# 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

**Abstract**

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

# 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

# 3 Theory

## 3.1 Problem Definition

Add description of problem here.

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 $\mu$ 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 two types of barrier functions

1. Log function:

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

2. Inverse function:

$$g_{d,\mu}(x) = \frac{1}{\mu(x-d)}, \quad g'_{d,\mu}(x) = \frac{-1}{\mu(x-d)^2}. \tag{4}$$

Using this barrier function we can rewrite $F$ as

$$F(P) = \int_0^1 \left[ \sum_{i=1}^{N} \left\| \mathbf{p}'_i(t) \right\|^2 + \sum_{i,j=1;i\neq j}^{N} g_{r,\mu_1} \left( \left\| \mathbf{p}_i(t) - \mathbf{p}_j(t) \right\|^2 \right) + \sum_{i=1}^{N} \sum_{j=1}^{M} g_{R,\mu_2} \left( \left\| \mathbf{p}_i(t) - \mathbf{c}_j \right\|^2 \right) \right] dt, \tag{5}$$

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 $L$,

$$L(Y,Z) = \sum_{i=1}^{N} \|\mathbf{z}_i\|^2 + \sum_{i,j=1;i\neq j}^{N} g_{r,\mu_1} \left( \|\mathbf{y}_i - \mathbf{y}_j\|^2 \right) + \sum_{i=1}^{N} \sum_{j=1}^{M} g_{R,\mu_2} \left( \|\mathbf{y}_i - \mathbf{c}_j\|^2 \right). \tag{6}$$

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

$$\frac{d}{dt} \left( \mathbf{p}'_{i,x} \right) = \sum_{j\neq i} g'_{r,\mu_1} \left( \|\mathbf{p}_i - \mathbf{p}_j\|^2 \right) (\mathbf{p}_{i,x} - \mathbf{p}_{j,x}) + \sum_{j=1}^{M} g'_{R,\mu_2} \left( \|\mathbf{p}_i - \mathbf{c}_j\|^2 \right) (\mathbf{p}_{i,x} - \mathbf{c}_{j,x}). \tag{7}$$

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

### 3.4 Differential Equation

The Euler-Lagrange equation derived in Equation (7) provides a system of differential equations that govern the motion of each robot's path.

## 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

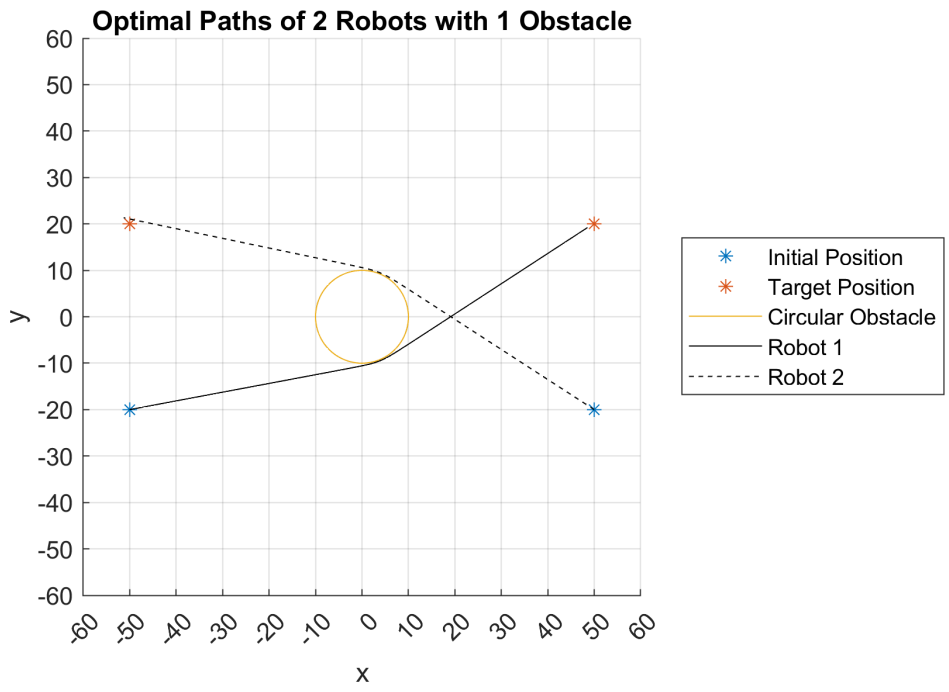


Figure 1: Optimal trajectories for two robots and one obstacles placed in an X shape. The trajectories were found by a manual shooting scheme.

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 $t = 1$.

### 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 $[-100, 100] \times [-100, 100]$.

Perhaps describe the secant method more in depth in the theory section before this. 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 $2\pi$. 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.

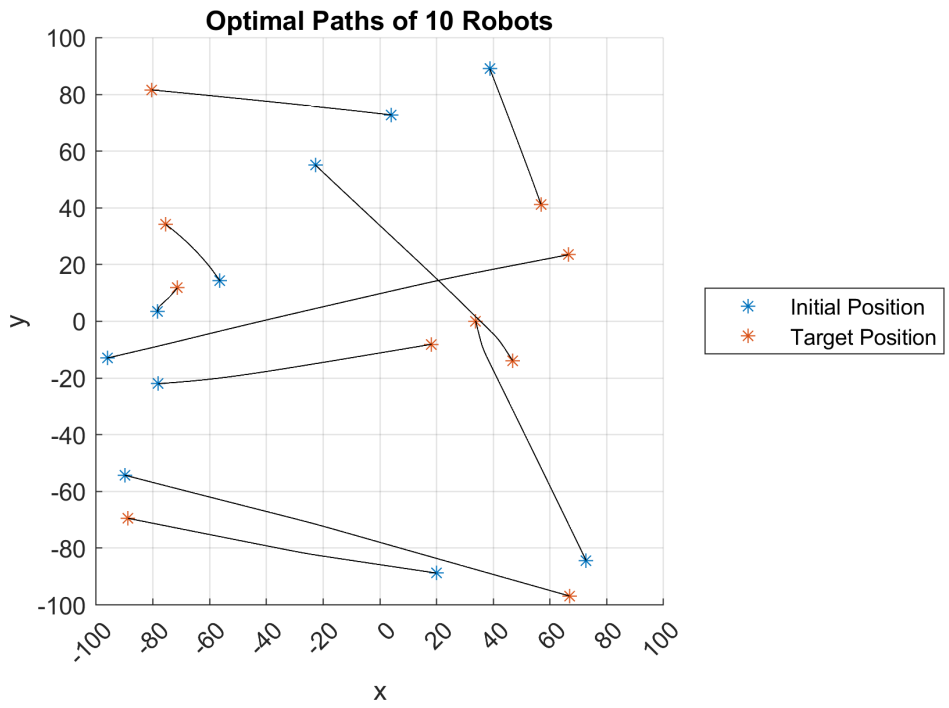


Figure 2: Optimal trajectories for 10 robots and no obstacles placed uniformly at random. The trajectories were found using a shooting algorithm with the secant method.

## 5 Results

## 6 Discussion

## 7 Conclusion

## 8 Acknowledgements

# Appendices

## A MATLAB Code