Scientific Computing - Interpolation

Mateusz Pełechaty

23 October 2022

1 Exercise

Write function, which calculates Newton's divided differences for given function. Function should be named *dividedDifference* and should take two arguments:

- $\vec{x}$: vector of nodes
- $\vec{f}$: vector of function values at nodes

Function should not use two-dimensional array.

Description

Newton's divided differences are defined as follows:

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0} \tag{1}$$

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

We can compute them in one-dimensional array by looping from last to first with the help of formula (1):

$$\begin{pmatrix} f[x_0] \\ f[x_1] \\ f[x_2] \\ \vdots \\ f[x_{n-1}] \\ f[x_n] \end{pmatrix} \rightarrow \begin{pmatrix} f[x_0] \\ f[x_0, x_1] \\ f[x_1, x_2] \\ \vdots \\ f[x_{n-2}, x_{n-1}] \\ f[x_{n-1}, x_n] \end{pmatrix} \rightarrow \begin{pmatrix} f[x_0] \\ f[x_0, x_1] \\ f[x_0, x_1, x_2] \\ \vdots \\ f[x_0 \ldots x_{n-2}, x_{n-1}] \\ f[x_0 \ldots x_{n-1}, x_n] \end{pmatrix} \tag{2}$$

Solutions and tests

Function with documentation can be found in *src/interpolation.jl*.
Tests are in *test/test_ dividedDifference.jl*

2 Exercise

Write function which calculates value of polynomial at a given point. Function should be named *valueNewton* and should take three arguments:

- $\vec{x}$: vector of nodes
- $\vec{f}$: vector of function values at nodes
- t : point at which polynomial should be evaluated

Function should compute in $O(n)$ time complexity

Description

Method of computation is based on Horner's rule:

$$\begin{aligned} p(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1)\dots(x - x_{n-1}) \\ &= a_0 + (a_1 + (a_2 + \dots (a_{n-1} + (a_n \cdot (x - x_{n-1}))(x - x_{n-2}))\dots)(x - x_0) \end{aligned}$$

We start from the innermost term, add coefficient and multiply by $x - x_i$ till we reach the outermost term.

Solutions and tests

Function with documentation can be found in *src/interpolation.jl*.
Tests are in *test/test_ valueNewton.jl*

3 Exercise

Write a function that can compute coefficients of a polynomial in general form by using Newton's divided differences. Function should be named *general* and should take two arguments:

- $\vec{x}$: vector of nodes
- $\vec{f}$: vector of function values at nodes

Function should compute in $O(n^2)$ time complexity

Solutions and tests

Function with documentation can be found in *src/interpolation.jl*.
Tests are in *test/test_ general.jl*

4 Exercise

Write a function that will interpolate given function on a given interval using Newton's interpolation formula. Then it should plot the function and its interpolation. Function should be named *drawInterpolation* and should take four arguments:

- f : anonymous function to interpolate
- $[a, b]$: left and right bounds of interval
- n : Interpolated polynomial's degree

Solutions and tests

Function with documentation can be found in *src/interpolation.jl*
Tests are in *test/test_ drawInterpolation.jl*

5 Exercise

Test function *drawInterpolation* with the following arguments:

- $e^x, [0, 1], n = 5, 10, 15$
- $x^2 \cdot \sin x, [-1, 1], n = 5, 10, 15$

Solutions and results

Solution can be found in *src/ex5/main.jl*. Plots can be found in *src/results/*:

Results

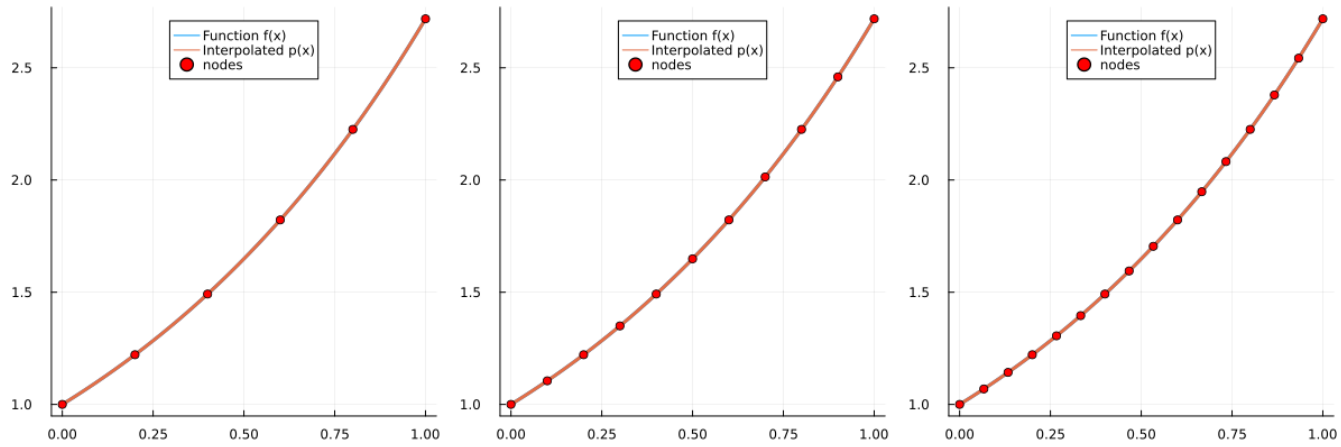

Figure 1: Interpolation of $f(x) = exp(x)$, $[a, b] = [0, 1]$ and $n = [5, 10, 15]$

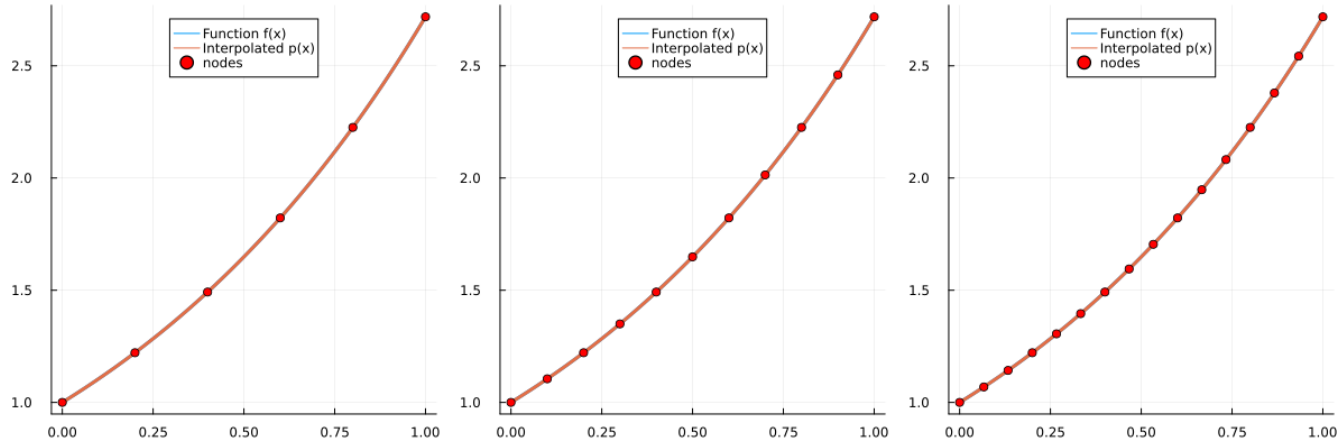

Figure 2: Interpolation of $f(x) = x^2$, $[a, b] = [-1, 1]$ and $n = [5, 10, 15]$

Conclusion

We can see that there is no visible difference between interpolation and function itself. This is due to two reasons:

- We are interpolating on a small interval
- We can bound maximum of $\frac{f^{(n+1)}(\xi)}{(n+1)!}$
- f is smooth and behaves well. One could say that f is *preety*

6 Exercise

Spot *discrepancy phenomena* by testing function *drawInterpolation* with the following arguments:

- $|x|, [-1, 1], n = 5, 10, 15$
- $\frac{1}{1+x^2}, [-5, 5], n = 5, 10, 15$; so called Runge's phenomenon

Solutions

Solution can be found in *src/ex6/main.jl*. Plots can be found in *src/results/*

Results

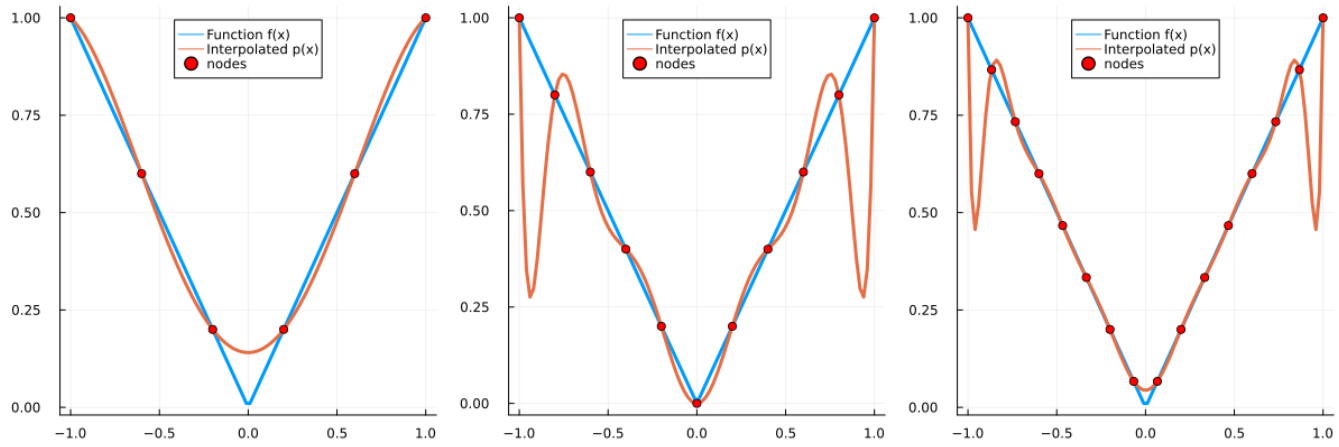

Figure 3: Interpolation of $f(x) = |x|$, $[a, b] = [-1, 1]$ and $n = [5, 10, 15]$

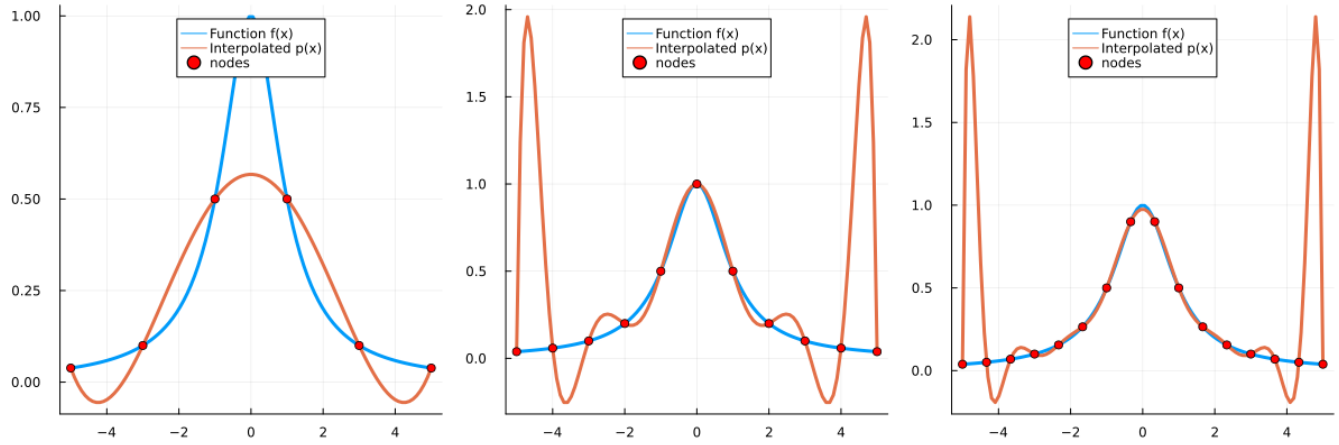

Figure 4: Interpolation of $f(x) = 1/(1 + x^2)$, $[a, b] = [-5, 5]$ and $n = [5, 10, 15]$

Conclusion

We can see that there is visible difference between interpolation and function itself. When amount of nodes gets bigger, we can see that magnitude of the oscillations in the interpolation is increased. We can analyse the error formula to see why this happens:

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot \Pi_{i=0}^{n}(x - x_i)$$

where ξ is some point between a and b
Even though I can't exactly say why this phenomena happens, I can spot that when when points are equidistanced, then for $x = a + \frac{b-a}{2n}$

$$\Pi_{i=0}^{n}(x - x_i) = (\frac{b-a}{2n})^2 \cdot \Pi_{i=2}^{n}(i \cdot \frac{b-a}{n} + \frac{b-a}{2n}) = (\frac{b-a}{2n})^{n+1} + (\frac{b-a}{n})^{n-1} \cdot n!$$

Let's calculate what happens when $n \to \infty$

$$\lim_{n\to\infty} \Pi_{i=0}^{n}(x - x_i) = \lim_{n\to\infty} (\frac{b-a}{n})^{n-1} \cdot n! = \lim_{n\to\infty} (\frac{b-a}{n})^{n-1} \cdot \sqrt{2\pi n} \cdot (\frac{n}{e})^n = \lim_{n\to\infty} (\frac{b-a}{e})^{n-1} \cdot n\sqrt{n} \cdot e^{-1}$$

We can see that if $b - a \geq e$ then the product grows exponentially. It may show the reason why the oscillations are growing at the edges of interval. To fully prove that the error grows exponentially, we would have to also grasp $f^{(n+1)}(\xi)$, to which I have no idea how to do.