Name: Andrew Brown	**Lab Time:** T 12:00
People Worked With: Cailin Moore	**Websites Used:**
Time spent on zyBooks (hrs):1	**Time spent on lab (hrs):4**
Submission Instructions Turn all work in to Lab 9 on Gradescope (PDF) and Canvas (.zip file), even if it is not complete yet. If you are not finished, complete the assignment outside of lab and re-submit to Lab 9 on Gradescope and Canvas. All labs are typically due at the same time on Monday every week, but check Canvas if in doubt.	

Learning objectives:

- *Practice building matrices and using matrix multiplication to move objects around*
  - *Translation*
  - *Rotation*
- *Plot a multi-variable 3D function*

New MATLAB commands

These are highlighted in **bold** in the instructions below.

- `plot3( xs, ys, zs)` – Plot in 3D
- `eye(d)` – make an identity matrix with dimension d
- `axis equal`
- `[S,T] = meshgrid(s,t) –` Turn the arrays s and t into 2D matrices
- `surf( S,T,Z ) –` plot the surface. S,T,Z must be 2D matrices that are all the same size
- `camlight left;` Place a light over your shoulder
- `lighting phong`; Nice lighting
- `view(az, el);` Set the view azimuth and elevation

Lab Problems

Files to download to your Lab 9 Folder
DrawHouse

Getting Started

Anything for experimentation goes here with bullet points

- Call DrawHouse(eye(3)) from the command line to make sure it works
- Call axis equatl to make it square
- Put a break point in DrawHouse
    - What dimension are the house, window, and door matrices?
    - What is the last row set to? Can you explain why?

<u>Problem 1</u>

Use a matrix to move a house around. Translate by (5, 1.5) and rotate by 25 degrees then try both combinations of rotating then translating

Deliverables:

1. Script to make the four plots
2. Make a translation matrix and check that it correctly translates
3. Make a rotation matrix and check that it correctly rotates
    a. Also verify that it is orthogonal
4. Plot with 4 versions of the house: Translated, rotated, translated then rotated, rotated then translated
    a. You must use your matrices to do this

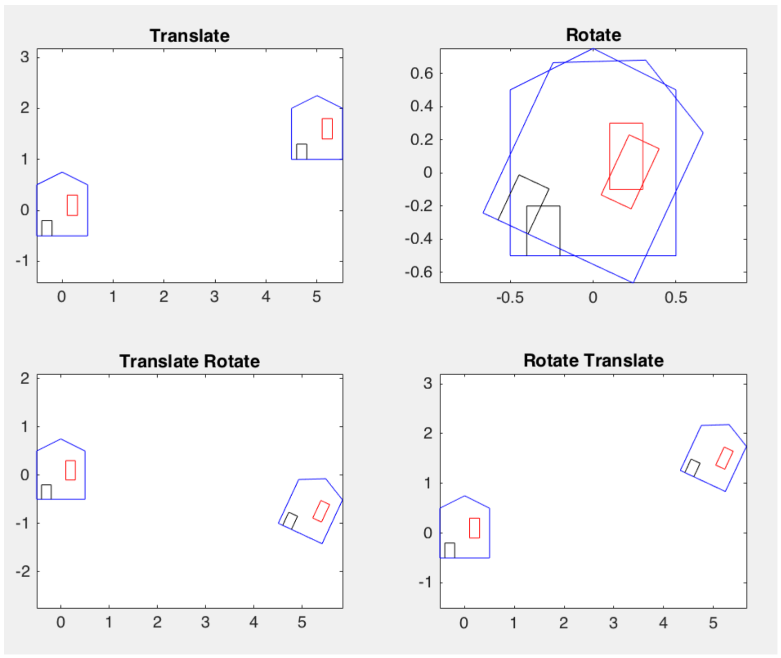
Step by Step Instructions:

- First make four plots and plot the house at the origin in each of them
    - Call **DrawHouse** with the identity matrix
    - Use axis equal to make it the right aspect ratio (not squished)
    - Put a title on the subplot but you can skip labels
- Make a translation matrix
    - Use eye to make an identity matrix mTrans. Must be a 3x3 matrix
    - Set dx to be 5, dy to be 1.5
        - The upper right two elements of mTrans
        - (refer to lecture notes)
    - To check that you have the correct matrix, try
        - mTrans * [0;0;1]
    - You should get the column vector 5, 1.5, 1
        - See self check below
    - Use DrawHouse with mTrans to make the first picture
- Make a rotation matrix
    - Use eye (again) to make an identity matrix mRot. Must also be a 3x3 matrix
    - Declare a variable for theta
    - Fill in the matrix; refer to lecture notes.
        - cos, sin
        - -sin, cos
    - Check that you have the correct matrix
        - dot(mRot(1,:), mRot(2,:)) is zero
            - as is every other dot product if the rows are different
        - dot(mRot(1,:), mRot(1,:)) is one
            - as is every other dot product if the rows are the same
        - mRot * mRot' is the identity matrix (or close enough)

                - Math fact: This is the matrix multiplied by its transpose. One property of rotation matrices is that their transpose is their inverse – i.e., rotating in the opposite direction. You'll notice that mRot' is the same as mRot, except the minus sign on the `sin` is swapped…
        - Note: you only need to check the conditions above for this lab's hand-in, but in general, if you're making a rotation matrix you should ALWAYS check all of these.
        - Use DrawHouse with mRot to make the second picture
- Now make the 3rd and 4th pictures
    - Note: The matrix that is on the *right* is the one that happens first. So
        - mTrans * mRot
    - rotates then translates (yes, that seems backwards for those of us who read left to right)
    - Math fact II: This is a visual demonstration of the fact that matrix multiplication is not communitive – A * B is **not** the same as B * A.

Self-check:

```
>> mTrans * [0;0;1]

ans =

    5.0000
    1.5000
    1.0000

>> dot( mRot(:,1), mRot(:,2) )

ans =

     0
>> dot( mRot(:,2), mRot(:,2) )

ans =

     1
>> mRot * mRot'

ans =

    1.0000   -0.0000         0
   -0.0000    1.0000         0
         0         0    1.0000
```



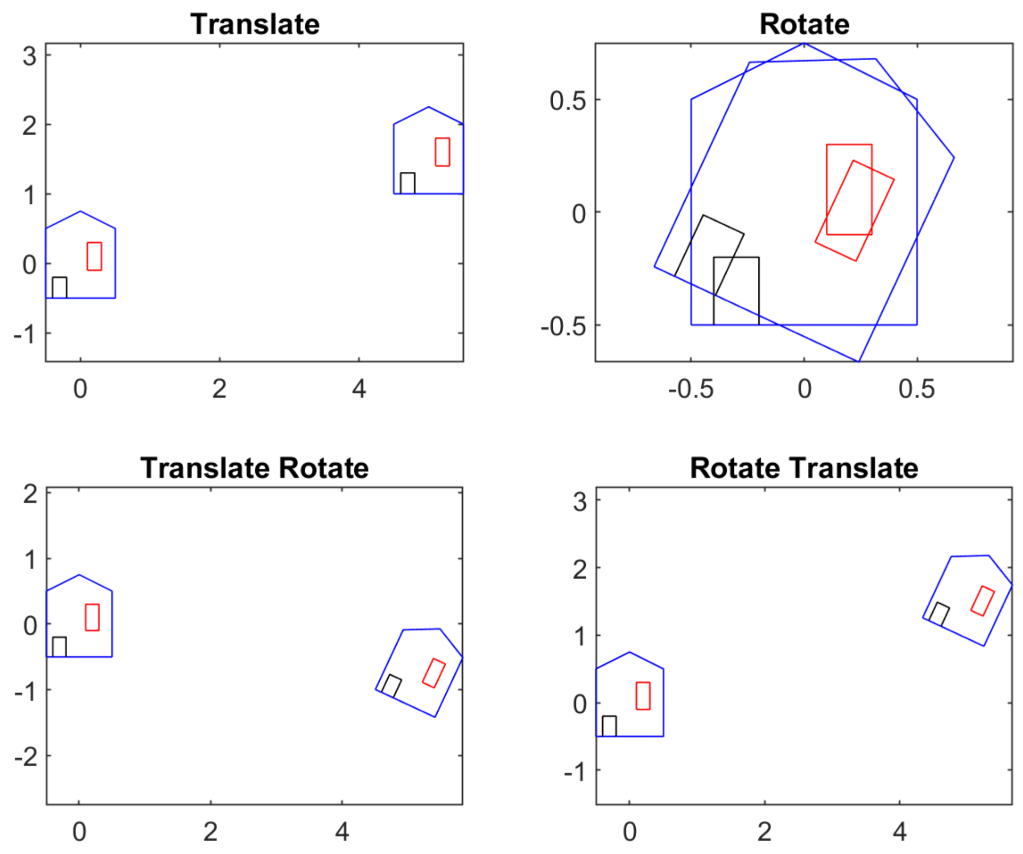
Grading Criteria:
[10 pts] [Setting up the 4 sub-plots with the house at the origin and the correct titles]
[10 pts] [Translation matrix]
[10 pts] [Translation matrix self-check]
[15 pts] [Rotation matrix]
[15 pts] [Rotation matrix self-check]
[10 pts ea] [Drawing second house – must use matrices and matrix multiplication]

Answer script here:

```
%Andrew Brown Lab 9 Problem 1

clc
clear
close all

%Use a matrix to move a house around. Transalte by (5,1.5) znd rotate by 25
%degrees then try both combination of rotating then translating

%Define given variables and translation matrix
identity3=eye(3);
mTrans=identity3; %idendtity matrix
dx=5; %change in x
dy=1.5; %change in y
mTrans(1,3)=dx; %make translation matrix
mTrans(2,3)=dy; %make translation matrix

%Translated House
subplot(2,2,1) %plot in 1st position
DrawHouse(identity3) %keep original house
hold on
DrawHouse(mTrans) %Translated house
axis equal
```

```
title('Translate')

%Define given variables and rotation matrix
mRot=identity3;
theta=25; %degrees
mRot(1,1)=cosd(theta); %make rotation matrix
mRot(1,2)=sind(theta); %make rotation matrix
mRot(2,1)=-sind(theta); %make rotation matrix
mRot(2,2)=cosd(theta); %make rotation matrix

%Rotated House
subplot(2,2,2) %plot second position
DrawHouse(identity3) %plot original house
hold on
DrawHouse(mRot) %plot rotated house
axis equal
title('Rotate')

%Translated then Rotated House
subplot(2,2,3) %plot 3rd position
DrawHouse(identity3)  %plot original house
hold on
DrawHouse(mRot*mTrans) %translated then rotated house
axis equal
title('Translate Rotate')

%Rotated then Translated House
subplot(2,2,4) %plot 4th position
DrawHouse(identity3) %plot original house
hold on
DrawHouse(mTrans*mRot) %rotated then translated house
axis equal
title('Rotate Translate')
```

Plot here:



Command window output

```
>> mTrans * [0;0;1]

ans =

    5.0000
    1.5000
    1.0000

>> dot( mRot(:,1), mRot(:,2) )

ans =

     0

>> dot( mRot(:,2), mRot(:,2) )

ans =

     1

>> mRot * mRot'

ans =
    1.0000    0.0000         0
    0.0000    1.0000         0
         0         0    1.0000

>>
```

Problem 2

Use a function (polar rose or spiral or anything else reasonable) to create (x,y) points. Move it to a new location using a rotation followed by a translation. Plot it again. Extra credit: Plot multiple copies in a circle or a grid or some other pattern using a for loop.

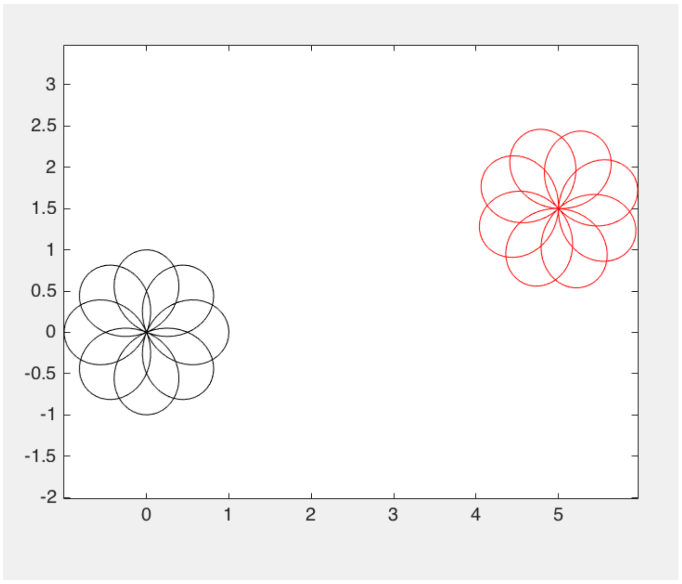
Deliverables:

5. Script to create the function and do the plot
    a. Function to create x,y values
    b. Put x,y values into a 3xn matrix
    c. Use a matrix to move the points
6. Picture with your original and moved points
7. EC: Plot with multiple copies. Must use a loop (don't do it by hand)

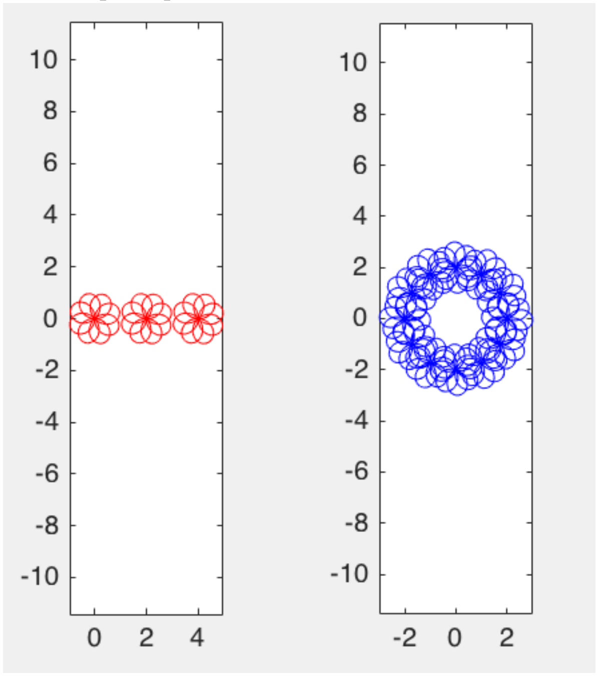
Step by Step Instructions:

- Create x and y points using either a spiral or a polar rose
    - Polar rose: r(t) is cos(n/d *t) – I used n = 4, d = 3 in the plot below
    - Spiral: r(t) is sqrt(t) (see lecture script)
    - x(t) = r(t) * cos(t)
    - y(t) = r(t) * sin(t)
- Plot and make sure you have either a spiral or a rose. Use axis equal to make sure the aspect ratio is correct
    - Note: You can do anything you want here – just produce x and y points that make an "interesting" plot – one that you can tell if you rotated it correctly or not
- Turn your x and y vectors into a 3xn matrix where n is the length of x and y
    - pts = [x; y; ones(1, length(x))]
- Copy your mTrans and mRot code from the previous problem
- Multiply the matrics by the points
    - ptsNew = mTrans * mRot * pts
- Plot the new points. Notice that ptsNew is also a 3xn matrix with ones in the 3rd row – ignore those

Self-check:



2 example possible extra credits



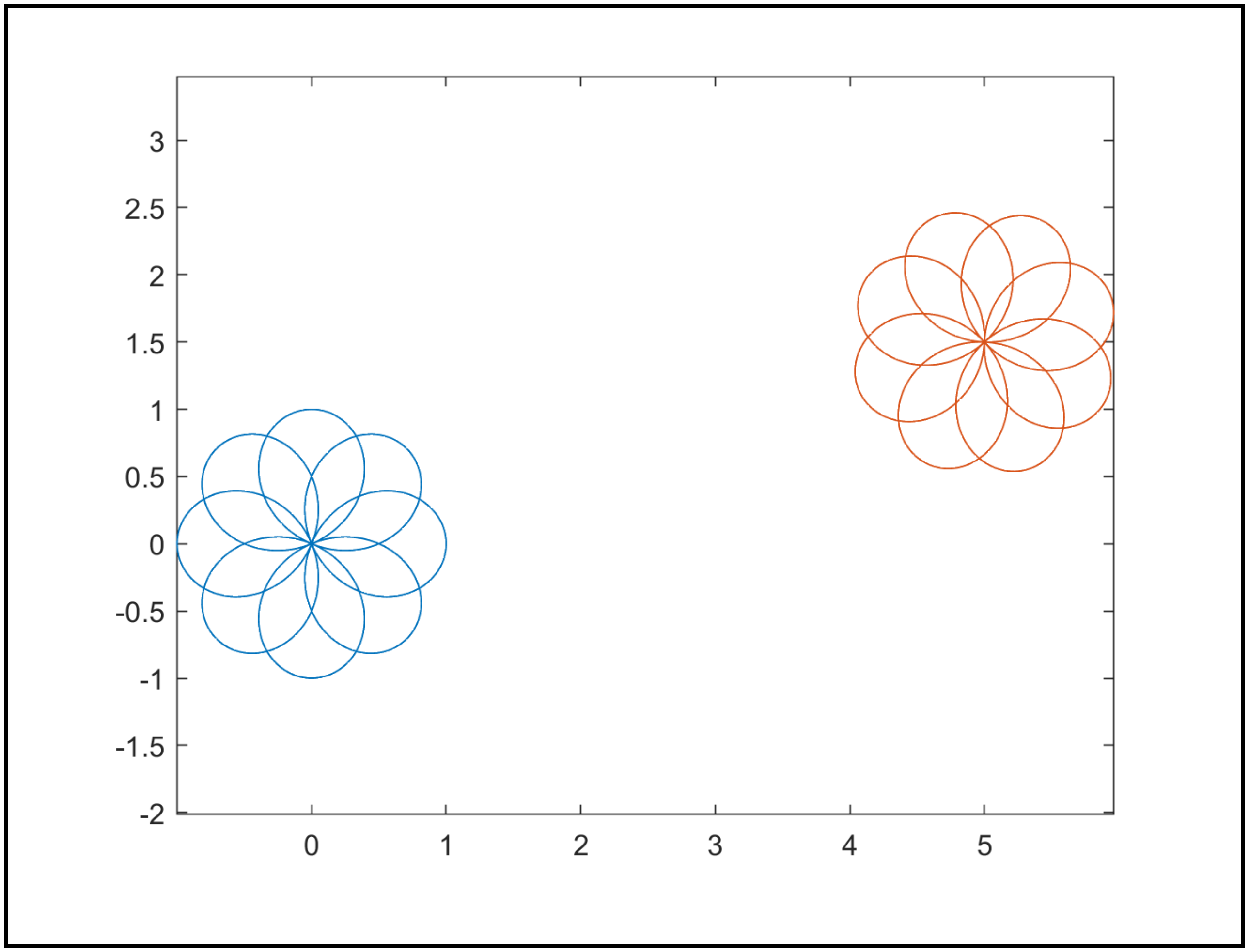
Grading Criteria:
[20 pts] [*x, y values]*
[25 pts] [*putting them into a 3xn matrix]*
[30 pts] [*Matrix multiplication to move points]*
[25 pts] [*Plot new points]*
[up to 30 pts] [*Extra credit, 15 pts plotting on a line, up to 30 for other]*

Answer script here:

```
%Andrew Brown Lab 9 Problem 2

clc
clear
```

```
close all

%Use a function (polar rose or spiral or anything else reasonable) to
%create (x,y) points. Move it to a new location using a rotation followed
%by a translation. Plot it again. Extra credit: Plot multiple copies in a
%circle or a grid or some other pattern using a for loop.

%Create a polar rose with given values
n=4;
d=3;
t=linspace(1,10000,10000);
rt=cosd(n/d*t); %given equation
xt=rt.*cosd(t); %given equation for x points
yt=rt.*sind(t); %given equation for y points
identity3=eye(3); %identity matrix

%Plot the original polar rose
plot(xt,yt)
axis equal

%Define given variables and rotation matrix
mRot=identity3;
theta=25; %degrees
mRot(1,1)=cosd(theta); %make rotation matrix
mRot(1,2)=sind(theta); %make rotation matrix
mRot(2,1)=-sind(theta); %make rotation matrix
mRot(2,2)=cosd(theta); %make rotation matrix

%Define given variables and translation matrix
identity3=eye(3);
mTrans=identity3; %idendtity matrix
dx=5; %change in x
dy=1.5; %change in y
mTrans(1,3)=dx; %make translation matrix
mTrans(2,3)=dy; %make translation matrix

%Turn x and y vectors into a 3xn matrix
pts=[xt;yt;ones(1,length(xt))];

%Multiply the matrices by the x and y points
ptsNew=mTrans*mRot*pts;

%Plot the new translated rose
hold on
plot(ptsNew(1,:),ptsNew(2,:))
```

Plot here:


3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
1
2
3
4
5

Problem 3

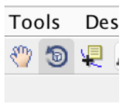
a) ***Plot your shape from problem 2 in 3D. Use a function to create x,y, and z points.***
b) ***Create a wavy surface and plot it using surf***

Deliverables:

8. Script to do the plots
9. Function to create the 3D data points for the first plot (can be any function you'd like)
10. Function to create the height data for the second plot (can be any function you'd like)
11. Screen shot of the plots from two different view points

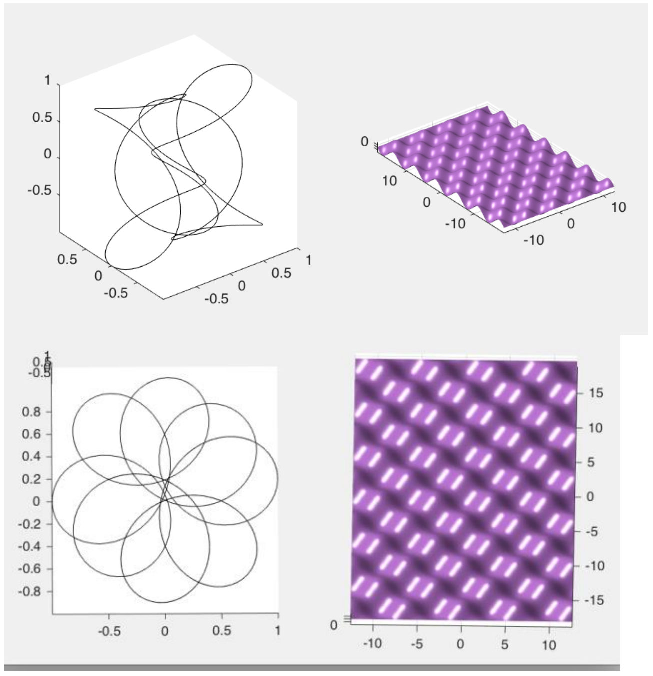
Step by Step Instructions:

- *The 3D plot:*
  - *Create a function file that takes in t and returns x,y and z*
    - *Copy in your code from the previous problem to make x and y from t*
      - *If you, eg, want to pass in n and d or some other parameters, feel free*
    - *Set z to be some function of t – I used cos(t) in the picture below*
  - In the calling script create t and pass it to your function to create the x,y,z values
    - [x, y, z] = My3DFc(t)
  - Use plot3 to plot this in 3D
- *The surface*
  - *Create a function file mySrfF that takes in s and t and returns z*
    - *This can be any function you'd like – you can use the one from lecture or the plywood function from the homework or just make something up*
    - *Make sure to use element-wise multiplication*
  - *In the calling script make a grid of values using meshgrid*
    - *s = linspace(-pi, pi);*
    - *t = linspace(-pi, pi);*
    - *[S, T] = meshgrid(s,t)*
      - *- note, I used a bigger range for the plot below*
  - *Use these values to get your z values*
    - *Z = mySrfFc(S,T)*
  - *Plot using surf*
    - *surf(S, T, Z');*
  - *Plot that way then try*
    - *surf(S, T, Z, 'FaceColor', [0.5 0.5 0.5], 'EdgeColor', 'None');*
    - *camlight left;*
    - *lighting phong;*
- *To rotate the camera use the rotate tool*

- 

- You can also set the camera viewpoint from within the code
  - view(45, 60)

Self-check:



Grading Criteria:
[20 pts] [*function file for line plot]*
[20 pts] [*Calling function file to get points]*
[10 pts] [*Actual line plot]*
[20 pts] [*function file for surface plot]*
[20 pts] [*Calling function file to get points]*
[10 pts] [*Actual surface plot]*

Answer script here:

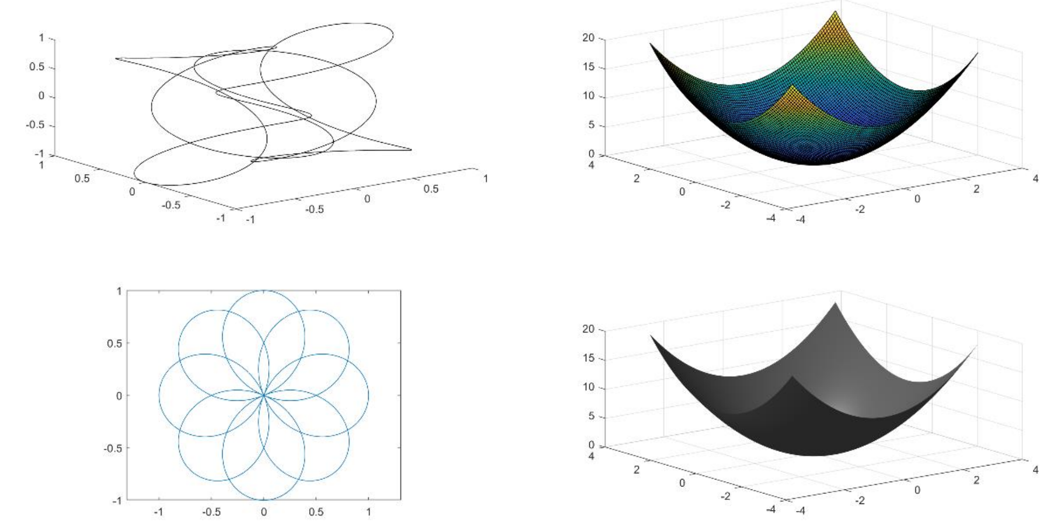
```
%Andrew Brown Lab 9 Script 3

clc
clear
close all

%Plot your shape from problem 2 in 3D. Use a function to create x,y, and z
%Create a wavy surface and plot is using surf

%Create a polar rose with given values
identity3=eye(3); %identity matrix
t=linspace(1,10000,10000);
```

```
%Call My3DFc to get the 3D points of a polar rose with input t
[x,y,z] = My3DFc(t);

%plot the original polar rose
subplot(2,2,3)
plot(x,y)
axis equal

%Plot the 3D polar rose
subplot(2,2,1)
plot3(x,y,z,'k')

%Make the survafe of the 3D polar rose
s=linspace(-pi,pi); %given value
t=linspace(-pi,pi); %given value

%make a blanket
[S,T]=meshgrid(s,t);

%Create the right values to pass through surf
[Z] = mySrfF(S,T);

%call surf and then the better surf
subplot(2,2,2)
surf(S,T,Z); %plot the regular surf
subplot(2,2,4)
surf(S,T,Z,'FaceColor',[0.5,0.5,0.5],'EdgeColor','None'); %plot the colored
structured surf
camlight left %add depth
lighting phong %add lighting
```

Function scripts here:

```
function [x,y,z] = My3DFc(t)
%MY3DFC Takes in an array 't' and creates points for a 3D polar rose
n=4; %Polar rose value
d=3; %Polar rose value
r=cosd(n/d*t); %given equation
x=r.*cosd(t); %given equation for x points
y=r.*sind(t); %given equation for y points
z=cosd(t); %given equation for z points
end

function [z] = mySrfF(s,t)
%MYSRFF makes the proper z variable for using surf
z=s.^(2)+t.^(2);
end
```

Plot here:



zyBooks Challenge Exercises

There are no challenge exercises for week 9.