1.

result:

```
>> Q1
(a)
   ans for (a) :
   0.775107121875000

(b)
   ans for (b) :
   0.775123777656250

(c)
   En(x) = value of next term that would be added to Pn(x)
   Error for (a) :
     1.665578124999997e-05

   Error for (b) :
    -2.450202148437495e-06

(d)
   Error when x0 = 0.24
     2.137906249999995e-05

   Error when x0 = 0.36
     2.485937499999995e-05

   It is better to start with x0 = 0.24 because it will have smaller error if getting f(0.42) quadratically
```

plug in the values into the polynomial, details are in code.

2.

code:

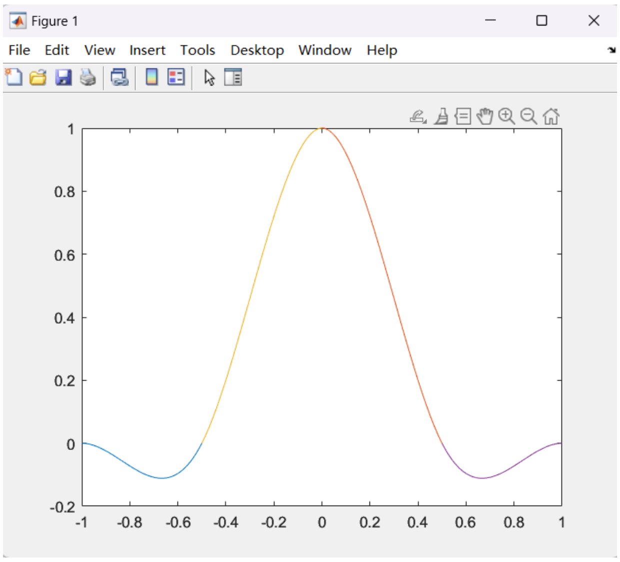
```
format long;
h = 1/2;                                      % x(n+1) - x(n)
y = [0 0 1 0 0]';                             % corresponding y value on x
Y = 6*[0 2 -4 2 0 ]';                         % Y
H = [2*h h 0 0 0;                             % H
    h 4*h h 0 0 ;
    0 h 4*h h 0 ;
    0 0 h 4*h h ;
    0 0 0 h 2*h];
S = H\Y;                                      % HS = Y
disp("S :")
disp(S)
a = 1/6/h.*[S(2)-S(1) S(3)-S(2) S(4)-S(3) S(5)-S(4)]';
b = 1/2.*S(1:4);
d = [0  0 1 0]';
c = (1/h).*(y(2:5)-y(1:4)) - (1/6).*(2*h.*S(1:4)+h.*S(2:5));

abcd = [a b c d];
disp("[a b c d] :")
disp(abcd)
temp = 0;
x0 = linspace(-1+temp*h, -1+temp*h+h, 101);      % segment x0 to x1
y0 = polyval(abcd(temp+1,1:4),x0+1);             % plug in y = a(x-x0)^3 + b(x-x0)^2 + c(x-x0) +d
temp = temp+1;

x1 = linspace(-1+temp*h, -1+temp*h+h, 101);
y1 = polyval(abcd(temp+1,1:4),x1+1/2);
temp = temp+1;

x2 = linspace(-1+temp*h, -1+temp*h+h, 101);
y2 = polyval(abcd(temp+1,1:4),x2);
temp = temp+1;
x3 = linspace(-1+temp*h, -1+temp*h+h, 101);
y3 = polyval(abcd(temp+1,1:4),x3-1/2);
temp = temp+1;
plot(x0,y0, x2,y2, x1, y1, x3,y3)  % plot
```

I choose x = [-1, -1/2, 0, 1/2, 1]’.

The following result shows the S I got and the coefficients of each polynomial in the form of [a0, b0, c0, d0; a1, b1, c1, d1; ….]

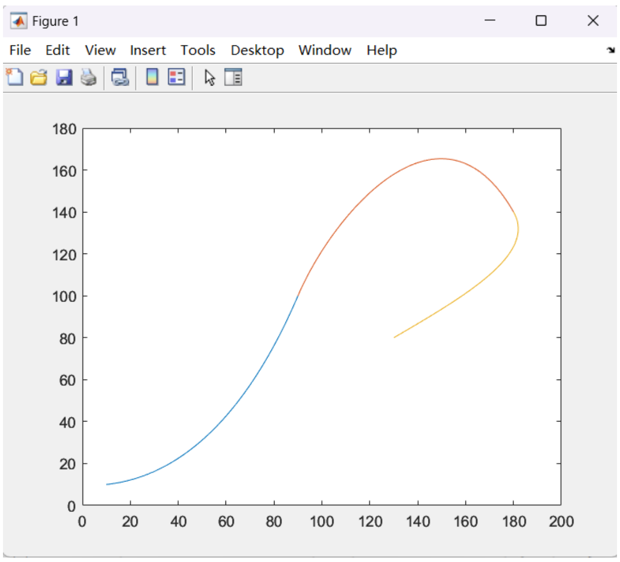
```
>> Q2
S :
  -6.000000000000000
  12.000000000000000
 -17.999999999999996
  11.999999999999998
  -5.999999999999999

[a b c d] :
   6.000000000000000  -3.000000000000000                   0                   0
  -9.999999999999998   6.000000000000000   1.500000000000000                   0
   9.999999999999996  -8.999999999999998  -0.000000000000001   1.000000000000000
  -5.999999999999998   5.999999999999999  -1.500000000000000                   0
```

figure:



3.

(a)



(b)

It is because point 2, 3, 4 form a straight line and point 4, 5, 6 also forms a straight line. Suppose there are two Bezier curve P1(p10, p11, p12, p13), P2(p21, p22, p23, p24), and p13 is equal to p21. The slope of P2'(0) is computed by 3(p11-p10) ∝ the slope of p21 and p22. Similarly, P1'(1) ∝ the slope of p12 and p13 = the slope of p12 and p21. Therefore, if the p12, p13, and p21 forms a straight line, the two will have same slope on p13 and achive C2 continuity and look smooth.

(c) I change Q3_a into

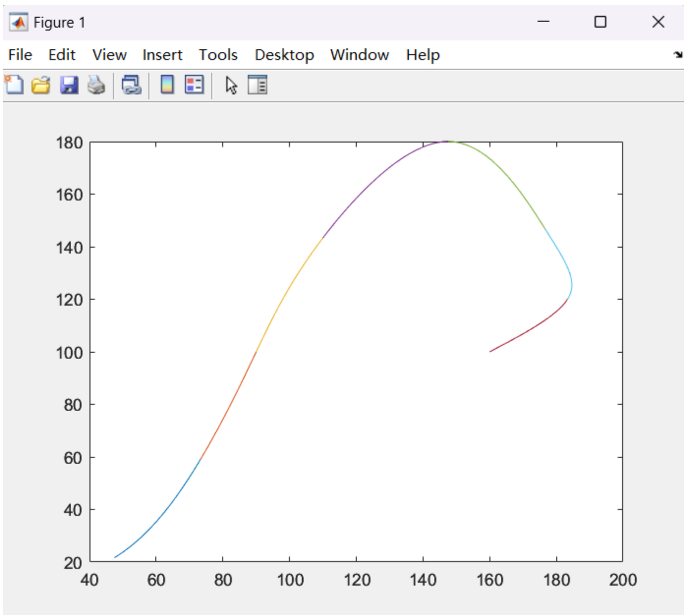
```
format long;
u1 = linspace(0,1);
matrix = [2 -2 1 1;-3 3 -2 -1;0 0 1 0 ;1 0 0 0]*[1 0 0 0;0 0 0 1;-3 3 0 0;0 0 -3 3];

p1 = matrix*[10 10;50 15;75 60 ;90 100];
x1 = u1.^3*p1(1,1)+u1.^2*p1(2,1)+u1*p1(3,1)+p1(4,1);
y1 = u1.^3*p1(1,2)+u1.^2*p1(2,2)+u1*p1(3,2)+p1(4,2);

u2 = linspace(1,2);
p2 = matrix*[90 100;105 140;150 200;180 140];
x2 = (u2-1).^3*p2(1,1)+(u2-1).^2*p2(2,1)+(u2-1)*p2(3,1)+p2(4,1);
y2 = (u2-1).^3*p2(1,2)+(u2-1).^2*p2(2,2)+(u2-1)*p2(3,2)+p2(4,2);

u3 = linspace(2,3);
p3 = matrix*[180 140;190 120;160 100;130 80];
x3 = (u3-2).^3*p3(1,1)+(u3-2).^2*p3(2,1)+(u3-2)*p3(3,1)+p3(4,1);
y3 = (u3-2).^3*p3(1,2)+(u3-2).^2*p3(2,2)+(u3-2)*p3(3,2)+p3(4,2);

plot(x1, y1, x2, y2,x3,y3)
```

The result is same as the graph in 3(a) and now u is in [0,1], [1,2], [2,3] for the segments.

4.

(a)

result:



compute the segments of all segments like the following code (the rest is in the code)

```
format long;
u1 = linspace(0,1);
matrix = (1/6)*[-1 3 -3 1;3 -6 3 0;-3 0 3 0; 1 4 1 0];

pts = [10 10;50 15;75 60;90 100;105 140;150 200;180 140;190 120;160 100;130 80];
count = 1;
p1 = matrix*[pts(count,1:2);pts(count+1,1:2);pts(count+2,1:2) ;pts(count+3,1:2)];
x1 = u1.^3*p1(1,1)+u1.^2*p1(2,1)+u1*p1(3,1)+p1(4,1);
y1 = u1.^3*p1(1,2)+u1.^2*p1(2,2)+u1*p1(3,2)+p1(4,2);
count = count +1;

u2 = u1;
p2 = matrix*[pts(count,1:2);pts(count+1,1:2);pts(count+2,1:2) ;pts(count+3,1:2)];
x2 = u2.^3*p2(1,1)+u2.^2*p2(2,1)+u2*p2(3,1)+p2(4,1);
y2 = u2.^3*p2(1,2)+u2.^2*p2(2,2)+u2*p2(3,2)+p2(4,2);
count = count +1;
```

(b)

The curve 6 since it is a B-spline curve, but it pass point 3 because point 3 is the average of point 2 and 4. The whole curve is smooth because B-spline curve is designed in a way achieving C2 continuity.

(c)

Do the same thing as 3. (c), details in code Q4_c.

5.

```
format long;
A = [1 0.4 0.7;1 1.2 2.1; 1 3.4 4 ;1 4.1 4.9; 1 5.7 6.3 ;1 7.2 8.1 ;1 9.3 8.9];
At = A';
AtA = At*A;
b = [0.031 0.933 3.058 3.349 4.87 5.757 8.921]';
Atb = At*b;
a = AtA\Atb;
disp("(a)")
disp("   A'Aa = A'b, where")
disp("   A = ")
disp(A)
disp("   B = ")
disp(b)
fprintf("(b)\n")
fprintf("   z = %d * x + %d * y + %d\n", a(2),a(3), a(1))

fprintf("(c)\n")

fprintf("   sum of square = %d\n",sum((b-A*a).*(b-A*a)))
```

result:

```
>> Q5
(a)
   A'Aa = A'b, where
   A = 
   1.000000000000000   0.400000000000000   0.700000000000000
   1.000000000000000   1.200000000000000   2.100000000000000
   1.000000000000000   3.400000000000000   4.000000000000000
   1.000000000000000   4.100000000000000   4.900000000000000
   1.000000000000000   5.700000000000000   6.300000000000000
   1.000000000000000   7.200000000000000   8.100000000000000
   1.000000000000000   9.300000000000001   8.900000000000000

   B = 
   0.031000000000000
   0.933000000000000
   3.058000000000000
   3.349000000000000
   4.870000000000000
   5.757000000000000
   8.920999999999999

(b)
   z = 1.596092e+00 * x + -7.023814e-01 * y + 2.206660e-01
(c)
   sum of square = 3.193951e-01
```

6.

code: add condition to the order of polynomials in the pade function in matlab

```
% find pade approximation

syms x;
p = pade(cos(x)^2, x, 0,'Order',[3 3]);
disp("for cos(x)^2 : ")
disp(p)

disp("for sin(x^4-x) :")
p = pade(sin(x^4-x), x, 0,'Order',[3 3]);
disp(p)

disp("for x*exp(x) :")
p = pade(x*exp(x), x, 0,'Order',[3 3]);
disp(p)
```

result:

```
>> Q6
for cos(x)^2 :
-(2*x^2 - 3)/(x^2 + 3)

for sin(x^4-x) :
-(x*(21649*x^2 - 6120*x + 129180))/(3*(42720*x^3 + 14393*x^2 - 2040*x + 43060))

for x*exp(x) :
-(3*x*(x^2 + 8*x + 20))/(x^3 - 9*x^2 + 36*x - 60)
```

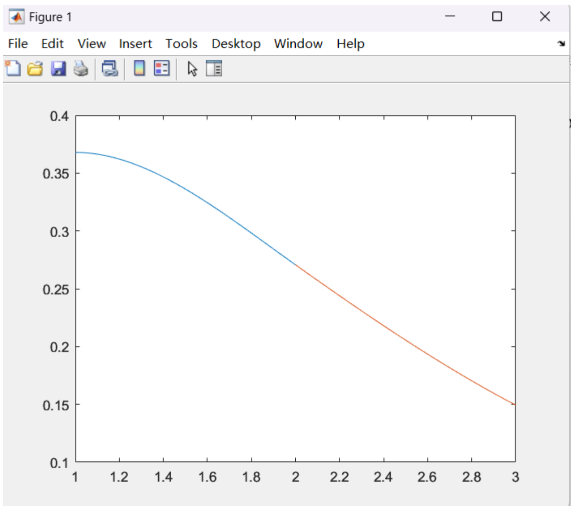
This can also be computed by hand with the following steps

1. Get the taylor series to the power of 4, which is f(0) + f'(0) x + f''(0) x^2 / 2! + f'''(0) x^3 / 3!
2. Multiply with 1 + b1*x + b2*x^2 + b3*x^3, and get a polynomial of order 6
3. Map each coefficients of x in the polynomial to a0 + a1*x + a2*x^2 /2! + a3 * x^3 / 3!, since there are seven variables and seven equations, you can solve all the variables.

7.

(a)

result:



code:

```
syms x;
f = x*exp(-x);
fd = diff(f);
matrix = [2 -2 1 1;-3 3 -2 -1 ;0  0 1 0; 1 0 0 0];                                     % M

u1 = linspace(0,1,101);
p1 = matrix*[1 vpa(subs(f,x,1)); 2 vpa(subs(f,x,2));1 vpa(subs(fd,x,1));1 vpa(subs(fd,x,2))]; % MP
x1 = u1.^3*p1(1,1)+u1.^2*p1(2,1)+u1*p1(3,1)+p1(4,1);                                   % u'MP(1) = x(u)
y1 = u1.^3*p1(1,2)+u1.^2*p1(2,2)+u1*p1(3,2)+p1(4,2);                                   % u'MP(2) = y(u)

p2 = matrix*[2 vpa(subs(f,x,2)); 3 vpa(subs(f,x,3));1 vpa(subs(fd,x,2));1 vpa(subs(fd,x,3)) ];
x2 = u1.^3*p2(1,1)+u1.^2*p2(2,1)+u1*p2(3,1)+p2(4,1);
y2 = u1.^3*p2(1,2)+u1.^2*p2(2,2)+u1*p2(3,2)+p2(4,2);

plot(x1,y1,x2,y2)
```

(b)

Since the 51th point of the first curve is 1.5, so I pick out the point from the curve directly.

```
fprintf("when x = %f\ny = %f\n", x1(51),y1(51));
```

result:

```
>> Q7
when x = 1.500000
y = 0.336192
```