Exam 1

Math 3607, Summer 2021

SMITH.11783

Table of Contents

Instructions

- Rename this file to "exam1_Lastname_Firstname.mlx" (e.g. "exam3written_Kim_TaeEun.mlx"). In addition, type in your OSU name.number at the top of this document where it says [NAME.NUMBER]. Failure to do so will result in point deduction.
- Insert your signature in the next section. Submission without a signature WILL NOT be accepted.
- Fill in the live script with your answers. Lengthy explanation of your code may be typed or handwritten. Do whichever is more convenient for you.
- Once completed, re-run the entire live script to ensure that requested outputs and figures are properly generated.
- Export to pdf. Do NOT manually modify the generated pdf file.
- Upload mlx and pdf files. Do not use any external m-files. Your submission must be self-contained.

Signature

Upon reading all the statements on the front page of the exam, please insert your signature (as a picture file) down below. Submission missing signature WILL NOT be accepted.

Academic Integrity Statements

- All of the work shown on this exam is my own.
- I will not consult with any resources (MATLAB documentation, online searches, etc.) other than the textbooks, lecture notes, and supplementary resources provided on the course Carmen pages.
- I will not discuss any part of this exam with anyone, online or offline.
- I understand that academic misconduct during an exam at The Ohio State University is very serious and can result in my failing this class or worse.
- I understand that any suspicious activity on my part will be automatically reported to the OSU Committee on Academic Misconduct for their review.

Signature

Problem 1 (Surface Plot)

This problem asks for a script that plots the surface represented by: $x = u(3 + \cos(v))\cos(2u)$,

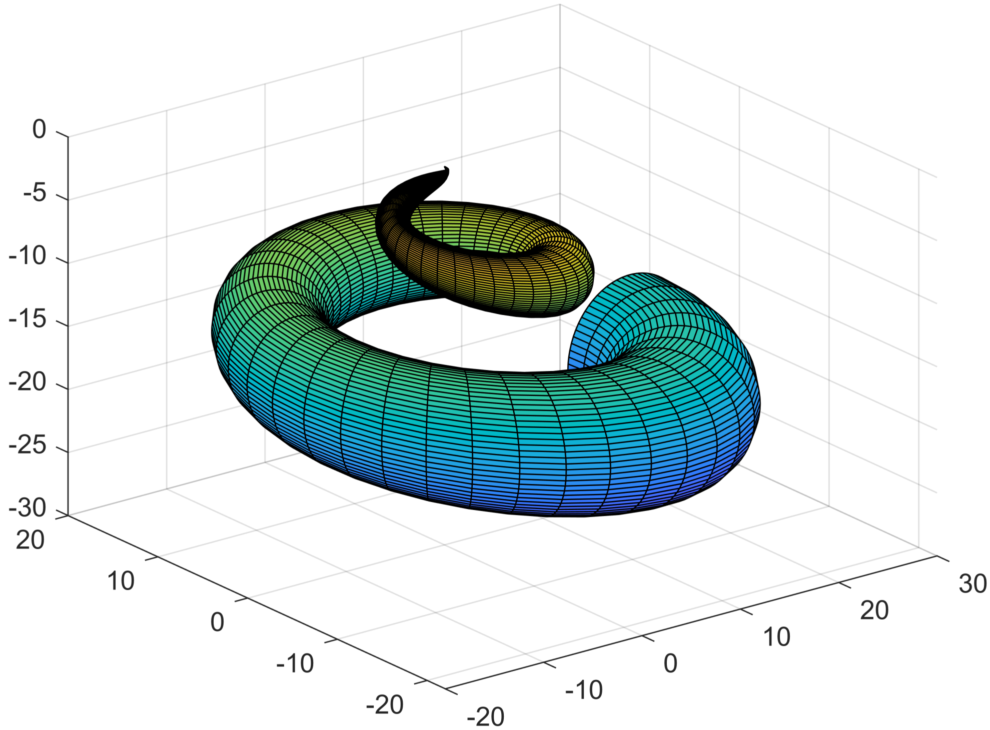
$y = u(3 + \cos(v))\sin(2u)$,

$z = \text{usin}(v) - 3u$

It is similar to ones found on the Lecture Slides for Module 1 (pg 127/128).

```
u=linspace(0,2*pi,75); %75 evenly spaced points in interval [0,2pi]
v=linspace(0,2*pi,75);
[U, V]=meshgrid(u,v);    %creates meshgrid so that we can call surf later

clf
x=(U.*(3+cos(V))).*cos(2*U); %transforms points
y=(U.*(3+cos(V))).*sin(2*U);
z=(U.*sin(V))-(3*U);
surf(x,y,z) %plots the surface using new points
```



Problem 2 (Continued Fraction)

This problem asks for a script that approximates the golden ratio until there is 16 correct digits after the decimal place. It is similar to the problem 2 from homework 2.

```
format long g
d_rot_deg=(1+sqrt(5))/2; %find golden ratio in decimal form
n=0;
phi=zeros(1,100); %create matrix large enough to store data
phi(1)=1+(1/1);   %find first iteration

string='n  phi_n                    error';
fprintf(string)
```

```
n  phi_n                    error
```

```
while (abs(phi(n+1)-d_rot_deg) > (10^-16)) %checks value of phi minus golden ratio
    n=n+1;
    phi(n+1)=1+(1/phi(n));  %applies pre-existing phi to formula to get new phi
    error=abs(phi(n)-d_rot_deg);    %calculates error b/w new phi and golden ratio
    fprintf('%i  %.16f    %.4e\n',n,phi(n),error)
end                           %YOU HAVE TO SCROLL DOWN ON OUTPUT TO GET REST OF VALUES
```

```
1  2.0000000000000000    3.8197e-01
2  1.5000000000000000    1.1803e-01
3  1.6666666666666665    4.8633e-02
4  1.6000000000000001    1.8034e-02
5  1.6250000000000000    6.9660e-03
6  1.6153846153846154    2.6494e-03
7  1.6190476190476191    1.0136e-03
8  1.6176470588235294    3.8693e-04
9  1.6181818181818182    1.4783e-04
10  1.6179775280898876    5.6461e-05
11  1.6180555555555556    2.1567e-05
12  1.6180257510729614    8.2377e-06
13  1.6180371352785146    3.1465e-06
14  1.6180327868852458    1.2019e-06
15  1.6180344478216819    4.5907e-07
16  1.6180338134001251    1.7535e-07
17  1.6180340557275543    6.6978e-08
18  1.6180339631667064    2.5583e-08
19  1.6180339985218035    9.7719e-09
20  1.6180339850173580    3.7325e-09
21  1.6180339901755971    1.4257e-09
22  1.6180339882053252    5.4457e-10
23  1.6180339889579018    2.0801e-10
24  1.6180339886704433    7.9452e-11
25  1.6180339887802426    3.0348e-11
26  1.6180339887383031    1.1592e-11
27  1.6180339887543225    4.4276e-12
28  1.6180339887482038    1.6911e-12
29  1.6180339887505406    6.4571e-13
30  1.6180339887496482    2.4669e-13
31  1.6180339887499890    9.4147e-14
32  1.6180339887498589    3.5971e-14
33  1.6180339887499087    1.3767e-14
34  1.6180339887498896    5.3291e-15
35  1.6180339887498969    1.9984e-15
36  1.6180339887498940    8.8818e-16
37  1.6180339887498951    2.2204e-16
```

Problem 3 (Array Operations)

This problem asks for a varitey of scripts that deal with array manipulation.

```
Miles = [0, 27, 69, 101, 120, ...
         154, 178, 211, 235, 278, ...
         306, 327, 356, 391, 400];
Dist=diff(Miles);
```

(a) This problem asks for the shortest distance you will travel on one of the days.

```
short=min(Dist);
fprintf('The shortest distance is: %i miles\n',short);
```

```
The shortest distance is: 9 miles
```

(b) This problem asks for the longest distance you will travel on one of the days.

```
long=max(Dist);
```

```
fprintf('The longest distance is: %i miles\n',long);
```

```
The longest distance is: 43 miles
```

(c) This problem asks for the average distance you will travel for all of the days.

```
average=mean(Dist);
fprintf('The average daily distance is: %5.2f miles\n',average);
```

```
The average daily distance is: 28.57 miles
```

(d) This problem asks for the distance you will travel on day 7.

```
day7=Dist(7);
fprintf('On day 7, you will have to go %i miles\n',day7);
```

```
On day 7, you will have to go 33 miles
```

(e) This problem asks for the halfway mileage markers of each day.

```
halfway=Dist/2;
newMiles=Miles(1:14);
lunch=halfway+newMiles;
fprintf('The mileage values for the halfway point of each day are:\n')
```

```
The mileage values for the halfway point of each day are:
```

```
disp(lunch);
```

```
  Columns 1 through 3
                    13.5                        48                        85
  Columns 4 through 6
                   110.5                       137                       166
  Columns 7 through 9
                   194.5                       223                     256.5
  Columns 10 through 12
                     292                     316.5                     341.5
  Columns 13 through 14
                   373.5                     395.5
```

(f) This problem asks for the original matrix to be produced from the Dist matrix.

```
f=[0 cumsum(Dist)];
fprintf('The original vector values are: \n')
```

```
The original vector values are:
```

```
disp(f)
```

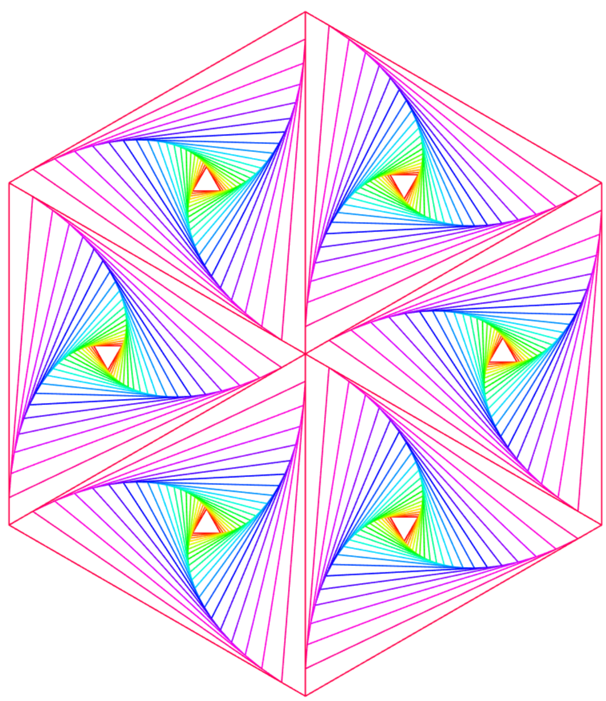
```
  Columns 1 through 14
     0    27    69   101   120   154   178   211   235   278   306   327   356   391
  Column 15
   400
```

Problem 4 (Tiling with Spiral Polygons)

(a) Modify the function `spiralgon` under "Functions Used".

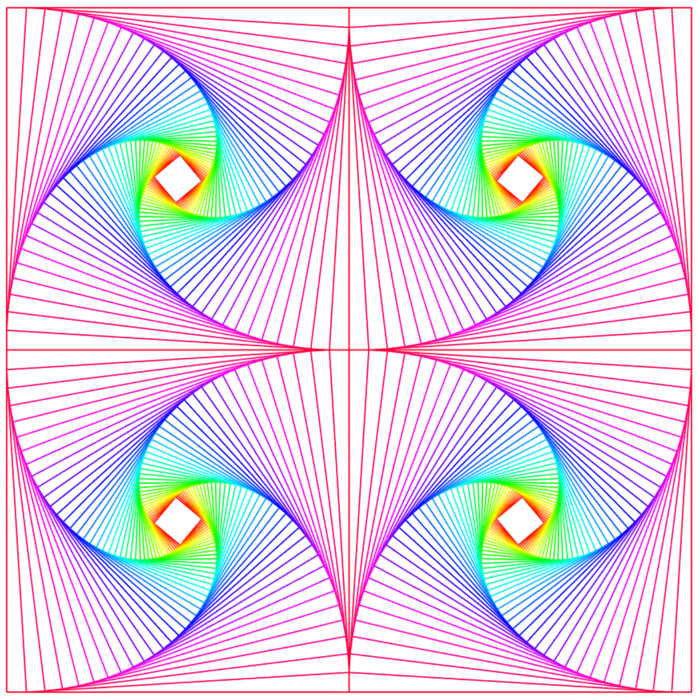
(b) For this problem I started by calling the spiralgon function to plot the middle right piece. Then I shifted its leftern-most vertex to the origin by appropriately adjusting the shift vector from [0 0] to [12.1105 0]. Next, I initialized my variables d_rot_deg and shift_deg so that I could use them in the for loop. I ran the for loop 5 times because I needed 5 more triangles. I incremented d_rot_deg by dividing 360 total degrees by 6 triangles to get 60 degrees of rotation added to that of the previous triangle. Then, rotating the shift variable gave me a lot more difficulty until I realized the relationship between this diagram and the unit circle. So, I ended up multiplying the radius (12.1105) by the corresponding cosine and sine value that was determined by the incrementing of shift_deg by $\frac{\pi}{3}$ each time.

```
%Triangles
clf
spiralgon(3,21,4.5,90,[12.1105 0]');
d_rot_deg=60;
shift_deg=(pi/3);
for n=1:5
    spiralgon(3,21,4.5,90+d_rot_deg,[12.1105*cos(shift_deg) 12.1105*sin(shift_deg)]);
    d_rot_deg=d_rot_deg+60;
    shift_deg=shift_deg+(pi/3);
end
```



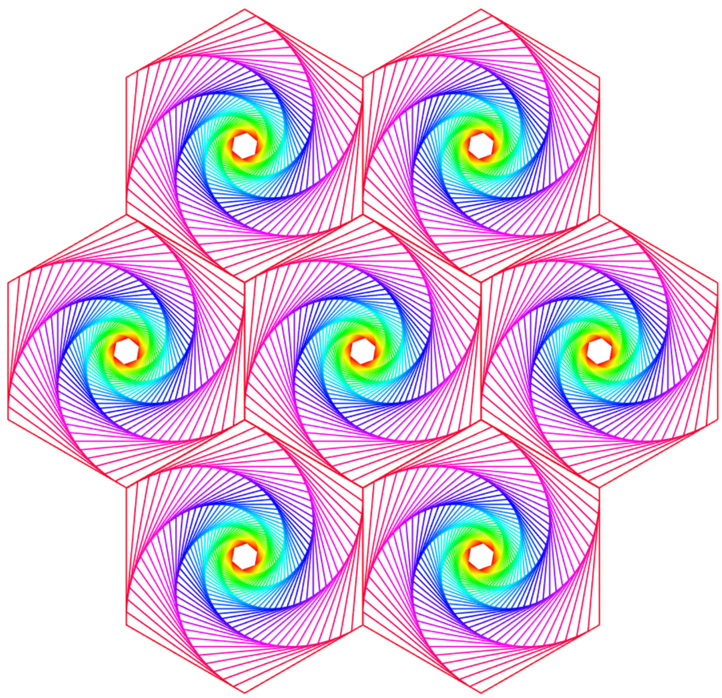
(c) For this problem I chose d_angle=-3.5 on both the bottom-left and top-right because they were the ones that had the squares rotating to the left. Similarly, I chose d_angle=3.5 for the bottom-right and top-left as they had squares rotating to the right. Then, I chose d_rot at 5 and 175 to get them in the proper orientation and used shift=[0 0], [14.1 0], [0 14.1], [14.1 14.1] to get them to align.

```
%Squares
clf
spiralgon(4,41,-3.5,5,[0 0]');
spiralgon(4,41,3.5,175, [14.1 0]');
spiralgon(4,41,3.5,175,[0 14.1]');
spiralgon(4,41,-3.5,5,[14.1 14.1]);
```



(d) For this problem I chose d_angle=5 because all of the hexagons were rotating to the right. Then, I chose d_rot=80 so that they were all vertically aligned. Then I chose the values of shift that I did in order to get them to line up with one another to form the correct shape.

```
%Hexagons
clf
spiralgon(6,51,5,80,[0 0]');
spiralgon(6,51,5,80,[16.8,0]);
spiralgon(6,51,5,80,[-16.8 0]');
spiralgon(6,51,5,80,[-8.4 14.6]');
spiralgon(6,51,5,80,[8.4 14.6]');
spiralgon(6,51,5,80,[-8.4 -14.6]');
spiralgon(6,51,5,80,[8.4 -14.6]');
```



Functions Used

`spiralgon.m` (from HW2 solution)

Modified (06/25 by Spenser Smith)

```
function V = spiralgon(n, m, d_angle, d_rot, shift)
% SPIRALGON plots spiraling regular n-gons
% input:   n = the number of vertices
%          m = the number of regular n-gons
%          d_angle = the degree angle between successive n-gons
%          (can be positive or negative)
%          d_rot = the degree angle by which the innermost n-gon
%                  is rotated
% output:  V = the vertices of the outermost n-gon
th = linspace(0, 360, n+1) + d_rot;
V = [cosd(th);
    sind(th)];
C = colormap(hsv(m));
scale = sind(90 + 180/n - abs(d_angle))/...
    sind(90 - 180/n);
R = [cosd(d_angle) -sind(d_angle);
    sind(d_angle) cosd(d_angle)];
%hold off
for i = 1:m
    if i > 1
```

```
        V = scale*R*V;
    end
    plot(V(1,:) + shift(1), V(2,:) + shift(2), 'Color', C(i,:))
    hold on
end
set(gcf, 'Color', 'w')
axis equal, axis off
end
```