

Lecture 07: Interpolation

Outline

- 1) Definitions, Motivation and Applications of Interpolation
- 2) Polynomial Interpolation!

Definition and **uniqueness** of the *interpolating polynomial* P_N

Finding P_N

Monomial Basis: Vandermonde matrices and Polyfit

Lagrange Basis:

Newton Basis:?

Properties of the Lagrange Polynomials

Examples

- 3) Error Estimates for Polynomial Interpolation

Lagrange Remainder Formula

Chebyshev points for reducing error (if you can)

- 4) There's more to life than polynomials...

Interpolation and Interpolants

Definition: Given a discrete set of values y_i at locations x_i , an interpolant is a (piecewise) continuous function $f(x)$ that passes **exactly** through the data (i.e. $f(x_i)=y_i$).

Comments:

lots of methods of interpolation, lots of different functions, works in n-Dimensions.

Interpolation and Interpolants

Some Applications:

Data filling: Connect the dots

Function Approximation:

Fundamental Component of other algorithms

Rootfinding: Secant method and IQI

Optimization: successive parabolic interpolation

Numerical integration and differentiation

Finite Element Methods!

Polynomial Interpolation (1-D)

The Interpolating Polynomial:

Theorem: There is a *unique* polynomial of degree N , $P_N(x)$ that passes exactly through $N+1$ values $y_1 \dots y_{N+1}$ at *distinct* positions $x_1 \dots x_{N+1}$ (i.e. $P_N(x_i) = y_i$)

Example: 2 points

3 points:

Polynomial Interpolation (1-D)

The Interpolating Polynomial:

Theorem: There is a *unique* polynomial of degree N , $P_N(x)$ that passes exactly through $N+1$ values $y_1 \dots y_{N+1}$ at *distinct* positions $x_1 \dots x_{N+1}$ (i.e. $P_N(x_i) = y_i$)

Proof: Let $P_N(x) = p_1 x^N + p_2 x^{N-1} + \dots + p_N x + p_{N+1}$
such that $P_N(x_i) = y_i$ for $i=1, \dots, N+1$ and $x_i \neq x_j \forall i, j$

Now assume that there is another degree N polynomial Q that passes through the same points i.e.

$Q_N(x) = q_1 x^N + q_2 x^{N-1} + \dots + q_N x + q_{N+1}$ and $Q_N(x_i) = y_i$

Finding the interpolating Polynomial:

Monomial Basis

Let $P_N(x) = p_1 x^N + p_2 x^{N-1} + \dots + p_N x + p_{N+1}$

which is a *Linear combination* of the *monomials*

$1, x, x^2, x^3, \dots, x^N$ with weights p_1, p_2, \dots, p_{N+1}

Finding the interpolating Polynomial: Matlab:

Easiest:

Next Easiest:

Break it down:

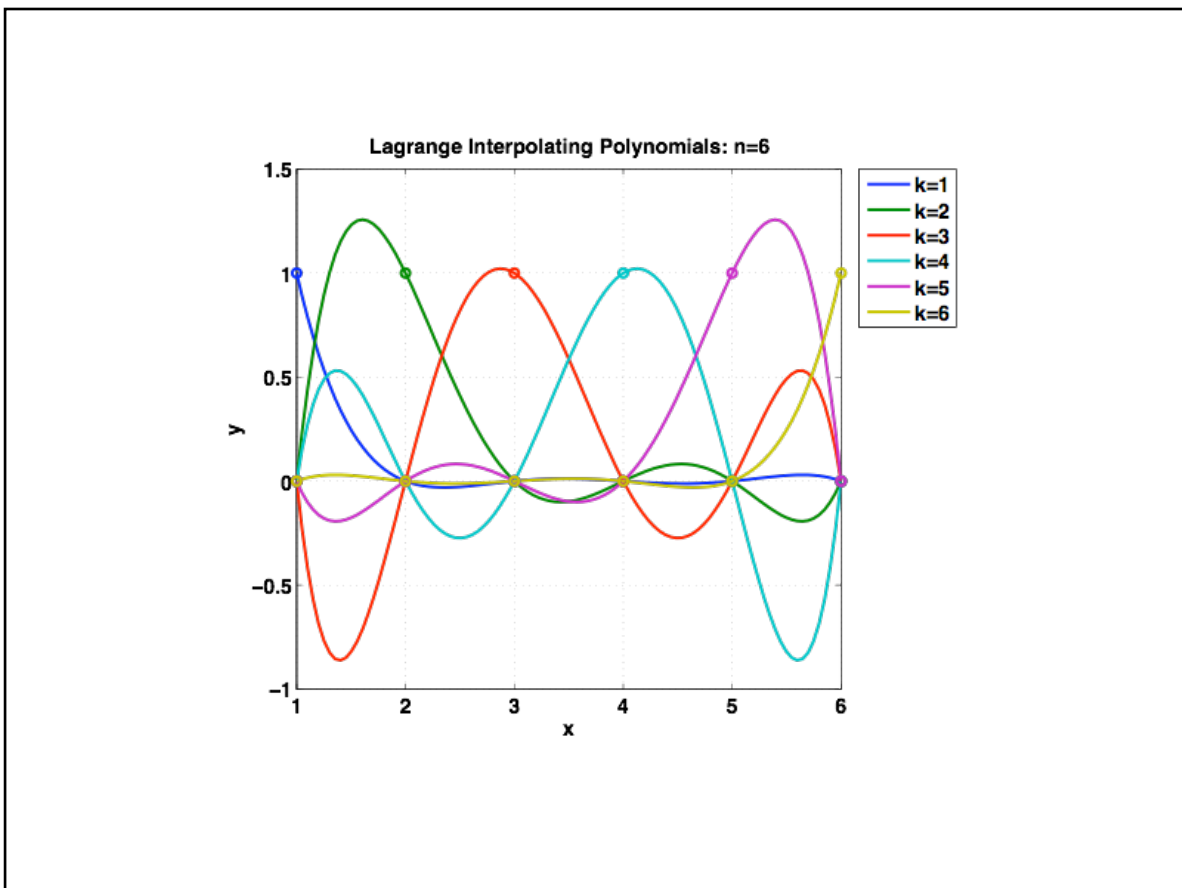
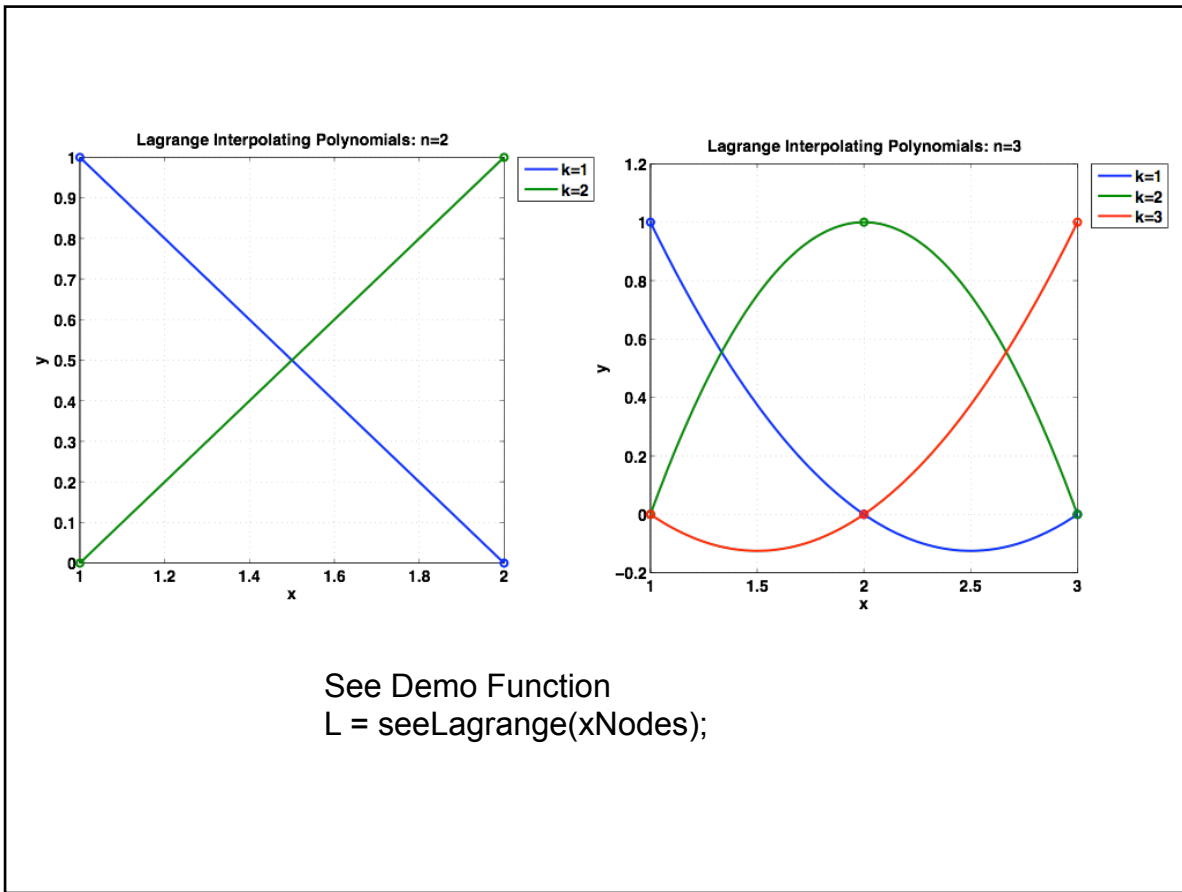
```
function A = vander(v)
n = length(v);
v = v(:);
A = ones(n);
for j = n-1:-1:1
    A(:,j) = v.*A(:,j+1);
end
```

Finding the interpolating Polynomial: A better way Lagrange basis:

define the Lagrange Polynomials of order N as

Examples: for 2 points, N=1 (linear)

3 points: N=2 (quadratic)



Finding the interpolating Polynomial: A better way Lagrange basis:

Fundamental properties of the Lagrange Polynomials

1) $L_k(x_i) =$

2) The interpolating polynomial can be written exactly as

$$P_N(x) =$$

Polynomial Interpolation using Lagrange:

Moler's polyinterp

```
function y = polyinterp(xNodes,yNodes,x)
%POLYINTERP Polynomial interpolation.
% y = POLYINTERP(xNodes,yNodes,x) computes y(j) = P(x(j)) where P is the
% polynomial of degree d = length(xNodes)-1 with P(xNodes(i)) = yNodes(i).

% Use Lagrangian representation.
% Evaluate at all elements of u simultaneously.
% Modified from Moler: NCM routines

n = length(xNodes); % number of interpolating points
y = zeros(size(x));
for k = 1:n
    w = ones(size(x)); % start calculating Lagrange Weights
    for j = [1:k-1 k+1:n] % skip over node k
        w = (x-xNodes(j))./(xNodes(k)-xNodes(j)).*w;
    end
    y = y + w*yNodes(k);
end
```

Polynomial Interpolation using Lagrange:

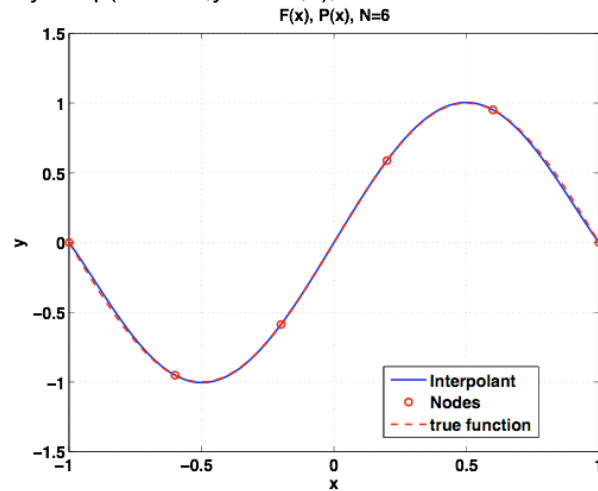
Examples: Interpolate $\sin(\pi x)$ using 6 equally spaced points on the interval $[-1, 1]$

```
xNodes = linspace(______);
```

```
yNodes =
```

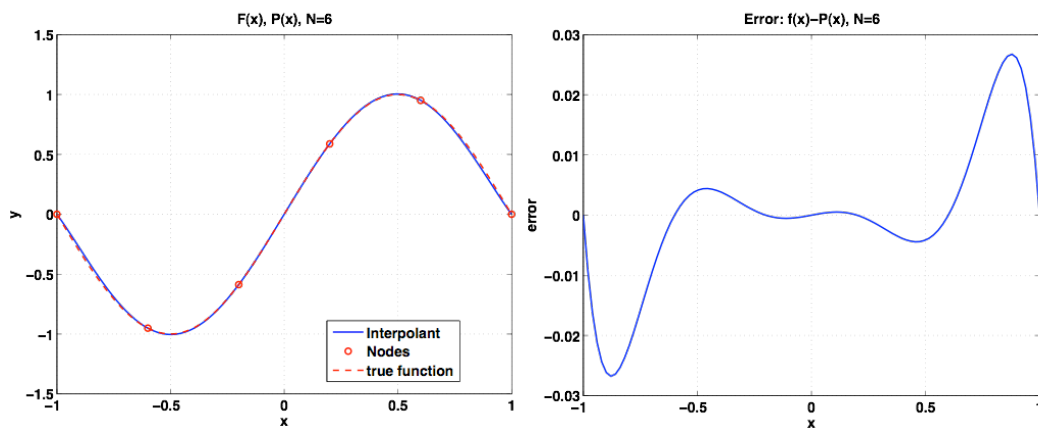
```
x =
```

```
y = polyinterp(xNodes,yNodes,x);
```



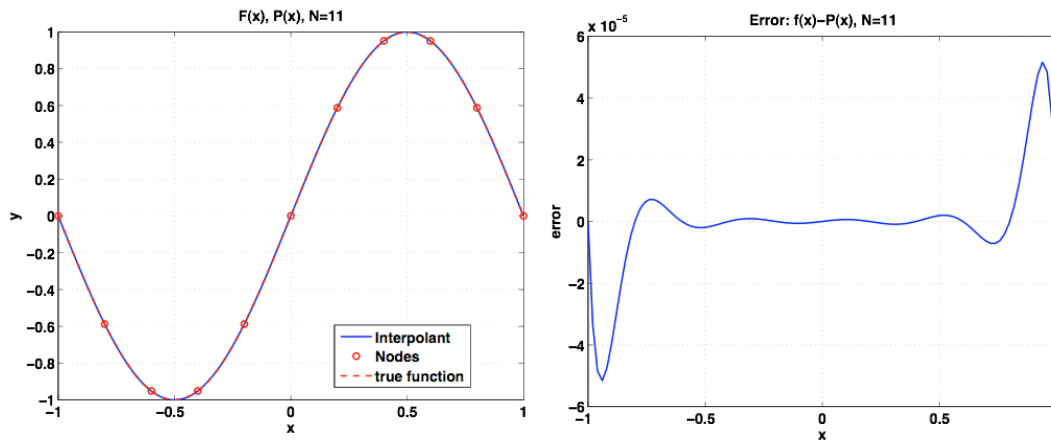
Polynomial Interpolation using Lagrange:

Examples: Interpolate $\sin(\pi x)$ using 6 equally spaced points on the interval $[-1, 1]$



Polynomial Interpolation using Lagrange:

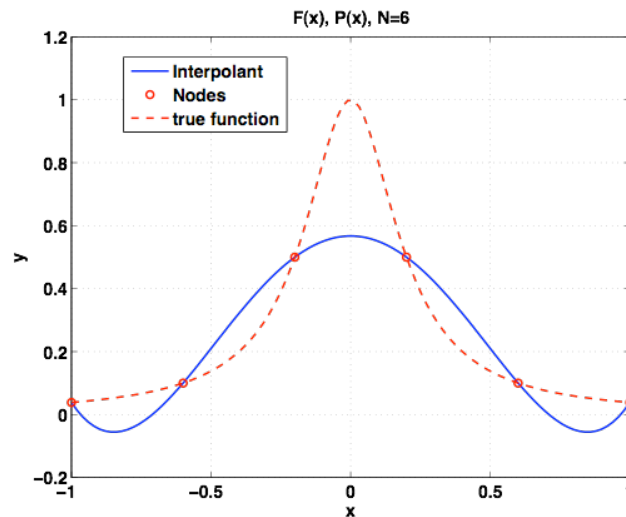
Examples: Interpolate $\sin(\pi x)$ using 11 equally spaced points on the interval $[-1, 1]$



Polynomial Interpolation using Lagrange:

Warning! High Order does not guarantee high accuracy!
(only if your function or data is well approximated by a high-order polynomial)

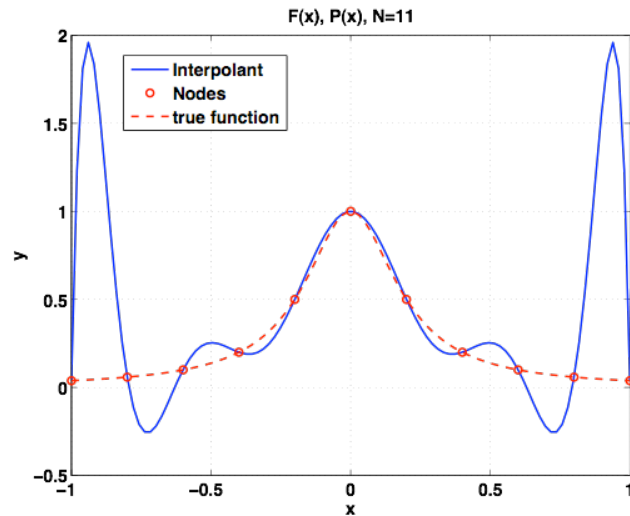
Example #2: Interpolate Runge's Function $f(x) = 1/(1+25x^2)$ using 6 points on the interval $[-1, 1]$;



Polynomial Interpolation using Lagrange:

Warning! High Order does not guarantee high accuracy!
(only if your function or data is well approximated by a high-order polynomial)

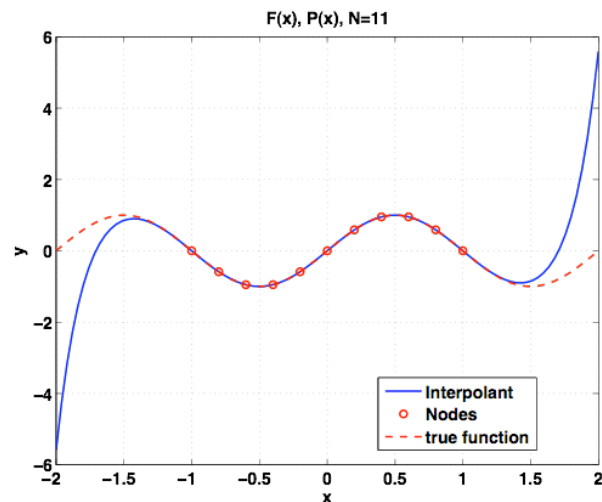
Example #2: Interpolate Runge's Function $f(x) = 1/(1+25x^2)$ using 11 points on the interval $[-1, 1]$;



Error Analysis for Polynomial Interpolation:

Simple Version #1: Avoid high-order polynomial interpolants unless you know what your doing...usually they can be highly oscillatory between nodes. $N=1:3$ (maybe ~ 5) good for government work.

Simple Version #2: Avoid **Extrapolation** with high-order polynomials altogether



Error Analysis for Polynomial Interpolation: Less Simple version:

Lagrange Remainder Theorem: similar to Taylor's Theorem

Theorem: let $f(x) \in C^{N+1}[-1,1]$, then

$$f(x) = P_N(x) + R_N(x)$$

where $P_N(x)$ is the interpolating polynomial and

$$R_N(x) = Q(x) f^{(N+1)}(c)/(N+1)! \quad \text{with } c \in [a,b] \text{ and}$$

$$Q(x) = (x-x_1)(x-x_2)(x-x_3)\dots(x-x_{N+1})$$

Comments:

$Q(x)$ is a $N+1$ order **monic** polynomial (leading coefficient = 1)

for Taylor's Theorem $Q(x) = \underline{\hspace{2cm}}$
and the error vanishes identically at $x =$

for Lagrange: $Q(x)$ vanishes at?

To minimize $R_N(x)$ requires minimizing $|Q(x)|$ for $x \in [-1,1]$. How?

Error Analysis for Polynomial Interpolation: Less Simple version:

The Magic of Chebyshev Polynomials

Definition: The Chebyshev polynomials are another basis for the space of polynomials with important properties

First 4 Chebyshev Polynomials:

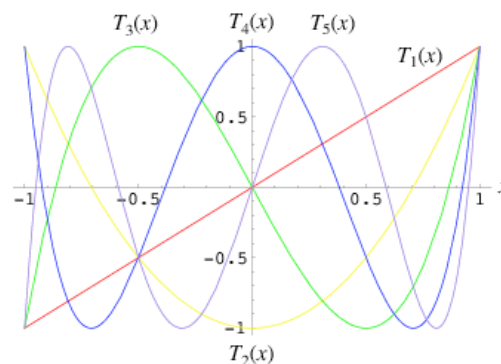
$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2 - 1$$

$$T_3(x) = 4x^3 - 3x$$

$$T_4(x) = 8x^4 - 8x^2 + 1$$



Error Analysis for Polynomial Interpolation: Less Simple version:

The Magic of Chebyshev Polynomials

Important Properties of the Chebyshev Polynomials

- 1) Recurrence relation $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$
- 2) Leading Coefficient of x^N in $T_N(x)$ is 2^{N-1} for $N \geq 1$
- 3) Extreme values: $|T_N(x)| \leq 1$ for $-1 \leq x \leq 1$

4)* Minimax principal: The polynomial

$T(x) = T_{N+1}(x)/2^N$ is a Monic Polynomial with the (amazing) property that

$$\max|T(x)| \leq \max|Q(x)| \text{ for } x \in [-1, 1] \text{ moreover}$$

$$\max|T(x)| = 1/2^N$$

Error Analysis for Polynomial Interpolation: Less Simple version:

The Magic of Chebyshev Polynomials

Therefore: To minimize the Lagrange remainder term, set

$Q(x) = T(x)$but $Q(x)$ is only defined by its roots

$$x_1, x_2, x_3, \dots, x_{N+1}$$

Sooo.....

Need to choose nodes to be the zeros of T_{N+1}

More Magic: the zeros of $T_N(x)$ in the interval $[-1, 1]$ are

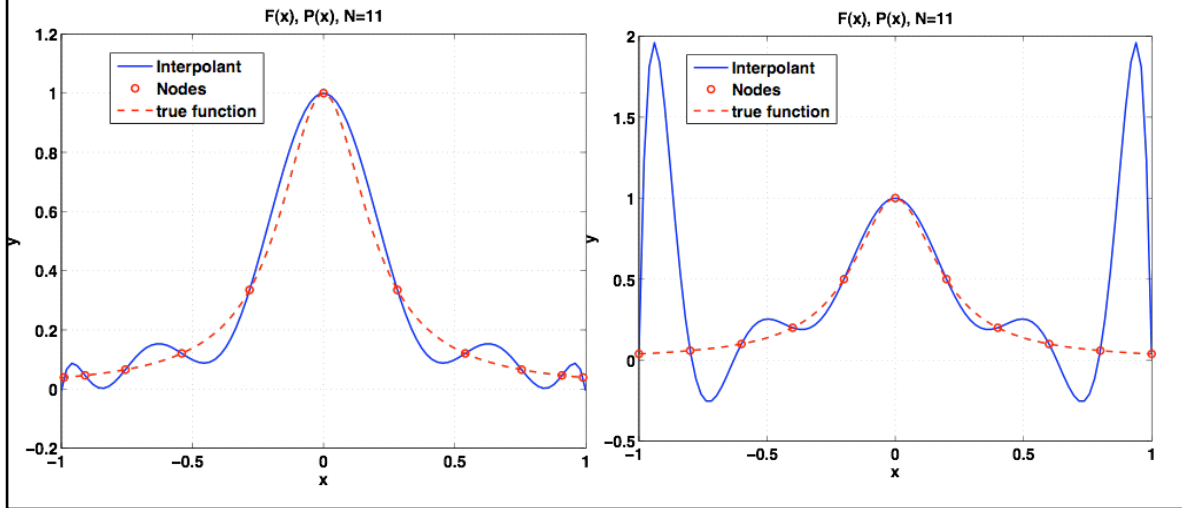
$$x_k = \cos((2k+1)\pi/2N) \text{ for } k=0, 1, \dots, N-1$$

Comparison of Interpolation with equally spaced points vs Chebyshev points

```

N = 11;
x=linspace(-1,1);
xNodes = chebyshevZeros(N-1);
f = @(x) 1./(1+25*x.^2)
[ y, error ] = seeInterp(f,x,xNodes,'showerror');

```

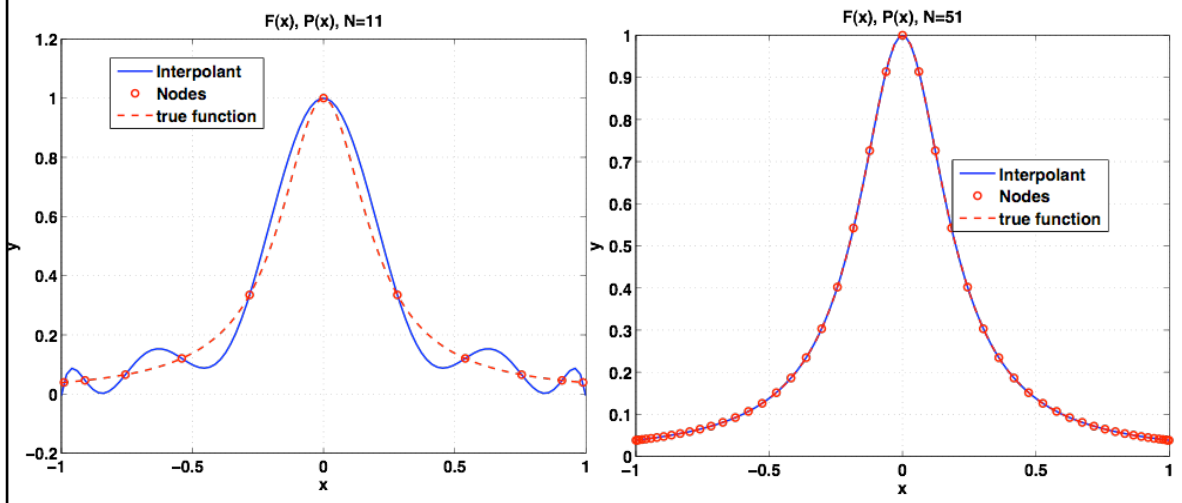


Comparison of Interpolation with equally spaced points vs Chebyshev points

```

N = 11;
x=linspace(-1,1);
xNodes = chebyshevZeros(N-1);
f = @(x) 1./(1+25*x.^2)
[ y, error ] = seeInterp(f,x,xNodes,'showerror');

```



Polynomial Interpolation:**Some final comments:**

- 0) Interpolation is a fundamental tool in the numerical quiver
- 1) The interpolating polynomial is only guaranteed to match the data at N points
- 2) High order polynomials can be wildly inaccurate between nodes
- 3) High order is not High Accuracy!
- 4) There are a lot more interpolants than full polynomials....
Onwards to piecewise-polynomial interpolation