CS 3200
Introduction to Scientific Computing

Instructor: Martin Berzins

Topic: **Interpolation**

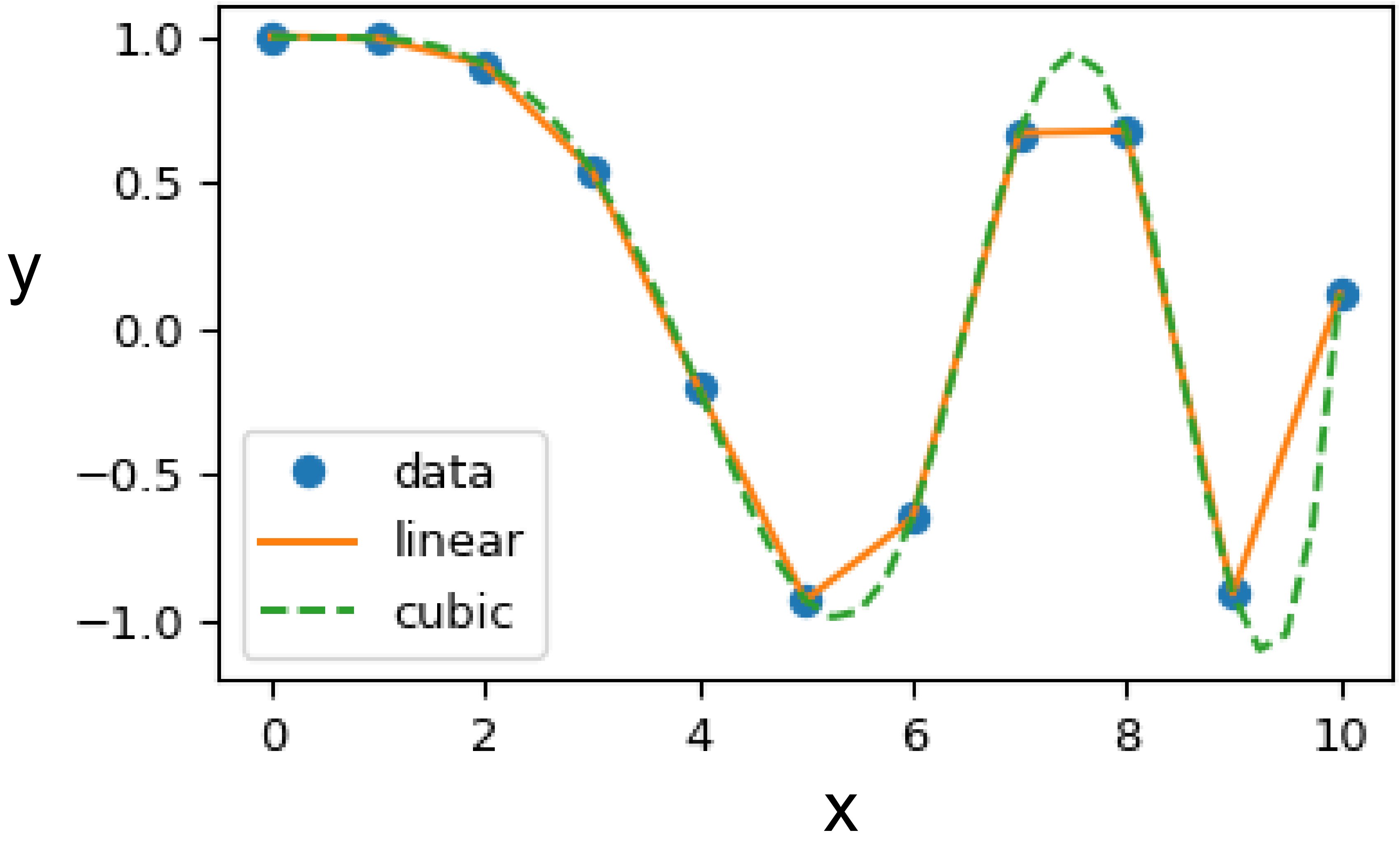
What is Interpolation?

Given

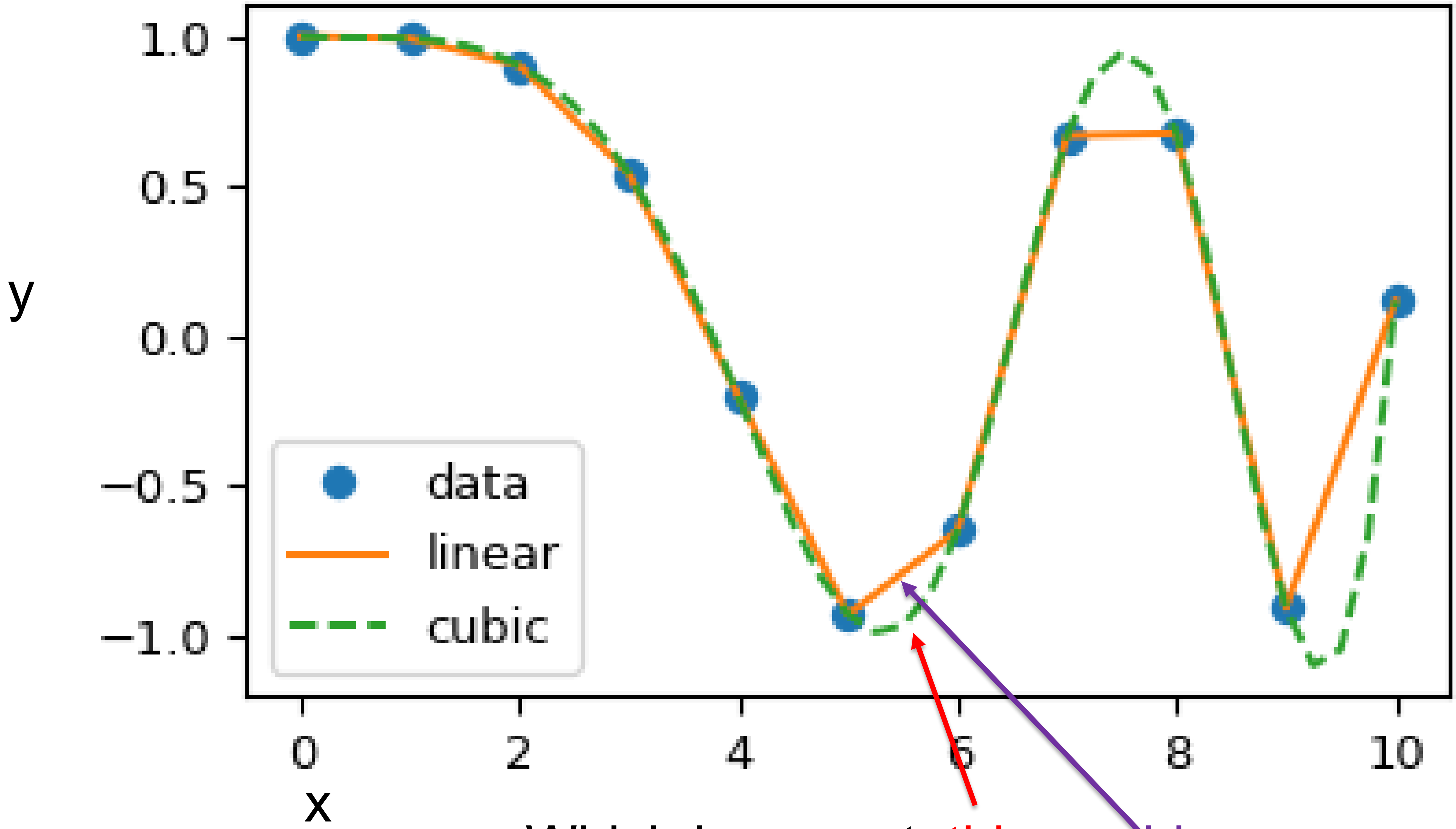
$\{(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)\}$

, find the value of y (or its gradient, derivative)

at a value of x that is not explicitly given

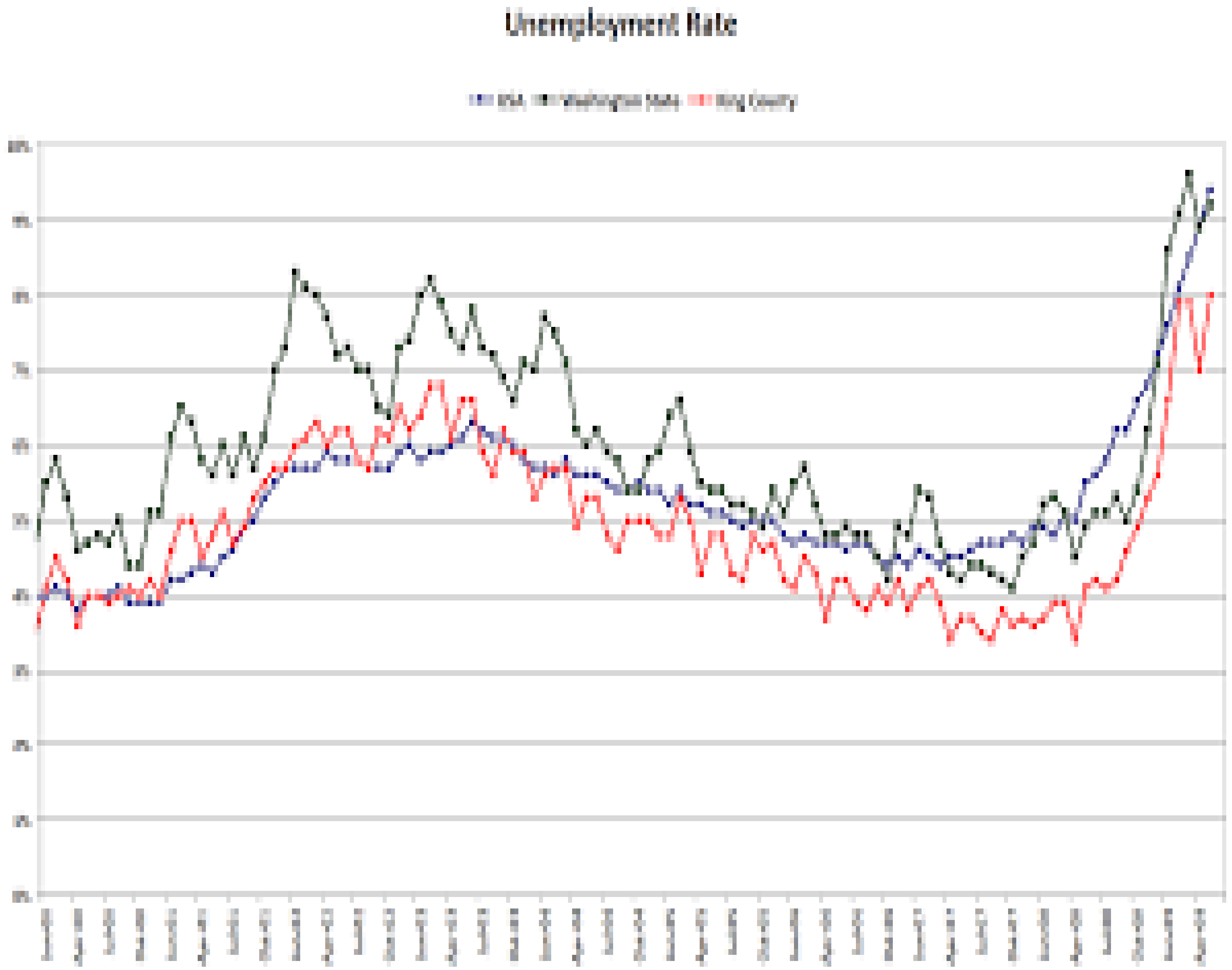


What is Interpolation?

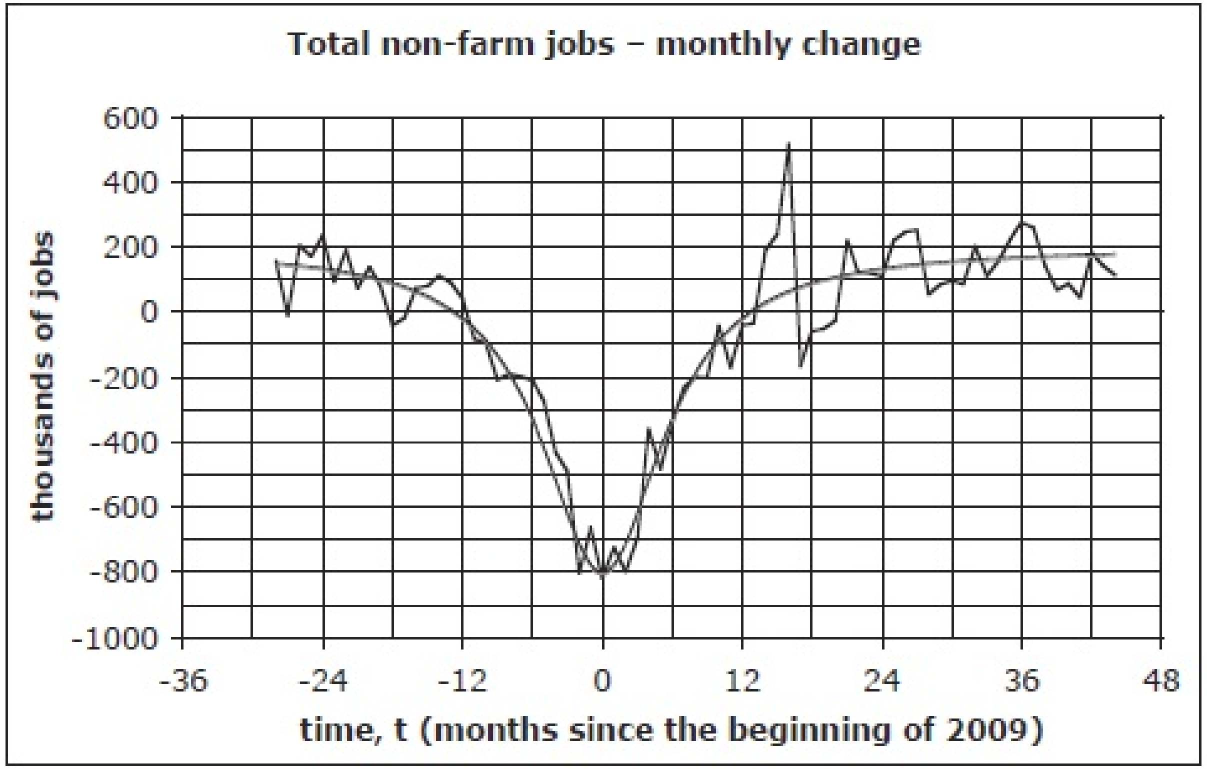


Which is correct this or this

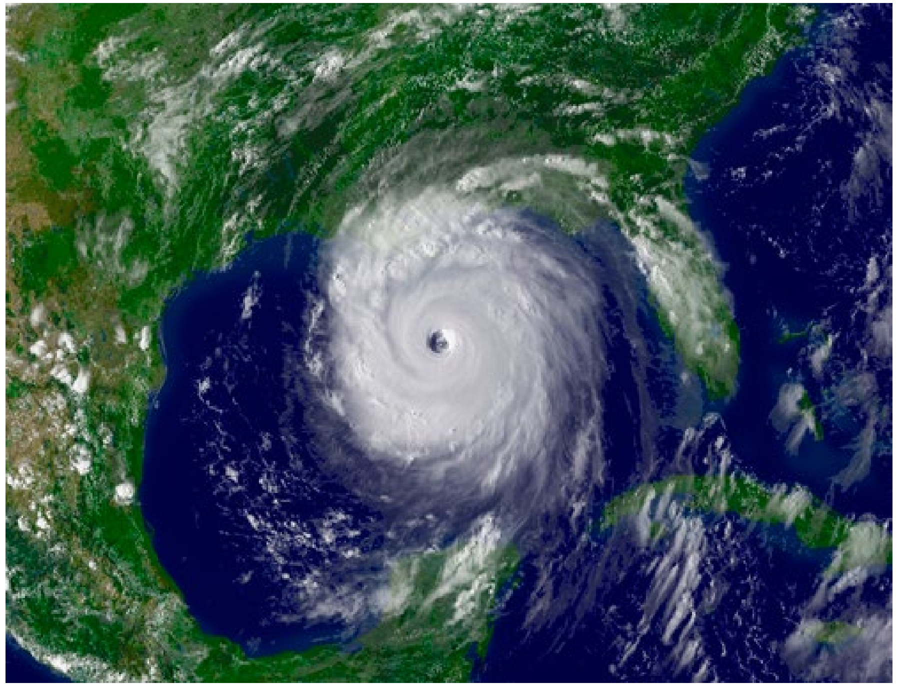
Real Data is often not smooth



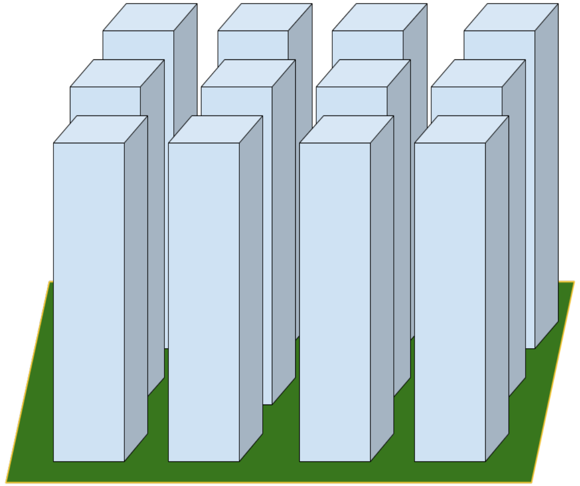
However we do use smooth fine-scale approximations at least to the level of floating point arithmetic



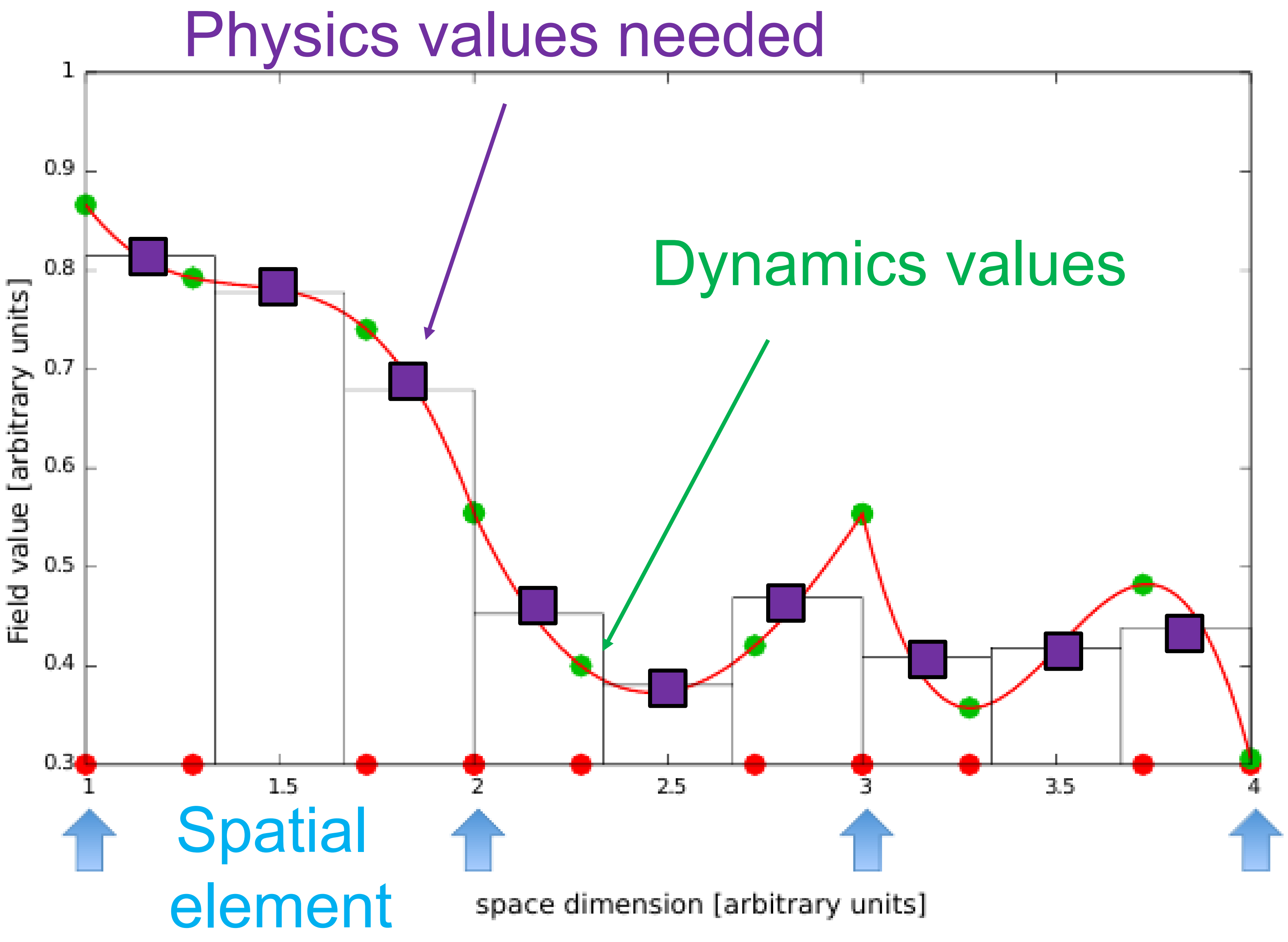
Interpolation in Weather Codes

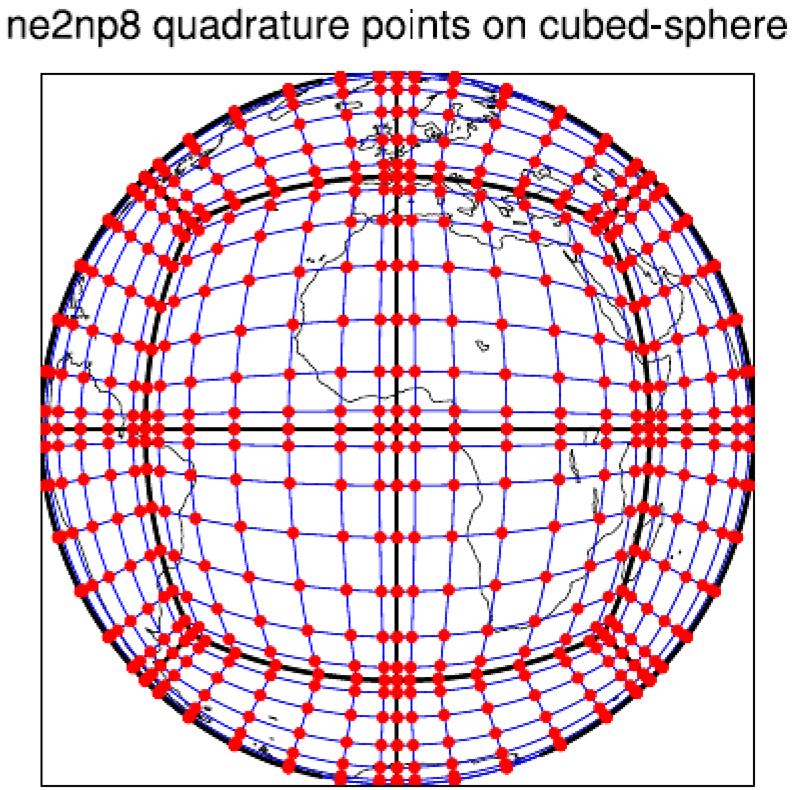


- Recent codes like Neptune split into dynamics and physics part
- Physics deals with rain snow grauple
- Physics considers vertical columns
- Physics routines need to be evenly spaced.
- Dynamics routines used a different mesh
- Need to map physical quantities between meshes
- Density velocities in x,y, and z and temperature are mapped back and forth



Weather Interpolation Problem

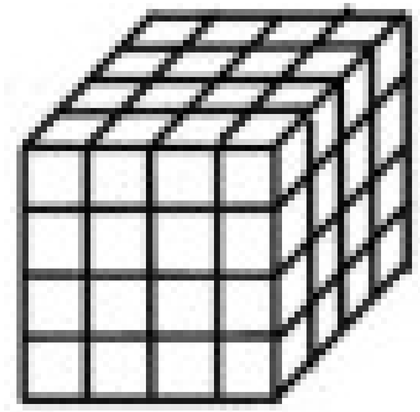


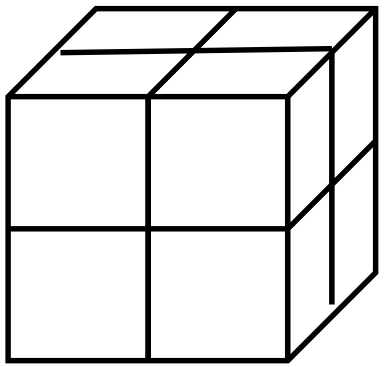


Need to use green values To find purple values

Algorithm-Based Fault Tolerance

- The largest machines today have minor faults about once a day and may have a node crash once a week.
- Approaches to deal with this may involve hardware, system software or algorithmic resilience.
- In the last case we may store a coarse copy of a mesh patch on another node. In three dimensions a mesh patch with ½ the variables in each dimension require only 1/8 of the storage.
- In order to rebuild the fine mesh patch from the coarse one we need to use interpolation that gives physically meaningful values..





Interpolation and approximation

1 Polynomial $y_I(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$

As there are n+1 degrees of freedom in there is one and only one n^{th}-order polynomial that fits n+1 points.

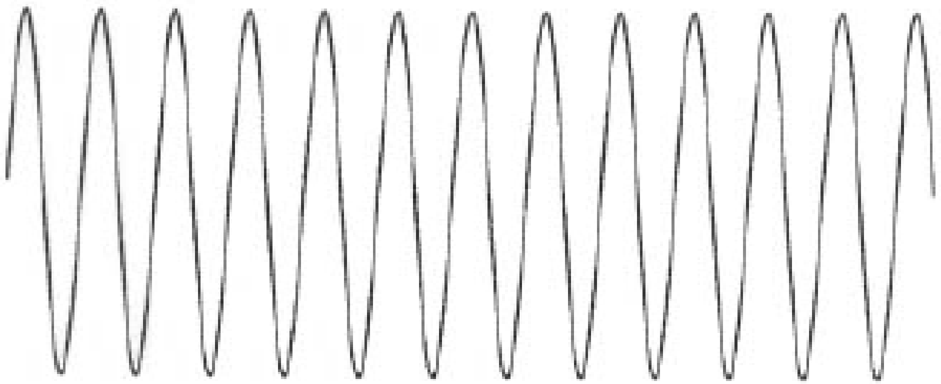
2 Rational Functions

$$y_I(x) = \frac{a + bx + cx^2 + dx^3 + ex^4}{f + gx + hx^2 + kx^3 + ex^4}$$

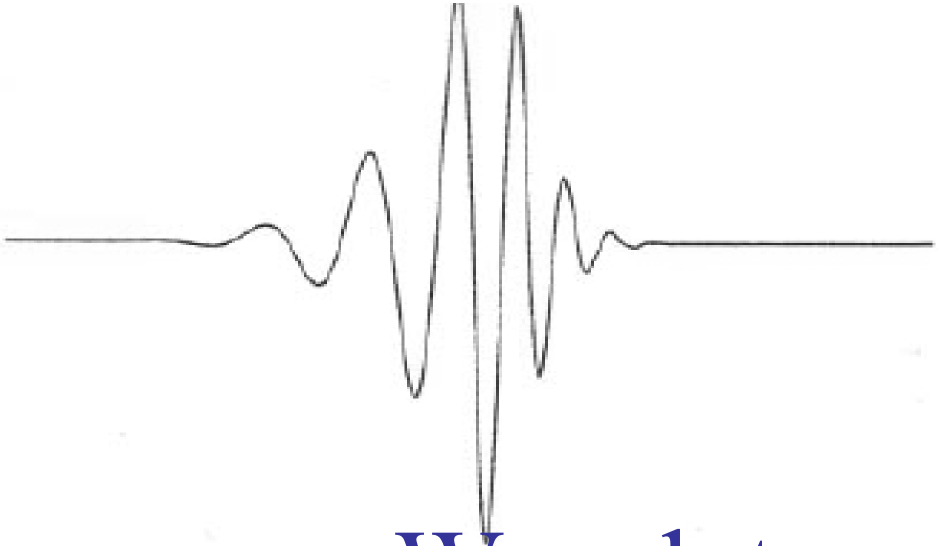
3 Trigonometric interpolants,

$$y_I(x) = a_{-n} e^{-inx} + \ldots + a_0 + \cdots + a_n e^{inx}$$

4. Wavelets alternate to Fourier methods



Sinusoid



Wavelet

5. Radial Basis Functions – used in Neural Nets

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{n_b} b_i \xi_i(\mathbf{x})$$

Radial basis functions $\xi_i(\mathbf{x}) = e^{-(\|\mathbf{x}-\mathbf{x}_i\|/\lambda)^2}$

Polynomial Interpolation Algorithm

- Decide on a set of functions (polynomials) that you will use to present the solution to the problem
- Pick a set of points at which these polynomials are defined $y(x_i)$
- Use interpolation based on this polynomial to recover the solution everywhere

Why Polynomials?

Weierstrass Approximation Theorem (1885).

Suppose f **is a continuous real-valued function** defined on the real interval $[a, b]$. For every $\varepsilon > 0$, **there exists a polynomial $p(x)$** such that for all x in $[a, b]$,

the max value $| f(x) - p(x) | < \varepsilon$.

More formally $\| f(x) - p(x) \|_\infty < \varepsilon$

What is problematic about this theorem?

Why Polynomials?

Weierstrass Approximation Theorem (1885).

Suppose *f* **is a continuous real-valued function** defined on the real interval [*a*, *b*]. For every $\varepsilon > 0$, **there exists a polynomial *p*(*x*)** such that for all *x* in [*a*, *b*],

the max value | *f* (x) – *p(x)* | < *ε*.

More formally $\| f(x) - p(x) \|_\infty < \varepsilon$

What is problematic about this theorem?

Bernstein proved a constructive version of this theory in 1912

Using Taylors Series Directly?

Example – consider approximating the function f(x) = 1/x close to x = 1 using Taylors series

$$f(x) = f(1) + (x-1)\frac{df}{dx}(1) + \frac{(x-1)^2}{2}\frac{d^2 f}{dx^2}(1) + \frac{(x-1)^3}{3!}\frac{d^3 f}{dx^3}(1)$$

$$+ \frac{(x-1)^4}{4!}\frac{d^4 f}{dx^4}(1) + ...$$

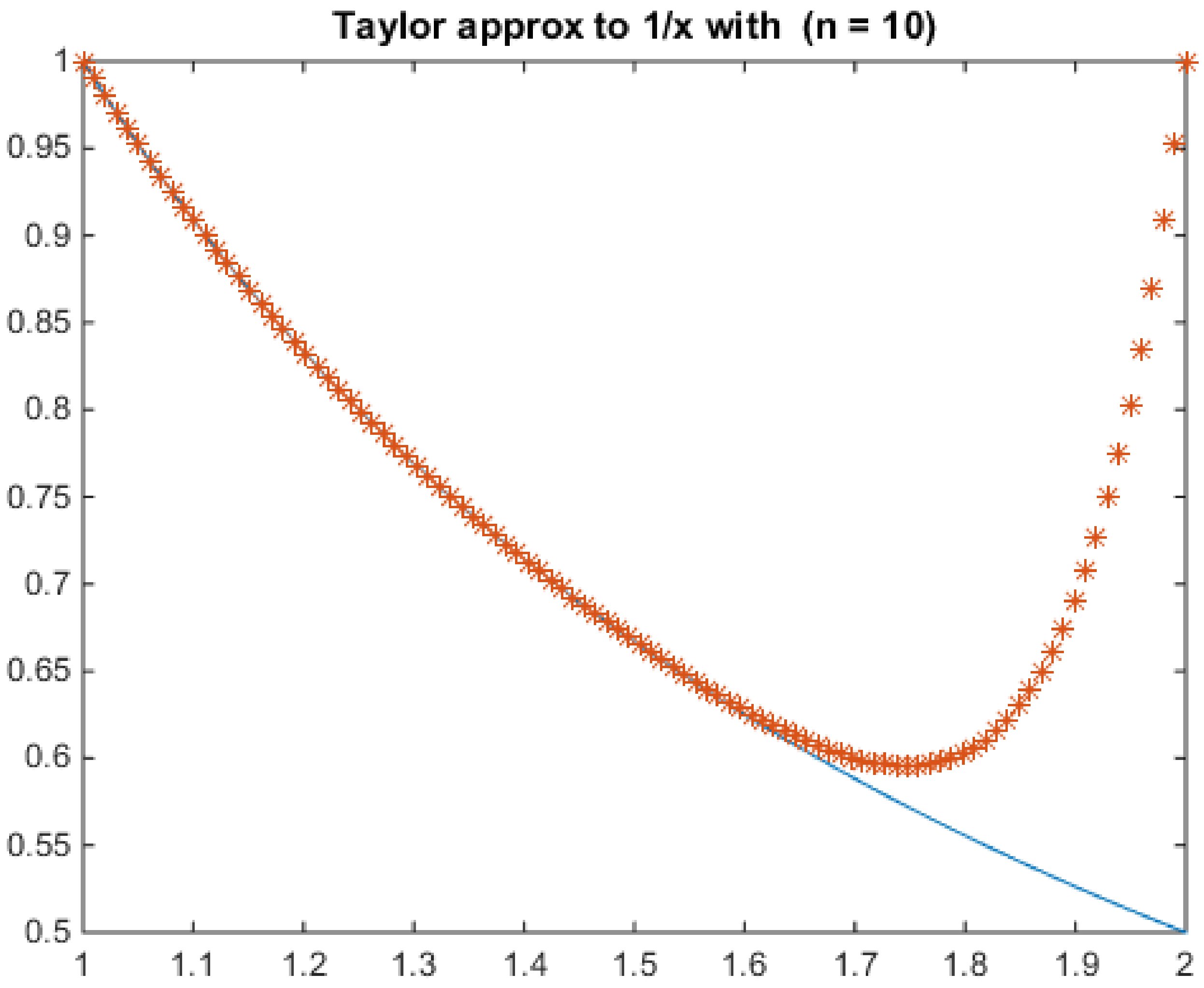
Using Taylors Series Directly?

Example – consider approximating the function f(x) = 1/x close to x = 1 using Taylors series

$$f(x) = f(1) - (x-1) + (x-1)^2 - (x-1)^3 + (x-1)^4 - (x-1)^5 + ...$$

Using Taylors Series Directly ?

- Example – consider approximating the function 1/x close to x = 1 using Taylors series with n = 10.
- Even if we make n much larger there are still problems close to 2
- The answer is completely wrong
- Why?



```
% x = linspace(1.95,2.050,100)';
 x = linspace(1.00,1.5,100)';
 y = ones(100,1)./(x);
 n=4;
 for i = 1:100
   ytaylor(i) = 1.0;
   term = (x(i)-1.0);
   for j = 1:n
       ytaylor(i)  = ytaylor(i)- term;
       term = term * (-1.0*(x(i)-1));
   end
   error(i) = y(i)- ytaylor(i);
 end
```

```
plot(x,y,x,ytaylor,'*',x,error,'+')
 legend('y','ytaylor','error')
 title(sprintf('Taylor approx to 1/x with (n = %2.0f)',n))
```

Coefficients of an Interpolating Polynomial

- Since $n+1$ data points are required to determine $n+1$ coefficients, simultaneous linear systems of equations can be used to calculate "a"s:

$$y(x_0) = a_0 + a_1 x_0 + a_2 x_0^2 \cdots + a_n x_0^n$$

$$y(x_1) = a_0 + a_1 x_1 + a_2 x_1^2 \cdots + a_n x_1^n$$

$$\vdots$$

$$y(x_n) = a_0 + a_1 x_n + a_2 x_n^2 \cdots + a_n x_n^n$$

where "x"s are the knowns and "a"s are the unknowns

We can represent this system of equations as

O(n^3) operations to solve linear system

$$\begin{pmatrix} 1, x_0, x_0^2, x_0^3, \ldots., & x_0^{n-1} & x_0^n \\ 1, x_1, x_1^2, x_1^3, \ldots. & x_1^{n-1} & x_1^n \\ \vdots & \vdots & \vdots \\ 1, x_n, x_n^2, x_n^3, \ldots & x_n^{n-1} & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}$$

and our solution will be the polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + \ldots + a_n x^n$$

Matlab Vandermonde

```
v = 1:.5:3
v = 1×5
1.0000 1.5000 2.0000 2.5000 3.0000
```

Find the alternate form of the Vandermonde matrix using fliplr.

```
A = fliplr(vander(v))

A = 5×5

 1.0000 1.0000 1.0000 1.0000 1.0000
 1.0000 1.5000 2.2500 3.3750 5.0625
 1.0000 2.0000 4.0000 8.0000 16.0000
 1.0000 2.5000 6.2500 15.6250 39.0625
 1.0000 3.0000 9.0000 27.0000 81.0000
```

Example

Polynomial of degree 2 through $(-2,-27),(0,-1),(1,0)$

$$\begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{pmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = y$$

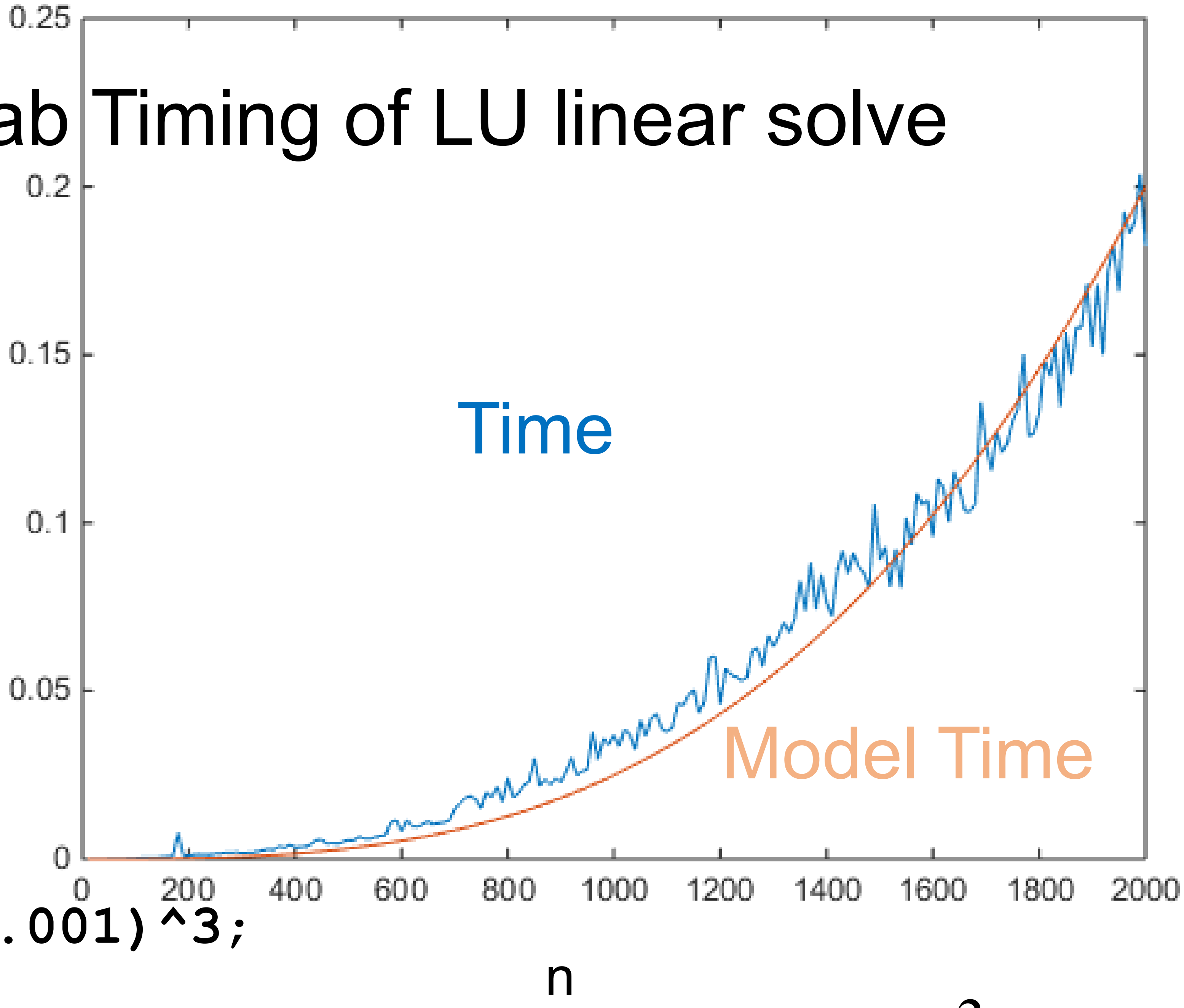
In this case

$$\begin{pmatrix} 1 & -2 & 4 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -27 \\ -1 \\ 0 \end{bmatrix}$$ with solution $$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 5 \\ 4 \end{bmatrix},$$

giving polynomial $-1+5x+4x^2$

```
clear all
t = zeros(1,200);
nt = zeros(1,200);
model = zeros(1,200);
for m = 1:200
    n = m*10;
    A = rand(n,n);
    b = rand(n,1);
    tic;
    x = A\b;
    t(m) = toc;
    nt(m) = n;
    model(m) = 0.25e-1*(n*0.001)^3;
    ratio = model(m)/t(m);
end
plot(nt,t,nt,model)
```

Matlab Timing of LU linear solve



$$O(n^3)$$

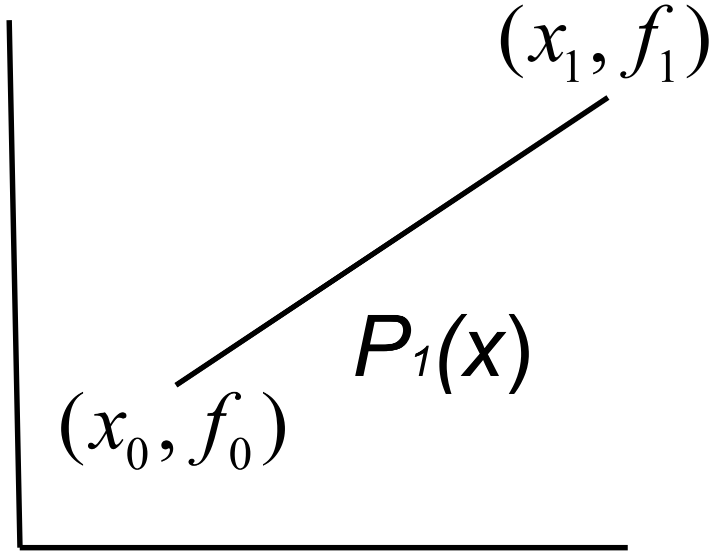
Vandermonde systems are notorious for having challenges with rounding errors, but special algorithms are availabl. These are still N^3.

Can we get a faster algorithm that doesn’t involve solving a set of equations ?

Yes if we explicitly break up the polynomial into simpler parts. Eventually we can get an O(N) algorithm using a special point set and Barycentric interpolation.

Linear Interpolation

- Given f($\boldsymbol{x_0}$) $= \mathrm{f}_{\mathbf{0}}$ and f($\boldsymbol{x_1}$) $= \mathrm{f}_{\mathbf{1}}$
  - Approximate the unknown $\mathrm{y}(x)$ with $p_1(x)$
  - Construct two linear interpolants each of which is zero at one point and one at the other point
  - $L_1(x) = \frac{(x-x_0)}{(x_1-x_0)}, L_0(x) = \frac{(x-x_1)}{(x_0-x_1)}$
  - Then $P_1(x) = L_0 f_0 + L_1 f_1$



Quadratic Lagrange Interpolating Polynomials

The quadratic case is similar construct three polynomials

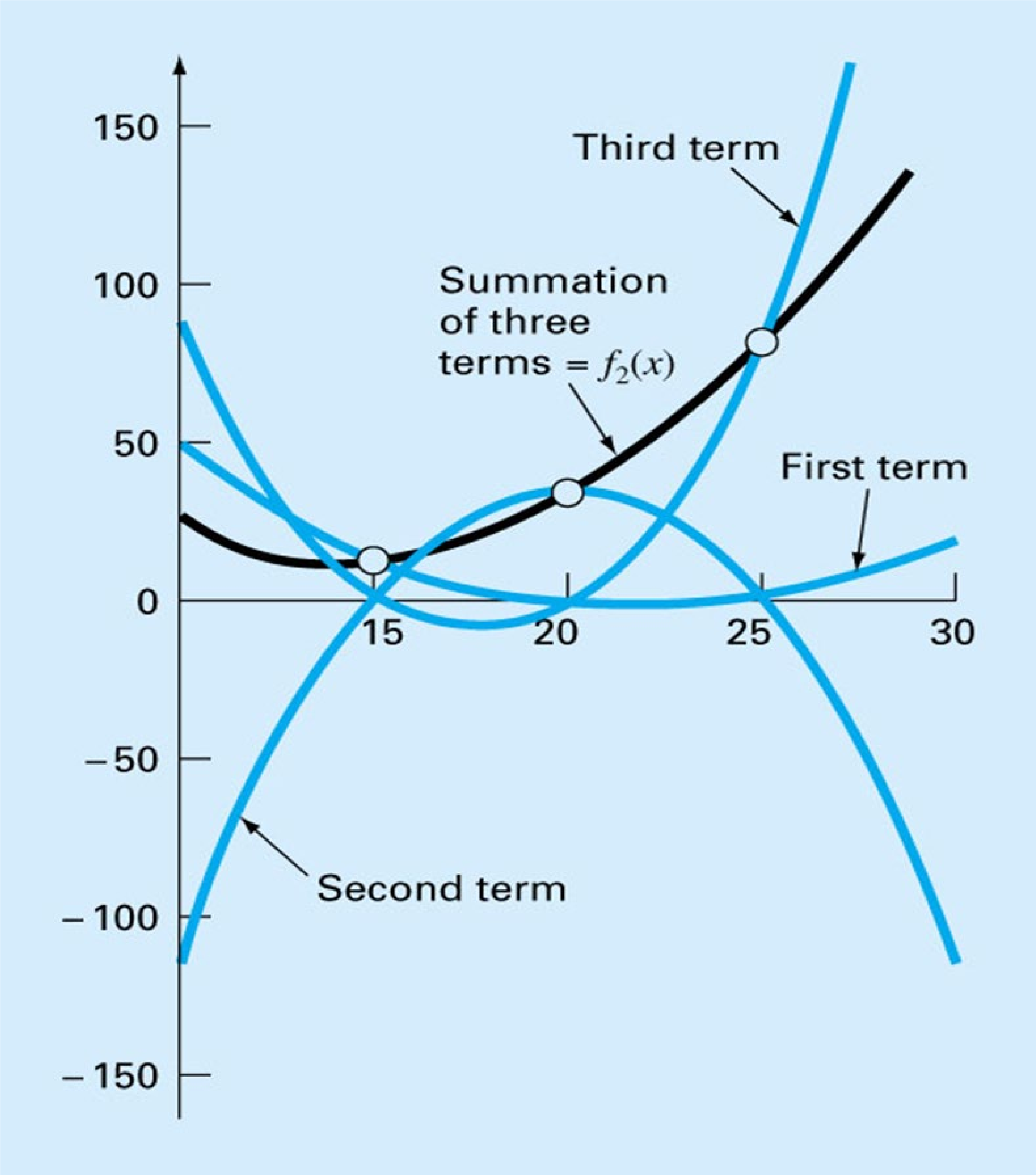
$L_0(x), L_1(x), L_2(x)$

Each of which has value one at one point and is zero at the other two points

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

$$f_2(x) = L_0(x) \quad f(x_0) + \quad L_1(x) \quad f(x_1) + \quad L_2(x) \quad f(x_2)$$

Quadratic Lagrange Interpolants



Lagrange Interpolation

- Given multiple points $(x_i, f(x_i))$
  - Calculate a polynomial interpolant in the same way

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

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_0)(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)}$$

Error of Polynomial Interpolation

Theorem

For an n^{th}-order interpolating polynomial, the error is given by the $(n+1)^{th}$ derivative of the unknown function evaluated at an unknown point ξ

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n)$$

x is located in some interval containing the unknown and the data: $[x, x_0, x_1, ..., x_n]$

Limitation of interpolating polynomials

Runge's phenomenon.

When approximating the function f(x) on [a,b] by an interpolating polynomial with evenly spaced points , an error does not necessary decrease as increase the degree of polynomial. The interpolation oscillates violently at the ends of the interval,

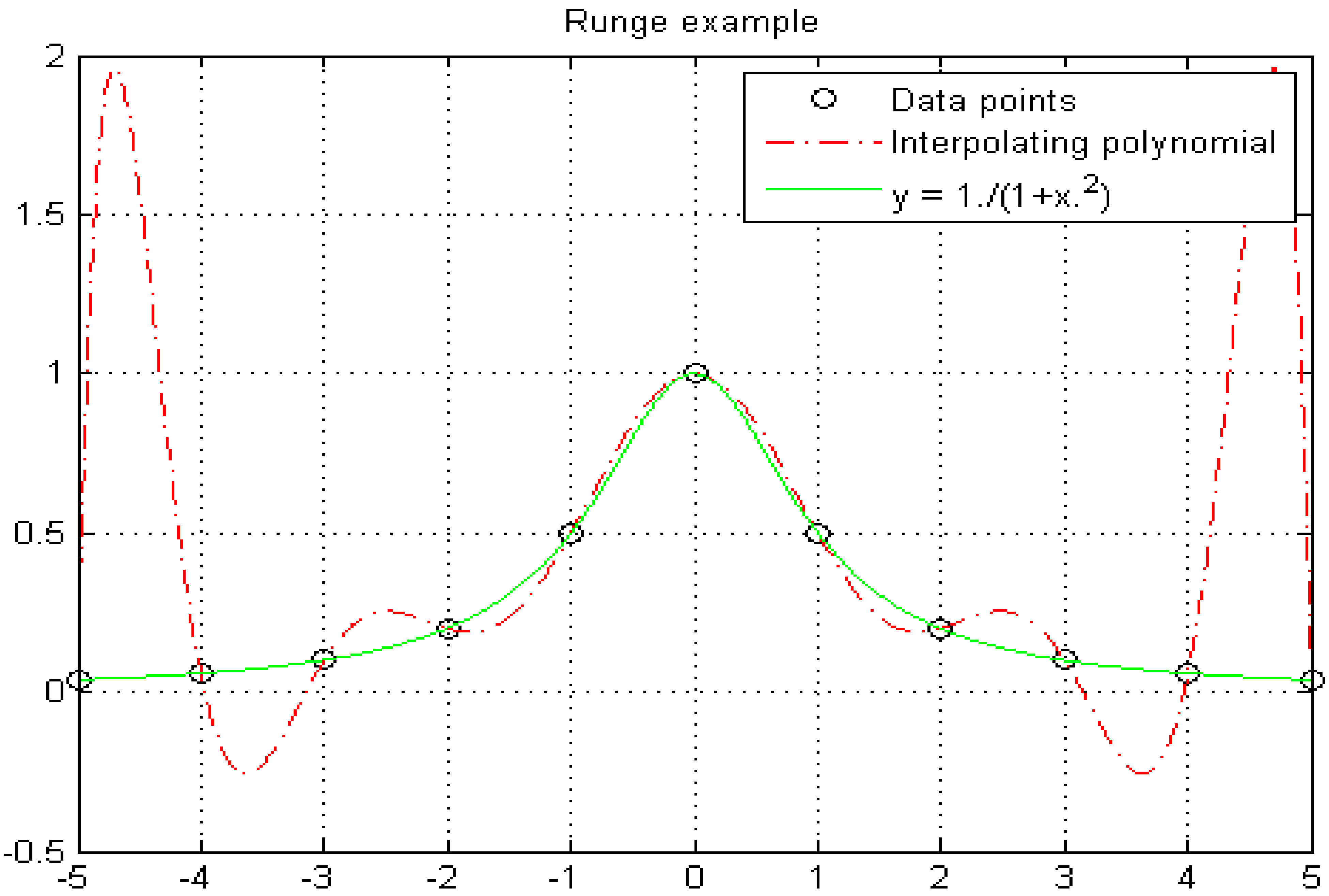
$$f(x) = \frac{1}{1+25x^2}, \text{ defined on an interval } [-1,1],$$

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

https://en.wikipedia.org/wiki/Runge%27s_phenomenon

Runge Function

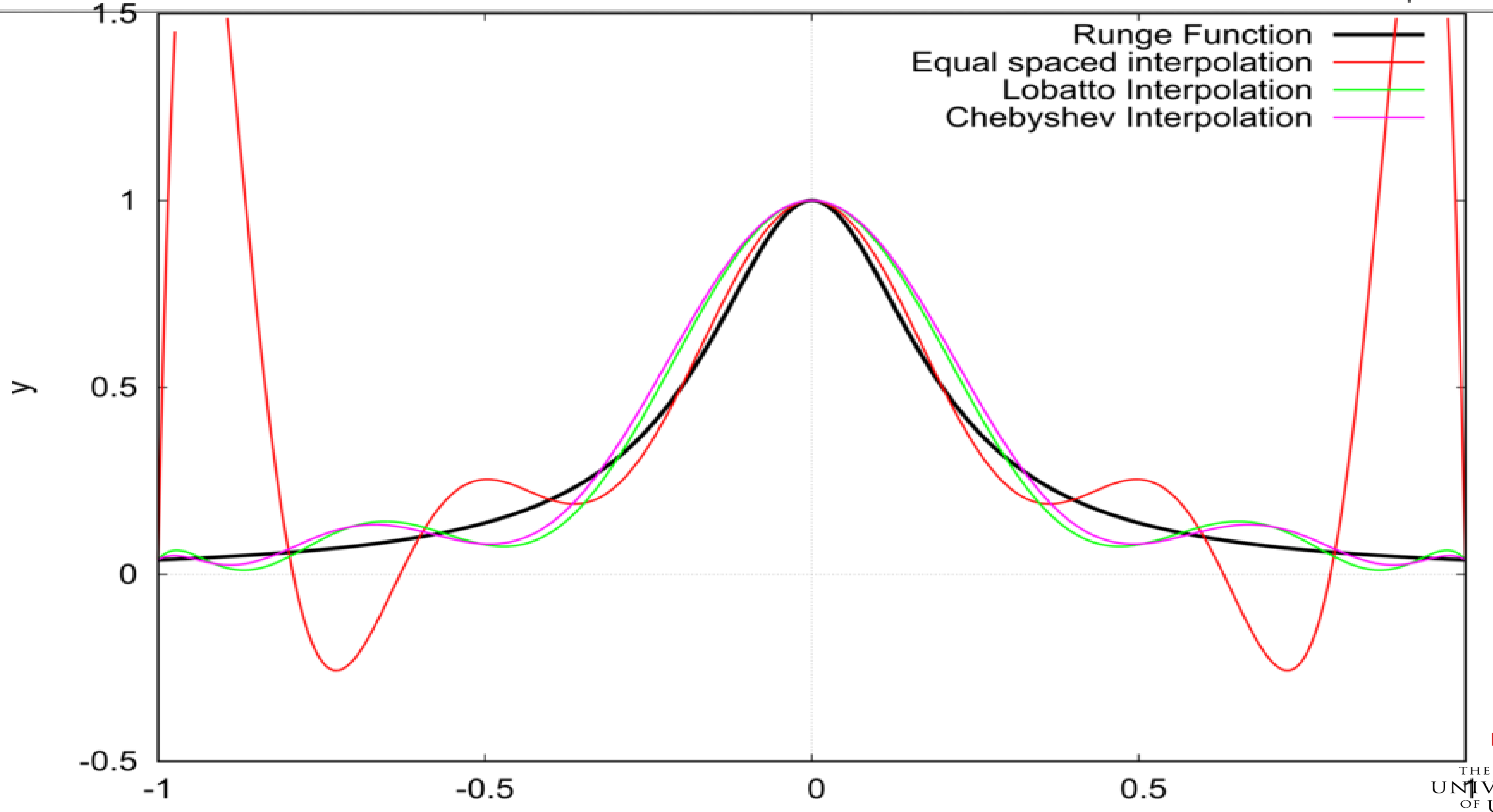
$$f(x) = \frac{1}{1+25x^2}$$



THE UNIVERSITY OF UTAH

Runge Function Point set Matters

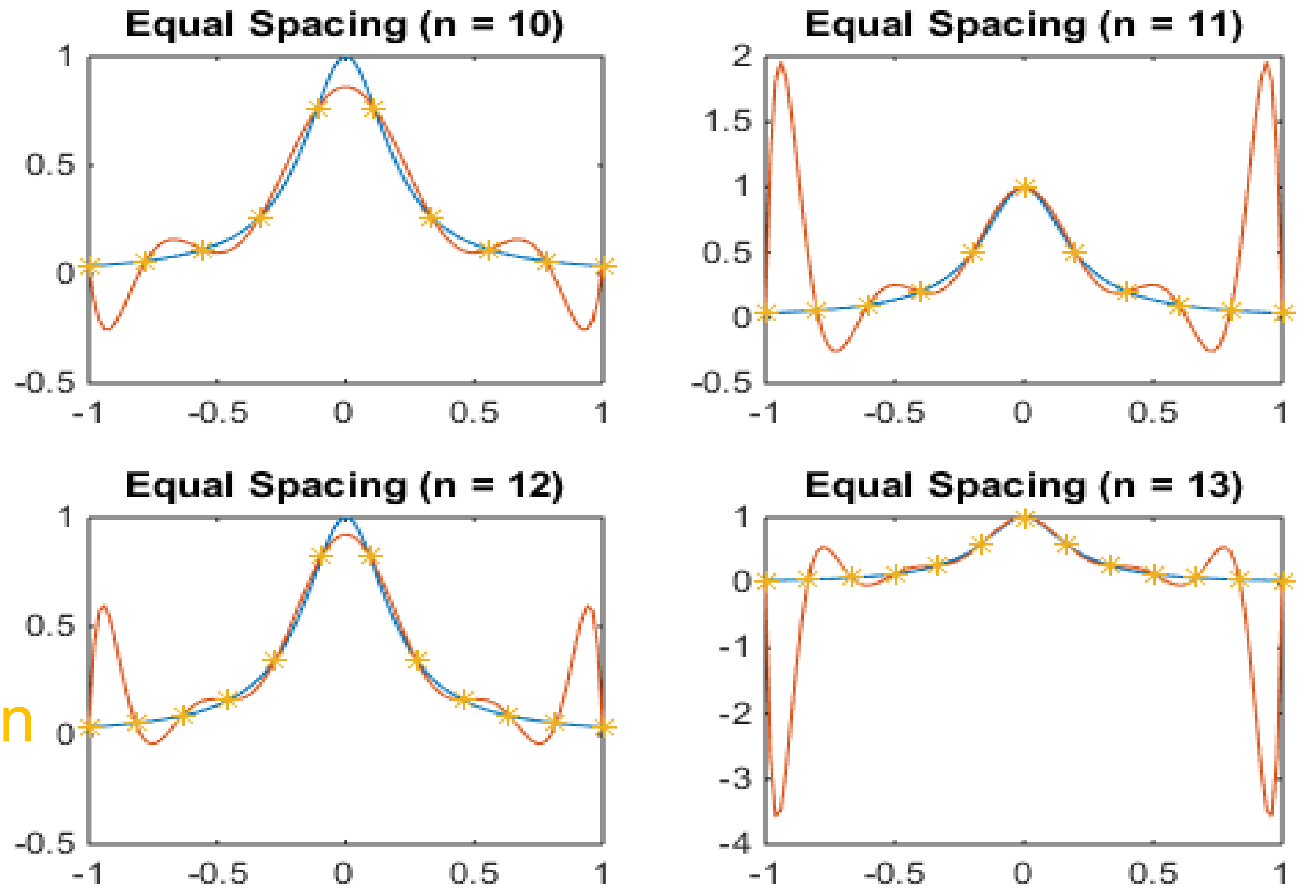
$$f(x) = \frac{1}{1+25x^2}$$



Example Code

```
% Script File: RungeEg
% For n=10:13, interpolants of f(x) = 1/(1+25x^2) on [-1,1]  are of plotted.
 close all
 x = linspace(-1,1,100);
 y = ones(100,1)./(1 + 25*x.^2);
 iplot = 0;
 for n=10:13
 xEqual = linspace(-1,1,n)';
    yEqual = ones(size(xEqual))./(1+25*xEqual.^2);;
    cEqual=              InterpN(xEqual,yEqual);
    pvalsEqual =       HornerN(cEqual,xEqual,x);
    iplot = iplot+1;
    subplot(2,2,iplot)
    plot(x,y,x,pvalsEqual,xEqual,yEqual,'*')
    title(sprintf('Equal Spacing (n = %2.0f)',n))
 end
```

Runge's Example Equally spaced polynomial interpolation points true solution blue polynomial approximation



Horner's Scheme for evaluating a Polynomial

Evaluating a polynomial

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

Optimal algorithm

Sum = a$_n$

For i = n-1 to 0

sum = x * sum + a$_i$

Why?

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)$$

$$\Downarrow$$

Grows very quickly for evenly spaced points

$$h^n \frac{(n-1)!}{4}$$

Use uneven points that are more closely spaced towards ends

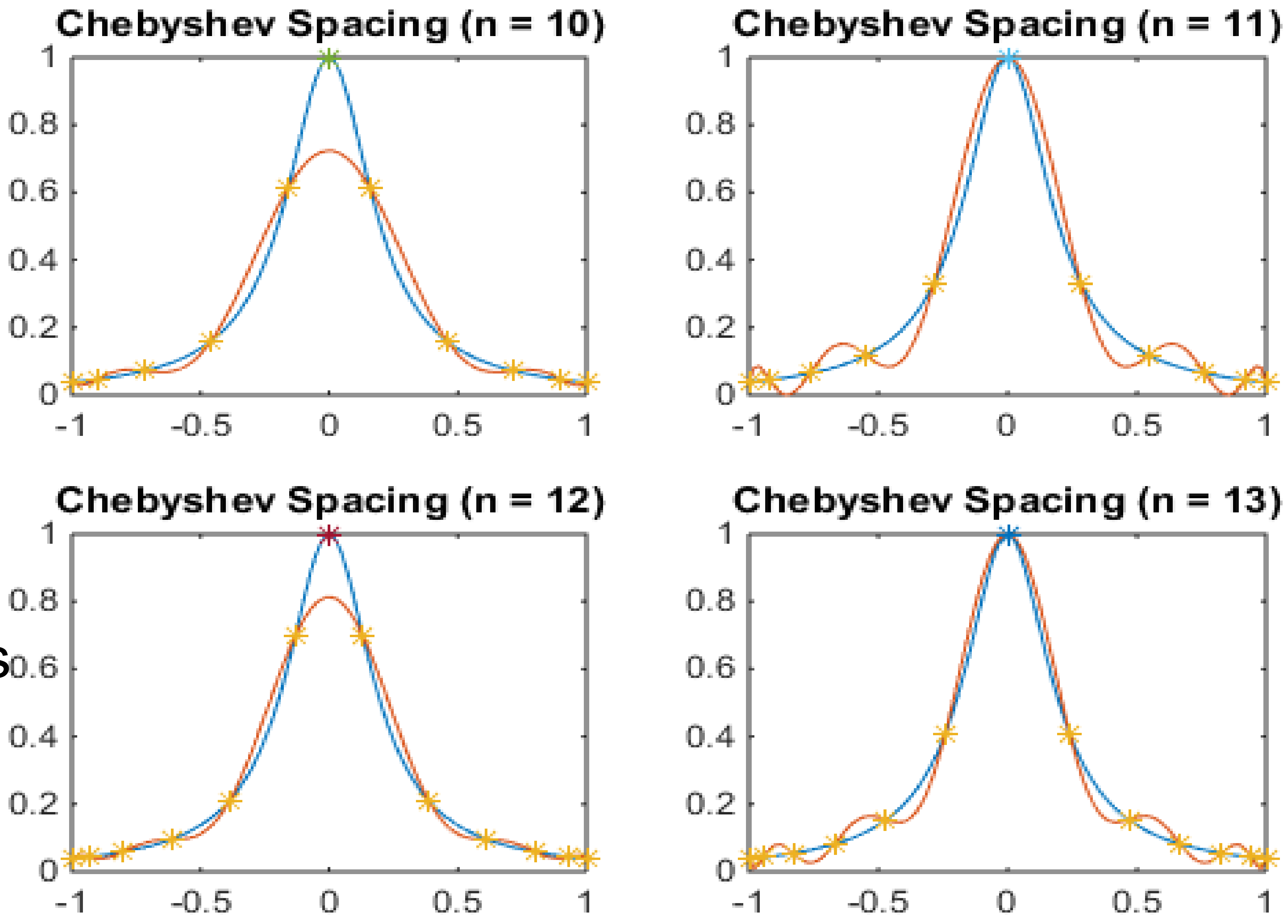
```
xEqual(kk) = -cos( (2.0*kk-1.0)*pi/(2.0*n)) / cos (pi/(2.0*n));
```

Runge Example Using Chebyshev Points

```
%  Script File: RungeEg
% For n=10:13, interpolants of f(x) = 1/(1+25x^2) on [-1,1]' are of plotted.
  close all
  x = linspace(-1,1,100);
  y = ones(1,100)./(1 + 25*x.^2);
  iplot = 0;
  for n=10:13
     xEqual = zeros(n);
     yEqual = zeros(n);
     for kk = 1:n
      xEqual(kk) = -cos( (2.0*kk-1.0)*pi/(2.0*n)) / cos (pi/(2.0*n));
     end
     yEqual = ones(n)./(1 +25*xEqual.^2);
     cEqual=InterpN(xEqual,yEqual);
     pvals = HornerN(cEqual,xEqual,x);
     iplot = iplot+1;
     subplot(2,2,iplot)
     plot(x,y,x,pvals,xEqual,yEqual,'*')
     title(sprintf('Chebyshev Spacing (n = %2.0f)',n))
  end
```

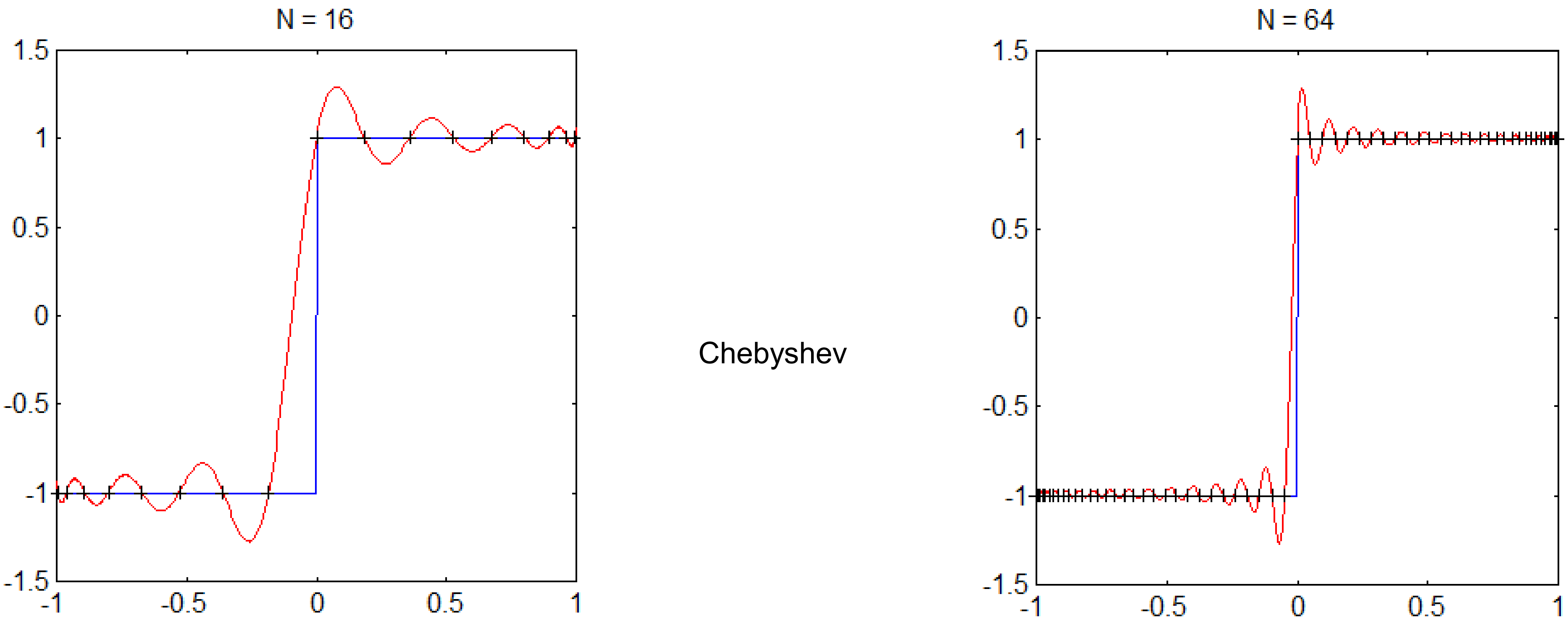
Runge's Example with Chebyshev Points

Runge's Example
Chebyshev polynomial interpolation points
true solution blue
polynomial approximation



Chebyshev - Gibb's Oscillations $f(x)=\text{sign}(x-\pi)$ => $f(x)$ - blue ; $F_N(x)$ - red; x_i - '+'

https://en.wikipedia.org/wiki/Gibbs_phenomenon



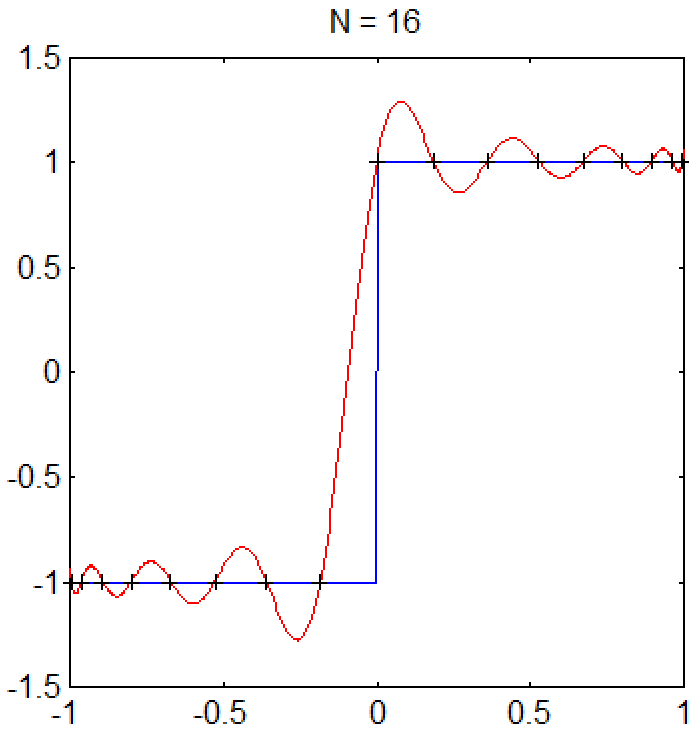
Chebyshev

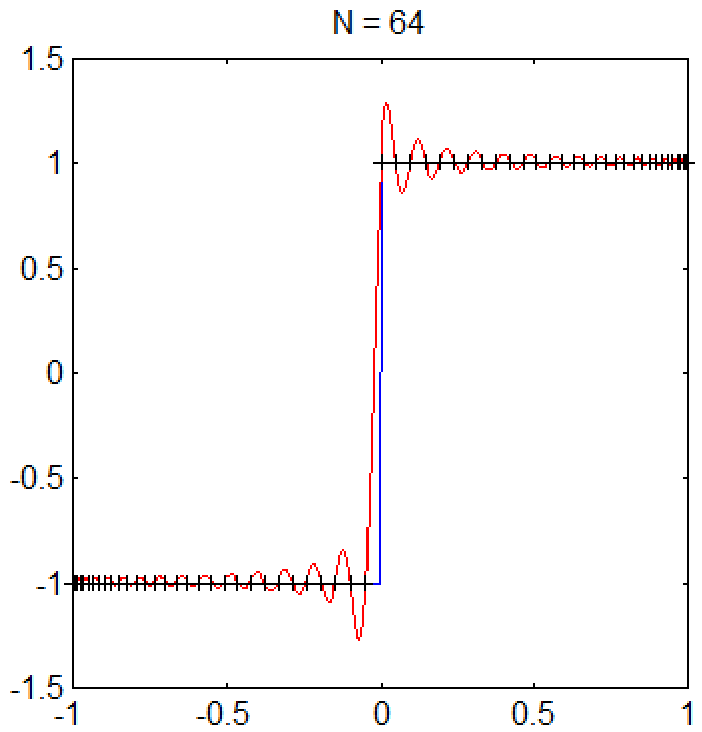
For functions with steep gradients we get overshoots and undershoots in the interpolating polynomial these are the famous Gibbs oscillations originally discovered with Fourier Series..

Chebyshev - Gibb's Oscillations

https://en.wikipedia.org/wiki/Gibbs_phenomenon $f(x)=\mathrm{sign}(x-\pi)$ => $f(x)$ - blue ; $F_N(x)$ - red; x_i - '+'

Chebyshev





For functions with steep gradients we get overshoots and undershoots in the interpolating polynomial these are the famous Gibbs oscillations originally discovered with Fourier Series..

Measuring the error, e(x), in an approximation?

For a polynomial of degree n, $p_n(\mathrm{x})$ approximating a function $f(x)$

The error $e(x) = p_n(x) - f(x)$, has to be calculated somehow.

We estimate the error over $[a,b]$ by using "norms".

The Lp norm of a function $g(x)$ is $\quad ||\mathrm{g}||_p = \left(\int_a^b |\, g(x)\,|^p \; dx\right)^{1/p}$,

We now estimate the norm of the error i.e. this integral using pointwise errors

Let $e_i = p_n(x_i) - f(x_i)$, where $x_i = a + (i-1)\dfrac{(b-a)}{(m-1)}, i = 1 \ldots m$,

where $m >> n$, so that the interpolation points don't dominate.

Measuring measure the error, e(x), in an approximation?

Let $e_i = p_n(x_i) - f(x_i)$, where $x_i = a + (i-1)\dfrac{(b-a)}{(n-1)}, i = 1 \ldots n$

$L1$ function norm

$$p = 1,\ ||\mathbf{e}||_1 = \left(\int_a^b |e(x)|\ dx\right) \approx \frac{(b-a)}{(n-1)} \sum_{j=1}^{n-1} 0.5(|e_j| + |e_{j+1}|)$$

$L2$ function norm

$$p = 2,\ ||\mathbf{e}||_2 = \left(\int_a^b |e(x)|^2\ dx\right)^{1/2} \approx \left[\frac{(b-a)}{(n-1)} \sum_{j=1}^{n-1} 0.5(|e_j|^2 + |e_{j+1}|^2)\right]^{\frac{1}{2}}$$

$L\inf$ function norm

$$p = \infty,\ ||\mathbf{e}||_\infty = \left(\int_a^b |e(x)|^\infty\ dx\right)^{1/\infty} \approx \max |e_j|, j = 1, \ldots, n$$

Matlab Vector Norms

Matlab has norms of vectors they are related to the function norms in the last slide.

$$\text{Let } e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

$L1$ vector norm

$$p = 1,\ norm(\mathrm{e},1) = ||\mathrm{e}||_1 = \sum_{j=1}^{n1} | e_j |$$

$L2$ vector norm

$$p = 2, norm(e,2) = ||\mathrm{e}||_2 = (\sum_{j=1}^{n} | e_j |^2)^{1/2}$$

$L\inf$ vector norm

$$p = \infty,\ \mathrm{norm(e,inf)} = ||\mathrm{e}||_\infty = \max | e_j |, j = 1, \ldots, n$$

We can use the matlab vectors as approximations to functions normd if we we scale them appropriately

$$p = 1,\ ||\mathbf{e}||_1 \approx \frac{(b-a)}{(n-1)} \sum_{j=1}^{n-1} 0.5(|e_j| + |e_{j+1}|) \approx \frac{(b-a)}{(n-1)} norm(e,1)$$

$L2$ function norm

$$p = 2,\ ||\mathbf{e}||_2 \approx \left[\frac{(b-a)}{(n-1)} \sum_{j=1}^{n-1} 0.5(|e_j|^2 + |e_{j+1}|^2) \right]^{\frac{1}{2}} \approx \sqrt{\frac{(b-a)}{(n-1)}} norm(e,2)$$

$L\inf$ function norm

$$p = \infty,\ ||\mathbf{e}||_\infty \approx \max |e_j|,\ j = 1,\ldots,n = norm(e,\inf)$$

Vectors and Functions

orthogonality a.b $(a,b)=0$ (f,g)=0

$$\sum_{i=1}^{n} a_i b_i = 0 \qquad \int_a^b f(x)g(x)dx = 0$$

Norm $\| a \|_2 = \sqrt{(a,a)}$

(or size)

$$\| f \|_2 = \sqrt{\int_a^b f(x)f(x)dx}$$

matlab $norm(a,2)$

$$\sqrt{\frac{\text{(b-a)}}{n}} norm(f_n, 2)$$

vector $f_i = f(x_i), x_i$

evenly spaced