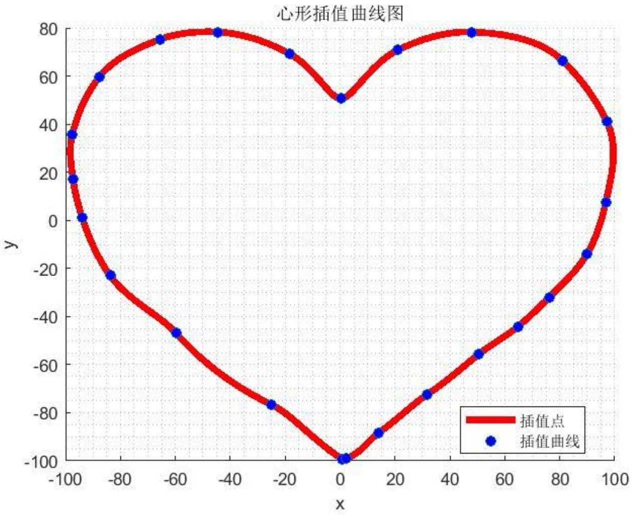
# MATLAB 第 4 次作业参考答案

几点反馈：

- 作业的参考答案来自于同学们的作业，只是稍作修正，为保护隐私，隐去姓名。
- 我们学的大多数算法其实都可以避免使用符号变量，使得运算效率更高。另外，大家也要注重对程序的优化，让程序的效率更高，可读性、简洁性更高。

第一题




```
test1.m
clear;clc;
pic=imread('h.jpg');% 输入图片
imshow(pic);% 显示图片
[x,y]=ginput;% 读取坐标
% 转换坐标并中心化
y=mean(y)-y;   x=x-mean(x);
%转换为极坐标形式
[theta,r]=cart2pol(x,y);
%调整角度到-pi/2~3*pi/2
k=find(theta<-pi/2);
%为使曲线闭合，将最小角度插值点角度增加 2pi，长度不变
theta(k)=theta(k)+2*pi;
l=find(theta==min(theta));
theta=[theta;theta(l)+2*pi]; r=[r;r(l)];
u=linspace(min(theta),max(theta),1000);% 生成线性序列
v=pchip(theta,r,u);% 插值运算
[u,v]=pol2cart(u,v);%还原至直角坐标
plot(u,v,'r','LineWidth',4);% 绘制插值曲线
hold on;grid minor;
plot(x,y,'o','MarkerFaceColor','b');% 绘制数据点
title('心形插值曲线图');xlabel('x');ylabel('y')
legend('插值点','插值曲线','Location','best')
```

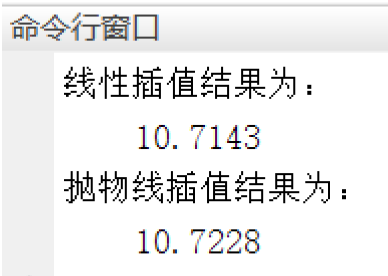
第二题

(1)Lagrange.m

```
function v = Lagrange(x,y,u)
% 实现拉格朗日插值，x,y 为插值点，u 为插值运算区间
v=0;
for k=1:length(x)
    s=1;
    for i=1:length(x)
        if i~=k
            s=s.*(u-x(i))/(x(k)-x(i));
        end
    end
    v=v+y(k)*s;
end
end
```

- 在课堂测试答案中给出了多种 lagrange 插值编写的示例

(2)




test2.m

```
clear;clc;%清除变量
x = [100,121,144]; y = [10,11,12];%设置插值点
u = 115;   %设置待求点
v_1 = Lagrange(x(1:2),y(1:2),u);%线性插值
v_2 = Lagrange(x,y,u);%抛物线插值
% 命令行输出结果
fprintf(['线性插值结果为：\n\t',num2str(v_1),'\n']);
fprintf(['抛物线插值结果为：\n\t',num2str(v_2),'\n']);
```

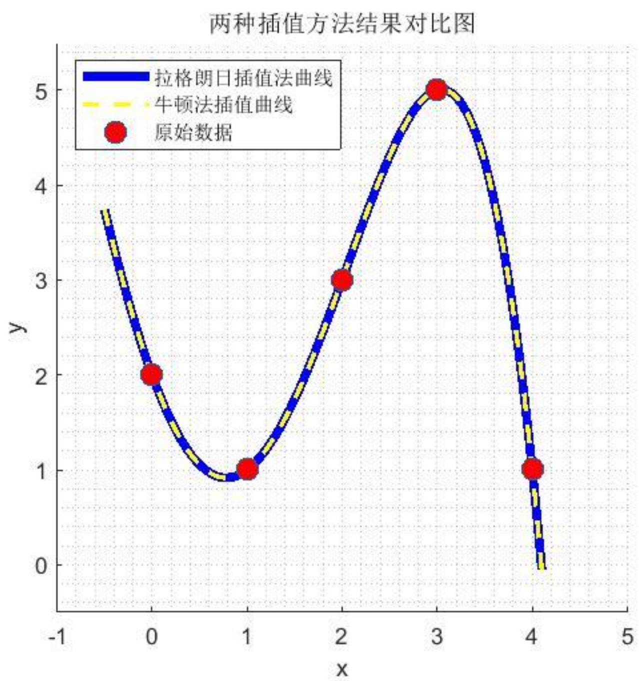
- 关于两个点的 Lagrange 插值就是线性插值

第三题

(1) Newton.m

```
function v = Newton(x,y,u)
% 实现牛顿法插值，x,y 为插值点，u 为插值区间
% 计算牛顿差商表
n=length(x);
D=zeros(n,n);
D(:,1)=y;%填充第一列
%逐行对差商表每个元素进行运算
for i=2:n
    for j=2:i
        D(i,j)=(D(i-1,j-1)-D(i,j-1))/(x(i-j+1)-x(i));
    end
end
% 计算牛顿插值多项式
v=D(1,1);%首项
for i=2:n
    % 运算多项式
     p=1;
     for j=1:i-1
         p=p.*(u-x(j));
     end
     v=v+D(i,i)*p;
end
end
```

(2) 插值结果对比图如下，



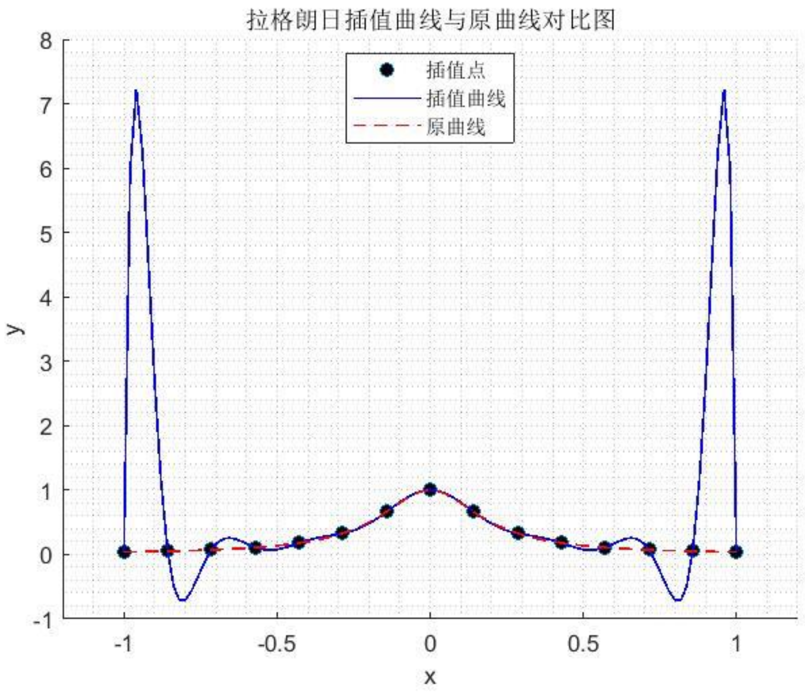
由图像可以看出，两种方法插值出的曲线完全相同，其区别在于求多项式系数的方法不同，Lagrange 法使用所有数据点求解系数，Newton 法则是逐点求解系数，应用了牛顿差商。相比较而言，Newton 法适应性更强，新增插值点时，可以有效利用以往的结果。

test3.m

```
clear;clc;%清除变量
load 3_x;load 3_y;%加载数据文件
u = linspace(-0.5,4.1,100);%设置插值区间
%牛顿插值
v_n = Newton(x,y,u);
%拉格朗日插值
v_l = Lagrange(x,y,u);
%绘图
hold on
plot(u,v_n,'b-','LineWidth',4)
plot(u,v_l,'y--','LineWidth',2);
plot(x,y,'o','MarkerFaceColor','red','MarkerSize',10);
%设置坐标属性
axis equal;grid minor;
axis([-1,5,-0.5,5.5])
xlabel('x');ylabel('y');
%设置标题与标签
title('两种插值方法结果对比图');
legend('拉格朗日插值法曲线','牛顿法插值曲线','原始数据',...
'Location','northwest');
```

第四题

解：(1) 图像如下，




test4_1.m

```
clear;clc;%清除变量
% 初始化插值点
x=linspace(-1,1,15);
y=1./(25.*x.^2+1);
%% 拉格朗日插值
u=linspace(-1,1,100);
v=Lagrange(x,y,u);
y0=1./(25.*u.^2+1);
hold on;grid minor;
plot(x,y,'o','MarkerFaceColor','k')
plot(u,v,'-b','LineWidth',1)
plot(u,y0,'--r','LineWidth',1)
%% 编辑图像
axis([-1.2,1.2,-1,8])
title('拉格朗日插值曲线与原曲线对比图')
legend('插值点','插值曲线','原曲线','Location','best')
xlabel('x');ylabel('y')
```

(2) 通过拉格朗日法得到的插值多项式 P(x)，误差为，

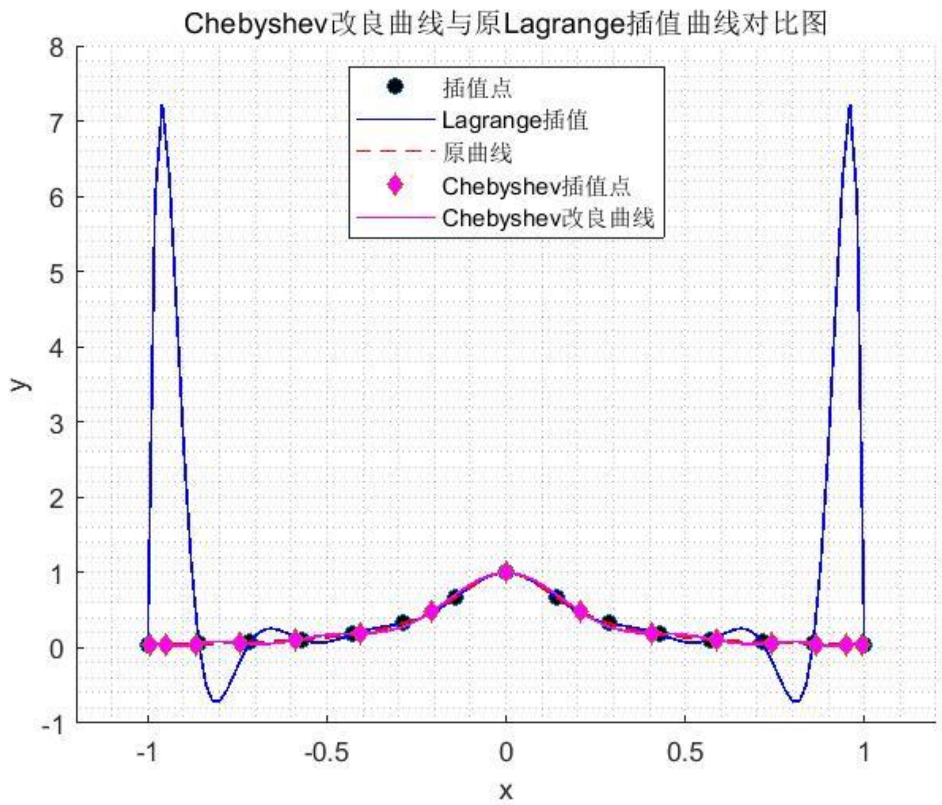
$$erro = \frac{(x - x_1)(x - x_2) \cdots (x - x_n)}{n!} f^{(n)}(c)$$

其中 c 为区间[min{x1,x2…xn} ,max{x1,x2…xn}]中的某点。由公式可见，每两个插值点之间必定存在误差的极大值或极小值，而且值的大小与$(x - x_1)(x - x_2) \cdots (x - x_n)$密

切相关，也就是说，插值点的选取会影响插值多项式的误差大小。

特点：当平均选取插值点时，由误差公式易得，会导致在边缘处出现过大的误差。

(3) 使用 Chebyshev 点之后，图像为，




test4_2.m

```
clear;clc;%清除变量
% 初始化插值点
x=linspace(-1,1,15);
y=1./(25.*x.^2+1);
%% 拉格朗日插值
u=linspace(-1,1,100);
v=Lagrange(x,y,u);
y0=1./(25.*u.^2+1);
hold on;grid minor;
plot(x,y,'o','MarkerFaceColor','k')
plot(u,v,'-b','LineWidth',1)
plot(u,y0,'--r','LineWidth',1)
%% 编辑图像（第二问将此部分注释）
axis([-1.2,1.2,-1,8])
title('拉格朗日插值曲线与原曲线对比图')
legend('插值点','插值曲线','原曲线','Location','best')
xlabel('x');ylabel('y')
%% 切比雪夫点优化
```

```
i=1:15;
x=cos((2*i-1)*pi/2/15);
y=1./(25.*x.^2+1);
u=linspace(x(1),x(end),100);
v=Lagrange(x,y,u);
%% 绘图（第一问将此部分注释）
hold on
plot(x,y,'d','MarkerFaceColor','m')
plot(u,v,'-m','LineWidth',1);
axis([-1.2,1.2,-1,8])
title('Chebyshev 改良曲线与原 Lagrange 插值曲线对比图')
legend('插值点','Lagrange 插值','原曲线',...
    'Chebyshev 插值点','Chebyshev 改良曲线','Location','north')
xlabel('x');ylabel('y')
```

(4) Chebyshev 定理给出了[-1,1]上的 Chebyshev 点，当区间变化到[a,b]时，点变化为

$$x = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{(2i-1)\pi}{2n}\right)$$

在该题中，a=1, b=5, n=15。从而 Chebyshev 点为：

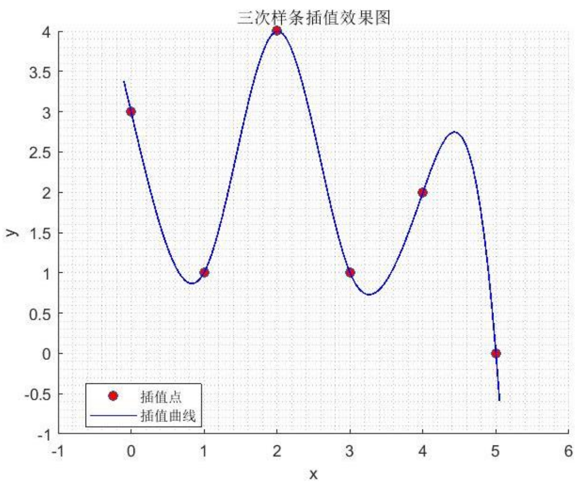
4.98904379073655
4.90211303259031
4.73205080756888
4.48628965095479
4.17557050458495
3.81347328615160
3.41582338163552
3.00000000000000
2.58417661836448
2.18652671384840
1.82442949541505
1.51371034904521
1.26794919243112
1.09788696740969
1.01095620926345

```
>> n=1:15
x=3+2*cos((2*n-1)*pi/(2*15))
reshape(x, [15, 1])
```

第五题

nature_spline.m

```
function v = nature_spline(x,y,u)
% 对若干数据点进行三次样条插值，x,y 为坐标数据,u 为插值区间
% 边界条件为 nature cubic spline
    n=length(x);%记录数据点个数
    dx=reshape(diff(x),[n-1,1]);%计算 δ 并重整为列向量
    dy=reshape(diff(y),[n-1,1]);%计算 Δ 并重整为列向量
    B=reshape(diff(dy(1:n-2)./dx(1:n-2)),[n-3,1]);
    B=3*[0;B;0];%生成矩阵方程右侧部分
    %生成求解 c 的参数矩阵
    D=zeros(n-1);
    D(1,1)=1;D(n-1,n-1)=1;
    for i=2:n-2
        D(i,i-1:i+1)=[dx(i-1),2*(dx(i-1)+dx(i)),dx(i)];
    end
    %求解 c
    c=D\B;
    %求解 b,d，范围为前 n-2 段的参数
    d=diff(c)./(3*dx(1:n-2));
    b=dy(1:n-2)./dx(1:n-2)-(2*c(1:n-2)+c(2:n-1))/3.*dx(1:n-2);
    %添加最后一段的 b,d
    b=[b;b(n-2)+2*c(n-2)*dx(n-2)+3*d(n-2)*dx(n-2)^2];
    d=[d;(dy(n-1)-b(n-1)*dx(n-1)-c(n-1)*dx(n-1)^2)/dx(n-1)^3];
    %生成插值曲线，若 u 小于 min(x)或大于 max(x)，则相应地按照最外层曲线做运算
    v=(y(1)+b(1)*(u-x(1))+c(1)*(u-x(1)).^2+d(1)*(u-x(1)).^3).*(u<=x(2))+...
    (y(n-1)+b(n-1)*(u-x(n-1))+c(n-1)*(u-x(n-1)).^2+d(n-1)*(u-x(n-1)).^3)...
    .*(u>x(n-1));
    for i=2:n-2
        v=v+(y(i)+b(i)*(u-x(i))+c(i)*(u-x(i)).^2+d(i)*(u-x(i)).^3)...
        .*(u>x(i)&u<=x(i+1));
    end
end
```



test5.m

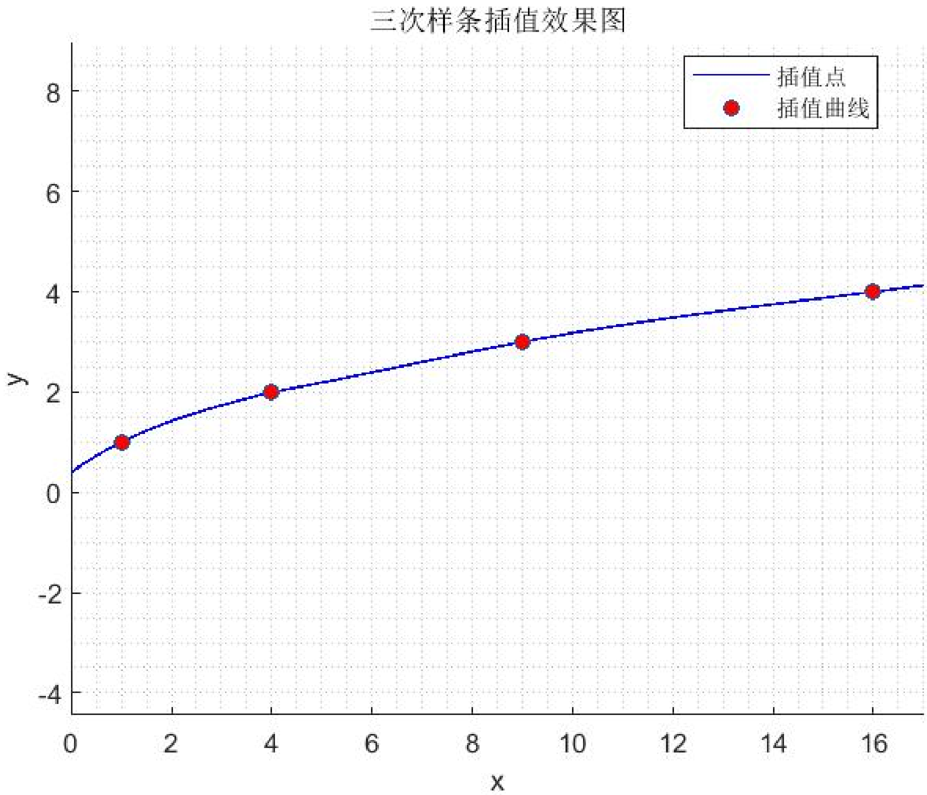
```
clear;clc;
%% 初始化插值点与插值区间
x=[0,1,2,3,4,5];
y=[3,1,4,1,2,0];
u=linspace(-0.1,5.05,100);
%% 进行插值运算
v = nature_spline(x,y,u);
%绘图
hold on;grid minor;
plot(x,y,'o','MarkerFaceColor','r')
plot(u,v,'-b','LineWidth',1)
title('三次样条插值效果图')
legend('插值点','插值曲线','Location','best')
xlabel('x');ylabel('y')
```

第六题

clamped 边界条件三次样条拟合函数：

clamped_spline.m

```
function v = clamped_spline(x,y,u,v0,vn)
%  对若干数据点进行三次样条插值，
%  边界条件为 campled cubic spline,
% x,y 为坐标数据,u 为插值区间，v0,vn 分别为起始与末尾斜率
    n=length(x);%记录数据点个数
    dx=reshape(diff(x),[n-1,1]);%计算 δ 并重整为列向量
    dy=reshape(diff(y),[n-1,1]);%计算 Δ 并重整为列向量
    B=reshape(diff(dy(1:n-2)./dx(1:n-2)),[n-3,1]);
    %生成矩阵方程右侧部分
    B=[3*(dy(1)/dx(1)-v0);B;3*dy(n-1)/dx(n-1)-2*dy(n-2)/dx(n-2)-vn];
    %生成求解 c 的参数矩阵
    D=zeros(n-1);
    D(1,1:2)=[2*dx(1),dx(1)];
    D(n-1,n-2:n-1)=[2/3*dx(n-2),dx(n-1)+4/3*dx(n-2)];
    for i=2:n-2
        D(i,i-1:i+1)=[dx(i-1),2*(dx(i-1)+dx(i)),dx(i)];
    end
    %求解 c
    c=D\B;
    %求解 b,d，范围为前 n-2 段的参数
    d=diff(c)./(3*dx(1:n-2));
    b=dy(1:n-2)./dx(1:n-2)-(2*c(1:n-2)+c(2:n-1))/3.*dx(1:n-2);
    %添加最后一段的 b,d
    b=[b;b(n-2)+2*c(n-2)*dx(n-2)+3*d(n-2)*dx(n-2)^2];
    d=[d;(dy(n-1)-b(n-1)*dx(n-1)-c(n-1)*dx(n-1)^2)/dx(n-1)^3];
    %生成插值曲线，若 u 小于 min(x)或大于 max(x)，则相应地按照最外层曲线做运算
    v=(y(1)+b(1)*(u-x(1))+c(1)*(u-x(1)).^2+d(1)*(u-x(1)).^3).*(u<=x(2))+...
    (y(n-1)+b(n-1)*(u-x(n-1))+c(n-1)*(u-x(n-1)).^2+d(n-1)*(u-x(n-1)).^3)...
    .*(u>x(n-1));
    for i=2:n-2
        v=v+(y(i)+b(i)*(u-x(i))+c(i)*(u-x(i)).^2+d(i)*(u-x(i)).^3)...
        .*(u>x(i)&u<=x(i+1));
    end
end
```



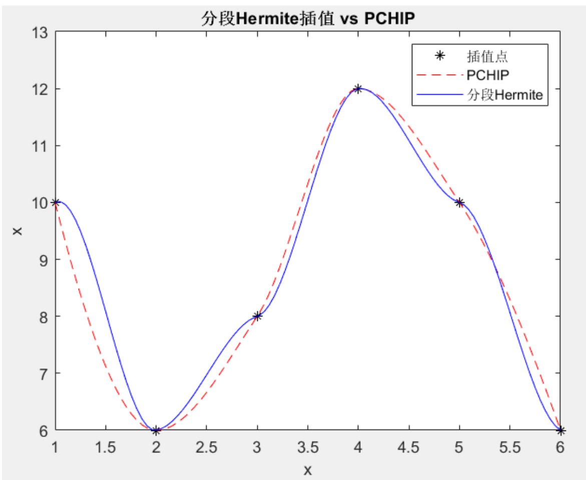

test6.m

```
clear;clc;
%% 初始化插值点与插值区间
x=[1,4,9,16];
y=[1,2,3,4];
u=linspace(0,17,100);
%% 进行插值运算,campled cubic spline
v = clamped_spline(x,y,u,1/2,1/8);
%绘图
hold on;grid minor;axis equal;
plot(u,v,'-b','LineWidth',1)
plot(x,y,'o','MarkerFaceColor','r')
title('三次样条插值效果图')
legend('插值点','插值曲线','Location','best')
xlabel('x');ylabel('y')
```

第七题

hw4_7.m

```
clc;clear;
x = [1,2,3,4,5,6];
y = [10,6,8,12,10,6];
dy = [1,0.3,0.5,0.2,-0.5,-1];
u = linspace(x(1),x(end),100);
v1 = pchip(x,y,u);
v2 = Hinterp(x,y,dy,u);
plot(x,y,'*k',u,v1,'--r',u,v2,'-b');
legend({'插值点' 'PCHIP' '分段 Hermite'})
title('分段 Hermite 插值  vs PCHIP')
xlabel('x'),ylabel('x');
function v = Hinterp(x,y,dy,u)
N = length(x);
for k = 1:length(u)
    for i = 1:N-1
        if x(i)<=u(k) && u(k)<=x(i+1)
            alpha1=(1+2*((u(k))-(x(i)))/((x(i+1))-(x(i)))).*...
                (((u(k))-(x(i+1)))/((x(i))-(x(i+1)))).^2;
            alpha2=(1+2*((u(k))-(x(i+1)))/((x(i))-(x(i+1)))).*...
                (((u(k))-(x(i)))/((x(i+1))-(x(i)))).^2;
            beta1=((u(k))-(x(i))).*(((u(k))-(x(i+1)))/((x(i))-(x(i+1)))).^2;
            beta2=((u(k))-(x(i+1))).*(((u(k))-(x(i)))/((x(i+1))-(x(i)))).^2;
            v(k)=alpha1*y(i)+alpha2*y(i+1)+beta1*dy(i)+beta2*dy(i+1);
        end
    end
end
end
```




区别：Herimite 可以指定点的导数；pchip 则用前向差分表示每点的导数，并若点前后的导数符号不一，则该点导数取零，若相邻点的步长不一，则使用前后两点前向差分的调和平均作为该点导数。

第八题

(1) 通过 fzero 与不动点法求解零点，结果为

```
x_fzero=
	0.56714
x_fix=
	0.56714
误差为：
	1.4229e-06
```

(2) 通过反插值法（Lagrange, spline, pchip 三种方法），得到结果为

```
x_Lagrange =

   0.567142623527871


x_spline =

   0.567142623527871


x_pchip =

   0.567142121741490
```

test8.m

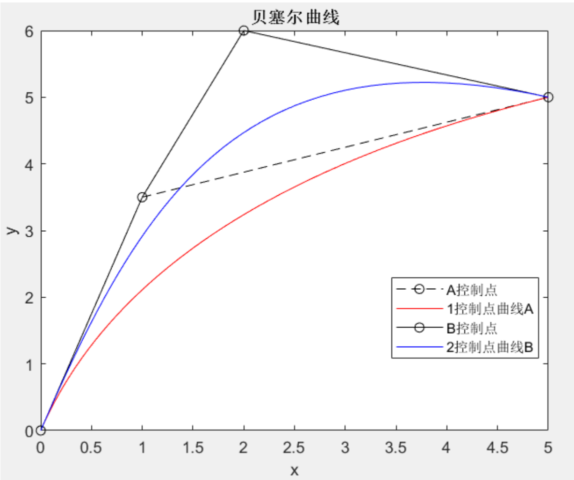
```
clear;clc;
format long
f=@(x)x-exp(-x);
x_fzero=fzero(f,0);
% 不动点求解
x0=0;
x_fix=1;
while abs(x_fix-x0)>0.5e-5
	x0=x_fix;
	x_fix=exp(-x0);
end
delta=abs(x_fix-x_fzero);
fprintf(['x_fzero=\n\t',num2str(x_fzero),'\n'])
fprintf(['x_fix=\n\t',num2str(x_fix),'\n'])
fprintf(['误差为：\n\t',num2str(delta),'\n'])
%% 反插值法求解
x=0.3:0.1:0.6;
y=[0.740818 0.670320 0.606531 0.548812];
y=x-y;
u=0;
x_Lagrange=Lagrange(y,x,u)
x_spline=spline(y,x,u)
x_pchip=pchip(y,x,u)
```

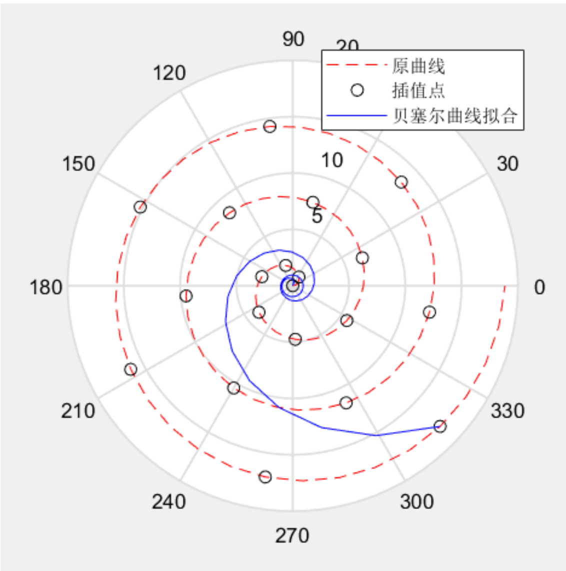
附加题

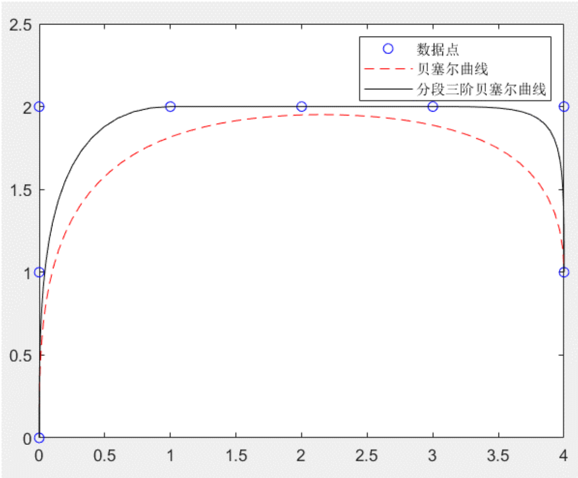
Hw4_9.m

```
clc;clear;close all
p1=[0,0]';
p2=[1,3.5]';
p3=[5,5]';
P=[p1 p2 p3];
v=Bezier(P,50);
figure
plot(P(1,[1:2,end-1:end]),P(2,[1:2,end-1:end]),'--ok',v(1,:),v(2,:),'b-')
hold on
p4=[2,6]';
P=[p1 p2 p4 p3];
v=Bezier(P,50);
plot(P(1,[1:2,end-1:end]),P(2,[1:2,end-1:end]),'-ok',v(1,:),v(2,:),'r-')
title('贝塞尔曲线')
legend({'A 控制点' '1 控制点曲线 A' 'B 控制点' '2 控制点曲线 B'})
xlabel('x'),ylabel('y')
% (2)
theta=linspace(0,2*pi*3,100);
r=theta;
figure
polar(theta,r,'--r')
x=r.*cos(theta);
y=r.*sin(theta);
x=x(1:5:end);
y=y(1:5:end);
hold on
plot(x,y,'ok')
v=Bezier([x;y],50);
hold on
plot(v(1,:),v(2,:),'-b')
legend({'原曲线' '插值点' '贝塞尔曲线拟合'})
% (3)
x=[0 0 0 1 2 3 4 4 ];
y=[0 1 2 2 2 2 2 1 ];
figure
plot(x,y,'bo')
hold on
P=[x;y];
vB=Bezier(P,50);
plot(vB(1,:),vB(2,:),'r--')

vB1=Bezier(P(:,1:4),20);
```

```
innP=2*P(:,4)-P(:,3);
vB2=Bezier([P(:,4) innP P(:,5:6)],20);
innP=2*P(:,6)-P(:,5);
vB3=Bezier([P(:,6) innP P(:,7:8)],20);
plot([vB1(1,:),vB2(1,:),vB3(1,:)],[vB1(2,:),vB2(2,:),vB3(2,:)],'k-');
legend({'数据点' '贝塞尔曲线' '分段三阶贝塞尔曲线'})

function [v]=Bezier(p,m)
n=size(p,2)-1;
u=linspace(0,1,m);
v=zeros(2,m);
c=zeros(1,n);
for i=1:m
    for j=0:n
        v(:,i)=v(:,i)+nchoosek(n,j)*p(:,j+1)*(1-u(i))^(n-j)*u(i)^j;
    end
end
end
```




数据点
贝塞尔曲线
分段三阶贝塞尔曲线
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
4