6. Interpolacja i aproksymacja

1. Interpolacja

Zagadnienie interpolacji można sformułować następująco: w przedziale <a,b> danych jest n+1 różnych punktów x_1, x_1, ..., x_n, które nazywamy ***węzłami interpolacji***, oraz wartości pewnej funkcji $y=f(x)$ w tych punktach $f(x_0)=y_0$, $f(x_1)=y_1$,..., $f(x_n)=y_n$.

Zadaniem interpolacji jest wyznaczenie przybliżonych wartości funkcji w punktach niebędących węzłami oraz oszacowanie błędu tych przybliżonych wartości. W tym celu należy znaleźć funkcję $F(x)$ zwaną ***funkcją interpolującą***, która w węzłach interpolacji przyjmuje takie same wartości jak funkcja $f(x)$.

Funkcje interpolujące

y_i=interp1(x, y, x_i, metoda) - zwraca wektor y_i, będący wartościami funkcji jednej zmiennej $y=f(x)$ w punktach określonych wektorem x_1; węzły interpolacji określają wektory x i y, metoda to łańcuch znaków określający metodę interpolującą.
z_i=interp2(x, y, z, x_i, y_i, metoda) - zwraca macierz z_1 zawierającą wartości funkcji dwóch zmiennych $z=f(x,y)$ w punktach określonych wektorami x_i i y_i; węzły interpolacji określają macierze x, y, z.
v_i=interp3(x, y, z, v, x_i, y_i, z_i, metoda) - interpolacja funkcją trzech zmiennych, analogicznie jak w **interp2**.
v_i=interpn(x_1, x_2, x_3,...v, x_i, y_i, z_i, metoda) - interpolacja funkcją n zmiennych, analogicznie jak w **interp2**.

Metody interpolacji
'linear' - interpolacja liniowa,
'spline' - interpolacja funkcjami sklejanymi stopnia trzeciego,
'cubic' lub 'pchip' - interpolacja wielomianami stopnia trzeciego,
'nearest' - interpolacja wartością najbliższego sąsiada.
Wszystkie metody interpolacji wymagają, aby ciąg **x** był monotoniczny.

Przykład 1

```
clear all;
clf;
%węzły interpolacji
x=[-1 0 1 2 3];
y=[3 5 4 2 6];
%punkty w ktorych znajdujemy wartosci interpolowane
xi=-1:0.2:3;
%interpolacja liniowa
yi_lin=interp1(x,y,xi, 'linear');
%interpolacja funkcjami sklejanymi
yi_spline=interp1(x,y,xi, 'spline');
%wykresy
plot(x,y,'*',xi,yi_lin,':',xi,yi_spline,'--');
```

Przykład 2. Testowanie różnych metod interpolacji

```
 clf;
f=inline('2.*sin(x)+sin(2.*x)+sin(3.*x+pi)') %interpolowana funkcja
%węzły interpolacji
x=[0:20];
y=f(x);
%metody interpolacji
x_int=[0:0.1:20];
y_int1=interp1(x,y,x_int,'spline');
y_int2=interp1(x,y,x_int,'pchip');
y_int3=interp1(x,y,x_int,'nearest');
y_int4=interp1(x,y,x_int,'linear');
%wykresy
subplot(4,1,1), plot(x,y,'*',x_int,y_int1,'-r')
input('press any key...')
hold on, ezplot(f,[0,20]);
subplot(4,1,2), plot(x,y,'*',x_int,y_int2,'-r')
input('press any key...')
hold on, ezplot(f,[0,20]);
subplot(4,1,3), plot(x,y,'*',x_int,y_int3,'-r')
input('press any key...')
hold on, ezplot(f,[0,20]);
subplot(4,1,4), plot(x,y,'*',x_int,y_int4,'-r')
input('press any key...')
hold on, ezplot(f,[0,20]);
```

2. Interpolacja Lagrange'a

Interpolacja za pomocą wielomianu polega na znalezieniu takiego wielomianu $L_n(x)$ stopnia co najwyżej n, aby wartości tego wielomianu i funkcji interpolowanej w węzłach były sobie równe, czyli: $L_n(x) = a_0+a_1x+a_2x^2+...+ a_nx^n$
Zadanie interpolacji wielomianowej ma jednoznaczne rozwiązanie, czyli istnieje tylko jeden wielomian spełniający warunek. Wartości współczynników wielomianu wyliczane są ze wzoru Lagrange'a. Szukany wielomian ma postać:

$$L_n(x) = \sum_{j=0}^{n} y_j \frac{(x-x_o)(x-x_1)...(x-x_{j-1})(x-x_{j+1})...(x-x_n)}{(x_j-x_o)(x_j-x_1)...(x_j-x_{j-1})(x_j-x_{j+1})...(x_j-x_n)}$$

przyjmując, że : $\omega_n(x) = (x-x_o)(x-x_1)...(x-x_{j-1})(x-x_{j+1})...(x-x_n)$

wzór Lagrange'a ma wtedy postać:

$$L_n(x) = \sum_{j=0}^{n} y_j \frac{\omega_n(x)}{(x-x_j)\omega_n(x_j)}$$

gdzie $y_j=y(x_j)$, a $\omega_n(x_j)$ jest wartością pochodnej wielomianu $\omega(x)$ w punkcie x_j. Pozwala to na napisanie procedury numerycznej do obliczania wartości wielomianu stopnia n w dowolnym punkcie leżącym w przedziale <a,b>.

Przykład 3. skrypt sprawdza interpolację wielomianami Lagrange'a

```
%skrypt sprawdza interpolację wielomianami Lagrange'a
clear all; close all;
funk=inline('1./(1+25*x.^2)');
x=linspace(-1,1,201);
y=funk(x);
ia=linspace(1,201,3);
ib=linspace(1,201,5);
ic=linspace(1,201,9);
xia=x(ia);
yia=y(ia);
x1a=x;
x1a(ia)=[];
ya=lagrange(xia,yia,x1a,2);
xib=x(ib); yib=y(ib);
x1b=x; x1b(ib)=[];
yb=lagrange(xib,yib,x1b,4);
xic=x(ic); yic=y(ic);
x1c=x; x1c(ic)=[];
yc=lagrange(xic,yic,x,8);
clf;
subplot(2,2,1);
ezplot(funk,[-1,1])
input('press any key...');
ylim([-1.5,1.5]);
title('funkcja interpolowana:1/(1+25x^2)');
subplot(2,2,2);
%ezplot(funk,[-1,1]),hold on
plot(xia,yia,'o'); hold on;
plot(x1a,ya,'--')
input('press any key...');
ylim([-1.5,1.5]);
title('f. interpolowana: wielomian 2-go rzędu');
subplot(2,2,3);
%ezplot(funk,[-1,1]),hold on,
plot(xib,yib,'o'); hold on;
plot(x1b,yb,'--')
input('press any key...');
ylim([-1.5,1.5]);
title('f. interpolowana: wielomian 4-go rzędu');
subplot(2,2,4);
ezplot(funk,[-1,1]),hold on,
plot(xic,yic,'o'), hold on;
plot(x,yc,'--')
ylim([-1.5,1.5]);
title('f. interpolowana: wielomian 8-go rzędu');
```

```
--------------to oddzielny plik o nazwie lagrange.m-------------------------------

function [y]=lagrange(xj,yj,x,n);
%wzór interpolacyjny Lagrange'a
%funkcja lagrange oblicza wartości wielomianu interpolacyjnego stopnia n w dowolnym
%punkcie dla węzłów xj i odpowiadających im wartości funkcji interpolowanej yj
%np.  y=lagrange([1 2 3], [1 2 3],1.7,2)  daje y=1.7
omega_w=1;s=0;
for j=1:n+1
    omega_w=omega_w.*(x-xj(j));
    omega_p=1;
    for i=1:n+1
        if i~=j
            omega_p=omega_p.*(xj(j)-xj(i));
        end
    end
    s=s+yj(j)./(omega_p.*(x-xj(j)));
end
y=omega_w.*s;
```

3. Aproksymacja

Załóżmy, że w przedziale <a, b> jest n+1 punktów x_o = a, x_1, x_2, ...,x_n=b oraz wartości pewnej funkcji $y=f(x)$ w tych punktach. Poszukiwana jest funkcja $F(x)$ dobrze przybliżająca $f(x)$ w przedziale <a,b>, z tym że w odróżnieniu od interpolacji, obydwie funkcje nie muszą mieć takich samych wartości w podanych punktach. Pojawia się zatem błąd aproksymacji zdefiniowany jako różnica między wartościami funkcji aproksymującej F i funkcji aproksymowanej f w punktach xi, i=0, 1, ..., n. Zadaniem metody numerycznej jest w tym przypadku minimalizacja błędu aproksymacji.

Aproksymacja jest nazywana również dopasowywaniem (fitowaniem) krzywej do danego zbioru danych. Jeżeli punkty (x_i, y_i) pochodzą z badań eksperymentalnych nad procesami fizycznymi, to zwykle ich wartości są obarczone błędami pomiaru. Interpolacja, czyli wymóg znalezienia funkcji przechodzącej dokładnie przez te punkty, w tym przypadku nie ma sensu.

Jedną z najpopularniejszych metod aproksymacji jest aproksymacja średniokwadratowa, zwana także metodą najmniejszych kwadratów. Polega ona na minimalizacji błędu aproksymacji zdefiniowanego jako suma kwadratów odchyłek we wszystkich punktach obliczeniowych:

$$\sum_{i=0}^{n} w(x_i)\left[F(x_i) - f(x_i)\right]^2$$

Do poprawy przybliżenia w wybranych węzłach może być wykorzystana funkcja wagowa w(x_i). Zwykle jest ona tożsamościowo równa jedności, zatem wszystkie odchyłki są jednakowo istotne. Podstawową funkcją jaką oferuje Matlab do rozwiązania zadania aproksymacji jest funkcja **polyfit**.

a = polyfit(x, y, n) - znajduje współczynniki wielomianu n-tego stopnia, który w sensie najmniejszych kwadratów najlepiej pasuje do serii danych pomiarowych (x, y).

Uwaga do obliczania wartości wielomianu: $W(x, a)=a_1x^n+ a_2x^{n-1}+...+ a_nx+ a_{n+1}$

o współczynnikach **a** w punkcie x można wykorzystać funkcję polyval(a, x), gdzie a(1) jest współczynnikiem stojącym przy x^n, a(2) jest współczynnikiem stojącym przy x^{n-1}, ..., a(n+1)jest wyrazem wolnym.

Przykład 4

Przypuśćmy, że mamy następujące dane x_i oraz y_i i chcielibyśmy znaleźć najlepsze liniowe dopasowanie do tych danych.

(x_i, y_i): (2,3); (4,5); (7,15); (12,22); (20,42)

```
%fit do danych pomiarowych
clf;
x=[2 4 7 12 20];
y=[3 5 15 22 42];
figure, plot(x, y,'o');
a=polyfit(x,y,1);
b=polyfit(x,y,2);
input('any key');
xt=0:25;
yt=polyval(a,xt);
yt1=polyval(b,xt);
hold on, plot(xt,yt,'r-');
hold on, plot(xt,yt1,'g-');
legend('dane pomiarowe', 'dopasowanie 1-go', 'dopasowanie 2-go');
```

Poza funkcją polyfit Matlab daje użytkownikowi narzędzie **BasicFitting**, uruchamiane w menu *Tools* w oknie graficznym.

Przykład 5 a i b

```
%Wykorzystanie narzędzia Basic Fitting
t=linspace(0,10,10);    %momenty pomiaru
y=sin(0.5*t)+0.1*randn(size(t));  %wygenerowane dane pomiarowe
plot(t,y,'o');
```

a.) Po narysowaniu surowych danych należy przejść do okna graficznego i z menu *Tools* wybrać *Basic Fitting*. Zaznaczamy kolejno opcje q*uadratic*, *qubic* oraz *4th degree polynomial*, a następnie zaznaczamy opcje: *Show equations*, *Plot residuals* oraz *Show norm of residuals*.

b.) Ponownie uruchamiamy skrypt, ale zamiast rysowania wykresu wpisujemy z linii komend >***cftool***. Próbujemy dopasowywać różne funkcje dopasowujące oferowane przez narzędzie *cftool.*

Przykad 6 - pompowanie (dopasowanie krzywej wykładniczej)

(t_i, p_i): (0, 760); (0.5, 625); (1, 528); (5, 85); (10, 14); (20, 0.16)

```
clear;
clc;
clf;
t=[0 0.5 1 5 10 20]
p=[760 625 528 85 14 0.16]
pp=log(p)  % dane przekształcone
a=polyfit(t,pp,1); %fit liniowy
po=exp(a(2)); %obliczenie stałej p₀
T=-1/a(1);%obliczenie stałej T
disp(['stala po = ',num2str(po),', stała T = ',num2str(T)]);
input('any key...');
%wykres w skali liniowej
twyk=linspace(0,20,20);  %dane do wykresu
pwyk=po*exp(-twyk/T);
subplot(2,1,1);
plot(t,p,'o',twyk,pwyk,'r-'), grid on
xlabel('czas t (s)'), ylabel('Cisnienie');
% wykres w skali półlogarytmicznej
pwyk_log=exp(polyval(a,twyk));
subplot(2,1,2)
semilogy(t,p,'o',twyk,pwyk_log,'r-'),grid on
xlabel('czas t [s]'), ylabel('Cisnienie');
```

4. Fitowanie powierzchni

```
% Strumień na obszarze Błoni
%
clear;
clc;
dane_blonia=importdata('blonia.txt');
x=dane_blonia(:,1);
y=dane_blonia(:,2);
f=dane_blonia(:,3);
```

1. Wykorzystanie narzędzia *sftool*

```
%{SFTool - Surface Fitting Tool - otwieramy komendą sftool i dopasowujemy
powierzchnię: wybieramy x,y,f jako X,Y,Z,
Createsurfacefit(x,y,f); %{utworzona przez nas w SFTOOL funkcja -
%dopasowuje i rysuje%};
title('polecenie CreateSurfaceFit');
```

2. Polecenie TriScatteredInterp

```
[X Y]=meshgrid(19.9:0.0001:19.925,50.057:0.0001:50.063);
```

```
A=TriScatteredInterp(x,y,f,'natural');
Z=A(X,Y);
figure;
surf(X,Y,Z);
Title('polecenie TriScatteredInterp');
shading interp;
```

3.Polecenie griddata

```
[XX YY]=meshgrid(19.9:0.00001:19.925,50.057:0.00001:50.063);
ZZ=griddata(x,y,f,XX,YY,'cubic');%jeszcze linear i nearest
%subplot(3,1,2);
figure;
surf(XX,YY,ZZ);
Title('polecenie griddata');
shading interp;
```

4. Polecenie *contour*.

```
 %subplot(3,1,3);
figure;
contour(X,Y,Z,10);
```