Dokumentacja do zadania 1.3

Marcin Horoszko, Radosław Głombiowski, Jacek Dermont

4 listopada 2012

1 Zadanie 1.3

Ustalić naturalną k_{max}. Wczytać k ε {1,2,...,k_{max}} oraz różne węzły $x_1, x_2, ..., x_n$ gdzie n = 3k. Następnie wczytać 2 komplety wartości $A_1, A_2, ..., A_n$ i $B_1, B_2, ..., B_k$. Wyznaczyć w postaci Newtona wielomian interpolacyjny Hermite'a $W = W(x)$ stopnia co najwyżej (4k - 1) spełniający warunki: $W(x_i) = A_i$ dla i = 1,2,...,n oraz $W'(x_{3i}) = B_i$ dla i = 1,2,...,k. Wynik przedstawić również w postaci ogólnej.

2 Podstawowe pojęcia

2.1 (Wielomian interpolacyjny Hermite'a funkcji f)**.** Wielomian W stopnia co najwyżej n, nazywamy wielomianem interpolacyjnym Hermite'a funkcji f, jeśli w każdym n–krotnym węźle x_k spełnia równania:

$$W(x_k) = f(x_k),\ W'(x_k) = f'(x_k),\ \dots\ ,\ W^{(n-1)}(x_k) = f^{(n-1)}(x_k).$$

2.2 (różnica dzielona)**.** Dla parami różnych liczb $x_0, \dots, x_n$ **różnice dzielone** funkcji f są określone w następujący sposób:

$$f[x_i] = f(x_i),$$

$$f[x_i, \dots, x_i + k] = \frac{f[x_i, \dots, x_{i+k-1}] - f[x_{i+1}, \dots, x_{i+k}]}{x_i - x_{i+k}}$$

Korzystając z powyższych wzorów, możemy wyliczyć różnicę dzieloną dla dowolnych $x_0, \dots, x_n$.

3 Metoda numeryczna

Posiadając węzły $x_1, x_2, ..., x_n$, komplety wartości $A_1, A_2, ..., A_n$ i $B_1, B_2, ..., B_k$, możemy wyznaczyć wielomian interpolacyjny Hermite'a w postaci Newtona. Zrobimy to w następujący sposób:

- tworzymy tablicę różnic dzielonych o wielkości n+k (co trzeci węzeł będzie się powtarzał)

- dla powtarzających się węzłów, różnica dzielona wyjdzie $\frac{0}{0}$; wtedy zastępujemy ją przez $W'(x_{3i})$=B_i
- pierwszy wiersz będzie zawierać współczynniki dla wielomianu interpolacyjnego; oznaczmy przez $a_0, ..., a_{n-1}$
- wielomian (w postaci Newtona) otrzymamy zgodnie ze wzorem:
  $W(x) = a_0 + a_1(x - x_1) + ... + a_{n-1}(x - x_n)$
- wymnażamy wielomian w postaci Newtona i otrzymujemy wielomian w postaci ogólnej

4 Opis programu

Program został w całości napisany w języku $C++$.

Wejście

- liczba całkowita k, w przedziale $1, ..., k_{max}$ (k_{max} ustalone, wynosi 5)
- n (n=3k) liczb typu double, **x**, niepowtarzających się
- n (n=3k) liczb typu double, **A**
- k liczb typu double, **B**

Wyjście

- wydruk wielomianu interpolacyjnego Hermite'a w postaci Newtona
- wydruk wielomianu interpolacyjnego Hermite'a w postaci ogólnej

Struktura programu

- *main.cpp* - główna część programu, pobiera dane od użytkownika, wywołuje funkcje liczące współczynniki do wielomianów i generuje wyjście w postaci tychże wielomianów
- *tablica.cpp* - zawiera funkcje wyliczające współczynniki wielomianu interpolacyjnego Hermite'a
- *drukuj.cpp* - zawiera funkcje drukujące wielomiany

5 Funkcje programu

tablica.cpp
Główna część roznice_dzielone:

```
double **roznice_dzielone(double *x, double *A,
     double *B, int rozmiar)
    ...
    // wypelnia pierwsza kolumne
    for (int i=0;i<rozmiar;i++)
       tablica[i][0] = A[i];

    // wypelnia druga kolumne
    int t = 0;
    for (int i=0;i<rozmiar-1;i++)
        double licznik = tablica[i+1][0] - tablica[i][0];
        double mianownik = x[i+1] - x[i];
        if (licznik == 0.0f && mianownik == 0.0f)
            tablica[i][1] = B[(i-t)/3];
            t++;
            continue;
        tablica[i][1] = licznik/mianownik;

    // wypelnia reszte kolumn
    for (int i=2;i<rozmiar;i++)
        int z = i;
        for (int j=0;j<rozmiar-i;j++)
            double licznik = tablica[j+1][i-1] - tablica[j][i-1];
            double mianownik = x[z]-x[j];
            z++;
            tablica[j][i] = licznik/mianownik;
    ...
```

Funkcja ta zawiera trzy pętle for. Pierwsza służy do wypełnienia pierwszej kolumnej kompletem wartości A. Drugą pętla wypełnia drugą kolumnę już za pomocą wzoru na różnice dzielone, przy czym wynik $\frac{0}{0}$ zastępuje poprzez odpowiedni element B. Trzecia pętla wypełnia resztę kolumn zgodnie ze wzorem na różnice dzielone.
tablica.cpp zawiera również funkcję *wspolczynniki*, która kopiuje z tablicy pierwszy wiersz i go zwraca.

drukuj.cpp
Funkcja drukuj_newton:

```
void drukuj_newton(double *wsp,double *x, int rozmiar) {
    cout << "W(x) = ";
    for (int i=0;i<rozmiar;i++)
        if (wsp[i] == 0.0f) continue;
        if (i == 0) cout << wsp[i];
        else
            if (wsp[i] > 1.0f || wsp[i] < 1.0f)
                if (i>0 && wsp[i] < 0.0f) cout << -1*wsp[i];
                else cout << wsp[i];
        for (int j=0;j<i;j++)
```

```
            if ((j+1)%4 == 0) cout << "^2";
            else
                if (x[j] < 0.0f) cout << "(x+" << -1*x[j] << ")";
                else if (x[j] == 0.0f) cout << "x";
                else cout << "(x-" << x[j] << ")";
        if (i+1 < rozmiar)
            if (!same_zera(wsp,i+1,rozmiar))
                if (wsp[i+1] > 0.0f) cout << " + ";
                else cout << " - ";
    cout << endl;
}
```

Funkcja przyjmuje współczynniki, tablicę x i rozmiar jako argumenty i drukuje wielomian iterpolacyjny Hermite'a w postaci Newtona. Odwołuje się do funkcji pomocniczej *same_zera*, która to sprawdza czy dalsze współczynniki są niezerowe (dzięki temu nie ma niepotrzebnego plusa na końcu).

Najważniejsza część drukuj_ogolna:

```
void drukuj_ogolna(double **tablica,double *x,int rozmiar) {
  double tablicaOgolna[rozmiar][rozmiar];
  // tworzymy tablice startowa ( 1 kolumna to wspolczynniki, na
     koncu to beda wartosci przy danym stopniu )
  for( int i = 0 ; i < rozmiar ; i++ )
    for( int j = 0 ; j < rozmiar ; j++ )
      if( j == 0 ){ tablicaOgolna[i][j] = tablica[0][i]; }
      else{ tablicaOgolna[i][j] = 0; }

  // tworzymy tablice x'ow ( do mnozenia ) ( miejsc zerowych )
  double xValues[rozmiar];
  for( int i = 0 ; i < rozmiar ; i++ )
    if( i == 0 ){ xValues[i] = 0; }
    if( i % 4 == 0 ){ xValues[i] = i == 0 ? 0 : x[i-2]; }
    else{ xValues[i] = x[i-1]; }

  // liczymy kazdy wiersz ( dzialanie mnozenia wszystkich stopni )
  for( int i = 0 ; i < rozmiar ; i++ )
    // dla mnozenia ( 1 ) bedzie 0 iteracji, dla mnozenia ( 1 * (x
       -1) ) 1 iteracja, itp itd.
    for( int j = 0 ; j < i ; j++ )
      // zapisanie aktualnej tablicy przed przesuwaniem by potem
         od tego mnozyc
      double currentOriginalTab[rozmiar];
      for( int oT = 0 ; oT < rozmiar; oT++ ){ currentOriginalTab[
         oT] = tablicaOgolna[i][oT]; }
      // przesuwamy wiersz glownej tablicy w prawo
      for( int pT = rozmiar - 1 ; pT > 0 ; pT-- )
 tablicaOgolna[i][pT] = tablicaOgolna[i][pT-1];
      tablicaOgolna[i][0] = 0;
      // mnozymy oryginalna tablice przez xValues i dodajemy do
         glownej
      for( int mT = 0 ; mT < rozmiar ; mT++ )
 tablicaOgolna[i][mT] = tablicaOgolna[i][mT] + (
    currentOriginalTab[mT] * ( -1 * xValues[j+1] ) );
```

```
  // zsumowanie kazdej kolumny
  for( int i = rozmiar - 1 ; i > 0 ; i-- )
    for( int j = 0 ; j < rozmiar ; j++ )
      tablicaOgolna[0][j] += tablicaOgolna[i][j];

  // wypisanie postaci ogolnej ( pierwszy wiersz jest wynikiem )
  cout << "W(x) = ";
  int firstExp = 1;
  for( int i = rozmiar-1 ; i >= 0 ; i-- )
    if( tablicaOgolna[0][i] != 0 )
      if( !(i > 0 && tablicaOgolna[0][i] == 1) )
if (!same_zera(tablicaOgolna[0],i+1,rozmiar))
   if( tablicaOgolna[0][i] < 0 )
     cout << " - ";
     if (tablicaOgolna[0][i] != -1) cout << tablicaOgolna[0][i] *
        -1;
   else
     cout << " + ";
     if (tablicaOgolna[0][i] != 1) cout << tablicaOgolna[0][i];
else
   if (tablicaOgolna[0][i] != -1 && tablicaOgolna[0][i] != 1) cout
        << tablicaOgolna[0][i];
   if( i == 1 ){ if (tablicaOgolna[0][i]==-1) cout << "-x"; else
       cout << "x"; }
   else if( i > 1 ){ if (tablicaOgolna[0][i]==-1) cout << "-x^";
       else cout << "x^"; cout << i; }
      else if (firstExp == 1)
firstExp = 0;
      if (same_zera(tablicaOgolna[0],0,rozmiar)) cout << "0";
  cout << endl;
```

Podobnie jak drukuj_newton, drukuj_ogolna przyjmuje tablicę różnic dzielonych, tablicę x i rozmiar jako argumenty. Za pomocą odpowiednich pętli funkcja ta wymnaża współczynniki i x'y, sumuje je i drukuje wielomian interpolacyjny Hermite'a w postaci ogólnej. Ta funkcja także odwołuje się do *same_zera*.

main.cpp
Podstawa programu, inicjalizuje zmienne, pobiera dane od użytkownika, odwołuje się do innych funkcji i odwołuje się funkcji wypisujących wyjście na ekran. Na końcu czyści niepotrzebną pamięć.
Część odpowiedzialna za pobieranie danych od użytkownika.

```
    #define ROZ 4*k
    ...
    int k = 1;
    cout << "Podaj k z zakresu 1.." << KMAX << endl;
    do
        if (k > KMAX || k < 1) cout << "k niepoprawne!" << endl;
        cout << "Podaj k: "; cin >> k;
    while (k > KMAX || k < 1);

    int n = 3*k;
    double x[ROZ]; // x i A maja rozmiar 4*k (a nie n) poniewaz
```

```
    double A[ROZ]; // co trzeci element sie powtarza!
    double B[k];
    int t = 0;
    double tmp_x;
    bool tmp;
    cout << "Podaj rozne wezly x." << endl;
    for (int i=0;i<n;i++)
        tmp = false;
        cout << "Podaj x" << i+1 << ": "; cin >> tmp_x;
        for (int j=0; j<(i+t); j++) {
            if (tmp_x == x[j])
                cout << "Podaj inny x." << endl;
                tmp = true;
                break;

        if (tmp == true)
            i--;
            continue;

        x[i+t] = tmp_x;
        if ((i+1)%3 == 0)
            t++;
            x[i+t] = x[i+t-1];
    t = 0;
    cout << "Podaj komplet wartosci A." << endl;
    for (int i=0;i<n;i++)
        cout << "Podaj A" << i+1 << ": "; cin >> A[i+t];
        if ((i+1)%3 == 0)
            t++;
            A[i+t] = A[i+t-1];
    cout << "Podaj komplet wartosci B." << endl;
    for (int i=0;i<k;i++)
        cout << "Podaj B" << i+1 << ": "; cin >> B[i];
    ...
```

Użytkownik kolejno podaje k z zakresu 1..KMAX (5), n (n=3k) **różnych** węzłów x, n wartości A i k wartości B. Za to odpowiedzialne są kolejne pętle for. Mimo, że tablica węzłów x i tablica wartości A są wielkości 4k, użytkownik podaje tylko n (n=3k) danych. W tych pętlach co trzeci element automatycznie się powtarza.

Część odwołująca się do wcześniej omówionych funkcji drukujących.

```
    #define ROZ 4*k
    ...
    // postac newtona
    cout << endl << "Wielomian interpolacyjny w postaci Newtona:"
       << endl;
    drukuj_newton(wsp,x,ROZ);

    // postac ogolna
    cout << endl << "Wielomian interpolacyjny w postaci ogolnej:"
       << endl;
    drukuj_ogolna(tablica,x,ROZ);
    ...
```

6 Przykładowe uruchomienia

Przykład 1

```
Podaj k z zakresu 1..5
Podaj k: 1
Podaj rozne wezly x.
Podaj x1: 1
Podaj x2: 2
Podaj x3: 3
Podaj komplet wartosci A.
Podaj A1: 2
Podaj A2: 2
Podaj A3: 2
Podaj komplet wartosci B.
Podaj B1: 0

Wielomian interpolacyjny w postaci Newtona:
W(x) = 2

Wielomian interpolacyjny w postaci ogolnej:
W(x) = 2
```

Przykład 2

```
Podaj k z zakresu 1..5
Podaj k: 1
Podaj rozne wezly x.
Podaj x1: 1
Podaj x2: 2
Podaj x3: 3
Podaj komplet wartosci A.
Podaj A1: 9
Podaj A2: 5
Podaj A3: 1
Podaj komplet wartosci B.
Podaj B1: -4

Wielomian interpolacyjny w postaci Newtona:
W(x) = 9 - 4(x-1)

Wielomian interpolacyjny w postaci ogolnej:
W(x) = -4x + 13
```

Przykład 3

```
Podaj k z zakresu 1..5
Podaj k: 0
k niepoprawne!
Podaj k: 2
Podaj rozne wezly x.
Podaj x1: 1
Podaj x2: 2
Podaj x3: 3
Podaj x4: 1
```

```
Podaj inny x.
Podaj x4: 3
Podaj inny x.
Podaj x4: 4
Podaj x5: 5
Podaj x6: 5
Podaj inny x.
Podaj x6: 6
Podaj komplet wartosci A.
Podaj A1: 2
Podaj A2: 4
Podaj A3: 8
Podaj A4: 13
Podaj A5: 19
Podaj A6: 27
Podaj komplet wartosci B.
Podaj B1: 3
Podaj B2: 6

Wielomian interpolacyjny w postaci Newtona:
W(x) = 2 + 2(x-1) + (x-1)(x-2) - 1(x-1)(x-2)(x-3) + 0.833(x-1)(x-2)(x-3)^2 -
    0.396(x-1)(x-2)(x-3)^2(x-4) + 0.132(x-1)(x-2)(x-3)^2(x-4)(x-5) - 0.0539(x
    -1)(x-2)(x-3)^2(x-4)(x-5)(x-6)

Wielomian interpolacyjny w postaci ogolnej:
W(x) = -0.0539x^7 + 1.43x^6 - 15.6x^5 + 91.1x^4 - 304x^3 + 576x^2 - 562x + 216
```

Przykład 4

```
Podaj k z zakresu 1..5
Podaj k: 6
k niepoprawne!
Podaj k: 1
Podaj rozne wezly x.
Podaj x1: 1
Podaj x2: 2
Podaj x3: 3
Podaj komplet wartosci A.
Podaj A1: 0
Podaj A2: 0
Podaj A3: 0
Podaj komplet wartosci B.
Podaj B1: 0

Wielomian interpolacyjny w postaci Newtona:
W(x) = 0

Wielomian interpolacyjny w postaci ogolnej:
W(x) = 0
```

Przykład 5

```
Podaj k z zakresu 1..5
Podaj k: 3
Podaj rozne wezly x.
Podaj x1: -1
Podaj x2: 0
Podaj x3: 1
Podaj x4: -1
Podaj inny x.
Podaj x4: 2
Podaj x5: 3
Podaj x6: 4
Podaj x7: 5
Podaj x8: 6
Podaj x9: 7
Podaj komplet wartosci A.
Podaj A1: -1
Podaj A2: 2
Podaj A3: 3
Podaj A4: 7
Podaj A5: 10
Podaj A6: 7
Podaj A7: 4
Podaj A8: 2
Podaj A9: 0
Podaj komplet wartosci B.
Podaj B1: 0
Podaj B2: 5
Podaj B3: 8

Wielomian interpolacyjny w postaci Newtona:
W(x) = -1 + 3(x+1) - 1(x+1)x - 0.833(x+1)x(x-1)^2 - 0.604(x+1)x(x-1)^2(x-2) +
    0.224(x+1)x(x-1)^2(x-2)(x-3) - 0.0431(x+1)x(x-1)^2(x-2)(x-3)(x-4) - 0.0114(
    x+1)x(x-1)^2(x-2)(x-3)(x-4)^2 + 0.0105(x+1)x(x-1)^2(x-2)(x-3)(x-4)^2(x-5) -
     0.00393(x+1)x(x-1)^2(x-2)(x-3)(x-4)^2(x-5)(x-6) + 0.00137(x+1)x(x-1)^2(x
    -2)(x-3)(x-4)^2(x-5)(x-6)(x-7)

Wielomian interpolacyjny w postaci ogolnej:
W(x) = 0.00137x^11 - 0.0477x^10 + 0.702x^9 - 5.63x^8 + 26.6x^7 - 73.5x^6 + 106x
    ^5 - 38x^4 - 92.7x^3 + 116x^2 - 38.7x + 2
```