A SEQUENTIAL CONVEX PROGRAMMING APPROACH FOR FREE-FLYING ROBOT TRAJECTORY GENERATION

Abhishek Cauligi[1], Andrew Bylard[1], Riccardo Bonalli[2], Marco Pavone[1]

[1]*Department of Aeronautics and Astronautics, Stanford University, 496 Lomita Mall, Stanford, CA 94305, USA*
[2]*ONERA - The French Aerospace Lab, 8 Chemin de la Hunière, 91120 Palaiseau, France*
[1]*E-mails: acauligi@stanford.edu, bylard@stanford.edu, riccardo.bonalli@gmail.com, pavone@stanford.edu*

ABSTRACT

Small, assistive free-flying robots have received increased attention in recent years as a platform for furthering robotic technologies and capabilities for operations inside and outside of spacecraft. This work seeks to develop a trajectory refinement method that considers six degree-of-freedom motion for a free-flying robot while satisfying the group structure constraints for a rigid body evolving on $SE(3)$. Simulation results are presented for a sequential convex programming based approach for generating locally optimal trajectories that refine and improve the quality of an initially dynamically feasible solution.

1 INTRODUCTION

Robotic spacecraft have gathered significant interest in recent years, due to advances in robotic technologies and increasing accessibility to testing opportunities in space. Despite the tremendous and growing need for major spacecraft operations such as satellite servicing (e.g. repair and refueling) or management of large orbital debris [1], such operations are rare and in many cases undemonstrated due to the excessive cost and expense of crewed missions for such tasks, as well as limited validation examples on the robotic side. Small, assistive free-flying robots (AFFs), such as the Astrobee shown in Figure 1, have emerged as a promising platform for demonstrating such autonomous capabilities. These robots could work both alongside astronauts or independent of astronaut supervision, performing various logistics, monitoring, and maintenance tasks, and saving valuable crew time on-board the International Space Station (ISS).

A key enabler for such operational capabilities is safe, autonomous navigation. Simple tasks such as retrieving tools from different modules on the ISS or repositioning cameras to record astronaut activities require the ability to plan dynamically feasible trajectories in real-time while avoiding collisions.

Of special interest for AFFs is planning across full six degree-of-freedom (DoF) motion on the special Euclidean group $SE(3)$:

$$SE(3) = \left\{ \begin{pmatrix} \mathbf{R} & \mathbf{r} \\ 0 & 1 \end{pmatrix} \middle| \; \mathbf{R} \in SO(3), \mathbf{r} \in \mathbb{R}^3 \right\} \quad (1)$$

Elements of $SE(3)$ represent rigid body poses using the combination of translation $\mathbf{r}$ and a rotation $\mathbf{R}$. The special orthogonal group of $SO(3)$ rotation matrices poses a particular challenge to planning, as it results in non-convex dynamics, on a manifold having a geometric structure that cannot be neglected.

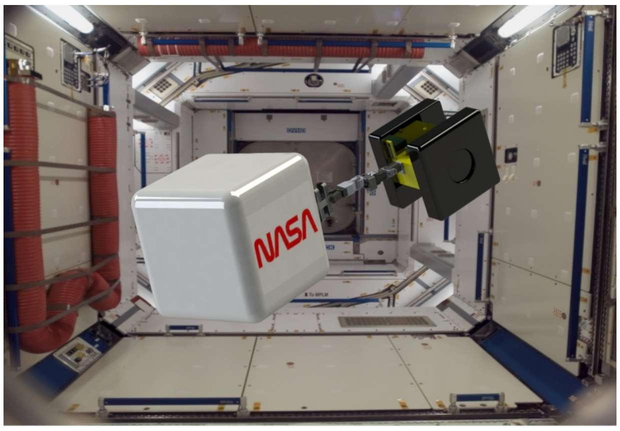


Figure 1 Example operations of an AFF on the ISS [2].

In this work, we present a trajectory refinement approach using sequential convex programming to locally improve the quality of plans generated by a sampling-based motion planning algorithm. We expect this tool to be used as a part of a short-range motion planning algorithm in which rotational degrees-of-freedom must be considered in order to increase the likelihood of finding a solution in a cluttered environment.

Paper contribution: The key contribution of this paper is a sequential convex programming-based approach for generating locally optimal trajectories (for any convex cost function) given an initially dynamically feasible trajectory. A convexification method is presented that allows for the optimization to adhere to the geometric structure of $SE(3)$ while using linearized models to successively satisfy the non-convex dynamics constraints.

Paper organization: In Sec. 2, we review relevant work in the field of AFF motion planning and sequential convex programming. Then in Sec. 3, we present our proposed methodology of using sequential convex programming for trajectory refinement. Simulation results are shown in Sec. 4. Finally, Sec. 5 summarizes our contributions and proposes future areas of investigation.

2 PRELIMINARIES

2.1 Related Works

Sampling-based motion planning is a promising technique for generating trajectories for free-flying robots, and its efficacy has been demonstrated on high-dimensional robotic systems with complex constraints [3]. Early works on motion planning for the SPHERES free-flying robots combined sampling-based approaches with trajectory refinement to enforce nonlinear dynamics constraints, but these methods were either not amenable to real-time applications or lacked a notion of optimality [4, 5]. Recently, [6] highlighted the importance of considering rotational motions to return solutions that would otherwise fail if only translational trajectories were considered with the robot radius inflated.

The advantages of using sampling-based approaches to address challenging pointing constraints have been illustrated in [7, 8, 9]. However, sampling-based planners can return solutions that are jagged and of poor quality, so they are often used in conjuction with a post-processing step to refine and improve the quality of the trajectory. Traditional refinement approaches such as polynomial shortcuts or elastic banding [10] methods available for flat Euclidean spaces cannot be directly applied for rotational motions on $SO(3)$.

Planning on the $SE(3)$ manifold requires that resulting trajectories satisfy group structure constraints associated with the rotational Lie group $SO(3)$. Lie group variational integrators [11] have been used in planning and optimization problems, but the discrete state update equations used to satisfy group constraints remain nonlinear. For quaternion parameterizations of the attitude, solutions must lie on the 3-sphere $\mathcal{S}^3$, a non-convex domain. A common approach simply normalizes the quaternion solutions, resulting in departures from S^3 which offers no guarantees of accuracy. In [12], a *big-M* integer formulation is used to constrain decision variables to one surface of an $\mathbb{R}^4$ polytope approximation of $\mathcal{S}^3$, but this approach scales exponentially with the number of binary variables.

2.2 Sequential Convex Programming

Recently, sequential convex programming (SCP) has emerged as a promising tool for solving problems with nonlinear dynamics and equality constraints in a principled manner, and it has been demonstrated on a class of challenging aerospace and robotics problems [13, 14, 15]. SCP provides real-time techniques for tackling the associated non-convexity of a problem without resorting to nonlinear programming methods such as simulated annealing or genetic programming. Unlike for these latter methods that lack convergence guarantees and often require well-tuned warm starts to find a solution, casting a non-convex problem in a convex framework through or constraint relaxation provides global convergence guarantees and polynomial-time complexity for each SCP iteration.

Convex optimization has demonstrated its effectiveness for applications requiring onboard autonomous guidance and control and advances in interior point solvers have made it even more amenable for real-time applications. SCP is a paradigm in which non-convex optimization problems are convexified and a series of convex optimization problems are solved. For dynamical systems, the constraints that render a problem non-convex are usually related the nonlinear equality constraints for the dynamics. SCP allows these non-convex dynamical equality constraints to be satisfied using a successive convexification approach while maintaining a trust region over which a local minimum is searched for.

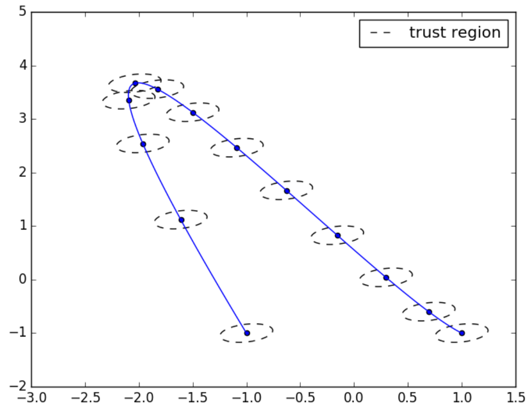


Figure 2 Trust regions about reference trajectory.

2.3 Notation

The kth vector in a trajectory is denoted with a subscript, and the iteration number n of the trajectory is denoted with the superscript in parentheses, e.g. $\mathbf{x}_k^{(n)}$.

3 TECHNICAL APPROACH

3.1 Problem Statement

The optimal control problem being approximately solved by the planning algorithm has the form:

$$\min \quad C(\mathbf{x},\mathbf{u},t) = \phi(\mathbf{x}_{\text{final}}, t_{\text{final}}) + \int_0^{t_f} J(\mathbf{x},\mathbf{u},t)\mathrm{d}t \qquad (2)$$

subject to:

$$\begin{aligned} \mathbf{x}(t_{\text{init}}) &= \mathbf{x}_{\text{init}} && \text{Initial condition} \\ \mathbf{x}(t_{\text{final}}) &\in \mathcal{X}_{\text{goal}} && \text{Terminal condition} \\ \mathbf{x}(t) &= f(\mathbf{x}(t),\mathbf{u}(t),t) && \text{System dynamics} \\ \mathbf{x}(t) &\in \mathcal{X}_{\text{free}} && \text{Obstacle avoidance} \\ \mathbf{u}(t) &\in \mathcal{U} && \text{Actuator constraints} \end{aligned}$$

The problem of interest is to guide the robot from an initial state $\mathbf{x}_{\text{init}}$ to a goal region $\mathcal{X}_{\text{goal}}$. $J(\mathbf{x},\mathbf{u},t)$ and $\phi(\mathbf{x}_{\text{final}}, t_{\text{final}})$ are the cost functional over the state trajectory and terminal cost penalty, respectively. The solution to this problem is the control input $\mathbf{u}(t)$ that satisfies the listed constraints while also minimizing J. The problem in Equation (2) is non-convex due to the system dynamics constraints evolving as a nonlinear function $f(\mathbf{x}(t),\mathbf{u}(t),t)$ and the obstacle avoidance constraints $\mathbf{x}(t) \in \mathcal{X}_{\text{free}}$.

Our proposed methodology for solving (2) entails a two-step approach:

1. Use a sampling-based planner to generate a dynamically-feasible, collision-free solution of a particular homotopy class (3.3)

2. Refine the trajectory using SCP to find a locally optimum solution (3.4)

3.2 Dynamics

We explicitly incorporate the rotational motion of the robot by including orientations and angular velocity in the configuration space used for sampling:

$$\mathbf{x} = (\mathbf{r}, \mathbf{v}, \mathbf{q}, \omega)^T \tag{3}$$

$\mathbf{r}, \mathbf{v} \in \mathbb{R}^3$ are the robot position and velocity, respectively, expressed in a locally fixed inertial frame. $\omega \in \mathbb{R}^3$ is the angular velocity also expressed in the inertial frame. From (1), the pose of a rigid body is given by its position and orientation. In this work, we choose to represent the orientation using the quaternion parameterization of attitude $\mathbf{q} \in \mathcal{S}^3$. The set of unit quaternions is:

$$\mathcal{S}^3 = \left\{\mathbf{q} \in \mathbb{R}^4 \;\middle|\; \|\mathbf{q}\|_2 = 1\right\}$$

The local geometry of $SO(3)$ is identical to that of $\mathcal{S}^3$ and it should be noted that there exists a two-to-one mapping between antipodal unit quaternions $\mathbf{q}$ and $-\mathbf{q}$ to the corresponding orientation $\mathbf{R} \in SO(3)$ [16].

The translational dynamics are given by a simple double integrator system:

$$\begin{pmatrix}\dot{\mathbf{r}}\\ \dot{\mathbf{v}}\end{pmatrix} = \begin{pmatrix}0 & I_3\\ 0 & 0\end{pmatrix}\begin{pmatrix}\mathbf{r}\\ \mathbf{v}\end{pmatrix} + \begin{pmatrix}0\\ \frac{1}{m}I_3\end{pmatrix}\mathbf{F} \tag{4}$$

m is the robot mass and $\mathbf{F} \in \mathbb{R}^3$ is the force. The nonlinear rotational dynamics are:

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\omega)\mathbf{q} \tag{5}$$

$$\dot{\omega} = \mathbf{J}^{-1}(\mathbf{M} - \omega \times \mathbf{J}\omega) \tag{6}$$

$\Omega(\omega)$ is the skew-symmetric matrix representation of the cross-product operation. $\mathbf{M} \in \mathbb{R}^3$ is the moment applied and $\mathbf{J}$ is the inertia tensor.

3.3 Initial Trajectory Generation

Sampling-based planning techniques search for feasible and approximately-optimal paths for a system by drawing samples from the system's configuration space and attempting edge connections between them. Provided a "black-box" collision checker, samples and edge connections can be checked for constraint satisfaction and thrown out if in violation, thus avoiding an explicit geometric construction of valid (i.e. collision-free) regions of the configuration space. This implicit description of the free configuration space is key to enabling scalability to high-dimensional problems. In this work, the initial collision-free trajectory is solved for using an implementation of the kinodynamic rapidly-exploring random trees (kino-RRT) [3]. We use the control-sampling-based approach from kino-RRT to ensure that the initial seed for the trajectory optimization is dynamically feasible. The output of the planner is a tree (V, E, U), where V is a set of vertices in the configuration space and the controls U corresponding to the edges E connecting the vertices.

3.4 Trajectory Refinement

Solutions of sampling-based planners may be of poor quality in the sense that they appear too unnatural or jerky for the robot to carry out even when dynamically feasible. SCP is an optimization-based method for smoothing the trajectory using a notion of smoothness in the objective function. Each iteration of SCP consists of convexifying the objective and constraints around a reference trajectory $(\mathbf{x}^{(n)}, \mathbf{u}^{(n)})$ and solving for a step $(\Delta\mathbf{x}^{(n)}, \Delta\mathbf{u}^{(n)})$ that leads to improvements in the objective function and constraint satisfaction.

Because convex approximations of the problem are only valid within a region close to the nominal trajectory, SCP keeps track of a trust region $\Delta^{(n)}$ near the previous solution $\mathbf{x}_k^{(n-1)}$ that the new solution $\mathbf{x}_k^{(n)}$ must lie within. The trust region constraint determines whether the trust region should shrink or grow based upon the accuracy of the local approximation of the non-convex constraint.

3.4.1 Trust Region Updates

The trust region size $\Delta^{(n)}$ is updated by calculating the ratio:

$$\rho^{(n)} = \frac{J(\mathbf{x}_k^{(n-1)}) - J(\mathbf{x}_k^{(n)})}{J(\mathbf{x}_k^{(n-1)}) - \hat{J}(\mathbf{x}_k^{(n)})} \tag{7}$$

Here, $\hat{J}$ is the convexified model of the problem at hand. The numerator and denominator in (7) represent the actual and predicted improvements in cost, respectively, between iterations $n-1$ and n. A value of $\rho^{(n)}$ close to 1 then indicates a good matchup between the nonlinear and convex models and the trust region expands by a preset factor β_{succ}. Correspondingly, a low or possibly negative value of $\rho^{(n)}$ indicates a poor local model of the problem and $\Delta^{(n)}$ shrinks by preset factor β_{fail}.

On $\mathbb{R}^n$, the trust region constraint is typically the Euclidean distance between the decision variables at the current iteration n and the previous one $n-1$:

$$\|\mathbf{r}_k^{(n)} - \mathbf{r}_k^{(n-1)}\|_2 \leq \Delta^{(n)}$$

For planning on $SE(3)$, we use the notion of a minimum *swept volume* to define the trust region in which refinements to the trajectory must lie within.

Because the Riemannian distance metric is more difficult to compute on $SE(3)$, we instead treat $(\mathbf{r}, \mathbf{q}) \in \mathcal{M} \equiv \mathbb{R}^3 \times \mathcal{S}^3$, such that $SE(3)$ is strictly contained in $\mathcal{M}$. $\mathcal{M}$ is equipped with the Riemannian metric $g_{\mathcal{M}} = g_{\mathbb{R}^3} + g_{\mathcal{S}^3}$, where $g_{\mathbb{R}^3}$ is the flat metric of $\mathbb{R}^3$ and $g_{\mathcal{S}^3}$ is the metric induced by the flat metric of $\mathbb{R}^4$ and the pull-back operator, i.e. $g_{\mathcal{S}^3} = i^* g_{\mathbb{R}^4}$, where $i : \mathcal{S}^3 \to \mathbb{R}^4$ is the canonical immersion.

By standard arguments of Riemannian geometry [17], we know that the Riemannian distance $d_{\mathcal{M}}(\mathbf{x}_0, \mathbf{x}_1)$ of two points $\mathbf{x}_0$ and $\mathbf{x}_1$ in $\mathcal{M}$ can be evaluated as

$$d_{\mathcal{M}}(\mathbf{x}_0, \mathbf{x}_1) = \int_0^1 \|\sigma'_{\mathcal{M}}(t)\|_{\sigma_{\mathcal{M}}(t)} \mathrm{d}t$$

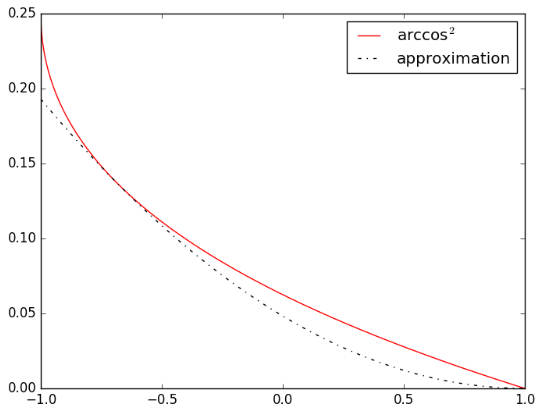


Figure 3 Quadratic approximation of $\arccos^2(x)$ for $x \in [-1, 1]$.

where $\sigma_{\mathcal{M}} : [0, 1] \to \mathcal{M}$ is a geodesic with respect to the Levi-Civita connection related to $g_{\mathcal{M}}$, joining $\mathbf{x}_0$ to $\mathbf{x}_1$. Similar arguments provide that $\sigma_{\mathcal{M}}$ has the form

$$(t\mathbf{r}_1 + (1-t)\mathbf{r}_0, \sigma_{\mathcal{S}^3}(t))$$

where $\sigma_{\mathcal{S}^3}(t) : [0, 1] \to \mathcal{S}^3$ is a geodesic in $\mathcal{S}^3$ joining $\mathbf{q}_0$ to $\mathbf{q}_1$. Thus,

$$d_{\mathcal{M}}(\mathbf{x}_0, \mathbf{x}_1) = \int_0^1 \sqrt{\|\mathbf{r}_1 - \mathbf{r}_0\|^2 + \sigma'_{\mathcal{S}^3}(t) \cdot \sigma'_{\mathcal{S}^3}(t)}\mathrm{d}t$$

Here, $\sigma'_{\mathcal{S}^3}(t) \cdot \sigma'_{\mathcal{S}^3}(t)$ is the standard inner product on $\mathbb{R}^4$, and it can be shown that this expression reduces to $\arccos^2(\mathbf{q}_0 \cdot \mathbf{q}_1)$. Thus, the Riemannian distance on $\mathcal{M}$ is given by:

$$d_{\mathcal{M}}(\mathbf{x}_0, \mathbf{x}_1) = \sqrt{\|\mathbf{r}_1 - \mathbf{r}_0\|^2 + \arccos^2(\mathbf{q}_0 \cdot \mathbf{q}_1)} \tag{8}$$

This $\arccos^2(\mathbf{q}_0 \cdot \mathbf{q}_1)$ term cannot be used in a disciplined convex programming framework [18] and must be approximated before being used as a trust region constraint on $\mathcal{M}$. As shown in Fig. 3, for arguments x in the range $-1 \le x \le 1$, $\arccos^2(x)$ is lower bounded by:

$$\left(\frac{\pi}{2}\alpha(1 - x)\right)^2 \le \arccos^2(x) \quad x \in [-1, 1] \tag{9}$$

A derivation of this constraint and the constant value α are given in [19]. The quadratic approximation of the constraint in (8) can then be written as:

$$\|\mathbf{r}_k^{(n)} - \mathbf{r}_k^{(n-1)}\|_2^2 + \left(\frac{\pi}{2}\alpha\left(1 - \left(\mathbf{q}_k^{(n-1)}\right)^T \mathbf{q}_k^{(n)}\right)\right)^2 \le \Delta_{SE(3)}^{(n)} \tag{10}$$

Because a quaternion on $\mathcal{S}^3$ has a unity norm constraint $\|\mathbf{q}_k\|_2 = 1$, the argument of $\mathbf{q}_0 \cdot \mathbf{q}_1$ is automatically guaranteed to lie in the valid range $x \in [-1, 1]$ for Equation (9), so the quadratic approximation will always be valid. The nonlinear unity norm constraint is satisfied iteratively in the convex optimization using the following two constraints:

$$\|\mathbf{q}_k\|_2 \le 1 \tag{11}$$

$$\|\mathbf{q}_k^{(n-1)}\|_2 - 1 + \nabla(\|\mathbf{q}_k^{(n-1)}\|_2)^T(\mathbf{q}_k^{(n)} - \mathbf{q}_k^{(n-1)}) = 0 \tag{12}$$

Further, we also constrain the scalar component of the quaternion q_w to one hemisphere of $\mathcal{S}^3$ by canonicalizing all quaternion decision variables with a non-negativity constraint to avoid wildly spinning steering solutions:

$$q_{w,k} \ge 0 \tag{13}$$

The trust region constraints for the velocity and angular velocity are simply the Euclidean norm:

$$\|\mathbf{v}_k^{(n)} - \mathbf{v}_k^{(n-1)}\|_2^2 \le \Delta_v^{(n)} \tag{14}$$

$$\|\boldsymbol{\omega}_k^{(n)} - \boldsymbol{\omega}_k^{(n-1)}\|_2^2 \le \Delta_\omega^{(n)} \tag{15}$$

3.4.2 Convexification of Nonlinear Equality Constraints

Equation (2) includes two nonlinear equality constraints that must be satisfied, the nonlinear dynamics over $\mathcal{S}^3$ and the quaternion norm constraint $\|\mathbf{q}\|_2 = 1$. We linearize the nonlinear rotational dynamics, denoted $f_{\mathcal{S}^3}$, using a trapezoidal approximation:

$$\begin{aligned} \mathbf{x}_{\mathcal{S}^3,k+1}^{(n)} &= \mathbf{x}_{\mathcal{S}^3,k}^{(n)} + \\ \frac{\Delta t}{2}\Big[f_{\mathcal{S}^3}(\mathbf{x}_{\mathcal{S}^3,k+1}^{(n-1)}) &+ A_{\mathcal{S}^3,k+1}^{(n-1)}(\mathbf{x}_{\mathcal{S}^3,k+1}^{(n)} - \mathbf{x}_{\mathcal{S}^3,k+1}^{(n-1)}) + \\ B_{\mathcal{S}^3}(\mathbf{M}_{k+1}^{(n)} &- \mathbf{M}_{k+1}^{(n-1)}) + f_{\mathcal{S}^3,k}(\mathbf{x}_{\mathcal{S}^3,k}^{(n-1)}) + \\ A_{\mathcal{S}^3,k}^{(n-1)}(\mathbf{x}_{\mathcal{S}^3,k}^{(n)} - \mathbf{x}_{\mathcal{S}^3,k}^{(n-1)}) &+ B_{\mathcal{S}^3}(\mathbf{M}_k^{(n)} - \mathbf{M}_k^{(n-1)})\Big] \end{aligned} \tag{16}$$

Here, $\mathbf{x}_{\mathcal{S}^3,k}^{(n)}$ refers strictly to the rotational components of the dynamics $\mathbf{q}_k \in \mathcal{S}^3$ and $\boldsymbol{\omega}_k \in \mathbb{R}^3$. The step size Δt is $(t_f - t_0)/N$ where $N+1$ is the total number of discretization nodes. As the translational and rotational control for AFFs are decoupled, only the torque $\mathbf{M}_k$ is considered for refining the rotational trajectory. $A_{\mathcal{S}^3,k}^{(n-1)}$ is the local linearization of $f_{\mathcal{S}^3}$ about $\mathbf{x}_{\mathcal{S}^3,k}^{(n-1)}$ and $B_{\mathcal{S}^3}$ is a constant matrix:

$$A_{\mathcal{S}^3,k}^{(n-1)} = \frac{\partial f_{\mathcal{S}^3}}{\partial \mathbf{x}_{\mathcal{S}^3,k}^{(n-1)}} \qquad B_{\mathcal{S}^3} = \frac{\partial f_{\mathcal{S}^3}}{\partial \mathbf{M}}$$

The discretized double integrator dynamics for (4) are:

$$\mathbf{x}_{\mathbb{R}^3,k+1}^{(n)} = A_{\mathbb{R}^3}\mathbf{x}_{\mathbb{R}^3,k}^{(n)} + B_{\mathbb{R}^3}\mathbf{F}_k^{(n)} \tag{17}$$

$$A_{\mathbb{R}^3} = \begin{pmatrix} I_3 & \Delta t I_3 \\ 0 & I_3 \end{pmatrix} \qquad B_{\mathbb{R}^3} = \begin{pmatrix} \frac{1}{2}\Delta t^2 I_3 \\ \Delta t I_3 \end{pmatrix}$$

$\mathbf{x}_{\mathbb{R}^3} \in \mathbb{R}^6$ includes the position $\mathbf{r}$ and velocity $\mathbf{v}$. $A_{\mathbb{R}^3}$ and $B_{\mathbb{R}^3}$ are the discrete state update matrices.

3.4.3 Thruster Constraint Satisfaction

The cost function to be minimized for the problem may be any convex cost function. AFFs such as Astrobee and SPHERES typically use two thrusters on each face, for a total of 12 for translational and rotational control. The chosen cost function for this configuration seeks to minimize chatter in the individual thruster commands $\mathbf{f} \in \mathbb{R}^{12}$:

$$J(\mathbf{x},\mathbf{u},t) = \sum_{k=0}^{N-2} \|\mathbf{f}_{k+1} - \mathbf{f}_k\|_1$$

This objective must be accomplished while satisfying control allocation and individual thruster limits as well:

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{M} \end{pmatrix} = R(\mathbf{q})G\mathbf{f} \tag{18}$$

$$0 \le \mathbf{f} \tag{19}$$

$G \in \mathbb{R}^{6\times 12}$ is the control allocation matrix. Note that the individual thruster commands must be converted from the body to inertial frame using the rotation matrix $R(\mathbf{q})$. As this conversion is a non-convex constraint, we approximate the R using the previous quaternion solution $R(\mathbf{q}_k^{(n-1)})$.

3.4.4 SCP Algorithm

We model the control inputs $\mathbf{F}_k$ and $\mathbf{M}_k$ as zero-order holds. The decision variables for the convex optimization are the pose $(\mathbf{r},\mathbf{q})$, twist $(\mathbf{v},\omega)$, and controls $(\mathbf{F},\mathbf{M})$. The convex optimization problem to be solved at each iteration is given as:

$$\min \sum_{k=0}^{N-2} \|\mathbf{f}_{k+1} - \mathbf{f}_k\|_1 \tag{20}$$

$$\text{subject to:}$$

$$\mathbf{x}_0 = \mathbf{x}_{\text{init}} \qquad \mathbf{x}_N = \mathbf{x}_{\text{final}}$$

$$\begin{aligned} \|\mathbf{v}_k\|_2 &\le v_{\max}, & k &= 0,\dots,N \\ \|\omega_k\|_2 &\le \omega_{\max}, & k &= 0,\dots,N \end{aligned}$$

$$\begin{aligned} \|\mathbf{F}_k\|_2 &\le F_{\max}, & k &= 0,\dots,N-1 \\ \|\mathbf{M}_k\|_2 &\le M_{\max}, & k &= 0,\dots,N-1 \end{aligned}$$

$$\begin{aligned} (10),(11),(12),(13),(14),(15) & \quad k = 0,\dots,N \\ (16),(17) & \quad k = 0,\dots,N-1 \\ (18),(19) & \quad k = 0,\dots,N-2 \end{aligned}$$

3.5 Planning Algorithm

In this work, we implemented kino-RRT detailed in [3], but we note that the trajectory refinement algorithm below can be integrated into any planner that outputs a set of edges E that form a dynamically-feasible trajectory to the goal. Algorithm 1 shows the steps in the SCP approach to trajectory refinement.

4 SIMULATIONS

4.1 Implementation

The planning algorithm was implemented in the Julia language and was run on a Linux system equipped with a

Algorithm 1 Sequential Convex Programming

```
Require: solution edges $E$, vertices $V$, and controls $U$ from kino-RRT [3], $\beta_{\text{succ}}$, $\beta_{\text{fail}}$, $\Delta^{(0)}$, $0 < \rho^{(0)} < \rho^{(1)} < \rho^{(2)} < 1$
 1: for ConvexifyIteration $n = 1,2,\dots$ do
 2:     Solve optimization problem in (20)
 3:     Calculate true and modeled improvements in cost
 4:     Calculate model accuracy ratio $\rho^{(n)}$ in (7)
 5:     if $\rho^{(k)} < \rho^{(0)}$ then
 6:         Reject solution $(\mathbf{x}^{(n)}, \mathbf{u}^{(n)})$
 7:         $\Delta^{(n+1)} \leftarrow \beta_{\text{fail}}\Delta^{(n)}$
 8:     else
 9:         Accept solution $(\mathbf{x}^{(n)}, \mathbf{u}^{(n)})$
10:         Update trust region size
11:         $\Delta^{(n+1)} = \begin{cases} \beta_{\text{fail}}\Delta^{(n)} & \rho^{(n)} < \rho^{(1)} \\ \Delta^{(n)} & \rho^{(n)} \in [\rho^{(1)}, \rho^{(2)}) \\ \beta_{\text{succ}}\Delta^{(n)} & \rho^{(n)} \ge \rho^{(2)} \end{cases}$
12:     end if
13: end for
```

Parameter	Value
N	51
$\rho^{(0)}$	0.10
$\rho^{(1)}$	0.25
$\rho^{(2)}$	0.90
β_{succ}	1.2
β_{fail}	0.5
$\Delta_{SE(3)}$	0.80
Δ_v	0.20
Δ_ω	0.25

Table 1 SCP simulation parameters.

2.80GHz Intel i7 processor with 32GB memory. For the convex optimization problems, we used Convex.jl [18] as the solver interface and Gurobi [20] as the solver. Initial simulation parameters for the SCP iterations are given in Table (1) and the constraint values for Equation (20) correspond to the Astrobee robot constraints from [2].

4.2 Results

The planning scenario shown in Figure 4 involves the robot moving across an ISS module to reach a goal region in position and orientation. The initial trajectory used for refinement was generated using kino-RRT such that the constraints in (20) were initially satisfied, resulting in a trajectory that was 28s long. In order to keep the number of discretization nodes low, the trajectory refinement is carried out in a receding horizon fashion where the first few waypoints from the kino-RRT output is sampled at a $\Delta t = 0.01$ to retain an accurate approximation of the nonlinear rotational dynamics. Points further along in the trajectory are sampled at intervals of $\Delta t = 1s$ and this initial trajectory is used as $(\mathbf{x}^0, \mathbf{u}^0)$. Each trajectory refinement iteration requires 0.35s to solve for a total of approximately 2.1s for six iterations of SCP.

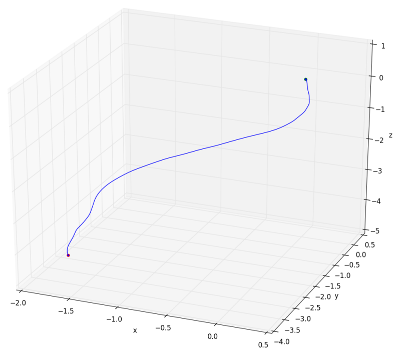


Figure 4 Translational trajectory of robot in workspace.

5 CONCLUSIONS

In this work, we presented results on using sequental convex programming as a means of refining dynamically free trajectories for free-flying robots while satisfying group constraints on $SE(3)$. The approach seeks to leverage the collision-free trajectory returned by the sampling-based planner and construct a locally optimal trajectory about this homotopy class.

Future areas of exploration include theoretical guarantees on (1) the convergence properties of the SCP optimizer to a solution and that (2) this solution satisfies the necessary conditions for the nonlinear optimal control problem in (2). We seek to demonstrate the efficacy of this approach via implementation of a planar air-bearing free-flying robot test bed. We would also like to investigate a free-final time formulation in which SCP solves for the optimal time step Δt from Equations (5) and (17) while accurately satisfying the non-linear dynamics constraints.

Real-time, kinodynamic motion planning remains a challenge with the dynamics in consideration for AFFs. As a part of our future work, we expect to leverage recent contributions from the fields of robotics, optimization, and control to develop a tractable algorithmic framework that can be deployed in space.

6 ACKNOWLEDGEMENTS

This work was supported in part by NASA under the Space Technology Research Program, Early Career Faculty Grant NNX12AQ43G and Early Stage Innovations Grant NNX16AD19G. A. Cauligi and A. Bylard are supported by NASA Space Technology Research Fellowship (NSTRF) Grants NNX16AM78H and NNX15AP67H, respectively. The authors would like to thank Ed Schmerling for his insightful suggestions.

References

[1] Bylard A, MacPherson R, Hockman B, Cutkosky MR and Pavone M (2017) Robust Capture and Deorbit of Rocket Body Debris Using Controllable Dry Adhesion. In: *IEEE Aerospace Conference*.

[2] Smith T, Barlow J, Bualat M, Fong T, Provencher C, Sanchez H and Smith E (2016) Astrobee: A New Platform for Free-Flying Robotics on the International Space Station. In: *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*.

[3] LaValle SM and Kuffner JJ (2001) Randomized Kinodynamic Planning. In: *Int. Journal of Robotics Research*, 20(5):pp.378–400.

[4] Aoudé GS (2007) *Two-Stage Path Planning Approach for Designing Multiple Spacecraft Reconfiguration Maneuvers and Applications to SPHERES onboard ISS*. Master's thesis, Massachusetts Inst. of Technology.

[5] Fejzić A (2008) *Development of control and autonomy algorithms for docking to complex tumbling satellites*. Master's thesis, Massachusetts Inst. of Technology.

[6] Liu S, Mohta K, Atanasov N and Kumar V (2018) Search-Based Motion Planning for Aggressive Flight on SE(3). In: *IEEE Robotics and Automation Letters*, 3(3):pp.2439 – 2446.

[7] Frazzoli E, Dahleh MA, Feron E and Kornfeld R (2001) A randomized attitude slew planning algorithm for autonomous spacecraft. In: *AIAA Conf. on Guidance, Navigation and Control*.

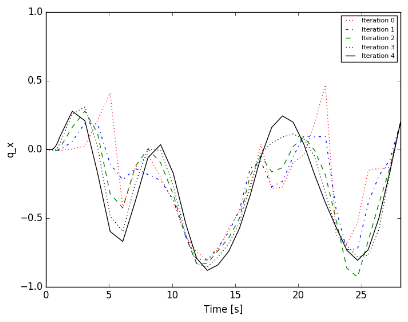
[8] Kjellberg HC and Lightsey EG (2013) Discretized Constrained Attitude Pathfinding and Control for Satellites. In: *AIAA Journal of Guidance, Control, and Dynamics*, 36(5):pp.1301 – 1309.

[9] Weiss A, Leve F, Baldwin M, Forbes J and Kolmanovsky I (2014) Spacecraft constrained attitude control using positively invariant constraint admissible sets on SO(3) x R3. In: *American Control Conference*.

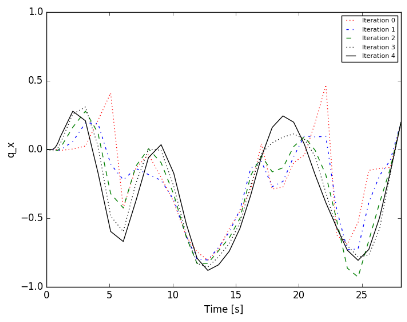
[10] Quinlan S and Khatib O (1993) Elastic Bands: Connecting Path Planning and Control. In: *Proc. IEEE Conf. on Robotics and Automation*.

[11] Lee T, McClamroch NH and Leok M (2005) A Lie group variational integrator for the attitude dynamics of a rigid body with applications to the 3D pendulum. In: *IEEE Conf. on Control Applications*.

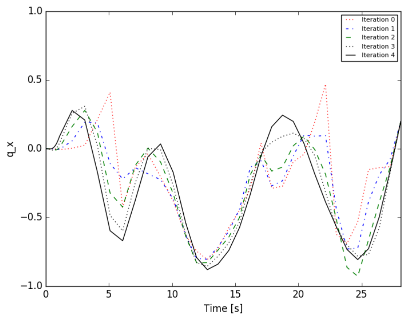
[12] Eren U, Açikmeşe B and Scharf DP (2015) A mixed integer convex programming approach to constrained attitude guidance. In: *European Control Conference*.



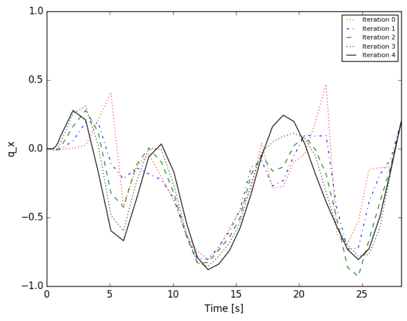
(a) q_x



(b) q_y



(c) q_z



(d) q_w

Figure 5 Quaternion components for rotational slew.

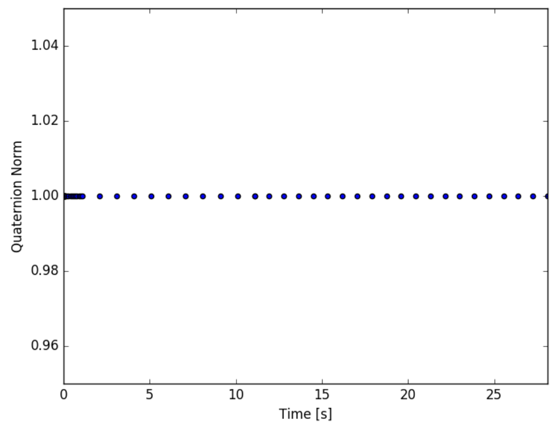


Figure 6 Quaternion norm for trajectory.

[13] Schulman J, Duan Y, Ho J, Lee A, Awwal I, Bradlow H, Pan J, Patil S, Goldberg K and Abbeel P (2014) Motion planning with sequential convex optimization and convex collision checking. In: *Int. Journal of Robotics Research*, 33(9):pp.1251–1270.

[14] Liu X and Lu P (2014) Solving Nonconvex Optimal Control Problems by Convex Optimization. In: *AIAA Journal of Guidance, Control, and Dynamics*, 37(3):pp.750 – 765.

[15] Mao Y, Szmuk M and Açikmeşe B (2016) Successive Convexification of Non-Convex Optimal Control Problems and Its Convergence Properties. In: *Proc. IEEE Conf. on Decision and Control*.

[16] Shuster MD (1993) A survey of attitude representations. In: *Journal of the Astronautical Sciences*, 41(4):pp.439 – 517.

[17] do Carmo M (1971) *Riemannian geometry*. Birkhäuser, first edition.

[18] Udell M, Mohan K, Zeng D, Hong J, Diamond S and Boyd S (2014) Convex Optimization in Julia. In: *High Performance Technical Computing in Dynamic Languages*.

[19] Goemans MX and Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. In: *Journal of the Association for Computing Machinery*, 42(6):pp.1115 – 1145.

[20] Gurobi Optimization, Inc. (2016) *Gurobi Optimizer reference manual*.