Obliczenia naukowe

Lista 4

Jakub Brodziński
229781

1 Zadanie 1

1.1 Opis problemu

Celem zadania było napisanie funkcji obliczającej ilorazy różnicowe dla przekazanych jako parametry węzłów oraz odpowiadających im wartości funkcji interpolowanej. Sygnatura funkcji powinna być postaci:

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

gdzie x to wektor długości $n{+}1$ zawierający węzły $x_0, .., x_n$, a f to wektor o tej samej długości zawierający wartości funkcji interpolowanej w węzłach $f(x_0), .., f(x_n)$. Funkcja powinna zwracać wektor fx o długości $n + 1$, gdzie $fx[i] = f[x_0, .., x_{i-1}]$. Ilorazy różnicowe można policzyć z ich ogólnego wzoru:

$$f[x_i, x_{i+1}, ..., x_{i+j}] = \frac{f[x_{i+1}, x_{i+2}, ..., x_{i+j}] - f[x_i, x_{i+1}, ..., x_{i+j-1}]}{x_{i+j} - x_i}$$

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

1.2 Opis rozwiązania

Przyjmując, że

$$c_{ij} = f[x_i, x_{i+1}, ..., x_{i+j}]$$

możemy stworzyć macierz trójkątna zawierącą wszystkie ilorazy różnicowe:

x_0	c_{00}	c_{01}	c_{02}	...	$c_{0,n-1}$	c_{0n}
x_1	c_{10}	c_{11}	c_{12}	...	$c_{1,n-1}$	
...	...	...	...			
x_{n-1}	$c_{n-1,0}$	$c_{n-1,1}$				
x_n	c_{n0}					

Wyrażając wzór ogólny na ilorazy róznicowe przy tych oznaczeniach otrzymujemy:

$$c_{ij} = \frac{c_{i+1,j-1} - c_{i,j-1}}{x_{i+j} - x_i}$$

Ze względu na fakt, że interesują nas tylko ilorazy róznicowe postaci $f[x_0, .., x_i]$ zamiast korzystać z macierzy trójkątnej (tj. tablicy dwuwymiarowej), potrzebne ilorazy policzyć przy użyciu jedynie tablicy jednowymiarowej. Początkowymi wartościami tablicy są wartości funkcji w węzłach $f(x_i)$. Tablice tworzymy kolumnami, a w każdej kolumnie - z góry do dołu (od najmniejszego do największego indeksu). Dzieki takiej kolejności tablica w momencie obliczania c_{ij} zawiera $c_{i+1,j-1}$ oraz $c_{i,j-1}$ z poprzedniej iteracji, które są konieczne do policzenia c_{ij}. Poniżej znajduje się pseudokod metody *ilorazyRoznicowe* korzystający bezpośrednio z wzoru rekurecyjnego przedstawionego powyżej.

```
function ilorazyRoznicowe(x,f)
    n ← length(x)
    for i = 1 to n do
        fx[i] ← f[i]
    end for
    for j = 2 to n do
        for i = n downto j do
            fx[i] ← (fx[i] − fx[i − 1])/(x[i] − x[i − j + 1])
        end for
    end for
    return fx
end function
```

1.3 Wyniki

Funkcja została poddana dwóm prostym testą, które na celu miały sprawdzić poprawność zwracanych wyników przez zaimplementowaną funkcje. Wywołania wraz z wynikami znajdują sie poniżej. W obu przypadkach policzone przez funkcje ilorazy różnicowe były prawidłowe.

$$ilorazyRoznicowe([1.0, 2.0], [1.0, 2.0]) = [1.0, 1.0]$$
$$ilorazyRoznicowe([3.0, 1.0, 5.0, 6.0], [1.0, -3.0, 2.0, 4.0]) = [1.0, 2.0, -0.375, 0.175]$$

2 Zadanie 2

2.1 Opis problemu

Celem zadania było napisanie funkcji obliczającej wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie t za pomocą uogólnionego algorytmu Hornera. Sygnatura funkcji powinna być postaci:

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

gdzie x to wektór zawierający węzły, fx to wektor zawierający ilorazy różnicowe (tak, że $fx[i] = f[x_0.x_1, ..., f_i]$), a t to punkt, w którym należy obliczyć wartość wielomianu.

2.2 Opis rozwiązania

Na wejsciu otrzymujemy wektory zawierające węzły oraz ilorazy różnicowe, dzięki czemu możemy wielomian N_n zapisać w takiej postaci:

$$N_n(x) = f[x_0] + (x - x_0)(f[x_0, x_1] + ... + (x - x_{n-1})(f[x_0, .., x_{n-1}] + (x - x_n)(f[x_0, ..., x_n]))...)$$

Co uwzględniając, że $f[x_0, .., x_i] = fx[i]$ oraz $x_i = x[i]$ możemy zapisać:

$$N_n(x) = fx[0]+(x-x[0])(fx[1]+...+(x-x[n-2])(fx[n-2]+(x-x[n-1])(fx[n-1]+(x-x[n])(fx[n])))...)$$

Mając taki wzór wielomianu w łatwy sposób można policzyć jego wartośc w punkcie t:

$$N_n(t) = fx[0]+(t-x[0])(fx[1]+...+(t-x[n-2])(fx[n-2]+(t-x[n-1])(fx[n-1]+(t-x[n])(fx[n])))...)$$

Pseudokod przedstawiony poniżej korzysta bezpośrednio z wzorów przedstawiowych wcześniej, zaczynając od lewej strony równania przesuwa się w jej prawą stronę. Zmienną nt po i iteracjach pętli for (należy zwrócić uwage, że $nt = fx[1]$ występuje poza pętlą, tak więc jest to tak naprawdę stan po $i + 1$ iteracjach) można w uproszczeniu zapisać jako:

$$nt_i = fx[0] + (t - x[0])fx[1] + (t - x[0])(t - x[1])fx[2] + ... + (t - x[0])(t - x[1])...(t - x[i])fx[i + 1]$$

```
function warNewton(x, fx, t)
    p ← 1
    nt ← fx[1]
    for i = 2 to length(n) do
        p ← p * (t − x[i − 1])
        nt ← nt + fx[i] * p
    end for
    return nt
end function
```

2.3 Wyniki

Funkcja została poddana dwóm testom, które wykorzystywały rownież funkcje zaimplementowaną w zadaniu poprzednim.Przy zadanych węzłach oraz wartości funkcji w węzłach funkcja *ilorazyRoznicowe* została wykorzystana do policzenia wektora fx zawierającego ilorazy różnicowe, które natomiast zostały wykorzystane do przetestowania funkcji *warNewton*. Wartość wielomianu interpolacyjnego zadanego węzłami $[3.0, 1.0, 5.0, 6.0]$ oraz wartościami w węzłach $[1.0, -3.0, 2.0, 4.0]$ w punktach $t_1 = 3$ oraz $t_2 = 5$ wynosi kolejno 1.0 oraz 2.0 co odpowiada otrzymanym wynikom uzyskanym przeze mnie przy użyciu innych dostępnych narzędzi.

3 Zadanie 3

3.1 Opis problemu

Celem zadania była implementacja funkcji, która policzy współczynniki wielomianu interpolacyjnego w postaci Newtona. Sygnatura funkcji powinna byc postaci:

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

gdzie x to wektor zawierający węzly,a fx to wektor zawierający iloraz różnicowe (taki że $fx[i] = f[x_0, .., x_i]$). Funkcja powinna zwracać taki wektor zawierający współczynniki wielomianu w postaci naturalnej, że $a[i] = a_i$.

3.2 Opis rozwiązania

Przed zaznajomieniem się rozwiązaniem warto zapoznać się z twierdzeniem Bézouta oraz schematem Hornera. Niech p będzie wielomianem, a $p(x)$ funkcją wielomianową odpowiadająca temu wielomianowi. Z twierdzenia Bézouta wynika, że wartość funkcji wielomianowej $p(x)$ w punkcie a jest równy reszcie z dzielenia wielomianu p przez dwumian $x - a$ oraz, że jeśli $p(x) = q(x)(x - a) + r$ to $p(a) = r$. Schemat Hornera jest sposobem na obliczanie wartości wielomianu w danym punkcie (do czego został wykorzystany w zadaniu numer 2) jak również jest algorytmem dzielenia wielomianu przez dowolny dwumian.
Posiadając wielomian postaci:

$$W(x) = a_0 + (x - x_0)(a_1 + ... + (x - x_{n-1})(a_{n-1} + (x - x_n)(a_n))...)$$

możemy go podzielić przez dwumian $x - c$ z wykorzystaniem schematu Hornera zaczynając od liczenia współczynników b_i (gdzie $i \in \{0..., deg(W)\}$) w taki sposób:

$$\begin{gathered} b_n = a_n \\ b_{n-1} = a_{n-1} + (c - x_n) * b_n \ b_{n-2} = a_{n-2} + (c - x_{n-1}) * b_{n-1} \\ \vdots \\ b_1 = a_1 + (c - x_2) * b_2 \\ b_0 = a_0 + (c - x_1) * b_1 \end{gathered}$$

Po uzyskaniu współczynników b_i dla $i \in \{0, ..., deg(W)\}$ możemy przedstawic wielomian W w postaci $W(x) = (x - c)W'(x) + r$, gdzie $deg(r) = 0$:

$$W(x) = (x - c)(b_1 + (x - x_0)(b_2 + ... + (x - x_{n-2})(b_{n-1} + (x - x_{n-1})(b_n))...)) + b_0$$

Algorytm obliczania współczynników wielomianu który początkowo jest w postaci Newtona bazuje właśnie na algorytmie przedstawionym powyżej. Otrzymany wielomian $N_n(x)$ na wejściu jest dzielony przez dwumian x i na podstawie wspólczynników b_i zostaje przedstawiony w postaci $N_n(x) = xN_{n_1}(x) + b_0$, gdzie b_0 jest wyrazem wolnym wielomianu N_n przedstawionego w postaci kanonicznej. Wielomian $N_{n_1}(x)$ jest postaci:

$$N_{n_1}(x) = b_1 + (x - x_0)(b_2 + ... + (x - x_{n-1})(b_{n-2} + (x - x_{n-1})(b_n))...)$$

i jest on stopnia $deg(N_n) - 1$. Wykorzystując po raz kolejny schemat Hornera możemy przedstawić ten wielomian w postaci:

$$\begin{gathered} N_{n_1}(x) = xN_{n_2}(x) + b'_1 \\ \text{, gdzie } N_{n_2}(x) = (x - x_0)(b'_2 + ... + (x - x_{n-3})(b'_{n-1} + (x - x_{n-2})(b'_n))...) \end{gathered}$$

Reszta z dzielenia N_{n_1} przez x jest tym razem współczynnikiem wielomianu w postaci kanonicznej przy x^1. Po i iteracjach tego algorytmu mamy

$$\begin{gathered} N_{n_i} = (x - x_0)(b^i_i + ... + (x - x_{n-i-1})(b^i_{n-1} + (x - x_{n-i})(b^i_n))...) \text{ , gdzie} \\ b^i_a \text{ to } b_a \text{ w i-tej iteracji } (b^1_a = b'_a, b^2_a = b''_a \text{ itd.}) \end{gathered}$$

a jako reszte z dzielenia wielomianu N_{n_2} przez dwumian x otrzymujemy współczynnik wielomianu interpolacyjnego w postaci kanonicznej przy x^{i+1}. Po użyciu $deg(N_n)$ razy schematu Hornera otrzymujemy zbiór o zawierający wszystkie współczynniki wielomianu interpolacyjnego w postaci kanonicznej. W pseudokodzie przed wykonaniem się drugiej pętli for tablica b przechowuje wspólczynniki z poprzedniej iteracji. Dzięki liczeniu "aktualnych" wartości współczynników od prawej strony, możemy pracować na jednej jednowymiarowej tablicy. Przed i-tą iteracją zewnętrznej pętli for współczynniki $b[1], ..., b[i-1]$ odpowiadają $a_0, ..., a_{i-2}$ wielomianu interpolacyjnego w postaci kanonicznej, natomiast $b[i], ..., b[n]$ są współczynnikami wielomianu $N_{n,i}(x)$.

```
function NATURALNA(x, fx)
    n ← length(fx)
    for i = 1 to n do
        b[i] ← fx[i]
    end for
    for i = 1 to n do
        for j = n − 1 downto i do
            b[j] ← b[j] − x[j − i + 1] ∗ b[j + 1]
        end for
    end for
    return b
end function
```

3.3 Wyniki

W celu zweryfikowania poprawności algorytmu został wykonany test obliczający wspólczynniki wielomianu interpolującego funkcje o węzłach $[-1.0, 0.0, 1.0, 2.0]$ oraz wartościach w węzłach $[3.0, -7.0, 8.0, -6.0]$ otrzymana wartość $[4.0, 7.0, 8.0, -6.0]$ odpowiada końcowemu wielomianowi postaci $-4 + 7x + 8x^2 - 6x^3$ i jest ona prawidłowa.

4 Zadanie 4

4.1 Opis problemu

Celem zadania była implementacja funkcji, która zinterpoluje zadaną anonimową funkcje $f(x)$ w przedziale $[a.b]$ za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona. Funkcja powinna rysować wielomian interpolacyjny, jak również interpolowaną funkcje. Sygnatura funkcji powinna być postaci:

$$function\ rysujNnfx(f, a :: Float64, b :: Float64, b :: Int)$$

gdzie f to funkcja zadana jako anonimowa funkcja, a oraz b to granice przedziału interpolacji, a n to stopień wielomianu interpolacyjnego. Funkcja $rysujNnfx$ powinno korzystać jedynie z $ilorazyRoznicowe$ oraz $warNewton$, a nie powinna wyznaczać wielomianu interpolacyjnego w jawnej postaci.

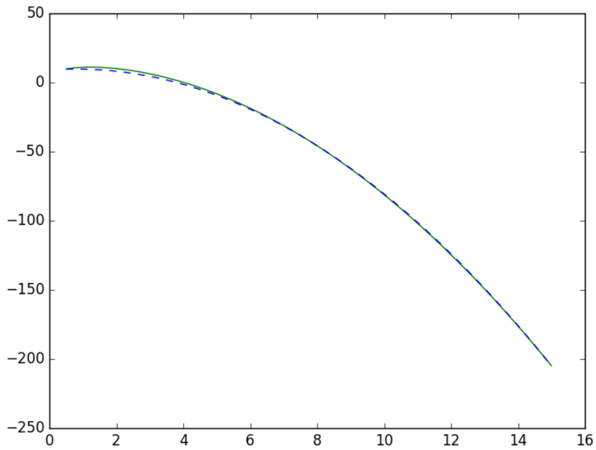
4.2 Opis rozwiązania

W celu uzyskania ilorazów różnicowych na samym początku liczone są węzły interpolacyjne oraz wartości funkcji w tych węzłach. Jako, że stopień wielomianu interpolacyjnego jest zadany poprzez parametr funkcji (i jest on równy n) obliczanych jest tylko $n + 1$ węzłow, które są od siebie odległe o taką samą wartość (tzn. $x[i + 1] - x[i] = x[j + 1] - x[j]$) oraz $n + 1$ wartości funkcji w węzłach. Wykorzystując funkcje zaimplementowaną wcześniej uzyskujemy ilorazy różnicowe wielomianu w postaci Newtona N_n. Aby skonstruować wykresy funkcji oraz wielomianu interpolacyjnego potrzebuje wartości funkcji oraz wielomianu w punktach. Aby to uzyskać korzystamy z $f(arg)$ oraz $warNewton(x, fx, arg)$, gdzie arg to punkt, w którym liczymy wartość. Ponizej znajduje się pseudokod zaimplementowanej funkcji.

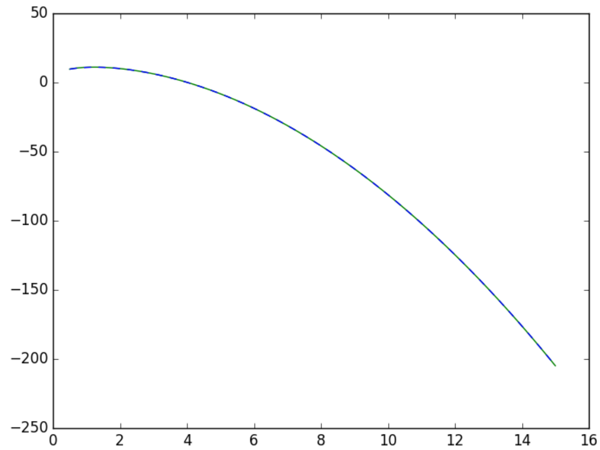
function RYSUJNNFX(f,a,b,n)
 $h \leftarrow (b - a)/n$
 for $i = 0$ **to** n **do**
  $x[i + 1] \leftarrow a + i * h$
  $y[i + 1] \leftarrow f(x[i + 1])$
 end for
 $fx \leftarrow ilorazyRoznicowe(x, y)$
 $n_{graph} \leftarrow \lfloor(b - a)\rfloor * 20$
 $h_{graph} \leftarrow (b - a)/n_{graph}$
 for $i = 0$ **to** n_{graph} **do**
  $arg \leftarrow a + i * h_{graph}$
  $f\_x_{graph}[i + 1] \leftarrow f(arg)$
  $w\_x_{graph}[i + 1] \leftarrow warNewton(x, fx, arg)$
 end for
 $plot(f\_x_{graph})$
 $plot(w\_x_{graph})$
end function

4.3 Wyniki

Poprawność działania procedury przedstawiającej zinterpolowaną funkcje na wykresach została przetestowana na funkcji $f(x) = 12 + ln(x) * 3 - x^2$ na przedziale $[0.5, 15.0]$, gdzie stopień wielomianu interpolującego n należał do $\{2, 10\}$.



Rysunek 1: przybliżenie funkcji f wielomianem stopnia 2



Rysunek 2: przybliżenie funkcji f wielomianem stopnia 10

5 Zadanie 5

5.1 Opis problemu

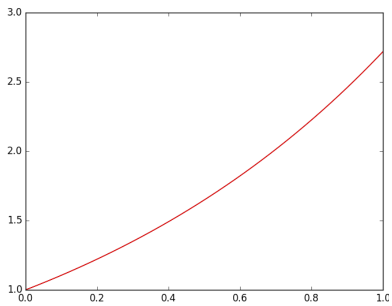
Celem zadania było przetestowanie zaimplementowanej wcześniej funkcji $rysunNnfx$ dla dwóch wskazanych przykładów $f_1(x) = e^x$ dla przedziału $[0, 1]$ i stopniu wielomianu $n \in \{5, 10, 15\}$ oraz $f_2(x) = x^2 sin(x)$ dla przedziału $[-1, 1]$ i stopnia wielomianu $n \in \{5, 10, 15\}$.

5.2 Opis rozwiązania

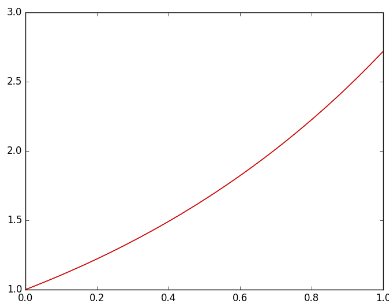
Dla każdej z dwóch wskazanych funkcji procedura $rysujNnfx$ została wywołana 4 razy. Pierwsze trzy wywołania prezentowały graficznie wielomiany o wskazanych stopniach na oddzielnych wykresach, natomiast czwarte wywołanie prezentowało wszystkie trzy wielomiany interpolacyjne wraz z funkcją, którą przybliżaliśmy na jednym wykresie.

5.3 Wyniki

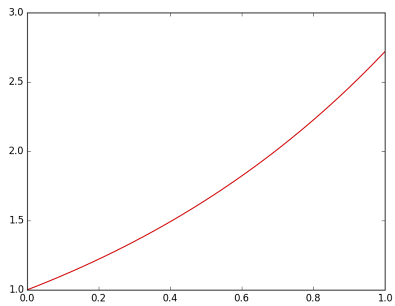
Trzy wykresy reprezntujące interpolacje tej samej funkcji wielomianem o róznych stopniach wyglądają wrecz identycznie. Dopiero przy odpowiednim przybliżeniu widoczne są odchylenia wielomianu interpolującego od funkcji, którą interpolujemy.



Rysunek 3: przybliżenie funkcji $f_1(x) = e^x$ wielomianem stopnia 5

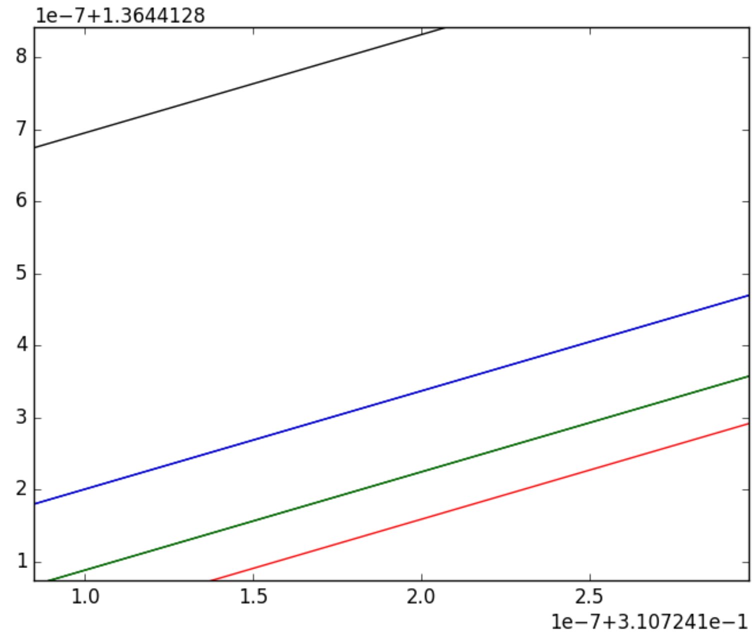


Rysunek 4: przybliżenie funkcji $f_1(x) = e^x$ wielomianem stopnia 10



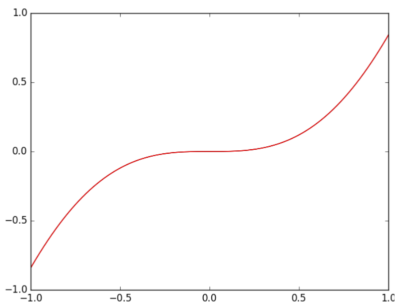
Rysunek 5: przybliżenie funkcji $f_1(x) = e^x$ wielomianem stopnia 15

Wielomany interpolujące zostały porównae na wykresie poniżej. Niebieskim kolejem został oznaczony wielomian o stopniu $n = 10$, zielony $n = 15$, natomiast czerwony $n = 5$. Kolorem czarnym została oznaczona funkcja, którą przybliżamy.

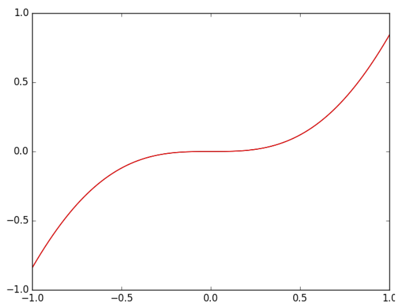


Rysunek 6: interpolacja funkcji $f_1(x) = e^x$

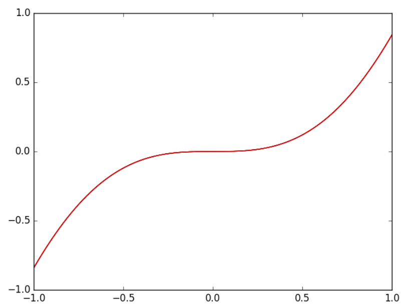
Tak samo jak w przypadku f_1 na początku przedstawione zostały trzy oddzielne wykresy, a dopiero następnie wykres porównujący przybliżenia. Podobnie jak w przypadku f_1 niebieskim kolejem został oznaczony wielomian o stopniu $n = 10$, zielony $n = 15$, natomiast czerwony $n = 5$. Kolorem czarnym została oznaczona funkcja, którą przybliżamy.



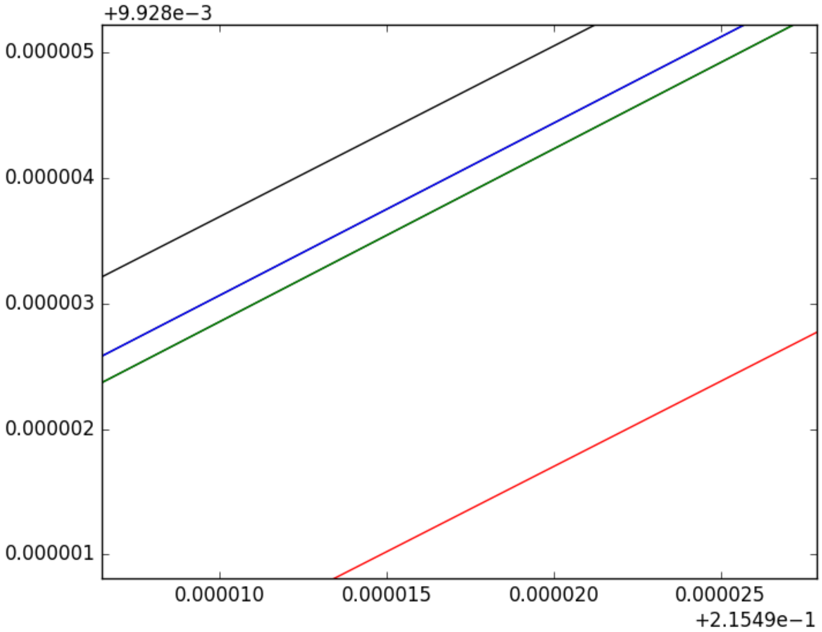
Rysunek 7: przybliżenie funkcji $f_2(x) = x^2 sin(x)$ wielomianem stopnia 5



Rysunek 8: przybliżenie funkcji $f_2(x) = x^2 sin(x)$ wielomianem stopnia 10



Rysunek 9: przybliżenie funkcji $f_2(x) = x^2 sin(x)$ wielomianem stopnia 15



Rysunek 10: interpolacja funkcji $f_2(x) = x^2 sin(x)$

5.4 Wnioski

Na wykresie, na którym zostały zawarte wszystkie 3 wielomiany interpolujące tę samą funkcje można zauważyć, że stopień wielomianu ma bezpośredni wpływ na jego dokładnośc względem pierwotnej funckji. Mimo tego, jeżeli stopień wielomianu będzie za duży otrzymamy mniej dokładne przybliżenie. Widać to wyraźnie w przypadku $n = 10$ oraz $n = 15$ dla f_1 oraz f_2 wielomiany o stopniu równym 10 są bardziej dokładne niz o stopniu 15. Może to być spowodowane zbyt małą dokładnością obliczeń, które były wykonywane w $Float64$.

6 Zadanie 6

6.1 Opis problemu

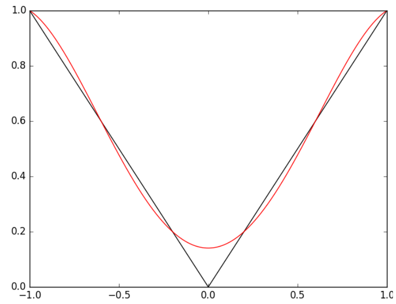
Celem zadania było przetestowanie zaimplementowanej wcześniej funkcji $rysunNnfx$ dla dwóch wskazanych przykładów $f_1(x) = |x|$ dla przedziału $[-1, 1]$ i stopniu wielomianu $n \in \{5, 10, 15\}$ oraz $f_2(x) = \frac{1}{1+x^2}$ dla przedziału $[-5, 5]$ i stopnia wielomianu $n \in \{5, 10, 15\}$.

6.2 Opis rozwiązania

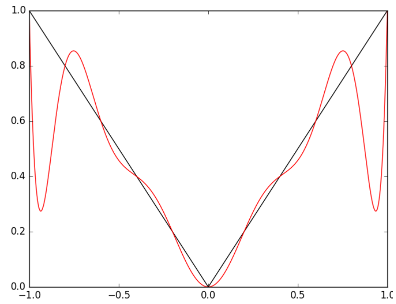
Dla każdej z dwóch wskazanych funkcji procedura $rysujNnfx$ została wywołana 3 razy. Każde wywołanie prezentowało wykres wielomianu interpolacyjnego innego stopnia.

6.3 Wyniki

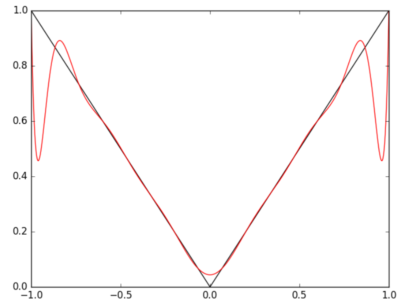
W odróżnieniu od zadania 5 przedstawione zostały jedynie wykresy zawierające tylko jeden wielomian interpolacyjny o danym stopniu, ze względu na fakt, że już na podstawie takich wykresach można zaobserwować różnice w prezentowanych wielomianach, wielomiany interpolacyjne zostały zaznaczone kolorem czerwonym.



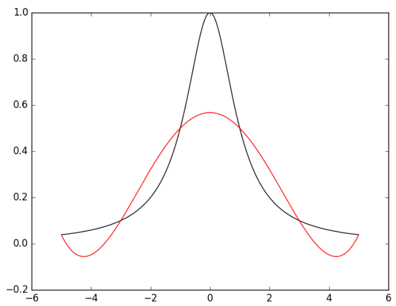
Rysunek 11: przybliżenie funkcji $f_1(x) = |x|$ wielomianem stopnia 5



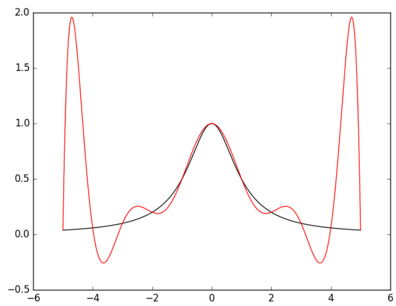
Rysunek 12: przybliżenie funkcji $f_1(x) = |x|$ wielomianem stopnia 10



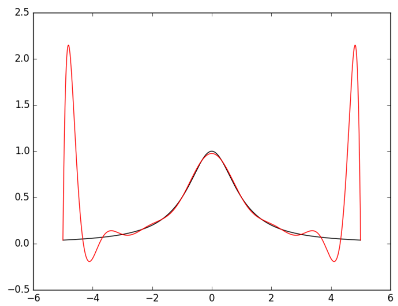
Rysunek 13: przybliżenie funkcji $f_1(x) = |x|$ wielomianem stopnia 15



Rysunek 14: przybliżenie funkcji $f_2(x) = \frac{1}{1+x^2}$ wielomianem stopnia 5



Rysunek 15: przybliżenie funkcji $f_2(x) = \frac{1}{1+x^2}$ wielomianem stopnia 10



Rysunek 16: przybliżenie funkcji $f_2(x) = \frac{1}{1+x^2}$ wielomianem stopnia 15

6.4 Wnioski

Na wykresach przedstawionych w poprzednim podrozdziale można było zaobserwować Efekt Rungeo. Polega on na pogorszeniu się jakości interpolacji wielomianowej, mimo zoeiekszenia liczby jej węzłów (tj. zwiększenia stopnia wielomianu). Początkowo ze wzrostem stopnia wielomianu n przybliżenie poprawia się, jednak po dalszym wzroście n, zaczyna się pogarszać, co jest szczególnie widoczne na końcach przedziałów. Takie zachowanie wielomianu interpolującego jest zjawiskiem typowym dla interpolacji za pomocą wielomianów wysokich stopni przy stałych odległościach węzłów. Występuje również, gdy interpolowana funkcja jest nieciągła lub odbiega znacząco od funkcji gładkeij ($f_1(x) = |x|$). Szczególnie wyraźnie efekt Runego można zaobserowwać dla $f_1(x) = |x|$, gdzie przy wzroście stopnia wielomianu interpolującego wzrasta dokładność przybliżenia blisko punktu $x = 0$, lecz błąd przybliżenia rośnie na granicach przedziału. Wzrost błędu na granicach przedziału jest nieporównywalnie większy od wzrostu dokładności blisko punktu $x = 0$ (tj. środka przedziału). W przypadku $f_2(x) = \frac{1}{1+x^2}$ obserwujemy tą samą sytuacje, z taką różnicą, że interpolacja na granicach już dla stopnia wielomianu $n = 5$ mocno odbiegała od funkcji. Efekt ten można minimalizować np. poprzez zwiekszenie gęstości węzłów interpolacji na krańcach przedziału (przy tym samym stopniu wielomianu interpolującego).