DECENTRALIZED MODEL PREDICTIVE CONTROL OF SWARMS OF SPACECRAFT

Daniel Morgan[(1)], **Soon-Jo Chung**[(2)], **and Fred Y. Hadaegh**[(3)]

[(1)] *University of Illinois at Urbana-Champaign, 306 Talbot Lab, 104 S. Wright St., Urbana, IL 61801, (608) 852-4824, morgan29@illinois.edu*
[(2)] *University of Illinois at Urbana-Champaign, 306 Talbot Lab, 104 S. Wright St., Urbana, IL 61801, (217) 244-2737, sjchung@illinois.edu*
[(3)] *Jet Propulsion Laboratory, M/S 198-326, 4800 Oak Grove Dr., Pasadena, CA 91109, (818) 354-8777, Fred.Y.Hadaegh@jpl.nasa.gov*

Abstract: *This paper presents a decentralized, model predictive control (MPC) algorithm for the reconfiguration of swarms of spacecraft composed of hundreds to thousands of agents with limited capabilities. This paper develops algorithms for the swarm reconfiguration which involves transferring spacecraft from one J_2-invariant orbit to another while avoiding collisions and minimizing fuel. The algorithm uses sequential convex programming (SCP) to solve a series of approximate path planning problems until the solution converges. The MPC-SCP algorithm results in decentralized computations and communications between neighboring spacecraft only.*

Keywords: *Swarms, Collision Avoidance, Guidance, Reconfiguration, Optimal*

1 Introduction

Spacecraft formation flying continues to be a major area of research. Recently, the idea of formation flying has been expanded to include swarms of spacecraft [1], which contain hundreds to thousands of 100-gram class spacecraft, known as femtosats. Due to their small size, the femtosats have limited communication and computation capabilities, which require the guidance and control algorithms to be decentralized so that they are computationally efficient. Additionally, the algorithms must be fuel efficient since the femtosats will not be able to carry much fuel due to their small size.

J_2-invariant orbits [2] are excellent choices for swarm keeping and provide collision-free motion for hundreds of orbits. However, guidance and control algorithms for swarm reconfiguration must also be developed. The goal of this paper is to develop a fuel and computationally efficient guidance algorithm for the reconfiguration of a swarm of spacecraft located in low Earth orbit (LEO). This algorithm must transfer each spacecraft from one J_2-invariant orbit to another. In addition to fuel and computational efficiency, collision avoidance is a key requirement of the algorithm. This is a difficult problem in the highly nonlinear dynamics of relative spacecraft motion in the presence of J_2, which is the dominant perturbation in LEO.

Previous work in spacecraft formation flying [3, 4] and robotic swarm research [5, 6, 7] has developed many multivehicle guidance methods. However, the work in formation flying usually deals with a small number of high-capability spacecraft, a dozen

at the most. The algorithm for swarm reconfiguration must be different from the current literature because it must simultaneously address the large number of agents, the modest capabilities of each individual agent, and the complex dynamic environment.

Recently, convex programming [8] has been used in multi-vehicle optimization problems and can be efficiently solved to find a global optimum. However, approximating the collision avoidance constraints with a convex constraint creates an overly conservative collision avoidance region. In this paper, sequential convex programming (SCP) [9] is used to solve for the trajectories required for swarm reconfiguration. SCP is an iterative process which ensures that the convex approximations of nonconvex constraints are accurate Additionally, using multiple iterations allows for more fuel-efficient trajectories. Additionally, freely available software, such as CVX [10], can be used to solve the convex programming problems.

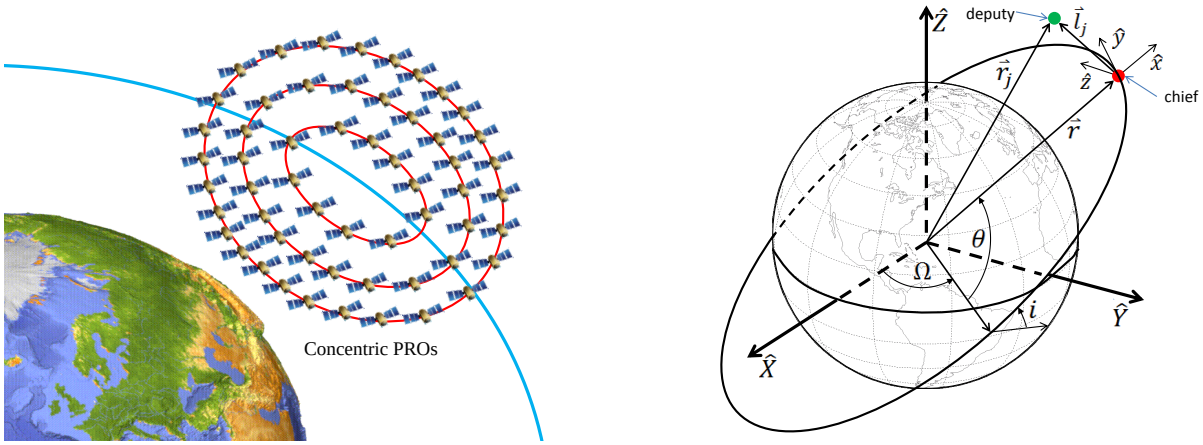
The SCP algorithm can run fast enough to be used in real-time and, therefore, it can be used in model predictive control (MPC) approach. This method updates the trajectories with new state information that may have arisen because of sensing or actuation errors, or unmodeled disturbances.

The most important feature of the MPC-SCP algorithm is that it decentralizes the computation and communication requirements for the swarm reconfiguration with collision avoidance. This allows the algorithm to be run onboard the femtosats in real time for a swarm with hundreds to thousands of spacecraft. Additionally, the MPC-SCP algorithm provides some robustness to errors or uncertainties by taking the actual state information into account when computing the future trajectories. This robustness is not present in an open-loop, optimal control approach.

The paper is organized as follows. In Section 2, the swarm reconfiguration is discussed and the optimal control formulation is described. In Section 3, the problem is converted to convex form and a decentralized SCP algorithm is developed. In Section 4, the MPC-SCP algorithm is implemented and the effectiveness of this algorithm is investigated. In Section 5, the results of simulations and the effectiveness of the MPC-SCP algorithm is discussed.

2 Guidance of Swarms of Spacecraft

In this section, the swarm reconfiguration is described by a nonlinear, continuous, optimal control problem. The swarm reconfiguration transfers hundreds to thousands of spacecraft from one J_2-invariant orbit [2],shown in Fig. 1a, to another while preventing collisions and minimizing the total fuel usage. To properly define the variables and constraints involved in the optimization problem, two coordinate systems must be defined. First, the Earth Centered Inertial (ECI) coordinate system is used to locate the chief, or reference, orbit (see Fig. 1b). This coordinate system is inertially fixed at the center of the Earth. The $\hat{X}$ direction points towards the vernal equinox, the $\hat{Z}$ direction points towards the north pole, and the $\hat{Y}$ direction is perpendicular to the other two and completes the right-handed coordinate system. The second coordinate system is the local vertical, local horizonal (LVLH) coordinate system. The LVLH frame is centered at the chief orbit. Figure 1b shows the LVLH frame with respect to a chief spacecraft. The $\hat{x}$, or radial, direction is always aligned with the position vector and points radially outward from the Earth, the $\hat{z}$, or crosstrack, direction is aligned with the angular momentum vector, and the $\hat{y}$, or alongtrack, direction completes the right-handed coordinate system. The LVLH frame is a rotating frame with a rotation rate of ω_x about the



(a) Spacecraft Swarm

(b) ECI $(\hat{X}, \hat{Y}, \hat{Z})$ and LVLH Frames $(\hat{x}, \hat{y}, \hat{z})$

Figure 1: A visualization of the relative coordinate system and a spacecraft swarm [2]

radial axis and ω_z about the crosstrack axis. The relative state of the deputy spacecraft in the LVLH frame is described by $\mathbf{x}_j = [\, x_j \quad y_j \quad z_j \quad \dot{x}_j \quad \dot{y}_j \quad \dot{z}_j \,]^T$.

The optimization problem for swarm reconfiguration is written using the LVLH frame. The equations of motion for spacecraft's position in the LVLH frame ($\boldsymbol{\ell}_j = (x_j, y_j, z_j)^T$) are [11]

$$\ddot{\boldsymbol{\ell}}_j = -2\mathbf{S}(\boldsymbol{\omega})\dot{\boldsymbol{\ell}}_j - \mathbf{g}(\boldsymbol{\ell}_j, \text{œ}) + \mathbf{u}_j \tag{1}$$

where the function $\mathbf{g}(\boldsymbol{\ell}_j, \text{œ}) \in \mathbb{R}^3$ is defined in Ref. [12] and the matrix $\mathbf{S}(\boldsymbol{\omega}) \in \mathbb{R}^{3\times3}$ is defined as $\mathbf{S}(\boldsymbol{\omega})\dot{\boldsymbol{\ell}} = \boldsymbol{\omega} \times \dot{\boldsymbol{\ell}}$. Additionally, the orbital elements of the chief (reference) orbit, $\text{œ} = (r, v_x, h, i, \Omega, \theta)^T$, are geocentric distance (r), radial velocity (v_x), angular momentum (h), inclination (i), right ascension of the ascending node (Ω), and argument of latitude (θ). Note that the angular rates of the LVLH frame, $\boldsymbol{\omega}$ are also determined by œ (see Ref. [12]). These values are independent of the deputy spacecraft's motion so it is assumed that these values can be computed by each spacecraft using some standard estimation process that uses communicated or measured information and propagation of the following equation of motion

$$\frac{d\text{œ}}{dt} = \mathbf{f}_{\text{chief}}(\text{œ}) \tag{2}$$

where the RHS of this equation is defined in Ref. [12]. Note that Eqs. (1) and (2) are hierarchical; Eq. (2) does not depend on Eq. (1). Hence, the reference orbital elements cannot be affected by individual spacecraft and do not need to be considered variables in the optimization. Therefore, the dynamics constraints are given by Eq. (1). In addition to the dynamics constraints, the following constraints must be enforced as well.

$$\|\mathbf{u}_j(t)\| \leq U_{\max} \qquad \forall t \in [0, t_f], \qquad j = 1, \ldots, N \tag{3}$$

$$\|C[\mathbf{x}_j(t) - \mathbf{x}_i(t)]\| \geq R_{\text{col}} \qquad \forall t \in [0, t_f], \qquad j = 1, \ldots, N-1 \tag{4}$$

$$\mathbf{x}_j(0) = \mathbf{x}_{j,0} \qquad j = 1, \ldots, N \tag{5}$$

$$\mathbf{x}_j(t_f) = \mathbf{x}_{j,f} \qquad j = 1, \ldots, N \tag{6}$$

where $C = [\mathbf{I}_{3\times3} \quad \mathbf{0}_{3\times3}]$ and $\mathbf{x}_j = (\boldsymbol{\ell}^T, \dot{\boldsymbol{\ell}}^T)^T$. Equation (3) is the constraint describing the maximum allowable control magnitude (U), Eq. (4) is the collision avoidance

constraint describing the minimum allowable distance between two spacecraft (R), and Eqs. (5)-(6) are the initial and terminal state constraint, respectively. Any reachable conditions can be used for Eqs. (5)-(6), but the simulations in Sec. 5 use the J_2-invariant conditions developed in our prior work [2], which provide the velocities required for J_2-invariant orbits for a given position.

The swarm reconfiguration goal is to minimize fuel. Therefore, the swarm reconfiguration is formulated as follows:

Problem 1: Nonconvex Optimization

$$\min_{\mathbf{u}_j, j=1,\ldots,N} \sum_{j=1}^{N} \int_0^{t_f} \|\mathbf{u}_j(t)\| dt \quad \text{subject to} \quad \{(1), (3)-(6)\} \tag{7}$$

The objective function and the constraints of Eqs. (3), (5), and (6) already meet the requirements for convex programming. Therefore, only the dynamics, Eq. (1), and the collision avoidance constraints, Eq. (4), need to be convexified.

3 Sequential Convex Programming

In this section, convexification is presented. This is done by converting both Eq. (1) and (4) into an acceptable form for convex programming. For the dynamics, this involves linearizing Eq. (1) and discretizing the state and control from Problem 1. This results in a finite number of affine equality constraints, which are acceptable in a convex programming problem. The collision avoidance constraints in Eq. (4) are converted to convex inequality constraints. Once the problem is in convex form, a SCP algorithm is applied to solve the swarm reconfiguration.

3.1 Linearization and Discretization of Dynamics

To convexify Problem 1, the dynamics constraints in Eq. (1) are linearized and the state and control variables are discretized. This process is shown in detail in our prior work [12] and approximates Eq. (1) as

$$\mathbf{x}_j[k+1] = A_d(\bar{\mathbf{x}}_j, \text{œ})\mathbf{x}_j[k] + B_d\mathbf{u}_j[k] + c_d(\bar{\mathbf{x}}_j, \text{œ}), \quad k = 0, \ldots, T-1, \quad j = 1, \ldots, N \tag{8}$$

where $\mathbf{x}_j[k] = \mathbf{x}_j(t_k)$, $\mathbf{u}_j[k] = \mathbf{u}_j(t_k)$, $\text{œ}[k] = \text{œ}(t_k)$. The constraints from Eqs. (3)-(6) are approximated by

$$\|\mathbf{u}_j[k]\| \leq U \qquad k = 0, \ldots, T-1, \qquad j = 1, \ldots, N \tag{9}$$

$$\|C(\mathbf{x}_j[k] - \mathbf{x}_i[k])\| \geq R \qquad k = 0, \ldots, T, \qquad j = 1, \ldots, N-1 \tag{10}$$

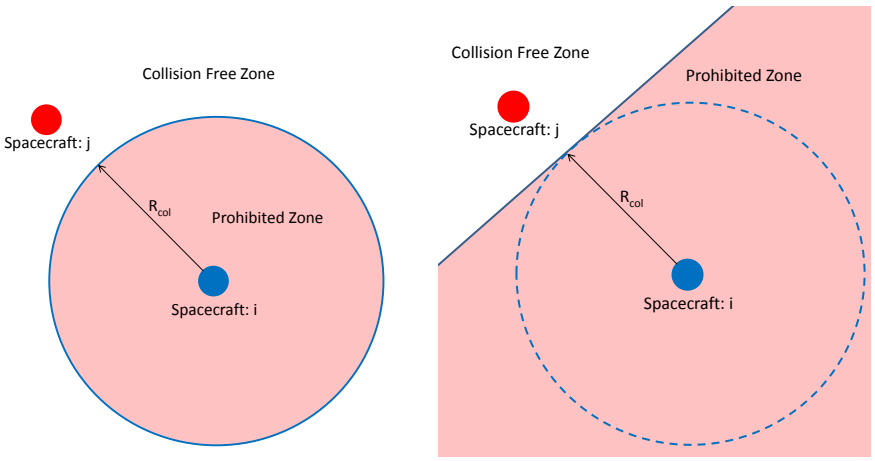
$$\mathbf{x}_j[0] = \mathbf{x}_{j,0} \qquad j = 1, \ldots, N \tag{11}$$

$$\mathbf{x}_j[T] = \mathbf{x}_{j,f} \qquad j = 1, \ldots, N \tag{12}$$

Note that the only constraint that does not satisfy the rules of convex programming is Eq. (10). This constraint will be modified in the next section so that it is a convex inequality constraint.

3.2 Convexification of Collision Avoidance Constraints

The final step in converting the swarm reconfiguration into a convex program is convexifying the collision avoidance constraints. Since the collision avoidance constraints currently form the complement of a convex set, the best convex approximations will be affine constraints. In other words, the sphere which defines the forbidden region is approximated by a half space which is tangent to the sphere and perpendicular to the line segment connecting the nominal positions ($\bar{\mathbf{x}}_j$) of the spacecraft and the object. This idea is shown in 2-D using a line and a circle in Fig. 2.



(a) Nonconvex prohibited zone

(b) Convex approximation of prohibited zone

Figure 2: Convexification of the 2-D collision avoidance constraint [12]

Figure 2a shows the prohibited zone for the initial collision avoidance constraint. Figure 2b demonstrates the convexification of the constraint from Fig. 2a. Based on the positions of the two spacecraft in the previous iteration, a line (or plane in the 3-D version) is defined to be tangent to the old prohibited zone and perpendicular to the line segment connecting the spacecraft. This line defines the half space, which is new prohibited zone. As can be seen in Fig. 2b, the new prohibited zone contains the old prohibited zone, so collision avoidance using the convex prohibited zone implies collision avoidance using the original prohibited zone, which is ultimately what is required. When multiple neighbors are considered, the collision free zone will be the intersection of the half spaces that define the collision free zones between each neighbor and spacecraft j. This results in a convex polytope around the nominal position of spacecraft j in which it is guaranteed to be collision free based on the position of the neighboring spacecraft.

Using this idea, a sufficient condition for the collision avoidance constraints to hold from Eq. (10) will be

$$(\bar{\mathbf{x}}_j[k]-\bar{\mathbf{x}}_i[k])^T C^T C(\mathbf{x}_j[k]-\mathbf{x}_i[k]) \geq R\|C(\bar{\mathbf{x}}_j[k]-\bar{\mathbf{x}}_i[k])\| \quad k=0,\ldots,T, \quad j=1,\ldots,N-1 \tag{13}$$

The sufficiency of this condition is shown in Ref. [12]. $\bar{\mathbf{x}}_i[k]$ and $\bar{\mathbf{x}}_j[k]$ are the nominal trajectories from the previous iteration of the SCP algorithm. These nominal values are assumed to be known and are not variables in the optimization. Therefore, the collision avoidance constraints in Eq. (13) are affine and can be used in a convex program.

3.3 Sequential Convex Programming Algorithm

Now that Problem 1 has been converted to convex programming form, it can be written as the following convex programming problem:

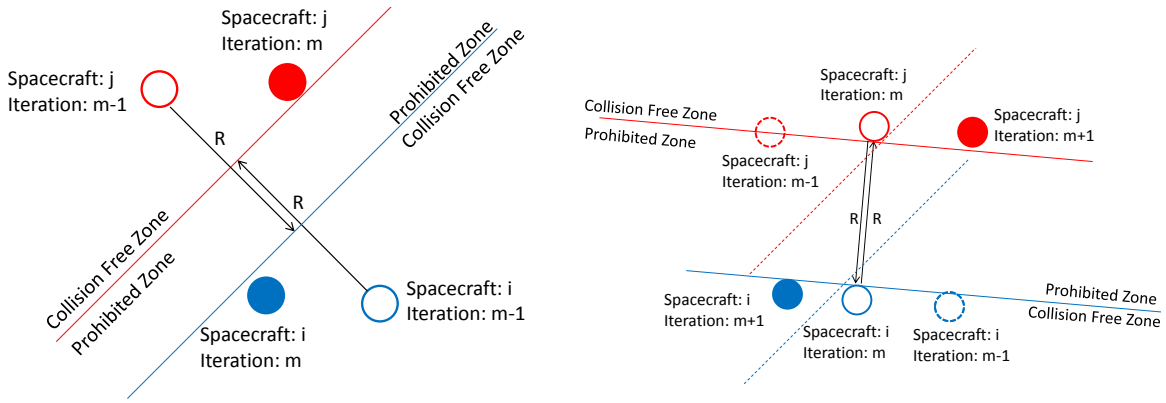
Problem 2: SCP

$$\min_{\mathbf{u}_j, j=1,\ldots,N} \sum_{j=1}^{N} \sum_{k=0}^{T-1} \|\mathbf{u}_j[k]\| \Delta t \quad \text{subject to} \quad \{(8), (9), (11), (12), (13)\} \tag{14}$$

where Problem 1 has been discretized and the constraints of Eqs. (1) and (4) have been approximated by Eqs. (8) and (13), respectively.

The approximations used to get the dynamics and collision avoidance constraints into convex forms, Eqs. (8) and (13), require a nominal state $\bar{\mathbf{x}}_j[k]$ for each spacecraft at each time step. Additionally, the nominal vectors must be accurate in order for the solution to the convex programming problem to be valid. To ensure that the nominal vectors are good estimates of the actual state vectors, SCP is used. In SCP, $\bar{\mathbf{x}}_j^m[k] = \mathbf{x}_j^{m-1}[k]$ for iteration m, i.e. the actual state trajectory from the previous iteration becomes the nominal state trajectory in the current iteration. To enforce the collision avoidance constraints, each spacecraft communicates its nominal trajectory to the other spacecraft.

One of the main advantages of using SCP compared to simply solving the convex programming problem is that the initial guess is not as critical. Because of the way that the collision avoidance constraint were convexified, the prohibited zone for each collision is overly conservative. With convex programming, this will prevent the spacecraft from achieving some trajectories that are safe. This can potentially lead to non-optimal trajectories if a poor initial guess is provided. In SCP, the iterations allow the spacecraft to move into an area that was originally prohibited but is actually safe. This is illustrated in Fig. 3. In iteration $m+1$, shown in Fig. 3b, the spacecraft move into areas that were originally prohibited in iteration m, shown in Fig. 3a. This idea is how SCP can achieve more optimal trajectories than a single convex programming problem.



(a) Collision free zone for iteration m

(b) Collision free zone for iteration m+1

Figure 3: Evolution of the 2-D collision avoidance constraint [12]

In the first iteration, an initial guess must be provided for the nominal vector but in the following iterations, the solution to the previous iteration is used as the nominal vector. This process continues until the solutions to the sequence of convex problems

converges. In each iteration, a trust region is defined for the convex program. The trust region represents the range of state vectors over which the linearization is accurate. It is defined as

$$\mathcal{X}_{L^m} = \{\mathbf{x}_j |\ \|\mathbf{x}_j - \bar{\mathbf{x}}_j\| \leq L^m\} \tag{15}$$

where L^m is the size of the trust region during iteration m. Additionally, the trust region in Eq. (15) can be contracted to ensure that the SCP algorithm converges. In order to ensure convergence, the size of the trust is updated according to the following equation.

$$L^{m+1} = \beta L^m \tag{16}$$

where $\beta \in (0, 1)$ is a parameter that determines the worst-case rate of convergence.

Even in convex form, the swarm reconfiguration algorithm will scale poorly because the collision avoidance constraints are coupled. Therefore, the problem must be decoupled so that the computations can be decentralized. Therefore, the collision avoidance constraints must be written in such a way that each spacecraft can compute its own trajectory yet the entire swarm is still collision free.

The first step to decentralizing the SCP algorithm is limiting the pairs of spacecraft for which collision avoidance is considered. By only checking for collisions between spacecraft pairs that were close to colliding in a previous iteration of the SCP algorithm, the number of constraints in each iteration of Problem 2 can be greatly reduced. Also, the nominal state vectors become better estimates of the actual state vectors as the number of iterations increases. This fact can be used to decentralize the optimizations by assuming that all other spacecraft are fixed objects, located at their positions from the preceding iteration. Using this assumption, Eq. (13) can be rewritten as

$$\begin{gathered}(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])^T C^T C(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k]) \geq R\|C(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\| \\ k = 0, \ldots, T, \quad i \in \mathcal{I}_j, \quad j = 1, \ldots, N-1\end{gathered} \tag{17}$$

where

$$\mathcal{I}_j = \{i |\ \exists\ k \in 1, \ldots, T \text{ such that } \|C(\mathbf{x}_i[k] - \mathbf{x}_j[k])\| \leq R_{\text{safe}} \text{ and } i < j\} \tag{18}$$

where R_{safe} is the minimum distance between two spacecraft so that collision avoidance is not considered in the next SCP iteration and the constraint $i < j$ guarantees that only one spacecraft will avoid the other one.

Defining the collision avoidance constraints by Eq. (17) allows each spacecraft to perform its own computations while still maintaining a collision-free swarm. The decentralized problem can be written as follows:

Problem 3: Decentralized SCP

$$\min_{\mathbf{u}_j, j=1,\ldots,N} \sum_{j=1}^{N} \sum_{k=0}^{T-1} \|\mathbf{u}_j[k]\| \Delta t \quad \text{subject to} \quad \{(8), (9), (11), (12), (17)\} \tag{19}$$

4 Model Predictive Control

4.1 Problem Formulation

In this section, the decentralized SCP algorithm described in Problem 3 is formulated as a MPC problem. To describe the MPC-SCP algorithm, Problem 4 and Problem 5 are defined. Problem 4 is defined for cases when the MPC horizon does not

reach the terminal time for the reconfiguration. For this reason, the terminal constraint, Eq. (12), is not enforced in Problem 4. Instead, a terminal cost ($h_j(\mathbf{x}[k], k)$) is added to the objective to estimate the cost of completing the reconfiguration from the state and time at the end of the horizon. Problem 5 is very similar to Problem 3 with the only difference being the starting time. Problem 5 is used in the MPC-SCP algorithm when the horizon reaches the terminal time. In both Problem 4 and Problem 5, the spacecraft are assumed to have limited communication range. Only pairs of spacecraft within a certain distance will have collision constraints. Problem 4 and Problem 5 are expressed as follows:

Problem 4: Convex Optimization with Terminal Cost for $k_0 + T_H < T$

$$\min_{\mathbf{u}_j} \sum_{k=k_0}^{k_0+T_H-1} \|\mathbf{u}_j[k]\|\Delta t + h_j\left(\mathbf{x}_j[k_0+T_H], k_0+T_H\right) \qquad \forall j = 1,\ldots,N \tag{20}$$

subject to

$$\mathbf{x}_j[k+1] = A_d(\bar{\mathbf{x}}_j, \text{œ})\mathbf{x}_j[k] + B_d\mathbf{u}_j[k] + c_d(\bar{\mathbf{x}}_j, \text{œ}), \tag{21}$$
$$k = k_0,\ldots,k_0+T_H-1, \quad j = 1,\ldots,N$$
$$(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])^T C^T C(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k]) \geq R_{\text{col}}\|C(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\| \tag{22}$$
$$k = k_0,\ldots,k_0+T_H, \quad \{i,j\} : i \in \mathcal{N}_j, \quad \mathcal{N}_j = \{i | i > j, \|\mathbf{x}_j[k_0] - \mathbf{x}_i[k_0]\| \leq R_{\text{comm}}\}$$
$$\|\mathbf{u}_j[k]\| \leq U_{\max} \qquad k = k_0,\ldots,k_0+T_H-1, \qquad j = 1,\ldots,N \tag{23}$$
$$\mathbf{x}_j[k_0] = \mathbf{x}_{j,\text{MPC}} \qquad j = 1,\ldots,N \tag{24}$$

Problem 5: Convex Optimization with Terminal Constraint for $T - T_H \leq k_0 < T$

$$\min_{\mathbf{u}_j} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|\Delta t \qquad \forall j = 1,\ldots,N \tag{25}$$

subject to (24) and

$$\mathbf{x}_j[k+1] = A_d(\bar{\mathbf{x}}_j, \text{œ})\mathbf{x}_j[k] + B_d\mathbf{u}_j[k] + c_d(\bar{\mathbf{x}}_j, \text{œ}), \tag{26}$$
$$k = k_0,\ldots,T-1, \qquad j = 1,\ldots,N$$
$$(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])^T C^T C(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k]) \geq R_{\text{col}}\|C(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\| \tag{27}$$
$$k = k_0,\ldots,T, \quad \{i,j\} : i \in \mathcal{N}_j, \quad \mathcal{N}_j = \{i | i > j, \|\mathbf{x}_j[k_0] - \mathbf{x}_i[k_0]\| \leq R_{\text{comm}}\}$$
$$\|\mathbf{u}_j[k]\| \leq U_{\max} \qquad k = k_0,\ldots,T-1, \qquad j = 1,\ldots,N \tag{28}$$
$$\mathbf{x}_j[T] = \mathbf{x}_{j,f} \qquad j = 1,\ldots,N \tag{29}$$

The MPC-SCP algorithm is performed by reducing the horizon of the SCP problem and then solving this problem repeatedly throughout the reconfiguration. Initially, the SCP algorithm is run a finite horizon (T_H). As the spacecraft approaches this horizon in real time, the SCP algorithm is rerun, using the current time (k_0) and position ($\mathbf{x}_{j,\text{MPC}}$), until the new horizon ($k_0 + T_H$). It is important to note that k_0 is the time at the beginning of each MPC-SCP iteration and increases with time. $\mathbf{x}_{j,\text{MPC}}$ is the real-time position and velocity of the spacecraft when the MPC-SCP algorithm is run. This value represents the initial condition used in the SCP algorithm. This process is repeated until the spacecraft reaches the desired position ($\mathbf{x}_{j,f}$) at the final time (T).

The result of the MPC-SCP algorithm is a fully decentralized algorithm that can be run in real time and has robustness to sensor and actuator errors. The decentralization of the swarm reconfiguration algorithm greatly reduces the communication and computation requirements of the femtosats. Additionally, the increased robustness properties of this algorithm reduce the fuel requirements for the femtosats.

4.2 Stability

The stability of the MPC-SCP algorithm is dependent on the terminal cost function ($h_j\left(\mathbf{x}_j[k], k\right)$) in Eq. (20) of Problem 4. To ensure the stability of the MPC-SCP algorithm, the terminal cost function ($h_j\left(\mathbf{x}_j[k], k\right)$) is defined as the solution to the following optimization problem.

Problem 6: Convex Terminal Cost Function

Minimize (25) subject to {(24),(26),(28),(29)}

This terminal cost function evaluates the fuel required to complete the reconfiguration without considering collision avoidance constraints. There are two reasons to ignore the collision avoidance constraints when calculating the terminal cost function. First, the spacecraft can only communicate with other spacecraft within a certain distance of them. Second, the collision avoidance constraints add complexity to the problem so removing them greatly reduces the time required for the computation.

Substituting the terminal cost function that results from Problem 6 into Problem 4 results in the following optimization problem.

Problem 7: Stable Convex Optimization for $k_0 + T_H < T$

Minimize (25) subject to {(22),(24),(26),(28),(29)}

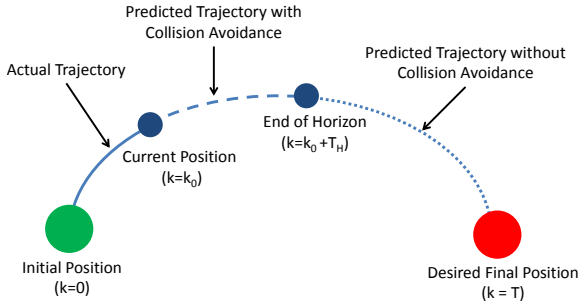


Figure 4: Illustration of the optimization horizon used in the MPC-SCP algorithm [13]

The concept of this terminal cost function is shown in Figure 4. This figure shows the various stages of the MPC-SCP algorithm. The first stage is the solid line in Figure 4. This is the actual trajectory that the spacecraft has traversed and it occurs between $k = 0$ and $k = k_0$. $k = k_0$ represents the current time, and the initial time in the optimization. The next stage occurs between $k = k_0$ and $k = k_0 + T_H$ and is the dashed line in the figure. This represents the predicted trajectory up to the horizon and collision avoidance is considered during this stage. The final stage is the dotted line and extends from $k = k_0 + T_H$ and $k = T$. During this time, collision avoidance is not enforced. If the second stage (dashed line) extends beyond the final time, the

final stage does not exist and Problem 5, which always enforces collision avoidance, should be used instead of Problem 7.

Using Problem 7, which contains the stabilizing terminal cost function, to replace Problem 4 in the MPC-SCP algorithm forces the algorithm to converge. This algorithm uses the solution to Problem 5 or Problem 7 and, in either case, the final state is fixed from Eq. (29). Therefore, the trajectory is forced to converge to the desired final state as long as the optimizations are feasible. The feasibility of the optimizations is discussed in the following subsection.

4.3 Feasibility

For the trajectories from the MPC-SCP algorithm to converge, the optimizations must be feasible. Infeasibility can be caused by two things: The collision avoidance constraints cannot all be satisfied or the terminal constraint cannot be satisfied without violating the limit on the velocity and/or control vectors. The collision avoidance infeasibility arises because other spacecraft can only be detected if they are within the communication radius (R_{comm}). Therefore, collisions that occur with spacecraft outside of the communication radius are not considered until a later time step. For this reason, several conditions are introduced to ensure that collision avoidance is guaranteed.

In order to guarantee feasibility, an artificial velocity constraint is imposed on the problem to bound the distance that each spacecraft can move during each time step. This condition is written as follows.

$$\|D\mathbf{x}_j[k]\| \leq V_{\max} \qquad k = k_0, \ldots, T, \qquad j = 1, \ldots, N \tag{30}$$

where $D = [0_{3\times3} \quad I_{3\times3}]$. Adding this constraint to Problem 7 and Problem 5 yields Problem 8 and Problem 9, respectively.

Problem 8: Feasible Convex Optimization for $k_0 + T_H < T$

Minimize (25) subject to {(22),(24),(26),(28),(29),(30)}

Problem 9: Feasible Convex Optimization for $T - T_H \leq k_0 < T$

Minimize (25) subject to {(24),(26),(27),(28),(29),(30)}

If the optimization problem in Problem 3 is feasible the following conditions ensure that the MPC-SCP algorithm using Problem 8 and Problem 9 is feasible:

Proposition 1: *Detectable Collisions*
All spacecraft that can cause collisions within the current horizon are able to be detected if

$$R_{\text{comm}} \geq 2V_{\max}T_H\Delta t + R_{\text{col}} \tag{31}$$

Proof: This condition guarantees that any spacecraft that could potentially cause a collision before the end of the MPC horizon is detected and therefore considered in the optimization. The length of the horizon is the number of time steps (T_H) multiplied by the length of each time step (Δt). Additionally, the maximum relative velocity between two spacecraft is $2V_{\max}$. Therefore, the maximum change in the relative distance between two spacecraft is given by the first term on the right hand side of Eq. (31). To ensure that all collisions are detected, the difference between the communication radius (R_{comm}) and the collision radius (R_{col}) must be at least as big as this distance. This establishes Eq. (31). □

Proposition 2: *Computational Feasibility*
The new control sequence can be computed before the previous horizon is reached if

$$t_{\text{run}} \leq T_H \Delta t \tag{32}$$

Proof: See Ref. [13].

Propositions 1-2 ensure that infeasibility of the MPC-SCP algorithm is not caused by violations to the collision avoidance constraints. Therefore, the collision avoidance constraints are satisfied and there are no collisions at the discrete time steps. However, it is still possible that collisions occur in between time steps. The following theorem addresses this issue.

Theorem 1: *Collision Avoidance between Time Steps*
If two spacecraft are collision free during two consecutive time steps k and $k+1$ and Eqs. (33)–(34) are satisfied, then the two spacecraft are collision free in the interval $t \in [k\Delta t, (k+1)\Delta t]$.

$$V_{\max} < \frac{R_{\text{col}}}{\Delta t} \tag{33}$$

$$R_{\text{col}} \geq \begin{cases} \sqrt{(\bar{R}_{\text{col}} + \frac{a^* \Delta t^2}{4})^2 + (V_{\max}\Delta t)^2 - \frac{(a^*\Delta t^2)^2}{4}} & \text{if} \quad a^* < \min\{\frac{2V_{\max}}{\Delta t}, a_{\max}\} \\ \sqrt{(\bar{R}_{\text{col}} + \frac{a_{\max} \Delta t^2}{4})^2 + (V_{\max}\Delta t)^2 - \frac{(a_{\max}\Delta t^2)^2}{4}} & \text{else if} \quad a_{\max} < \frac{2V_{\max}}{\Delta t} \\ \bar{R}_{\text{col}} + \frac{V_{\max}\Delta t}{2} & \text{else} \end{cases} \tag{34}$$

where

$$a^* = \frac{2}{\sqrt{3}} \sqrt{\frac{R_{\text{col}}^2}{\Delta t^4} - \frac{V_{\max}^2}{\Delta t^2}} \tag{35}$$

Proof: See Ref. [13].

In addition to infeasibility caused by collision avoidance constraints, infeasibility can also be caused by the constraints on maximum velocity and control magnitudes. This occurs because the spacecraft have a limited communication radius in the MPC-SCP formulation and, therefore, cannot detect collisions occurring after the MPC-SCP horizon. To reduce the likelihood that the maximum velocity causes infeasibility, $V_{\max}$ should be chosen to be as large as possible while still satisfying Propositions 1-2 and Theorem 1. If the optimization is infeasible due to maximum velocity or control constraints, the final time or the terminal state set can be extended to make the optimization feasible.

Remark 1:
When uncertainties are included in the system, forcing the spacecraft to reach a terminal position is unrealistic. Therefore, the terminal state can be replaced with a terminal set as follows:

$$\|\mathbf{x}_j[T] - \mathbf{x}_{j,f}\| \leq \delta \tag{36}$$

where $\delta > 0$ is the radius of a ball around $\mathbf{x}_{j,f}$ which must contain the terminal position of spacecraft j.

Remark 2:
The time step (Δt) is a critical parameter in Propositions 1-2 and Theorem 1. Reducing Δt decreases the required communication radius (R_{comm}) and collision radius (R_{col}) as described in Proposition 1 (Eq. (31)) and Theorem 1 (Eq. (34)), respectively. Additionally, the time step is also used as the discretization time step so reducing it also

reduces the discretization errors in the convexification process. However, reducing the time step increases the number of variables in the optimization, which will increase the run time (t_{run}). This will make it more difficult to satisfy Proposition 2 (Eq. (32)).

To achieve the benefits of a small time step without making the optimization too large for MPC, separate time steps are defined for the time period before the end of the horizon (Δt_1) and for the time period after the horizon (Δt_2). Since collisions and communications are not considered after the horizon, Δt_2 does not affect the conditions in Propositions 1-2 or Theorem 1. Therefore, reducing Δt_1 achieves the same benefits as reducing the time step for the entire optimization but since most of the optimization occurs after the horizon, the number of variables in the optimization does not increase significantly.

5 Numerical Simulations

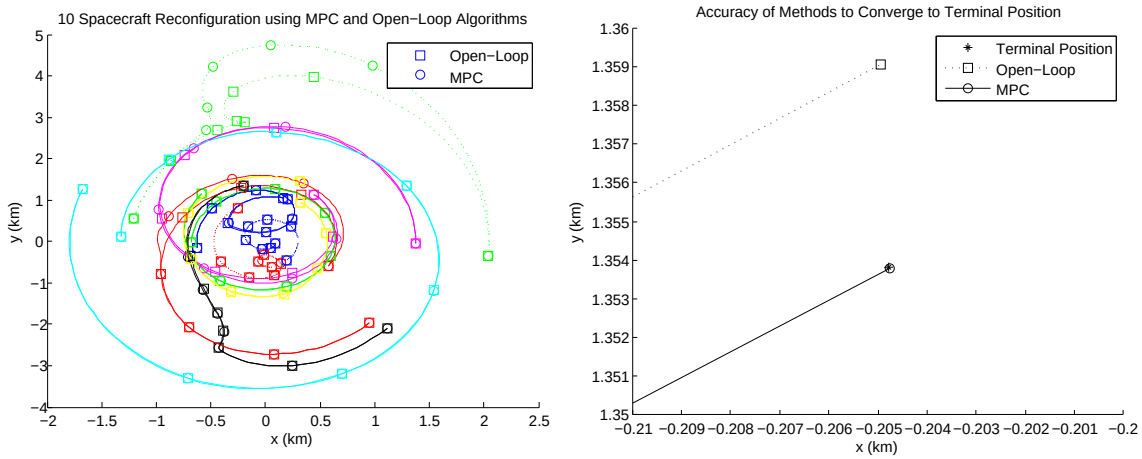
In this section, simulations of the swarm reconfiguration are presented using the MPC-SCP algorithm. The MPC-SCP algorithm is compared to an open-loop optimization, for a 10 spacecraft formation, and the fuel efficiency and accuracy of the trajectories are compared.

All of the simulations are run with a reference orbit having the following initial orbital elements: 6878 km semimajor axis, 0 eccentricity, 45 deg inclination, 60 deg right ascension , 0 deg argument of perigee, and 0 deg true anomaly. Additionally, the length of the transfer, t_f, is 5677 s, or one orbit and the SCP algorithm is considered to be converged if the trajectories for two consecutive iterations have no point exceeding $\epsilon = 10^{-3}$. In all the simulations, the initial and terminal conditions are determined by randomly generating the positions and then applying the J_2-invariant conditions from out prior work [2] to determine the desired velocities. All of the convex optimizations were performed using CVX [10].

The simulation results for the 10 spacecraft formation reconfiguration are shown in Figure 5 and Table 1. Both the MPC-SCP and the open-loop algorithms were run using a collision radius (R) of 150 m, a time step (Δt) of 60 s and an optimization horizon (T_H) of 3 time steps (MPC-SCP only).

Figure 5a shows the trajectories for 10 spacecraft using both the MPC-SCP and open-loop algorithms. In some cases, such as the green dashed trajectory, the trajectories from the two methods are quite different. This can be caused by errors that arise from the convexification process. In the open-loop case, these errors are not accounted for and the resulting trajectory may not reach the terminal state. On the other hand, the MPC-SCP algorithm updates the trajectory using real-time positions so the terminal error is much smaller. The difference in terminal error between the two methods is shown in Figure 5b. The MPC-SCP algorithm (solid line) reaches a terminal position (circle) that is less than 50 mm from the desired terminal state (star). This is over two orders of magnitude better than the open-loop trajectory (dotted line), which reaches a terminal position (square) over 5 m from the desired state. The average terminal errors and fuel usage are shown in Table 1.

Table 1 shows the average terminal position error and fuel usage for both the MPC-SCP and open-loop trajectories. The terminal error decreases dramatically when the MPC-SCP algorithm is used instead of the open-loop optimization. However, the fuel usage required using the MPC-SCP algorithm is about 7% more than the fuel usage of the open-loop method. The increase in fuel consumption is largely due to the



(a) View of a 10 spacecraft reconfiguration

(b) Enlargement of the terminal position for one spacecraft

Figure 5: x-y projection of the reconfiguration of 10 spacecraft using MPC-SCP and open-loop algorithms [13]

Table 1: Simulation results for the reconfiguration of a 10 spacecraft formation using MPC-SCP and Open-Loop SCP algorithms

	Average Algorithm Performance	
Algorithm	Fuel Cost (m/s)	Terminal Error (m)
MPC-SCP Algorithm	2.242	0.032
Open-Loop SCP Optimization	2.100	2.293

decentralized communication. In the MPC-SCP algorithm, spacecraft only consider collisions up to the end of the horizon. In Figure 5, spacecraft 2 (solid red line) performs a large maneuver when the MPC-SCP algorithm is used. This maneuver occurs because a collision is detected and must be avoided in less than three time steps. This is an aggressive maneuver, which does not occur in the open-loop case because all collisions are known at the initial time because all-to-all communication is assumed. Overall, the MPC-SCP algorithm greatly improves the robustness of the trajectories and decentralizes the communication of the swarm with the only a small increase in fuel consumption.

In the next simulation, the conditions in Propositions 1-2 and Theorem 1 are enforced to ensure that the algorithm is feasible and avoids collisions between time steps. Additionally, the idea of different time steps before and after the horizon, which is discussed in Remark 2, is implemented. This simulation has the same parameters as the previous one with the following exceptions: $V_{\max} = 0.005$ km/s, $U_{\max} = 0.001$ km/s^2, $R_{\text{comm}} = 2$ km, $T_H = 12$, $\Delta t_1 = 15$ s, and $\Delta t_2 = 60$ s.

The results of this simulation are compared to the results of a simulation with the same parameters except using a single time step of $\Delta t = 60$ s and maintaining a 180 s horizon by setting $T_H = 3$. The results of these simulations are shown in Table 2.

Table 2 shows the simulation results using the MPC-SCP algorithm satisfying Propositions 1-2 and Theorem 1. When using a constant time step of 60 s, the algorithm performs similarly to how it did in the previous simulation, shown in Table 1. However, when a time step of 15 s is used before the end of the horizon, the algorithm shows

Table 2: Simulation results for the reconfiguration of a 10 spacecraft formation using the MPC-SCP algorithm satisfying Proposition 1-2 and Theorem 1

	Average Algorithm Performance	
Time Step Size	Fuel Cost (m/s)	Terminal Error (mm)
$\Delta t = 60$ s	2.242	12.974
$\Delta t_1 = 15$ s, $\Delta t_2 = 60$ s	2.178	0.709

improvements in both fuel efficiency and terminal error compared to the constant time step case. Additionally, the run time time using smaller time steps is still short enough that Proposition 2 is satisfied.

6 Conclusion

In this paper, a SCP algorithm was developed to solve for the optimal trajectories for swarm reconfiguration. To use SCP, the problem was convexified. To reduce the computation time required, the collision avoidance constraints were decentralized by having each spacecraft treat the other spacecraft's trajectories as fixed. This allowed each spacecraft to run its own SCP algorithm to solve for its optimal trajectory as long as the trajectories of the other spacecraft were known.

This SCP algorithm was then used to compute the optimizations in a MPC framework. Using MPC-SCP decreased the length of the optimizations that needed be solved, which allowed smaller time steps in the optimizations. By using smaller time steps and shorter horizons, the MPC-SCP algorithm bounded the distance each spacecraft could travel during one optimization. This allowed us to relax the communication requirements on each spacecraft by considering communication between two spacecraft only if they were within a certain distance.

To ensure that the trajectories resulting from the MPC-SCP algorithm converged, the terminal cost function was converted to a convex optimization problem with a terminal constraint, which ensured that if the optimization was feasible, it would converge. Also, an upper bound on the magnitude of the velocity was introduced so that two propositions could be developed to ensure that each of the receding horizon optimizations had a solution. Additionally, a theorem was developed to guarantee that the spacecraft do not collide in between discrete time steps.

The MPC-SCP algorithm was used to compute the optimal trajectories for a randomly distributed swarm. The MPC-SCP algorithm drove the spacecraft to within several mm of the desired terminal state despite the convexification errors. Additionally, the time required to run each optimization of the MPC-SCP algorithm was much less than the length of the MPC horizon.

Acknowledgments

This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. ©2013 California Institute of Technology. This work was supported by a NASA Office of the Chief Technologists Space Technology Research Fellowship. Government sponsorship acknowledged.

References

[1] S.-J. Chung and F. Y. Hadaegh, "Swarms of Femtosats for Synthetic Aperture Applications," in *Proceedings of the Fourth International Conference on Spacecraft Formation Flying Missions & Technologies*, (St-Hubert, Quebec), May 2011.

[2] D. Morgan, S.-J. Chung, L. Blackmore, B. Acikmese, D. Bayard, and F. Y. Hadaegh, "Swarm-Keeping Strategies for Spacecraft Under J_2 and Atmospheric Drag Perturbations," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 5, pp. 1492–1506, 2012.

[3] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, "A Survey of Spacecraft Formation Flying Guidance and Control (Part I): Guidance," in *Proceedings of the American Control Conference*, pp. 1733–1739, Jun. 2003.

[4] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, "A Survey of Spacecraft Formation Flying Guidance and Control (Part II): Control," in *Proceedings of the American Control Conference*, pp. 2976–2984, Jun. 2004.

[5] R. M. Murray, "Recent Research in Cooperative Control of Multivehicle Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 51, 2007.

[6] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 2976–2984, 2003.

[7] M. G. Earl and R. D'Andrea, "Iterative MILP Methods for Vehicle-Control Problems," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1158–1167, 2005.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[9] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming," *Math. Program. A*, vol. 89, no. 1, pp. 149–185, 2000.

[10] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming (version 1.22)," May 2012. http://cvxr.com/cvx/.

[11] G. Xu and D. Wang, "Nonlinear Dynamic Equations of Satellite Relative Motion Around an Oblate Earth," *Journal of Guidance, Control, and Dynamics*, vol. 31, pp. 1521–1524, Sep.-Oct. 2008.

[12] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Spacecraft Swarm Guidance Using a Sequence of Decentralized Convex Optimizations," in *AIAA/AAS Astrodynamics Specialist Conference*, (Minneapolis, MN), August 2012. AIAA 2012-4583.

[13] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Decentralized Model Predictive Control of Swarms of Spacecraft using Sequential Convex Programming," in *AAS/AIAA Space Flight Mechanics Conference*, (Kauai, HI), February 2013. AAS 13-439.