JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 37, No. 6, November–December 2014

Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming

Daniel Morgan* and Soon-Jo Chung†
University of Illinois at Urbana–Champaign, Urbana, Illinois 61801
and
Fred Y. Hadaegh‡
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109

DOI: 10.2514/1.G000218

This paper presents a decentralized, model predictive control algorithm for the optimal guidance and reconfiguration of swarms of spacecraft composed of hundreds to thousands of agents with limited capabilities. In previous work, J_2-invariant orbits have been found to provide collision-free motion for hundreds of orbits in a low Earth orbit. This paper develops real-time optimal control algorithms for the swarm reconfiguration that involve transferring from one J_2-invariant orbit to another while avoiding collisions and minimizing fuel. The proposed model predictive control-sequential convex programming algorithm uses sequential convex programming to solve a series of approximate path planning problems until the solution converges. By updating the optimal trajectories during the reconfiguration, the model predictive control algorithm results in decentralized computations and communication between neighboring spacecraft only. Additionally, model predictive control reduces the horizon of the convex optimizations, which reduces the run time of the algorithm. Multiple time steps, time-varying collision constraints, and communication requirements are developed to guarantee stability, feasibility, and robustness of the model predictive control-sequential convex programming algorithm.

Nomenclature

a	=	magnitude of the acceleration vector
$a_{\max}$	=	maximum possible acceleration of a spacecraft
a^*	=	acceleration magnitude that minimizes the distance between two spacecraft
h	=	magnitude of the specific angular momentum of orbit
$\mathcal{I}$	=	set of spacecraft that are to be avoided
i	=	orbit inclination
J_2	=	second harmonic coefficient of Earth
$\mathcal{K}$	=	set of spacecraft that have not converged
k	=	time step k
k_{J2}	=	$\frac{3}{2}J_2\mu R_e^2$, 2.633×10^{10} km^2/s^2
k_0	=	time step at the start of the model predictive control horizon
L	=	size of trust region for convex optimization
$\ell = (x, y, z)^T$	=	relative position vector in the local-vertical/local-horizontal coordinate system
$\dot{\ell} = (\dot{x}, \dot{y}, \dot{z})^T$	=	relative velocity vector in the local-vertical/local-horizontal coordinate system
M	=	final iteration of sequential convex programming
N	=	number of spacecraft
œ	=	orbital element vector
R_{col}	=	minimum collision-free distance enforced at the discrete points in the optimization
$\bar{R}_{\text{col}}$	=	minimum collision-free distance achieved in the continuous trajectories ($\bar{R}_{\text{col}} < R_{\text{col}}$)
R_{comm}	=	maximum distance a spacecraft can communicate ($R_{\text{comm}} > R_{\text{col}}$)
R_e	=	radius of the Earth
r	=	geocentric distance
T	=	final time step
T_H	=	number of time steps in the model predictive control horizon
t	=	time
t_f	=	final time
t_{run}	=	time required to compute the optimization
$U_{\max}$	=	maximum allowable magnitude of the control (acceleration) vector
$\boldsymbol{u}$	=	control (acceleration) vector in local-vertical/local-horizontal frame
$V_{\max}$	=	maximum allowable magnitude of the relative velocity vector
v_x	=	radial velocity
$(\hat{X}, \hat{Y}, \hat{Z})$	=	Earth-centered inertial coordinate system
$\boldsymbol{x} = (\ell^T, \dot{\ell}^T)^T$	=	state vector in local-vertical/local-horizontal frame
$\bar{\boldsymbol{x}}$	=	nominal state vector
$\boldsymbol{x}_{\text{actual}}$	=	actual state vector
(x, y, z)	=	coordinate values in the local-vertical/local-horizontal coordinate system
$(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$	=	unit vectors of the local-vertical/local-horizontal coordinate system
$\mathcal{Y}$	=	function defining the J_2-invariant orbit for a given relative position and reference orbit
$(\alpha_x, \alpha_y, \alpha_z)$	=	angular acceleration of coordinate system about (x, y, z) axes
β	=	rate at which the size of the trust region decreases
Δt	=	length of time step

Presented as Paper 2012-4583 at the AIAA/AAS Astrodynamics Specialist Conference, Minneapolis, MN, 13–16 August 2012; received 14 August 2013; revision received 24 December 2013; accepted for publication 5 January 2014; published online 22 April 2014.

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

ϵ	=	tolerance of sequential convex programming convergence
θ	=	argument of latitude
μ	=	gravitational constant
$\mathcal{X}_L$	=	trust region for convex optimization
Ω	=	right ascension of the ascending node
$\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$	=	vector of rotation rates of the local-vertical/local-horizontal frame
$\|\cdot\|_p$	=	ℓ_p norm of a vector, $p \in [1, \infty]$

Subscripts

f	=	final condition (t is equal to t_f)
i	=	spacecraft i
j	=	spacecraft j
m	=	iteration m
0	=	initial condition (t is equal to 0)

I. Introduction

SPACECRAFT formation flying has been a major area of research over the past decades. Recently, the idea of formation flying has been extended to create swarms of spacecraft [1,2] that contain a large number (hundreds to thousands) of femtosatellites (100-gram-class spacecraft), also known as femtosats. 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 [3] have been shown to maintain the swarm shape and provide collision-free motion for hundreds of orbits. These orbits are very effective at swarm keeping once the swarm is in a desired formation. However, another important requirement for swarm missions is the guidance and control of the swarm reconfiguration. The goal of this paper is to develop a fuel and computationally efficient guidance and control algorithm for the reconfiguration of a swarm of spacecraft located in low Earth orbit (LEO). This algorithm will transfer the spacecraft from one set of J_2-invariant passive relative orbits (PROs) to another. In addition to being fuel and computationally efficient, the algorithm should provide collision-free motion 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 [4–12] and multivehicle control research [13–16] has presented multivehicle guidance and control methods. 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 must be different from previous research because they need to simultaneously address the large number of agents, the modest capabilities of each individual agent, and the complex dynamic environment. Specifically, the large number of spacecraft makes collision avoidance a major challenge. Also, the limited computational capabilities of each agent require that the swarm reconfiguration algorithm is very simple so that it can be run onboard the femtosats in real time.

A purely centralized algorithm can find fuel-efficient trajectories for reconfiguration but scales very poorly with the number of spacecraft [17]. On the other hand, a decentralized algorithm can generate trajectories with computational efficiency but will need a reactive collision-avoidance algorithm [18], 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 [18]. 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 [19,20]. Therefore, many optimal control 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 [21] have been used for trajectory optimization, but these methods solve a centralized problem that scales poorly with the number of spacecraft due to the coupling of spacecraft in the collision-avoidance requirements. Mixed integer linear programming can be used to enforce collision-avoidance constraints and has been implemented in real time [22] as well as used for preplanning trajectories [23,24]. However, these algorithms 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 [25] has been used in multivehicle trajectory design, and it has been shown that it can be efficiently solved to achieve a global optimum by state-of-the-art interior point methods. Convex optimization has been used to implement a receding horizon controller for a convex problem [26]. Additionally, convex optimization has been used to find collision-free trajectories for a formation reconfiguration [27] and robotic motion planning [28]. However, convexifying the collision constraints results in an overly conservative approximation of the collision-avoidance region. In the present paper, sequential convex programming (SCP) [29] 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 [30,31], to convert the convex programs to semidefinite programs (SDPs) or second-order-cone programs (SOCPs). These programs can then be solved by SDP or SOCP solvers, such as SDPT3 [32,33] (MATLAB) or MOSEK [34] (C/C++ or MATLAB).

By solving the swarm reconfiguration as an optimization problem, the entire trajectory is generated for each spacecraft at the initial time. In our prior work [35], it is shown that these trajectories can be computed onboard the femtosats. However, calculating the entire trajectory, with collision avoidance, for each spacecraft at the initial time requires each spacecraft to have all-to-all communication capabilities. To relax this assumption, the swarm reconfiguration is formulated as a decentralized model predictive control (MPC), or receding horizon control, problem using SCP to solve the optimizations.

MPC has been a major research area for over a decade [36,37]. In recent years, the original MPC problem has been modified to create robust MPC [38–40] and fast MPC [41,42]. Additionally, MPC has been used in applications similar to swarm guidance, such as vehicle maneuvering [38], formation flying [43], and spacecraft landing [44]. In all of these variations and applications of MPC, the basic idea remains the same. MPC computes the control input by optimizing over a finite-horizon subject to control and state constraints with the current state as the initial state of the optimization. Then, the control input is applied to the system until a new computation is completed giving an updated control input.

The goal of this paper is to develop a model predictive control implementation, which provides fuel-optimal collision-free motion for the reconfiguration of swarms of spacecraft and can be implemented on a femtosat with limited computation and communication capabilities. The MPC–SCP algorithm presented in this paper will build upon prior work by Morgan et al. on J_2-invariant orbits [3] and the current authors on optimal swarm trajectories [35]. The J_2-invariant conditions from prior work by Morgan et al. [3] are used as the boundary conditions for the swarm reconfiguration. The SCP algorithm [35] will be used to compute the optimizations used in the MPC implementation, which results in a fully decentralized reconfiguration algorithm. The algorithms developed in this paper provide real-time collision-free trajectories for a large number of spacecraft (in hundreds or thousands). Additionally, our algorithms are equally applicable to formation flying of a smaller number (3–10) of larger spacecraft.

The novelty of the MPC implementation using SCP (MPC–SCP) is that it decentralizes the computations and communications

required for swarm reconfiguration with collision avoidance. This allows the algorithm to handle hundreds to thousands of spacecraft in real time with calculations performed onboard the femtosats. Additionally, the MPC–SCP implementation offers several other advantages. First, the limited horizon of the MPC–SCP implementation greatly reduces the size of the SCP problem and, therefore, the run time. Additionally, the limited horizon allows the algorithm to include collision-avoidance constraints for only the neighboring spacecraft. This decentralizes the communication requirements of the SCP algorithm. Finally, by running the SCP algorithm multiple times, any differences between the desired and actual trajectories, which can be caused by errors or uncertainties, are accounted for when computing the future trajectories. This provides some robustness to the MPC–SCP implementation that is not present when the SCP algorithm is run only once at the initial time.

The paper is organized as follows. In Sec. II, the swarm reconfiguration is discussed and the SCP method is described. In Sec. III, the problem of converting to convex form is discussed and SCP is applied to the problem. In Sec. IV.A, the collision-avoidance constraints are discussed and the decentralized algorithm is presented. In the remainder of Sec. IV, the SCP problem is implemented using MPC and the effectiveness of this algorithm is investigated, along with the stability, feasibility, and robustness. In Sec. V, the results of simulations and the effectiveness of each algorithm are discussed.

II. Guidance of Swarms of Spacecraft

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 [3] 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 optimal control 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 toward the vernal equinox, the $\hat{Z}$ direction points toward 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 $\boldsymbol{x}_j = [\,x_j \quad y_j \quad z_j \quad \dot{x}_j \quad \dot{y}_j \quad \dot{z}_j\,]^T$.

The optimal control problem for swarm reconfiguration is written using the LVLH coordinates and dynamics. The equations of motion for spacecraft in the LVLH frame ($\ell_j = (x_j, y_j, z_j)^T$) are [45]

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

where the function $\boldsymbol{g}(\ell_j, \text{œ}) \in \mathbb{R}^3$ is defined in the Appendix, and the matrix $\boldsymbol{S}(\boldsymbol{\omega}) \in \mathbb{R}^{3\times3}$ is defined as $\boldsymbol{S}(\boldsymbol{\omega})\dot{\ell} = \boldsymbol{\omega} \times \dot{\ell}$. 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, magnitude of the specific 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 $\boldsymbol{\omega}(t)$ are also determined by $\text{œ}(t)$ (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:

$$\dot{\text{œ}} = \boldsymbol{f}_{\text{chief}}(\text{œ}(t), \boldsymbol{u}_{\text{chief}}) \tag{2}$$

where the right-hand side of this equation is defined in the Appendix. Note that Eqs. (1–2) 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 optimal control problem. Therefore, the dynamics constraints are given by Eq. (1) with known parameters given by Eq. (2).

The objective of the optimal swarm reconfiguration is to minimize the $\mathcal{L}_1$-norm of the control input. The $\mathcal{L}_1$-norm of the control input is equivalent to the total fuel used during the transfer [46]. Therefore, we can define the swarm reconfiguration as follows:

Problem 1: The nonlinear optimal control problem is

$$\min_{\boldsymbol{u}_j, j=1,\ldots,N} \sum_{j=1}^{N} \int_0^{t_f} \|\boldsymbol{u}_j(t)\|_p \, \mathrm{d}t \text{ subject to Eq.(1)} \tag{3}$$

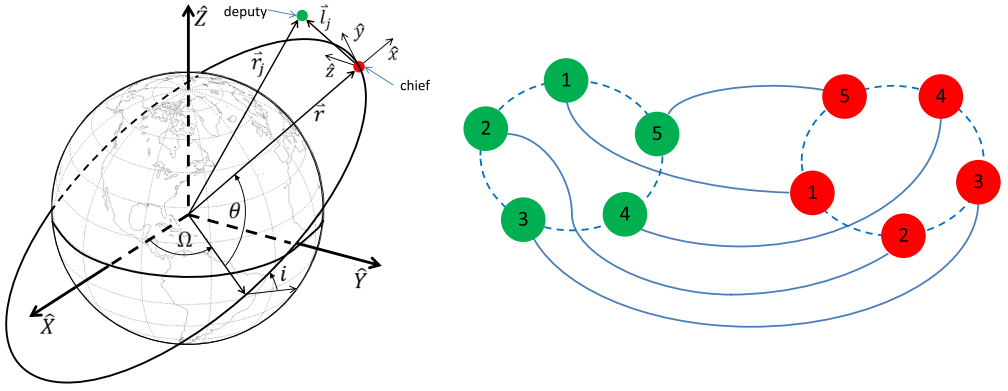
and

$$\|\boldsymbol{u}_j(t)\|_q \le U_{\max} \quad \forall\, t \in [0, t_f], \qquad j = 1, \ldots, N \tag{4}$$

$$\|C[\boldsymbol{x}_j(t) - \boldsymbol{x}_i(t)]\|_2 \ge \bar{R}_{\text{col}} \quad \forall\, t \in [0, t_f], \quad i > j, \qquad j = 1, \ldots, N-1 \tag{5}$$

$$\boldsymbol{x}_j(0) = \boldsymbol{x}_{j,0}, \qquad \boldsymbol{x}_j(t_f) = \boldsymbol{x}_{j,f} \qquad j = 1, \ldots, N \tag{6}$$

where $C = [\,\mathbf{I}_{3\times3} \quad 0_{3\times3}\,]$ and $\boldsymbol{x}_j = (\ell^T, \dot{\ell}^T)^T$. Equation (4) represents the limitation on the magnitude of the control vector, with $U_{\max}$ being the maximum allowable control magnitude; Eq. (5) is the collision-avoidance constraint, with R_{col} being the minimum



a) ECI ($\hat{X}$, $\hat{Y}$, $\hat{Z}$) and LVLH frames ($\hat{x}$, $\hat{y}$, $\hat{z}$) **b) Spacecraft swarm reconfiguration**

Fig. 1 Visualization of the relative coordinate system and a spacecraft swarm reconfiguration [3].

allowable distance between two spacecraft; and Eq. (6) contains the initial state constraint and the final state constraint. Although any reachable initial and terminal conditions can be used for Eq. (6), the simulations in Sec. V use the J_2-invariant conditions developed in prior work by Morgan et al. [3]. These J_2-invariant relative orbits have been shown to provide collision-free motion of hundreds of spacecraft for hundreds of orbits. Therefore, these orbits are chosen as the parking orbits for the spacecraft before and after the reconfiguration. Given the relative position $\ell_j(\tau), \tau \in \{0, t_f\}$, and the chief orbit $\mathfrak{oe}(\tau)$, the velocity vector that yields J_2-invariant PRO is given by some function $\dot{\ell}_j(\tau) = \mathcal{Y}(\ell_j(\tau), \mathfrak{oe}(\tau))$ (see [3] for details).

Remark 1 (p-norm): The norms used in Eqs. (3) and (4), $\|\cdot\|_p$ and $\|\cdot\|_q$, respectively, are dependent on the thruster architecture used on the spacecraft. Usually, $p, q \in \{1, 2, \infty\}$, where p and q can be the same or different. For a spacecraft with a single thruster, the values for p and q will both be 2. However, the femtosats considered in this paper are assumed to have thrusters in each direction $(\hat{x}, \hat{y}, \hat{z})$ with a single fuel tank. Each thruster has a limit on the amount of thrust that can be produced, which requires that $q = \infty$ in Eq. (4). Additionally, the lifetime of each femtosat is limited by the fuel remaining. Since each thruster requires fuel from the same tank, the goal is to minimize the sum of the magnitudes of the control components, i.e., the 1-norm. Therefore, $p = 1$ in Eq. (3). Throughout this paper, the 1-norm will be used for the objective function and the ∞-norm will be used for the control constraint. However, it is important to note that the convex optimizations developed in the following sections are valid for $p, q \in \{1, 2, \infty\}$.

It is important to note that the objective function and the constraints of Eqs. (4) and (6) already satisfy the requirements for a convex programming problem. Therefore, only the dynamics [Eq. (1)] and the collision-avoidance constraints [Eq. (5)] 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 (5), 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. (5) 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

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. Equation (1) can be rewritten as follows $\forall\ j = 1, \ldots, N$:

$$\dot{\boldsymbol{x}}_j = \boldsymbol{f}(\mathbf{x}_j, \mathfrak{oe}) + B\boldsymbol{u}_j \tag{7}$$

where $B = [\,0_{3\times3} \quad \boldsymbol{I}_{3\times3}\,]^T$. Linearizing Eq. (7) yields

$$\dot{\boldsymbol{x}}_j = A(\bar{\mathbf{x}}_j, \mathfrak{oe})\boldsymbol{x}_j + B\boldsymbol{u}_j + c(\bar{\boldsymbol{x}}_j, \mathfrak{oe}) \tag{8}$$

where $\bar{\boldsymbol{x}}_j$ 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, $A(\bar{\boldsymbol{x}}_j, \mathfrak{oe})$ and $c(\bar{\boldsymbol{x}}_j, \mathfrak{oe})$ are

$$A(\bar{\boldsymbol{x}}_j, \mathfrak{oe}) = \begin{bmatrix} 0_{3\times3} & \boldsymbol{I}_3 \\ -\frac{\partial \mathbf{g}}{\partial \mathbf{x}_j}\Big|_{\bar{\boldsymbol{x}}_j} & -2\boldsymbol{S}(\boldsymbol{\omega}) \end{bmatrix},$$
$$c(\bar{\boldsymbol{x}}_j, \mathfrak{oe}) = \begin{bmatrix} 0_{3\times1} \\ -\boldsymbol{g}(\bar{\ell}_j, \mathfrak{oe}) + \frac{\partial \mathbf{g}}{\partial \mathbf{x}_j}|_{\bar{\boldsymbol{x}}_j}\bar{\boldsymbol{x}}_j \end{bmatrix} \tag{9}$$

Hence, Eq. (8) is a fully controllable system.

The next step in the process of converting Eq. (1) into a constraint that can be used in convex programming is to convert the ordinary differential equation in Eq. (8) to a finite number of algebraic constraints. To do this, the problem is discretized using a zero-order-hold approach such that

$$\boldsymbol{u}_j(t) = \boldsymbol{u}_j[k], \qquad t \in [t_k, t_{k+1}), \qquad k = 0, \ldots, T-1 \tag{10}$$

where $t_f = T\Delta t$; $t_0 = 0$; $t_T = t_f$; and $\Delta t = t_{k+1} - t_k$ for $k = 0, \ldots, T-1$. This method of discretization reduces Eq. (8) to

$$\boldsymbol{x}_j[k+1] = A_j[k]\boldsymbol{x}_j[k] + B_j[k]\boldsymbol{u}_j[k] + c_j[k],$$
$$k = 0, \ldots, T-1, \qquad j = 1, \ldots, N \tag{11}$$

where $\boldsymbol{x}_j[k] = \boldsymbol{x}_j(t_k)$, $\boldsymbol{u}_j[k] = \boldsymbol{u}_j(t_k)$, $\mathfrak{oe}[k] = \mathfrak{oe}(t_k)$, and

$$A_j[k] = e^{A(\bar{\boldsymbol{x}}_j(t_k), \mathfrak{oe}(t_k))\Delta t}, \qquad B_j[k] = \int_0^{\Delta t} e^{A(\bar{\boldsymbol{x}}_j(t_k), \mathfrak{oe}(t_k))\tau} B\, \mathrm{d}\tau,$$
$$c_j[k] = \int_0^{\Delta t} e^{A(\bar{\boldsymbol{x}}_j(t_k), \mathfrak{oe}(t_k))\tau} c(\bar{\boldsymbol{x}}_j(t_k), \mathfrak{oe}(t_k))\, \mathrm{d}\tau \tag{12}$$

Now that the nonlinear continuous-time equations of motion from Eq. (1) have been rewritten as linear, finite-dimensional constraints in Eq. (11), they can be used in a convex programming problem. The constraints from Eqs. (4–6) can be written in discretized form as

$$\|\boldsymbol{u}_j[k]\|_\infty \le U_{\max} \qquad k = 0, \ldots, T-1, \qquad j = 1, \ldots, N \tag{13}$$

$$\|C(\boldsymbol{x}_j[k] - \boldsymbol{x}_i[k])\|_2 \ge R_{\mathrm{col}} \qquad k = 0, \ldots, T,$$
$$i > j, \qquad j = 1, \ldots, N-1 \tag{14}$$

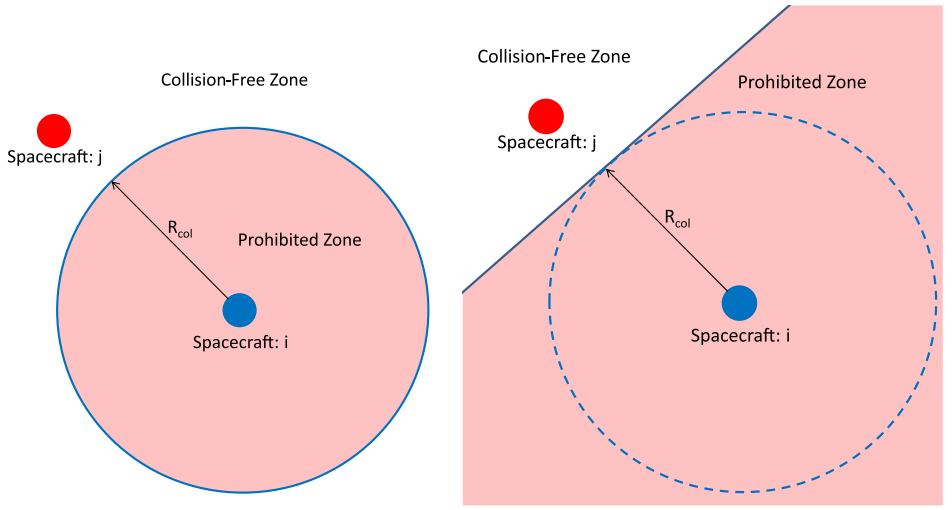
$$\boldsymbol{x}_j[0] = \boldsymbol{x}_{j,0}, \qquad \boldsymbol{x}_j[T] = \boldsymbol{x}_{j,f} \qquad j = 1, \ldots, N \tag{15}$$

Note that the only constraint that does not satisfy the requirements of convex programming is Eq. (14). 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 that defines the forbidden region is approximated by a plane that is tangent to the sphere and perpendicular to the line segment connecting the nominal position $\bar{\boldsymbol{x}}_j$ of the spacecraft and the object. This idea is shown in two dimensions using a line and a circle in Fig. 2.

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 three-dimensional 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.



a) Nonconvex prohibited zone **b) Convex approximation of prohibited zone**

Fig. 2 Convexification of the two-dimensional (2-D) collision-avoidance constraint.

Figure 3 shows the collision-free zone for a spacecraft surrounded by multiple neighbors. When multiple neighboring spacecraft (center dot) are in the vicinity of spacecraft j (outer area dots), 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.

Proposition 1 (convexification of collision avoidance constraint): A sufficient condition for the collision-avoidance constraints to hold from Eq. (14) is

$$(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])^T C^T C(\boldsymbol{x}_j[k]-\boldsymbol{x}_i[k]) \geq R_{\text{col}}\|C(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])\|_2$$
$$k=0,\ldots,T, \qquad i>j, \qquad j=1,\ldots,N-1 \tag{16}$$

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

$$(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])^T C^T C(\boldsymbol{x}_j[k]-\boldsymbol{x}_i[k]) \geq R_{\text{col}}\|C(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])\|_2$$
$$\|C(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])\|_2\|C(\boldsymbol{x}_j[k]-\boldsymbol{x}_i[k])\|_2\cos\phi \geq R_{\text{col}}\|C(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])\|_2$$
$$\|C(\boldsymbol{x}_j[k]-\boldsymbol{x}_i[k])\|_2\cos\phi \geq R_{\text{col}}$$
$$\|C(\boldsymbol{x}_j[k]-\boldsymbol{x}_i[k])\|_2 \geq \|C(\boldsymbol{x}_j[k]-\boldsymbol{x}_i[k])\|_2\cos\phi \geq R_{\text{col}}$$

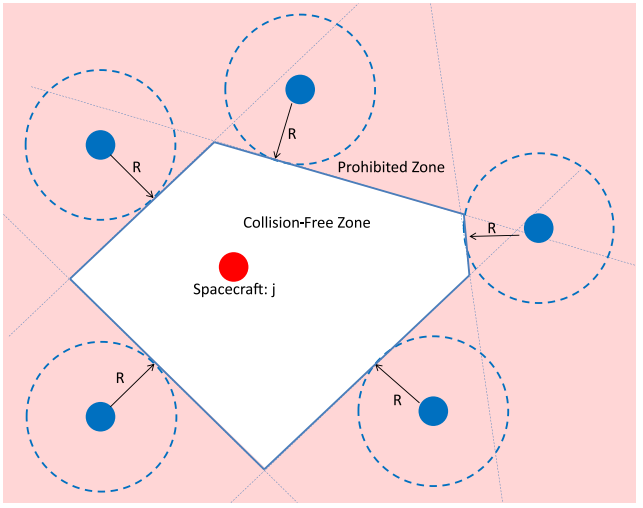


Fig. 3 Collision-free zone for a spacecraft with five neighbors using affine collision-avoidance constraints.

This reestablishes Eq. (14) and proves sufficiency. The nominal trajectories from the previous iteration of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{x}}_j$, 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. (16) are affine and in a form that can be used in a convex programming problem. □

Remark 2 (nominal trajectories): The nominal trajectory $\bar{\boldsymbol{x}}_j$ represents an initial guess for the actual trajectory $\boldsymbol{x}_j$ and is used to convexify the collision-avoidance constraint. The closer the nominal trajectory is to the actual trajectory, the more accurate the convex program will be. The nominal trajectory plays an important role in the iterative method developed in Sec. III.C, where it is further defined.

C. Sequential Convex Programming Method

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 (convex program):

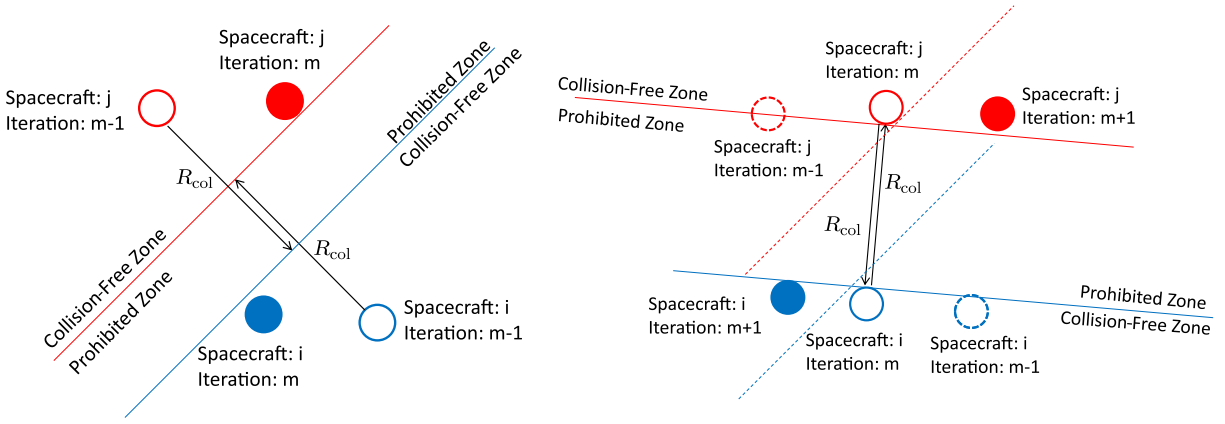
$$\min_{\boldsymbol{u}_j, j=1,\ldots,N} \sum_{j=1}^{N}\sum_{k=0}^{T-1}\|\boldsymbol{u}_j[k]\|_1\Delta t \text{ subject to } \{(11),(13),(15),(16)\} \tag{17}$$

where Problem 1 has been discretized, and the constraints of Eqs. (1) and (5) have been approximated by Eqs. (11) and (16), respectively.

The approximations used to get the dynamics and collision-avoidance constraints into their convex forms [Eqs. (11) and (16)] require a nominal state $\bar{\boldsymbol{x}}_j[k]$ 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. To ensure that the nominal vectors are good estimates of the actual state vectors, SCP is used. SCP is a method for solving nonconvex optimizations using convex programming [29]. To use SCP, the nonconvex problem is approximated by a convex problem, as has been done in Secs. III.A and 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, i.e., $\bar{\boldsymbol{x}}_j[k]=\boldsymbol{x}_{j,m-1}[k]$, $\forall\ k$ for iteration m. This process continues until the following condition is satisfied:

$$\|\boldsymbol{x}_{j,m}[k]-\boldsymbol{x}_{j,m-1}[k]\|_\infty < \epsilon, \qquad \forall\ j,k \tag{18}$$

To enforce the collision-avoidance constraints, each spacecraft communicates its own nominal trajectory to the other spacecraft. This requirement will be relaxed in Sec. IV.



a) Collision-free zone for iteration m **b) Collision-free zone for iteration m+1**

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

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 nonoptimal 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 a single iteration of convex programming.

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 $\forall\ j$ as

$$\mathcal{X}_{j,L_m} = \{\boldsymbol{x}_j \,|\, \|\boldsymbol{x}_{j,m}[k] - \boldsymbol{x}_{j,m-1}[k]\|_\infty \le L_m, \forall\ k\} \tag{19}$$

where L_m is the size of the trust region during iteration m. Additionally, the trust region in Eq. (19) can be used to ensure that the SCP algorithm converges. To ensure convergence, the size of the trust region is updated as follows:

$$L_{m+1} = \beta L_m \tag{20}$$

where $\beta \in (0,1)$ is a parameter that determines the worst-case rate of convergence, and $\boldsymbol{x}_{j,m}[k]$ denotes the relative state vector of the jth spacecraft, at the kth time step, after the mth iteration of the convex program.

If the run time of each iteration t_{iter} and the total allowable run time t_{run} of the algorithm are known, the maximum allowable number of iterations M can be determined by the following equation:

$$M = \left\lfloor \frac{t_{\text{run}}}{t_{\text{iter}}} \right\rfloor \tag{21}$$

Where t_{iter} is dependent on the computer and optimization software used, and t_{run} is further defined in Proposition 5 in Sec. IV. This maximum number of iterations M can then be used to determine the required value of β. From Eq. (20),

$$L_M = \beta^M L_0 \tag{22}$$

To guarantee convergence by iteration M, L_M must be less than ϵ from Eq. (18). Substituting Eq. (22) for L_M and solving for β yields

$$\beta < \left(\frac{\epsilon}{L_0}\right)^{\frac{1}{M}} \tag{23}$$

where ϵ is the tolerance of SCP convergence, and L_0 is the initial trust region size.

The SCP method is effective for formations containing tens of spacecraft 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. Model Predictive Control Using Sequential Convex Programming

A. Decentralized Sequential Convex Programming Method

As mentioned in Sec. III, even in convex form, the centralized swarm reconfiguration algorithm still scales poorly because of the number of collision-avoidance constraints. Therefore, the problem must be 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 the SCP algorithm 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, R_{safe}, so that $R_{\text{safe}} > R_{\text{col}}$, 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 SCP that can be used to reduce the computational complexity is the fact that, as the number of iterations increases, the difference between $\boldsymbol{x}_{j,m}[k]$ and $\boldsymbol{x}_{j,m-1}[k]$ 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, the optimization can be rewritten as follows:

Problem 3 (decentralized convex program):

$$\min_{\boldsymbol{u}_j, j=1,\ldots,N} \sum_{j=1}^{N} \sum_{k=0}^{T-1} \|\boldsymbol{u}_j[k]\|_1 \Delta t \text{ subject to Eqs. (11), (13), (15)} \tag{24}$$

and

$$(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])^T C^T C(\boldsymbol{x}_j[k]-\bar{\boldsymbol{x}}_i[k]) \geq R_{\text{col}}\|C(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])\|_2$$
$$k=0,\ldots,T, \qquad i\in\mathcal{I}_j, \qquad j=1,\ldots,N-1 \tag{25}$$

where the nominal trajectory is calculated using $\bar{\boldsymbol{x}}_j[k]=\boldsymbol{x}_{j,m-1}[k]$ and

$$\mathcal{I}_j=\{i|\exists k\in 0,\ldots,T, \text{ such that } \|C(\bar{\boldsymbol{x}}_i[k]-\bar{\boldsymbol{x}}_j[k])\|_2 \leq R_{\text{safe}} \text{ and } i<j\} \tag{26}$$

The decentralized SCP method is described in Method 1. The nominal values are not updated until after every spacecraft has completed its computation, as seen in line 15 of Method 1. 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 (open circle on the right) moving to a position (solid circle to the right of center) that is safe based on the previous location of spacecraft j (open circle on the left). However, spacecraft j has updated its position (solid circle in center), 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.

By adding the constraint $i<j$ to Eq. (26), only one of the spacecraft will try to avoid the other one. Using these modifications, the SCP method is shown in Method 1.

Remark 3 (spacecraft ordering): By forcing one spacecraft to avoid the other rather than allowing cooperative avoidance, the trajectories can potentially be farther from the optimal than one derived in a centralized 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 minimizes the disadvantages of using the decentralized 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.

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.

B. Model Predictive Control Method

To describe the MPC algorithm, Problem 4 and Problem 5 are defined. Problem 4 is defined so that the horizon for the optimization T_H does not reach the terminal time T for the reconfiguration. For this reason, a terminal cost is added to the objective [second term in Eq. (27)] to estimate the cost of completing the reconfiguration from the state and time at the end of the optimization horizon. Problem 4 is used in the MPC algorithm when the horizon of the optimization does not reach the terminal time of the reconfiguration. Problem 5 is very similar to Problem 3, with the only difference being the starting time. Problem 5 is used in the MPC algorithm when the horizon of the optimization goes beyond the terminal time of the reconfiguration. In both Problem 4 and Problem 5, the spacecraft are assumed to have limited communication ranges. Therefore, they can only communicate with the other spacecraft that are within their communication ranges. Problem 4 and Problem 5 are expressed as follows:

Problem 4 (convex optimization used in MPC–SCP if $k_0+T_H<T$):

$$\min_{\boldsymbol{u}_j} \sum_{k=k_0}^{k_0+T_H-1} \|\boldsymbol{u}_j[k]\|_1\Delta t_1 + \sum_{k=k_0+T_H}^{T-1} \|\boldsymbol{u}_j[k]\|_1\Delta t_2 \quad \forall\ j=1,\ldots,N \tag{27}$$

subject to

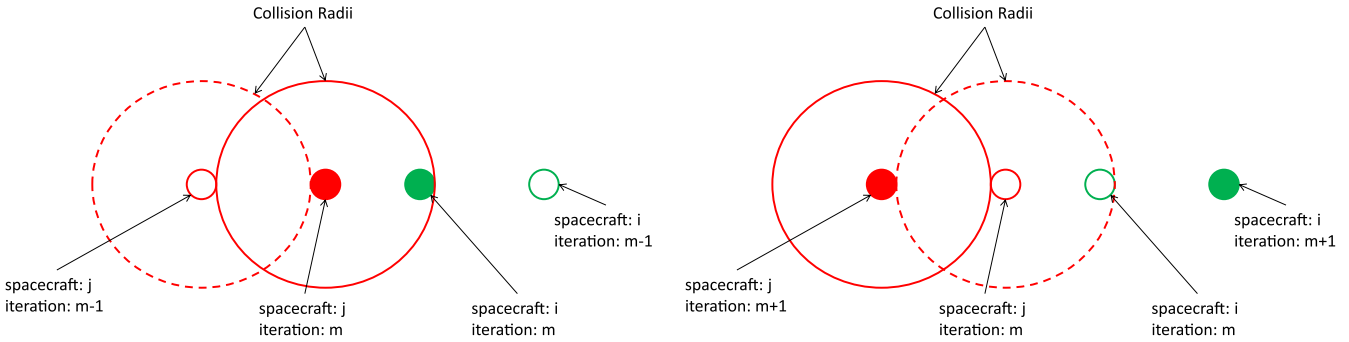
$$\boldsymbol{x}_j[k+1]=A_j[k]\boldsymbol{x}_j[k]+B_j[k]\boldsymbol{u}_j[k]+c_j[k],$$
$$k=k_0,\ldots,T-1, \qquad j=1,\ldots,N \tag{28}$$

$$(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])^T C^T C(\boldsymbol{x}_j[k]-\bar{\boldsymbol{x}}_i[k]) \geq R_{i,j}[k]\|C(\bar{\boldsymbol{x}}_j[k]-\bar{\boldsymbol{x}}_i[k])\|_2$$
$$k=k_0,\ldots,k_0+T_H, \qquad \{i,j\}:\ i\in\mathcal{N}_j,$$
$$\mathcal{N}_j=\{i|i<j, \|\boldsymbol{x}_j[k_0]-\boldsymbol{x}_i[k_0]\|_2\leq R_{\text{comm}}\} \tag{29}$$

Method 1 SCP method for decentralized problem

1: $\bar{\boldsymbol{x}}_j[k]:=\boldsymbol{0}_{6\times 1}, \forall\ j,k$
2: $\boldsymbol{x}_{j,0}[k]:=$ the solution to Problem 3 (Problem 4 or 5 when called from Method 2) excluding Eq. (25) (Eq. (29) or (34) when using Problem 4 or 5, respectively), $\forall\ j,k$
3: $\bar{\boldsymbol{x}}_j[k]:=\boldsymbol{x}_{j,0}[k], \forall\ j,k$
4: $\mathcal{K}:=\{1,\ldots,N\}$
5: $m:=1$
6: **while** $\mathcal{K}\neq\emptyset$ **do**
7: **for all** $j\in\mathcal{K}$ **do**
8: $\boldsymbol{x}_{j,m}[k]:=$ the solution to Problem 3 (problem 4 or 5 when called from Method 2), $\forall\ k$
9: **end for**
10: **for all** $j\in\mathcal{K}$ **do**
11: $\boldsymbol{x}_{j,m}[k]:=\boldsymbol{x}_{j,m-1}[k], \forall\ k$
12: **end for**
13: $\mathcal{K}:=\{1,\ldots,N\}$
14: **for all** j **do**
15: $\bar{\boldsymbol{x}}_j[k]:=\boldsymbol{x}_{j,m}[k], \forall\ k$
16: **if** $\|\boldsymbol{x}_{j,m}[k]-\boldsymbol{x}_{j,m-1}[k]\|_\infty<\varepsilon\ \forall\ k$ and $\|C(\boldsymbol{x}_{j,m}[k]-\boldsymbol{x}_{i,m}[k])\|_2>R_{\text{col}}, \forall\ k\ \forall\ i\in\mathcal{N}_j$[a] **then**
17: Remove j from $\mathcal{K}$
18: **end if**
19: **end for**
20: $m:=m+1$
21: **end while**
22: $M:=m-1$
23: $\boldsymbol{x}_j^M[k]$ is the approximate solution to Problem 1

[a]In the SCP-only (no MPC) algorithm, $\mathcal{N}_j=\{i|i\neq j\}$. $\mathcal{N}_j$ is further defined in Sec. IV.B.



a) Update from iteration m-1 to iteration m: collision **b) Update from iteration m to iteration m+1: overly conservative**

Fig. 5 Example of two spacecraft that have difficulty converging. This problem is solved by adding the second constraint in Eq. (26).

$$\|D\boldsymbol{x}_j[k]\|_2 \le V_{\max} \qquad k = k_0, \ldots, T, \qquad j = 1, \ldots, N \tag{30}$$

$$\|\boldsymbol{u}_j[k]\|_\infty \le U_{\max} \quad k = k_0, \ldots, T-1, \quad j = 1, \ldots, N \tag{31}$$

$$\boldsymbol{x}_j[k_0] = \boldsymbol{x}_{j,\text{actual}}, \qquad \boldsymbol{x}_j[T] = \boldsymbol{x}_{j,f} \qquad j = 1, \ldots, N \tag{32}$$

where $D = [\,0_{3\times3} \quad I_{3\times3}\,]$; Δt_1 and Δt_2 are the time step size before and after the end of the horizon, respectively; and $R_{i,j}[k] \ge R_{\text{col}}$ is the collision radius for spacecraft i and j at time k when perturbations are considered. The importance of having multiple time step sizes and time-varying collision-avoidance distances will be discussed in Sec. IV.E.

Problem 5 (convex optimization used in MPC–SCP if $T - T_H \le k_0 < T$):

$$\min_{\boldsymbol{u}_j} \sum_{k=k_0}^{T-1} \|\boldsymbol{u}_j[k]\|_1 \Delta t_1 \quad \forall\, j = 1, \ldots, N \tag{33}$$

subject to Eqs. (28), (30), (31), (32), and

$$\begin{aligned}
&(\bar{\boldsymbol{x}}_j[k] - \bar{\boldsymbol{x}}_i[k])^T C^T C(\boldsymbol{x}_j[k] - \bar{\boldsymbol{x}}_i[k]) \ge R_{i,j}[k]\|C(\bar{\boldsymbol{x}}_j[k] - \bar{\boldsymbol{x}}_i[k])\|_2 \\
&k = k_0, \ldots, T, \qquad \{i, j\}\colon i \in \mathcal{N}_j, \\
&\mathcal{N}_j = \{i | i < j, \|\boldsymbol{x}_j[k_0] - \boldsymbol{x}_i[k_0]\|_2 \le R_{\text{comm}}\}
\end{aligned} \tag{34}$$

The MPC implementation of the 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 for the optimal trajectory up to a finite horizon T_H. As the spacecraft approaches this horizon in real time, the SCP algorithm is rerun from the current time k_0 and position $\boldsymbol{x}_{j,\text{actual}}$ up to the new horizon $(k_0 + T_H)$. It is important to note that k_0 is the current time at the beginning of each MPC iteration and increases with time. In Eq. (32), $\boldsymbol{x}_{j,\text{actual}}$ is the real-time position and velocity of the spacecraft when the MPC algorithm is run. This value represents the initial condition of the MPC algorithm. This process is repeated until the spacecraft reaches the desired position $\boldsymbol{x}_{j,f}$ at the final time T. This process is shown in more detail in Method 2.

The SCP algorithm used to solve the optimizations in the MPC algorithms was given by Method 1. The SCP algorithm is written very generally, and it is assumed that the optimization problem to be solved, the time range of the optimization, and the pairs of spacecraft that can communicate are specified by the MPC algorithm.

The result of the MPC–SCP implementation is a fully decentralized optimal guidance algorithm with improved computation times as well as better robustness when sensor and actuator errors are included, as will be shown in Sec. IV.C. The decentralization of the swarm guidance algorithm greatly reduces the communication and computation requirements of the femtosats. Additionally, the increased robustness properties of this algorithm will reduce the fuel requirements for the femtosats. The benefits of the MPC implementation with respect to robustness and fuel efficiency are shown in Fig. 6. Figure 6a shows how an initial actuator or sensor error can cause the actual final position (large circle, upper right) to have a significant error with respect to the desired final position (large circle, bottom right) if the SCP algorithm is only run once. However, the MPC implementation in Fig. 6b can reduce this error by updating the desired trajectory based on the actual position and velocity ($\boldsymbol{x}_{j,\text{actual}}[k]$) at various points (small circles) throughout the reconfiguration. In addition to reducing final position errors, the MPC implementation reduces the computation, communication, and fuel requirements, which are especially important for femtosats due to their very limited volume and mass.

C. Sensor and Actuator Uncertainties

A major benefit of the MPC–SCP algorithm is the robustness to sensor and actuator uncertainties. Before analyzing the stability and feasibility of the MPC–SCP algorithm, in Secs. IV.D and IV.E, respectively, this subsection introduces the sensor and actuator uncertainties and their effects on the spacecraft's trajectories.

Assumption 1: The linearization and discretization errors of the convexification process are negligible compared to the errors in the control actuation and state measurements.

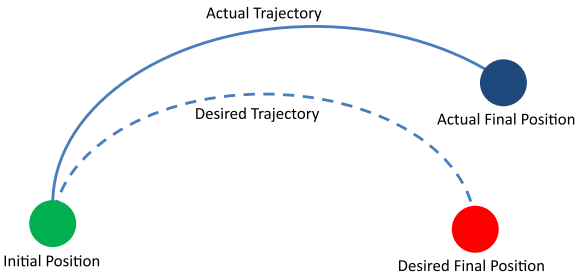
Remark 4 (negligible errors): The linearization errors are negligible because the nominal trajectory about which the linearization occurs ($\bar{\boldsymbol{x}}_j = \boldsymbol{x}_{j,m-1}$) and the actual trajectory ($\boldsymbol{x}_{j,m}$) are within ϵ, as given by line 16 of Method 1. Additionally, the discretization error is caused by using a zero-order hold on time-varying dynamics. However, the relative spacecraft dynamics do not change much over the length of a single time step, so using constant dynamics (A_j, B_j, c_j) does not introduce much error.

Assumption 2: The errors in the state measurement at time k_0 and control actuation at all times k are $\Delta\boldsymbol{x}_j[k_0] = \hat{\boldsymbol{x}}_j[k_0] - \boldsymbol{x}_j[k_0]$ and $\Delta\boldsymbol{u}_j[k] = \hat{\boldsymbol{u}}_j[k] - \boldsymbol{u}_j[k]$, respectively. These errors satisfy the following conditions:

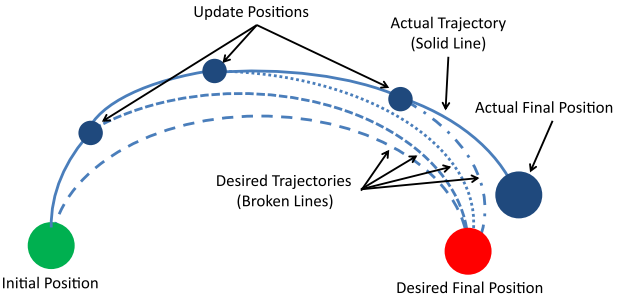
$$\|C\Delta\boldsymbol{x}_j[k]\|_2 \le \xi_x, \quad \|D\Delta\boldsymbol{x}_j[k]\|_2 \le \xi_v, \quad \|\Delta\boldsymbol{u}_j[k]\|_2 \le \xi_u, \quad \forall\, j, k \tag{35}$$

Method 2 MPC–SCP (main result)

1: $k_0 = 0$
2: **while** $k_0 \le T - T_H$, **do**
3: Solve Problem 4 using SCP (Method 1)
4: $\boldsymbol{x}_j[k]$ = state solution to Problem 4, $\forall\, j, k = k_0 \ldots k_0 + T_H$
5: $\boldsymbol{u}_j[k]$ = control solution to Problem 4, $\forall\, j, k = k_0 \ldots k_0 + T_H - 1$
6: Update k_0 to current time
7: **end while**
8: **while** $k_0 \le T$ **do**
9: Solve Problem 5 using SCP (Method 1)
10: $\boldsymbol{x}_j[k]$ = state solution to Problem 5, $\forall\, j, k = k_0 \ldots T$
11: $\boldsymbol{u}_j[k]$ = control solution to Problem 5, $\forall\, j, k = k_0 \ldots T - 1$
12: Update k_0 to current time
13: **end while**



a) Trajectory from running the SCP algorithm at the initial time only



b) Trajectory from running the MPC implementation of the SCP algorithm

Fig. 6 Convexification of the 2-D collision-avoidance constraint.

Proposition 2 (bound on the perturbed trajectory): Under Assumptions 1 and 2, the error ($\boldsymbol{d}_j(k,n)$) in the position of spacecraft accumulated between time steps k and $k+n$ can be bounded as follows:

$$\|\boldsymbol{d}_j(k,n)\|_2 \le \left\|\prod_{s=0}^{n-1} A_j[k+s]\right\|_{UL} \xi_x + \left\|\prod_{s=0}^{n-1} A_j[k+s]\right\|_{UR} \xi_v + \sum_{l=0}^{n-1} \left\|\prod_{s=l+1}^{n-l} A_j[k+s] B_j[k+l]\right\|_U \xi_u =: \bar{d}_j(k,n) \tag{36}$$

where $\boldsymbol{d}_j(k,n) = C\tilde{\boldsymbol{x}}_j[k+n]$, the propagated error between the perturbed state ($\hat{\boldsymbol{x}}_j[k]$) and the original state is $\tilde{\boldsymbol{x}}_j[k] = \hat{\boldsymbol{x}}_j[k] - \boldsymbol{x}_j[k]$, and

$$\prod_{s=0}^{n} Z[s] = Z[n]Z[n-1]\ldots Z[2]Z[1]Z[0], \qquad \prod_{s=n}^{n-1} Z[s] = I$$

$$Z_{6\times 6} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}, \qquad Z_{6\times 3} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}$$

$$\|Z_{6\times 6}\|_{UL} = \bar{\sigma}(Z_{11}), \qquad \|Z_{6\times 6}\|_{UR} = \bar{\sigma}(Z_{12}), \qquad \|Z_{6\times 3}\|_U = \bar{\sigma}(Z_1)$$

Proof: Define $\hat{\boldsymbol{x}}_j[k]$ as the perturbed state of spacecraft j at time k. The dynamics for $\hat{\boldsymbol{x}}_j[k]$ given Assumption 1 can be written as

$$\hat{\boldsymbol{x}}_j[k+1] = A_j[k](\boldsymbol{x}_j[k] + \Delta\boldsymbol{x}_j[k]) + B_j[k](\boldsymbol{u}_j[k] + \Delta\boldsymbol{u}_j[k]) + c_j[k] \tag{37}$$

Subtracting Eq. (11) from Eq. (37) yields

$$\tilde{\boldsymbol{x}}_j[k+1] = A_j[k]\Delta\boldsymbol{x}_j[k] + B_j[k]\Delta\boldsymbol{u}_j[k] \tag{38}$$

Expanding this equation for n time steps yields

$$\tilde{\boldsymbol{x}}_j[k+n] = \prod_{s=0}^{n-1} A_j[k+s]\Delta\boldsymbol{x}_j + \sum_{l=0}^{n-1}\prod_{s=l+1}^{n-l} A_j[k+s]B_j[k+l]\Delta\boldsymbol{u}_j[k+l] \tag{39}$$

Considering only the position components of the state and taking the norm of both sides results in

$$\|d_j(k,n)\|_2 \le \left\|\prod_{s=0}^{n-1} A_j[k+s]\right\|_{UL} \|C\Delta\boldsymbol{x}_j[k]\|_2 + \left\|\prod_{s=0}^{n-1} A_j[k+s]\right\|_{UR} \|D\Delta\boldsymbol{x}_j[k]\|_2 + \sum_{l=0}^{n-1}\left\|\prod_{s=l+1}^{n-l} A_j[k+s]B_j[k+l]\right\|_U \|\Delta\boldsymbol{u}_j[k]\|_2 \tag{40}$$

Applying the conditions in Assumption 2 establishes Eq. (36). □

Proposition 2 calculates an upper bound ($\bar{d}_j(k,n)$) on the drift of the perturbed trajectory of spacecraft j between time steps k and $k+n$. This allows a terminal ball to be constructed so that the perturbed terminal position lies inside the ball.

D. Stability

The stability of an MPC algorithm is dependent on the terminal cost function [second term in Eq. (27)]. In Method 2 (MPC–SCP), the terminal cost function evaluates the fuel required to go from $\boldsymbol{x}_{j,\text{actual}}[k]$ at $k_0 + T_H$ to $\boldsymbol{x}_{j,f}$ at T without considering collision-avoidance constraints during this part of the trajectory. There are two reasons not to enforce the collision-avoidance constraints when calculating the terminal cost function. First, the collision-avoidance constraints add complexity to the problem so removing them greatly reduces the time required for the computation. Second, the spacecraft can only communicate with other spacecraft within a certain distance of them.

The concept of Method 2 (MPC–SCP) is shown in Fig. 7. This figure shows the various stages of the MPC algorithm. The first stage is shown by the solid line in Fig. 7. This represents the actual trajectory that the spacecraft has traversed, and it occurs between $k = 0$ and $k = k_0$. The current time, and the initial time in the optimization, is represented by $k = k_0$. The next stage occurs between $k = k_0$ and $k = k_0 + T_H$, and it is shown as a dashed line in the figure. This represents the predicted trajectory, and collision avoidance is considered during this time period. The final stage is illustrated by the dotted line and extends from $k = k_0 + T_H$ to $k = T$. During this time, the predicted trajectory does not take into

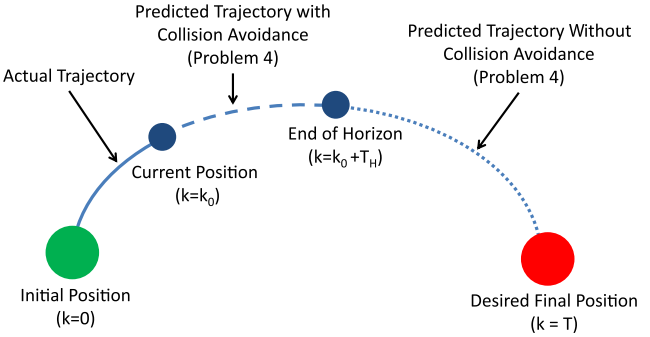


Fig. 7 Illustration of the optimization horizon used in the MPC algorithms.

account collision avoidance. It is important to note that, if the second stage (dashed line) extends beyond the final time, the final stage does not exist, and Problem 5 should be used instead of Problem 4.

Method 2 (MPC–SCP) uses the solution to Problem 4 or Problem 5 depending on whether or not the horizon of the optimization reaches the final time. In either case, the final state is bound from Eq. (32). Therefore, the trajectory is guaranteed to converge to the desired final set as long as the optimizations described in Problems 4 and 5 are feasible. The feasibility of the optimizations is discussed Sec. IV.E.

When the uncertainties described in Sec. IV.C are considered, the perturbed trajectories must still converge to the desired terminal positions.

Theorem 3 (stability of the perturbed trajectory): If the last MPC update occurs at time $T-n$ and Assumptions 1 and 2 hold, the perturbed trajectory will reach a terminal position such that

$$\|C(\hat{\boldsymbol{x}}_j[T]-\boldsymbol{x}_{j,f})\|_2 \le \bar{d}_j(T-n,n) \tag{41}$$

Proof: The MPC–SCP algorithm's terminal constraint [Eq. (32)] ensures that the unperturbed trajectory reaches the desired terminal state. The rest of the proof follows from Proposition 2 with $k=T-n$. □

E. Feasibility

For the trajectories in Method 2 (MPC–SCP) to converge, the optimizations must be feasible. Infeasibility of the optimization can result for two reasons: The collision-avoidance constraints cannot all be satisfied or the terminal constraint ($\boldsymbol{x}_j[T]=\boldsymbol{x}_{j,f}$) cannot be satisfied without violating the limit on the velocity and/or control vectors. The collision-avoidance infeasibility arises because collision avoidance is only considered up to the horizon of the optimization and other spacecraft can only be detected if they are within the communication radius R_{comm}. Therefore, collisions that occur after the optimization horizon (k_0+T_H) or with spacecraft outside of the communication radius R_{comm} are not considered until a later time step. For this reason, several conditions are introduced to ensure that collision avoidance is guaranteed.

To guarantee feasibility, an artificial constraint [Eq. (30)] is imposed on the problem in order to bound the distance that each spacecraft can move during each time step. The maximum velocity $V_{\max}$ can be approximated from the relative dynamics in Eq. (28). Assuming that the original optimization problem described by Problem 2 is feasible, i.e., the initial and terminal constraints can be satisfied without violating the collision-avoidance constraints, the following conditions ensure that Method 2 (MPC–SCP) is feasible:

Proposition 4 (detectable collisions): All spacecraft that can cause collisions within the current horizon are able to be detected if

$$R_{\text{comm}} \ge 2V_{\max}T_H\Delta t_1 + R_{\text{col}} \tag{42}$$

Proof: The length of the horizon is the number of time steps T_H multiplied by the length of each time step Δt_1. Additionally, the maximum relative velocity between two spacecraft is $2V_{\max}$. Therefore, the maximum change in the relative distance between two spacecraft results in $2V_{\max}T_H\Delta t_1$. Any pair of spacecraft can change their relative distance by this amount before the end of the MPC horizon. Therefore, this distance must be less than the difference between the communication radius R_{comm} and the collision radius R_{col}. This establishes Eq. (42). □

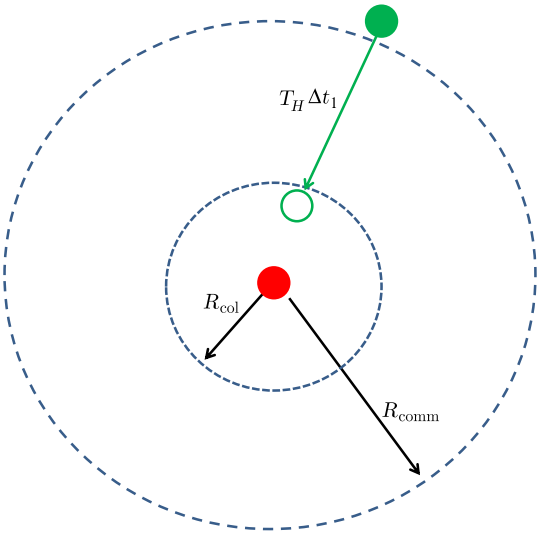


Fig. 8 Illustration of a pair of spacecraft that do not have sufficient communication radii to guarantee detectable collisions.

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. An illustration of a pair of spacecraft that violate this condition is shown in Fig. 8.

Proposition 5 (computational feasibility): The new control sequence can be computed before the previous horizon is reached if

$$t_{\text{run}} \le T_H\Delta t_1 \tag{43}$$

Proof: Since collision avoidance is not enforced after the end of the MPC horizon, a new control sequence must be computed before the current control sequence reaches the end of the horizon. Otherwise, the control sequence will not necessarily avoid collisions. Therefore, the computation time of each step of the MPC algorithm t_{run} must be less than the length of the MPC horizon ($T_H\Delta t_1$). This results in Eq. (43). □

Propositions 4 and 5 ensure the feasibility of Method 2 (MPC–SCP). Since the optimizations performed by this algorithm are feasible, the collision-avoidance constraints are satisfied and there are no collisions at the discrete time steps. However, there is still a possibility that collisions occur in between time steps when the collision-avoidance constraints are not enforced. The following theorem addresses this issue.

Theorem 6 (collision avoidance between time steps): If two spacecraft are collision free during two consecutive time steps k and $k+1$, and Eqs. (44) and (45) are satisfied, then the two spacecraft are collision free in the interval $t\in[k\Delta t_1,(k+1)\Delta t_1]$:

$$\Delta t_1 < \frac{R_{\text{col}}}{V_{\max}} \tag{44}$$

$$R_{\text{col}} \ge \begin{cases} \sqrt{\left(\bar{R}_{\text{col}}+\frac{a^*\Delta t_1^2}{4}\right)^2+(V_{\max}\Delta t_1)^2-\frac{(a^*\Delta t_1^2)^2}{4}} & \text{if } a^* < \min\left\{\frac{2V_{\max}}{\Delta t_1}, a_{\max}\right\} \\ \sqrt{\left(\bar{R}_{\text{col}}+\frac{a_{\max}\Delta t_1^2}{4}\right)^2+(V_{\max}\Delta t_1)^2-\frac{(a_{\max}\Delta t_1^2)^2}{4}} & \text{else if } a_{\max} < \frac{2V_{\max}}{\Delta t_1} \\ \bar{R}_{\text{col}}+\frac{V_{\max}\Delta t_1}{2} & \text{else} \end{cases} \tag{45}$$

where

$$a^* = \frac{2}{\sqrt{3}}\sqrt{\frac{R_{\text{col}}^2}{\Delta t_1^4} - \frac{V_{\max}^2}{\Delta t_1^2}} \tag{46}$$

and $\bar{R}_{\text{col}}$ is the minimum allowable distance between every pair of spacecraft in continuous time.

Proof: The MPC–SCP algorithm (Method 2) guarantees that the trajectories are collision free at the discrete time steps. However, the trajectories must also be collision free in between time steps in order to guarantee collision-free trajectories. The first step to ensuring that collisions do not occur in between time steps is to establish a condition which prevents two spacecraft from passing through each other. Two spacecraft can move by a relative distance of $2V_{\max}\Delta t_1$ in one time step. This distance must be less than twice the collision radius R_{col} to prevent the spacecraft from passing through each other. This establishes Eq. (44).

Now that the spacecraft cannot pass through each other, the minimum possible relative distance between the spacecraft is established for a given $(V_{\max}, a_{\max})$, where $V_{\max}$ is the maximum allowable velocity and $a_{\max}$ is the maximum allowable acceleration, which includes acceleration due to both the control and the dynamics. The discretization method uses a constant acceleration in between time steps. Consider this scenario from the reference frame centered at spacecraft j, as shown in Fig. 9. In this figure, the subscript $i|j$ denotes the location of spacecraft i with respect to j, where $i \in \mathcal{N}_j$. In the worst-case scenario, the distance between the two spacecraft is R_{col} at both time steps, and both the relative velocity and acceleration vectors are in the plane defined by spacecraft j and the initial and final positions of spacecraft i. This scenario is depicted in Fig. 9, and the distances in Eqs. (47)–(50) are defined in this figure.

Since the acceleration vector is constant, the minimum distance $\bar{R}_{\text{col}}$ occurs when the problem is symmetric, as shown in Fig. 9:

$$L = 2V_{\max}\cos\phi\Delta t_1 \tag{47}$$

$$b = \sqrt{R_{\text{col}}^2 - \left(\frac{L}{2}\right)^2} \tag{48}$$

$$d = V_{\max}\sin\phi\Delta t_1 - a\frac{\Delta t_1^2}{4} \tag{49}$$

$$\bar{R}_{\text{col}} = b - d \tag{50}$$

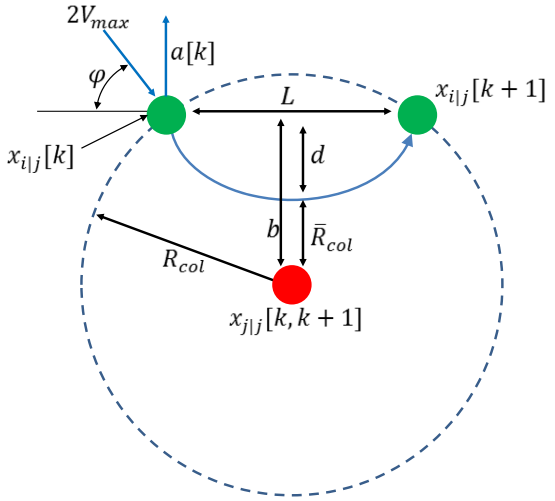


Fig. 9 Illustration of the worst-case scenario for collisions in between time steps.

where L is length of the segment connecting the position of spacecraft i relative to spacecraft j at time steps k and $k+1$, b is the minimum distance between this segment and spacecraft j, d is the distance between the line segment measured by L and the point of closest approach of spacecraft i relative to spacecraft j, and ϕ is the angle between the initial velocity vector and the line segment measured by L. Since the closest distance occurs at $\Delta t_1/2$ due to symmetry, the velocity in the direction of the acceleration vector must be zero at this time:

$$2V_{\max}\sin\phi = a\Delta t_1 \tag{51}$$

This equation holds for $a \le 2V_{\max}/\Delta t_1$. Substituting Eq. (51) into Eq. (49) results in

$$d = \frac{a\Delta t_1^2}{4} \tag{52}$$

Additionally, solving Eq. (51) for $\sin\phi$ and substituting it into the identity $\cos^2\phi = 1 - \sin^2\phi$ yields

$$\cos^2\phi = 1 - \frac{a^2\Delta t_1^2}{4V_{\max}^2} \tag{53}$$

Combining Eqs. (47), (48), (50), (52), and (53) results in

$$\bar{R}_{\text{col}} = \sqrt{R_{\text{col}}^2 - (V_{\max}\Delta t_1)^2 + \frac{a^2\Delta t_1^4}{4}} - \frac{a\Delta t_1^2}{4} \tag{54}$$

To find the closest approach, Eq. (54) is minimized with respect to a. Taking the first and second derivatives yields

$$\frac{\mathrm{d}\bar{R}_{\text{col}}}{\mathrm{d}a} = \left(R_{\text{col}}^2 - (V_{\max}\Delta t_1)^2 + \frac{a^2\Delta t_1^4}{4}\right)^{-1/2}\left(\frac{a\Delta t_1^4}{4}\right) - \frac{\Delta t_1^2}{4} \tag{55}$$

$$\frac{\mathrm{d}^2\bar{R}_{\text{col}}}{\mathrm{d}a^2} = -\left(R_{\text{col}}^2 - (V_{\max}\Delta t_1)^2 + \frac{a^2\Delta t_1^4}{4}\right)^{-3/2}\left(\frac{a\Delta t_1^4}{4}\right)^2 + \left(\frac{\Delta t_1^4}{4}\right)\left(R_{\text{col}}^2 - (V_{\max}\Delta t_1)^2 + \frac{a^2\Delta t_1^4}{4}\right)^{-1/2} \tag{56}$$

Setting Eq. (55) equal to zero establishes Eq. (46), and rearranging Eq. (56) shows that

$$\frac{\mathrm{d}^2\bar{R}_{\text{col}}}{\mathrm{d}a^2} > 0 \quad \text{if } V_{\max} < \frac{R_{\text{col}}}{\Delta t_1}$$

This condition has already been established in Eq. (44). Therefore, a^* minimizes $\bar{R}_{\text{col}}$ so long as $a^* \le \min\{a_{\max}, 2V_{\max}/\Delta t_1\}$. In fact, the minimizing feasible a is the minimum of a^*, $a_{\max}$, and $2V_{\max}/\Delta t_1$. Substituting these three values into Eq. (54) and solving for $\bar{R}_{\text{col}}$ establishes the three conditions in Eq. (45). □

Remark 5 (extending the terminal time): In addition to infeasibility caused by collision-avoidance constraints, infeasibility can also arise due to the constraints on maximum velocity and control magnitudes. Once again, assume that the original optimization described by Problem 2 is feasible. It is possible that the MPC–SCP optimizations are infeasible even when the original problem is feasible. This occurs because the spacecraft have a limited communication radius in the MPC formulation and, therefore, cannot detect collisions occurring after the MPC horizon. This infeasibility is much more likely to occur in situations where the maximum velocity and/or control are achieved in the original problem. Therefore, the swarm reconfiguration should be strictly feasible with respect to the maximum velocity and control constraints when solved using Problem 2. Additionally, $V_{\max}$ is an artificial constraint that was introduced to guarantee collision avoidance. Therefore, it is an optimization parameter rather than a value determined by the problem. 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 4 and 5 and Theorem 6.

These methods will greatly reduce the probability that infeasibility will occur but do not guarantee that it cannot happen. If the optimization is infeasible due to maximum velocity or control constraints, the final time can be extended to make the optimization feasible. This can be done in a few different ways depending on the swarm reconfiguration. First, the final time can be extended by a fraction of an orbit without adjusting the final position. This is desirable if the elapsed time of the reconfiguration is critical. The drawback to this method is that it is not as fuel efficient as other options since most of the spacecraft will have to adjust their trajectories for the new final time. The next option is to extend the final time and adjust the final position to the position on the J_2-invariant orbit [3] where the spacecraft would have been if the optimization was completed as planned. This saves both time and fuel but requires an extra calculation for the updated terminal positions. Finally, the final time can be extended by a full orbit. This option is fuel efficient compared to the first one but requires more time. Any of these options will result in a feasible problem but the mission will dictate which one is the most practical.

Remark 6 (multiple time step sizes): The time step Δt_1 is a critical parameter in Propositions 4 and 5 and Theorem 6. Reducing Δt_1 decreases the required communication radius R_{comm} and collision radius R_{col} as described in Proposition 4 [Eq. (42)] and Theorem 6 [Eq. (45)], 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 5 [Eq. (43)].

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 4 and 5 or Theorem 6. 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.

Additionally, the perturbed trajectory must still satisfy the original collision-avoidance constraint in order to guarantee collision-free initial conditions for the following MPC iteration.

Theorem 7 (resolvability of collision-avoidance constraints): Under Assumptions 1 and 2, the initial constraints and collision-avoidance constraints, Eq. (32) and Eq. (29), respectively, are simultaneously solvable if

$$R_{i,j}[k] \geq \bar{d}_j(k_0, k-k_0) + \bar{d}_i(k_0, k-k_0) + R_{\text{col}} + 2\xi_x \tag{57}$$

where R_{col} satisfies the conditions of Theorem 6.

Proof: The unperturbed trajectories calculated at k_0 satisfy the collision-avoidance constraint [Eq. (29)]. Proposition 1 states that Eq. (29) implies

$$\|C(\boldsymbol{x}_i[k] - \boldsymbol{x}_j[k])\|_2 \geq R_{i,j}[k] \quad \forall\ k = k_0, \ldots, k_0 + T_H \tag{58}$$

Rewriting the left-hand side in terms of the perturbed trajectory and applying Eq. (57) to the right-hand side yields

$$\begin{aligned}&\|C(\hat{\boldsymbol{x}}_i[k] - \hat{\boldsymbol{x}}_j[k])\|_2 + \|C(\boldsymbol{x}_i[k] - \hat{\boldsymbol{x}}_i[k])\|_2 + \|C(\hat{\boldsymbol{x}}_j[k] - \boldsymbol{x}_j[k])\|_2 \\ &\quad \geq \bar{d}_i(k_0, k-k_0) + \bar{d}_j(k_0, k-k_0) + R_{\text{col}} + 2\xi_x \\ &\quad \forall\ k = k_0, \ldots, k_0 + T_H\end{aligned} \tag{59}$$

Applying the results of Proposition 2 and canceling terms results in

$$\|C(\hat{\boldsymbol{x}}_i[k] - \hat{\boldsymbol{x}}_j[k])\|_2 \geq R_{\text{col}} + 2\xi_x \quad \forall\ k = k_0, \ldots, k_0 + T_H \tag{60}$$

Equation (60) states that the actual trajectories never violate the collision-avoidance constraint with R_{col}. However, the initial positions of the following MPC–SCP iteration will have measurement errors. In the worst case, both errors will be in the direction of the other spacecraft. The term $2\xi_x$ ensures that, after the state errors are added, the initial positions of the next MPC–SCP iteration will still maintain a separation of at least R_{col}. Therefore, the MPC–SCP algorithm is resolvable at any time $k \leq k_0 + T_H$. □

An illustration of this condition is shown in Fig. 10.

V. Numerical Simulations

In this section, simulation results 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 Method 1. 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. Each simulation is run using both the SCP algorithm (Method 1) and the MPC–SCP algorithm (Method 2), and the results are compared. The SCP algorithm has a convergence tolerance of $\epsilon = 10^{-3}$, and the simulation parameters are as follows: $V_{\max} = 7.5$ m/s, $U_{\max} = 0.1$ m/s^2, $R_{\text{col}} = 0.125$ km, $R_{\text{safe}} = 0.875$ km, $R_{\text{comm}} = 3$ km, $T_H = 12$, $\Delta t_1 = 15$ s, and $\Delta t_2 = 60$ s. Four simulations, which vary the number of spacecraft and the maximum allowable perturbation, are solved using both methods. In all the simulations, the initial and terminal conditions, $\boldsymbol{x}_{j,0}$ and $\boldsymbol{x}_{j,f}$, respectively, are determined by randomly generating the positions and then applying the J_2-invariant conditions from prior work by Morgan et al. [3] to determine the desired velocities. All of the convex optimizations were performed using CVX [30,31] with the SDPT3 [32,33] solver.

A. Unperturbed Simulations

The simulation results for the 10-spacecraft formation reconfiguration with no actuator or sensor errors are shown in Fig. 11. Figure 11a shows the trajectories when using SCP, and Fig. 11b shows the trajectories resulting from the MPC–SCP algorithm.

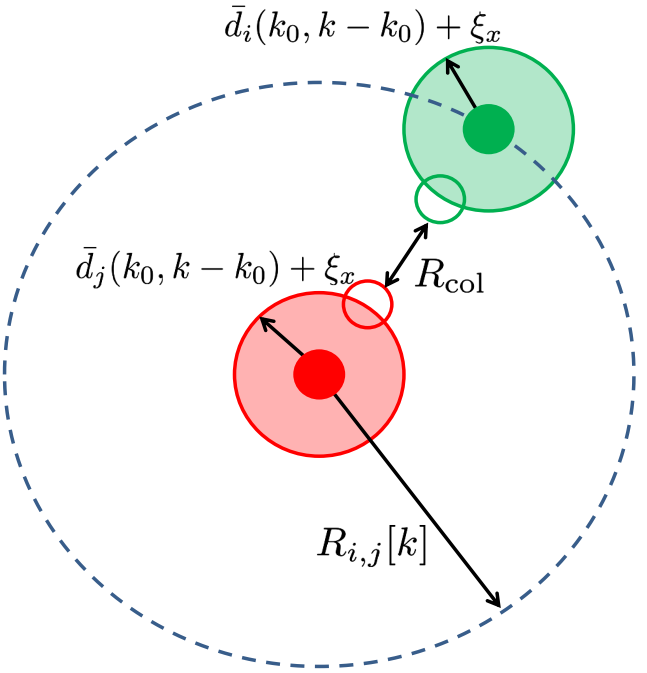


Fig. 10 Illustration of the resolvability condition with sensor and actuator errors.

Without actuator or sensor errors, both sets of trajectories converge to within 1 m of the desired terminal position, with the average terminal error being 0.613 m with SCP and 0.0047 m with MPC–SCP. Figure 11 also shows that both the SCP and MPC–SCP trajectories (squares) follow the optimal trajectory (lines) closely. Additionally, the performances of each of the algorithms for both the 10-spacecraft and 100-spacecraft simulations are shown in Table 1.

Remark 7 (centralized algorithm): As a baseline for the SCP and MPC–SCP algorithms, a centralized approach was used to solve the 10-spacecraft reconfiguration without collision avoidance. The run time for this algorithm was 358 s, which is an order of magnitude longer than the SCP algorithm took to solve the reconfiguration with collision-avoidance constraints, as shown in Table 1. Additionally, including the collision-avoidance constraints prevents the centralized method from finding a feasible solution, and increasing the number of spacecraft to 100 causes the run time of the centralized algorithm to exceed 12 hours.

Table 1 shows the average terminal position error and fuel usage for both the SCP and MPC–SCP trajectories. On average, the terminal error decreases by almost two orders of magnitude when the MPC–SCP algorithm is used instead of the open-loop optimization. Additionally, the computation time of the MPC–SCP algorithm is an order of magnitude better than the SCP algorithm. Since the MPC–SCP algorithm is continuously solving optimizations, the reported algorithm time is the average over every optimization. On the other hand, the fuel usage (ΔV) required to correct for these errors using MPC–SCP is slightly more than the fuel usage of the SCP method. It is important to note that the increase in fuel consumption is largely due to the decentralized communication in the MPC–SCP algorithm. In the MPC–SCP algorithm, spacecraft only consider collisions with spacecraft within the communication range. This can cause aggressive maneuvers to occur because a future collision is detected and must be avoided in a short amount of time. This maneuver requires a large amount of fuel and does not occur in the SCP case because all-to-all communication is considered so all collisions are known at the initial time. Additionally, the MPC–SCP algorithm satisfies the collision-avoidance constraints when nonlinear, continuous dynamics are used to simulate the motion. This is not necessarily true for the SCP case. While the SCP trajectories are collision free, the actual trajectories that result from simulating the open-loop control do not necessarily satisfy the collision-avoidance requirements. Overall, the MPC–SCP algorithm greatly improves the accuracy of the terminal state, decentralizes the communication of the swarm, and guarantees collision avoidance, with the only disadvantage being a small increase in fuel consumption.

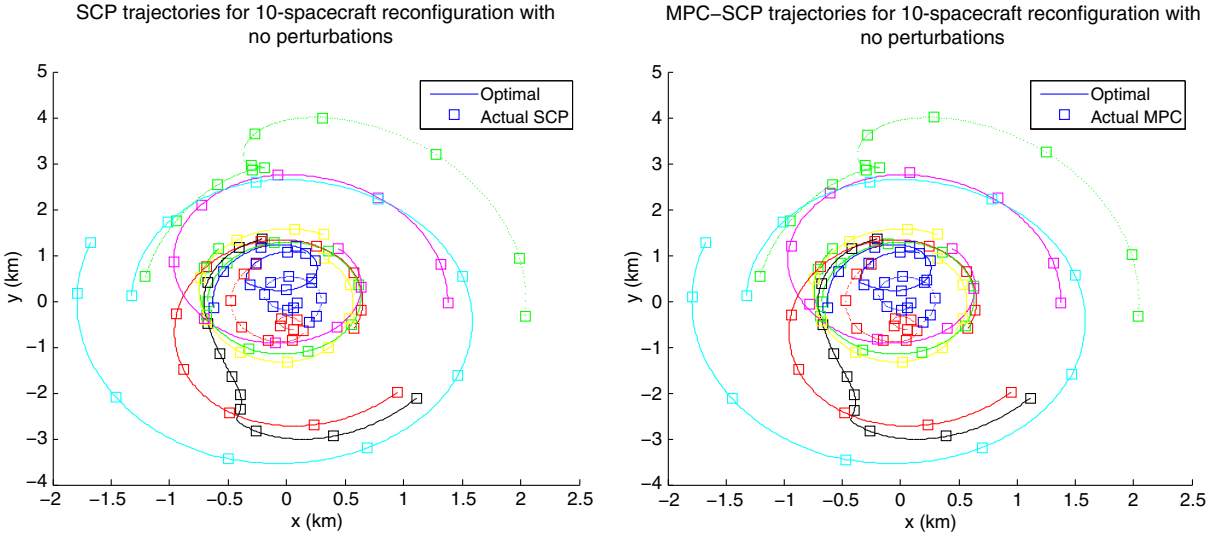
Table 1 Simulation results for swarm reconfigurations using the SCP and MPC-SCP algorithms with no actuator or sensor errors

		Method performance		
Method	No. of spacecraft	Average $\Delta V \times$, m/s	Average terminal error, m	Computation time, s
SCP (Method 1)	10	2.100	0.613	35.03
	100	1.915	5.997	163.585
MPC–SCP (Method 2)	10	2.101	0.0047	3.69
	100	2.356	0.0069	12.58

B. Perturbed Simulations

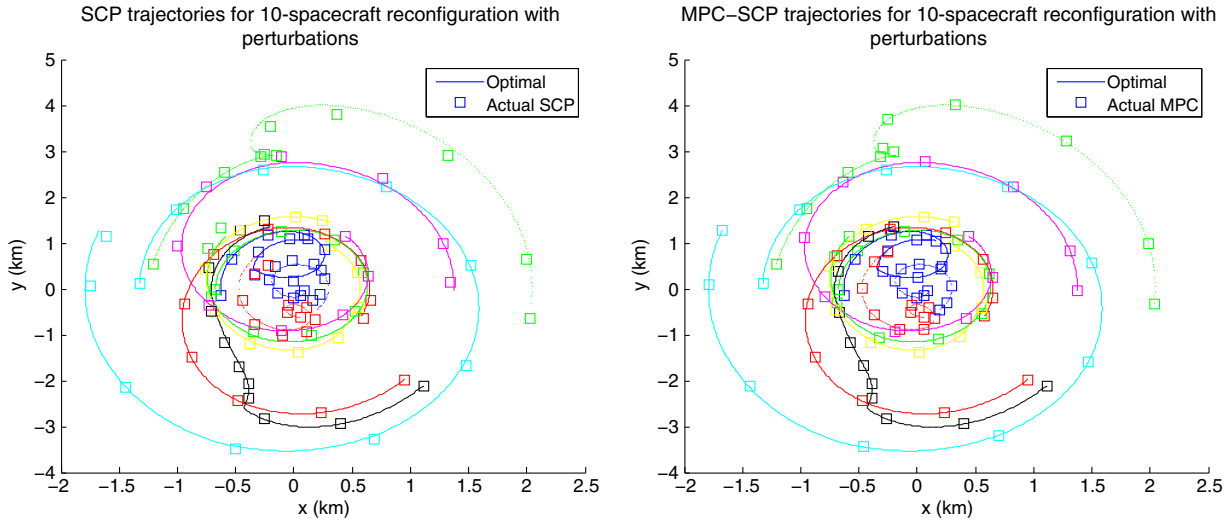
In the following simulations, actuator and sensor errors are included. The maximum allowable errors are $\xi_x = 1$ m, $\xi_v = 7.5$ mm/s, and $\xi_u = 0.1$ mm/s^2. The simulation results for the 10-spacecraft formation reconfiguration are shown in Fig. 12. Figure 12a shows the trajectories when using SCP, and Fig. 12b shows the trajectories resulting from the MPC–SCP algorithm. While the average terminal error has increased due to the inclusion of the actuator and sensor perturbations, the SCP algorithm still has a much larger average terminal error than the MPC–SCP algorithm, with the average terminal error of SCP and MPC–SCP being 163 m and just over 1 m, respectively. The addition of perturbations also causes the SCP trajectory (squares) to deviate from the optimal trajectory (lines), as can be seen in Fig. 12a. In Figure 12b, however, the MPC–SCP trajectories (squares) still closely follow the optimal trajectory closely. Additionally, the performances of each of the algorithms for both the 10-spacecraft and 100-spacecraft simulations are shown in Table 2.

Table 2 shows the simulation results using the SCP and MPC–SCP algorithms when perturbations are included on the actuators and sensors. As in the unperturbed case, the MPC–SCP algorithm greatly reduces the average terminal error when compared to the SCP algorithm. In addition to reducing the terminal error, the MPC–SCP algorithm satisfies the conditions developed in Sec. IV. Therefore, the MPC–SCP algorithm guarantees stability, resolvability, and collision-free trajectories, whereas the SCP algorithm does not have the same guarantees. Similar to the unperturbed case, the fuel required in the MPC–SCP method is larger than in the SCP method due to the decentralized communication and disturbance rejection that occur in the MPC–SCP algorithm. Also, the computation times are of the same order of magnitude as they were in the unperturbed case



a) SCP trajectories **b) MPC-SCP trajectories**

Fig. 11 *x-y* projection of the entire reconfiguration of 10 spacecraft with no perturbations.



a) SCP trajectories **b) MPC-SCP trajectories**

Fig. 12 x-y projection of the reconfiguration of 10 spacecraft with sensor and actuator perturbations.

VI. Conclusions

The optimal swarm guidance problem with collision avoidance was solved using an MPC algorithm with SCP. To use SCP to solve the optimal control problem, the dynamics and collision-avoidance constraints were linearized and the problem was discretized. The resulting problem was a convex optimization, which improved the efficiency at which the problem could be solved. However, the optimization problem was still centralized, meaning it became very large as the number of spacecraft became large. 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's 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. A decentralized SCP algorithm (Method 1) was developed to use SCP to solve the problem.

Once the problem was convexified and decentralized, a receding horizon was introduced and MPC–SCP was applied. Using MPC–SCP decreased the number of variables and constraints in 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 restricted the distances each spacecraft could travel during one optimization. This allowed relaxation of the communication requirements on each spacecraft by considering communication between two spacecraft only if they were within a certain distance from one another.

To ensure that the trajectories resulting from the MPC–SCP optimizations converged to the terminal states, the terminal cost function was converted to a convex optimization problem with a terminal constraint. This constraint ensured that, if the optimization was feasible, it would satisfy the terminal conditions. 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 spacecraft converged to their desired terminal positions and to show that the receding horizon optimizations had a solution. Additionally, a theorem was developed to guarantee that the spacecraft do not collide in between time steps.

These feasibility conditions were then applied to a randomly distributed swarm, and the MPC–SCP algorithm was used to compute the optimal trajectories. These results performed well compared to the trajectories that resulted from solving a single optimization at the initial time. The MPC–SCP algorithm drove the spacecraft to within several millimeters of the desired terminal state, despite the linearization and discretization errors of the optimization. On the other hand, the single optimization trajectory missed the desired terminal state by several meters. Additionally, the time required to run each optimization of the MPC–SCP algorithm was much less than the time required to solve for the entire trajectory at the initial time.

Swarms of spacecraft can be an extremely useful tool for interferometry and distributed sensing, but in order for these missions to become practical, fuel and computationally efficient guidance and control algorithms must be developed. The fuel- and computationally efficient MPC–SCP algorithm developed in this paper is a necessary step toward this goal. Due to the orders-of-magnitude increase in the number of spacecraft and the decrease in the size of the spacecraft, the fuel, communication, and computational requirements become very restrictive. The MPC–SCP algorithm developed in this paper enables swarms of spacecraft to change their formation using minimal fuel and computational resources.

Table 2 Simulation results for swarm reconfigurations using the SCP and MPC-SCP algorithms with actuator or sensor errors

		Method performance		
Method	No. of Spacecraft	Average, $\Delta V \times$ [m/s]	Average terminal error, m	Computation time, s
SCP (Method 1)	10	2.102	163.6	37.36
	100	1.916	166.6	413.29
MPC–SCP (Method 2)	10	2.995	1.067	3.79
	100	2.894	1.013	10.32

Appendix: Dynamics of Chief and Relative Motion

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

$$\boldsymbol{g}(\ell, \text{œ}) = \begin{bmatrix} \eta_j^2 - \omega_z^2 & -\alpha_z & \omega_x\omega_z \\ \alpha_z & \eta_j^2 - \omega_z^2 - \omega_x^2 & -\alpha_x \\ \omega_x\omega_z & \alpha_x & \eta_j^2 - \omega_x^2 \end{bmatrix} \ell + (\zeta_j - \zeta) \begin{pmatrix} \sin i \sin\theta \\ \sin i \cos\theta \\ \cos i \end{pmatrix} + \begin{pmatrix} r(\eta_j^2 - \eta^2) \\ 0 \\ 0 \end{pmatrix} + \mathbf{u}_{\text{chief}} \tag{A1}$$

where

$$\zeta = \frac{2k_{J2}\sin i \sin\theta}{r^4} \qquad \zeta_j = \frac{2k_{J2}r_{jZ}}{r_j^5}$$

$$\eta^2 = \frac{\mu}{r^3} + \frac{k_{J2}}{r^5} - \frac{5k_{J2}\sin^2 i \sin^2\theta}{r^5} \qquad \eta_j^2 = \frac{\mu}{r_j^3} + \frac{k_{J2}}{r_j^5} - \frac{5k_{J2}r_{jZ}^2}{r_j^7}$$

$$r_j = \sqrt{(r+x_j)^2 + y_j^2 + z_j^2}$$

$$r_{jZ} = (r+x_j)\sin i \sin\theta + y_j \sin i \cos\theta + z_j \cos i$$

$$\omega_x = -\frac{k_{J2}\sin 2i \sin\theta}{hr^3} + \frac{r}{h}u_N \qquad \omega_z = \frac{h}{r^2} \qquad \alpha_x = \dot{\omega}_x \qquad \alpha_z = \dot{\omega}_z \tag{A2}$$

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

$$\dot{r} = v_x \qquad \dot{v}_x = -\frac{\mu}{r^2} + \frac{h^2}{r^3} - \frac{k_{J2}}{r^4}(1 - 3\sin^2 i \sin^2\theta) + u_R$$

$$\dot{h} = -\frac{k_{J2}\sin^2 i \sin 2\theta}{r^3} + (r)u_T$$

$$\dot{\Omega} = -\frac{2k_{J2}\cos i \sin^2\theta}{hr^3} + \left(\frac{r\sin\theta}{h\sin i}\right)u_N$$

$$\dot{i} = -\frac{k_{J2}\sin 2i \sin 2\theta}{2\,hr^3} + \left(\frac{r\cos\theta}{h}\right)u_N$$

$$\dot{\theta} = \frac{h}{r^2} + \frac{2k_{J2}\cos^2 i \sin^2\theta}{hr^3} - \left(\frac{r\sin\theta\cos i}{h\sin i}\right)u_N \tag{A3}$$

where $\boldsymbol{u}_{\text{chief}} = (u_R, u_T, u_N)^T$ is the chief control in the radial, tangential, and normal directions, respectively.

Acknowledgments

This work was supported by a NASA Office of the Chief Technologist Space Technology Research Fellowship. Government sponsorship is acknowledged. This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the NASA.

References

[1] Hadaegh, F. Y., Chung, S.-J., and Manaroha, H. M., "On Development of 100-Gram-Class Spacecraft for Swarm Applications," *IEEE Systems Journal* (accepted for publication), available online at http://arcl.ae.illinois.edu/FemtoSat_Swarm_Final2013.pdf [retrieved 2014].

[2] 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 and Technologies* [CD-ROM], National Research Council Canada, Ottawa, May 2011.

[3] 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*, Vol. 35, No. 5, 2012, pp. 1492–1506. doi:10.2514/1.55705

[4] 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*, Evanston, IL, June 2003, pp. 1733–1739.

[5] 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*, Evanston, IL, June 2004, pp. 2976–2984.

[6] Alfriend, K. T., Vadali, S. R., Gurfil, P., How, J. P., and Breger, L., *Spacecraft Formation Flying: Dynamics, Control and Navigation*, Elsevier, Oxford, 2009, pp. 143–222.

[7] D'Amico, S., and Montenbruck, O., "Proximity Operations of Formation-Flying Spacecraft Using an Eccentricity/Inclination Vector Separation," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 3, 2006, pp. 554–563. doi:10.2514/1.15114

[8] Breger, L., and How, J. P., "Gauss's Variational Equation-Based Dynamics and Control for Formation Flying Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 437–448. doi:10.2514/1.22649

[9] Vaddi, S. S., Alfriend, K. T., Vadali, S. R., and Sengupta, P., "Formation Establishment and Reconfiguration Using Impulsive Control," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 2, 2005, pp. 262–268. doi:10.2514/1.6687

[10] Campbell, M. E., "Planning Algorithm for Multiple Satellites Clusters," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 5, 2003, pp. 770–780.

[11] Campbell, M. E., "Collision Monitoring within Satellite Clusters," *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 1, 2005, pp. 42–55.

[12] Zanon, D. J., and Campbell, M. E., "Optimal Planner for Spacecraft Formations in Elliptical Orbits," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 1, 2006, pp. 161–171.

[13] Murray, R. M., "Recent Research in Cooperative Control of Multivehicle Systems," *Journal of Dynamic Systems, Measurement and Control*, Vol. 129, No. 5, 2007, pp. 571–583. doi:10.1115/1.2766721

[14] 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. doi:10.1109/TAC.2003.812781

[15] 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. doi:10.1109/TRO.2005.853499

[16] Chung, S.-J., Bandyopadhyay, S., Chang, I., and Hadaegh, F. Y., "Phase Synchronization Control of Complex Networks of Lagrangian Systems on Adaptive Digraphs," *Automatica*, Vol. 49, No. 5, 2013, pp. 1148–1161. doi:10.1016/j.automatica.2013.01.048

[17] 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. doi:10.1145/179812.179911

[18] 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 and Technologies*, National Research Council Canada, Ottawa, May 2011.

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

[20] Conway, B. A., *Spacecraft Trajectory Optimization*, Cambridge Univ. Press, New York, 2010, pp. 16–36.

[21] Ross, I. M., and Fahroo, F., "Legendre Pseudospectral Approximations of Optimal Control Problems," *Lecture Notes in Control and Information Systems*, Vol. 295, 2003, pp. 327–342. doi:10.1007/978-3-540-45056-6_21

[22] Richards, A., Kuwata, Y., and How, J., "Experimental Demonstration of Real-Time MILP Control," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2003-5802, 2003.

[23] 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*, AIAA Paper 2008-7132, 2008.

[24] 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*, IEEE, Piscataway, NJ, 2008, pp. 540–545.

[25] Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge Univ. Press, Cambridge, England, U.K., 2004, pp. 21–189.

[26] 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.

[27] 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*, AIAA Paper 2006-6070, 2006.

[28] Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., and Abbeel, P., "Finding Locally Optimal, Collision-Free Trajectories with Sequential

Convex Optimization," *Robotics: Science and Systems*, Vol. 9, No. 1, June 2013, pp. 1–10.

[29] Byrd, R. H., Gilbert, J. C., and Nocedal, J., "A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming," *Mathematical Programming*, Vol. 89, No. 1, 2000, pp. 149–185. doi:10.1007/PL00011391

[30] Grant, M., and Boyd, S., "CVX: Matlab Software for Disciplined Convex Programming, Ver. 2.0 Beta," Sept. 2012, available online at http://cvxr.com/cvx [retrieved 28 June 2013].

[31] Grant, M., and Boyd, S., "Graph Implementations for Nonsmooth Convex Programs," *Recent Advances in Learning and Control*, edited by Blondel, V., Boyd, S., and Kimura, H., Lecture Notes in Control and Information Sciences, Springer–Verlag, New York, 2008, pp. 95–110, http://stanford.edu/~boyd/papers/pdf/graph_dcp.html [retrieved 2014].

[32] Toh, K. C., Todd, M. J., and Tutuncu, R. H., "SDPT3: A Matlab Software Package for Semidefinite Programming," *Optimization Methods and Software*, Vol. 11, Nos. 1–4, 1999, pp. 545–581. doi:10.1080/10556789908805762

[33] Tutuncu, R. H., Toh, K. C., and Todd, M. J., "Solving semidefinite-quadratic-linear programs using SDPT3," *Mathematical Programming Ser. B*, Vol. 95, No. 2, 2003, pp. 189–217.

[34] Anderson, E. D., and Anderson, K. D., "MOSEK: High Performance Software for Large-Scale LP, QP, SOCP, SDP and MIP Including Interfaces to C, Java, MATLAB,.NET and Python," MOSEK, Copenhagen, 2012, http://www.mosek.com/ [retrieved 2014].

[35] Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Spacecraft Swarm Guidance Using a Sequence of Decentralized Convex Optimizations," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA Paper 2012-4583, 2012.

[36] Bemporad, A., and Morari, M., "Robust Model Predictive Control: A Survey," *Robustness in Identification and Control*, Vol. 245, 1999, pp. 207–226. doi:10.1007/BFb0109870

[37] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M., "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, Vol. 36, No. 6, 2000, pp. 789–814. doi:10.1016/S0005-1098(99)00214-9

[38] Richards, A., and How, J. P., "Robust Variable Horizon Model Predictive Control for Vehicle Maneuvering," *International Journal of Robust and Nonlinear Control*, Vol. 16, No. 7, 2006, pp. 333–351. doi:10.1002/(ISSN)1099-1239

[39] Acikmese, B., Carson, J. M., and Bayard, D. S., "A Robust Model Predictive Control Algorithm for Incrementally Conic Uncertain/Nonlinear Systems," *International Journal of Robust and Nonlinear Control*, Vol. 21, No. 5, 2011, pp. 563–590. doi:10.1002/rnc.1613

[40] Richards, A., and How, J., "Robust Model Predictive Control with Imperfect Information," *American Control Conference*, IEEE, Portland, OR, June 2005, pp. 268–273.

[41] Wang, Y., and Boyd, S., "Fast Model Predictive Control Using Online Optimization," *IEEE Transactions on Control Systems Technology*, Vol. 18, No. 2, 2010, pp. 267–278. doi:10.1109/TCST.2009.2017934

[42] Canale, M., Cerone, V., Piga, D., and Regruto, D., "Fast Implementation of Model Predictive Control with Guaranteed Performance," *IEEE Conference on Decision and Control*, IEEE, Piscataway, NJ, Dec. 2011, pp. 3375–3380.

[43] Keviczky, T., Borrelli, F., Fregene, K., Godbole, D., and Balas, G. J., "Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations," *IEEE Transactions on Control Systems Technology*, Vol. 16, No. 1, 2008, pp. 19–33. doi:10.1109/TCST.2007.903066

[44] Carson, J. M., and Acikmese, B., "A Model-Predictive Control Technique with Guaranteed Resolvability and Required Thruster Silent Times for Small-Body Proximity Operations," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2006-6780, 2006.

[45] 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, 2008, pp. 1521–1524. doi:10.2514/1.33616

[46] Ross, I. M., "Space Trajectory Optimization and L^1-Optimal Control Problems," *Modern Astrodynamics*, Butterworth-Heinemann, New York, 2006, pp. 155–188.