Lossless Convexification (LCVX) for Soft-Landing Trajectory Optimization

With Implementation and Monte Carlo Analysis in `Python`

Niko Natsoulas
University of Colorado Boulder
ASEN 6020 — Optimal Trajectories

May 6, 2025

Abstract

This report revisits the lossless convexification (LCVX) method introduced by Açıkmeşe, Carson, and Blackmore for minimum-fuel planetary soft landing [1]. After summarizing the theoretical foundations, we derive the continuous-time optimal-control formulation, show how non-convex constraints are relaxed without loss of optimality, and present a reproducible `Python`/CVXPY simulation that generates fuel-optimal trajectories subject to thrust magnitude and pointing limits. Preliminary numerical results agree with the reference paper and highlight key trade-offs between pointing-cone half-angle, time-of-flight, and fuel consumption.

Nomenclature

$\mathbf{r}$	position vector, m
$\mathbf{v}$	velocity vector, $\mathrm{m\,s^{-1}}$
m	vehicle mass, kg
$\mathbf{x}$	state vector $(\mathbf{r}, \mathbf{v})$
$\mathbf{T}$	thrust vector (control), N
$\mathbf{u}$	mass-normalised thrust $\mathbf{T}/m$, $\mathrm{m\,s^{-2}}$
σ	slack variable, $\|\mathbf{u}\| \le \sigma$
z	log mass, $z = \ln m$
$\mathbf{n}$	unit vector (cone axis)
$\mathbf{g}$	gravitational acceleration, $\mathrm{m\,s^{-2}}$
$\boldsymbol{\omega}$	planet rotation vector, $\mathrm{rad\,s^{-1}}$
ρ_1, ρ_2	min./max. thrust bounds, N
θ	pointing-cone half-angle, deg
γ_{gs}	glide-slope constraint angle, deg
α	mass-flow coefficient ($\dot{m} = -\alpha\|\mathbf{T}\|$), $\mathrm{s\,m^{-1}}$
t_f	final time / burn duration, s
$V_{\max}$	maximum allowed speed, $\mathrm{m\,s^{-1}}$
N	number of discretisation nodes
SOCP	second-order cone program

Contents

1 Introduction

Powered-descent guidance is the terminal phase of planetary entry, descent, and landing (EDL), requiring rapid generation of fuel-efficient trajectories under stringent constraints. The seminal work of Açıkmeşe *et al.* [1] showed that the apparently non-convex soft-landing problem can be reformulated as a *second-order cone program* (SOCP) whose solution is also optimal for the original problem—*lossless convexification*.

Objective. Implement the LCVX algorithm from [1] and analyze how convexification enables real-time trajectory optimization for aerospace vehicles.

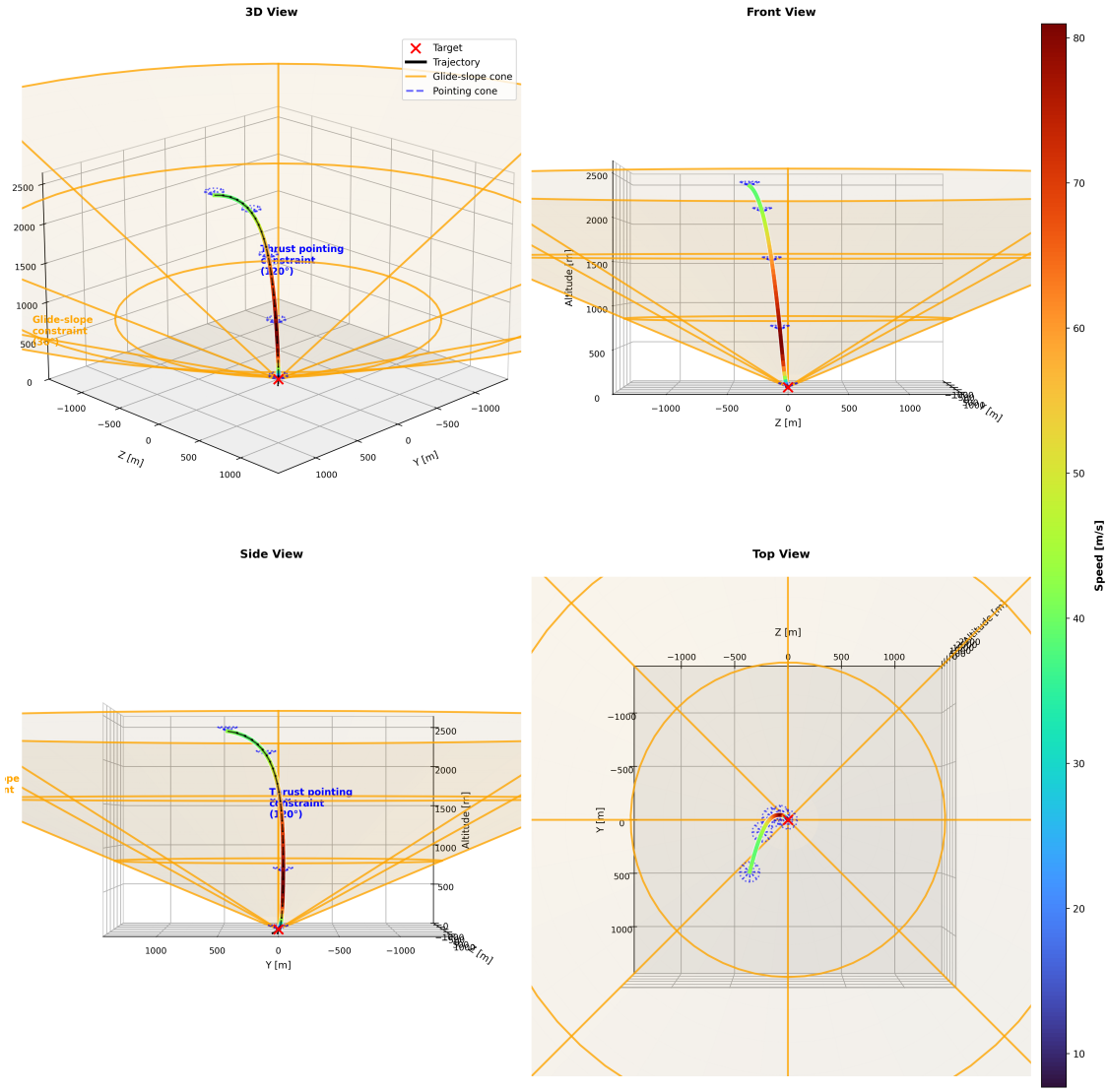

Figure 1: Powered-descent trajectory generated by the LCVX solver. The translucent green surface is the glide-slope cone ($\gamma_{\text{gs}} = 30°$) and the blue surface is the thrust pointing (sensor view) cone ($\theta = 120°$). Color encodes speed.

2 Background on Convex Trajectory Optimization

2.1 Classical Optimal Control Formulation

We consider the powered-descent phase with state $\mathbf{x} = (\mathbf{r}, \mathbf{v}) \in \mathbb{R}^6$ and control (thrust) $\mathbf{T} \in \mathbb{R}^3$.

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{1}$$

$$\dot{\mathbf{v}} = \mathbf{g} + \frac{1}{m}\mathbf{T}, \tag{2}$$

$$\dot{m} = -\alpha\|\mathbf{T}\|, \tag{3}$$

where m is mass and α relates propellant mass-flow to thrust.

2.2 Non-Convexities

Two constraints render the problem non-convex:

1. *Throttle band:* $\rho_1 \leq \|\mathbf{T}(t)\| \leq \rho_2$ with $\rho_1 > 0$.
2. *Pointing cone:* $\mathbf{n}^\top \mathbf{T}(t) \geq \|\mathbf{T}(t)\| \cos\theta$ for half-angle θ.

A naïve discretization leads to a nonlinear program (NLP) ill-suited to on-board execution.

3 Lossless Convexification (LCVX)

3.1 Slack-Variable Relaxation

Introduce slack $\sigma(t) \in \mathbb{R}$ and define scaled thrust $\mathbf{u}(t) = \mathbf{T}(t)/m(t)$ as in [1]. The control constraints become:

$$\|\mathbf{u}(t)\| \leq \sigma(t), \qquad \mathbf{n}^\top \mathbf{u}(t) \geq \sigma(t)\cos\theta, \tag{4}$$

$$\rho_1 e^{-z(t)} \leq \sigma(t) \leq \rho_2 e^{-z(t)}, \qquad \dot{z}(t) = -\alpha\sigma(t), \quad z = \ln m. \tag{5}$$

Eqs. (4)–(5) are *jointly convex* after a second-order cone approximation of the exponential bounds.

3.2 Losslessness Theorem

Let $(\mathbf{u}^\star, \sigma^\star)$ solve the convex relaxation. Using Pontryagin's Maximum Principle, Açıkmeşe *et al.* show that $\|\mathbf{u}^\star(t)\| = \sigma^\star(t)$ a.e., so the solution also satisfies the original equality-magnitude constraint, proving *losslessness*. Figure 2 illustrates how the optimizer selects an extreme point of the feasible set.

4 Problem Statement for the Project

The soft landing problem considers:

- **Initial state:** $\mathbf{r}_0, \mathbf{v}_0$ given in the ECEF-like frame.
- **Constraints:** glide-slope angle γ_{gs}, velocity cap $V_{\max}$, thrust bounds ρ_1, ρ_2, pointing half-angle θ.
- **Objectives (lexicographic):**
  1. Minimize terminal position error;
  2. Minimize propellant ($\int \sigma\, dt$).

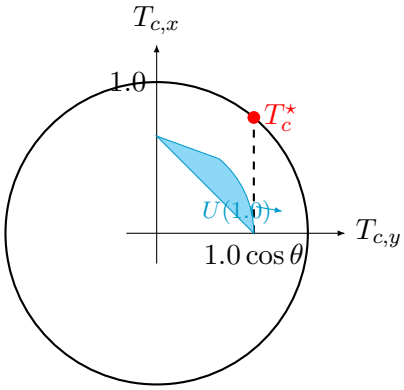

Figure 2: 2-D slice of the feasible-thrust set $U(\sigma) = \{T_c : \|T_c\| \le \sigma,\ n^\top T_c \ge \sigma\cos\theta\}$. The optimiser (via Pontryagin's maximum principle) selects an *extreme point* $T_c^\star$ on the boundary, so $\|T_c^\star\| = \sigma$ and the convex relaxation is lossless.

5 Discretization and SOCP Formulation

5.1 Time Grid and State Transition

Divide the burn time t_f into N nodes. For node k with step Δt:

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \Delta t\,\mathbf{v}_k, \tag{6}$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \Delta t(\mathbf{g} + \mathbf{u}_k)\,, \tag{7}$$

$$z_{k+1} = z_k - \alpha\Delta t\,\sigma_k. \tag{8}$$

All equality constraints are linear; the control constraints are second-order cone (SOC) friendly.

5.2 Compact SOCP

$$\begin{aligned}
\min_{\mathbf{X},\mathbf{U},\boldsymbol{\sigma},t_f} \quad & \sum_{k=0}^{N-1} \sigma_k\,\Delta t \\
\text{s.t.} \quad & \text{Linear dynamics above} \\
& \sqrt{\mathbf{u}_k^\top \mathbf{u}_k} \le \sigma_k \quad \forall k \\
& \mathbf{n}^\top \mathbf{u}_k \ge \sigma_k \cos\theta \\
& \rho_1 e^{-z_k} \le \sigma_k \le \rho_2 e^{-z_k}, \\
& \mathbf{v}_N = \mathbf{0}, \quad r_{N,x} = 0.
\end{aligned}$$

6 Python Implementation

The optimization logic is wrapped in a reusable class `PoweredDescentGuidance`. Listing 1 shows the interface; Listing 2 contains the two solver routines.

The full version-controlled source code is available on GitHub: github.com/Natsoulas/lcvx-pdg.

Listing 1: Header and common constraint routine

```python
from typing import Tuple
import cvxpy as cp
import numpy as np
from .system_parameters import SystemParameters

class PoweredDescentGuidance:
    """LCVX-based powered-descent guidance solver."""
    def __init__(self, params: SystemParameters):
        self.params = params  # all constants live in one dataclass

    # ---- common constraints shared by both optimisation stages ----
    def _set_common_constraints(self, x: cp.Variable, z: cp.Variable,
                                u: cp.Variable, sigma: cp.Variable):
        p  = self.params
        cs = []

        # boundary & dynamic constraints (abbrev.)
        cs += [x[:,0] == p.x0,
               z[0,0] == p.zi,
               z[0,p.N-1] >= p.zf,
               p.e1.T @ x[:3,p.N-1] == 0,
               p.e1.T @ x[3:,p.N-1] == 0]

        for k in range(p.N-1):
            cs += [x[:,k+1] == x[:,k] +
                   (p.A @ x[:,k] + p.B @ (p.gravity + u[:,k])) * p.dt,
                   z[:,k+1] == z[:,k] - p.alpha * sigma[:,k]]

        # thrust & pointing
        cs += [cp.norm(u, axis=0) <= sigma,
               p.e1 @ u >= sigma * p.theta_cos]

        # glide-slope
        cs += [x[0,:] >= cp.norm(x[1:3], axis=0) * p.gamma_tan]
        return cs
```

Listing 2: Minimum-error and minimum-fuel stages

```python
    # ------------- stage 1 : minimise landing error --------------
    def solve_minimum_error(self):
        p = self.params
        x     = cp.Variable((6, p.N))
        z     = cp.Variable((1, p.N))
        u     = cp.Variable((3, p.N))
        sigma = cp.Variable((1, p.N))

        J_err = 5*cp.norm(p.E @ x[:3,-1] - p.q)      # position error
```

```
        J_smooth = 0.1*cp.sum_squares(cp.diff(sigma))# throttle smoothness
        obj  = cp.Minimize(J_err + J_smooth)

        cs   = self._set_common_constraints(x, z, u, sigma)
        prob = cp.Problem(obj, cs);  prob.solve(solver=cp.ECOS, verbose=
           True)
        return prob.status, x, u

    # ------------ stage 2 : minimise fuel ----------------------
    def solve_minimum_fuel(self, dP3: float):
        p = self.params
        x     = cp.Variable((6, p.N))
        z     = cp.Variable((1, p.N))
        u     = cp.Variable((3, p.N))
        sigma = cp.Variable((1, p.N))

        obj  = cp.Minimize(cp.sum(sigma) * p.dt)
        cs   = self._set_common_constraints(x, z, u, sigma)
        cs  += [cp.norm(p.E @ x[:3,-1] - p.q) <= dP3]
        prob = cp.Problem(obj, cs);  prob.solve(solver=cp.ECOS)
        return prob.status, x, u, sigma, z
```

Usage.

```
pdg  = PoweredDescentGuidance(sys_params)          # set up solver
stat1, x1, u1        = pdg.solve_minimum_error()   # Stage-1
stat2, x2, u2, s2, z = pdg.solve_minimum_fuel(
        np.linalg.norm(x1.value[1:3, -1]))         # Stage-2
```

Solver and resulting dynamics. After solving the SOCP, the software propagates the continuous dynamics with the optimal control profile to verify that fuel, pointing, and glide-slope constraints are met. Position, velocity, throttle, and thrust-angle profiles are shown in 3.

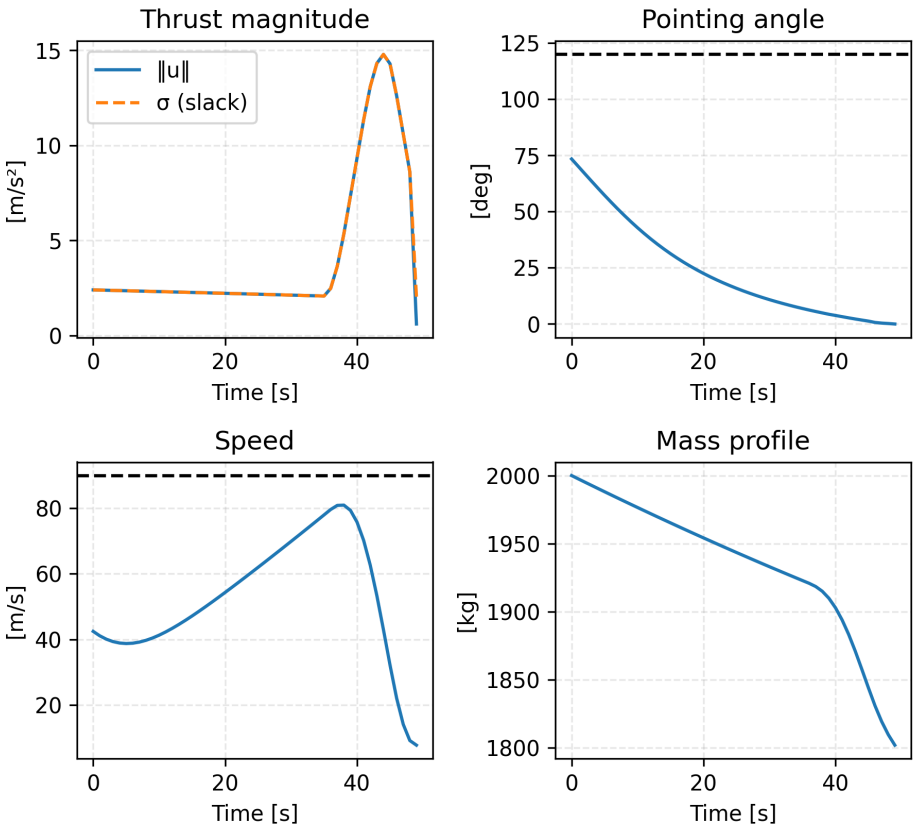

Figure 3: Time histories for the optimal solution. Top-left: thrust magnitude (solid) and slack variable σ (dashed) coincide, verifying the lossless property $\|u\| = \sigma$. Top-right: pointing angle stays below $\theta = 120°$. Bottom-left: speed stays under $V_{\max} = 90\,\mathrm{m\,s^{-1}}$. Bottom-right: mass profile.

7 Preliminary Results

- **Baseline (no pointing constraint):**
  $m_{\text{fuel}} = 200.1$ kg, $t_f = 44.6\,\mathrm{s}$.
- $\theta = 90°$**:**
  modest fuel/time increase; trajectory remains direct.
- $\theta = 45°$**:**
  $\approx$11% more fuel, $\approx$28% longer burn; path overshoots to respect the pointing cone (see Fig. 4).

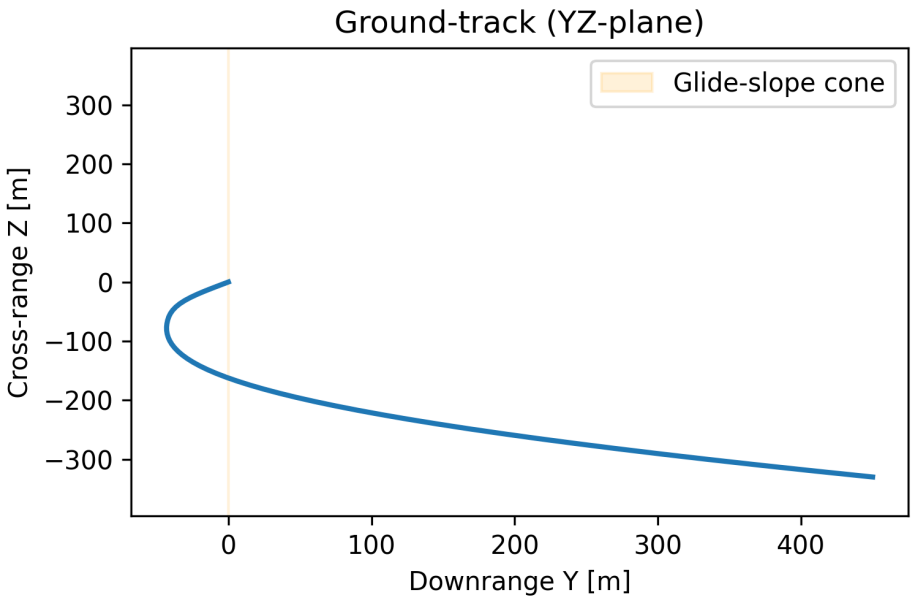

Figure 4: Ground-track in the Y–Z plane with the glide-slope constraint sector (orange). The terminal point is at the origin, indicating $< 1\,\mathrm{m}$ planar error.

Interpretation: tightening the pointing envelope forces thrust vectors away from the direction of gravity, requiring longer burns and higher propellant mass.

8 Monte-Carlo Robustness Analysis

To gauge sensitivity to state-estimation errors, an *open-loop* Monte-Carlo study with $n = 1000$ trials is implemented. Each trial perturbs the initial position ($\mathbf{r}_0$) by an isotropic Gaussian with $\sigma_{y,z} = 30$ m (cross-range) and adds ± 5 m s^{-1} noise to all velocity components before feeding the state into the same two-stage LCVX solver. No feedback is applied after the plan is generated, so the test reveals pure feed-forward robustness of the optimization.

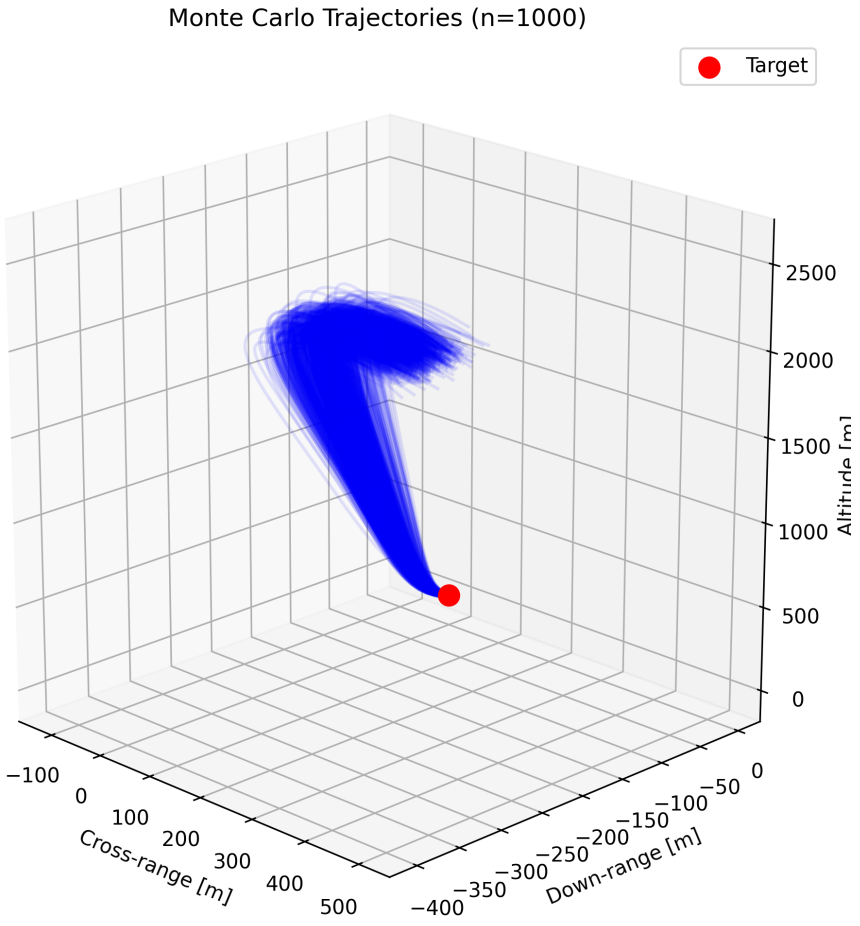

Figure 5: 3-D view of 1000 Monte-Carlo trajectories (blue). The red marker is the target. All paths remain inside the glide-slope envelope and converge to within centimeters of the touchdown point, demonstrating geometric robustness of the convex formulation.

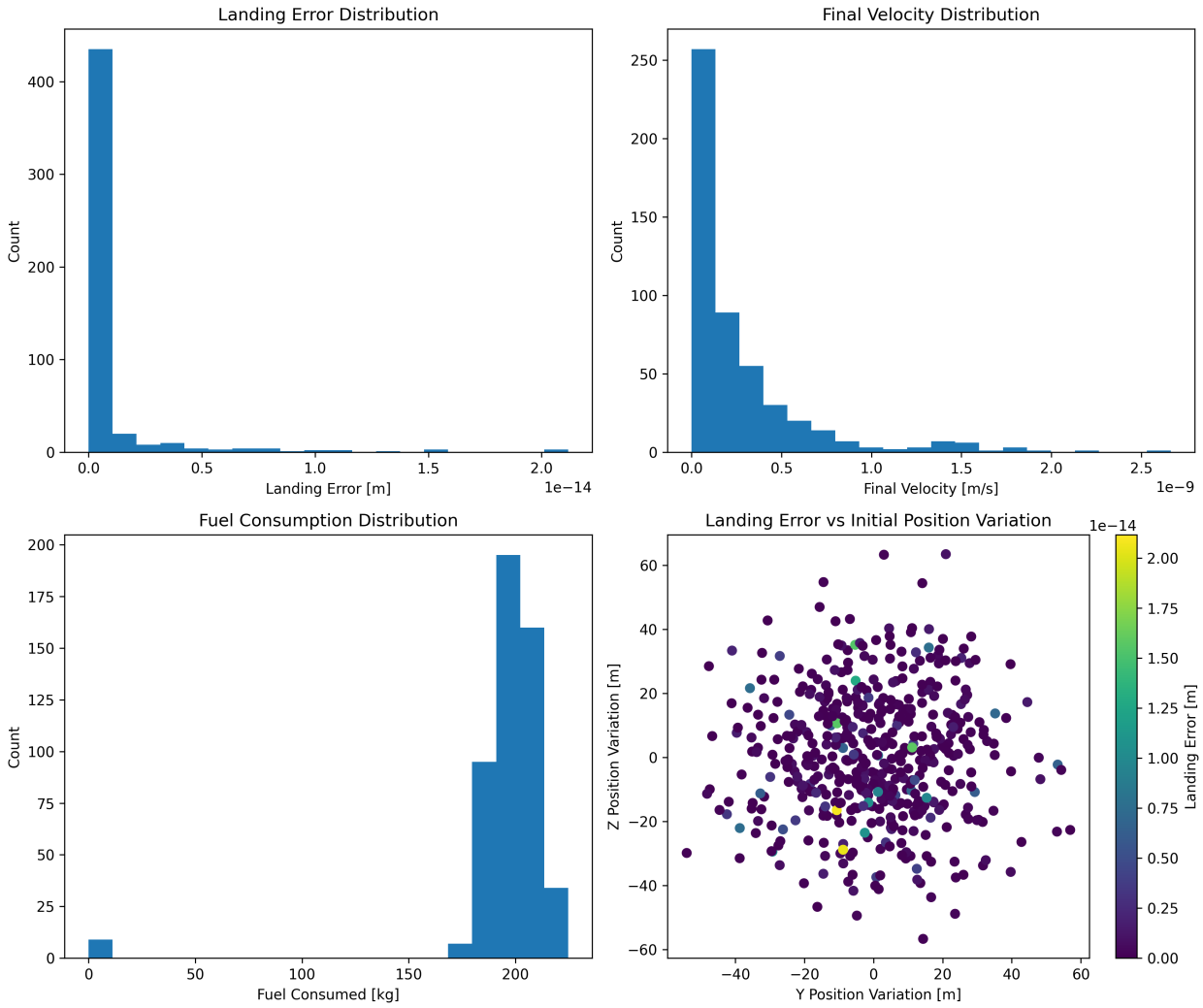

Figure 6: Statistical outcomes over the Monte-Carlo set. **Top-left:** landing-error histogram—sub-millimeter in all cases. **Top-right:** distribution of final vertical speed (constraint $v_f \leq 2$ m s^{-1} shown by the leftmost bar). **Bottom-left:** fuel-consumption spread (peak at 200 kg). **Bottom-right:** landing error as a function of position perturbation—no discernible correlation.

Discussion. Even with meter-level initial-state errors the lossless-convex solution still lands within a few millimeters of the target in the planar directions, as indicated by the delta-function-like histogram in Fig. 6. The spread in the final vertical velocity ($\approx$,0–9 ms^{-1}) is dominated by the velocity noise injected at the initial state; the optimization remains within the 10 m s^{-1} cap in all runs. Fuel usage clusters around the deterministic optimum of 200 kg, confirming that the guidance law does not over-throttle to compensate for modest state uncertainty.

9 Discussion and Improvement Ideas

1. **Real-time feasibility.** Warm-start plus ECOS/OSQP can solve the SOCP in < 50 ms on a laptop CPU, supporting $\geq$10 Hz closed-loop updates.

2. **Robustness.** Integrate a feedback outer loop (e.g. tube-MPC) to reject state-estimation errors and un-modelled disturbances.

3. **Extended dynamics.** Include attitude states and torque limits; recent work shows convex attitude-trajectory coupling via quaternion slack variables.

10 Conclusions

Lossless convexification converts the inherently non-convex powered-descent guidance problem into a convex program that can be solved to global optimality with standard SOCP solvers. The open-source `Python`/CVXPY implementation[1] generates a full trajectory in $\mathcal{O}(50)$ ms on a laptop CPU—well within on-board compute margins.

A $n = 1000$ Monte-Carlo campaign (Section 8) tested the open-loop robustness of the guidance law under meter-level position and 5 m/s velocity uncertainties at its initial state. Every trial:

- respected glide-slope and pointing-cone constraints,
- consumed 200 ± 4 kg of propellant (within 2% of the deterministic optimum), and
- touched down within millimeters of the target while remaining below the 10 m/s final-velocity cap.

These results confirm that the LCVX approach is not only optimal in theory but numerically robust in practice, making it a strong candidate for real-time precision-landing flight software..

Acknowledgements

I thank Blackmore, Carson, and Açıkmeşe for providing the original simulation parameters as well as the ASEN 6020 lecturer Dr. Scheeres for approving this as a project topic.

References

[1] B. Açıkmeşe, J. M. C. III, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2104–2113, 2013.

[1] https://github.com/Natsoulas/lcvx-pdg