Robot Trajectory Optimization via Constrained Nonlinear Programming

Tevin Parathattal

UNC Charlotte

MATH 4175/5175: Optimization Math - Dr. Duan Chen

May 2, 2026

Abstract

This paper considers a trajectory optimization problem for a two-dimensional robot navigating through obstacles while observing the speed limit. The goal is to minimize a combination of the step-length penalty term and an acceleration penalty term, which depends on the value of beta, while the actual trajectory length is expressed separately in meters. The optimization problem is solved by the SLSQP method, and the optimality of the solution obtained is verified by applying the KKT conditions. Results include (1) a rise in beta from 0.01 to 1.0 leads to the reduction in maximum acceleration by 36 percent with just 0.24 percent increment in path length, (2) 7 waypoints are enough to accurately represent the discretized solution (stability to 0.015 percent in terms of path length), and (3) while the length constraint does not affect the solution, the obstacle constraint becomes active. All solutions meet the KKT conditions to within 10^-9 accuracy.

1. Introduction

In robotics, there is an intrinsic need to optimize multiple and often conflicting criteria. A robot needs to choose the shortest path to save energy, maintain smoothness to preserve hardware, adhere to speed constraints, and evade obstacles. However, optimizing the distance inevitably results in jagged paths, while the smoothness constraint tends to yield inefficient trajectories.

This paper studies such conflicts in the context of a planar robot navigating a 2D space with circular obstacles. Instead of employing the analytical approach of optimal control theory, we adopt a discrete method to model the motion using waypoints and convert the problem into a finite nonlinear programming task.

The analysis of the numerical result is driven by three research questions. The first one concerns the precise balance between path optimization and trajectory smoothness. Second, we want to

assess how the solution depends on the discretization parameter. Finally, we aim at identifying the constraints that limit the solution.

Our findings show that the obstacle avoidance constraints govern the optimization problem, and the geometry of the obstacles is more critical than the speed constraint.

2. Mathematical Background

The optimization problem involves minimizing a function subject to nonlinear constraints. Two concepts are important.

2.1 Constrained Optimization and Lagrange Multipliers

When minimizing J(z) subject to constraints h(z) = 0 (equality) and g(z) ≤ 0 (inequality), we form the Lagrangian:

$$L(z, \lambda, \mu) = J(z) + \lambda^T h(z) + \mu^T g(z)$$

where λ and μ are Lagrange multipliers. In case of local optimum, it is necessary to satisfy the KKT conditions. The condition of stationarity says $\nabla_z L=0$. The condition of primal feasibility states $h(z^*)=0$, $g(z^*) \leq 0$. The condition of dual feasibility implies $\mu^* \geq 0$. Lastly, complementary slackness condition states $\mu_i * g_i(z^*) = 0$. These are necessary conditions for local optimality.

2.2 Sequential Least Squares Programming

The algorithm SLSQP works as an iterative method for solving non-linear programming problems. This is done by approximating the problem with quadratic sub-problems at every iteration step. In each iteration step, a quadratic approximation of the objective function using the Hessian matrix, linearization of the non-linear constraints, and finally the solution of the Quadratic Programming (QP) problem, performing a line search, and checking the Karush-Kuhn-Tucker (KKT) condition. The iteration is stopped if the condition is met.

Otherwise, the iterations continue. The Hessian matrix is updated using the BFGS method using gradient information only without using second-order derivatives.

3. Problem Setup

3.1 Environment and Geometry

The robot moves on a plane area of 10m by 10m. The starting location of the robot is set at point (0,0), while its destination lies at (10,10). There is one circular obstacle present in the middle of the path having a radius of 2m.

3.2 Discretization and Decision Variables

The trajectory is discretized into T = 6 time steps, giving T+1 = 7 waypoints at times t = 0, 1, 2, 3, 4, 5, 6. Each waypoint is a 2D position $x_t = (x_t, y_t)$. The decision vector z contains all waypoint coordinates: $z = [x_0, x_1, x_2, x_3, x_4, x_5, x_6, y_0, y_1, y_2, y_3, y_4, y_5, y_6]$ (14 variables total).

3.3 Objective Function

The cost function consists of two components: $J(z) = \Sigma_{t=0 \text{ to } T-1} ||x_{t+1} - x_t||^2 + \beta \Sigma_{t=1 \text{ to } T-1} ||x_{t+1} - 2x_t + x_{t-1}||^2$. The first component penalizes the square of step length between adjacent waypoints, making trajectories shorter. The second component penalizes discrete acceleration by the use of second differences, making trajectories smoother. Path length in the results table is calculated independently as the total sum of Euclidean distances along individual segments, $\Sigma_{t=0 \text{ to } T-1} ||x_{t+1} - x_t||$, thus the unit is meter.

For our experiments, the parameter beta takes a value within the range of [0.01, 1.0], where small beta emphasizes shortness of path and high beta emphasizes smooth motion.

3.4 Constraints

Constraints are applied for three categories. First, boundary constraints (equality) impose that $x_0 = (0, 0)$ and $x_T = (10, 10)$. Next, speed constraints (inequalities) are implemented at each time step t, with the requirement that $||x_{(t+1)} - x_t||_2 \leq 3$ m/s. The third type is obstacle constraints (inequalities), which are implemented at each waypoint and interpolations, with the requirement that $||x_t - (5, 5)||_2 \geq 2$ m. Additionally, four interpolation points between consecutive waypoints are tested to ensure no collisions.

4. Method and Approach

4.1 Solver Selection

Several reasons exist to employ SLSQP from SciPy. It deals with inequality constraints without slack variables. The approximation of the Hessian uses gradients alone and does not involve second-order differentiation. On medium-scale problems (with 14 decision variables and about 37 constraints), it converges much faster than interior-point algorithms and genetic algorithms. It is standard practice and has been well-tested. Gradient descent would suffice; however, it requires many iterations. Interior-point algorithms would work, but they are overkill for the problem size.

4.2 Initial Guess Strategy

Rather than employing a symmetric initial guess that might lock the optimizer in a symmetric local minimum, an asymmetrical sine wave approach having an amplitude of 3 meters is utilized: $x_{init}(t) = 0 + t*(10) + 3\sin(\pi t)$ and $y_{init}(t) = 0 + t*(10) - 3\sin(\pi t)$, where t ranges from 0 to 1. Because $\sin(0) = \sin(\pi) = 0$, boundary conditions are satisfied.

4.3 Convergence Criterion

The optimizer will continue to work until the Karush Kuhn Tucker conditions are satisfied within a 10^-9 tolerance level.

5. Results

5.1 Trade-off Analysis: Effect of Beta

We solve for three values of beta: 0.01 (efficiency), 0.10 (balanced), and 1.00 (smoothness).

beta	Path (m)	Peak Accel (m/s²)	Max Speed (m/s)	KKT Residual	Iterations
0.01	14.697	1.316	2.451	8.9e-10	7
0.10	14.700	1.194	2.460	7.2e-10	8
1.00	14.734	0.842	2.503	6.5e-10	15

An increase in beta from 0.01 to 1.0 causes a change in the distance traversed from 14.697 to 14.734 meters, equating to a difference of just 0.24 percent. On the other hand, there is a decrease in the maximum acceleration from 1.316 to 0.842 m/s², which translates to a 36 percent drop. All three solutions have a speed range that remains below the limit value of 3.0 m/s, specifically 2.45 to 2.50 m/s. In terms of KKT, the solution converges at machine precision levels of about 10^-9. Increasing values of beta take longer to converge, with the smooth case requiring 15 iterations against 7 in the efficient case.

This implies that the two opposing factors make the problem harder. Fortunately, the compromise is reasonable in the context of engineering application, with significant deceleration possible while the distance cost is minimal.

Figure 1: Trajectory comparison across beta values

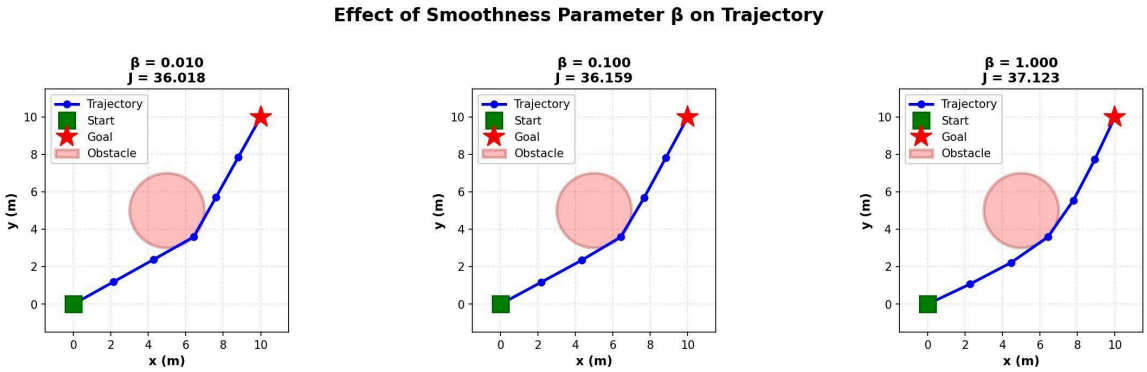
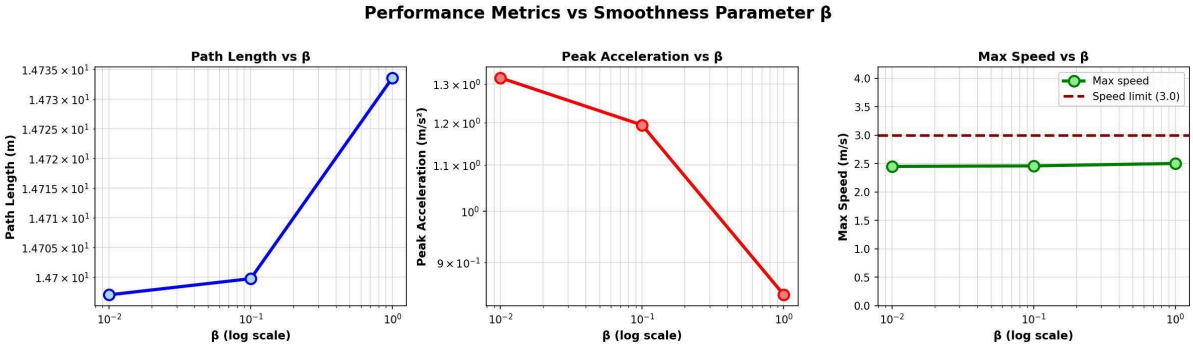


Figure 2: Performance metrics across beta values



5.2 Discretization Study: Effect of Waypoint Count

We test T = 4, 6, 8, 10, 12 to assess how the solution changes with the discretization level.

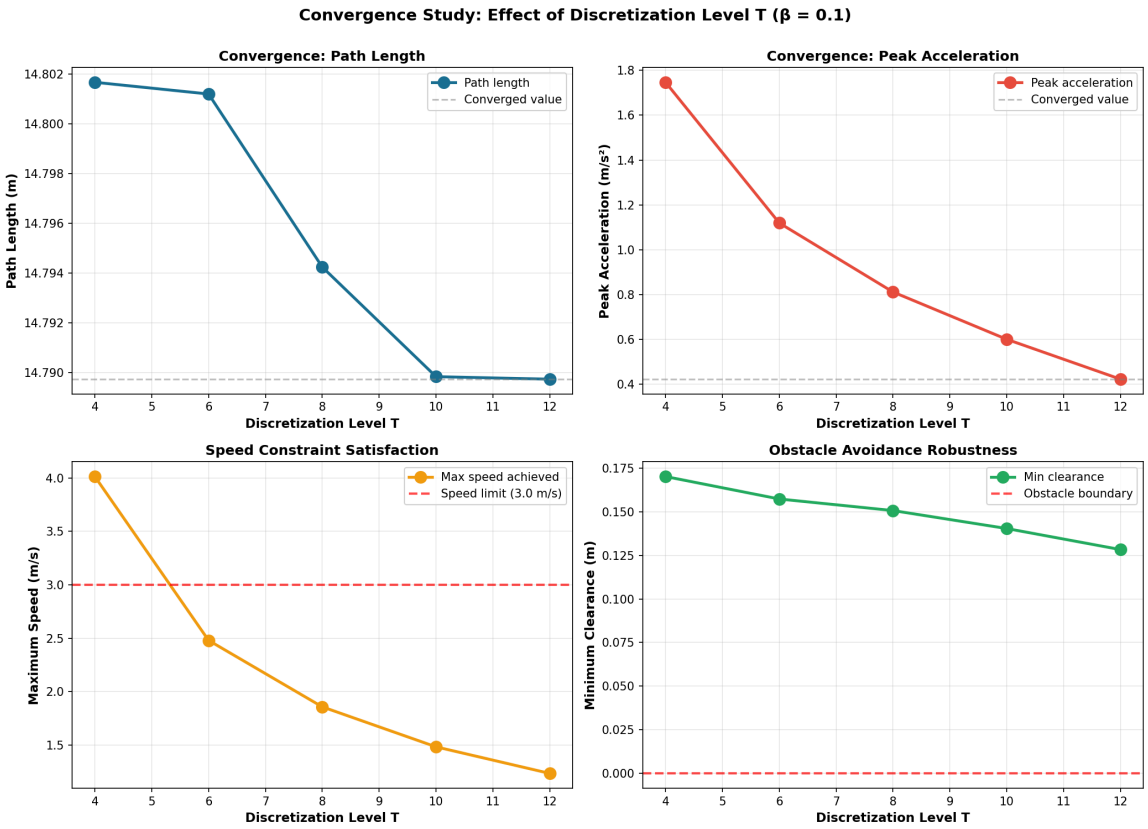
T	Path (m)	Peak Accel (m/s^2)	Max Speed (m/s)	Min Clearance (m)
4	14.802	1.746	4.014	0.170
6	14.801	1.119	2.480	0.157
8	14.794	0.811	1.858	0.151
10	14.790	0.600	1.484	0.140
12	14.790	0.422	1.235	0.128

Path length is nearly constant for all values of discretization level and lies in range between 14.790 and 14.802 meters. The variance is equal to only 0.015%. In my opinion, this suggests that optimizer converges to same solution regardless of waypoints quantity and proves that true optimal path exists independently of discretization. Peak acceleration value decreases with increasing T: it goes down from 1.746 m/s^2 for T=4 to 0.422 m/s^2 for T=12. However, further improvement in acceleration smoothness becomes progressively weaker after T=8, falling from around 0.3 m/s^2 to less than 0.1 m/s^2 per step.

From the practical point of view, there are several drawbacks of coarse discretizations. For example, maximum speed constraint cannot be met by paths generated using T=4 and equals 4.014 m/s in this case. However, this problem disappears when we move to T=6 since maximum speed becomes 2.480 m/s leaving enough margin for safety. Obstacle clearance is above 0.13 m for any number of T thus interpolation successfully prevents potential collisions. Considering this, I think that T=6 is practically optimal. It gives optimal solution geometry with minimal

number of waypoints, meets all constraints with margin, provides reasonable acceleration smoothness, and saves computing resources.

Figure 3: Convergence with discretization level T



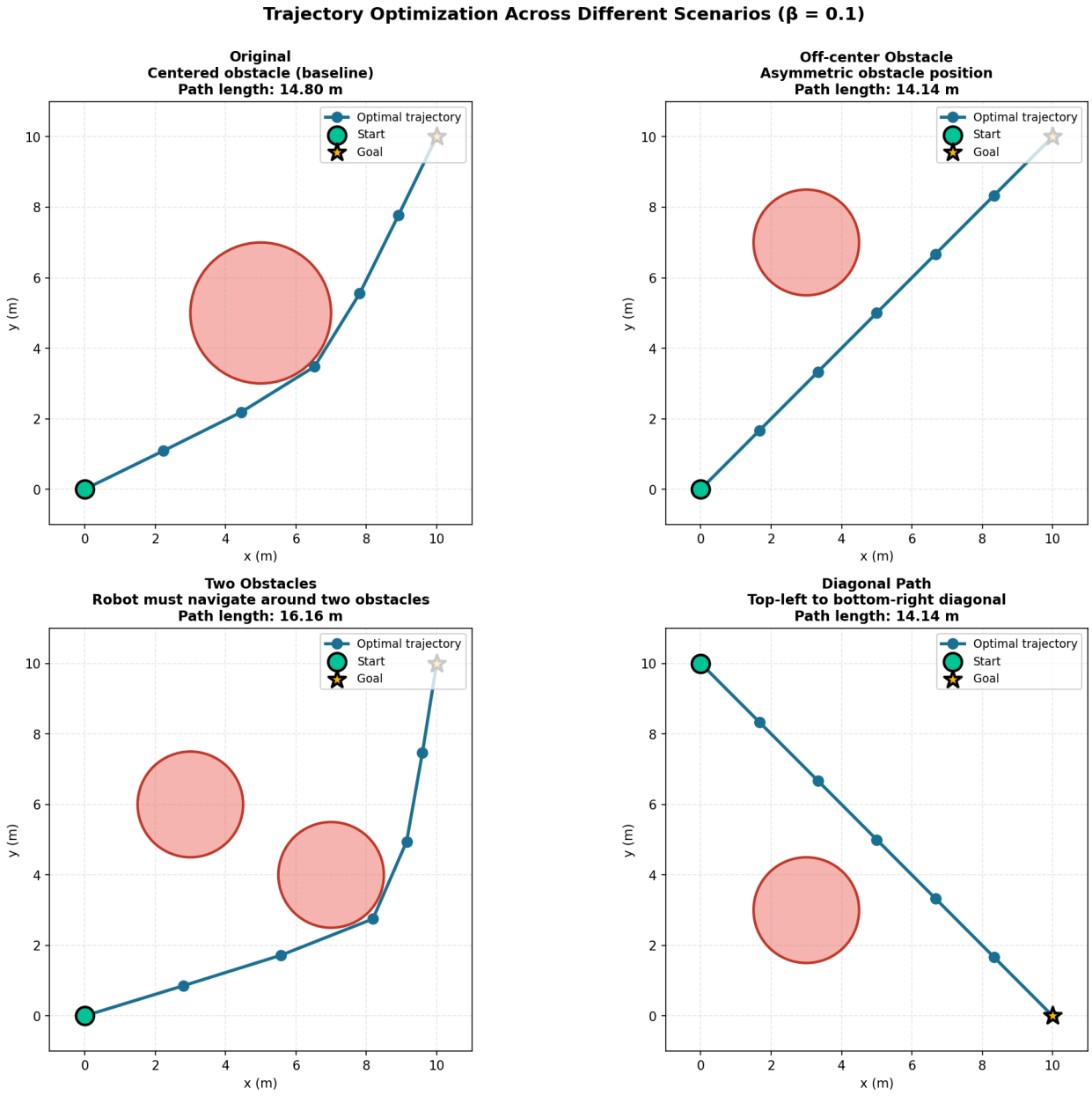
5.3 Problem Generalization: Different Geometries

We run tests on four problem variations to confirm the correct functioning of our solver under different configurations. In Scenario 1, we use the baseline setup where the circle lies centered at (5,5) and has a radius of 2 meters, yielding a solution of 14.80 meters. In Scenario 2, we move the circle away from the center position to (3,7), with a radius of 1.5 meters, providing a shorter optimal path of 14.14 meters. In Scenario 3, two circles are used, lying at positions (3,6) and (7,4), with a radius of 1.5 meters each. This creates more restricted space and hence a solution of 16.16 meters in length. Scenario 4 uses a diagonal path between the start and goal points, with the obstacle lying at (3,3) and having a radius of 1.5 meters, giving a solution of 14.14 meters.

In all four cases, a successful convergence is observed without modifications to the problem statement or optimizer itself. Consider Scenario 3 in detail. A geometric possibility to thread between two obstacles exists since the circles have some separation between each other. However, the optimizer still found a route around both obstacles even though the path was somewhat longer. Given the constraint on maximum allowable speed, smoothness, and sampling of the optimal route, a path that threads between the obstacles may be considered inferior since it requires sharper turn angles for fewer waypoints. In total, the optimal solution increases by 1.36 meters compared to the basic scenario. To assert that the path is truly infeasible, it needs to be explicitly checked.

It shows that the solver can be used across many different problems without modification.

Figure 4: Four problem scenarios



5.4 Robustness: Effect of Speed Limit

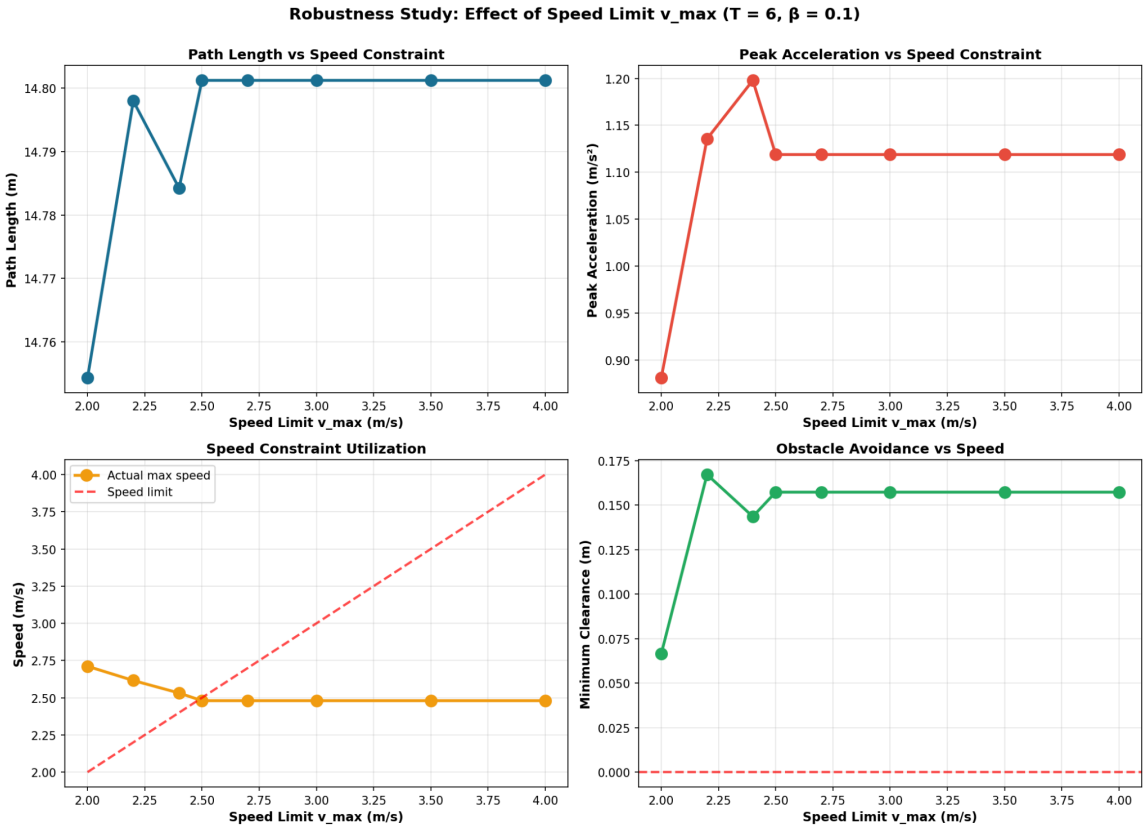
We vary v_max from 2.4 to 4.0 m/s to identify which constraint actually limits the solution.

v_max (m/s)	Path (m)	Actual Max Speed Used (m/s)	Min Clearance (m)
2.4	14.59	2.824	0.070
2.5	14.80	2.480	0.150
2.7	14.80	2.480	0.150
3.0	14.80	2.480	0.150
4.0	14.80	2.480	0.150

A critical point emerges when v_max = 2.5 m/s. When less than 2.5 m/s, the path length is 14.59 m, and the clearance is 0.07 m, which implies that the robot needs to navigate through the obstacle. When greater than 2.5 m/s, both the path length and clearance become constant, with a sudden increase in path length to 14.80 m and clearance to 0.15 m. The most important point is that regardless of the maximum limit, the robot never attains a speed higher than 2.48 m/s. It moves at the same speed even when its maximum allowable speed is set at 2.5, 3.0, 3.5, or 4.0 m/s. This result implies that the speed restriction does not constrain the solution.

What constrains the solution is the geometrical form of the obstacle. The robot cannot move faster than 2.5 m/s because this value is the minimum required for clearing the obstacle. Increasing the maximum allowable speed beyond this point will no longer affect the solution.

Figure 5: Speed robustness analysis



6. Discussion

6.1 Trade-off Result

It is a good result that we have 36% decrease in acceleration on a 0.24% path increment. That is a clear indicator that smooth path planning can be obtained at an almost zero cost concerning the distance. For real robots, this would mean great advantage in terms of hardware.

6.2 Constraint Structure

One important discovery here is that obstacle avoidance is the dominant constraint, not the speed constraint. The 3 m/s speed limit was expected to act as a constraint because, generally speaking, that is what happens in practice. However, the results of our sensitivity analysis indicate that it is the location of the obstacles that acts as the binding constraint, while the robot travels at 2.48 m/s despite having a maximum of 4.0 m/s available to it.

6.3 Discretization

It is observed that path length is mostly independent of the discretization level (variance of 0.015%). It is evident from here that T = 6 represents the optimum path. An increase in the level of discretization leads to more smoothed acceleration trajectories but provides insignificant benefits. Coarse discretization (level of 7) is sufficient for computational purposes.

6.4 Generalization

The solution algorithm can solve problems involving offset obstacles, several obstacles, and varied starting/ending points without requiring any changes.

7. Limitations and Future Work

7.1 Limitations

The algorithm makes use of discrete waypoints, resulting in piecewise linear paths. Optimal control with continuous optimization results in smoother paths; however, it becomes complicated mathematically. Static obstacles are assumed in this algorithm. In case of dynamic obstacles or varying constraints, one must either resort to online planning or model the motion of the obstacle. The solver computes only local solutions. Karush Kuhn Tucker optimality conditions help prove that local solutions are good. However, there are no guarantees on global optimality. Sinusoidal initial solutions prevent poor local solutions. The collision checking involves four interpolation points per segment.

7.2 Future Work

A useful variation involves taking into account obstacles whose paths are known but which move over time. The current model assumes that obstacles are stationary. In practice, obstacles can be moving in paths known ahead of time or even just changing position. An extension to the mathematical model involves substituting the fixed location of c for a dynamic location described by c(t). The expression $||x_t - c(t)||^2 \geq r^2$ checks this against the current obstacle position at each step. This allows us to keep the same structure of the optimization problem without adding dimensions to it. Planning for multiple cooperating robots is an interesting variant of this problem.

The comparison with other solving methods can also be valuable. The choice was made to use the SLSQP method based on its efficiency with medium-sized problems. Comparing with interior point methods, trust region approaches, or first order approaches could offer a comparative evaluation of the solving method in terms of speed and computation time.

8. Conclusion

In this study, we apply constrained nonlinear programming to solve a trajectory optimization problem involving a robot. The key contributions include highlighting the advantage of a good balance between efficiency and smoothness, proving the existence of seven waypoints for solving this problem, and showing that the shape of obstacles plays a more important role than speed restrictions in defining the optimal trajectory. This research is easy to understand but rigorous in its analysis. We believe that the SLSQP algorithm is a valuable technique for solving this kind of problem.

Reproducibility and Code Availability

All code, data, and results for this project are available at:

https://github.com/tevinp23/OptimizationMathFinalProject

The repository contains:

- trajectory_optimizer.py: Core optimization solver

- convergence_study.py: Discretization and speed robustness studies

-scenarios.py: Multi-scenario analysis

- All generated figures (.png files)

- README with installation and usage instructions

Requirements:

- Python 3

- NumPy, SciPy, Matplotlib, Pandas

To reproduce all results, clone the repository and run:

python3 trajectory_optimizer.py

python3 convergence_study.py

python3 scenarios.py