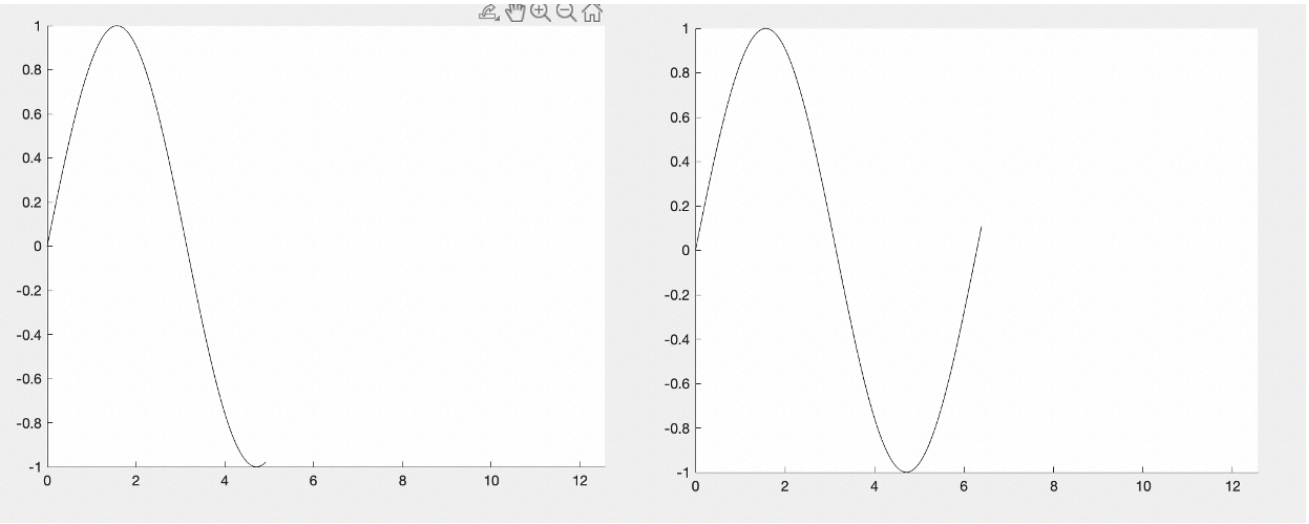
```
%% How we start all scripts

clc;
close all;
clear;

%% Let's make a fun shape!
%参数为数据点的形状：'.'为实心黑点；'*'为八线符；'<'为左三角；'d'为菱形；'o'为空心圆圈；'s'为方块符；'+'为十字符号；'^'为上三角符；'>'为右三角符；'h'为六角星；'p'为五角星；'x'为叉字符。
%点的大小可以用markersize设置,更改plot(x,y,'参数','markersize',N)里面N的大小即可。

point0 = [0, 0]';
point1 = [1, 0]';
point2 = [0.25, 0.25]';
point3 = [0, 1]';
point4 = [-0.25, 0.25]';
point5 = [-1, 0]';
point6 = [-0.25, -0.25]';
point7 = [0, -1]';
point8 = [0.25, -0.25]';
point9 = point1;



points = [point0 point1 point2 point3 ...
          point4 point5 point6 point7 ...
          point8 point9];

%% Let's plot!

plot(points(1,:), points(2,:), '-o')
axis([-10, 10, -10, 10], 'equal');

%% Let's also define an inline function to plot these points

plot_points = @(list_of_points) plot(list_of_points(1,:), list_of_points(2,:), '-o');
```

```
%% Transformers
% This is an arbitrary translation matrix
T = @(tx, ty) [eye(3,2) [tx ty 1]'];

% This is an arbitrary scaling matrix
S = @(sx, sy) [sx, 0,  0;
               0,  sy, 0;
               0,  0,  1];
% This is a rotation matrix (clockwise)
h_R = @(theta) [cos(theta)  sin(theta) 0;
                -sin(theta) cos(theta) 0;
                0           0          1];
```

```
% To rotate about this point we have to translate back to the origin,
% rotate, and then translate back to the current center

rotated_new_shape = T(shape_center(1), shape_center(2))*R(20*pi/180)*T(-
shape_center(1), -shape_center(2))*new_shape;
plot_points(rotated_new_shape);
```

```
%% Drawnow
% update figures and process callbacks
h = animatedline;
axis([0 4*pi -1 1])
x = linspace(0,4*pi,2000);

for k = 1:length(x)
    y = sin(x(k));
    addpoints(h,x(k),y);
    drawnow
end
```




```
%% What about for an arbitrary shape??
clf;
axis equal;
axis([-10, 10, -10, 10]);

my_points = [];

while true
    new_point = ginput(1);    % ginput allows us to input points by clicking
    if isempty(new_point)     % If no points are selected we break
        break
    end

    my_points(:,end+1) = [new_point 1]';  % Add the next point to the matrix

    clf;
    plot_points(my_points);
    axis equal;
    axis([-10, 10, -10, 10]);

end

my_points(:,end+1) = my_points(:,1);

%这一串其实能直接用gintput(n)代替，内置了循环和判断，缺点是要预判点击量上限，优点是可以偷懒
close all
clf
clear

p1=[];
```

```
[p1(1,:),p1(2,:)]= ginput(10);
```

```
%%make a circle
theta = linspace(0, 2*pi, 100);
[x, y] = pol2cart(theta, ones(size(theta)));
%Transform polar or cylindrical coordinates to Cartesian
circle = [x; y; ones(size(x))];
```

```
%eigenvalue
clc;clear;

% [V,D] = eig(A)
syms a
%A = [ 0.75 -0.6124 -0.25; 0.25 0.6124 -0.75; 0.6124 0.5 0.6124]
%B=[0.0975 0.9575 0.9706; 0.2785 0.9649 0.9572; 0.5469 0.1576 0.4854]
C = [ cos(a) -sin(a) 0; sin(a) cos(a) 0; 0 0 1]
[v,d] = eig(C)
```

```
%%hw1p5
% This is an arbitrary translation matrix
T = @(tx, ty) [eye(3,2) [tx ty 1]'];

% This is an arbitrary scaling matrix
S = @(sx, sy) [sx, 0,  0;
               0,  sy, 0;
               0,  0,  1];

%%
clf;
% make a circle
theta = linspace(0, 2*pi, 1000);
[x, y] = pol2cart(theta, ones(size(theta)));
circle = [x; y; ones(size(x))];

% squash it a bit
%squashed = S(1, 0.7)*circle;
%plot_points(squashed);
%axis equal;
```

```
%axis([-4, 4, -4, 4], 'equal');

%%
clc;clf;

v_scale = 1;              % vertical scale
h_scale = 2;            % horizontal offset

d = [1, 6, 11, 16, 21, 26];
array = [];

clf;
for i = 1:6
    sx = 1;
    sy = 1;
    dxtotal = 0;
    sd = 0;
    arrayy=[];
    for ii = 1:(i-1)
        sd = sd+d(ii);
    end
    dxtotal =  d(i)/2 - d(1)/2+sd;

    new_circle = T(((dxtotal+(i-1)*h_scale)), d(i)/2)*S(d(i)/2, d(i)/2)*circle;
    arrayy = horzcat(arrayy, new_circle);

    j=1;
    sumy = mean(new_circle(2,:));

    while(j< idivide(49, int16(d(i)+1),'floor'))
        new_circle1 = T(0, j*(v_scale+d(i)))*new_circle;
        arrayy = horzcat(arrayy, new_circle1);
        j = j+1;
        sumy = sumy +mean(new_circle1(2,:));
    end

    yavg = sumy/j;
    arrayy(2,:) = arrayy(2,:)  +25-yavg;
    array = horzcat(array, arrayy);
end

plot(array(1,:), array(2,:), '.')

rectangle('Position',[-5,0,100,50])

axis([-10, 120, -10, 60], 'equal');
```

```
%%integral
clc;
clear;
c = 3e8;
h = 6.626e-34;
k = 1.38e-23;
t = 5800;
vg = 1.618e14;
Qfun = @(v) 2*(v.^2)./(c.^2)./(exp((h.*v)./(k*t))-1);
pfun = @(v) 2*h*(v.^3)./(c.^2)./(exp((h.*v)./(k*t))-1);
q = integral(Qfun,vg,3.45e14);
q
p = integral(pfun, 0 ,10e20);
p
u = q*h*vg/p;
u
```

```
%%lagrange
close all;
syms x_l
data(:,1)=angle;
data(:,2)=height;
lagrange_poly = 0;
x_real = [0:0.01:2*pi];
for i=1:size(data,1)
    tmp_numerator = 1;
    tmp_denominator = 1;
    for j=1:size(data,1)
        if j ~= i
            tmp_numerator = tmp_numerator*(x_l - data(j,1));
            tmp_denominator = tmp_denominator*(data(i,1) - data(j,1));
        end
    end
    lagrange_poly = lagrange_poly + tmp_numerator/tmp_denominator * data(i,2);
end
simplify(lagrange_poly);
hold on;

axis([0, 2*pi, 0, 25])
plot(data(:,1),data(:,2),"r o")
```

```
plot(x_real, double(subs(lagrange_poly, x_l, x_real)), 'k')
xlabel('Angle')
ylabel('Follower height')
title('Lagrange fits', 'Fontweight','b')
```

```
%c
% Spline & pchip
% clf;

hold on;

sp = spline(angle, height, x_real);
plot(x_real, sp, 'black');

pc = pchip(angle, height, x_real);
plot(x_real, pc, 'g');

axis([0,2*pi,0,30 ])

title('Spline fits')
xlabel('Angle')
ylabel('Follower height')
```

```
%%d
%picth curve & cam surface
%(xpc,ypc) & (xps,yps)
% pchip
% clf;
close all
clear;
clf;

time = linspace(0,20,21);
angle = 2*pi*time/20;
height = [0, 2.4, 8.6, 16.4, 22.6, 25, 25, 24.9, 23.9, 21.8, 18.9,16.5, 15.2, 15,
15,12.9, 7.5, 2, 0.1, 0, 0];

axis([0,2*pi,0,25 ])
theta = [0:0.1:2*pi];
```

```
hold on;

rf = 2;
rp = 10;
pc = pchip(angle, height, theta);

rpc = rp+pc;
xpc = rpc.*cos(theta);
ypc = rpc.*sin(theta);
time = 1;
dxpc = gradient(xpc)/time;
dypc = gradient(ypc)/time;
speed = sqrt(dxpc.^2+dypc.^2);
xcs = xpc+rf.*(-dypc./speed);
ycs = ypc+rf.*(dxpc./speed);
plot(xpc, ypc, 'g--');
plot(xcs, ycs, 'black');

axis([-50,50,-50,50 ])
daspect([1,1,1])
```

```
%%wirte csv file
%e
%rotation
% This is an arbitrary translation matrix
T = @(tx, ty) [eye(3,2) [tx ty 1]'];

% This is an arbitrary scaling matrix
S = @(sx, sy) [sx, 0,  0;
               0,  sy, 0;
               0,  0,  1];
% This is an arbitrary rotation matric
R = @(ro) [cos(ro),sin(ro);
           -sin(ro),cos(ro)];
% unit circle
    thetar = linspace(0, 2*pi, 1000);
    [x, y] = pol2cart(thetar, ones(size(thetar)));
    circle = [x; y; ones(size(x))];
%Rotation
for i = 0:4
    angler = i/5*2*pi;
    pcr = R(angler)*[xpc;ypc];
    xcsr = R(angler)*[xcs;ycs];
```

```
    subplot(1,5,i+1);
    plot(pcr(1,:),pcr(2,:),'g--');
    hold on;
    plot(xcsr(1,:),xcsr(2,:),'black');
    title(['Theta=' num2str(i/5*2*pi) 'pi']);

    new_height = pchip(angle, height, angler);
    new_circle = T(new_height+rp,0)*S(2,2)*circle;
    plot(new_circle(1,:),new_circle(2,:),'r')

    axis([-50,50,-50,50 ])
    daspect([1,1,1])
end
xcsr(3,:)=0;

writematrix(xcsr','xcsr.csv')
```

```
%%Fibonacci
%Tansformation
clear;
clf;
T = @(tx, ty) [eye(3,2) [tx ty 1]'];

% This is an arbitrary scaling matrix
S = @(sx, sy) [sx, 0,  0;
               0,  sy, 0;
               0,  0,  1];

% This is an arbitrary rotation matric
R = @(ro) [cos(ro), sin(ro),0;
           -sin(ro), cos(ro),0;
           0,      0 ,1];
%%
%Fibonacci siries
f = [];
f(1) = 1;
f(2) = 1;
for i = 3:16
    f(i) = f(i-1)+f(i-2);
end

fiboarc = [];
```

```
%%
%Unit arc
thetar = linspace(0, 1/2*pi, 1000);
[x, y] = pol2cart(thetar, ones(size(thetar)));
unitarc = [x; y; ones(size(x))];
plot(unitarc(1,:),unitarc(2,:),'red');

%%
%create unit square
side = linspace(0, 1, 1000);
x1=0;
y1=side;
x2=side;
y2=0;
x3=1;
y3=side;
x4 =side;
y4 =1;
line1 = [0.*ones(size(y1));
         y1;
         ones(size(y1));];
line2 = [x2;
         0.*ones(size(x2));
         ones(size(x2));];
line3 = [1.*ones(size(y3));
         y3;
         ones(size(y3));];
line4 = [x4;
         1.*ones(size(x4));
         ones(size(x4));];
unitsquare = [line1, line2, line3, line4];
plot(unitsquare(1,:),unitsquare(2,:))
    axis([-10,10,-10,10])

%%
% unit arc and unit square
clf;
plot(unitarc(1,:),unitarc(2,:),'red');
hold on
plot(unitsquare(1,:),unitsquare(2,:),'blue')
axis equal

%orgin arc and orgin square
originarc = R(pi)*unitarc;
originsquare = R(pi)*unitsquare;
clf;
```

```
plot(originarc(1,:),originarc(2,:),'red');
hold on
plot(originsquare(1,:),originsquare(2,:),'blue')
axis equal

%%
% origin xy position
x =[];
y = [];
x(1) =0;
y(1) =0;
for i = 2:16
    if mod(i,2)==1 % i is odd
        x(i) = x(i-1)+((-1)^floor((i-1)/2))*(f(i)-f(i-1));
        y(i) = y(i-1);
    else % i is even
        x(i) = x(i-1);
        y(i) = y(i-1)+((-1)^(floor((i-1)/2)))*(f(i)-f(i-1));
    end
end

%%
%plot fibonacci arc and fibonacci square
clc;


fiboarc = [];
fibosquare =[];
for j = 1:16
    newarc=T(x(j),y(j))*R(-(j-1)/2*pi)*S(f(j),f(j))*originarc;
    fiboarc= horzcat(fiboarc, newarc);
    newsquare =  T(x(j),y(j))*R(-(j-1)/2*pi)*S(f(j),f(j))*originsquare;
    fibosquare = horzcat(fibosquare, newsquare);
end

plot(fibosquare(1,:),fibosquare(2,:),'b')
plot(fiboarc(1,:),fiboarc(2,:), 'r')
axis equal
```