Math 4543: Numerical Methods

Lecture 8 — Lagrangian Interpolation Method

Syed Rifat Raiyan
Lecturer
Department of Computer Science & Engineering
Islamic University of Technology, Dhaka, Bangladesh
Email: rifatraiyan@iut-dhaka.edu

Lecture Plan

The agenda for today

- Represent interpolant polynomials using Lagrangian method
- Generalize the formula for finding the n^{th} order interpolant
- Understand the advantage of Lagrangian method over the NDD and Direct Method
- See the pitfalls of choosing higher order interpolant polynomials (the Runge Phenomenon)

Lagrangian Interpolation

What is it?

Joseph-Louis Lagrange represented the interpolant polynomial in such a manner so that each term of the polynomial is a *product* between a *Lagrange weight functional value* and a *given functional value* from the set of data points.

As given in Figure 1, data is given at discrete points such as $(x_0, y_0), (x_1, y_1), \ldots\ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)$.

A continuous function $f(x)$ may be used to represent the $n+1$ data values with $f(x)$ passing through the $n+1$ points.

Then one can find the value of y at any other value x.

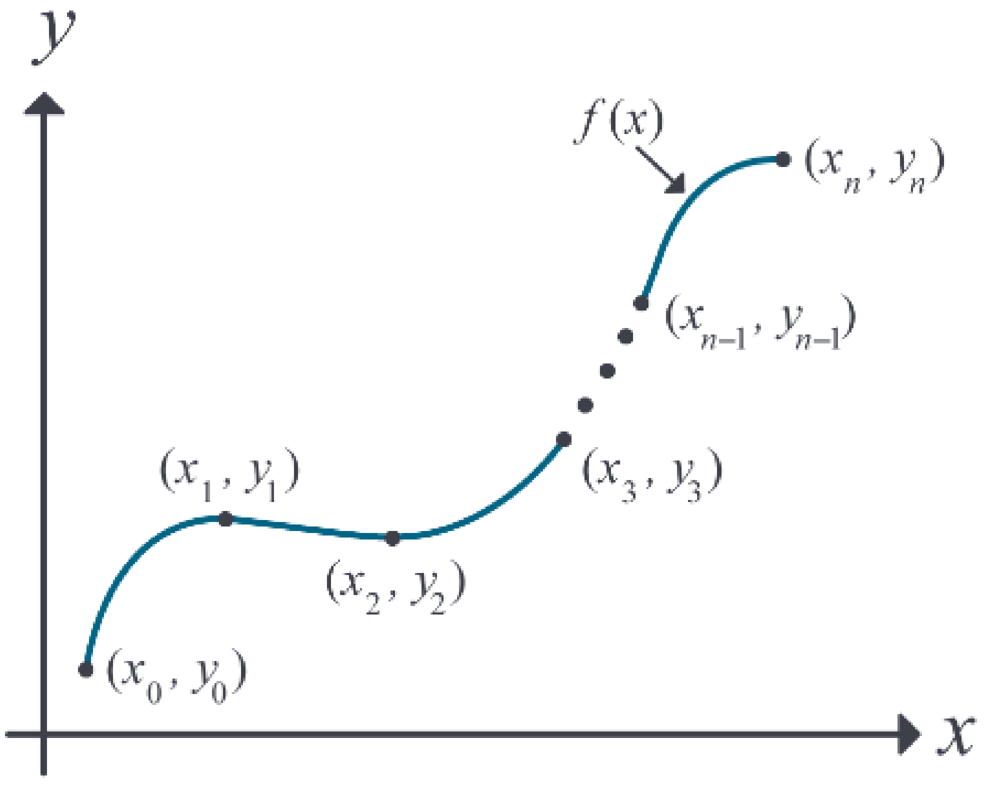


Figure 1. Interpolation of a function given at discrete points

Lagrangian Interpolation

General n^{th} order interpolation

Proof/Derivation: Page-**11** of http://ccnet.vidyasagar.ac.in:8450/pluginfile.php/606/mod_resource/content/1/Lagrange%20Interpolation.pdf

In this method, given $(x_0, y_0), \ldots, (x_n, y_n)$, one can fit a n^{th} order Lagrangian polynomial given by

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

where n in $f_n(x)$ stands for the n^{th} order polynomial that approximates the function $y = f(x)$ and

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

$L_i(x)$ is a weighting function that includes a product of n. terms with terms of $j = i$ omitted.

For example, the second order Lagrange polynomial passing through $(x_0, y_0), (x_1, y_1), and\,(x_2, y_2)$ is

$$f_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2)$$

Lagrangian Interpolation

Why do we need it?

The advantages are —

- ✓ Overall time complexity to obtain the interpolant is $O(n^2)$.
- ✓ Can be optimized further down to $O(n\ log^2 n\ )$ [**Link:** https://codeforces.com/blog/entry/94143]

For the **NDD method**, the best we could do was $O(n^2)$.

For the **Direct method**, we needed to calculate the ***inverse of a matrix*** and simultaneously solve ***all the equations*** to obtain all the coefficients. The time complexities of the algorithms that are used to do this are,

- Naïve Gaussian Elimination — $O\left(n^3 \log\left(\left|\left|A\right|\right| + \left|\left|b\right|\right|\right)\right)$, for $\boldsymbol{A}x = \boldsymbol{b}$
- LU Decomposition — $O(n^3)$
- Cramer's Rule — $O((n+1)!)$

Lagrangian Interpolation

A first-order polynomial example

The upward velocity of a rocket is given as a function of time in Table 1.

Table 1. Velocity as a function of time.

t (s)	$v(t)$ (m/s)
0	0
10	227.04
15	362.78
20	517.35
22.5	602.97
30	901.67

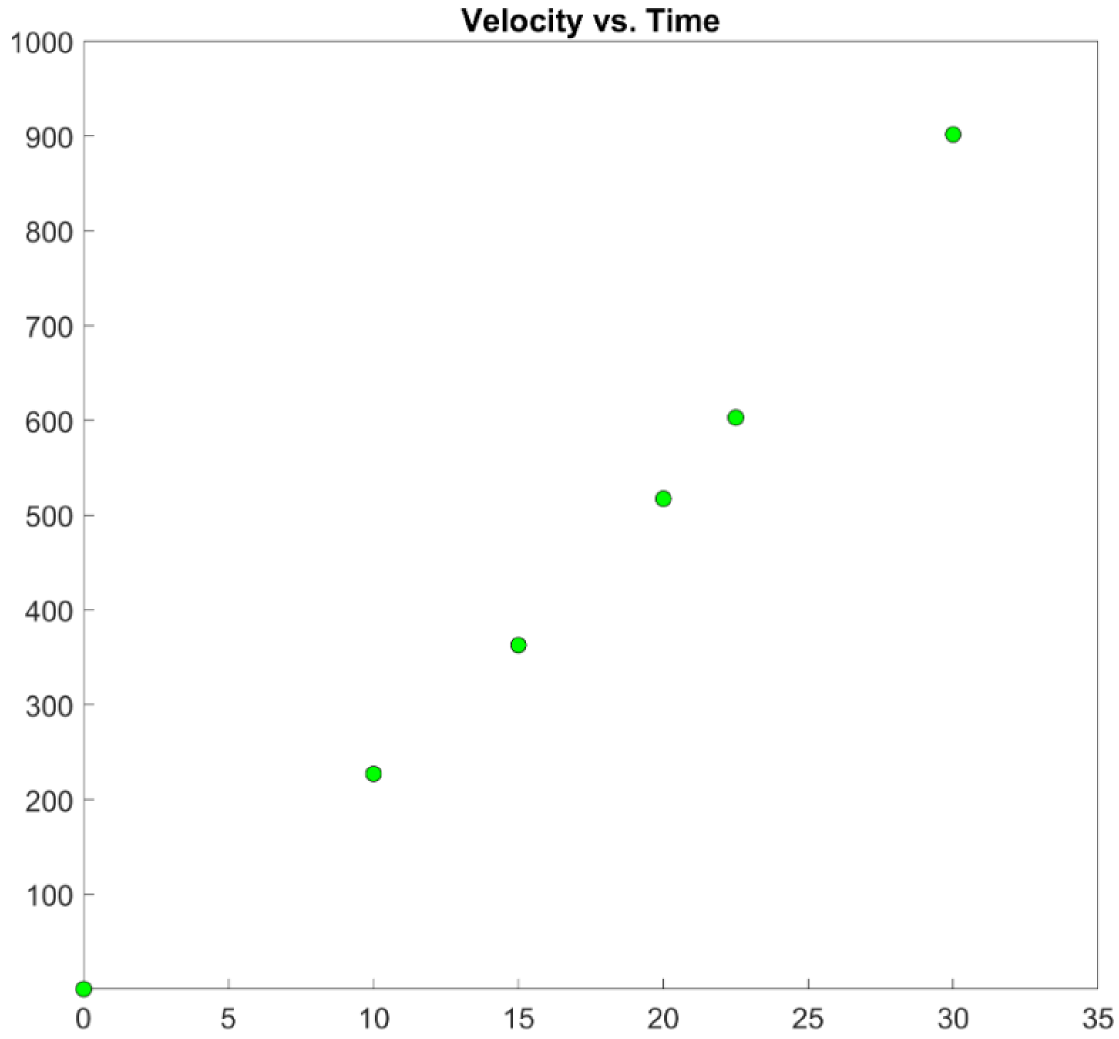


Figure 1. Graph of velocity vs. time data for the rocket example.

Determine the value of the velocity at $t = 16$ seconds using a first order Lagrange polynomial.

Lagrangian Interpolation

A first-order polynomial example

Solution

For first order polynomial interpolation (also called linear interpolation), the velocity is given by

$$v(t) = \sum_{i=0}^{1} L_i(t) v(t_i)$$
$$= L_0(t)v(t_0) + L_1(t)v(t_1)$$

Since we want to find the velocity at $t = 16$, and we are using a first order polynomial, we need to choose the two data points that are closest to $t = 16$ that also bracket $t = 16$ to evaluate it. The two points are $t_0 = 15$ and $t_1 = 20$.

Then

$$t_0 = 15,\ v(t_0) = 362.78$$
$$t_1 = 20,\ v(t_1) = 517.35$$

gives

$$L_0(t) = \prod_{\substack{j=0 \\ j \neq 0}}^{1} \frac{t - t_j}{t_0 - t_j} = \frac{t - t_1}{t_0 - t_1}$$

$$L_1(t) = \prod_{\substack{j=0 \\ j \neq 1}}^{1} \frac{t - t_j}{t_1 - t_j} = \frac{t - t_0}{t_1 - t_0}$$

Hence

$$v(t) = \frac{t - t_1}{t_0 - t_1} v(t_0) + \frac{t - t_0}{t_1 - t_0} v(t_1)$$
$$= \frac{t - 20}{15 - 20}(362.78) + \frac{t - 15}{20 - 15}(517.35), \quad 15 \le t \le 20$$

$$\boxed{\begin{aligned} v(16) &= \frac{16 - 20}{15 - 20}(362.78) + \frac{16 - 15}{20 - 15}(517.35) \\ &= 0.8(362.78) + 0.2(517.35) \\ &= 393.69 \text{ m/s} \end{aligned}}$$

Lagrangian Interpolation

A second-order polynomial example

The upward velocity of a rocket is given as a function of time in Table 1.

Table 1. Velocity as a function of time.

t (s)	$v(t)$ (m/s)
0	0
10	227.04
15	362.78
20	517.35
22.5	602.97
30	901.67

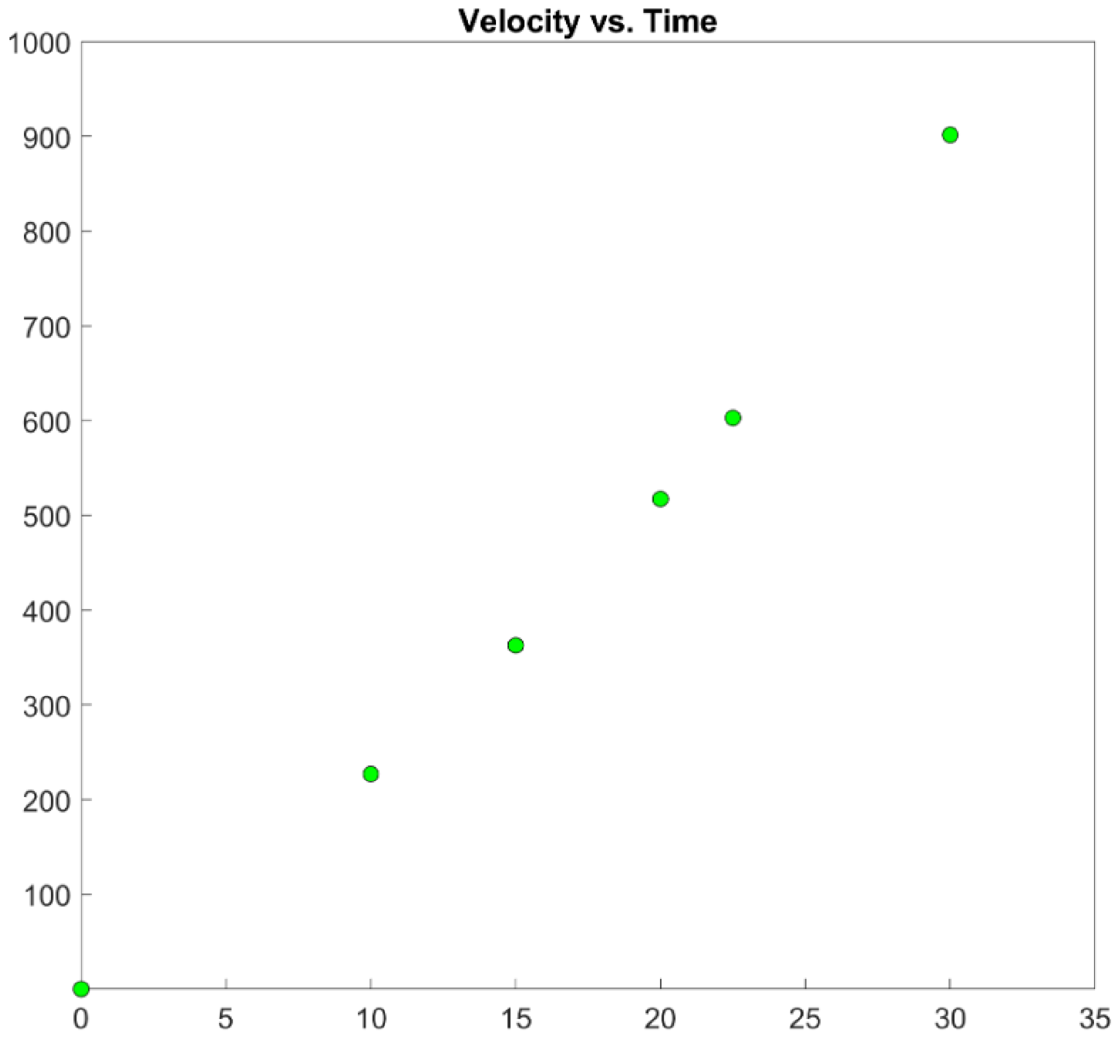


Figure 1. Graph of velocity vs. time data for the rocket example.

a) Determine the value of the velocity at $t = 16$ seconds with second order polynomial interpolation using Lagrangian polynomial interpolation.

Lagrangian Interpolation

A second-order polynomial example

Solution

a) For second order polynomial interpolation (also called quadratic interpolation), the velocity is given by

$$v(t) = \sum_{i=0}^{2} L_i(t) v(t_i)$$
$$= L_0(t)v(t_0) + L_1(t)v(t_1) + L_2(t)v(t_2)$$

Since we want to find the velocity at $t = 16$, and we are using a second order polynomial, we need to choose the three data points that are closest to $t = 16$ that also bracket $t = 16$ to evaluate it. The three points are $t_0 = 10,\ t_1 = 15,$ and $t_2 = 20$.

Then

$$t_0 = 10,\ v(t_0) = 227.04$$
$$t_1 = 15,\ v(t_1) = 362.78$$
$$t_2 = 20,\ v(t_2) = 517.35$$

gives

$$L_0(t) = \prod_{\substack{j=0 \\ j\neq 0}}^{2} \frac{t - t_j}{t_0 - t_j} = \left(\frac{t - t_1}{t_0 - t_1}\right)\left(\frac{t - t_2}{t_0 - t_2}\right)$$

$$L_1(t) = \prod_{\substack{j=0 \\ j\neq 1}}^{2} \frac{t - t_j}{t_1 - t_j} = \left(\frac{t - t_0}{t_1 - t_0}\right)\left(\frac{t - t_2}{t_1 - t_2}\right)$$

$$L_2(t) = \prod_{\substack{j=0 \\ j\neq 2}}^{2} \frac{t - t_j}{t_2 - t_j} = \left(\frac{t - t_0}{t_2 - t_0}\right)\left(\frac{t - t_1}{t_2 - t_1}\right)$$

Lagrangian Interpolation

A second-order polynomial example

Hence

$$v(t) = \left(\frac{t-t_1}{t_0-t_1}\right)\left(\frac{t-t_2}{t_0-t_2}\right)v(t_0) + \left(\frac{t-t_0}{t_1-t_0}\right)\left(\frac{t-t_2}{t_1-t_2}\right)v(t_1) + \left(\frac{t-t_0}{t_2-t_0}\right)\left(\frac{t-t_1}{t_2-t_1}\right)v(t_2),\ t_0 \le t \le t_2$$

$$\boxed{\begin{aligned} v(16) &= \frac{(16-15)(16-20)}{(10-15)(10-20)}(227.04) + \frac{(16-10)(16-20)}{(15-10)(15-20)}(362.78) \\ &\quad + \frac{(16-10)(16-15)}{(20-10)(20-15)}(517.35) \\ &= (-0.08)(227.04) + (0.96)(362.78) + (0.12)(517.35) \\ &= 392.19 \text{ m/s} \end{aligned}}$$

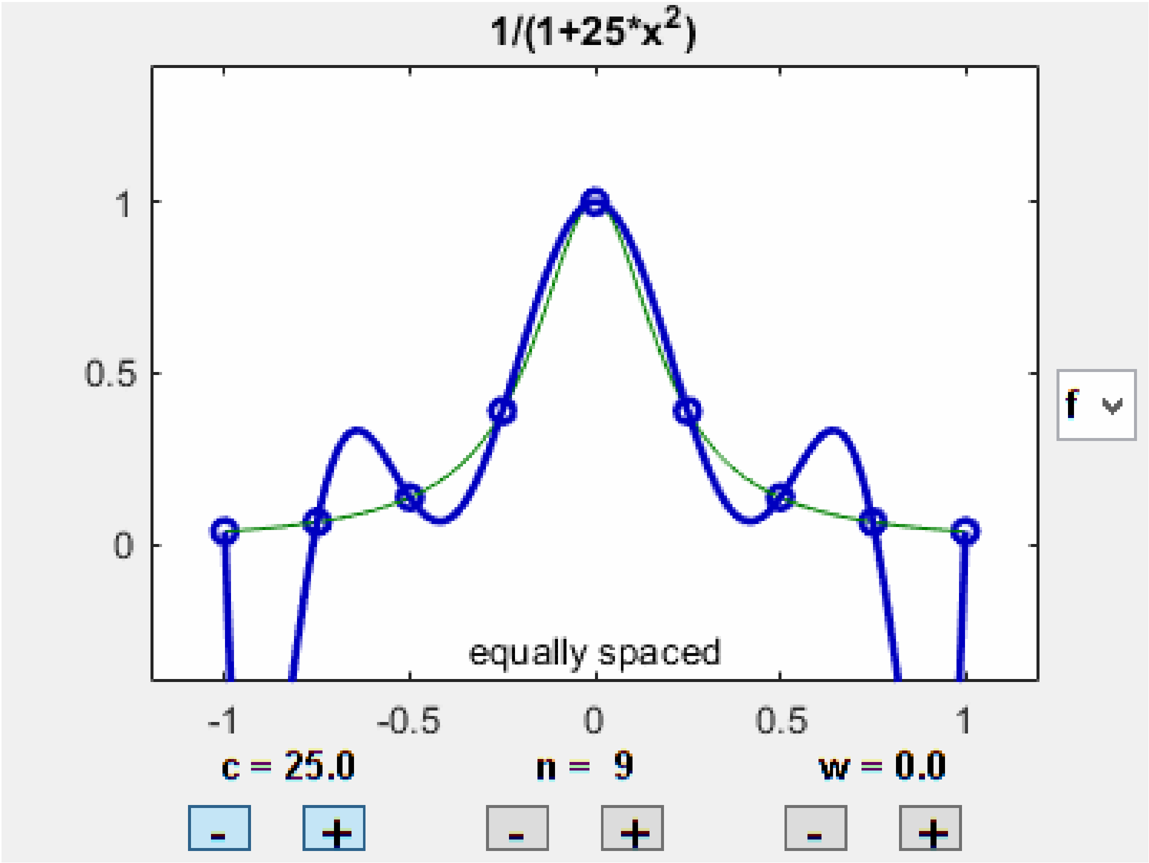
Runge Phenomenon

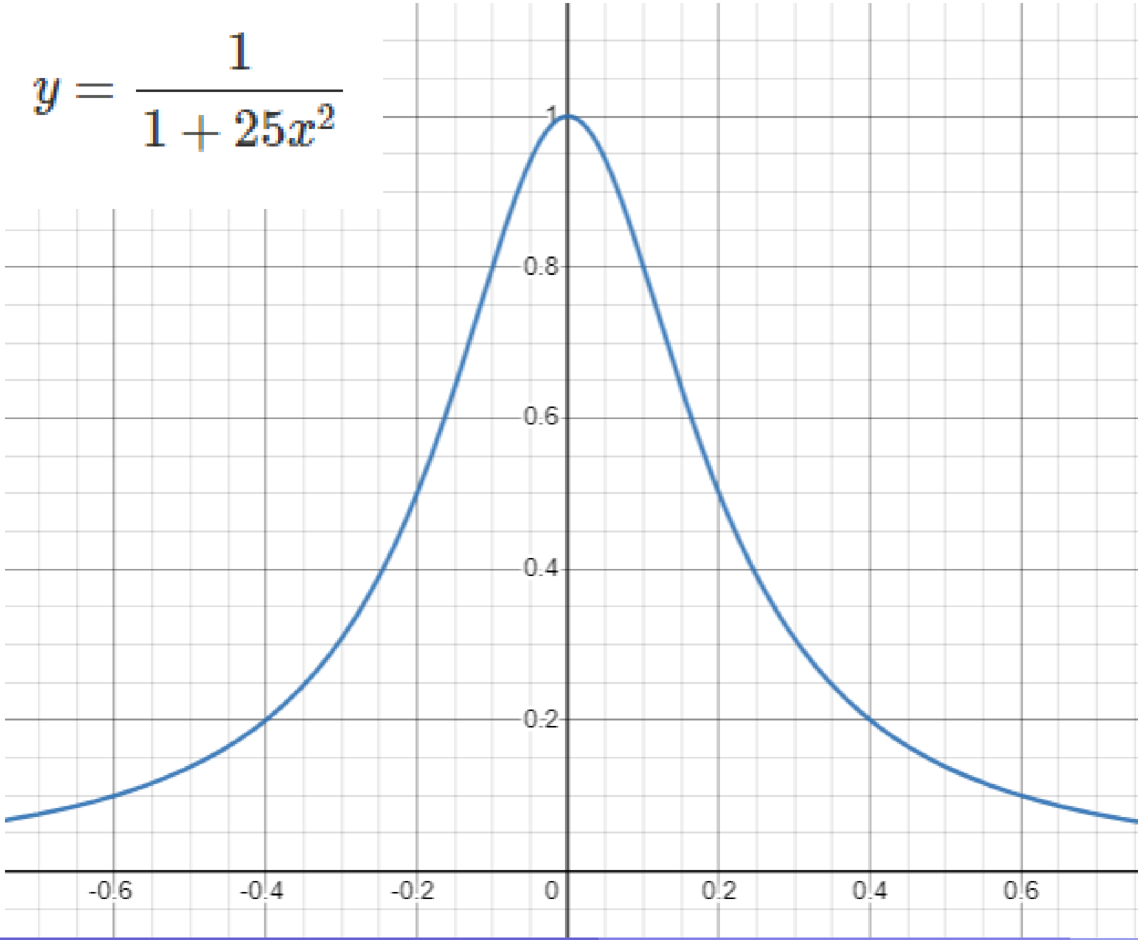
How higher order interpolation becomes a bad idea

Demo Link:
https://demonstrations.wolfram.com/RungesPhenomenon/#embed

When n becomes ***large***, in many cases, one may get ***oscillatory behavior*** in the resulting polynomial.

This was shown by Runge when he interpolated data based on a simple function on an interval of $[-1, 1]$.





Runge Phenomenon

How higher order interpolation becomes a bad idea

Table 1. Six equidistantly spaced points in $[-1, 1]$.

x	$y = \dfrac{1}{1+25x^2}$
-1.0	0.038461
-0.6	0.1
-0.2	0.5
0.2	0.5
0.6	0.1
1.0	0.038461

Now through these six data points, one can pass a fifth-order interpolating polynomial.

$$f_5(x) = 1.2019x^4 - 1.7308x^2 + 0.56731, \quad -1 \le x \le 1$$

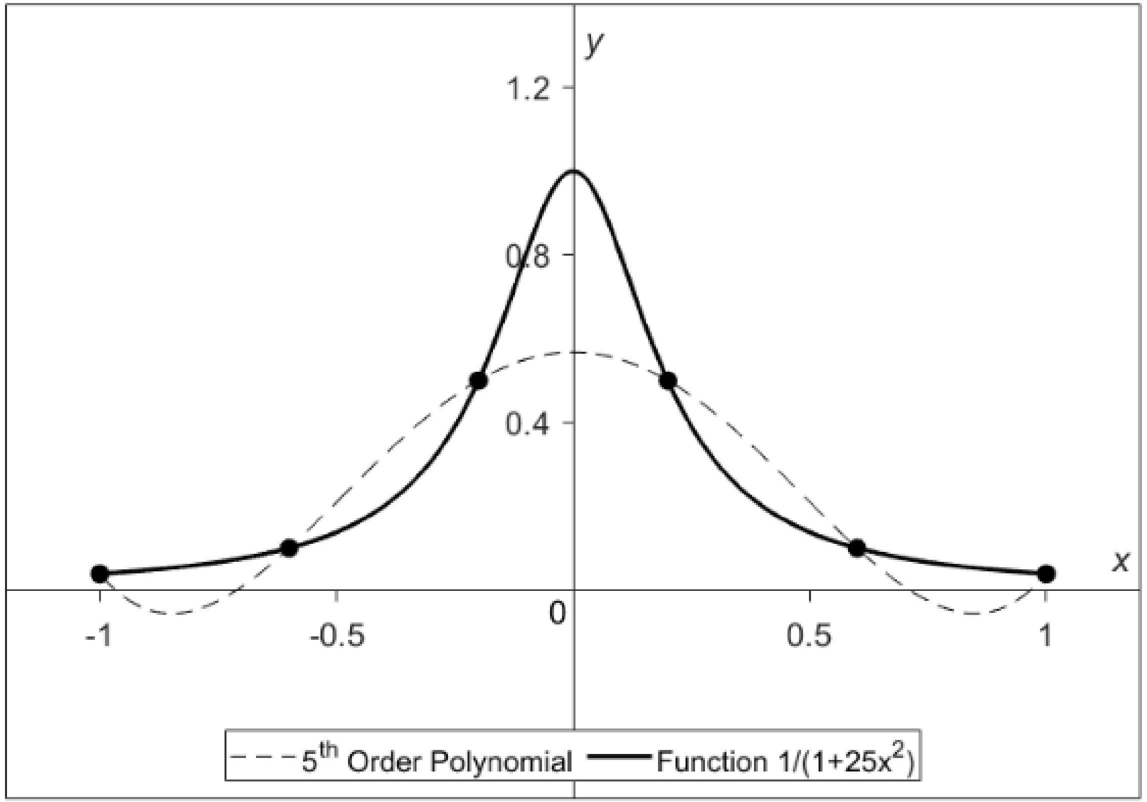


Figure 1. Fifth order polynomial interpolation with six equidistant points.

Runge Phenomenon

How higher order interpolation becomes a bad idea

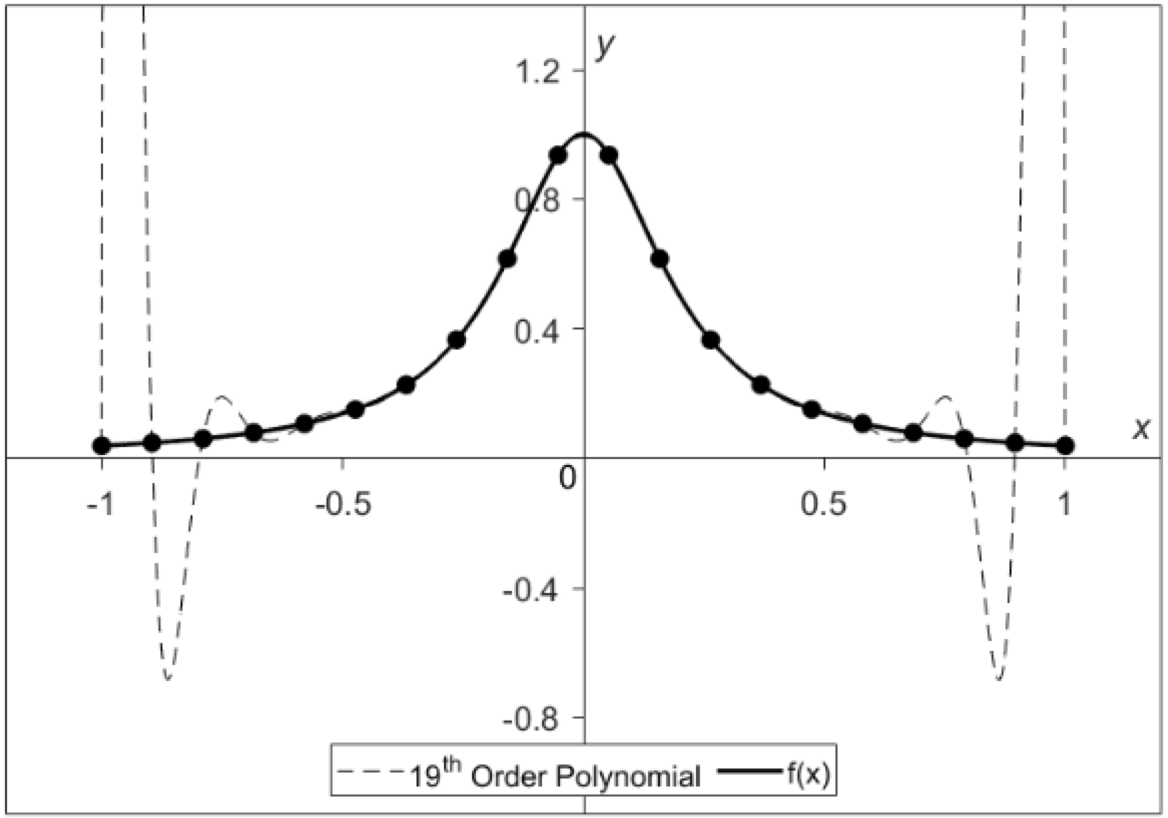
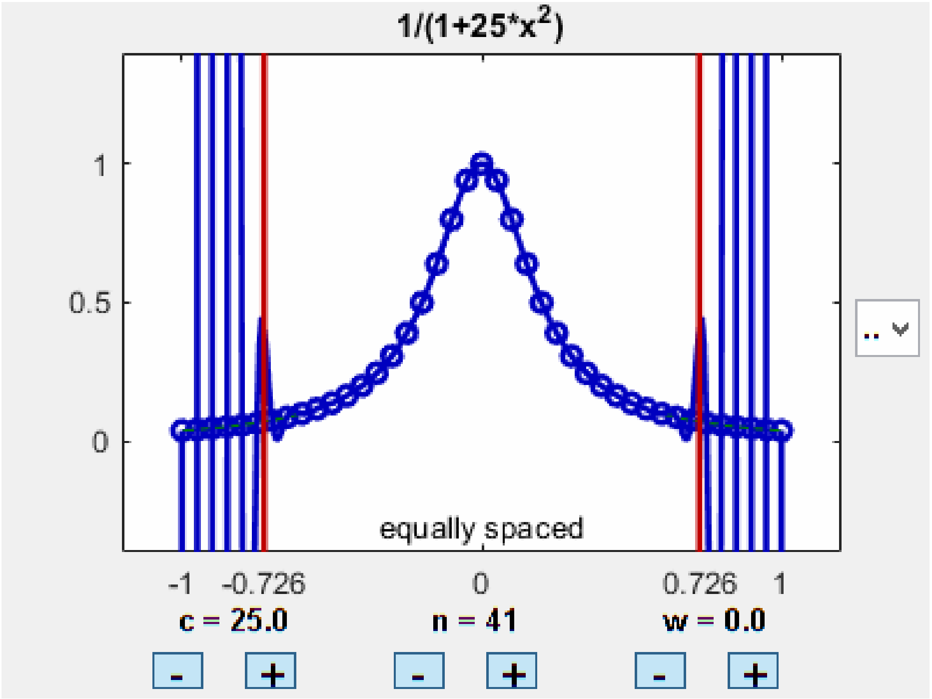


Figure 2. Nineteenth order polynomial interpolation with twenty equidistant points



$$\lim_{n\to\infty}\left(\sup_{-1\leq x\leq 1}|f(x)-P_n(x)|\right)=\infty.$$

It can even be proven that the interpolation error increases (without bound) when the degree of the polynomial is increased.

In fact, Runge found that as the order of the polynomial approaches infinity, the polynomial diverges even more in the interval of $-1 < x < -0.726$ and $0.726 < x < 1$.

Runge Phenomenon

How to ameliorate this?

We need the **Spline Interpolation method**.

Will be discussed in the next class!