Homework 5
Jingmin Sun
661849071

1. (a) i. For direct method, we can get the formula of

$$P_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

Since $P_3(x_j) = y_j$, for $j = 1, 2, 3, 4$,

$$A \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 3 \\ -6 \end{bmatrix}$$

$$\text{where } A = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{bmatrix}$$

$$\therefore \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 3 \\ -6 \end{bmatrix}$$

And by Matlab we can get:

Listing 1: Result

```
A=[1 -1 1 -1;1 1 1 1;1 2 4 8;1 3 9 27];b=[6;4;3;-6];x=A\b

x =

     3
     0
     2
    -1

diary off;
```

so

$$a_0 = 3, a_1 = 0, a_2 = 2, a_3 = -1$$

which means $P_3(x) = 3 + 2x^2 - x^3$

ii. For Lagrange Approach, we have the formula of

$$P_3(x) = \sum_{j=1}^{4} y_j l^{(j)}(x)$$

So, our goal is to find $l^{(j)}(x)$, since

$$
\begin{aligned}
l^{(1)}(x) &= \frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)} \\
&= \frac{(x-1)(x-2)(x-3)}{(-2)(-3)(-4)} \\
&= -\frac{(x-1)(x-2)(x-3)}{24} \\
l^{(2)}(x) &= \frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)} \\
&= \frac{(x+1)(x-2)(x-3)}{(2)(-1)(-2)} \\
&= \frac{(x+1)(x-2)(x-3)}{4} \\
l^{(3)}(x) &= \frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)} \\
&= \frac{(x+1)(x-1)(x-3)}{(3)(1)(-1)} \\
&= -\frac{(x+1)(x-1)(x-3)}{3} \\
l^{(4)}(x) &= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} \\
&= \frac{(x+1)(x-1)(x-2)}{(4)(2)(1)} \\
&= \frac{(x+1)(x-1)(x-2)}{8}
\end{aligned}
$$

Thus,

$$
\begin{aligned}
P_3(x) &= -6\frac{(x-1)(x-2)(x-3)}{24} + 4\frac{(x+1)(x-2)(x-3)}{4} - 3\frac{(x+1)(x-1)(x-3)}{3} - 6\frac{(x+1)(x-1)(x-2)}{8} \\
&= -\frac{(x-1)(x-2)(x-3)}{4} + (x+1)(x-2)(x-3) - (x+1)(x-1)(x-3) - \frac{3(x+1)(x-1)(x-2)}{4}
\end{aligned}
$$

iii. For Newton Divided difference, we can get the form of

$$P_3(x) = \square + \square\,(x-x_1) + \square\,(x-x_1)(x-x_2) + \square\,(x-x_1)(x-x_2)(x-x_3)$$

And for the coefficient, we can make a table for it:

$x_1 = -1$	$g[x_1] = y_1 = 6$			
$x_2 = 1$	$g[x_2] = y_2 = 4$	$g[x_1, x_2] = \frac{4-6}{1-(-1)} = -1$		
$x_3 = 2$	$g[x_3] = y_3 = 3$	$g[x_2, x_3] = \frac{3-4}{2-1} = -1$	$g[x_1, x_2, x_3] = \frac{-1-(-1)}{2-(-1)} = 0$	
$x_4 = 3$	$g[x_4] = y_4 = -6$	$g[x_2, x_3] = \frac{-6-3}{3-2} = -9$	$g[x_2, x_3, x_4] = \frac{-9-(-1)}{3-1} = -4$	$g[x_1, x_2, x_3, x_4] = \frac{-4-0}{3-(-1)} = -1$

Thus,

$$\begin{aligned} P_3(x) &= 6 - (x - x_1) - (x - x_1)(x - x_2)(x - x_3) \\ &= 6 - (x + 1) - (x + 1)(x - 1)(x - 2) \end{aligned}$$

(b)

$$\begin{aligned} g(x) &= \begin{cases} g_1(x) & x \in [-1, 1] \\ g_2(x) & x \in (1, 2] \\ g_3(x) & x \in (2, 3] \end{cases} \\ &= \begin{cases} a_1 x + b_1 & x \in [-1, 1] \\ a_2 x + b_2 & x \in (1, 2] \\ a_3 x + b_3 & x \in (2, 3] \end{cases} \\ \therefore a_1 \cdot (-1) + b_1 &= 6 \\ a_1 \cdot 1 + b_1 &= 4 \\ a_2 \cdot 1 + b_2 &= 4 \\ a_2 \cdot 2 + b_2 &= 3 \\ a_3 \cdot 2 + b_3 &= 3 \\ a_3 \cdot 3 + b_3 &= -6 \\ \therefore a_1 = -1, b_1 = 5, a_2 = -1, b_2 &= 5, a_3 = -9, b_3 = 21 \\ g(x) &= \begin{cases} -x + 5 & x \in [-1, 1] \\ -x + 5 & x \in (1, 2] \\ -9x + 21 & x \in (2, 3] \end{cases} \\ &= \begin{cases} -x + 5 & x \in [-1, 2] \\ -9x + 21 & x \in (2, 3] \end{cases} \end{aligned}$$

(c) We can use direct method here, such that

$$P_6(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6$$

Since $P_6(x_j) = y_j$, for $j = 1, 2, 3, 4$,

$$A \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 3 \\ -6 \end{bmatrix}$$

$$\text{where} \quad A = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 & x_1^6 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & x_2^5 & x_2^6 \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 & x_3^5 & x_3^6 \\ 1 & x_4 & x_4^2 & x_4^3 & x_4^4 & x_4^5 & x_4^6 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 \\ 1 & 3 & 9 & 27 & 81 & 243 & 729 \end{bmatrix}$$

$$\therefore \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 \\ 1 & 3 & 9 & 27 & 81 & 243 & 729 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 3 \\ -6 \end{bmatrix}$$

To solve the linear system, we can put it into an augmented matrix and reduced it to the "diagonal " form: And by Matlab we can get:

Listing 2: Result

```
A=[1 -1 1 -1 1 -1 1 6;1 1 1 1 1 1 1 4;1 2 4 8 16 32 64 3;1 3 9 27 81 243 729 -6];
B=rref(A)

B =

     1     0     0     0     6    30   120     3
     0     1     0     0    -5   -19   -70     0
     0     0     1     0    -5   -30  -119     2
     0     0     0     1     5    20    70    -1

diary off
```

And we can get for all $\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \in \mathbb{R}^6$, such that

$$\begin{aligned} a_0 + 6a_4 + 30a_5 + 120a_6 &= 3 \\ a_1 - 5a_4 - 19a_5 - 70a_6 &= 0 \\ a_2 - 5a_4 - 30a_5 - 119a_6 &= 2 \\ a_3 + 5a_4 + 20a_5 + 70a_6 &= -1 \end{aligned}$$

will be a interpolate of $f(x)$ of degree 6, since $a_6 \neq 0$ and one example can be find by setting $a_6 = a_5 = a_4 = 1$, and we can get by matlab that:

Listing 3: Result

```
B(:,8) = B(:,8)- B(:,7)- B(:,6) - B(:,5);
B(:,7)=[];B(:,6)=[]; B(:,5)=[]

B =

     1     0     0     0  -153
     0     1     0     0    94
     0     0     1     0   156
     0     0     0     1   -96

diary off
```

Thus $a_0 = -153, a_1 = 94, a_2 = 156, a_3 = -96$, and this example can be written as

$$P_6(x) = -153 + 94x + 156x^2 - 96x^3 + x^4 + x^5 + x^6$$

2. (a) Since

$$\begin{aligned} g_1''(x_2) &= g_2''(x_2) \\ 3x_2 &= 2c + 6d(x_2 - 2) \\ 3 \times 2 &= 2c + 0 \\ c &= 3 \end{aligned}$$

   (b) Since spline is natural, so that

$$\begin{aligned} g_2''(x_3) &= 0 \\ 2c + 6d(x_3 - 2) &= 0 \\ 6 &= -6d(1) \\ d &= -1 \end{aligned}$$

3. (a) Since we have need the degree 2 interpolating polynomial $p(x)$, we can use Lagrange approach

with

$$p(x) = \sum_{j=1}^{3} y_j l^{(j)}(x)$$

where

$$\begin{aligned}
l^{(1)}(x) &= \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} \\
&= \frac{(x-2)(x-4)}{(1-2)(1-4)} \\
&= \frac{(x-2)(x-4)}{3} \\
l^{(2)}(x) &= \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} \\
&= \frac{(x-1)(x-4)}{(2-1)(2-4)} \\
&= -\frac{(x-1)(x-4)}{2} \\
l^{(3)}(x) &= \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} \\
&= \frac{(x-1)(x-2)}{(4-1)(4-2)} \\
&= \frac{(x-1)(x-2)}{6} \\
\therefore p(x) &= -\frac{\ln(2)(x-1)(x-4)}{2} + \frac{\ln(4)(x-1)(x-2)}{6}
\end{aligned}$$

(b)

$$\begin{aligned}
|f(x)-p(x)| &= \left|\frac{(x-x_1)(x-x_2)(x-x_3)}{3!} f'''(c)\right| && \text{for } c \in [\min(x,x_1,x_2,x_3), \max(x,x_1,x_2,x_3)] \\
&= \left|2\frac{(x-1)(x-2)(x-4)}{6c^3}\right| && \text{for } c \in [\min(x,x_1,x_2,x_3), \max(x,x_1,x_2,x_3)] \\
&= \left|\frac{(x-1)(x-2)(x-4)}{3c^2}\right| && \text{for } c \in [\min(x,x_1,x_2,x_3), \max(x,x_1,x_2,x_3)] \\
|f(3)-p(3)| &= \left|\frac{(3-1)(3-2)(3-4)}{3c^2}\right| && \text{for } c \in [1,4] \\
&= \left|\frac{-2}{3c^2}\right| && \text{for } c \in [1,4] \\
&= \left|\frac{2}{3c^2}\right| && \text{for } c \in [1,4] \\
&\leq \frac{2}{3}
\end{aligned}$$

(c)

$$
\begin{aligned}
|f(3)-p(3)| &= \left|\ln(3) - \left(-\ln(2)\cdot\frac{(3-1)(3-4)}{2} + \ln(4)\cdot\frac{(3-1)(3-2)}{6}\right)\right| \\
&= \left|\ln(3) - \left(-\ln(2)\cdot\frac{-2}{2} + \ln(4)\cdot\frac{2}{6}\right)\right| \\
&= \left|\ln(3) - \left(\ln(2) + \frac{\ln(4)}{3}\right)\right| \\
&= \left|\frac{\ln(\frac{27}{32})}{3}\right| \\
&\approx 0.056633
\end{aligned}
$$

4. (a) Here, we can define z_j as $(x - x_j)$

$$
\begin{aligned}
p(x) &= a_1 + \sum_{j=2}^{n} a_j (x-x_1)(x-x_2)\cdots(x-x_{j-1}) \\
&= a_1 + \sum_{j=2}^{n} a_j \Pi_{i=1}^{i-1} z_i \\
&= a_1 + z_1 a_2 + z_1 z_2 a_3 + \cdots + z_1 z_2 \cdots z_{n-1} a_n \\
&= a_1 + z_1(a_2 + z_2(a_3 + z_3(\cdots a_{n-1} + z_{n-1}(a_n)))) \\
&= a_1 + (x-x_1)(a_2 + (x-x_2)(a_3 + (x-x_3)(\cdots a_{n-1} + (x-x_{n-1})(a_n))))
\end{aligned}
$$

(b) The function to calculate the coefficient of Newton Divided Difference Method:

Listing 4: myPolyCoef.m function

```
%% This program computes the coefficients of Newton Divided Difference Method
%input: x and y are vectors containing the x and y coordinates
%output: coefficients c of interpolating polynomial in nested form
function c=myPolyCoef(x,y)
n=length(x);
c=zeros(size(x));
for j=1:n
v(j,1)=y(j); % Fill in y column of Newton triangle
end
for i=2:n % For column i,
for j=i:n % fill in column from top to bottom
v(j,i)=(v(j,i-1)-v(j-1,i-1))/(x(j)-x(j+1-i));
end
end
for i=1:n
c(i)=v(i,i); % Read along diagonal entries

end % for output coefficien
```

(c) The function to evaluate the interpolation at point x:

Listing 5: myPolyEval.m function

```
function y = myPolyEval( x, xvec, a)
%% to evaluate a polynomial at x
```

```
% inputs: x: where the polynomial p(x) is evalute
% xvec(j): the input nodes to interpolate the function
% a: a vector of length d+1, that contains% the coefficients of p(x), with the constant
% term p(0) being the last
% output: y = p(x)
%%%%%%%%%%%%%%%%%%
n = length(a);
y = a(n);
for j = n-1:-1:1
  y = y*(x-xvec(j))+a(j);
end
```

(d) Firstly, the most important property is that $f(x) = p(x) = y$ at the points $\{x_j, y_j\}_{j=1}^{n}$, so we can choose write a test code for that

Listing 6: test_original.m function

```
clear all; clc;
format long;


% Initailize the data points of x,y
r = [2,3,4];
for ra =r
randn('seed',20190316);
x = randn(ra+1,1);
y = randn(ra+1,1);
a = myPolyCoef(x,y);
fprintf('r = %d\n', ra);
result1 =[];
for k = 1:length(x)
result = myPolyEval( x(k), x, a);
fprintf('When x = % 5.4f, p(x) = % 5.4f, f(x) = % 5.4f, error = %5.4e\n',x(k),
    result,y(k), abs(result-y(k)));
end
end
```

and we can check the output:

Listing 7: Output

```
test_original
r = 2
When x = -0.0187, p(x) = 1.0790, f(x) = 1.0790, error = 0.0000e+00
When x = 0.3203, p(x) = 1.0792, f(x) = 1.0792, error = 0.0000e+00
When x = -1.1637, p(x) = 0.8123, f(x) = 0.8123, error = 0.0000e+00
r = 3
When x = -0.0187, p(x) = 1.0792, f(x) = 1.0792, error = 0.0000e+00
When x = 0.3203, p(x) = 0.8123, f(x) = 0.8123, error = 0.0000e+00
When x = -1.1637, p(x) = 2.5759, f(x) = 2.5759, error = 0.0000e+00
When x = 1.0790, p(x) = 0.1425, f(x) = 0.1425, error = 2.4980e-16
r = 4
When x = -0.0187, p(x) = 0.8123, f(x) = 0.8123, error = 0.0000e+00
When x = 0.3203, p(x) = 2.5759, f(x) = 2.5759, error = 0.0000e+00
When x = -1.1637, p(x) = 0.1425, f(x) = 0.1425, error = 1.0270e-15
When x = 1.0790, p(x) = 0.6650, f(x) = 0.6650, error = 2.2204e-16
When x = 1.0792, p(x) = 0.3783, f(x) = 0.3783, error = 4.9960e-16
```

And we can observe that the error is approximately zero, and the error might be caused by the rounding process of computer, so we can conclude that our implementation may be correct.

Besides that, since we know that $n+1$ points can exactly interpolate polynomial of degree n, so I randomly choose coefficients $a_0, a_1 \cdots a_n$ to make a random polynomial $a_0 + a_1 x + a_2 x^2 + \cdots a_n x^n$, and use Horner's nested method to calculate the y for randomly choosen $n+1$ xs. And I then randomly choose three points z_1, z_2, z_3, and calculate the error between $f(z_i)$ and $p(z_i)$, which interpolated by x_j, y_j

Listing 8: test_z.m function

```
clear all; clc;
format long;


% Initailize the data points of x,y
r = [2,3,4];
for ra =r
randn('seed',20190316);
coef = ra*randn(ra+1,1);
x =  randn(ra+1,1);
y = zeros(ra+1,1);
for i = 1:ra+1
y(i) = coef(ra+1);
for j = ra:-1:1
  y(i) = y(i)*x(i)+coef(j);
end
end

z = randn(3,1);
b = zeros(3,1);
for i = 1:length(z)
b(i) = coef(ra+1);
for j = ra:-1:1
  b(i) = b(i)*z(i)+coef(j);
end
end
a = myPolyCoef(x,y);
fprintf('r = %d\n', ra);
for k = 1:length(z)
result = myPolyEval(z(k),x, a);
fprintf('When z = % 5.4f, p(z) = % 7.3f, f(z) = % 7.3f, error = %5.4e\n',z(k),
    result,b(k), abs(result-b(k)));
end
end
```

and we can check the output:

Listing 9: Output

```
test_z
r = 2
When z = 2.5759, p(z) = -13.830, f(z) = -13.830, error = 1.2699e-11
When z = 0.1425, p(z) = 0.007, f(z) = 0.007, error = 3.0165e-12
When z = 0.6650, p(z) = -0.641, f(z) = -0.641, error = 2.9343e-13
```

```
r = 3
When z = 0.6650, p(z) = -0.009, f(z) = -0.009, error = 2.7062e-16
When z = 0.3783, p(z) = -0.017, f(z) = -0.017, error = 4.8572e-16
When z = -0.6228, p(z) = -2.791, f(z) = -2.791, error = 1.7764e-15
r = 4
When z = -0.6228, p(z) = -3.071, f(z) = -3.071, error = 5.6577e-13
When z = 0.3731, p(z) = 0.063, f(z) = 0.063, error = 1.7028e-14
When z = -0.3550, p(z) = -1.241, f(z) = -1.241, error = 2.2515e-13
```

And we can observe that the error is near machine epsilon as well, so we can conclude that our implementation is approximately correct.

Finally, we can check the upper bound for the interpolation error, with the sin function, since $\dfrac{d\sin(x)}{dx} \leq 1$

$$\begin{aligned}\therefore |f(x) - P_n(x)| &= \left|\frac{(x-x_1)(x-x_2)\cdots(x-x_{n+1}))}{(n+1)!}f^{(n+1)}(c)\right| \\ &\leq \left|\frac{(x-x_1)(x-x_2)\cdots(x-x_{n+1}))}{(n+1)!}\right|\end{aligned}$$

Listing 10: test_sin.m function

```
clear all; clc;
format long;


% Initailize the data points of x,y
r = [2,3,4];
for ra =r
randn('seed',20190316);
x = linspace(0,pi/2,ra+1);
y = sin(x);

z =  pi/2*randn(3,1);
b = sin(z);

a = myPolyCoef(x,y);
fprintf('r = %d\n', ra);
for k = 1:length(z)
result = myPolyEval(z(k),x, a);
upperbound =1/factorial(ra+1);
for h = 1:length(x)
    upperbound = abs(upperbound *(z(k) - x(h)));
end
fprintf('When z = % 5.4f, p(z) = % 5.3f, f(z) = % 5.3f, error = %5.4e ub =
    %5.4e\n',z(k), result,b(k), abs(result-b(k)),upperbound);
end
end
```

And the output follows,

Listing 11: Output

```
test_sin
r = 2
When z = -0.0294, p(z) = -0.035, f(z) = -0.029, error = 5.1230e-03 ub = 6.3970e-03
When z = 0.5031, p(z) = 0.501, f(z) = 0.482, error = 1.8491e-02 ub = 2.5274e-02
When z = -1.8279, p(z) = -3.249, f(z) = -0.967, error = 2.2823e+00 ub = 2.7057e+00
r = 3
When z = -0.0294, p(z) = -0.030, f(z) = -0.029, error = 6.5941e-04 ub = 1.1686e-03
When z = 0.5031, p(z) = 0.482, f(z) = 0.482, error = 1.6241e-04 ub = 2.4984e-04
When z = -1.8279, p(z) = -1.389, f(z) = -0.967, error = 4.2139e-01 ub = 1.7500e+00
r = 4
When z = -0.0294, p(z) = -0.029, f(z) = -0.029, error = 1.2666e-04 ub = 1.6304e-04
When z = 0.5031, p(z) = 0.482, f(z) = 0.482, error = 6.8330e-05 ub = 9.4160e-05
When z = -1.8279, p(z) = -0.191, f(z) = -0.967, error = 7.7651e-01 ub = 9.0303e-01
```

We can see that all the error are bounded by the upperbound, so that the algorithm is correct.

(e) I implement the following form to interpolating $f(x) = \dfrac{1}{1+4x^2}$ with uniformly distributed $x \in [-2, 2]$:

Listing 12: uniform.m function

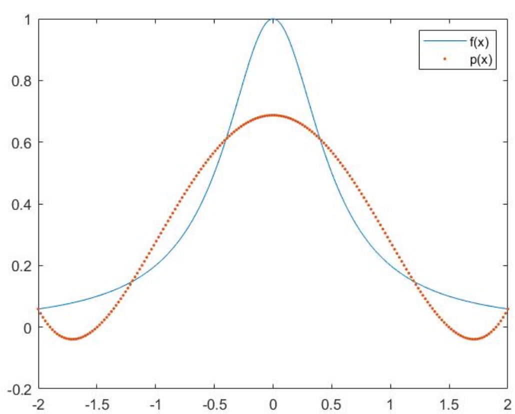
```
%% This program is to interplate f(x) = 1/(1+4x^2) with uniformly distributed x

n = 13; %We uniform distributed n points in [-2,2]
f = @(x) (1./(1+4.*x.^2)); % Define f(x)

%Initialize (x_j, y_j)
x=linspace(-2,2,n);
y=arrayfun(f,x);

%% Find the interpolation function
fprintf('n = %i\n',n);
%Find Coefficient
a = myPolyCoef(x,y);
for i = 1:n
    fprintf('a(%i) = % 15.14f\n', i, a(i));
end

%Find the polynomial function
p = @(b) a(n);
for j = n-1:-1:1
 p = @(b) p(b)*(b-x(1,j))+a(j);
end
x1=linspace(-2,2,200);

%Find the function value for each x
polyvec=zeros(1,200);
for i = 1:200
    polyvec(1,i) = p(x1(1,i));
end

%Draw the graph
plot(x1,f(x1));
hold on
plot(x1,polyvec,'.');
legend("f(x)", "p(x)");
```

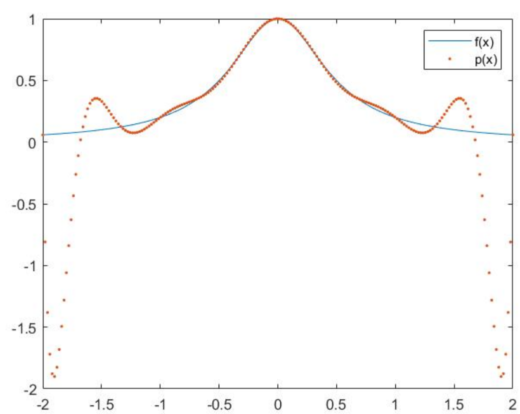
And we can get the output

Listing 13: Output

```
uniform
n = 6
a(1) = 0.05882352941176
a(2) = 0.11138183083884
a(3) = 0.29118878031802
a(4) = -0.27166300204596
a(5) = 0.08489468813936
a(6) = -0.00000000000000
uniform
n = 13
a(1) = 0.05882352941176
a(2) = 0.07123583378305
a(3) = 0.07638113684584
a(4) = 0.08558945508579
a(5) = 0.09542585514789
a(6) = 0.02723926171043
a(7) = -0.34538440497766
a(8) = 0.24338457216589
a(9) = 0.14687975924895
a(10) = -0.39054733673131
a(11) = 0.37696308154066
a(12) = -0.24451659343178
a(13) = 0.12225829671589
```

And from the figure below, $n = 6$ on the left, and $n = 13$ on the right, we can discover that when $n = 13$, the interpolated polynomial gets far away from the original function at some points than $n = 6$, this is because of "Runge's" phenomenon.





(f) Since the uniform distributed seems not good at some point, so we can use "Chebyshev nodes":

Listing 14: Chebyshev.m function

```
%% This program is to interplate f(x) = 1/(1+4x^2) with chebyshev nodes


n = 6; %# of Chebyshev nodes
```

```
f = @(x) (1./(1+4.*x.^2));% Define f(x)

%Initialize (x_j, y_j)
x = zeros(1,n);
for i = 1:n
    x(i) = 2*cos((2*i-1)*pi/(2*n));
end
y=arrayfun(f,x);
%% Find the interpolation function
fprintf('n = %i\n',n);
%Find Coefficient
a = myPolyCoef(x,y);
for i = 1:n
    fprintf('a(%i) = % 15.14f\n', i, a(i));
end

%Find the polynomial function
p = @(b) a(n);
for j = n-1:-1:1
 p = @(b) p(b)*(b-x(1,j))+a(j);
end
x1=linspace(-2,2,200);

%Find the function value for each x
polyvec=zeros(1,200);
for i = 1:200
    polyvec(1,i) = p(x1(1,i));
end

%Draw the graph
plot(x1,f(x1));
hold on
plot(x1,polyvec,'.');

legend("f(x)", "p(x)");
```

And we can get the output

Listing 15: Output

```
chebyshev
n = 6
a(1) = 0.06278172029468
a(2) = -0.09336521351681
a(3) = 0.22702230923301
a(4) = 0.18025940551926
a(5) = 0.05387205387205
a(6) = -0.00000000000000
chebyshev
n = 13
a(1) = 0.05963906009257
a(2) = -0.06136485429509
a(3) = 0.05275725374458
a(4) = -0.04750842653948
a(5) = 0.04884509239445
a(6) = -0.04439530891621
```

```
a(7) = -0.11038442682151
a(8) = -0.01761291631395
a(9) = 0.07683484938961
a(10) = 0.09504012818094
a(11) = 0.06949309964432
a(12) = 0.03984434714485
a(13) = 0.02006849549978
```

And from the figure below, we can see that the polynomial meet much better to the original function than before, when we do it with uniform distributed nodes.