# Obliczenia Naukowe
# Lista 4
# Laboratoria

Piotr Szyma

7 grudnia 2017

# 1 Zadanie 1

## 1.1 Opis problemu

Napisać funkcję obliczającą ilorazy różnicowe zgodnie ze specyfikacją podaną w treści zadania bez użycia tablicy dwuwymiarowej (macierzy).
`function ilorazyRoznicowe (x::VectorFloat64, f::VectorFloat64)`

## 1.2 Analiza

Implementacja metody wyliczania ilorazów różnicowych to pierwszy krok do stworzenia algorytmu aproksymującego funkcję metodą Newtona. Podstawą do zbudowania algorymu jest następujący wzór rekurencyjny:

$$f([x_k, x_{k+1}, \ldots, x_{k+m}]) = \frac{f[x_{k+1}, x_{k+2}, \ldots, x_{k+m}] - f[x_k, x_{k+1}, \ldots, x_{k+m-1}]}{x_{k+m} - x_k}$$

który rozwijamy, aż do postaci ilorazu pojedynczego węzła, który jest nam znany i wynosi $f[x_0] = f(x_0)$. Oto tablica trójkątna obrazująca wartości ilorazów potrzebne do wyliczenia interesującego nas ilorazu - w tym wypadku $f[x_0, x_1, x_2, x_3]$. Do wyliczenia ilorazu w komórce $(x, y)$ potrzebujemy wartości z komórek $(x-1, y)$ oraz $(x-1, y-1)$. Wartości $f[x_k] = f(x_k)$ znamy od początku, więc - idąc od lewej strony - jesteśmy w stanie wypełnić całą tabelę trójkątną.

$$\begin{array}{c|cccc}
x_0 & f[x_0] & f[x_0x_1] & f[x_0x_1x_2] & f[x_0x_1x_2x_3] \\
x_1 & f[x_1] & f[x_1x_2] & f[x_1x_2x_3] & \\
x_2 & f[x_2] & f[x_2x_3] & & \\
x_3 & f[x_3] & & &
\end{array}$$

Wielomian, jaki można odczytać z powyższej tabeli to:

$$w(x) = f[x_0] + f[x_0x_1](x - x_0) + f[x_0x_1x_2](x - x_0)(x - x_1) + f[x_0x_1x_2x_3](x - x_0)(x - x_1)(x - x_2)$$

Specyfikacja funkcji, jaką mieliśmy zaimplementować w zadaniu wymagała, aby wartość zwracana była wektorem ilorazów. Zestawiając to z przykładem macierzy trójkątnej i odczytując z niej odpowiednie komórki, funkcja powina zwrócić wektor $(f[x_0], f[x_0x_1], f[x_0x_1x_2], f[x_0x_1x_2x_3])$ - znajdujący się w pierwszym wierszu macierzy.

## 1.3 Implementacja

Założenie zadania wymagało, aby wyliczenia ilorazów nie używać tablicy dwuwymiarowej. To założenie wyeliminowało możliwość implementacji w postaci iteracyjnej, ani nie pozwoliło na użycie tablicy będącej *cachem* już obliczonych wartości. W mojej implementacji posłużyłem się rekurencją, która oddaje matematyczny wzór rekurencyjny przytoczony w części dotyczacej analizy zadania.

Moja implementacja algorytmu w języku Julia znajduje się w module `MyModule` znajdującym się w pliku załączonym do tego sprawozdania.

# 2 Zadanie 2

## 2.1 Opis problemu

Celem zadania było stworzenie funkcji obliczającej wartość wielomianu interpolacyjnego stopnia $n$ w postaci Newtona $N_n(x)$ w punkcie $x = t$ za pomocą uogólninonego algorytmu Hornera w czasie $O(n)$. (implementacja zadania 8 z listy nr 4 z ćwiczeń)

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

## 2.2 Analiza

Na zajęciach pokazaliśmy, że poniższa formuła (uogólniony algorytm Hornera) pozwala na wyliczenie wartości wielomianu interpolacyjnego Newtona:

$$\begin{aligned} w_n(x) &= f[x_0, x_1, \ldots, x_n] \\ w_k(x) &= f[x_0, x_1, \ldots, x_k] + (x - x_k)w_{k+1} \quad (k = n-1, \ldots, 0) \\ N_n(x) &= w_0(x) \end{aligned}$$

Rzeczywiście, starając się wyliczyć interesującą nas wartość $N_n(x)$ rozwijamy rekurencyjny wzór:

$$\begin{aligned} N_n(x) &= f[x_0, x_1, \ldots, x_k] + (x - x_k)(f[x_0, x_1, \ldots, x_{k+1}] + (x - x_{k+1})w_{k+2}) \\ &= f[x_0, x_1, \ldots, x_k] + (x - x_k)(f[x_0, x_1, \ldots, x_{k+1}] + (x - x_{k+1})(f[x_0, x_1, \ldots, x_{k+2}] + (x - x_{k+2})w_{k+3})) \\ &= \ldots \\ &= f[x_0] + f[x_0, x_1](x - x_1) + \ldots + f[x_0, \ldots, x_n](x - x_1)\ldots(x - x_n) \end{aligned}$$

który zakończy się w momencie, gdy dojdziemy do $k = n$, czyli dla $w_n(x)$, za $x$ podstawiając nasz argument.

## 2.3 Implementacja

Warunkiem postawionym w treści zadania było stworzenie algorytmu działającego w czasie $O(n)$, dostając na wstępie wektor węzłów `x` oraz wektor ilorazów różnicowych `fx`. Po dokonaniu analizy wyżej wymienionego wzoru, udało mi się stworzyć algorytm, który na wstępie generuje tablicę `W` długości $n$ i wypełnia jej ostatnią komórkę wartością $\mathtt{W}[\mathtt{n}] = f[x_0, x_1, \ldots, x_n]$ (którą bierze z wektora `fx`). W kolejnych krokach przesuwam się do poprzednich komórek, wypełniając je wg. schematu $\mathtt{W}[\mathtt{k}] = f[x_0, x_1, \ldots, x_k] + (x - x_k) * \mathtt{W}[\mathtt{k} + 1]$. Do wypełnienia całej tabeli `W` potrzeba jednego przejścia pętli - co gwarantuje liniową złożoność algorytmu.

Moja implementacja algorytmu w języku Julia znajduje się w module `MyModule` znajdującym się w pliku załączonym do tego sprawozdania.

# 3 Zadanie 3

## 3.1 Opis problemu

W tym zadaniu należało dla zadanych współczynników wielomianu interpolacyjnego w postaci Newtona $c_0 = f[x_0], c_1 = f[x_0, x_1], c_2 = f[x_0, x_1, x_2], \ldots, c_n = f[x_0, \ldots, x_n]$ (ilorazy różnicowe) oraz węzłów $x_0, x_1, \ldots, x_n$ napisać funkcję obliczającą w czasie $O(n^2)$ współczynniki $a_0, \ldots, a_n$ jego postaci naturalnej, tzn. $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

## 3.2 Analiza

To zadanie bazowało na wynikach zadania 9 z listy 4 z ćwiczeń, gdzie pokazaliśmy, że mając wielomian w postaci Newtona, jesteśmy w stanie, przy pomocy wielomianów pomocniczych z metody Hornera, wyznaczyć współczynniki postaci naturalnej.

## 3.3 Implementacja

Algorytm rozpoczynamy od stworzenia tablicy `A[n]`, w której będziemy trzymać kolejne współczynniki. Zaczynamy od końca, do komórki `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 kolejnym kroku współczynniki z tego wielomianu pomocniczego zestawiamy z dotychczas wyliczonymi współczynnikami. Te kroki powtarzamy aż do momentu zejścia do współczynnika $a_0$. Po przejściu całego algorytmu otrzymujemy tabelę `A[n]` będącą wektorem współczynników wielomianu postaci normalnej. W każdej z $n$ iteracji algorytmu musimy dokonać zestawienia do $n$ współczynników. Złożoność algorytmu wynosi więc $O(n) * O(n)$, tj. $O(n^2)$.

Moja implementacja algorytmu w języku Julia znajduje się w module `MyModule` znajdującym się w pliku załączonym do tego sprawozdania.

# 4 Zadanie 4

## 4.1 Opis problemu

W tym zadaniu należało napisać funkcję, która zinterpoluje zadaną funkcję $f(x)$ w przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego stopnia $n$ w postaci Newtona, a następnie wygeneruje wielomian interpolacyjny i interpolowaną funkcję. (np. przy pomocy pakietu `Plots`, `PyPlot` lub `Gadfly`). Do interpolacji należało użyć węzłów równoodległych, tj. $x_k = a + kh, k = \frac{b-a}{n}, k = 0, 1, \ldots, n$.

## 4.2 Analiza

W celu wykonania interpolacji wykonałem następujące kroki:

1. Wygenerowałem wektor węzłów $\mathtt{W}$
2. Wyliczyłem wartości funkcji w węzłach i zapisałem do tablicy $\mathtt{T_{wynik}}$
3. Używając własnoręcznie zaimplementowanej funkcji $\mathtt{ilorazyRoznicowe}(\mathtt{W}, \mathtt{T_{wynik}})$ wygenerowałem wektor ilorazów $\mathtt{I_{ilorazy}}$.
4. Wykorzystując $\mathtt{W}$ oraz $\mathtt{I_{ilorazy}}$ przy pomocy $\mathtt{warNewton}(\mathtt{W}, \mathtt{I_{ilorazy}})$ wygenerowałem tablicę $\mathtt{N_{wynik}}$ wartości funkcji w zakresie, w którym zostanie wygenerowany wykres
5. Biorąc $\mathtt{N_{wynik}}$ wygenerowałem wykres wielomianu interpolacyjnego
6. Powyższy wykres zestawiłem z rzeczywistym wykresem interpolowanej funkcji

## 4.3 Implementacja

Moja implementacja algorytmu w języku Julia znajduje się w module `MyModule` znajdującym się w pliku załączonym do tego sprawozdania.

# 5 Zadanie 5

## 5.1 Opis problemu

Zadanie polegało na wygenerowaniu wykresów wielomianów interpolowanych z poniższych funkcji za pomocą wcześniej zaimplementowanych metod.

1. $f(x) = e^x, [0, 1], n = 5, 10, 15$
2. $g(x) = x^2 \sin(x), [-1, 1], n = 5, 10, 15$

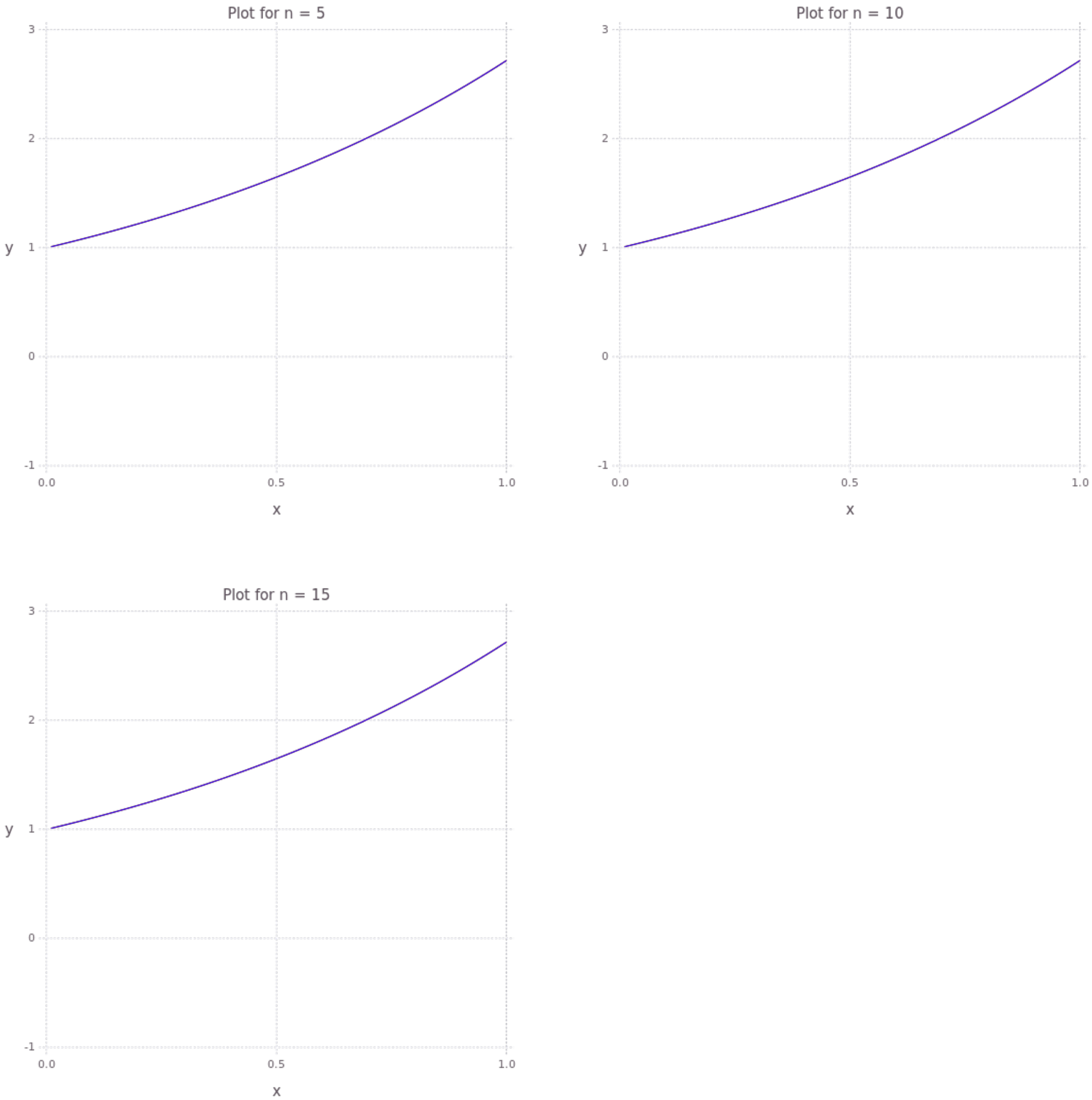
## 5.2 Rozwiązanie

Na poniższych rysunkach przedstawiłem wykresy dla funkcji $f(x)$ oraz $g(x)$ dla poszczególych $n = 5, 10, 15$

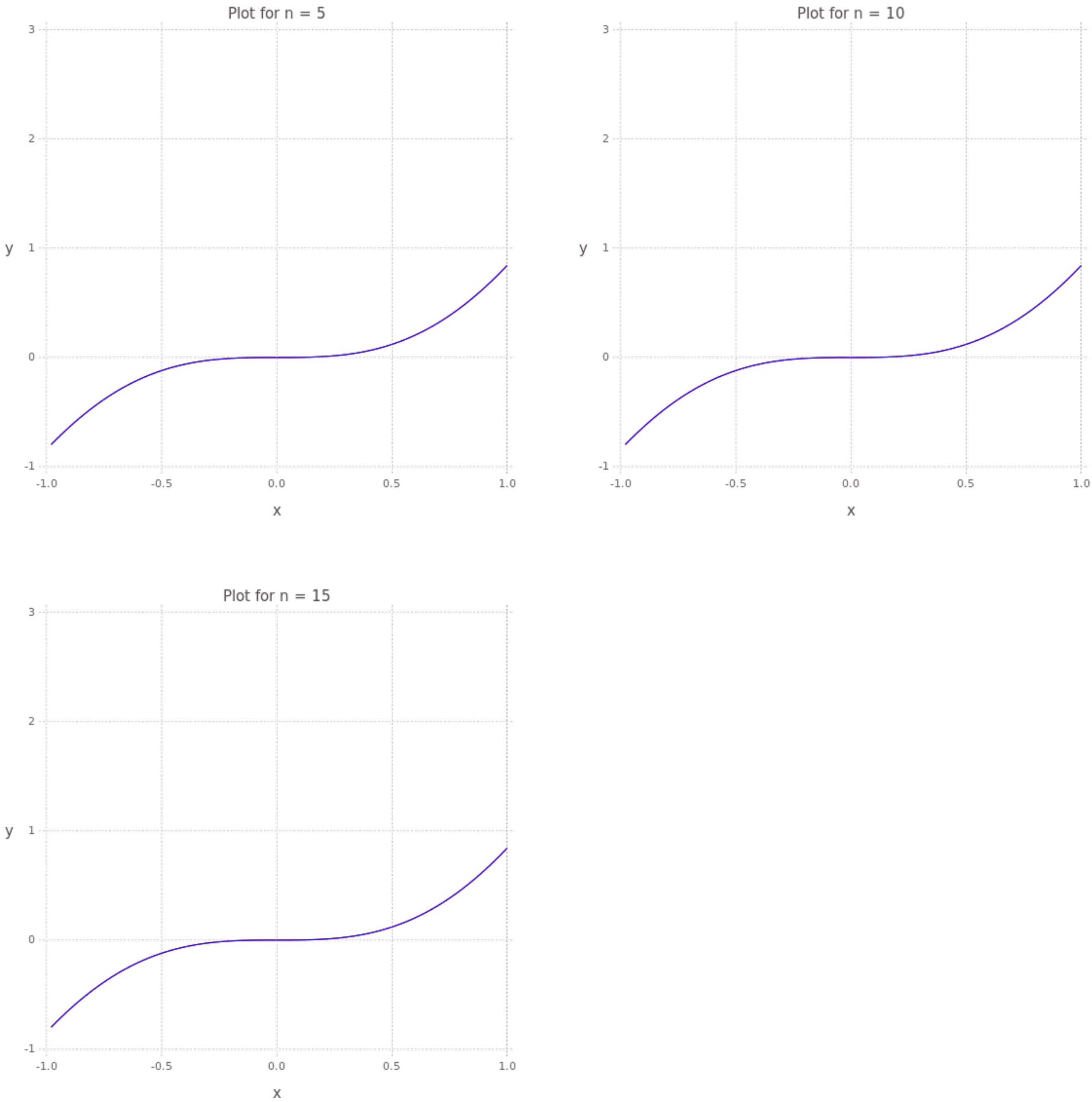
## 5.3 Analiza wyników

W przypadku przykładów z zadania piątego, wykresy wielomianów będących interpolacją funkcji pokryły się z wykresami tych funkcji. Interpolacja poprawnie - z pewnym małym błędem - odwzorowała zadane funkcje. Funkcje interpolowane nie spełniały żadnych warunków, które mogły powodować efekt Rungego[1], mimo tego, że odległość między poszczególnymi węzłami jest stała i ich ilość nie gęstniała na krańcach przedziału.

[1] Efekt opisany w analizie kolejnego zadania




Rysunek 1: Wykresy dla $f(x) = e^x$




Rysunek 2: Wykresy dla $g(x) = x^2 \sin(x)$

# 6 Zadanie 5

## 6.1 Opis problemu

Zadanie polegało na wygenerowaniu wykresów wielomianów interpolowanych z poniższych funkcji za pomocą wcześniej zaimplementowanych metod.

1. $f(x) = |x|, [-1, 1], n = 5, 10, 15$
2. $g(x) = \frac{1}{1+x^2}, [-5, 5], n = 5, 10, 15$

## 6.2 Rozwiązanie

Na poniższych rysunkach przedstawiłem wykresy dla funkcji $f(x)$ oraz $g(x)$ dla poszczególych $n = 5, 10, 15$

## 6.3 Analiza wyników

W przypadku tego zadania mieliśmy do czynienia z funkcjami, dla których wielomiany interpolacyjne są podatne na tzw. EFEKT RUNGEGO. (pogorszenie jakości interpolacji mimo zwiększenia ilości węzłów) Efekt ten często występuje, gdy:

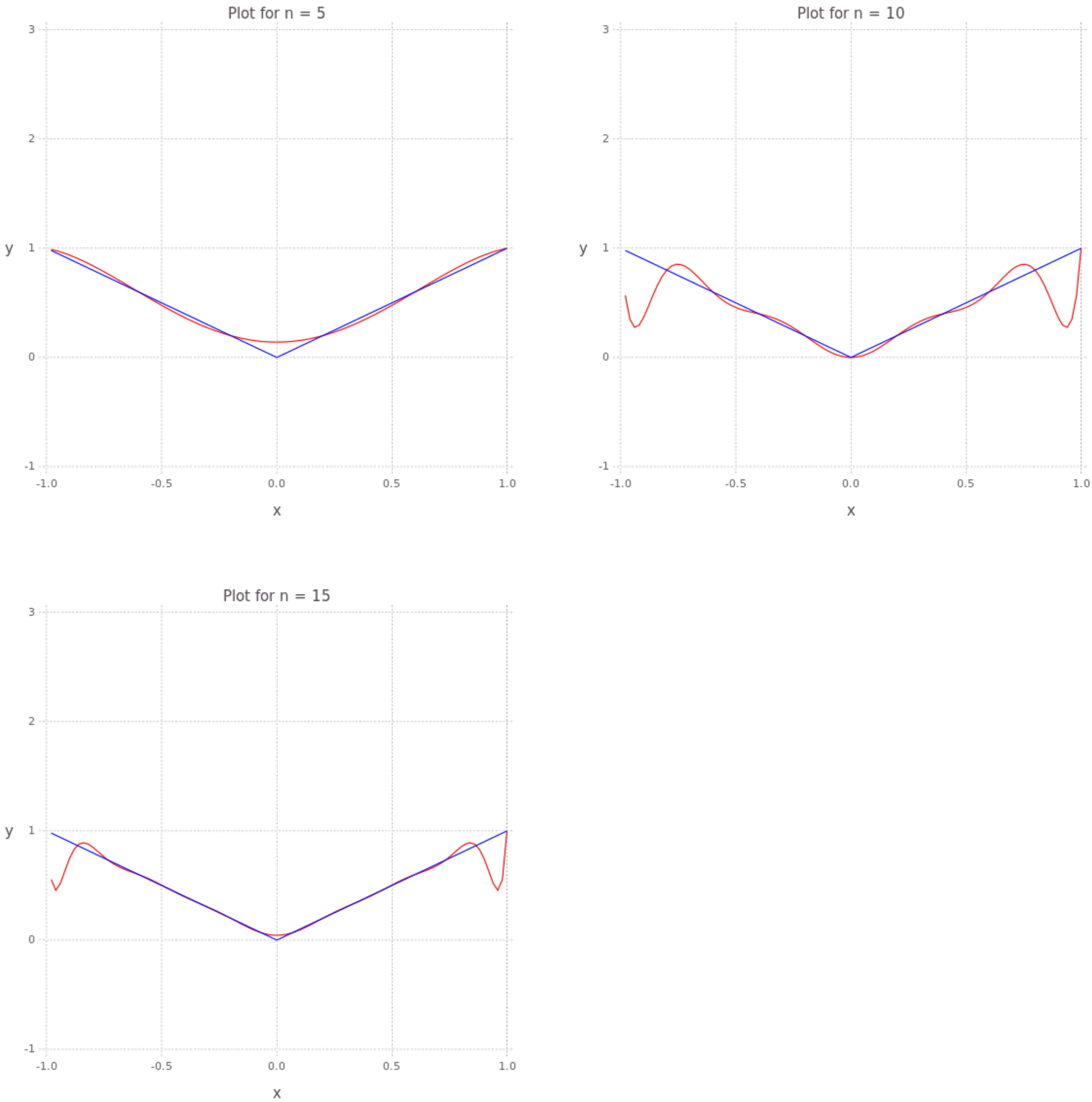
1. interpolowana funkcja jest nieciągła
2. funkcja odbiega znacząco od funkcji gładkiej[2]
3. wielomiany interpolujące mają wysokie stopnie, a odległość między więzłami jest stała

W przypadku funkcji $f(x) = |x|$ dla $n = 5$ interpolacja nie była dokładna w okolicach $x = 0$, lecz bliska $f(x)$ na krańcach przedziału. Po zwiększeniu ilości węzłów do $n = 10$, wygenerowany wielomian był bardziej dokładny w centrum, lecz na krańcach pojawił się wyżej wspomniany efekt Rungego, który nie został zniwelowany dla $n = 15$. Powodem wystąpienia efektu był brak ciągłości funkcji oraz równe odstępy między poszczególnymi węzłami $x_i$.

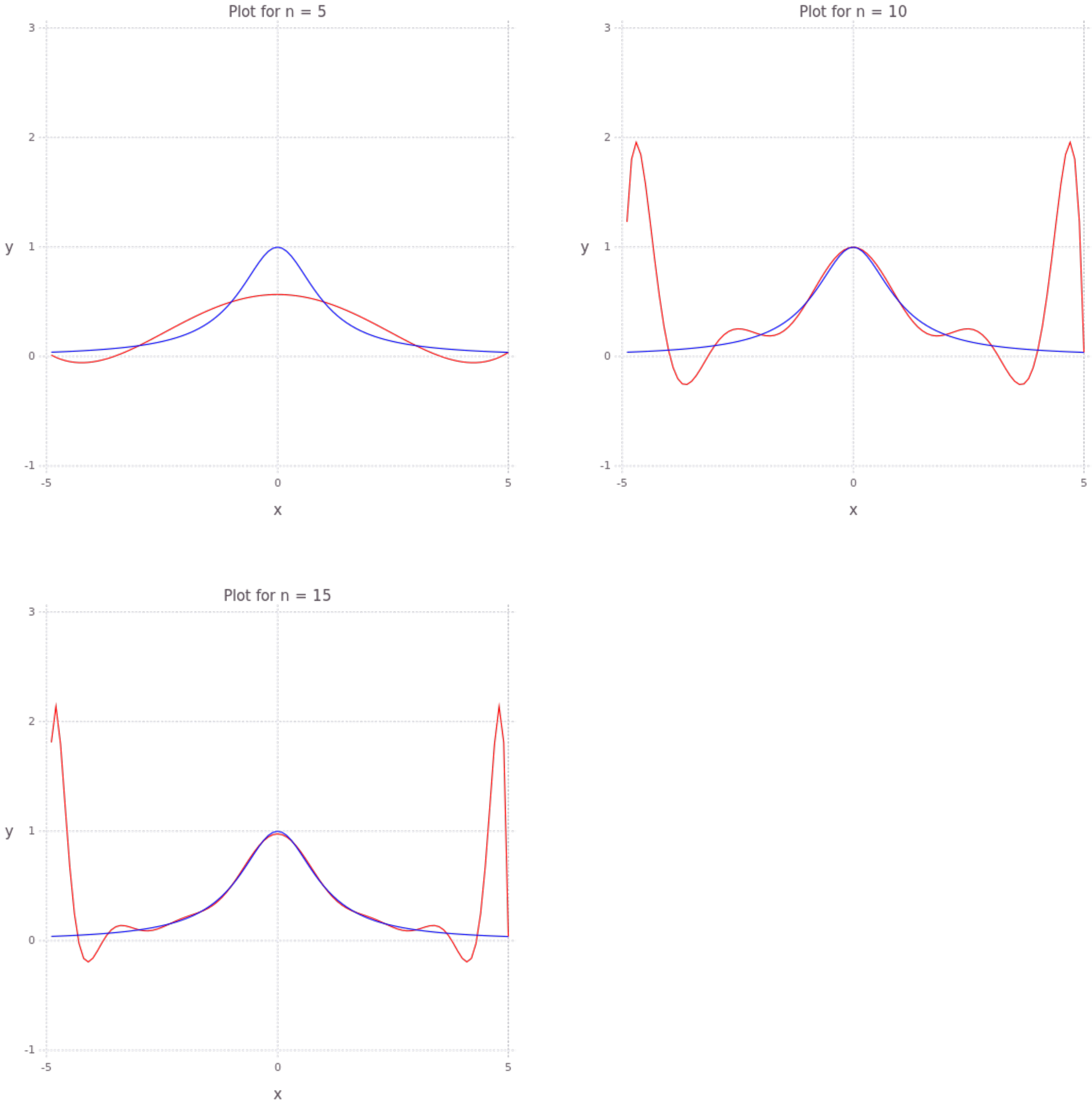
Wielomian będący interpolacją $g(x) = \frac{1}{1+x^2}$ dla $n = 5$ odbiega od rzeczywistego wykresu $g(x)$. Po zwiększeniu ilości węzłów do $n = 10$, interpolacja staje się bliższa wyjściowej funkcji w okolicach $x_i = 0$, lecz na krańcach przedziału pojawia się efekt Rungego. Zwiększając ilość węzłów do $n = 15$ efekt nie zniknął. Powodem wystąpienia efektu Rungego w tym wypadku była nieciągłość funkcji i jednakowa odległość między poszczególnymi węzłami

W celu zniwelowania błędów wynikających z efektu Rungego - dużych oscylacji wielomianu interpolacyjnego na krańcach przedziału - należy gęściej upakować węzły na tych krańcach przedziału interpolacji, np. wykorzystując WIELOMIANY CZYBYSZEWA, których miejsca zerowe zagęszczają się na krańcach.

[2]Funkcja gładka - funkcja ciągła i mająca pochodne wszystkich rzędów




Rysunek 3: Wykresy dla $f(x) = |x|$




Rysunek 4: Wykresy dla $g(x) = \frac{1}{1+x^2}$