Multi-Agent Path Optimization using Calculus of Variations

Course Project
MATH 146 — Methods of Applied Mathematics

Author: Robert Lee and Vedant Yogishwar
Instructor: Dr. Shiba Biswal

March 20, 2024

Contents

1   List of Symbols

N number of robots
M number of circular obstacles
$\mathbf{x}_i$ initial position of the i-th robot
$\mathbf{y}_i$ target position of the i-th robot
$\mathbf{p}_i$ path of the i-th robot
r radius or repulsive distance of robots
$\mathbf{c}_j$ center of the j-th circular obstacle
R radius of circular obstacles
F objective function
L Lagrangian function
g barrier function

2   Introduction

Given N robots and M circular obstacles, the purpose of this investigation is to model optimal trajectories from initial positions to target positions given the constraints of the model, circular obstacles, and other robots within the domain attempting to achieve their optimality simultaneously without collision. As a evaluative tool, recursive tools like the secant method will be relevant functions that are subjected to the initial derivations of optimising conditions.

3   Theory

3.1   Problem Definition

Let $\mathbf{p}_i \in C^1([0,1])^2$ be the vector valued function corresponding to the path of the i-th robot. Then we have that $\mathbf{p}_i(0) = \mathbf{x}_i$ and $\mathbf{p}_i(1) = \mathbf{y}_i$. The length of the path is defined as

$$l_i = \int_0^1 \left\| \mathbf{p}_i'(t) \right\|^2 dt. \tag{1}$$

3.2   Objective Function

We wish to minimize the total length of all robot paths. Define F as the total path length

$$F(P) = \int_0^1 \sum_{i=1}^N \left\| \mathbf{p}_i'(t) \right\|^2 dt, \tag{2}$$

where $P = [\mathbf{p}_1, \ldots, \mathbf{p}_N]^\intercal$ is the path ensemble.

We will modify the objective function in order to represent the no-collision and obstacle constraints. As an example, consider the obstacle constraint. In order to check whether or not a robot collides with an obstacle, a function g can be defined such that it evaluates to infinity if the distance between the robot and the center of the obstacle is less than R and evaluates to 0 when not. However, g would be discontinuous and forbid the use of calculus of variations techniques.

A barrier function $g_{d,\mu}$ is a continuous function that goes to infinity as it approaches a "barrier" value d. A parameter μ can be varied in order for g to approach the discontinuous form. In this

case, we wish to devise a function that goes to infinity when approached from above to act as a repulsor. In this project, we will consider the inverse function barrier function:

$$g_{d,\mu}(x) = \frac{1}{\mu(x-d)}, \quad g'_{d,\mu}(x) = \frac{-1}{\mu(x-d)^2}. \tag{3}$$

Other barrier functions such as the log barrier could also be used.

Using this barrier function we can rewrite F as

$$F(P) = \int_0^1 \left[\sum_{i=1}^N \|\mathbf{p}'_i(t)\|^2 + \sum_{i,j=1;i\neq j}^N g_{r,\mu_1}\left(\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\|^2\right) + \sum_{i=1}^N \sum_{j=1}^M g_{R,\mu_2}\left(\|\mathbf{p}_i(t) - \mathbf{c}_j\|^2\right) \right] dt, \tag{4}$$

and the optimization problem becomes an unconstrained optimization problem.

3.3 First-Order Necessary Conditions

To derive first-order necessary condition, we first identify the Lagrangian L,

$$L(Y, Z) = \sum_{i=1}^N \|\mathbf{z}_i\|^2 + \sum_{i,j=1;i\neq j}^N g_{r,\mu_1}\left(\|\mathbf{y}_i - \mathbf{y}_j\|^2\right) + \sum_{i=1}^N \sum_{j=1}^M g_{R,\mu_2}\left(\|\mathbf{y}_i - \mathbf{c}_j\|^2\right). \tag{5}$$

Applying the Euler–Lagrange equation w.r.t. the x-component of the i-th robot,

$$\frac{d}{dt}\left(\mathbf{p}'_{i,x}\right) = \sum_{j\neq i} g'_{r,\mu_1}\left(\|\mathbf{p}_i - \mathbf{p}_j\|^2\right)(\mathbf{p}_{i,x} - \mathbf{p}_{j,x}) + \sum_{j=1}^M g'_{R,\mu_2}\left(\|\mathbf{p}_i - \mathbf{c}_j\|^2\right)(\mathbf{p}_{i,x} - \mathbf{c}_{j,x}). \tag{6}$$

An analogous equation can be derived w.r.t. its y-component by replacing x with y.

3.4 Shooting Method

The Euler-Lagrange equation derived in Equation (6) provides a system of differential equations that govern the motion of each robot's path. In order to recover the optimal trajectory, we must solve the boundary-value problem. One method to do so is known as the shooting method, where we guess the unknown initial conditions of the differential equation, propagate it and compare the final value to the boundary condition, and repeat until our initial condition results in the appropriate boundary value. This can be reformulated as a root-finding problem.

3.5 The Secant Method

In simple terms, the secant method is an iterative root-finding method that—through initial guesses—formulates a sequence of approximations layered onto each other. This method relays the chosen initial guesses through a function (see Equation (7)) which allows the algorithm to predict the next iterative term. This term is then subjected to the same equation, formulating a recurrence relation until the desired level of accuracy is achieved. In the simple case of two initial conditions x_0 and x_1, the next iterative term in this model is given by

$$x_2 = x_1 - \frac{f(x_1) \cdot (x_1 - x_0)}{f(x_1) - f(x_0)}. \tag{7}$$
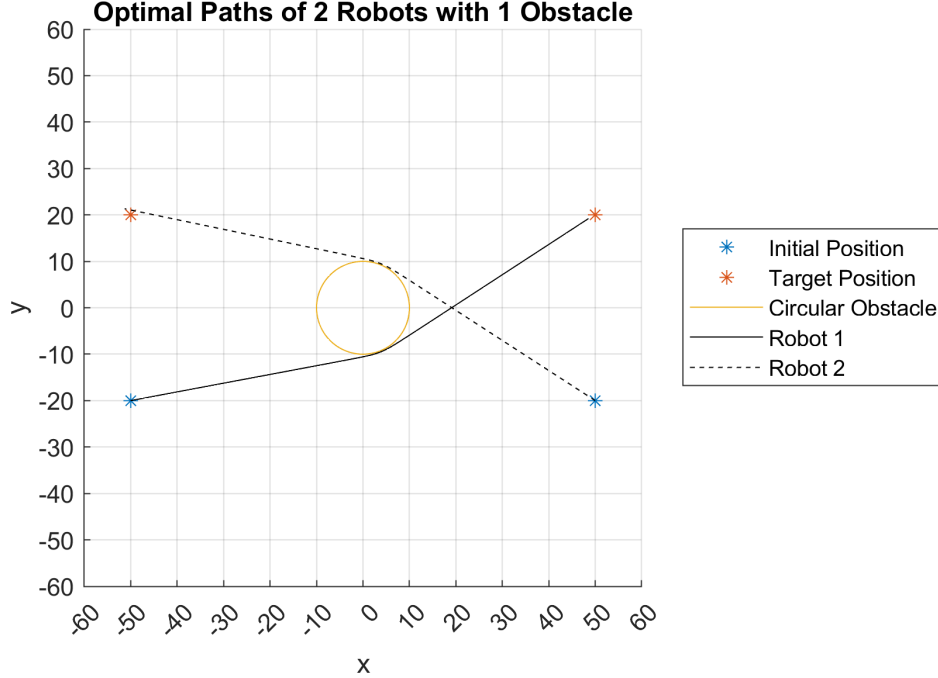
Figure 1: Optimal trajectories for two robots and one obstacles placed in an X shape. The trajectories were found by a manual shooting scheme.

4   Cases

4.1   Validating Obstacle Avoidance

In order to validate the transformation of the constrained optimization problem into an unconstrained one, we consider the simplest possible case that exhibits all phenomena in the system: two robots and one obstacle. It has the parameters

$$\mathbf{x}_1 = (-50, -20),\ \mathbf{y}_1 = (50, 20),\ \mathbf{x}_2 = (50, -20),\ \mathbf{y}_2 = (-50, 20),\ \mathbf{c}_1 = (0, 0),\ \text{and } R = 10.$$

The initial and target positions of the two robots are placed such that if they travel in a straight line, they will cross each other at the origin and create the shape of an X. At this intersection, a circular obstacle is placed.

The soft-penalty tuning parameters are set to $\mu_1 = 10^{-3}$, $r = 0$, and $\mu_2 = 5 \times 10^{-4}$. We set a small robot radius and make the effect of the obstacle double that of the robots themselves. This is because the obstacle is the dominant force in the scenario.

A manual "guess-and-check" shooting scheme is used. The initial velocity is set to aim tangent below and above for robot 1 and 2, respectively, with a magnitude of $\sqrt{100^2 + 40^2} \approx 107.7$—which is the distance both robots must travel if they move in a straight line. This is further tuned to produce the trajectory in Figure 1.

A shooting scheme algorithm that used the secant method was attempted, but convergence did not occur due to the sensitivity of the barrier function used. A small change in initial condition could result in the trajectory hitting the obstacle, which would cause `ode45` to fail and terminate early—before reaching time $t = 1$.
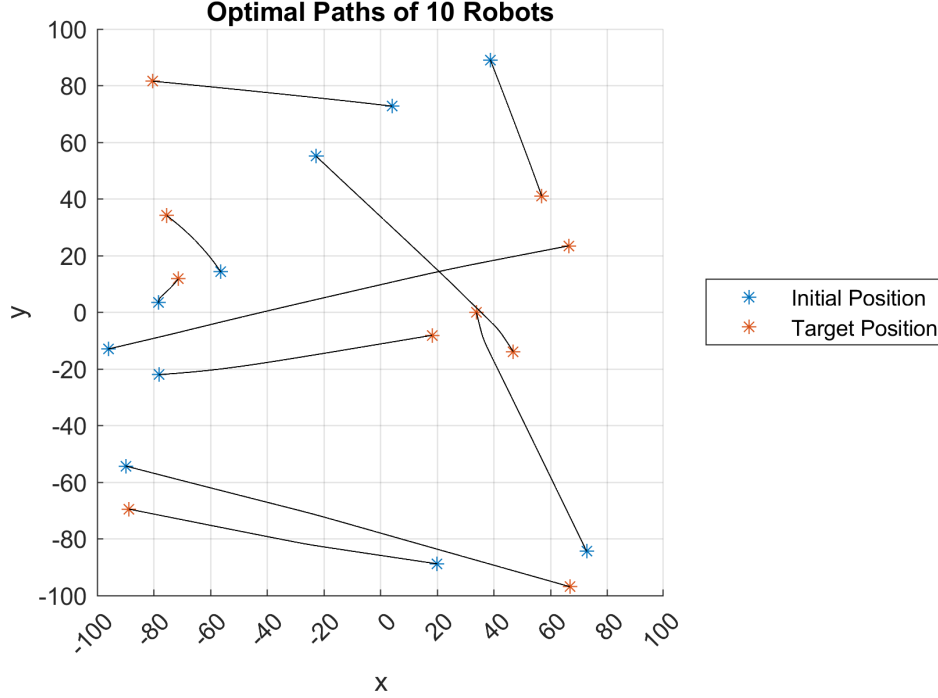
Figure 2: Optimal trajectories for 10 robots and no obstacles placed uniformly at random. The trajectories were found using a shooting algorithm with the secant method.

4.2   More Robots

Due to the sensitivity of obstacles, we decided to investigate the efficacy of the secant method with only robot repulsion. 10 robots were used in the first case and their initial and target positions were uniformly and randomly placed within the cell $[-100, 100] \times [-100, 100]$.

As the secant method must be seeded by two initial guesses, the following scheme was used. For the first initial guess, a velocity direction was chosen, for each robot, uniformly at random from 0 to 2π. The magnitude of its initial velocity was set to the distance between its initial position and target position. The second initial guess perturbed each robot's velocity direction by up to 1 degree. This was chosen as it would allow for a relatively smooth approximation of the derivative, as opposed to two diametric initial velocity guesses.

50 iterations were used to find the optimal trajectory shown in Figure 2. We can see the repulsive effect between robots in the curved trajectory of the robot moving from $\sim (-60, 20)$ to $\sim (-80, 40)$, which is caused by the influence of the robot southwest of it. Figure 3 shows a zoomed in section of the domain where two paths cross and turn to reach the target position.

The convergence of the secant method can be seen in Figure 4. A high rate of convergence can been seen from iteration 0 to 10–15, from where it slows down slightly. Great convergence can be seen, with an average target position error of $\sim 10^{-6}$ for each robot. The distribution of target position errors is shown in Figure 5 with all trajectories ending less than 10^{-5} from their target position.
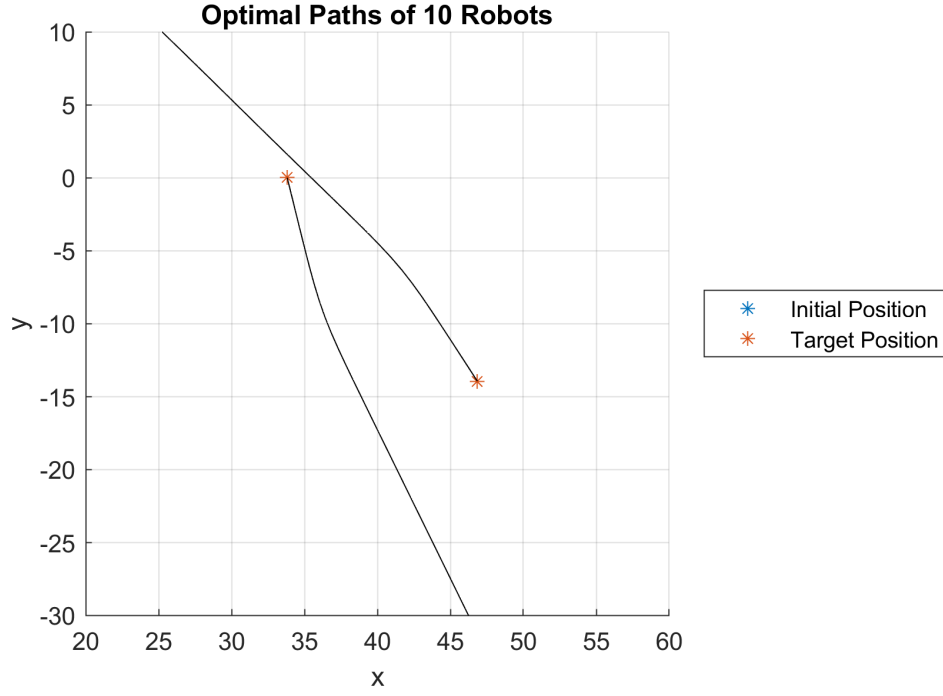
Figure 3: The region $[20, 60] \times [-30, 10]$ in Figure 2. The curvature of the path can clearly be seen from two target positions.
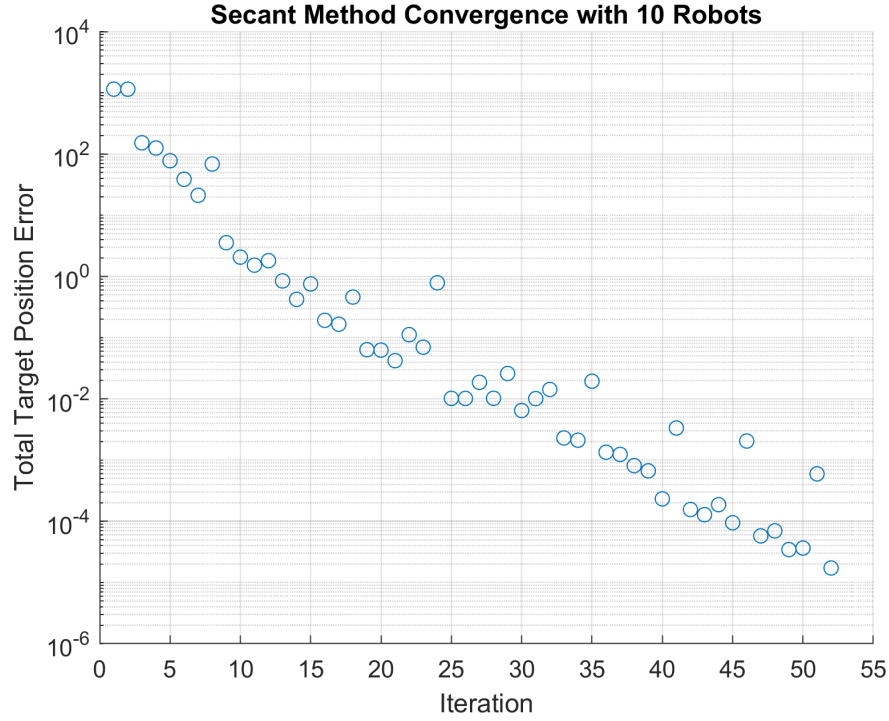

Figure 4: The total (sum of 10 robots) error between final trajectory position and target position for each iteration of secant method. 52 points evaluations are made as the first two initialization guesses are not included in the 50 total iterations, but their total errors are included.
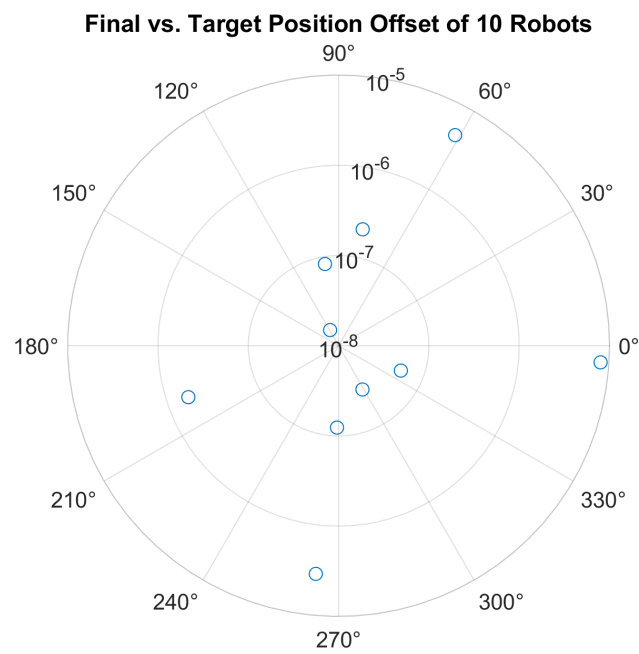
Figure 5: A log-polar graph of all target position errors for 10 robots.
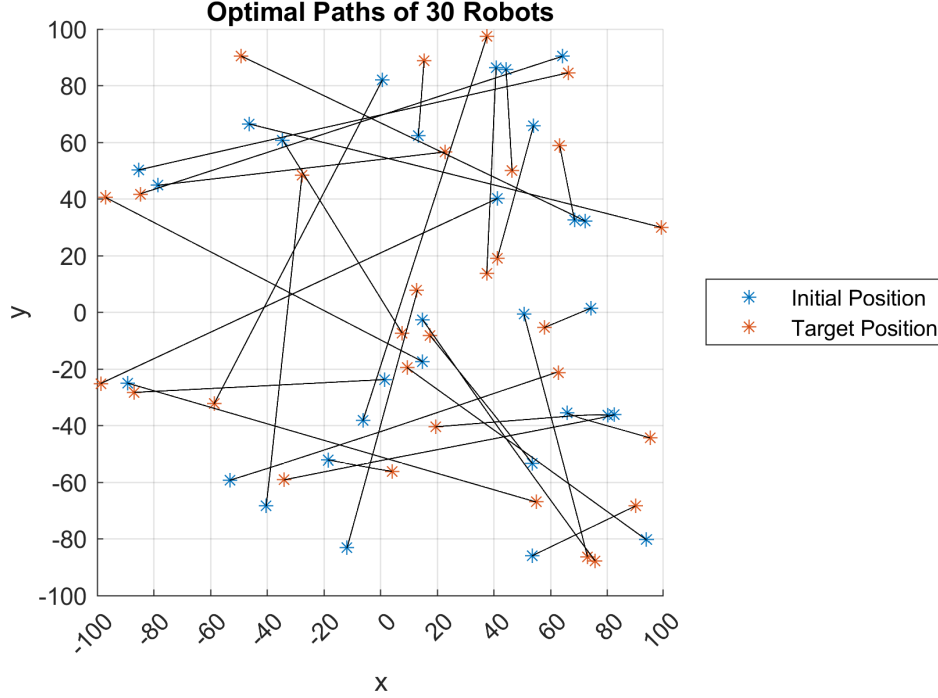
Figure 6: Optimal trajectories for 30 robots and no obstacles placed uniformly at random. The trajectories were found using a shooting algorithm with the secant method.

4.3 Even More Robots!

These same steps were repeated for 30 and 60 robots; their optimal trajectories, convergence of their total error, and their distribution of target position errors can be found in Figures 6 to 9.

Relatively good convergence was found for 30 robots, but exhibited plateaus in the first 30 iterations and the last 25. All target destinations were reached within a distance of 1 and curvature in the trajectories can be seen for close initial or target position pairs.

Unfortunately, due to the high dimensionality when there is 60 robots, the secant method was unable to converge, with some robots grossly overshooting their target positions by a magnitude greater than the space their restricted to. Other more robust or computationally efficient methods may be considered to combat this issue.

5 Discussion

Throughout the workings of the model derived, an embedded factor of evaluation was to determine when the dimensionality would supersede the constraints of the model. The secant method was found to computationally fail when robots exceeded 60. As aforementioned, other dimensionally apt models may be incorporated, but their relation to the divergence in total position error as a function of convergence rates must be studied independently. The "guess and check" shooting scheme as described hitherto, for example, resulted in various other segments of the model requiring the relevant adaptations. The incorporation of the inverse function without the interactive log function was, from a mathematical standpoint, a key factor of convergence disparities. This omission, however, was critical because an elementary log function defined for our domain wouldn't tend to 0 as distance shot to infinity.
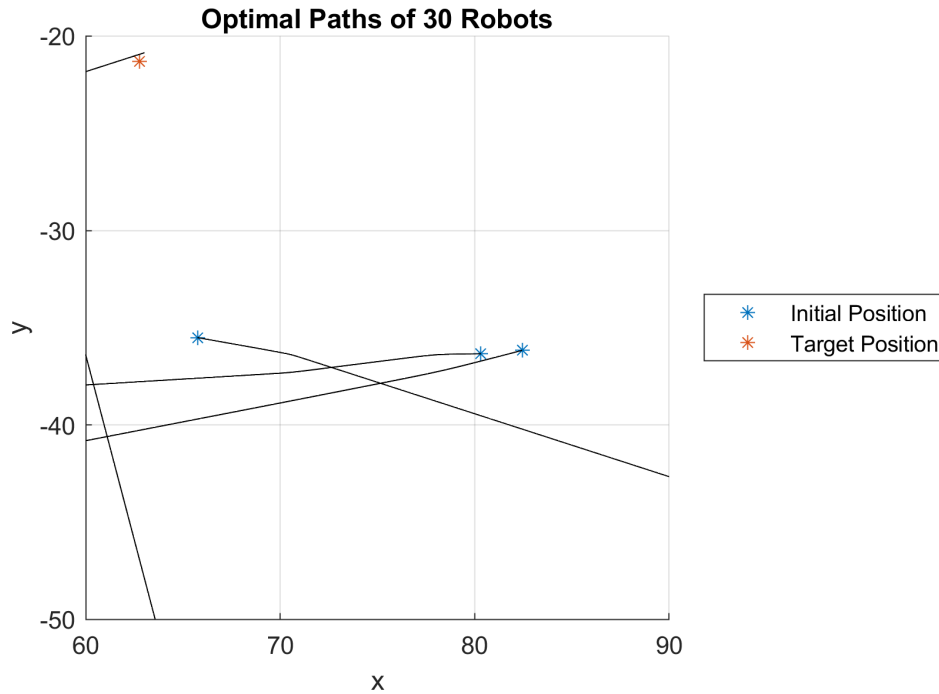
Figure 7: The region $[60, 90] \times [-50, -20]$ in Figure 6. The curvature of the path can clearly be seen from two initial positions.
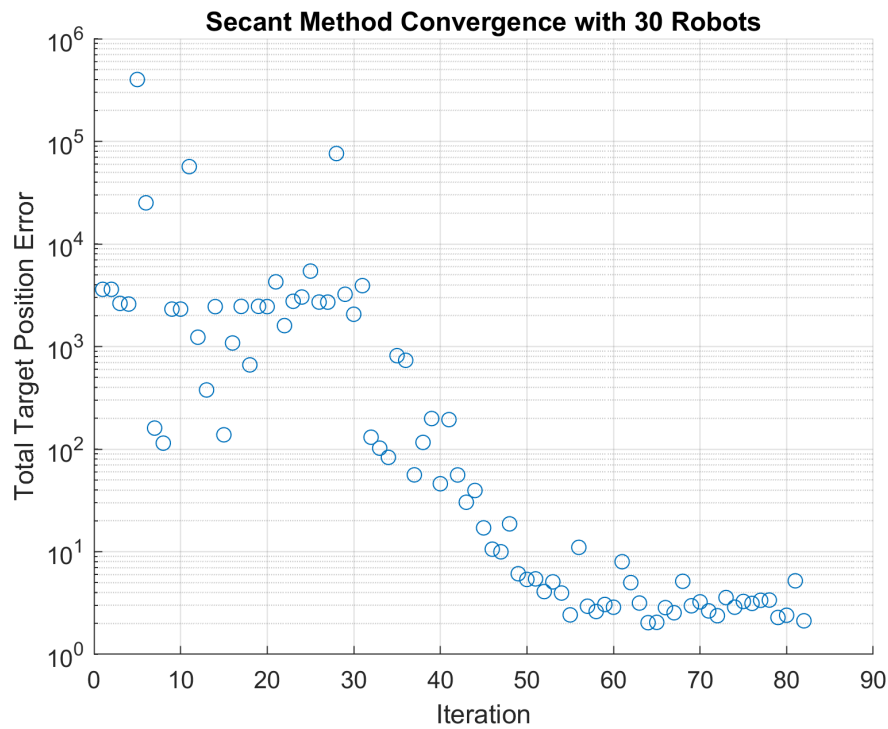

Figure 8: The total (sum of 30 robots) error between final trajectory position and target position for each iteration of secant method. 82 points evaluations are made as the first two initialization guesses are not included in the 80 total iterations, but their total errors are included.
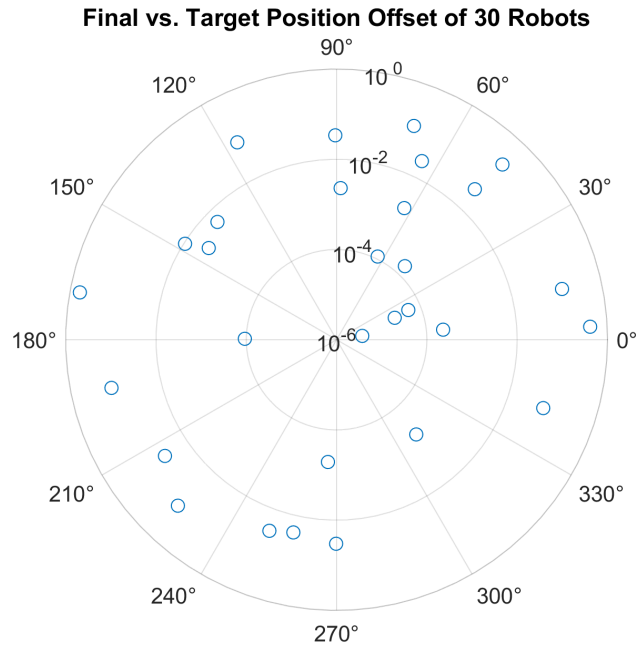
Figure 9: A log-polar graph of all target position errors for 30 robots.
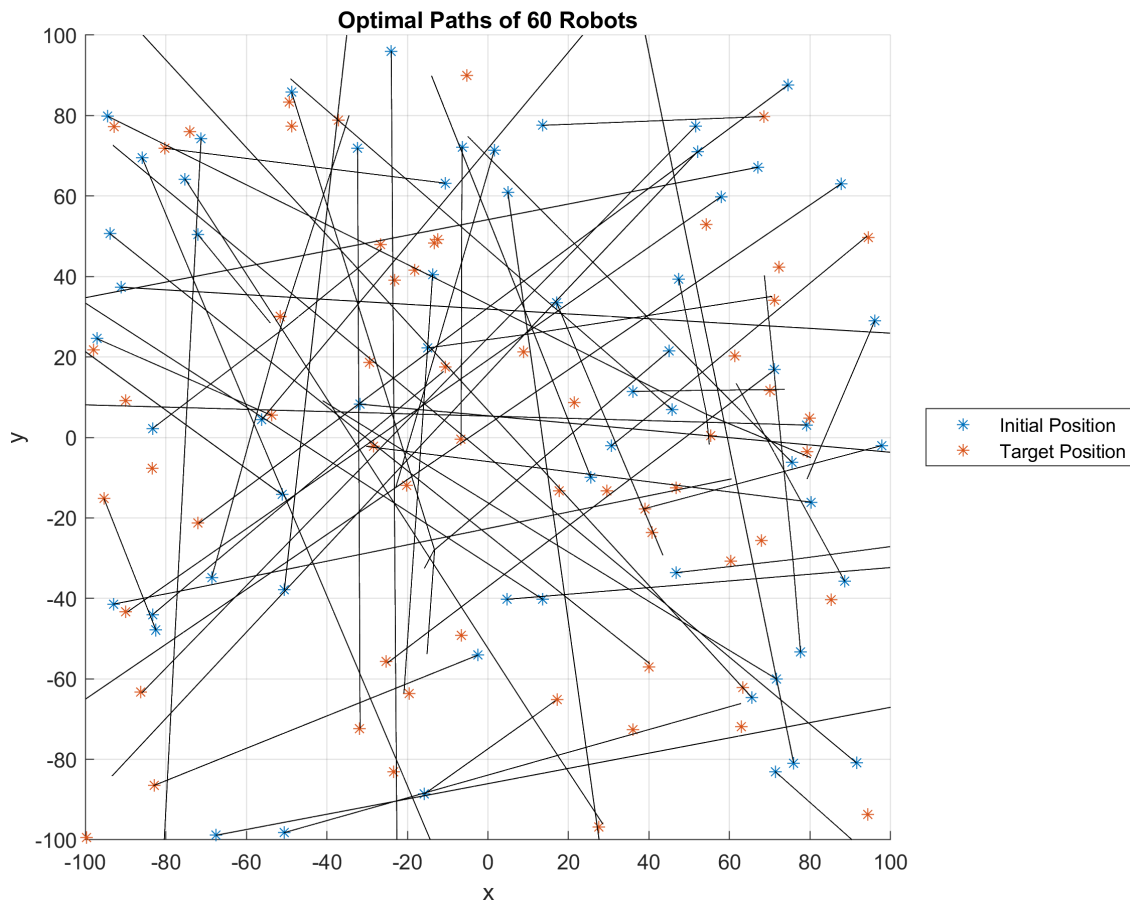

Figure 10: Optimal trajectories for 60 robots and no obstacles placed uniformly at random. The trajectories were found using a shooting algorithm with the secant method.
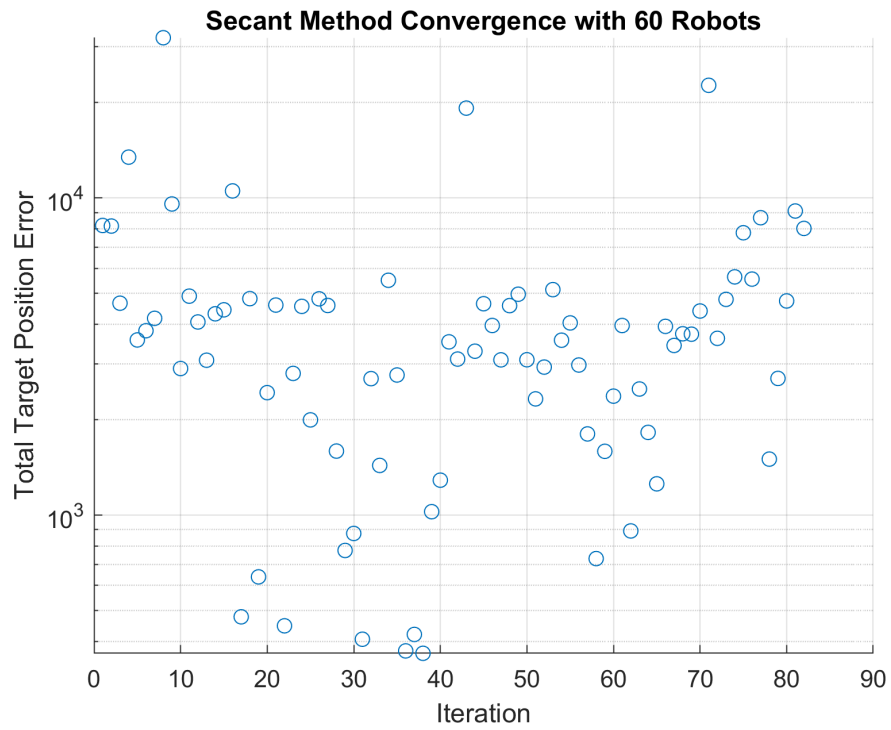
Figure 11: The total (sum of 60 robots) error between final trajectory position and target position for 80 iteration of secant method. Convergence is not seen due to high dimensionality of the search space.
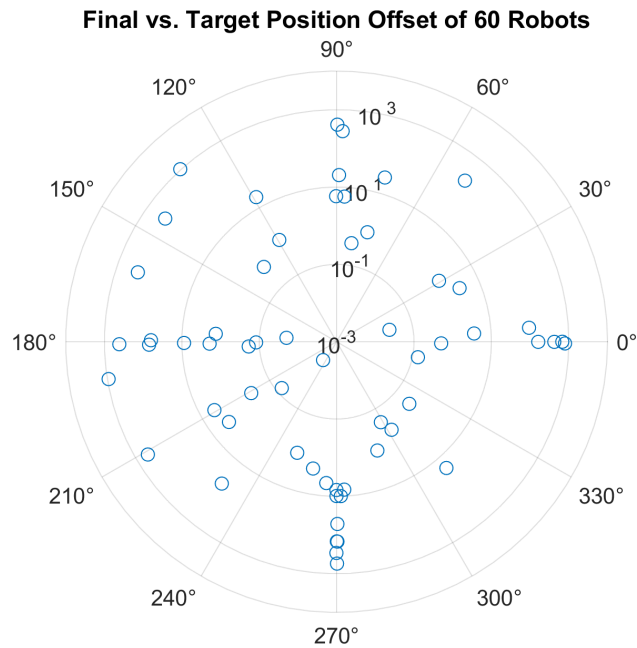

Figure 12: A log-polar graph of all target position errors for 60 robots.

6   Conclusion

An important takeaway during the entirety of the process is that optimal trajectories, when subjected to initial conditions, need not rely on the linearity of the optimal method set in place (in our case, the secant method). Within the discussion, the convergence of the shooting method, with high emphasis on reduction of the target position error, was subjected to curvature within the trajectories. The shooting method, though fairly iterative, was also subjected to conditions that were significantly determined by the initial conditions of our obstacles. Perhaps an introspective component of such a study hereafter would be the evolution of said trajectories with methods that relied on variation of polynomial terms. Muller's method, for example, is a tool that operates in a similar recursive manner to the secant method, but instead varies the iterative approximations using a polynomial of degree 2. When analyzed contextually with the 50 step iteration shown for the 10 robots in Figure 4, an arguable evaluation is that such a method may result in a quicker rate of convergence, thus potentially fewer iterations. That being said, external factors such as high dimensionality (especially in the case of 60 robots and beyond) may force the incorporation of such tools, but their efficacy would depend heavily on the relevant quadratic extrapolation.

Appendices

A   MATLAB Code

All Matlab code can be found at https://github.com/RobertYL/math-146-project. Run all scripts in version R2022b.