Marcin Rociek
Informatyka, II rok

Metody Obliczeniowe w Nauce i Technice

laboratorium

zestaw 2: aproksymacja

Zadanie 1: Wyniki pomiarów zebrane są w tabeli:

x	1	3	4	6	8	9	11	14
y	1	2	4	4	5	7	8	9

Znaleźć wielomian drugiego stopnia przybliżający najlepiej (w sensie aproksymacji średniokwadratowej) dany układ wartości.

Do rozwiązania tego zadania napisałem prostą funkcję `polapprox` wyznaczającą wielomian aproksymujący stopnia *m* dla *n* punktów. Najpierw wyliczane są wyrażenia:

$$s_1 = \sum_i x_i \,, \qquad s_2 = \sum_i x_i^2 \,, \qquad s_3 = \sum_i x_i^3 \,, \ldots$$

$$p_1 = \sum_i y_i \,, \qquad p_2 = \sum_i y_i x_i \,, \quad p_3 = \sum_i y_i x_i^2 \,, \quad p_4 = \sum_i y_i x_i^3 \,, \ldots$$

Następnie rozwiązywany jest układ *m+1* równań liniowych o *m+1* niewiadomych, postaci:

$$\begin{bmatrix} n & s_1 & s_2 & \mathbf{L} & s_m \\ s_1 & s_2 & s_3 & & s_{m+1} \\ s_2 & s_3 & s_4 & \mathbf{L} & s_{m+2} \\ \mathbb{M} & & \mathbb{M} & \mathbf{O} & \mathbb{M} \\ s_m & s_{m+1} & s_{m+2} & \mathbf{L} & s_{2m} \end{bmatrix} \bullet \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \mathbb{M} \\ a_m \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \mathbb{M} \\ p_{m+1} \end{bmatrix}$$

którego rozwiązaniem są współczynniki wielomianu aproksymującego. Do rozwiązania tego układu równań wykorzystałem funkcje `ludcmp` oraz `lubksb` z biblioteki *Numerical Recipes*.

Treść funkcji `polapprox`:

```
/*
 * aproksymacja wielomianowa.
 * wejscie: x, y - punkty
 *          n - ilosc punktow
 *          m - stopien wielomianu aproksymujacego
 * wyjscie: coef - wspolczynniki wielomianu
 */
void polapprox(float *x, float *y, int n, int m, float *coef)
{
  float **a, *b, d;
  float *s;
  float *p;
  int *idx;
  int i,j;

  int sdim = 2*m;

  s = vector(1, sdim+1);
  p = vector(1, m+1);

  for(i=1; i<=sdim; i++)
    s[i] = 0;
  for(j=0; j<n; j++)
  {
    float ss = x[j];
    for(i=1; i<=sdim; i++)
    {                           // s1 = x1   + ... + xn
      s[i] += ss;               // s2 = x1^2 + ... + xn^2
      ss *= x[j];               // s3 = x1^3 + ... + xn^3
    }                           //  ...
  }

  for(i=1; i<=m+1; i++)
    p[i] = 0;
  for(j=0; j<n; j++)
```

```
  {
    float pp = 1;
    for(i=1; i<=m+1; i++)       // p1 = y1      + ... + yn
    {                           // p2 = y1*x1   + ... + yn*xn
      p[i] += pp*y[j];          // p3 = y1*x1^2 + ... + yn*xn^2
      pp *= x[j];               //  ...
    }
  }

  idx = ivector(1, m+1);

  a = matrix(1, m+1, 1, m+1);
  b = vector(1, m+1);

  a[1][1] = n;                  //    | n  s1 s2 ..  |
  for(j=1; j<=m+1; j++)         //    | s1 s2 s3     |
    for(i=1; i<=m+1; i++)       // A= | s2 s3 s4     |
      if(j>1 || i>1)            //    | s3 s4 s5     |
        a[j][i] = s[j+i-2];     //    | ..           |

                                //    | p1 |
  for(i=1; i<=m+1; i++)         //    | p2 |
    b[i] = p[i];                // B= | p3 |
                                //    | p4 |
                                //    | .. |

  ludcmp(a, m+1, idx, &d);
  lubksb(a, m+1, idx, b);       // rozwiazanie ukladu Ax=B

  for(i=0; i<m+1; i++)
    coef[i] = b[i+1];
}
```

Treść programu realizującego aproksymację danych podanych w zadaniu:

```
#include <stdlib.h>
#include "nr.h"

void main(void)
{
  float x[] = {  1,  3,  4,  6,  8,  9, 11, 14};
  float y[] = {  1,  2,  4,  4,  5,  7,  8,  9};
  float coef[3];
  int i;

  polapprox(x, y, 8, 2, coef);

  printf("wielomian aproksymujacy:\n");
  for(i=0; i<3; i++)
    printf("%f x^%d\n", coef[i], i);
}
```

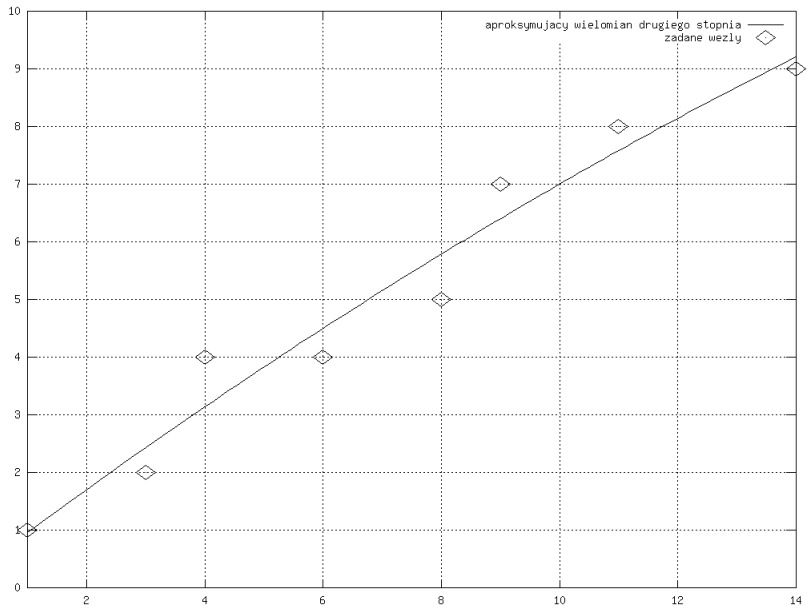
Wynik działania tego programu:

```
wielomian aproksymujacy:
0.194808 x^0
0.772286 x^1
-0.009173 x^2
```

Wielomianem aproksymującym zestaw punktów jest wielomian

$y = -0.009173x^2 + 0.772286x + 0.194808$.

Poniżej znajduje się jego wykres:



Zadanie 2: Wyznaczyć wielomian stopnia *m* aproksymujący funkcję $f(x)=1/(1+x^2)$ dla $-1<x<1$ określoną $n+1$ dyskretnymi punktami. Porównać przybliżenie uzyskane dla aproksymacji średniokwadratowej z przybliżeniem uzyskanym metodą aproksymacji jednostajnej.

Do rozwiązania tego zadania posłużyłem się tą samą funkcją co poprzednio (`polapprox`) oraz funkcjami `chebft` i `chebpc` z biblioteki *Numerical Recipes*.

Treść programu:

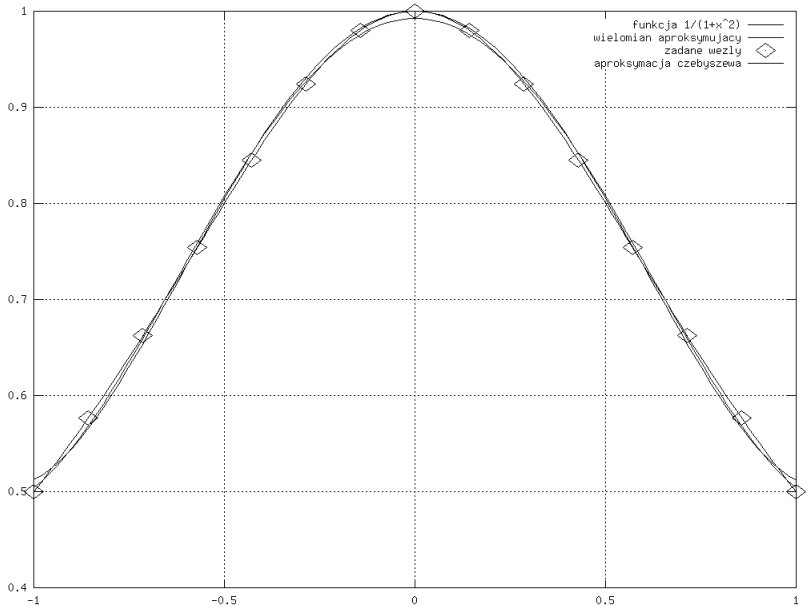
```
#include <stdlib.h>
#include "nr.h"

float funkcja(float x)
{
  return 1.0/(1.0+x*x);
}

void main(void)
{
  float *x, *y;
  float *coef, *chcoef, *pchcoef;
  int i;
  int n = 15;
  int m = 5;

  x = (float*)malloc(n*sizeof(float));
  y = (float*)malloc(n*sizeof(float));
  coef = (float*)malloc((m+1)*sizeof(float));
  chcoef = (float*)malloc((m+1)*sizeof(float));
  pchcoef = (float*)calloc((m+1)*sizeof(float), 1);

  printf("%d wezlow:\n", n);
  for(i=0; i<n; i++)
  {
    x[i] = ((2.0*i)/((float)(n-1)))-1;
    y[i] = funkcja(x[i]);
    printf("%f %f\n", x[i], y[i]);
  }

  polapprox(x, y, n, m, coef);

  printf("\nwielomian aproksymujacy %d stopnia:\n", m);
  for(i=0; i<m+1; i++)
    printf("%f x^%d\n", coef[i], i);

  chebft(-1, 1, chcoef, m, funkcja);
  chebpc(chcoef, pchcoef, m);

  printf("\nwielomian czebyszewa:\n");
  for(i=0; i<m+1; i++)
    printf("%f x^%d\n", pchcoef[i], i);
```

```
}
```

Wynik działania programu dla (przykładowo) 15 węzłów i wielomianów 5 stopnia:

```
15 wezlow:
-1.000000 0.500000
-0.857143 0.576471
-0.714286 0.662162
-0.571429 0.753846
-0.428571 0.844828
-0.285714 0.924528
-0.142857 0.980000
0.000000 1.000000
0.142857 0.980000
0.285714 0.924528
0.428571 0.844828
0.571429 0.753846
0.714286 0.662162
0.857143 0.576471
1.000000 0.500000

wielomian aproksymujacy 5 stopnia:
0.992224 x^0
0.000004 x^1
-0.824281 x^2
-0.000013 x^3
0.336452 x^4
0.000010 x^5

wielomian czebyszewa:
1.000000 x^0
-0.000000 x^1
-0.878049 x^2
0.000000 x^3
0.390244 x^4
0.000000 x^5
```

Wykresy funkcji:



Zadanie 3: Utworzyć tablicę wartości funkcji (*n* punktów) *f(x)=2-2sin(x)+cos(x)*. Lekko zaburzyć wartości w tablicy i dla tych wartości znaleźć trygonometryczne przybliżenie średniokwadratowe.

Do rozwiązania tego zadania napisałem program, który wyznacza współczynniki aproksymującego wielomianu trygonometrycznego dla *2L* punktów.
Współczynniki wyznaczane są wg zależności:

$$a_j = \frac{1}{L}\sum_{i=0}^{2L-1} f(x_i)\cos jx_i \,, \quad b_j = \frac{1}{L}\sum_{i=0}^{2L-1} f(x_i)\sin jx_i \quad \text{dla} \quad j=1, 2, \ldots, n$$

Wielomian zaś ma postać:

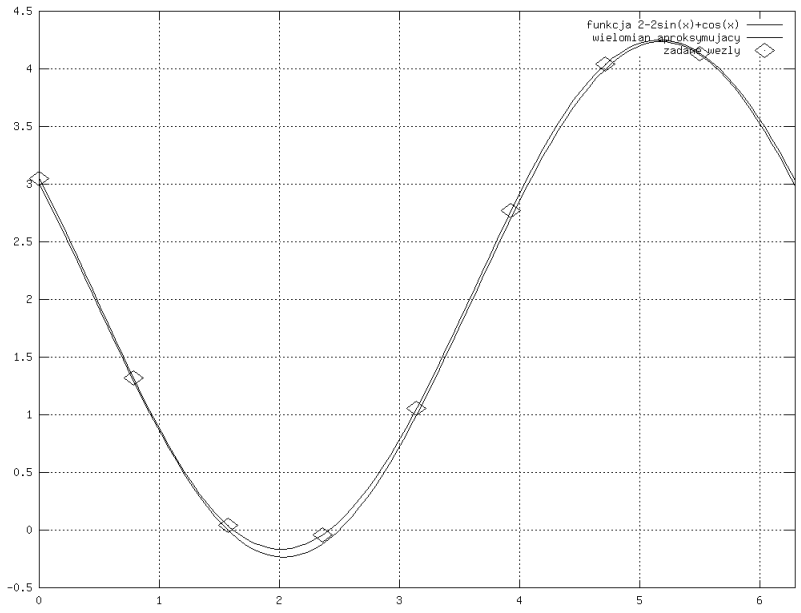
$$y_n(x) = \frac{1}{2}a_0 + \sum_{j=1}^{n}\left(a_j \cos jx + b_j \sin jx\right), \quad n<L$$

Treść programu:

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

void main(void)
{
  const int n = 8;
  float *x, *y, *ab;
  int i, j, l;

  x = (float*)malloc(n*sizeof(float));
  y = (float*)malloc(n*sizeof(float));
  ab = (float*)calloc(n*sizeof(float),1);

  l = (n/2);

  for(i=0; i<n; i++)
  {
    x[i] = PI*i/l;
    y[i] = 2.0-2.0*sin(x[i])+cos(x[i]) +0.001*(random()%100);
  }

  // obliczenie a0
  for(i=0; i<n; i++)
    ab[0] += y[i];
  ab[0] /= l;

  // obliczenie aj i bj
  for(j=1; j<l; j++)
  {
    for(i=0; i<n; i++)
    {
      ab[j*2] += y[i]*cos(x[i]*(float)j);
      ab[j*2+1] += y[i]*sin(x[i]*(float)j);
    }
    ab[j*2] /= (float)l;
    ab[j*2+1] /= (float)l;
  }

  printf("wezly:\n");
  for(i=0; i<n; i++)
    printf("%f %f\n", x[i], y[i]);

  printf("wielomian aproksymujacy:\n");
  printf("%f\n", 0.5*ab[0]);
  for(j=1; j<l; j++)
    printf("%f*cos(%d*x)\n%f*sin(%d*x)\n", ab[j*2], j, ab[j*2+1], j);
}
```

Wynik działania programu (dla n=8):

```
wezly:
0.000000 3.049000
0.785398 1.316893
1.570796 0.041000
2.356194 -0.043320
3.141593 1.057000
3.926991 2.772107
4.712389 4.042000
5.497787 4.134320
wielomian aproksymujacy:
2.046125
0.979261*cos(1*x)
-1.996007*sin(1*x)
0.005750*cos(2*x)
-0.000500*sin(2*x)
0.016738*cos(3*x)
0.004493*sin(3*x)
```


funkcja 2-2sin(x)+cos(x)
wielomian aproksymujacy
zadane wezly
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
0
1
2
3
4
5
6