List 4 report

Albert Kołodziejski

December 3, 2023

Exercise 1

Description of problem:

This exercise requires to implementation of the differential quotients without using a 2-dimensional array.

Description of method:

Let's take a closer look at the 2-dimensional array:

$f[x_0]$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$
$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	
$f[x_2]$	$f[x_2, x_3]$		
$f[x_3$			

Our goal is to compute and return the first row starting from the first column.

$f[x_0]$
$f[x_1]$
$f[x_2]$
$f[x_3]$

Take note that we need the last element in this column only once to compute $f[x_2, x_3]$, so after computing it we can put this value where $f[x_3]$ have been

$f[x_0]$
$f[x_1]$
$f[x_2]$
$f[x_2, x_3]$

Now take a look at f[x_2], it is used to compute two things, f[x_2, x_3] and f[x_1, x_2], but former of them is already computed, so we once again have element that is used to compute only one thing.

f[x_0]
f[x_1]
f[x_1, x_2]
f[x_2, x_3]

After doing this once again for f[x_1] we get:

f[x_0]
f[x_0, x_1]
f[x_1, x_2]
f[x_2, x_3]

As we can see, in this way we have computed 2 elements from the first row, now we need to go back to the last element and compute 3rd element.

$f[x_0]$
$f[x_0, x_1]$
$f[x_1, x_2]$
$f[x_1, x_2, x_3]$

$f[x_0]$
$f[x_0, x_1]$
$f[x_0, x_1, x_2]$
$f[x_1, x_2, x_3]$

And in the end, we can compute the last element:

$f[x_0]$
$f[x_0, x_1]$
$f[x_0, x_1, x_2]$
$f[x_0, x_1, x_2, x_3]$

In this way, we can compute the first row of the 2-dimensional array without storing all values, but only does what we need.

Exercise 2

Description of problem:

This exercise requires computing the value of the polynomial in the Newton form using the Horner scheme.

Description of method:

Let's take a closer look at the formula to compute the value of the polynomial in the Newton form:

$N_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + ... + f[x_0, ..., x_n](x - x_0)...(x - x_n)$

a lot of elements in this formula are multiplied by $(x - x_0)$, similar to the Horner scheme, we can remove this before parenthesis:

$N_n(x) = f[x_0] + (x - x_0)(f[x_0, x_1] + f[x_0, x_1, x_2](x - x_1)... + f[x_0, ..., x_n](x - x_1)...(x - x_n))$
We can do the same thing with $(x - x_1)$ and so on, finally we get:
$N_n(x) = f[x_0] + (x - x_0)(f[x_0, x_1] + (x - x_1)(f[x_0, x_1, x_2] + ... + (x - x_n)(f[x_0, ..., x_n]))...)$
Starting from the most inner parenthesis we can compute the value of the polynomial, while also minimizing the number of multiplications.

Exercise 3

Description of problem:

This exercise requires constructing the natural form of a polynomial from Newton one.

Description of method:

Similar to the previous exercise, we can use the Horner scheme, every single time we would multiply the polynomial by $(x - x_i)$ and add $f[x_0, ..., x_i]$ to the result. we can store polynomials as vectors of coefficients. This way computation simplifies to a few steps:

1. To multiply polynomials by x, we simply move every coefficient to the right by one place and add 0 at the beginning.

2. To add $f[x_0, ..., x_i]$ we add it to the last coefficient of polynomial from 1st step.

3. To mulitiply polynomial by x_i we mulitiply every coefficient by x_i.

4. Substract polynomial from 3rd step from polynomial from 2nd step.

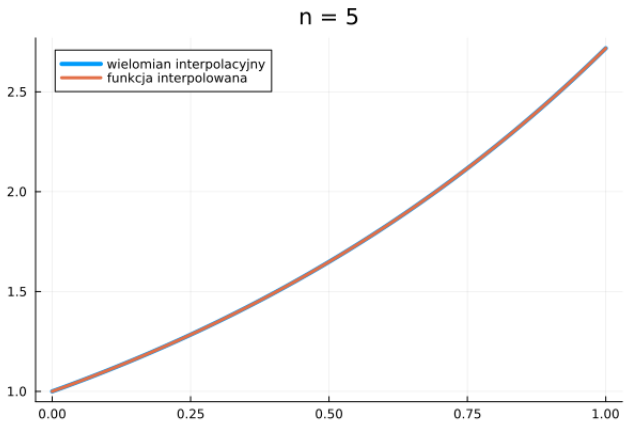
Exercise 5

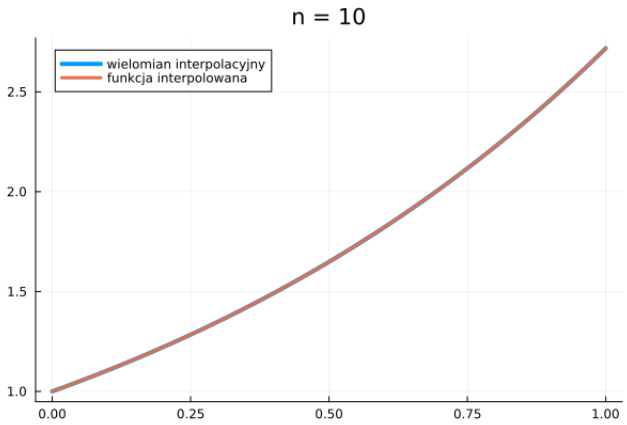
Description of problem:

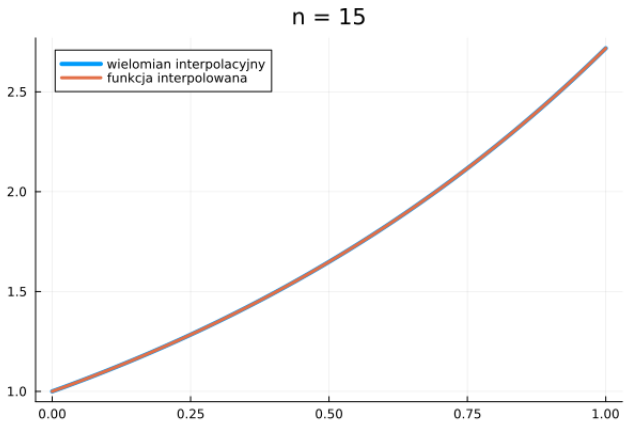
In this exercise we are going to compare interpolating polynomials with interpolated functions for different numbers of n:

Results:

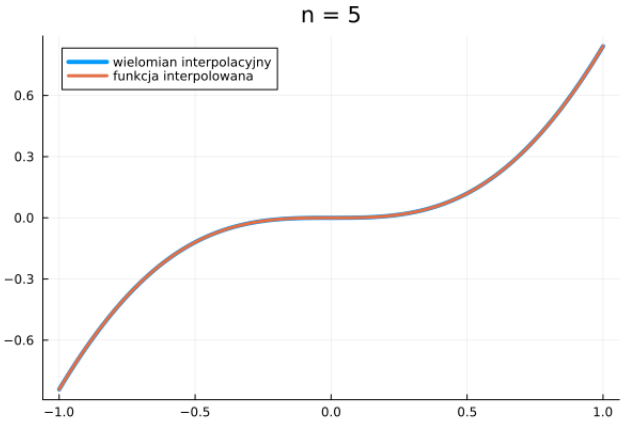
$$e^x, [0, 1], n = 5, 10, 15$$

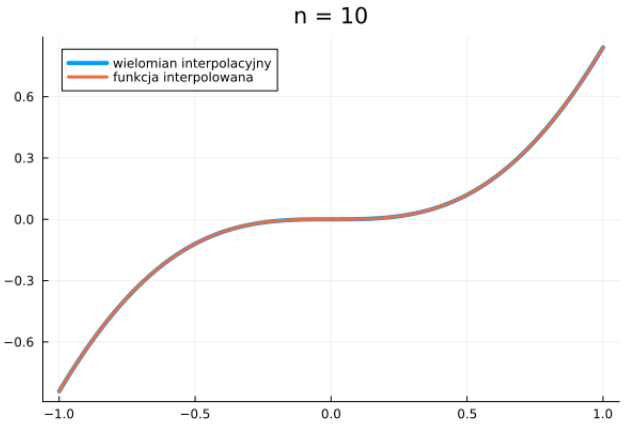


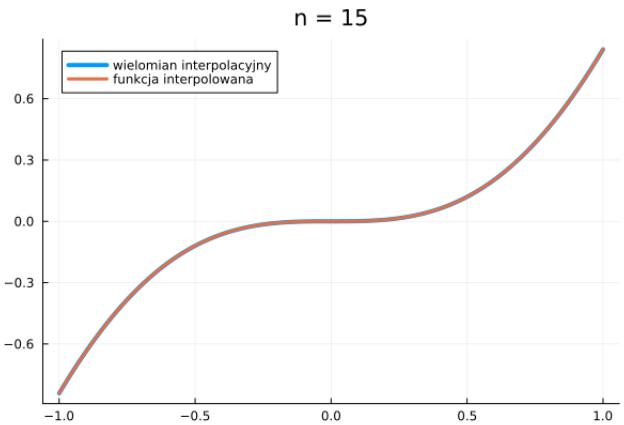




$$x^2 sin(x), [-1, 1], n = 5, 10, 15$$







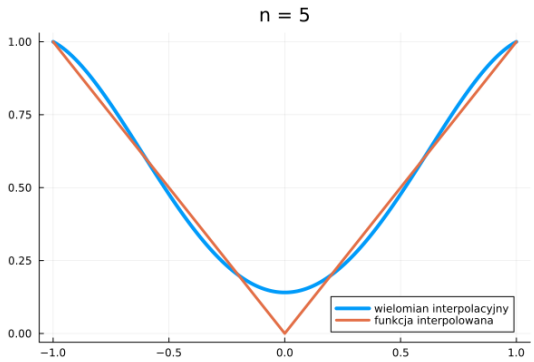
Exercise 6

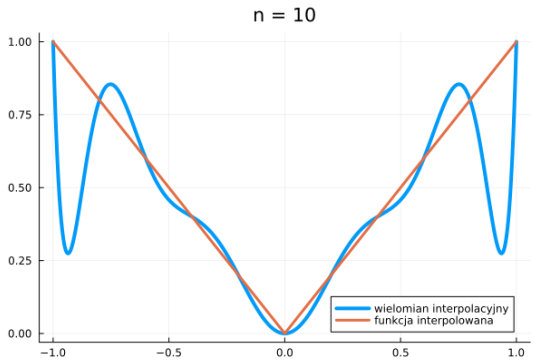
Description of problem:

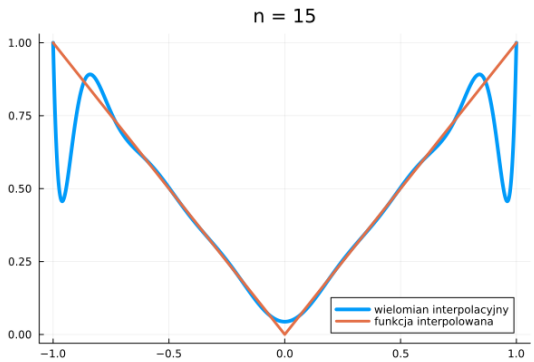
In this exercise we are going to compare interpolating polynomials with interpolated functions for different numbers of n:

Results:

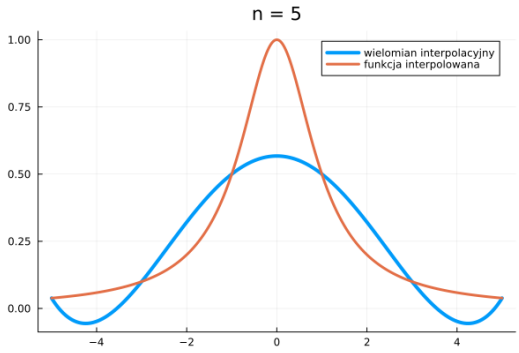
$$|x|, [-1, 1], n = 5, 10, 15$$

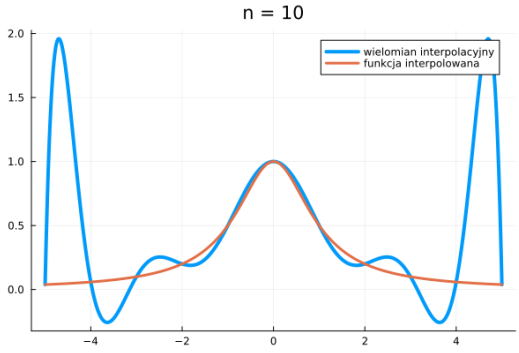


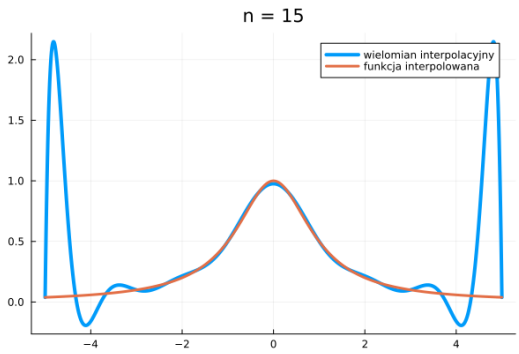




$$\frac{1}{1+x^2}, [-5, 5], n = 5, 10, 15$$







Interpretation and conclusions:

As we can see, for those functions, our approximation is not that good, especially near the edges of the range. What's more important is that the error will not go smaller with bigger n. A possible solution is to not take points evenly distributed, but instead take them from Chebyshev nodes, more nodes near the edges of the range, and less in the middle.