Interpolacja funkcjami sklejanymi

Funkcje sklejane:
Załóżmy, że mamy $n+1$ węzłów $t_0, t_1, \ldots, t_n$ takich, że $t_0 < t_1 < \ldots < t_n$.
Dla danej liczby całkowitej, nieujemnej k funkcją sklejaną stopnia k nazywamy taką funkcję S która:

1. W każdym z przedziałów $[t_i, t_{i+1})$ $(0 \le i \le n-1)$ jest wielomianem co najwyżej rzędu k,
2. Ma ciągłą $(k-1)$-szą pochodną w przedziale $[t_0, t_n]$.

Funkcje sklejane stopnia trzeciego:
Funkcja taka musi spełniać następujące warunki:

1. $S_{i-1}(t_i) = y_i = S_i(t_i)$ $(1 \le i \le n)$ - warunek ten zapewnia ciągłość funkcji S i pokrywanie się jej z węzłami,
2. $S'_{i-1}(t_i) = S'_i(t_i)$ $(1 \le i \le n-1)$ - ciągłość pierwszej pochodnej,
3. $S''_{i-1}(t_i) = S''_i(t_i)$ $(1 \le i \le n-1)$ - ciągłość drugiej pochodnej.

W każdym z przedziałów funkcja będzie miała następującą postać:

$$S_k(x) = S_{k,0} + S_{k,1} \cdot x + S_{k,2} \cdot x^2 + S_{k,3} \cdot x^3 \qquad \begin{array}{c} k = 0, 1, 2, \ldots, n-1 \\ x \in [t_k, t_{k+1}] \end{array}$$

Aby jednoznacznie określić taką funkcję musimy wyznaczyć cztery współczynniki. Ponieważ takich krzywych jest n to mamy $4n$ niewiadomych. Z pierwszego przytoczonego warunku otrzymamy $2n$ równań. Z drugiego i trzeciego po $2n-1$ równań. Wynika stąd, że mamy $4n-2$ równania. Zostają dwa stopnie swobody, które można wyzyskać na różne sposoby. Często przyjmuje się, że $S''(t_0) = S''(t_n) = 0$ lub określa się pochodne na końcach przedziału np. $S'(t_0) = 0$ i $S'(t_n) = 0$.

Wprowadźmy oznaczenia: $z_i = S''(t_i)$, $h_i = t_{i+1} - t_i$
Ponieważ, $S_k(x)$ jest funkcją stopnia co najwyżej trzeciego, to $S''_k(x)$ jest funkcją liniową. Możemy więc zapisać:

$$S''_i(x) = \frac{z_i}{h_i}(t_{i+1} - x) + \frac{z_{i+1}}{h_{i+1}}(x - t_i)$$

Całkując dwukrotnie obie strony tej równości otrzymujemy wzór na wielomian S_i

$$S_i(x) = \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \frac{z_{i+1}}{h_i}(x - t_i)^3 + C(x - t_i) + D(t_{i+1} - x)$$

gdzie C i D są stałymi całkowania, które wynikają z warunków interpolacyjnych $S_i(t_i) = y_i$, $S_i(t_{i+1}) = y_{i+1}$. Ostatecznie:

$$S_i(x) = \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \frac{z_{i+1}}{h_i}(x - t_i)^3 + (\frac{y_{i+1}}{h_i} - \frac{z_{i+1}h_i}{6})(x - t_i) + (\frac{y_i}{h_i} - \frac{z_i h_i}{6})(t_{i+1} - x)$$

Aby wyznaczyć nieznane wartości z_i różniczkujemy powyższy wielomian i podstawiamy $x = t_i$:

$$S'_i(t_i) = -\frac{h_i}{3}z_i - \frac{h_i}{6}z_{i+1} - \frac{y_i}{h_i} + \frac{y_{i+1}}{h_i}$$
$$S'_{i-1}(t_i) = \frac{h_{i-1}}{6}z_{i-1} + \frac{h_{i-1}}{3}z_i - \frac{y_{i-1}}{h_{i-1}} + \frac{y_i}{h_{i-1}}$$

Korzystając z warunku 2. otrzymujemy:

$$h_{i-1}z_{i-1} + 2(h_{i-1}+h_i) + h_i z_{i+1} = \frac{6}{h_i}(y_{i+1}-y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \qquad (1 \le i \le n-1)$$

Przyjmijmy oznaczenia:

$$u_i = 2(h_{i-1}+h_i), \qquad b_i = \frac{6}{h_i}(y_{i+1}-y_i), \qquad v_i = b_i - b_{i-1}.$$

Zakładając dodatkowo, że $z_0 = z_n = 0$, otrzymujemy układ równań:

$$\begin{bmatrix} u_1 & h_1 & & & \\ h_1 & u_2 & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}$$

Układ ten można rozwiązać metodą eliminacji Gaussa lub rozkładem LU.
Po obliczeniu wartości $z_0, z_1, \ldots, z_n$ można znaleźć wartości funkcji sklejanej S w dowolnym punkcie x. W tym celu najpierw należy wyznaczyć do którego z przedzaiałów

$$(-\infty, t_1), \quad [t_1, t_2), \quad \ldots, \quad [t_{n-2}, t_{n-1}), \quad [t_{n-11}, \infty)$$

należy x. Po wyznaczeniu przedziału stosujemy wzór:

$$S_i(x) = \frac{z_i}{6}\left[\frac{(t_{i+1}-x)^3}{h_i} - h_i(t_{i+1}-x)\right] - \frac{z_{i+1}}{6}\left[\frac{(t_i-x)^3}{h_i} - h_i(t_i-x)\right] + \frac{y_i(t_{i+1}-x) - y_{i+1}(t_i-x)}{h_i}$$

Rozwiązywanie układów trójprzekątniowych:
Do rozwiązywania układów trójprzekątniowych postaci:

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

używana jest specjalna forma metody eliminacji Gaussa, zwana także algorytmem Thomasa. Obliczenia rozpoczynamy następująco:

$$\begin{aligned} m &= \frac{a_k}{b_{k-1}} \\ b'_k &= b_k - mc_{k-1} \qquad k = 2, ..., n \\ d'_k &= d_k - md_{k-1} \end{aligned}$$

Następnie obliczamy wartość x_n:

$$x_n = \frac{d'_n}{b'_n}$$

Resztę wartości wektora x obliczmy ze wzoru:

$$x_k = \frac{d'_k - c_k x_{k+1}}{b'_k} \qquad k = n-1, ..., 1$$

Rozkład LU:
Rozważmy układ n równań liniowych z n niewiadomymi

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$$

Przypuśćmy, że macierz $\mathbb{A}$ można wyrazić jako iloczyn macierzy trójkątnej dolnej $\mathbb{L}$ i trójkątnej górnej $\mathbb{U}$: $\mathbb{A} = \mathbb{L}\mathbb{U}$. Macierze $\mathbb{L}$ i $\mathbb{U}$ przedstawiaja się następująco:

$$\mathbb{L} = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \qquad \mathbb{U} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

Wtedy rozwiązanie układu równań $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$ dzieli się na dwa etapy:

$$\mathbb{L}\boldsymbol{z} = \boldsymbol{b}$$
$$\mathbb{U}\boldsymbol{x} = \boldsymbol{z}$$

Jest to prostsze ponieważ każdy układ zawiera macierz trójkątną. Jeśli istnieją macierze $\mathbb{L}$ i $\mathbb{U}$ takie, że $\mathbb{A} = \mathbb{L}\mathbb{U}$, to mówimy, że $\mathbb{A}$ ma rozkład LU. Rozkład taki nie jest jednoznaczny. Dla każdego i możemy wybrać dowolną wartość, różną od zera dla jednej z liczb l_{ii} lub u_{ii} (ale nie dla obu). Najprostszym wyborem zdaje się być $l_{ii} = 1$ dla $i = 1, 2, ..., n$. Wtedy $\mathbb{L}$ jest jedynkowa trójkątna dolna. Inny prosty wybór to $u_{ii} = 1$ dla każdego i wtedy $\mathbb{U}$ jest jedynkowa trójkątna górna. Aby wyznaczyć elementy macierzy $\mathbb{L}$ i $\mathbb{U}$ zaczynamy od wzoru na mnożenie macierzy:

$$a_{ij} = \sum_{s=1}^{n} l_{is} u_{sj}$$

Wykorzystując własności macierzy $\mathbb{L}$ i $\mathbb{U}$: $l_{is} = 0$ dla $s > i$ oraz $u_{sj} = 0$ dla $s > j$ otrzymujemy:

$$a_{ij} = \sum_{s=1}^{min(i,j)} l_{is} u_{sj}$$

W każdym kolejnym etapie obliczeń będziemy wyznaczali jeden nowy wiersz macierzy $\mathbb{U}$ i jedną nową kolumnę macierzy $\mathbb{L}$. W k-tym kroku możemy więc założyć, że znamy już początkowych $k-1$ wierszy macierzy $\mathbb{U}$ i tyle samo kolumn macierzy $\mathbb{L}$:

$$a_{kk} = \sum_{s=1}^{k-1} l_{ks} u_{sj} + l_{kk} u_{kk}$$

Ponieważ ustaliliśmy jeden z elementów u_{kk} lub l_{kk}, to możemy wyznaczyć drugi z nich. Znając go możemy obliczyć k-ty wiersz macierzy $\mathbb{U}$ i k-tą kolumne macierzy $\mathbb{L}$:

$$a_{kj} = \sum_{s=1}^{k-1} l_{ks} u_{sj} + l_{kk} u_{kj} \qquad k+1 \le j \le n$$

$$a_{ik} = \sum_{s=1}^{k-1} l_{is} u_{sk} + l_{ik} u_{kk} \qquad k+1 \le i \le n$$

Skrypt 1:

```
function k = splineCurv(xData, yData)
% Zwraca wartosc drugich pochodnych w wezlach
% dane wejsciowe: xData - wspolrzedne x wezlow,
%                 yData - wspolrzedne y wezlow.

n = length(xData); c = zeros(n-1, 1); d = ones(n,1);
e = zeros(n-1,1); k = zeros(n,1);

c(1:n-2) = xData(1:n-2) - xData(2:n-1);
d(2:n-1) = 2*(xData(1:n-2) - xData(3:n);
e(2:n-1) = xData(2:n-1) - xData(3:n);

k(2:n-1) = 6*(yData(1:n-2)-yData(2:n-1))./(xData(1:n-2)-xData(2:n-1))
         - 6*(yData(2:n-1)-yData(3:n))./(xData(2:n-1)-xData(3:n));

[c,d,e] = LUdec3(c,d,e);
k = LUsol3(c,d,e,k);
```

Skrypt 2:

```
function y = splineEval(xData,yData,k,x)
% Zwraca wartosc funkcji interpolujacej w x
% dane wejsciowe: xData - wspolrzedne x wezlow,
%                 yData - wspolrzedne y wezlow,
% dane wyjsciowe: k - wartosci drugich pochodnych w wezlach.

i = findSeg(xData,x);
h = xData(i) - xData(i+1);

y = ((x - xData(i+1))^3/h - (x - xData(i+1))*h)*k(i)/6.0
  - ((x - xData(i))^3/h - (x - xData(i))*h)*k(i+1)/6.0
  + yData(i)*(x - xData(i+1))/h - yData(i+1)*(x - xData(i))/h;
```

Skrypt 3:

```
function i = findSeg(xData,x)
% Zwraca numer przedzialu w ktorym znajduje sie x

iLeft = 1;
iRight = length(xData);

while 1
  if (iRight - iLeft) <= 1
    i = iLeft;
    return
  end
  i = fix((iLeft + iRight)/2);
  if x < xData(i)
    iRight = i;
  else
    iLeft = i;
  end
end
```

Skrypt 4:

```
function [c,d,e] = LUdec3(c,d,e)
% Dekompozucja na macierz LU macierzy trojdiagonalnej A = [c\d\e]

n = length(d);
for k = 2:n
  lambda = c(k-1)/d(k-1);
  d(k) = d(k) - lambda * e(k-1);
  c(k-1) = lambda;
end
```

Skrypt 5:

```
function x = LUsol3(c,d,e,b)
% Rozwiazuje uklad A*x=b, gdzie A=[c/d/e] jest dekompozycja LU
% oryginalnej trojdiagonalnej macierzy A.

n = length(d);
for k = 2:n
  b(k) = b(k) - c(k-1)*b(k-1);
end

b(n) = b(n) / d(n);

for k = n-1:-1:1
  b(k) = (b(k) - e(k) * b(k+1)) / d(k);
end
x = b
```

Skrypt 6:

```
function x = tridiag(a,b,c,d)
% funkcja rozwiazuje uklad trojprzekatniowy postaci:
%
% |   b_1    c_1      0    ...     0 |   |   x_1 |   |   d_1 |
% |   a_2    b_2    c_2    ...     0 |   |   x_2 |   |   d_2 |
% |           ...    ...    ...       | * |   ... | = |   ... |
% |               a_n-1  b_n-1 c_n-1 |   | x_n-1 |   | d_n-1 |
% |                        b_n   c_n |   |   x_n |   |   d_n |
%
% Wszystkie wektory wejsciowe musza byc tej samej dlugosci.

n  = length(a); for i=2:n
    m = a(i)/b(i-1);
    b(i) = b(i) - m*c(i-1);
    d(i) = d(i) - m*d(i-1);
end

x(n) = d(n)/b(n);

for k=n-1:-1:1
    x(k) = (d(k) - c(k) * x(k+1)) / b(k);
end
```

Skrypt 7:

```
function k = splineCurv(xData, yData)
% Zwraca wartosc drugich pochodnych w wezlach
% dane wejsciowe: xData - wspolrzedne x wezlow,
%                 yData - wspolrzedne y wezlow.

n = length(xData); c = zeros(n, 1); d = ones(n,1);
e = zeros(n,1); k = zeros(n,1);

c(2:n-1) = xData(1:n-2) - xData(2:n-1);
d(2:n-1) = 2*(xData(1:n-2) - xData(3:n);
e(2:n-1) = xData(2:n-1) - xData(3:n);

k(2:n-1) = 6*(yData(1:n-2)-yData(2:n-1))./(xData(1:n-2)-xData(2:n-1))
         - 6*(yData(2:n-1)-yData(3:n))./(xData(2:n-1)-xData(3:n));

k = tridiag(c,d,e,k);
```

Zadanie:

Wyznacz przybliżenie funkcji

$$y = |x|$$

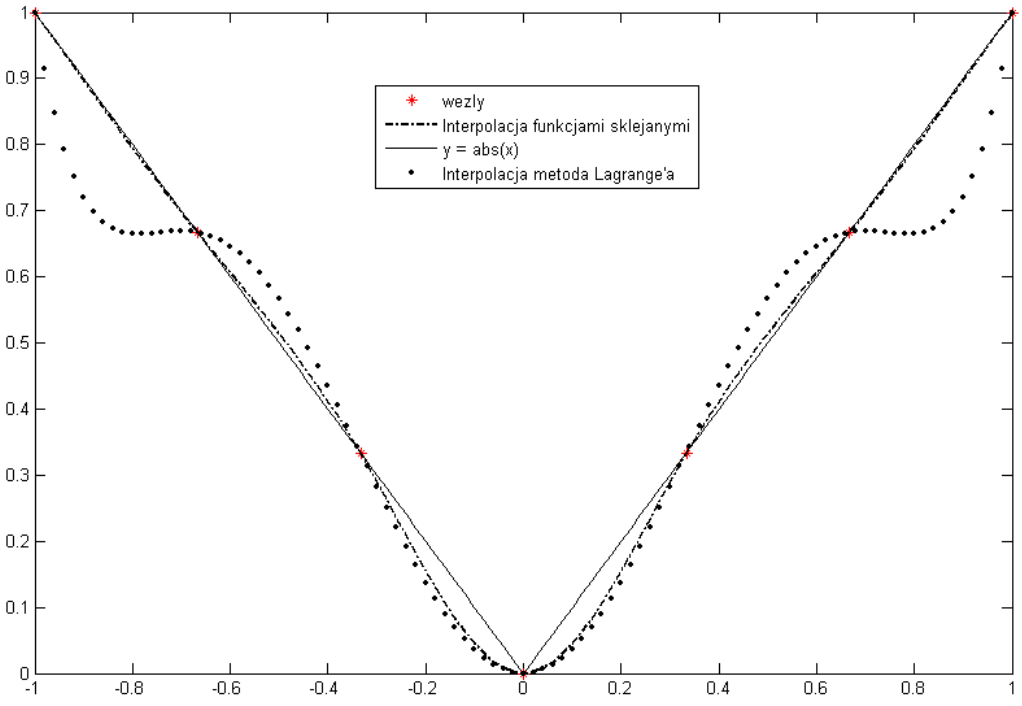
w przedziale <-1,1> przy użyciu metody interpolacji funkcjami sklejanymi w oparciu o 7 równoodległych węzłów.

Rozwizanie w programie MATLAB z użyciem dekompozycji LU:

```
clc
xData = linspace(-1,1,7);
yData = abs(x);
k = splineCurv(x,y);
xx = linspace(-1,1,101);
for i = 1:101
  yy(i) = splineEval(xData,yData,k,xx(i));
end
plot(xData,yData,'r*',xx,yy);
```

Rozwizanie w programie MATLAB z użyciem macierzy trójprzekątniowej:

```
clc
xData = linspace(-1,1,7);
yData = abs(x);
k = splineCurvTri(x,y);
xx = linspace(-1,1,101);
for i = 1:101
  yy(i) = splineEval(xData,yData,k,xx(i));
end
plot(xData,yData,'r*',xx,yy);
```


wezly
Interpolacja funkcjami sklejanymi
y = abs(x)
Interpolacja metoda Lagrange'a
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1