Interpolacja

1. Wstęp

Interpolacja jest zagadnieniem dotyczącym znajdowania funkcji przechodzącej przez zadane punkty, tzw. węzły interpolacji. Ma zasadnicze znaczenie w przetwarzaniu sygnałów i obrazów.
Najbardziej popularne metody interpolacji to:

- wielomianowa (dopasowywanie wielomianów do punktów pomiarowych)
- oparta na transformacie Fouriera (wektor pomiarów funkcji okresowej poddaje się transformacie, a następnie wykonuje się transformatę odwrotną używając większej liczby punktów)

2. Zadania praktyczne

2.1. Interpolacja jednowymiarowa

Wygeneruj przykładowe punkty pomiarowe (np. za pomocą funkcji *sin*) dla współrzędnych *x* od 0 do 9 z krokiem 1. Narysuj je:

```
x = 0:9;          %wektor od 0 do 9 z krokiem 1
y = sin(x);       %sinus
plot(y)
```

Przeprowadź interpolację liniową danych „pomiarowych” używając czterokrotnie mniejszej wartości kroku niż ta, której użyto w trakcie pomiarów:

```
xi = 0:0.25:9;              %wektor od 0 do 9 z krokiem 0.25
yi = interp1(x,y,xi);       %interpolacja
plot(x,y,'o',xi,yi)         %wykres oryginalny i interpolowany
```

Pokaż punkty dla których policzona została interpolacja:

```
plot(x,y,'o',xi,yi,'.-')    %wykres oryginalny i interpolowany
```

Do przeprowadzenia obliczeń została użyta funkcja *interp1* o składni:
yi = interp1(x, y, xi, ‘metoda’)
gdzie:
x – zbiór posortowanych rosnąco współrzędnych węzłów interpolacji,
y – zbiór odpowiadających wartości dla tych współrzędnych,
xi – zbiór nowych współrzędnych dla których przeprowadzona ma być interpolacja,
yi – zbiór odpowiadających wartości dla nowych współrzędnych,
‘metoda’ może przybierać wartości:

- **‘linear’** – interpolacja funkcją liniową (metoda domyślna),
- **‘spline’** - interpolacja funkcjami sklejanymi trzeciego stopnia,
- **‘cubic’** – interpolacja wielomianami trzeciego stopnia.

Porównaj wyniki interpolacji dla pozostałych metod:

```
yi2 = interp1(x,y,xi,'cubic');
yi3 = interp1(x,y,xi,'spline');
plot(x,y,'o',xi,yi2,xi,yi3)
legend('pomiary','cubic','spline',0)
```

Interpolacja oparta na transformacie Fouriera za pomocą funkcji *interpft*:

```
N = length(y);        %długość wektora "danych"
L = 4;                %współczynnik "gęstości" interpolacji
M = N*L;              %liczba danych po interpolacji
yi4 = interpft(y,M);  %interpolacja
%przeliczony wektor współrzędnych x
xi4 = 0:0.25:9+0.25*(length(yi4)-length(yi));
plot(x,y,'o',xi4,yi4,'*-')
legend('pomiary','interpolacja',0)
```

Narysuj wszystkie metody na jednym rysunku:

```
plot(x,y,'o',xi,yi2,xi,yi3,xi4,yi4)
```

Różnice między poszczególnymi metodami interpolacji wyraźniej widać w przypadku gdy zjawisko opisywane zbiorem węzłów interpolacji nie ma charakteru monotonicznego, jak w poniższym ćwiczeniu.

- **Zadanie do samodzielnego wykonania**

Porównaj jakość dostępnych w funkcji **`interp1`** metod interpolacji na przykładzie funkcji

y(x) = exp(-0.5) cos(5x)

w przypadku równoodległych węzłów interpolacji, których współrzędne leżą w przedziale [0,2]. Przyjmij wektor współrzędnych x = 0:0.4:2 oraz nowy wektor współrzędnych xi = 0:0.05:2. Narysuj wykresy funkcji oraz policz błąd średniokwadratowy dla poszczególnych metod.

2.2. Interpolacja dwuwymiarowa

Wygeneruj funkcję dwóch zmiennych w niskiej rozdzielczości:

```
close all
[x,y] = meshgrid(-3:1:3);  %płaszczyzna w niskiej rozdzielczości (7x7)
z = peaks(x,y);            %przykładowa funkcja
surf(x,y,z)                %wizualizacja trójwymiarowa
```

Funkcja *meshgrid* oblicza punkty siatki dwuwymiarowej, na której będziemy generować powierzchnię trójwymiarową. Funkcja *peaks* oblicza wartości funkcji dwóch zmiennych poprzez translację i skalowanie rozkładu Gaussa – posłuży nam jako funkcja testowa. *Surf* rysuje funkcję w przestrzeni trójwymiarowej.

Przeprowadź interpolację danych „pomiarowych” używając czterokrotnie mniejszej wartości kroku niż ta, której użyto w trakcie pomiarów:

```
%siatka o większej rozdzielczości
[xi,yi] = meshgrid(-3:0.25:3);
%interpolacja metodą "najbliższego sąsiada"
zi1 = interp2(x,y,z,xi,yi,'nearest');
%interpolacja biliniowa
zi2 = interp2(x,y,z,xi,yi,'bilinear');
%interpolacja sześcienna
zi3 = interp2(x,y,z,xi,yi,'cubic');
```

Do przeprowadzenia obliczeń została użyta funkcja *interp2* o składni:
zi = interp2(x, y, z, xi, yi,'metoda')
gdzie:
z – macierz prostokątna zawierająca wartości funkcji dwuwymiarowej,
x, y – wektory o takiej samej długości, zawierające współrzędne punktów dla których obliczone są wartości **z**,
xi, yi – zbiór nowych współrzędnych dla których przeprowadzona ma być interpolacja,
zi – macierz prostokątna zawierająca wartości funkcji dla nowych współrzędnych
string **metoda** może przybierać wartości:

- **nearest** – interpolacja metodą „najbliższego sąsiada" (wartość punktu interpolowanego jest taka sama jak wartość najbliższego sąsiada)
- **linear** – dopasowuje biliniową powierzchnię, przechodzącą przez istniejące punkty (wartość punktu interpolowanego jest kombinacją wartości czterech najbliższych punktów)
- **cubic** – wartość punktu interpolowanego jest kombinacją wartości szesnastu najbliższych punktów

Narysuj wyniki:

```
%wizualizacja
figure
surf(xi,yi,zi1),title('nearest')
figure
surf(xi,yi,zi2),title('bilinear')
figure
surf(xi,yi,zi3),title('cubic')
```

Sprawdź także różne metody wizualizacji za pomocą funkcji:

```
surfc(xi,yi,zi3)      % dodaje kontur

surfl(xi,yi,zi3)      % dodaje efekt oświetlenia
shading interp        % metoda cieniowania
colormap(gray)        % mapa kolorów
```

Spróbuj różnych metod cieniowania:

```
shading flat
shading faceted
shading interp
```

Spróbuj różnych map kolorów, np.:

```
colormap(hsv)
colormap(jet)
colormap(pink)
```

(sprawdź w helpie możliwe argumenty funkcji *colormap*)