Operating Small Sat Swarms as a Single Entity: Introducing SODA

Tracie Conn, Laura Plice, Andres Dono Perez, Michael Ho
NASA Ames Research Center / Mission Design Division
Moffett Field, CA; (650) 604-5365
tracie.conn@nasa.gov

Abstract

NASA's decadal survey determined that simultaneous measurements from a 3D volume of space are advantageous for a variety of studies in space physics and Earth science. Therefore, swarm concepts with multiple spacecraft in close proximity are a growing topic of interest in the small satellite community. Among the capabilities needed for swarm missions is a means to maintain operator-specified geometry, alignment, or separation. Swarm stationkeeping poses a planning challenge due to the limited scalability of ground resources. To address scalable control of orbital dynamics, we introduce SODA – Swarm Orbital Dynamics Advisor – a tool that accepts high-level configuration commands and provides the orbital maneuvers needed to achieve the desired type of swarm relative motion. Rather than conventional path planning, SODA's innovation is the use of artificial potential functions to define boundaries and keepout regions. The software architecture includes high fidelity propagation, accommodates manual or automated inputs, displays motion animations, and returns maneuver commands and analytical results. Currently, two swarm types are enabled: in-train distribution and an ellipsoid volume container. Additional swarm types, simulation applications, and orbital destinations are in planning stages.

Nomenclature

$\bar{x}_i$	3 x 1 position vector of satellite i in the inertial reference frame
$\bar{x}_t$	3 x 1 position vector of target location in the inertial reference frame
$\bar{r}_i$	3 x 1 position vector of satellite i in the LVLH frame
$\bar{\rho}_{j,i}$	3 x 1 position vector of satellite j with respect to satellite i, i.e. $\bar{x}_j - \bar{x}_i$
ϕ	Potential function, scalar-valued
a, r	Subscripts on ϕ denoting attractive or repulsive potential functions
κ	Selectable scaling factor for the magnitude of the impulsive maneuver
P	3×3 positive-definite matrix that describes the shape of the attractive potential
A, B	Scalars that may be tuned to yield desired repulsive potential functions
APF	Artificial potential function

Introduction and Motivation

In 2016, the Space Studies Board of the National Academies of Sciences, Engineering, and Medicine states that *satellite swarm missions are of high priority for multiple disciplines and deserve focused investment and development*.[1] As defined by the Board, a swarm comprises multiple satellites flying in formation near one another in similar orbits. More specifically, we envision swarms to have capabilities for cross-link communication and station-keeping. Of particular interest to the scientific community is the ability of a satellite swarm to achieve and maintain a specified geometry, alignment, or separation.

The focus of this paper is how to control inter-satellite relative motion to achieve the objectives of the swarm as a whole. We introduce SODA, Swarm Orbital Dynamics Advisor, a tool that provides the orbital maneuvers required to achieve a desired type of relative swarm motion. SODA is under development in the Mission Design Division at NASA Ames Research Center to enable new science return possibilities for future spacecraft swarm mission architectures.

Swarms of large numbers of cooperating satellites will introduce new space mission capabilities and complexities. The differences from conventional missions will be manifold, spanning science goals, instrument design, concept of operations, spacecraft capabilities, and inter-

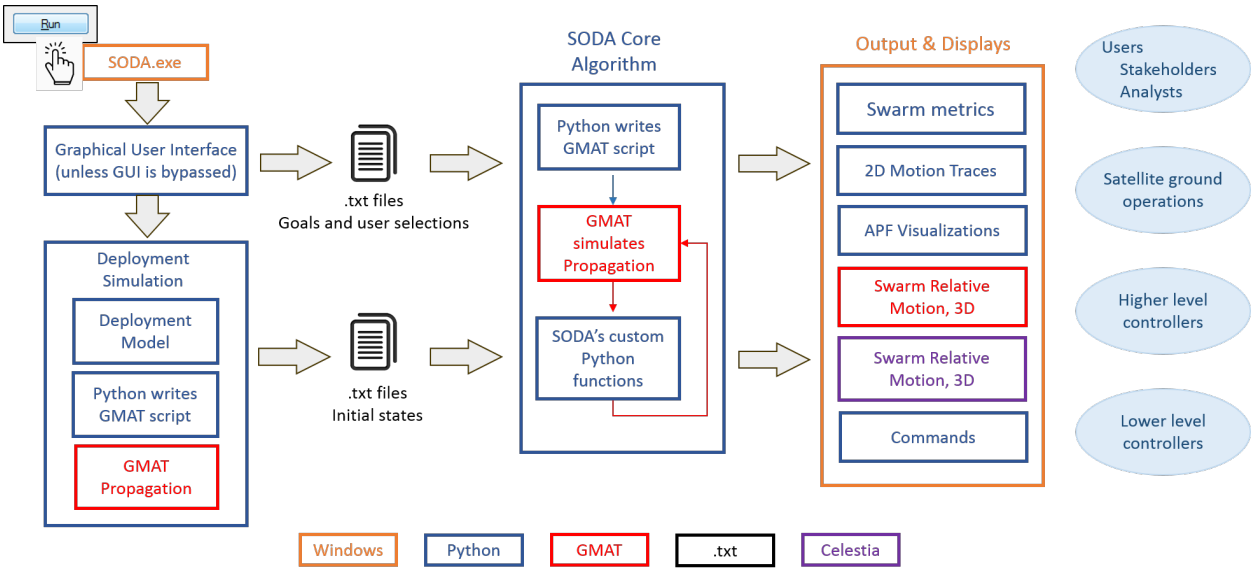

Figure 1: The different functions of SODA are completed via GUI scripts, custom Python functions, and the applications GMAT and Celestia.

satellite cooperation. From a mission operations perspective, swarms pose a planning challenge due to the limited scalability of ground operations. The approach of planning and commanding individual satellites simply does not scale for multi-sat swarms of tens or hundreds. If the current state-of-practice continues to be applied, operation of large swarms (e.g. 100 spacecraft or more) will become intractable and cost prohibitive.

To avoid this operations bottleneck, a new approach is required: the swarm must operate as a unit, responding to high level commands and constraints. SODA enables high level user inputs in a single planning cycle. From one high level command, SODA determines all of the required individual satellite maneuvers over time, relieving ground personnel of the tasks of designing and commanding the placement of the swarm members. SODA accomplishes this by applying the most appropriate algorithm, which can be a basic Hohmann transfer or a more complex, nonlinear control routine.

Prior formation flying studies used guidance and control algorithms for very particular mission concepts, such as assigning individual spacecraft to a target location and guiding each to its destination via artificial potential functions or explicit solutions to Lambert's problem.[2, 3] While SODA builds upon these concepts, the goal of the tool is somewhat more general. We have identified several swarm types that are particularly interesting in terms of science data return; subsequent sections provide details. A user may choose from one of the available types, and the algorithms in SODA handle how to achieve and maintain the chosen configuration. The resulting products include tabular listings of required Δv maneuvers, plots, animations, and numerical metrics describing how well the swarm type was maintained in the simulation.

SODA is early in its development and its current state emphasizes interactive use. Subsequent sections describe each of the components of SODA in detail; the high level design of SODA is as follows. First, the tool prompts the user to specify initial and final conditions via a GUI interface. Initial state vectors (Keplerian or Cartesian) may be provided as a model of estimations for spacecraft already in orbit, or the user may elect for SODA to simulate a swarm deployment from a specified orbit. Depending on the swarm type selected, the user will be prompted for the specific parameters that define that swarm type. Alternatively, the user may bypass the GUI altogether and provide inputs via text files. From the user's inputs, SODA generates a custom script to produce a high-fidelity simulation to be performed by the General Mission Analysis Tool (GMAT).[4] During the simulation, GMAT links to Python scripts that have been specifically written for SODA to enable the desired swarm types. After the simulation completes, the user receives results in the form of animation, plots, and statistics. Figure 1 illustrates SODA's overall structure. Note that SODA's dependencies are completely free and open-source, with the exception of the Windows operating system itself. These dependencies include: Python (with `numpy`, `tkinter`, and `matplotlib` libraries), GMAT, and Celestia. Our long-term goal is to implement several more swarm types in SODA and, ultimately, to make the tool available open-source.

Specifying the Swarm Mission

SODA's architecture design accommodates several options for the source of inputs: graphical user interface (GUI) for small analyses or developing stakeholder awareness, text files for "hands-off" applications such as Monte Carlo analyses, and a defined software interface for future integration with higher- and lower-level controllers.

Mission descriptions include initial and target conditions, the mission orbit, number of spacecraft and their characteristics, and simulation parameters. This section summarizes the mission design inputs and implementation details appear in following sections.

When a swarm mission begins, the member spacecraft will have initial orbital conditions, either from ground estimates of existing on-orbit assets, or from a dedicated deployment phase of mission ops. SODA's core algorithms receive state vectors for the swarm members uniformly for all input sources and a powerful feature for users is the ability to simulate initial states resulting from the deployment impulse.

The target conditions for the swarm entail descriptions of the desired orbit and the desired swarm configuration. While SODA's planned development encompasses multiple orbital and non-Keplerian regimes, Low Earth Orbit (LEO) serves as a valuable, stressing case for the tool and is commonly used for small sat missions. Early test cases have revealed an important subtlety in specifying the target orbit in LEO: the use of orbital elements excludes perturbation effects. Orbit maintenance and swarm maintenance may be combined as drivers on SODA's control algorithms, however to isolate the swarm maintenance approach under study, it is necessary to identify target orbital motion that includes gravitational perturbations and atmospheric decay. One approach for accommodating realistic target motion is to define a pseudo-satellite whose trajectory will be a reference point for the swarm.

SODA categorizes swarms by type, with "In Train" and "Ellipsoid Container" discussed below. Examples of additional planned types include "Ellipsoid with Distribution," and some commonly referenced formations. Users choose the target swarm type, orientation, and dimensions. Maneuvers are constrained to allowable magnitude and frequency.

The spacecraft participating in the SODA swarm are currently all identical. Specialized swarm elements will be part of future implementations. Specifications of most relevance to orbital motion are propulsion subsystem capabilities and ballistic coefficient. SODA currently offers users several CubeSat options, from 1U to 6U. Attitude control capabilities are very important for mitigating differential drag among swarm members in LEO. While SODA is not a mission simulator with high fidelity attitude models, it does offer a useful approximation of differential drag effects with an option to apply random drag areas from Gaussian distribution with user settings for mean and standard deviation.

SODA's calculations rely on accurate knowledge of swarm member positions and velocities. For the first release, the model excludes issues of real-world communication and navigation. Future versions will allow users to input restrictions on state knowledge.

Using the SODA GUI

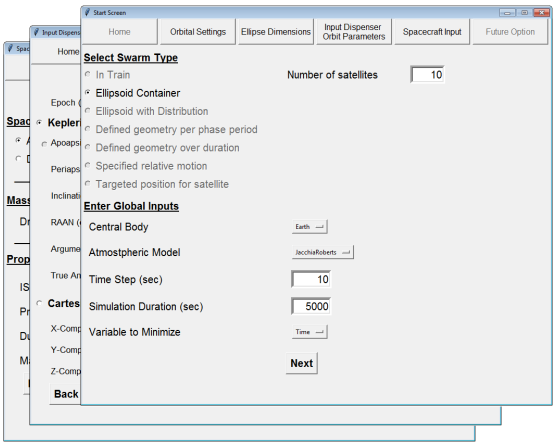

Figure 3: The first SODA GUI window is shown here. Subsequent windows contain inputs for the high level parameters required to simulate a spacecraft swarm.

Unless it is bypassed via the command line, the SODA GUI is the first part of SODA application. The GUI prompts the user for inputs required for SODA's use cases and dynamically generates files needed for subsequent modules of the program. Each window contains a category of inputs, with buttons and tabs to provide navigation through the screens (Figure 3). As a user progresses through the GUI, logical checks ensure valid inputs, such as realistic orbital elements. GUI functionality concludes with the creation of prerequisite files needed for other programs in the SODA application.

Because of a Python version dependency in GMAT, SODA's GUI is written in Python version 3.4.3. We use the `tkinter` module that comes prepackaged with Python. The GUI consists of classes, including a parent class that initializes and stacks multiple components of the GUI.

SODA's early development has followed a rapid prototyping approach, with requirements readily defined, prioritized and implemented. Functionality benefits from early user feedback. Future efforts will continue to ensure that the code structure is flexible, expandable, and manageable.

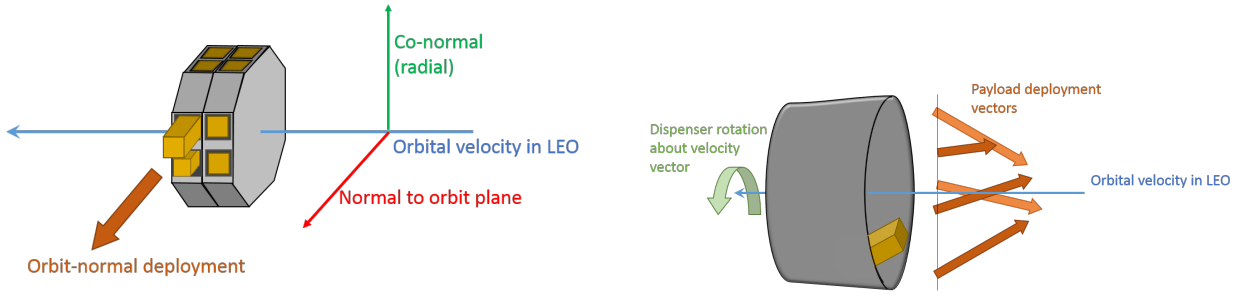

Figure 2: (Left) In the example depicted, small satellites are deployed in the orbit-normal direction. Deployments in the velocity or co-normal direction are also simulation options. (Right) Satellites are deployed from a rotating vehicle

Bypassing the SODA GUI

In the Windows command line the user can bypass the GUI and SODA will look for all necessary parameters in user-provided text files. Executing SODA via scripts enables Monte Carlo style analyses, a feature that is especially useful if the user needs to define unique initial states and propulsion system properties for tens or hundreds of satellites.

Deployment Simulators

Rather than specifying the initial state of each individual satellite, a user may rely on SODA to perform a simulation of a small satellite deployment. Currently, two deployment simulators are available in SODA. We describe them as "VNC Directional" (velocity, normal, co-normal) and "Rotating" and illustrate the concepts inFigure 2.

For both deployment simulations, the orbit of the dispenser is specified by the user. There is a five second delay between the individual deployments. To simulate the imperfect nature of small satellite deployments, random errors of up to 10% create a distribution for the deployment spring force. The VNC Directional deployer simulation is for deployments along one of the three axes of the deploying vehicle's VNC reference frame. Individual deployment vectors have pointing errors of up to 0.5º. The Rotating deployer simulation models CubeSat deployments from a rotating vehicle. The deployment vector includes components from the spring force and a tangential component due to the dispenser's rotation.

Propagation via GMAT

To model the individual spacecraft trajectories, we rely on GMAT, an open-source mission analysis tool developed by a team of NASA, private industry, public, and private contributors.[4] Each time SODA runs, user inputs define a custom GMAT script. User creation of a simulation script in GMAT's GUI is also possible, but SODA takes care of this automatically and avoids what could become a tedious task to configure a swarm of a large number of satellites. GMAT propagates the motion of the satellites for the specified time step, and then passes all state data to the custom Python function written for that swarm type. If Δv maneuvers are required for this epoch, GMAT implements them as impulsive burns. Fuel depletion modeling by GMAT reflects propulsion system parameters given by the user (I_{sp}, max thrust magnitude, and duty cycle). GMAT then propagates forward another time step, using a high fidelity model that includes perturbations due to atmospheric drag, solar radiation pressure, and J2 effects. At the next epoch, the calls back and forth to the Python function repeat. The entire propagation process continues for the specified duration of the simulation. GMAT outputs data of interest for each satellite, such as state vectors, fuel use, and maneuvers, to text files. Output files support subsequent scripts that generate final products. GMAT also displays an animation of the swarm motion.

Swarm Types

Missions have unique requirements for science data return, and swarm types will support different science objectives. To support swarm mission design, SODA currently has two swarm types enabled, each with its own control logic. Additional swarm types are in development.

Swarm Type: In-Train Distribution

In this swarm type, the objective is to phase the satellites ahead and behind each other to achieve an in-track, or string-of-pearls, relative position configuration. SODA maneuvers each satellite by performing a two-impulse elliptical transfer orbit from and back to the same orbit, known as a phasing maneuver.

If the spacecraft's relative position is trailing the target position, then the phasing orbital period must be less

than that of the current orbit. A retrofire is required; the spacecraft must slow down to speed up. The retrograde Δv occurs at apoapsis of the phasing orbit. On the other hand, if the target is "behind" or trailing in the along-track direction, the phasing orbit must have an orbital period greater than that of the current orbit. A forward fire thruster is required to boost the spacecraft's velocity. The prograde Δv occurs at periapsis of the phasing orbit. We illustrate two possible phasing orbits relative to a circular baseline in Figure 4, where both Δv maneuvers would occur at point P.

If the baseline orbit is not circular, then the resulting in-train formation will have relative motion in the radial and along-track directions. The residual relative motion is due to the speed of the satellites changing as they move around the elliptical orbit. As shown in Figure 5, the magnitude of the resulting relative motion will be a function of the in-train separation distances; satellites that maneuver further ahead or behind will have greater relative motion than those spacecraft closest to the point of reference.

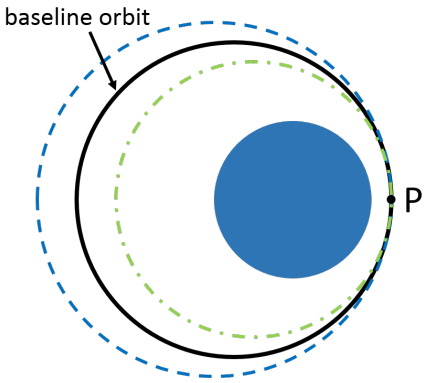

Figure 4: A baseline orbit is shown with two potential phasing orbits: a smaller semimajor axis (green dash-dot) to results in moving forward in-track, and a larger semimajor axis (blue dashed) to regress in relative true anomaly.

Swarm Type: Ellipsoid Container

The purpose of the ellipsoid container swarm type is to maneuver the satellites to within a defined ellipsoid, the center of which is on its own specified (mathematical) orbit, shown conceptually in Figure 6. Within this volume, the satellites may wander, but not escape.

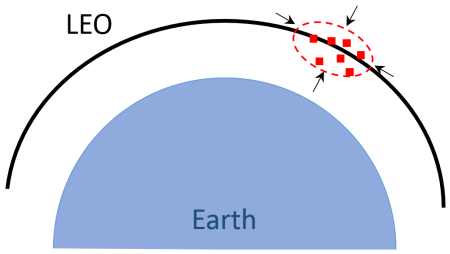

Figure 6: For this swarm type, prescribed maneuvers constrain satellite motion to within an ellipsoid volume, as illustrated in red.

The parameters a, b, and c specify the dimensions of the ellipsoid container, as in:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1 \tag{1}$$

where a, b, and c define the ellipsoid dimensions in the x-, y-, and z-axis directions of the local-vertical, local-horizontal frame with origin at the center of the ellipsoid container. In other words, for a local frame with its origin on the user-specified orbit, a would be the dimension in the radial direction, c would be the dimension in the orbit normal direction, and y would be in the direction of the unit vector that completes the triad.

For the ellipsoid container swarm type, we rely on artificial potential functions (APFs), a method for autonomous spacecraft control receiving extensive study in the past two decades.[2, 5, 6, 7] APFs provide the maneuvers that guide the spacecraft to the ellipsoid volume, constrain their motion to within this volume, and prevent collisions. To accomplish simultaneous attraction and repulsion, we define the global potential function such that the negative gradient of the potential leads to the desired target area.

For the APF method described in this section to be feasible for autonomous control, there is a non-trivial requirement that each satellite has knowledge of the estimated states of all other spacecraft in the swarm at a given instant of query. The relative position of each satellite in the swarm is simply:

$$\bar{r}_i=\bar{x}_i-\bar{x}_t \tag{2}$$

where $\bar{x}_i$ is the position vector of satellite i and $\bar{x}_t$ is the position vector of the ellipsoid center in the inertial reference frame. We take the same approach as others,[2, 6, 7] and define the general form of the attractive potential function for satellite i to be:

$$\phi_{a,i}=\frac{1}{2}\bar{r}_i^T P\bar{r}_i \tag{3}$$

where $\phi_{a,i}$ is scalar-valued. Taking the time derivative Equation (3) gives:

$$\dot{\phi}_{a,i}=(P\bar{r}_i)^T\dot{\bar{r}}_i \tag{4}$$

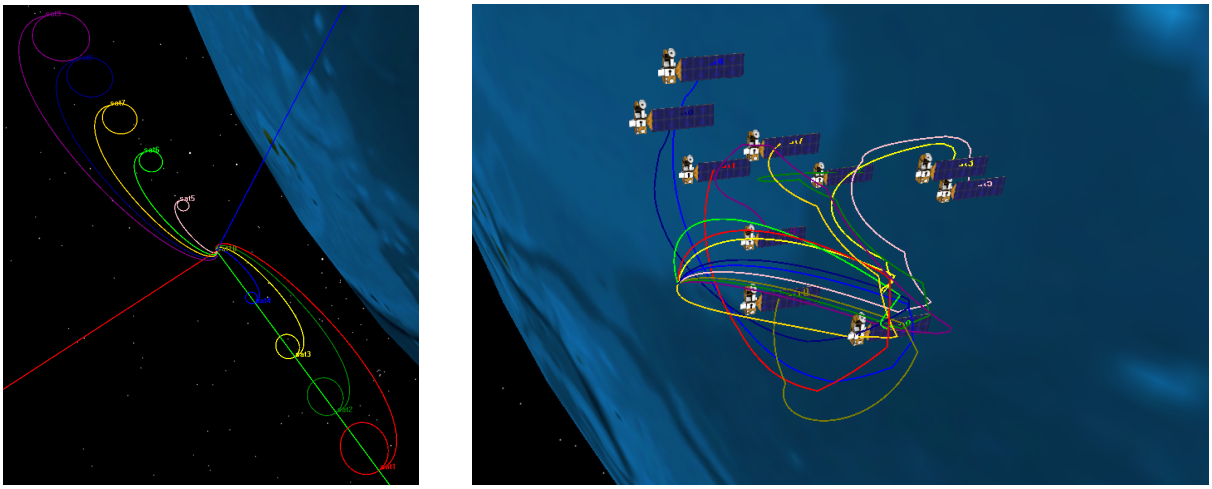

Figure 5: (Left) In-train distribution satellites. (Right) Snapshot of a ten satellite swarm simulation. An animation of this swarm type is available online.[?]

Because the purpose of this swarm type is to constrain the satellites within the ellipsoid volume, we define:

$$\phi_{a,i} = \dot{\phi}_{a,i} = 0 \tag{5}$$

if satellite i is is within the ellipsoid perimeter.

The repulsive potential is a function of the distances between each pair of satellites. We apply the obstacle-avoiding approach described by McQuade and McInnes and use a Gaussian function.[6] For a swarm of n satellites, we define the total repulsive potential function for satellite i to be:

$$\phi_{r,i} = \sum_{j=1}^{n,j\neq i} Ae^{-B\left(\overline{\rho}_{j,i}^T P \overline{\rho}_{j,i}\right)} \tag{6}$$

where $\overline{\rho}_{j,i}$ is the position of satellite j with respect to satellite i. Taking the time derivative of Equation (6) gives:

$$\dot{\phi}_{r,i} = \sum_{j=1}^{n,j\neq i} -4\mathrm{AB}\left(\overline{\rho}_{j,i}^T P \dot{\overline{\rho}}_{j,i}\right) e^{-B\left(\overline{\rho}_{j,i}^T P \overline{\rho}_{j,i}\right)} \tag{7}$$

The global potential for spacecraft i is the sum of the attractive and repulsive potential functions:

$$\phi_i = \phi_{a,i} + \phi_{r,i} \tag{8}$$

SODA queries the states of all satellites in the swarm at a user-specified frequency. The algorithm first checks a wait condition, verifying that enough time has elapsed since the last impulsive maneuver. For example, a particular spacecraft design may require a duty cycle of 30 seconds. If either the wait condition is unsatisfied or the time derivative of the potential is negative, no maneuvers are performed.

If the wait condition is satisfied and $\dot{\phi}_i \geq 0$, the gradient for spacecraft i is found as:

$$\nabla\phi = \nabla\phi_{a,i} + \nabla\phi_{r,i} \tag{9}$$

$$\nabla\phi = P\bar{r}_i + \sum_{j=1}^{n,j\neq i} -2ABP\overline{\rho}_{j,i} e^{-B\left(\overline{\rho}_{j,i}^T P \overline{\rho}_{j,i}\right)} \tag{10}$$

The next step calculates a scaling factor for the magnitude of the impulsive maneuver:[6]

$$\kappa = v_{max,i}\left(1 - e^{-\lambda\phi_i}\right) \tag{11}$$

where λ is a constant and $v_{max,i}$ is the maximum possible Δv magnitude achievable by the satellite. Next, the desired relative velocity vector for satellite i is calculated:

$$\dot{\bar{r}}_i = -\kappa\frac{\nabla\phi}{|\nabla\phi|} \tag{12}$$

Finally, the relative velocity change given by Equation (12) executes via an impulsive maneuver.

There are several parameters in the above equations that may be either tuned to constants or defined as time-varying functions: A, B, λ, and P. Choosing these parameters is a swarm design choice, as different values can yield very different maneuvers and thus impact fuel use and mission life. For example, Figure 7 illustrates how the total potential field is represented for different values of A and B.

A wider base and steeper curve for the repulsive function creates a more conservative "keep out" zone, but may unnecessarily cause more maneuvers as the satellites move within the ellipsoid volume. Monte Carlo analysis is an option for choosing the optimal values of A, B, λ,

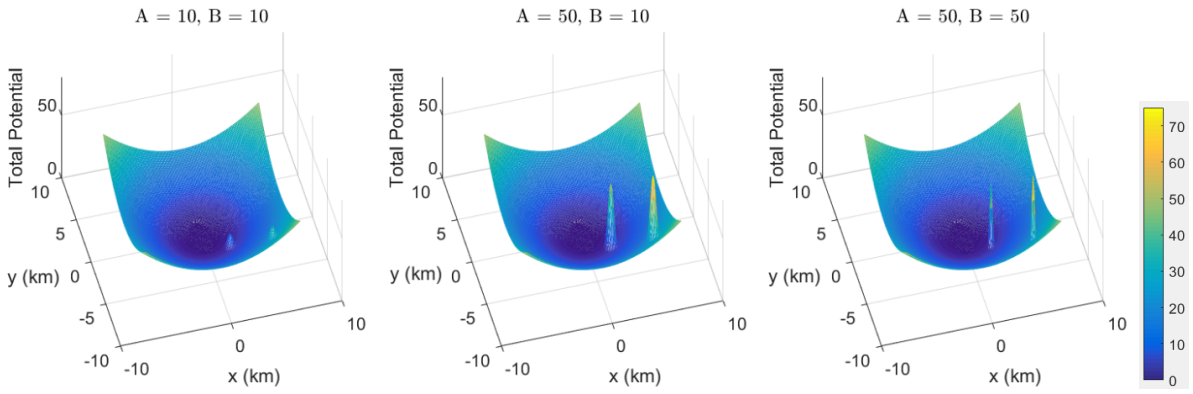

Figure 7: Three examples of unique choices for the repulsive function's A and B parameters. An animation illustrating how the peaks of the repulsive potential function vary with A and B is available online.[?]

and P for a specific mission concept. Currently SODA defaults to: $A = B = 50$, $\lambda = 1$, and $P = [I]_{3x3}$. In Figure 8, we illustrate a representative APF calculated by SODA at the start of a simulation.

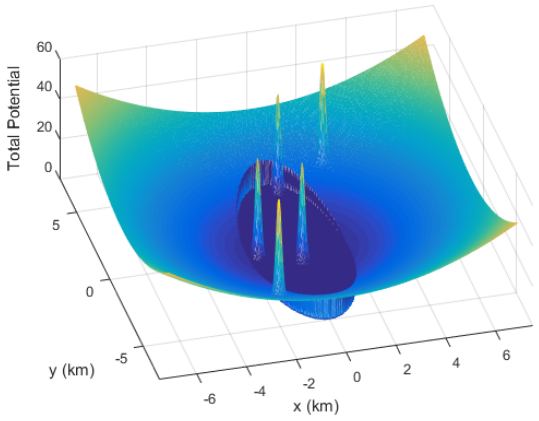

Figure 8: Snapshot of the APF field for a five satellite swarm in an equatorial, circular LEO.

Products and Results

SODA creates output products which serve its uses as a development environment for algorithms to control swarm orbital dynamics, simulation support for swarm technologies, and enhancing stakeholder awareness of swarm mission concepts and issues.

Among the most powerful display capabilities are 3D animations of the trajectories of the satellites using GMAT; a screen capture of the In-Train Distribution case and an Ellipsoid Container example with 10 satellites are shown in Figure 5. Impulsive burns are recognizable as the cusp points in the trajectory curves. At present, the spacecraft image defaults to a standard GMAT model; pending capability will import custom designs.

Several aspects of on orbit relative motion may seem counter-intuitive for stakeholders and multi-disciplinary collaborators. For example, the effects of differential drag in LEO are generally more pronounced than mission planners expect and can be a design driver on post-deployment attitude control. It is beneficial to see how large a dispersion would be achieved in a given time, or how changes to deployment vectors affect the spreading of the satellites. For supporting stakeholder awareness, SODA has a "free drift" swarm type where no corrective maneuvers are performed; initial conditions are propagated only using the high fidelity model. We find that the free drift mode is useful mainly for studies comparing different swarm types, or to illustrate the underlying relative motion dynamics.

Design analyses and trades studies rely on quantitative evaluations and SODA captures metrics to characterize swarm performance. The plot on the left in Figure 9 is an example study of the collision avoidance feature of the artificial potential functions. We see that the average separation distance grows after deployment, but there are instances where a pair of satellites come within 15 meters of each other in this particular case. Close approach information allows us to study and tune the repulsive functions of the APFs. We realize that in certain missions, the "keep-out" zones around satellites should be much more conservative than in other cases. An additional concern is fuel use. The plot on the right side of Figure 9 shows that one satellite used 1/2 kg of fuel over the duration of this simulation. We would expect that more conservative repulsive APFs, as well as smaller ellipsoid dimensions, would require higher fuel use. Visualizations of the time varying APFs allow a user to study different swarm design choices and how they would impact performance.

Lastly, SODA generates tabular output files representing maneuver commands for swarm maintenance. Ad-

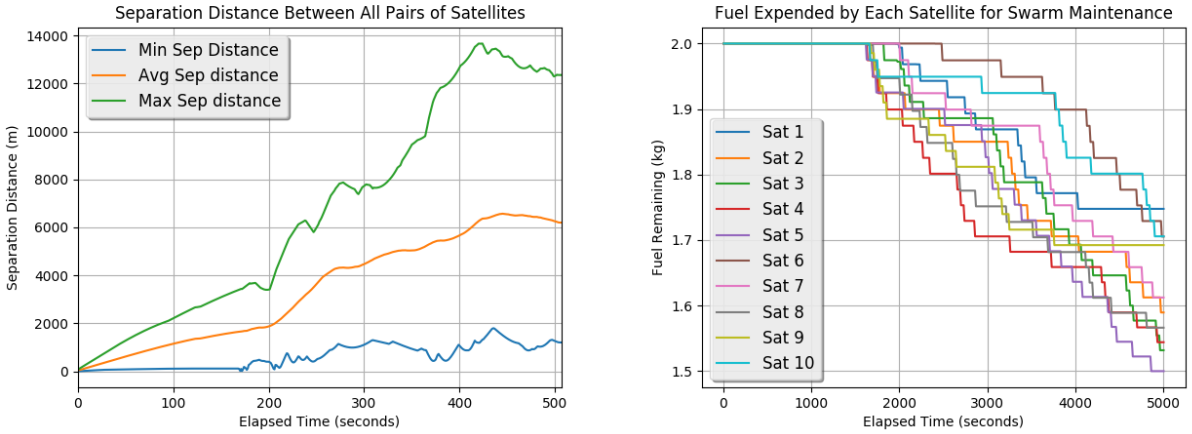

Figure 9: After completing a swarm type simulation, SODA displays performance metrics.

vanced evolution of a swarm controller could become a component of ground support and would leverage the central advantage of SODA's approach by streamlining operations of a large number of swarm members using input commands abstracted for the swarm as a single entity. Closer on the development timeline are interfaces to other mission components that use the maneuver commands, such as control software or analytical simulations.

Conclusions and Forward Work

SODA has proven useful as a tool for both mission concept development and propulsion system research. Work is ongoing to enable additional swarm types, including: statistical distribution, short-hold geometric formation, periodic geometric formation, and specified relative motion. Furthermore, we intend to incorporate techniques that optimize fuel, rather than time, by applying constraints to the timing of maneuvers.

Future research efforts include studying the effectiveness of state of the art propulsion systems for small satellite swarms. SODA enables trade analysis, performance evaluation, and the generation of valuable parametric results to address the technical challenges associated with maintaining a satellite swarm in close proximity.

Orbit Determination

We acknowledge that swarm maintenance requires knowledge of the relative position and velocity of each satellite - i.e. the orbital parameters of each swarm agent. Up to this point, we have made the assumption of accurate state information. In the context of swarms, this is admittedly a big assumption.

The theory of orbit determination is as old as satellites themselves; the orbit of Sputnik was determined from Doppler shift measurements.[8] In today's mission design, accurate measurement data and successful orbit determination algorithms are assumed. For example, the recent Radio Aurora eXplorer II (RAX-2) CubeSat mission reported a maximum GPS-derived error of 4.02 m for position and 0.48 m/s for velocity.[9] This may or may not be sufficient accuracy for swarm satellites, and the requirements will be mission-specific. Deep space swarm missions beyond the help of GPS will rely on alternate orbit determination methods using radio- or optical-based measurements. Analysis of orbital estimation requirements for satellite swarms has been beyond the scope of this initial paper, but imperfect measurements will be modeled in future SODA versions.

Acknowledgments

We would like express gratitude to Scott Richey, Division Chief of Ames' Mission Design Division, for funding this development effort. Also we thank both Scott and Dr. Butler Hine for their countless conversations about satellite swarm autonomy and enhancing the science data return of future missions.

References

[1] Space Studies Board, National Academies of Sciences, Engineering, and Medicine. *Achieving Science with CubeSats: Thinking Inside the Box*. National Academies Press, November 2016.

[2] Theodore Wahl and Kathleen Howell. Autonomous guidance algorithm for multiple spacecraft and for-

mation reconfiguration maneuvers. In *Proc. of the AAS/AIAA 26th Space Flight Mechanics Meeting*, February 2016. (accessed: Mar 2, 2017).

[3] Thomas V Peters, João Branco, Diego Escorial, Lorenzo Tarabini Castellani, and Alex Cropp. Mission analysis for PROBA-3 nominal operations. *Acta Astronautica*, 102:296–310, 2014.

[4] GMAT Development Team. GMAT Central. `gmatcentral.org`, 2016. (accessed: May 8, 2017).

[5] Jeremy Neubauer and Michael Swartwout. Controlling swarms of bandit inspector spacecraft. In *Proc. of the 20th Annual AIAA/USU Conference on Small Satellites, SSC06-V-6*, 2006. (accessed: Mar 14, 2017).

[6] F McQuade and CR McInnes. Autonomous control for on-orbit assembly using potential function methods. *The Aeronautical Journal (1968)*, 101(1006):255–262, 1997.

[7] Andrew R Tatsch. *Artificial Potential Function Guidance for Autonomous In-Space Operations*. PhD thesis, University of Florida, 2006.

[8] William H Guier and George C Weiffenbach. The Doppler Determination of Orbits. Technical report, DTIC Document, 1959.

[9] Jessica Arlas and Sara Spangelo. GPS Results for the Radio Aurora Explorer II CubeSat Mission. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 123, 2013.