Algorytmy Numeryczne - Faza 4

Bartłomiej Kruk, Mateusz Motyl, Szymon Lichota

26 maja 2021

1 Treść zadania

Dana jest tabela wartości funkcji i jej pochodnej

x_i	-1	0	1	2
$f(x_i)$	-1	-1	-1	-1
$f'(x_i)$	-6	2	-2	6
$f''(x_i)$		-2	-2	22

Rozwiązując zagadnienie interpolacyjne z węzłami wielokrotnymi znaleźć funkcję wielomianową pasującą do podanych danych. Program powinien umożliwiać użycie tylko niektórych informacji z tabeli pozwalając użytkownikowi wybrać krotność użytych węzłów.

2 Teoretyczny opis metody

2.1 Teoria

Niech $x_0 < x_1 < ... < x_n$ i każdy x_j ma krotność $m_j \geq 1$. Definiujemy $t_0 \leq t_1 \leq ... \leq t_m$,
gdzie $m+1 = \sum_{j=0}^{n} m_j$, jako $t_0 = t_1 = ... = t_{m_0} - 1 = x_0,\ t_{m_0} = t_{m_0+1} = ... = t_{m_0+m_1-1} = x_1 ...$
Definiujemy ilorazy różnicowe:

$$f[t_j, t_{j+1}, ..., t_{j+k}] = \frac{f^{(k)}(t_j)}{k!}$$

jeśli $t_j = t_{j+k}$, oraz

$$f[t_j, t_{j+1}, ..., t_{j+k}] = \frac{f[t_{j+1}, ..., t_{j+k}] - f[t_j, ..., t_{j+k-1}]}{t_{j+k} - t_j}$$

gdy $t_j < t_{j+k}$.
Jeśli znane są wartości

$$f^{(i)}(x_j),\ j = 0, 1, ..., n,\ i = 0, 1, ..., m_j - 1,$$

gdzie $x_0 < x_1 < ... < x_n$ oraz $t_0 \leq t_1 \leq ... \leq t_m,\ m+1 = \sum_{j=0}^{n} m_j$, są dane przez x_j z uwzględnieniem krotności, to wielomian

$$H(x) = b_0 + b_1(x - t_0) + b_2(x - t_0) + b_2(x - t_0)(x - t_1) + ...$$
$$... + b_m(x - t_0) \cdot ... \cdot (x - t_{m-1})$$

spełniający warunki interpolacji Hermite'a

$$H^{(i)}(x_j) = f^{(i)}(x_j),\ j = 0, 1, ..., n,\ i = 0, 1, ..., m_j - 1,$$

ma współczynniki

$$b_j = f[t_0, t_1, ..., t_j],\ j = 0, 1, ..., m.$$

2.2 Przykład

Do interpolacji danych

x_i	0	1	2
$f(x_i)$	0	1	2
$f'(x_i)$	1	0	
$f''(x_i)$		2	

wykorzystamy postać Newtona wielomianu Hermite'a $H = H(x)$. Mamy:

$$x_0 = 0,\ m_0 = 2,\ x_1 = 1,\ m_1 = 3,\ x_2 = 2,\ m_2 = 1,$$
$$t_0 = t_1 = 0 = x_0,\ t_2 = t_3 = t_4 = 1 = x_1,\ t_5 = 2 = x_2$$

$$H(x) = b_0 + b_1(x - t_0) + b_2(x - t_0)(x - t_1) + b_3(x - t_0)(x - t_1)(x - t_2) +$$
$$+b_4(x - t_0)(x - t_1)(x - t_2)(x - t_3) + b_5(x - t_0)(x - t_1)(x - t_2)(x - t_3)(x - t_4),$$

$$H(x) = b_0 + b_1x + b_2x^2 + b_3x^2(x - 1) + b_4x^2(x - 1)^2 + b_5x^2(x - 1)^3.$$

Współczynniki dla wielomianu w postaci Newtona wyznaczamy za pomocą ilorazów różnicowych.
$f[t_0] = f[0] = 0 = b_0$
$f[t_1] = 0,\ f[t_0, t_1] = \frac{f'(t_0)}{1!} = 1 = b_1,$
$f[t_2] = 1,\ f[t_1, t_2] = \frac{f[t_2] - f[t_1]}{t_2 - t_1} = 1,\ f[t_0, t_1, t_2] = \frac{f[t_1, t_2] - f[t_0, t_1]}{t_2 - t_0} = 0 = b_2$
$f[t_3] = 1,\ f[t_2, t_3] = 0,\ f[t_1, t_2, t_3] = -1,\ f[t_0, t_1, t_2, t_3] = -1 = b_3$
$f[t_4] = 1,\ f[t_3, t_4] = 0,\ f[t_2, t_3, t_4] = 1,\ f[t_1, t_2, t_3, t_4] = 2,\ f[t_0, t_1, t_2, t_3, t_4] = 3 = b_4$
$f[t_5] = 2,\ f[t_4, t_5] = 1,\ f[t_3, t_4, t_5] = 1,\ f[t_2, t_3, t_4, t_5] = 0,\ f[t_1, t_2, t_3, t_4, t_5] = -1,\ f[t_0, t_1, t_2, t_3, t_4, t_5] = -2 = b_5$
Otrzymujemy

$$b_0 = 0,\ b_1 = 1,\ b_2 = 0,\ b_3 = -1,\ b_4 = 3,\ b_5 = -2,$$

a szukanym wielomianem jest

$$H(x) = -2x^5 + 9x^4 - 13x^3 + 6x^2 + x.$$

3 Opis implementacji

Procedura interpolacji Hermite'a w postaci Newtona jest następująca:
Tworzymy tabelkę uzupełniając ją wartościami:
$x_i\ f(x_i)$
$x_1\ f(x_1)$
$x_2\ f(x_2)$
$\vdots$
$x_n\ f(x_n)$
przy czym x_i może równać się x_{i-1} (jeżeli węzeł jest k-krotny to umieszczamy go w tabeli k razy).
Uzupełniamy tabelkę dopisując do kolejnych kolumn ilorazy różnicowe:
$x_i\ f(x_i)\ f[x_{i-1}, x_i]$
$x_0\ f(x_0)$
$x_1\ f(x_1)\ f[x_0, x_1]$
$x_2\ f(x_2)\ f[x_1, x_2] \quad f[x_0, x_1, x_2]$
$\vdots$
$\vdots \quad \vdots \quad \vdots$
$x_n f(x_n)\ f[x_{n-1}, x_n]\ f[x_{n-2}, x_{n-1}, x_n]$ ... $f[x_0$ ... $x_n]$
aż skończy się możliwość dalszego dopisywania. Współczynnikami b_i są wartości leżące na przekątnej.

Algorithm 1: Wyznaczenie długości pomocniczej tablicy do interporacji Hermite'a

```
len - liczba wierszy w tablicy ilorazów różnicowych.
Jest ich dokładnie tyle, ile podanych wartości funkcji f oraz jej
pochodnych w punktach x_i. Zliczamy ilość niepustych komórek tabeli:
for i = 1, 2, ..., columnsSize do
    for j = 0, 1, ..., rowsSize do
        if F_{i,j} != null then
            len = len + 1
        end
    end
end
```

Algorithm 2: Wypełnienie pomocniczej tablicy ilorazów różnicowych

Q - dwuwymiarowa tablica ilorazów różnicowych.
Dwie pierwsze kolumny tabeli pomocniczej zostają wypełnione wartościami z tabeli.
Pierwsza: x_i, druga: $f(x_i)$.

for $b = 0,\ 1,\ ...,\ columnsSize$ **do**
  for $a = 1,\ 2,\ ...,\ rowsSize$ **do**
    if $F_{a,b}\ != null$ **then**
      $Q_{i,0} = F_{0,b}$
      $Q_{i,1} = F_{1,b}$
      $i = i + 1$

Wypełnienie pozostałych kolumn za pomocą ilorazów różnicowch.

for $i = 1,\ 2,\ ...,\ dlugosc$ **do**
  Dla różnych t_j i t_{j+k} :
  if $Q_{i,0}! = Q_{i-1,0}$ **then**
    for $j = 2,\ ...,\ i + 1$ **do**
      $Q_{i_j} = \frac{Q_{i,j-1} - Q_{i-1,j-1}}{Q_{i,0} - Q_{i-j+1,0}}$
  else
    Dla $t_j = t_{j+k}$ używamy węzłów większej krotności:
    for $l = i, ...,\ 1$ **do**
      if $Q_{l,0} == Q_{l-1,0}$ **then**
        $counter = counter + 1$
      else
        break
    for $d = 0,\ ...,\ m$ **do**
      if $F_{0,d} == Q_{i,0}$ **then**
        $p = 2$
        for $j = 2,\ ...,\ 1 + counter$ **do**
          $Q_{i,j} = \frac{F_{p,d}}{(j-1)!}$
          $p = p + 1$
        if $counter == i$ **then**
          break
        else
          for $j = 2 + counter,\ ...,\ i + 1$ **do**
            $Q_{i,j} = \frac{Q_{i,j-1} - Qi-1,j-1}{Q_{i,0} - Qi-j+1,0}$

4 Demonstracja

4.1 Opis programu

Specyfikacja wejścia
W pierwszym wierszu standardowego wejścia podajemy liczbę wierszy dwuwymiarowej tablicy danych. Następnie podajemy liczbę kolumn. Najpierw do tablicy danych wstawiamy kolejne wartości całkowite x_i takie, że $0 \leq i < m$, gdzie m to liczba kolumn oraz $x_0 < x_1 < ... < x_i$. Dalej podajemy wartości całkowite $f^j(x_i)$, przy założeniu, że $0 \leq j \leq n$, gdzie $n + 2$ to liczba wierszy. W przypadku braku wartości wstawiamy N.
Specyfikacja wyjścia
Program symuluje interpolacje z węzłami wielokrotnymi i znajduje funkcję wielomianową pasującą do podanych danych w postaci Newtona: $H(x) = b_0 + b_1(x - t_0) + ... + b_m(x - t_0) \cdot ... \cdot (x - t_{m-1})$.

4.2 Przykład

x_i	0	1	2
$f(x_i)$	0	1	2
$f'(x_i)$	1	0	
$f''(x_i)$		2	

Wejście:
4
3
0
1
2
0
1
2
1
0
N
N
2
N

Wyjście:
$H(x) = (0) + (1)(x - 0) + (-1)(x - 0)(x - 0)(x - 1)+$
+ (3)(x - 0)(x - 0)(x - 1)(x - 1) + (-2)(x - 0)(x - 0)(x - 1)(x - 1)(x - 1)

Inne przykłady:

x_i	-1	0	1	2
$f(x_i)$	-1	-1	-1	-1
$f'(x_i)$	-6	2	-2	6
$f''(x_i)$		-2	-2	22

Wyjście:
$H(x) = (-1) + (-6)(x + 1) + (6)(x + 1)(x + 1)+$
+ (-4)(x + 1)(x + 1)(x - 0) + (1)(x + 1)(x + 1)(x - 0)(x - 0)

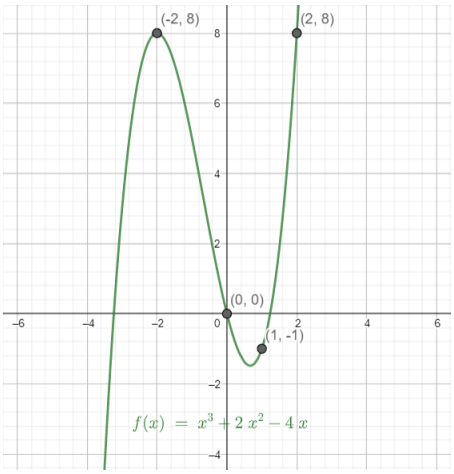
x_i	-1	0	1
$f(x_i)$	2	1	2
$f'(x_i)$	-8	0	8
$f''(x_i)$	56	0	56

Wyjście:
$H(X) = (2) + (-8)(x + 1) + (28)(x + 1)(x + 1) +$
$+$ (-21)(x + 1)(x + 1)(x + 1) + (15)(x + 1)(x + 1)(x + 1)(x - 0) +
$+$ (-10)(x + 1)(x + 1)(x + 1)(x - 0)(x - 0) +
$+$ (4)(x + 1)(x + 1)(x + 1)(x - 0)(x - 0)(x - 0) +
$+$ (-1)(x + 1)(x + 1)(x + 1)(x - 0)(x - 0)(x - 0)(x - 1) +
$+$ (1)(x + 1)(x + 1)(x + 1)(x - 0)(x - 0)(x - 0)(x - 1)(x - 1)

x_i	-2	0	1	2
$f(x_i)$	8	0	-1	8
$f'(x_i)$	0	-4	3	16

Wyjście:
$H(x) = (8) + (-2)(x + 2)(x + 2) + (1)(x + 2)(x + 2)(x - 0)$

$\Rightarrow \mathrm{x}^3 + 2x^2 - 4x$



Rysunek 1: Wykres przedstawia funkcję $f(x) = x^3 + 2x^2 - 4x$ wraz z nałożonymi punktami: $A = (-2, 8)$, $B = (0, 0)$, $C = (1, -1)$, $D = (2, 8)$.

5 Wnioski

Istnienie i jednoznaczność rozwiązania:
Jeżeli liczby $x_0, x_1, ..., x_n$ są parami różne, to dla wartości $y_0, y_1, ..., y_n$ istnieje dokładnie jeden wielomian P_n stopnia nie wyższego niż n spełniający warunki

$$y_i = W_n(x_i) \ (i = 0, 1, ..., n).$$

Dowód. Jednoznaczność. Zakłada się, że istnieją dwa różne wielomiany $W_1(x)$ i $W_2(x)$ stopnia n, przyjmujące w węzłach $x_0, x_1, ..., x_n$ takie same wartości.
Niech

$$W_3(x) = W_1(x) - W_2(x)$$

$W_3(x)$ jest wielomianem stopnia co najwyżej n (co wynika z własności odejmowania wielomianów).
Ponieważ $W_1(x)$ i $W_2(x)$ w węzłach $x_i, i \in 0, 1, ..., n$ interpolują tę samą funkcję, to $W_1(x_i) = W_2(x_i)$, a więc $W_3(x_i) = 0$ (węzły interpolacji są pierwiastkami $W_3(x)$).
Każdy niezerowy wielomian stopnia n ma co najwyżej n pierwiastków rzeczywistych, a ponieważ z powyższego wiadomo, że $W_3(x)$ ma $n + 1$ pierwiastków, to $W_3(x)$ musi być wielomianem tożsamściowo równym zeru, a ponieważ:

$$W_3(x) = W_1(x) - W_2(x) = 0$$

to

$$W_1(x) = W_2(x)$$

co jest sprzeczne z założeniem, że $W_1(x)$ i $W_2(x)$ są różne. □

Dowód. Istnienie. Niech $x_0, x_1, ..., x_n$ będą węzłami interpolacji funkcji f takimi, że znane są wartości: $f(x_0) = y_0, f(x_1) = y_1, ..., f(x_n) = y_n$. Można zdefiniować funkcję:

$$L_i(x) = \prod_{j=0 \wedge j \neq i}^{n} \frac{x - x_j}{x_i - x_j}, i \in 0, 1, ..., n$$

taką, że dla $x \notin \{x_0, x_1, ..., x_n\}$ $L_i(x)$ jest wielomianem stopnia n (mianownik jest liczbą, a licznik iloczynem n wyrazów postaci $(x - x_j)$).
Gdy $x_k \in \{x_0, x_1, ..., x_n\}$ i $k = i$:

$$L_i(x) = \prod_{j=0 \wedge j \neq i}^{n} \frac{x - x_j}{x_i - x_j} = \prod_{j=0 \wedge j \neq i}^{n} (1) = 1.$$

Gdy $x_k \in \{x_0, x_1, ..., x_n\}$ i $k \neq i$:

$$L_i(x) = \prod_{j=0 \wedge j \neq i}^{n} \frac{x - x_j}{x_i - x_j} = \frac{(x_k - x_0)(x_k - x_1) \cdot ... \cdot (x_k - x_k) \cdot ... \cdot (x_k - x_n)}{(x_i - x_0)(x_i - x_1) \cdot ... \cdot (x_i - x_{i-1})(x_i - x_{i+1}) \cdot ... \cdot (x_i - x_n)} = 0$$

(licznik jest zerem, ponieważ występuje element $(x_k - x_k)$).
Niech $W(x)$ będzie wielomianem stopnia co najwyżej n, określonym jako:

$$W(x) = y_0 L_0(x) + y_1 L_1(x) + ... + y_n L_n(x).$$

Dla $X_i \in \{x_0, x_1, ..., x_n\}$:

$$W(x_i) = y_0 L_0(x_i) + y_1 L_1(x_i) + ... + y_i L_i(x_i) + ... + y_n L_n(x_i).$$

Wszystkie składniki sumy o indeksach różnych od i są równe zeru (ponieważ dla $j \neq i L_i(x_j) = 0$), składnik o indeksie i jest równy:

$$L_i(x_i) y_i = 1 \cdot y_i = y_i,$$

a więc

$$W(x_i) = y_i,$$

z czego wynika, że $W(x)$ jest wielomianem interpolującym funkcję $f(x)$ na zbiorze punktów $x_0, x_1, ..., x_n$. □

6 Kod źródłowy

```
import java.util.Scanner;
public class Interpolation {
    String F[][];
    int Q[][];
    Scanner input = new Scanner(System.in);
    int argX;
    String valY, s;
    int n, m, length;

    public Interpolation(int n, int m){
        this.n = n;
        this.m = m;
        F = new String[n][m];
        System.out.println("Podaj wartości Xi, od najmniejszej do najwiekszej:");
        for(int i = 0; i < m; i++){
            argX = input.nextInt();
            s = String.valueOf(argX);
            F[0][i] = s;
            System.out.print(" Xi | ");
            for(int p = 0; p <= i; p++){
                System.out.print(F[0][p] + " | ");
            }
            System.out.println();
        }
        System.out.println("Teraz podaj wartości funkcji w tych punktach:");
        System.out.println("Jeśli wskazane przez '?' pole jest liczbą, wpisz ją.
                                        Jeśli jest puste, wpisz literę 'N'");
        for(int i = 1; i < n; i++){
            for(int j = 0; j < m; j++){
                F[i][j] = "  ?  ";
                showData();
                while(true){
                    valY = input.next();
                    if(valY.equals("N")){
                        F[i][j] = valY;
                        break;
                    }
                    else if(isNumeric(valY)){
                        F[i][j] = valY;
                        break;
                    }
                    else{
                        System.out.println("Podano zla wartosc, podaj jeszcze raz");
                        return;
                    }
                }
            }
        }
        showData();
        getLength();
        Q = new int[length][length+1];
        diffQuotients();
        //showDiffQuotientsArr(); //wyświetlenie tablicy 2d ilorazów różnicowych
        //showCoeff(); //wyswietlenie wspolczynników b_0, ... b_n.
        showPoly();
    }
```

```
private void showCoeff() {
    System.out.println();
    for(int i = 0; i < length; i++){
        System.out.println("b" + i + ": " + Q[i][i + 1]);
    }
}
private int getLength() {
    length = 0;
    for(int i = 1; i < n; i++){
        for(int j = 0; j < m; j++){
            if(!F[i][j].equals("N")){
                length += 1;
            }
        }
    }
    return length;
}
private void diffQuotients() {
    int w = 0;
    for(int b = 0; b < m; b++){
        for(int a = 1; a < n; a++){
            if(!F[a][b].equals("N")){
                Q[w][0] = Integer.parseInt(F[0][b]);
                Q[w][1] = Integer.parseInt(F[1][b]);
                w += 1;
            }
        }
    }
    for(int i = 1; i < length; i++){
        if(Q[i][0] != Q[i-1][0]) {
            for (int j = 2; j <= i + 1; j++) {
                Q[i][j] = (Q[i][j - 1] - Q[i - 1][j - 1]) / (Q[i][0] - Q[i - j + 1][0]);
            }
        }
        else{
            int counter = 0;
            for(int l = i; l > 0; l--){
                if(Q[l][0] == Q[l-1][0])
                    counter += 1;
                else break;
            }
            for(int d = 0; d < m; d++){
                if(Integer.parseInt(F[0][d]) == Q[i][0]){
                    int p = 2;
                    for(int j = 2; j < 2 + counter; j++){
                        Q[i][j] = Integer.parseInt(F[p][d]) / factorial(j - 1);
                        p+=1;
                    }
                    if(counter == i) break;
                    else {
                        for (int j = 2 + counter; j <= i + 1; j++)
                            Q[i][j] = (Q[i][j - 1] - Q[i - 1][j - 1]) / (Q[i][0] - Q[i - j + 1][0]);
                    }
                }
            }
        }
    }
}
```

```
private void showDiffQuotientsArr() {
    System.out.println();
    for(int i = 0; i < length; i++){
        for(int j = 0; j < length + 1; j++){
            System.out.print(Q[i][j] + " | ");
        }
        System.out.println();
    }
}
public static int factorial(int n) {
    int fact = 1;
    for (int i = 1; i <= n; i++) {
        fact *= i;
    }
    return fact;
}
public void showData(){
    System.out.print(" Xi | ");
    for(int i = 0; i < m; i++){
        System.out.print(F[0][i] + " | ");
    }
    System.out.println();
    for(int i = 1; i < n; i++){

        s = "f";
        for(int st = 0; st < i; st++){
            if(st>0)
                s = s + "'";
        }
        s = s + "(argX)";
        System.out.print(s + " | ");

        for(int j = 0; j < m; j++){
            System.out.print(F[i][j] + " | ");
        }
        System.out.println();
    }
}
public void showPoly(){
    System.out.print("H(x) = (" + Q[0][1] + ") ");
    for(int i = 1; i < length; i++){
        if(Q[i][i + 1] == 0){
            continue;
        }
        if(i != length)
            System.out.print(" + ");

        System.out.print("(" + Q[i][i+1] + ")");
        for(int j = 1; j <= i; j++) {
            if(Q[j - 1][0] < 0) {
                System.out.print("(x + " + Math.abs(Q[j - 1][0]) + ")");
            }
            else {
                System.out.print("(x - " + Q[j - 1][0] + ")");
            }
        }
    }
}
```

```
    public static boolean isNumeric(String string) {
        int intValue;

        if(string == null || string.equals("")) {
            System.out.println("String cannot be parsed, it is null or empty.");
            return false;
        }

        try {
            intValue = Integer.parseInt(string);
            return true;
        } catch (NumberFormatException e) {
            System.out.println("Input String cannot be parsed to Integer.");
        }
        return false;
    }
}

import java.util.Scanner;
public class Main {
    public static void main(String [] args){
        Scanner input = new Scanner(System.in);
        System.out.println("Podaj wymiary tablicy danych");
        System.out.println("Liczba wierszy:");
        int row = input.nextInt();
        System.out.println("Liczba kolumn:");
        int col = input.nextInt();

        Interpolation in = new Interpolation(row, col);

    }
}
```

Literatura

[1] Interporacja Hermite'a

https://pl.wikipedia.org/wiki/Interpolacja_Hermite'a

[2] Interpolacja wielomianowa

https://pl.wikipedia.org/wiki/Interpolacja_wielomianowa

[3] Jaruszewska-Walczak Danuta. Wykład nr 3: interpolacja Hermite'a, ilorazy różnicowe dla węzłów wielokrotnych, postać Newtona wielomianu Hermite'a, reszta interpolacji