Lyapunov-based Control for Sequential Convex Programming Initial Guess Generation

Julia Briden
Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA, USA
jbriden@mit.edu

***Abstract*—To improve the convergence rate of the real-time Sequential Convex Programming (SCP) powered descent guidance algorithm, a Lyapunov-based control algorithm was implemented to provide an asymptotically stable control input guess. The time-varying maximum region of attraction sum-of-squares program was implemented in PyDrake to compute the region of attraction, Lyapunov function, and control input. Semidefinite programming, with the method of alternations, was utilized to solve the resulting bilinear optimization problem. The SCP algorithm for the rocket landing problem was implemented in Julia with both interpolation-based and Lyapunov-based initial guesses. While memory limitations inhibited iteration until convergence for the semidefinite program, preliminary results show a 6% reduction in runtime for the Lyapunov-based initial guess, when compared to the straight-line method.**

***Index Terms*—trajectory, optimization, Lyapunov methods, sums-of-squares programming, nonlinear control**

I. Introduction

Guidance algorithms determine the desired path of travel for a spacecraft. These path calculations can be performed autonomously or with interventions from mission planners. During rendezvous and atmospheric entry, descent, and landing (EDL), time constraints make most human interventions infeasible. Missions such as Europa Clipper and Mars Sample Return will require large maneuvers, or diverts, to take a lander to a precise surface location [7]. By 2027, future crewed Mars Surface Missions must be capable of autonomous hazard avoidance and pinpoint landing. The large diverts required to achieve pinpoint landing require large amounts of fuel. Allocating more weight for fuel storage limits the spacecraft's capacity for telemetry and increases launch vehicle costs. Therefore, it is advantageous to calculate a fuel-optimal guidance solution. Current technologies can compute constraint-satisfying and fuel-optimal diverts but guidance solutions to problems larger than 1000 solution variables remain computationally infeasible on flight processors [5]. To enable real-time autonomous guidance for a wider range of landing scenarios it is of considerable importance to develop algorithms with faster convergence rates than state-of-the-art guidance algorithms.

Guidance for Fuel Optimal Large Diverts (G-FOLD), developed by Açıkmeşe et al., is the only current constraint-satisfying, fuel-optimal autonomous guidance algorithm capable of performing diverts for pinpoint landing [2]. By convexifying non-convex minimum thrust constraints, using the method of lossless convexification, the Interior Point Method (IPM) guarantees a solution in bounded polynomial time. G-FOLD achieves a maximum position error of less than 1 meter over a 750 meter divert and has accurately executed an 800 meter three-dimensional divert [2], [3]. However, the current implementation is constrained to 3-degrees-of-freedom (DoF) and employs a cone constraint. For missions where sensing and guidance are coupled, such as proximity operations or uncertain environmental conditions, the guidance algorithm must be capable of full 6-DoF powered descent guidance. In addition, missions with non-conic constraints, such as landing in a crater, or with infeasible solutions using lossless convexification methods remain unsolvable.

Recently, real-time Sequential Convex Programming (SCP) was applied to 6-DoF powered descent guidance problems [8]. The algorithm applies Penalized Trust Region (PTR) trajectory generation through successive convexification and solving steps. A time complexity analysis and a case study were conducted to demonstrate the algorithm's computational performance for a flight-test computer. From the case study, the total run time was bound by a cubic polynomial fit, primarily from the solver implementation. For problems using a small number of discrete nodes to compute the solution ($N < 20$), SCP can arrive at a solution on the order of 100 milliseconds with less than 1% sub-optimality. Although capable of achieving fast enough runtimes for 25 meter diverts, large diverts (1-10 kilometers) require a larger number of discrete nodes, resulting in infeasible total solution times (> 100 milliseconds). The time complexity of the algorithm also relies on two assumptions: the inertia tensor remains constant and the runtime of trust region calculations will remain small for a variety of problem formulations. As fuel is consumed during the maneuver, the inertia tensor changes. When a large divert is planned, the large quantity of fuel consumed will likely require the inertia tensor to be updated. A repeated update to the inertia tensor will increase the time complexity of the program. An additional source of uncertainty in total runtime is the selection of the trust region. The 2-norm trust region was chosen manually after observing a 11 milliseconds per iteration increase in solver time for the infinite-norm trust region. Since the optimal trust region type may vary by problem type, it provides a sense of uncertainty in the calculated runtime complexity. To account for uncertainties

in the effects of problem formulation while maintaining a guidance path robust to disturbances, additional developments must be made to minimize solver runtime.

Convex optimization problem-solving methods provide potential near-term solutions for the challenges associated with computational complexity and uncertainties. To advance large divert guidance, new methods must be formulated for solving multi-constrained optimization problems to improve runtime on current flight processors. The SCP powered descent guidance algorithm currently initializes the state solution guess by using the straight-line initialization method [8]. Reynolds et al. has shown that certain initial guesses will result in a faster convergence, demonstrating that methods could be designed to generate initial guesses to minimize runtime [10]. Developments in control design along trajectories with Sums-Of-Squares (SOS) programming could be implemented to generate an asymptotically stable and quickly converging initial guess [1]. With an initial guess trajectory that maximizes the region of attraction for the rocket landing problem, a single SOS-generated guess could significantly reduce the average number of iterations required to reach the optimal solution for the SCP program. Figure 1 shows an example of the convergence and runtime for the 6-DoF free-flyer problem solved by SCP with a linear initial guess [4]. Instead of rapidly approaching the optimal solution after iteration 10, the SOS-generated initial guess may start significantly closer to the optimal solution.

To evaluate the performance of SOS programming for SCP initial guess generation, the process for generating a valid Lyapunov guess for the SOS program and resulting SCP-generated trajectories are visualized. In addition, the Penalized Trust Region (PTR) solver runtimes are quantified to compare the straight line initial guess and the SOS-generated initial guess strategies.

II. Related Work

The development of initial guess strategies for optimization problems is a prominent topic in optimal control literature; a Lyapunov-based predictive control algorithm has been designed to utilize the Lyapunov function to formulate an initial guess for the model predictive control algorithm [6]. In addition, the guided policy search initial guess strategy has been explored for the SCP trajectory optimization problem [9]. While the guided policy search provides initial guesses that are close to optimal, the process it takes to determine the guess is not entirely observable; a neural network policy and the trajectory optimization problem dataset are used to formulate guesses without a closed-form mathematical description of the guess's origin. In contrast, SOS-methods use Lyapunov functions, which provide stability guarantees as well as develop a bounded control design. While a policy-based initial guess may make unexpected decisions due to conditions outside of the training dataset, SOS guesses will always remain inside of the region of attraction for the linearized system. Since Lyapunov-based initial guess strategies have not yet been applied to the spacecraft large divert problem, this project constitutes an approach new to its applied field with the stability guarantees required for safety-critical missions.

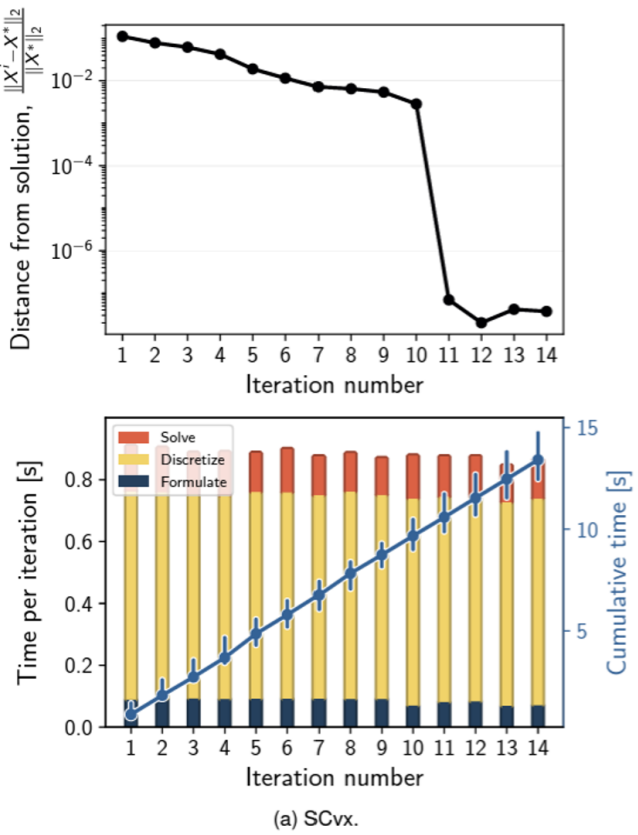

Fig. 1. Convergence and runtime for the 6-DoF free-flyer problem.

III. Approach

The system dynamics for the large divert guidance problem are shown in Figure 2. Where T, the thrust, and δ, the gimbal angle, are the system's inputs. The problem's objective is to minimize fuel usage subject to state and control constraints.

Figure 3 illustrates the goal of this project: developing an informed initial guess strategy for SCP that dominates the convergence rate when compared to the straight line method.

To develop an initial trajectory guess strategy for the SCP rocket landing problem, two software products were developed: a PTR SCP solver in Julia and a time-varying maximum region of attraction SOS program in PyDrake. All code and videos for this project can be found at juliabriden.com/projects.

A. Sequential Convex Programming Implementation in Julia

SCP algorithms solve nonconvex programming problems by solving a sequence of local convex approximations to a nonconvex optimization problem. An inner loop of defining an initial trajectory guess, creating a linearized and discretized convex subproblem, and updating the initial guess trajectory occurs until the outer loop's solution converges. For this

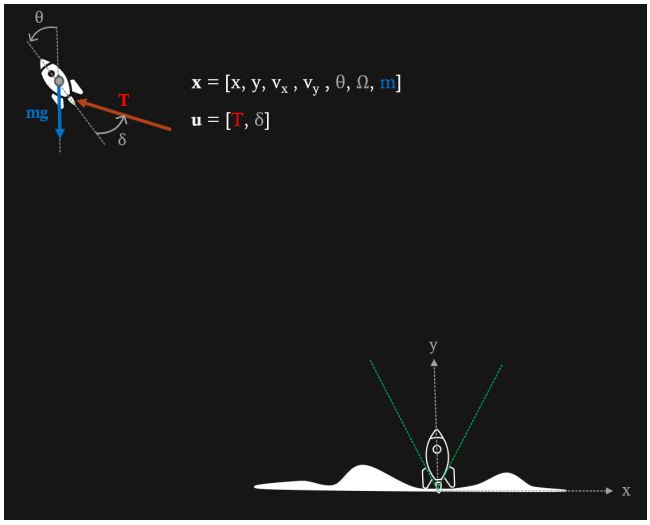

Fig. 2. Rocket landing problem.

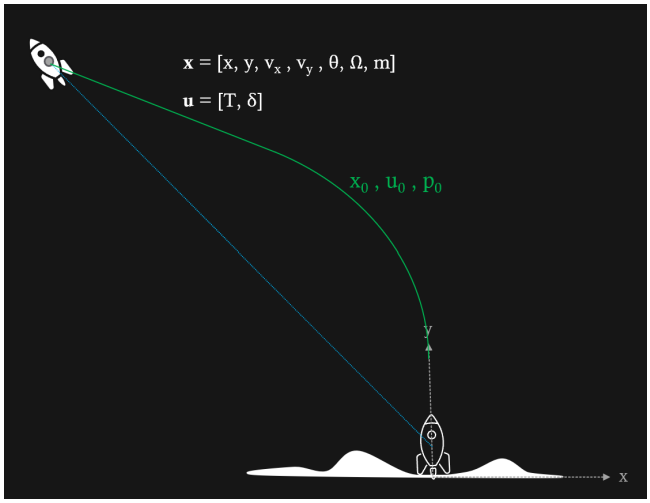

Fig. 3. Initial guess proposal.

project, the SCP Toolbox was utilized to define and solve the large divert guidance problem with SCP [4].

1) Rocket Dynamics: The rocket dynamics are defined in the form:

$$\dot{x} = f_{rocket}(x, u) \tag{1}$$

Where x is the rocket state (Equation 2) and u is the control input (Equation 3).

$$\mathbf{x} = [x, y, v_x, v_y, \theta, \omega, m] \tag{2}$$

$$\mathbf{u} = [T, \delta] \tag{3}$$

The ordinary differential equations describing the rocket dynamics are determined from Malyuta et al. and shown in Equation 4 [4].

$$\dot{\mathbf{x}} = \begin{vmatrix} v_x, \\ v_y, \\ -\frac{T}{m} sin(\theta + \delta), \\ \frac{T}{m} cos(\theta + \delta) - g, \\ \omega, \\ -\frac{LT}{J} sin(\delta), \\ -\alpha T \end{vmatrix} \tag{4}$$

All values and limits for the parameters in the differential equations are defined in Table 1.

TABLE I
ROCKET LANDING PARAMETERS FOR THE JULIA SCP IMPLEMENTATION

Rocket Parameters		
Variable	***Description***	***Value***
g	gravitational acceleration of the planet	1.625 m/s^2
m_{wet}	initial mass	25,000 kg
L	thrust lever arm	0.5 m
J	moment of inertia	100,000 kgm^2
T	thrust	$20,000N \leq T \leq 80,000N$
α	fuel consumption rate coefficient	2.756e-4 km/s/N
δ	gimbal angle	$-10^\circ \leq \delta \leq 10^\circ$
γ	glideslope angle	45°

Jacobians for the rocket dynamics are defined with respect to the state, input, and parameter vectors (Equations 5, 6, and 7).

$$A(x, u, p) = \nabla_x f(x, u, p)$$
$$= \begin{bmatrix} 0 & 0 & t_f & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{T t_f \cos(\delta+\theta)}{m} & 0 & \frac{T t_f \sin(\delta+\theta)}{m^2} \\ 0 & 0 & 0 & 0 & -\frac{T t_f \sin(\delta+\theta)}{m} & 0 & -\frac{T t_f \cos(\delta+\theta)}{m^2} \\ 0 & 0 & 0 & 0 & 0 & t_f & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

$$B(x, u, p) = \nabla_u f(x, u, p)$$
$$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{t_f \sin(\delta+\theta)}{m} & -\frac{T t_f \cos(\delta+\theta)}{m} \\ \frac{t_f \cos(\delta+\theta)}{m} & -\frac{T t_f \sin(\delta+\theta)}{m} \\ 0 & 0 \\ -\frac{L t_f \sin(\delta)}{J} & -\frac{LT t_f \cos(\delta)}{J} \\ -\alpha t_f & 0 \end{bmatrix} \tag{6}$$

$$F(x, u, p) = \nabla_p f(x, u, p) = f_{rocket}(x, u) \tag{7}$$

2) Boundary Conditions: The boundary conditions are defined for the state as a fixed point above the landing site and a point at the landing site at the end of the trajectory (Equation 8). The final mass is unspecified since the system depletes mass over time.

$$\mathbf{x_0} = \begin{vmatrix} 0.5km, \\ 1.4km, \\ 80km/h, \\ -100km/h, \\ \text{-}30^o, \\ 0^o, \\ \mathrm{m}_{wet} \end{vmatrix} \tag{8}$$

3) State Constraints: To constrain the rocket from colliding with the ground, a glideslope constraint is enforced (Equation 9).

$$|x| - \frac{y}{\tan(\gamma_{gs})} \leq 0 \tag{9}$$

In the SCP program, the constraint is defined as an L1 cone constraint.

4) Control Constraints: Control constraints ensure that the control inputs are attainable for the rocket's propulsion system. Equation 10 shows the three control constraints for the rocket landing problem.

$$T_{\min} - T, T - T_{\max}, \begin{bmatrix} \delta_{\max} \\ \delta \end{bmatrix}. \tag{10}$$

The constraints bound the thrust vector and gimbal angle and are written as three cone constraints.

5) Objective Function: The objective function is written to minimize fuel usage, which is equivalent to maximizing the final mass of the rocket (Equation 11).

$$J(x, u, p) = \phi(x(1), p) + \int_0^1 \Gamma(x(t), u(t), p)dt \tag{11}$$

Where ϕ and Γ are defined in Equation 12.

$$\phi(x(1), p) = -\frac{m(1)}{m_{wet}}, \Gamma(x(t), u(t), p) = 0 \tag{12}$$

6) Initial Trajectory Guess: The initial trajectory guess is the portion of the SCP algorithm that will be changed to the Lyapuonv-based trajectory. The straight-line guess is implemented using interpolation and is the baseline for the performance assessment. The state, control input, and parameters are interpolated between the initial and final conditions of the rocket.

7) Variable Scaling: To improve the performance of the SCP algorithm, automatic scaling is done using the expected value ranges for each variable.

8) Solving with Penalized Trust Region: After all parts of the problem are defined, the PTR solver is called to complete successive convexification of the optimization problem until convergence. The TickTok package is Julia is used to time the solving process. Finally, plots for all states and parameters for the problem are created to visualize the final computed trajectory.

B. Maximum Region of Attraction Sum-of-Squares Programming in PyDrake

The implementation of the program to determine the initial guess for the rocket landing problem is outlined by Majumdar et al. [1]. First, the linearized dynamical system is used in the Linear Quadratic Regulator (LQR) equation to determine the matrix S(t). Then, the initial guesses for the Lyapunov function and region of attraction for the SOS program are defined. Finally, the maximum region of attraction, Lyapunov function, and control input are found using semidefinite programming and the method of alternations.

1) Time-Varying Linear Quadratic Regulator: The control law for LQR is obtained by solving 13 for S(t), the matrix that defines the optimal quadratic cost-to-go.

$$-\dot{S}(t) = Q - S(t)B(t)R^{-1}B^T S(t) + S(t)A(t) + A(t)^T S(t) \tag{13}$$

Where Q and R are defined as identity matrices and A(t) and B(t) are the linearized system dynamics defined in Equations 5 and 6. The nominal trajectory, $\bar{x}$ and $\bar{u}$, chosen for the linearization is a previous trajectory computed by the SCP algorithm.

2) Initial Candidates for the Lyapunov Function and Region of Attraction: The initial Lyapunov candidate, $V_{guess}(\bar{\mathbf{x}}, t)$, is defined in Equation 14.

$$V_{guess}(\bar{\mathbf{x}}, t) = (\mathbf{x} - \mathbf{x_0})^T S(t)(\mathbf{x} - \mathbf{x_0}) = \bar{\mathbf{x}}^T S(t)\bar{\mathbf{x}} \tag{14}$$

Which is in the same form as the optimal cost-to-go function for the LQR. To determine the region of attraction candidate, Equation 15 was analyzed.

$$B_f = \{\bar{\mathbf{x}} | \bar{\mathbf{x}}^T S_f \bar{\mathbf{x}}\} \tag{15}$$

Where S_f is S(t) evaluated at t_f. ρ was set to a small constant of 1, since a small enough constant should work for the SOS program [1].

3) Semidefinite Programming with the Method of Alternations: Due to restrictions in memory for PyDrake, the semidefinite SOS program was written for the problem cases with and without input limits. The approaches solve the SOS program in Equations 16-19.

$$\max_{\rho(t_i), \mathbf{L_i}, \mathbf{V_i}, \bar{\mathbf{u}_i}} \sum_{i=1}^{N} \rho(t_i) \tag{16}$$

$$st\, \mathbf{V_i}\, is\, SOS \tag{17}$$

$$-\dot{\mathbf{V}}_\mathbf{i} + \dot{\rho}(\mathbf{t_i}) + \mathbf{L_i}(\mathbf{V_i} - \rho(t_i))\, SOS \tag{18}$$

$$\mathbf{V_i}(\sum_j \mathbf{e_j}) = \mathbf{V_{guess}}(\sum_j \mathbf{e_j}, \mathbf{t_i}) \tag{19}$$

The is the equality-constrained formulation of the S-procedure. The L terms are multipliers that enforce invariance and Equation 19 is the normalization constraint.

The Time-varying controller design algorithm from Majumdar et al. is shown in Figure 4 [1].

Algorithm 1 Time-Varying Controller Design

Initialize $V_i(\bar{x})$ and $\rho(t_i)$, $\quad \forall i = 1 \ldots N$
$\rho_{prev}(t_i) = 0, \quad \forall i = 1 \ldots N.$
converged = false;
while ¬converged **do**
  STEP 1 : Solve feasibility problem by searching for $L_i(\bar{x})$ and $\bar{u}_i(\bar{x})$ and fixing $V_i(\bar{x})$ and $\rho(t_i)$.
  STEP 2 : Maximize $\sum_{i=1}^{N} \rho(t_i)$ by searching for $\bar{u}_i(\bar{x})$ and $\rho(t_i)$, and fixing $L_i(\bar{x})$ and $V_i(\bar{x})$.
  STEP 3 : Maximize $\sum_{i=1}^{N} \rho(t_i)$ by searching for $V_i(\bar{x})$ and $\rho(t_i)$, and fixing $L_i(\bar{x})$ and $\bar{u}_i(\bar{x})$.
  if $\frac{\sum_{i=1}^{N} \rho(t_i) - \sum_{i=1}^{N} \rho_{prev}(t_i)}{\sum_{i=1}^{N} \rho_{prev}(t_i)} < \varepsilon$ **then**
    converged = true;
  end if
  $\rho_{prev}(t_i) = \rho(t_i), \quad \forall i = 1 \ldots N.$
end while

Fig. 4. Time-varying controller design algorithm.

This algorithm was implemented in PyDrake with the following steps. First, a create_indeterminates function was implemented to compute x and $\bar{x}$. Then, a function to create new mathematical programs was designed. By formulating the Lyapunov and control input candidates in terms of the indeterminate $\bar{x}$ vector, the bilinear optimization problem can be formulated as a series of convex optimization problems. Each step in Figure 4 was formulated as a new mathematical program and solved by fixing all except two variables.

For the input limit SOS program, Figure 4 was applied with the addition of inequality constraints on $\dot{V}$ and $\dot{\rho}$ (Equations 20-22).

$$\dot{\mathbf{V}}_{\mathbf{min}}(\bar{\mathbf{x}}, t) < \dot{\rho}(t) \tag{20}$$

$$\dot{\mathbf{V}}_{\mathbf{max}}(\bar{\mathbf{x}}, t) < \dot{\rho}(t) \tag{21}$$

$$\dot{\mathbf{V}}(\bar{\mathbf{x}}, t) < \dot{\rho}(t) \tag{22}$$

Graphs of S(t), V, and the rocket trajectory were created. The final control input determined by the SOS program was saved and imported as the initial guess in the Julia SCP program.

IV. Evaluation

Figure 5 shows the evolution of the S(t) matrix determined by the LQR equation. The darker values represent small magnitude values in the matrix and light values represent large magnitude values. Since S(t) is proportional to the optimal cost to go, the small magnitude of values at the landing location correctly indicates a low cost-to-go. On the other hand, S(t) evaluated at t_0 has v_y and theta values that are much larger in magnitude, as expected.

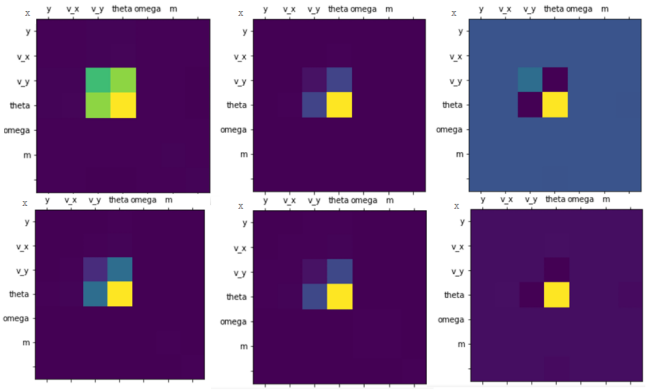

Fig. 5. Evolution of S(t) for the Linear Quadratic Regulator for the first three timesteps (top row) and last three timesteps (bottom row) of the N=20 initial trajectory.

The Lyapunov candidate, determined by Equation 14, is shown in Figure 6. As time progresses, the function develops sharp peaks. Since the LQR cost-to-go equation was used for the Lyapunov candidate, these curves can be intuitively explained; there is a fairly level cost-to-go at t_0, when all locations are sufficiently far from the final destination. As t_f is approached, the value of the cost-to-go is only close to zero if you are near the landing site.

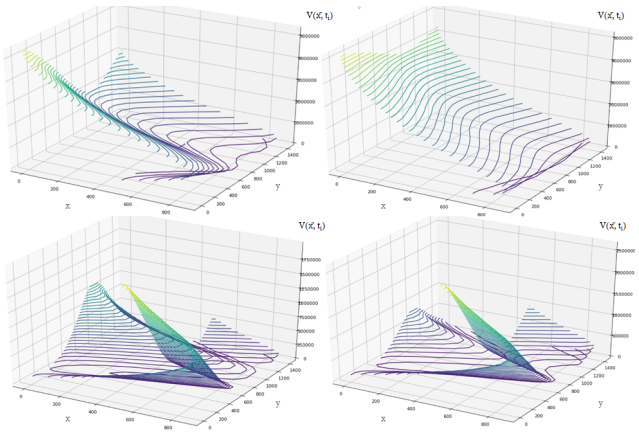

Fig. 6. The Lyapunov function guess for the first two timesteps (top row) and last two timesteps (bottom row) of the $N = 7$ initial trajectory.

Figures 7-12 show the landing trajectory, states, and parameters for the rocket landing problem with the Lyapunov-generated initial guess strategy. A solution was successfully computed using the Lyapunov-generated initial guess, all state and control constraints were met, and the PTR solver converged in only six steps.

Table 2 shows the elapsed time for the straight-line method and Lyapunov-based initial guesses to converge in the PTR

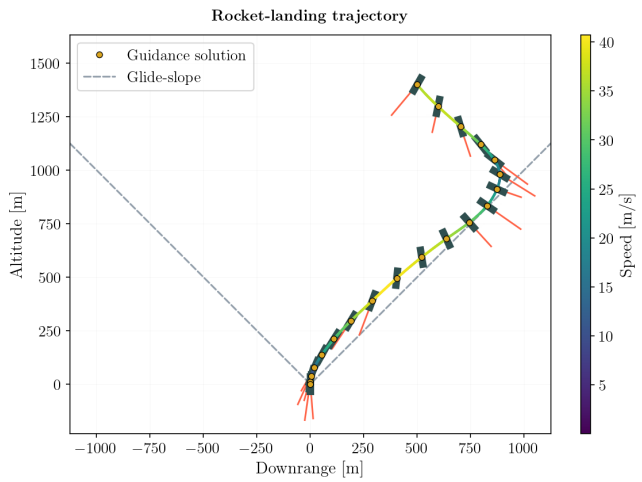

Fig. 7. The rocket landing trajectory for the Sequential Convex Programming solution to the Lyapunov-based initial guess.

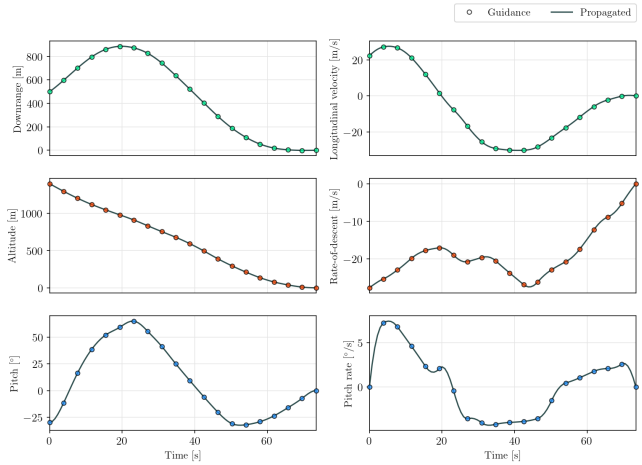

Fig. 8. The system states for the Sequential Convex Programming solution to the Lyapunov-based initial guess.

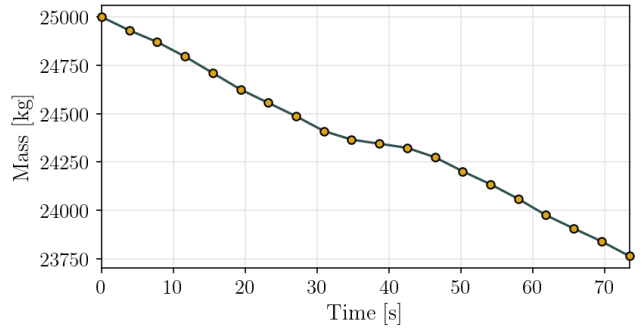

Fig. 9. The mass for the Sequential Convex Programming solution to the Lyapunov-based initial guess.

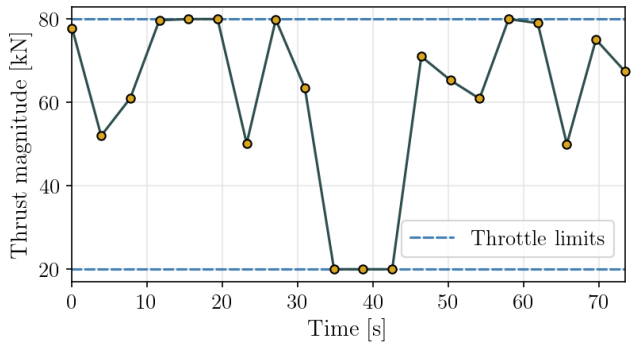

Fig. 10. The thrust magnitude for the Sequential Convex Programming solution to the Lyapunov-based initial guess.

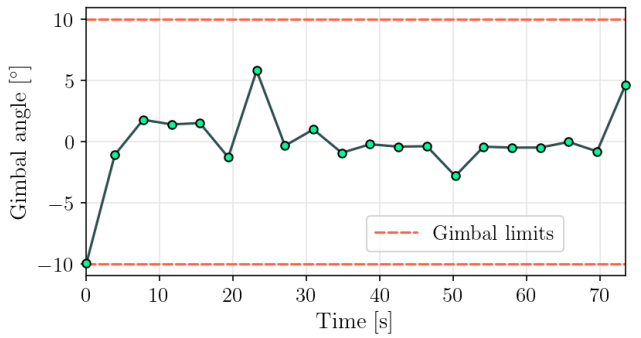

Fig. 11. The gimbal angle for the Sequential Convex Programming solution to the Lyapunov-based initial guess.

solver after four runs. Since the standard deviations for both methods are quite small, it can be concluded that the Lyapunov-based initial guess strategy resulted in about a 6% runtime reduction.

V. Discussion

From completing this project, I was most surprised by the methods of determining the Lyapunov function and region of attraction-maximizing control input taking the most time for the project. In addition, as input constraints were added, the

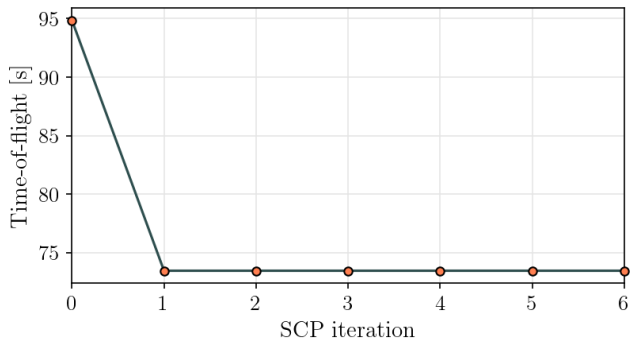

Fig. 12. The time-of-flight for the Sequential Convex Programming solution to the Lyapunov-based initial guess.

TABLE II
CONVERGENCE TIMES FOR SCP INITIALIZATION METHODS

Convergence Times		
Initialization Method	***Elapsed Time / s***	***Standard Deviation / s***
straight-line	3.3153	6.8762e-3
Lyapunov	3.1116	6.6380e-3

number of alternations and time it took to solve the SOS grew significantly.

This project also increased my skills in PyDrake syntax and formulating semidefinite programs. I discovered that only x should be defined as indeterminate and which types of polynomials were required for decision variables. I was also not fully aware of the time-varying LQR's use in formulating Lyapunov function candidates. From reading papers as well as developing the semidefinite programming functions, I gained an intuitive understanding of how semidefinite SOS programs can search along trajectories.

While I have learned a lot from this project, there were a significant amount of memory errors that I encountered along with some initial confusion when implementing linearization. With the amount of memory on the PyDrake notebook, I was only able to get a solution for the first alternation of the program without input constraints. When making the algorithms no longer time-varying, I was able to get solutions for the program. This suggests that the memory issues could be improved when run on a different system. I had also attempted to use the linearization that approximates sin and cos as variables. Unfortunately, the implementation with the control input became convoluted. Therefore, linearization was conducted with the first order Taylor series expansion instead.

Since this approach was limited by memory, the decrease in runtime for the SCP program to converge when the Lyapunov initial guess is applied should be further assessed with a system that has greater computational resources. The SOS algorithm with input constraints should be run until convergence and run along multiple trajectory inputs to determine if the Lyapunov initial guess dominates the straight-line initial guess in most SCP programs for rocket landing problems.

While this project is relevant to my research, Computationally Efficient Large Divert Guidance, it implements methods from class, Lyapunov theory, SOS, LQR, sequential convex programming, the S-procedure, and method of alternations, to solve the research-relevant problem of fast convergence for SCP-based powered descent guidance algorithms. Without this class, I would not have considered this approach to improving the computational efficiency of SCP methods.

VI. CONCLUSION

By applying a Lyapunov-based initial guess for the rocket landing SCP problem, a 6% runtime reduction was attained. While promising, additional Lyapunov-based guesses must be tested to determine if this method always dominates the straight line method. In addition, the full method of alternations must be run on a system with enough memory to allow the algorithm to converge on the maximum region of attraction. The method of determining an initial guess with SOS programming is also limited with increasing dimensionality; the number of constraints continue to increase and the time required to converge to the maximum region of attraction solution also grows. To determine if this method is viable for future guidance algorithms, work must be completed to expand the current implementation into three dimensions. If feasible, Lyapunov-based initial guess strategies could be promising methods for reducing online runtime for SCP algorithms.

VII. CODE AND SIMULATION VIDEOS

juliabriden.com/projects.

REFERENCES

[1] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in Robotics and Automation (ICRA), 2013 IEEE International Conference on, 2013, pp. 4054–4061.

[2] Açıkmeşe, Behçet Casoliva, Jordi Carson, John Blackmore, Lars. (2012). G-FOLD: A Real-Time Implementable Fuel Optimal Large Divert Guidance Algorithm for Planetary Pinpoint Landing. LPI Contributions. 4193-.

[3] Conner, Monroe. "JPL Completes G-FOLD Tests on Masten's Xombie." NASA, NASA, 9 Mar. 2015, tinyurl.com/y25nnkn8.

[4] D. Malyuta, T. Reynolds, M. Szmuk, and T. Lew, "Convex Optimization for Trajectory Generation," 2021. Available: https://doi.org/10.48550/arxiv.2106.09125

[5] D. Dueri, Real-time Optimization in Aerospace Systems. Ph.D., University of Washington, 2018.

[6] Mhaskar, El-Farra, Christofides, Stabilization of Nonlinear Systems with State and Control Constraints using Lyapunov-based Predictive Control, Systems Control Letters, Volume 55, Issue 8, 2006, Pages 650-659, ISSN 0167-6911.

[7] "NASA Technology Roadmaps TA 9: Entry, Descent, and Landing Systems." NASA, NASA, July 2015, tinyurl.com/y6y8x4lw.

[8] Reynolds, Taylor Malyuta, Danylo Mesbahi, Mehran Açıkmeşe, Behçet Carson, John. (2020). A Real-Time Algorithm for Non-Convex Powered Descent Guidance. 10.2514/6.2020-0844.

[9] T. Kim, P. Elango, D. Malyuta, and B. Acikmese. Guided Policy Search using Sequential Convex Programming for Initialization of Trajectory Optimization Algorithms. arXiv, 2021, https://arxiv.org/abs/2110.06975.

[10] T. P. Reynolds, M. Szmuk, D. Malyuta, M. Mesbahi, B. A¸cıkme¸se, and J. M. Carson III, "Dual Quaternion Based 6-DoF Powered Descent Guidance with State-Triggered Constraints," arXiv e-prints, 2019. arXiv:1904.09248.