UCSD_MAE_292_Computer_Aided_Design&Analysis_1

这一个系列算是对2020Spring这个学期的MAE292课程的总结了，梳理了我的一些思路、想法和MATLAB代码块（可能是因为我懒或者是我写的不多，都是有了思路之后再去找找用MATLAB的语言表述的代码，对我来说无论是python还是MATLAB都只是像积木块，抄抄改改、拼拼凑凑，总是能搭好的）。从对MATLAB一无所知甚至有点点嫌弃（开学的时候跑xugroup的超声成像模拟频频死机🤣）到MATLAB真香。虽然感觉没学到什么，但是至少是比材料学更加实用一些吧。

本课的内容主要可以分为基本图形变换、数值拟合、凸轮装置（cam）、连杆装置（linkage）以及有限元模拟优化（简单介绍）。

1. 基本图形变换

基本图形变化涉及到了三种图形变化：缩放，旋转，平移。实时这三种变化都可以归纳为坐标轴的变换。这个超nice的理解方式是在3blue1brown的线性代数的本质中学到的。（安利！！！3blue1brown的视频可以在YouTube上订阅，也可以在bilibili中三连。后者提供了中文字幕，而且做了内容分类，比较友好。不得不说，因为3blue1brown、半佛老师、巫师财经我才开始用bilibili的。在此以前只用它偶尔看过鸭子侦探和工作细胞😂）。

首先我们用矩阵表示向量(x, y)，可以写为 $\begin{bmatrix} x \\ y \end{bmatrix}$，那么初始向量$V = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$。同理表示二维直角坐标系的单位向量$e_i = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ 和$e_j = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$。将这些向量矩阵并列，就可以得到直角坐标系的基向量，$e = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (Fig. 1)。

接着我们将直角坐标系进行旋转伸缩变换得到新的坐标系(Fig. 2) 。原x轴单位向量(1, 0)伸缩变换为(1, -2), 原y轴单位向量 (0,1)伸缩变换为(3, 0) ，即它的新单位向量为$e_i' = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$ 和$e_j' = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$，以及新基向量 $e' = \begin{bmatrix} 1 & -2 \\ 3 & 0 \end{bmatrix}$。此时, 在新坐标系里用新基向量$e_1'$与$e_2'$ 表示(-1, 2)的新向量（Fig. 2中黄线向量指向的蓝色网格部分) ，就相当于旧坐标中（Fig. 2中黄线向量指向的灰色网格部分）表示(5, 2)的向量。写成矩阵为：

$$\begin{bmatrix} 1 & -2 \\ 3 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

所以图形的旋转伸缩变换就相当于矩阵基向量网格的压缩，将新的基向量矩阵乘以旧向量或者位置，就能得到图形被伸缩变换之后的向量或者位置。

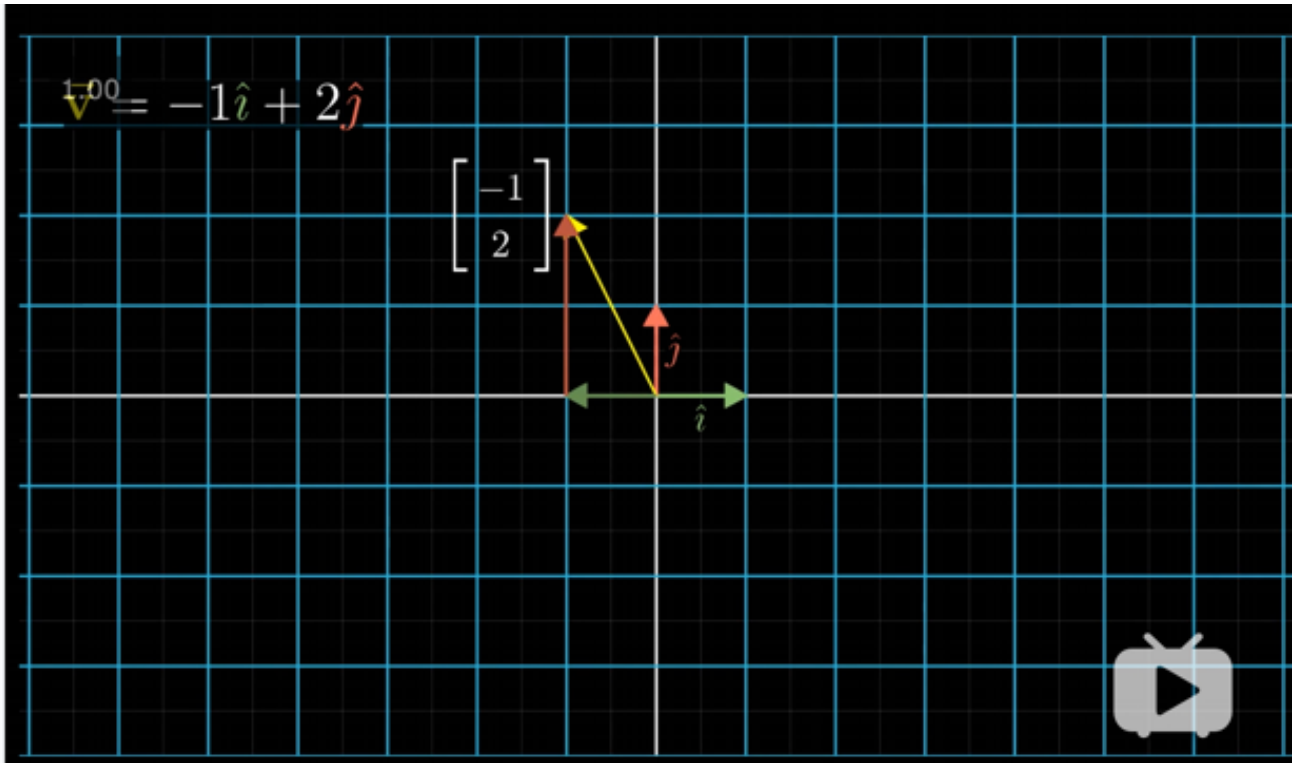


Fig. 1：黄线箭头为初始向量；向右的绿色箭头与向上的红色箭头分别为基向量i、j；蓝色网格为二维正交直角坐标系

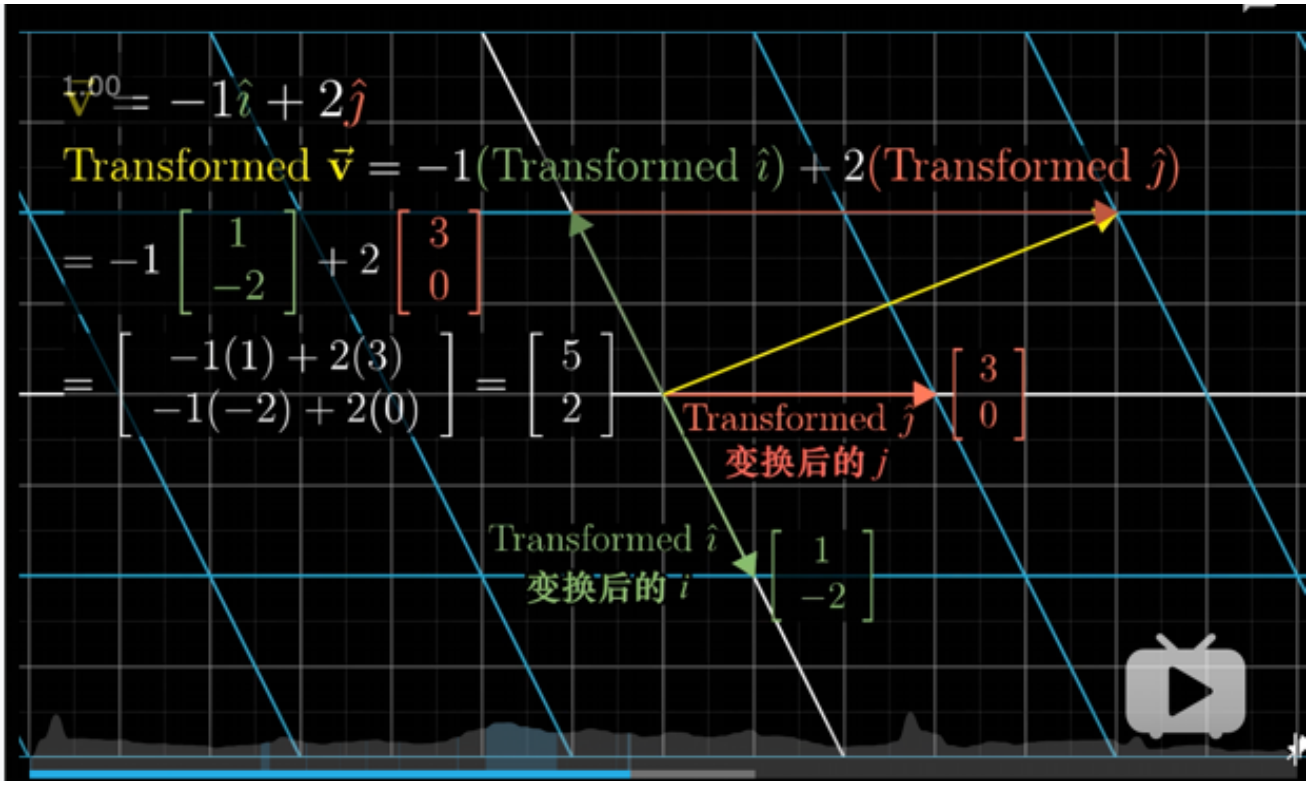


Fig. 2：黄线箭头为变化后的向量；向下的绿色箭头与向右的红色箭头分别为基向量i、j；蓝色网格为新坐标系，灰色网格为原正交直角坐标系

2. Transformers

将上述的图形变换拆分之后，我们其实可以得到分别控制伸缩（scaling）、旋转（rotation）、平移（旋转）的矩阵。

伸缩向量S第一列相当于将x轴的单位向量放大或者缩小了sx倍（1，0）。

旋转向量R第一列相当于x轴单位向量向上旋转theta角度之后分解在x，y方向的值。如果不熟悉极坐标的表示方法，其实就相当于旋转之后的单位向量在x轴分量=1·cos(theta)，在y轴分量=1·sin(theta) 。

平移变化变换的是坐标原点，所以表示方法略有不同，因此在第三列加入tx，ty表示坐标原点的偏移量，也是图形的平移量。举个例子就是火车没有开动时你在车尾。当你往车头走的时候，火车也开动了。而当你走到车头的时候你的位置相当于你走过的车长（x）+火车开动的距离（tx）。

也正是因为平移变化时候矩阵扩充了一列，所以所有的矩阵都变成了3X3矩阵，(x, y)向量变为(x,y,1)。不然会造成矩阵相乘时维度不一致而无法相乘。

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

% This is a rotation matrix (anti-clockwise)
R = @(theta) [cos(theta)  -sin(theta) 0;
              sin(theta) cos(theta) 0;
              0           0          1];

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

%eye(3,2)相当于生成一个三行两列的单位矩阵，只是因为行数和列数不相等，导致其前两行是2*2的单位向量，后两行为0。等效于
%T= @(theta) [1 0 tx;
%             0 1 ty;
%             0 0 1];
```

这个理解方式在 Prof. Cai 的“MATS 231 MAE 276 - Mechanics of Soft Materialsi”课上的材料形变上还是蛮有用的。(X, Y, Z)为未变形材料中的某向量，变形之后成为(x, y, z)，描述材料变形过程的 F（Deformation gradient tensor）就相当于这个基坐标变换。这门课的知识点很多，和Prof. Nic 的CAD以及Prof. Luo老师的Thermodynamics是我这三个学期学的最充实的三门课，也许之后我会也出几篇总结吧。

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [F] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

3. 案例

这是HW1的最后一题，我觉得很有代表性，很好的把这几个知识点串起来。第一次看到这道题的时候，这筛子用AutoCAD什么绘图的软件来做的话简直不要太容易。但是他的要求是用MATLAB绘制。当时我们几个都不会MATLAB，整个人都是懵的，简直有毒。说这道题很有代表性是当时突然想到的做题思路以及他确实能很好的用上所有的知识点，还是很有必要记录和分享一下吧。

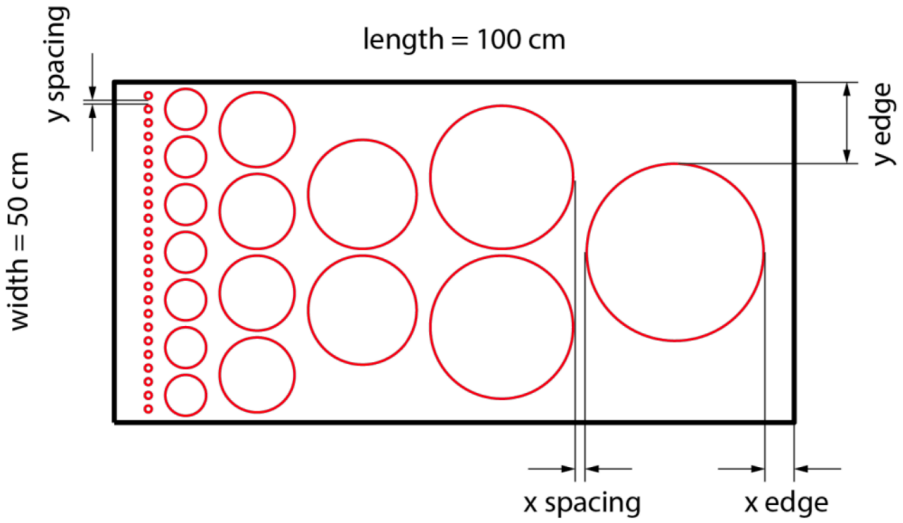
Problem 5: Computational Design of a Particle-Size Sorting Sieve (20 points)

We want to design a particle size sorting sieve on an acrylic board with a series of holes of diameters $[1, 6, 11, 16, 21, 26]$ cm, as shown in the figure below. The size of the acrylic board is 50×100 cm. The holes in each column are the same size and the diameters increase from left to right columns. We want to fill in as many holes as possible while also meet some design requirements listed below.

x edge	y edge	x spacing	y spacing
$\geq$ 1cm	$\geq$ 1cm	= 2 cm	= 1 cm

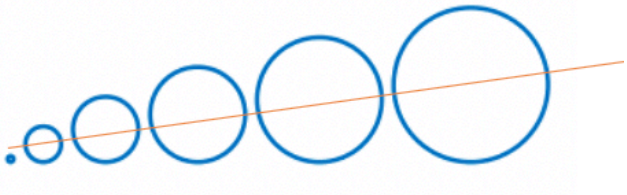
Use MATALB to draw the design pattern of the particle size sorting sieve. It is important that your design be parameterized through the use of one or more loops. We do not want you to just hand plot each circle in a script. Your final figure should look similar to the one present below.

Please submit a picture of the plotted geometry and your code.

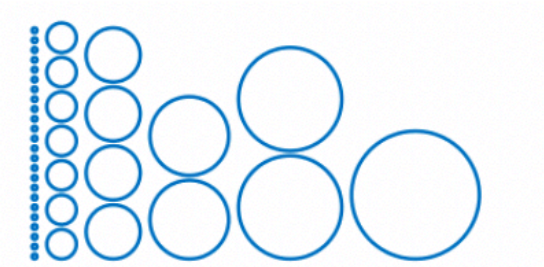


粗看这道题的确很复杂。但当仔细想想自己学过什么的时候，无非就是旋转，伸缩，以及平移这三个操作。那么将这道题往这三个变换操作靠可以发现。这道题其实是1. 先做出一个小圆；2. 然后进行平移操作调整x方向间距，画出一排六列的六个圆形；3.接着再调整这六个圆的直径大小，画出一排六个不同直径的圆形；4.然后再将每列的圆形向上复制扩充并调整y方向间距；5.将每排圆形对着x方向中心线上下对称；6. 最后画出矩形外框。主要框架就是这六步，剩下的就是各种填充实现，比如怎么画圆（别笑，当时真的连圆都不会画）；23步用等差数列去表示x方向间距等等等等。但整个方向是对的，有条不紊推进就好了。用图表示就是:

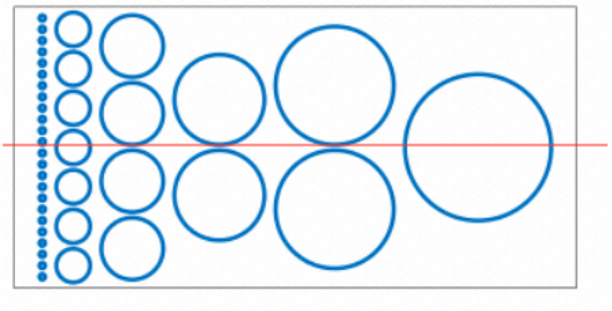
1



23

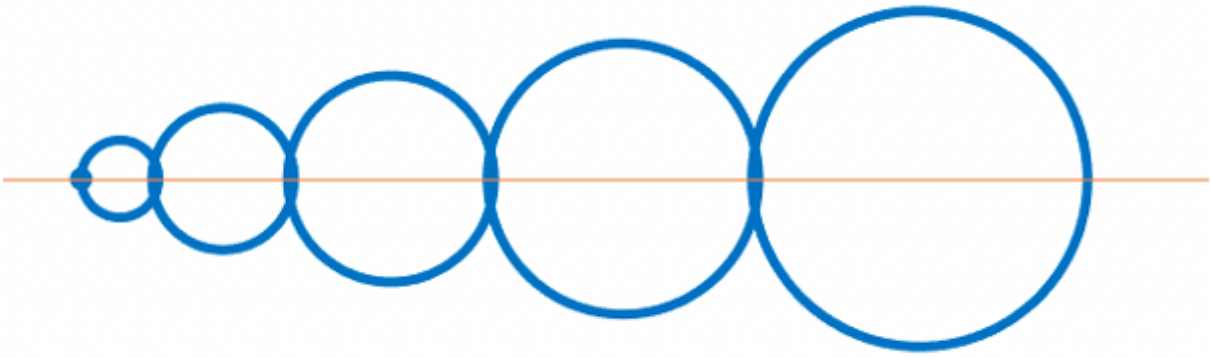


4



56

其实这道题当时做的时候也是有瑕疵的，因为在第1步第2步的时候并没有实现平移，圆心逐渐上移。根本原因是在生成用pol2cart函数生成初始圆的时候，圆心不在原点。导致后面矩阵变换的时候其实圆心的横纵坐标也被遗弃放大了逐渐。这个问题就导致最后的中心对称会比较难调的，不过类似于第2步x间距的等差数列。这23本应该的结果是（我没有重新调整x间距，只做了一个示意图）：



具体实现过程的代码和注释我附在最后面了。 写这道题就像小时候写奥数题，思考的时候很享受，也很痛苦，做出来的时候是真的很有成就感。

4 港话

这道题对我的还有些触动。

一是我真的挺懒的，得失心可能太重了。因为我们当时是临时选课，选课时已经过了别人两个星期的ddl，特意向老师申请了三天的宽限时间，其实也熬夜连轴转满满的。这个样子白手起家做到这个时候我怕其实挺满足了，感觉大家可能都做不出来吧。毕竟睿哥已经提前放弃，直接手动计算了所有的xy距离，用数列一个个标上去的。有这个参照在，我确实觉得最后这一点不算什么，没有必要再花那么大的力气。而当时立哥在参考完我的代码之后还是自己琢磨一遍。当时确实也是被他刺激到了吧，自己又去想了想，其实也就多花了半个多小时。说起来这个复盘的习惯也是跟他学的。讲道理我能混到现在真是个奇迹。

其实我一直没有自己的内驱力。预计工作量会巨大的情况面前，觉得边际效益太低的我，真的就是觉得跟大部分人差不多就好了吧。所以一直都是别人做到什么样子，我就做到什么样子，关键时候比别人稍稍前一点就好，平时不会去逼自己走的更远。万年老二就是俺。

与ChenC姐姐之后也聊起过这事。当时是想问她如何对整个房屋内空气流动进行热学模拟，她当时跟我说：你为了一个小project，要去做这个有限元模拟不值得。她这个想法就跟我差不多。

跟睿哥聊的时候，他说他也会这样。但是如果不继续学习做这件事，反正也都是打游戏睡觉。虽然打游戏睡觉也很香。

但其实很多时候，也确实自己多做一些，努力一些，会让别人觉得靠谱安心吧。特别在团队里的时候，如果真的没有其他什么更重要的事情，耐着性子逼自己多做一会儿也好，照顾别人的情绪。现在也确实比以前做的好多了。也算是我这一年最大的收获之一了。

二是这次拆解问题，列好提纲的做事方法。其实仔细想想，就是我写了这么多年的实验预习报告啊。以前确实也只是把他当成任务完成了，却没有仔细想想他的意义在哪。想起来带我物化实验的许新华老师，他的要求就是不准照抄实验教材，做实验的时候要求做到只有自己的笔记和提纲就能独立完成实验。确实在他的高要求下，做物化实验极度自信，学到的多，也了解多。

另外在想起之前建个人主页也是采取了这种方法。当时是想着按照做实验的部分。先是查了很多资料找了一下搭建个人主页大致需要的整个流程。然后也再一个个按着流程去走。有问题就边做边学，大不了推到重来，至少大方向还在。（回头也会总结一下发出来的，虽然建完了就没用过，但是我怕再拖着我连回忆都没了呢）。

当时开学在实验室呆了一个月就退出去了。确实是无法平衡学业。但根本原因可能是自己觉得很盲目吧，盲目感令人窒息。这个交叉学科的实验室，其实看来最重要的部分在算法，以及对医学成像的了解，绝不是我短时间内可以上手的。更重要的是，我不知道整个流程究竟是什么，我做的每一步意义在哪。也不知道预计会花多少时间；加上白天不干活，晚上疯狂熬夜这个确实跟我的习惯不一样，短期突击还可以，长期不是一个长久之计，我真的好佩服那些说可以直接从项目入手，边做边学的同学。也许我没有能够整体了解整个实验项目吧，每天像个螺丝钉一样（卑微）。

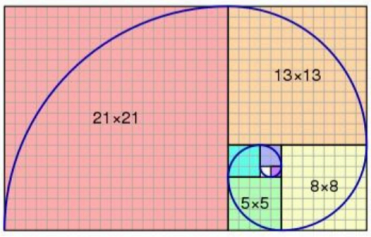
5. Bonus

还有一道比较类似的题目也挺有趣的。代码我也附在后面了。

Problem 1: CAD transformations to generate fractals (20 points)

Many structures in nature are self-similar, formed from repeating patterns of scaling, translation, and rotation operations. In this problem you will use Matlab to construct two fractals. Turn in all plots in your final pdf and all Matlab code.

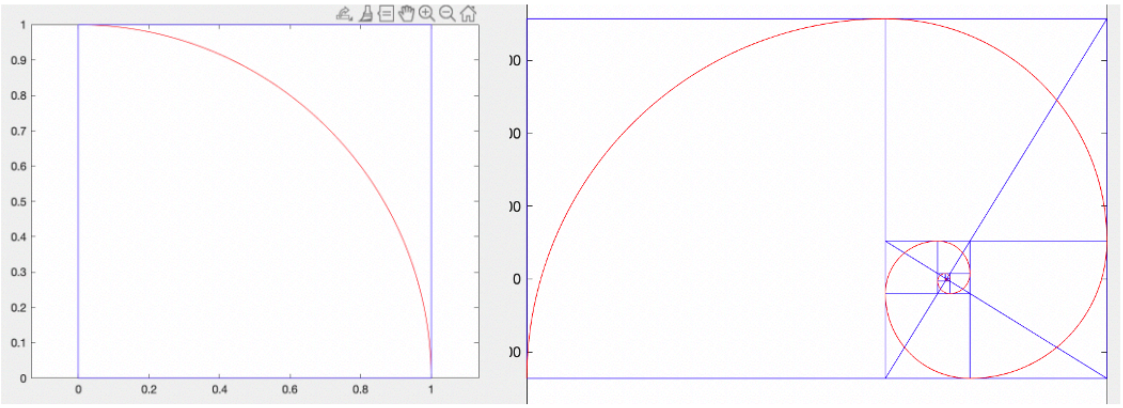
1. The golden spiral shown below can be constructed from a series of iterative CAD transformations of rotation, scaling, and translation using the Fibonacci sequence to generate the scaling amount (sequence definition below). To generate this shape in Matlab, start with one 1 × 1 square (side length equal to the first Fibonacci number) and add an arc (quarter of circle). Represent the arc as a set of points. To expand the spiral we scale the box and arc by the next Fibonacci number, and rotate and translate the shape so that the arcs meet. To continue building the spiral we keep adding square and arc shapes with side lengths given by the fibonacci series. **Generate a program to draw the golden spiral up to the 16th Fibonacci number using CAD transformations.** For reference the spiral below is drawn to the 8th Fibonacci number (21). The bounding boxes should be blue and the arcs red in your final plot.



$$F(1) = 1, F(2) = 1$$
$$F(n) = F(n-2) + F(n-1) \text{ for } n > 2$$
which gives the numbers:
$1, 1, 2, 3, 5, 8, 13, 21...$

大致思路：这个图形其实是一个正方形以及其中最大的1/4圆弧组成的单元。然后旋转，只不过弧长和正方形的边长是按照斐波那契数列变换的。以及初始位置也是依赖于前一个值。1. 生成斐波那契数列；2.生成第一个单位弧和单位方框作为旋转缩放的参考对象；3.确定每个方框的起点位置作为平移变换的依据（这一步我记得是一个比较复杂的数列，每两个一组，xy还错开了一步，下图所示）；4.写出循环赋予半径画图就好了。

```
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
```



6. 代码

```
%%hw1p5筛子
% 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 这一步是因为样例里有个椭圆阵列
%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 %画出第一排的圆形
```

```
    sx = 1;
    sy = 1;
    dxtotal = 0;
    sd = 0;
    arrayy=[];
    for ii = 1:(i-1)
    %每一列的间距仔细分析其实是个有规律的数列
        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;
    %向右平移以及确定直径
    arrayy = horzcat(arrayy, new_circle);

    j=1;
    sumy = mean(new_circle(2,:));
    %确定圆心的y值位置
    while(j< idivide(49, int16(d(i)+1),'floor'))
    %画每一列的圆形，idivide是为了根据y方向宽度自动判断能够容纳的圆圈个数
        new_circle1 = T(0, j*(v_scale+d(i)))*new_circle;
        %向y方向拓展
        arrayy = horzcat(arrayy, new_circle1);
        j = j+1;
        sumy = sumy +mean(new_circle1(2,:));
    end

    yavg = sumy/j;
    %确定一列的中心对称线
    arrayy(2,:) = arrayy(2,:)  +25-yavg;
    %调整y轴高度，使得每一列圆圈均匀分布在对称线两边
    array = horzcat(array, arrayy);
end

plot(array(1,:), array(2,:), '.')
rectangle('Position',[-5,0,100,50])
axis([-10, 120, -10, 60], 'equal');
```

```
%%
%midtermproblem1
%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
```