AERO7970 - Trajectory Optimiztion

Project 04

Matt Boler

December 9, 2022

Abstract

This report details the implementation of an obstacle-avoiding trajectory solver using successive convexification. The solver was implemented in `Matlab` using both the built-in `coneprog` solver and the open-source `CVX` solver. The trajectories, commanded control inputs, and runtimes of the two solvers are compared.

1 Introduction

We are given the task of applying *successive convexification* (SCvx)[1] to generate optimal trajectories between two points while avoiding obstacles. SCvx is a method of solving non-convex problems by iteratively approximating the original problem as a locally convex problem. In this case, requiring the trajectory to avoid obstacles makes the search space non-convex. An example of the problem is shown below in Figure 1.

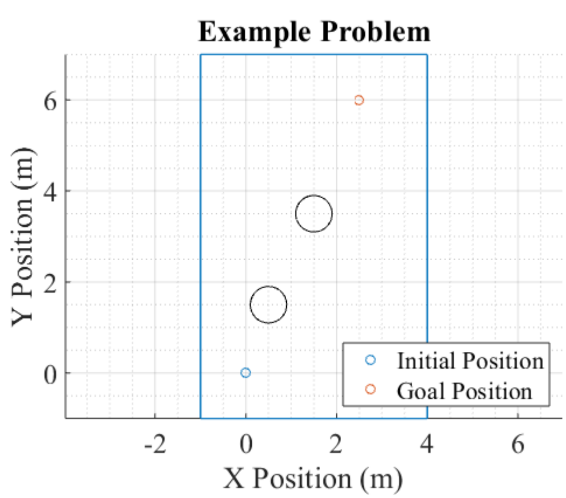


Figure 1: An example obstacle avoidance problem

2 Problem Modeling

The optimization problem is given by Problem 2 of Szmuk et. al [2]. The platform dynamics are modeled as a 3DOF 2^{nd}-order integrator and the commanded thrust is bounded above and below. Initial and final conditions are given. The optimization problem is reproduced below in Equation 1.

$$
\begin{aligned}
\min_{u^k(t),\Gamma^k(t)} \quad & w\int_t^{t_f}(\Gamma^k(t)^2)dt+\sum_{j\in\mathbb{J}}\nu_j && (1)\\
\text{s.t.} \quad & r^k(0)=r_i, \quad v^k(0)=0, \quad u^k(0)=ge_3\\
& r^k(t_f)=r_f, \quad v^k(t_f)=0, \quad u^k(t_f)=ge_3\\
& \dot{r}^k(t)=v^k(t)\\
& \dot{v}^k(t)=u^k(t)\\
& ||u^k(t)||_2\le\Gamma^k(t)\\
& 0<u_{min}\le\Gamma^k(t)\le u_{max}\\
& \Gamma^k(t)\cos(\theta_{max})\le e_3^Tu^k(t)\\
\forall j\in\mathbb{J}, \quad t\in[0,t_f]: &\\
& \nu_j\ge 0\\
& H_j\succeq 0\\
& \Delta r^{k,j}(t)\triangleq(r^{k-1}(t)-p_j)\\
& \delta r^k(t)\triangleq(r^k(t)-r^{k-1}(t))\\
& \xi^{k,j}\triangleq||H_j\Delta r^{k,j}(t)||_2\\
& \zeta^{k,j}\triangleq\frac{H_j^TH_j\Delta r^{k,j}(t)}{\xi^{k,j}}\\
& \xi^{k,j}+\zeta^{k,j}(t)^T\delta r^k(t)\ge R_j-\nu_j && (2)
\end{aligned}
$$

where Equation 2 and the preceding lines describe the relaxed obstacle constraints for each obstacle $p_j\in\mathbb{J}$.

In order to solve using SCvx, an initial solution is required. We find this initial solution by solving Equation 1 without the linearized obstacle constraints to find $\hat{r}$, $\hat{v}$, $\hat{u}$, and $\hat{\Gamma}$. These solutions are then used as the previous solution for the first iteration of SCvx.

3 Results

3.1 Coneprog Solver

SCvx was first implemented in `Matlab` using the built-in `coneprog` solver and a problem-based approach. Of note, while minimization of a squared norm can be converted from a quadratic problem to a second-order cone program, `coneprog` will not do this for you. To do so, a slack variable Y was introduced to bound the quadratic term as described in Mathworks documentation. Generated trajectories and control efforts from `coneprog` are shown below in Figures 2 and 3.

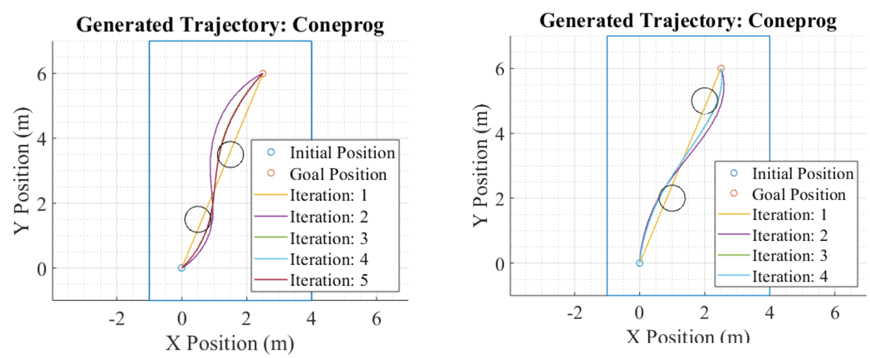


Figure 2: Trajectories generated with `coneprog`

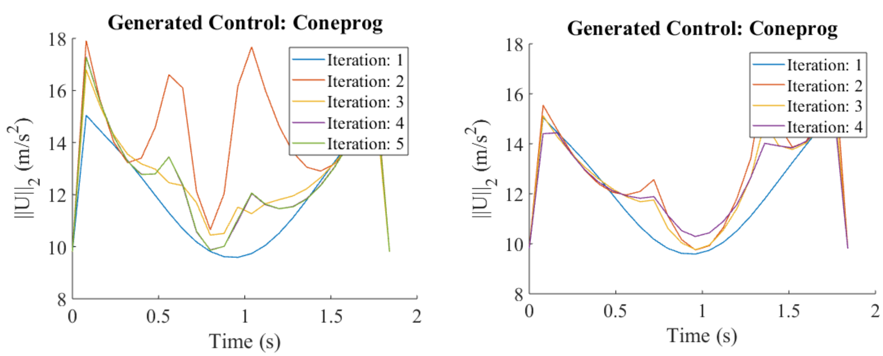
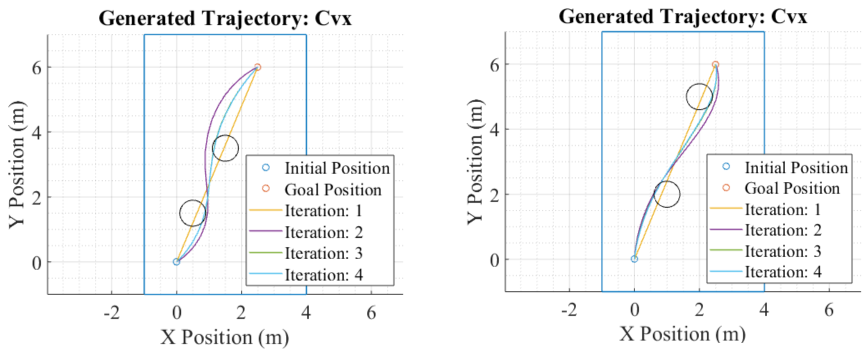


Figure 3: Control commands generated with `coneprog`

3.2 CVX Solver

Next, SCvx was implemented in the open-source convex optimization toolbox `CVX`. `CVX` has several built-in niceties such as automatic conversion of quadratic programs with SOCP constraints to SOCP programs. As a result, SCvx was able to be implemented as in Equations 1 and 2 without modification. Generated trajectories and control effort from `CVX` are shown below in Figures 4 and 5.



(a) A subfigure (b) A subfigure

Figure 4: Trajectories generated with `CVX`

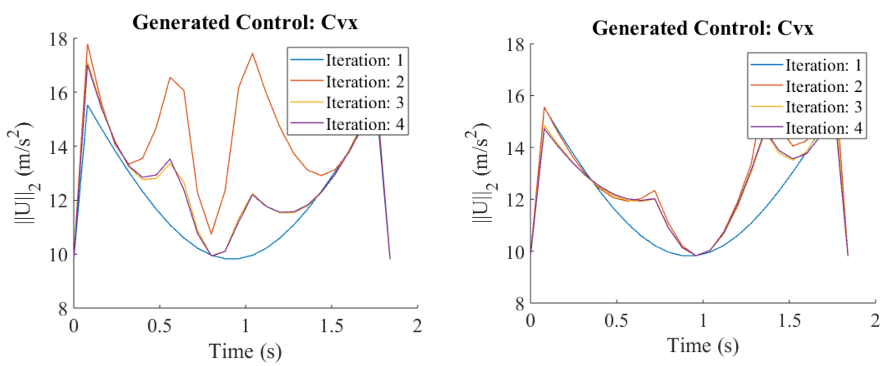


Figure 5: Control commands generated with `CVX`

3.3 Performance Comparison

The performance results of the two implementations are shown below. The `coneprog` solver is about 1 second faster than the `CVX` solver on average.

```
Testing ConeProg performance
  - Iteration: 1, 1.6008
  - Iteration: 2, 1.5742
  - Iteration: 3, 1.569
  - Iteration: 4, 1.5223
  - Iteration: 5, 1.5066
  - Iteration: 6, 1.5125
  - Iteration: 7, 1.5437
  - Iteration: 8, 1.5087
  - Iteration: 9, 1.4817
  - Iteration: 10, 1.4791
Average time to solve with ConeProg:
    1.5299
```

Performance results from `coneprog` implementation of SCvx

```
Testing CVX performance
  - Iteration: 1, 2.3492
  - Iteration: 2, 2.3063
  - Iteration: 3, 2.3183
  - Iteration: 4, 2.2605
  - Iteration: 5, 2.2148
  - Iteration: 6, 2.1939
  - Iteration: 7, 2.2043
  - Iteration: 8, 2.2814
  - Iteration: 9, 2.3704
  - Iteration: 10, 2.1954
Average time to solve with CVX:
    2.2695
```

Performance results from `CVX` implementation of SCvx

Conclusion

This report has presented results from implementing SCvx using two SOCP solvers. While the `coneprog` implementation runs faster than the `CVX` implmentation, the `CVX` implementation was significantly easier to write, test, and generally work with. I spent around 25 hours on this project.

References

[1] Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behcet Acikmese. Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems, February 2019. arXiv:1804.06539 [math].

[2] Michael Szmuk, Carlo Alberto Pascucci, Daniel Dueri, and Behcet Acikmese. Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4862–4868, Vancouver, BC, September 2017. IEEE.