5.7. Calcul numeric cu polinoame

Polinomul este o formă algebrică utilizată frecvent în inginerie. De exemplu modelarea şi analiza sistemelor liniare pe baza funcţiilor de transfer reprezintă o categorie semnificativă de probleme cu aplicabilitate în teoria controlului automat. Polinoamele se folosesc uzual şi în probleme curente de evaluare numerică cum sunt interpolarea şi aproximarea datelor.

Polinomul este o expresie matematică de o variabilă (ridicată la diferite puteri), care poate fi exprimat sub forma generala:

$$P(x) = a_1 x^n + a_2 x^{n-1} + a_3 x^{n-2} + \ldots + a_{n-1} x^2 + a_n x + a_{n+1} .$$

Puterea cea mai mare care apare în expresia polinomului determină *ordinul* acestuia.

În sintaxa Matlab, un polinom se descrie şi se defineşte prin vectorul linie al coeficienţilor săi aranjaţi *în ordinea descrescătoare a puterilor*:

$$p = [a_1 \; a_2 \; a_3 \ldots a_{n+1}]$$

De exemplu, cu vectorul de şase elemente `c=[-1 3 5 0 4 -2]`, respectiv

```
c =
    -1     3     5     0     4    -2
```

se defineşte următorul polinom de ordinul cinci:

```
P(x)=-x^5+3*x^4+5*x^3+4*x-2,
```

la care, se observă absenţa termenului de ordinul doi, al cărui coeficient este egal cu zero.

5.7.1 Evaluarea polinoamelor

A evalua un polinom înseamnă a calcula valoarea sa pentru o anumită valoare a variabilei argument, rezolvând operaţiile matematice din expresia sa: ridicări la putere, înmulţiri şi însumare algebrică.

Evaluarea polinoamelor se poate face în mai multe moduri, pentru argument scalar (o singură valoare), respectiv vector/tablou numeric (pentru valori multiple), în conformitate cu descrierile şi exemplele din tabelul următor.

Mod de evaluare	**Exemplificare**

Cu scalari
(pentru o valoare singulară)
	`>>x=10;`
	`>>p=5*x^3-x^2+x-1`
	
	Rezultatul evaluării :
	p =
	4909

Cu tablouri
(pentru mai multe valori)
	`>>x=[-1 0 2 3.5 10]`
	`>>p=5.*x.^3-x.^2+x-1`
	
	Rezultatul evaluării :
	p =
	1.0e+003 *
	-0.0080 -0.0010 0.0370 0.2046 4.9090

Cu funcţia
`polyval`
	`f=polyval(p,s)`
	
	p – vectorul coeficientilor polinomului
	s – valoarea/valorile de evaluare (vector, matrice).
	
	a) Evaluarea într-o mulţime de puncte
	
	`>>p=[5 -1 1 -1]`
	`>>x=[-1 0 2 3.5 10]`
	`>>f=polyval(p,x)`
	
	Rezultatul evaluării :
	f =
	1.0e+003 *
	-0.0080 -0.0010 0.0370 0.2046 4.9090
	
	b) Evaluarea pe un interval (mulţime compactă)
	
	`>>p=[5 -1 1 -1]`
	`>>x=-1:(11/10):10`
	`>>f=polyval(p,x)`
	
	Rezultatul evaluării :
	f =
	1.0e+003 *
	-0.0080 -0.0009 0.0074 0.0568 0.1874 0.4389
	0.8513 1.4646 2.3187 3.4535 4.9090

5.7.2 Operații aritmetice cu polinoame

Calculele cu polinoame apar atunci când cel puțin unul dintre operanzi este un polinom de un anumit ordin. Pentru operațiile de adunare şi scădere se operează asupra vectorilor coeficienților, în modul de operare cu tablouri de date. Înmulțirea respectiv împărțirea unui polinom cu o constantă (scalar) se efectuează tot ca operații cu tablouri. Operațiile de înmulțire şi împărțire prezintă particularități în cazul în care cei doi operanzi sunt polinoame, în sensul că presupun operații termen cu termen urmate de *concatenarea* şi *reordonarea* termenilor rezultanți.

În Matlab există funcții foarte utile pentru efectuarea rapidă a operațiilor de înmulțire şi împărțire a două polinoame, precum şi pentru exprimarea raportului a două polinoame sub formă de fracții simple.

Operații	Descriere

Adunarea şi scăderea	`s=g+h`
	`u=g-h`
	Observație. Dacă nu au aceeaşi lungime, vectorii coeficienților se completeaza la aceeaşi dimensiune cu elemente zero.

Înmulțirea şi împărțirea	- <u>înmulțirea</u>: convoluția
	`c=conv(a,b)`
	`a,b` - vectorii coeficienților polinoamelor operanzi
	`c` – vectorul coeficienților polinomului rezultat
	-<u>împărțirea</u>: deconvoluția
	`[d,r]=deconv(a,b)`
	d-vectorul coeficienților polinomului cât,
	r- vectorul coeficienților polinomului rest.

Descompunerea în fracții simple	Scrierea expresiei raport de polinoame $E(x)=\frac{P_1(x)}{P_2(x)}$ ca sumă de fracții cu polinoame de ordinul 1, ireductibile, în forma : $\frac{P_1(x)}{P_2(x)}=\frac{r_1}{x-p_1}+\frac{r_2}{x-p_2}+\ldots+\frac{r_n}{x-p_n}+k(x)$, unde
	$r_1, \ldots, r_n$ reziduri,
	$p_1, \ldots, p_n$ poli,
	k(x) termen liber.
	`[r,p,k]=residue(p1,p2)`
	sau reciproc (dacă se cunosc rezidurire, polii şi termenul liber)
	`[p1,p2]=residue(r,p,k)`

Exemple de operații cu polinoame.

```
>> a=[-1 3 5 -6 0 10]

a =
     -1     3     5    -6     0    10


>> b=[0 2 -1 10 5 -2]

b =
     0     2    -1    10     5    -2


>> suma=a+b

suma =
       -1     5     4     4     5     8


>> dif1=a-b

dif1 =
       -1     1     6   -16    -5    12


>> dif2=b-a

dif2 =
        1    -1    -6    16     5   -12

% inmultire cu un scalar c=10
>> c=10
c =
    10


>> pc=c.*a

pc =
   -10    30    50   -60     0   100



>> conv(c,a)

ans =
   -10    30    50   -60     0   100
```

```
%inmultirea si impartirea a doua polinoame A(x) •i B(x)
>> A=[4 -1 0 5 10]
A =
     4    -1     0     5    10


>> B=[-2 3 1 -1 5]
B =
    -2     3     1    -1     5


>> P=conv(A,B)

P =
    -8    14     1   -15    16    30     5    15    50


>> [d,r]=deconv(A,B)

d =
    -2
r =
     0     5     2     3    20
```

```
% decompunerea in fractii a raportului de doua polinoame
>> a=[-1 3 5 -6 0 10]
a =
     -1     3     5    -6     0    10


>> b=[0 2 -1 10 5 -2]
b =
     0     2    -1    10     5    -2


>> [r,p,k]=residue(a,b)

r =
   2.5922 + 1.7651i
   2.5922 - 1.7651i
  -0.5090
   0.9497

p =
   0.4598 + 2.3139i
   0.4598 - 2.3139i
  -0.6827
   0.2632

k =
   -0.5000    1.2500
```

Rezultatul descompunerii în fracţii din exemplul anterior este expresia următoare:

$$\frac{A(x)}{B(x)} = \frac{2.5922+1.7651i}{x-(0.4598+2.3139i)} + \frac{2.5922-1.7651i}{x-(0.4598-2.3139i)} + \frac{-0.509}{x+0.6827} + \frac{0.9497}{x-0.2632} - 0.5x + 1.25$$

Abordând *problema inversă*: când se cunosc rezidurile, polii şi polinomul termenilor liberi, se poate genera raportul polinoamelor care conţin aceste elemente în descompunerea respectivă, apelând aceeaşi funcţie `residue`, dar cu sintaxa diferită în ceea ce priveşte lista parametrilor de intrare şi lista parametrilor de ieşire.

Exemplu. Fie în ordinea dată următoarele elemente:

- rezidurile: -5, 1, 0, 4;
- polii: 1, -2, 0 ,3;
- polinomul termenilor liberi k(x)=0.75x–2.

Să se determine raportul polinoamelor ce admite această descompunere.

```
>> r=[-5 1 0 4]
r =
    -5     1     0     4


>> p=[1 -2 0 3]
p =
     1    -2     0     3


>> k=[0.75 -2]
k =
    0.7500   -2.0000

>> [A,B]=residue(r,p,k)

A =
    0.750   -3.500    0.250   19.500   13.000    0

B =
     1    -2    -5     6     0
```

Expresia raportului polinoamelor este:

$$\frac{A(x)}{B(x)} = \frac{0,75x^5 - 3,5x^4 + 0,25x^3 + 19,5x^2 + 13x}{x^4 - 2x^3 - 5x^2 + 6x}$$

5.7.3 Calculul derivatei polinoamelor

Operația de derivare a expresiilor polinomiale intervine curent în problemele de extremum (minim şi maxim funcțional) atunci când funcțiile sunt sau pot fi aduse sub o formă polinomială.

Funcția Matlab `polyder` este foarte eficientă în calculul derivatei pentru un polinoam singular, pentru produsul a două polinoame şi pentru raportul a două polinoame, cu returnarea rezultatului tot sub formă de raport a două polinoame. Mai jos sunt centralizate variantele de utilizare ale acestei funcții.

Pentru un singur polinom	`d=polyder(p)`	$d = p'$
Pentru un produs de polinoame	`d=polyder(p1,p2)`	$d = (p1 \cdot p2)' = p1' \cdot p2 + p1 \cdot p2'$
Pentru un raport de polinoame	`[q1,q2]=polyder(p1,p2)`, returnează rezultatul derivatei raportului de polinoame p1/p2 sub forma raportului q1/q2	$\frac{q1}{q2} = \left(\frac{p1}{p2}\right)' = \frac{p1' \cdot p2 - p1 \cdot p2'}{(p2)^2}$

Exemplu de aplicare :

```
>> p1=[2 5 -1 0 1]
p1 =
     2     5    -1     0     1
>> p2=[1 -1 4 0 2]
p2 =
     1    -1     4     0     2

>> d=polyder(p1)
d =
     8    15    -2     0

>> dp=polyder(p1,p2)
dp =
    16    21    12   105     4    27     4     0


>> [q1,q2]=polyder(p1,p2)

q1 =
    -7    18    19    12    33   -12     0
q2 =
     1    -2     9    -8    20    -4    16     0     4
```

Rezultatele obținute corespund următoarelor exprimări analitice:

$$pl(x)' = \left(2x^4 + 5x^3 - x^2 + 1\right)' = 8x^3 + 15x^2 - 2x \ ,$$

$$d = (pl \cdot p2)' = 16x^7 + 21x^6 + 12x^5 + 105x^4 + 4x^3 + 27x^2 + 4x \ ,$$

$$\frac{ql}{q2} = \left(\frac{pl}{p2}\right)' = \frac{-7x^6 + 18x^5 + 19x^4 + 12x^3 + 33x^2 - 12x}{x^8 - 2x^7 + 9x^6 - 8x^5 + 20x^4 - 4x^3 + 16x^2 + 4} \ .$$

5.7.4 Calculul rădăcinilor şi coeficienților polinoamelor

Un număr $\alpha \in C$ se numeşte rădăcină a polinomului P(x), dacă şi numai dacă $P(\alpha)=0$. În baza *teoremei lui Bézout* numărul α este rădăcină a unui polinom nenul P(x) dacă şi numai dacă $(x-\alpha)$ divide P(x). Prin definiție, α se numeşte *rădăcină multiplă de ordinul p* a polinomului P(x), dacă şi numai dacă $(x-\alpha)^p$ divide P(x), iar $(x-\alpha)^{p+1}$ nu divide P(x). În general se vorbeşte de *ordinul de multiplicitate* al rădăcinilor unui polinom. Pentru p=1 rădăcina se numeşte *simplă*, iar pentru p=2 ea se numeşte *rădăcină dublă*.

În Matlab există două funcții cu efect reciproc, ce returnează rădăcinile unui polinom, respectiv generează coeficienții polinomului care are rădăcini date.

Calculul rădăcinilor (când se cunosc coeficienții)	`r=roots(p)`
Calculul coeficienților (când se cunosc rădăcinile)	`p=poly(r)`

Mai jos este exemplificată aplicarea acestor funcții pentru două polinoame $p_1(x)$ şi $p_2(x)$ care admit rădăcini reale şi complexe, respectiv numai rădăcini complexe. Se generează de asemenea, un polinom $p(x)$ care admite rădăcini reale conținute într-un vector *r*.

```
%calculul radacinilor
>> p1=[2 5 -1 0 1]

p1 =
    2    5   -1    0    1

>> p2=[1 -1 4 0 2]

p2 =
    1   -1    4    0    2

>> r=roots(p1)

r =
  -2.6613
  -0.5611
   0.3612 + 0.4520i
   0.3612 - 0.4520i

>> r=roots(p2)

r =
   0.5878 + 1.8205i
   0.5878 - 1.8205i
  -0.0878 + 0.7340i
  -0.0878 - 0.7340i
```

```
% calculul coeficientilor
% cand se cunosc
% radacinile

>> r = [1 5 -5 0 2]

r =

     1    5   -5    0
2


>> p=poly(r)

p =

   1  -3  -23   75  -50
0
```

5.7.5 Polinoame pentru interpolarea şi aproximarea datelor

Prin interpolare se rezolvă problema generală ca o funcție de N variabile, cunoscută numai prin valorile ei într-un număr finit de puncte distincte date din domeniul ei de definiție (denumite *noduri*), să fie cunoscută în oricare punct situat printre cele date. Cu alte cuvinte, se pune problema de a obține o determinare ″mai fină″ a unei funcții atunci când nu se cunoaşte expresia analitică a acesteia. Problema generală a interpolării se particularizează în practică pentru funcții de o singură variabilă, de două variabile, etc.

Interpolarea este una dintre cele mai folosite metode pentru aproximarea funcțiilor cunoscute discret şi se bazează pe construirea unui *polinom de interpolare*.

Definiție. Se numeşte polinom de interpolare pe nodurile $x_1, x_2, ..., x_n$, al unei funcții $f:[a,b]\to R$, cunoscută prin valorile sale $f(x_i)$, $i=1,...,n$, un

polinom $P(x)$ de grad cel mult $n-1$, care verifică relațiile: $P(x_i)=f(x_i)$, $i=1,...,n$.

Cea mai simplă metodă de interpolare, dar care furnizează însă cea mai slabă aproximare a unei funcții discrete este *interpolarea liniară*. Aceasta nu asigură însă o funcție de interpolare netedă, fiind vorba de o aproximare cu segmente de dreaptă.

Din punct de vedere practic, una din clasele cele mai importante de metode de interpolare utilizează polinoamele algebrice sau trigonometrice. Metodele de interpolare polinomială se disting prin tipul polinomului de interpolare utilizat şi ordinul acestuia. Metodele polinomiale implică utilizarea unor tipuri de polinoame particulare: Lagrange, Newton, Hermite, spline, cubice, etc..

Extrapolarea este metoda prin care se determină funcția în puncte situate în afara intervalului nodurilor date (prelungirea funcției). În general, vorbim de extrapolare atunci cînd x este în afara celui mai mic interval care conține nodurile date.

În figura 5.9 este ilustrat conceptul de interpolare şi cel de extrapolare pentru funcții de o variabilă.

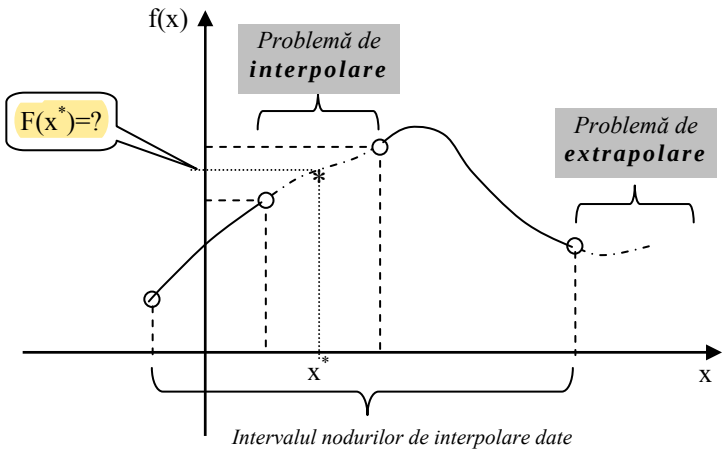


Fig. 5.9. Ilustrarea conceptului de interpolare şi extrapolare

Problema *aproximării datelor* constă în înlocuirea unei distribuții oarecare de valori ale unei funcții discrete cu o funcție analitică, de formă cât mai simplă (chiar liniară) îndeplinind o condiție globală de eroare

minimă. Această problemă se întâlneşte frecvent în statistică şi poartă numele de *regresie* (liniară sau în general polinomială) şi are ca scop găsirea unor dependenţe grafo-analitice între două variabile aleatoare. Se aplică în probleme inginereşti atunci când se lucrează cu date care provin din măsurători mai puţin precise.

A. Interpolarea funcţiilor de o singură variabilă

Interpolarea liniară presupune aproximarea funcţiilor discrete cu segmente liniare ce unesc punctele date. Această metodă se poate efectua în Matlab printr-un procedeu de tip "table lookup" folosind funcţia tradiţională `table1` cu sintaxa `yi=table1(nume_tabel,xi)`, în care:

- `nume_tabel` este o structură de date organizată ca un tablou care are în prima coloană valorile lui x (abscisa ordonată crescător), iar în coloana a doua valorile funcţiei în punctele de abscisă date y(x). Variabila `nume_tabel` poate să fie şi un fişier ce se încărcă cu funcţia `load`,
- `xi` este punctul în care se solicită interpolarea,
- `yi` este valoarea interpolată a funcţiei.

Deşi funcţia `table1` este considerată depăşită (obsolete) în versiunile actuale ale Matlab-ului, ea fiind înlocuită de funcţia `interp1`, se dă mai jos un exemplu de utilizare a ei pentru evidenţierea conceptului de reprezentare tabelară a unei funcţii discrete şi utilizarea acesteia în procesul de interpolare liniară.

Exemplu. Fie tabelul ce defineşte prin valori discrete funcţia $f:[-10,5]\to R$ astfel:

x	y
-10	8
-1	2
0	0
3	-2
5	1

Se cere să se verifice valorea funcţiei în punctul xi=−1 şi să se determine valoarea funcţiei în punctul −8. Problema se rezolvă cu secvenţa de instrucţiuni următoare:

```
>> TAB=[-10 -1 0 3 5; 8 2 0 -2 1]'
TAB =

   -10     8
    -1     2
     0     0
     3    -2
     5     1

>> y1=table1(TAB,-1)
y1 =
     2

>> y1=table1(TAB,-8)
y1 =
    6.6667
```

O funcție Matlab generală pentru interpolarea funcțiilor de o singură variabilă este `interp1` care poate fi apelată cu diferite sintaxe:

```
yi = interp1(x,y,xi)
yi = interp1(y,xi)
yi = interp1(x,y,xi,'method')
yi = interp1(x,y,xi,'method','extrap')
```

unde:
- `x` este vectorul punctelor (nodurilor) date, care trebuie să fie ordonat monoton,
- `y` este vectorul valorilor date (cunoscute) ale funcției în noduri,
- `xi` este vectorul valorilor (punctelor) în care se solicită interpolarea,
- `yi` este vectorul valorilor obținute prin interpolare,
- `method` este un descriptor care specifică opțional una din metodele de interpolare, conferind astfel funcției caracterul de *interpolare multiplă* cu diferite metode:
  - 'nearest' - interpolare la cea mai apropiată valoare (vecină),
  - 'linear' - interpolare liniară (metodă implicită),
  - 'spline' - interpolare cu porțiuni potrivite de polinoame spline cubice,
  - 'pchip' - interpolare cu porțiuni potrivite de polinoame Hermite cubice,
  - 'cubic' - similară cu 'pchip',

  - 'v5cubic' - interpolare cubică, care nu extrapolează şi utilizează automat metoda *spline* dacă x nu are elementele echidistante (nodurile egal depărtate),

- `extrap` permite extrapolarea prin metoda specificată (`method`), iar dacă aceasta nu se specifică extrapolarea se efectuează implicit pentru metodele 'spline' si 'pchip' şi returnează valoarea `extrapval=NaN` pentru celelalte metode.

Observaţii.

- Utilizarea sintaxelor cu patru respectiv cinci parametri în lista de intrare (vezi ultimele două) permite *interpolarea multiplă* în sensul selectării unei anumite metode dintre cele puse la dispoziție;
- Utilizarea funcției cu sintaxa `yi=interp1(y,xi)` presupune automat că punctele date (nodurile) sunt elementele vectorului `x=1:N`, unde `N=length(y);` altfel spus interpolarea se face pe intervalul [1,length(y)];
- Dacă `y` este o matrice, atunci interpolarea se efectuează pentru fiecare coloană;
- Dacă nu se specifică altfel, implicit metoda de interpolare cu care operează funcția este cea liniară;
- Valorile tipice returnate de `extrapval` sunt NaN şi 0;
- Alte funcții Matlab care rezolvă interpolarea funcțiilor de o variabilă sunt:
  - `interp1q` pentru `x` monoton crescător,
  - `interpft` pentru interpolarea cu transformata Fourier rapidă,
  - `spline` pentru interpolarea cu polinoame spline cubice.

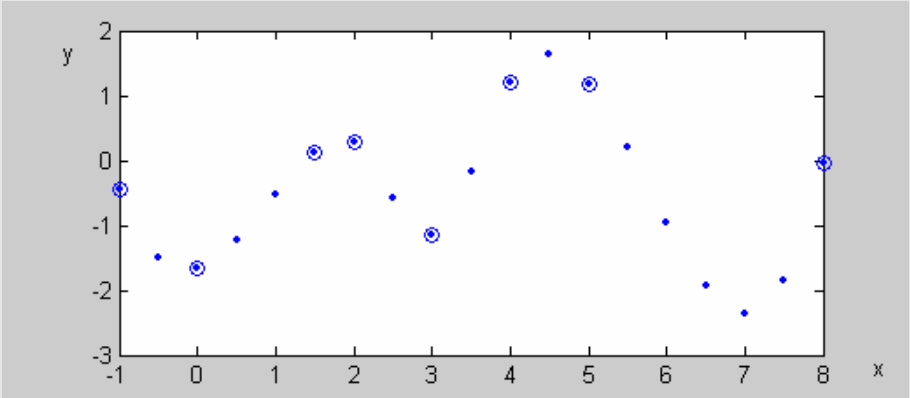
- Funcții similare care determină valori şi anumite elemente ale unor polinoame de aproximare sunt: `ppval`, `mkpp`, `unmkpp`. Pentru mai multe detalii în legătură cu acestea recomandăm consultarea help-ului Matlab.

Exemplu. Folosind aceleaşi date numerice ca în exemplul precedent să se aplice funcția `interp1` pentru diferite metode de interpolare.

```
>> x=[-10 -1 0 3 5]

x =
   -10    -1     0     3     5


>> y=[8 2 0 -2 1]

y =
    8     2     0    -2     1


>> yi=interp1(x,y,-8,'nearst')
yi =
    8


>> yi=interp1(x,y,-8,'linear')
yi =
    6.6667


>> yi=interp1(x,y,-8)
yi =
    6.6667


>> yi=interp1(x,y,-8,'cubic')
yi =
    7.6417


>> yi=interp1(x,y,-8,'pchip')
yi =
    7.6417


>> yi=interp1(x,y,-8,'spline')
yi =
    13.0286


>> yi=interp1(x,y,-8,'v5cubic')
yi =
    13.0286
```

În continuare, se exemplifică interpolarea prin metoda *spline* a unei funcții discrete generată aleator în 8 noduri, pentru 19 puncte de interpolare. Rezultatul este reprezentat grafic astfel: nodurile de interpolare- cu simbolul cerculeț, punctele interpolare- cu simbolul punct.

```
>> x=[-1 0 1.5 2 3 4 5 8]

x =
   -1.0000    0    1.5000   2.0000   3.0000   4.0000   5.0000   8.0000

>> y=randn(1,8)

  y =
     -0.4326  -1.6656  0.1253  0.2877  -1.1465   1.1909   1.1892  -0.0376


>> xi=-1:0.5:8

  xi =
    Columns 1 through 10

     -1.0000   -0.5000   0   0.5000   1.0000   1.5000   2.0000   2.5000
3.0000   3.5000

    Columns 11 through 19

   4.0000   4.5000   5.0000   5.5000   6.0000   6.5000   7.0000  7.5000
8.0000


  >> yi=interp1(x,y,xi,'spline')

  yi =
    Columns 1 through 10

    -0.4326  -1.5024  -1.6656  -1.2381  -0.5359  0.1253  0.2877  -0.5766
-1.1465  -0.1884

    Columns 11 through 19

      1.1909  1.6424  1.1892  0.1971  -0.9679  -1.9398  -2.3527  -1.8406  -
0.0376
```



Interpolarea prin *metoda transformatei Fourier* este potrivită pentru funcții periodice utilizând funcția `interpft` cu sintaxa:

```
yi=interpft(y,n).
```

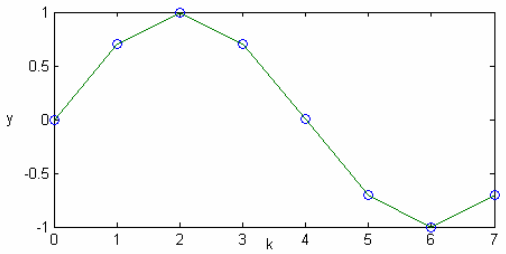
Aceasta returnează un vector `yi` de lungime `n` obținut din vectorul `y`, unde `n` este mai mare decât dimensiunea vectorului `y`: length(yi)>length(y).

Aplicație. Să se refacă un semnal armonic din eşantioane pe baza interpolării cu transformata Fourier. Fie funcția *sinus* eşantionată cu opt paşi (echidistanți) pe perioadă, adică $y = \sin\left(\frac{2\pi \cdot k}{8}\right)$. Să se refacă funcția prin interpolare cu pas de trei ori mai mic (mai fin) şi să se evalueze eroarea maximă de interpolare.

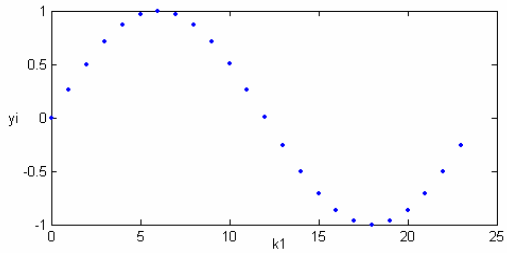
Cerințele acestei probleme se rezolvă cu următoarea secvență de instrucțiuni Matlab:

```
%interpolare prin metoda transformatei Fourier
k=0:7;
y=sin(2*pi*k/8);
yi=interpft(y,24);
%evaluare eroare
k1=0:23;
yr=sin(2*pi*k1/24);
d=max(yi-yr)
```

Punctele de eşantionare a semnalului sinusoidal:



Valorile interpolate pentru refacerea semnalului:



B. Interpolarea funcțiilor de două variabile

Din punct de vedere geometric, funcțiile de două variabile reprezintă suprafețe. Acest tip de interpolare se mai numeşte şi *bidimensională* sau interpolare 2D.

Prin interpolarea unei funcții de două variabile $f(x,y)=z$ în puncte (x_i,y_i) intermediare celor date (nodurilor) se obțin valori interpolate z_i, care generează (aproximativ) suprafața. În funcție de tipul expresiei matematice a curbei ce uneşte două puncte pe fiecare din cele două direcții de coordonate x, respectiv y, există tipul armonic, liniar, cubic. Deoarece interpolarea se aplică pentru două direcții, metodele se numesc: biarmonică, biliniară, bicubică, etc.

Funcțiile Matlab pentru interpolarea funcțiilor de două variabile sunt similare ca sintaxă cu cele pentru funcții de o singură variabilă, singura deosebire fiind aceea că, în mod evident, trebuie să accepte mai mulți parametrii de intrare în lista argumentelor. Sintaxele de apel sunt:

```
zi = interp2(x,y,z,xi,yi)
zi = interp2(z,xi,yi)
zi = interp2(z,ntimes)
zi = interp2(x,y,z,xi,yi,'method')
```

unde:

- `x` şi `y` sunt matrici care specifică punctele (nodurilor) în care funcția este cunoscută prin valori `z` date,
- `xi` şi `yi` sunt matrici care specifică punctele în care se solicită interpolarea,
- `zi` este o matrice (rezultat) care conține valorile interpolate,
- `ntimes` este un specificator pentru recursivitatea interpolării intercalate între fiecare element al matricei date `z`,
- `method` este un descriptor care specifică opțional una din metodele de interpolare:
  - 'nearest' - interpolare la cea mai apropiată valoare (vecină),
  - 'linear' - interpolare biliniară (metodă implicită),
  - 'spline' - interpolare spline,
  - 'cubic' - interpolare bicubică,

Observaţie importantă. Interpolarea bidimensională necesită evaluarea funcţiei $z=f(x,y)$ pe un domeniu plan al unor puncte de coordonate (x,y). În mod normal coordonatele x şi y se dau sub formă de vectori *monotoni şi cu elementele echidistante*. În acest caz este necesară o operaţiunea suplimentară de asociere în combinaţii carteziene a valorilor în planul (tabloul) de definiţie. Acest lucru se face cu funcţia `meshgrid`.

Funcţia Matlab `meshgrid` apelată cu sintaxa `[X,Y]=meshgrid(x,y)` sau `[X,Y]=meshgrid(x)` transformă domeniul specificat de vectorii x şi y în tablourile X şi Y cu aceleaşi dimensiuni, utile în evaluarea funcţiilor de două variabile precum şi pentru reprezentări 3D de tipul mesh (plasă) sau surface (suprafaţă), care acceptă argumente matrice.

Prin urmare, utilizarea funcţiilor de interpolare bidimensională necesită transformarea variabilelor de intrare în matrici, astfel:

- xi vectorul linie al absciselor este specificat prin extindere ca o matrice cu coloane constante (adică linii identice),
- yi vectorul coloană al ordonatelor este specificat prin extindere ca o matrice cu linii constante (adică coloane identice).

În continuare, se dă un exemplu de utilizare a funcţiei `meshgrid` pentru furnizarea unei reţele de puncte de coordonate carteziene pornind de la valori date abscisei şi ordonatei.

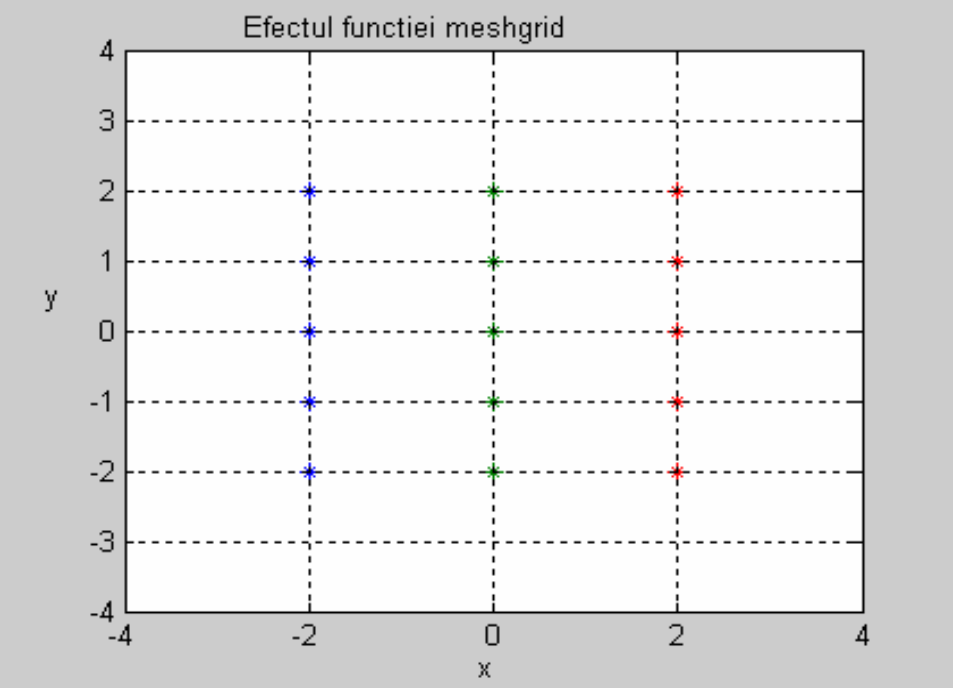
Fie domeniul xy definit de vectorii x= -2:2:2 şi y= -2:1:2 , adică x=[-2 0 2], respectiv y=[-2 –1 0 1 2]. Domeniul discretizat în puncte se obţine cu instrucţiunea, `[X,Y]=meshgrid(-2:2:2,-2:2)`, care furnizează reţeaua (grila) de puncte de coordonate [X,Y] prin extinderea vectorilor iniţiali, astfel :

X= **-2 0 2**
-2 0 2
-2 0 2
-2 0 2
-2 0 2

Y= **-2** -2 -2 -2
-1 -1 -1 -1
0 0 0 0
1 1 1 1
2 2 2 2

```
>> x=[-2 0 2]

x =
    -2     0     2


>> y=[-2 -1 0 1 2]

y =
    -2    -1     0     1     2
```

```
>> [X,Y]=meshgrid(-2:2:2,-2:2)

X =
     -2      0      2
     -2      0      2
     -2      0      2
     -2      0      2
     -2      0      2

Y =
     -2     -2     -2
     -1     -1     -1
      0      0      0
      1      1      1
      2      2      2


>> plot(X,Y,'*')
```

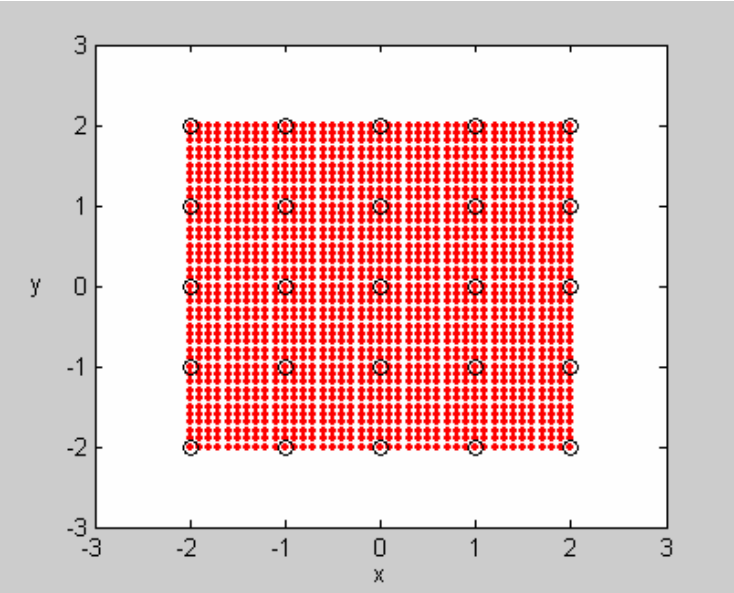


Aplicație 1. Se cere să se interpoleze funcția de două variabile $z=\sqrt{x^2+y^2}$ pe domeniul $x\in[-2,2]$ şi $y\in[-2,2]$ definită în nodurile rețelei cu pasul egal cu 1, pe o rețea de puncte mai fină, cu pasul de 0,1. Să se reprezinte grafic marcând cu cerculețe nodurile de interpolare date şi cu simbolul punct rețeaua (grila) punctelor în care se cere interpolarea.

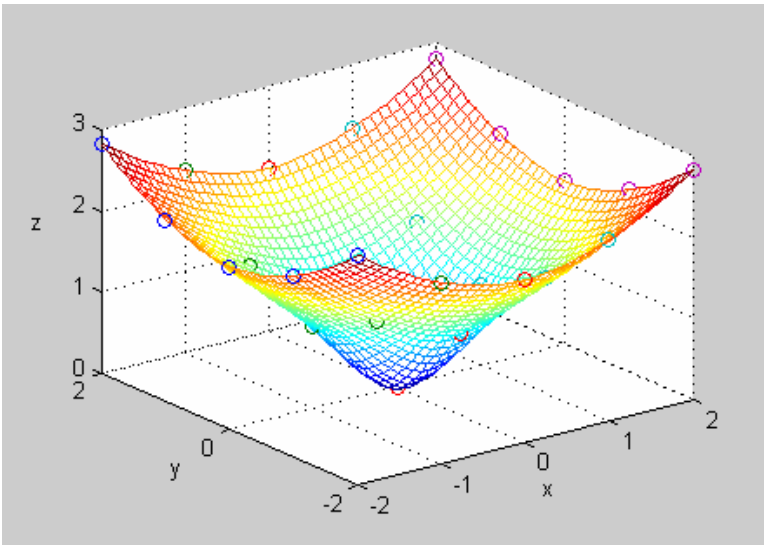
```
clear
x=-2:2
y=-2:2
[X,Y]=meshgrid(x,y)
z=sqrt(X.^2+Y.^2)
xi=-2:0.1:2
yi=-2:0.1:2
[Xi,Yi]=meshgrid(xi,yi);
zi=interp2(X,Y,z,Xi,Yi,'cubic');
mesh(Xi,Yi,zi)
hold on
plot3(X,Y,z,'o')
```

```
x =
    -2    -1     0     1     2


y =
    -2    -1     0     1     2


X =
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2

Y =
    -2    -2    -2    -2    -2
    -1    -1    -1    -1    -1
     0     0     0     0     0
     1     1     1     1     1
     2     2     2     2     2


z =
    2.8284    2.2361    2.0000    2.2361    2.8284
    2.2361    1.4142    1.0000    1.4142    2.2361
    2.0000    1.0000         0    1.0000    2.0000
    2.2361    1.4142    1.0000    1.4142    2.2361
    2.8284    2.2361    2.0000    2.2361    2.8284
```

Rețeaua nodurilor (cerculețe) și a punctelor de interpolare (puncte) obținute cu funcția `meshgrid` are următoarea reprezentare:



Rezultatul grafic al interpolării cu evidențierea valorilor funcției date inițial (reprezentate prin cerculețe) și a rețelei punctelor calculate prin interpolare cubică s-a obținut cu ajutorul funcției grafice `plot3`, respectiv `mesh`, așa cum se arată mai jos:



Aplicaţie 2. Se cere să se determine valorile funcţiei dată în aplicaţia precedentă în punctul (xi,yi)=(1.5,1.8) prin interpolare bidimensională folosind comparativ metodele 'nearest', 'linear', 'spline' şi 'cubic'.

```
>> zi=interp2(X,Y,z,1.5,1.8,'nearst')

zi =
    2.8284

>> zi=interp2(X,Y,z,1.5,1.8,'linear')

zi =
    2.3908

>> zi=interp2(X,Y,z,1.5,1.8,'spline')

zi =
    2.2948

>> zi=interp2(X,Y,z,1.5,1.8,'cubic')

zi =
    2.3099
```

Notă. Comparând valoarea rezultatelor numerice obţinute se constată caracterul aproximativ şi relativ al metodelor de interpolare.

Interpolarea bidimensională a datelor împrăştiate

În multe situaţii practice punctele (nodurile) disponibile pentru interpolare nu sunt echidistante din punct de vedere numeric. Astfel, în cazul unor valori numerice împrăştiate, elementele vectorilor x respectiv y nu mai au pas egal. Una dintre cele mai utilizate metode de interpolare folosită în astfel de cazuri este metoda *distanţei inverse* sau *distanţa inversă ponderată*. Această tehnică de interpolare permite aproximarea suprafeţei (funcţiei $z = z(x, y)$) pe baza nodurilor date de vectorii x şi y ale căror elemente nu sunt în mod necesar echidistante. Esenţa metodei constă în presupunerea intuitivă că suprafaţa de interpolare este influenţată mai puternic de punctele apropiate unele de altele şi într-o măsură mai mică de punctele mai împrăştiate. Astfel, suprafaţa interpolată reprezintă o medie

ponderată a punctelor împrăştiate şi ponderea asociată fiecărui punct se diminuează pe măsură ce distanța de la nod la punctul de interpolare creşte. În continuare se dă una dintre cele mai utilizate metode de ponderare a distanței numită *metoda lui Shepards*, care se remarcă prin simplitatea algoritmului său.

Ecuația utilizată pentru aproximarea funcției este:

$$F(x,y)=\sum_{k=1}^{n} w_k f_k \ ,$$

în care n este numărul nodurilor împrăştiate, f_k sunt valorile date ale funcției în noduri şi w_k reprezintă funcția pondere asociată fiecărui nod, care are valori de la 0 la 1.

Expresia clasică a funcției de ponderare este:

$$w_k=\frac{d_k^{-p}}{\sum_{j=1}^{n} d_j^{-p}},$$

unde p este un număr arbitrar real, pozitiv numit exponent de ponderare cu valoarea implicită egală cu 2, iar d_k este valoarea distanței dintre nodul de interpolare şi punctul de interpolare curent, adică:

$$d_k=\sqrt{(x_i-x_k)^2+(y_i-y_k)^2}$$

unde (x_i,y_i) sunt coordonatele punctului curent de interpolare şi (x_k,y_k) sunt coordonatele fiecărui nod dat, k=1,…,n.

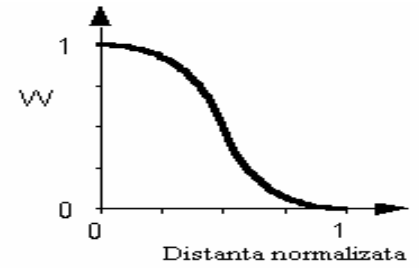
Se observă că expresia clasică prezentată mai sus pentru calculul ponderii ia în considerare distanțele la toate nodurile de interpolare, lucru care în general nu este avantajos. Din acest motiv, în cele mai multe implementări practice, metoda distanței inverse ponderate este ameliorată prin:

- reducerea numărului de noduri luate în considerare la calculul funcției pondere,
- adoptarea unei expresii analitice particulare pentru calculul funcției pondere.

O asemenea variantă îmbunătățită stă la baza *metodei triangulației* a lui *Delaunay* care se aplică datelor în procesul de interpolare bidimensională după o schemă locală astfel:

- în jurul fiecărui punct de interpolare se construieşte câte un *triunghi împrejmuitor* cu vârfurile în nodurile vecine,
- pentru fiecare punct de interpolare, în interiorul unui triunghi există trei ponderi diferite de zero,
- pentru calculul ponderilor se adoptă funcții cubice sau liniare (implicit).

Alura tipică a funcției pondere corespunde unei cubice (S-shaped)



În Matlab, interpolarea prin metoda distanței inverse se bazează pe algoritmul Delaunay şi este implementată prin funcția `griddata`, care poate fi apelată cu una din sintaxele următoare:

```
          zi=griddata(x,y,z,xi,yi)
[xi,yi,zi]=griddata(x,y,z,xi,yi)
[xi,yi,zi]=griddata(x,y,z,xi,yi,'method')
```

în care parametrii au semnificația descrisă anterior la funcția `interp2`, cu mențiunea că ansamblul metodelor de interpolare nu acceptă metoda 'spline' în schimb include metoda '`v4`', care furnizează o aproximare mai netedă a suprafeței interpolate (similară medodei 'cubic') dar care nu foloseşte algoritmul triangulației Delaunay.

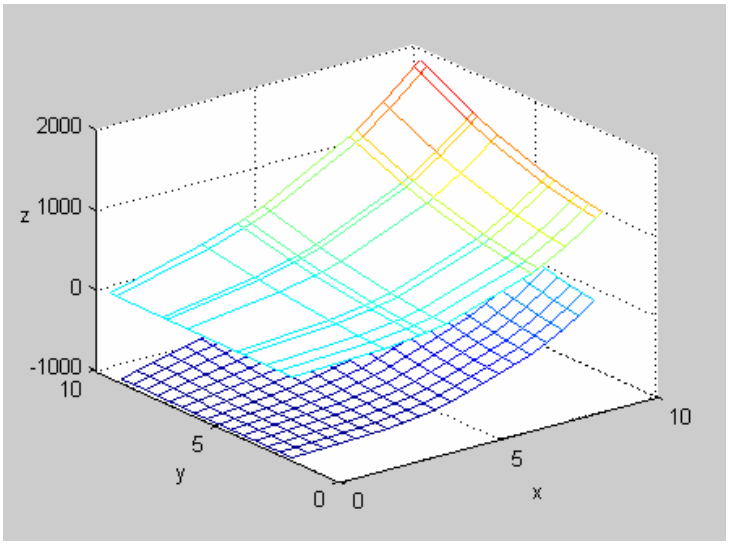
Aplicație. Exemplu de utilizare a funcției `griddata` pentru aproximarea funcției $z = x \cdot (x^2 + y^2)$ prin interpolare pe puncte împrăştiate, generate aleator.

Se utilizează programul de mai jos care efectuează următoarele operații: generează aleator vectorii coordonatelor nodurilor (evident, cu pas neuniform) pe care îi ordonează apoi monoton

crescător, generează rețeaua nodurilor în planul xy cu funcția `meshgrid`, determină valorile funcției în noduri, generează rețeaua punctelor de interpolare (cu pas constant), efectuează interpolarea propriuzisă cu funcția `griddata` (metoda cubic) şi apoi reprezintă pe acelaşi grafic suprafața definită de noduri şi suprafața obținută prin interpolare.

```
clear
x = 10.*rand(1,10);
y = 10.*rand(1,10);
x=sort(x)
y=sort(y)
y = y';
[X,Y] = meshgrid(x,y)
% functia de reprezentat
z = X.*(X.^2 + Y.^2)
xi = min(x): 0.5 : max(x);
yi = min(y): 0.5 : max(y);
[Xi ,Yi] = meshgrid(xi,yi)
zi=griddata(X,Y,z,Xi,Yi,'cubic')
mesh(X,Y,z);
hold on;
mesh(Xi, Yi, zi-1000)
```

Suprafața definită de nodurile de interpolare (sus) şi suprafața definită de punctele obținute în urma interpolării bidimensionale au aspectul următor:



C. Interpolarea funcţiilor de trei variabile

Aproximarea funcţiilor de trei variabile $w = f(x, y, z)$ prin interpolare se efectuează cu funcţia `interp3` în cazul datelor uniforme (echidistante) care se apelază cu una din următoarele sintaxe:

```
wi=interp3(x,y,z,w,xi,yi,zi)
wi=interp3(x,y,z,w,xi,yi,zi,'method')
```

Argumentul `'method'` admite specificaţiile 'nearest', 'linear', 'cubic' şi 'spline'.

În cazul datelor distribuite neuniform pentru interpolare se foloseşte funcţia griddata3 cu sintaxele:

```
wi=griddata3(x,y,z,w,xi,yi,zi)
wi=griddata3(x,y,z,w,xi,yi,zi,'method'),
```

unde argumentul `'method'` admite numai metodele 'linear' (implicit) sau 'nearest'.

Din punct de vedere geometric, funcţiile de trei sau mai multe variabile reprezintă *hipersuprafeţe* în spaţiul 3D sau n-dimensional.

Funcţia `[X,Y,Z] = meshgrid(x,y,z)` generează tablourile 3D necesare evaluării funcţiilor de trei variabile în spaţiul (volumul) de definiţie al acestora.

D. Interpolarea funcţiilor de *n* variabile

Pe un spaţiu n-dimensional se pot aproxima prin interpolare funcţii de forma $v = f(x_1, x_2, ..., x_n)$ folosind funcţia Matlab `interpn` cu sintaxe sub forma generală:

```
vi = interpn(x1,x2,x3,...,v,y1,y2,y3,...)
vi = interpn(v,y1,y2,y3,...)
vi = interpn(v,ntimes)
vi=interpn(...,'method')
```

Funcţia `interpn` lucrează pentru orice tablou de date multidimensional cu două sau mai multe dimensiuni. Se obţin valorile interpolate vi, pe baza valorilor funcţiei v în punctele date x1, x2,... xn. Pentru *n* variabile, funcţia `interpn` trebuie apelată cu *2n+1* argumente.

Condiții de utilizare:

- Tablourile x1, x2, x3, etc. conțin punctele (nodurile) în care sunt date valorile v ale funcției. Valorile care depăşesc limitele admise de Matlab sunt returnate ca NaN.
- y1, y2, y3, etc. trebuie să fie matrici sau vectori de aceeaşi dimensiune ce conțin punctele de interpolare.
- Funcția `interpn` impune ca vectorii x1, x2, x3, etc. să fie monotoni şi cu elementele echidistante. Acest lucru se poate face cu funcția `ndgrid`, similară cu `meshgrid`.
- Se admite ca vectorii x1, x2, x3, etc. să aibă elementele neechidistande.

Observație. Vectorii argument care nu au aceeaşi mărime fie sunt identic orientați (adică ambii sunt ori vectori coloană ori vectori linie), sau sunt prelucrați cu funcția `ndgrid` pentru a crea tablourile y1,y2,y3, etc..

Utilizarea sintaxei `vi=interpn(v,y1,y2,y3,...)` presupune `x1=1:size(v,1)`, `x2=1:size(v,2)`, etc.

Cu sintaxa `vi=interpn(v,ntimes)` tabloul multidimensional v este expandat prin interpolarea intercalată între fiecare element, lucrând recursiv de `ntimes`-ori. Apelul `vi=interpn(v)` este echivalent cu `interpn(v,1)`.

Cu sintaxa `vi=interpn(...,'method')` se pot specifica metodele de interpolare acceptate: 'linear' (metoda implicită), 'cubic', 'nearest', 'spline'.

5.7.6. Aproximarea datelor

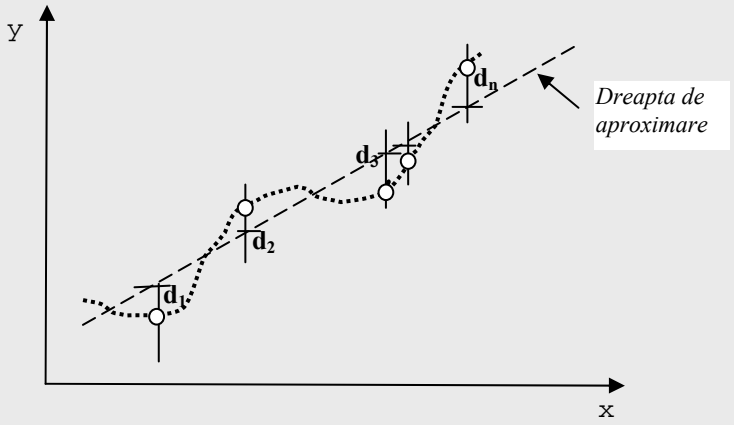
Se pune problema aproximării unui set de date printr-o singură formă analitică cu erori minime acceptate. Această metodă matematică se numeşte *regresie*, iar criteriul utilizat pentru minimizarea erorilor de aproximare este de natură pătratică (criteriul celor mai mici pătrate).

a) Regresia liniară

Dacă avem la dispoziție un set de valori y_i, i=1,...,n observate în funcție de o anumită variabilă x_i, i=1,...,n, se propune o aproximare a tendinței de distribuție a acestora printr-o dreaptă de evoluție. Regresia liniară reprezintă astfel, aproximarea unui set de date printr-o dependență

liniară utilizând drept criteriu de aproximare suma pătratelor distanțelor dintre dreapta propusă pentru aproximare şi punctele date. Se pune problema determinării dreptei care aproximează cel mai bine întregul set de date aplicând un criteriu de optim global constând în minimizarea sumei pătratelor distanțelor. Această condiție reprezintă o măsură a calității aproximării şi în acelaşi timp constituie un concept şi o tehnică generală de optimizare cunoscută sub denumirea de *metoda celor mai mici pătrate*. Această metodă nu impune condiția ca dreapta (curba) de aproximare să treacă prin punctele date, ceea ce diferențiază fundamental conceptul de aproximare a datelor, de cel de interpolare.

Mai jos este sintetizată sub formă grafică şi analitică esența acestei metode de aproximare optimă.



Criteriul celor mai mici pătrate evaluează măsura calității aproximării prin:

$$d_{pm} = \min_{i=\overline{1,n}} \left(\sum_{i=1}^{n} d_i^2 \right), \text{ unde } d_i = \left| y - y_i \right| .$$

În Matlab determinarea dreptei de aproximare în sensul celor mai mici pătrate se face cu funcția `polyfit` cu sintaxa următoare:

```
coef=polyfit(x,y,1)
```

Aceasta returnează prin variabila de ieşire (de tip vector linie) notată `coef` coeficienții `m`-*panta* şi `n`-*ordonata la origine* corespunzători dreptei de

aproximare $y1 = mx + n$, în sensul celor mai mici pătrate. Variabilele de intrare `x, y` sunt vectori linie cu aceeaşi dimensiune, conținând coordonatele punctelor date (cu `x` ordonat crescător), iar ultimul argument de intrare este 1 pentru dreaptă.

Criteriul de aproximare constând în evaluarea sumei pătratelor distanțelor se poate calcula cu secvența de instrucțiuni Matlab următoare:

```
coef=polyfit(x,y,1)
m=coef(1)
n=coef(2)
y1=m*x+n
sum_p=sum((y-y1).^2)
```

b) Regresia polinomială

Prin regresie polinomială se efectuează o aproximare a unui set de date cu un polinom de forma:

$$p(x) = \sum_{i=1}^{n} a_i x^{n+1-i} = a_1 x^n + a_2 x^{n-1} + ... + a_{n-1} x^2 + a_n x + a_{n-1},$$

care în punctele date x, are *în sensul celor mai mici pătrate*, valorile conținute de vectorul $y(x_{date})$. Acest polinom de regresie se determină tot cu funcția Matlab `polyfit` cu următoarea sintaxă:

```
p=polyfit(x,y,n),
```

unde `p` este vectorul coeficienților polinomului de aproximare, iar argumentul `n` precizează gradul acestuia. Vectorii x, y conțin datele de aproximat.

Observații.

Dacă setul de date are n elemente, toate datele se află pe curba de aproximare. Deci, polinomul de aproximare va trece prin toate punctele date dacă `n=length(x)`.

Pentru gradul polinomului mai mic decât numărul de date, adică `n<length(x)`, acesta nu mai trece prin toate punctele date, iar aproximarea este cu atât mai bună cu cât gradul polinomului este mai apropiat de numărul de date.

Utilizarea unui polinom cu grad mai mare decât numărul de date `n>length(x)` conduce în general la erori de aproximare considerabile. În acest caz polinomul de aproximare nu este unic.

Particularități de utilizare a funcției `polyfit`.

Funcția `polyfit` se poate utiliza şi cu sintaxe care extind opțional lista parametrilor de ieşire cu anumite variabile, astfel:

```
   [p,s] = polyfit(x,y,n)
[p,s,mu] = polyfit(x,y,n)
```

unde, pe lângă coeficienții polinomului de aproximare p, funcția mai returnează unele date utile în procesul de aproximare, după cum urmează:

- Structura de date `s` care va fi utilizată ulterior de funcția `polyval` la estimarea erorilor de predicție în procesul de generare a polinomului de aproximare. Astfel, dacă erorile datelor de aproximat `y` au distribuție normală, fiind independente cu varianță (σ^2) constantă, funcția `polyval` va produce limite ale erorii ce vor include cel puțin 50% dintre predicții. Structura de date `s` conține trei variabile: factorul Cholesky al matricei Vandermonde (`R`), numărul gradelor de libertate (`df`) şi norma reziduurilor (`normr`).
- Variabila `mu` este de tip vector coloană cu două elemente, astfel: `mu(1)=mean(x)`, respectiv `mu(2)=std(x)`. În acest caz, funcția `polyfit` va căuta coeficienții polinomului de aproximare nu pentru punctele vectorului `x` ci în punctele unui vector centrat şi scalat `xhat` de medie 0 şi varianță 1, calculat cu expresia:

$$\text{xhat} = \frac{\left[\text{x} - \text{mu}(1)\right]}{\text{mu}(2)} = \frac{\text{x} - \mu}{\sigma},$$

Această transformare a vectorului de date inițial, prin centrare şi scalare simetrică, îmbunătățeşte proprietățile numerice ale polinomului şi ale algoritmului de aproximare în sine.

Exemplu. În continuare se prezintă un program Matlab care rezolvă comparativ problema aproximării a şase date furnizate prin vectorii `x`, `y`, utilizând regresia liniară şi regresia polinomială de diferite ordine cu funcția `polyfit`. Rezultatele se reprezintă grafic, construind polinoamele cu funcția `polyval` într-un număr mai mare de puncte si utilizând funcția

`plot` succesiv pe acelaşi sistem de axe. Valorile date sunt reprezentate cu simbolul cerculeț (o).

```
%aproximarea datelor
clear
x=[0, 1, 2, 3, 4, 5];
y=[-10 10 0 100 60 20];
%cu dreapta de regresie
 coef=polyfit(x,y,1)
y1=coef(1)*x+coef(2);
 plot(x,y,'ok',x,y1,'-k')
%cu polinom de ordin 2
 p2=polyfit(x,y,2)
 x1=min(x):0.1:max(x);
 poli2=polyval(p2,x1);
 hold on
 plot(x1,poli2,'-.k')
%cu polinom de ordin 4
 p4=polyfit(x,y,4)
 poli4=polyval(p4,x1);
 hold on
 plot(x1,poli4,'--k')
%cu polinom de ordin 6
 p6=polyfit(x,y,6)
 poli6=polyval(p6,x1);
 hold on
 plot(x1,poli6,'.k')
%cu polinom de ordin 8
 p8=polyfit(x,y,8)
 poli8=polyval(p8,x1);
 hold on
 plot(x1,poli8,':k')
```

```
coef =

   11.4286    1.4286


p2 =

   -7.5000   48.9286  -23.5714


p4 =

     0.0000   -5.5556   34.1667
    -27.1825   -6.9048


p6 =

     1.5481  -16.7222   50.3426
0  -164.5574  149.3889  -
10.0000


 p8 =

     0.2023   -2.1265    6.1172
0  -13.7658   0          0
29.5727    -10.0000
```