Cooperative Shared-Load Carrying by Quadrotors

Seiji Shaw
Massachusetts Institute of Technology
Cambridge, Massachusetts
seijis@mit.edu

Thomas Cohn
Massachusetts Institute of Technology
Cambridge, Massachusetts
tcohn@mit.edu

***Abstract*—In our final project, we implement and test a set of control strategies for a multiple-quadrotor team to carry a slung load. Our goals are to stabilize the system to a set equilibrium point and to compute and stabilize to a dynamically-feasible trajectory specified by user-supplied waypoints. We derive a system model with hybrid dynamics, where the carrying cables are modeled as springs. To stabilize the system, we solve a nonlinear mathematical program to find an equilibrium point and compute a finite-horizon LQR controller. To construct trajectories through waypoints, we show that our system is differentially flat. We solve a quadratic program to find a piecewise-polynomial trajectory in the system's output space and use a finite-horizon LQR controller to stabilize to the trajectory. Video: https://youtu.be/c17hjDMm7Gw**

***Index Terms*—Unmanned aerial vehicles, Trajectory optimization, Differential flatness, Cooperative control**

I. Introduction

A common application for unmanned aerial vehicles is to carry loads to areas too cumbersome for humans or ground vehicles to reach. When an object is too large to be carried within a quadrotor, a natural solution is to carry it as a slung load. A tensioned cable is used to tether the object to the quadrotor, similar to what is done with helicopter transports (albeit at a larger scale). A single quadrotor may be sufficient for smaller objects, but heavier objects may require multiple quadrotors working together. While a multi-bot solution could provide greater flexibility and greater robustness to failure than using a bigger drone, it presents a challenging control problem, due to the multiple causes of underactuation.

The first source of underactuation is straightforward: the quadrotors themselves are underactuated. Due to the fixed mounting of the propellers, it is impossible to exert a lateral force on the quadrotor (with respect to its own local coordinate frame). The quadrotor also has input limits – the propellers can only exert so much lifting force.

The control of quadrotor systems is well-established, so this source of underactuation should not lead to major difficulties. However the slung load setup leads to a much greater challenge, because the tension from the cables can only act as a pulling force. The cables can also go slack if allowed to contract past their nominal length and cannot exert any forces. Thus, we are faced with controlling an underactuated hybrid system; a problem which, in its general form, is at the cutting edge of current robotics research.

In this project, we endeavour to create a set of tools for planning and controlling a multiple-quadrotor team to carry a slung load. We primarily model our approach on the work presented by Sreenath and Kumar [1]. We use an idealized hybrid-dynamics model to account for the slack and tension modes of the cables between the drones and the mass. In the tension mode, we assume that the cables act as Hookian springs, i.e., the exerted tension force is proportional to the displacement of the cable from its nominal length.

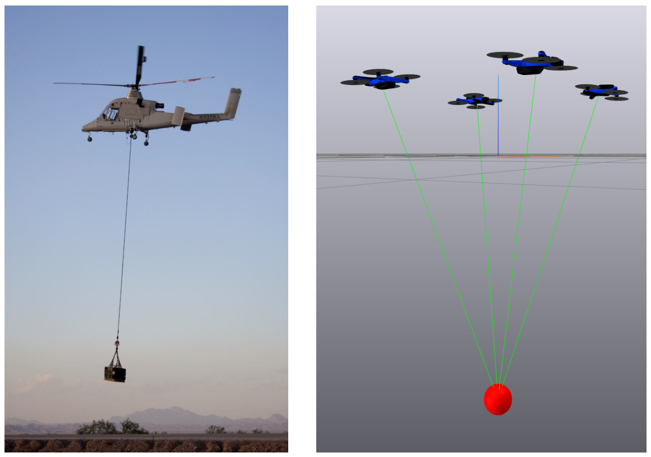


Fig. 1: A slung load carried by a helicopter (left, from [3]) or a team of quadrotors (right).

Our control techniques are based on theoretical and empirical analysis of the system. We use nonlinear optimization to identify equilibrium points of the system, to serve as stable initial and final configurations for these trajectories. We are able to construct locally stabilizing controllers about such fixed points. Then, we demonstrate that our system is differentiably flat – a fact we leverage to plan aggressive trajectories in a manner similar to [2]. We demonstrate that our approach can be used to produce a variety of load trajectories, including instances where the load can be moved at high speeds, or change direction very quickly. Finally, we track these trajectories with a finite horizon linear-quadratic regulator.

II. Related Work

Cooperative control of suspended masses is a widely studied problem. The system dynamics has been modeled under many considerations (elastic cables, air-resistance, response time of a rope-slacking) [4]. Sreenath and Kumar [1] present an idealized hybrid system that models cables that produce a tensile force when the distance between the drone and the mass exceeds a specified length.

Unsurprisingly, since quadrotors and multi-quadrotor systems have a rich set of models, there are also a rich set of well-established control methodologies for these systems. Mellinger et al. [2] popularized a trajectory optimization-based approach that leverages the *differential flatness* property of quadrotor dynamics. If a system is differentiably flat, rather than planning in the control space directly, a controller can optimizes a trajectory in an flat *output space*, which can lead to more tractable optimization. The corresponding state trajectory and control inputs can be computed from this output trajectory and used for control. Since the state of an individual quadrotor is $\mathsf{SE}(3)$, a non-Euclidean manifold, Lee et al. applied geometric-control based methods [5] to provide convergence and/or stability guarantees for quadrotors to track trajectories and stabilize to fixed points.

Many of the control strategies developed for single quadrotors have been extended to multi-quadrotor systems. Sreenath and Kumar [1] show their hybrid-system is differentiably flat, and leverage this property for control. Wu et al. [6] also exploit differential flatness, and additionally apply geometric-control methods to track trajectories.

Our approach takes after Sreenath and Kumar's work [1]. Rather than modelling cables as reactive tensile forces to gravity, we model them as springs that will exert a force when the distance between the drone and the mass exceeds a pre-specified length. To follow aggressive trajectories, we follow an approach guided by Mellinger et al. [2].

III. System Model

We model our system as a mass load suspended by cables from $n \geq 2$ quadrotors. We assume the quadrotors do not experience drag and will not collide with each other. The cables are massless, will never tangle, and are connected to the quadrotors at their respective centers of mass.

The symbols we choose to use in our model closely follow the same scheme used in [1]. For a system of n quadrotors, we denote the pose of the ith quadrotor with position x_i and orientation r_i. The tension force induced by the cable connected to quadrotor i will be written as $T_i q_i$, where $T_i \in \mathbb{R}^+$ is the magnitude of the force and $q_i \in S^2$ is the unit vector pointing from the mass to quadrotor i. Since we are modeling our system with a point mass (with mass denoted m_m), we denote the position of the mass as x_m and do not need to specify an orientation. We give all cables the same length L and Hooke constant K, measured in meters and Newtons per meter respectively. For all units and standard symbols, please refer to Table I.

Our cables are modeled as springs and will only exert a force if they are in tension, which introduces the hybrid component of our system. We express the tension of the cable in terms of the positions of the mass and the ith quadrotor:

$$T_i q_i = \begin{cases} K(x_i - x_m - Lq_i) & ||x_i - x_m|| \geq L \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

As in [1], the influence of the tension forces and gravity on the system are accounted for by the standard Newton-Euler dynamics for quadrotors:

$$m_i \ddot{x}_i = f_i r_i e_3 - m_i g e_3 + T_i q_i \tag{2}$$

$$M_i = J_i \dot{\omega}_i + \omega_i \times J_i \omega_i \tag{3}$$

$$m_m \ddot{x}_m = -\sum_{i=1}^{n} T_i q_i - m_m g e_3, \tag{4}$$

where ω_i, f_i, M_i, and J_i are the angular velocity, thrust, moment, and moment of inertia of the ith quadrotor, respectively, and ge_3 is the acceleration vector due to gravity expressed in the world frame.

Symbol	Definition
L	Minimum length for spring to exert force.
K	Hooke constant of cable.
g	Acceleration due to gravity.
e_1, e_2, e_3	Standard basis of x, y, z axes in the world frame.
$m_m \in \mathbb{R}^+$	Mass of slung load.
$x_m \in \mathbb{R}^3$	Position of slung load.
$x_i \in \mathbb{R}^3$	Position of quadrotor i.
$r_i \in \mathsf{SO}(3)$	Orientation of quadrotor i.
$\psi_i \in \mathbb{R}$	Yaw of quadrotor i.
$\omega_i \in \mathbb{R}^3$	Angular velocity of quadrotor i.
$m_i \in \mathbb{R}^+$	Mass of quadrotor i.
$T_i \in \mathbb{R}$	Tension of spring of quadrotor i.
$q_i \in S^2$	Direction of tension force from load to quadrotor i.
$f_i \in \mathbb{R}$	Thrust produced by propellors of quadrotor i.
$M_i \in \mathbb{R}^3$	Moment produced by propellors of quadrotor i.

TABLE I: Symbols used to describe system parameters and state.

IV. Planning

The first key planning tool we present can be used to find equilibrium points so that the system can stabilize to a static configuration. We then present a naive nonlinear trajectory optimization method, before describing the differential flatness property of our system and how to leverage it for more principled optimization.

A. Finding Equilibrium Points

In any nonlinear planning and control problem, equilibrium points serve a vital role. It is natural to expect the start and end point of any trajectory to be equilibrium points, and techniques such as model predictive control rely on treating these points as worst-case endpoints. However, for a complicated nonlinear system such as the one at hand, it may be difficult to find equilibrium points by hand. Thus, we turn our attention to general nonlinear programming, which can be handled by powerful solvers such as SNOPT [7], [8].

Given a system $\dot{x} = f(x, u)$, we can search for an equilibrium point by solving an optimization problem of the form

$$\begin{aligned} \min_{x,u} \quad & \mu(x, u) \\ \text{s.t.} \quad & f(x, u) = 0 \\ & g_i(x, u) \geq 0, \quad \forall i \in I \end{aligned} \tag{5}$$

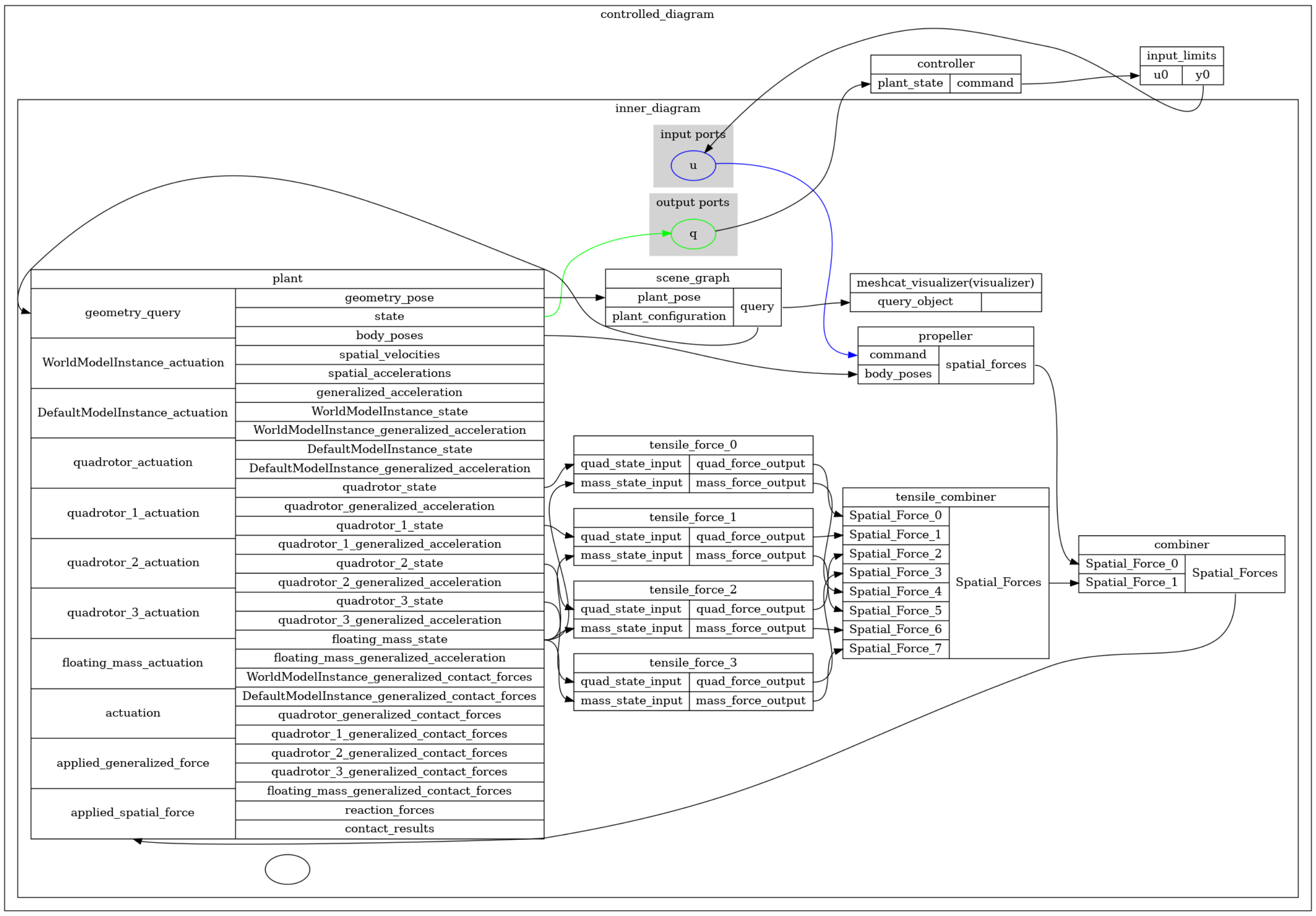


Fig. 2: The complete block diagram for our system. All propeller thrust and tensile forces are modeled as state-dependent external forces on the system.

where $\mu(x,u)$ is an arbitrary user-specified objective and the $g_i(x,u) \geq 0$ are arbitrary user-specified constraints. In the simplest case, a trivial objective $\mu(x,u) := 0$ is used, with no additional constraints. But this is inadvisable, as it leaves everything up to the whims of the nonlinear solver.

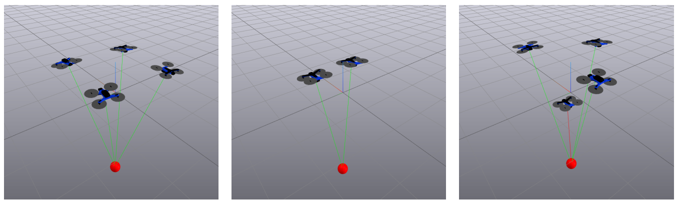
In our case, we place a uniform quadratic cost on the position and control signal. The cost on the position encourages the solver to find an equilibrium point near the origin (which we can then translate arbitrarily after the fact, since the system is translation-invariant). The cost on the control input encourages the solver to find an equilibrium point with minimal required control effort. This way, the quadrotors need not waste energy pulling against each other.

We also leverage several constraints, besides the essential $f(x,u) = 0$. We enforce input limits, to ensure that the equilibrium point will not exceed the maximum thrust of the propellers. We also require that all of the cables are in tension, by setting a minimum distance between the slung load and each quadrotor. Without this, SNOPT will frequently find equilibrium points where only a few quadrotors are holding the slung load up, and the remaining quadrotors are simply hovering nearby, with slack cables. This is undesirable, as in such configurations, it is harder to control the system. Finally, we add a minimum distance constraint between each pair of quadrotors. Spreading the quadrotors out improves the stability of the system, as it is possible to exert a wider range of forces on the slung load. Furthermore, while we do not concern ourselves with collision avoidance (and indeed, disable collision physics in our simulations), this does make collisions less likely. Such a feature would be desirable in a more advanced system, that does handle collision avoidance.

In practice, we are able to find a variety of equilibrium points for any number of quadrotors. SNOPT does not always find a solution on the first try, but by rerunning the solver with a new, random intialization, we will always eventually find an equilibrium point. We include an example of an equilibrium point in Figure 3, and compare it to equilibrium points found without the additional constraints.

B. *Naive Trajectory Optimization via Direct Collocation*

Given an initial and final configuration, the problem of finding a path from one to the other can be transcribed as a trajectory optimization problem. Such problems are generally



(a) (b) (c)

Fig. 3: Equilibrium points found with nonlinear programming. (a) is an ideal equilibrium point. (b) is obtained when there is no minimum quadrotor distance constraint (there are actually four quadrotors, organized into two perfectly interposed pairs). (c) is obtained when the cables are not required to be in tension, so one of the quadrotors cannot exert any force on the slung load.

nonconvex, but SNOPT can still sometimes come up with a solution (although it may not be the global optimum). In order to handle nonlinear dynamics, a common technique for trajectory optimization is *direct collocation* [9]. In this formulation, the state and control are represented as cubic splines, and the system dynamics are enforced at so-called "collocation points", located at the midpoints of the spline.

The formulation we use is described in [10, §10.3.2]. Given system dynamics $\dot{x} = f(x, u)$, initial conditions $x[0] = x_0$, incremental cost $\ell(x[n], u[n])$, and final cost $\ell_f(x[N])$, the trajectory optimization problem is formulation as

$$\begin{aligned} \min_{x[\cdot],u[\cdot]} \quad & \ell_f(x[N]) + \sum_{n=0}^{N-1} \ell(x[n], u[n]) \\ \text{s. t.} \quad & \dot{x}(t_{c,n}) = f(x(t_{c,n}), u(t_{c,n})) \qquad \forall n \in [0, N-1] \\ & x[0] = x_0 \end{aligned} \tag{6}$$

where $t_{c,n}$ denotes the midpoint between two time steps t_n and t_{n+1}, which are decision variables in the optimization problem. The dynamics constraint approximately enforces the true system dynamics, and leverages the fact that derivatives of splines can be computed analytically. Note that this framework supports arbitrary additional constraints.

In practice, this optimization problem is very brittle. There are a limited number of constraints which we have been able to add, including input limits and initial and final state constraints. However, adding even a simple objective (such as summing) leads to the optimization problem taking a prohibitive amount of time, or failing altogether. System initialization is essential, as SNOPT's default initial conditions frequently lead to a local minimum. We initialize the system with a zero-order hold on the initial control signal, and a first order hold from the initial configuration to the goal configuration.

C. Trajectory Optimization via Differential Flatness

Differential flatness is a property of a system that allows the state and controls of the system to be recovered from trajectories in an abstract output space that has no implicit coupling between any of the output variables. We use the definition given in [1], [11]:

Definition. *A system $\dot{x} = f(x, u)$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, is differentially flat if there exists a mapping $z = z(x, u\dot{u}, \ldots, u^{(p)})$, such that the states and inputs can be expressed by mappings $x = x(z, \dot{z}, \ldots, y^{(q)})$, and $u = u(z, \dot{z}, \ldots, z^{(q)})$, where p and q are finite integers.*

While optimization procedures to plan trajectories are often nonconvex in nonlinear systems, finding trajectories in the output space may be convex. State and control trajectories can then be recovered by applying the definition of differential flatness.

Our system, while using Hookian springs, is still differentially flat with respect to the output map given by [1]. We subsantiate our claim by showing that we can solve for the state and control variables.

Theorem 1. *Let $f(x, u)$ be a system as defined by Eqs. 1- 4. Suppose that $T_i > 0$ for all i. Let $z(x_L, x_1, r_1, \ldots x_n, r_n) = (T_2 q_2, \ldots, T_n q_n, \psi_1, \ldots, \psi_n, x_L)$ be an output mapping. Then there exists mapping x, u such that $(x_L, x_1, r_1, \ldots x_n, r_n) = x(z, z', '', \ldots, z^{(6)})$ and $(u_1^1, u_2^1, u_3^1, u_4^1, \ldots u_1^n, u_2^n, u_3^n, u_4^n) = u(z', z'', \ldots, z^{(6)})$. In other words, the system $f(x, u)$ is differentially flat.*

Proof. We can solve for the first tension force $T_1 q_1$ by manipulating equation (4):

$$T_1 q_1 = -m_L(\ddot{x}_L + g) - \sum_{i=2}^{n} T_i q_i.$$

We must solve $T_1 q_1$ and its derivatives up to the fourth degree, which requires the remaining $T_i q_i$, $i \in [2, \ldots, n]$, to be four-times differentiable and the mass trajectory x_m to be six-times differentiable.

Given $T_i q_i$, we can compute T_i by computing $\langle T_i q_i, q_i \rangle$. Higher derivatives of T_i can be obtained by differentiating that expression. Tension directions q_i can be computed by $T_i q_i / T_i$. Differentating $T_i q_i = T_i \cdot q_i$ can be used to compute the higher derivates of q_i.

Since $T_i > 0$ for all i, then we know that $||x_i - x_L|| > L$. We can compute the quadrotor position x_i by manipulating equation (1):

$$x_i = \frac{T_i q_i}{K} + x_L + L q_i$$

Higher derivatives of x_i can be obtained by differentiating the expression above and substituting in the known quantities.

According to [2], quadrotor dynamics also have the differential flatness property with respect to the output mapping $(x_i, r_i) \mapsto (x_i, \psi_i)$. The full state and control of the ith quadrotor (x_i, r_i) and $(u_1^i, u_2^i, u_3^i, u_4^i)$ can be computed from a four-times differentiable output trajectory $(x_i, \psi_i, x_i', \psi_i', \ldots, x_i^{(4)}, \psi_i^{(4)})$. □

Remark. *This proof is largely the same as the one seen in [1], though the solve order changes because we model cables as springs.*

We exploit the differential flatness property by planning six-times differentiable trajectories in the flat output space, and then compute the corresponding state and control trajectory used for trajectory stabilization. We expect a user to designate a finite sequence of waypoints $\{(x_1, r_1, \ldots, x_n, r_n, x_L)\}_{j=1}^N$ and corresponding times $\{t_0, \ldots, t_N\}$ through which the system should pass. We apply our output map to the sequence of waypoints to yield new waypoints in the output space $\{(z_1^{(i)}, \ldots, z_m^{(i)})\}_{i=1}^N$. We then solve the following quadratic program to obtain a six-continuously differentiable piecewise-polynomial for the kth component of the output waypoint trajectory:

$$\min_{\substack{a_0^{(i)}, a_1^{(i)}, \ldots, a_7^{(i)} \\ \text{for } i=1,\ldots,N}} \left[a_0^{(1)}, \ldots, a_7^{(1)}, a_1^{(2)}, \ldots, a_7^{(N)}\right] \begin{bmatrix} a_0^{(1)} \\ \vdots \\ a_7^{(1)} \\ a_1^{(2)} \\ \vdots \\ a_7^{(N)} \end{bmatrix}$$

$$\begin{aligned} \text{s.t.} \sum_{j=0}^{7} a_j^{(i)} (t_i)^j - \sum_{j=1}^{7} a_j^{(i+1)} (t_i)^j = 0 \\ \sum_{j=d}^{7} \frac{(j!) a_j^{(i)}}{(7-d)!} (t_i)^{j-d} - \sum_{j=d}^{7} \frac{(j!) a_j^{(i+1)}}{(7-d)!} (t_i)^{j-d} = 0 \\ \text{for } i = 1, \ldots, n-1, \; d = 1, \ldots 6 \\ \sum_{j=d}^{7} a_j^{(i)} (t_i)^j = z_k^{(i)}, \; \text{for } i = 1, \ldots, N \end{aligned}$$

where our objective minimizes the coefficients of our piecewise polynomials. Our first constraint ensures that the polynomials connect at each of the waypoints and the second constraint ensures that their derivatives match at the waypoints as well. The last constraint ensures the polynomials pass through the waypoints.

We then solve for the corresponding states and control actions by applying the technique outlined in the proof of Theorem 1. We treat the solution as a nominal trajectory to be used for trajectory stabilization.

V. Control

The optimal control problem asks for the minimum cost controller for a given dynamical system. We primarily focus on controllers based on the *linear quadratic regulator* [10, §8]. Given a linear dynamical system $\dot{x} = Ax + Bu$, the linear quadratic regulator (henceforth abbreviated LQR) is the controller that minimizes the infinite horizon cost

$$J = \int_0^\infty x^T Q x + u^T R u \, dt \tag{7}$$

where $Q \succeq 0$ represents the infinite horizon state cost, and $R \succ 0$ represents the infinite horizon control cost. The optimal stabilizing controller will be of the form $u = -Kx$, where $K = R^{-1} B^T S$, and S is a matrix satisfying the algebraic Riccati equation

$$0 = SA + A^T S - SBR^{-1} B^T S + Q \tag{8}$$

The LQR can be extended to handle smooth, nonlinear systems via a linearization of the dynamics about a fixed point. However, such a controller does not model the nonsmooth hybrid dynamics of our system when cables change between slack and taut. LQR can also be generalized to finite-horizon and time-varying problems, which enables stabilizing about a trajectory.

When stabilizing to fixed points obtained from SNOPT, the numerics are often imperfect. This can make it impossible to solve Equation (8). The finite-horizon cost is much more robust, so we are able to use it in all of our experiments.

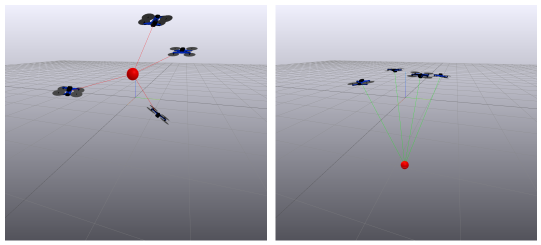

(a) Initial state (b) Final state

Fig. 4: The hybrid dynamics do not seem to harm the ability of the finite-horizon LQR controller to stabilize the trajectory. Even when the system is initialized with all cables slack, as shown in, the controller is able to drive the system to the fixed point.

VI. Results

We demonstrate our ability to stabilize the system to an equilibrium point, and to follow a dynamically feasible trajectory. We then compare the trajectories from nonlinear optimization with those obtained using differential flatness.

A. Stabilization to a Point

We first demonstrate our system's ability to stabilize to an equilibrium point. Given a fixed point from the nonlinear optimization described in Section IV-A, we construct a finite-horizon LQR controller. For costs, we use diagonal Q and R matrices. R is set to the identity, and Q is defined by $Q_{ii} = 10$ if state index i corresponds to a quadrotor pose or free body position (but not orientation), and 1 otherwise. We use a time horizon of 10 seconds for our controller.

Empirically, this controller is very effective. Under a wide variety of initial conditions, it is able to stabilize the system to the fixed point. This even includes cases where some or all of the cables are initially slack, as shown in Figure 4. In this case, the quadrotors begin to navigate themselves to the equilibrium point, independent of the mass falling. Once the mass has fallen enough for the cables to come into tension, the quadrotors are then able to exert a force on the mass, pulling it to the desired position.

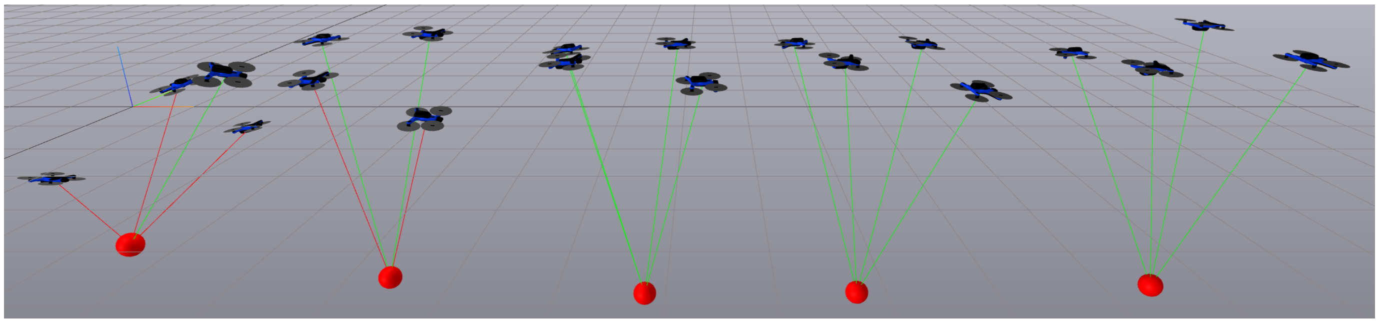
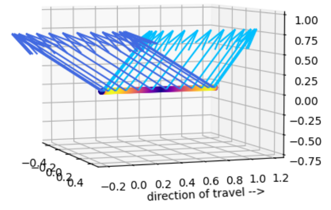
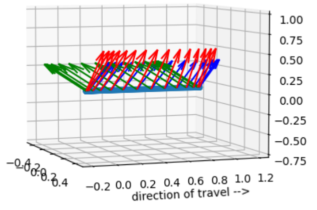


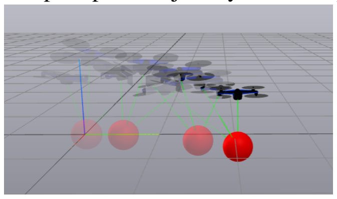
Fig. 5: The finite-horizon LQR controller stabilizing the system around a simple trajectory. (The system evolves left to right.) The trajectory was obtained via the direct collocation method, as described in Section IV-B.



(a) Trajectory of mass and tension forces planned in output space.



(b) Computed corresponding trajectory in state space.



(c) Trajectory produced by the tracking finite-horizon LQR controller.

Fig. 6: In (a), mass trajectory is shown by the line, with acceleration magnitude corresponding to color brightness. Arrows represent the tension forces in the output map. In (b), the mass trajectory is shown by the line and quadrotor positions shown by the arrows.

B. Stabilization to a Trajectory

Next, we demonstrate our system's ability to stabilize along an approximately dynamically feasible trajectory, obtained from the optimization problem described in Section IV-B. This trajectory is given as a piecewise polynomial, which does not perfectly satisfy the dynamics of the system at all points in time. However, if the optimization was successful, it should be close enough that a local stabilizing controller can follow it.

We construct a finite-horizon LQR controller by linearizing around this nominal trajectory. Once again, the controller is very robust to initial conditions, including cables being slack initially. An example trajectory is shown in Figure 5.

C. Optimization Dynamics Directly vs. Differential Flatness

To show the robustness of our differential-flatness-based pipeline against a nonlinear trajectory optimization in the state space, we found a simple trajectory (Fig. 6) that could be solved by our method, but not by SNOPT. The trajectory problem was to find a path between the two waypoints, with the system starting and ending at an equilibrium position subject to a smooth acceleration constraint. While SNOPT is able to find solutions when specifying nonlinear programs with only our dynamic constraints and waypoints, the solver fails to find a solution when we add any additional constraints or even specify a cost objective.

While solving for the trajectory is no longer a difficult problem, since we initialize our system with a random configuration, there are instances when the finite horizon LQR controller does not completely converge to the trajectory (see Fig. 6c).

VII. Discussion

To ensure that we had a well-defined differential-flatness mapping, we assumed the quadrotor cables would always be in tension. Our last assumption removed a large component that made our system interesting: the hybrid nature of the cable tension. While our assumption that the cables would always be in tension was not crippling (the mass is *very likely* to fall and bring all the cables back into tension), explicitly incorporating the hybrid nature of the system in our control system design would enable us to design more interesting trajectories.

We chose to use the same output map as [1] for our trajectory optimization, but found the output space difficult to work with designing trajectories. Since the map includes the tension forces from all cables, except for one, a common strategy (also used by Sreenath and Kumar) was to fix the tension forces for all the cables in the output map, and let the remaining one do the work of accelerating the mass in more extreme maneuvers. While not obvious, finding (or solving a mathematical program to produce an) output map that serves a more user-friendly design space is an interesting theoretical and user design-oriented question.

When attempting to solve the quadratic program for the coefficients of the piecewise-polynomial trajectory, Drake's default quadratic program solver was brittle. Even for problems with two waypoints (like the one shown in 6, the solver failed to find third-degree polynomials to connect them. After switching to Mosek, we could reliably find piecewise polynomial trajectories using degree-7 polynomials that were six-times differentiably continuous.

As always, our system had a large number of simplfying assumptions: no tangling of massless cables, no collisions, etc. Relaxing these assumptions and account for them in our control system design will be necessary to bring these ideas to a physical quadrotor team.

Acknowledgment

We thank Bernhard Paus Græsdal for pointing to the right papers to write down our differential-flatness output maps for our quadrotors.

Statement of Contributions

Thomas Cohn implemented the code to find equilibrium points, solve the naive trajectory optimization problem, and compute stabilizing controllers. Seiji Shaw derived the differential flatness results, and implemented the associated output map and trajectory optimization.

References

[1] K. Sreenath and V. Kumar, "Dynamics, Control and Planning for Cooperative Manipulation of Payloads Suspended by Cables from Multiple Quadrotor Robots," in *Robotics: Science and Systems IX*, Robotics: Science and Systems Foundation, June 2013.

[2] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2520–2525, May 2011. ISSN: 1050-4729.

[3] C. J. Adams, *Modeling and control of helicopters carrying suspended loads*. PhD thesis, Georgia Institute of Technology, 2012.

[4] D. K. D. Villa, A. S. Brandão, and M. Sarcinelli-Filho, "A Survey on Load Transportation Using Multirotor UAVs," *Journal of Intelligent & Robotic Systems*, vol. 98, pp. 267–296, May 2020.

[5] T. Lee, M. Leok, and N. H. McClamroch, "Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3)," Sept. 2011. arXiv:1003.2005 [cs, math].

[6] G. Wu and K. Sreenath, "Geometric control of multiple quadrotors transporting a rigid-body load," in *53rd IEEE Conference on Decision and Control*, pp. 6141–6148, Dec. 2014. ISSN: 0191-2216.

[7] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, pp. 99–131, 2005.

[8] P. E. Gill, W. Murray, M. A. Saunders, and E. Wong, "User's guide for SNOPT 7.7: Software for large-scale nonlinear programming," Center for Computational Mathematics Report CCoM 18-1, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2018.

[9] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[10] R. Tedrake, *Underactuated Robotics*. 2023.

[11] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," in *ASME international mechanical engineering congress and exposition*, Citeseer, 1995.