Lecture 16

Interpolation Cont.

Owen L. Lewis

Department of Mathematics and Statistics
University of New Mexico

October 22, 2024

Recall

Given $n+1$ distinct points $x_0, \ldots, x_n$, and values $y_0, \ldots, y_n$, there exists a unique polynomial $p(x)$ of degree at most n so that

$$p(x_i) = y_i \quad i = 0, \ldots, n$$

Recall: Monomials

Obvious attempt: try picking

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

To find the a_k's

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

That is,

$$a = V^{-1} y$$

Very bad matrix: terribly ill-conditioned.
Good: Easy to evaluate at any other location (nested):

$$p(x) = a_0 + x(a_1 + x(a_2 \dots x(a_{n-1} + a_n x) \dots)).$$

Recall: Lagrange

The general Lagrange form is

$$\ell_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i} \quad \Rightarrow \ell_k(x_j) = \begin{cases} 1, & \text{if } j = k \\ 0, & \text{if } j \neq k \end{cases}$$

The resulting interpolating polynomial is

$$p(x) = \sum_{k=0}^{n} \ell_k(x) y_k$$

Very good: no need to invert a matrix. Can just build directly out of our data.
Bad: No easy way to evaluate. Must store many polynomials/coefficients.

Resolution: Newton Polynomials

By definition, its nested

$$p(x) = \tilde{a}_0 + \tilde{a}_1(x - x_0) + \tilde{a}_2(x - x_0)(x - x_1) + \tilde{a}_3(x - x_0)(x - x_1)(x - x_2) + \dots$$

We can find an algorithm to compute the coefficients:

$$\tilde{a}_k = f[x_0, \dots x_k].$$

where

$$f[x_i, \dots x_j] = \frac{f[x_i, \dots x_{j-1}] - f[x_{i+1}, \dots x_j]}{x_j - x_i}, \quad f[x_j] = y_j.$$

The very good: More stable that monomials
The good: Almost as computationally efficient (nested evaluation)
The good: Easier to add more data points

Divided Differences

Recursive Property

$$f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}$$

With the first two defined by

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

$$f[x_i, x_j] = \frac{f[x_j] - f[x_i]}{x_j - x_i}$$

Divided Differences

Invariance Theorem

$f[x_0, \dots, x_k]$ is invariant under all permutations of the arguments $x_0, \dots, x_k$

Simple "proof": $f[x_0, x_1, ..., x_k]$ is the coefficient of the x^k term in the polynomial interpolating f at $x_0, \dots, x_k$. But any permutation of the x_i still gives the same polynomial. That is, the order that you consider the interpolation points does not matter.

This says that we can also write

$$f[x_i, \dots, x_j] = \frac{f[x_{i+1}, \dots, x_j] - f[x_i, \dots, x_{j-1}]}{x_j - x_i}$$

Divided Differences

the easy way: tables

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Divided Differences

the easy way: tables

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Divided Differences

the easy way: tables

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Divided Differences

the easy way: tables

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Divided Differences

the easy way: tables

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Divided Differences

the easy way: tables

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0,x_1]$		
x_1	$f[x_1]$		$f[x_0,x_1,x_2]$	
		$f[x_1,x_2]$		$f[x_0,x_1,x_2,x_3]$
x_2	$f[x_2]$		$f[x_1,x_2,x_3]$	
		$f[x_2,x_3]$		
x_3	$f[x_3]$			

Divided Differences

the easy way: tables

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Divided Differences

the easy way: tables

Now just read the coefficients out of your table

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Divided Differences

the easy way: tables

Now just read the coefficients out of your table

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	a_0			
		a_1		
x_1	$f[x_1]$		a_2	
		$f[x_1, x_2]$		a_3
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Notes

To the Board

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.

We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.

We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.

We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.

We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.

We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.

We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.

We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

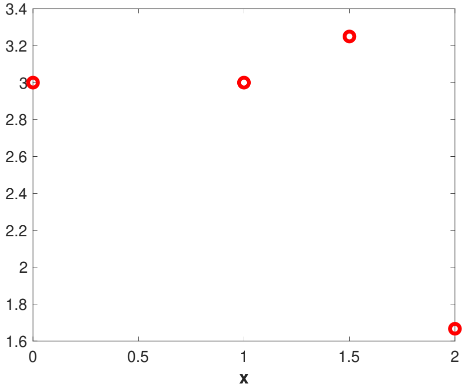
the easy way: example

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

The coeffiecients are readily available and we arrive at

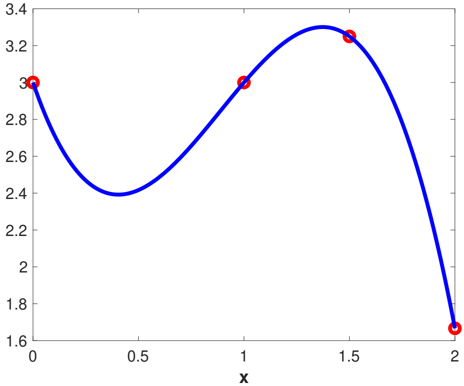
$$p_3(x) = 3 + \frac{1}{2}(x-1) + \frac{1}{3}(x-1)(x-\frac{3}{2}) - 2(x-1)(x-\frac{3}{2})x$$

Newton Polynomial



$$p_3(x) = 3 + \frac{1}{2}(x-1) + \frac{1}{3}(x-1)(x-\frac{3}{2}) - 2(x-1)(x-\frac{3}{2})x$$

Newton Polynomial



$$p_3(x) = 3 + \frac{1}{2}(x-1) + \frac{1}{3}(x-1)(x-\frac{3}{2}) - 2(x-1)(x-\frac{3}{2})x$$

Important

Theorem says there is exactly one polynomial of degree n which interpolates the $n+1$ data points

$$(x_0, y_0), \quad (x_1, y_1), \ldots \quad (x_n, y_n).$$

Monomial, Lagrange, Newton are just three different ways to write/find/store the same polynomial:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

$$p(x) = \sum_{k=0}^{n} y_k \ell_k(x) = \sum_{k=0}^{n} y_k \left(\prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i} \right)$$

$$p(x) = \tilde{a}_0 + \tilde{a}_1(x - x_0) + \tilde{a}_2(x - x_0)(x - x_1) + \tilde{a}_3(x - x_0)(x - x_1)(x - x_2) + \ldots$$

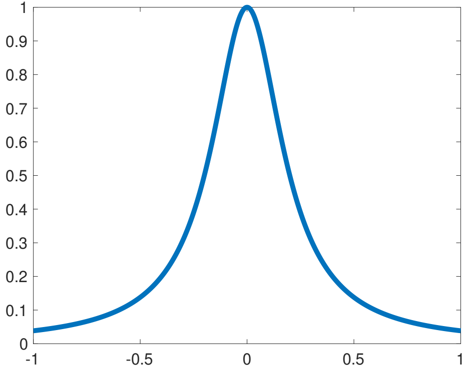
Bad News

Polynomial interpolation can have catastrophic drawbacks.

How bad is polynomial interpolation?

Let's take something very smooth function (Runge's function)

$$f(x) = \frac{1}{1 + 25x^2}$$

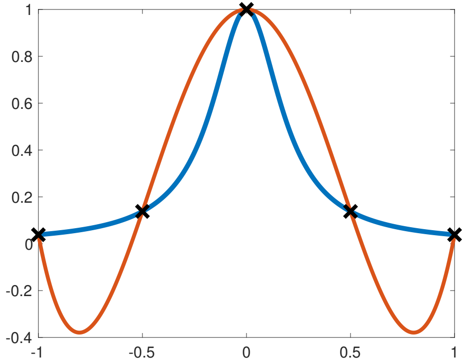


How does interpolation behave?

How bad is polynomial interpolation?

Let's take something very smooth function (Runge's function)

$$f(x) = \frac{1}{1 + 25x^2}$$

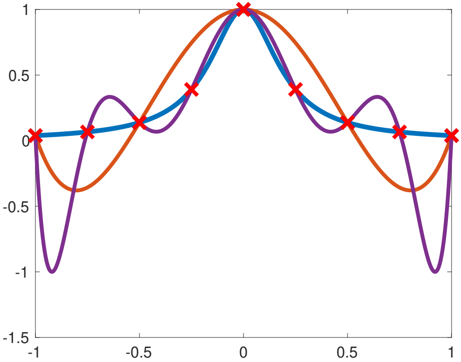


Five equally spaced points doesn't do well.

How bad is polynomial interpolation?

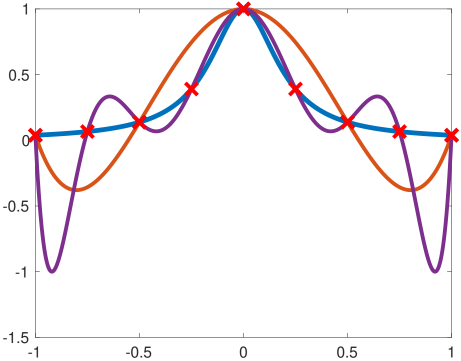
Let's take something very smooth function (Runge's function)

$$f(x) = \frac{1}{1 + 25x^2}$$



Nine equally spaced points is even worse!

How bad is polynomial interpolation?



Can show that, when using equispaced data points (for this f),

$$\lim_{n\to\infty}\left(\max_{-1\leqslant x\leqslant 1}|f(x)-p_n(x)|=\infty\right)$$

Notes

To the board!

Some analysis...

What can we say about

$$e(t) = f(t) - p(t)$$

at some point x? Consider $n = 1$: linear interpolation of a function at x_0 & x_1

- want: error at x, $e(x)$
- look at

$$g(t) = e(t) - \frac{(t - x_0)(t - x_1)}{(x - x_0)(x - x_1)} e(x)$$

- $g(t)$ is 0 at $t = x_0, x_1, x$
- so $g'(t)$ is zero at two points
- so $g''(t)$ is zero at one point, call it c

Some analysis...

We know $g''(c) = 0$ for some point c:

$$
\begin{aligned}
0 = g''(c) = e''(c) - 2\frac{e(x)}{(x - x_0)(x - x_1)} \\
= f''(c) - 2\frac{e(x)}{(x - x_0)(x - x_1)} \\
e(x) = \frac{(x - x_0)(x - x_1)}{2} f''(c)
\end{aligned}
$$

Analysis of Interpolation Error: General result

Theorem: Interpolation Error I

If $p_n(x)$ is the (at most) n degree polynomial interpolating $f(x)$ at $n+1$ distinct points and if $f^{(n+1)}$ is continuous, then

$$e(x) = f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(c) \prod_{i=0}^{n} (x - x_i)$$

Can you explain the problem with Runge's function with this theorem?

Analysis of Interpolation Error: Equispaced Points

Theorem: Bounding Lemma

Suppose x_i are equispaced in $[a, b]$ for $i = 0, \ldots, n$. Then

$$\prod_{i=0}^{n} |x - x_i| \leqslant \frac{h^{n+1}}{4} n!$$

(here, $h = x_i - x_{i-1}$ is the distance between our points).

Theorem: Interpolation Error II

Let $|f^{(n+1)}(x)| \leqslant M$, then with the above,

$$|f(x) - p_n(x)| \leqslant \frac{Mh^{n+1}}{4(n+1)}$$

Fixes

We have two options:

1. move the nodes: Chebychev nodes
2. piecewise polynomials (splines)

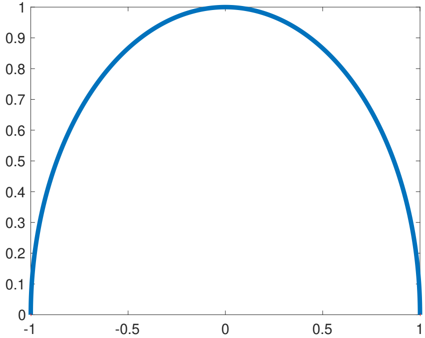
Option #1: Chebychev nodes in $[-1, 1]$

$$x_i = cos\left(\pi\frac{2i+1}{2n+2}\right), \quad i = 0, \dots, n$$

Option #2: piecewise polynomials...

Chebychev Nodes

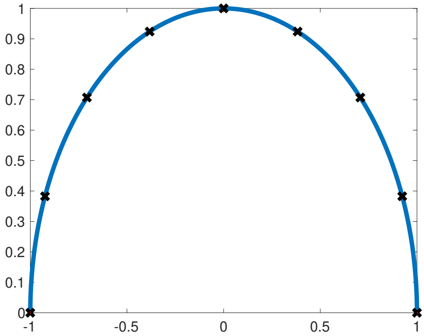
$$x_i = cos\left(\pi\frac{2i+1}{2n+2}\right), \quad i = 0,\ldots,n$$



Start with a semi-circle above the interval.

Chebychev Nodes

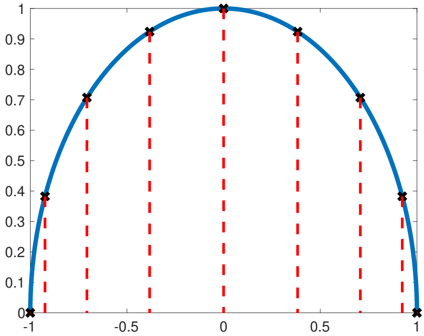
$$x_i = cos\left(\pi \frac{2i+1}{2n+2}\right), \quad i = 0, \ldots, n$$



Equally space points on that circle.

Chebychev Nodes

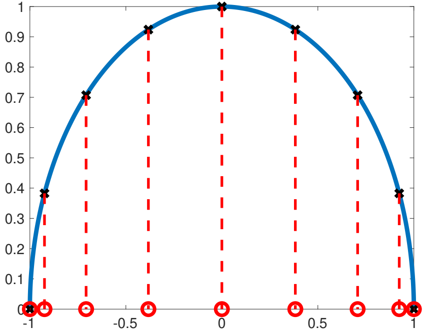
$$x_i = cos\left(\pi\frac{2i+1}{2n+2}\right), \quad i = 0, \ldots, n$$



Project down to the x axis

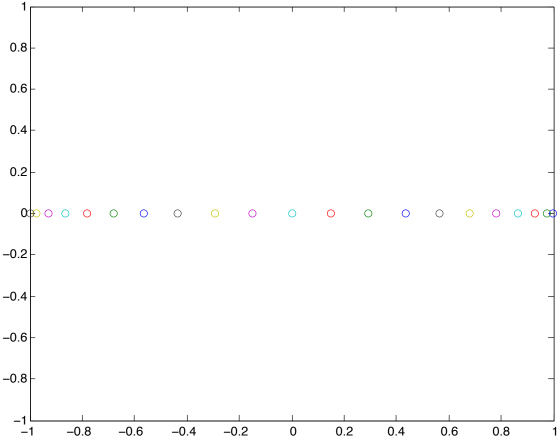
Chebychev Nodes

$$x_i = cos\left(\pi\frac{2i+1}{2n+2}\right), \quad i = 0, \ldots, n$$



Use *those* as your interpolation points.

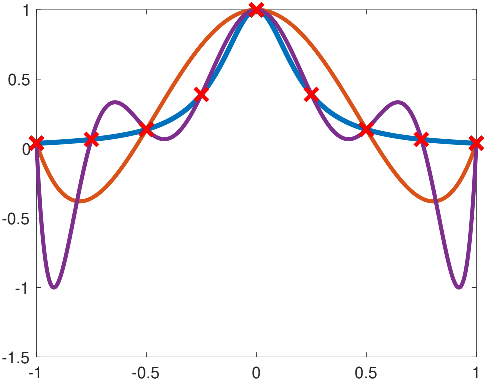
Chebychev Nodes



- Can obtain nodes from equidistant points on a circle projected down
- Nodes are non uniform and non nested

Chebychev Nodes

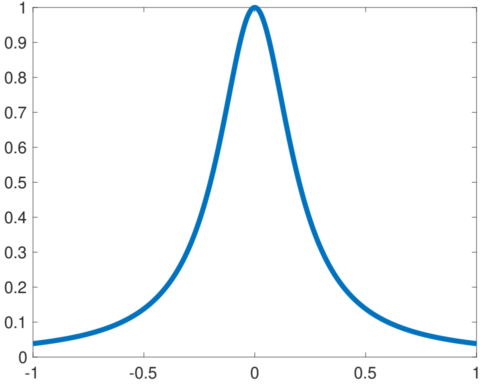
High degree polynomials using equispaced points suffer from many oscillations



- Worst behavior is near the end points ("Runge Phenomenon").
- Things get worse as we add more (equally spaced) points and make higher order polynomial.

Chebychev Nodes

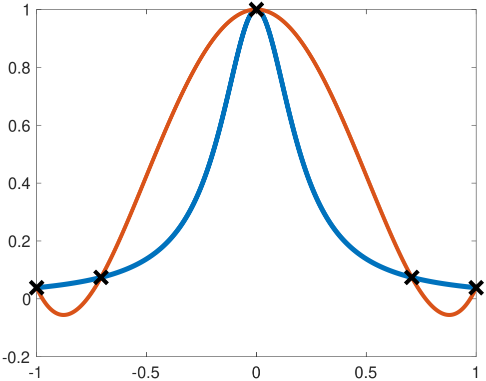
High degree polynomials using equispaced points suffer from many oscillations



- Chebyshev bunches the points towards the ends of the interval
- This ”ties” the function down at the ends, and the error is distributed more evenly

Chebychev Nodes

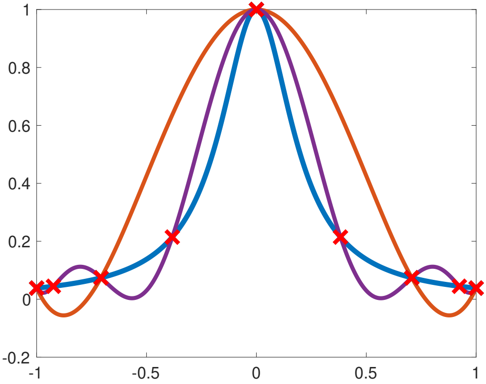
High degree polynomials using equispaced points suffer from many oscillations



- Chebyshev bunches the points towards the ends of the interval
- This ”ties” the function down at the ends, and the error is distributed more evenly

Chebychev Nodes

High degree polynomials using equispaced points suffer from many oscillations



- Chebyshev bunches the points towards the ends of the interval
- This ”ties” the function down at the ends, and the error is distributed more evenly

Why not Chebychev?

Chebychev points are "optimal" in that they minimize Runge phenomenon as n increases.

Unfortunately this presumes *we* get to choose where our "data" points lie on the x-axis.