Monika Tworek
229776

Laboratorium – Lista 4

1. Zadanie 1

1.1. Opis problemu

W zadaniu należało napisać funkcję obliczającą ilorazy różnicowe:
function ilorazyRoznicowe (x::Vector{Float64}, f::Vector{Float64})

1.2. Opis rozwiązania

Dane:
x - wektor długosci n+1 zawierajacy wezły $x_0, ..., x_n$
f - wektor długosci n+1 zawierajacy wartosci interpolowanej funkcji w wezłach $f(x_0), ..., f(x_n)$
Wyniki:
fx - wektor długosci n+1 zawierajacy obliczone ilorazy róznicowe

1.3. Algorytm

Algorytm 1: Ilorazy Roznicowe

```
1  ilorazyRoznicowe ( x, f)
2  for i ← 1 to n do
3      fx(i) ← f(i)
4  end
5  for j ← 1 to n do
6      for i ← n to j step -1 do
7          fx(i) ← (fx(i)fx(i−1)) / (x(i)x(i−j))
8      end
9  end
10 return (fx)
```

1.4. Zasady działania

Funkcja opiera się na podstawie równania (Twierdzenie 3 Ilorazy różnicowe wyższych rzędów z wykładu): $f[x_0, x_1, ..., x_n] = \frac{f[x_1,...,x_n]-f[x_0,x_1,...,x_{n-1}]}{x_n-x_0}$

W pierwszej pętli kopiuje wszystkie wartości funkcji f do tablicy fx, by w kolejnej pętli korzystać ze wzoru i twierdzenia 4, że ilorazy różnicowe nie zależą od kolejności węzłów.

2. Zadanie 2

2.1. Opis problemu

W zadaniu należało napisać funkcję obliczającą wartość wielomianu interpolacyjnego stopnia n w postaci Newtona Nn(x) w punkcie x=t za pomocą algorytmu Hornera:

function warNewton (x::VectorFloat64, fx::VectorFloat64, t::Float64)

2.2. Opis rozwiązania

Dane:

x - wektor długosci n+1 zawierajacy wezły $x_0, ..., x_n$

f - wektor długosci n+1 zawierajacy wartosci interpolowanej funkcji w wezłach $f(x_0), ..., f(x_n)$

t – punkt, w którym należy obliczyć wartość wielomianu

Wyniki:

nt - wartość wielomianu w punkcie t

2.3. Algorytm

Algorytm 2: warNewtona

```
1 warNewtona ( x, f, t)
2 nt(i) ← fx(length(fx))
3 for i ← n to j step -1 do
4     nt(i) ← nt * (t − x[i] + fx[i])
5 end
6 return (fx)
```

2.3.1. Zasady działania

Do powyższego programu został użyty algorytm Hornera. Zaimplementowano algorytm z zadania 8 lista 4 z ćwiczeń.

3. Zadanie 3

3.1. Opis problemu

Znając współczynniki wielomianu interpolacyjnego w postaci Newtona oraz węzły napisać funkcję obliczającą współczynniki w postaci naturalnej :

function naturalna(x::VectorFloat64. Fx::VectorFloat64)

Funkcja obliczająca ma mieć złożoność $O(n^2)$

3.2. Opis rozwiązania

Dane:

x - wektor długości n+1 zawierający węzły x0, , xn

fx - wektor długości n+1 zawierający ilorazy różnicowe

Wyniki:

a - wektor długości n+1 zawierający obliczone współczynniki w postaci naturalnej

3.3. Algorytm

Algorytm 3: naturalna

```
1  naturalna ( x, fx)
2  n ← length(x)
3  poly[1, 1] ← 1
4  for j ← 1 to n − 1 do
5      poly[j + 1, 1] ← 1
6      poly[j + 1, 2 : j + 1] ← poly[j, 2 : j + 1] − x[j] ∗ poly[j, 1 : j]
7  end
8  for j ← n to 1 step -1 do
9      a[j] ← fx[j]
10     for i ← 1 to 1 do
11         a[j] ← a[j] + fx[j + 1] ∗ poly[j + i, i + 1]
12     end
13 end
14 return (a)
```

3.4. Zasady działania

Wiedząc, że współczynnik przy x^k to $f[x_0, x_1, ..., x_k]$, czyli c_n przy najwyżej potędze wielomianu w postaci Newtona. Korzystamy ponownie z uogólnionego schematu Hornera, a potem korzystamy ze wzoru na postać Newtona ze wzoru z wykładu:

$p(x) = \sum_{k=0}^{n} f[x_0, ..., x_k] \prod_{j=0}^{k-1} (x - x_j)$

Na tej podstawie najpierw tworzymy wielomian odpowiedniego stopnia zaczynawszy od współczynnika przy najwyższej potędze a następnie kolejne współczynniki. Ponieważ ta nierówność nie ma wpływ również na wcześniej obliczone współczynniki.

4. Zadanie 4

4.1. Opis problemu

Napisać funkcję interpolującą zadaną funkcjęf(x) w przedziale [a,b] za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona, a następnie rysująca wielomian interpolacyjny i interpolowaną funkcję korzystając z ilorazyRoznicowe i warNewton:

function rysujNnfx(f, a::Float64, b::Float64, n::Int)

4.2. Opis rozwiązania

Dane:

f - funkcja f(x) zadana jako anonimowa funkcja

a, b - przedział interpolacji

n - stopień wielomianu interpolacyjnego

Wyniki:

Wykres

4.3. Algorytm

Algorytm 4: rysujNnfx

```
1 rysujNnfx ( f, a, b, n)
2 h = (b−a)/n for i ← to n + 1 do
3     x[i] ← a + (i − 1) ∗ h y[i](x[i])
4 end
5 fxilorazyRoznicowe(x, y) for i ← to 100 do
6     x[i] ← a + (i − 1) ∗ h functionResults[i]f(t) interpResults[i] ← warnewton(x, fx, t)
7 end
8 return (plot)
```

4.4. Zasady działania

Do wyznaczenia wykresu trzeba było najpierw wyznaczyć równoodległe węzły i wyliczenie odległości między nimi korzystając z h = $\frac{b-a}{n}$ .Wyliczenie wartość zadanej funkcji w węzłach. Za pomocą funkcji z zadania 1 ilorazyRoznicowe oraz tablic z węzłami i wartościami funkcji wyznaczenie wektora ilorazów różnicowych. Następnie wyznaczenie wartości funkcji interpolowanej i wielomianu interpolacyjnego w m równoodległych punktach (dla zwiększenia dokładności m = 100) na zadanym przedziale i użycia pakietu Plots do narysowania wykresu.

5. Zadanie 5

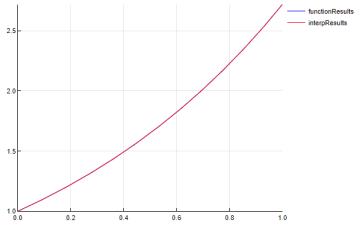
5.1. Opis problemu

Przetestować funkcję rysujNnfx(f, a, b, n) na następujących przykładach:
a) f(x) = $e^x, [0, 1], n = 5, 10, 15$
b) f(x) = $x^2 sinx, [-1, 1], n = 5, 10, 15$

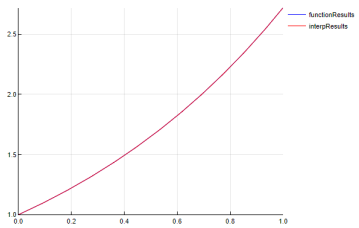
5.2. Opis rozwiązania

Napisanie programu, w którym wywołuję funkcję rysujNnfx z obu podpunktów dla każdego n.

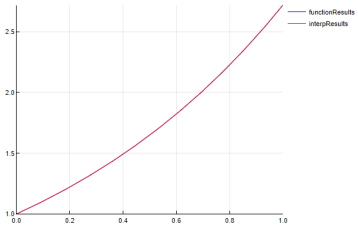
5.3. Wyniki



(a) n=5

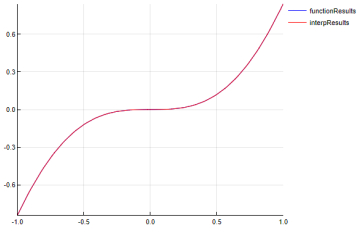


(b) n=10

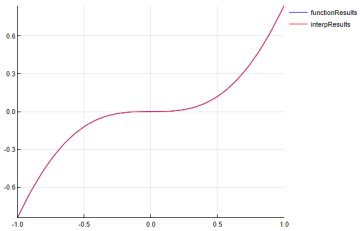


(c) n=15

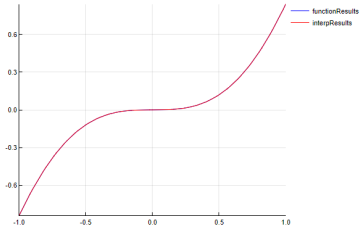
Rysunek 1: Dla przykładu a



(a) n=5



(b) n=10



(c) n=15

Rysunek 2: Dla przykładu b

5.4. Wnioski

Możemy zauważyć, że dla funkcji danych w zadaniu zwiększenie stopnia wielomianu wpływa na poprawę interpolacji (linie czerwone odpowiadające wielomianowi interpolacyjnemu coraz bardziej pokrywają się z zielonymi, odpowiadającymi funkcjom interpolowanym). Wykresy wielomianów interpolacyjnych oraz opisanych we wstępie funkcji, prawie się pokrywają. Co jednak ważniejsze, można zauważyć, że wraz ze wzrostem stopnia wielomianu, bardzo mocno wzrasta także jego dokładność (podobieństwo do oryginalnej funkcji).

6. Zadanie 6

6.1. Opis problemu

Przetestować funkcję rysujNnfx(f, a, b, n) w następujących przykładach obrazujących zjawisko rozbieżności:

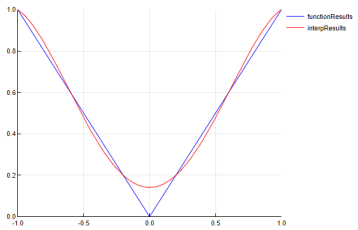
a) f(x) = —x—, [-1, 1], n = 5, 10, 15

b) f(x) = $1/(1+x^2), [-5, 5], n = 5, 10, 15 (zjawiskoRungego)$

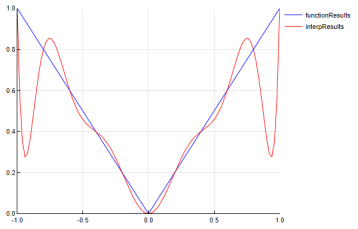
6.2. Opis rozwiązania

Napisanie programu, w którym wywołuję funkcję rysujNnfx z obu podpunktów dla każdego n w celu zauważenia zjawiska rozbieżności.

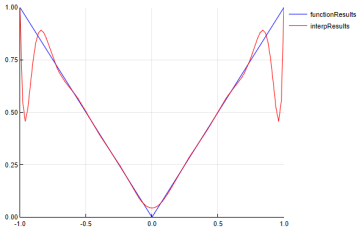
6.3. Wyniki



(a) n=5

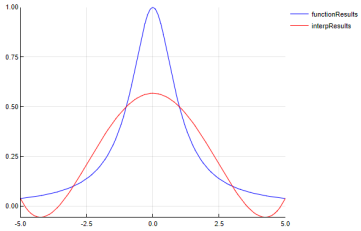


(b) n=10

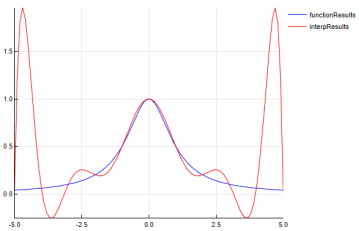


(c) n=15

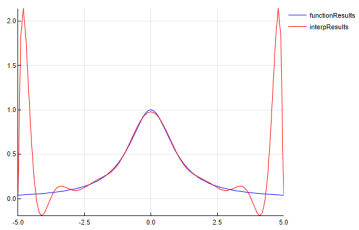
Rysunek 3: Dla przykładu a



(a) n=5



(b) n=10



(c) n=15

Rysunek 4: Dla przykładu b

6.4. Wnioski

Niestety, od razu można zauważyć, że wielomian interpolacyjny zdecydowanie mniej przypomina funkcje interpolowaną niż w przypadku poprzedniego zadania. Zjawisko to szczególnie mocno rzuca się w oczy w okolicach końców badanego przedziału. Możemy zauważyć, że dla danych funkcji zwiększenie stopnia wielomianu poprawia interpolację funkcji tylko do pewnego momentu (w naszym przypadku do n = 10). Dalsze zwiększanie stopnia wielomianu wpływa na pogorszenie interpolacji funkcji. Jest to przykład zjawiska Runge'ego. Jest ono typowe dla wielomianow interpolacyjnych, w ktorych odległość między kolejnymi węzłami jest stała. Aby uniknąć tego zjawiska stosuje się zmienną odległość między węzłami (odległość między kolejnymi węzłami na krańcach przedziału jest mniejsza, niż w środku przedziału). Inne rozwiązanie to wybieranie jako węzłów zer wielomianu Czebyszewa.