Spacecraft Swarm Guidance Using a Sequence of Decentralized Convex Optimizations

Daniel Morgan[*] and Soon-Jo Chung[†]

University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

Fred Y. Hadaegh[‡]

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, USA

This paper presents partially decentralized path planning algorithms for swarms of spacecraft composed of hundreds to thousands of agents with each spacecraft having limited computational capabilities. In our prior work, J_2-invariant orbits have been found to provide collision free motion for hundreds of orbits. This paper develops algorithms for the swarm reconfiguration which involves transferring from one J_2-invariant orbit to another avoiding collisions and minimizing fuel. To perform collision avoidance, it is assumed that the spacecraft can communicate their trajectories with each other. The algorithm uses sequential convex programming to solve a series of approximate path planning problems until the solution converges. Two decentralized methods are developed: a serial method where the spacecraft take turn updating their trajectories and a parallel method where all of the spacecraft update their trajectories simultaneously.

Nomenclature

$\mathcal{I}$	set of spacecraft that are to be avoided
J_2	second harmonic coefficient of earth
$\mathcal{K}$	set of spacecraft that have not converged
L	size of trust region for convex optimization
M	final iteration of sequential convex programming
N	number of spacecraft
R	minimum distance between spacecraft to avoid a collision
R_e	radius of the earth
R_{safe}	secondary collision radius, $1.5R$
T	number of time steps
U	maximum allowable magnitude of the control vector
$\mathcal{X}_L$	trust region for convex optimization
$(\hat{X}, \hat{Y}, \hat{Z})$	Earth centered inertial coordinate system
a	semimajor axis
e	eccentricity
h	angular momentum
i	orbit inclination
k	time step k
k_{J2}	$\frac{3}{2}J_2\mu R_e^2, 2.633\times 10^{10}$ [km^5/s^2]
$\boldsymbol{\ell} = (x, y, z)^T$	relative position vector in the local vertical, local horizontal coordinate system
$\dot{\boldsymbol{\ell}} = (\dot{x}, \dot{y}, \dot{z})^T$	relative velocity vector in the local vertical, local horizontal coordinate system
r	geocentric distance

[*]Graduate Research Assistant, Department of Aerospace Engineering, morgan29@illinois.edu, AIAA Student Member
[†]Assistant Professor, Department of Aerospace Engineering, sjchung@illinois.edu, AIAA Senior Member
[‡]Senior Research Scientist and Technical Fellow, fred.y.hadaegh@jpl.nasa.gov, AIAA Fellow

r_{jZ}	distance from satellite to equator
t	time
t_f	final time
Δt	length of time step
$\mathbf{u}$	control vector in local vertical, local horizontal frame
v_x	radial velocity
$\mathbf{x} = (\boldsymbol{\ell}^T, \dot{\boldsymbol{\ell}}^T)^T$	state vector in local vertical, local horizontal frame
$\bar{\mathbf{x}}$	nominal state vector
Ω	right ascension of the ascending node
α_x	angular acceleration of coordinate system about x-axis
α_y	angular acceleration of coordinate system about y-axis
α_z	angular acceleration of coordinate system about z-axis
β	rate at which the size of the trust region decreases
ϵ	tolerance of sequential convex programming convergence
μ	gravitational constant
ν	true anomaly
θ	argument of latitude
ω	argument of perigee
$\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$	vector of rotation rates of the local vertical, local horizontal frame

Subscripts

0	initial condition ($t = 0$)
d	discretized
f	final condition ($t = t_f$)
i	spacecraft i
j	spacecraft j

Superscripts

m	iteration m

I. Introduction

Spacecraft formation flying has been a major area of research over the past decade. Recently, the idea of formation flying has been extended to create swarms of spacecraft,[1] which contain a large number (hundreds to thousands) of femtosatellites (100-gram class spacecraft). Due to their small size, the femtosats have limited sensing, actuation, and computation capabilities, which require the guidance and control algorithms of the swarm to be both fuel and computationally efficient.

J_2-invariant orbits[2] have been shown to provide collision-free motion for hundreds of orbits. The conditions for these orbits are very good at maintaining a desired swarm shape. This paper addresses the problem of changing the swarm shape so that the spacecraft transfer from one J_2-invariant passive relative orbit (PRO) to another. The goal of this paper is to create fuel and computationally efficient guidance algorithms for the reconfiguration of swarms of spacecraft in lower Earth orbit (LEO). In addition to being fuel and computationally efficient, the algorithms must provide collision-free motion in the highly nonlinear, coupled, time-varying dynamics seen in relative spacecraft motion in the presence of J_2, which is the dominant perturbation in LEO. Between previous work in spacecraft formation flying[3,4] and multivehicle control research,[5–7] many multivehicle guidance methods have been developed. However, the previous work in formation flying usually deals with a small number of spacecraft, a dozen at the most. Additionally, the spacecraft are much larger than femtosats with greater capabilities. The swarm guidance algorithms are different from previous research because they need to simultaneously address the large number of agents, the limited capabilities of each individual agent, and the complex dynamic environment.

A purely centralized algorithm can find fuel-efficient trajectories for reconfiguration but scales very poorly with the number of spacecraft.[8] On the other hand, a decentralized algorithm can generate trajectories with computational efficiency but will need a reactive collision avoidance algorithm,[9] where the spacecraft do not preplan to avoid collisions but rather perform maneuvers once a potential collision is detected, which will reduce the fuel efficiency of the reconfiguration. Depending on the number of spacecraft and the reconfigured state of the swarm, a decentralized approach can be implemented without much loss in fuel efficiency.[9] However, with hundreds to thousands of spacecraft, there is a larger potential for collisions, which will reduce the fuel efficiency of a decentralized algorithm.

Many methods have been developed for solving nonlinear optimal control problems. Due to the complicated nonlinear dynamics of swarms of spacecraft, indirect methods become very difficult to use because they require the derivation of the first-order necessary conditions for optimality.[10,11] Therefore, many optimization problems are solved using direct methods, which parameterize the control space, and sometimes the state space, reducing the problem to a nonlinear optimization. Pseudospectral methods[12] have been used for trajectory optimization but this method solves a centralized problem which will scale poorly with the number of spacecraft due to the coupling of spacecraft in the collision avoidance requirements. Mixed integer linear programming (MILP) can be used to enforce collision avoidance constraints and has been implemented in real-time[13] as well as used for preplanning trajectories.[14,15] However, these algorithms will also scale poorly as the number of spacecraft increases due to the increase in integer variables caused by the increase in the number of collision constraints.

Recently, convex optimization[16] has been used in multi-vehicle trajectory design and shown that it can be efficiently solved to achieve a global optimum. Mattingley et al.[17] used convex optimization to implement a receding horizon controller for a convex problem. Acikmese et al.[18] used convex optimization to find trajectories for a formation reconfiguration with collision avoidance. However, convexifying the collision constraints results in an overly conservative approximation of the collision avoidance region. In the present paper, sequential convex programming (SCP)[19] is applied to the swarm reconfiguration. SCP uses multiple iterations to ensure that the convex approximations of nonconvex constraints are accurate resulting in more fuel-efficient trajectories. Additionally, the SCP algorithms can be written using freely available software, such as CVX,[20] to solve the convex programming problems at each iteration.

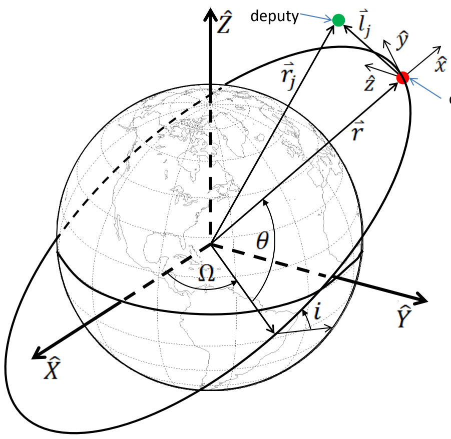
The main contribution of this paper is the method used to decentralize parts of the swarm reconfiguration so that the resulting algorithm can be implemented in real-time and provides collision-free, fuel-efficient trajectories. The swarm reconfiguration is made more computationally efficient in two steps. First, the problem is convexified and discretized using sequential convex programming. To partially decentralize the problem and improve the way it scales with spacecraft number, the second step is to fix the trajectories of the other spacecraft using the most up to date, or nominal, trajectory. This process requires that all spacecraft can communicate their nominal trajectory with all of the others, but that is the only part of the process which is centralized.

The algorithms developed in Sec. IV.B can run fast enough that it can be used to update the trajectories in real-time and be used in model predictive control (MPC). This method can be used to update the trajectories with new state information that may have arisen because of linearization and discretization error, or unmodeled disturbances.

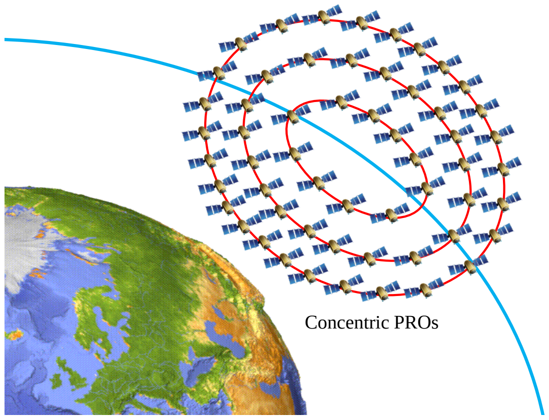
The paper is organized as follows. In Section II, the swarm reconfiguration is discussed and the SCP method is described. In Section III, the problem of converting to convex form is discussed and SCP is applied to the problem. In Section IV, the collision avoidance constraints are discussed and the decentralized optimizations are presented. In Section V, the results of simulations and the effectiveness of each algorithm are discussed.

II. Problem Statement

In this section, the swarm reconfiguration is presented as a continuous, finite horizon optimal control problem. The swarm reconfiguration involves the transfer of hundreds to thousands of spacecraft from one J_2-invariant PRO[2] to another while avoiding collisions between spacecraft and minimizing the total fuel used during the transfer. 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 spacecraft or a virtual reference point called the chief orbit (see Fig. 1a). This coordinate system is inertially fixed and located 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 spacecraft or chief orbit. Figure 1a 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 away 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 radial axis and ω_z about the crosstrack axis. The relative state of the deputy spacecraft in the LVLH frame is expressed by $\mathbf{x}_j = [\, x_j \quad y_j \quad z_j \quad \dot{x}_j \quad \dot{y}_j \quad \dot{z}_j \,]^T$.



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



(b) Spacecraft Swarm

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

The optimization problem for swarm reconfiguration is written using the LVLH coordinates and dynamics. The equations of motion for spacecraft in the LVLH frame ($\boldsymbol{\ell}_j = (x_j, y_j, z_j)^T$) are[21]

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

where the function $\mathbf{g}(\boldsymbol{\ell}_j, \boldsymbol{œ}) \in \mathbb{R}^3$ is defined in the appendix 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 $\boldsymbol{œ} = (r, v_x, h, i, \Omega, \theta)^T$ 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 $\boldsymbol{œ}$ (see the appendix). In this paper, it is assumed that these values are known to each spacecraft by some standard estimation process that might use communicated or measured information about the actual location of the chief spacecraft and propagation of the following equation of motion

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

where the RHS of this equation is defined in the appendix. Note that Eqs. (1) and (2) are hierarchically combined; Eq. (1) does not affect the reference motion given in Eq. (2). Hence, the reference orbital elements are assumed to be known values in the optimization problem. Therefore, the dynamics constraints are given by Eq. (1) with known parameters given by Eq. (2). In addition to the dynamics constraints, the following constraints must be enforced as well.

$$\|\mathbf{u}_j(t)\| \leq U \qquad \forall t \in [0, t_f], \qquad j = 1, \ldots, N \tag{3}$$
$$\|C[\mathbf{x}_j(t) - \mathbf{x}_i(t)]\| \geq R \qquad \forall t \in [0, t_f], \qquad i > j, \qquad j = 1, \ldots, N-1 \tag{4}$$
$$\mathbf{x}_j(0) = \mathbf{x}_{j,0} \qquad j = 1, \ldots, N \tag{5}$$
$$\mathbf{x}_j(t_f) = \mathbf{x}_{j,f} \qquad j = 1, \ldots, N \tag{6}$$

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) represents the limitation on the magnitude of the control vector with U being the maximum allowable control magnitude, Eq. (4) is the collision avoidance constraint with R being the minimum allowable distance between two spacecraft, and Eqs. (5)-(6) are the initial state constraint and the final state constraint, respectively. Although any reachable initial and terminal conditions can be used for Eqs. (5)-(6), the simulations in Sec. V use the J_2-invariant conditions developed in our prior work.[2] Given the relative position, $\boldsymbol{\ell}_j(\tau)$, $\tau \in \{0, t_f\}$, and the chief orbit, $\boldsymbol{œ}(\tau)$, the velocity vector that yields J_2-invariant PRO is given by some function $\dot{\boldsymbol{\ell}}_{\boldsymbol{j}}\boldsymbol{(\tau)} = \mathcal{Y}(\boldsymbol{\ell}_j(\tau), \boldsymbol{œ}(\tau))$ (see Ref. 2 for details).

The objective of the swarm reconfiguration is to minimize fuel. Therefore, we can define the swarm reconfiguration as follows

Problem 1:

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

It is important to note that the objective function and the constraints of Eqs. (3), (5), and (6) already satisfy the requirements for a convex programming problem. Therefore, only the dynamics, Eq. (1), and the collision avoidance constraints, Eq. (4), need to be converted in order to make Problem 1 convex.

III. Sequential Convex Programming

In this section, conversion to SCP is presented. This is done by converting both the dynamics constraints and the collision avoidance constraints, Eqs. (1) and (4) respectively, into an acceptable form for convex programming. For the dynamics, this involves linearizing Eq. (1) and discretizing Problem 1. This results in a finite number of linear equality constraints, which are acceptable in a convex programming problem. The collision avoidance constraints in Eq. (4) are converted to convex inequality constraints so that they are in convex form as well. Once the problem is converted to convex form, a SCP algorithm is applied to solve the modified version of the swarm reconfiguration.

A. Linearization and Discretization of Dynamics

In order to rewrite the dynamics in Eq. (1) as a constraint that can be used in a convex programming problem, these equations must first be linearized. This is necessary because the rules of convex programming state that equality constraints must be affine. Eq. (1) can be rewritten as

$$\dot{\mathbf{x}}_j = \mathbf{f}(\mathbf{x}_j, \text{œ}) + B\mathbf{u}_j \tag{8}$$

where $B = [\mathbf{0}_{3\times3}\ \mathbf{I}_{3\times3}]^T$. Linearizing $\mathbf{f}(\mathbf{x}_j, \text{œ})$ yields

$$\mathbf{f}(\mathbf{x}_j, \text{œ}) \approx \mathbf{f}(\bar{\mathbf{x}}_j, \text{œ}) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_j}\right|_{\bar{\mathbf{x}}_j} (\mathbf{x}_j - \bar{\mathbf{x}}_j) = A(\bar{\mathbf{x}}_j, \text{œ})\mathbf{x}_j + c(\bar{\mathbf{x}}_j, \text{œ}) \tag{9}$$

where $\bar{\mathbf{x}}_j$ is the nominal trajectory about which the equations are linearized. The method for determining these nominal trajectories will be described in Sec. III.C. Additionally, $A(\bar{\mathbf{x}}_j, \text{œ})$ and $c(\bar{\mathbf{x}}_j, \text{œ})$ are

$$A(\bar{\mathbf{x}}_j, \text{œ}) = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_3 \\ -\left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}_j}\right|_{\bar{\mathbf{x}}_j} & -2\mathbf{S}(\boldsymbol{\omega}) \end{bmatrix}, \quad c(\bar{\mathbf{x}}_j, \text{œ}) = \begin{bmatrix} \mathbf{0}_{3\times1} \\ -\mathbf{g}(\bar{\boldsymbol{\ell}}_j, \text{œ}) + \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}_j}\right|_{\bar{\mathbf{x}}_j} \bar{\mathbf{x}}_j \end{bmatrix} \tag{10}$$

Substituting Eq. (9) into Eq. (8) yields the following linear dynamics equation

$$\dot{\mathbf{x}}_j = A(\bar{\mathbf{x}}_j, \text{œ})\mathbf{x}_j + B\mathbf{u}_j + c(\bar{\mathbf{x}}_j, \text{œ}) \tag{11}$$

The next step in the process of converting Eq. (1) into a constraint that can be used in convex programming is to reduce Eq. (11) to a finite number of constraints. In order to do this, the problem is discretized using a zero-order-hold approach such that

$$\mathbf{u}_j(t) = \mathbf{u}_j[k], \qquad t \in [t_k, t_{k+1}), \qquad k = 1,\dots,T-1, \qquad j = 1,\dots,N \tag{12}$$

where $t_f = T\Delta t$, $t_1 = 0$, $t_T = t_f$, and $\Delta t = t_{k+1} - t_k$ for $k = 1,\dots,T-1$. This method of discretization reduces Eq. (11) to

$$\mathbf{x}_j[k+1] = A_d(\bar{\mathbf{x}}_j, \text{œ})\mathbf{x}_j[k] + B_d\mathbf{u}_j[k] + c_d(\bar{\mathbf{x}}_j, \text{œ}), \qquad k = 1,\dots,T-1, \qquad j = 1,\dots,N \tag{13}$$

where $\mathbf{x}_j[k] = \mathbf{x}_j(t_k)$, $\mathbf{u}_j[k] = \mathbf{u}_j(t_k)$, $\text{œ}[k] = \text{œ}(t_k)$, and

$$A_d(\bar{\mathbf{x}}_j, \text{œ}) = e^{A(\bar{\mathbf{x}}_j, \text{œ})\Delta t}, \qquad B_d = \int_0^{\Delta t} e^{A(\bar{\mathbf{x}}_j, \text{œ})\tau} B\, d\tau, \qquad c_d = c(\bar{\mathbf{x}}_j, \text{œ})\Delta t \tag{14}$$

Now that the nonlinear, continuous time equations of motion from Eq. (1) have been rewritten as linear, finite dimensional constraints in Eq. (13), they can be used in a convex programming problem. The constraints from Eqs. (3)-(6) can be written in discretized form as

$$\|\mathbf{u}_j[k]\| \leq U \qquad k = 1, \dots, T-1, \qquad j = 1, \dots, N \tag{15}$$

$$\|C(\mathbf{x}_j[k] - \mathbf{x}_i[k])\| \geq R \qquad k = 1, \dots, T, \qquad i > j, \qquad j = 1, \dots, N-1 \tag{16}$$

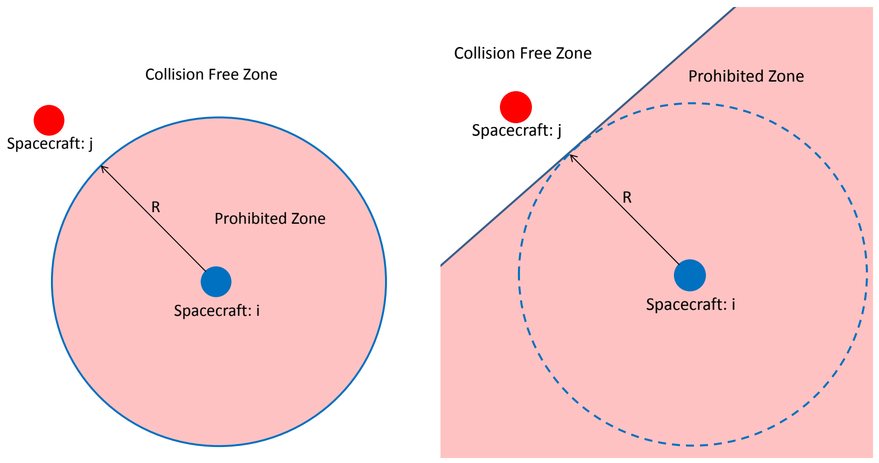
$$\mathbf{x}_j[1] = \mathbf{x}_{j,0} \qquad j = 1, \dots, N \tag{17}$$

$$\mathbf{x}_j[T] = \mathbf{x}_{j,f} \qquad j = 1, \dots, N \tag{18}$$

Note that the only constraint that does not satisfy the rules of convex programming is Eq. (16). This constraint will be modified in the next section so that it can be used in a convex programming problem.

B. Convexification of Collision Avoidance Constraints

The final step in converting the swarm reconfiguration into a convex programming problem is converting the collision avoidance constraints to convex constraints. Since the collision avoidance constraints in their current form are concave, the best convex approximations will be affine constraints. In other words, the sphere which defines the forbidden region is approximated by a plane which is tangent to the sphere and perpendicular to the line segment connecting the nominal position ($\bar{\mathbf{x}}_j$) 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

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 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 new prohibited zone. As can be seen in Fig. 2b, the new prohibited zone includes the old prohibited zone so collision avoidance is still guaranteed using this convexification method.

Figure 3 shows the collision free zone for a spacecraft surrounded by multiple neighbors. When multiple neighboring spacecraft (red) are in the vicinity of spacecraft j (blue), 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. (16) will be

$$(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])^T C^T C(\mathbf{x}_j[k] - \mathbf{x}_i[k]) \geq R\|C(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\| \qquad k = 1, \dots, T, \qquad i > j, \qquad j = 1, \dots, N-1 \tag{19}$$

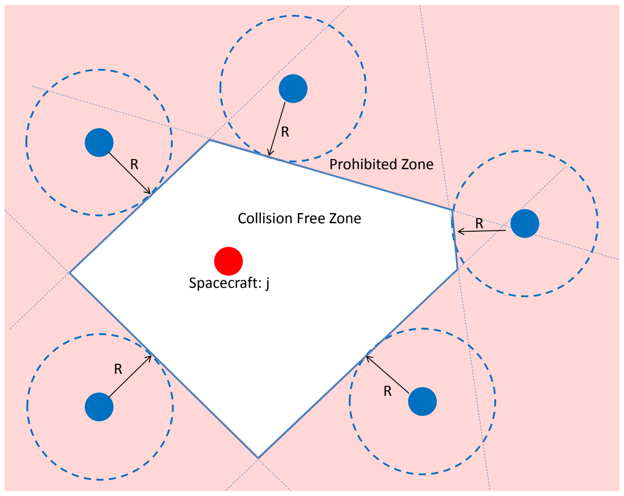


Figure 3. Collision free zone for a spacecraft with 5 neighbors using affine collision avoidance constraints

To show sufficiency, it is assumed that the above condition is satisfied. The following steps are valid for all i, j, k.

$$\begin{aligned}
&(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])^T C^T C(\mathbf{x}_j[k] - \mathbf{x}_i[k]) \geq R\|C(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\| \\
&\|C(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\| \; \|C(\mathbf{x}_j[k] - \mathbf{x}_i[k])\| \cos\phi \geq R\|C(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\| \\
&\|C(\mathbf{x}_j[k] - \mathbf{x}_i[k])\| \cos\phi \geq R \\
&\|C(\mathbf{x}_j[k] - \mathbf{x}_i[k])\| \geq \|C(\mathbf{x}_j[k] - \mathbf{x}_i[k])\| \cos\phi \geq R
\end{aligned}$$

This reestablishes Eq. (16) and proves sufficiency. $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}_j$ are the nominal trajectories from the previous iteration of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, and ϕ is the angle between the two vectors. These nominal values are assumed to be known and are not variables in the optimization. Therefore, the collision avoidance constraints in Eq. (19) are affine and in a form that can be used in a convex programming problem.

C. Sequential Convex Programming Algorithm

Now that all of the constraints in Problem 1 have been written in convex programming form, Problem 1 can be written as the following convex programming problem:

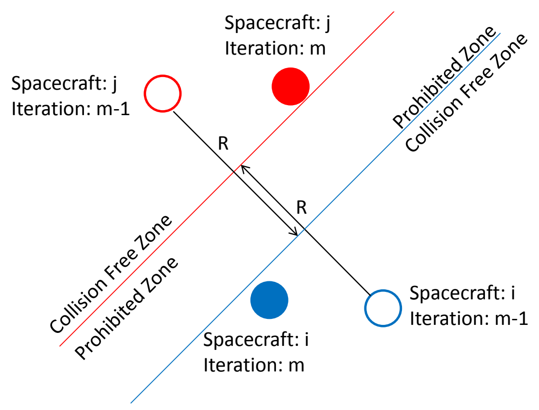
Problem 2 (SCP):

$$\min_{\mathbf{u}_j, j=1,\ldots,N} \sum_{j=1}^{N} \sum_{k=1}^{T-1} \|\mathbf{u}_j[k]\|\Delta t \quad \text{subject to} \quad \{(13), (15), (17), (18), (19)\} \tag{20}$$

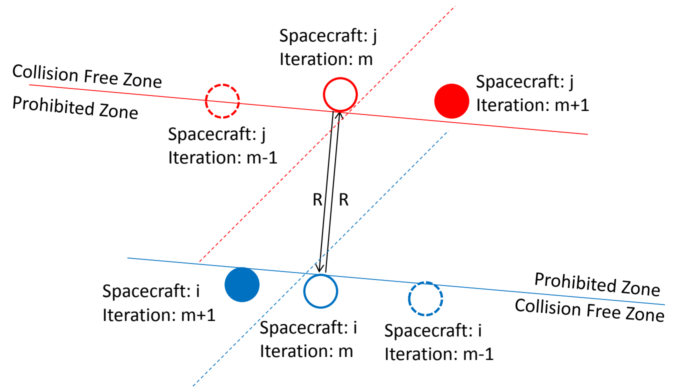
where Problem 1 has been discretized and the constraints of Eqs. (1) and (4) have been approximated by Eqs. (13) and (19), respectively.

The approximations used to get the dynamics and collision avoidance constraints into their convex forms, Eqs. (13) and (19), require a nominal state $\bar{\mathbf{x}}_j[k]$ for each spacecraft at each time step. Additionally, the nominal vectors must be close to the actual state vectors in order for the solution to the convex programming problem to be valid. In order to ensure that the nominal vectors are good estimates of the actual state vectors, SCP is used. In SCP, $\bar{\mathbf{x}}_j[k] = \mathbf{x}_j^{m-1}[k]$ for iteration m, i.e. the calculation of $\mathbf{x}_j^m[k]$. To enforce the collision avoidance constraints, each spacecraft communicates its own 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 resulting solution is not as dependent on the initial guess. Because of the way that the collision avoidance constraint must be convexified, the prohibited zone for each collision is a half space, which is overly conservative. With convex programming, this will prevent the spacecraft from passing through certain areas that are, in fact, 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 prohibited in the initial convex programming problem. This is illustrated in Fig. 4. In the iteration $m+1$, shown in Fig. 4b, the spacecraft move into areas that were originally prohibited in iteration m, shown in Fig. 4a. This idea allows SCP to achieve better trajectories than the convex programming problem.



(a) Collision free zone for iteration m



(b) Collision free zone for iteration m+1

Figure 4. Evolution of the 2-D collision avoidance constraint

SCP is a method for solving nonconvex optimizations using convex programming. In order to use SCP, the nonconvex problem is approximated by a convex problem as has been done in Subsections III.A & III.B. Then, the convex problem is solved using an iterative method. In the first iteration, an initial guess is provided for the nominal vector for the dynamics 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. The process is demonstrated in Algorithm 1. In each iteration, a trust region is defined for the convex problem. The region represents the range of state vectors over which the linearization provides accurate approximations. It is defined as

$$\mathcal{X}_{L^m} = \{\mathbf{x}_j | \; \|\mathbf{x}_j - \bar{\mathbf{x}}_j\| \leq L^m\} \tag{21}$$

where L^m is the size of the trust region during iteration m. Additionally, the trust region in Eq. (21) can be used to ensure that the SCP algorithm converges. In order to ensure converges, the size of the trust is updated according to the following equation.

$$L^{m+1} = \beta L^m \tag{22}$$

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

Algorithm 1 Centralized SCP Algorithm

```
1: $\bar{\mathbf{x}}_j := \mathbf{0}_{6\times 1} \; \forall j$
2: $\mathbf{x}_j^0 :=$ solution to Problem 2 without the collision avoidance constraint in Eq. (19)
3: $\bar{\mathbf{x}}_j := \mathbf{x}_j^0$
4: $\mathbf{x}_j^1 :=$ solution to Problem 2
5: $m := 1$
6: while $\|\mathbf{x}_j^m[k] - \mathbf{x}_j^{m-1}[k]\| > \epsilon \; \forall k$ do
7:     $m := m+1$
8:     $\bar{\mathbf{x}}_j = \mathbf{x}_j^{m-1}$
9:     $\mathbf{x}_j^m :=$ the solution to Problem 2
10: end while
11: $M := m$
12: $\mathbf{x}_j^M$ is the approximate solution to Problem 1
```

$\mathbf{x}_j^m[k]$ denotes the relative state vector of the jth spacecraft, at the kth time step, after the mth iteration of the convex program and $\mathbf{x}_j^m$ is the same as $\mathbf{x}_j^m[k]$ except over all time steps. Additionally, M is the final iteration performed in the SCP problem and ϵ is the tolerance of the SCP algorithm.

The SCP method described in Algorithm 1 is effective for small spacecraft formations but does not scale well because the number of collision avoidance constraints increases quadratically with the number of spacecraft. Additionally, while this method reduces the problem to convex form in Problem 2, which is much simpler than the original nonconvex form in Problem 1, it is still a centralized problem where all of the spacecraft's trajectories are solved for at the same time. Due to the limited size of the spacecraft in a swarm, it is unlikely that any of them will have the computational ability to solve the entire swarm reconfiguration. Therefore, decentralizing the swarm reconfiguration will make it much more feasible.

IV. Decentralized Methods for Sequential Convex Programming

As mentioned in the preceding section, even in convex form, the centralized swarm reconfiguration algorithm will still scale poorly because of the number of collision avoidance constraints. Therefore, the problem must be partially decentralized so that it can be run using the limited computational capabilities of spacecraft in the swarm. The collision avoidance constraints are the only constraints involving more than one spacecraft. Therefore, the goal of this section is to rewrite the collision constraints 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 Algorithm 1 is noticing that many of the spacecraft do not come close to each other at any time during the reconfiguration. For this reason, it is not necessary to include the collision avoidance constraints for every pair of spacecraft in the SCP algorithm. By defining a second collision distance as $R_{\text{safe}} = 1.5\ R$, where R is the distance that must exist between two spacecraft in order to avoid a collision, and only checking for collisions for spacecraft pairs that violated this distance in a previous iteration of the SCP, the number of constraints in each iteration of Problem 2 can be greatly reduced. Another property of Algorithm 1 that can be used to reduce the computational complexity is the fact that as the number of iterations increases, the difference between $\mathbf{x}_j^m[k]$ and $\mathbf{x}_j^{m-1}[k]$ decreases. In other words, 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, which must be avoided. Using this assumption, Eq. (19) can be rewritten as

$$(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])^T C^T C(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k]) \geq R\|C(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\| \qquad k = 1, \ldots, T, \qquad i \in \mathcal{I}_j, \qquad j = 1, \ldots, N-1 \tag{23}$$

where

$$\mathcal{I}_j = \{i |\ \exists\ k \in 1, \ldots, T \text{ such that } \|C(\mathbf{x}_i[k] - \mathbf{x}_j[k])\| \leq R_{\text{safe}}\} \tag{24}$$

This allows each spacecraft to solve its own trajectory optimization while still maintaining a collision-free swarm. The decentralized problem can be written as follows:

Problem 3:

$$\min_{\mathbf{u}_j, j=1,\ldots,N} \sum_{j=1}^{N} \sum_{k=1}^{T-1} \|\mathbf{u}_j[k]\|\Delta t \quad \text{subject to} \quad \{(13), (15), (17), (18), (23)\} \tag{25}$$

There are two different types of decentralized algorithms that are described in the following subsections: a serial method and a parallel method. These algorithms are similar and only differ on when the spacecraft communicate their updated position to the rest of the swarm. The serial method provides slightly better fuel efficiency and less individual computation time but the spacecraft cannot run their programs simultaneously so the total elapsed time is longer. On the other hand, the parallel method allows all of the spacecraft to run their computations simultaneously, which decreases the total time but each spacecraft's individual run time will be longer and the fuel efficiency will be slightly worse.

A. Serial Method

In the serial method, the spacecraft wait to receive the most updated positions from the other spacecraft before beginning their own computations. This leads to faster convergence of the SCP problem and better fuel efficiency. A detailed description of the decentralized serial method is given in Algorithm 2.

In the serial algorithm shown in Algorithm 2, each spacecraft solves an iteration of the SCP problem and then updates the rest of the spacecraft as to its new trajectory. This provides the spacecraft that follow it with more up to date information for collision avoidance but it also requires that these spacecraft wait until

Algorithm 2 Serial Algorithm for Decentralized Problem

```
1: $\bar{\mathbf{x}}_j := \mathbf{0}_{6\times 1}\ \forall j$
2: $\mathcal{I}_j := \emptyset\ \forall j$
3: $\mathbf{x}_j^0 :=$ the solution to Problem 3
4: $\bar{\mathbf{x}}_j := \mathbf{x}_j^0$
5: Update $\mathcal{I}_j$ using Eq. (24)
6: $\mathcal{K} := \{1,\ldots,N\}$
7: $m := 1$
8: while $\mathcal{K} \neq \emptyset$ do
9:     for all $j \in \mathcal{K}$ do
10:        $\mathbf{x}_j^m :=$the solution to Problem 3
11:        $\mathcal{I}_j$ is updated using Eq. (24)
12:        $\bar{\mathbf{x}}_j := \mathbf{x}_j^m$
13:    end for
14:    for all $j$ do
15:        if $\|\mathbf{x}_j^m[k] - \mathbf{x}_j^{m-1}[k]\| < \epsilon\ \forall k$ and $\|C(\mathbf{x}_j^m[k] - \mathbf{x}_i^m[k])\| > R\ \forall k, \forall i \neq j$ then
16:            Remove $j$ from $\mathcal{K}$
17:        end if
18:    end for
19:    $m = m + 1$
20: end while
21: $M := m - 1$
22: $\mathbf{x}_j^M$ is the approximate solution to Problem 1
```

they receive the updated trajectory to perform their own calculations as seen in line 11 of Algorithm 2. For this reason, the total time required to run the algorithm will be larger than in the parallel method described in the following subsection. On the other hand, this algorithm has the advantage of converging much faster and to a slightly more optimal solution than the parallel algorithm. Ultimately, it is necessary to decrease the total elapsed time of the algorithm, which is why a parallel algorithm is developed in the next subsection.

B. Parallel Method

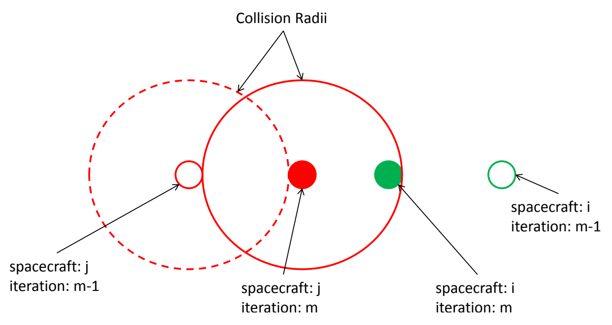
The parallel method described in Algorithm 3 is very similar to Algorithm 2 but with some small changes in order to decrease the total run time of the method. The main change is that the nominal values are not updated until after every spacecraft has completed its computation as seen in line 13 of Algorithm **??**. This allows all of the spacecraft to run the SCP algorithm simultaneously, which greatly reduces the total elapsed time. Unfortunately, this can cause the SCP algorithm to have trouble converging when trying to avoid collisions. This occurs because two spacecraft that are trying to avoid each other are now simultaneously updating their trajectories. Because neither spacecraft knows where the other will be, they may choose trajectories that are collision free based on the other spacecraft's previous trajectory but are not collision free based on the new trajectories. This situation is shown in Fig. 5.

Figure 5a shows spacecraft i (green) moving to a position (solid green circle) that is safe based on the previous location of spacecraft j (red open circle). However, spacecraft j has updated its position (solid red circle) and the spacecraft are within each other's collision radii. Figure 5b shows the following iteration where the spacecraft are overly conservative because both spacecraft think the other will be closer to them based on the previous trajectory. It is possible for the spacecraft to oscillate back and forth in this manner, which prevents the SCP algorithm from converging. In order to avoid this situation, Eq. (24) is redefined as

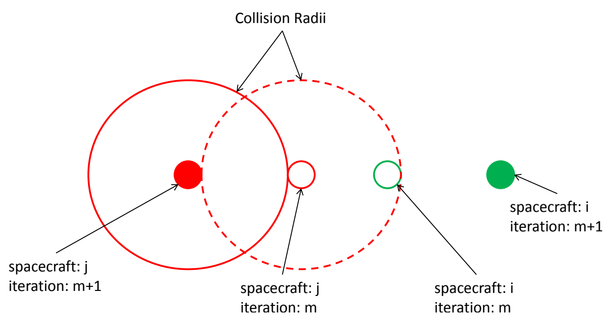
$$\mathcal{I}_{j,par} = \{i |\ \exists\ k \in 1,\ldots,T \text{ such that } \|C(\mathbf{x}_i[k] - \mathbf{x}_j[k])\| \leq R_{\text{safe}} \text{ and } i < j\} \tag{26}$$

By adding the constraint $i < j$, only one of the spacecraft will try to avoid the other one. Using these modifications, the parallel method is shown in Algorithm 3.

By forcing one spacecraft to avoid the other rather than allowing cooperative avoidance, the optimality can potentially be lower than in the serial method. However, the total run time of the algorithm is greatly reduced. Additionally, the numbering of the spacecraft is arbitrary so they can be numbered in a way that



(a) Update from iteration m-1 to iteration m: collision



(b) Update from iteration m to iteration m+1: overly conservative

Figure 5. An example of two spacecraft that have difficulty converging. This problem is solved by adding a second constraint to Eq. (24) **as seen in Eq.** (26)

Algorithm 3 Parallel Algorithm for Decentralized Problem

```
1: $\bar{\mathbf{x}}_j := \mathbf{0}_{6\times 1}\ \forall j$
2: $\mathcal{I}_{j,par} := \emptyset\ \forall j$
3: $\mathbf{x}_j^0 :=$ the solution to Problem 3
4: $\bar{\mathbf{x}}_j := \mathbf{x}_j^0$
5: Update $\mathcal{I}_{j,par}$ using Eq. (26)
6: $\mathcal{K} := \{1, \ldots, N\}$
7: $m := 1$
8: while $\mathcal{K} \neq \emptyset$ do
9:   for all $j \in \mathcal{K}$ do
10:     $\mathbf{x}_j^m :=$the solution to Problem 3
11:   end for
12:   for all $j$ do
13:     Update $\mathcal{I}_{j,par}$ using Eq. (26)
14:     $\bar{\mathbf{x}}_j := \mathbf{x}_j^m$
15:     if $\|\mathbf{x}_j^m[k] - \mathbf{x}_j^{m-1}[k]\| < \epsilon\ \forall k$ and $\|C(\mathbf{x}_j^m[k] - \mathbf{x}_i^m[k])\| > R\ \forall k, \forall i \neq j$ then
16:       Remove $j$ from $\mathcal{K}$
17:     end if
18:   end for
19:   $m := m + 1$
20: end while
21: $M := m - 1$
22: $\mathbf{x}_j^M$ is the approximate solution to Problem 1
```

minimizes the disadvantages of using the parallel method. For example, ordering the spacecraft based on their efficiency or amount of fuel remaining will ensure that the spacecraft avoiding the collision is better suited to do so. In this case, the decrease in optimality may not be as significant.

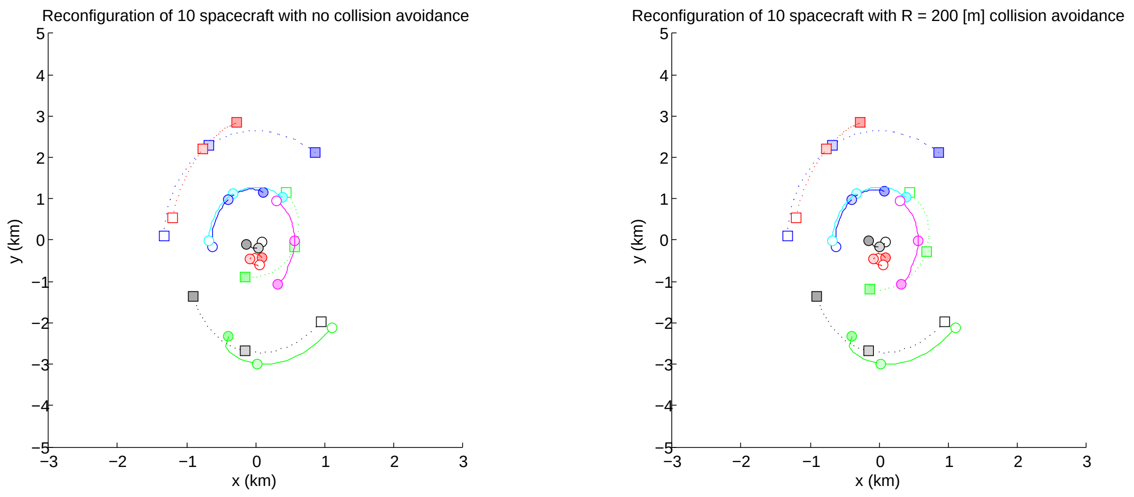
The large decrease in run time between Algorithms 2 and 3 allows Algorithm 3 to be used much more versatilely. For example, because the run time is now on the order of a time step or two, the algorithm can be run using MPC by updating the future control commands based on new state information that includes unmodeled disturbances and other errors. This can provide some robustness improvements compared to running the algorithm only once at the beginning.

V. Simulations

In this section, simulations of the swarm reconfiguration are presented using the algorithms developed in Sec. IV. A formation reconfiguration with 10 spacecraft and a swarm reconfiguration with 100 spacecraft are solved using Algorithm 2-3. The reconfigurations are not solved using the centralized method represented by Algorithm 1 because the huge number of variables and constraints makes it intractable. The fuel and computational efficiencies of the methods are presented and compared.

All of the simulations are run with a reference orbit having the following initial orbital elements: $[a,\ e,\ i,\ \Omega,\ \omega,\ \nu] = [6878\text{ km},\ 0,\ 45\text{ deg},\ 60\text{ deg},\ 0\text{ deg},\ 0\text{ deg}]$. Additionally, the length of the transfer, t_f, is 5677 s, or one orbit. The problem is discretized into 60 s intervals and $\epsilon = 10^{-3}$. The number of spacecraft and the collision radius are varied throughout the simulations. In all the simulations, the initial and terminal conditions, $\mathbf{x}_{j,0}$ and $\mathbf{x}_{j,f}$, respectively, 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.[20]

The simulation results for the 10 spacecraft formation reconfiguration is shown in Table 1. Both the serial and parallel algorithms were run for collision radii R of 0 m, 150 m, and 200 m. Additionally, the trajectories resulting from the parallel method with $R = 0$ m and $R = 200$ m are shown in Figs. 6-8.



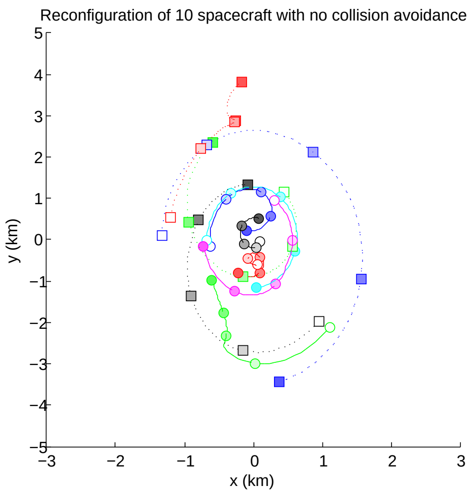
(a) No collision avoidance

(b) Collision avoidance with R = 200 m

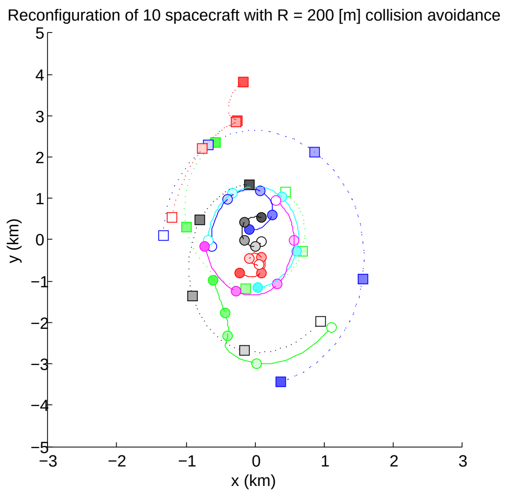
Figure 6. x-y projection of the beginning 1/3 of the reconfiguration of 10 spacecraft. Markers fade from empty to solid as time moves forward.

The collision avoidance maneuver can be seen in Fig. 6-8. The most obvious collision avoidance can be seen in Fig. 6 as the green square avoids the magenta circle around the time represented by the second marker. There are other collision avoidance maneuvers but this example is the most dramatic in the x-y projection.

Table 1 shows that for small formations with only a couple of potential collisions, the parallel method in Algorithm 3 reduces the run time by a factor between 3 and 5 compared to Algorithm 2 without increasing the fuel cost. Therefore, the advantages of this method greatly outweigh the disadvantages for this formation.

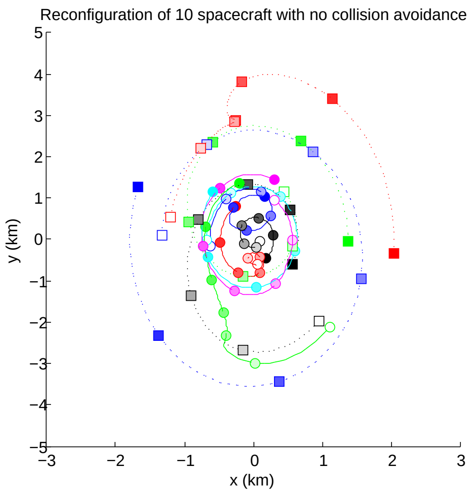


(a) No collision avoidance

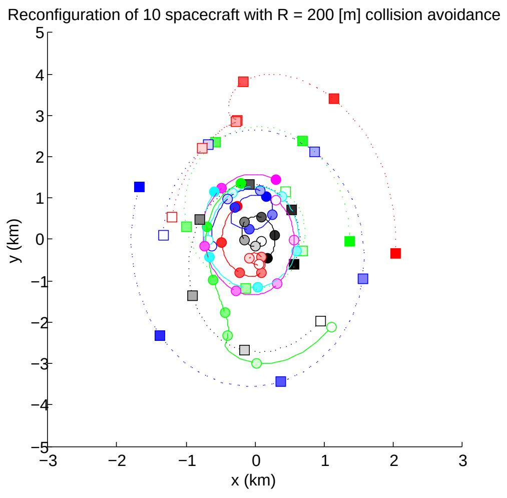


(b) Collision avoidance with R = 200 m

Figure 7. x-y projection of the beginning 2/3 of the reconfiguration of 10 spacecraft. Markers fade from empty to solid as time moves forward.



(a) No collision avoidance



(b) Collision avoidance with R = 200 m

Figure 8. x-y projection of the entire reconfiguration of 10 spacecraft. Markers fade from empty to solid as time moves forward.

Table 1. Simulation results for the reconfiguration of a 10 spacecraft formation using decentralized SCP algorithms

			Algorithm Performance		
Algorithm	R [m]	Col. Avoided	Fuel Cost [m/s]	Run Time [s]	Serial Run Time [s]
Serial (Algorithm 2)	0	0	21.00	69.98	69.98
	150	2	21.01	103.35	103.35
	200	3	21.13	107.12	107.12
Parallel (Algorithm 3)	0	0	21.00	16.03	80.13
	150	2	21.01	20.63	104.21
	200	3	21.13	30.71	104.16

Table 2. Simulation results for the reconfiguration of a swarm of 100 spacecraft using decentralized SCP algorithms

			Algorithm Performance		
Algorithm	R [m]	Col. Avoided	Fuel Cost [m/s]	Run Time [s]	Serial Run Time [s]
Serial (Algorithm 2)	0	0	191.50	707.75	707.75
	100	31	191.57	927.51	927.51
	150	68	192.38	1768.7	1768.7
Parallel (Algorithm 3)	0	0	191.50	19.94	765.25
	100	31	191.58	45.48	928.02
	150	68	192.39	75.35	1486.6

Table 2 shows the simulation results for reconfiguring a swarm of 100 spacecraft. Both the serial and parallel algorithms were run for collision radii R of 0 m, 100 m, and 150 m. As with the 10 spacecraft formation, the swarm of 100 spacecraft reconfiguration can be completed much faster using the parallel algorithm (Algorithm 3). In this case, it is more than an order of magnitude faster than the serial method (Algorithm 2). This indicates that the Algorithm 3 scales well with the number of spacecraft. However, this algorithm does use slightly more fuel when there are many collisions to be avoided. In this case, the specific requirements of the mission may dictate which algorithm is better, but in cases where the potential collision probability is low, i.e. the swarm is large relative to the number of spacecraft and the size of the collision radii, then the parallel method performs better.

Remark 1: There is room for improvement in the convex programming solver used in the SCP algorithms. Currently, it takes about 2.5s to run the convex solver but only 0.8s to run the semidefinite programming solver that is called. This means over half of the time is spent converting the convex problem into a semidefinite programming problem. This work can be done off line and a significant amount of time can be saved when running the SCP algorithms.

VI. Conclusion

In this paper, the swarm reconfiguration was solved by using SCP and by decentralizing the collision avoidance constraints. To use SCP, the dynamics and collision avoidance constraints were linearized and the problem was discretized. The resulting problem was in convex form and Algorithm 1 was developed using SCP to solve the problem. The drawback to this method was that the problem was still centralized and a large number of spacecraft caused the algorithm to have difficulty solving the problem due the large number of constraints.

To reduce the size of the optimization that needed to be solved, the collision avoidance constraints were decentralized by having each spacecraft treat the other spacecraft trajectories as fixed. This allowed each spacecraft to run its own SCP algorithm to solve for its optimal trajectory as long as it knew the trajectories of the other spacecraft. Two methods resulted from the decentralized SCP algorithm. The first method, Algorithm 2, was run with the computations in series. In this case, one spacecraft ran its algorithm and then updated the rest of the spacecraft with its new trajectory. Then, the next spacecraft ran its algorithm, and so on. This method provided good results from a fuel-efficiency stand point because each spacecraft was able to converge to an optimal trajectory quickly. However, because only one spacecraft was running computations at a time, the total algorithm was very long.

The other method, Algorithm 3, had the spacecraft run their computations in parallel. This greatly decreased the total time of the algorithm but led to some small problems for the individual computations. Mainly, the spacecraft could not receive updates of the other trajectories so they were using older information. This caused the spacecraft to have difficulty converging on an optimal solution. For this reason, the spacecraft were ordered and only the higher ranking spacecraft attempted to avoid the collision. This improved the convergence rate of the SCP algorithm but potentially led to less optimal solutions. However, the benefits gained in terms of total elapsed time were much more significant than the slight increase in fuel usage. The benefit of reducing the run time can be magnified by implementing the algorithm in real-time as MPC. This allows the algorithm to have some robustness with respect to unmodeled errors and disturbances.

Several methods for using SCP to solve the swarm reconfiguration were presented in this paper. The main trade off seen in these methods was between fuel efficiency and computational efficiency. In general, the decentralized parallel method was much more computationally efficient and only slightly less fuel efficient so it was recommended unless computational efficiency and the run time of the algorithm are unimportant for the specific mission.

Appendix: Dynamics of Chief and Relative Motion

The translational dynamics of spacecraft in the LVLH frame is described by Eq. (1) with

$$\mathbf{g}(\boldsymbol{\ell}, \text{œ}) = \begin{bmatrix} \eta_j^2 - \omega_z^2 & -\alpha_z & \omega_x\omega_z \\ \alpha_z & \eta_j^2 - \omega_z^2 - \omega_x^2 & -\alpha_x \\ \omega_x\omega_z & \alpha_x & \eta_j^2 - \omega_x^2 \end{bmatrix} \boldsymbol{\ell} + (\zeta_j - \zeta) \begin{pmatrix} \sin i \sin\theta \\ \sin i \cos\theta \\ \cos i \end{pmatrix} + \begin{pmatrix} r(\eta_j^2 - \eta^2) \\ 0 \\ 0 \end{pmatrix} \tag{27}$$

where

$$\begin{aligned}
\zeta &= \frac{2k_{J2}\sin i \sin\theta}{r^4} & \zeta_j &= \frac{2k_{J2} r_{jZ}}{r_j^5} \\
\eta^2 &= \frac{\mu}{r^3} + \frac{k_{J2}}{r^5} - \frac{5k_{J2}\sin^2 i \sin^2\theta}{r^5} & \eta_j^2 &= \frac{\mu}{r_j^3} + \frac{k_{J2}}{r_j^5} - \frac{5k_{J2} r_{jZ}^2}{r_j^7} \\
r_j &= \sqrt{(r+x_j)^2 + y_j^2 + z_j^2} & r_{jZ} &= (r+x_j)\sin i\sin\theta + y_j \sin i \cos\theta + z_j\cos i \\
\omega_x &= -\frac{k_{J2}\sin 2i \sin\theta}{hr^3} & \omega_y &= 0 \qquad \omega_z = \frac{h}{r^2} \\
\alpha_x &= -\frac{k_{J2}\sin 2i\cos\theta}{r^5} + \frac{3v_x k_{J2}\sin 2i \sin\theta}{r^4 h} - \frac{8k_{J2}^2 \sin^3 i \cos i \sin^2\theta\cos\theta}{r^6h^2} \\
\alpha_z &= -\frac{2hv_x}{r^3} - \frac{k_{J2}\sin^2 i \sin 2\theta}{r^5}
\end{aligned} \tag{28}$$

The orbital parameters of the chief orbit (origin of the LVLH frame) are governed by the following equation of motion with J_2 effects

$$\begin{aligned}
\dot{r} &= v_x & \dot{v}_x &= -\frac{\mu}{r^2} + \frac{h^2}{r^3} - \frac{k_{J2}}{r^4}(1 - 3\sin^2 i \sin^2\theta) \\
\dot{h} &= -\frac{k_{J2}\sin^2 i \sin 2\theta}{r^3} & \dot{\Omega} &= -\frac{2k_{J2}\cos i \sin^2\theta}{hr^3} \\
\dot{i} &= -\frac{k_{J2}\sin 2i \sin 2\theta}{2hr^3} & \dot{\theta} &= \frac{h}{r^2} + \frac{2k_{J2}\cos^2 i \sin^2\theta}{hr^3}
\end{aligned} \tag{29}$$

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. ©2012 California Institute of Technology. This work was supported by a NASA Office of the Chief Technologists Space Technology Research Fellowship. Government sponsorship acknowledged. Additional thanks to Saptarshi Bandyopadhyay (University of Illinois), Behcet Acikmese and Dan Scharf (Jet Propulsion Laboratory) for constructive comments.

References

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

[2]Morgan, D., Chung, S.-J., Blackmore, L., Acikmese, B., Bayard, D., and Hadaegh, F. Y., "Swarm-Keeping Strategies for Spacecraft Under J_2 and Atmospheric Drag Perturbations," *Journal of Guidance, Control, and Dynamics (to be published)*, 2012.

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

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

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

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

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

[8]Reif, J. and Sharir, M., "Motion Planning in the Presence of Moving Obstacles," *Journal of the Association for Computing Machinery*, Vol. 41, No. 4, 1994, pp. 764–790.

[9]Scharf, D. P., Acikmese, B., Ploen, S. R., and Hadaegh, F. Y., "Three-Dimensional Reactive Collision Avoidance with Multiple Colliding Spacecraft for Deep-Space and Earth-Orbiting Formations," *Proceedings of the Fourth International Conference on Spacecraft Formation Flying Missions & Technologies*, St-Hubert, QC, May.

[10]Rao, A. V., "A Survey of Numerical Methods for Optimal Control," *Advances in the Astronautical Sciences*, Vol. 135, 2010, pp. 497–528.

[11]Conway, B. A., *Spacecraft Trajectory Optimization*, Cambridge Univ. Press, New York, USA, 2010.

[12]Ross, I. M. and Fahroo, F., "Legendre Pseudospectral Approximations of Optimal Control Problems," *Lecture Notes in Control and Information Systems*, Vol. 295, 2003.

[13]Richards, A., Kuwata, Y., and How, J., "Experimental Demonstration of Real-time MILP Control," *AIAA Guidance, Navigation, and Control Conference*, Austin, Texas, August 2003.

[14]Vitus, M. P., Pradeep, V., Hoffman, G. M., Waslander, S. L., and Tomlin, C. J., "Tunnel-MILP: Path Planning with Sequential Convex Polytopes," *AIAA Guidance, Navigation, and Control Conference*, Honolulu, HI, August 2008.

[15]Vitus, M. P., Waslander, S. L., and Tomlin, C. J., "Locally Optimal Decomposition for Autonomous Obstacle Avoidance with the Tunnel-MILP Algorithm," *IEEE Conference on Decision and Control*, 2008, pp. 540–545.

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

[17]Mattingley, J., Wang, Y., and Boyd, S., "Receding Horizon Control: Automatic Generation of High-Speed Solvers," *IEEE Control Systems Magazine*, Vol. 31, No. 3, 2011, pp. 52–65.

[18]Acikmese, B., Scharf, D. P., Murray, E. A., and Hadaegh, F. Y., "A Convex Guidance Algorithm for Formation Reconfiguration," *AIAA Guidance, Navigation, and Control Conference*, Keystone, Colorado, August 2006.

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

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

[21]Xu, G. and Wang, D., "Nonlinear Dynamic Equations of Satellite Relative Motion Around an Oblate Earth," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, Sep.-Oct. 2008, pp. 1521–1524.