Marcin Rociek
Informatyka, II rok

Metody Obliczeniowe w Nauce i Technice

laboratorium

zestaw 3: funkcje sklejane

Funkcja $2/(3+x^2)$ dla x z $[0,2]$ określona jest n dyskretnymi punktami.

Zadanie 1: Wyznaczyć interpolacyjną funkcję sklejaną 3-go stopnia.

Funkcja sklejana określona jest wzorem:

$$s(x)=\sum_{i=-1}^{n+1} c_i \Phi^3(x)$$

gdzie:

$$\Phi^3(x)=\frac{1}{h^3}\begin{cases}(x-x_{i-2})^3, & x\in\langle x_{i-2};x_{i-1}\rangle\\ h^3+3h^2(x-x_{i-1})+3h(x-x_{i-1})^2-3(x-x_{i-1})^3, & x\in\langle x_{i-1};x_i\rangle\\ h^3+3h^2(x_{i+1}-x)+3h(x_{i+1}-x)^2-3(x_{i+1}-x)^3, & x\in\langle x_i;x_{i+1}\rangle\\ (x_{i+2}-x)^3, & x\in\langle x_{i+1};x_{i+2}\rangle\end{cases}$$

zaś współczynniki c wyznacza się z układu równań:

$$f'(0)=0$$

$$\begin{bmatrix}4 & 2 & \mathbf{L} & 0 & 0\\ 1 & 4 & & 0 & 0\\ \mathbb{M} & & \mathbf{O} & & \mathbb{M}\\ 0 & 0 & & 4 & 1\\ 0 & 0 & \mathbf{L} & 2 & 4\end{bmatrix}\cdot\begin{bmatrix}c_0\\ c_1\\ \mathbb{M}\\ c_{n-1}\\ c_n\end{bmatrix}=\begin{bmatrix}y_0+\frac{h}{3}a_1\\ y_1\\ \mathbb{M}\\ y_{n-1}\\ y_n+\frac{h}{3}b_1\end{bmatrix}$$

W którym wartości y to wartości funkcji dla kolejnych argumentów x:

$x_i=x_0+ih$, $i=0,1,...,n$

Wielkość h to odległość między kolejnymi argumentami:

$$h=\frac{b-a}{n}$$

Badana funkcja ma postać:

$$f(x)=\frac{2}{3+x^2}$$

Jej pierwsza pochodna:

$$f'(x)=\frac{-4x}{(3+x^2)^2}$$

Wartości pierwszej pochodnej na krańcach badanego przedziału wynoszą:

$$f'(0)=0$$

$$f'(2)=\frac{-8}{49}$$

Wartości te podstawiamy do układu z którego wyznaczamy współczynniki $c_0...c_n$

$$a_1 = 0$$

$$b_1 = \frac{-8}{49}$$

Współczynniki c_{-1} i c_{n+1} wyznacza sięz zależności:

$$c_{-1} = c_1 - \frac{h}{3} a_1$$

$$c_{n+1} = c_{n-1} + \frac{h}{3} b_1$$

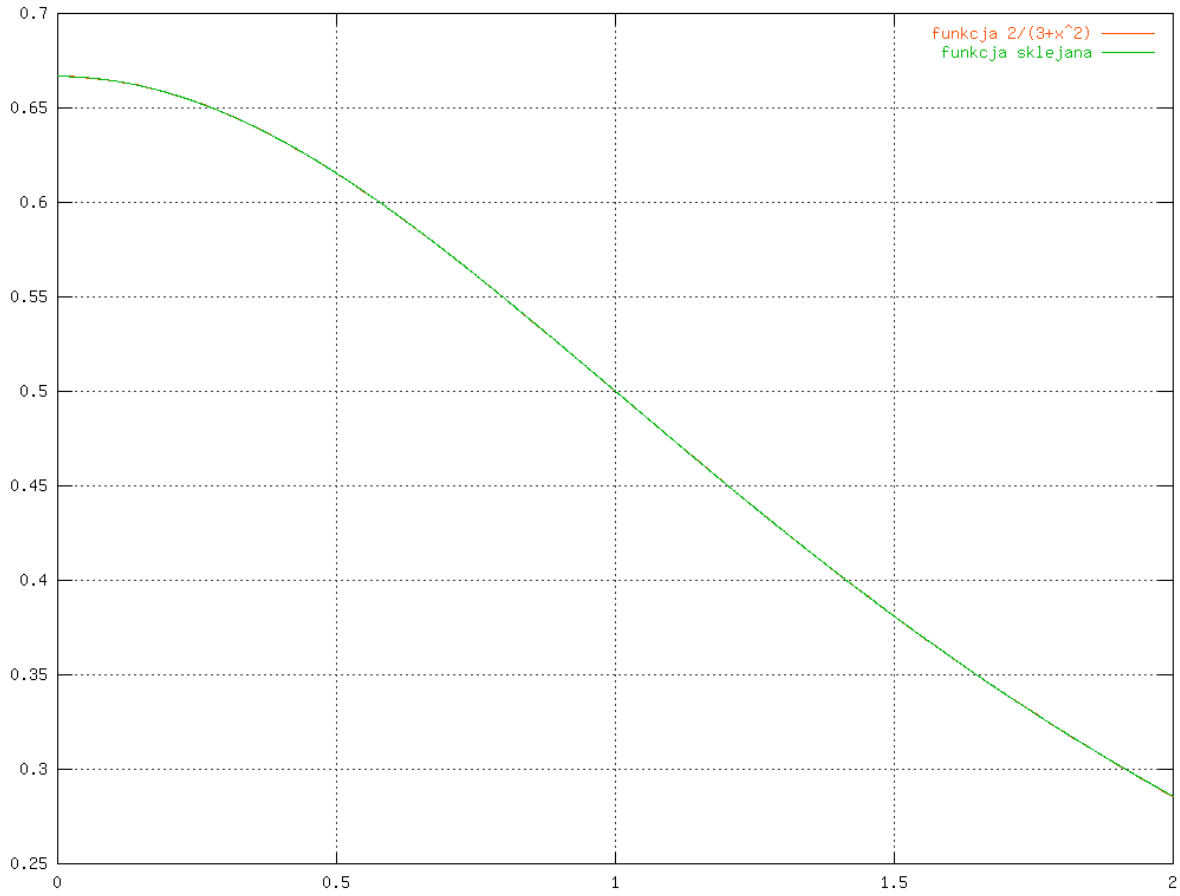
Do rozwiązania zadania wykorzystałem dwie funkcje z biblioteki *Numerical Recipes*: **spline** oraz **splint**. Pierwsza z nich tworzy funkcje sklejaną dla zadanego zbioru wartości funkcji, zaś druga oblicza wartość funkcji sklejanej dla konkretnej wartości *x*.

Treść programu:

```
#include "nr.h"
#include "nrutil.h"
#include <math.h>

void main(int argc, char**argv)
{
  int n = 6;

  float yp1, ypn, *x, *y, *y2;
  float xt, yt, dif;
  int i;

  if(argc==2)
    n = atoi(argv[1]);

  x = vector(1,n);  // argumenty funkcji
  y = vector(1,n);  // wartosci funkcji
  y2 = vector(1,n);

  for (i=1;i<=n;i++)
  {
    x[i] = 2.0*(i-1.0)/(n-1.0);
    y[i] = 2.0/(3.0+x[i]*x[i]);
  }

  spline(x, y, n, 0, -8.0/49.0, y2);

  dif = 0;
  for(xt=0; xt<=2; xt+=0.001)
  {
    float tempdif;
    splint(x, y, y2, n, xt, &yt);
    printf("%f %f\n",xt, yt);
    tempdif = fabs((2.0/(3.0+xt*xt))-yt);
    if(tempdif>dif)
      dif = tempdif;
  }

  printf("maksymalna roznica dla %d punktow: %f\n", n, dif);
}
```

Program wyznacza również maksymalną różnicę między funkcją interpolowaną a wyznaczoną funkcją sklejaną. Dla 6 punktów maksymalna różnica wynosi 0,000135, zaś dla 10 punktów: 0,000034.

Wykresy obu funkcji (praktycznie pokrywające się):



Zadanie 2: Wykorzystując aproksymację średniokwadratową przybliżyć tą funkcję za pomocą funkcji sklejanych 3-go stopnia z równoodległymi węzłami.

Aby stworzyć aproksymującą funkcje sklejaną dla dyskretnego zbioru punktów należy rozwiącać następujący układ równań:

$$\sum_{i=-1}^{n+1} b_{ij} c_i = \sum_{k=0}^{n1} f(x_k)\Phi_j^3(x_k),$$

$$j = -1,0,1,\ldots,n,n+1$$

gdzie b_{ij} określone są następująco:

$$b_{ij} = \sum_{k=0}^{n1} \Phi_i^3(x_k)\Phi_j^3(x_k).$$

Treść programu realizującego zadanie:

```
#include <math.h>
#include "nrutil.h"
#include <stdlib.h>
#include <stdio.h>

extern void gaussj(double **a, int n, double **b, int m);

/*          3
 * Funkcja fi (x)
 *          i
 */
double fi(int i, double x, double x0, double h)
{
  double f = 0.0, tmp, tmp2;

  if((x<=(x0+(i-1)*h)) && (x>=(x0+(i-2)*h)))
```

```
  {
    tmp = x-(x0+(i-2)*h);
    f = tmp*tmp*tmp;
  }
  else
  if((x>=(x0+(i-1)*h)) && (x<=(x0+i*h)))
  {
    double h2 = h*h;
    double h3 = h2*h;
    tmp = x-(x0+(i-1)*h);
    f = h3 +3.0*h2*tmp;
    tmp2 = tmp*tmp;
    f += 3.0*h*tmp2;
    tmp2 *= tmp;
    f -= 3.0*tmp2;
  }
  else
  if ((x>=(x0+i*h)) && (x<=(x0+(i+1)*h)))
  {
    double h2 = h*h;
    double h3 = h2*h;
    tmp = (x0+(i+1)*h)-x;
    f = h3 +3.0*h2*tmp;
    tmp2 = tmp*tmp;
    f += 3.0*h*tmp2;
    tmp2 *= tmp;
    f -= 3.0*tmp2;
  }
  else
  if((x>=(x0+(i+1)*h)) && (x<=(x0+(i+2)*h)))
  {
    tmp = (x0+(i+2)*h)-x;
    f = tmp*tmp*tmp;
  }
  return f;
}

/*
 * Wyznaczenie sklejanej funkcji aproksymujacej zadany zbior punktow
 */
double **splineapprox(int n, int npoints, double x[], double (*func)(double))
{
  int i, j, k;
  double tmp, tmp2, tmp3;
  double x0 = x[0];
  double h = (x[npoints-1]-x0)/n;
  double h2, h3;
  double **a, **b;

  b = dmatrix(1, n + 3, 1, 1);
  a = dmatrix(1, n + 3, 1, n + 3);

  /* prawa strona ukladu rownan */
  for (j=-1;j<=n+1;j++)
  {
    b[j+2][1] = 0;
    for(k=0; k<=npoints-1; k++)
      b[j+2][1] += func(x[k])*fi(j, x[k], x0, h);
  }

  /* macierz bedaca lewa strona ukladu rownan */
  for(j=-1; j<=n+1; j++)
    for (i=-1; i<=n+1; i++)
    {
      a[j+2][i+2] = 0;
      for(k=0; k<npoints; k++)
        a[j+2][i+2] += fi(i, x[k], x0, h) * fi(j, x[k], x0, h);
    }

  /* rozwiazanie ukladu rownan */
  gaussj(a, n + 3, b, 1);

  free_dmatrix(a, 1, n + 3, 1, n + 3);
  a = dmatrix(1, n + 7, 1, n + 7);

  for (i=1; i<=n+7; i++)
    for (j=1; j<=n+7; j++)
      a[i][j] = 0;

  h2 = h*h;
  h3 = h2*h;
```

```
  /* wyznaczenie wspolczynnikow funkcji sklejanej */
  for(i=-1; i<=n+1; i++)
  {
    tmp = x0+(i-2)*h;
    tmp2 = tmp*tmp;
    tmp3 = tmp2*tmp;
    a[i+2][1] += -tmp3*b[i+2][1];
    a[i+2][2] += 3.0*tmp2*b[i+2][1];
    a[i+2][3] += -3*tmp*b[i+2][1];
    a[i+2][4] += b[i+2][1];

    tmp = x0+(i-1)*h;
    tmp2 = tmp*tmp;
    tmp3 = tmp2*tmp;
    a[i+3][1] += ( h3 -3.0*h2*tmp +3.0*tmp3 +3.0*h*tmp2 )*b[i+2][1];
    a[i+3][2] += ( 3.0*h2 -6.0*h*tmp -9.0*tmp2 )*b[i+2][1];
    a[i+3][3] += ( 3.0*h +9.0*tmp )*b[i+2][1];
    a[i+3][4] += -3.0*b[i+2][1];

    tmp = x0+(i+1)*h;
    tmp2 = tmp*tmp;
    tmp3 = tmp2*tmp;
    a[i+4][1] += ( h3 +3.0*h2*tmp +3.0*h*tmp2 -3.0*tmp3 )*b[i+2][1];
    a[i+4][2] += ( -3.0*h2 -6.0*h*tmp +9.0*tmp2 )*b[i+2][1];
    a[i+4][3] += ( 3.0*h -9.0*tmp )*b[i+2][1];
    a[i+4][4] += 3.0*b[i+2][1];

    tmp = x0+(i+2)*h;
    tmp2 = tmp*tmp;
    tmp3 = tmp2*tmp;
    a[i+5][1] += tmp3*b[i+2][1];
    a[i+5][2] += ( -3.0*tmp2 )*b[i+2][1];
    a[i+5][3] += 3.0*tmp*b[i+2][1];
    a[i+5][4] += -b[i+2][1];
  }
  free_dmatrix(b, 1, n + 3, 1, 1);

  return a;
}

/*
 * funkcja zwracajaca wartosc funkcji sklejanej dla danego argumentu x
 */
double getsplineval(double x, double **polys, double x0, double h)
{
  int i, j = (int)((x-x0)/h);
  double y = 0.0, xt = 1.0;
  for(i=1; i<=4; i++)
  {
    y += xt*polys[j+4][i];
    xt *= x;
  }
  return y;
}

/*
 * funkcja aproksymowana
 */
double func(double x)
{
  return 2.0/(3.0+x*x);
}

/*
 * funkcja tworzaca funkcje sklejana dla okreslonej liczby przedzialow,
 * oraz wypisujaca jej parametry
 */
void spline(int n)
{
  double *x, h, **wyn, xt, diff, maxdiff;
  int i, j, dec;

  x = (double*)malloc((n+4)*sizeof(double));
  h = 2.0/(double)(n+3.0);
  x[0] = 0;
  x[n+3] = 2;
  for(j=1; j<n+3; j++)
    x[j] = x[0]+(j*h);
  h = 2.0/(double)n;
  wyn = splineapprox(n, n+4, x, &func);

  dec = n%10;
```

```
  printf("\n\nilosc przedzialow: %d\n", n);
  printf("     przedzial              wielomian\n");
  for(j=0; j<n; j++)
  {
    printf("<%f, %f>  ", x[0]+j*h, x[0]+(j+1)*h);
    for(i=1; i<4; i++)
      printf("%f*x^%d+", wyn[j+4][i], i-1);
    printf("%f*x^3\n", wyn[j+4][4]);
  }

  maxdiff = 0.0;
  for(xt = 0.0; xt<=2.0; xt+=0.01)
  {
    diff = fabs(func(xt)-getsplineval(xt, wyn, x[0], h));
    if(diff>maxdiff)
      maxdiff = diff;
  }
  printf("Maksymalna roznica miedzy funkcja aproksymowana a sklejana wynosi: %f\n", maxdiff);

  free_dmatrix(wyn, 1, n+7, 1, n+7);
  free(x);
}

/*
 * glowna czesc programu
 */
void main(void)
{
  spline(6);
  spline(10);
}
```

Wynik działania programu (dla przykładowo 6 i 10 przedziałów):

```
ilosc przedzialow: 6
     przedzial              wielomian
<0.000000, 0.333333>  0.666667*x^0+0.001215*x^1+-0.236660*x^2+0.056553*x^3
<0.333333, 0.666667>  0.665073*x^0+0.015555*x^1+-0.279679*x^2+0.099572*x^3
<0.666667, 1.000000>  0.669454*x^0+-0.004159*x^1+-0.250108*x^2+0.084786*x^3
<1.000000, 1.333333>  0.713762*x^0+-0.137083*x^1+-0.117184*x^2+0.040478*x^3
<1.333333, 1.666667>  0.780117*x^0+-0.286382*x^1+-0.005210*x^2+0.012484*x^3
<1.666667, 2.000000>  0.864611*x^0+-0.438471*x^1+0.086044*x^2+-0.005766*x^3
Maksymalna roznica miedzy funkcja aproksymowana a sklejana wynosi: 0.000032


ilosc przedzialow: 10
     przedzial              wielomian
<0.000000, 0.200000>  0.666667*x^0+-0.000037*x^1+-0.224505*x^2+0.027724*x^3
<0.200000, 0.400000>  0.666259*x^0+0.006075*x^1+-0.255068*x^2+0.078662*x^3
<0.400000, 0.600000>  0.664926*x^0+0.016070*x^1+-0.280055*x^2+0.099486*x^3
<0.600000, 0.800000>  0.665844*x^0+0.011483*x^1+-0.272409*x^2+0.095238*x^3
<0.800000, 1.000000>  0.675993*x^0+-0.026577*x^1+-0.224834*x^2+0.075415*x^3
<1.000000, 1.200000>  0.701765*x^0+-0.103893*x^1+-0.147519*x^2+0.049643*x^3
<1.200000, 1.400000>  0.739158*x^0+-0.197374*x^1+-0.069618*x^2+0.028004*x^3
<1.400000, 1.600000>  0.783803*x^0+-0.293044*x^1+-0.001282*x^2+0.011733*x^3
<1.600000, 1.800000>  0.828229*x^0+-0.376342*x^1+0.050779*x^2+0.000887*x^3
<1.800000, 2.000000>  0.861501*x^0+-0.431796*x^1+0.081587*x^2+-0.004818*x^3
Maksymalna roznica miedzy funkcja aproksymowana a sklejana wynosi: 0.000006
```

Jak widać, odchyłka funkcji sklejanej od funkcji oryginalnej jest w przypadku aproksymacji dużo mniejsza. Ilustruje to poniższe zestawienie:

Ilość Przedziałów	Maksymalna różnica:	
	Interpolacja	Aproksymacja
6	0,000135	0,000032
10	0,000034	0,000006