x_1)(x - x_2)\cdot \Delta^2 y_1 + ... + (x - x_1)...(x - x_{n-1})\cdot \Delta^{n-1} y_1,$$

unde:

$$\Delta y_1 = \frac{y_2 - y_1}{x_2 - x_1},\ \Delta^2 y_1 = \frac{\Delta y_2 - \Delta y_1}{x_3 - x_1},\ \ldots\ \Delta^{n-1} y_1 = \frac{\Delta^{n-2} y_2 - \Delta^{n-2} y_1}{x_n - x_1}$$

sunt *diferenţele finite* succesive calculate pentru variabila y, a căror formulă recursivă generală este:

$$(2) \qquad \Delta^j y_i = \frac{\Delta^{j-1} y_{i+1} - \Delta^{j-1} y_i}{x_{i+j} - x_i}, \quad j = \overline{1, n-1}, \quad i = \overline{j, n-1},$$

unde s-a notat $\Delta^0 y_i \equiv y_i$. Relaţia (2) conduce la ideea de a calcula aceste diferenţe ca elementele triunghiului inferior al unei matrice de n-1 linii şi n-1 coloane, după cum urmează:

$$D=\begin{bmatrix} \Delta y_1 & & & & & \\ \Delta y_2 & \Delta^2 y_1 & & & & \\ \Delta y_3 & \Delta^2 y_2 & \Delta^3 y_1 & & & \\ . & . & \dots & \dots & & \\ . & . & \Delta^j y_i & \dots & \Delta^j y_1 & \\ . & . & \dots & \dots & \dots & \\ \Delta y_{n-1} & \Delta^2 y_{n-2} & \dots & \Delta^j y_{n-j} & \dots & \Delta^{n-1} y_1 \end{bmatrix}$$

Conform relaţiei (1), din matricea D interesează doar elementele de pe diagonala principală (încercuite cu linie punctată).

Elementele primei coloane a matricei D se calculează exclusiv pe baza nodurilor de interpolare (x_i, y_i), i=1,2,...n, pentru restul elementelor se foloseşte formula recursivă (2), care se rescrie în spiritul observaţiilor de mai jos.

Elementul generic $\Delta^j y_i$ se află pe coloana $l=j$ şi linia $k=i+j-1$ ale matricei D:

$$(3) \qquad D_{kl}=\Delta^j y_i, \quad l=\overline{1,n-1}, \quad k=\overline{l,n-1}$$

Se exprimă perechea de indici (i, j) în funcţie de perechea de indici (k, l):

$j=l$, $i=k-j+1=k-l+1$

ceea ce conduce la deducerea următoarelor relaţii:

$$(4) \qquad \left\{\begin{aligned} &\Delta^j y_i = D_{kl} \\ &\Delta^{j-1} y_{i+1} = D_{k,l-1} \\ &\Delta^{j-1} y_i = D_{k-1,l-1} \end{aligned}\right\}, \quad l=\overline{2,n-1}, \quad k=\overline{l,n-1}$$

Corespondenţele din relaţia (4) se înlocuiesc în relaţia (2) şi rezultă astfel relaţia de calcul al elementelor triunghiului inferior al matricei D, ţinând cont că elementele primei coloane (l=1) se calculează separat:

$$(5) \qquad \begin{cases} D_{k1}=\dfrac{y_{k+1}-y_k}{x_{k+1}-x_k}, \quad k=\overline{1,n-1} \\ D_{kl}=\dfrac{D_{k,l-1}-D_{k-1,l-1}}{x_{k+1}-x_{k-l+1}}, \quad l=\overline{2,n-1}, \quad k=\overline{l,n-1} \end{cases},$$

În cazul particular în care punctele x_i sunt *echidistante*, adică $x_{i+1}-x_i=h$, $i=\overline{1,n-1}$, atunci formula (1) a polinomului de interpolare Newton devine:

$$(2.15') \quad y=y_1+(x-x_1)\frac{\Delta y_1}{1!h}+(x-x_1)(x-x_2)\frac{\Delta^2 y_1}{2!h^2}+\dots+(x-x_1)\dots(x-x_{n-1})\frac{\Delta^{n-1} y_1}{(n-1)!h^{n-1}}$$

Pentru n=1 se obţine interpolarea *liniară*, pentru n=2 interpolarea *pătratică*, iar pentru n=3 interpolarea *cubică*.

Mai jos se prezintă programul script `int_newt` care calculează polinomul de interpolare al lui Newton şi exemplifică trasarea graficului acestuia prin nodurile de interpolare.

```
%INTERPOLARE CU POLINOM NEWTON

%exemplu de iniţializare
x=[1 5 8 9 10];
y=[2 30 11 4 8];

n=length(x);
%calculul primei coloane din matricea de diferenţe finite, D
for k=1:(n-1),
   D(k,1)=(y(k+1)-y(k))/(x(k+1)-x(k));
end;

%calculul restului elementelor din triunghiul inferior al matricei D
for l=2:n-1,
   for k=l:n-1,
      D(k,l)=(D(k,l-1)-D(k-1,l-1))/(x(k+1)-x(k-l+1));
   end;
end;

tt=fliplr(diag(D)');%extragerea diagonalei principale a matricei D
    %într-un vector linie şi reaşezarea lui în ordinea inversă a rangurilor

%calculul coeficienţilor polinomului de interpolare al lui Newton
coef=[zeros(1,n-1) y(1)];
for i=1:(n-1),
   coef=coef+[zeros(1,i-1) poly(x(1:n-i))*tt(i)];
end;
%construirea vectorilor pentru reprezentările grafice
pas=(max(x)-min(x))/250;
xx=min(x):pas:max(x);
for i=1:length(xx),
   yy(i)=polyval(coef,xx(i));
end;
plot(x,y,'mo',xx,yy,'r-');grid;
title('INTERPOLARE PRIN POLINOM NEWTON');
```

Programul calculează elementele matricei D a diferenţelor finite folosind relaţiile (5) şi construieşte vectorul `tt` al acelor diferenţe care intervin în calculul coeficienţilor polinomului lui Newton, şi anume $\Delta^{n-1}y_1$, $\Delta^{n-2}y_1$, ... Δy_1 (în această ordine, corespunzătoare ordinii descrescătoare a rangurilor coeficienţilor în vectorul `coef`). În calculul vectorului `coef` s-a utilizat funcţia `poly` din biblioteca Matlab. Aceasta primeşte un vector de dimensiune n şi returnează într-un alt vector, de dimensiune n+1, coeficienţii polinomului care are drept rădăcini elementele vectorului de intrare.

Pentru iniţializările considerate, polinomul de interpolare Newton are gradul 4 şi este:

$$N(x) = 0.1018 \cdot x^4 - 2.1238 \cdot x^3 + 13.2732 \cdot x^2 - 22.6798 \cdot x + 13.4286$$

1.2 Formula de interpolare a lui Lagrange

În cazul general, un polinom de gradul n-1, constrâns să ia pentru $x=x_i$ valorile y_i, i=1,2,...n, este dat de ***formula de interpolare a lui Lagrange***:

$$(6) \qquad L(x)=\sum_{i=1}^{n} y_i \cdot \frac{(x-x_1)\cdot(x-x_2)\cdot\ldots\cdot(x-x_{i-1})\cdot(x-x_{i+1})\cdot\ldots\cdot(x-x_n)}{(x_i-x_1)\cdot(x_i-x_2)\cdot\ldots\cdot(x_i-x_{i-1})\cdot(x_i-x_{i+1})\cdot\ldots\cdot(x_i-x_n)}$$

Observaţie:

Formula (6) şi formula (1), de interpolare a lui Newton, conduc la ***acelaşi polinom de interpolare***; acesta poate fi referit ca „polinom Newton" sau ca „polinom Lagrange", după cum este obţinut cu una sau cu cealaltă din formule.

Dacă se notează:

$$(7) \qquad s_i(x)=(x-x_1)\cdot(x-x_2)\cdot\ldots\cdot(x-x_{i-1})\cdot(x-x_{i+1})\cdot\ldots\cdot(x-x_n),\ i=\overline{1,n},$$

atunci relaţia (2.20) se scrie:

$$(8) \qquad L(x)=\sum_{i=1}^{n} y_i \cdot \frac{s_i(x)}{s_i(x_i)}=\sum_{i=1}^{n} \underbrace{\frac{y_i}{s_i(x_i)}}_{p_i(x)} \cdot s_i(x)$$

Toate polinoamele s_i au gradul n-1, deci şi polinoamele notate cu p_i în relaţia (8) au gradul n-1. Prin urmare, coeficienţii acestor polinoame se pot reprezenta prin vectori de dimensiune n. Vectorul coeficienţilor polinomului Lagrange se obţine prin sumarea vectorilor coeficienţilor celor n polinoame p_i; algoritmul de calcul este schiţat mai jos.

Date de intrare: $\{x_i\}$, $\{y_i\}$, i=1,2,...n (cu $\{x_i\}$ strict monoton)

#0. Se iniţializează vectorul L al coeficienţilor polinomului Lagrange la vectorul nul de dimensiune n-1.

Pentru i de la 1 *la n*

#1. Se calculează vectorul *cs* al coeficienţilor polinomului s_i.

#2. Se calculează valoarea polinomului s_i în x_i: $v \leftarrow s_i(x_i)$.

#3. Se calculează vectorul *cp* al coeficienţilor polinomului p_i: $cp \leftarrow cs \cdot y_i/v$.

#4. $L \leftarrow L+cp$

Repetă

Mai jos este listat un program Matlab care realizează calculul polinomului Lagrange şi evaluează polinomul într-o valoare intermediară dată de utilizator. S-a introdus o secvenţă de comenzi prin care se îndeplineşte condiţia ca valorile x_i să fie ordonate monoton (folosirea funcţiei `sort`).

```
%Interpolarea funcţiilor prin POLINOMUL DE INTERPOLARE LAGRANGE

%exemplu de iniţializare
x=[1 5 8 9 10];
y=[2 30 11 4 8];

[m n]=size(x);

%sortarea datelor:  ordonarea crescătoare a valorilor lui x
```

```
%                    şi ordonarea corespunzătoare a valorilor lui y
[xsort,ind]=sort(x);
l=1;
for i=1:n,
   ysort(l)=y(ind(i));
   l=l+1;
end;
x=xsort;
y=ysort;

%calculul coeficienţilor polinomului de interpolare
L=zeros(1,n);
for i=1:n,
   cs=poly([x(1:i-1) x(i+1:n)]);
   cp=cs*y(i)/polyval(cs,x(i));
   L=L+cp;
end;

pas=(max(x)-min(x))/250;
xx=min(x):pas:max(x);
for i=1:length(xx),
   yy(i)=polyval(L,xx(i));
end;

val=input('Introduceti valoarea in care doriti interpolarea:');
yval=polyval(L,val);
%grafic
plot(xx,yy,'r-',x,y,'g*',xv,yv,'m--');grid;
s1=sprintf('INTERPOLARE PRIN POLINOM LAGRANGE');
title(s1);
xlabel('x');
ylabel('y');
```

S-a utilizat acelaşi exemplu de iniţializare ca şi în programul `int_newt`, de calcul al polinomului de interpolare Newton; se obţine următorul polinom Lagrange:

$$L(x) = 0.1018 \cdot x^4 - 2.1238 \cdot x^3 + 13.2732 \cdot x^2 - 22.6798 \cdot x + 13.4286$$

Se observă identitatea polinoamelor Newton şi Lagrange.

1.3 Funcţii de interpolare din biblioteca Matlab

Interpolarea unidimensională se realizează în Matlab cu ajutorul funcţiei `interp1`; o altă funcţie, `interpft`, este dedicată interpolării funcţiilor periodice, cu ajutorul transformatei Fourier rapide. Funcţia Matlab `spline` implementează interpolarea unidimensională cu funcţii spline cubice. Funcţiile `interp2`, `griddata` şi `meshgrid` realizează interpolare bidimensională.

Funcţia `interp1` primeşte trei argumente de intrare: doi vectori de aceeaşi dimensiune,

reprezentând respectiv abscisele şi ordonatele nodurilor de interpolare, şi un al treilea vector, conţinând abscisele în care se doreşte interpolarea. Funcţia returnează un vector de aceeaşi dimensiune cu al treilea argument de intrare, care conţine valorile interpolate. Abscisele nodurilor trebuie să fie monotone (ordonate crescător sau descrescător).

În mod implicit, `interp1` realizează interpolare *liniară* între oricare două noduri cu abscise succesive. O a doua formă de apel a acestei funcţii prevede furnizarea unui al patrulea argument de intrare, de tip şir de caractere, prin care se specifică metoda de interpolare: `'linear'` (implicit), `'spline'` (folosind funcţii spline cubice) şi `'cubic'` (interpolare cubică).

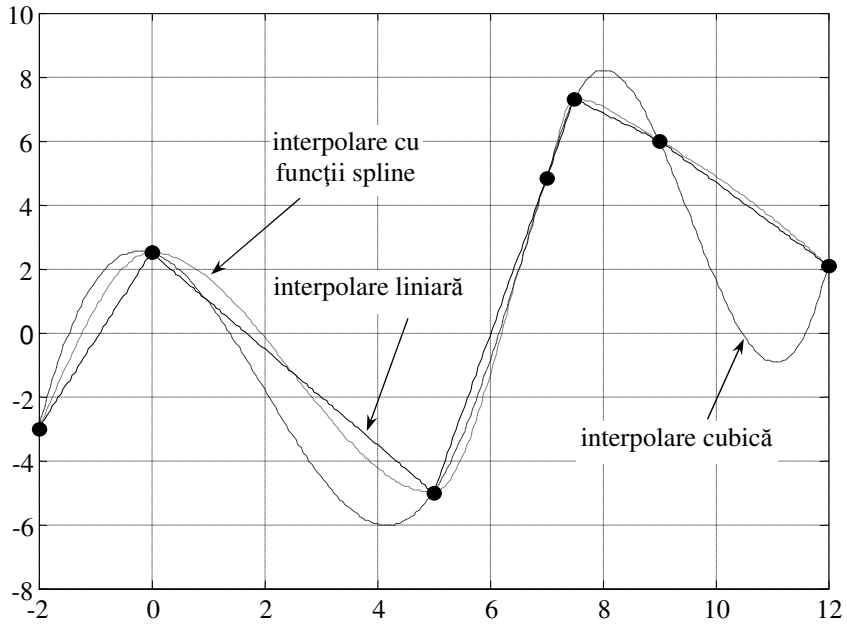

Fig. 1 Trei metode de interpolare aplicate aceloraşi date de intrare

O aplicaţie tipică a interpolării este creşterea preciziei de trasare a graficelor; blocul de comenzi de mai jos exemplifică aceasta cu folosirea funcţiei `interp1`.

```
x=[-2 0 5 7 7.5 9 12];
y=[-3 2.5 -5 4.8 7.3 6 2.1];
xx=min(x):0.05:max(x);
yy1=interp1(x,y,xx);%metoda implicită este cea liniară
yy2=interp1(x,y,xx,'spline');
yy3=interp1(x,y,xx,'cubic');
plot(x,y,'ko',xx,yy1,'k-',xx,yy2,'k--',xx,yy3,'k:');grid
```

Figura 1 (rezultată în urma apelului funcţiei `plot`) arată deosebirile între diferitele metode de interpolare folosite de `interp1`.