Enhanced Formation Flying Validation Report
(GSFC Algorithm)

March 15, 2002

David Folta and John Bristow
NASA/GSFC
Greenbelt, Maryland 20771

Albin Hawkins and Greg Dell
a.i. solutions Inc.
Lanham, Maryland 20706

NASA/GSFC

Table of Contents

List of Illustrations

List of Tables

1. INTRODUCTION

NASA's first-ever autonomous formation flying mission has been successfully demonstrated. The Earth Observing-1 (EO-1) satellite was launched in November 2000 as a technology mission designed to fly in formation with Landsat 7, another NASA satellite. Both satellites carry instruments that enable scientists to study high-resolution images and climatic trends in the Earth's environment.

The EO-1 satellite flew only 1 minute (450 kilometers) behind Landsat 7 in the same ground track and maintained the separation within 2 seconds, guaranteeing that EO-1 met the requirement that its ground track in the cross-track direction remain within +/- 3 kilometers of Landsat 7's at the equator. This close separation enabled EO-1 to observe the same ground location through the same atmospheric region so that paired scene comparisons between the two satellites could be made. It also demonstrated significantly improved return of science data. The mission allowed engineers to compare technological advances made in ground observing instruments that are smaller, cheaper, and more powerful. EO-1 also demonstrated spacecraft technologies for propulsion, thermal control, antenna systems, and data storage.

The advanced technology demonstrated by the EO-1 mission is called Enhanced Formation Flying (EFF), which was developed by NASA's Goddard Space Flight Center (GSFC). Using this technology, satellites can react to each other and maintain their proximity without human intervention. EFF allows satellites to autonomously react to each other's orbit changes quickly and efficiently. It also permits scientists to obtain unique measurements by combining data from several satellites rather than by flying the full complement of instruments on one costly satellite. Further, it enables the collection of different types of scientific data unavailable from a single satellite, such as stereo views, or simultaneously collecting data of the same ground scene at different angles. The EO-1 EFF validation successfully accomplished ten formation-flying maneuvers that included reactionary maneuvers, formation maneuvers, and an inclination maneuver.

Formation flying, as seen in Figures 1 and 2, is exactly that: satellites flying in a predetermined relative position, and maintaining their respective positions by using onboard control. Therefore, when one satellite moves, the others move to coordinate their measurements.

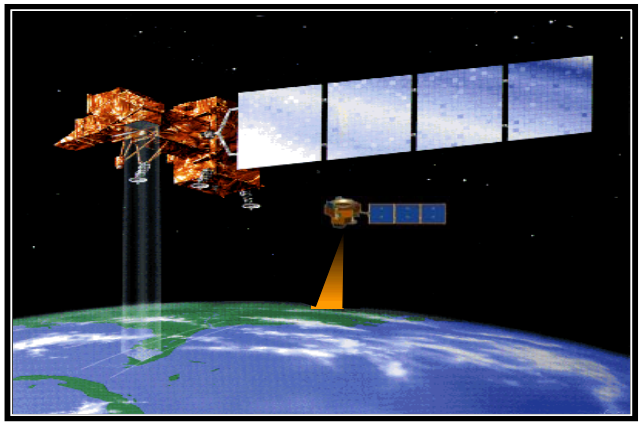

Figure 1. NASA's EO-1 in Formation with Landsat 7

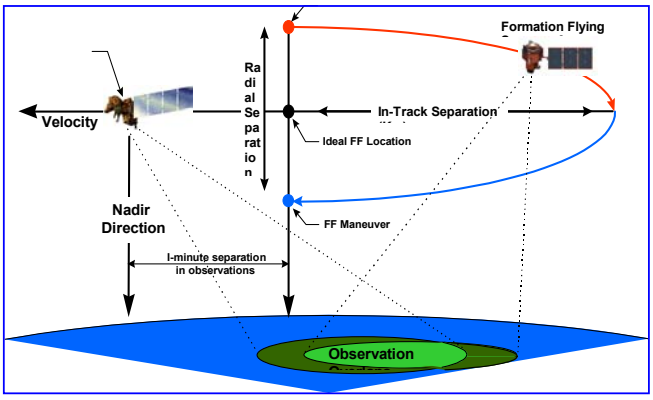

Figure 2. EO-1 and Landsat 7 Formation Geometry

Previously, satellites did not communicate directly with each other, did not plan and execute orbital maneuvers onboard, nor were they equipped to autonomously accommodate the actions of any other satellite in support of a desired scientific experiment. EO-1 flew an advanced technological controller

capable of autonomously planning, executing, and calibrating satellite orbit maneuvers. On EO-1 it was used for the computation of maneuvers to maintain the separation between the EO-1 and Landsat 7 satellites.

The idea and algorithm for an autonomous maneuver control system for this NASA first was conceived and developed by Dave Folta, John Bristow, and Dave Quinn, aerospace engineers at GSFC. This maneuver-control algorithm, called the Folta-Quinn (FQ) formation-flying algorithm, was designed as a universal 3-dimensional method for controlling the relative motion of multiple satellites in any orbit (see references 3, 7 and 8 for mathematical details of the algorithm). The algorithm was combined with a new ground software package that was then converted to a flight software version and interfaced with the EO-1 command and data handling (C&DH) component. This combined flight software package utilized an onboard flight code called AutoCon™, developed by a.i. solutions Inc. under contract to GSFC, that incorporated artificial intelligence (AI) technologies such as fuzzy logic and natural language scripting to resolve multiple conflicting constraints and provide automated maneuver planning. The software package also provided for the ingest and smoothing of real-time navigation data from any orbit determination system (the onboard Global Positioning System (GPS) in the case of EO-1), the transfer of data from the maneuver algorithm for maneuver commands, the computation of onboard predictions of where the satellites will be in the future, the generation of necessary attitude pointing information, and the actual commanding of thruster firings. This total software system constitutes the EFF system.

Because maneuver calculations and decisions can be performed onboard the satellite, the lengthy period of ground-based planning currently required prior to maneuver execution will eventually be eliminated. The EFF system was also modular so that it can be easily extended to other mission objectives such as simple orbit maintenance. Furthermore, the flight controller was designed to be compatible with various onboard navigation systems.

Formation flying technologies are primarily concerned with the maintenance of the relative location between many satellites. Much shorter and more precise baselines can be established between the satellites. The satellites can then be combined as part of a “virtual satellite” that can provide previously unobtainable science data using mass-produced, single-string, relatively cheap satellites. Multiple scientific instruments often present competing and conflicting requirements on a satellite design and its operation. So much science at stake for a single satellite often requires a great deal of onboard redundancy, which imposes significant overhead on the design process. Separating scientific payloads onto several simpler single-string satellites can accomplish the same complex missions without the added design and operational overhead, while risking only one payload at a time. The proposed approach for onboard formation control will enable a large number of satellites to be managed with a minimum of ground support. The result will be a group of satellites with the ability to detect errors and cooperatively agree on the appropriate maneuvers to maintain the desired positions and orientations.

Another reason to use formations is that the sensitivity of scientific instruments can often be increased by expanding the effective observation baselines (separation distances). This can be achieved by distributing the scientific instruments over many separate satellites. The formation flying technologies flown onboard EO-1 makes these missions routine and cost effective.

Since this technology has now been fully developed and demonstrated, synchronous science measurements occurring on multiple space vehicles will become commonplace and the concept of Earth observing “virtual platforms” will become a reality. In the process, this technology enables the development of autonomous rendezvous. Scientific payloads could be launched from any launch vehicle, rendezvous with and join a formation already in place, and then autonomously maintain this condition or respond to specific requests for science data collection by altering its own orbit. Thus, this technology addresses all of the NASA directives to build revolutionary satellites that are better, faster, and cheaper.

2. TECHNOLOGY DESCRIPTION

2.1 Enhanced Formation Flying (EFF) Description

Enhanced Formation Flying is a new autonomous onboard technology that involves data ingest and propagation and maneuver calculations using any navigation system such as GPS, celestial navigation, Tracking and Data Relay Satellite System (TDRSS) Onboard Navigation System (TONS), or state vector uplinks. Its flight software is capable of autonomously planning, executing, and calibrating any routine spacecraft orbital maneuvers using navigation system inputs. The autonomous formation flying control software (the executive) is called AutoCon™ and builds on GSFC Guidance, Navigation, and Control (GN&C) existing capabilities for the maneuver planning, calibration, and evaluation tasks. AutoCon™ also includes a fuzzy control engine, ideal for this application because it can easily handle conflicting constraints between spacecraft subsystems. As part of the AutoCon™ executive system, the Folta-Quinn (FQ) maneuver-planning algorithm, has been implemented. The output of the FQ algorithm provides the AutoCon system with a maneuver ΔV and attitude information. AutoCon™ then takes this maneuver data and computes a maneuver duration and attitude control. The maneuver start, duration, and attitude are then generated as part of the overall absolute time sequence. The onboard flight version of the ground development code consists of a subset of the ground-based version.

2.2 What is "Enhanced" Formation Flying

The term "enhanced" in the EFF context refers to the improvements over basic formation flying computations. The basic computation is a simple maneuver used to alter the semi-major axis (SMA) or to allow orbit period changes as a derived means of relative orbit maintenance. The enhancements included in EFF for the New Millennium Program (NMP) EO-1 are:

- Natural language script (ASCII) to control the system without flight software modifications.
- A universal 3-dimensional targeting algorithm that allows maneuvers in any direction to maintain the formation from direct measurements. This also allows formation maintenance maneuvers and inclination maneuvers.
- A GPS smoother that ensures consistent and accurate GPS input.
- Autonomous calibration of maneuvers that allow the maneuver performance to be measured and used in the planning of the next burn.
- Interface to telemetry and spacecraft attitude and propulsion data through a common C&DH path.
- Interface to the command system through both the C&DH (received) and the stored command processor (issued).
- A fuzzy logic control system that allows multiple constraint evaluations in script form.
- Additional tasks, such as propagation of Tracking and Data Relay Satellite (TDRS) states for antenna pointing or, as in the EO-1 case, propagation of Landsat 7 state for target computations.
- The computation of the attitude quaternion to orient the spacecraft in the correct direction based on thruster location and pointing.

All of these enhancements were fully tested during integration and test (I&T) and during spacecraft comprehensive performance tests. These enhancements were also used onboard during the formation flying experiment.

2.3 Software Architecture

The EFF flight control system ingests data from EO-1 sensors and subsystems such as propulsion, navigation, and attitude data. It then autonomously generates, analyzes, and executes the maneuvers required to initialize and maintain the formation between Landsat 7 and EO-1. Figure 3 shows a functional diagram of EFF and the AutoCon™ system. Because these calculations and decisions are performed onboard the spacecraft, the lengthy period of ground-based planning currently required prior to maneuver execution is eliminated. The system is general and modular so that it can be easily extended for future missions. Furthermore, the AutoCon™ flight control system is designed to be compatible with various onboard navigation systems (e.g., GPS, or an uploaded ground-based ephemeris). The AutoCon™ system executes on the Mongoose 5 (M-5) EO-1 spacecraft computer. Interfaces are handled by a single direct interface to the C&DH system. This is used to ingest the GPS states information, AutoCon™ commands, EFF telemetry, and maneuver commands for EO-1. The FQ algorithm requires input data for the current EO-1 state, the target state, and the desired state. This data is provided by AutoCon™. AutoCon™ takes the current EO-1 and uploaded Landsat 7 states and then propagates them for a user-specified fraction of the period. Autonomous orbit control of a single spacecraft requires that a known control regime be established by the ground that is consistent with mission parameters. That data must then be provided to the spacecraft. When orbital perturbations carry the spacecraft close to any of the established boundaries, the spacecraft reacts (via maneuver) to maintain itself within its error box. The system is currently set to check the tolerance requirements every 12 hours. From this point, AutoCon™ propagates the states for 48 hours (a commandable setting) and will execute a maneuver plan if needed.

The overall size of the code is designed to run on the M-5 within limits of ~600 kilobits and utilize the central processing unit (CPU) only over 4-second time intervals. The size of the AutoCon™ system varies depending on its utilization, that is, whether propagation is required for predictable maneuver planning and forecasting and if a selectable method for determining the location and time of the maneuver is necessary. Thus size and execution speed will vary on implementation in other spacecraft computers.

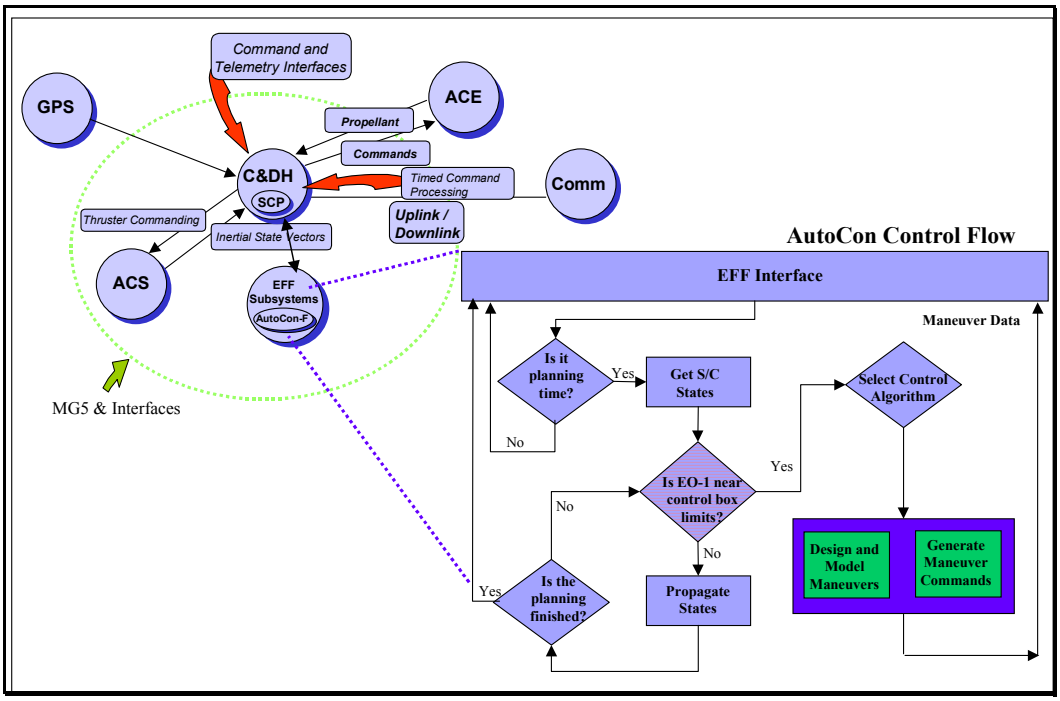

Figure 3. EO-1 EFF Functional Diagram (GSFC approach)

2.4 Maneuver Control Modes

There were five EFF maneuver control modes onboard EO-1, as shown in Figure 4. All control modes were verified onboard during this validation process. These modes were established to allow an incremental validation of the system performance, data interfaces, and maneuver computations before commands were generated onboard for an executable maneuver. Modes 1 and 2 were validated in a functional test while modes 3, 4, and 5 were validated as executed EO-1 maneuvers. Modes 1 and 2 provided for the testing of the onboard interface and basic functions. These modes were executed in the initial validation process. Modes 3, 4, and 5 were executed to validate that the maneuver planning process was correct and led up to the full autonomous maneuver planning.

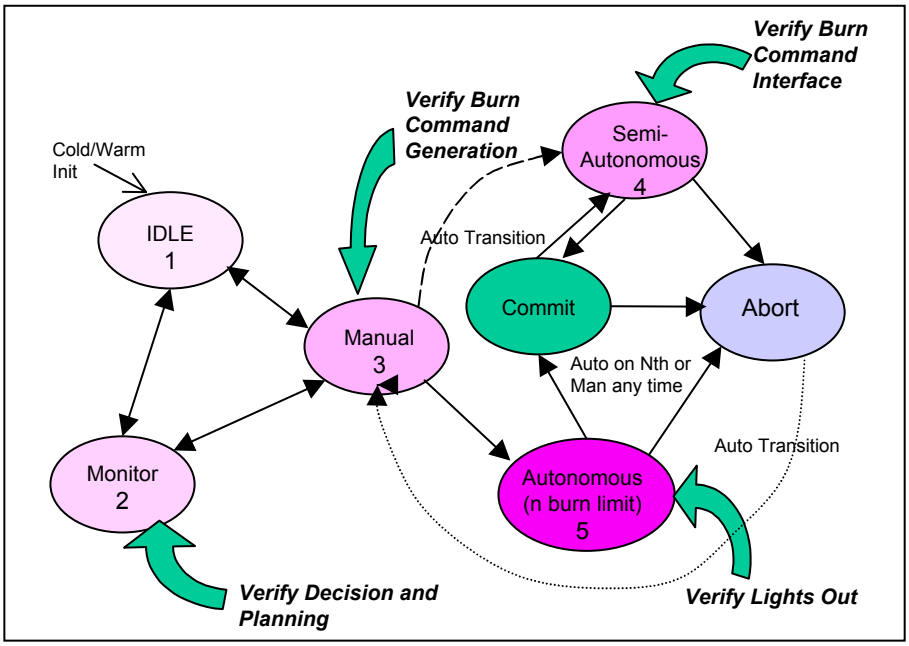

Figure 4. EFF Commandable Modes

2.5 EO-1 ΔV Computations and Quantization of Maneuvers

The computation of the EO-1 maneuver ΔVs was performed using a sequence of two methods. The first method used the FQ algorithm for the calculation of the maneuver to reach the targeted position relative to Landsat 7. Subsequently, a simple velocity-matching maneuver was performed once the targeted position was attained. The FQ algorithm could also be used, but in an effort to simplify onboard processes, as no state propagation was necessary, a velocity matching method was employed. This velocity matching was computed from the predicted difference in the velocity of the EO-1 transfer orbit and the targeted state at the target position.

The EO-1 spacecraft propulsion system was designed so that the minimum maneuver duration was one second with larger burns selectable at 1-second increments. This meant that commands generated either onboard or on the ground underwent a rounding of the maneuver duration based on the computed ΔV. For example, if a maneuver was such that the computed maneuver duration was 5.49 seconds, the commanded maneuver would actually be 5 seconds, and a 5.51-second duration would become 6 seconds. This resulted in a quantized maneuver duration for each maneuver and thus the achieved Keplerian trajectory differed slightly from the targeted trajectory. To compensate for this effect, the final ΔV was adjusted. The velocity match was perturbed slightly to compensate for the position error resulting from the prior maneuver's quantized burn duration. This allowed the targeted orbit's SMA to be achieved with a trivial sacrifice of eccentricity.

This FQ algorithm combined with AutoCon™'s ability to autonomously plan, execute, and calibrate routine spacecraft orbital maneuvers enabled formation flying. AutoCon™'s fuzzy control engine was ideal for this application because it could easily handle conflicting constraints between spacecraft subsystems.

2.6 Folta-Quinn Algorithm Control

The FQ algorithm uses input data of the current spacecraft state to compute the target state and the desired state. This data is provided by AutoCon™, which takes the current state of the control spacecraft and calculates its orbital period. It then propagates this state for a user-specified fraction of the period. This propagation provides the location of the control spacecraft at the target epoch. User-specified offsets are applied to this state to create the target state. The target state is then propagated back to the epoch of the initial state, producing the desired state. This procedure creates the required inputs to the FQ algorithm.

Establishing the desired state of a spacecraft's location is as varied as spacecraft missions themselves. Autonomous orbit control of a single spacecraft requires that the ground establish a known control regime, which is consistent with mission parameters. That data must then be provided to the spacecraft. When orbital perturbations carry the spacecraft close to any of the established boundaries, the spacecraft reacts (via maneuver) to maintain itself within its error box. Once an error box is provided to the spacecraft, no further ground interaction is required. EFF takes the next step up the technological ladder by permitting the spacecraft themselves to establish where their own control boxes should be. This requires cooperation between all the members of the formation, and therefore a depth of communication between all the individual satellites that is not practical (or in some cases even possible) from the ground. This may occur through cooperative "agreement" by controllers of all the spacecraft in the formation or by maintaining a relative position from a designated "lead," or by some hybrid of these two methods.

The AutoCon™ flight control system ingests data from additional sensors and spacecraft subsystems such as propulsion, groundtrack, navigation, and attitude subsystems. It is then possible to generate, analyze, and execute autonomously the maneuvers required to initialize and maintain the formation between satellites such as Landsat 7 and EO-1.

2.7 Orbit Mechanics of EO-1 Formation Flying With Landsat 7

In Figure 5, EO-1 starts a formation at the red dot located at a distance of 450 kilometers behind Landsat 7 and above by approximately 50 meters. Due to the difference in the accelerations from atmospheric drag and spacecraft design, the EO-1 satellite orbit decays faster than that of Landsat 7. While above Landsat 7, EO-1 drifts away from Landsat 7. After several days of atmospheric drag, EO-1 is below Landsat 7 and is drifting toward it. When EO-1 is outside the required separation distance or if the Landsat 7 satellite has maneuvered away, EO-1 autonomously computes and performs a maneuver to reposition it to an initial condition to repeat the relative motion and meet science data collection requirements.

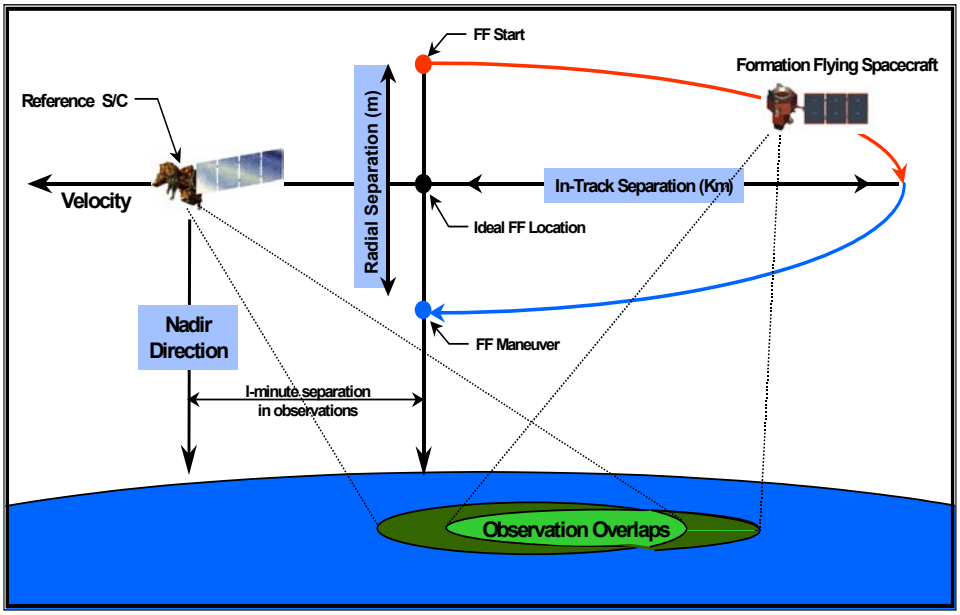

Figure 5. EO-1 and Landsat 7 Formation Geometry

3. TECHNOLOGY VALIDATION

The EO-1 software validation activity certified that all software requirements had been properly implemented and that the EFF software met all operational objectives. This section summarizes the approach used to accomplish these goals. The core AutoCon™ flight control software was qualified by executing a series of test plans, test data, and test scenarios. The results of each stage of validation were checked and documented. These activities had inputs from both the developers of AutoCon™ and the EO-1 attitude control system (ACS) software engineers. Quality assurance was integrated into each stage.

3.1 Ground Verification Process

The qualifications of the processes that were used to monitor verification were by analysis, inspection, test, and demonstration. The requirements by which the test show qualification were by ACS external interfaces, functionality, sizing, timing, and tractability. The verification of each of these tests was performed at the following levels. Note that Levels 1-4 were the ground verification process required to support onboard, Level-5 comprises validation of EFF. All levels, 1-5, were successfully completed.

- Level 1: AutoCon™, using a PC or workstation environment to develop, test, and provide high fidelity simulations, and proof-of-concept fuzzy logic rules.
- Level 2: Virtual Simulation, using a virtual simulation of the ACS with an embedded AutoCon™ core architecture flight code design to test the interfaces, telemetry, and commands with the ACS.
- Level 3: Software Test Facility, using a full spacecraft simulation of the ACS and GPS data to test AutoCon™. Test all interfaces to the ACS and C&DH for telemetry and commanding. Perform on a Mongoose breadboard with supporting hardware.
- Level 4: FlatSat, testing of the AutoCon™ flight code on flight hardware and ACS software.
- Level 5: Operational testing/validation of the core AutoCon™ flight code. These tests require a minimum amount of testing to verify proper execution of the AutoCon™ flight control system.

To minimize costs associated with these tests, the following approach was recommended.

- For each functional requirement, develop scenarios for the mission.
- Develop system test for each scenario.
- Develop system unit, integration test for EO-1 AutoCon™ to develop a system checkout matrix.
- Perform system tests for the mission scenarios and catalog results in a matrix.

The EO-1 maneuvers were computed onboard under a single system architecture that employs separate maneuver decision/design modules or algorithms. AutoCon™ controls execution of the modules through an onboard mode switch and performs constraint evaluation via fuzzy logic control. The AutoCon™ specifications were levied on the industry partners in order to facilitate uploading algorithms, patches, scripts, and required tables during the mission. Data and processing requirements from any potential industry partners were assessed during this initial phase of the technology. Figure 6 shows the ground development and test architecture used to verify the AutoCon™ executive, its interfaces, and the FQ algorithm. The software and hardware architecture is specified in Table 1.

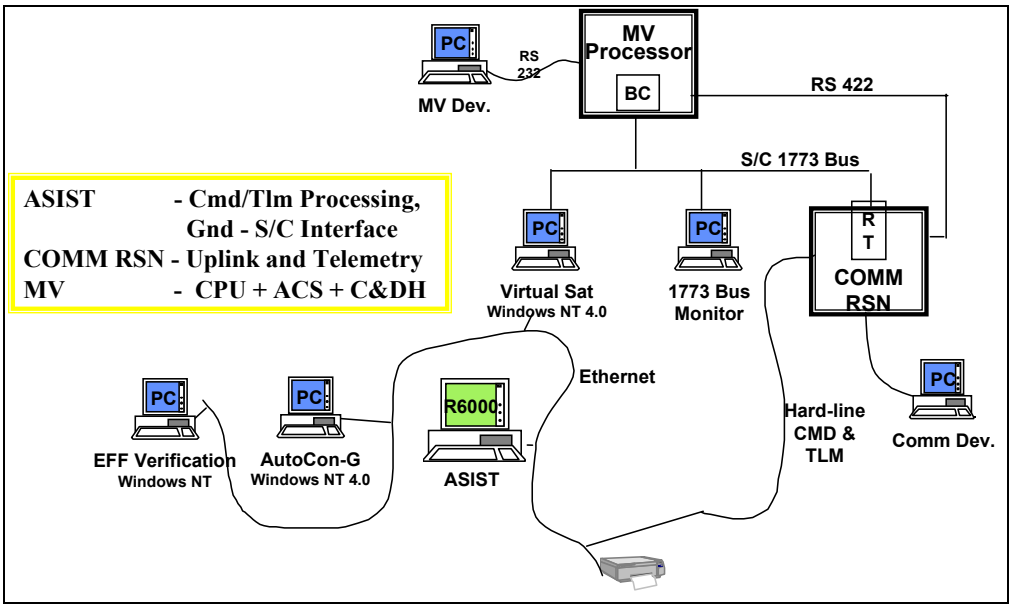

Figure 6. EFF Ground Verification

Table 1. Supporting Integration and Test (I&T) Hardware and Software

Data Verified	Software	Hardware
Orbital Data	Freeflyer	PC/Windows-NT
Interface Checkout	VirtualSAT	PC/Windows-NT
AutoCon-Ground	AutoCon	PC/Windows-NT
Table Loads, Algorithms, etc.	Flight S/W TestBed	PC/Workstations,MG5
Telemetry Data	Telemetry Processor	EO-1 Control Center H/W

3.1.1 AutoCon™ Executive and Fuzzy Logic Verification

Verification of the core AutoCon™ architecture executive was performed during the first year of the EO-1 mission development. This build was the system-level control of all of the EFF algorithms. The

objective was to test the fuzzy logic control and the development of the overall architecture. The tests ensure that the input, output, CPU memory, storage, processing speed requirements and the interface to the ACS-provided data perform as expected and that controls were invoke at the proper time for maneuver algorithms.

3.1.2 Required Data and Necessary Measurements

The data required to validate AutoCon™ are listed below from references 6, 7, 8, and 10. Data requirements were real data sets of EO-1 position state vectors from the EO-1 GPS orbit determination solutions and the Landsat 7 state vectors from the uplink of these vectors. The ACS provided data in memory locations for input to the fuzzy logic control. Output files for placement into the interface with the ACS for telemetry were exercised.

3.1.3 Approach

The validation approach was to execute AutoCon™ onboard with these input data values listed and allow AutoCon to process the data using the control algorithms. These algorithms both notify the ACS and ground through telemetry of a maneuver and invoke the maneuver planning algorithms within AutoCon. The validation showed that the fuzzy logic properly resolved conflicting constraints, that AutoCon™ can ingest the data from the ACS correctly for internal use, and that the interfaces with the ACS for all telemetry and command were working correctly. The final result of the validation was that the telemetry output confirmed the maneuver decision had selected a proper time for a maneuver. Also, the validation proved the interface to AutoCon™ via ACS uplinked tables functioned properly and confirmed the required memory sizing of the onboard computer

3.1.4 Ground Testing Results

The results were that AutoCon™ returned a maneuver-required flag and related information for the planning of the maneuver. There were no interface errors. The AutoCon™ software ran within the tolerance specified for the memory and timing requirements of the onboard computer. This verified the AutoCon™ interface to the ACS. An analysis of the downlinked telemetry showed that the data provided through memory to the AutoCon™ system and that the execution of the high level AutoCon™ system in terms of fuzzy logic, system control limits and flags were as expected. An indication by AutoCon™, that the data for the maneuver algorithms had been generated and control passed to the correct maneuver process, was as expected. The results were that the data within the telemetry data packets matched the ground-generated data. The observed differences between the ground and onboard AutoCon™ were as expected and were due only to differences in the software (constrained software run times or precision) and hardware (PC-based running Windows-NT versus flight hardware). Scenarios for the validation address each difference.

3.1.5 Supporting Integration and Test Data

Supporting I&T data was required for propulsion, health and safety area, and uplinked constraint for AutoCon™ control. The input data included preloaded fuzzy rule set and constraint checking limits. The validation required that these data be commandable for a complete checkout of this algorithm. The validation required software and hardware used for independent checking of orbital data, the use of the ground operational version of AutoCon™ for the validation of the fuzzy logic and rules, and the use of the Hammers Company's VirtualSat and the Flight Software Testbed for checking of all interfaces and the associated timing requirements.

3.1.6 Rationale

The reason for this verification was to test the control methodology of the AutoCon™ executive through the processing of the fuzzy logic rules and the fuzzy logic engines. The differences indicated above were minimal and due only to implementation in the spacecraft-specified hardware and software.

3.2 EFF Onboard Validation

The onboard validation planning process originally assumed that the Landsat 7 maneuvers occurred every two to three weeks. The reality was that Landsat 7 performed maneuvers every three to four weeks and then for a short period more frequently at one-week intervals. The requirement to maintain a one-minute separation between EO-1 and Landsat 7 was always met, but frequently EO-1 maneuvered before a complete "revolution" of the EO-1 formation cycle. The on-board validation testing centered on interfaces, the FQ algorithm, performance of the propagation, and overall system loading and safety.

3.2.1 Validation of Interfaces and Algorithm

The purpose was to validate the interfaces for the input of the EO-1 state vector information from either GPS or state vector uploads. The research team also began to validate the overall performance of the FQ algorithm, starting with collecting Loral Tensor (GPS) and S-band tracking data as soon as the Loral Tensor receiver was tracking. The intent here was to analyze GPS characteristics and test GPS accuracy against S-band tracking orbit determination. The Tensor data was fed into AutoCon™-Flight (F)-NT, a PC version of AutoCon™-F, for refinement of the smoother setup. This data provided the first GPS performance measurements. Smoothing requirements and targeting accuracy could be ascertained from the measurements. A smoothing sample consisted of at least two orbits of data. Longer continuous sampling of Tensor data was not truly required but was highly desirable. (Collection of at least three smoothing samples is the minimum required with two days of nearly continuous data is desired.) This collection represented a good sampling of GPS conditions but more data was required to prove this. Collection of the data began as soon as the Tensor was processing to give adequate analysis time for setup of the smoother operations to follow. For the best comparison, the highest accuracy S-band tracking OD solutions were required to coincide with the sampling of the Tensor data. The data of interest from the Loral Tensor could be extracted from EO-1 virtual channel (VC)-1 data replays. Note that this step did not require running the total EFF executable.

At this point testing concentrated on basic EFF operation. The intent was to provide quick functional confidence. When EFF was first turned on (taken out of its idle state), a default script was executed to provide overall functional checkout of the onboard software. One AutoCon™ script was executed to perform this quick test and is described in the procedure "EFF Quick Test of AutoCon™-F." Script 1 of the procedures can be executed beyond the quick checkout to provide a simulated CPU loading. This was the first occurrence of heavy loading that EFF can produce.

The research team then began to validate the operation of the EFF Smoother to determine its usefulness and need. The intent was to analyze basic smoother operation and tweak its setup. EFF was operated in the **MONITOR** mode during this phase and EO-1 state data flows were verified. A script was loaded and the smoother operated, providing smoothed solutions continuously. With the best case Tensor data, smoothing cycles take slightly over 2 hours each to be generated. This provided over ten solutions a day. During this shakedown of the smoother, daily analysis of the results was performed by the EFF/AutoCon™ design team. Any reconfiguration was accomplished by table uploads. After a workable smoother configuration was obtained, continuous operation of the smoother was done for several days to verify successful behavior. During this verification period, the highest accuracy S-band tracking solutions were desired for the best comparison.

This script was executed in the **MONITOR** mode. Note that this was the first operation of EFF at a current epoch and most tables needed to be loaded. The data of interest from this phase was extracted from EO-1 VC-1 data replays using a Standard Test Operating Language (STOL) procedure. At this point, testing was concentrating on EFF and the proper transfer of state data.

Next the research team validated the GSFC Targeter in **MONITOR** mode for quick checkout of data flow. The intent here was to extend the quick checkout performed in step two. During this phase,

additional data flows were verified. This process was short as it covered only several passes. Operation of AutoCon™-F expanded to encompass most of the tables at this point. Telemetry and table dumps were analyzed to ensure proper data flow. The EFF team monitored telemetry and analyzed dumps of AutoCon™ tables, which contained data crucial to EFF.

The AutoCon script run during this test was designated "eff_gsfc_test_1.autocon." Execution of this script was performed in the **MONITOR** mode. (See EFF/AutoCon Initialization and Operation for description of how to bring up EFF and AutoCon™ and execute a script.) Note that this is a script change from the previous phase. All other data was consistent and did not need to be reloaded. The data of interest from this phase was extracted from EO-1 VC-1 data replays using STOL procedures. Note that this step transitions to the next step by switching to monitor mode.

At this point testing was concentrating on the GSFC Targeter. The GSFC Targeter was operated in **MANUAL** mode with a script planning maneuvers continuously. In a true operation, planning occurred every 12 hours. The intent was to analyze GSFC Targeter performance on orbit. The continuous planning maximized the number of burn plans generated. There was one "forced" maneuver generated on the order of every three hours. The maneuvers generated were not at the desired maneuver times and locations but rather provided a complete sampling of Targeter performance. During this test, maneuvers of varying magnitude were generated as well as sampling of varying orbital parameters. Several Landsat 7 maneuvers occurred during this phase. At the completion of this phase, the level of confidence in the GSFC Targeter's ability to formation fly EO-1 was established.

The AutoCon™ script to run during this test was designated "eff_gsfc_test_1.autocon." Execution of this script was done in the **MANUAL** mode. (See EFF/AutoCon™ Initialization and Operation for a description of how to bring up EFF and AutoCon™ and execute a script.) This was only a mode change from the previous phase and should require only that operation. The data of interest from this phase was extracted from EO-1 VC-1 data replays using STOL procedures.

The next step was to operate the GSFC Targeter in **SEMIAUTO** mode with a script planning the desired maneuvers. The intent here was to take an onboard-generated maneuver and allow it to progress into the implementation phase to complete the data flow verification and test the EFF burn implementation. The essential items during this phase were the verification of the burn command sequence generation and the absolute time sequence (ATS) patch of these commands. Operation in this phase encompassed two to three Landsat 7 maneuvers or EO-1-only maneuvers. At the completion of this phase, the level of confidence in the EFF/AutoCon™ process was established with the ability to execute a script that planned maneuvers around the input constraints and implement maneuvers successfully.

The AutoCon™ script run during this test was designated "eff_gsfc_ops_1.autocon." This test was executed in a **SEMIAUTO** mode. Note that this was a script change from the previous phase. All other data was consistent and did not need to be reloaded. The data of interest from this phase was extracted from EO-1 VC-1 data replays using STOL procedures.

The final step was to operate the GSFC Targeter in **FULLAUTO** mode. The intent here was to demonstrate autonomous maneuver capability and provide the Flight Operations Team with an opportunity to independently operate EFF and the GSFC Targeter while the EFF design team observed. Operation in this phase encompassed at least two Landsat 7 maneuvers with four or more desired.

The AutoCon™ script run during this test was designated "eff_gsfc_ops_1.autocon." This script was executed in **FULLAUTO** mode. Note, this was only a mode change from the previous phase and only required that operation. The data of interest from this phase was extracted from EO-1 VC-1 data replays using STOL procedures.

At this point (July 20, 2001), testing was temporarily completed. The choice of SEMIAUTO vs. FULLAUTO depends upon the requirements of operations and the results of the first year's operations. Also the EFF/AutoCon™ design team has begun to relax its continual observation and assume an on-call posture. A key aspect for this experiment is for the Flight Operations Team to be able to operate the system. A summary of the onboard tests conducted is shown in Table 2.

Table 2. Validation Test Completion

Checkout and Monitor	Tests	Date	Complete
Uploads of Tables and Scripts	Correctly Accepts Data	1/31/01	Yes
Propagation With Forces	Propagate for Duration	1/31/01	Yes
Two-body Propagation	Prop Model	1/31/01	Yes
Conditional Constraint Check	Formation Constraints	1/31/01	Yes
GSFC Targeter	Folta/Quinn Algorithm	2/02/01	Yes
GPS Data Smoother	GPS Position Smoother	5/01/01	Yes
Manual Mode			
Conditional Constraint Check	Formation Constraints	3/30/01	Yes
GSFC Targeter	Folta/Quinn Algorithm	3/30/01	Yes
GPS Data Smoother	GPS Position Smoother	4/12/01	Yes
Semi-autonomous			
Conditional Constraint Check	Formation Constraints	5/01/01	Yes
GSFC Targeter	Folta/Quinn Algorithm	5/01/01	Yes
GPS Data Smoother	GPS Position Smoother	5/01/01	Yes
Autonomous			
GSFC Targeter	Folta/Quinn Algorithm	7/01/01	Yes
GPS Data Smoother	GPS Position Smoother	7/01/01	Yes
JPL Targeter Upload & Exec	Upload of JPL Algorithm	7/01/01	Yes

3.2.2 Validation Results and Period of Performance

The EFF experiment was executed over a several-month period, from January 12, 2001 through July 12, 2001 and then again in November 2001. The executions generated more than 600 maneuver test plans and ten successful maneuver commands to control the formation. The validation tests were divided into two areas, functional tests of modes 1 and 2, and autonomous maneuver execution tests of modes 3, 4, and 5.

Functional Tests

Functional maneuver tests were planned in sets of three based on three propagation durations. GPS data was ingested 177 times while tables were uploaded approximately 30 times for script control, Landsat 7 data, and environmental data updates. The functional validation was accomplished by comparing several events and computations[7]. These tests included:

- EO-1 GPS and Landsat 7 state ingest
- EO-1 and Landsat 7 propagation events (generate target and desired states)
- FQ Targeting Algorithm output
- Absolute and quantized maneuver ΔV

- Three-axis maneuver ΔV (any direction)
- Internal calculations (matrices, variables, and states)

Autonomous Tests
The maneuver execution tests were accomplished less frequently as they were tied to the operational maneuver timeline. The manual, semi-autonomous, and fully autonomous maneuvers were computed in much the same manner as the functional continuous tests with the following exceptions. Maneuvers were planned at a required maneuver epoch with the output used for planning of EO-1 formation flying maintenance maneuvers. The autonomous maneuver was planned with both a ground based S-band definitive orbit determination solution and the output of the GPS system onboard EO-1. The radial component targets varied over the demonstration as the atmospheric density was changing and the relative decay rates of both spacecraft needed to be considered. The radial targets relative to Landsat 7 were 40m, 60m, and 20m.

On January 12, 2001, the EFF experiment onboard EO-1 became operational. EFF was started in modes 1 and 2 whereby GPS data would flow though the C&DH interface into the AutoCon™ executable and maneuvers were computed continuously. Scripts and data uploaded via tables were enabled though the execution of EFF. With this data, maneuvers were calculated at specified intervals. The overall computational interval was approximately 3 hours in duration and began with the ingest of a single GPS EO-1 state. This state, along with an uploaded Landsat 7 state, was then propagated onboard for durations of 12 hours, 24 hours, and 48 hours. Maneuvers were computed at the 12-, 24-, and 48-hour epoch marks. After the last maneuver was computed, a new GPS EO-1 state was ingested and the process began again. This enabled continuous computation of maneuvers while verifying the ingest and data interfaces and propagation of states onboard EO-1.

3.2.3 Functional Validation of Modes 1 and 2
This section presents onboard and ground comparison results in terms of the absolute difference in the computed ΔV (cm/s) and the related percentage error for several maneuver scenarios. A total of 12 scenarios consisting of three maneuver sets (two maneuvers per set) for a total of 36 combined maneuvers were verified. The locations and epochs of these maneuvers were chosen randomly at approximately one per day over a three-week span. Figures 7 and 8 present the overall performance of each quantized maneuver as an absolute difference in the ΔV magnitude and its percent error. The mean value of the quantized difference is 0.0001890cm/s with a standard deviation of 0.000133 cm/s. These resulting data show that there was excellent agreement between the onboard system and ground validation system. The larger residual in figure 7 is due to a 1-second quantization of a velocity-matching maneuver. This difference was due to the onboard system yielding a maneuver duration near the midpoint that rounded down while the ground system rounded up. The difference was still small at 1.4% as it was a small residual on a small total maneuver duration. Figures, 9 and 10 present maneuver comparisons for the 3-dimensional computation. This provides the comparisons for the total ΔV required to align EO-1 directly behind Landsat 7 and involves all three ΔV components of radial, alongtrack, and crosstrack.

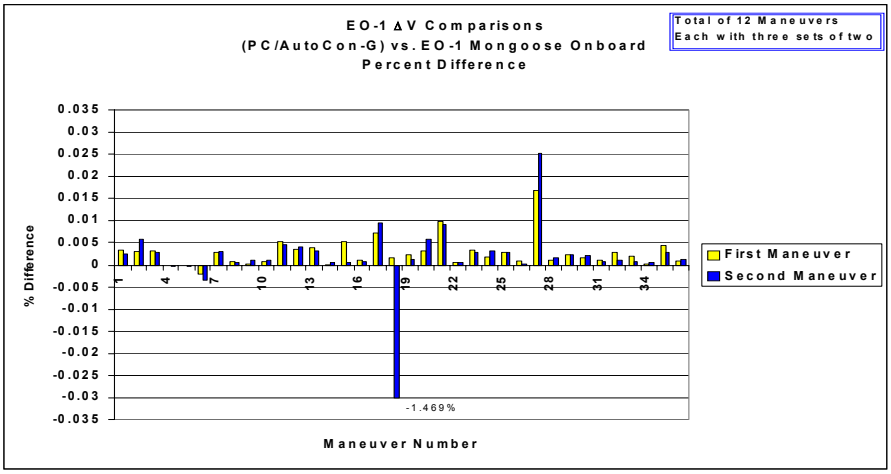

Figure 7. Percentage Difference in EO-1 Onboard and Ground Absolute ΔVs

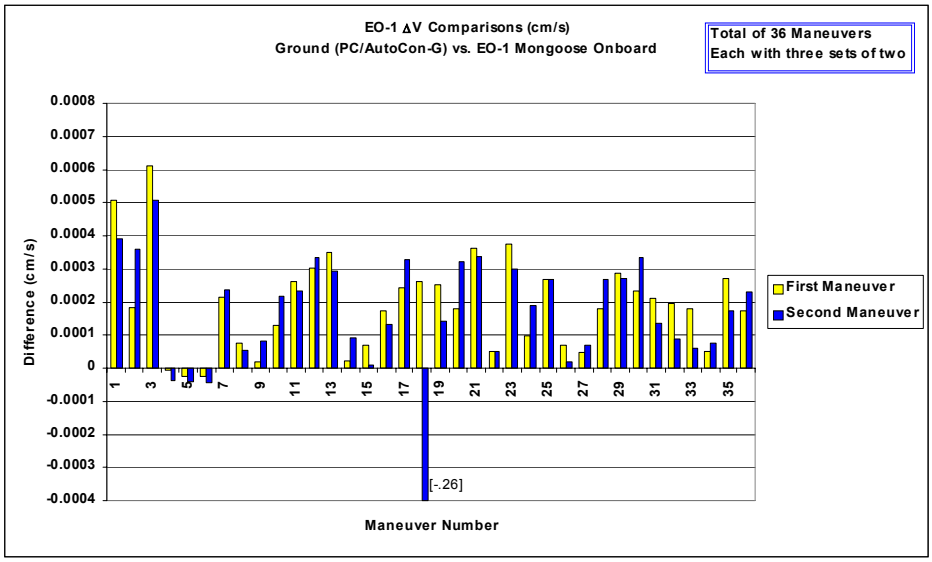

Figure 8. Difference in EO-1 Onboard and Ground Absolute ΔVs

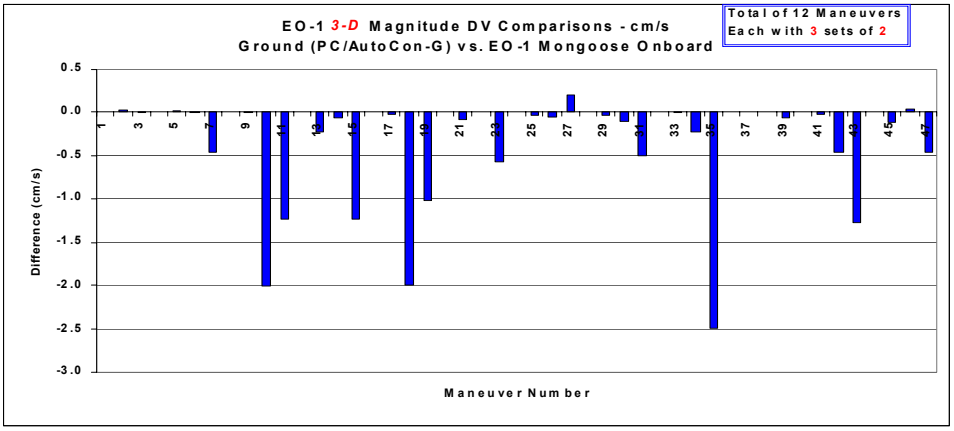

Figure 9. Absolute Difference in 3-D Onboard and Ground ΔVs

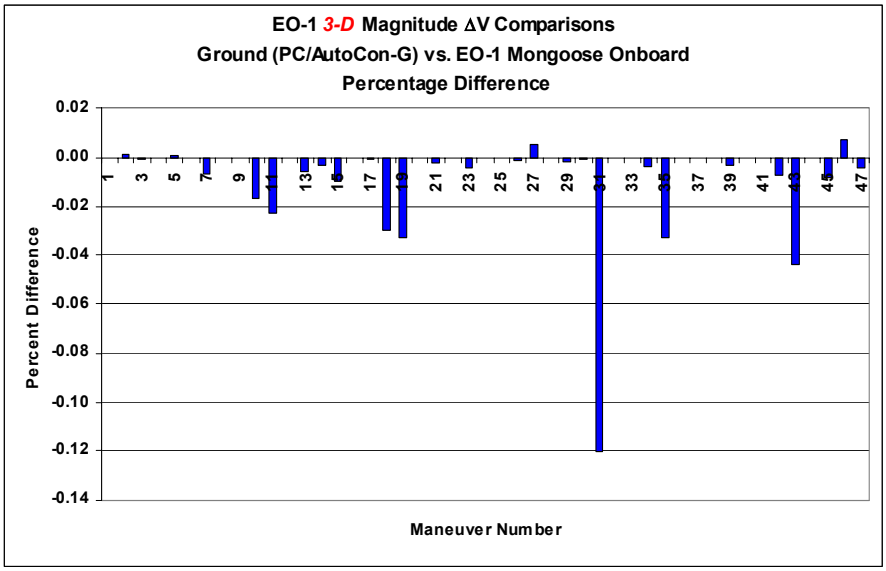

Figure 10. Percentage Difference in 3-D Onboard and Ground ΔVs

Obviously the crosstrack component was the driver with the largest magnitude. The comparisons show only the total ΔV magnitude, as this was the only information available in EO-1 playback telemetry.

With the comparisons between the ground and operational onboard version of the EFF completed, a comparison between the original FQ algorithm code and the onboard system was then performed. This comparison was done for only the first FQ targeted maneuver of each maneuver scenario. The state data from the playback telemetry was input into a MATLAB™ script with the FQ algorithm computing the maneuver without any propagation. Figure 11 shows the difference as a percentage respectively for the 3-dimensional ΔV and an alongtrack ΔV. The alongtrack ΔV was represented in the MATLAB™ script by using a local-vertical local horizontal coordinate system based on the input states that are comparable to the EO-1 nominal attitude for maneuvers. The resulting ΔV difference gave a mean of 0.0727 cm/s and a standard deviation of 0.348058 for the 3-D and a mean of -0.03997 cm/s and a standard deviation of 0.278402 for the alongtrack. The mean percentage difference was 0.003 for the 3-D and 0.006 for the alongtrack. These results showed excellent comparisons.

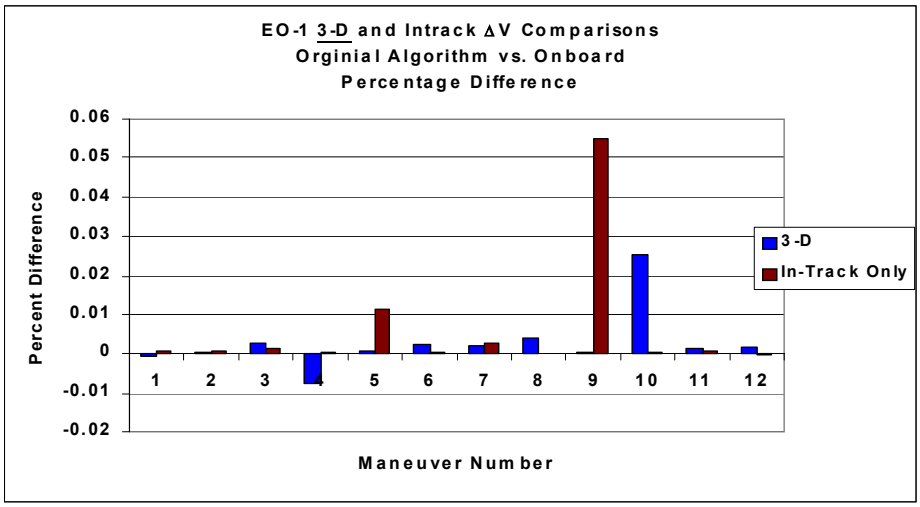

Figure 11. Percentage Difference in Original Algorithm and Onboard

3.2.4 Functional Propagation Comparisons

The FQ algorithm is dependent upon the generation of the target and desired states. These states were propagated onboard using a Runge-Kutta 4/5 with an 8x8 Geopotential model and a Jacchia-Roberts atmospheric drag model. The accuracy of the computed ΔV is dependent upon the accuracy of these propagated states. For EO-1, the states were propagated forward 1.5 orbits to compute the target state and then propagated 1.5 orbits backward to compute the desired state. As the desired state incorporated the longest propagation duration with a restart, a comparison was made in the onboard and ground states. Figure 12 shows the position component and magnitude differences for six maneuver plans. Figure 13 shows the velocity differences. The maximum difference observed was 1.35 meters in the y-component of position and 1.4 cm/s in the velocity z-component. These small differences are still being investigated, but are believed to be due to the integration methods and performance of the EO-1 computer. The mean and standard deviations for position and velocity are listed in Table 3.

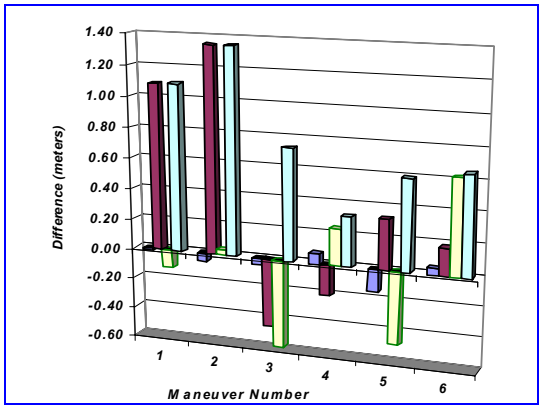

Figure 12. Targeter Orbit Propagation Position Difference

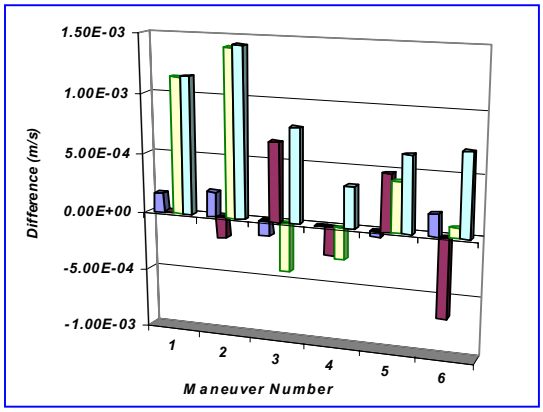

Figure 13.Targeter Orbit Propagation Velocity Difference

Table 3. Propagation Mean and Standard Deviation for Desired State Computation

	X	Y	Z	Magnitude
Position Mean (m)	-0.02279	0.38221	-0.04550	0.79088
Position StDev (m)	0.07676	0.70684	0.45024	0.36886
Velocity Mean (m/s)	0.00007	0.00001	0.00040	0.00084
Velocity StDev (m/s)	0.00014	0.00049	0.00074	0.00039

3.2.5 Autonomous Maneuver Validation Results

A total of ten maneuvers were planned and validated in the manual, semi-autonomous, and fully autonomous mode with seven reported herein. All were used to plan a formation flying maintenance maneuver with the semi-autonomous and autonomous mode generating commands onboard that were used onboard as well. The commands generated onboard in the fully autonomous mode were placed in the absolute time sequence with other spacecraft commands at approximately 12 hours before the maneuver execution. The locations and epochs of these maneuvers were chosen to meet the EO-1 orbit and science requirements in response to Landsat 7 maneuvers or to an EO-1 maneuver to maintain formation. The results presented in Tables 4 and 5 showed that there was excellent agreement between the onboard system and the ground validation system. Tables 4 and 5 present the maneuver mode and absolute ΔV difference and absolute percentage difference in the quantized and three-axis maneuvers. Table 4 gives results for the quantized maneuvers. Note that the percent error of the first ΔV computed from the FQ algorithm ($\Delta V1$) range from 0.00011% to 1.569%, the larger difference resulting from variations in the input target and desired states after propagation.

The effects of the quantization were more noticeable on the second maneuvers. That is, the velocity-matching maneuver was more prone to maneuver residuals. The residual of the first maneuver was "carried" over to the second maneuver, which also was quantized. The software was modified to account for most of this discrepancy in the computation of velocity matching. For example, the larger residuals of the second velocity matching ($\Delta V2$) were mostly due to a 1-second quantization of a very small velocity-matching maneuver. This difference was due to the onboard system yielding a maneuver duration near the midpoint that rounded down (up) while the ground system rounded up (down). Table 5 provides the comparisons for the three-axis ΔVs required to align EO-1 directly behind Landsat 7 and involves all three ΔV components of radial, alongtrack, and crosstrack. The ΔVs for these maneuvers ranged from 2.8 m/s to a maximum of 15.6 m/s. Again the comparisons were excellent with the range of percentage differences from the ground system at nearly zero to 0.66%. Additionally, a comparison was performed against the original algorithm, with excellent results as the percentage differences for all but one were under 0.005%. It should be noted that the last two autonomous maneuvers were computed using the GPS state that was filtered in the EFF system for an improved solution.

Table 4. Quantized Maneuver Comparisons

Date	Mode	Onboard ΔV1 (cm/s)	Onboard ΔV2 (cm/s)	Ground ΔV1 (cm/s)	Ground ΔV2 Difference (cm/s)	% Diff ΔV1 vs. Ground %	% Diff ΔV2 vs. Ground %
11/14/02	Auto (GPS)	4.162500	1.8500000	0.0000045	0.2313035	.00108159	12.502891
06/27/02	Auto (GPS)	5.359500	4.3387000	-0.000003	0.2552395	-0.0005535	5.8828571
06/07/02	Auto	4.9854078	0.0000000	0.0000001	0.0000000	0.00015645	0.00000000
05/17/02	Auto	2.4376271	3.7919202	0.0000003	0.0000002	0.00111324	0.00053176
05/10/02	Semi-Auto	1.0831335	1.6247106	0.0000063	-.0026969	0.05852198	-14.2361365
05/03/02	Semi-Auto	2.3841027	0.2649020	0.0000000	0.0000000	0.00011329	0.00073822
04/26/02	Semi-Auto	5.2980985	1.8543658	-0.0008450	-0.0002963	-1.56990117	-1.57294248
04/08/02	Manual	2.1915358	5.2049883	0.0000004	-0.0332099	0.00163366	-0.00022414
04/05/02	Manual	3.5555711	7.9318735	-0.0000003	-0.0272687	-0.00081327	3.57089537

Table 5. Three-Axis Maneuver Comparisons

Mode	Onboard ΔV1 (m/s)	Ground ΔV1 Difference (cm/s)	3-axis ΔV1 vs. Ground %	Algorithm ΔV1 Difference (cm/s)	3-Axis ΔV1 vs. Algorithm %
Auto (GPS)	2.8334	1.1222786	3.960890	-0.5083689	-1.794201
Auto (GPS)	8.45107	68.3310	-8.085483	.0462777	-0.005475
Auto	10.8468	-0.0005441	-0.0000502	0.0003217	0.0000297
Auto	11.8633	0.0178726	0.0015066	-0.0101756	-0.0008577
Semi-Auto	12.6416	0.0311944	0.0024677	0.0091362	0.0002867
Semi-Auto	14.7610	0.1888158	0.0127932	0.0000000	0.0001196
Semi-Auto	15.3797	-0.2526237	-0.0164231	-0.0633549	-0.0045164
Manual	15.5790	10.4109426	0.6682668	-0.0117851	-0.0007565
Manual	15.4749	0.0018465	0.0001193	-0.0307683	-0.0021934

3.2.6 Inclination Maneuver Validation Results

Beyond meeting requirements to maintain the EO-1 - Landsat 7 formation, The EFF also was used to plan an inclination maneuver. This maneuver was planned in a semi-auto mode using state vectors. The maneuver was also planned after the execution of the EO-1 inclination burn was completed as the Flight Operations Team needed to focus on maintaining the EO-1 orbit with respect to Landsat 7 and was iterating the ground-planned maneuver. Using the same state input information as the ground plan, the inclination maneuver was computed onboard using EFF with a script set to meet inclination targets. This script was a modification of the routine formation flying script, which specified that the maneuver was to be performed at the node crossing. The targeting tables were also updated to select a single maneuver, versus two maneuvers as was normally the case to meet eccentricity requirements. The EFF-planned maneuver was successfully computed and gave a duration of 114 seconds for an applied ΔV of 0.2389m/s and a quantized ΔV of 0.239m/s. The Flight Operations Team ground results gave a 115-second maneuver.

3.2.7 Propagation Comparisons for Autonomous Maneuvers

As with the functional validation, a comparison of the propagated states used in computing the targeted and desired states was made. Figure 14 shows the comparisons of the inertial positions (x, y, and z) for the target and desired states. These states were computed using the same models as discussed in the

functional validation. The largest difference can be seen in the columns marked 10-12. These differences occurred in the first semi-auto and last manual mode maneuvers. All the differences were less than 500 meters in all components with a standard deviation of less than 177 meters and less than 50 meters if the largest difference was excluded. Even so, these variations contributed to the differences between the onboard algorithm and the ground. The intervals of propagation for these states were 13 hours for the manual maneuver modes and less than 2 hours for the semi-autonomous or fully autonomous mode.

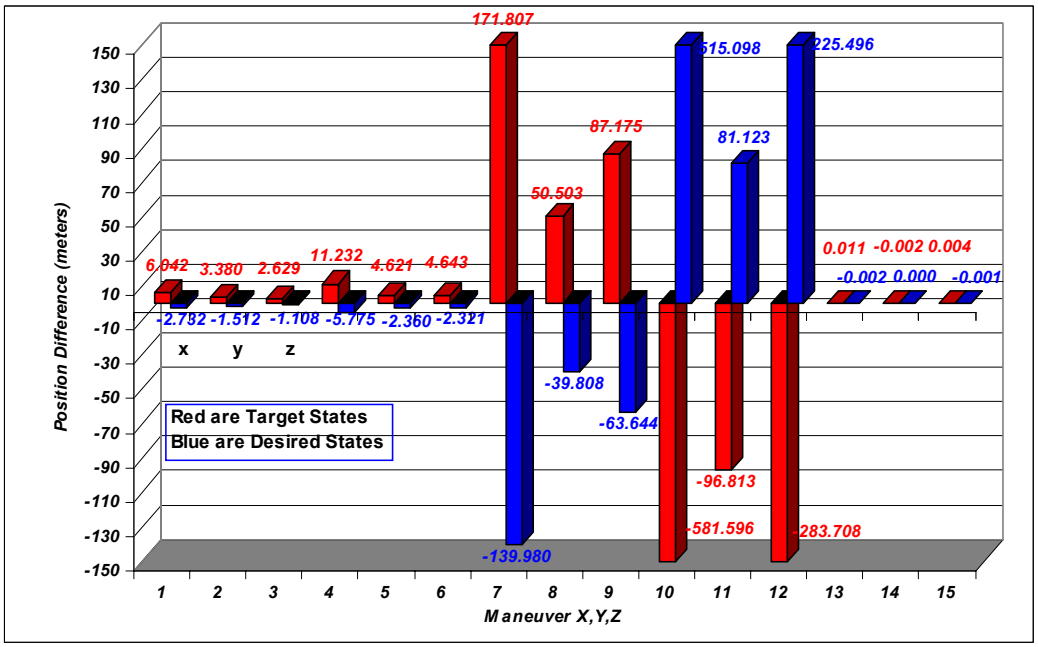

Figure 14. Target and Desired Propagation Position Differences

3.2.8 Independent Performance Assessment: EO-1 Formation History of Relative Motion and Keplerian Orbit Parameters

The following relative motion and Keplerian parameter plots are taken from the definitive ephemeris of EO-1 and Landsat 7 orbit determination process as an independent check to verify that the formation requirements of 450km with a tolerance of 75km (+/- 42.5 km yields 407.5 km to 492.5km) and the ground track of +/-3 km were meet. Additionally, one can observe that the relative eccentricity and SMA of the frozen orbit eccentricity was also maintained as a result of the formation flying maneuvers. Figures 15 and 17 show the general formation flying evolution of the alongtrack and radial components presented in a Landsat 7-centered rotating coordinate system with the radial direction (ordinate) being the difference in radius magnitude and the alongtrack direction (abscissa) being the arc between the position vectors.

Figure 16 shows the effect on the mission groundtrack made by the formation-flying maneuver and that it met NMP requirements. The figure shows both EO-1 and Landsat 7 groundtracks as an offset from the exact world reference grid. The time span is over the duration of the formation flying demonstration of five months from February 2001 to June 2001. At the beginning of the demonstration, EO-1 maneuvers only occurred in response to Landsat 7 maneuvers as the formation cycle where EO-1 exceeded the front of the control box was not completed before a Landsat 7 maneuver was required. Figure 17 shows the alongtrack separations over the demonstration duration. Figure 18 shows the SMA evolution in which one can see the effects of the differential ballistic properties of each spacecraft. Figures 19 and 20 show the frozen orbit eccentricity and argument of periapsis. The data for these plots was generated independently

from the formation flying system and further shows that the formation flying demonstration was a success.

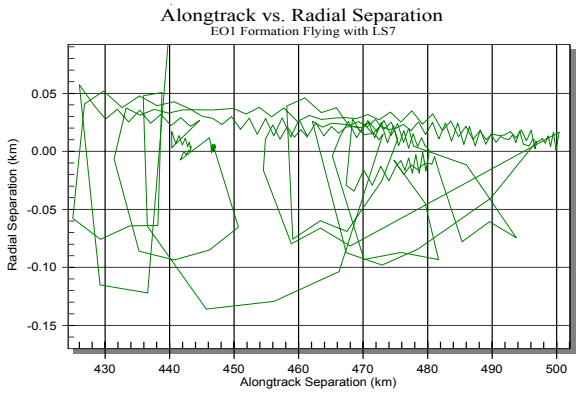

Figure 15. Relative Radial vs. Alongtrack Separation

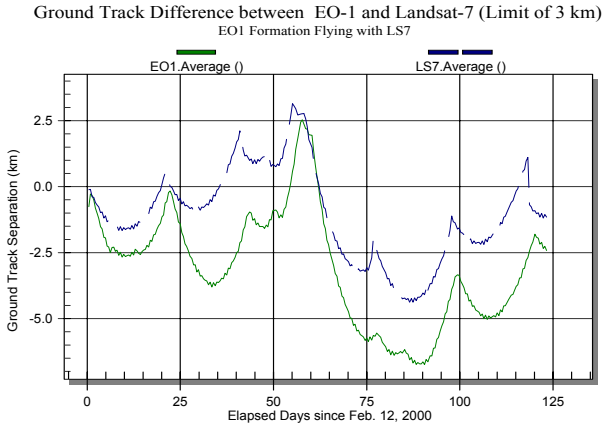

Figure 16. Ground Track Separation

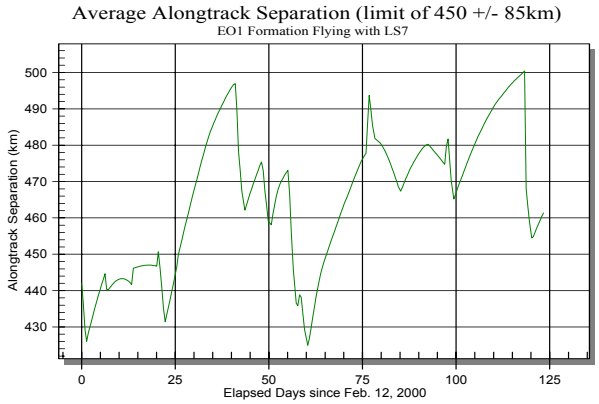

Figure 17. Alongtrack Separation vs. Elapsed Time

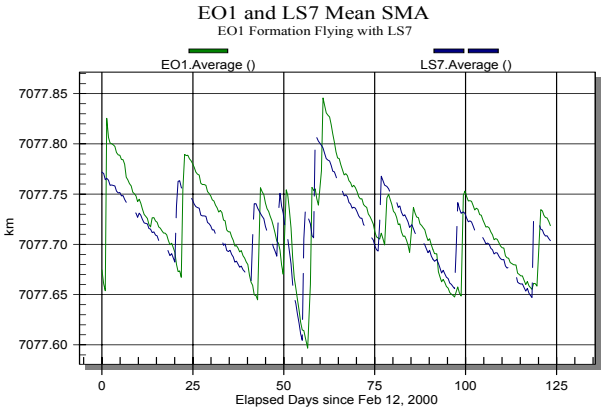

Figure 18. Semimajor Axis Profiles

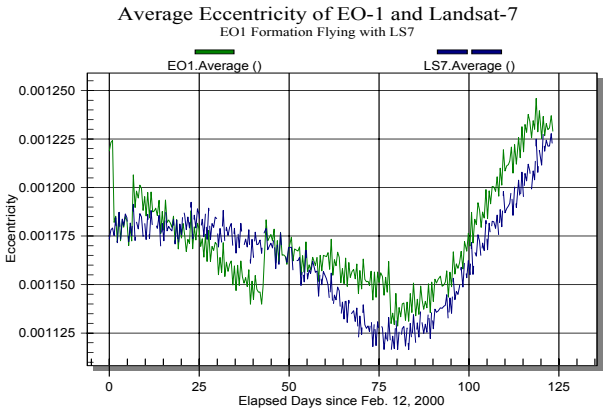

Figure 19. Eccentricity vs. Elapsed Time

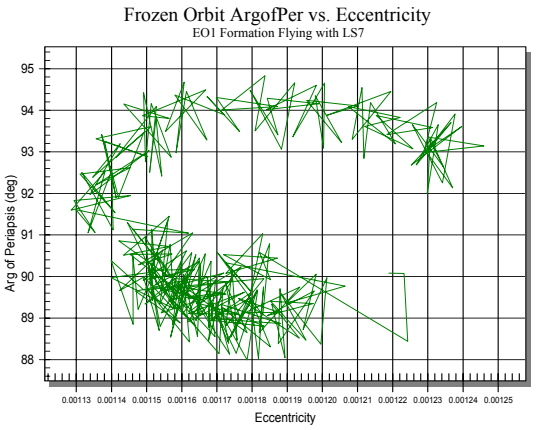

Figure 20. Eccentricity vs. Argument of Periapsis

3.3 EO-1 Safety

One of the major concerns of the EO-1 mission was to make sure that the autonomous maneuver system was as safe as possible. There was considerable concern, for example, that an autonomous system would cause the thrusters to fire for too long and jeopardize the mission. Several safeguards were created to

alleviate such concerns. These included a standard of 48 hours notice before any planned maneuver (the time length is adjustable) and a phased approach to autonomy. The 48-hour notice gave the ground time to review the planned maneuver before its execution. Figure 21 displays the "levels" of autonomy or phases and transition flow. These included a *monitor mode*, which allowed burn plans to be generated and reviewed, a *manual mode*, which allowed maneuvers to be predicted but not implemented, and a *semi-autonomous mode*, which allowed burns to be verified by the ground before execution. Even the autonomous mode could be interrupted by ground command. Also, the autonomous mode was limited to a specified number of burns before it automatically transitioned back to manual mode. A complete description of the safety modes is provided in Table 6.

In addition to AutoCon™-F's built-in safety features, the ACS limited all burns to 60 seconds or less. The stored command sequence also limited burn duration. Additionally, EO-1 had a watchdog timer to make sure no task, such as AutoCon™-F, exceeded CPU utilization, depriving other critical tasks processing time. Finally, the spacecraft had a safehold mode that could disable AutoCon™, if necessary.

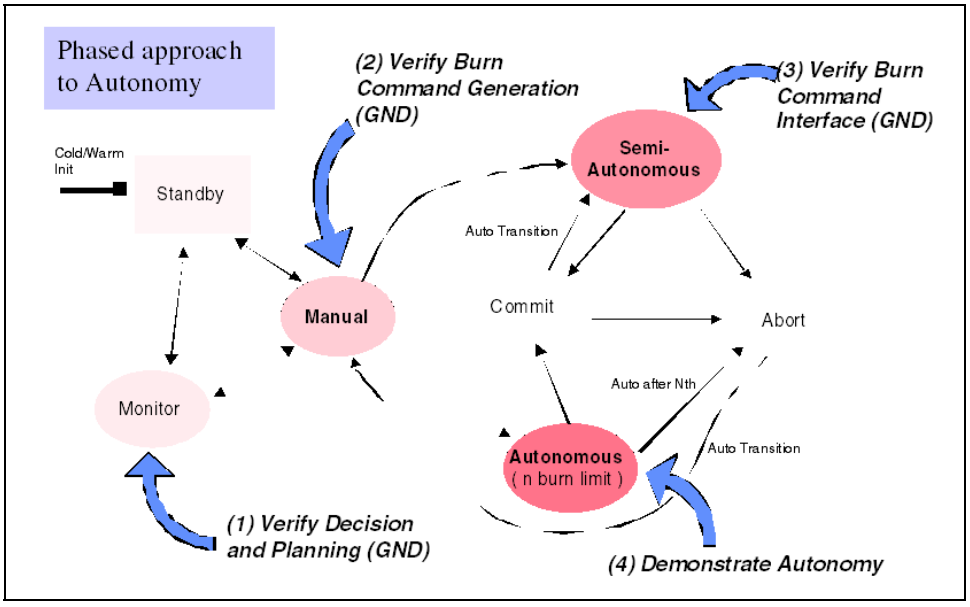

Figure 21. EO-1 Control Safety Modes

Table 6. Safety Modes

Standby • Pend on incoming data and send it to the bit bucket • Otherwise do nothing Monitor - (AutoCon™-F executes with maximum safety for S/C) • Invoke AutoCon™-F only • Report maneuver planning data to ground • No maneuver commands are generated Manual - (AutoCon™-F executes with ground as safety) • Generate maneuver commands (table loads) and send to ground only • All burns must be command from the ground in their entirety • Ground can loopback command from EFF telemetry if desired to execute burn

Semi-Autonomous - (Ground still in loop for go/no-go)
- Send maneuver commands (table loads) to the Stored Command Processor (SCP)
- Do not enable Absolute Time Sequence (ATS), Relative Time Sequence (RTS) in SCP
- Must switch to Commit Mode to allow loaded burn to execute
- Inaction will cause loaded burn to expire

Commit - (allow an EFF loaded burn to execute)
- Enable ATS and RTSs in SCP to permit loaded burn to be executed
- Required at least 2 hours before time of burn
- Autonomously switch to Semi Autonomous Mode upon completion

Abort - (abort an EFF loaded burn and clean up)
- Disable the ATS and RTSs in SCP to prevent execution of burn
- Clean up from any preparation for burn
- Autonomously switch to Manual Mode upon completion

Autonomous - (allow EFF to control the orbit)
- Closed loop orbit maintenance
- Use Commit Mode to switch back to Semi-Autonomous Mode and not abort a planned burn
- Ground can still monitor with 48 hour notice to burn
- Switch to Semi-Autonomous Mode after N burns. Safety for unattended operation

4. APPLICATION POSSIBILITIES

Recent discoveries in areas of climate changes, the Earth's space environment, weather predictions, space weather, and investigation of planets beyond the solar system have been drivers for the implementation of formation flying spacecraft. The diversity in the collection and measurement types of science data related to these areas has been expanding. These diverse areas have included:

- Viewing: temporal and spatial
- Collection types: large-scale apertures, multiple pointing, multiple in-situ, imaging
- Measurements types: spectrum, low-temperatures, fields
- Pointing accuracy: repeatability, stability, and control.

Using this technology, significant improvements in science can be made. This technology can be used to increase the number of instruments comprising the system and eliminate the restrictions imposed by the use of physical structures to establish, maintain, and control instrument separation. Formation flying enables extensive co-observing programs to be conducted autonomously without complex multi-instrument observatories and extensive ground support.

Another area of interest for formation flying is instrument calibration and correlation. Such as with the EO-1 technology demonstration, the EO-1 spacecraft was required to fly so to allow scene comparisons. Also better performing instruments and the associated increased data and miniaturization will drive onboard and formation requirements. The need for inter-satellite communications for data distribution and sensor cooperation is also becoming a reality with the advanced crosslink applications such as PiVoT and lower-power transceiver (LPT), sponsored and developed by GSFC, as is the need for unique geometrical views from various orbits (sensor webs and unique non-Keplerian orbits), including low-Earth orbits, elliptical orbits, libration point orbits, and unique orbits. Examples of these types of science needs include bi-directional reflectance flux measured by the Leonardo formation and x-ray detection using the Constellation-X mission.

There is also a need for image comparison and calibration achieved through closeness between spacecraft looking through the same atmospheric conditions at the same location. Landsat 7 and EO-1 maintained a

constant separation of 1 minute in time between each other while imaging the Earth. Challenges for these types of missions include autonomous maneuver control and autonomous navigation.

Future missions include formations such as those of the Constellation-X mission that utilizes multiple spacecraft to observe the same target black holes, galaxy clusters formation, and missing matter spacecraft deployed and maintained to a relative position and attitude in the same reference orbit. Here the reference orbit is a libration orbit. The challenges here are

- Autonomous maneuver control
- Autonomous navigation
- Collective attitude control
- Mission design

Another mission is Maxim, a formation of two spacecraft to image black holes. Maxim combines both constant separation and constant attitude/pointing. The detector spacecraft must "fly" around the optics continuously during an observation. The associated challenges include:

- Complex closed-loop autonomous maneuver control
- Autonomous absolute and relative navigation
- Precise attitude pointing and control
- Mission design
- Inter-spacecraft ranging and communication

5. TECHNOLOGY INFUSION OPPORTUNITIES

Upcoming GSFC Earth-orbiting mission infusion opportunities include any Earth-observing missions that require instruments to fly either in a formation or in a constellation. Also, another benefit to the NMP technology is the capability to perform simple orbit maintenance or SMA maintenance or orbital corrections such as inclination maneuvers, that is, to follow a desired SMA using any propulsion system effort, such as small, near-continuous maneuvers or less frequent large maneuvers to maintain orbital conditions. It can obviously be applied to a requirement to view the same or similar areas, such as those of the Earth-Observing System (EOS)-PM (renamed Aqua) train. Missions such as Picaso/Cena, COACH, and Leonardo are prime examples of the formation applications while the Global Precipitation Monitoring (GPM) mission is a prime example of autonomous orbit control. Also missions to libration orbits for interferometry such as Maxim, Constellation-X, SPECS, and others can use this technology directly with changes only for the differences in the dynamic properties of the orbit. As the EFF technology provides for three-axis control capability and uses AutoCon™ for the system-level executive and interfaces, it has potential applications to all missions having any autonomous control requirements.

6. LESSONS LEARNED

A description of the lessons learned from the integration, test, and validation of the EFF are addressed in some detail. Other topics that were identified to a lesser extent are discussed briefly below.

The implementation of an autonomous orbit control system facilitates lights-out operation, reducing costs and streamlining the spacecraft operations. While the motivation for implementing such a system is apparent, the challenge in this case is deriving the flight system from a ground system. Flight hardware is several generations behind ground hardware in processing power and available memory. Since AutoCon™ was originally designed as a ground system automation tool, a number of steps needed to be

taken to convert the system for use on-board the EO-1 satellite. These steps ranged from changing the interface to the system, to porting the system, to scaling the size and controlling the CPU processing. The steps in the conversion were straightforward to identify. Issues raised while implementing these steps are discussed below.

6.1 Interface Changes From Ground System to Flight System

The first step taken in making AutoCon™ flight-ready was to change the interface of AutoCon™. The design of AutoCon™ eased the potential complexity of this task. AutoCon™ was structured with a separate graphical user interface portion and a computational portion. Since the dialog boxes and windows would not be used onboard, that portion of AutoCon™ was replaced by the EFF interface. For user friendliness, the underlying data inputs to the core AutoCon™ system are ASCII files, with only a few exceptions. The flight system interface requires binary table inputs, control of the system through commands and operating modes, and the collection of results through telemetry.

The ground version of AutoCon™ was first modified to accept binary table files as an alternative input method to the ASCII file input. Binary tables were used because they were more compact and provided an accepted format for upload to the satellite. To accommodate all the inputs and maintain the flexibility of AutoCon™, 12 different inputs had to be converted to tables. Because EO-1 had a table size limit of 3000 bytes, the planetary input data had to be broken out into three separate tables.

To ensure table data integrity, AutoCon™ was implemented with the capability to validate the tables before use. Validation design required that all tables include three fields. The first two fields in the table were the table identifier number and the table size in bytes. These two fields were checked to ensure that the correct table was uploaded and accessed by AutoCon™. The last field in all tables was a checksum that was computed using a standard 32-bit cyclic-redundancy check (CRC) method.

Implementation of this interface upgrade provided AutoCon™-G the capability to ingest data in either the table format or the ASCII format for each input. This approach allowed for systematically testing each input table, and provided a complete code base with which to begin porting to the flight system environment.

6.2 Porting From PC/Windows NT to the Mongoose 5/5xWorks

AutoCon™-G was developed under Windows NT. The EO-1 flight system was built around a M-5 processor with VxWorks/Tornado Operating System. The system was built with the Tornado compiler, a derivative of GNU for this environment. Since an M-5 system was unavailable at the time of the initial port, the approach was taken to first port AutoCon™ to a similar system. AutoCon™ was ported to Hewlett-Packard (HP) UNIX and built with the GNU compiler. Although AutoCon™ was designed from the beginning to be portable, this step was the first real test of the system portability. The following issues were addressed during this port.

The first issues were the easiest to resolve technically. Because the UNIX environment is case sensitive, all references to files had to reflect the true file name. As a simple solution, all filenames and references to filenames were changed to lower case. Next, the system had to be built in the environment. The AutoCon™ system was originally designed as a collection of dynamic link libraries (DLLs) with an executable driver for the Windows environment. For the flight system, a single executable had to be built. This change was resolved by generating an all-inclusive makefile. The major porting issues came during the compiling and linking of the system.

The compilation issues with the largest scope were the use of the GNU string and math libraries. To resolve issues with the compiler-provided string class, a simplified string class specific to AutoCon™ was developed to overload the system string class. To support the change in the math library, AutoCon™

required its own definition of PI and the reevaluation of error handling for the math functions. For example, the fmod function provided in the Windows environment returned a value of zero when one of the two passed arguments was zero. In the flight environment, a +NAN (not a number) was returned from the same method if a zero was passed as the second argument.

Other compilation issues were associated with compiler limitations from the ANSI (American National Standard Institute) standard. One such limitation was the return of a value of an indexed array, Figure 22.

Original Code

```
return array_value[index];
```

Fixed Code

```
float temp_value = array_value[index];
return temp_value;
```

Figure 22. Resolution to Compiler Limitation

The largest obstacle to porting to the flight system was the restriction on dynamic memory allocation. While the object-oriented design of AutoCon™ is based on the creation of objects at run time, the flight operating system forbade the use of dynamic memory allocation. To overcome this hurdle, AutoCon™-F was fitted with a memory manager. The memory manager contained in its data segment a 1.5-MB block of space, and overloaded the C++ new and delete operators to manage the use of the space for AutoCon™. A number of redesigns were implemented before the memory manager operators would compile without conflicts with other parts of the flight system.

6.3 Size Reduction

The next challenge was to fit the AutoCon™ system into the available space on-board the satellite. Since AutoCon™ was originally designed as a ground system, the size of the system was not a major concern. Although AutoCon™ is a relatively small Windows system, it exceeded the memory limitations for flight code. When first ported to the UNIX environment as a single executable, the AutoCon™ executable size was over 7MB. The spacecraft requirement was to get AutoCon™ under 500kb in the flight environment.

Figure 23 shows the AutoCon™-F size history after the first build in the M-5 environment, which included an initial attempt at size reduction. The first five months included seven builds that were focused solely on reducing the size of the system. Once the size requirement was achieved, subsequent builds focused on modifying the capabilities of AutoCon™-F to support the flight system interfaces and mission requirements. As new capabilities were added, the code size was reevaluated and reduction efforts were implemented.

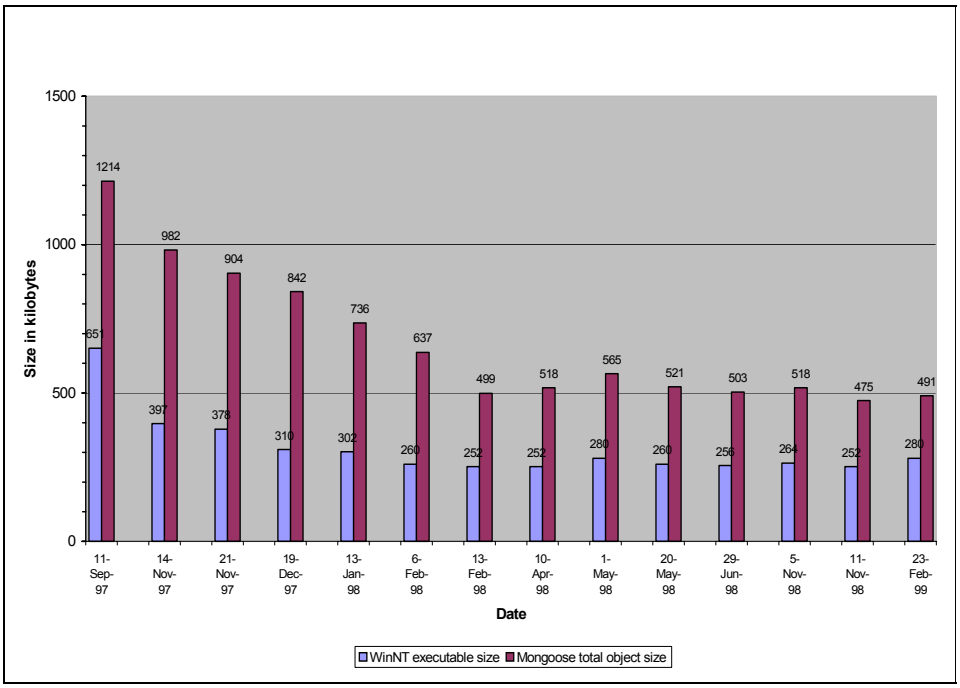

Figure 23. AutoCon™-F Size History

The first step was to remove unnecessary capabilities. Since AutoCon™ is object-oriented, this simply entailed removing whole objects from the build of the system. The original system contained 256 classes that defined the objects. The first two builds removed a total of almost 160 classes. Most of these classes were related to the calculation of event data that was not necessary for flight requirements. To remove additional objects, AutoCon™ was modified to use the math functions of the ACS in place of its own math classes. The final core AutoCon™-F system included 85 C++ classes.

While removing the objects offered a significant reduction, the system size still exceeded the spacecraft requirement. The next step in code reduction was to eliminate code methods from within the remaining classes. Methods associated with file input and debug output were obvious candidates for removal, as were methods required by these. Methods associated with unnecessary coordinate transformations and flight regimes outside of the EO-1 orbit were also omitted.

Another size reduction technique explored was compiler flag settings. Figure 23 shows a large size reduction in February 1998 between the sixth and seventh builds. The primary difference between these two builds was that the flag for compiling with debug was turned off for the seventh build; it had been activated for all previous builds.

During subsequent size reduction activities, other methods were implemented to reduce the size of the system. One technique was to collapse the inheritance of derived classes into their base class. This technique proved to provide minimal savings at approximately one kilobyte per collapse. Another technique used, which provided significant savings, was the static allocation and initialization of arrays. Initialization methods in some classes were filling large arrays using direct element-by-element assignments. Two classes had a combined total of 943 elements initialized by direct assignment. The change in initialization saved over 36 kb of space.

6.4 Parsed CPU Execution

The EO-1 flight environment system required that the executing tasks use CPU time slices. The AutoCon™-F system received a CPU slice every two seconds. AutoCon™-F was expected to complete processing within a fraction of a second. If AutoCon™-F, or any task, failed to complete processing within 5 seconds, the flight system would perform a warm restart.

AutoCon™ was originally designed as a parsed execution system for UI messaging such that one script command executes and processing returns to the controlling UI system. AutoCon™-F simply retained this design for the interface with the flight system. AutoCon™-F operates through the sequential execution of scripted commands. Using the original design, AutoCon™-F executed one command per time slice. The commands however took varying amounts of time to execute. Some commands used little CPU processing while others exceeded the allotted time slice. The commands that exceeded the time slice were segmented to complete a portion of the command, return processing to the main system, and continue with the next part of the command segment at the next time slice. To make best use of the available time slice, the commands that used very little processing time were identified and grouped with the next command within the same time slice.

Near the final stages of development of AutoCon™-F, a new capability was required to provide burn durations and burn start times on the whole second. The modification was minor, requiring only a few lines of additional code. As a result of the change, AutoCon™-F consistently used too much CPU time to perform targeting. The cause of the CPU over-utilization was traced to the customer-supplied fmod math function. The purpose of the fmod function was to return the remainder of one number divided by another. The function was implemented to perform successive subtractions and evaluations until the desired result was achieved. Because the use of fmod in the targeter had a very large number being divided by a relatively small number, the number of execution cycles to complete the processing was unreasonable. To ensure proper CPU use, the fmod function was replaced in AutoCon™-F with a less CPU intensive algorithm.

6.5 Issues During Testing

Upon successful compilation of the AutoCon™-F system in the flight environment, testing began. The test approach was to define a series of benchmark tests using AutoCon™-F in the Windows NT environment. The same tests were then duplicated in the flight environment and the results were compared. The flight environment tests were designed to exercise data table uplink, telemetry downlink, commanding, as well as computational accuracy. Initially, the results between the Windows NT and the M-5 were unacceptably different.

AutoCon™-F is required to propagate two spacecraft states (EO-1 and Landsat 7) into the future and plan any maneuvers required during that time to maintain the formation. The propagation differences between the benchmark NT result and the flight environment result were 540 meters root mean square (RSS) after 36 hours of propagation. The tools for debugging this problem on the flight environment were limited.

The investigation of this difference required embedding debug statements throughout the code. Since the largest perturbation on the spacecraft, besides the mass of the Earth, is drag, the drag model was investigated first. This choice was correct. Something in the code was causing the model to return an atmospheric density of zero without returning a processing error. The problem was found to be in a conditional statement in the Jacchia-Roberts drag model class, where a variable was set to the result of a function call and then tested. While the code complied with ANSI standards, the compiler did not handle the syntax properly. The problem continued even when the variable assignment was removed from the conditional. The problem was fixed when an interceding function call (*taskDelay*) was added. The original code and modified code syntax is provided in Figure 24.

```
Original Code
        if((*istat = jaccwf(a1_time, new_file, &geo)) != 0)
        {
                return 0.0;
        }

Fixed Code
        *istat = jaccwf(a1_time, new_file, &geo);
        taskDelay(1);
        if(*istat != 0)
        {
                return 0.0;
        }
```

Figure 24. Fix to Compiler Problem

After the problem with the atmospheric drag modeling was found, there were still approximately 36 meters of propagation difference using the full force model. To investigate this difference, separate results were produced for each force in the force model. This force-by-force testing showed that all forces modeled except for geopotential produced small (sub-meter) errors, but the largest discrepancy was related to the effects of the moon. Further tests revealed that, on the M-5, after the initial calculation of the moon's position the moon's position remained static, while the epoch was being advanced.

Inspection of the code shown in Figure 25 revealed the same type of structure found in the drag problem: the calculation and testing of a value in a conditional statement. This statement was broken apart, but did not correct the problem. Further analysis revealed that the error was caused by a chained assignment. The correct path was being executed, but the variables were not properly updated, causing the subsequent test to incorrectly bypass recalculating the position of the sun and moon. The correction required breaking the chained assignments into separate statements. Chained assignments were subsequently broken up in all other parts of the code even though they were not currently experiencing problems.

Once the compiler issues were resolved, the comparisons between the Windows NT and the M-5 results agreed.

```
Original Code
ABOOL ASolarSystem::GetSunMoonPosition(FSIZE * s)
{
        if (fabs (s[0] - mjt) > timeTolerance ) {                    // recalculate
        sunCalculated = moonCalculated = FALSE;                      // must recalculate
        mjt    = s[0];                                               // calculate new times
}
if (sunCalculated == FALSE) {                                        // sun not yet calculated
         sun[0] = mjt;
        moon[0] = mjt;
        if ( planetaryTable.CalculateSunMoonVectors(sun,moon) == FALSE )
                return FALSE;
}
memcpy(&s[1],&sun[1] ,sizeof(FSIZE)*3);                              // copy sun
sunCalculated = moonCalculated = TRUE;
return TRUE;
}

Fixed Code
ABOOL ASolarSystem::GetSunMoonPosition(FSIZE * s)
{
if (fabs (s[0] - mjt) > timeTolerance ) {                            // recalculate
```

```
        sunCalculated = FALSE;
        moonCalculated = FALSE;                          // must recalculate
        mjt    = s[0];                                   // calculate new times
}
if (sunCalculated == FALSE)  {                           // sun not yet calculated
        sun[0] = mjt;
        moon[0] = mjt;
        if ( planetaryTable.CalculateSunMoonVectors(sun,moon) == FALSE )
                return FALSE;
}
memcpy(&s[1],&sun[1] ,sizeof(FSIZE)*3);                  // copy sun
sunCalculated = TRUE;
moonCalculated = TRUE;
return TRUE;
}
```

Figure 25. Fix to Another Compiler Problem

6.6 GPS Data Smoother

Ensuring an accurate input state to AutoCon™-F is crucial for lights-out operation. On EO-1, the GPS TENSOR™ software using a Kalman filter processes raw GPS data that consists of pseudo-range and Doppler measurements. Orbital states obtained from the GPS TENSOR™ have RMS position and velocity errors of 35.7 m and 5.2 cm/s, respectively (ref. 4). The requirement for an AutoCon™-F input state for the GSFC algorithm is that the errors in radial position and velocity be no larger than 5m and 2 cm/s, respectively. Thus, the GPS TENSOR™ solution alone is not adequate, and an additional stage of optimization must be provided. This stage has been implemented as a discrete fixed interval data smoother, which uses the Rauch, Tung and Striebel (hereafter, RTS) algorithm (ref. 5). While the solutions provided by the GPS TENSOR™ software are actually processed, they will be referred to in this section as *measurements*. These measurements are assumed to be from a converged Kalman filter solution.

The RTS algorithm is itself based on an iterative Kalman filter. Each of N measurements is collected and processed, while storing at each interval the filter's a-priori state and state error covariance matrix (current iterate's state and error covariance advanced to the current measurement epoch), the state transition matrix, and the optimized (a-posteriori) state and error covariance. These stored quantities are referred to collectively as the *data arc*. Measurements are processed at one-minute intervals, and a number of measurements, corresponding to somewhat more than one orbital period (N~120), are required for optimal state determination. The RTS algorithm is a backward iteration through the data arc, taking the initial smoothed state to be the final filter computation in the forward sweep. In this way, the original filter estimate is updated to provide an improved smoothed estimate based on all the measurement data collected. Unlike the situation with the Kalman filter (forward sweep) updates, the smoothed state error covariance is not required in the computation of the smoother estimates in the backward iteration.

While, in principle, the RTS algorithm allows for computation of smoothed estimates at any or all of the interior intervals, care must be taken to ensure that the estimate is meaningful. Thus, the following convergence criteria have been established for the smoothed estimate, and the backward sweep is terminated if one of them occurs:

- A diagonal element of a current iterate's error covariance matrix is not positive definite
- Position or velocity error diverges, i.e., a current iterate's RSS position or velocity error exceeds that of the previous iterate.

The first criterion is required because of the decoupling of the error covariance computation from the state estimate. The second criterion simply ensures that the state estimate converges in the usual sense. Two additional convergence criteria may be turned on in operation, causing the backward iteration to terminate when:

- A current iterate's RSS position or velocity error exceeds a commandable tolerance
- A current iterate's state element estimate is out of bounds.

6.6.1 Kalman Filter

The Kalman filter underlying the smoother is adapted from the GPS Enhanced Orbit Determination Experiment (GEODE)-lite software (ref. 6). The state elements consist of the three components of position and velocity, together with a drag term coefficient, and the GPS receiver timing bias and bias rate. The dynamic quantities are computed in Mean of J2000 coordinates. The measurement model assumes that the point solution measurements are converted to Earth-Centered-Earth-Fixed (ECEF) coordinates. This model has been enhanced to incorporate the velocity components that are also provided by the GPS TENSOR™ software, and to incorporate the Jacchia-Roberts drag model, required by AutoCon™-F.

The Kalman optimization is performed in two stages: (1) propagation of the current estimate to the new measurement epoch, and (2) updating of the a-priori state due to the new measurement data. The original GEODElite code was modified so that AutoCon™-F performs the propagation step. AutoCon™-F time conversions are also called by the modified GEODElite code.

6.6.2 Smoother Integration and Testing

The GPS data smoother is implemented as an AutoCon™-F object, and appears to the user just as any other object does. It is created with an associated spacecraft object that holds the final smoothed state. This smoothed state is the input state for maneuver planning. As with other phases of the flight code conversion, the smoother used a parsed execution scheme, i.e., the two stages of Kalman optimization are handled in separate execution cycles. In the back sweep, only a small number of iterations are performed in each cycle to prevent CPU task overloading.

As with other facets of AutoCon™-F, the smoother is table-driven, and may be monitored via telemetry packets. Tables exist to control the operational modes of the smoother, the process and measurement noise characteristics, the data arc characteristics (e.g., queue size), as well as to provide data required for performing coordinate transformations. New data may be uploaded to the spacecraft via these tables in order to alter the secular behavior of the smoother. The uploaded tables are validated and checked for integrity in the same way as for the other AutoCon™-F tables. Similarly, the smoother measurement acquisition cycle and final state may be monitored dynamically through telemetry packets.

In testing scenarios, the definitive smoother state has proven to be nearly always better when compared to a reference ephemeris than the Kalman estimate. Indeed, despite the lag in time between the Kalman and smoothed estimates, multi-day propagation of the smoothed solutions are comparable to, and often much better than, the propagated Kalman estimates, producing the desired behavior. This result may be related to the fact that the definitive angular momentum and specific energy are better determined for the smoothed states than for the Kalman filtered states.

6.6.3 Operational Modes

The smoother is designed to provide a state estimate with minimal ground support under normal operating conditions. The initial seed state is derived from the most recently updated GPS TENSOR™ data. This can be changed, however, by uploading the seed state and error covariance to the EFF. Usually it is desirable to allow the smoother to complete the back sweep until one of the convergence criteria is encountered, as this process provides the best definitive solution. However, the smoother may be

commanded to provide a solution at a fixed amount of time (i.e., *lag interval*) before the final measurement epoch. Setting the lag interval to 0, for example, will tell the smoother to provide only the iterated Kalman filter estimate. Indeed, the smoother may even be commanded to provide the GPS TENSOR™ state alone.

7. CONTACT INFORMATION

David Folta
NASA Goddard Space Flight Center
Code 572
Greenbelt, MD 20771
Email: dave.folta@gsfc.nasa.gov
Phone: 301-286-6082

John Bristow
NASA Goddard Space Flight Center
Code 583
Greenbelt, MD 20771
Email: jbristow@pop500.gsfc.nasa.gov
Phone: 301-286-3647

8. SUMMARY

EFF, developed under direction of NASA's GSFC, is an advanced technology that allows satellites to autonomously react to each other's orbit changes quickly and efficiently. It was successfully validated onboard EO-1 in a predefined formation-flying relative position with Landsat 7 for more than a year. EFF uses an onboard software package called AutoCon™, a tool for planning and executing autonomous orbit control maneuvers while analyzing and resolving mission constraints. AutoCon™ software incorporates advanced AI technologies, such as fuzzy logic and natural language scripting, to resolve multiple conflicting constraints and provide automated maneuver planning. EFF can support all Earth-orbit mission needs and is especially useful for missions requiring frequent maneuvers and resolving complex conflicting constraints. The AutoCon™ architecture of EFF also supports distributive processing which can be critical for formation control missions. It is completely object-oriented and can easily be enhanced by adding new objects and "events."

A technology critical to EFF is the FQ formation-flying algorithm developed by GSFC aerospace engineers Dave Folta, John Bristow, and Dave Quinn. Onboard validation of this algorithm proved that the EO-1 formation flying requirements could be easily met. To ensure the accuracy of the onboard algorithm, several comparisons were performed against both original analytical calculations and ground-based numerical computations. The FQ algorithm was validated by direct inputs of the initial states taken from the onboard system. The ΔV results agreed to the millimeters/sec level for the numerical tests, which included the effects of propagation. During operations, EFF used the FQ algorithm to target new trajectories, place and size the maneuvers, and execute spacecraft burns autonomously. The FQ formation-flying algorithm is an innovative technology that can be used on-board in a closed-loop design to meet science and mission requirements of all low-Earth-orbiting formation flying missions.

The EFF validation effort established the following:

- A demonstrated, validated fully non-linear autonomous system for formation flying
- A precision algorithm for user-defined control accuracy
- A point-to-point formation flying algorithm using discretized maneuvers at user-defined time intervals
- A universal algorithm that incorporates
  - In-track velocity changes for semimajor axis control
  - Radial altitude changes for formation maintenance and eccentricity control (frozen orbit control)

    - Crosstrack velocity changes for inclination control or node changes
    - Any combination of the above for maintenance maneuvers
- Proven executive flight code
- A system that incorporates multiple constraint checking using fuzzy logic for maneuver planning and control
- A natural language script input
- Shared codebase for ground and flight systems
- Isolated flight code interfaces to the command and data handling systems
- The option of single or multiple maneuver computations
- The use of multiple/generalized navigation inputs
- Generation of attitude (quaternion) information required of the spacecraft to meet the ΔV components
- Maintenance of combinations of Keplerian orbit requirements (eccentricity, arg of perigee, etc.)

9. CONCLUSIONS

The EFF system flown onboard EO-1 was composed of AutoCon™'s technologies, the maneuver-targeting FQ algorithm, and all interfaces that enable frequently maneuvering spacecraft missions. All of these components are especially critical to formation flying and constellation missions. EFF enabled EO-1 to fly one minute behind and in the same ground track as Landsat 7 to allow paired scene comparisons. It enabled EO-1's operations team to meet these tight mission constraints without exceeding projected costs. Although EFF is an extremely powerful tool, which contains some very complex concepts and capabilities, it was designed for ease of learning and use. EFF is critical to NASA's ability to successfully support the tight requirements of formation flying. It is the enabling technology that provides quick and accurate response, of a "following" formation-flying satellite, to orbit variations and maneuvers of an independently controlled "target" satellite, such as EO-1 is to Landsat 7. This capability does not exist in any other maneuver-planning software system. EFF's targeting capabilities allowed EO-1 to adjust to any changes in Landsat's orbit and re-establish the formation. EFF's responsiveness and ease of use provided the operations support team with the accurate maneuver data required to successfully meet the challenging demands imposed by this first true controlled formation flying mission. EFF is very robust in that it supports autonomous operations for relative separation control, demanding three-axis control for inclination and non-Keplerian transfers, and in that it can be applied to any orbit about any celestial body.

10. TECHNICAL REFERENCES

[1] F. Bauer, D. Quinn, K. Hartman, D. Folta, and J. Bristow, "Enhanced Formation Flying Experiments For The New Millennium Program Earth Orbiting (EO)-1 Mission - Challenging Technology Program Management," AIAA, 5/97.

[2] J. Bristow, D. Folta, K. Hartman, and J. Leitner, "A Formation Flying Technology Vision," AIAA-2000-5194, February 2000.

[3] D. Folta and D. Quinn, "A Universal Three-axis Method for Controlling the Relative Motion of Multiple Spacecraft in Any Orbit," Proceedings of the AIAA/AAS Astrodynamics Specialists Conference, August 10-12, Boston, MA.

[4] D. Quinn and D. Folta (1996) *Patent Rights Application and Derivations of Autonomous Closed Loop 3-Axis Navigation Control Of EO-1.*

[5] R. Battin, (1987) *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA Education Series, Chapters 9 and 11.

[6] R. Sperling, (1997) *AutoCon User's Guide Version 3.0, February 1998,* a.i. solutions Inc., Greenbelt, MD 20770.
[7] D. Folta and A. Hawkins, "Preliminary Results of NASA's First Autonomous Formation Flying Experiment: Earth Observing-1 (EO-1)," AIAA GSFC Flight Mechanics Symposium, Greenbelt MD, June 2001.
[8] D. Folta, J. Bristow, G. Dell, and A. Hawkins, "Results of NASA's First Autonomous Formation Flying Experiment: Earth Observing-1 (EO-1)," AIAA/AAS Astrodynamics Specialist Conference, Monterey, CA, August 2002.
[9] Matlab, *The Math Works, Inc. Users Guide*, 1995.
[10] AutoCon/ACS ICD.

Acronym List

ACS	Attitude Control System
AI	Artificial Intelligence
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
ATS	Absolute Time Sequence
ATS	Absolute Time Sequence
C&DH	Command & Data Handling
CPU	Central Processing Unit
DLL	Dynamic Link Library
ECEE	Earth-Centered-Earth-Fixed
EFF	Enhanced Formation Flying
EO-1	Earth Observing-1
EOS	Earth Observing System
FQ	Folta-Quinn
GEODE	GPS Enhanced Orbit Determination Experiment
GN&C	Guidance, Navigation & Control
GPS	Global Positioning System
GSFC	Goddard Space Flight Center
HP	Hewlett-Packard
I&T	Integration and Test
LPT	Lower Power Transceiver
M-5	Mongoose 5
NMP	New Millennium Program
OD	Orbit Determination
RSS	Root Mean Square
RTS	Rauch, Tung, and Striebel; Relative Time Sequence
SCP	Stored Command Processor
SMA	Semi-major Axis
STOL	Standard Test Operating Language
TDRS	Tracking and Data Relay Satellite
TDRSS	Tracking and Data Relay Satellite System
TONS	TDRSS Onboard Navigation System
VC	Virtual Channel

Appendix-A. The Folta – Quinn Formation Flying Algorithm

The Folta-Quinn (FQ) algorithm is a new technology that is based on mathematics derived by Battin and adapted to the formation-flying problem. This technology allows full closed-loop maneuver autonomy onboard any spacecraft rather than the tedious and costly operational activity historically associated with ground-based operations and control.

FQ Algorithm Description

The FQ algorithm for formation flying solves the position maintenance problem by combining a modified Lambert's two point boundary value problem and Battin's 'C*' matrix with an autonomous system developed by a.i.-solution, Inc. of Lanham Maryland, called AutoCon™. The algorithm enables the spacecraft to execute complex three-axis orbital maneuvers autonomously. Figure 1A illustrates the basic sets of information required for the EO-1 formation targeting as it is incorporated into AutoCon™. The FQ algorithm is well suited for multiple three-axis burn scenarios but is more easily explained using a two-burn, co-planar example for clarity.

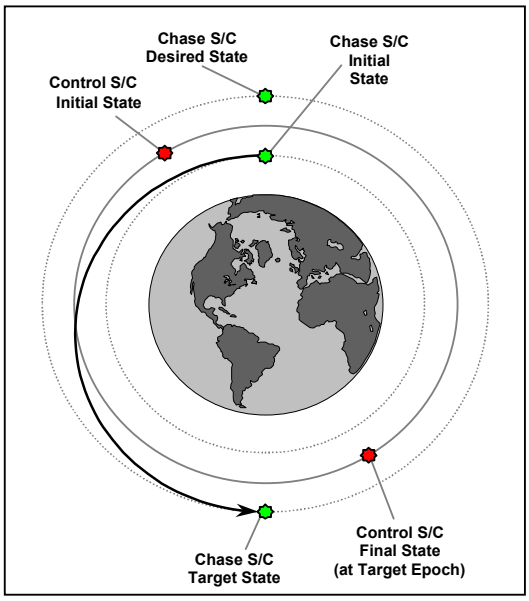

Figure 1A. FQ Algorithm Inputs for Formation

The formation-flying problem in this example involves two spacecraft orbiting the Earth. Landsat 7, the control spacecraft, orbits without performing any formation flying maneuvers. EO-1, the chase spacecraft monitors the control spacecraft, and performs maneuvers designed to maintain the relative position imposed by the formation requirements. In this example, the goal of the formation-flying algorithm is for EO-1 to perform maneuvers that cause it to move along a specific transfer orbit. The transfer orbit is established by determining a path (in this case a Keplerian path) that will carry the EO-1 spacecraft from some initial state, ($\mathbf{r}_0$, $\mathbf{v}_0$), at a given time, t_0, to a target state, ($\mathbf{r}_t$, $\mathbf{v}_t$), at a later time, t_t. The target state is found to be one that will place EO-1 in a location relative to Landsat 7 so as to maintain the formation. A desired state is also computed. This is accomplished by back propagating the target state to find the initial state that EO-1 would need at time t_0 for it to achieve the target state at time t_t without executing a maneuver. This back propagation of the target state gives rise to the desired state, ($\mathbf{r}_d$, $\mathbf{v}_d$) at time t_0. The initial state can now be differenced from the desired state to find:

$$\begin{pmatrix} \delta r \\ \delta v \end{pmatrix} = \begin{pmatrix} r_0 - r_d \\ v_0 - v_d \end{pmatrix}$$

STM Formulation

The FQ algorithm uses state transition matrices, described below, for the calculation of the maneuver ΔV. Selecting initial conditions prescribed at a time t_0 so that the state at this time has all zero components except the jth term, which is unity, a state transition matrix, $\Phi(t_1,t_0)$, can be constructed such that it will be a function of both t and t_0 and satisfies matrix differential equation relationships[5]. The initial conditions of $\Phi(t_1,t_0)$ are the identity matrix.

Having partitioned the state transition matrix, $\Phi(t_1,t_0)$ for time $t_0 < t_1$,

$$\phi(t_1,t_0) \equiv \begin{bmatrix} \phi_1(t_1,t_0)\phi_2(t_1,t_0) \\ \phi_3(t_1,t_0)\phi_4(t_1,t_0) \end{bmatrix}$$

We find the inverse may be directly obtained by employing symplectic properties

$$\phi^{-1}(t_1,t_0) \equiv \phi(t_0,t_1) \equiv \begin{bmatrix} \phi_1(t_0,t_1)\phi_2(t_0,t_1) \\ \phi_3(t_0,t_1)\phi_4(t_0,t_1) \end{bmatrix}$$

$$\phi^{-1}(t_1,t_0) \equiv \begin{bmatrix} \phi^T{}_4(t_1,t_0)\phi^T{}_2(t_1,t_0) \\ \phi^T{}_3(t_1,t_0)\phi^T{}_1(t_1,t_0) \end{bmatrix}$$

Where the matrix $\Phi(t_0,t_1)$ is based on a propagation forward in time from t_0 to t_1 and is sometimes referred to as the navigation matrix, and $\Phi(t_1,t_0)$ is based on a propagation backward in time from t_1 to t_0, and is sometimes referred to as the guidance matrix. We can further define the transition matrix partitions as follows:

$$\begin{aligned} \tilde{R}^*(t_0) &\equiv \phi_1(t_0,t_1) \quad \tilde{R}\ (t_1) \equiv \phi_1(t_1,t_0) \\ R^*(t_0) &\equiv \phi_2(t_0,t_1) \quad R\ (t_1) \equiv \phi_2(t_1,t_0) \\ \tilde{V}^*(t_0) &\equiv \phi_3(t_0,t_1) \quad \tilde{V}\ (t_1) \equiv \phi_3(t_1,t_0) \\ V^*(t_0) &\equiv \phi_4(t_0,t_1) \quad V\ (t_1) \equiv \phi_4(t_1,t_0) \end{aligned}$$

Substituting yields the following useful identities:

$$\begin{bmatrix} \tilde{R}^*(t_0) & R^*(t_0) \\ \tilde{V}^*(t_0) & V^*(t_0) \end{bmatrix} = \begin{bmatrix} V^T(t_1) & -R\ (t_1) \\ -\tilde{V}^T(t_1) & \tilde{R}\ (t_1) \end{bmatrix}$$

Where the starred quantities are based upon a guidance matrix and unstarred quantities are based on a navigation matrix. If a reversible Keplerian path is assumed between the two states, one should expect the forward projection of the state from t_0 to t_1 to be related to the backward projection of the state from t_1 to t_0. When the fundamental matrices C and C* are defined as

$$\tilde{C}^* \equiv \tilde{V}^* \tilde{R}^{*-1} \text{ and } C^* \equiv V^* R^{*-1}$$

We find the following:

$$\tilde{C}^* \equiv \left.\frac{\partial v_0}{\partial r_0}\right|_{r1=cons\tan t} \text{ and } C^* \equiv \left.\frac{\partial v_0}{\partial r_0}\right|_{r1=cons\tan t}$$

so that $C^*\delta r = \delta v_0$ becomes the velocity deviation required at time t_0 (as a function of the measured position error δr at time t_0) if the spacecraft is to arrive at the reference position r_1 at time t_1 (with arbitrary

velocity). Recalling that the starred quantities were obtained based on the guidance matrix, the sympletic property allows them to be computed based on a navigation projection. It can therefore be shown that

$$\left[C^*(t_0)\right] = \left[V^*(t_0)\right]\left[R^*(t_0)\right]^{-1} = \left[\widetilde{R}^T(t_1)\right]\left[-R^T(t_1)\right]^{-1}$$

Applying a universal variable formulation of the closed-form state transition matrix, the relevant state transition matrix submatrices are computed.[4,5] The expressions for F, G, F_t and G_t are derived from the Gauss problem of planar motion; K is a quantity derived from the Universal Variable (U) formulation.[5] These variables are dependent upon each other in their formulation, i.e. U(6) is dependent upon U(4) and on intermediate variables related to the classic f and g series. The target and desired states, r_d ,v_d, r_t , and v_t, are computed from the propagated states. μ is the universal gravitational constant. R and R are then defined from the target and desired states as:

$$\mathbf{R}(t_t) = \frac{|\mathbf{r}_d|}{\mu}(1-F)\left[(\mathbf{r}_t - \mathbf{r}_d)\mathbf{v}_d^{\mathrm{T}} - (\mathbf{v}_t - \mathbf{v}_d)\mathbf{r}_d^{\mathrm{T}}\right] + \frac{K}{\mu}\left[\mathbf{v}_t\mathbf{v}_d^{\mathrm{T}}\right] + G\left[\mathbf{I}\right]$$

$$\widetilde{\mathbf{R}}(t_t) = \frac{|\mathbf{r}_t|}{\mu}\left[(\mathbf{v}_t - \mathbf{v}_d)(\mathbf{v}_t - \mathbf{v}_d)^{\mathrm{T}}\right] + \frac{1}{|\mathbf{r}_t^3|}\left[|\mathbf{r}_t|(1-F)\mathbf{r}_t\mathbf{r}_d^{\mathrm{T}} + K\mathbf{v}_t\mathbf{r}_d^{\mathrm{T}}\right] + F\left[\mathbf{I}\right]$$

From these variables and sub-matrices, the C* matrix is computed as follows:

$$R^*(t_0) = -R^T(t_t)$$

$$V^*(t_0) = \widetilde{R}^T(t_t)$$

$$C^*(t_0) = V^*(t_0)\left[R^*(t_0)\right]^{-1}$$

The expression for the impulsive maneuver follows immediately:

$$\Delta\mathrm{V} = \left[\mathrm{C}^*(t_0)\right]\delta\mathrm{r}_0 - \delta\mathrm{v}_0$$

At each step in the process, the next control point on the reference path can be examined and back-propagated along a Keplerian path to determine small differences between spacecraft position and velocity on the reference path and determine which Keplerian path would intersect the reference path at the next control point. These differences are then fed into the propagator via the state transition matrices to determine the incremental ΔV required to get the spacecraft to the next control position on the reference trajectory. At the conclusion of the maneuver window, a final burn is required to match the velocity required to maintain the new Keplerian trajectory. One can use single or multiple maneuvers to achieve the target condition. For EO-1's orbit a long, iterative window requiring many small burns is not necessary and therefore ΔV maneuvers resemble a Hohmann transfer.