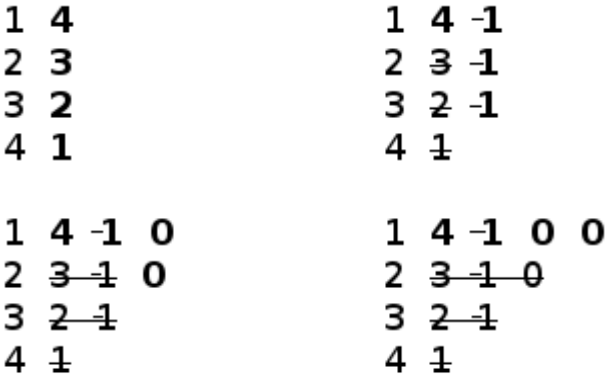
Tomasz Beneś

Sprawozdanie – Lista 4

Zadanie 1.

Implementacja wykorzystuje fakt, że w kolejnych iteracjach obliczania ilorazów różnicowych wykorzystujemy (lub musimy pamiętać jako wynik) tylko N wartości z macierzy N x N. Obrazuje to poniższy rysunek (przekreślone zostały niepotrzebne już liczby):



W związku z tym, zadanie można wykonać wykorzystując jedną tablicę N – elementową.

Rozwiązanie w pseudokodzie:

```
// dane: f[N], x[N]
for i from 2 to N
      for j from N downto i
            f[j] = (f[j]-f[j-1]) / (x[j]-x[j+1-i])
```

Zadanie 2.

Implementacja korzysta z algorytmu Hornera w postaci:

$w_n(x) = f[x_0, x_1, \ldots, x_n]$
$w_k(x) = f[x_0, x_1, \ldots, x_k] + (x - x_k)w_{k+1} \qquad 0 \le k < n$
$N_n(x) = w_0(x).$

Rozwiązanie w pseudokodzie:

```
// dane: x[N] węzły, fx[N] ilorazy, t
nt = fx[N]
for k from N-1 downto 1 // 1 to ostatni indeks czyli w0
      nt = fx[k] + (t - x[k])·nt
```

Mamy jedną pętlę o N-1 krokach, dostęp do x[i] i fx[i] jest stały, więc złożoność to O(N).

Zadanie 3.

Spostrzeżenie: w_n w uogólnionym algorytmie Hornera jest równe a_n. Mając a_n, możemy, bazując na wartościach pośrednich tworzonych przez algorytm Hornera, odzyskiwać kolejne a_{n-1}, bo nie zmieniają one wyższych a_n.

Pseudokod:

Uruchamiamy algorytm Hornera, zapamiętując w macierzy pośrednie wartości.

```
poly[1,1] = 1

for i in 1:n-1
  poly[i+1, 1] = 1
  poly[i+1, 2:i+1] = poly[i, 2:i+1] - x[i] * poly[i, 1:i]
end
```

Później wykorzystujemy je, by odtwarzać kolejne współczynniki.

```
for i in n:-1:1
  a[i] = fx[i]
  for j in 1:n-i
    a[i] = a[i] + fx[i+j] * poly[i+j, j+1]
  end
end
```

Pierwsza część algorytmu wykonuje się w czasie $O(n^2)$, druga również, co w sumie daje żądany czas.

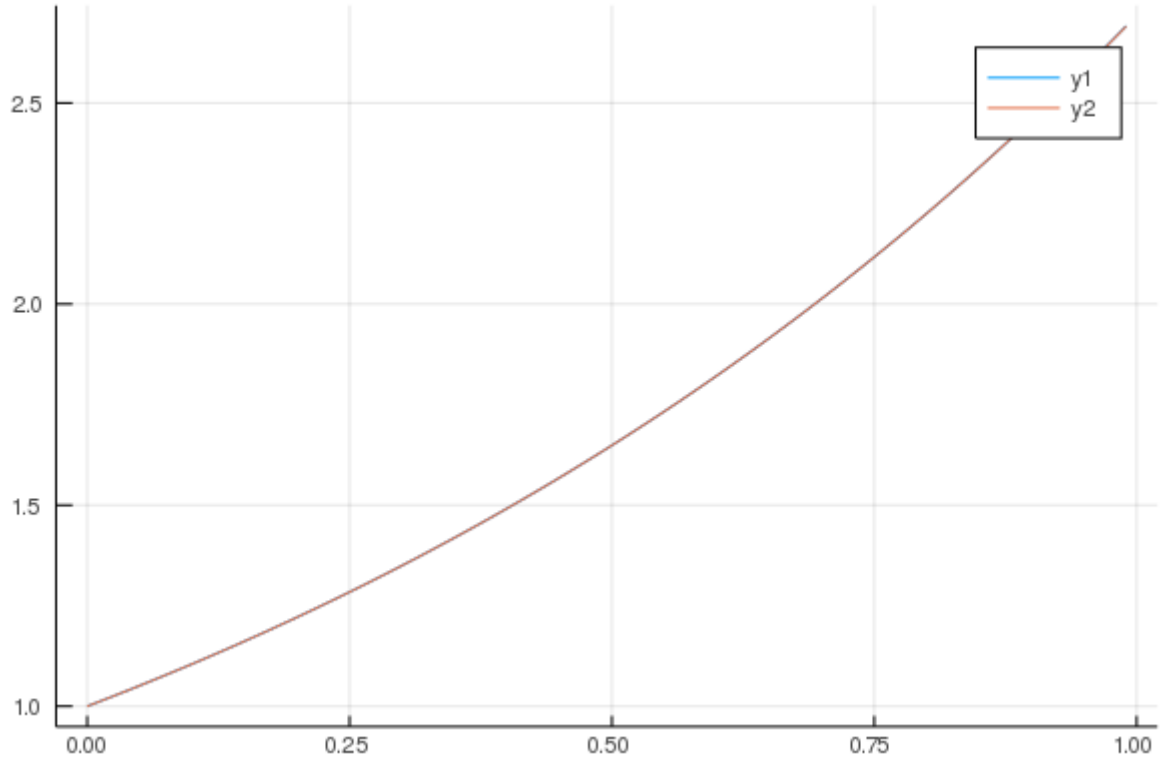
Zadanie 4.

Funkcja wykorzystuje pakiet Plots do rysowania wykresów. Kolejne kroki:

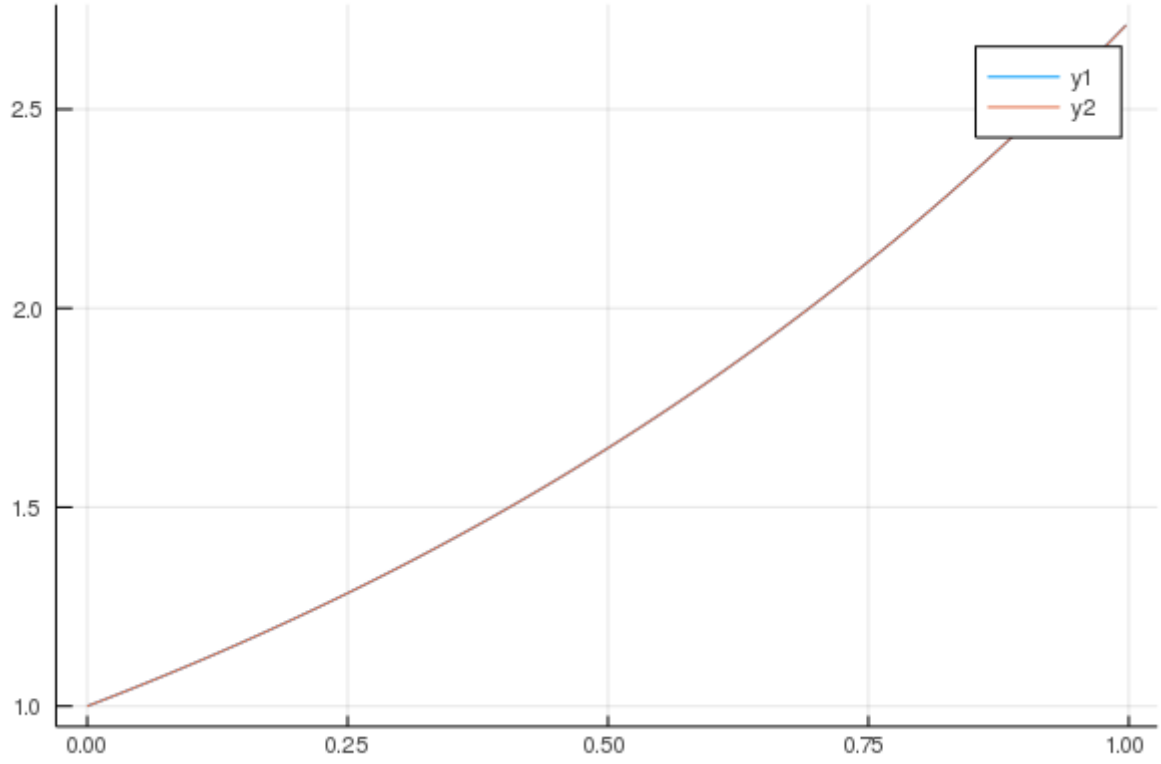
1. Wydzielenie n równoodległych węzłów wykorzystując wzór z zadania
2. Wywołanie `ilorazyRoznicowe`
3. Dla $(2n)^2$ równoodległych punktów w przedziale:
    1. Obliczenie explicite wartości funkcji
    2. Obliczenie wartości interpolowanej wywołaniem `warNewton`
4. Narysowanie obu krzywych

Zadanie 5.

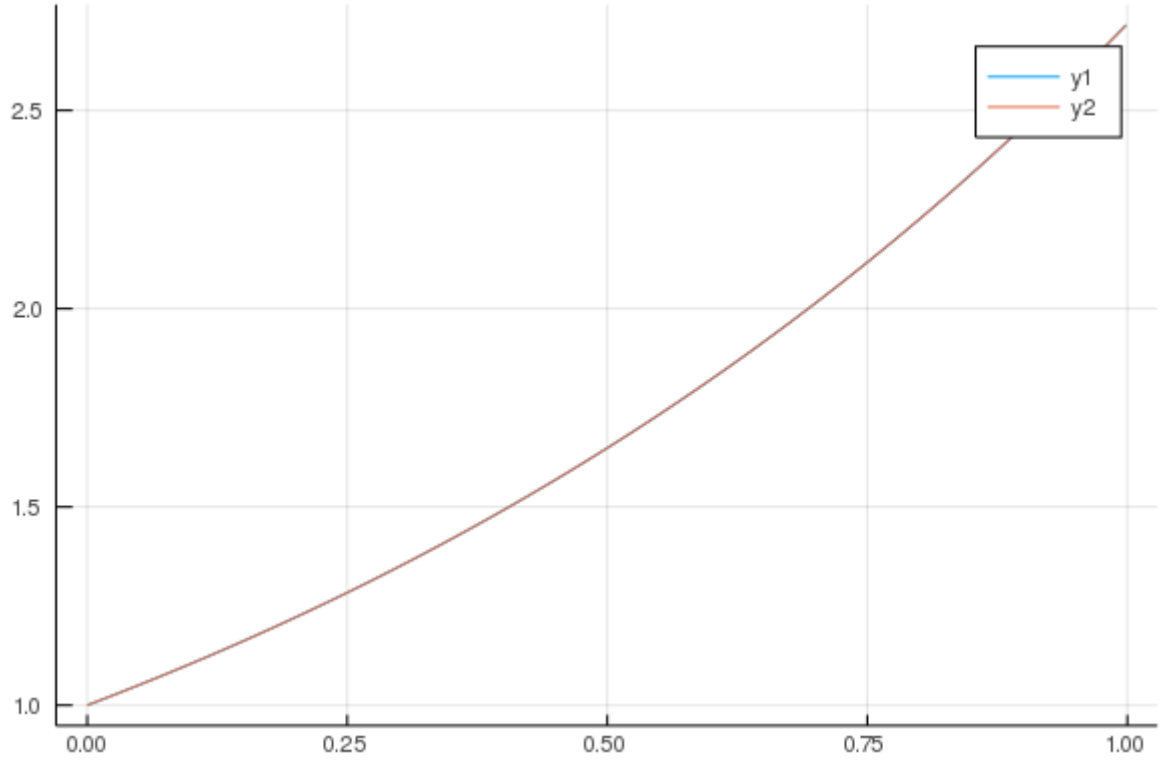
a) $e^x, [0,1]$



n = 5



n = 10

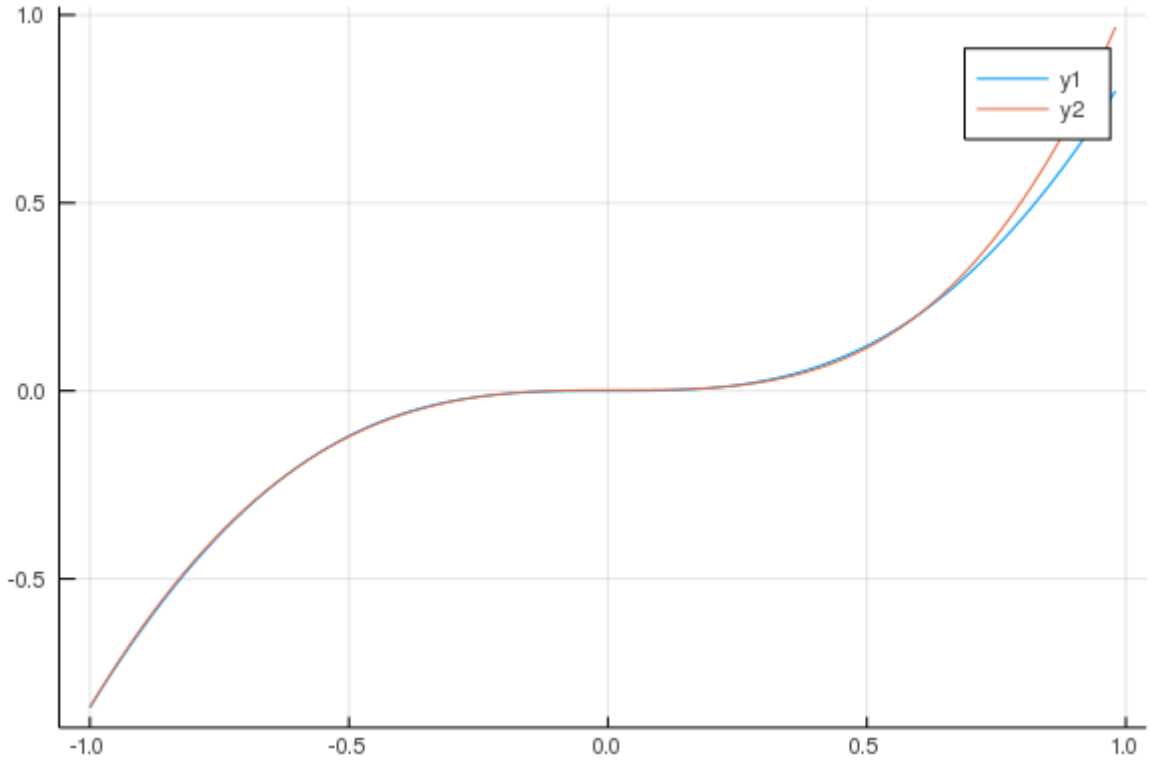


n = 15

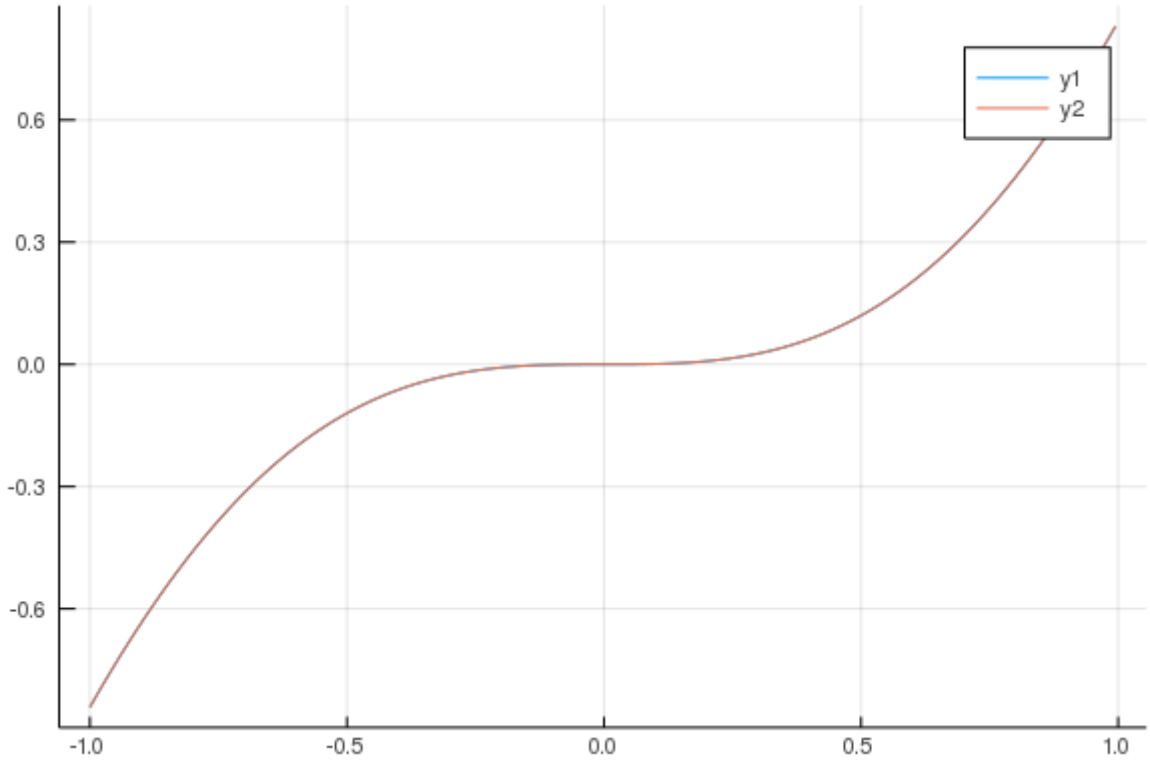
Obserwacje i wnioski:

- Zwiększanie stopnia wielomianu nie poprawia jakości interpolacji
- Stopień równy 5 wystarcza do interpolacji funkcji e^x na przedziale [0.0, 1.0].

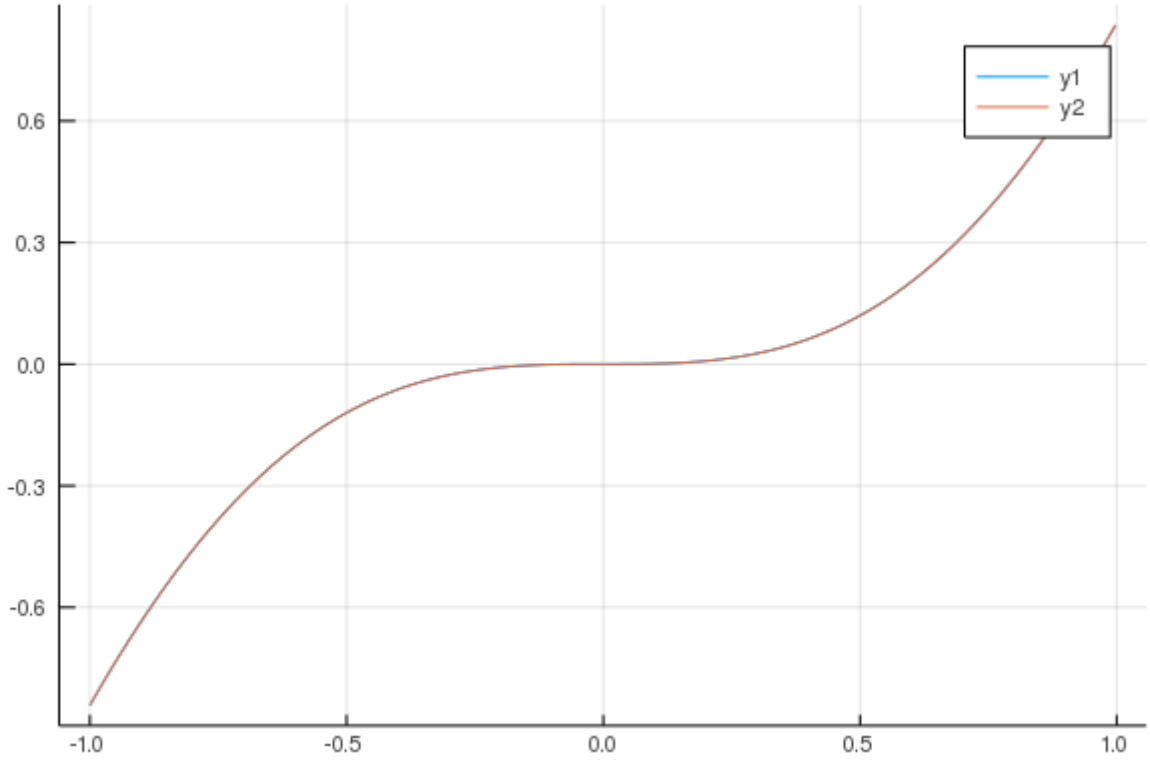
b) $x^2 sinx, [-1,1]$



n = 5



n = 10



n = 15

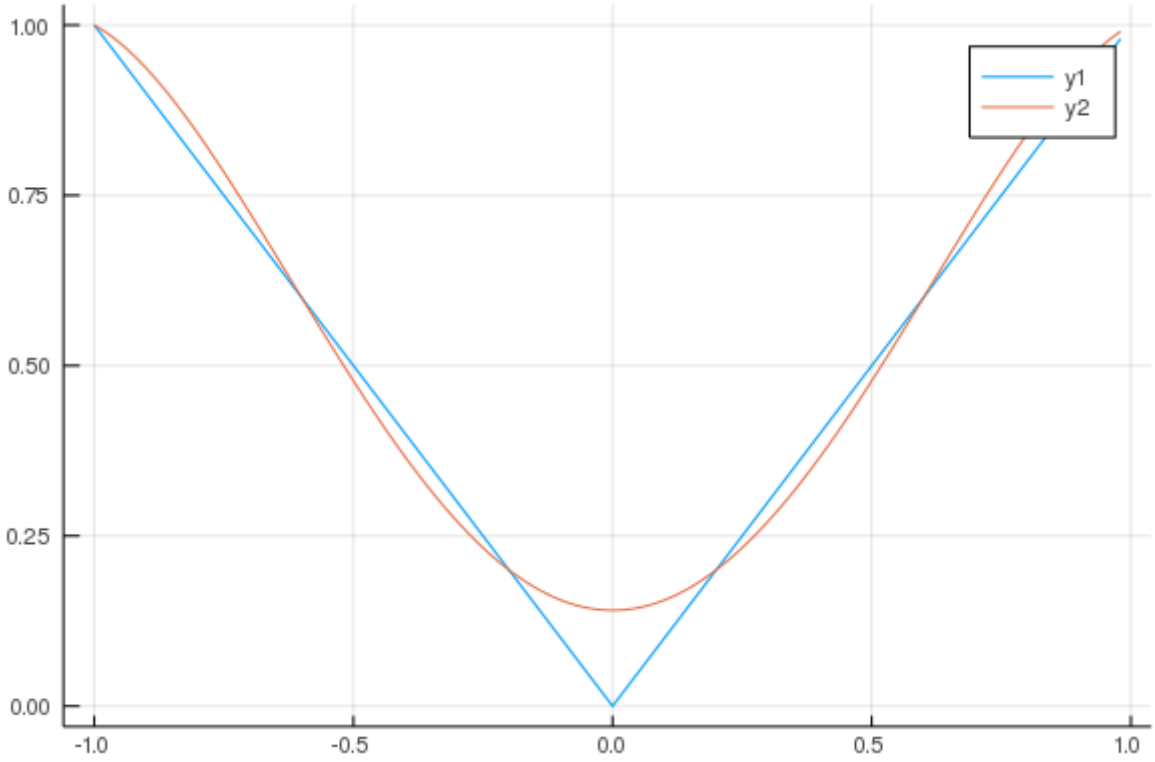
Obserwacje i wnioski:

- Zwiększenie stopnia wielomianu poprawiło jakość interpolacji.
- Wielomian 10-ego stopnia wystarczył do interpolacji na przedziale [-1, 1].

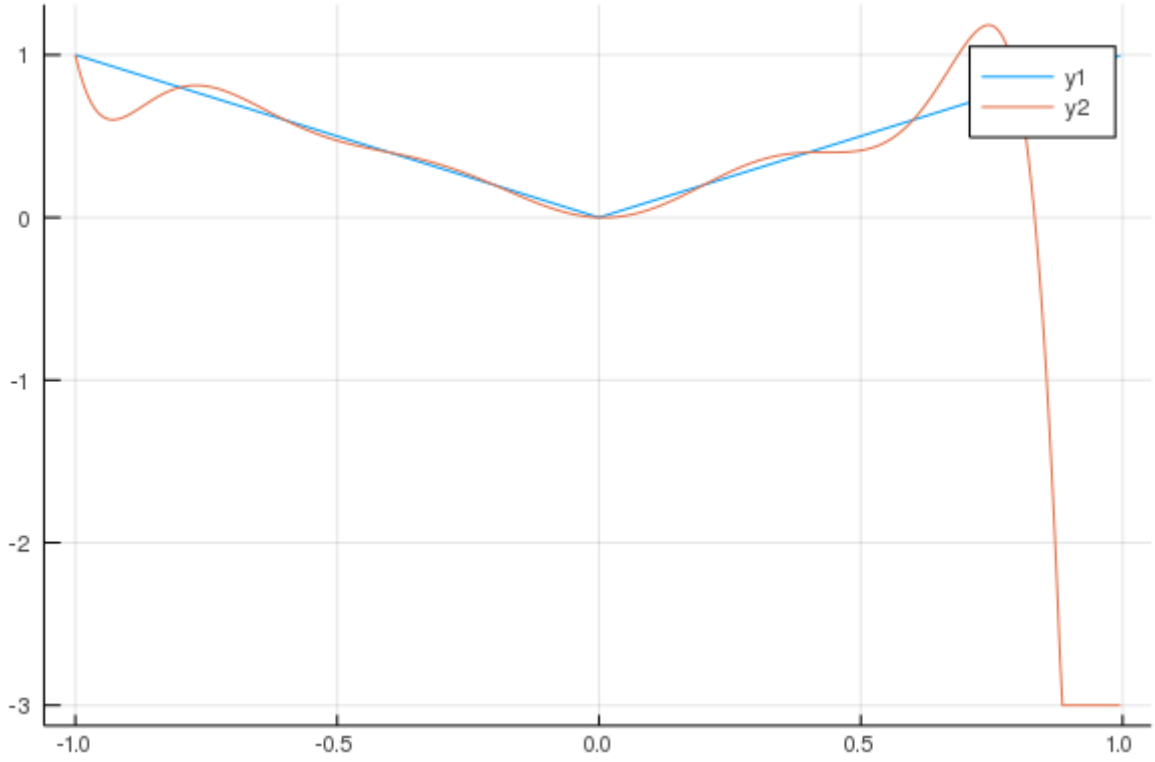
Zadanie 6.

W celu lepszego zobrazowania funkcji dla n = 10, wartości zostały ograniczone do [-3; 3].

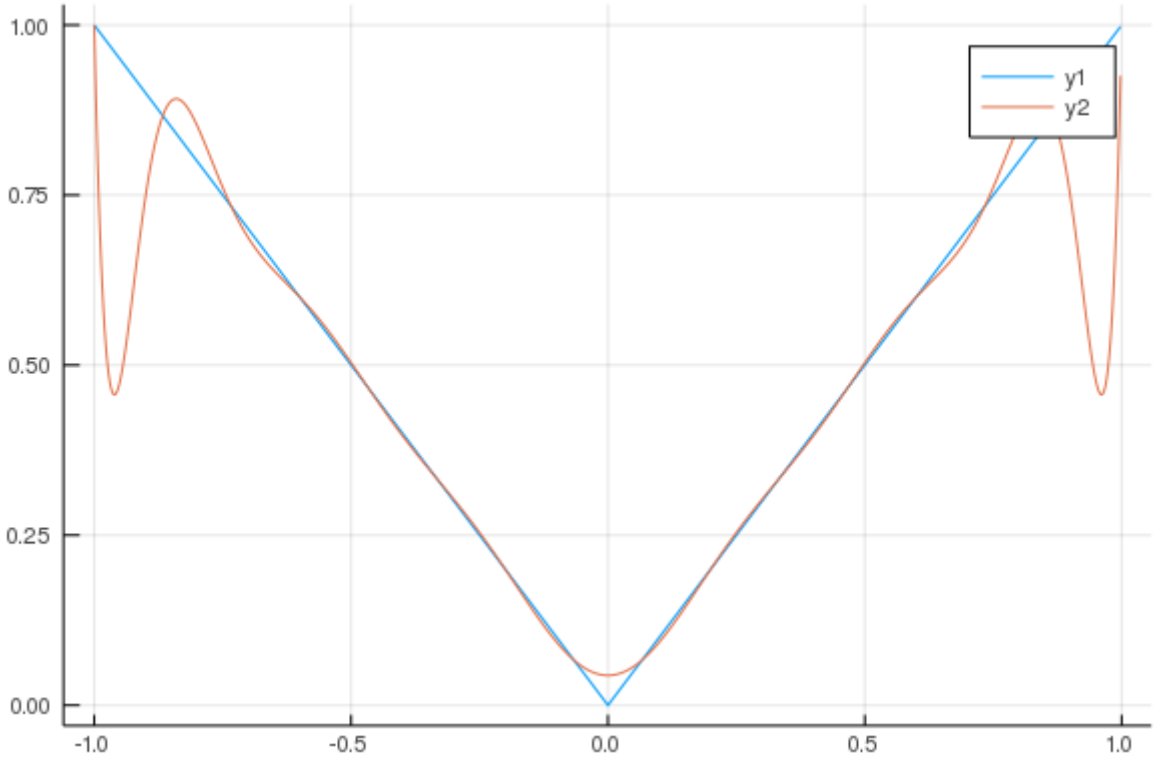
a) $|x|, [-1, 1]$



n = 5



n = 10

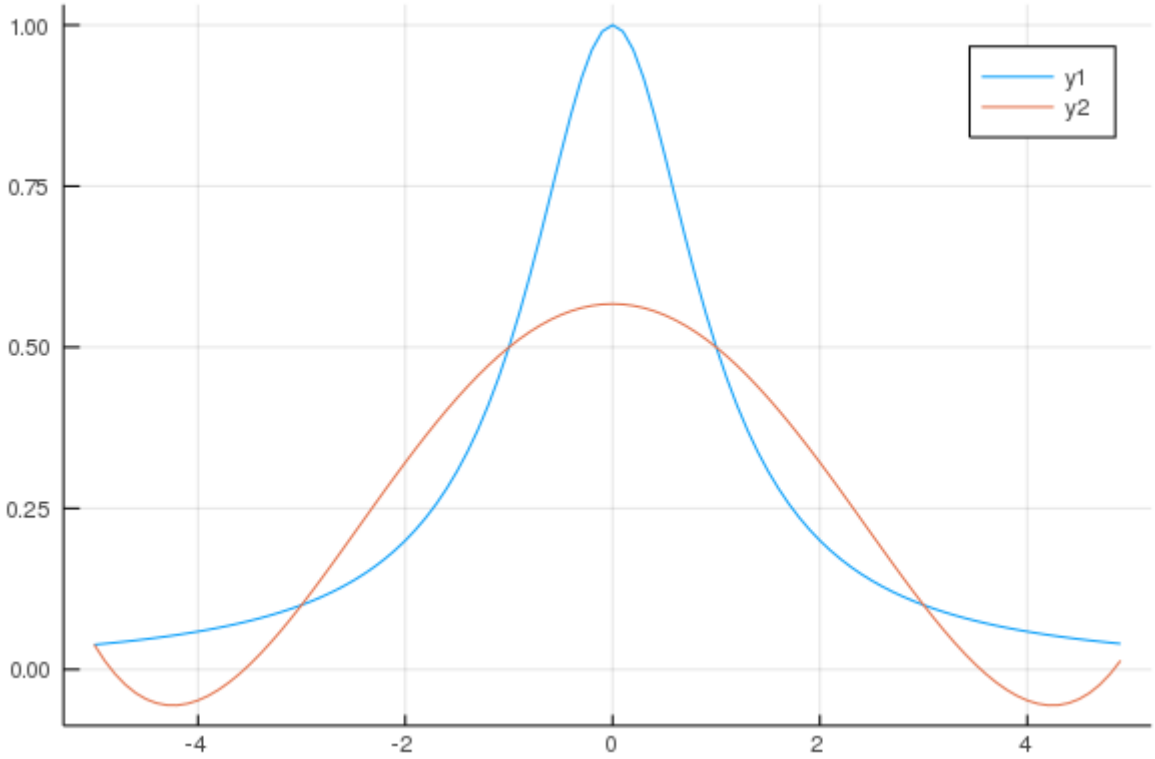


n = 15

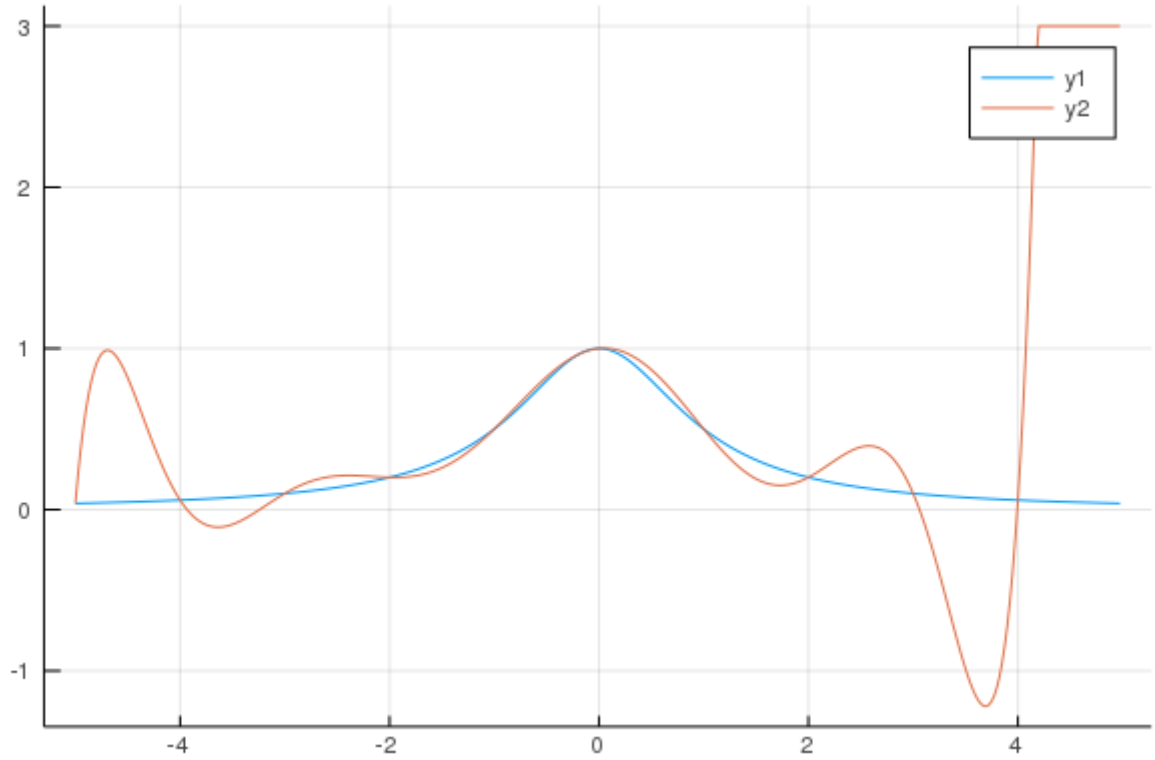
Obserwacje i wnioski:

- Wyższe stopnie wielomianu powodują utratę jakości interpolacji przy skrajach przedziału.
- Ostrza w rodzaju tego przy 0 są bardzo trudne do osiągnięcia przez wielomiany interpolacyjne, szczególnie, jeżeli nie znajduje się na nich węzeł – zjawisko typowe dla funkcji niegładkich.

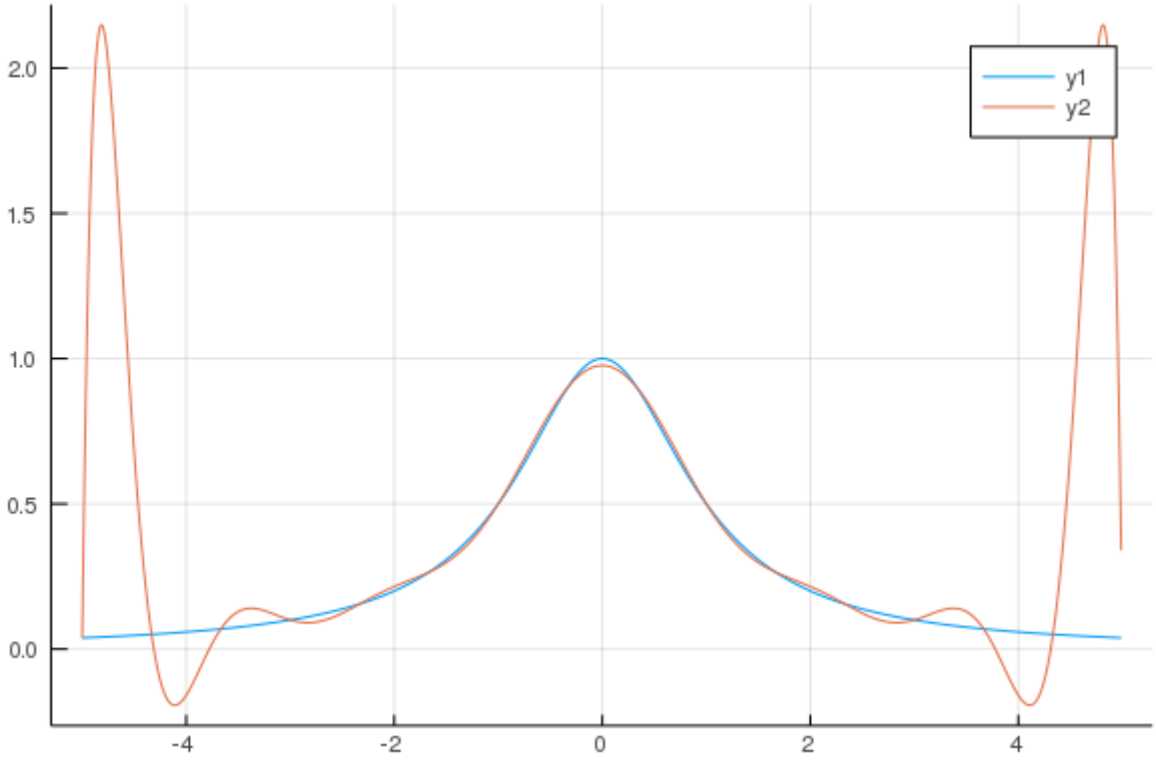
b) $\frac{1}{1+x^2}, [-5,5]$



n = 5



n = 10



n =15

Obserwacje i wnioski:

- Po pierwszym zwiększeniu ilości węzłów jakość interpolacji uległa polepszeniu. Widać to w szczególności w okolicach 0.
- Po kolejnym zwiększeniu stopnia wielomianu interpolacja pogorszyła się przy skrajach przedziału.
- Obserwujemy efekt Runge'go, ponieważ funkcja rysujNnfx wykorzystuje węzły równoodległe. Zjawiska można uniknąć, stosując inny rozkład węzłów, np. normalny.