Optimal Control for Systems with non-Euclidean State Spaces

Presenter: Spencer Kraisler

RAIN Lab
William E. Boeing Department of Aeronautics & Astronautics
University of Washington

November 2024

Table of Contents

Who Am I?

- Seattle
- Hobbies: curling, shotokan karate (2nd degree black belt), reading
- PhD candidate at UW Aerospace
- Passion for executing complex math solutions to practical engineering applications; establishing human presence in space

Figure: My lab

Research Interests

Designing controllers through optimization and (Riemannian geometry)

- Solve the LQG problem through *Riemannian optimization* methods
- **Trajectory optimization for systems whose state spaces are smooth manifolds**
- Satellite constellations
- Developing quadcopter hardware testbed (Python, ROS)
- Distributed consensus for systems whose state spaces are smooth manifolds

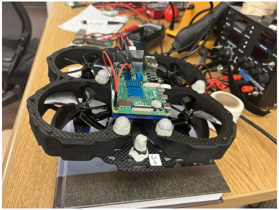


Figure: DROPLET

Table of Contents

Optimal Control over Smooth Manifolds

$$\min_{\mathbf{x},\mathbf{u}} J(\mathbf{x},\mathbf{u}) = \sum_{k=0}^{T-1} c(x_k, u_k) + h(x_T)$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k)$$

$$s(x_k, u_k) \leq 0$$

What if the state space is (implictly) a *smooth manifold*? Is that *important*?

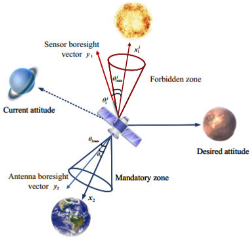


Figure: Constrained attitude trajectory problem $SO(3) \times \mathbb{R}^3$

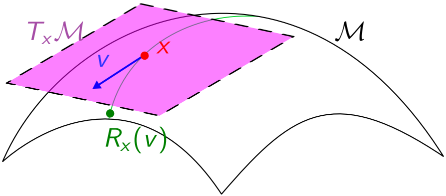
Figure: Powered descent of a rocket $SE(3) \times \mathbb{R}^6$

Figure: Mobile manipulator (Lie group)

Smooth manifold $\mathcal{M}$

- A space which is *locally Euclidean*
  - Unit quaternions $\mathcal{Q}$
- Compatible with calculus
- Tangent space $T_x\mathcal{M} \cong \mathbb{R}^n$
  - $T_q\mathcal{Q} = \{v \in \mathbb{R}^4 : q^T v = 0\} \cong \mathbb{R}^3$

Retraction: $R_x : T_x\mathcal{M} \to \mathcal{M}$



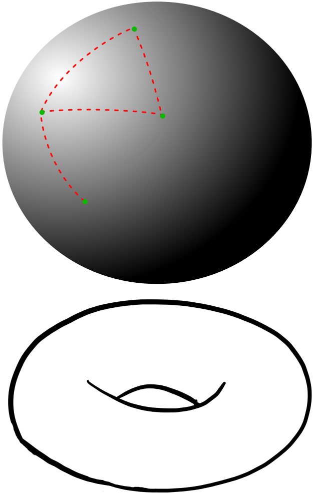


Figure: Smooth manifold examples

What is Successive Convexification (SCvx)?

- 1st-order solver for non-convex trajectory optimization problems
- allows infeasible initial trajectories
- highly scalable
- How SCvx works:
  1. Start with infeasible trajectory $(\mathbf{x}, \mathbf{u})$
  2. Solves a “zoomed-in” convex sub-problem with linearizing dynamics and constraints, obtaining trajectory perturbations $(\boldsymbol{\eta}, \boldsymbol{\xi})$
  3. adds perturbations to trajectory: $\mathbf{x}^+ \leftarrow \mathbf{x} + \boldsymbol{\eta}$, $\mathbf{u}^+ \leftarrow \mathbf{u} + \boldsymbol{\xi}$
  4. Repeat

Shortcomings:

1. Depends on state-space parameterization
2. Redundant dimensions
3. Next trajectory won't be on manifold

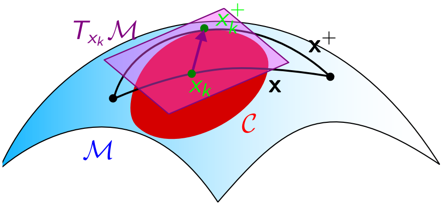


Figure: Visualization of perturbing a trajectory along a tangent space.

Goals:

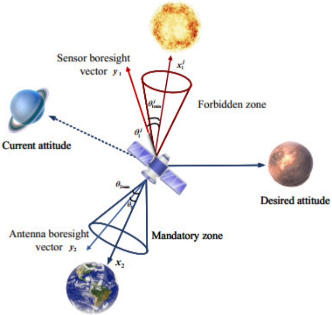
1. Perform SCvx in a way that is intrinsic to the manifold, not the parameterization
2. Ensure perturbations are tangent to the manifold surface
3. Use a retraction to "add" the perturbations to the trajectory in an intrinsic way

Problem Set-up

1. Write Python script for SCvx using only CVXPY and NumPy
2. Simple problem: rotate $q_k \in \mathcal{Q}$ with angular velocity ω_k for (fixed) Δt seconds:

$$q_{k+1} = q_k \otimes \exp(\Delta t \cdot \omega_k)$$

3. Constraints: keep-out zone
4. Visualization script (plots the boresight vector trajectory and the keepout zone)



Next step: achieve the goals of developing an intrinsic SCvx methodology...

Linearizing Dynamics over a Smooth Manifold

$f : \mathcal{M} \to \mathcal{M}$

Best linear approximation of f at x: the differential $df_x : T_x\mathcal{M} \to T_{f(x)}\mathcal{M}$

Idea: if $\mathcal{M} \subset \mathbb{R}^m$ and $\dim(\mathcal{M}) = n < m$, then

$$[Df(x)] \in \mathbb{R}^{m\times m}$$
$$df_x : v \in T_x\mathcal{M} \mapsto \mathrm{Proj}_{T_x\mathcal{M}}[Df(x)v]$$
$$[df_x] \in \mathbb{R}^{n\times n}$$

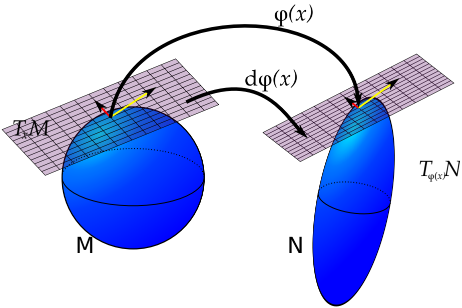


Figure: Visualization of the differential

Intrinsic SCvx: same methodology, but use the differential instead of the Jacobian for linearizing the dynamics and constraints

Next steps:

1. implement intrinsic SCvx methodology in Python and CVXPY
2. Compare SCvx and intrinsic SCvx

Experiments and Results

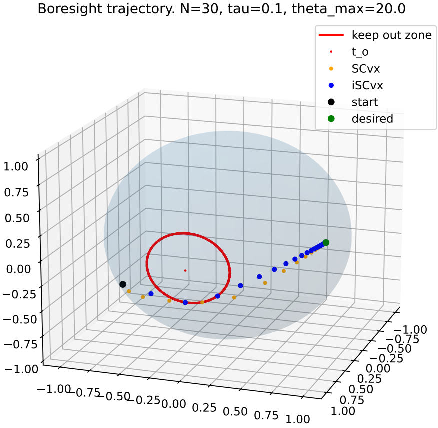


Figure: Trajectory of boresight vector, red circle is keep-out zone

N=30, tau = .1, eps_tol = 1e-3	Ave Iteration ratio	Iteration ratio std	Ave Cost ratio
theta_max = 10	0.896	0.117	0.956
theta_max = 20	0.937	0.112	0.92
theta_max = 30	0.932	0.122	0.893
N=60, tau = .05, eps_tol = 1e-3	**Ave Iteration ratio**	**Iteration ratio std**	**Ave Cost ratio**
theta_max = 10	0.646	0.121	0.955
theta_max = 20	0.742	0.1	1
theta_max = 30	0.732	0.093	0.994
N=30, tau = .1, eps_tol = 1e-5	**Ave Iteration ratio**	**Iteration ratio std**	**Ave Cost ratio**
theta_max = 10	0.646	0.121	0.954
theta_max = 20	0.64	0.128	0.922
theta_max = 30	0.651	0.142	0.896
N=60, tau = .05, eps_tol = 1e-5	**Ave Iteration ratio**	**Iteration ratio std**	**Ave Cost ratio**
theta_max = 10	0.428	0.125	0.988
theta_max = 20	0.458	0.144	0.997
theta_max = 30	0.445	0.124	1.001

Future Work and Conclusion

- Obtain similar convergence guarnatees for intrinsic SCvx, as was obtained for SCvx
- Test on more complicated problem set-ups, such as powered descent of a rocket
  - Plan: Write library in Python to handle even more general problem-setups, possibly include a Riemannian auto-differentiator (ManOpt)
- Submit journal paper
- Implement some online MPC on quadcopter testbed when finished