AI505/AI801, Optimization – Obligatory Assignment 1

2026-03-28

Case 1: Smooth Path Planning by Unconstrained Optimization

Problem Statement

A robot must move from a given start position to a given goal position in a 2D environment while avoiding obstacles and producing a smooth trajectory.

The trajectory is represented by a sequence of points

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n \in \mathbb{R}^2$$

where

- $\boldsymbol{x}_1$ is the start position (fixed)
- $\boldsymbol{x}_n$ is the goal position (fixed)
- $\boldsymbol{x}_2, \dots, \boldsymbol{x}_{n-1}$ are optimization variables

The goal is to compute a smooth and short path that avoids obstacles.

Optimization Model

We can define the objective function

$$f(\boldsymbol{x}) = f_L(\boldsymbol{x}) + \lambda f_S(\boldsymbol{x}) + \mu f_O(\boldsymbol{x})$$

where

- first term $f_L(\boldsymbol{x})$: takes into account the path length
- the second term $f_S(\boldsymbol{x})$: favours smoothness
- and the third term $f_O(\boldsymbol{x})$: promotes the avoidance of obstacles
- $\lambda > 0$ is a parameter that controls smoothness
- $\mu > 0$ is a parameter that controls obstacle avoidance

** Path length **

It is easy to see that a good choice for path length is

$$f_L(\boldsymbol{x}) = \sum_{i=1}^{n-1} \|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|^2$$

that is, the sum of the successive segment lengths.

Smoothness

Assume for a moment the robot trajectory is a continuous curve

$$\boldsymbol{x}(t) \in \mathbb{R}^2, t \in [0, 1]$$

We want the path to be smooth.

In calculus of variations, smoothness is enforced by minimizing curvature or acceleration:

$$\int_0^1 \left\| \frac{d^2\boldsymbol{x}(t)}{dt^2} \right\|^2 dt$$

In our discretized case where the path is determined by $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ with uniform spacing, we can use the finite difference approximation of the second derivative:

$$\frac{d^2\boldsymbol{x}(t)}{dt^2} \approx \boldsymbol{x}_{i+1} - 2\boldsymbol{x}_i + \boldsymbol{x}_{i-1}$$

From Taylor expansion:

$$\boldsymbol{x}(t+h) = \boldsymbol{x}(t) + h\boldsymbol{x}'(t) + \frac{h^2}{2}\boldsymbol{x}''(t) + O(h^3)$$

$$\boldsymbol{x}(t-h) = \boldsymbol{x}(t) - h\boldsymbol{x}'(t) + \frac{h^2}{2}\boldsymbol{x}''(t) + O(h^3)$$

and adding

$$\boldsymbol{x}(t+h) + \boldsymbol{x}(t-h) - 2\boldsymbol{x}(t) = h^2\boldsymbol{x}''(t)$$

So:

$$\boldsymbol{x}''(t) \approx \frac{\boldsymbol{x}_{i+1} - 2\boldsymbol{x}_i + \boldsymbol{x}_{i-1}}{h^2}$$

since h is a constant and the integral is approximated in the discrete case by summation we have that:

$$f_S(\boldsymbol{x}) = \sum_{i=2}^{n-1} \|\boldsymbol{x}_{i+1} - 2\boldsymbol{x}_i + \boldsymbol{x}_{i-1}\|^2$$

which can also be written in matrix form as:

$$f_S(\boldsymbol{x}) = \|D\boldsymbol{x}\|^2$$

here D is a tridiagonal matrix, a sparse matrix whose non-zero entries are confined to a diagonal band, comprising the main diagonal and one more diagonals on either side.

Obstacle Model

We can define a total penalty that needs to be minimized. In particular, every trajectory point $\boldsymbol{x}_i$ has associated a penalty function $\phi(\boldsymbol{x}_i)$ and

$$f_O(\boldsymbol{x}) = \sum_{i=1}^{n} \phi(\boldsymbol{x}_i)$$

We will assume circular obstacles and for an obstacle with center c and radius r, we can define

$$d(x_i) = \|x_i - c\|$$

We will define two penalty functions:

$$\phi_1(x_i) = \begin{cases} \frac{1}{(d(\boldsymbol{x}_i)-r)^2}, & d(\boldsymbol{x}_i) > r \\ \infty, & d(\boldsymbol{x}_i) \leq r \end{cases} \quad \text{for } i = 1, \dots, n$$

$$\phi_2(\boldsymbol{x}_i\ ) = \exp(-\alpha(d(\boldsymbol{x}_i\ )^2 - r^2))$$

Optimization Problem

With these definitions the problem becomes the following:

$$\min_{\boldsymbol{x}_2,\dots,\boldsymbol{x}_{n-1}} f(\boldsymbol{x})$$

This is an unconstrained nonlinear optimization problem. In the following it might be helpful noting that:

$$\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|^2 = (\boldsymbol{x}_{i+1} - \boldsymbol{x}_i)^\top (\boldsymbol{x}_{i+1} - \boldsymbol{x}_i)$$

Your Tasks

Task 1: Problem Setup

- Choose start and goal points
- Generate an initial straight-line path
- Define at least two circular obstacles
- Plot initial path and obstacles

Task 2: Objective Function

Implement the objective function $f(\boldsymbol{x})$ in Python.

The function should return:

- objective value
- gradient

Hint: Represent the path as a NumPy array of shape `(n,2)` for implementation. When using optimization solvers, flatten the array into a vector of dimension `2(n-2)`.

Bonus Carry out this task using nanograd and forward and backward accumulation.

Task 3: Optimization Algorithms

Implement two or more optimization algorithms suitable for solving this problem. You are free to choose among those treated in the lectures.

- Motivate your choices
- Define stopping criterion
- Plot convergence
- visualize path evolution
- Compare the use of $\phi_1(\boldsymbol{x}_i)$ and $\phi_2(\boldsymbol{x}_i)$ as penalty functions for obstacle avoidance.

Task 4: Comparison

Compare your selected algorithms and your own implementation of them with possibly others chosen among those available for the python function `scipy.optimize.minimize()`.

Remarks

Consider running experiments with:

- different number of path points ($n = 20, 50, 100$)
- different λ
- different μ
- different obstacle positions

Compare algorithms using:

- number of iterations
- runtime
- final objective value
- convergence plots
- resulting path

Some inspiration for discussion:

1. Which method converges fastest?

2. Which method is most robust to initialization?
3. How does the obstacle penalty affect convergence?
4. How does problem dimension influence the methods?
5. Do results reflect the indications from theory?