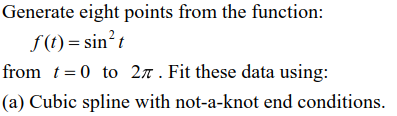
Numerical Methods Homework-7

B10602110 四電子三乙 呂和軒

1.



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

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;

[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"]

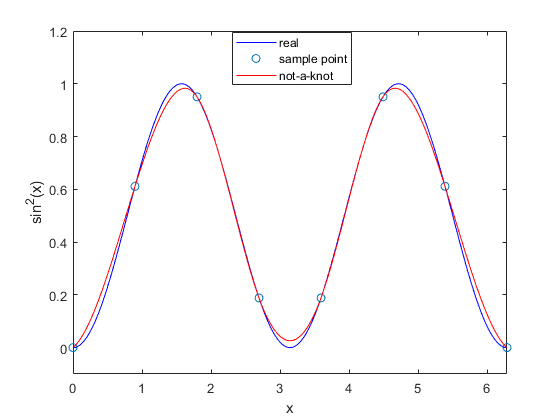
hold on

legend(M)

xlim([0,2\*pi])

ylim([-0.1,1.2])

1. Result:



real

spline



ANS :

1. Code\_function :

Equal to part(a)

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;

[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"]

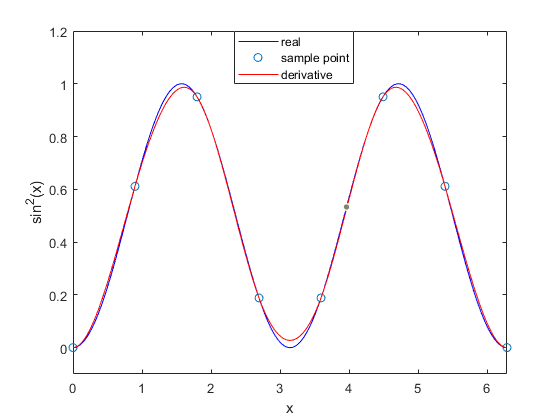
hold on

legend(M)

xlim([0,2\*pi])

ylim([-0.1,1.2])

1. Result:



spline

real



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"]

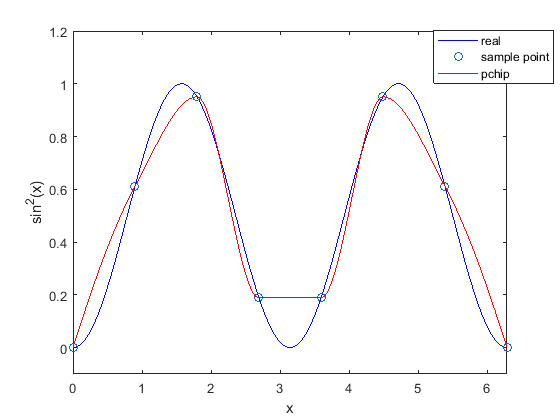
hold on

legend(M)

xlim([0,2\*pi])

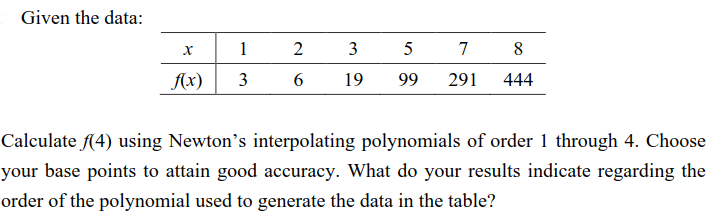
ylim([-0.1,1.2])

1. Result:



pchip

real

2. 

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

1. 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\_)+')')

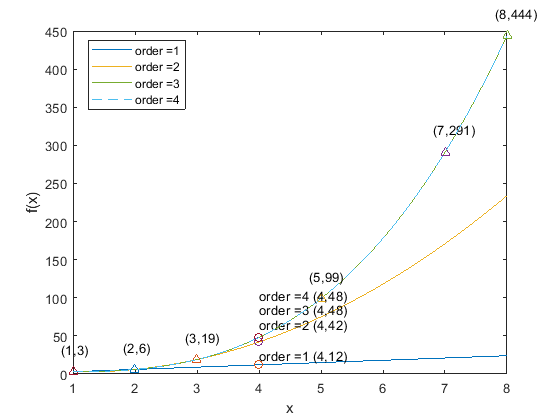
end

end

end

legend(l,M)

1. Result:



Order3、4

Order2

Order1

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. 

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

1. 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')

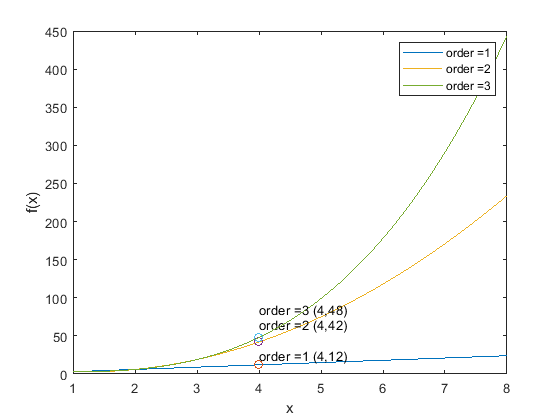
text(x\_point,yy(find(x\_it==4))+n\*n\*2+5,'order =' + string(n-1)+' ('+string(x\_point)+','+string(yy(find(x\_it==4)))+')')

M = [M;'order =' + string(n-1)]

end

legend(l,M)

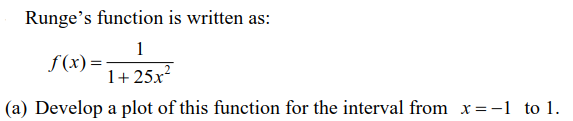
1. Result:



Order1

Order3

Order2

4. 

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))

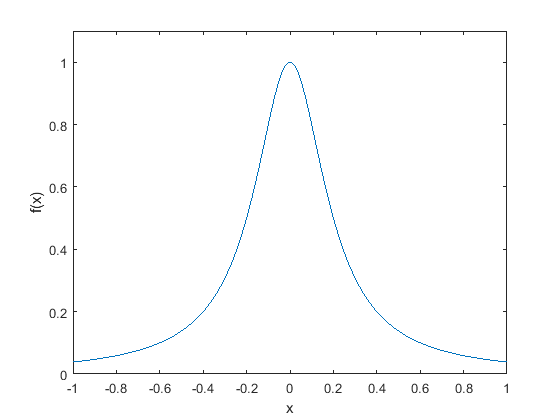
xlabel('x')

ylabel('f(x)')

xlim([-1,1])

ylim([0,1.1])

1. Result:





ANS :

1. Code\_function :

Equal to Part 3 (Lagrange\_pol)

1. 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

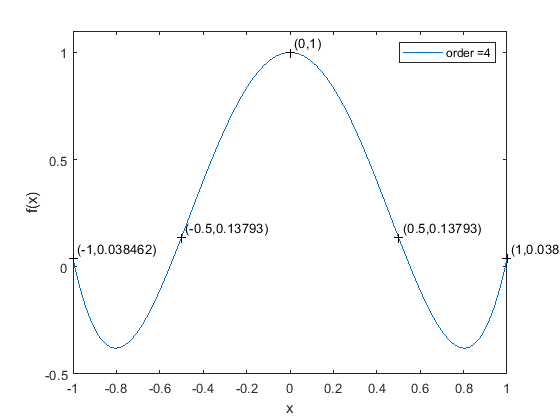
M = [M;'order =' + string(n-1)]

legend(l,M)

xlim([-1,1])

ylim([-0.5,1.1])

1. Result:





ANS :

1. Code\_function :

Equal to Part 2 (Newton\_interpo)

1. 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)))+')\rightarrow', ...,

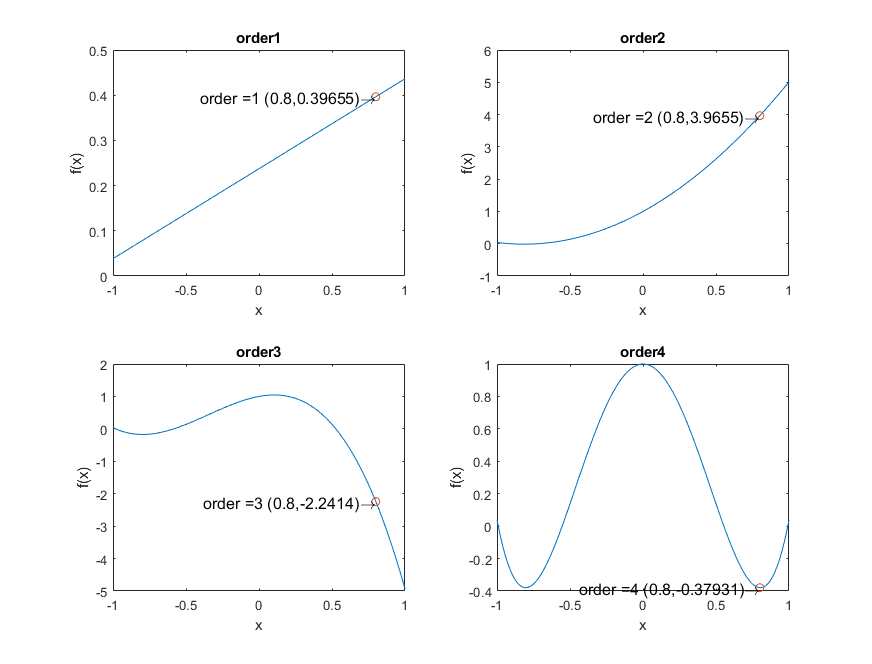
'HorizontalAlignment','right','FontSize',8)

M = [M;'order =' + string(n-1)];

title('order'+string(n-1))

end

1. Result:





ANS :

1. Code\_function :

Equal to Part 1 (Cubic\_spline)

1. 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"]

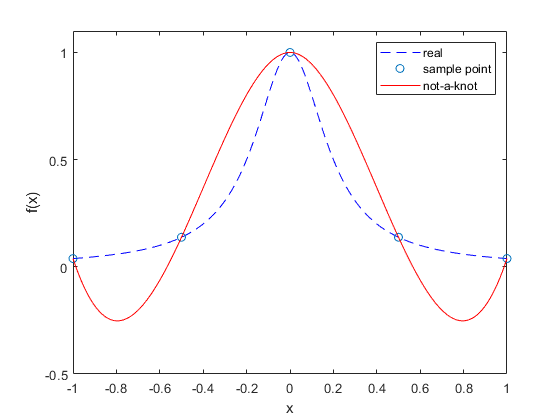
hold on

legend(M)

xlim([-1,1])

ylim([-0.5,1.1])

1. Result:





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。