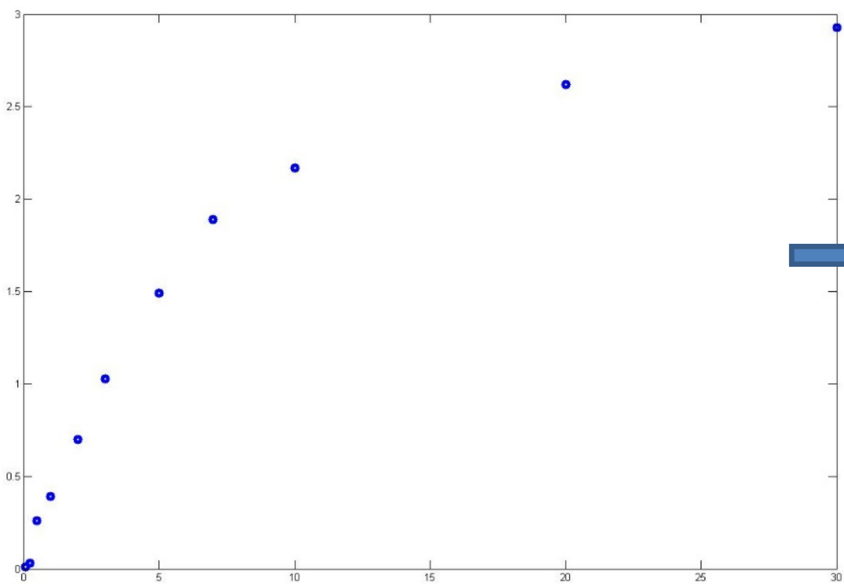
3.1 Interpolation and Lagrange Polynomial

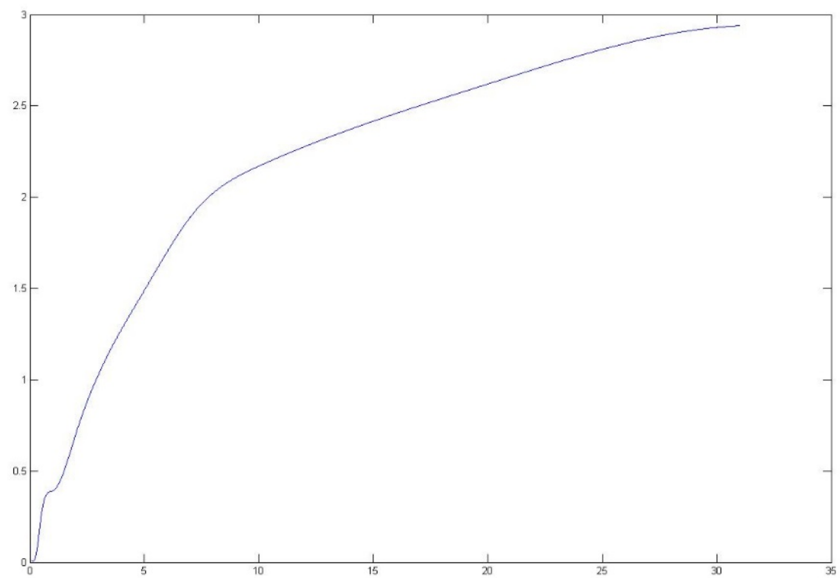
Example. Daily Treasury Yield Curve Rates

Date	1 Mo	3 Mo	6 Mo	1 Yr	2 Yr	3 Yr	5 Yr	7 Yr	10 Yr	20 Yr	30 Yr
09/01/15	0.01	0.03	0.26	0.39	0.70	1.03	1.49	1.89	2.17	2.62	2.93

Suppose we want yield rate for a four-years maturity bond, what shall we do?

Solution: Draw a smooth curve passing through these data points (interpolation).





Ref: http://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=yield

- **Interpolation problem**: Find a smooth function $P(x)$ which interpolates (passes) the data $\{(x_i, y_i)\}_{i=0}^{N}$.

- **Remark**: In this class, we always assume that the data $\{y_i\}_{i=0}^{N}$ represent measured or computed values of a underlying function $f(x)$, i.e., $y_i = f(x_i)$. Thus $P(x)$ can be considered as an approximation to f.

Polynomial Interpolation

Polynomials $P_n(x) = a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0$ are commonly used for interpolation.

- Advantages for using polynomial: efficient, simple mathematical operation such as differentiation and integration.

Theorem 3.1 Weierstrass Approximation theorem

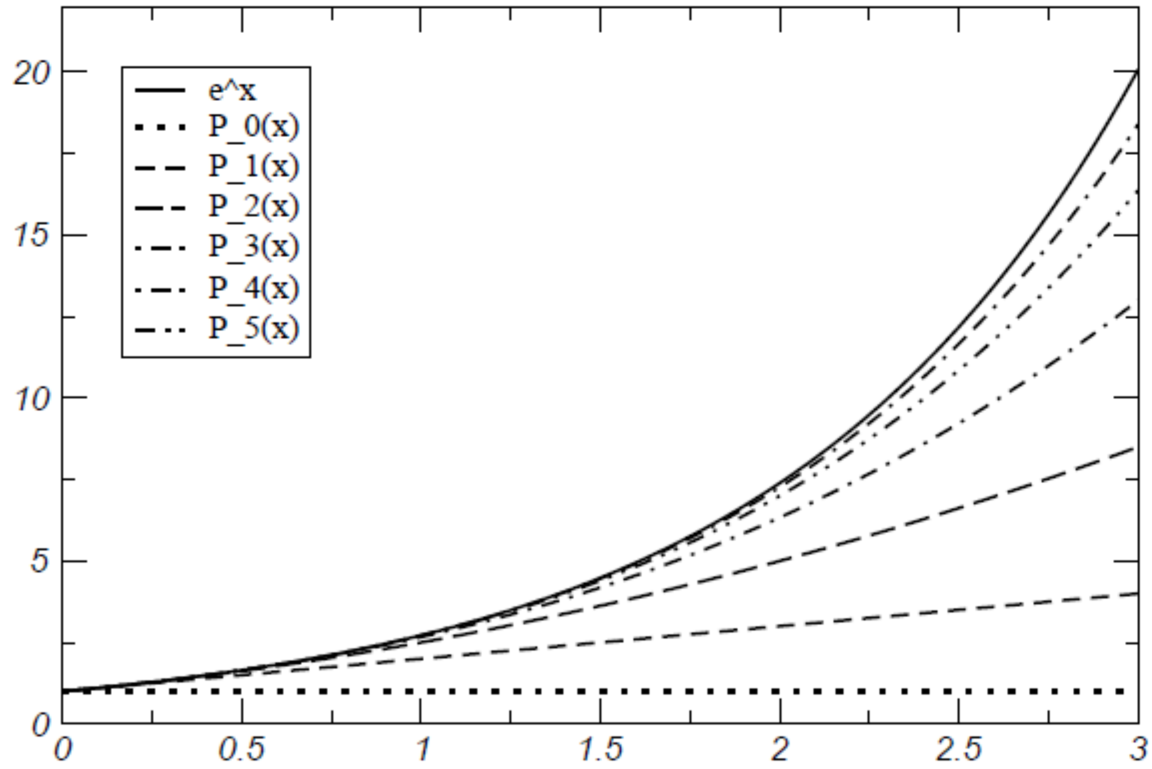
Suppose $f \in C[a, b]$. Then $\forall \epsilon > 0$, $\exists$ a polynomial $P(x)$: $|f(x) - P(x)| < \epsilon, \forall x \in [a, b]$.

Remark:

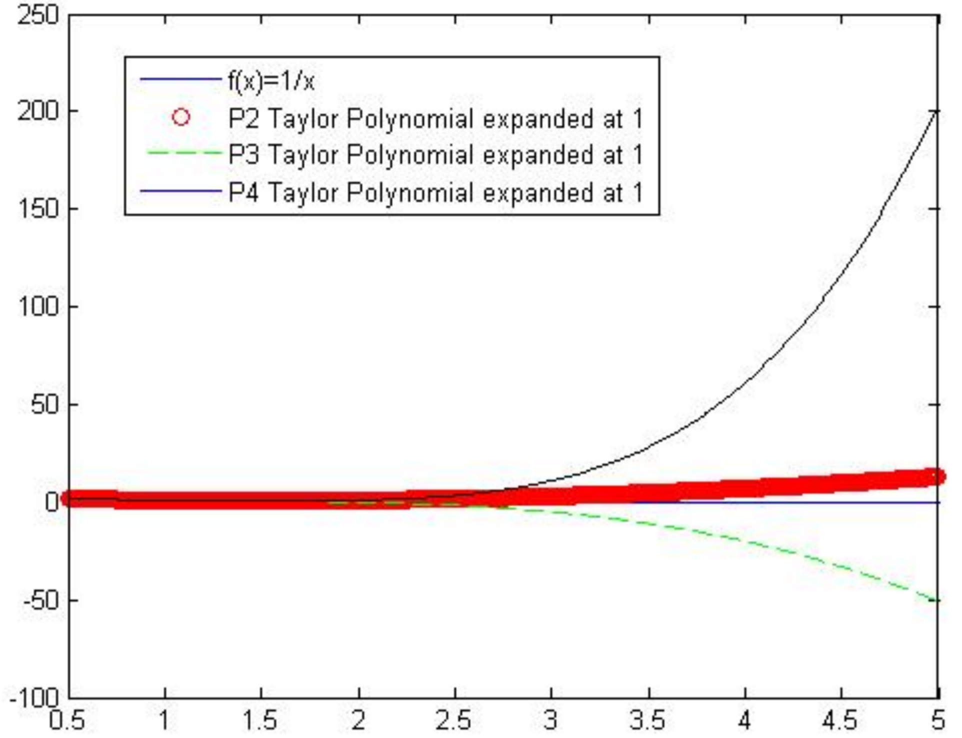
1. The bound is uniform, i.e. valid for all x in $[a, b]$. This means polynomials are good at approximating general functions.
2. But the way to find $P(x)$ is unknown.

- **Question:** Can Taylor polynomial be used here?
- Taylor expansion is accurate in the neighborhood of **one** point. But we need the (interpolating) polynomial to pass many points.

- **Example**. Taylor polynomial approximation of e^x for $x \in [0,3]$



- **Another bad Example**. Taylor polynomial approximation of $\frac{1}{x}$ for $x \in [0.5,5]$. Taylor polynomials of different degrees are expanded at $x_0 = 1$



$2nd$-degree Lagrange Interpolating Polynomial

Goal: construct a polynomial of degree 2 passing 3 data points $(x_0, y_0), (x_1, y_1), (x_2, y_2)$.

Step 1: construct a set of *basis polynomials* $L_{2,k}(x),\ k = 0,1,2$ satisfying

$$L_{2,k}(x_j) = \begin{cases} 1, & \text{when } j = k \\ 0, & \text{when } j \neq k \end{cases}$$

These polynomials are:

$$L_{2,0}(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)},$$

$$L_{2,1}(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)},$$

$$L_{2,2}(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Step 2: form the 2nd-degree Lagrange interpolating polynomial $P(x)$:

$$P(x) = y_0 L_{2,0}(x) + y_1 L_{2,1}(x) + y_2 L_{2,2}(x)$$

Verification:

a) $P(x)$ is a 2nd –degree polynomial

b) $P(x)$ satisfy the interpolation property:
$P(x_0) = y_0, P(x_1) = y_1, P(x_2) = y_2.$

Example 1. Use nodes $x_0 = 0, x_1 = 1, x_2 = 2$ to find 2nd Lagrange interpolating polynomial $P(x)$ for $f(x) = \frac{1}{x+1}$. And use $P(x)$ to approximate $f\left(\frac{3}{2}\right)$.

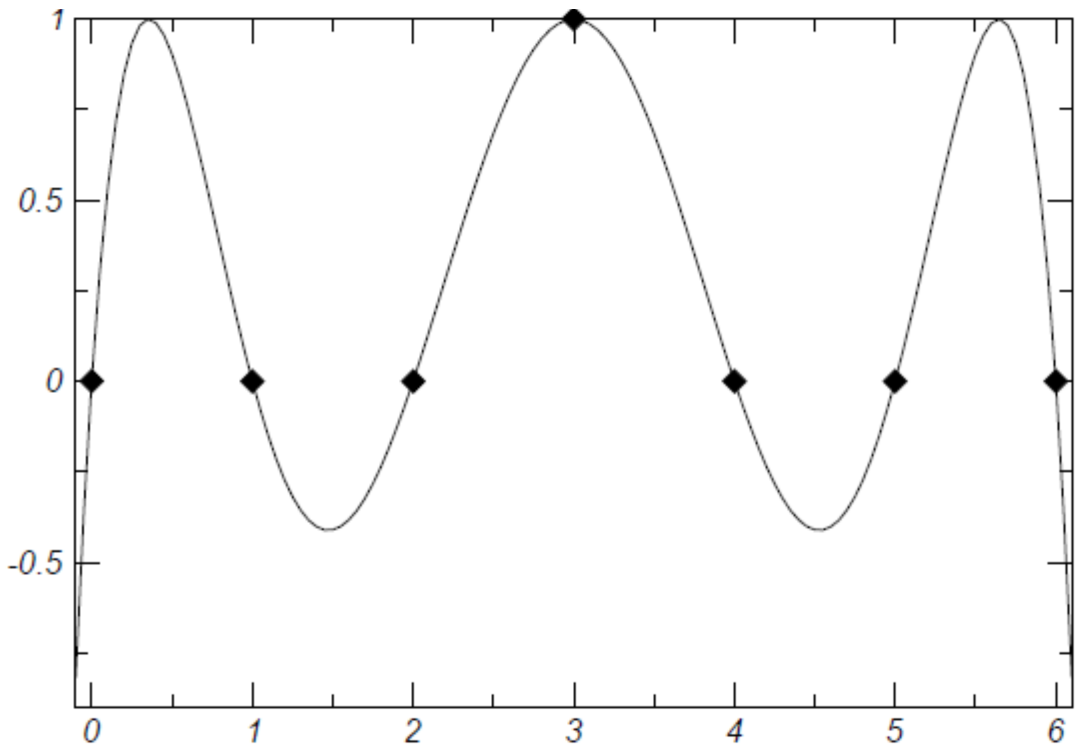
n-degree Interpolating Polynomial through $n+1$ Points

Constructing a Lagrange interpolating polynomial $P(x)$ passing through the points $(x_0, f(x_0))$, $\left(x_1, f(x_1)\right)$, $\left(x_2, f(x_2)\right)$, …, $(x_n, f(x_n))$.

1. Define Lagrange basis functions $L_{n,k}(x) = \prod_{i=0, i\neq k}^{n} \frac{x-x_i}{x_k-x_i} = \frac{x-x_0}{x_k-x_0} \dots \frac{x-x_{k-1}}{x_k-x_{k-1}} \cdot \frac{x-x_{k+1}}{x_k-x_{k+1}} \dots \frac{x-x_n}{x_k-x_n}$ for $k = 0,1 \dots n$.

   Remark: $L_{n,k}(x_k) = 1; L_{n,k}(x_i) = 0,\ \forall i \neq k$

2. $P(x) = f(x_0)L_{n,0}(x) + \cdots + f(x_n)L_{n,n}(x)$.



- $L_{6,3}(x)$ for points $x_i = i,\ i = 0, \ldots, 6.$

Theorem 3.2 If $x_0, \dots, x_n$ are $n+1$distinct numbers (called nodes) and f is a function whose values are given at these numbers, then a **unique polynomial** $P(x)$ of **degree at most n** exists with $P(x_k) = f(x_k)$, for each $k = 0, 1, \dots n$.

$P(x) = f(x_0)L_{n,0}(x) + \cdots + f(x_n)L_{n,n}(x)$.

Where $L_{n,k}(x) = \prod_{i=0, i \neq k}^{n} \frac{x - x_i}{x_k - x_i}$.

Error Bound for the Lagrange Interpolating Polynomial

Theorem 3.3. Suppose $x_0, \dots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$. Then, for each x in $[a, b]$, a number $\xi(x)$ (generally unknown) in (a, b) exists with

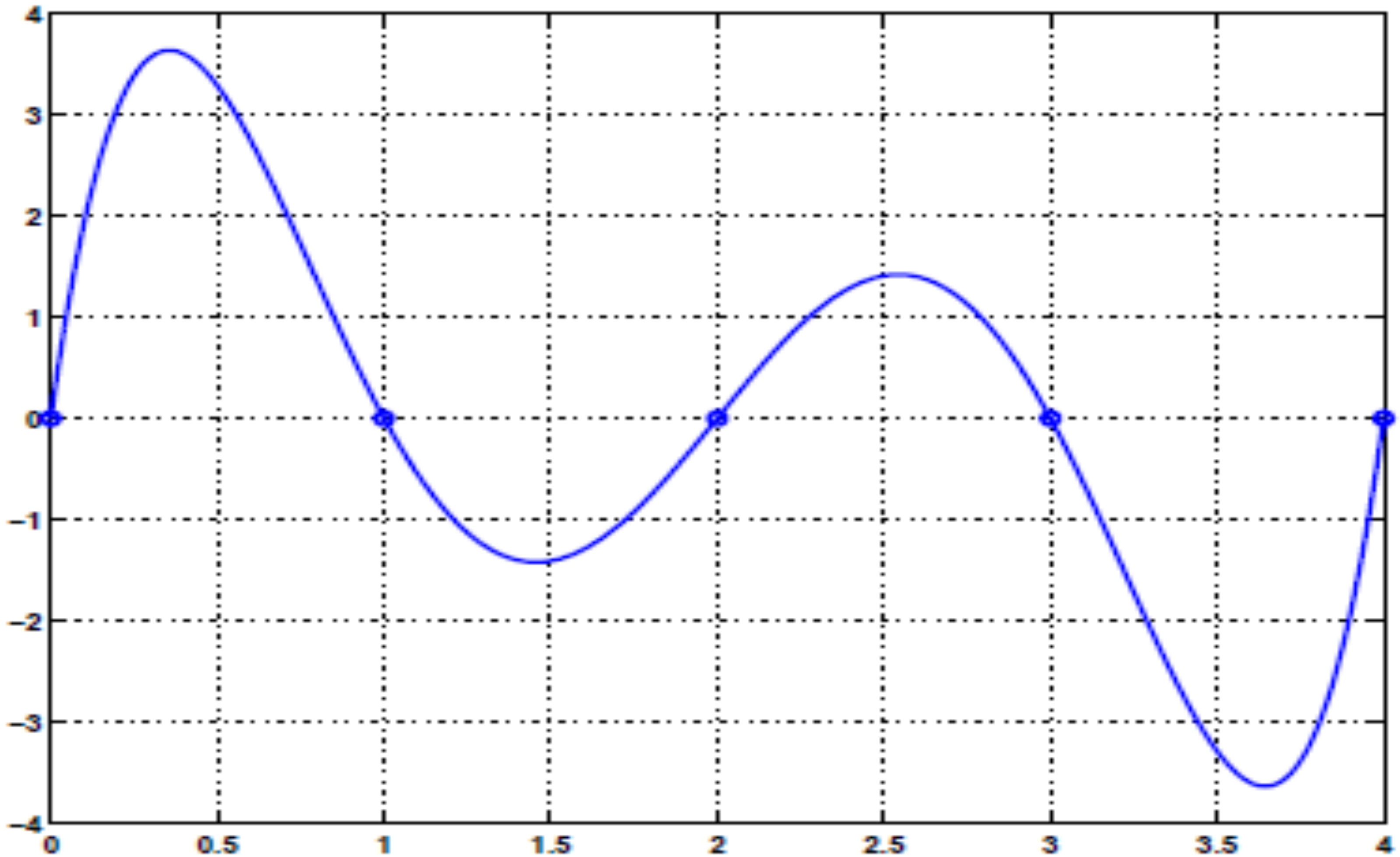
$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\dots(x - x_n)$$

Where $P(x)$ is the nth-degree Lagrange interpolating polynomial.

Remark: It is usually very difficult to estimate the absolute error $|f(x) - P(x)|$ using *Theorem 3.3*:

(1) the term $|f^{(n+1)}(\xi(x))|$ is hard to estimate for a general function f.

(2) the term $(x - x_0)(x - x_1)\dots(x - x_n)$ is oscillatory and its extreme value is hard to calculate for large value n.



Graph of $(x-0)(x-1)(x-2)(x-3)(x-4)$

Example 2. The 2[nd] Lagrange polynomial for $f(x) = \frac{1}{x+1}$ on [0, 2] using nodes $x_0 = 0, x_1 = 1, x_2 = 2$ is $P(x) = \frac{1}{6}x^2 - \frac{2}{3}x + 1$.

Determine the error form for $P(x)$, and maximum error when the polynomial is used to approximate $f(x)$ for $x \in [0,2]$.

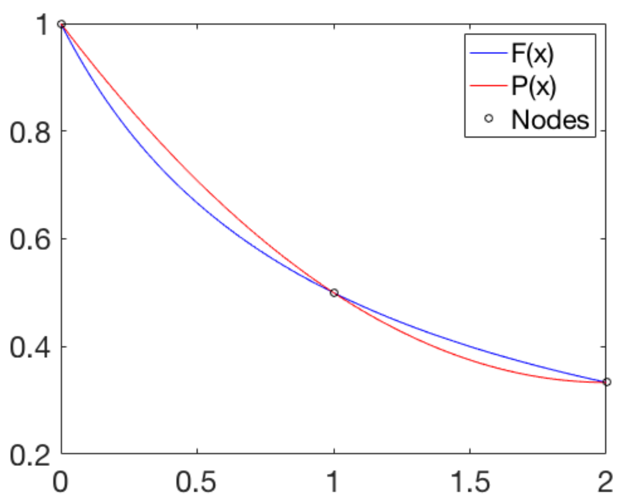
[MATLAB demo next slide]

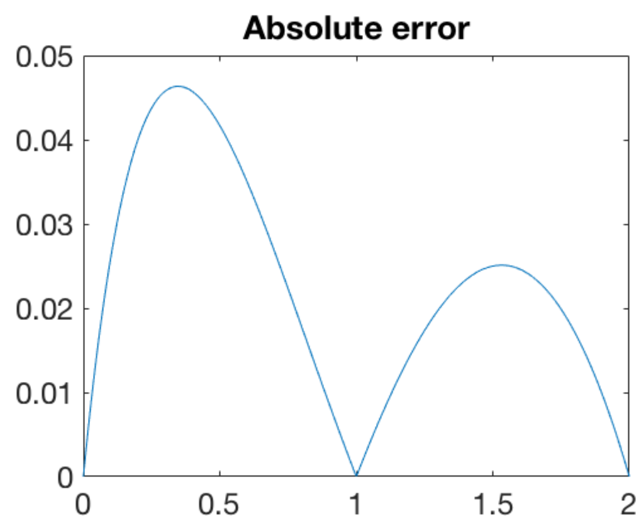
```
F = @(x) 1./(x+1);
P = @(x) 1/6*x.^2-2/3*x+1;
xNodes = [0,1,2];
yNodes = F(xNodes);

X = linspace(0,2,10000); % sample 10000 points
FX = F(X);
PX = P(X);
fprintf('\n Max-err: %.4e\n', max(abs(FX-PX))) % the maximum err

figure(1) % plot function and interpolant
plot(X,FX, 'b', X, PX, 'r', xNodes, yNodes, 'ko','LineWidth',4)
legend('F(x)', 'P(x)','Nodes')
set(gca, 'FontSize',24)

figure(2) % plot the absolute error
plot(X, abs(FX-PX),'LineWidth',4)
title('Absolute error')
set(gca, 'FontSize',24)
```





Example 3 (MATLAB). Plot the 4th Lagrange interpolating polynomial for

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

on the interval $[-1, 1]$ using 5 uniform nodes $x_0 = -1, x_1 = -0.5, x_2 = 0, x_3 = 0.5, x_4 = 1.$ On the same figure, plot the original function $f(x)$ and the interpolation nodes.

STEP 1: Lagrange Basis function (function .m file)

lagrange_basis.m

```
function phi_k = lagrange_basis(x, xnodes, k)
% function phi_k = lagrange_basis(x, xnodes, k)
% ### The Lagrange Basis/ or Lagrange Characteristic poly ###
%
% Input:
% x: a vector of x-values/ or a symbolic variable
% xnodes:  a vector of size (n+1) storing the values of x_k(k=0,...,n)
% k:       a integer in the range 0 - n.
% Output:
% phi_k: k-th (degree n) Lagrange basis eval @ x

n = length(xnodes)-1; % poly degree
xk = xnodes(k+1);

phi_k = 1;
for i = 0:n
   if i == k
       continue;
   end
   phi_k = phi_k.*(x-xnodes(i+1))/(xk-xnodes(i+1));
end

return
```

STEP 2: Lagrange Interpolation (script .m file)

lagrange_interp.m

```
% function and interpolation nodes
f = @(x) 1./(1+25*x.*x);
n = 4; % degree 4 interpolation
xNodes = linspace(-1,1,n+1);
yNodes = f(xNodes);

% evaluate function at 1001 uniform points
m = 1001;
xGrid = linspace(-1,1,m);
pGrid = zeros(size(xGrid));

for k = 0:n
    yk = yNodes(k+1);
    phi_k = lagrange_basis(xGrid, xNodes, k); % k-th basis eval @
 xGrid
    pGrid = pGrid + yk*phi_k;
end

plot(xGrid, f(xGrid),'b', xGrid, pGrid, 'r', xNodes, yNodes, 'ko',...
    'LineWidth', 4)
legend('F(x)', 'P(x)','Nodes')
set(gca, 'FontSize',24)
```

The figure (run ≫ lagrange_interp on command window)