

Numerical Practice 1

Aakash Patil

aakash.patil@master.centraledlille.fr

Abstract

1. Study of polynomial interpolation methods for Runge function and a Trigonometric function.
 2. Study of interpolation error as a function of number of discretization points.
-

1. Interpolation

Interpolation is used to determine intermediate values between finite data points. The most common method used for this purpose is polynomial interpolation.

$$f(x) = a_0 + a_1 * x + a_2 * x^2 + \dots + a_n * x^n$$

For $n + 1$ data points, there is one and only one polynomial of order n that passes through all the points. For example, there is only one straight line (that is, a first-order polynomial) that connects two points. Similarly, only one parabola connects a set of three points. Polynomial interpolation consists of determining the unique n th-order polynomial that fits $n + 1$ data points. This polynomial then provides a relation to compute intermediate values.

2. Newton Divided Difference Interpolation Method

There are a variety of alternative forms for expressing an interpolating polynomial. Newton's divided-difference interpolating polynomial is among the most popular and useful forms.

The generalized equation to fit an n th-order polynomial to $n + 1$ data points. The n th-order polynomial as per Newton's method is:

$$f_n(x) = b_0 + b_1 * (x - x_0) + \dots + b_n * (x - x_0) * (x - x_1) * \dots * (x - x_{n-1})$$

Data points can be used to evaluate the coefficients b_0, b_1, \dots, b_n . For an n th-order polynomial, $n + 1$ data points are required: $[x_0, f(x_0)], [x_1, f(x_1)], \dots, [x_n, f(x_n)]$.

We use these data points and the following equations to evaluate the coefficients:

$$b_0 = f(x_0)$$

$$\begin{aligned}
b1 &= f[x1, x0] \\
b2 &= f[x2, x1, x0] \\
- \\
- \\
- \\
bn &= f[xn, xn - 1, \dots, x1, x0]
\end{aligned}$$

where the bracketed function evaluations are finite divided differences. For example, the first finite divided difference is represented generally as

$$f[xi, xj] = (f(xi) - f(xj)) / (xi - xj)$$

Similarly, the nth finite divided difference is

$$f[xn, xn - 1, \dots, x1, x0] = (f[xn, xn - 1, \dots, x1] - f[xn - 1, xn - 2, \dots, x0]) / (xn - x0)$$

These coefficients can be evaluated and put in the equation to yield the interpolation polynomial.

3. Lagrange Interpolation Method

The Lagrange interpolating polynomial is simply a reformulation of the Newton polynomial that avoids the computation of divided differences. It can be represented concisely as

$$f(x) = \sum_{i=0}^n Li(x) * f(xi)$$

where

$$Li(x) = \prod_{j=0, j \neq i}^b (x - xj) / (xi - xj)$$

For example, the linear version (n = 1) is

$$f1(x) = (x - x1) * f(x0) / (x0 - x1) + (x - x0) * f(x1) / (x1 - x0)$$

the second-order version is

$$\begin{aligned}
f2(x) &= (x - x1) * (x - x2) * f(x0) / ((x0 - x1)(x0 - x2)) + (x - x0) * (x - x2) * f(x1) / ((x1 - x0)(x1 - x2)) \\
&+ (x - x0) * (x - x1) * f(x2) / ((x2 - x0)(x2 - x1))
\end{aligned}$$

4. Code

The code has been uploaded to

<https://github.com/aakash30jan/ClassWork-TurbulenceMaster/tree/master/06-11-2017/src/>

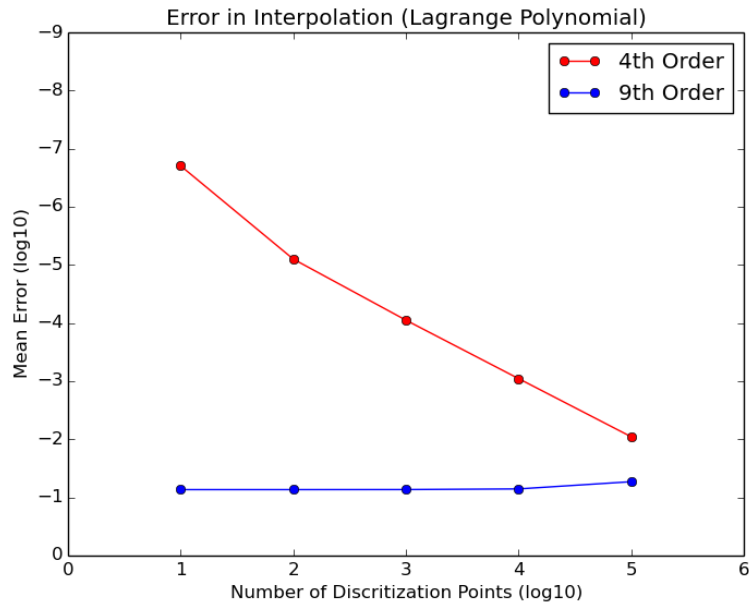


Figure 1: Lagrange Interpolation Error for Equation 1 (4th and 9th order)

5. Results

5.1. Functions

Following two functions were used to study polynomial interpolation:

Eq 1. $f(x) = 1/(1 + 25x^2)$ where $x \in [-1, 1]$

Eq 2. $f(x) = \cos(x * \pi)$ where $x \in [-1, 1]$

5.2. Plots

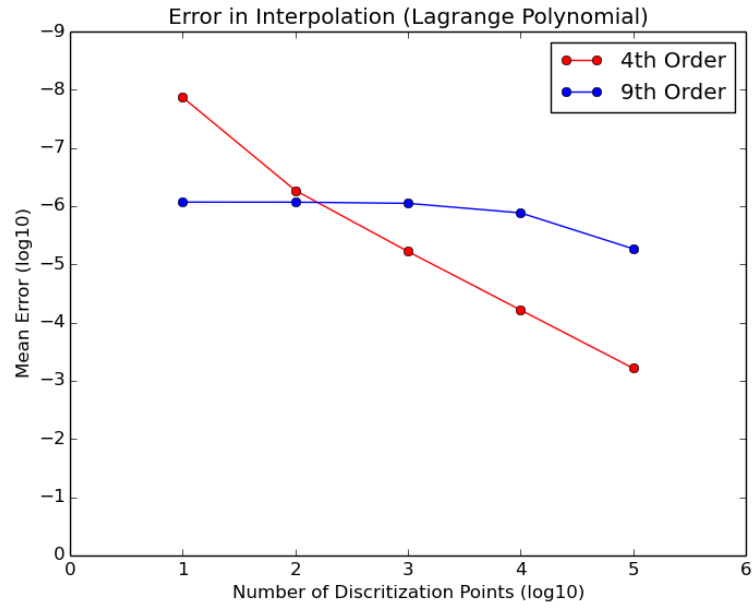


Figure 2: Lagrange Interpolation Error for Equation 2 (4th and 9th order)

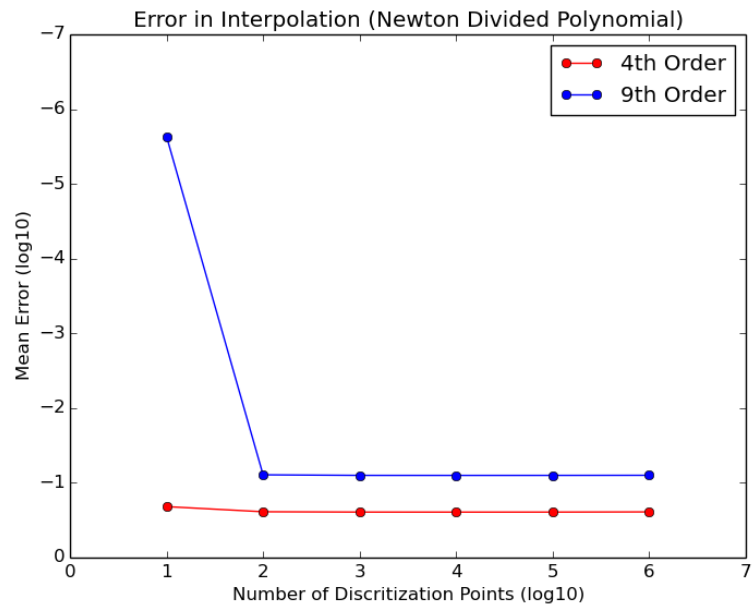


Figure 3: Newton Interpolation Error for Equation 1 (4th and 9th order)

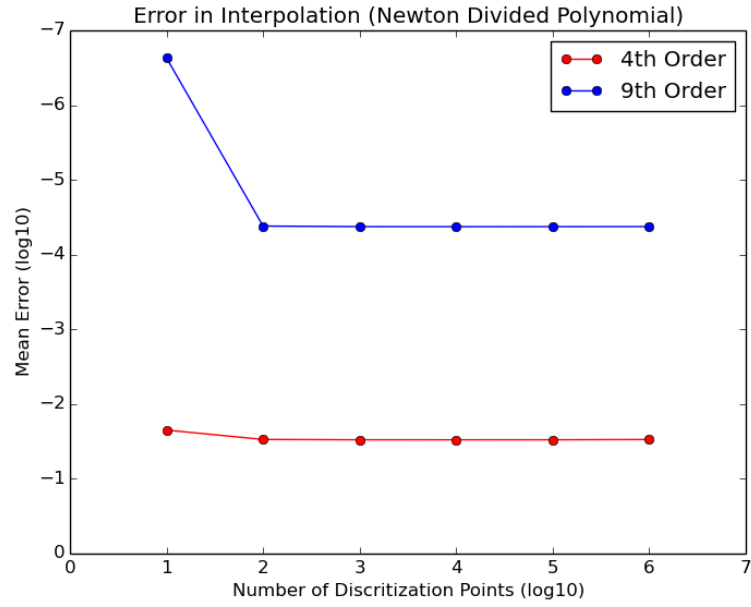


Figure 4: Newton Interpolation Error for Equation 2 (4th and 9th order)

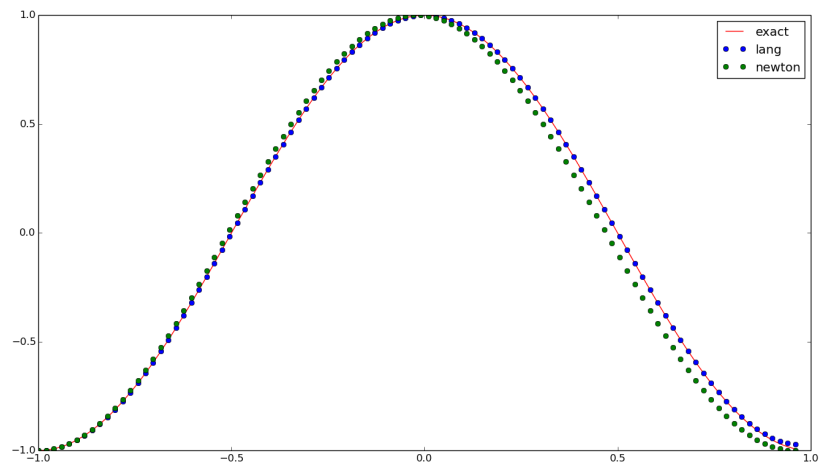


Figure 5: Comparison of interpolated points from 9th order polynomial for Equation 2