Obliczenia naukowe

Sprawozdanie

Lista 4

Mateusz Laskowski

09.12.2018

1. **Zadanie 1**
   1. **Opis problemu**

Napisać funkcję obliczająca ilorazy różnicowe. Zaimplementować funkcję bez użycia tablicy dwuwymiarowej, czyli macierzy.

* 1. **Opis rozwiązania**

**Model funkcji:**

function ilorazyRoznicowe (x::Vector{Float64}, f::Vector{Float64})

**Dane wejściowe:**

x – wektor długości zawierający węzły , gdzie

,

f – wektor długości zawierających wartości interpolowanej funkcji w

węzłach

**Dane wyjściowe** (fx):

fx – wektor długości zawierający obliczone ilorazy różnicowe

,

**Analiza:**

Implementacja metody wyliczania ilorazów różnicowych to pierwszy krok do stworzenia algorytmu aproksymującego funkcję metodą Newtona. Podstawowy wzór rekurencyjny, który jest potrzebny do stworzenia odpowiedniego algorytmu:

Wyżej wypisany wzór rekurencyjny, rozwijamy do postaci ilorazu pojedynczego węzła, który jest znany i wynosi . W tablicy trójkątnej można zobrazować wartości ilorazów potrzebne do wyliczenia interesującego nas ilorazu. Przykładowo dla , aby wyliczyć iloraz w komórce potrzebne są wartości z komórek oraz . Wartości znane są, więc postępując od lewej strony tablicy trójkątnej, można wypełnić całą tablicę.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |
|  |  |  |
|  |  |

Z powyższej tablicy można rozpisać wielomian:

W zadaniu trzeba zaimplementować funkcję, która będzie zwracała wektor ilorazów. Na wyżej podanym przykładzie z tablicy trójkątnej, powinniśmy zwrócić wektor .

* 1. **Implementacja**

Założenie zadanie wymagało, aby przy wyliczeniach ilorazów nie używać tablicy dwuwymiarowej, więc w mojej implementacji posłużyłem się wzorem rekurencyjnym, wypisanym w analizie problemu.

1. **Zadanie 2**
   1. **Opis problemu**

Napisać funkcję obliczającą wartość wielomianu interpolacyjnego stopnia w postaci Newtona w punkcie za pomocą uogólnionego algorytmu Hornera, w czasie .

* 1. **Rozwiązanie problemu**

**Model funkcji:**

function warNewton (x::Vector{Float64}, fx::Vector{Float64}, t::Float64)

**Dane wejściowe:**

x – wektor długości zawierający węzły , gdzie

,

fx – wektor długości zawierający obliczone ilorazy różnicowe

,

t – punkt, w którym należy obliczyć wartość wielomianu

**Dane wyjściowe** (**nt**):

nt – wartość wielomianu w punkcie t

**Analiza:**

Uogólniony algorytm Hornera pozwala na wyliczenie wartości wielomianu interpolacyjnego Newtona:

Aby wyliczyć daną wartość trzeba rozwinąć wzór rekurencyjny:

który zakończy się, gdy , czyli dla , za *x* podstawiając wybrany argument.

* 1. **Implementacja**

Algorytm trzeba było zaimplementować w czasie , gdzie na wejściu otrzymuje wektor węzłów x oraz wektor ilorazów różnicowych fx. Algorytm generuje tablicę W długości n i wypełnia jej ostatnią komórkę wartością (wartość brana z wektora fx). W każdym kolejnym kroku pętli przeskakuję do poprzednich komórek, wypełniając je według schematu . Aby wypełnić całą tabele W trzeba wykonać tylko jedno przejście więc złożoność algorytmu jest liniowa.

1. **Zadanie 3**
   1. **Opis problemu**

Napisać funkcję obliczającą w czasie wpółczynniki postaci naturalnej wielomianu interpolacyjnego, tzn. .

Funkcja ma działać dla zadanych współczynników wielomianu interpolacyjnego w postaci Newtona (ilorazy różnicowe) oraz węzłów .

* 1. **Rozwiązanie problemu**

**Model funkcji:**

function naturalna (x::Vector{Float64}, fx::Vector{Float64})

**Dane wejściowe:**

x – wektor długości zawierający węzły , gdzie

,

fx – wektor długości zawierający obliczone ilorazy różnicowe

,

**Dane wyjściowe**:

a – wektor długości zawierający obliczone współczynniki postaci

naturalnej

*,*

**Analiza:**

Zadanie bazuje na wynikach z zadania 9 z listy 4 z ćwiczeń, gdzie przedstawione zostało, że za pomocą wielomianu w postaci Newtona oraz użyciu wielomianów pomocniczych z metody Hornera, można wyznaczyć współczynniki postaci naturalnej. Ponieważ przy każdym kroku algorytmu Hornera, każdy kolejny zależy tylko od dwóch współczynników z poprzedniej iteracji algorytmu Hornera.

* 1. **Implementacja**

Na początku tworzona jest tablica A[n], w której będą trzymane kolejne współczynniki. Zaczyna się od końca, czyli od A[n] podstawiając iloraz różnicowy z n-tego węzła. Kolejne współczynniki obliczamy, biorąc wielomian dla wcześniejszego elementu i wyliczając wielomian pomocniczy dla tego, kolejnego współczynnika. W każdym następnym kroku współczynniki z wielomianu pomocniczego zestawiamy z dotychczas wyliczonymi współczynnikami. Powtarzane jest dopóki nie dojdziemy do współczynnika . Po przejściu całego algorytmu otrzymujemy tabelę A[n] będącą wektorem współczynników wielomianu postaci normalnej. W każdej iteracji algorytmu należy dokonać zestawienia do n współczynników, co daje złożoność , bo .

1. **Zadanie 4**
   1. **Opis problemu**

Napisać funkcję, która zinterpoluje zadaną funkcję w przedziale , za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona. Następnie wygenerować wielomian interpolacyjny i interpolowaną funkcję, przy pomocy pakietu Plots, PyPlot lub Gadfly. Do interpolacji należy użyć węzłów równoodległych, tzn.

*,*

* 1. **Rozwiązanie problemu**

**Model funkcji:**

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

**Dane wejściowe:**

f – funkcja zadana jako anonimowa funkcja,

a, b – przedział interpolacji

n – stopień wielomianu interpolacyjnego

**Dane wyjściowe**:

– funkcja rysuje wielomian interpolacyjny i interpolowaną funkcję w przedziale

* 1. **Implementacja**

Kroki, które wykonuje algorytm przy interpolacji:

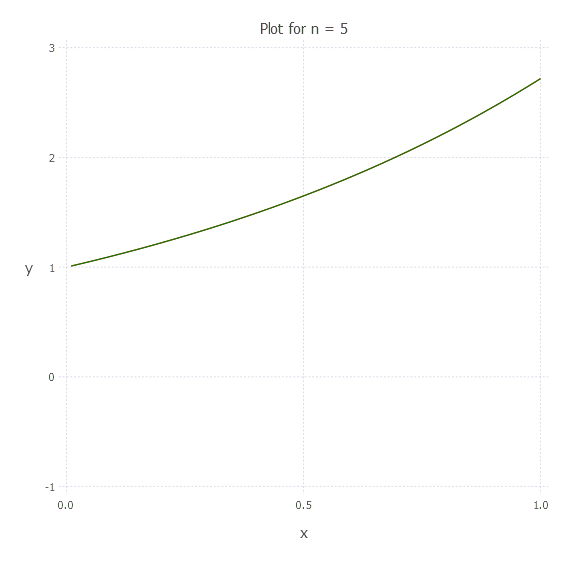
1. Generowanie wektora węzłów W
2. Wyliczanie wartości funkcji w węzłach, wpisywanie do tablicy
3. Użycie funkcji ilorazyRoznicowe (W, ), które generuje wektor ilorazów
4. Użycie funkcji function warNewton (W, ), które generuje tablicę , wartości funkcji w zakresie (w którym zostanie wygenerowany wykres)
5. Za pomocą tablicy generowanie wykresu wielomianu interpolacyjnego
6. Zestawienie generowanego wykresu z rzeczywistym wykresem interpolowanej funkcji
7. **Zadanie 5**
   1. **Opis problemu**

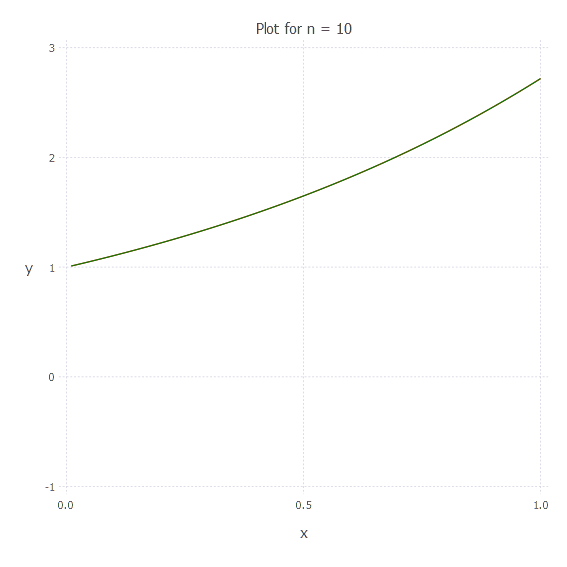
Wygenerować wykresy wielomianów interpolowanych (za pomocą wcześniej zaimplementowanych funkcji) z poniższych funkcji:

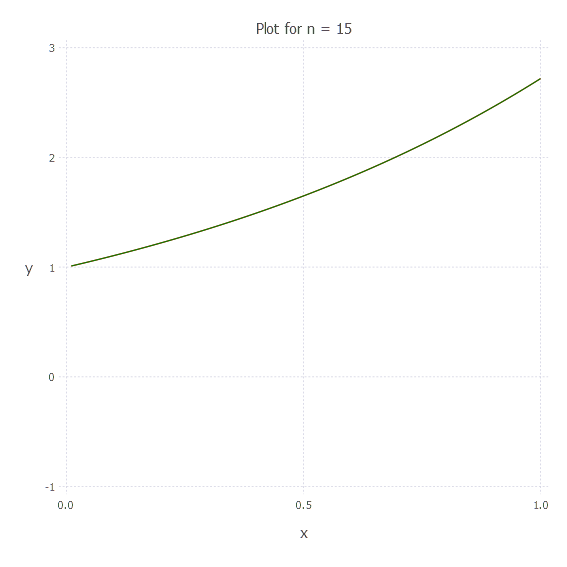
* 1. **Rozwiązanie problemu**

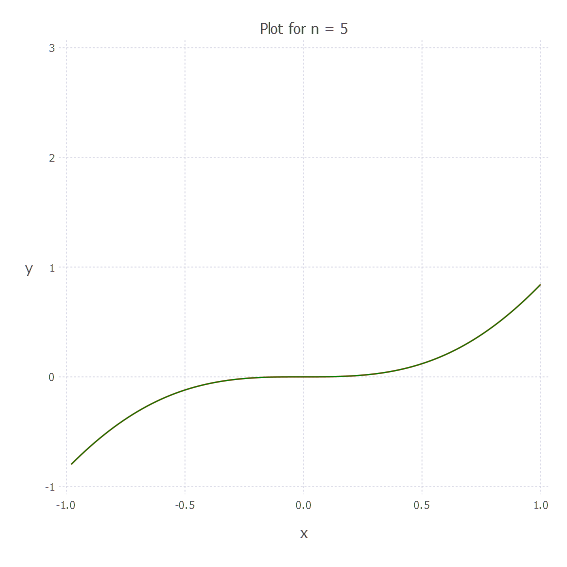
Wygenerowałem wykresy za pomocą funkcji rysujNnfx().

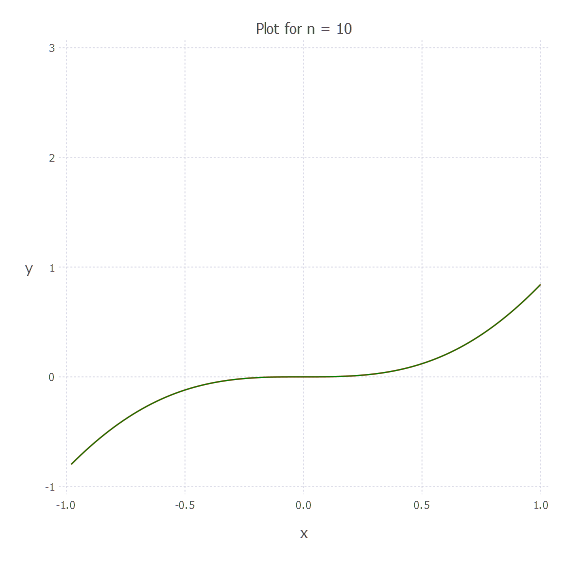
* 1. **Wyniki**

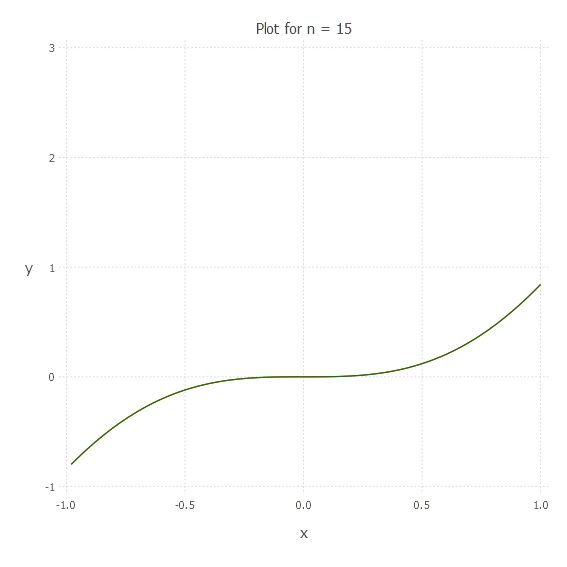
Ryc. 5.1

 Ryc. 5.2

 Ryc. 5.3

 Ryc. 5.4

 Ryc. 5.5

 Ryc. 5.6

* 1. **Wnioski**

Wykresy wielomianów będących interpolacją funkcji, pokryły się z wykresami tych funkcji. Interpolacja bardzo dobrze odwzorowała funkcje. Oczywiście trzeba mieć na uwadze, że wyliczenia są obarczone pewnymi małymi błędami obliczeń.

1. **Zadanie 6**
   1. **Opis problemu**

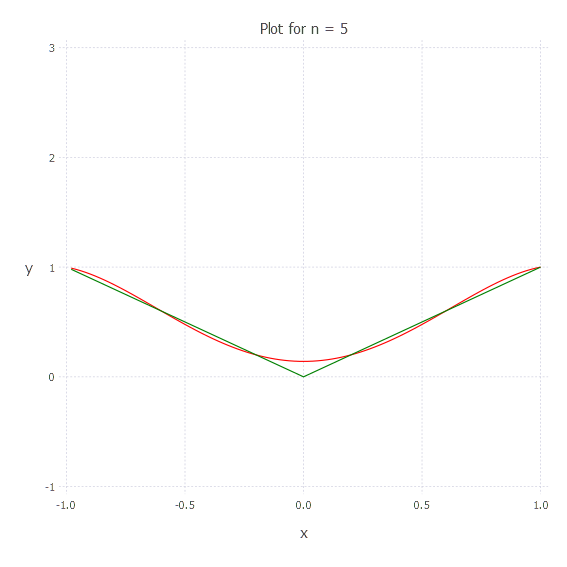
Wygenerować wykresy wielomianów interpolowanych (za pomocą wcześniej

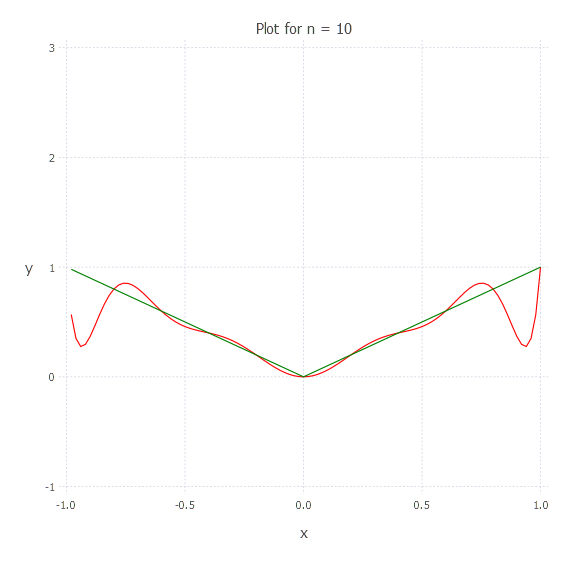
zaimplementowanych funkcji) z poniższych funkcji:

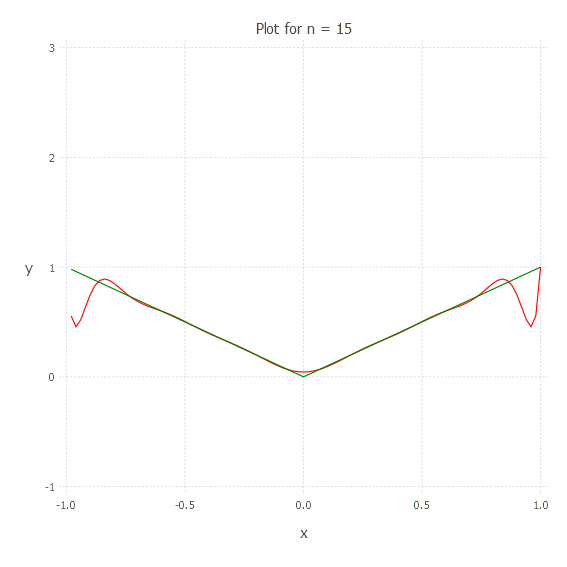
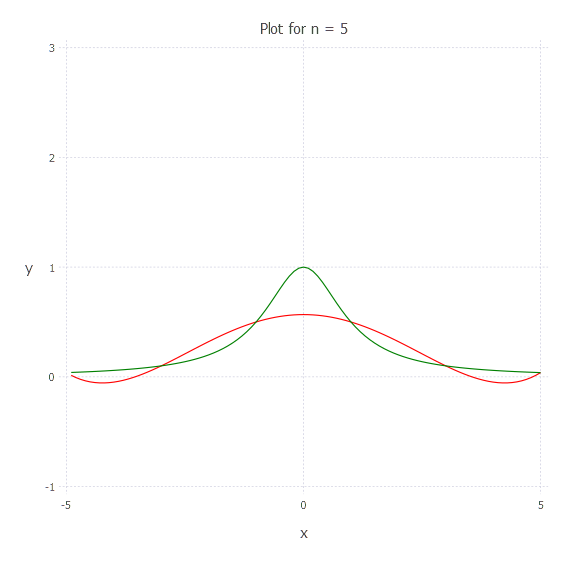
* 1. **Rozwiązanie problemu**

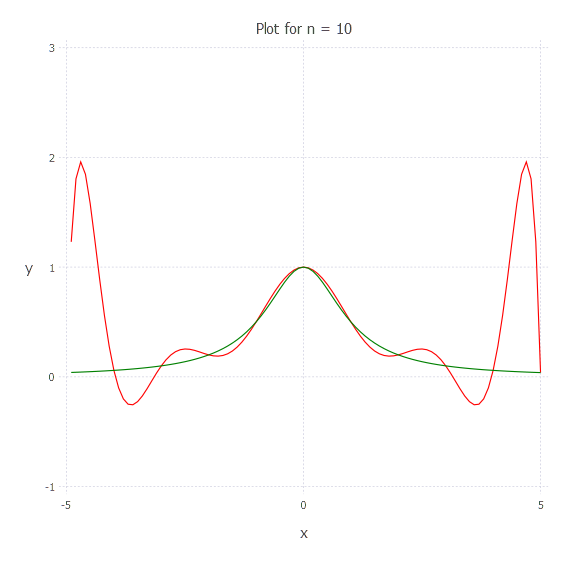
Wygenerowałem wykresy za pomocą funkcji rysujNnfx().

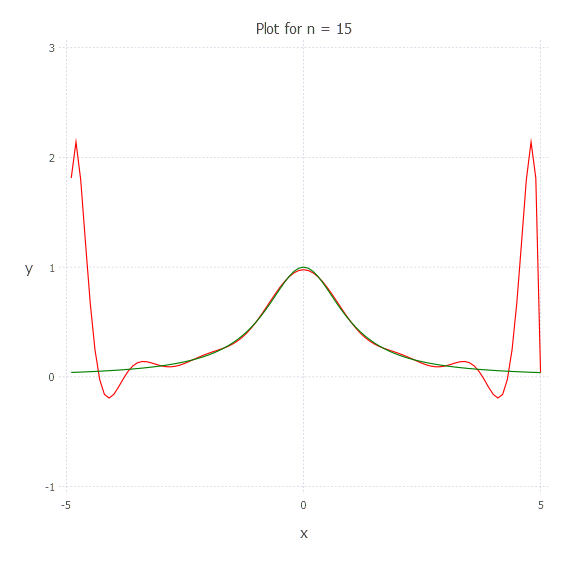
* 1. **Wyniki**

Ryc. 6.1 -

 Ryc. 6.2 -

 Ryc. 6.3 - Ryc. 6.4 -

 Ryc. 6.5 -

 Ryc. 6.6 -

* 1. **Wnioski**

W tym zadaniu są funkcje, dla których wielomiany interpolacyjne są podatne na tzw. Efekt Rungego, czyli pogorszenie jakości interpolacji mimo zwiększania ilości węzłów. Efekt Ryngego występuje gdy:

* + - Interpolowana funkcja jest nieciągła
    - Funkcja odbiega znacząco od funkcji gładkiej (funkcja gładka – funkcja ciągła mająca pochodne wszystkich rzędów)
    - Wielomiany interpolujące mają wysokie stopnie, a odległość między węzłami jest stała

Funkcja dla interpolacyjna nie była dokładna w okolicach , lecz bliska na krańcach przedziału. Po zwiększeniu ilości węzłów, wygenerowany wielomian był bardziej dokładny w centrum, lecz na krańcach pojawił się efekt Rungego, który pomimo ponownego zwiększenia ilości węzłów nie został zniwelowany. Głównym powodem wystąpienia efektu Rungego w tej funkcji jest brak ciągłości funkcji oraz równe odstępy między poszczególnymi węzłami.

W przypadku wielomianu będący interpolacją funkcji dla jest różny od rzeczywistego wykresu . Przy zwiększeniu ilości węzłów, interpolacja staje się bliższa rzeczywistej funkcji w okolicach , lecz na krańcach przedziału pojawia się jak w poprzednim przykładzie efekt Rungego. Zwiększając ilość węzłów efekt znów nie znikł, jak w przykładzie funkcji . Powodem wystąpienia efektu Rungego w funkcji jest nieciągłość funkcji oraz takie same odstępy między węzłami.

Można zniwelować błędy wynikające z efektu Rungego. Należy gęściej uwzględnić węzły na krańcach przedziału interpolowanej funkcji, wykorzystując wielomiany Czybyszczewa, których miejsca zerowe zagęszczają się na krańcach.