Wrocław University of Science and Technology

Scientific Calculations - Task List 4

Analysis of accuracy of approximations of a function using Polynomial Interpolation

Marcel Jerzyk
December 7, 2019

1 Calculating the Quotient of Differential Functions

1.1 Problem

Create a function that calculates *differential quotients* without using a two-dimensional matrix (array).

1.2 Implementation

Let $f : X \to Y$ and $x_1, x_2 \in X$. Then *differential quotient* is calculated from:

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

We can set up an array with initial values of *differential quotient* equal to f values at x_i and then starting from bottom to top update it with:

$$f(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-k}}$$

to achieve correct array of calculated *differential quotients*.

Code 1: ilorazyRoznicowe(x, f)

```
n <- length(x)
for k <- (2) to (n)
  for i <- (n) to (k) step (-1)
    do fx[i] <- (fx[i] - fx[i-1]) /
                (x[i] - x[i - k + 1])
  end
end
return fx
```

2 Value of Interpolative Polynomial of a Degree n in Newton Form

2.1 Problem

Write a function that calculates *interpolative polynomial* value of coefficient n in Newton Form $N_n(x)$ at point $x = t$ with generalized Horner algorithm in time $O(n)$.

2.2 Implementation

Interpolative polynomial can be written as

$$\sum_{i=0}^{n} fx[i] \prod_{j=0}^{i-1} (x - x_j)$$

where $fx[i]$ is a vector of form $fx[i] = f(x_0, x_1, ..., x_{i-1})$ containing differential quotients and where x_j is an interpolating node. Generalized Horner Scheme can be written as:

$$w_n(x) = fx[n]$$
$$w_i(x) = fx[i] + (x - x_i) * w_{i+1}(x)$$
$$i = n - 1, ..., 0$$
$$N_n(x) = w_0(x)$$

Code 2: warNewton(x, fx, t)

```
n <- length(x)
nt <- fx[n]
for i <- (n-1) to (1) step (-1)
    do nt <- fx[i] + (t - x[i]) * nt
end
return nt
```

3 Coefficients of the Natural Form of Interpolative Polynomial in Newton's form

3.1 Problem

Knowing the coefficients of interpolation polynomial in the Newton form $c_0 = f[x_0]$, $c_1 = f[x_0, x_1], ..., c_n = f[x_0, ..., x_n]$ (differential quotients) and nodes $x_0, x_2, ..., x_n$ write a function calculating, in $O(n^2)$ time, coefficients of it's Normal Form $a_0, ..., a_n$, it means: $a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$.

3.2 Implementation

So the coefficient standing at the highest power polynomials in natural form is known at the input ($c_n = fx[x_0, ..., x_n]$) so we can copy them into a new array "coefficients".

- We start iterating first loop indexed by i with n iterations. On every step it calculates a new coefficient value by subtracting $c_{i+1} * node[i]$ subtracting current iteration differential quotient.
- Then the inner loop starts indexed by k with iterations amount equal to range between n and i (so $n - i$ iterations). In every step it updates values of previous coefficients by multiplying them with subsequent factors ($c_k = c_k - c_{k+1} * node[i]$).

On finish the newly created array will contain values equal to coefficients values of the interpolated polynomial in it's canonical form (value on c_i is equal to coefficient at i power).

Code 3: naturalna(x, fx)

```
1  n <- length(fx)
2  coef[n] <- fx[n]
3  for i <- (n-1) to (-1) step (1)
4    do t <- coef[i+1] * x[i]
5       coef[i] = fx[i] - t
6    for k in (i+1, n-1)
7      do t <- coef[k+1] * x[i]
8         coef[k] = coef[k] - t
9    end
10 end
11 return coef
```

4 Drawing the interpolated function

4.1 Problem

Write a function that will interpolate $f(x)$ in interval $[a, b]$ with interpolative polynomial of a degree n in Newton Form. Then it draws interpolated polynomial and interpolated function. In interpolation - use equidistant nodes ($x_k = a + kh, h = \frac{(b-a)}{n}, k = 0, 1, ..., n$.

4.2 Implementation

First we calculate space between nodes equal to $\frac{b-a}{n}$. Then with help of two additional vectors (to store nodes and values) we start iterating and on every step we create a node $x_i = a + i * h$ with value $y_i = f(x_i)$. After the loop is finished, we use those arrays to calculate differential quotients (with *ilorazyRoznicowe*) and in next loop with *warNewton* we get values in all points of interpolated polynomial. Then we draw it using *Plots* with the $f(x)$.

Code 4: rysujNnfx(f, a, b, n)

```
1  h <- (b-a)/n
2  for i <- (0) to (n)
3    do x[i+1] = a + i*h
4       y[i+1] = f(x[i+1]
5  end
6
7  fx <- ilorazyRoznicowe(x, y)
8  accuracy = (b - a) / points
9
10 for i <- (0) to (points)
11   do t = a + i * accuracy
12      fxVal[i+1] = f(t)
13      ip[i+1] = warNewton(x, fx, t)
14 end
15
16 drawGraph(fxVal, ip)
```

5 Graphs of Functions whose values Coincide

5.1 Problem

Test function from the previous task on these examples:

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 Graphs

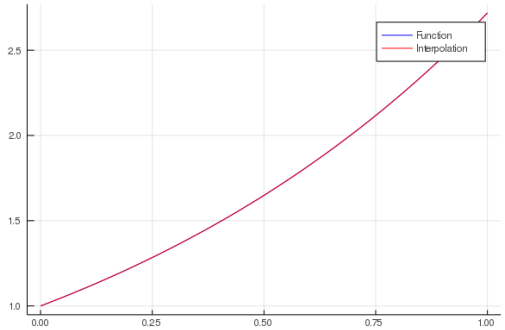

Figure 1: $f(x)$ with $n = 5$

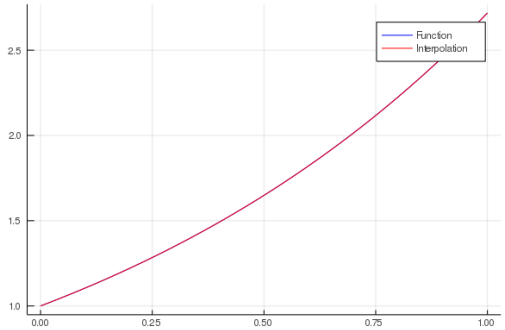

Figure 2: $f(x)$ with $n = 10$

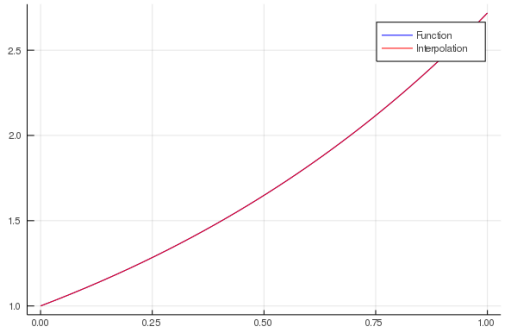

Figure 3: $f(x)$ with $n = 15$

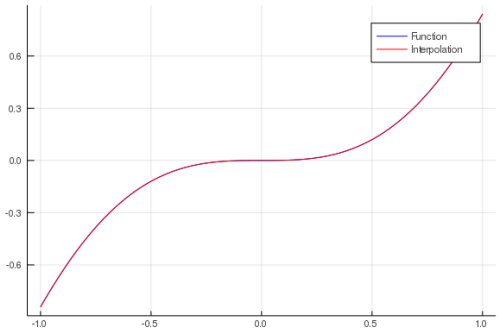

Figure 4: $g(x)$ with $n = 5$

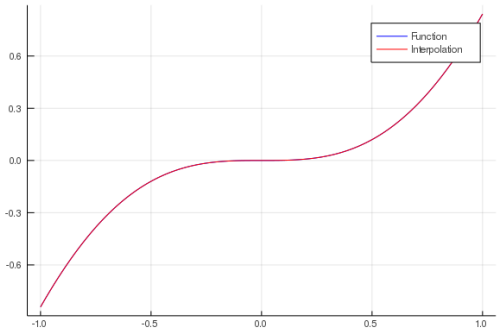

Figure 5: $g(x)$ with $n = 10$

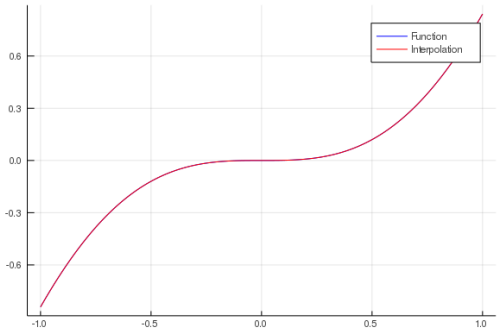

Figure 6: $g(x)$ with $n = 15$

5.3 Conclusions

At given numerical intervals, the function values overlaps very tight with interpolation polynomial values. It is because there are very tiny differences between these two functions, which further decrease as the n degree of polynomial increases.

6 Graphs of Functions whose values do not Coincide

6.1 Problem

Test function from Task 4 on these examples (discrepancy phenomenon):

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

6.2 Graphs

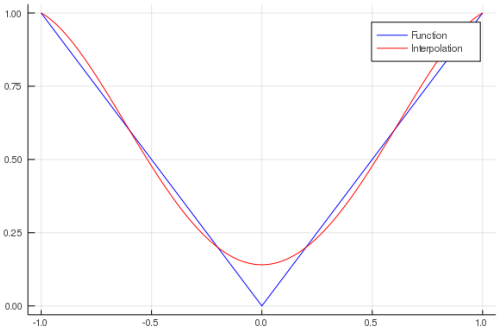

Figure 7: $f(x)$ with $n = 5$

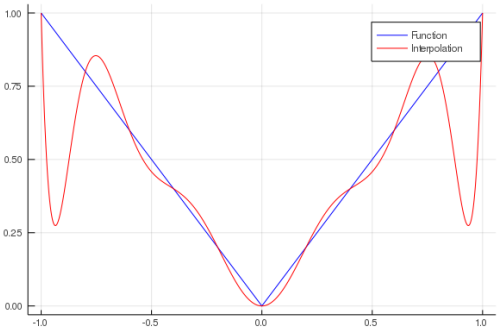

Figure 8: $f(x)$ with $n = 10$

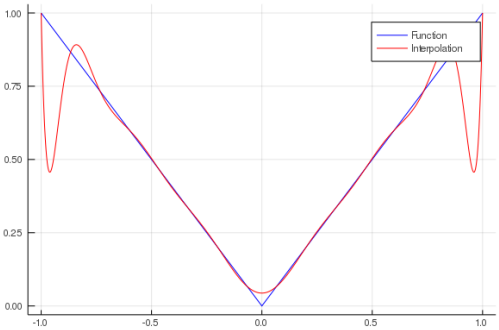

Figure 9: $f(x)$ with $n = 15$

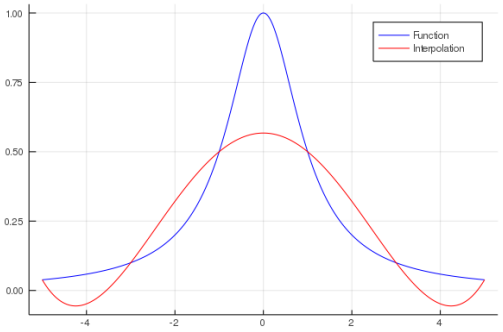

Figure 10: $g(x)$ with $n = 5$

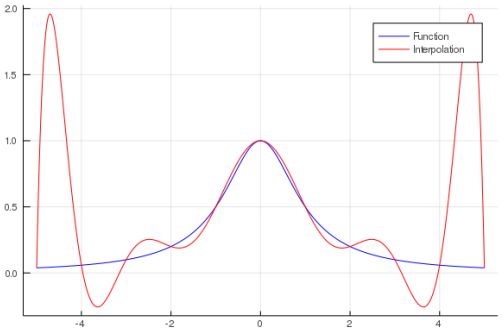

Figure 11: $g(x)$ with $n = 10$

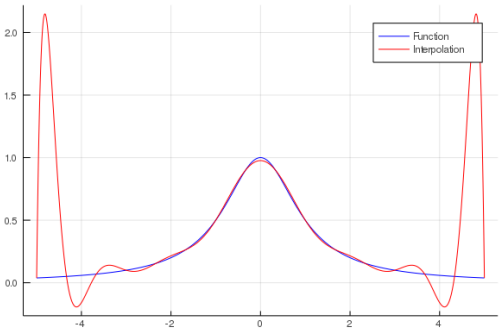

Figure 12: $g(x)$ with $n = 15$

6.3 Conclusions

For these functions, increasing the degree of interpolation polynomial improves the accuracy of the function overlapping only up to a point (further increase of n impairs interpolation). Functions near the end of range are more and more deflected while achieving better accuracy at the middle of it. The problem occurs because of equal distance between successive nodes (*Runge's phenomenon*). So to fix that we can for example use varying distance at the ends of the range or use Czybyszew Polynomial zeros as nodes.