SIMULATED ANNEALING MANEUVER PLANNER FOR SEMI-AUTONOMOUS CLUSTER FLIGHT

Allen G. Brown[1], Dr. Matthew C. Ruschmann[2], Dr. Brenton Duffy[3], Lucas Ward[4], Dr. Sun Hur-Diaz[5], Eric Ferguson[6] and Shaun M. Stewart[7]
Emergent Space Technologies Inc., 6411 Ivy Lane, Greenbelt, MD 20770
(301) 345-1535
[1] *allen.brown@emergentspace.com*
[2] *matthew.ruschmann @emergentspace.com*
[3] *brenton.duffy @emergentspace.com*
[4] *lucas.ward @emergentspace.com*
[5] *sun.hur-diaz @emergentspace.com*
[6] *(Eric Ferguson contact TBD)*
[7] *shaun.stewart@outlook.com*

Keywords: *cluster flight, simulated annealing, maneuver planning, F6*

ABSTRACT

A simulated annealing (SA) heuristic search algorithm has been implemented for creating maneuver plans for guidance of a satellite cluster. The algorithm is designed to support a variable number of satellites (also called modules) and multiple maneuver types including: ingress into the cluster, egress from the cluster, station-keeping, cluster reconfiguration, emergency cluster defensive scatter, and regathering the cluster after scattering. The maneuver planner searches over a discretized parameter space varying the earliest and latest maneuver times and target relative orbit elements (ROEs) while using a lower-level linear programming multi-burn solver to evaluate optimal maneuvers that satisfy a set of cluster-wide and module-specific constraints and objectives. These algorithms allow for management of competing operational objectives and constraints while minimizing module ΔV and the probability of inter-module collision. This paper presents an overview of these maneuver planning algorithms and the message-based software service that encapsulates them for supporting semi-autonomous cluster flight.

One challenge of autonomous cluster flight is the system capability to quickly derive maneuver plans for multiple modules operating in close proximity while satisfying complex non-linear constraints with competing objectives for minimizing individual module propellant use and reducing the risk of inter-module collision. To address these requirements, a Maneuver Planning Service (MPS) has been designed to run in a message-oriented-middleware architecture. The MPS is implemented using a combination of C++ classes and MATLAB Coder generated C code. The C++ handcode is developed to provide an interface between the application message layer and the underlying GN&C algorithms which are developed in MATLAB. Originally developed for DARPA's System F6, the development of a reusable service and the supporting GN&C algorithms independently of the flight operating system enables rapid prototyping and GN&C performance testing in an isolated, repeatable, and faster than real-time environment. One of the key features of the MPS design is that most of the same SA algorithm elements are utilized for every maneuver type, thereby reducing the number and scale of unique algorithms that must be created, implemented, and validated before flight.

The MPS incorporates flexible algorithms for generating maneuver plans through an optimization architecture that is scalable with the number of cluster modules. The SA algorithm is designed with flexibility in mind for handling the wide variety of required maneuver types, ranging from single module maneuvers (e.g., a module that must egress the cluster) to simultaneous module cluster-wide maneuvers (e.g., a commanded defensive scatter and regather). The service relies on the application of heuristic optimization methods to achieve a feasible and reasonably optimal maneuver plan for cluster flight maneuver goals within the limited processing capabilities available throughout the cluster. The trade-off between risk of collision and propellant optimality is managed by a set of weighted constraints and objectives evaluated during the simulated annealing search which are selectively applied by the MPS for the given maneuver type.

Results from an example module ingress into the cluster are presented below. Ingress maneuver planning consists of maneuvering the ingressing module from a specified "parking orbit" into its desired trajectory as part of the cluster. This parking orbit is determined by the ground and is passively safe in the radial-crosstrack plane with respect to the modules in the cluster. The ingress problem is bounded by the module's initial navigation state and the designated final ROE state. Otherwise, the MPS is allowed to search over the ingress start time, an optional constrained range of ROE parameters, and burn plan. Standard constraints and objectives for inter-module distance and ΔV are maintained to reduce the risk of collision while minimizing fuel consumption. The maneuver times are allowed to span several orbits in order to take advantage of any multi-orbit transfers that require less ΔV. Figure 1 depicts a module ingressing from its initial parking orbit into a reserved location in a small four-module cluster configuration.

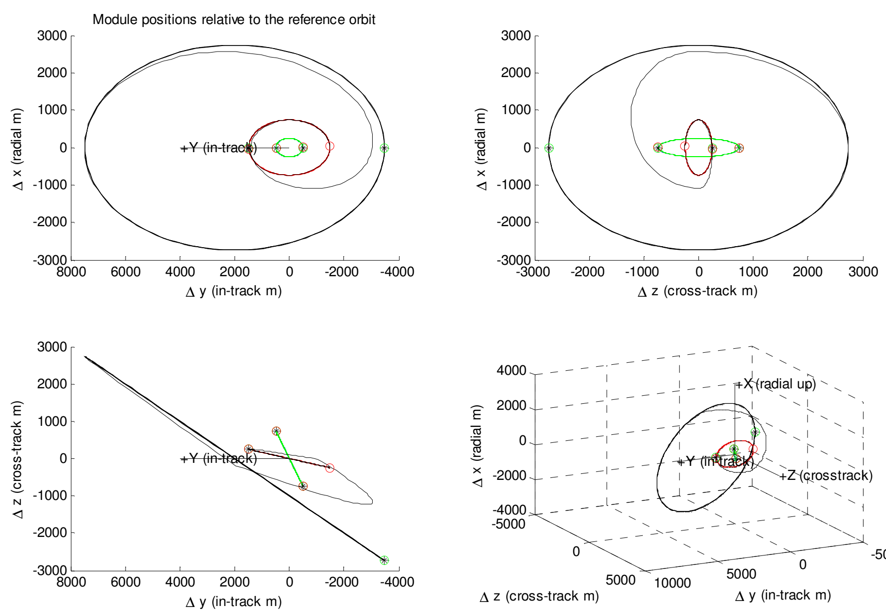

Figure 1: Module Ingress into 4-Module Cluster Configuration