Minimum Point Between Two Orbits

Connor Adams

November 30, 2021

Abstract

In this project we discuss the minimum distance between two orbits. We are given two orbits that don't touch each other, one orbit resides in the other, and they each have $(0,0)$ as a focal point. We will be using MATLAB to construct a visual for our orbits as well as contour plots, and the steepest decent method. We are given the fixed parameters $(A_1, P_1, \phi_1) = (10, 2, \pi/8)$ and $(A_2, P_2, \phi_2) = (4, 1, -\pi/7)$ to help us construct the orbits and we use a different type of formula then one typically uses to calculate the distance between a given set of points. Once we construct our orbits, then we look at the contour plot to visually see where our feasible region is. Here we hypothesized that the optimal solution for t_1 and t_2 repeat every 2π. Using MATLAB we construct the steepest decent method to locate the optimal values for t_1 and t_2. Then, we tested our hypothesis by subtracting 2π from each of our optimal values to discover that we landed on another optimal value. This discovery allowed us to create a closed form for our solutions, t_1 and t_2. After this, we generated (x_1, y_1) and (x_2, y_2) using our optimal values to find the points on each orbit where the distance is the smallest. We explored other fixed parameters to see if our steepest decent method holds for other orbits. Finally, we finished up by calculating the minimum distance of each elliptical orbit scenario and stating our conclusion. In the future we may want to explore other scenarios such as: multiple orbits, orbits in 3 dimensions, and an object interfering with the elliptical orbits.

0.1 Introduction

In the problem we have two orbits that has $(0,0)$ as a focal point. These orbits are written as:

$$\begin{bmatrix} x_1(t_1) \\ y_1(t_1) \end{bmatrix} = \begin{bmatrix} \cos(\phi_1) & \sin(\phi_1) \\ -\sin(\phi_1) & \cos(\phi_1) \end{bmatrix} \begin{bmatrix} \frac{P_1-A_1}{2} + \frac{P_1+A_1}{2}\cos(t_1) \\ \sqrt{P_1 A_1} sin(t_1) \end{bmatrix}$$

and

$$\begin{bmatrix} x_2(t_2) \\ y_2(t_2) \end{bmatrix} = \begin{bmatrix} \cos(\phi_2) & \sin(\phi_2) \\ -\sin(\phi_2) & \cos(\phi_2) \end{bmatrix} \begin{bmatrix} \frac{P_2-A_2}{2} + \frac{P_2+A_2}{2}\cos(t_2) \\ \sqrt{P_2 A_2} sin(t_2) \end{bmatrix}$$

Where (A_1, P_1, ϕ_1) and (A_2, P_2, ϕ_2) are the fixed parameters designed to determine the width, height, and angle of rotation of each elliptical orbit. We want to know the minimum distance between the 2 orbits, $(A_1, P_1, \phi_1) = (10, 2, \pi/8)$ and $(A_2, P_2, \phi_2) = (4, 1, -\pi/7)$. We use the below formula to measure the distance between two points.

$$dis(t_1, t_2) = \frac{1}{2}\left[(x_1(t_1) - x_2(t_2))^2 + (y_1(t_1) - y_2(t_2))^2\right]$$

Notice, that this function is dependent on t_1 and t_2 which will become useful later on. In order to find the minimum we will first plot the two orbits, so we have an idea of where our minimum points should be. Then, we will plot the contour lines of $dis(t_!, t_2)$ to check where the feasible region is. Lastly, derive the gradient and use steepest decent to find our minimum distance.

0.2 Discussion

Before we begin finding the minimum points, first let's plot the two orbits so we can see where the minimum should be around. We will plug in the parameters for our orbits and see what our vectors looks like.

$$\begin{bmatrix} x_1(t_1) \\ y_1(t_1) \end{bmatrix} = \begin{bmatrix} \cos(\frac{\pi}{8}) & \sin(\frac{\pi}{8}) \\ -\sin(\frac{\pi}{8}) & \cos(\frac{\pi}{8}) \end{bmatrix} \begin{bmatrix} \frac{2-10}{2} + \frac{2+10}{2}\cos(t_1) \\ \sqrt{2*10}\sin(t_1) \end{bmatrix} = \begin{bmatrix} \cos(\frac{\pi}{8}) & \sin(\frac{\pi}{8}) \\ -\sin(\frac{\pi}{8}) & \cos(\frac{\pi}{8}) \end{bmatrix} \begin{bmatrix} -4 + 6\cos(t_1) \\ 2\sqrt{5}\sin(t_1) \end{bmatrix}$$

$$\begin{bmatrix} x_1(t_1) \\ y_1(t_1) \end{bmatrix} = \begin{bmatrix} -4 * \cos(\frac{\pi}{8}) + 6\cos(\frac{\pi}{8})\cos(t_1) + 2\sqrt{5} * \sin(\frac{\pi}{8})\sin(t_1) \\ 4\sin(\frac{\pi}{8}) - 6\sin(\frac{\pi}{8})\cos(t_1) + 2\sqrt(5) * \cos(\frac{\pi}{8})\sin(t_1) \end{bmatrix}$$

$$\begin{bmatrix} x_2(t_2) \\ y_2(t_2) \end{bmatrix} = \begin{bmatrix} \cos(-\frac{\pi}{7}) & \sin(-\frac{\pi}{7}) \\ -\sin(-\frac{\pi}{7}) & \cos(-\frac{\pi}{7}) \end{bmatrix} \begin{bmatrix} \frac{1-4}{2} + \frac{1+4}{2}\cos(t_2) \\ \sqrt{1*4}\sin(t_2) \end{bmatrix} = \begin{bmatrix} \cos(\frac{\pi}{7}) & -\sin(\frac{\pi}{7}) \\ \sin(\frac{\pi}{7}) & \cos(\frac{\pi}{7}) \end{bmatrix} \begin{bmatrix} -1.5 + 2.5\cos(t_2) \\ 2\sin(t_2) \end{bmatrix}$$

$$\begin{bmatrix} x_2(t_2) \\ y_2(t_2) \end{bmatrix} = \begin{bmatrix} -1.5 * \cos(\frac{\pi}{7}) + 2.5\cos(\frac{\pi}{7})\cos(t_2) - 2\sin(\frac{\pi}{7})\sin(t_2) \\ -1.5\sin(\frac{\pi}{7}) + 2.5\sin(\frac{\pi}{7})\cos(t_2) + 2\cos(\frac{\pi}{7})\sin(t_2) \end{bmatrix}$$

Since t_1 and t_2 are both inside trig functions without any scalars attached to them, then this implies that our formula has a period of 2π. So, assuming t_1 and t_2 start at 0 then after $t_1 = t_2 = 2\pi$ the values that make up these orbits will just repeat which indeed creates the orbit.

So, this means we can plug in an interval of points for t_1 and t_2 as long as the length of the interval is greater than 2π to get the below figure.

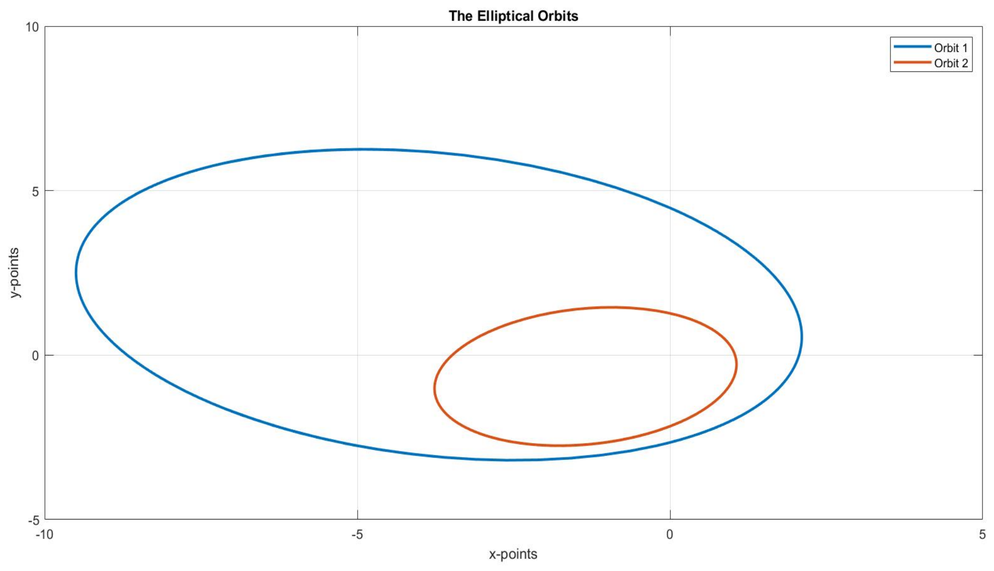

Figure 1: This plot represents both elliptical orbits with the initial conditions $(A_1, P_1, \phi_1) = (10, 2, \pi/8)$ and $(A_2, P_2, \phi_2) = (4, 1, -\pi/7)$.

We can see where the minimum values should be around,(maybe $(x_1, y_1) = (-1, -3)$ and $(x_2, y_2) = (-1, -2.5)$. So let's look at the contour plot to see approximately which t_1 and t_2 values will give us our optimal (x_1, y_1) and (x_2, y_2) solutions.

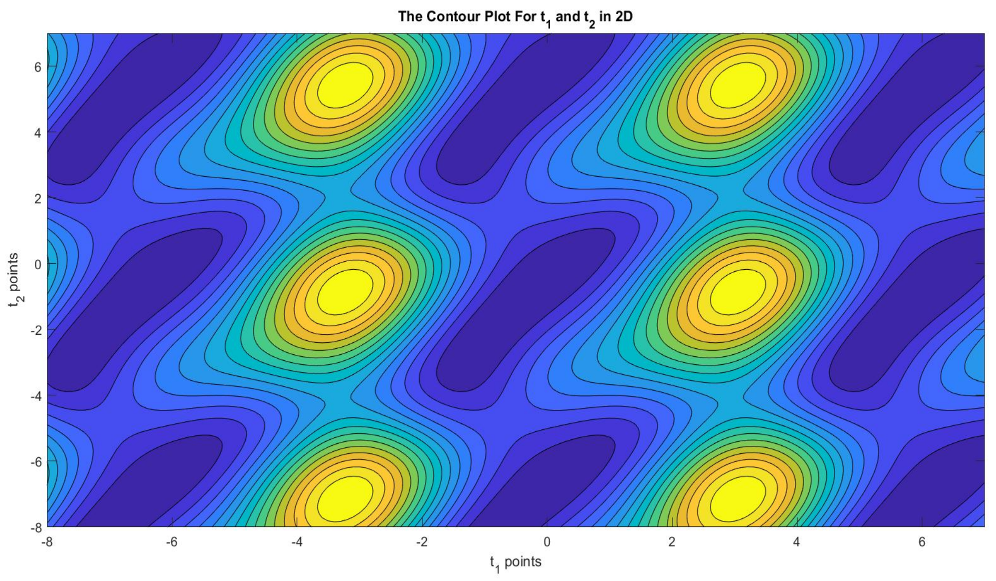

Figure 2: The Contour plot in 2D to give us an idea where our t_1 and t_2 values are optimal to find the minimum distance.

With our contour plot we noticed that there are quite a few points that could be our t_1 and t_2 solutions. This is because for every time t_1 and t_2 complete a full period, 2π, there is another set of solutions. So, from here we can create a closed form for our solutions once we find our first optimal t_1 and t_2.

To get a better idea of our minimums we can look at the contour plot in 3D.

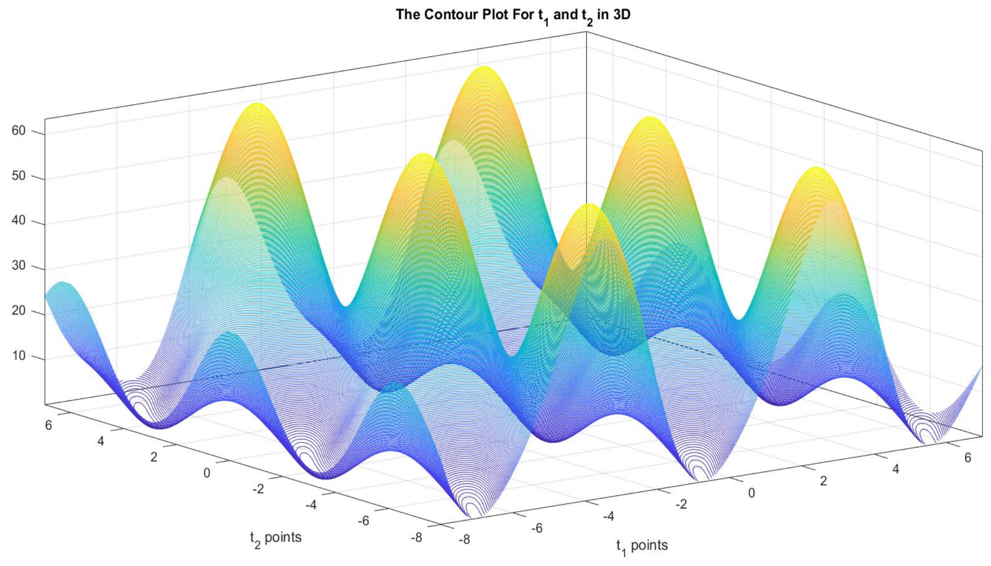

Figure 3: Contour plot in 3D to give a better picture of our feasible regions.

I really like looking at the contour map in 3D because we can start to really see that the feasible solutions repeat over some period because of its sinusoidal appearance.

So, now that we have an idea of where our solutions are at, we can then start finding them and checking to see if they are correct. First, let's begin by finding the gradient of our distance formula.

$$\nabla dis(t_1,t_2) = \frac{1}{2}\begin{bmatrix} 2x_1'(t_1)(x_1(t_1)-x_2(t_2))+2y_1'(t_1)(y_1(t_1)-y_2(t_2)) \\ -2x_2'(t_1)(x_1(t_1)-x_2(t_2))-2y_2'(t_1)(y_1(t_1)-y_2(t_2)) \end{bmatrix} = \begin{bmatrix} 2x_1'(t_1)(x_1(t_1)-x_2(t_2))+2y_1'(t_1)(y_1(t_1)-y_2(t_2)) \\ -x_2'(t_1)(x_1(t_1)-x_2(t_2))-y_2'(t_1)(y_1(t_1)-y_2(t_2)) \end{bmatrix}$$

Remember, we know what x_1, x_2, y_1, and y_2 are so if see what the derivative of those look like then, we could use MATLAB to piece together the gradient when solving for the solutions.

$$\begin{aligned} x_1'(t_1) &= -6\cos(\tfrac{\pi}{8})\sin(t_1) + 2\sqrt{5} * \sin(\tfrac{\pi}{8})\cos(t_1) \\ y_1'(t_1) &= 6\sin(\tfrac{\pi}{8})\sin(t_1) + +2\sqrt{5} * \cos(\tfrac{\pi}{8})\cos(t_1) \\ x_2'(t_2) &= -2.5\cos(\tfrac{\pi}{7})\sin(t_2) - 2\sin(\tfrac{\pi}{7})\cos(t_2) \\ y_2'(t_2) &= -2.5\sin(\tfrac{\pi}{7})\sin(t_2) + 2\sin(\tfrac{\pi}{7})\cos(t_2) \end{aligned}$$

So, now that we have the derivatives of x_1, x_2, y_1 and y_2 we can use MATLAB to build the gradient and use it to find the minimum points. We will be using steepest decent method to find these points. Below explains how it works.

First we want to start with an initial point, $t_{1,0}$ and $t_{2,0}$. We can make these point anything we like. For example, we will start with $t_{1,0} = 2$ and $t_{2,0} = 4$. We then, plug these initial values into our gradient to get: . Now, we will use the equation:

$$x_{k+1} = x_k - \lambda_k \nabla dis(t_{1,k}, t_{2,k})$$

where, $x_{k+1} = [t_{1,k+1}, t_{2,k+1}]$. Now, think of λ_k as the step sizes that this sequence takes in order to get to the optimal solution. Remember, that we want to find the minimum distance between 2 elliptical orbits. So, we are trying to:

$$\text{Minimize } dist(x_k - \lambda_k \nabla dis(t_{1,k}, t_{2,k}))$$

So, we need to pick a λ_k such that it keeps us in the direction we want to go and is small enough that we do not overshoot our minimum point. Since, we are dealing with sine and cosine functions within our distance formula, we are likely to experience a lot of minimums. This means we must sort through all the minimums to find the find the one where λ_k is closest to 0 to keep our step sizes as small and controlled as possible.

I did this by first by finding a large number of the critical values by taking the derivative of our distance formula with respect to λ_k. Then, I tested each critical value by plugging it into the 2nd derivative with respect to λ_k of our distance formula. We can tell if a critical value is at a minimum if the 2nd derivative is positive. Once, I collected all the potential λ_ks that satisfy these conditions from my data set, I found the value that is closest to

0. I then used this λ_k as my step size as discussed above. I used 20 iterations of steepest decent to approximate the optimal t_1 and t_2 values. Below, is the steepest decent iterations plotted on our contour map starting from our initial point $(3,2)$. Notice that during each iteration the direction of the line seems to move perpendicular to the contour lines. This result arises because of how we acquired our step sizes as discussed earlier.

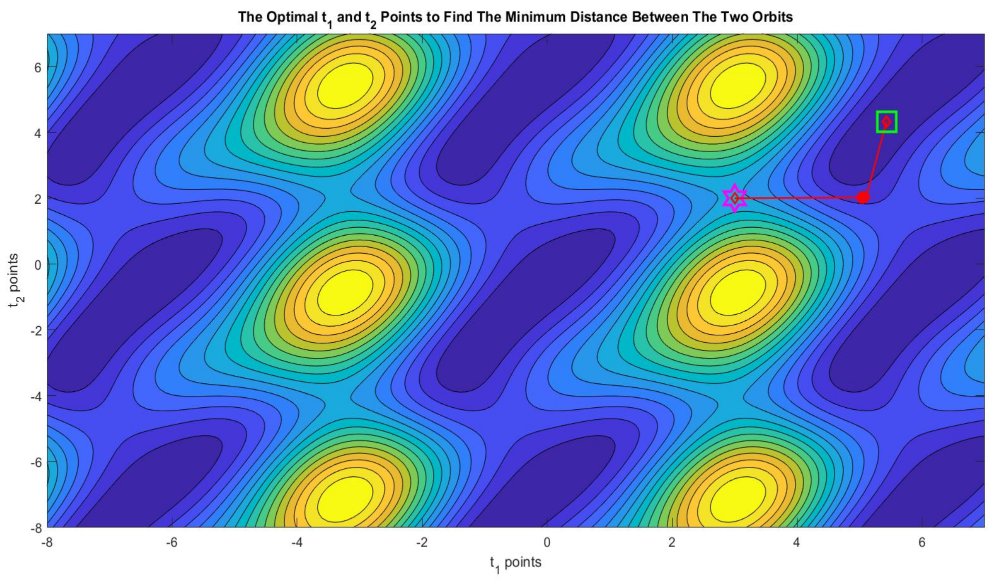

Figure 4: This is our contour map that illustrates each step of the steepest decent method starting from the magenta star to the green box which represents our optimal solution.

We can see that the initial point picked was on a saddle point and converges to a minimum. Below illustrates the same plot, but with the data points labeled on the graph.

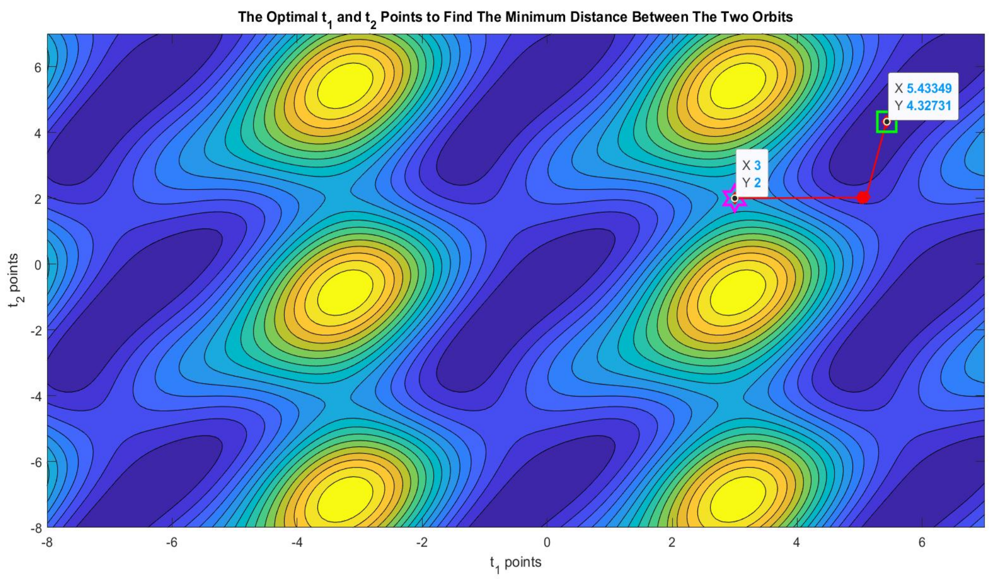

Figure 5: This is our contour map that illustrates each step of the steepest decent method starting from $(3,2)$ and converges at $(5.433, 4.327)$.

This even gets more interesting when we include a quiver plot over our existing plot as shown below.

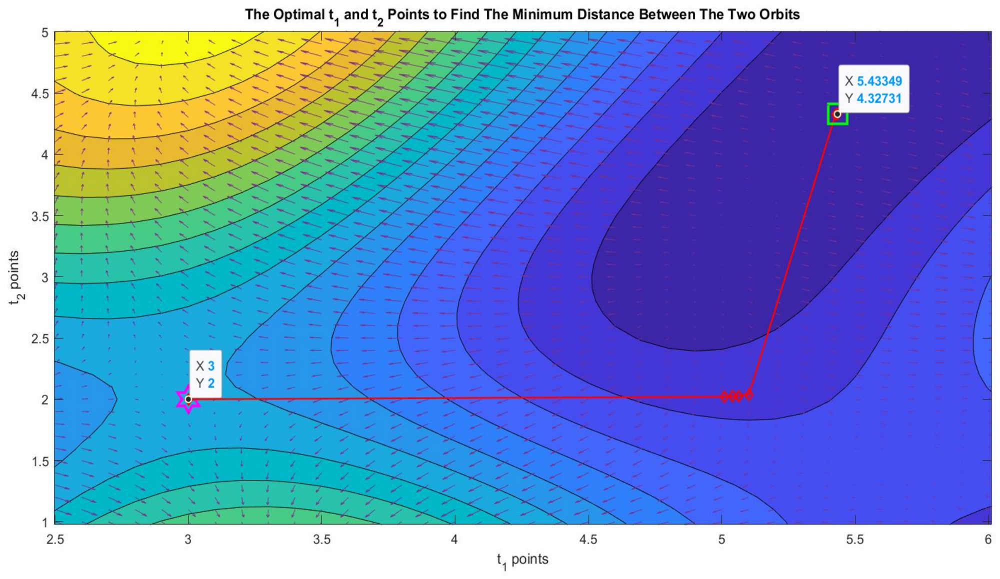

Figure 6: Same as fig. 5, but with a quiver plot on top and zoomed in around our points.

Notice, how all the arrows show the direction of the curve leading to the maximum. Also, all the arrows seem to be pointing away from our optimal point $(5.433, 4.327)$ which gives us a pretty good indication that this is our minimum/optimal (t_1, t_2).

If we plug these values in for our $\begin{bmatrix} x_1(t_1) \\ y_1(t_1) \end{bmatrix}$ and $\begin{bmatrix} x_2(t_2) \\ y_2(t_2) \end{bmatrix}$ to get

$$\begin{bmatrix} x_1(5.433) \\ y_1(5.433) \end{bmatrix} \approx \begin{bmatrix} -1.32 \\ -3.09 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} x_2(4.327) \\ y_2(4.327) \end{bmatrix} \approx \begin{bmatrix} -1.39 \\ -2.73 \end{bmatrix}$$

We can then plot this on our two orbits to get a better picture of what we are talking about.

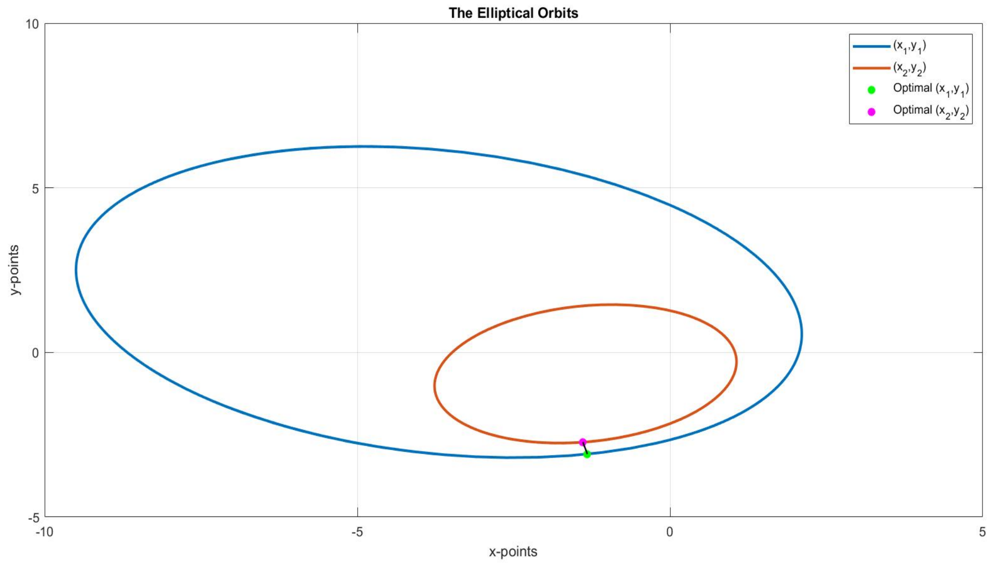

Figure 7: A plot of the orbits with the points that represent the minimum distance between the 2 orbits.

We can see that this is right around our prediction and seems to be decently accurate to our minimum. Keep in mind that this is probably not the actual minimum but it's close and we can probably get closer if we increase the number of iterations in our steepest decent method.

Remember when I spoke earlier about how the optimal points should repeat every 2π, so we can use this to find some new optimal points. We can create a closed form of our optimal solutions now that we have our first optimal solution.

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \approx \begin{bmatrix} 5.433 + 2\pi k \\ 4.327 + 2\pi r \end{bmatrix} \quad \text{Where } r \text{ \& } k \text{ are integers.}$$

So, we can create a new optimal t_1 and t_2 by making $k = r = -1$:

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \approx \begin{bmatrix} -0.850 \\ -1.956 \end{bmatrix}$$

So, if we look back at our contour map, then we can see that this belongs in a new feasible region.

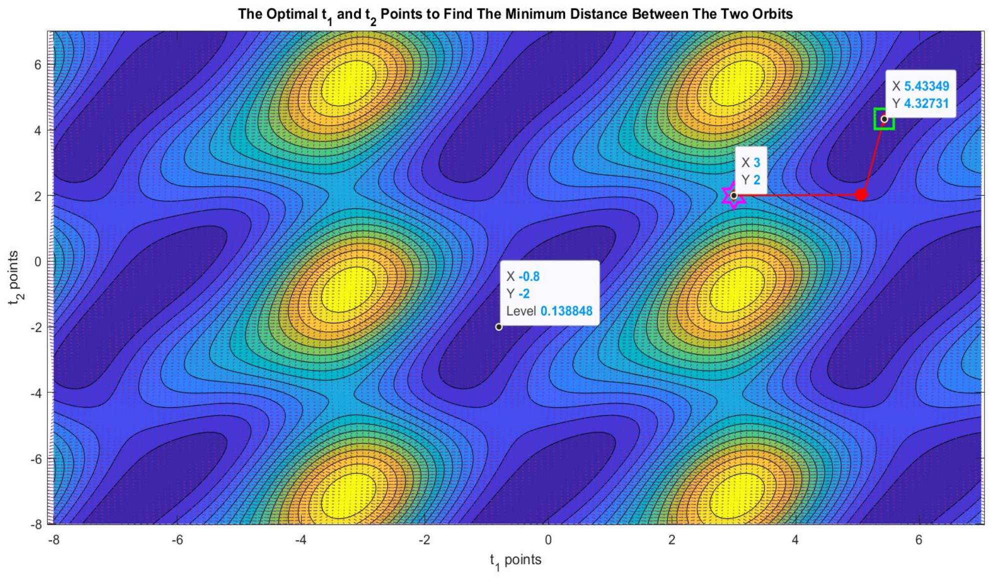

Figure 8: We can see that our prediction of our new optimal solution for t_1 and t_2 is in a feasible region

Now, if we plug these new optimal solutions to solve for (x_1, y_1) and (x_2, y_2) we get the points $(-1.32, -3.09)$ and $(-1.39, -2.73)$ which are the same points we got before which implies that our prediction was correct.

Lastly, I plugged our points into the formula we used in this project to measure distance and got 1.562 units. Then, I plugged the two points into the standard distance formula and found that the closest these 2 orbits get to each other is 1.767 units.

0.3 New Parameters

We can mess with the parameters a little bit to see what new orbits will appear. My new parameters are $(12, 9, \pi/6)$ and $(5, 2, -2pi/3)$. Then this gives us 2 orbits that look like:

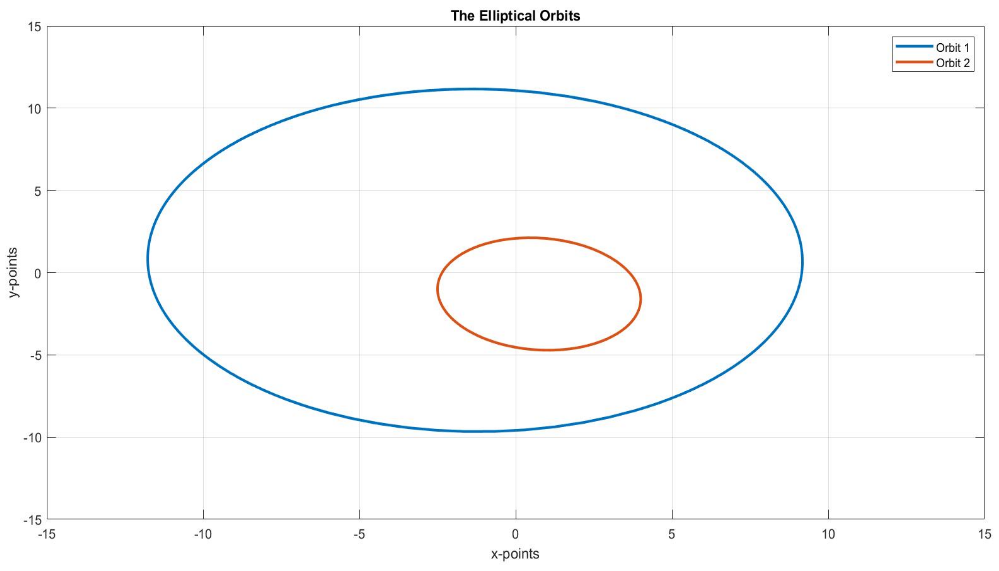

Figure 9: These are our new orbits that represented by our new initial parameters.

From the plot we can see that these two orbits don't look very close together, but it will be interesting to see where the orbits are closest. Now, we can look at the contour map to see where one of the feasible regions are.

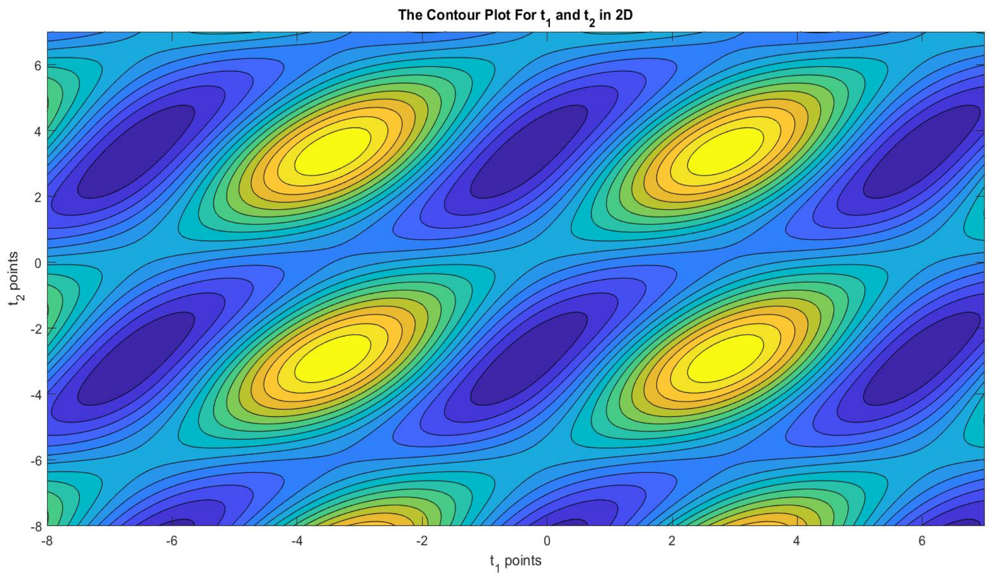

Figure 10: Our new contour map show us where our new feasible regions are.

Now, that we know where one of our feasible regions are we can pick an initial starting point for our steepest decent method. I decided to pick $t_{1,0} = -1.5$ and $t_{2,0} = -2$ as our initial conditions. So when we run our steepest decent method we get:

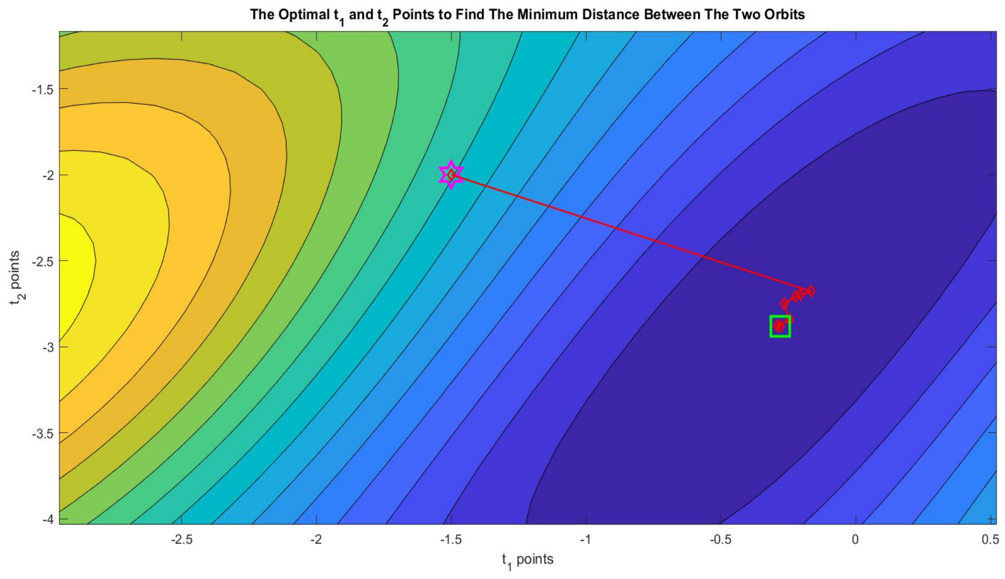

Figure 11: Our new optimal points plotted on our new contour plot.

We can see that it converges quite nicely. We can add some points to gain a better picture:

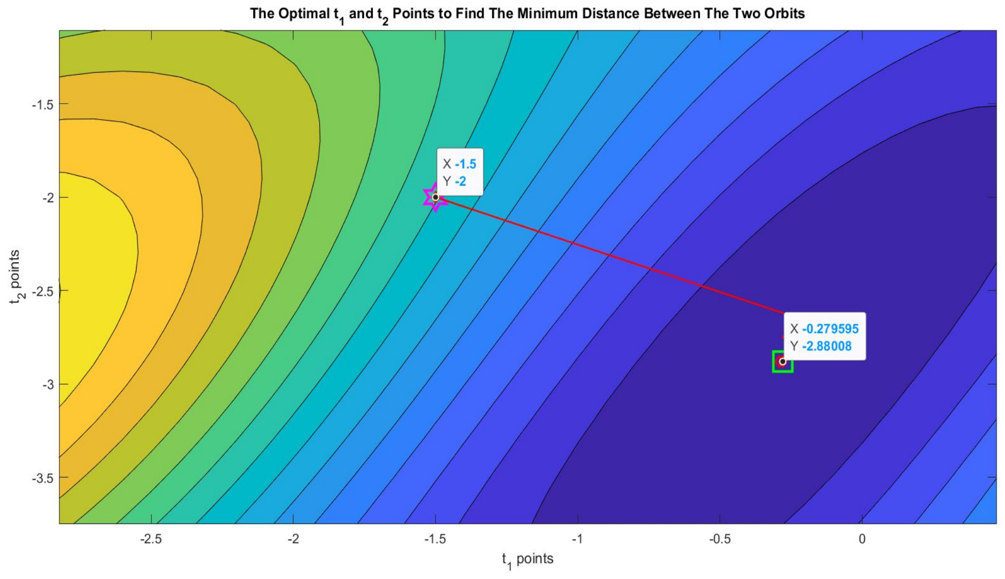

Figure 12: Our new optimal points plotted and labeled on our new contour plot.

From our picture we can now determine that our new optimal solution is:

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \approx \begin{bmatrix} -0.280 + 2\pi k \\ -2.88 + 2\pi r \end{bmatrix} \quad \text{Where } r \text{ \& } k \text{ are integers.}$$

Then if we plug our optimal t_1 and t_2 values in we get:

$$\begin{bmatrix} x_1(-0.280) \\ y_1(-0.280) \end{bmatrix} \approx \begin{bmatrix} 6.007 \\ -6.770 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} x_2(-2.88) \\ y_2(-2.88) \end{bmatrix} \approx \begin{bmatrix} 3.149 \\ -3.818 \end{bmatrix}$$

Now, that we know our coordinates we can then plot them back on our orbits to visually see the distance.

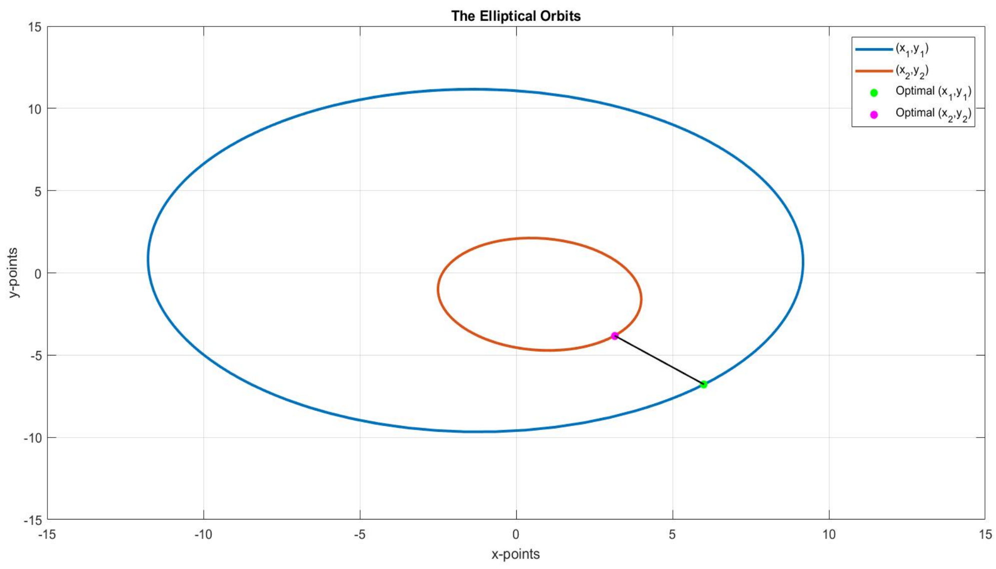

Figure 13: We can visually see the minimum distance clearly when we have the points plotted

I plugged our points into the formula we used in this project to measure distance and got 8.471 units. Then, I plugged the two points into the standard distance formula and found the closest point these orbits are from each other is 4.116 units away.

0.4 Source Code

PROJECT CODE

```
% Project for MAT 5800

clear;
close all;
clc;
% We can play with parameters that are (P+A)/2 = sqrt(PA)
par1 = [12,9,pi/6];
par2 = [5,2,-2*pi/3];
par = [par1;par2];
[m,n] = size(par);
syms t

for i = 1:m
   alpha = (par(i,2) - par(i,1)) / 2;
   beta = (par(i,2) + par(i,1)) / 2;
   gamma = sqrt(par(i,2) * par(i,1));
   orbit_vec(:,i) = [cos(par(i,3)), sin(par(i,3));...
       -sin(par(i,3)), cos(par(i,3))]*[alpha + beta*cos(t); ...
       gamma * sin(t)];
end

[t1,t2] = meshgrid(-8:.1:7);
x1 = subs(orbit_vec(1,1),t,t1);
y1 = subs(orbit_vec(2,1),t,t1);
x2 = subs(orbit_vec(1,2),t,t2);
y2 = subs(orbit_vec(2,2),t,t2);

dist_cont = .5 .* ((x1-x2).^2 + (y1-y2).^2);

figure(1)
contourf(t1,t2,dist_cont,15);
xlabel('t_1 points')
ylabel('t_2 points')
title('The Optimal t_1 and t_2 Points to Find The Minimum Distance Between The Two
    Orbits')
hold on
figure(2)
contour3(t1,t2,dist_cont,200);
xlabel('t_1 points')
ylabel('t_2 points')
title('The Contour Plot For t_1 and t_2 in 2D')


syms t t1 t2

x1 = subs(orbit_vec(1,1),t,t1);
y1 = subs(orbit_vec(2,1),t,t1);
x2 = subs(orbit_vec(1,2),t,t2);
y2 = subs(orbit_vec(2,2),t,t2);

figure(3)
fplot(orbit_vec(1,1),orbit_vec(2,1),'linewidth',2)
hold on
fplot(orbit_vec(1,2),orbit_vec(2,2),'linewidth',2)
hold on
axis([-15, 15, -15, 15])
grid on
xlabel('x-points')
ylabel('y-points')
title('The Elliptical Orbits')
legend('Orbit 1', 'Orbit 2','Location','NorthEast')


x_1=@(t1) subs(orbit_vec(1,1),t,t1);
y_1=@(t1) subs(orbit_vec(2,1),t,t1);
x_2=@(t2) subs(orbit_vec(1,2),t,t2);
```

```
y_2=@(t2) subs(orbit_vec(2,2),t,t2);

dist=@(x1,x2,y1,y2) .5 * ((x1-x2)^2 + (y1-y2)^2);

    % Creating quiver plot
    [x,y] = meshgrid(-8:.1:7);
    dist1=@(t1,t2) dist(x_1(t1),x_2(t2),y_1(t1),y_2(t2));

    for i = 1:length(x)
        for j = 1:length(y)
        dist2(i,j) = double(dist1(x(i,j),y(i,j)));
        end
    end

    dist2 = double(dist2);

    [Dt1,Dt2] = gradient(dist2,.5);

    figure(1)
    quiver(x,y,Dt1,Dt2)
    hold on

% I want to perform steepest decent 10 times.
num_point=4;
t1_0 = [-2, -3.2, 1.8, -1.5]; t2_0 = [2.5, -1, -.5, -2];
figure(1)
plot(t1_0(num_point),t2_0(num_point),'mh','MarkerSize',20,'LineWidth',2)
hold on

% My initial search
n = 20;
[x,y] = Project_Steepest_Decent_MAT_5800(dist, x1, x2, y1, y2,...
    x_1, x_2, y_1, y_2, t1_0(num_point), t2_0(num_point), n);
t1 = x;
t2 = y;


Opt_t1 = t1(n+1);
Opt_t2 = t2(n+1);

figure(1)
plot(t1,t2,'rd-','LineWidth',1.3)
hold on
plot(Opt_t1,Opt_t2,'gs','MarkerSize',20,'LineWidth',2)
hold on
hold off


x1 = double(subs(orbit_vec(1,1),t,Opt_t1))
y1 = double(subs(orbit_vec(2,1),t,Opt_t1))
x2 = double(subs(orbit_vec(1,2),t,Opt_t2))
y2 = double(subs(orbit_vec(2,2),t,Opt_t2))

distance = dist(x1,x2,y1,y2);

figure(3)
plot(x1,y1,'g*','LineWidth',2)
hold on
plot(x2,y2,'m*','LineWidth',2)
hold on
plot([x1,x2],[y1,y2],'k-','LineWidth',1.2)
hold on
legend('(x_1,y_1)','(x_2,y_2)','Optimal (x_1,y_1)','Optimal (x_2,y_2)',...
    'Location','NorthEast')
hold on
```

STEEPEST DECENT METHOD

```
% Steepest Decent of Project MAT 5800
% Connor Adams
function [x,y] = Project_Steepest_Decent_MAT_5800(f, x1, x2, y1, y2,...
    x_1, x_2, y_1, y_2, x, y, n)


syms t1 t2
for i = 1:n
    i
            f_sym = f(x1,y1,x2,y2);
            Df = gradient(f_sym);
            df = subs(Df,[t1,t2],[x(i),y(i)]);

            dfdx = df(1);
            dfdy = df(2);

        slope(i) = -double(dfdx);
        slope(i+1) = -double(dfdy);

    syms d
% Trouble spot

dist_dir=@(d) f(x_1((x(i)+d*slope(i))),x_2(y(i)+d*slope(i+1)),...
    y_1(x(i)+d*slope(i)),y_2(y(i)+d*slope(i+1)));

hess_dist = -1;
upper = 2;
lower = -2;
count1 = 1;
count2 = 1;
while hess_dist < 0
    dir_star1 = double(vpasolve(gradient(dist_dir(d)),d,[lower,0]));
    dir_star2 = double(vpasolve(gradient(dist_dir(d)),d,[0,upper]));
    if count1==1 && count2==1 && (isequal(dir_star1,double.empty(0,1))==1 ||...
            isequal(dir_star2,double.empty(0,1)) == 1)
       if isequal(dir_star1,double.empty(0,1))==1
        lower = lower - 2;
       elseif isequal(dir_star2,double.empty(0,1))==1
        upper = upper + 2;
       end
       continue;
    end
    if lower >= 0 || upper <= 0
        dir_star1 = unique(dir_star1_collect);
        dir_star2 = unique(dir_star2_collect);
        dir_star1 = sort(dir_star1,'descend');
        for k = 1:length(dir_star1)
            hess1 = double(subs(hessian(dist_dir(d)),d,dir_star1(k)));
            if hess1 > 0
                dir_star1 = dir_star1(k);
                break
            end
        end
        for k = 1:length(dir_star2)
            hess2 = double(subs(hessian(dist_dir(d)),d,dir_star2(k)));
            if hess2 > 0
                dir_star2 = dir_star2(k);
                break
            end
        end
    else
    if isequal(dir_star1,double.empty(0,1)) == logical(false)...
            && isequal(dir_star2,double.empty(0,1)) == logical(false)
       lower = lower + .1;
       upper = upper - .1;
       dir_star1_collect(count1) = dir_star1;
       dir_star2_collect(count2) = dir_star2;
       count1 = count1 + 1;
       count2 = count2 + 1;
       continue;
    elseif isequal(dir_star1,double.empty(0,1)) == logical(false)
        lower = lower + .1;
        dir_star1_collect(count1) = dir_star1;
        count1 = count1 + 1;
        continue;
    elseif isequal(dir_star2,double.empty(0,1)) == logical(false)
        upper = upper - .1;
        dir_star2_collect(count2) = dir_star2;
        count2 = count2 + 1;
        continue;
    else
        dir_star1 = unique(dir_star1_collect);
        dir_star2 = unique(dir_star2_collect);
        dir_star1 = sort(dir_star1,'descend');
        for k = 1:length(dir_star1)
            hess1 = double(subs(hessian(dist_dir(d)),d,dir_star1(k)));
            if hess1 > 0
                dir_star1 = dir_star1(k);
                break
            end
        end
        for k = 1:length(dir_star2)
            hess2 = double(subs(hessian(dist_dir(d)),d,dir_star2(k)));
            if hess2 > 0
                dir_star2 = dir_star2(k);
                break
            end
        end
    end
    end
    if hess1 > hess2 && abs(dir_star2 + dir_star1) < .01
        dir_star = dir_star1;
        hess_dist = hess1;
    elseif hess1 < hess2 && abs(dir_star2 + dir_star1) < .01
        dir_star = dir_star2;
        hess_dist = dir_star2;
    else
        if abs(dir_star1) > abs(dir_star2)
            dir_star = dir_star2;
        else
            dir_star = dir_star1;
        end
        hess_dist = dir_star2;
    end

end
dir_star
% dir_star = fminsearch(dist_dir,0)


x(i+1) = x(i)+dir_star*slope(i);
y(i+1) = y(i)+dir_star*slope(i+1);
clear dir_star1_collect dir_star2_collect
end

disp('t_1 is:')
disp(x(n+1))
disp('t_2 is:')
disp(y(n+1))


% So the answer I have found for t1 = -0.7816 + 2*k*pi, k is an integer.
% The answer for t2 = -1.8157 +2*r*pi, r is an integer.
% This follows to be true to the periodic property in trigonometry.
end
```