Numerical Methods Homework-7

B10602110 四電子三乙 呂和軒

1.

Generate eight points from the function:

$$f(t) = \sin^2 t$$

from $t = 0$ to 2π. Fit these data using:

(a) Cubic spline with not-a-knot end conditions.

ANS :

1.Code_function :

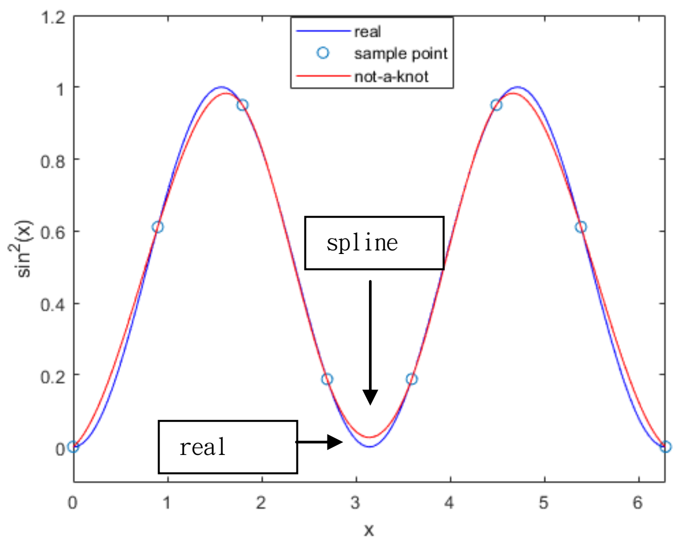
```
function [p] = Cubic_spline(t,y,xx,cond)
    n = length(t) - 1;
    p = [];
    f =@(x,x_o,a,b,c,d) a+b.*(x-x_o)+c.*(x-x_o).^2+d.*(x-x_o).^3
    A = zeros(n+1,n+1);
    if(cond == 'not-a-knot')
       for i=1:n
           h(i) = t(i+1)-t(i);
           if(i==1)
              yy(i) = 0;
           else
              yy(i) = (y(i+1)-y(i))/h(i) - (y(i)-y(i-1))/h(i-1);
           end
       end
       yy(n+1) = 0;
       yy = yy.*6;
       A(1,1:3) = [-h(2),h(1)+h(2),-h(1)];
       A(n+1,n+1-2:n+1) = [-h(n),h(n-1)+h(n),-h(n-1)];
       for i = 2:n
         A(i,i-1) = h(i-1);
         A(i,i)   = 2*(h(i-1)+h(i));
         A(i,i+1) = h(i);
       end
       m = inv(A)*yy'
    end
    if(cond == 'derivative')
       for i=1:n
           h(i) = t(i+1)-t(i);
           if(i==1)
              yy(i) = (y(i+1)-y(i))/h(i) - y(1); % A = y(1)
           else
              yy(i) = (y(i+1)-y(i))/h(i) - (y(i)-y(i-1))/h(i-1);
           end
       end
       yy(n+1) = y(n+1)-(y(n+1)-y(n))/h(n); % B = y(n+1);
       yy = yy.*6;
       A(1,1:3) = [2*h(1),h(1),0];
       A(n+1,n+1-2:n+1) = [0,h(n),2*h(n)];
       for i = 2:n
         A(i,i-1) = h(i-1);
         A(i,i)   = 2*(h(i-1)+h(i));
         A(i,i+1) = h(i);
       end
       m = inv(A)*yy'
```

```
    end
    for i=1:n
       a(i) = y(i)
       b(i) = (y(i+1)-y(i))/h(i)-h(i)*m(i)/2-(h(i)/6)*(m(i+1)-m(i));
       c(i) = m(i)/2;
       d(i) = (m(i+1)-m(i))/(6*h(i));
       if(i == 1)
        tt = xx(xx >= t(i) & xx <= t(i+1));
       else
        tt = xx(xx > t(i) & xx <= t(i+1));
       end
       p = [p,f(tt,t(i),a(i),b(i),c(i),d(i))];
    end
end
```

2.Code_main:

```
close all
clear all
format long

f = @(x) sin(x).^2;
t= linspace(0,2*pi,8);
y = f(t);
xx = 0:0.01:2*pi;
[p] = Cubic_spline(t,y,xx,'not-a-knot');
plot(xx,f(xx),'b-')
hold on
plot(t,f(t),'o')
hold on
plot(xx,p,'r-')
hold on
xlabel('x')
ylabel('sin^2(x)')
M = ["real";"sample point";"not-a-knot"]
hold on
legend(M)
xlim([0,2*pi])
ylim([-0.1,1.2])
```

3.Result:



(b) Cubic spline with derivative end conditions equal to the exact values calculated with differentiation.

ANS :

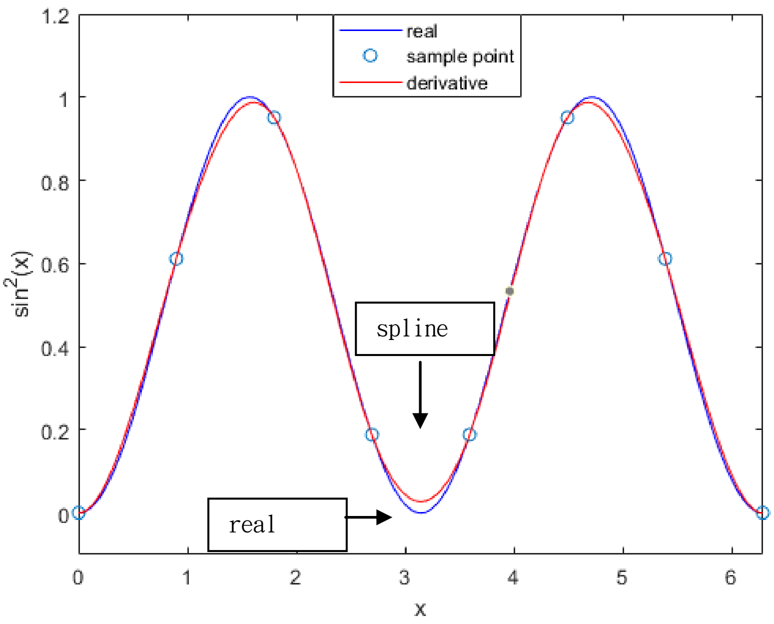
1.Code_function :

Equal to part(a)

2.Code_main:

```
close all
clear all
format long

f = @(x) sin(x).^2;
t= linspace(0,2*pi,8);
y = f(t);
xx = 0:0.01:2*pi;
[p] = Cubic_spline(t,y,xx,'derivative');
plot(xx,f(xx),'b-')
hold on
plot(t,f(t),'o')
hold on
plot(xx,p,'r-')
hold on
xlabel('x')
ylabel('sin^2(x)')
M = ["real";"sample point";"derivative"]
hold on
legend(M)
xlim([0,2*pi])
ylim([-0.1,1.2])
```

3.Result:

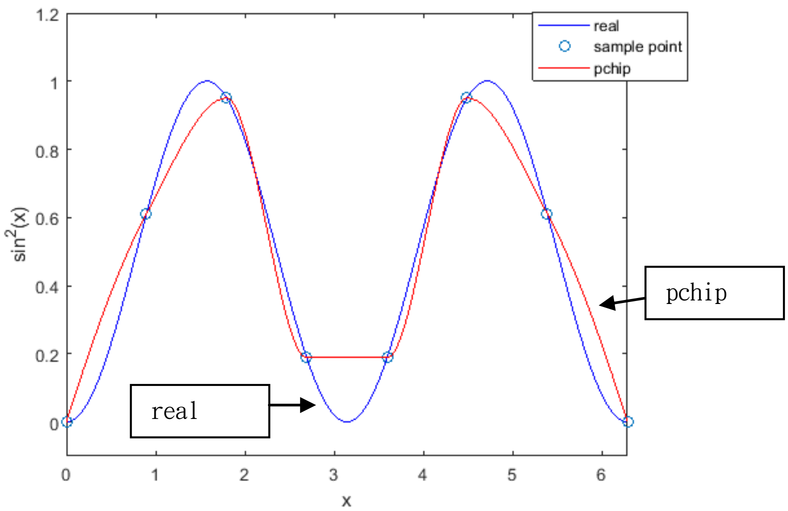


(c) Piecewise cubic Hermite interpolation (PCHIP).

1.Code_main:

```
close all
clear all
format long

f = @(x) sin(x).^2;
t= linspace(0,2*pi,8);
y = f(t);
xx = 0:0.01:2*pi;
yy = pchip(t,y,xx);
plot(xx,f(xx),'b-')
hold on
plot(t,f(t),'o')
hold on
plot(xx,yy,'r-')
xlabel('x')
ylabel('sin^2(x)')
M = ["real";"sample point";"pchip"]
hold on
legend(M)
xlim([0,2*pi])
ylim([-0.1,1.2])
```

2.Result:



2.

Given the data:

x	1	2	3	5	7	8
$f(x)$	3	6	19	99	291	444

Calculate $f(4)$ using Newton's interpolating polynomials of order 1 through 4. Choose your base points to attain good accuracy. What do your results indicate regarding the order of the polynomial used to generate the data in the table?

ANS :

1.Code_function :

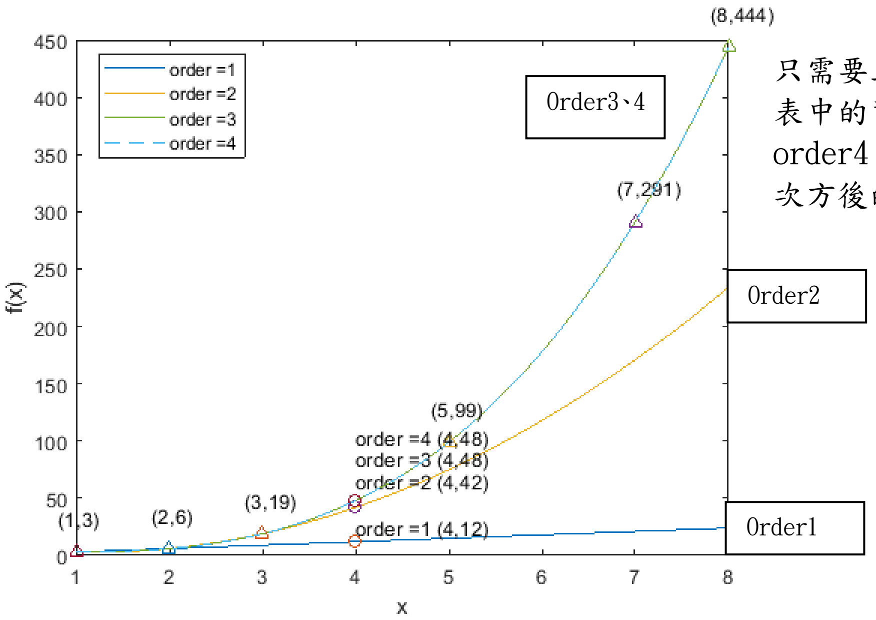
```
function [d] = Newton_interpo(x,y)
    cof = [];
    n = length(y);
    d = zeros(n,n);
    d(:,1) = y';
    for i = 2:n
        for j = 1:n-i+1
            d(j,i) = (d(j+1,i-1)-d(j,i-1))/(x(i+j-1)-x(j))
        end
    end
end
```

2.Code_main:

```
close all
clear all
format long

x = [1,2,3,5,7,8];
y = [3,6,19,99,291,444];
cof = Newton_interpo(x,y);

n_t = 4; % order
M = [];
l = [];
for n = 2:n_t+1 % a1~an
    yy = cof(1,1);
    x_it = 1:0.01:8;
    x_point = 4
    for i = 2:n
        p = 1;
        for j = 1:i-1
           p  = p.*(x_it-x(j)) % x-x0 ...
        end
        yy = yy + p.*cof(1,i); % a1 *(x-x0)...
    end
    if(n == 5)
         l=[l;plot(x_it,yy,'--')]
    else
         l=[l;plot(x_it,yy,'-')]
    end
    xlabel('x')
    ylabel('f(x)')
    hold on
    plot(x_point,yy(find(x_it==4)),'o')
    text(x_point,yy(find(x_it==4))+n*n*2+5,'order =' + string(n-1)+'
('+string(x_point)+','+string(yy(find(x_it==4)))+')')
    M = [M;'order =' + string(n-1)]
    if(n==4)
        for i =1:length(x)
            y_ = yy(find(x_it == x(i)));
            plot(x(i),y_,'^')
            hold on

text(x(i)-0.2,y_+30,'('+string(x(i))+','+string(y_)+')')
        end
    end
end
```

```
legend(l,M)
```

3.Result:



只需要三次多項式即可滿足表中的資料，因為 order 3 及 order4 為相同曲線，且 x 三次方後的係數皆為 0。

3.

Repeat the above question using Lagrange polynomials of order 1 through 3.

ANS :

1.Code_function :

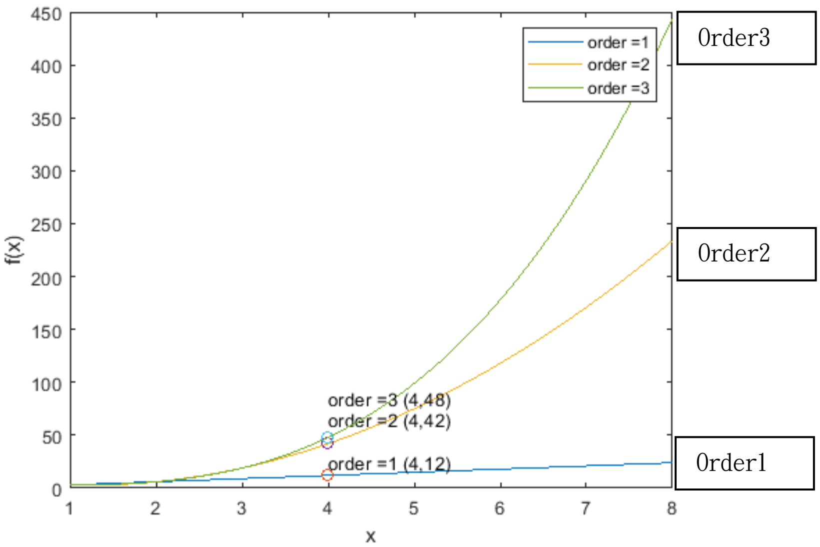
```
function [d] = Lagrange_pol(x,y,n)
    %n = length(y);
    d = y
    for i = 1:n
        for j = 1:n
            if(j ~= i)
                d(1,i) = d(1,i)/(x(i)-x(j))
            end
        end
    end
end
```

2.Code_main:

```
close all
clear all
format long

x = [1,2,3,5,7,8];
y = [3,6,19,99,291,444];

n_t = 3;
M = [];
l = [];
for n = 2:n_t+1
    cof = Lagrange_pol(x,y,n);
    yy = 0;
```

```
    x_it = 1:0.01:8;
    x_point = 4
    for i = 1:n
        p = 1;
        for j = 1:n
            if(i~=j)
                p  = p.*(x_it-x(j)); % x-x0 ...
            end
        end
        yy = yy + p.*cof(1,i); % a1 *(x-x0)...
    end
    l=[l;plot(x_it,yy,'-')]
    xlabel('x')
    ylabel('f(x)')
    hold on
    plot(x_point,yy(find(x_it==4)),'o')
    text(x_point,yy(find(x_it==4))+n*n*2+5,'order =' + string(n-1)+'
('+string(x_point)+','+string(yy(find(x_it==4)))+')')
    M = [M;'order =' + string(n-1)]
end
legend(l,M)
```

3.Result:



4.

Runge's function is written as:

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

(a) Develop a plot of this function for the interval from $x = -1$ to 1.

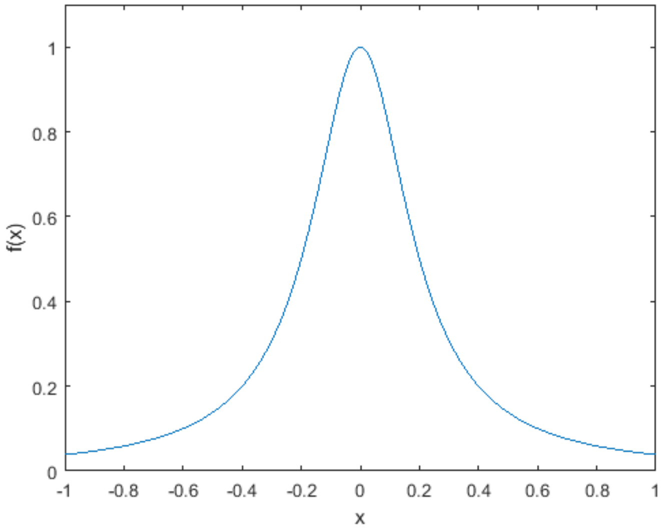
ANS:

1.Code_main:

```
close all
clear all
format long
```

```
f = @(x) 1./(1+25.*x.^2);
n= linspace(-1,1,1000);
plot(n,f(n))
xlabel('x')
ylabel('f(x)')
xlim([-1,1])
ylim([0,1.1])
```

2.Result:



(b) Generate and plot e forth-order Lagrange interpolating polynomial using equispaced function values corresponding to $x = -1, -0.5, 0, 0.5, \text{and } 1$.

ANS :

1.Code_function :

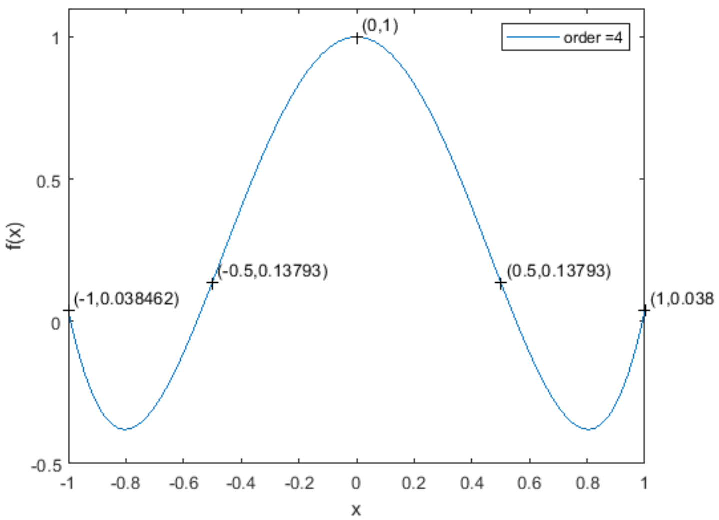
Equal to Part 3 (`Lagrange_pol`)

2.Code_main:

```
close all
clear all
format long

f = @(x) 1./(1+25.*x.^2);
x = -1:0.5:1;
y = f(x);

n = 5; %order 4
M = [];
l = [];
cof = Lagrange_pol(x,y,n);
yy = 0;
x_it = -1:0.01:1;
for i = 1:n
    p = 1;
    for j = 1:n
        if(i~=j)
            p  = p.*(x_it-x(j)); % x-x0 ...
        end
```

```
    end
    yy = yy + p.*cof(1,i); % a1 *(x-x0)...
end
l=[l;plot(x_it,yy,'-')]
xlabel('x')
ylabel('f(x)')
hold on
for i = 1:length(x)
    y_ = yy(find(x_it==x(i)))
    plot(x(i),y_,'k+')
    hold on
    text(x(i),y_+0.05,' ('+string(x(i))+','+string(y_)+')')
end
M = [M;'order =' + string(n-1)]
legend(l,M)
xlim([-1,1])
ylim([-0.5,1.1])
```

3.Result:



(c) Use the five points from (b) to estimate *f*(0.8) with first- through fourth-order Newton interpolating polynomials.

ANS :

1.Code_function :

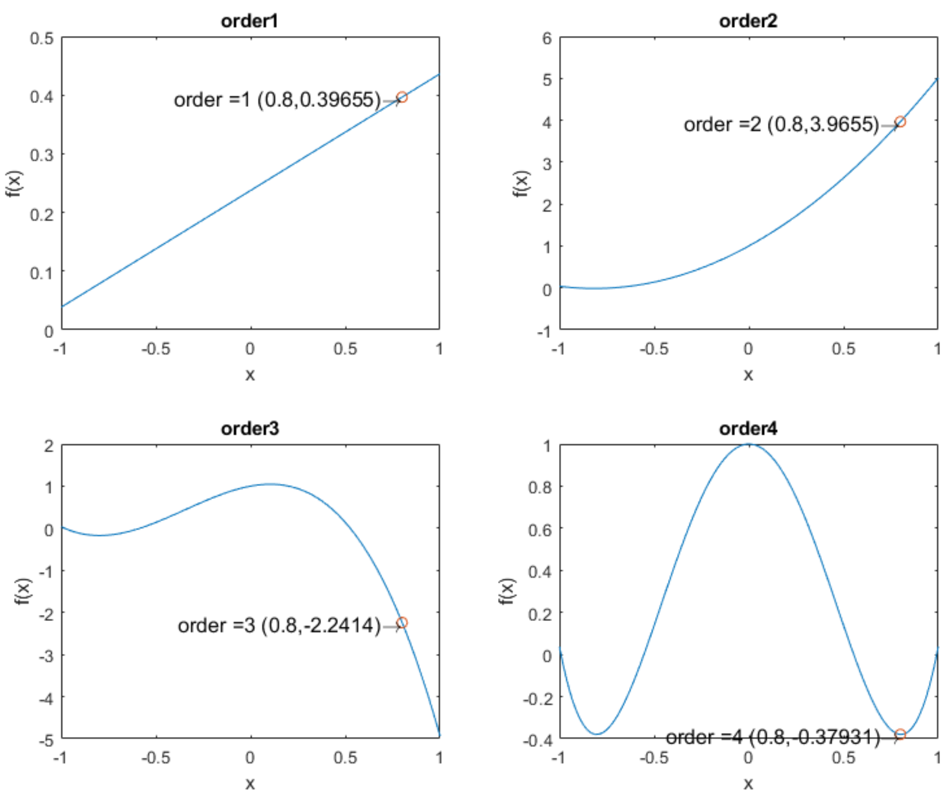
Equal to Part 2 (`Newton_interpo`)

2.Code_main:

```
close all
clear all
format long

f = @(x) 1./(1+25.*x.^2);
x = -1:0.5:1;
y = f(x);
cof = Newton_interpo(x,y);

n_t = 4; % order
```

```
for n = 2:n_t+1 % a1~an
    M = [];
    l = [];
    subplot(2,2,n-1);

    yy = cof(1,1);
    x_it = -1:0.01:1;
    x_point = 0.8
    for i = 2:n
        p = 1;
        for j = 1:i-1
            p  = p.*(x_it-x(j)); % x-x0 ...
        end
        yy = yy + p.*cof(1,i); % a1 *(x-x0)...
    end
    l=[l;plot(x_it,yy,'-')];
    xlabel('x')
    ylabel('f(x)')
    hold on
    plot(x_point,yy(find(x_it==x_point)),'o')
    hold on
    text(x_point,yy(find(x_it==x_point)),'order =' + string(n-1)+'
('+string(x_point)+','+string(yy(find(x_it==x_point)))+')\rightar
row', ...,
        'HorizontalAlignment','right','FontSize',8)
    M = [M;'order =' + string(n-1)];
    title('order'+string(n-1))
end
```

3.Result:



(d) Generate and plot a cubic spline using the five points from (b).

ANS :

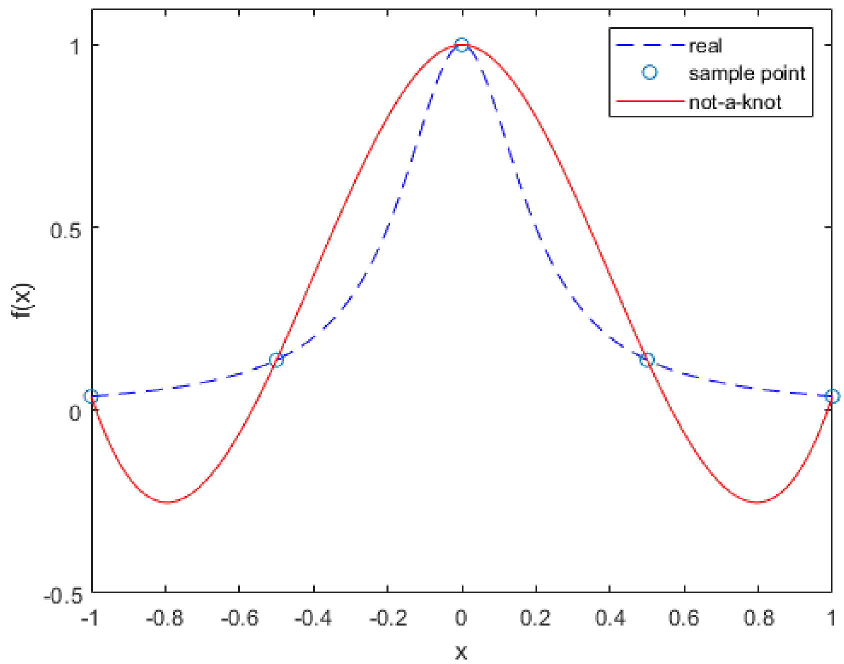
1. Code_function :

Equal to Part 1 (`Cubic_spline`)

2.Code_main:

```
close all
clear all
format long

f = @(x) 1./(1+25.*x.^2);
x = -1:0.5:1;
y = f(x);

xx = -1:0.01:1;
[p] = Cubic_spline(x,y,xx,'not-a-knot');
plot(xx,f(xx),'b--')
hold on
plot(x,f(x),'o')
hold on
plot(xx,p,'r-')
hold on
xlabel('x')
ylabel('f(x)')
M = ["real";"sample point";"not-a-knot"]
hold on
legend(M)
xlim([-1,1])
ylim([-0.5,1.1])
```

3.Result:



(e) Discuss your results.

ANS :

由函式 f(x)的泰勒展開式可知，若要在-1 至 1 之間使用多項式擬合 f(x)，需要無窮多項，因此只使用四次方多項式內插出來的曲線與原曲線誤差較大。