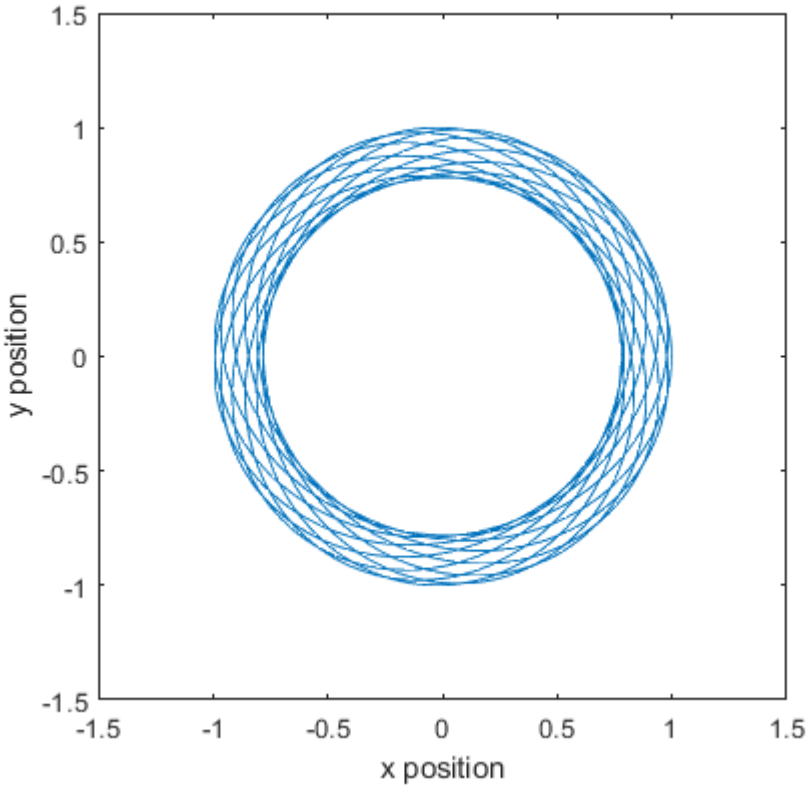
What periodic trajectories exist for a particle subject to a central force $F(r) = -r^{-\frac{1}{2}}$?



Brian Lui (bl569)
Cornell MAE
January-May 2019

Contents

1 Abstract

Unique periodic trajectories that are neither circular nor linear are found for a particle subject to a central force $F = -r^{-\frac{1}{2}}$. For this particular central force, the only periodic solutions that appear to impossible are those with 2 lobes, 3 lobes, 4 lobes, 6 lobes, 10 lobes and 14 lobes. There are some lobed solutions with more than one possible solution; the lowest lobed solution in which this is the case is the 13-lobed solution. We can also do the same analysis to look at possible trajectories for other central forces, specifically $F = -r^{-2}$, $F = -r$, and $F = -r^{-\frac{3}{2}}$, and notice that the possible trajectories differ for those central forces. Periodic trajectories were found using trajectory optimization, primarily through single shooting or trapezoidal collocation.

2 Introduction

2.1 Central Force

A central force is a force that acts only in the radial direction between a particle and the origin. The magnitude of this force is only a function of the distance the particle is from the origin. It can be written in the form $\mathbf{F} = F(r)\hat{\mathbf{r}}$, in which $F(r)$ is the magnitude of the central force and $\hat{\mathbf{r}}$ is the unit vector from the origin to the particle. The free body diagram for a particle subjected to a central force is given by Figure 1. The acceleration of the particle in both Cartesian and polar coordinates is given by

$$\begin{aligned} \mathbf{a} &= \ddot{x}\hat{\mathbf{e_x}} + \ddot{y}\hat{\mathbf{e_y}} \\ \mathbf{a} &= (\ddot{r} - r\dot{\theta}^2)\hat{\mathbf{e_r}} + (r\ddot{\theta} + 2\dot{r}\dot{\theta})\hat{\mathbf{e}_\theta} \end{aligned} \tag{1}$$

To find the equations of motion in Cartesian coordinates, we can take a linear momentum balance and dot with $\hat{\mathbf{e}}_\mathbf{x}$ and $\hat{\mathbf{e}}_\mathbf{y}$. For simplicity, assume that the mass of the particle is equal to 1. This results in the following equations of motion:

$$\begin{aligned} \ddot{x} &= F(r)\frac{x}{r} \\ \ddot{y} &= F(r)\frac{y}{r} \end{aligned} \tag{2}$$

in which $r = \sqrt{x^2 + y^2}$ is the distance the particle is away from the origin.

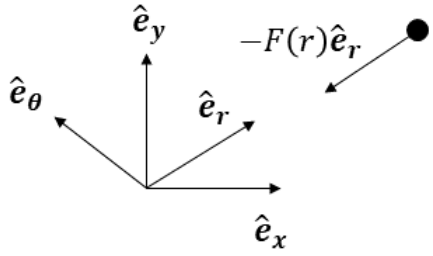


Figure 1: Free body diagram for a particle subjected to only a central force $\mathbf{F} = -F(r)\hat{\mathbf{e_r}}$

There are many examples of central forces, the most well known of which are $F(r) = -\frac{k}{r^2}$ and $F(r) = -kr$, in which k is a constant and r is the distance from the object to the origin. An example of a central force of the form $F(r) = -\frac{k}{r^2}$ would be the gravitational force $F(r) = \frac{Gm_1m_2}{r^2}$, in which G is the gravitational constant, and m_1 and m_2 are the masses of the two particles. This could be thought of as a central force by taking one of the masses as the origin for the second mass; the resulting force on the second mass is always directed towards the first mass. An example of a central force $F(r) = -kr$ would be the restoring force in a spring, in which k is now the spring constant. The origin in this case would be the other end of the spring.

The trajectories of particles subject to a central force of the form $F(r) = -\frac{k}{r^2}$ or $F(r) = -kr$ will always have an elliptical or linear orbit. This was proved by Joseph Bertrand in 1873, and this assertion is known as Bertrand's theorem. [1]

However, particles subjected to central forces not of the above form may have some more unique trajectories. Looking at the central force $F(r) = -r^{-\frac{1}{2}}$, we can plot the trajectories for several different initial conditions. Although linear and circular solutions do exist, as can be seen in Figures 2 and 3, other more interesting solutions can also exist, as can be seen in Figures 4 and 5. In fact, all trajectories that are not linear or elliptical will look like Figures 4 and 5.

Figure 4 suggests that a periodic orbit with 5 lobes exists; Figure 5 suggests that a periodic orbit with 7 lobes exist. If such solutions exist, which n-lobed periodic trajectories are possible? For a desired solution containing n lobes, how many unique periodic trajectories have this number of lobes?

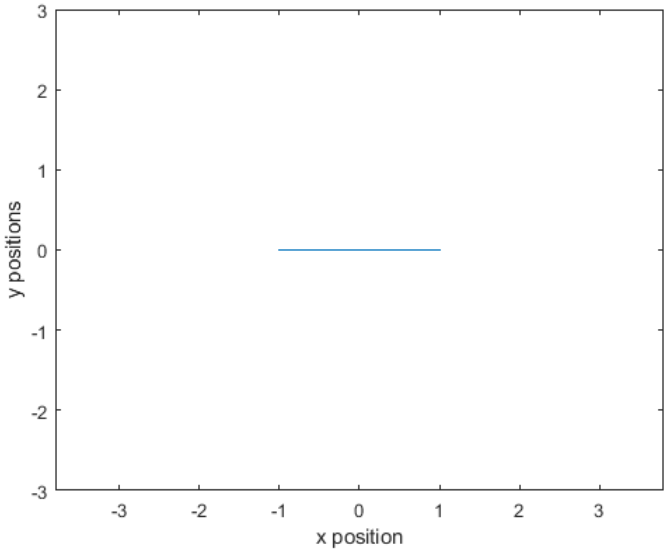


Figure 2: Initial condition of [x y $\dot{x}$ $\dot{y}$] = [1 0 0 0] over a period of 10 seconds using ode45.

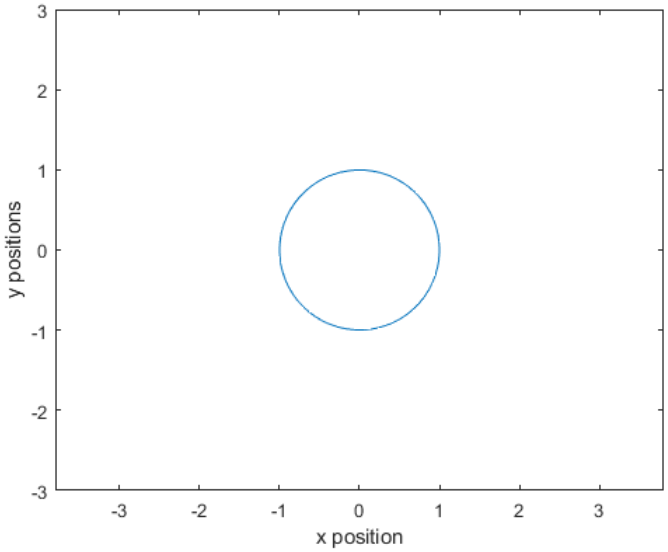


Figure 3: Initial condition of [x y $\dot{x}$ $\dot{y}$] = [1 0 0 1] over a period of 10 seconds using ode45.

2.2 Trajectory Optimization

Trajectory optimization is a method that can be used to find such trajectories. If the solution of a problem is given by the global minimum to some cost function, trajectory optimization is a way to find that global minimum. It is a subset of optimal control that involves finding the open-loop solution to the problem, which is dependent on the initial conditions of the system. This is in contrast to the closed-loop solution, which finds the global solution for the system. The open-loop solution finds the solution for one given initial and final state of the system; the closed loop solution finds the solution for all possible given initial conditions and a given final state of the system.

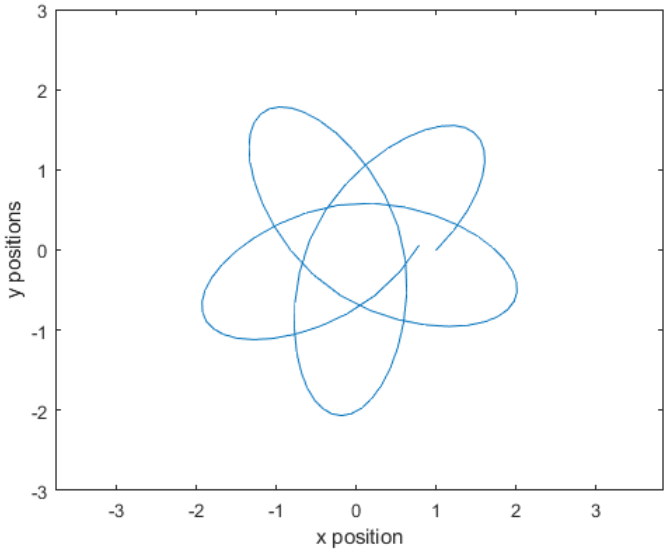


Figure 4: Initial condition of [x y $\dot{x}$ $\dot{y}$] = [1 0 1 1] over a period of 25.2 seconds using ode45.

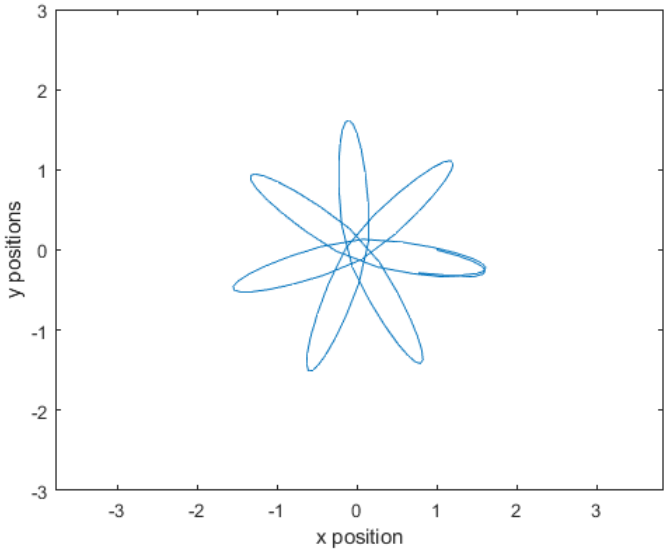


Figure 5: Initial condition of [x y $\dot{x}$ $\dot{y}$] = [1 0 1.03 -0.25] over a period of 30 seconds using ode45.

The open-loop solution is much easier to compute than the closed-loop solution and therefore is used for higher dimensional problems, but may run into an issue of finding a local minimum instead of the global minimum.

Given a state $\mathbf{x}(t)$ and control input $\mathbf{u}(t)$ over the timespan t = $[t_0, t_f]$, solving a trajectory optimization problem involves solving the continuous equation

$$\min_{t_0, t_F, \mathbf{x}(t), \mathbf{u}(t)} \quad J(t_0, t_F, \mathbf{x}(t_0), \mathbf{x}(t_F)) + \int_{t_0}^{t_F} w(\tau, \mathbf{x}(\tau), \mathbf{u}(\tau)) d\tau$$

when subjected to the constraints:

1.	Boundary constraints	$\mathbf{g}(t_0, t_F, \mathbf{x}(t_0), \mathbf{x}(t_F)) \leq \mathbf{0}$
2.	Bounds on initial and final time	$t_{low} \leq t_0 \leq t_F \leq t_{upp}$
3.	Bounds on initial state	$\mathbf{x}_{0,low} \leq \mathbf{x}(t_0) \leq \mathbf{x}_{0,upp}$
4.	Bounds on final state	$\mathbf{x}_{F,low} \leq \mathbf{x}(t_F) \leq \mathbf{x}_{F,upp}$
5.	Dynamics	$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t))$
6.	Path constraints	$\mathbf{h}(t, \mathbf{x}(t), \mathbf{u}(t)) \leq \mathbf{0}$
7.	Continuous bounds on state	$\mathbf{x}_{low} \leq \mathbf{x}(t) \leq \mathbf{x}_{upp}$
8.	Continuous bounds on control	$\mathbf{u}_{low} \leq \mathbf{u}(t) \leq \mathbf{u}_{upp}$

The term $J(t_0, t_F, \mathbf{x}(t_0), \mathbf{x}(t_F))$ is the boundary objective function, and the term $\int_{t_0}^{t_F} w(\tau, \mathbf{x}(\tau), \mathbf{u}(\tau)) d\tau$ is the integral objective function. The boundary objective function is a function that penalizes some value at the bounds of the trajectory; the integral objective function is a function that penalizes some value along the trajectory.

In this paper, there are a few assumptions that are made. One assumption is that the system is continuous in both time and state; there are no jumps in the state. As a result, in integral form, the system dynamics can be written as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{x}(t_{k+1}) &= \mathbf{x}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{f}(\tau, \mathbf{x}(\tau), \mathbf{u}(\tau)) d\tau \\ k &\in [0, N-1] \end{aligned} \tag{3}$$

in which N is the number of grid points when discretized. We also assume that the dynamics, cost function and constraint equations are smooth and consistent.

In order to solve the trajectory optimization problem, we have to transcribe the problem into a nonlinear program (NLP) of the form:

$$\begin{aligned} \min_{\mathbf{z}} f(\mathbf{z}) \qquad & \text{subject to:} \\ & \mathbf{g}(\mathbf{z}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{z}) = \mathbf{0} \\ & \mathbf{z}_{low} \leq \mathbf{z} \leq \mathbf{z}_{upp} \end{aligned}$$

Note that this $f(z)$ is different from the dynamics $\mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t))$.

This is done because these problems are not solvable by hand and require a computer to find a solution; these problems require that the problem be discretized. The problem can be discretized then optimized (direct methods), or optimized then discretized (indirect methods); direct methods are less accurate but easier to pose and solve.[2] In this paper, I used direct methods: this involved breaking the trajectory into N points, each representing the particle at a different timestep of the trajectory. The first point of the trajectory corresponding to the particle at t_0, the next point corresponded to the particle at t_1, and so on until t_N.

There many NLP solvers that are available, such as FMINCON, IPOPT, and SNOPT. There also exists software that solves a trajectory optimization problem directly, such as GPOPS-II; such software internally calls one of the NLPs available. In this paper, I used MATLAB's FMINCON because that was the most easily accessible NLP to me, and I had some experience using it in the past. FMINCON takes several inputs: the cost function, an initial guess, linear constraints, and nonlinear constraints. The cost function is the function that will be minimized, the initial condition is the first guess to the cost function, and the linear and nonlinear constraints are given in terms of inequality or equality constraints. The NLP will seek to minimize the cost function while meeting all of the constraint to the desired tolerance. Transcribing the problem into the NLP is one of the harder challenges of trajectory optimization, especially if there are constraints that are not easily written mathematically.

Trajectory optimization can also be classified by either collocation methods or shooting methods: the difference between the two is how the NLP handles the dynamics of the system after system has been discretized. [3] Using a collocation method, the NLP directly controls both the state of the system and the controls that influence the state. As a result, it is subject to the same tolerance as the other constraints. The most commonly used simultaneous methods are trapezoidal collocation. In a trapezoidal collocation, the dynamics and controls to the system are assumed to be linear between grid points, and the state is therefore quadratic. In other words, the system dynamics given by Equation 3 is approximated by

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \frac{h_k}{2}(\mathbf{f}_{k+1} + \mathbf{f}_k) \\ h_k &= t_{k+1} - t_k \end{aligned} \tag{4}$$

In shooting methods, the dynamics of the system is not directly controlled by the NLP. Instead, a simulation of the dynamics occurs within the NLP, such as a 4th order Runge-Kutta (RK4). One can also run ode45 within the NLP to simulate the dynamics of the system, but each iteration would be significantly slower. Shooting methods can also be broken down into single shooting and multiple shooting. The difference between these are in how many points the trajectory contains when it is discretized. For single shooting, there are only two: the initial state and the end state. For multiple shooting, there are also many intermediary states between the initial state and the end state, similar to collocation. In fact, in the limit as the number of particles N approaches ∞, the results of multiple shooting is very similar to that of collocation. For large enough N, the simulation occurs over such a small timescale that a linear approximation is very good.

3 Results and Discussion

3.1 Possible Solutions

Excluding the case in which a particle starts at the origin (this is undefined because the magnitude of the central force is ∞), all the possible trajectories is a straight line, circle, or somewhere in between. In the case of a straight line, there is no net acceleration in the $\hat{\mathbf{e}}_\theta$ direction; in the case of a circle, there is no net acceleration in the $\hat{\mathbf{e}}_\mathbf{r}$ direction. For trajectories that are not linear or circular, there is some component of the acceleration that is in both the $\hat{\mathbf{e}}_\mathbf{r}$ and $\hat{\mathbf{e}}_\theta$ direction.

For example, look at the case in which a particle starts at (x,y) = (1,0). If it starts without any initial velocity in either the $\hat{\mathbf{e}}_\mathbf{x}$ or $\hat{\mathbf{e}}_\mathbf{y}$ direction, the particle will oscillate about the origin in a linear fashion; the only acceleration is in the $\hat{\mathbf{e}}_\mathbf{x} = \hat{\mathbf{e}}_\mathbf{r}$ direction. It will also do so if the velocity in the $\hat{\mathbf{e}}_\mathbf{x}$ direction is nonzero but the velocity in the $\hat{\mathbf{e}}_\mathbf{y}$ direction is 0. On the other hand, if it moves with an initial velocity purely in the $\hat{\mathbf{e}}_\mathbf{y} = \hat{\mathbf{e}}_\theta$ direction with just the right magnitude, it will move in a circle. In order to find this velocity, we look at the acceleration given in Equation 1. The circular trajectory solution would occur when there is no change in the radial components; $\dot{r} = \ddot{r} = 0$. This simplifies the acceleration given in Equation 1 to that as follows:

$$\mathbf{a} = -r\dot{\theta}^2\hat{\mathbf{e}_\mathbf{r}} + r\ddot{\theta}\hat{\mathbf{e}_\theta} \tag{5}$$

Applying a linear momentum balance on the particle and taking the dot product in the $\hat{\mathbf{e}}_\mathbf{r}$ direction, we get the following relationship:

$$F(r) = \frac{mv^2}{r} \tag{6}$$

For the central force, $F(r) = -r^{-\frac{1}{2}}$, the velocity required for the particle to move in a circle is equal to $v_0 = r^{\frac{1}{4}}$. Therefore, for a particle that starts at (1,0), the velocity must be $v_0 = 1$; for a particle that starts at (2,0), the velocity must be $v_0 = 2^{\frac{1}{4}}$.

If the particle starts with a velocity that is nonzero in both the $\hat{\mathbf{e}}_\mathbf{x}$ and $\hat{\mathbf{e}}_\mathbf{y}$ direction, then the solution will look like one of the lobed solutions, similar to that seen in Figures 4 or 5. Looking at those solutions, the points that are the greatest distance away from the origin corresponds to the ends of each of the lobes. If the point (1,0) lies on the end of a lobe ($r_{max} = 1$), then the velocity of the particle is perpendicular to the lobe; it must be purely in the $\hat{\mathbf{e}}_\mathbf{y}$ direction. If the point (1,0) does not lie on the end of a lobe, then it is part of a larger lobed solution ($r_{max} > 1$). (1,0) would not lie at the end of a lobe if there is a velocity in both the $\hat{\mathbf{e}}_\mathbf{x}$ and $\hat{\mathbf{e}}_\mathbf{y}$ direction at (1,0), or if the velocity in the $\hat{\mathbf{e}}_\mathbf{x}$ direction is 0 but the velocity in the $\hat{\mathbf{e}}_\mathbf{y}$ is greater than v_0, the velocity for the particle to move in a circle. These can be seen in Figures 3, 6, and 7.

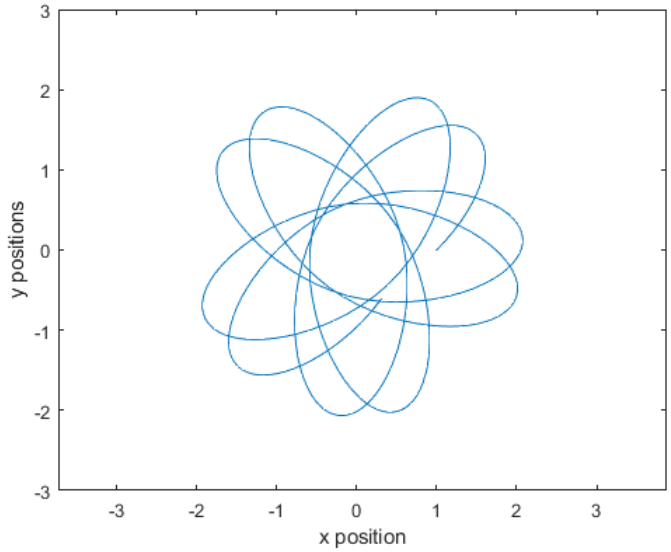


Figure 6: Initial condition of [x y $\dot{x}$ $\dot{y}$] = [1 0 1 1] over a period of 50 seconds using ode45.

In these lobed solutions, there is a certain symmetry that exists. Looking at the 5 lobed solution, there exists an infinite number of 5 lobed solutions for a given r_{max}: one can take any periodic trajectory and rotate by some angle, and that trajectory will also be a solution. There also will exist an infinite number 5 lobed solutions that are dilations of each other as you increase r_{max}. Such solutions exists because the central force doesn't differentiate between where exactly a particle is, only how far away it is from the origin; the central force always acts radially towards the origin. Also, being further away from the origin, one can scale the initial velocities accordingly to get the same trajectory shape. As a result, a particle located at (1,0) will experience the same force as that located at (0,1), albeit in a different direction. We are interested by the shape of solutions possible, so we can ignore the cases in which the point (1,0) does not lie at the end of a lobe. Therefore, all the possible shapes of periodic trajectories to a particle subjected to a central force can be found by looking at the initial conditions [x y $\dot{x}$ $\dot{y}$] = [1 0 0 v] for v $\in$ [0,1].

For there to be an n-lobed solution, the ends of the lobes must be equally spaced and be $\frac{2\pi}{n}$ rad apart. For example, for a 5 lobe solution, each of the adjacent ends of the 5 lobes must be $\frac{2\pi}{5} = 72^o$ apart. Looking at Figure 4, this appears roughly true. If the lobes were not exactly $\frac{2\pi}{n}$ apart, then the solution is quasiperiodic; the trajectory will get very close to being periodic but will never quite reach its initial starting position with the right velocity.

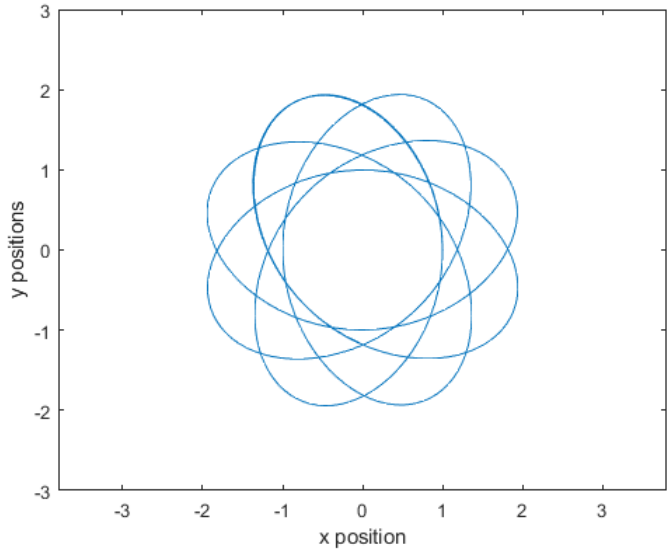


Figure 7: Initial condition of [x y $\dot{x}$ $\dot{y}$] = [1 0 0 1.5] over a period of 50 seconds using ode45.

Because we are interested at the limiting cases of the possible solutions, we look at the limiting cases in which $v \approx 0$ and $v \approx 1$. The case when v = 0.02 and v = 0.98 can be seen in Figures 8 and 9. Let's define the angle θ to be the angle from the line connecting the origin to the initial point and the line connecting the origin and the end of the second lode to appear in the trajectory. These two points are labeled with an 'x' in Figures 8 and 9. In the limit as v approaches 0 and v approaches 1, θ appears to be bounded by approximately (180^o,227.7^o). Therefore, if the ends of the first two lobes of the trajectory are separated by $\theta \in (180^o,227.7^o)$, then that a periodic trajectory will exist. These θ corresponds to the end of the lobe travelling approximately (0.5, 0.6325) of a circle.

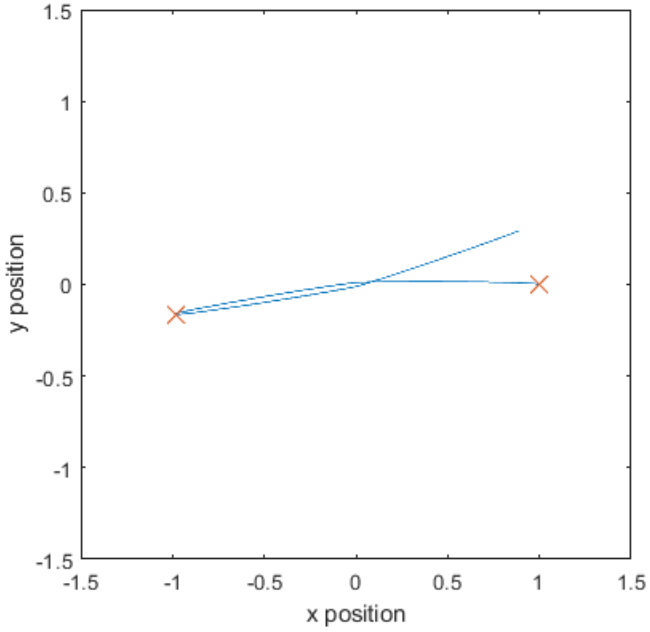


Figure 8: Initial condition of [x y $\dot{x}$ $\dot{y}$] = [1 0 1 1] over a period of 5 seconds using ode45. The 'x' marks the points on the trajectory that are within 1e-6 of r_{max}

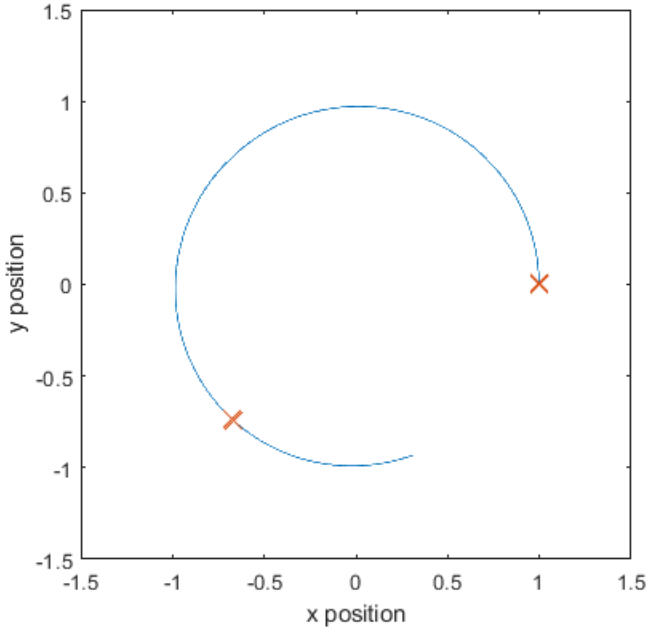


Figure 9: Initial condition of [x y $\dot{x}$ $\dot{y}$] = [1 0 0 1.5] over a period of 5 seconds using ode45. The 'x' marks the points on the trajectory that are within 1e-6 of r_{max}

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	0.5	0.33	0.25	0.2	0.17	0.143	0.13	0.11	0.1	0.091	0.083	0.077	0.071	0.067	0.063	0.059	0.056	0.053	0.05
2		1	0.67	0.5	0.4	0.33	0.29	0.25	0.22	0.2	0.18	0.167	0.154	0.143	0.133	0.125	0.118	0.111	0.105	0.1
3			1	0.75	0.6	0.5	0.43	0.375	0.33	0.3	0.27	0.25	0.23	0.21	0.19	0.18	0.18	0.17	0.16	0.15
4				1	0.8	0.67	0.57	0.5	0.44	0.4	0.36	0.33	0.31	0.29	0.27	0.25	0.24	0.22	0.21	0.2
5					1	0.83	0.71	0.625	0.55	0.5	0.45	0.42	0.38	0.36	0.33	0.31	0.26	0.28	0.26	0.25
6						1	0.88	0.75	0.66	0.6	0.545	0.5	0.46	0.43	0.4	0.375	0.35	0.33	0.32	0.3
7							1	0.875	0.77	0.7	0.636	0.583	0.538	0.5	0.467	0.43	0.41	0.39	0.37	0.35
8								1	0.88	0.8	0.73	0.67	0.615	0.57	0.53	0.5	0.47	0.44	0.42	0.4
9									1	0.9	0.81	0.75	0.69	0.643	0.6	0.563	0.529	0.5	0.47	0.45
10										1	0.91	0.83	0.77	0.71	0.67	0.625	0.589	0.56	0.523	0.5
11											1	0.92	0.87	0.79	0.73	0.69	0.65	0.61	0.579	0.55
12												1	0.93	0.87	0.8	0.75	0.71	0.67	0.631	0.6
13													1	0.93	0.88	0.81	0.76	0.72	0.68	0.65
14														1	0.94	0.88	0.82	0.78	0.74	0.7
15															1	0.94	0.88	0.83	0.79	0.75
16																1	0.94	0.89	0.84	0.8
17																	1	0.94	0.89	0.85
18																		1	0.94	0.9
19																			1	0.95
20																				1

Table 1: Each column is the maximum lobe number, each row is the specific lobe if that solution existed. The value of each is is the row number divided by the column number. The green shows a possible good trajectory, the yellow shows when the solution would be linear, and the orange shows repeated solutions (multiple of another, and therefore not allowed).

Table 1 shows which unique solutions are possible for a desired trajectory with n number of lobes, as shown in green. For a 5 lobe solution, there exist 1 possible solution; for a 7 lobe solution, there exist 1 possible solutions; for a 13 lobe solution, there exists 2 possible solution, and so on. The 1 lobe solution also technically does exist: it is the circular trajectory. Looking at the table, there is a correlation between the number of lobes and the number of different solutions to get to that trajectory: as one increases the number of lobes, the greater the number of different solutions that can exist for that number of lobes. As one increases the number of lobes, there is a greater chance of one of the lobes falling in the good region between travelling (0.5,0.06325) of the circle.

It is interesting to note for which cases there exist no solutions: up until 20 lobes, there does not exist a trajectory for 2 lobes, 3 lobes, 4 lobes, 6 lobes, 10 lobes, and 14 lobes. Are these the only lobes in which no solutions exist, or do the cases of no solution continue to exist as the number of lobes increase? In order to find out, one can use the same argument to look at higher number of lobes. I did this for n up to 3000. I found that the only cases in which no solutions existed were still the same: 2, 3, 4, 6, 10 and 14. This is not a rigid proof, but it does suggest that those are the only cases for which no solutions exist for the central force $F = -r^{\frac{1}{2}}$.

We can also apply the same logic and reasoning to look at other central forces. Two obvious choices are $F = -r^{-2}$ and $F = -r$ because they occur so often in nature. The bounds for the central force $F = -r^{-2}$ is $(360^o, 360^o)$, and the bounds for the central force $F = -r$ is $(180^o, 180^o)$. These conditions can be seen in Figures 10-13. This is as expected, and is the reason that only elliptical or linear solutions exist for those central forces. We can also apply it for another central force, such as $F = -r^{-\frac{3}{2}}$. In this case, the bounds for the central force is approximately $(245^o, 343^o)$, and can be seen in Figure 14 and Figure 15. These bounds are different than that for the central force $F = -r^{-\frac{1}{2}}$, and therefore the types of possible solutions should be different. The actual trajectories for solutions with the same number of lobes should also be different because the central force attracts the particle towards the origin differently at the same (x,y) position.

We can do this repeatedly for many central forces and generate Figure 16. There is a slightly rough region in the figure; this was because of how the data was manually collected. These angles are also not exact- to be exact, you must take the limit as the velocity approaches 0 and 1. These angles used to find the bounds are v = 1e-3 to get the lower bound and v = 0.9999 to get the upper bound. As a result, Figure 16 should primarily be analyzed qualitatively and be the starting point when looking for trajectories.

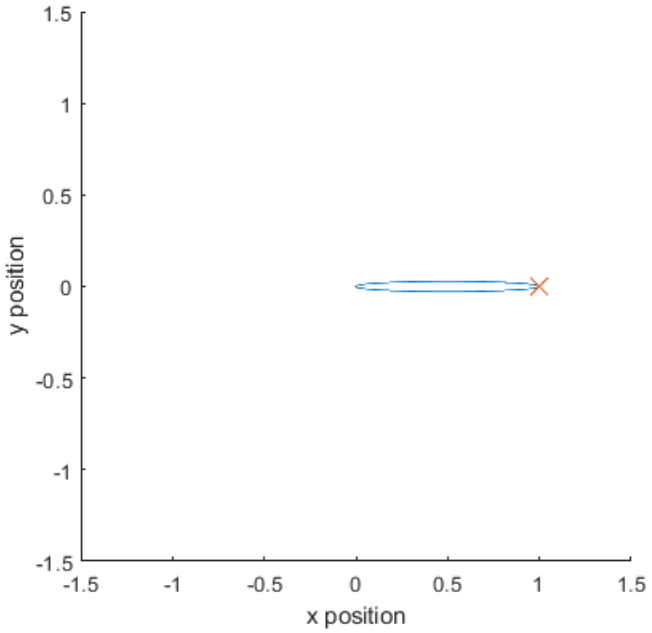


Figure 10: The smallest angle that the second lobe for a particle subject to a central force $F = -r^{-2}$.

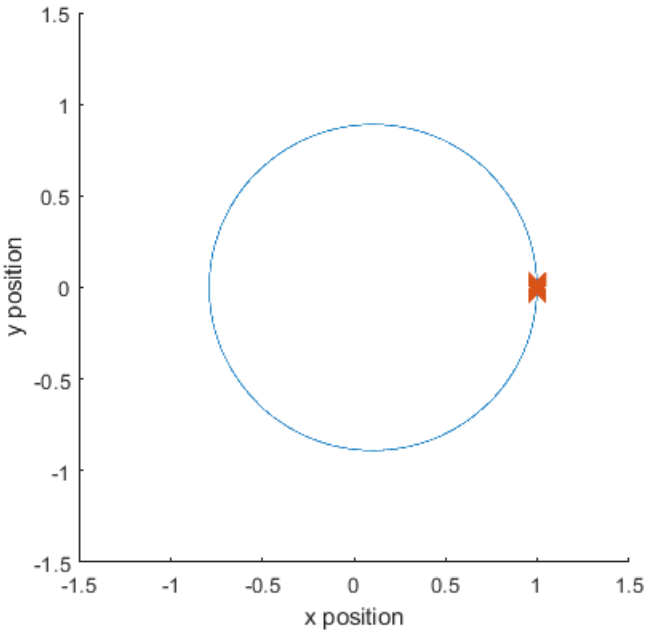


Figure 11: The largest angle that the second lobe for a particle subject to a central force $F = -r^{-2}$.

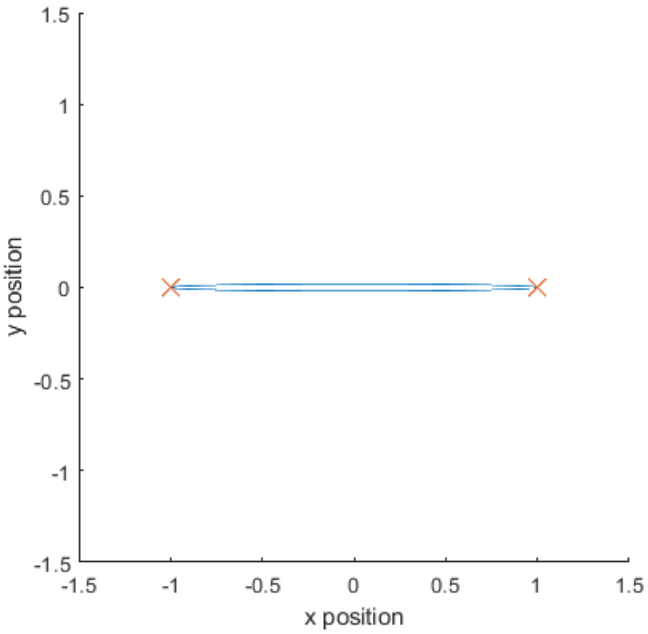


Figure 12: The smallest angle that the second lobe for a particle subject to a central force $F = -r$.

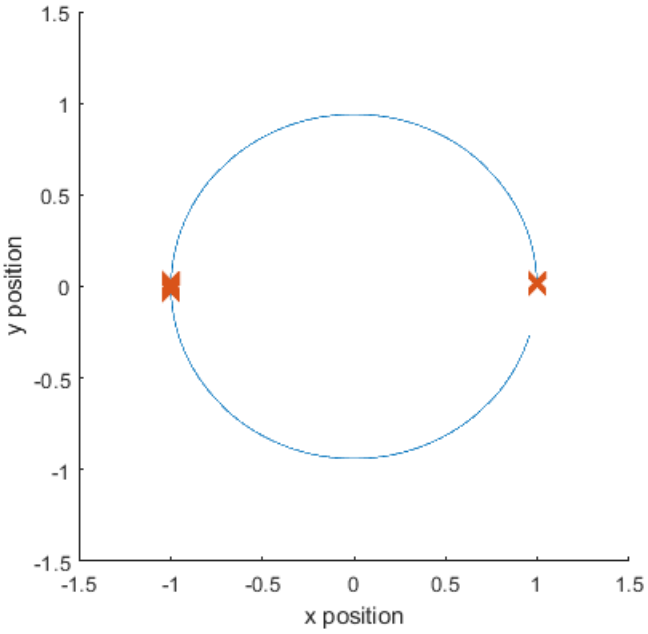


Figure 13: The largest angle that the second lobe for a particle subject to a central force $F = -r$.

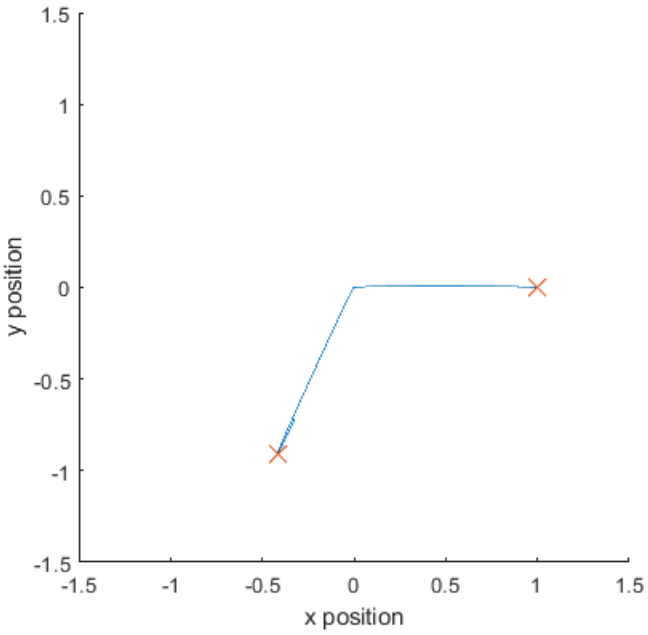


Figure 14: The smallest angle that the second lobe for a particle subject to a central force $F = -r^{-\frac{3}{2}}$.

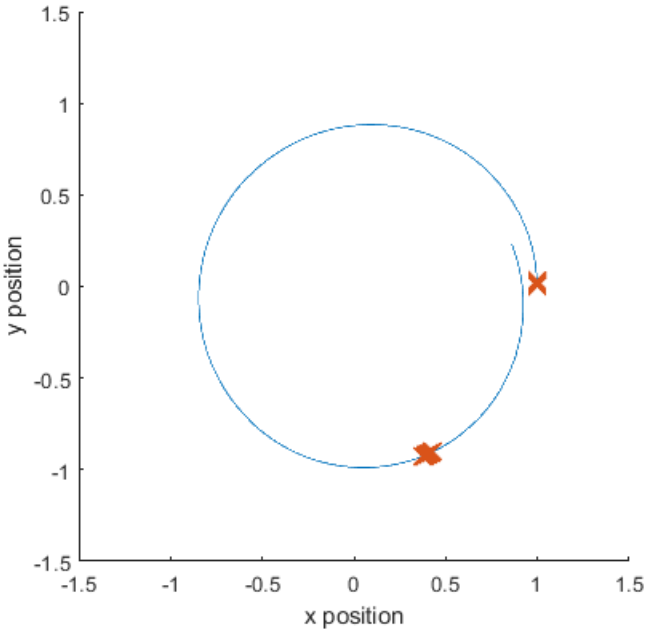


Figure 15: The largest angle that the second lobe for a particle subject to a central force $F = -r^{-\frac{3}{2}}$.

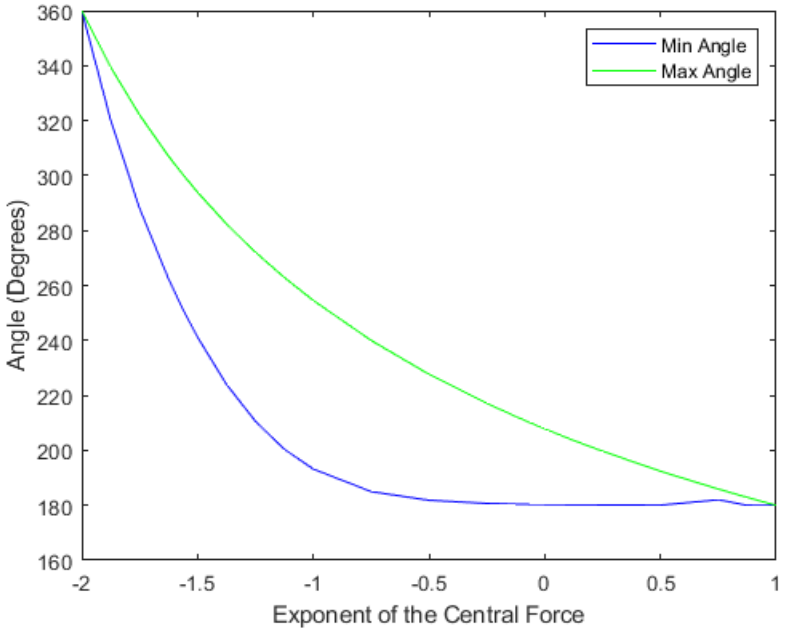


Figure 16: The power/exponent of the central force vs the bounding angles of the trajectory.

3.2 Using Trajectory Optimization to Find Desired Solutions

By blindly guessing initial conditions, trajectories that are almost periodic can be found. However, although these trajectories appear to start and end in the same spot, the trajectories are not truly periodic, as can be seen by looking at the trajectory over multiple periods. A better way to find periodic solutions involves using trajectory optimization.

3.2.1 Setting up Trajectory Optimization

Constraints help limit the region in which the NLP searches over for a solution. By limiting the region, it will help the NLP better find a solution by decreasing the number of bad leads, and can help prevent the NLP from converging at a local minimum that is not the global minimum. This is similar to searching for your missing keys in only your kitchen instead of your entire house (if the keys are in the kitchen). Sometimes you may get stuck search for your bedroom when it is not actually there. However, constraints should be applied carefully. If bad constraints are imposed on the NLP, the solution may never converge. In the analogy, it would be as if the keys were actually in the bedroom instead of the kitchen, but you never search the bedroom. Constraints should always be checked before running the NLP and if the NLP is taking a long time to converge; I often mistakenly attempted to get the NLP to solve an unsolvable problem.

One constraint is on the starting position and velocity of the particle. Because I am interested in only the shape of the trajectory and not the size or orientation of the trajectory, I bounded the starting position of the particle to be at (x,y) = (1,0). This was done for two reasons: from the previous analysis of the trajectories, I have a set of good starting conditions in which to get solutions with varying number of lobes, and by starting on the x axis, I can also put a constraint on the starting x velocity to be 0 because I know the solution I seek will not have any starting velocity in the $\hat{\mathbf{e}}_\mathbf{x}$ direction. By limiting the number of free parameters that FMINCON has control over, it should help it converge faster; it would not waste time playing around with parameters that it does not need to.

In order for the trajectory to be periodic, a constraint is imposed on the boundary of the trajectory, specifically

$$\begin{aligned} x(t=0) &= x(t=t_F) \\ y(t=0) &= y(t=t_F) \\ \dot{x}(t=0) &= \dot{x}(t=t_F) \\ \dot{y}(t=0) &= \dot{y}(t=t_F) \end{aligned} \tag{7}$$

To prevent the trajectory from converging towards a linear solution, I put the following condition as a nonlinear constraint:

$$\left|\frac{dy}{dx}(t=0) - \frac{dy}{dx}(t=t_1)\right| > \epsilon \tag{8}$$

This constraint was chosen because the magnitude of the slope of a linear trajectory is constant. By only looking at only the first two timesteps, they would also have the same magnitude. Because there isn't a way to impose an unequal constraint in FMINCON, an inequality constraint was used instead. Choosing an ϵ value of 1e-2 worked well for getting the NLP to converge to a good trajectory.

To prevent the trajectory from converging towards a circular solution, I put the following condition as a nonlinear constraint:

$$r_{max} - r_{min} > \delta \tag{9}$$

For a circular trajectory, the distance of the particle from the origin is constant. For a trajectory that contains lobes,

the particle oscillates between moving closer and further away from the origin. By constraining the absolute value of the difference to be greater than δ, then the trajectory should avoid converging towards a circle. Choosing δ based on the expected solution, such as 0.7 for the 5-lobed solution or 0.3 for the 8-lobed solution, this can greatly limit the parameter space for the NLP.

Finally, there was a constraint on the period length. Without the bound on time, the solution tends towards a trajectory in which the period length is very small. This corresponds to a trajectory in which the particle barely moved from its initial position, which if small enough, will satisfy the constraint within the desired tolerances. This is an unintended solution: although that solution does indeed satisfy all of the constraints, it doesn't give any useful information about whether the initial conditions may have produced a periodic trajectory. One could have put a bound on the time using the inequality constraint; I instead put an upper and lower bound on the time because I had a very good idea of what I thought the period should be (based on a very good initial guess), therefore further bounding the NLP and helping it converge.

The cost function is used to get the trajectory to converge towards the desired solution. Because I am seeking trajectories that are neither circular nor linear, that condition would be the most obvious choice to put within the cost function. However, I decided against using a cost function; the cost function was set to be a constant instead because it tries to impose the condition more rigidly. The other reasonable choice of a function to be placed within the cost function would be the desired lobed solution.

For each method, I attempted to get trajectories that satisfy a constraint tolerance of 1e-6. Afterward, I compare the trajectory with that from running ode45 using the initial starting position and velocity of the particle. This compares the solution from each method with a baseline so that the methods can be compared with each other. By imposing a tighter constraint tolerance, the solution found will better obey the dynamics. This assumes that the constraints are imposed in the same order of magnitude.

3.2.2 Initial condition

Whether or not the NLP will solve is usually dependent on how good the initial guess of the trajectory is. If the initial guess is very close to the actual desired solution, then the NLP is more likely to solve. Playing around with the initial conditions from shooting trajectories from [1 0 0 v], one can easily iterate and find solutions that appear to be periodic. This time, instead of looking at the velocity interval v = [0, 1], v should be a smaller interval to be able to see at higher resolution. By first looking at where the second lobe of the trajectory appears, we can compare that location with that on Table 1. By fine tuning the exact v and then adjusting the period, trajectories with the desired number of lobes can be found. Afterward, the trajectory is simulated for a much longer period in order to capture the entire periodic behavior of the particle, and the one with the best starting condition. The period was also fine tuned so that the starting and ending point of the trajectory was roughly in the same spot. Although generating this list of good trajectories requires more time, it allows the initial guess to the NLP to be very good, helping the NLP converge faster. Although not done, this process of finding good initial conditions can be automated.

3.2.3 Single Shooting

One way to solve for solutions would be to use single shooting. In this transcription method, a simulation of the dynamics is done within the NLP. The state of the solution from using this transcription method is $[x_0, y_0, \dot{x_0}, \dot{y_0}, T]$; these are the only five parameters that the NLP can vary as it attempts to satisfy the imposed constraints. The dynamics are all done by the simulation, and the resulting position and velocity is compared to see if it equals the initial starting position and velocity. If a very good initial guess is chosen, a tighter initial tolerance can be used, and fewer iterations of the NLP is required. Because the number of parameters that the NLP can change is not very large, I decided to use ode45 within the optimizer to simulate the dynamics. Although each iteration of the simulator took a much longer time, there were not many iterations required for the constraint tolerance to be below the desired value.

In order for the NLP to solve quickly the constraints of the problem should be made as tight as possible. This is especially the case if a long integrator such as ode45 is used. Specifically, this meant that bounds on time, the

linear trajectory constraint, and the circular trajectory constraint may need its values to be tuned for every desired trajectory. For example if the period for a trajectory that appears to close is around 15 seconds, then the period component of the time should be bounded close to this value, such as between [10,20] (or often even tighter, like [13,17]!). This method was good at finding the solution if the desired solution is known.

3.2.4 Multiple Shooting

Another way to solve for solutions would be to use multiple shooting. In this transcription method, the trajectory of the particle is broken down into N points, each point representing the particle at a different time in its path. The state of the solution from using this transcription method is as follows:

$$\begin{bmatrix} x_0 & x_1 & x_2 & ... & x_N \\ y_0 & y_1 & y_2 & ... & y_N \\ \dot{x}_0 & \dot{x}_1 & \dot{x}_2 & ... & \dot{x}_N \\ \dot{y}_0 & \dot{y}_1 & \dot{y}_2 & ... & \dot{y}_N \\ T & 0 & 0 & ... & 0 \end{bmatrix}$$

x_0 is the x position of the particle at time 0, x_1 is the x position of the particle at time t_1, and so on until time T. A simulation of the dynamics is done within the NLP using each of these N points. Unlike single shooting, we now have to enforce a dynamics constraint. Because the NLP is able to move each of the points x_0, x_1, ..., y_1,y_2,..., the NLP may move the points in such a way that the trajectory violates the dynamics. To enforce the dynamics, we add a constraint that the state

$$\begin{bmatrix} x_1 & x_2 & ... & x_N \\ y_1 & y_2 & ... & y_N \\ \dot{x}_1 & \dot{x}_2 & ... & \dot{x}_N \\ \dot{y}_1 & \dot{y}_2 & ... & \dot{y}_N \end{bmatrix} = \begin{bmatrix} x_0 + h\dot{x}_0 & x_1 + h\dot{x}_1 & ... & x_{N-1} + h\dot{x}_{N-1} \\ y_0 + h\dot{y}_0 & y_1 + h\dot{y}_1 & ... & y_{N-1} + h\dot{y}_{N-1} \\ \dot{x}_0 + h\ddot{x}_0 & \dot{x}_1 + h\ddot{x}_1 & ... & \dot{x}_{N-1} + h\ddot{x}_{N-1} \\ \dot{y}_0 + h\ddot{y}_0 & \dot{y}_1 + h\ddot{y}_1 & ... & \dot{y}_{N-1} + h\ddot{y}_{N-1} \end{bmatrix}$$

in which h is the timestep. Without this constraint, even if the NLP converges, the solution is meaningless; as can be seen if compared with the actual solution from running ode45.

Because of the much larger size of this parameter space, we should not use ode45 to simulate the dynamics. Doing so would mean that each iteration of the NLP will take many hours (if not days!). Instead, I used RK4 is used to simulate the dynamics. Because the timestep over which the simulation occurs is very small and there are many particles, RK4 will be very close to the actual trajectory of the particle.

3.2.5 Collocation

Like multiple shooting, the trajectory of the particle is broken down into N points, each point representing the particle at a different time in its path. As a result, the state of the solution is the same as that from multiple shooting. There also is the dynamics constraint, except the derivatives are not simulated but instead estimated using Equation 4 for trapezoidal collocation. Because the estimation of the dynamics is much simpler and the optimizer more directly controls the parameters, the optimizer is able to iterate more quickly than from shooting.

3.2.6 Comparison Between Single Shooting, Multiple Shooting and Trapezoidal Collocation

In terms of convergence time, I found that in some situations, single shooting worked best; in other situations, trapezoidal collocation worked best. This seems to be very dependent on the problem that the NLP was attempting to solve. If the initial guess was very very good, then collocation may find a solution more quickly than single shooting, to an order of magnitude of 10. This may be because the iteration time is so much faster; the optimizer

may handle constraint tolerances well and just needs to iterate enough times to deal with them. However, sometimes collocation also sometimes takes a very long time to converge to a solution. In these cases, I often stopped the optimizer before it could solve. Another potential negative of using collocation is that you may need to tighten the constraint tolerance much more in order to get the same quality of solution as that of single shooting. This is because the dynamics are inherently not satisfied by the trapezoidal collocation, but it is when using ode45 to integrate the dynamics in single shooting. Still, I would recommend starting with trapezoidal collocation to see if it can find a solution quickly, and if it cannot, switch over to single shooting. With a good enough initial guess, single shooting for this problem is able to find the solution very well. In almost every case, I found that using multiple shooting took too long to converge to make it useful. In this problem, it seemed that a smaller parameter space or faster iterations is more beneficial towards helping the optimizer converge.

The benefit of using multiple shooting or collocation over single shooting is that the initial guess of the trajectory need not be as good. Because the NLP has control of more parameters, it can vary more things to get the trajectory to meet the constraint tolerance. This however usually means that the dynamics tends to be violated more. Because an integrator such as RK4 is used, it is not going to have the same accuracy as if ode45 is used. To compensate, one can increase the number of points N, which decreases the constraint violation, and the NLP converged solution approaches that of the actual dynamics. However, the negative of using a larger number of points is the increased computation time per iteration. Also, one may use single shooting because if simulated with ode45, the trajectory when the NLP converged is more realistic in obeying the dynamics.

3.2.7 Improving convergence

FMINCON is very susceptible to the initial conditions applied to the system. When the initial condition is very good, the NLP would solve the problem much more quickly. In fact, an initial guess that is twice as close to the actual trajectory may sometimes get a solution that converges ten times as quickly.

One method to improve convergence to the desired tolerance is by running multiple iterations of FMINCON, with each iteration having a tighter constraint tolerance and a larger number of particles. This is essentially allowing the optimizer to solve a simpler problem and feeding this solution as an initial guess back into the optimizer. Extrapolating this idea, this would be very useful if one did not know the actual desired trajectory. However, there is a limit to this. If the constraint tolerance is set initially too low, it may not improve the guess. For example, if the tolerance was set to 1e-2 or 1e-1 when the optimizer is fed a solution that may have a tighter tolerance, the optimizer would output a worst nonperiodic trajectory. Be careful with the constraints that are fed into the NLP; it will try very hard to find unintended solutions.

3.3 Converged Solutions

This section contain all the solutions that I have found for a particle under a central force $F = -r^{-\frac{1}{2}}$. Specifically, this is For all the possible solutions describes in Table 1; they are all the possible solutions up until 20-lobes. These specific solutions have all been found using single shooting with a constraint toleraance of 1e-6, and RelTol = AbsTol = 1e-8. This section also contains the periodic trajectories for the 4 and 9-lobed solutions for a particle under a central force $F = -r^{-\frac{3}{2}}$

3.3.1 Central Force $F = -r^{-\frac{1}{2}}$

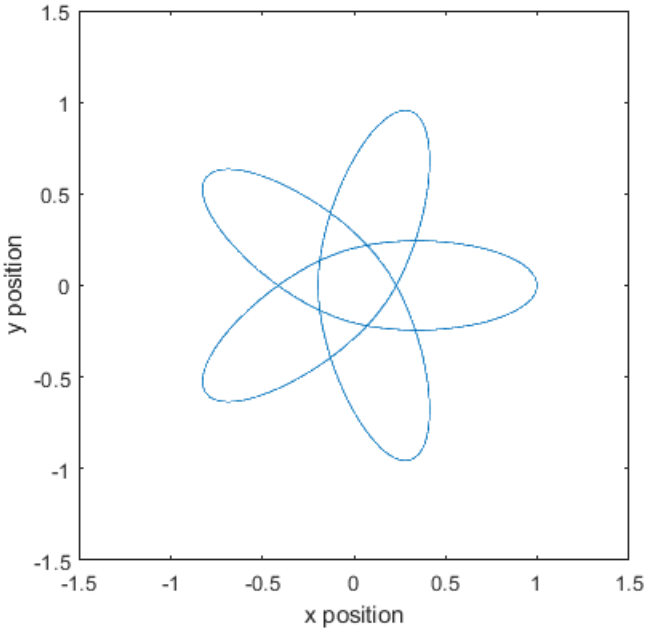


Figure 17: 5-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 14.1586 s.

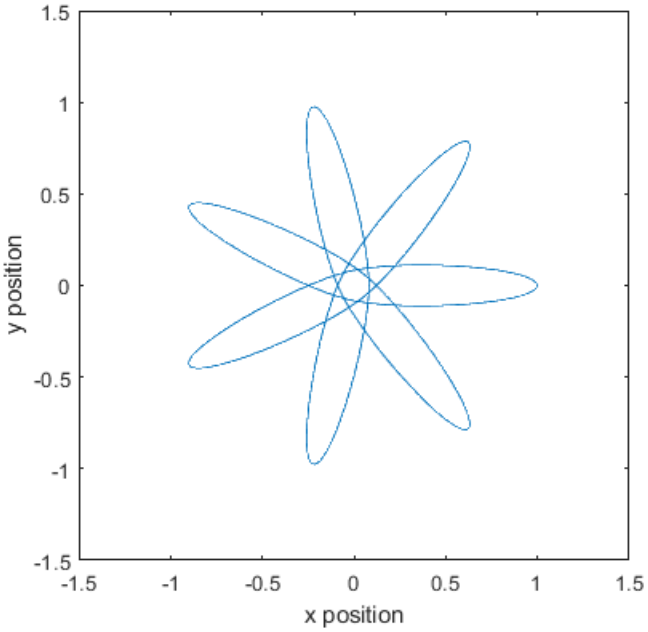


Figure 18: 7-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 18.9964 s.

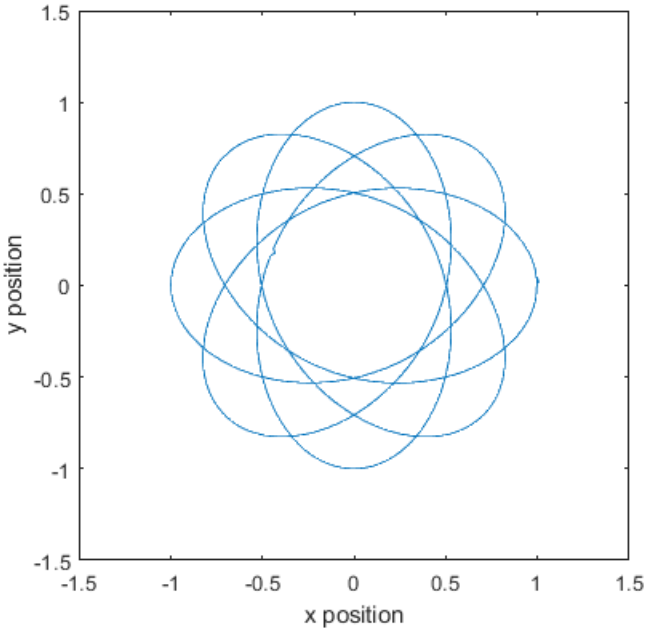


Figure 19: 8-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 25.7385 s.

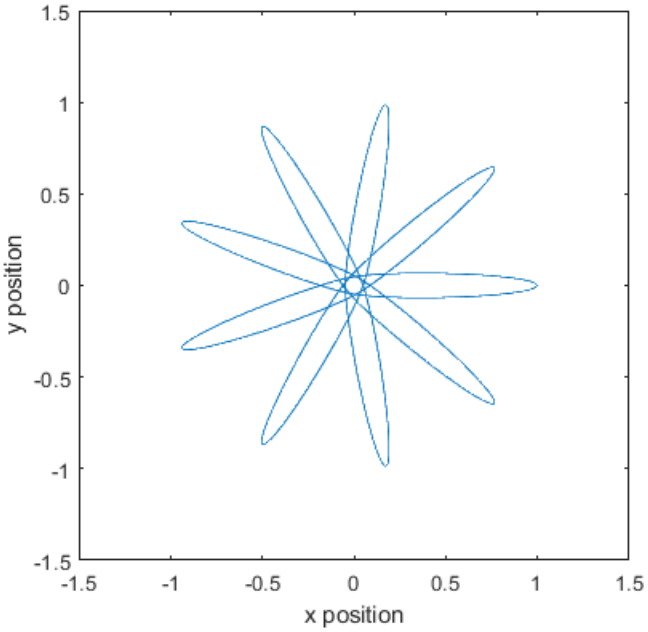


Figure 20: 9-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 24.1761 s.

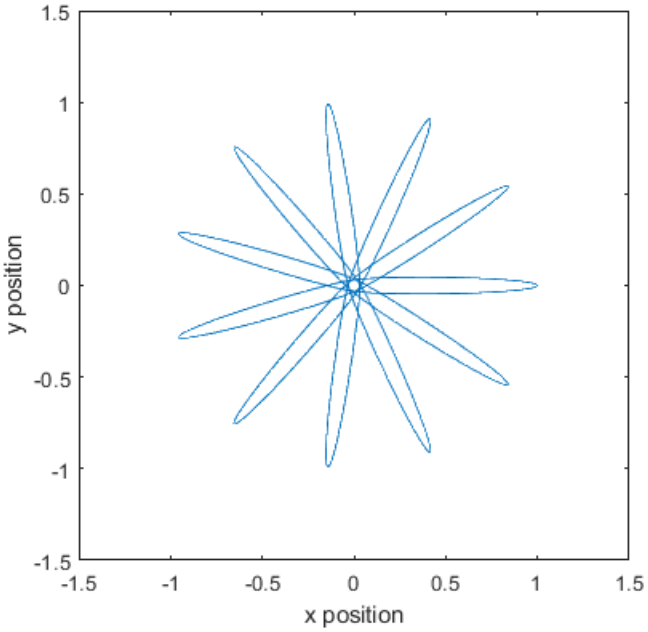


Figure 21: 11-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 29.4467 s.

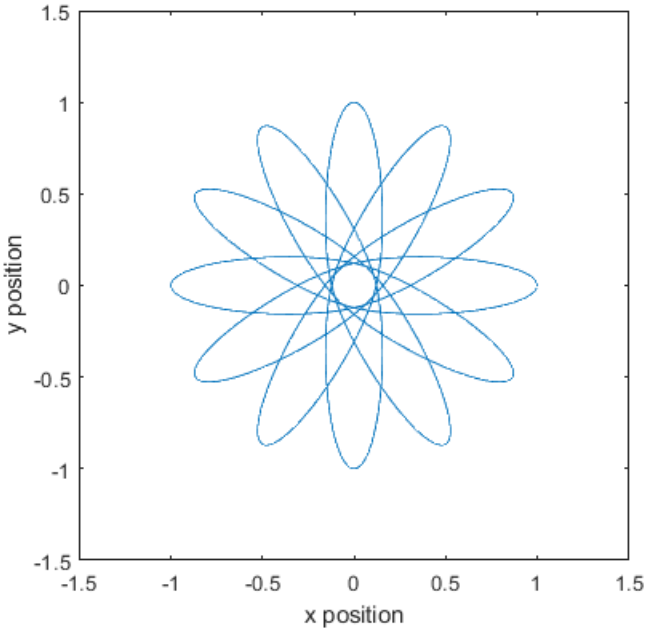


Figure 22: 12-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 32.9478 s.

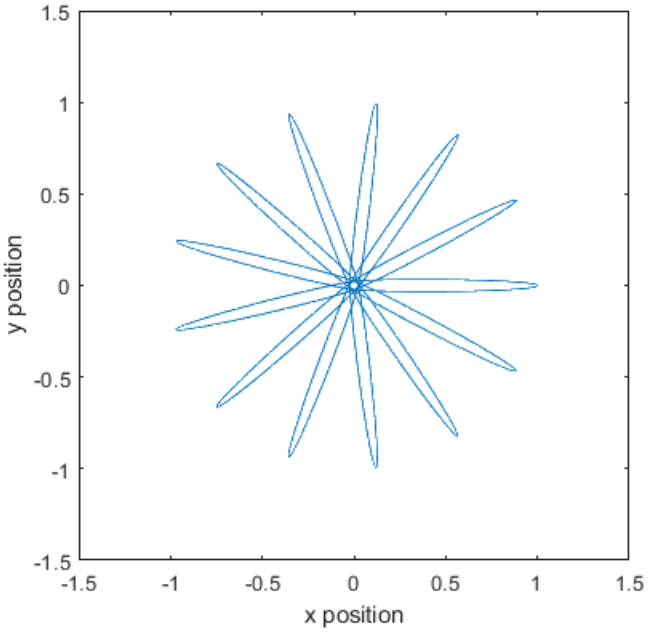


Figure 23: 1^{st} 13-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 34.7463 s.

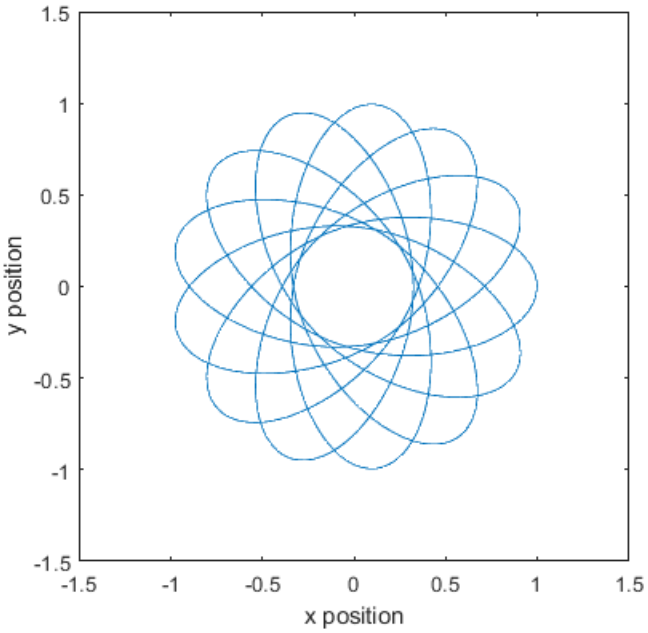


Figure 24: 2^{nd} 13-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 38.9035 s.

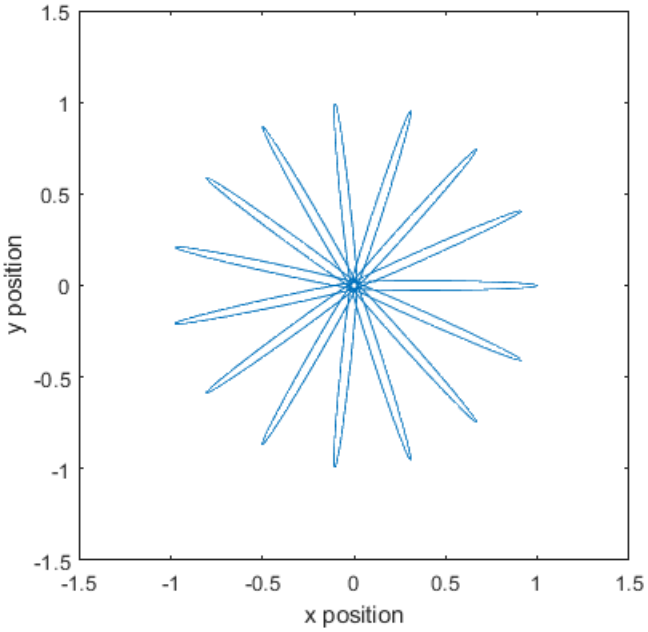


Figure 25: 15-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 40.0592 s.

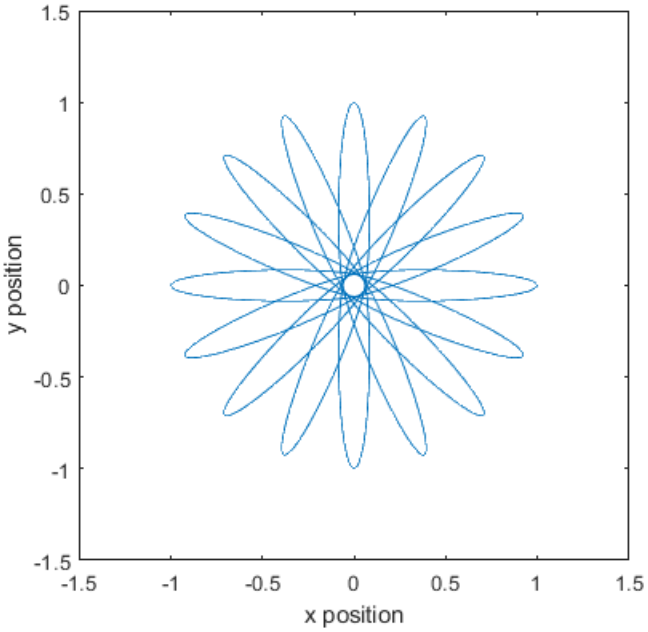


Figure 26: 16-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 43.1279 s.

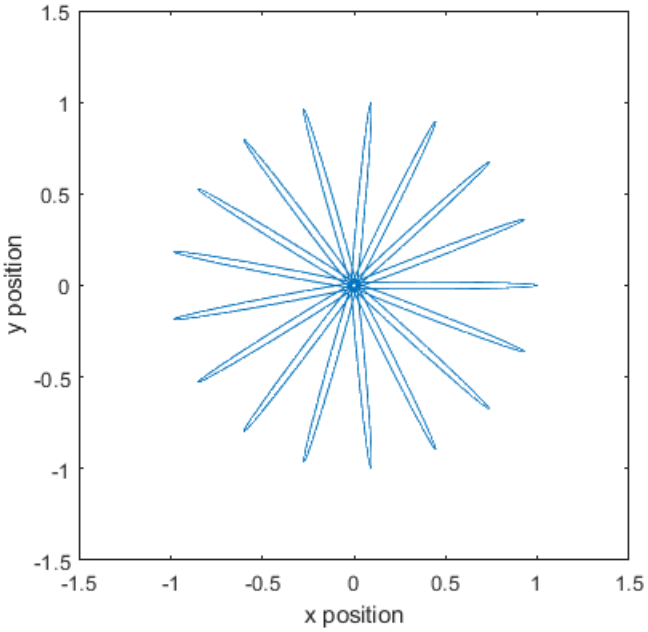


Figure 27: 1^{st} 17-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 45.3792 s.

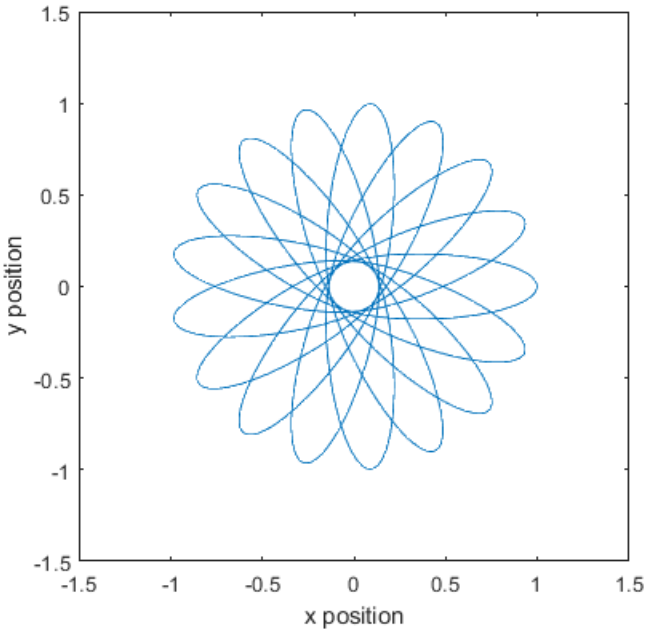


Figure 28: 2^{nd} 17-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 47.0012 s.

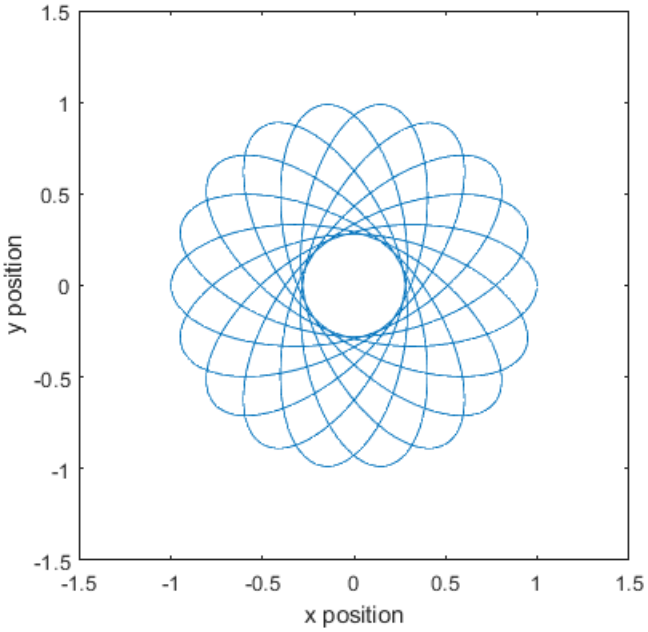


Figure 29: 18-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 52.8003 s.

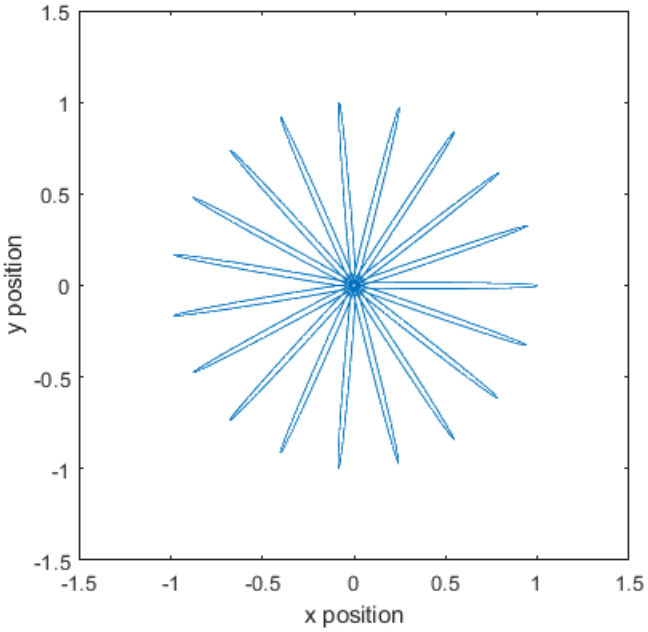


Figure 30: 1^{st} 19-lobed solution. Constraint tolerance of 1e-5, 1600 particles. T = 50.7033 s.

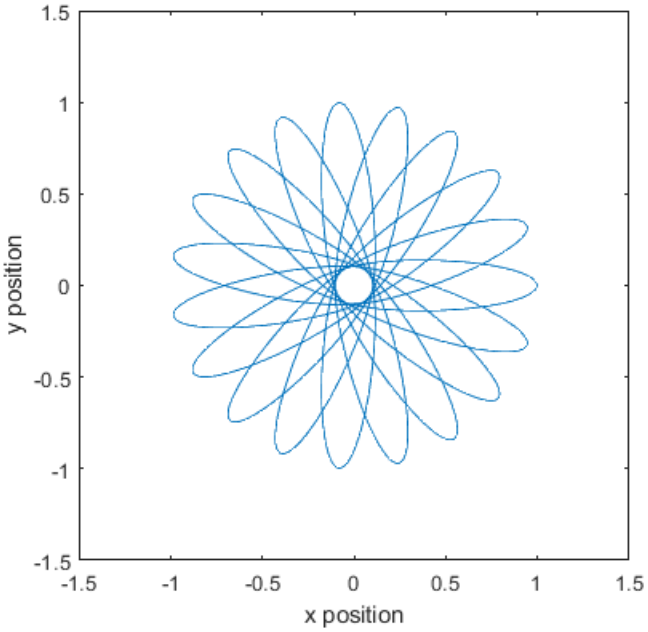


Figure 31: 2^{nd} 19-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 51.9294 s.

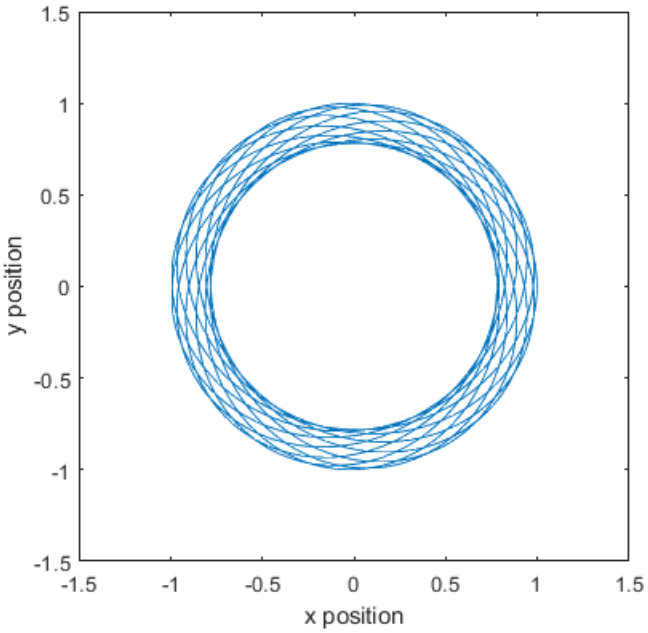


Figure 32: 3^{rd} 19-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 69.3873 s.

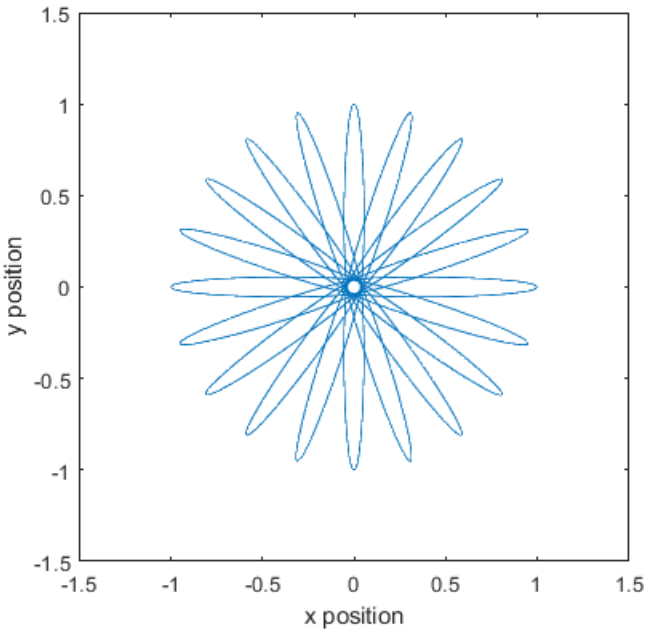


Figure 33: 20-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 53.6122 s.

3.3.2 Central Force $F = -r^{-\frac{3}{2}}$

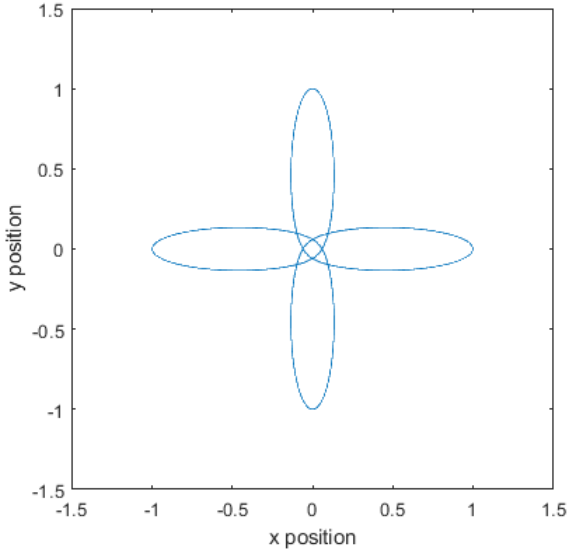


Figure 34: 4-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 9.7257 s.

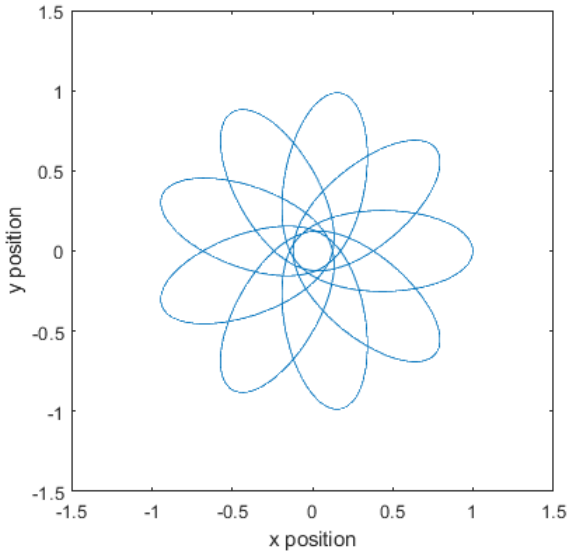


Figure 35: 9-lobed solution. Constraint tolerance of 1e-6, 1600 particles. T = 23.3174 s.

4 Conclusion

For a particle subject to a central force that is not analogous to gravity or Hooke's law, there may exist some interesting periodic orbits containing many lobes. Trajectory optimization is a very powerful tool that can be used to find periodic trajectories containing various lobes, and sometimes even multiple solutions for a given number of lobes. I found all the possible periodic trajectories up to 20 lobes. Past a certain point, there also exists multiple solutions for a given n-lobe. The analysis of the central force suggests that the only solutions that do not exist are the 2, 3, 4, 6, 10, and 14 lobed solutions. For each different force, the types of possible and impossible solutions differ.

Acknowledgements

The author would like to thank Andy Ruina for the direction, feedback, and scintillating conversations. This class definitely took a lot of time, but from an educational standpoint, the author definitely learned a lot. The author would also like to thank Rebecca Jiang, Pat Voorhees, and Anshuman Das for helping drive that conversation through all the interesting directions they took their work and asking the big brained questions. Also shoutout to Rebecca for helping keep my computers in Upson from restarting.

References

[1] Profumo, Stephano. http://scipp.ucsc.edu/ profumo/teaching/phys210_12/bertrand.pdf. 2012.

[2] Kelly, Matthew. *An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation*. SIAM Rev., 59(4), 849–904. 2017.

[3] Kelly, Matthew. *Transcription Methods for Trajectory Optimization*: *A beginners tutorial*. eprint: arXiv:1707.00284. July 2017.