Efficiency comparison of DDP and SQP on trajectory optimization

ISEN 623 Final Report

Ta-Wei Yeh (twy359)

1 Abstract

Trajectory optimization is an optimal control problem that outputs an optimal control trajectory considering a dynamical system over a period of time. It is usually a nonlinear optimization problem, since the dynamical system and physical constraints are generally nonlinear in the real world. We select two optimal control solvers for comparison. The first solver is Sequential Quadratic Programming (SQP), and the second solver is Differential Dynamic Programming (DDP). When applying SQP to the control problem, the system dynamic model needs to be discretized over the whole trajectory which makes the problem dimension larger. In the conclusion, we showed that DDP is more efficient than SQP in terms of computation time.

2 Introduction

Trajectory optimization is an optimal control problem that optimizes the control input over the entire trajectory. A sequence of system states composed a trajectory. Each state is the consequent result from the control input and the system dynamics at each time step. In an optimal control problem, the decision variables need to consider the sysetm dynamics constraint which could easily be nonlinear in the real world. There are many methods to solve an optimal control problem. In general, we can divide the methods into two categories, classical and alternative. There are direct, indirect, and dynamic programming methods in the classical methods. Whereas alternative methods included learning-based and geometric methods to name a few.

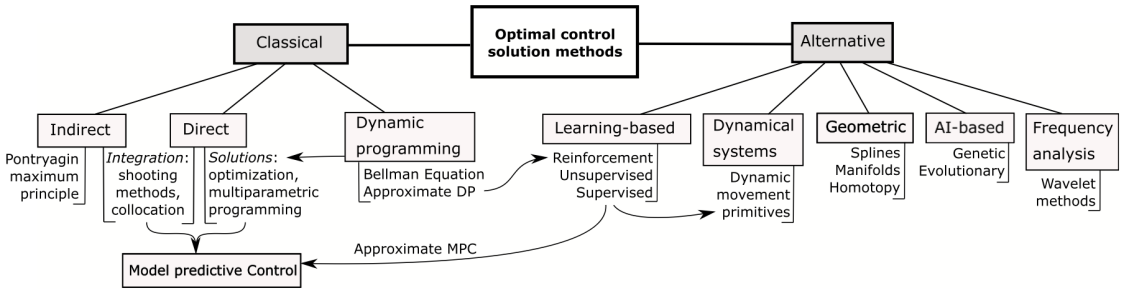

Figure 1: Approximate taxonomy of optimal control solution methods [1]

Now, we only focus on the classical methods that utilized well studied quadratic optimization solvers. In the direct method, Sequential Quadratic Programming (SQP) is widely adopted to the optimal control problem. In dynamic programming, Differential Dynamic Programming is considered because it reduces the problem dimension. Xie's Constraint Differential Dynamic Programming paper solves the control problem considering nonlinear constraints [2] which is included in the result for comparison. Both SQP and CDDP are nonlinear optimization solvers which could optimize not only nonlinear objective function but also nonlinear constraints.

In the following sections, we will introduce the trajectory optimization problem and the two solvers, SQP and DDP. Finally, we will compare the efficiency of these two solvers in terms of computation time.

3 Problem

3.1 What is a trajectory optimization problem?

A trajectory optimization problem is an optimal control problem that aims to find the optimal control inputs over a given time horizon to minimize a cost function while satisfying system dynamics and constraints. It involves finding the control inputs that result in the desired trajectory of a system while considering various constraints such as state bounds, control bounds, and path constraints. The goal is to find the control inputs that optimize a certain performance criterion, such as minimizing energy consumption or maximizing system stability.

Here is a figure 2 that simulates a car trajectory optimization problem. The car needs to reach the goal position while avoiding obstacles and minimizing the control effort. The trajectory optimization problem involves finding the optimal control inputs (steering angle, forward velocity) that guarantee the car to reach the final destination while satisfying the system dynamics and constraints.

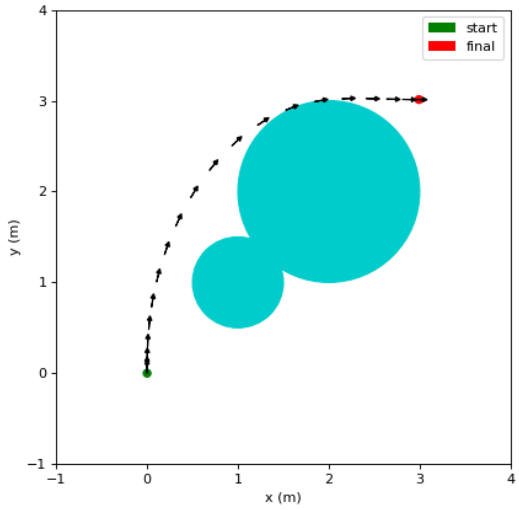

Figure 2: Trajectory optimization problem with obstacles

3.2 Problem Formulation

The trajectory optimization problem in continuous time can be formulated as follows:

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

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

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

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

where 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, and g is the inequality constraint which could include trajectory obstacles.

The goal is to minimize the trajectory cost function $l(x, u)$ within time horizon T while satisfying the system dynamics $f(x, u)$ and constraints $g(x, u)$. The final cost function guarantee the final state reach the final position. The decision variables are the control inputs $\mathbf{u}$.

4 Methods

4.1 Sequential Quadratic Programming (SQP)

Sequential Quadratic Programming is an iterative method for solving nonlinear optimization problems. It is a direct method that requires the system dynamics to be discretized over the entire trajectory. The optimization problem is solved by iteratively solving a sequence of quadratic subproblems that approximate the original nonlinear problem. At each iteration, the objective function and constraints are approximated by a quadratic model, and the solution of the quadratic subproblem is used to update the current solution. The process is repeated until convergence is achieved.

In trajectory optimization problem, each system state is the output from the system dynamics given the control input at each time step. To integrate SQP with trajectory optimization, we need to discretize the dynamic function which is usually described by a differential equation. The trajectory is composed by a sequence of the system states and control inputs over the time horizon. There are three ways to discretize the system dynamics, single shooting, multiple shooting, and collocation. In the single shooting method, the control inputs are the decision variables. In the multiple shooting method (figure 4), the state variables and control inputs at each time step are the decision variables. In the collocation method [3], additional state variables are introduced to approximate the system dynamics.

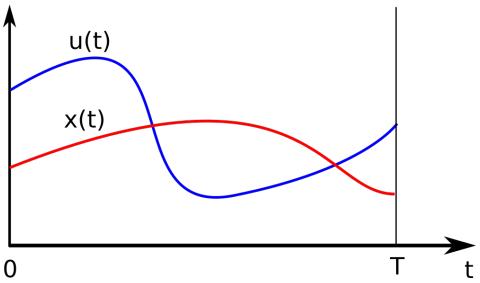

Figure 3: Continuous system dynamics

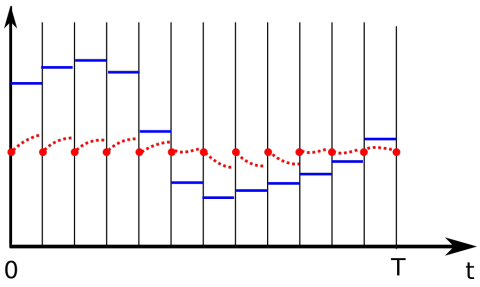

Figure 4: Multiple shooting discretized system dynamics

We decided to discretize the dynamic function with multiple shooting method since it is more accurate than single shooting method and easier to implement than collocation method. The optimization problem after discretization can be formulated as follows:

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

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

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

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

where h is the time step, and N is the time horizon given by T/h.

After discretization, the continuous system dynamics function (2) becomes (6) and the optimization problem becomes (5). The objective function discretizes the entire trajectory using multiple shooting methods. Optimizing over the entire trajectory rapidly increase the computational complexity [4] and causes problem dimension to be large making the optimization slower. Therefore, we are introducing the Differential Dynamic Programming (DDP) method to compare the efficiency.

4.2 Differential Dynamic Programming (DDP)

Differential Dynamic Programming is a dynamic programming method that solves optimal control problems by iteratively approximating the value function and the optimal control policy. It is an indirect method that does not require the system dynamics to be discretized over the entire trajectory (9). Instead, it uses a local linearization of the system dynamics to compute the optimal control inputs at each time step. This is possible thanks to the Bellman's Optimal Principle which states that the optimal control policy only depends on the resulting states from the initial states. As a result, the

value function over time horizon could be broken down into a sub-problem (10) at each time step. The sub-problem solves the optimal control input at each time step given the value function from the previous time step.

$$V_k(\mathbf{x}) = \min_{\mathbf{u}} \sum_{j=k}^{N-1} l(\mathbf{x}_j, \mathbf{u}_j) + l^f(\mathbf{x}_N) \tag{9}$$

$$= \min_{\mathbf{u}} l(\mathbf{x}_k, \mathbf{u}) + V_{k+1}(f(\mathbf{x}, \mathbf{u})) \tag{10}$$

where the boundary condition is $V_N(x) = l^f(x)$.

The optimization solves backward from time step T to 0 which is called backward pass. The backward pass determined the step directions of a nominal trajectory. However, the trajectory might violate the constraints. Thus, in the forward pass, it ensures the trajectory feasibility is satisfied and decrease the trajectory cost using a trust region method. The forward optimizaiton starts from time 0 to T. The backward and forward optimization repeats until the nominal trajectory is converged.

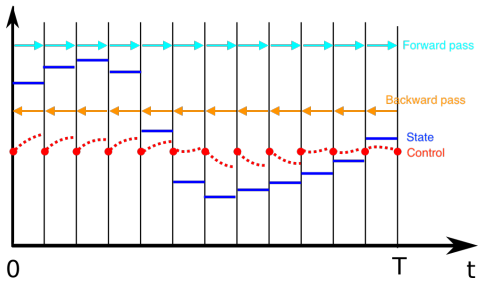

Figure 5: Differential Dynamic Programming

5 Result

The objective of the optimal control problem are the same for SQP and DDP which is driving a 2D car to reach the final destination while avoiding obstacles. To achieve this, the objective function is consisted of the trajectory cost function $l(x, u) = hu^T Ru$ and the final cost function $l^f(x) = (x - x^{goal})^T Q^f (x - x_{goal})$, where h is the time step, R is a identity matrix, Q^f is a diagonal matrix with larger diagonal values on car positions. The time horizon T is 5 seconds and multiple shooting discretize the trajectory into 100 time steps.

The 2D car follows the simplified vehicle dynamic in the CDDP paper [2]:

$$\begin{aligned} x_{k+1} &= x_k + hv_k sin(\theta_k) \\ y_{k+1} &= y_k + hv_k cos(\theta_k) \\ \theta_{k+1} &= \theta_k + hu^\theta v_k \\ v_{k+1} &= v_k + hu^v \end{aligned}$$

where the state x $= \{x, y, \theta, v\}$, includes car position (x, y), car heading angle θ, and forward velocity v. There are two control inputs which are angular velocity u^θ, and forward velocity u^k.

The implementation of Differential Dynamic Programming is pulled from Xie's CDDP GitHub repository [1]. I implemented the trajectory optimization on SQP with multiple shooting method. To have a fare comparison, both programming languages are Python and execute under Apple M2 Silicon. The SQP solver is provided by SciPy, a free optimizer. I was trying to integrate SNOPT [5], a professional SQP solver, with Casadi [6], a popular and free optimal control solver. However, it was not straightforward. If the integration was successful, the SQP computation result could have been faster since

[1] CDDP GitHub https://github.com/ZhaomingXie/CDDP.

SNOPT considers objective function sparsity, Hessian, and Jacobian matrix. My code and instructions could be found in my course GitHub repository [2].

The result figures 6 and 7 show the comparison between DDP and SQP on the trajectory optimization problem with obstacles. The computation time of DDP, 1.27 seconds, is significantly shorter than SQP, 71.97 seconds, thus the computation time of DDP is faster than SQP.

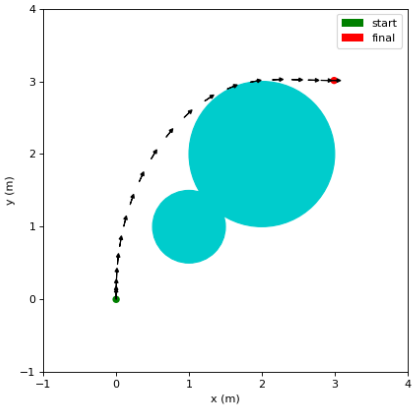

Figure 6: DDP trajectory optimization with obstacles

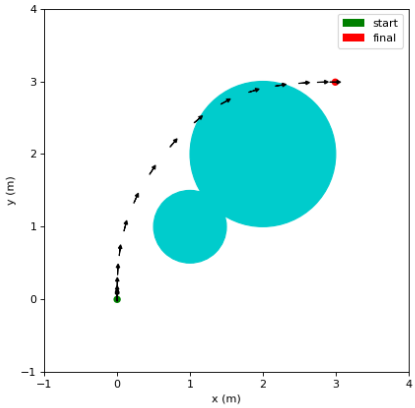

Figure 7: SQP trajectory optimization with obstacles

6 Conclusion

In conclusion, we compared the efficiency of two optimal control solvers, SQP and DDP, on the trajectory optimization problem. We found that DDP is more efficient than SQP in terms of computation time. The result shows that DDP solves faster than SQP and requires less computation time to solve the trajectory optimization problem. Therefore, DDP is a more efficient solver for trajectory optimization problems with nonlinear dynamics and constraints.

The conclusion could be exaggerated since the SQP solver could be optimized based on the objective function Hessian and Jacobian matrix. Also, the matrix sparsity could be utilized by the professional SQP optimizer, SNOPT [5]. The comparison could be more accurate if the professional optimizer is used for the SQP solver.

Nonetheless, the optimization dimension problem is discussed by Xie [2] and Lantoine [4]. SQP suffers from the problem dimension whereas Dynamic Programming overcomes the "curse of dimensionality" of pure dynamic programming. Thus, I believe DDP would be an efficient solver when it comes to a trajectory optimization problem.

References

[1] J. Drgoňa, J. Arroyo, I. Cupeiro, D. Blum, K. Arendt, D. Kim, E. Olle, J. Oravec, M. Wetter, D. Vrabie, and L. Helsen, "All you need to know about model predictive control for buildings," *Annual Reviews in Control*, vol. 50, 10 2020.

[2] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 695–702, 2017.

[3] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[2] ISEN623 Course Project https://github.com/TaWeiYeh/ISEN-623-Project

[4] G. Lantoine and R. P. Russell, "A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory," *Journal of Optimization Theory and Applications*, vol. 154, pp. 382–417, 2012.

[5] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[6] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.