Efficiency comparison of DDP and SQP on trajectory optimization

Ta-Wei Yeh

Texas A&M University, College Station, TX

April 25, 2024

Section 1

What is trajectory optimization?

1. Trajectory optimization is an optimal control problem that outputs an optimal control trajectory considering a dynamical system over a period of time.
2. In an optimal control problem, the decision variables need to consider the system dynamics constraint which could easily be nonlinear in the real world.

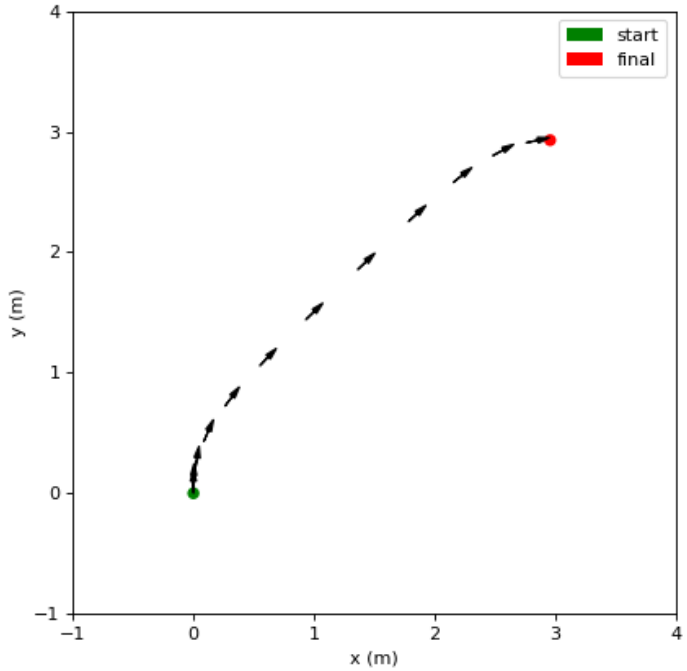

Figure: Trajectory optimization without obstacles

Problem formulation

The trajectory optimization problem is usually formulated as,

$$\min_{\mathbf{x},\mathbf{u}} \quad \int_{t=0}^{T} l(\mathbf{x}_t, \mathbf{u}_t) + l^f(\mathbf{x}_T) \tag{1}$$

$$s.t. \quad \dot{\mathbf{x}}_t = f(\mathbf{x}_t, \mathbf{u}_t) \tag{2}$$

$$\mathbf{x}_{t=0} = \mathbf{x}_0 \tag{3}$$

$$g(\mathbf{x}_t, \mathbf{u}_t) \leq 0 \tag{4}$$

- l is the cost-to-go function
- l^f is the final cost function which considers the final state
- $\mathbf{x}_t$ is the state variables (e.g. positions and orientation) at time t
- $\mathbf{u}_t$ is the control input (e.g. forward speed and angular speed) at time t
- T is the total control time horizon
- f is the differential equation of the system dynamic model
- $\mathbf{x}_0$ is the initial state
- g is the inequality constraint which could include trajectory obstacles

Trajectory with obstacles

When function g considers trajectory obstacles, it will become nonlinear.

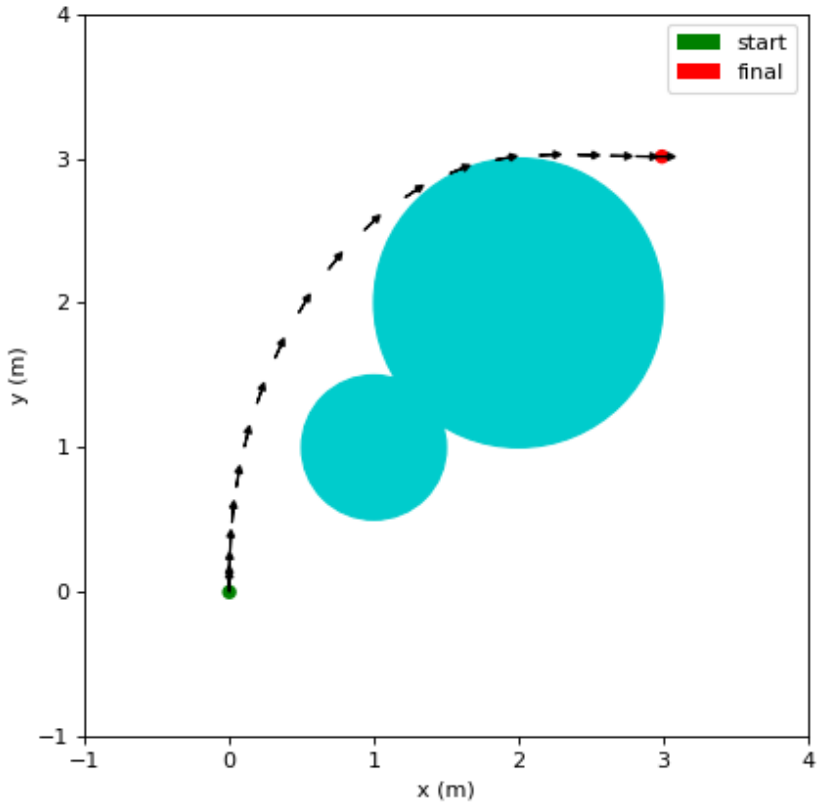

Figure: Trajectory with obstacles

Section 2

Sequential Quadratic Programming

1. Definition:
   It is an iterative method to solve a constrained nonlinear problem. Each iteration solves a sub-problem which optimizes the quadratic objective function and the linearization of the constraints.

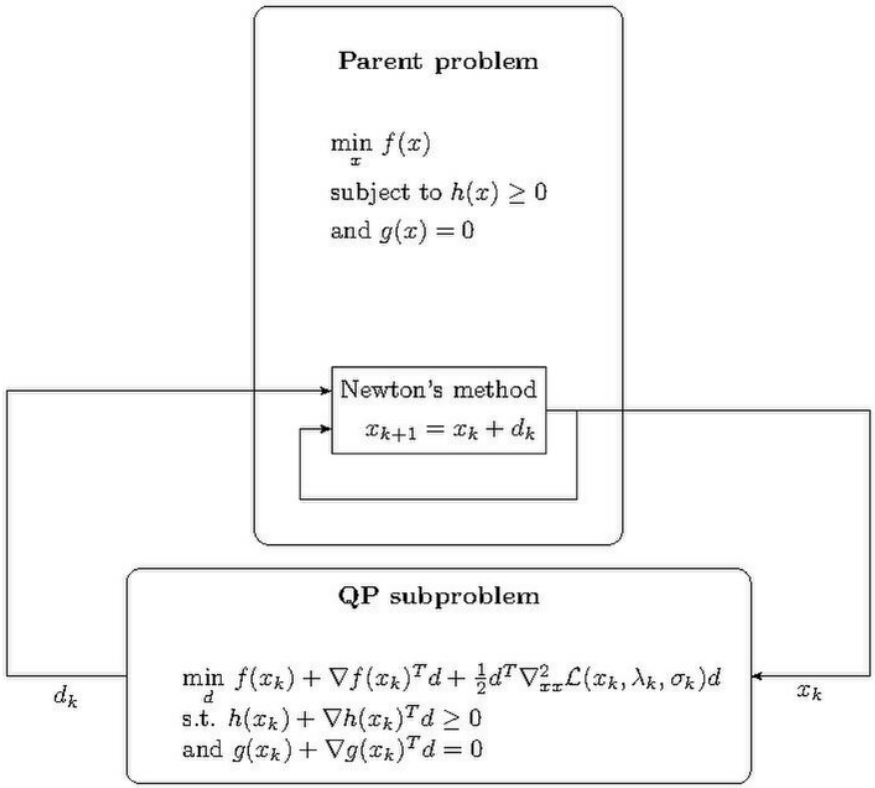

Figure: SQP schematic

How to consider dynamic function?

1. Discretizing the dynamic function
   1. Single shooting
   2. Multiple Shooting
   3. Collocation
2. Multiple Shooting
   Instead of predicting the entire trajectory (single shooting), it breaks it into many shorter segments. Each segment predicts the next state based on the current state and control.

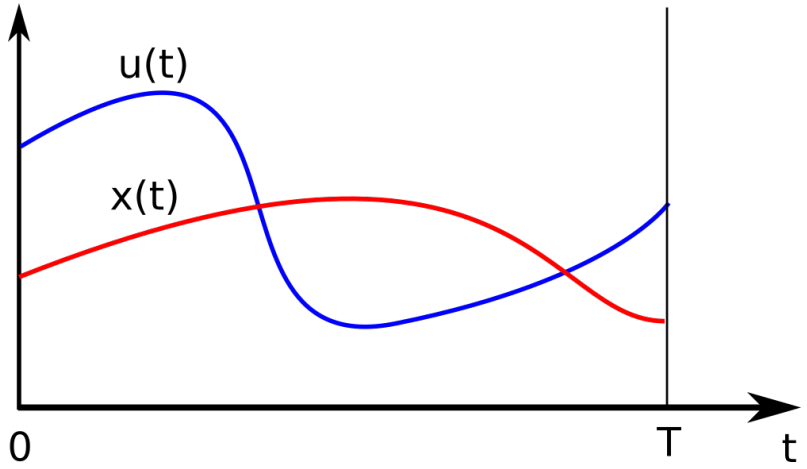

Figure: Continuous state and control

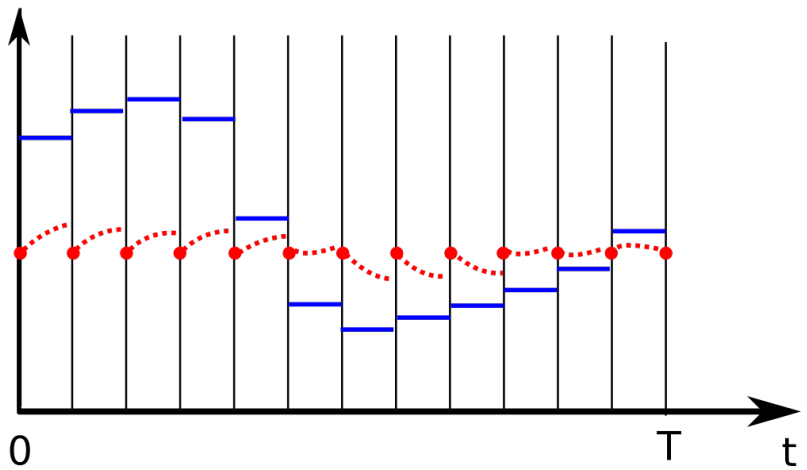

Figure: Discretized state and control

Trajectory optimization after discretization

- The objective function size expanded after discretizing the entire trajectory using multiple shooting methods. The new optimization becomes

$$\min_{\mathbf{u}} \quad \sum_{k=0}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + l^f(\mathbf{x}_N) \tag{5}$$

$$s.t. \quad \mathbf{x}_{k+1} = \mathbf{x}_k + dt \cdot f(\mathbf{x}_k, \mathbf{u}_k), \quad k = [0, N-1] \tag{6}$$

$$\mathbf{x}_{k=0} = \mathbf{x}_0 \tag{7}$$

$$g(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \qquad k = [0, N] \tag{8}$$

- where dt is the time step, N is the time horizon given by T/dt.

Differential Dynamic Programming

1. DDP transforms the large problem into a succession of low-dimensional sub-problems. The method is based on Bellman's Principle of Optimality which the current sub-problem requires the best decision from the previous sub-problem.
2. Thus, instead of optimizing the entire trajectory all at once, each sub-problem solves the control problem at a single time step. And the objective function becomes a value function,

$$\begin{aligned} V_k(\mathbf{x}) &= \min_{\mathbf{u}} \sum_{j=k}^{N-1} l(\mathbf{x}_j, \mathbf{u}_j) + l^f(\mathbf{x}_N) \\ &= \min_{\mathbf{u}} l(\mathbf{x}_k, \mathbf{u}) + V_{k+1}(f(\mathbf{x}, \mathbf{u})) \end{aligned}$$

where $V_N(\mathbf{x}) = l^f(\mathbf{x})$.

Differential Dynamic Programming Algorithm

1. Backward Pass
   Solved the value function as a quadratic problem, which creates control step directions. Thus, creates a new nominal trajectory.
2. Forward Pass
   Ensures the feasibility of the new nominal trajectory and decreases cost function by using the trust-region method.

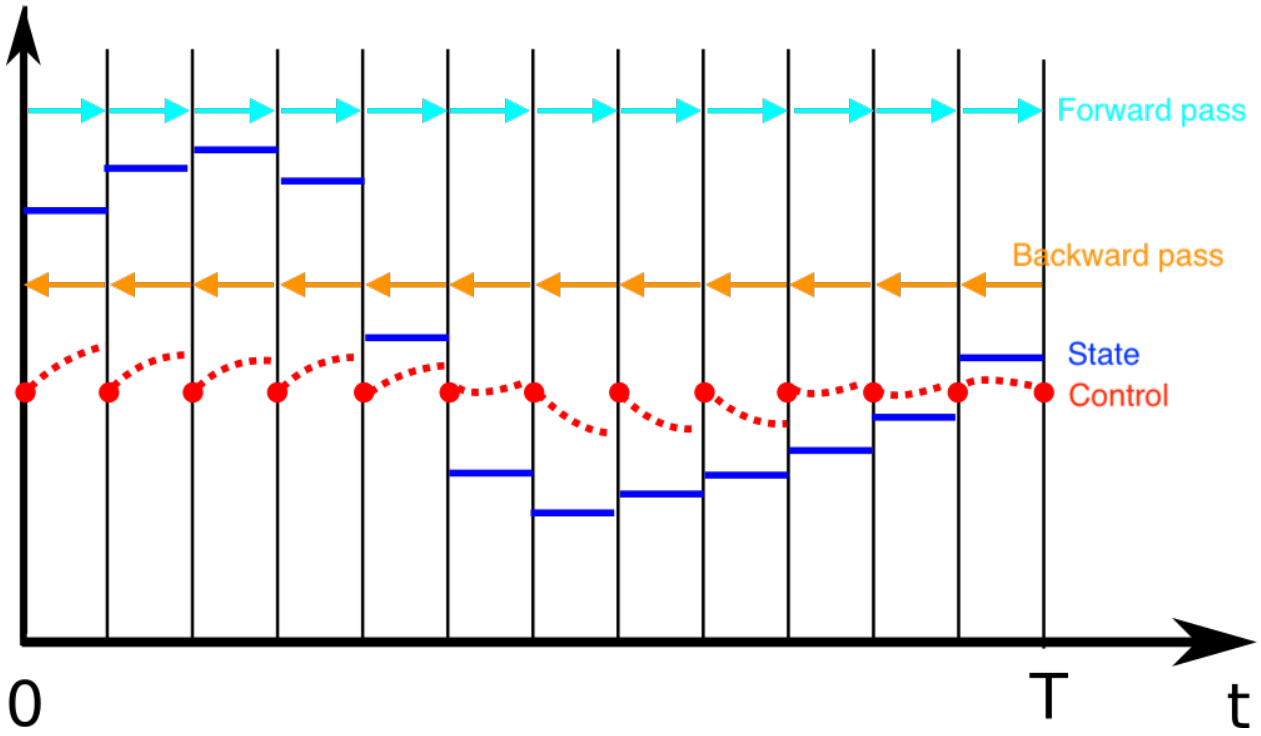

Figure: DDP Schematics

Section 3

Convergence time comparison

- DDP Convergence time: 1.27 seconds.

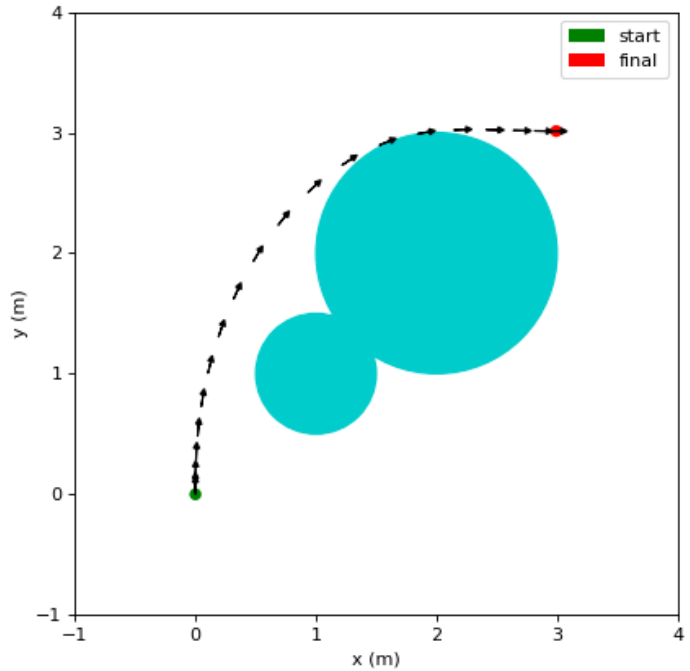

Figure: DDP trajectory with obstacles

- SQP Convergence time: 71.97 seconds.

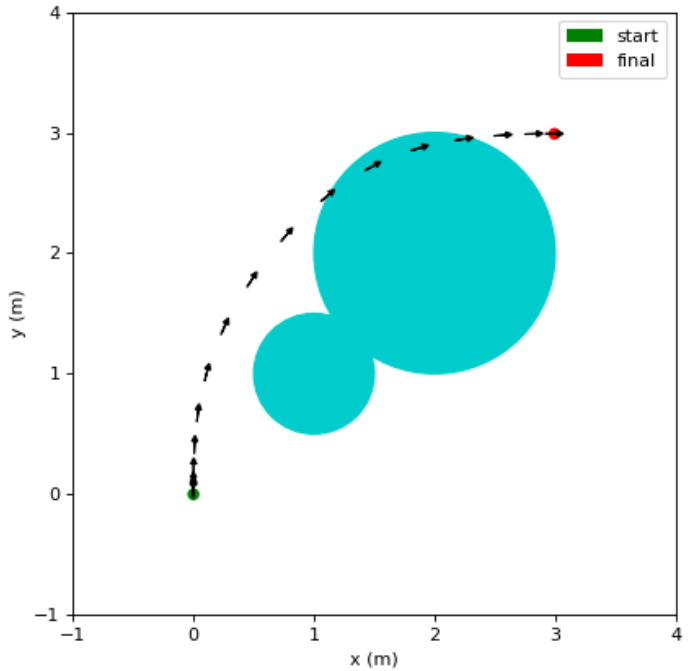

Figure: SQP trajectory with obstacles

Section 4

Conclusion

1. DDP decreases the trajectory optimization problem to a lower dimension problem which greatly improves the convergence speed.
2. SQP method could be optimized with the objective Hessian and Jacobian matrix. Moreover, considering the sparsity would accelerate the convergence speed as well. For example, the professional SQP solver, SNOPT would handle this option.

References

Z. Xie, C. K. Liu and K. Hauser

Differential dynamic programming with nonlinear constraints

2017 IEEE International Conference on Robotics and Automation (ICRA)

Lantoine, G., Russell, R.P

A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems

Part 1: Theory. J Optim Theory Appl 154, 382–417 (2012).

CasADi - Blog - Optimal control problems in a nutshell

https://web.casadi.org/blog/ocp/

Thank You!