Autocommander – A Supervisory Controller for Integrated Guidance and Control for the 2nd Generation Reusable Launch Vehicle

J. E. Fisher, D. A. Lawrence[1] and J. J. Zhu
Avionics Engineering Center
School of Electrical Engineering and Computer Science
Ohio University
Athens, OH 45701 – 2971

An Extended Abstract Submitted to the 2002 AIAA GNC Conference
January 2002

ABSTRACT

This paper presents a hierarchical architecture for integrated guidance and control that achieves risk and cost reduction for NASA's 2nd generation reusable launch vehicle (RLV). Guidance, attitude control, and control allocation subsystems that heretofore operated independently will now work cooperatively under the coordination of a top-level autocommander. In addition to delivering improved performance from a flight mechanics perspective, the autocommander is intended to provide an autonomous supervisory control capability for traditional mission management under nominal conditions, G&C reconfiguration in response to effector saturation, and abort mode decision-making upon vehicle malfunction. This high-level functionality is to be implemented through the development of a relational database that is populated with the broad range of vehicle and mission specific data and translated into a discrete event system model for analysis, simulation, and onboard implementation. A Stateflow Autocoder software tool that translates the database into the Stateflow component of a Matlab/Simulink simulation is also presented.

[1] Corresponding author, e-mail: dal@ohio.edu, Tel. 740-593-1578, Fax. 740-593-0007

Introduction

Flight mechanics technology is perhaps of the highest abstraction in an RLV system in the sense that, unlike hardware and software, it is algorithmic in nature. Yet it is vitally important to vehicle safety, reliability and operational costs, as inadequate and inflexible G&C algorithms can cause unrecoverable departure in flight leading to Loss of Vehicle (LOV) or Loss of Crew (LOC) in the presence of severe adverse flight conditions, or otherwise nonfatal hardware or software failures. Indeed, history has witnessed spectacular failures due to the lack of understanding in vehicle dynamics and control in the early days of rocketry. In fact, the evolution of space exploration and exploitation has been a major driving force for the development of modern G&C theory and technology. In turn, the flourish of modern G&C theory in the late 50's and early 60's had rewarded the aerospace community with an enabling technology that contributed to the success of Apollo moon landings and routine space flight by the 1st generation RLV, *viz.* the Space Shuttle. It has been so successful that, during the more than 100 Shuttle flights to date, there has not been an LOV or LOC incident caused by guidance or control.

However, the success of Shuttle G&C technology comes at the price of stringent launch and entry windows that lead to excessive launch and entry rescheduling. The large number of I-loads and the difficulty in validating guidance and control gains for a particular mission contribute significantly to the high operational cost of the Shuttle. This is especially true when this technology is applied to the more challenging 2nd generation RLV architectures. The Shuttle G&C technology has two major deficiencies. First, the G&C algorithms are gain-scheduled which, albeit may give good performance under nominal conditions, create large I-loads, and are difficult and time consuming to design, tune and validate, yet not very robust in off nominal flight conditions. Second, it lacks autonomous control reconfiguration and abort guidance. Another problem that is common to most if not all G&C subsystems is the segregated guidance and control design approach, which limits the performance and robustness of the overall G&C subsystem. These problems contribute significantly to the high risk and cost of Shuttle operation.

During the recent X-33 advanced G&C (AG&C) research program at NASA MSFC, the MSFC baseline G&C algorithms along with several alternative G&C algorithms developed in-house or by contractors were tested with high fidelity 6-DOF Monte Carlo dispersion simulations under realistic flight conditions [1]. The baseline algorithms are based on Shuttle technology, whereas the alternatives are based on more advanced nonlinear and time-varying G&C techniques. While none of the designs aced all simulation scenarios, some of the more advanced AG&C algorithms did show improvements over the baseline designs in performance, mission success rate, robustness in the presence of adverse flight conditions and mis-modeling of vehicle dynamics, as well as reduced developmental and routine operational costs [2]. However, it is alarming and disturbing that the best combination of the AG&C algorithms in the aforementioned simulation tests achieved only about 1 in 100 LOV for the X-33, a 2nd generation RLV technology demonstrator. The G&C algorithms alone need an order of magnitude improvement in safety and reliability to reach NASA's mission safety goal, let alone leave room for the possibility of other subsystem failures.

Compelled by the current state of affairs in RLV G&C technology, and armed with the experience gained in the R&D for the X-33 G&C and the latest achievements in G&C

theoretical research, the authors are part of a multi-university team developing advanced G&C technologies for the NASA 2nd generation RLV program. These technologies are cast in an integrated framework and include onboard trajectory generation for closed-loop ascent guidance [12] and adaptive entry guidance [13], autonomous abort guidance that relies on the aforementioned onboard trajectory generation capability, robust and scalable attitude control [14], direct fault tolerant control [15], closed-loop dynamic control allocation [18] with on line estimation of local attainable moment set [19], autonomous control reconfiguration [20]. A top-level *autocommander* discussed herein integrates guidance, attitude control and control allocation to facilitate autonomous fault handling. Also under development is a Matlab / Simulink / Stateflow based discrete event driven hybrid control system analysis, design and simulation software tool called the Stateflow AutoCoder [17]. A much needed stability verification technique for off-line and on-line stability assessment that includes robustness metrics for nonlinear, time-varying RLV flight control systems based on the concepts of Lyapunov exponents, Lyapunov's 2nd method, passivity approach, the small gain theorem, and the generalized gain margin and phase margin [16].

The G&C algorithms under development are based on the most mature algorithms developed in the X-33 AG&C program that are estimated at TRL 3-4. They will be brought to TRL 6 at the end of the 4.25-year project. The proposed G&C technologies are crosscutting in the sense that they do not rely on gain scheduling and therefore are largely trajectory independent and easily scalable to a full size RLV. It is ultimately planned to work with the selected 2nd generation RLV architect(s) in order to help ensure that the chosen 2nd generation RLV architecture will achieve NASA's goals of risk and cost reduction.

NASA's primary goals for the 2nd generation RLV are : (i) reducing ascent LOV risk to 1 in 1,000 missions, LOC risk to 1 in 10,000, and providing crew survivable abort capability through intact ascent abort, and possibly descent and landing abort; and (ii) reducing the recurring operational cost to $1,000 per pound of payload and the non-recurring costs in vehicle design and validation. The integrated G&C (IG&C) technology under development will achieve risk reduction by: (i) improved robustness of G&C algorithms, which reduces the risk of departure in adverse flight conditions or mis-modeling of the vehicle dynamics; (ii) utilization of the IVHM and the IG&C in autonomous abort guidance and control reconfiguration, which improves survivability in case of other subsystem failures. The cost reduction will be accomplished by: (i) improved robustness of the G&C subsystem to allow widening of the launch and entry windows; (ii) streamlined preflight operation owing to the small set of I-loads that are independent of trajectory and mission specifics, and are easily tuned and validated.

Autocommander - Overview

Performance, safety, and reliability of next generation reusable launch vehicles (RLVs) can be greatly enhanced through the effective integration of the flight management/control system, IVHM system, and vehicle/actuator dynamics under the supervisory control of a top-level *autocommander*. A hierarchical architecture for integrated G&C, depicted in Figure 1, will be capable of delivering improved performance, robustness, and fault-tolerance due to the coordinated operation of the RLV subsystems as controlled by the autocommander.

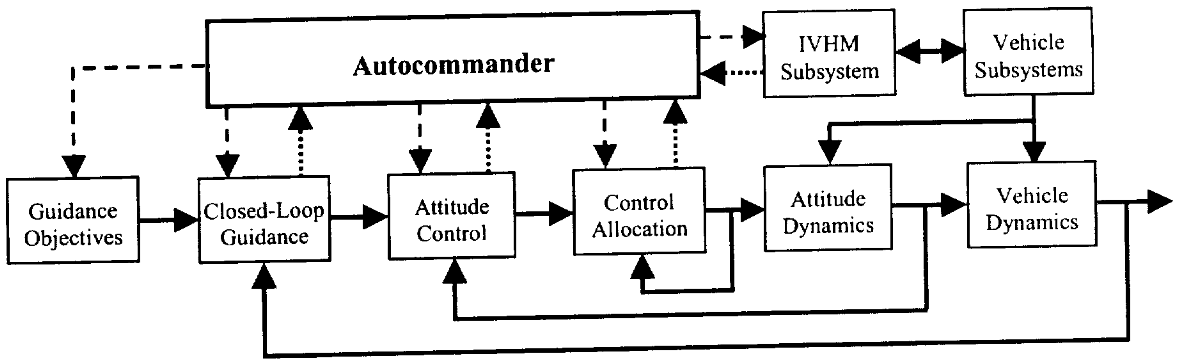

Figure 1. Integrated Guidance and Control Architecture for 2nd Generation RLV

Conventional G&C architectures constitute a multi-loop control system with interdependent causal effects. Segregated guidance and control law development, by ignoring such complex coupling, may be unable to deliver the robustness and fault-tolerance necessary for autonomous vehicle operation. For example, under aggressive guidance commands, adverse flight conditions or effector failures, the attitude controller may produce commanded torques based upon which the control allocation scheme would drive the effectors to saturation. In a segregated design, the controller would cut back its gain to avoid integrator windup. However, this reduced inner loop gain may cause the guidance loop, without knowing what is happening, to drive the attitude command even higher. This may cause saturation in the guidance loop, or lead to gross tracking errors that render the linearization based guidance and control laws invalid, thereby causing loss of stability. In contrast, in the integrated G&C architecture considered here, the autocommander, with information from all subsystems as reported by the IVHM system, can decide on the best course of action from a global perspective and command on-line adjustments to be made by any or all subsystems. Such a capability for system-wide on-line adaptation will provide improved robustness to variations in vehicle characteristics, improved fault-tolerance in the presence of severe failure modes, and mission-level decision-making in response to catastrophic malfunctions.

This integrated, hierarchical architecture, combining continuous-time vehicle and actuator dynamics, sampled-data guidance and control laws, an event-detecting IVHM system, and an event-driven autocommander capable of logical decision-making, forms a *hybrid system*. There has recently been a tremendous surge in research activity on hybrid systems in response to the realization that highly complex control systems naturally possess a hierarchical structure with continuous-variable, time-driven subsystems at the lowest level and logical, event-driven subsystems at the highest level. Researchers have proposed a variety modeling frameworks for hybrid systems that either augment conventional time-driven models (differential or difference equations) with event-driven dynamics, or augment conventional discrete event models (Petri nets or automata) with time-driven dynamics, or strive to be even more general. Analysis and design methods have accompanied the various modeling formalisms as exemplified in [4][5][6][7][8][9][10].

A hybrid systems approach holds significant potential to aid in the analysis, design, and simulation phases of IG&C technology development. For example, hierarchical architectures are often necessary in order to manage complexity in large-scale systems [9]. Various layers in the hierarchy are designed based on course or aggregated models of lower level subsystems. Here it is crucial that aggregation is performed in a hierarchically consistent fashion, which essentially means that constraints of lower level subsystems are adequately reflected in any simplified model so that high-level policies meeting high level objectives do not violate low-level constraints. The highest level in the hierarchy is often purely logical in nature and modeled by a discrete event system. The theory of discrete-event systems is now fairly mature and tools for analysis, design, testing, and optimization are widely available [3].

Autocommander - Functional Description

The autocommander is expected to autonomously perform three main high-level functions: (i) to act as mission manager under nominal conditions, (ii) to reconfigure the G&C subsystems under off-nominal conditions for improved fault tolerance, and (iii) to declare an appropriate abort mode in the event of an unrecoverable failure in coordination with on-line trajectory re-planning. The interrelationships between these main operational modes are captured in the high-level diagram in Figure 2. Currently absent from this figure is an indication of the mechanisms that cause appropriate transitions between these modes or states. It is clear even at this level of detail that equipping the autocommander with the necessary algorithms to distill information from various sources into decision-making logic and associated actions is a considerable challenge.

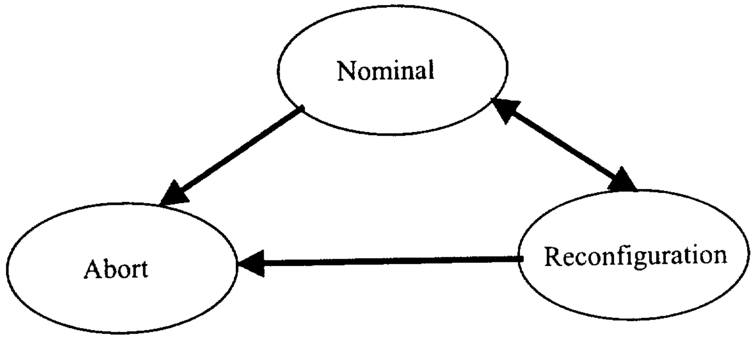

Figure 2 Autocommander top-level logical diagram

To meet this challenge, a key goal of the autocommander effort is the development of an automated process by which the vast body of vehicle specifications independently formulated by numerous subsystem design specialists is captured in a relational database and subsequently used to formulate discrete event system models of the autocommander's logical decision-making behavior. Vehicle information upon which autocommander decisions are to be based comes from a variety of sources such as an onboard performance predictor, G&C subsystems, vehicle health monitoring, navigation, and operator input. Information provided from these sources is, at least in part, of the form of continuous-valued, continuous-time signals. For example, the onboard performance predictor may independently utilize the on-line trajectory generation algorithms within the guidance subsystem or implement a faster-than-real-time simulation that produces performance prediction, the G&C subsystems may provide tracking error measurements, and the navigation subsystem may report various sensor measurements. In order to construct a discrete event system representation, all such continuous quantities must be quantized according to designer specified thresholds. This quantization leads to the definition of discrete events and states from which an automaton model can be constructed [3]. Here it is expected that this process will initially proceed in parallel for each information source leading to separate and conceptually simple automata for each. The next step involves the parallel composition (with appropriate synchronization) of the individual automata in order to represent their concurrent operation. Here the so-called curse of dimensionality is encountered. For instance, if each of the five individual automata has just four states, the combined automaton may have as many as $4^5 = 1{,}024$ states. Since it is expected that autocommander events can be characterized by simple and compound logical operations involving states of the individual automata, the parallel composition operation need not be performed explicitly, thereby avoiding the complexity issue.

The remainder of this section discusses the three main autocommander functions listed above are considered in greater detail, with an emphasis on discrete-event system representations.

Nominal Mission Manager

Here the autocommander is primarily responsible for coordinating the transitions between flight regimes under nominal conditions as indicated by the automaton or state machine depicted in Figure 3. In this representation, the ovals constitute automaton states and the directed arcs emanating from each state correspond to a feasible or active event for that state. The events that trigger the transitions between nominal mission phases, labeled generically as e_1 through e_7, are to be derived from vehicle specifications and mission parameters captured in the database. Also, within each mission phase, the autocommander must be capable of transitioning to a G&C reconfiguration mode under off-nominal conditions or recoverable failures and transitioning to an abort mode in response to a more serious situation. These capabilities are described in more detail below.

G&C Reconfiguration

Suppose that for different flight regimes the control effectors are designated as *primary* or *secondary* depending on their effectiveness and the desirability for using them. These designations are to be made during control allocation design and captured in the master database. Using the tools under development, the database information will be transformed into a discrete event system model given by an automaton of the form shown in Figure 4 with states and events exemplified by Tables 1 and 2. To illustrate the behavior of this automaton, consider State A which represents nominal usage of primary effectors. Active or feasible events for this state are r_1, primary effector commands within limits, and f_1, primary effector command saturation. While in State A, event r_1 (recovery) triggers a state transition to a nominal operational mode, corresponding to a state in the automaton of Figure 3. Alternatively, event f_1 (failure) triggers a transition to State B that corresponds to an adjustment in the control allocation logic in which the usage of primary effectors is reduced, counterbalanced with increased utilization of the secondary effectors. Events other than r_1 or f_1 are not active for State A and therefore do not trigger a transition.

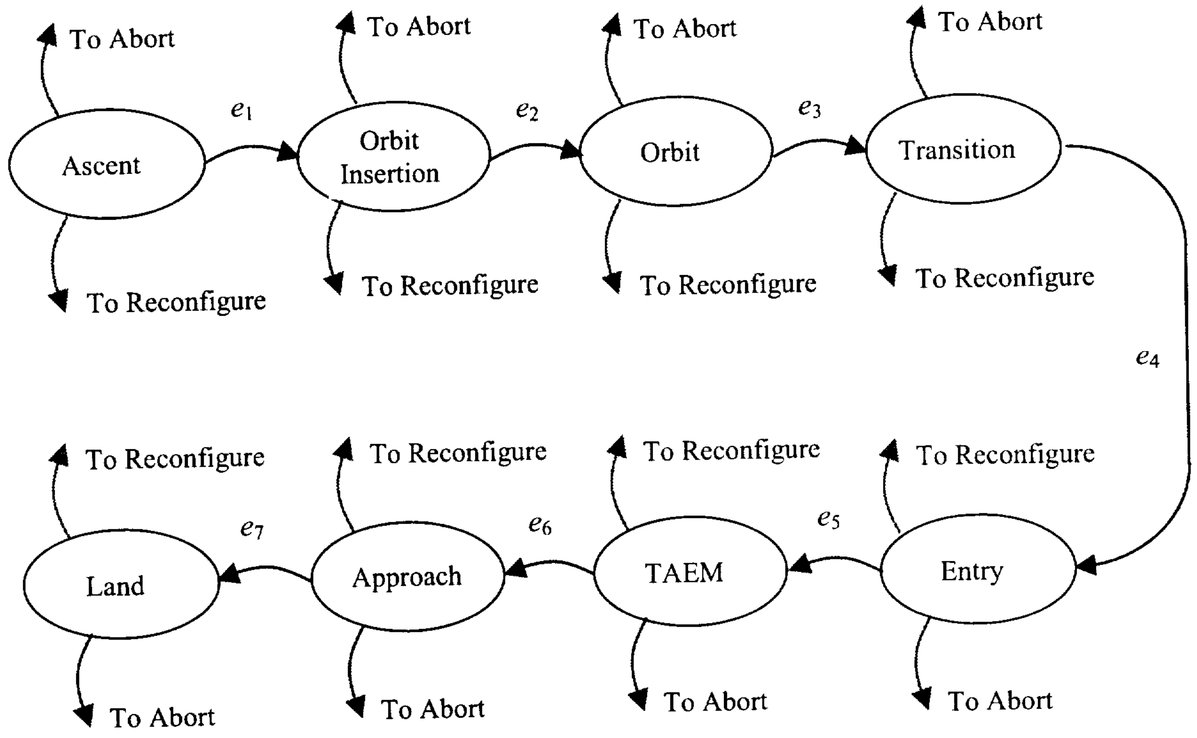

Figure 3. Automaton representation of nominal G & C modes

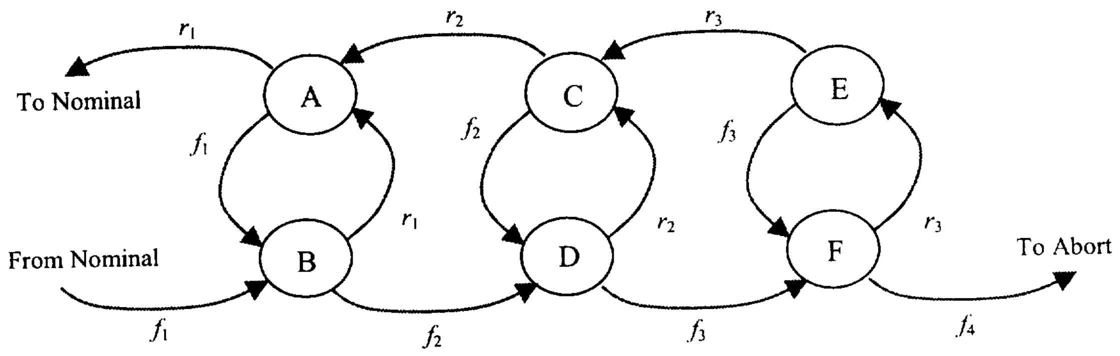

Figure 4. Automaton representation of G & C reconfiguration logic

Table 1. States of G & C Reconfiguration logic automaton

State Label	**Description**
A	Nominal usage of primary effectors
B	Decreased usage of primary effectors
C	Nominal attitude control bandwidth with nominal control mixing logic
D	Decreased attitude control bandwidth and/or alternate control mixing logic
E	Nominal guidance gain with nominal guidance law
F	Decreased guidance gain and/or alternate guidance law

Table 2. Events of G & C Reconfiguration logic automaton

Failure Events		**Recovery Events**	
Label	Description	Label	Description
f_1	Primary effector command saturation	r_1	Primary effector commands within limits
f_2	Secondary effector command saturation	r_2	Secondary effector ommands within limits
f_3	Torque command saturation	r_3	Torque commands within limits
f_4	Unrecoverable guidance command saturation		

Autonomous Abort

Depending on the vehicle architecture and flight regimes, different abort modes may exist when a need arises. Each abort mode can be modeled as a discrete state in an automaton consisting of a sequence of operational procedures such as failure confinement, fuel dump, preparation for crew bailout, etc, and a feasible abort trajectory and target (alternative orbit or landing site). For discussion purposes, we use the following generic abort modes as examples.

- Abort to Orbit (ATO): the vehicle will fly to an alternative orbit with at least 24 hours staying time, such as an 105 nm circular orbit. The ATO is used when the energy of the vehicle at MECO cannot reach the nominal orbit. This mode of abort consists of several submodes depending on the number of engine failure modes, e.g. single engine out, dual-engine out, etc. It is noted that engine failures in various combinations may have a distinct effect on flight mechanics; in that case, each distinct engine out condition constitutes a distinct engine out event. Each submode is a discrete state.
- Abort Once Around (AOA): the vehicle will fly around the earth once in order to better target a desired abort landing site, including the launch site or landing sites that are to the west of the launch site. The AOA is used when the MECO energy level does not allow the vehicle to reach the ATO orbit. Similar to ATO, the AOA consists of a number of submodes depending on the number of engine failure modes that have distinct effects on flight mechanics and the number of available landing sites.
- Down Range Abort (DRA): the vehicle will fly a sub orbital entry trajectory targeted at a landing site other than, and to the east of the launch site. (For this reason, it is also known as Transoceanic Abort Landing (TAL) for the Space Shuttle, which is routinely launched from the east coast.) The DRA is used when either the MECO energy level or vehicle condition does not allow the vehicle to fly once around, or when the desired time-to-landing is less than an hour. The submodes for DRA are a combination of the number of distinct engine failure modes and all alternative landing sites.
- Return to Launch Site (RTLS) Abort: during early stage of ascent, the vehicle may return to the launch site if an abort is necessary. The RTLS abort window exists from liftoff until the Negative Return (NE) time when the vehicle energy level and remaining fuel are such that RTLS maneuvers cannot be completed. The RTLS abort affords the shortest time-to-landing. The submodes for RTLS depend on the number of distinct engine failure modes.

Since each of the abort modes depends on the attainable energy level, which in turn depends mainly on the vehicle MECO velocity (since potential energy is negligible comparing to kinetic energy during launch), the window for each abort mode comes sequentially from liftoff with RTLS first, followed by DRA, AOA and ATO. Associated with each abort (sub) mode, there is a risk factor and a cost factor. These factors may vary with the vehicle condition and environmental conditions. For example, thermal stress during entry increases with the vehicle energy level. Thus, the risk factor would increase for a vehicle with thermal protection damage. However, from the flight control point of view, the risk factor decreases with the RTLS-DRA-AOA-ATO sequence, as the entry maneuvers will be less demanding with higher energy level at MECO. If the time-to-landing is of first priority, then the cost factor increases with the RTLS-DRA-AOA-ATO sequence. Otherwise, the cost factor may depend on whether the mission can be partially carried out, as may be the case with ATO, or the cost associated with each abort landing site.

In addition to the above intact abort scenarios, there may be cases where the vehicle system failure poses eminent danger to the crew or ground assets, or no feasible abort trajectories can be found. In these cases, the first priority is to fly the vehicle to a safe condition to bail out the crew. Once the crew is safely bailed out, the vehicle will either fly autonomously under the autocommander's control, or remotely controlled by the Ground Mission Control (GMC) to a safe location for possible unmanned landing or intentional crash/splash landing. The crash site should be selected to minimize loss of ground assets, minimize the damage to the vehicle or to facilitate recovery of the vehicle wreckage.

The following figure depicts the concept for the Autonomous Abort Automaton, where each node should be understood as representing a set of discrete states. Each state has a preloaded risk factor and cost factor that can be updated by the autocommander during flight based on vehicle conditions, flight phases, current weather conditions, etc. These factors can also be updated by the Crew in flight. These factors will be used by the autocommander in optimizing abort decisions.

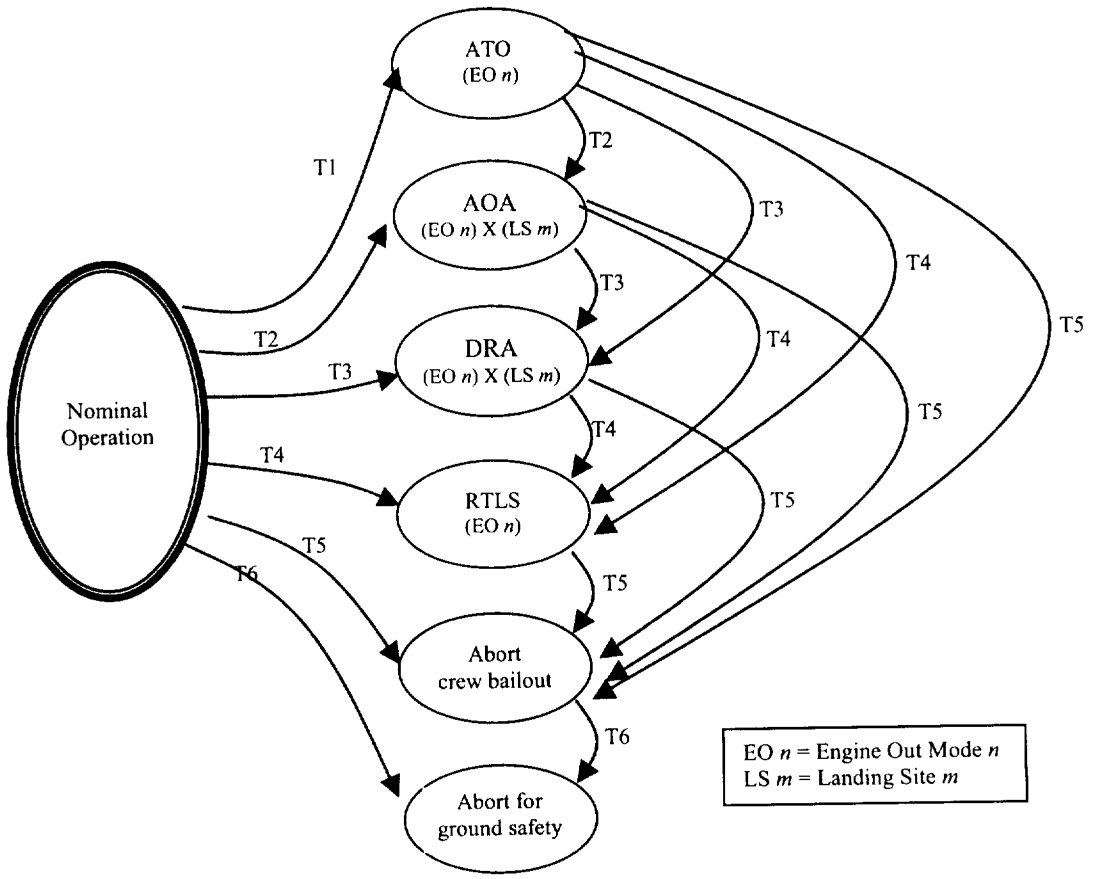

Figure 5. Autonomous Abort Automaton

The events and conditions that trigger an abort can be categorized into four classes:

- Vehicle health conditions, hereafter called Type V
- Flight mechanics performance, Type P

- Flight control capabilities, Type C
- Operational concerns, Type O

The occurrence of these events are detected and provided to the autocommander by the following four information sources, respectively:

- Type V: Integrated Vehicle Health Management (IVHM) System
- Type P: Onboard Performance Predictor (OPP)
- Type C: Direct Fault Tolerant Control (DFTC) System
- Type O: Predetermined optimization criteria and real time input from the Crew or Ground Mission Control (GMC) for decision making

Occurrence of a Type V event is based on diagnosis by the IVHM system, usually associated with a confidence level of occurrence and consequence of the event. A Type P event is flagged by the OPP, which consists of onboard computer simulations of continuous dynamics and discrete events. The former simulates the trajectory based on the current vehicle navigation and energy states, remaining fuel and propulsion capability, and physical and operational constraints. The latter simulates contingency scenarios as advised by the IVHM or crew and GMC. Type P events can also be associated with a confidence level and a risk factor. The DFTC, coordinated by the autocommander, provides Type C event information regarding degradation of flight control authority, onset of instability (departure or excitation of parasitic modes), and inference of control actuator/effector failure with a confidence level. Finally, Type O events are controllable events that influence the abort decision and selection of the abort strategy. Predetermined optimization criteria can be either mission independent, which are permanently stored in onboard flight computers, or mission dependent, which are day-of-launch I-loads. Parameters associated with these criteria, such as weighting coefficients and thresholds, can be altered and new criteria can be added by the crew and GMC in flight as needed.

Example events of each type and how they may be combined in selection of an abort strategy using an automaton are given in the Tables below. It is emphasized here that in general these events are vehicle or mission dependent.

Table 3. Example Type V Events of Launch Abort

Events	Description
V0	Vehicle healthy, with confidence level
V1	Main avionics cooling system down (still have backup)
V2	Main power bus down (still have backup)
V3	Cabin decompression, with level of severity
V4	Failure of one (say, out of five) flight computers
V5	Failure of two (out of five) flight computers
V6	One (say, out of three) engine out
V7	Two (out of three) engines out
V8	Three (out of three) engines out
V9	Thermal protection defects, with level of severity
V10	Eminent danger to Crew (vehicle may still be landable autonomously)
V11	Crew has bailed out
V12	Catastrophic vehicle system failure (intact abort negative)
V13	Eminent danger to ground assets
V*n*	Etc.

Table 4. Example Type P Events of Launch Abort

Events	Description
P0	Nominal orbit reachable with remaining fuel at current energy level, with confidence level (risk factor = 0)
P1	An alternative orbit (min. 24-hour stay) is reachable with remaining fuel at current energy level, with confidence level and risk factor
P2	Predicted MECO energy level allows for flying once around, with confidence level and risk factor
P3	Launch site is reachable with available energy and vehicle maneuverability, with confidence level and risk factor
P4	Landing site A is reachable with predicted MECO energy level and cross-range, with confidence level and risk factor
P5	Landing site B is reachable with predicted MECO energy level and cross-range, with confidence level and risk factor
P6	Landing site C is reachable with predicted MECO energy level and cross-range, with confidence level and risk factor
P7	Landing site D is reachable with predicted MECO energy level and cross-range, with confidence level and risk factor
P8	Crew bailout window available, with confidence level and risk factor
P9	Crew bailout window open, with confidence level and risk factor
P*n*	Etc.

Note:

1. The risk factors are evaluated based on the difficulty and mechanical/thermal stress of the required maneuvers for the trajectory, and the predicted weather conditions along the trajectory and at the landing site.

Table 5. Example Type C Events of Launch Abort

Events	Description
C0	Flight control system normal, with confidence level
C1	Loss of roll control authority, with level of degradation
C2	Loss of pitch control authority, with level of degradation
C3	Loss of yaw control authority, with level of degradation
C4	Unmodeled mode in roll, with estimated frequency, amplitude and damping factor
C5	Unmodeled mode in pitch, with estimated frequency, amplitude and damping factor
C6	Unmodeled mode in yaw, with estimated frequency, amplitude and damping factor
C7	Eminent departure in roll, with level of confidence
C8	Eminent departure in pitch, with level of confidence
C9	Eminent departure in yaw, with level of confidence
C*n*	Etc.

Table 6. Example Type O Events of Launch Abort

Events	Description
O0	No restriction on time-to-landing (Default)
O1	Required time-to-landing less than 60 minutes (See Note 2 below)
O2	Required time-to-landing less than 30 minutes (See Note 2 below)
O3	(Composite) confidence level acceptable (based on current acceptance threshold)
O4	(Composite) risk factor acceptable (based on current acceptance threshold)
O5	(Composite) cost factor acceptable (based on current acceptance threshold)
O6	Optimal decision when multiple options exist (see Note 3 below)
O*n*	Etc.

Notes:

1. The Type O events can be set by the Crew, or influenced by the crew by adjusting the acceptance threshold.
2. The Events O0, O1 and O2 are mutually exclusive.
3. When only one abort option exists, O6 is automatically set. When multiple abort options exist, the autocommander will make a decision using the built-in optimization algorithms. However, this decision may be overridden by the Crew or the GMC.

Table 7. Events of Launch Abort Logic Automaton

Events	Description
T1	!V12 & !V9 & P1 & C0 & O0 & O6
T2	!V12 & !V9 & P2 & C0 & O0 & O6
T3	!V12 & (P4 | P5 | P6 | P7) & C0 & (O0 | O1) & O6
T4	!V12 & !V9 & P3 & C0 & O6
T5	V10 & P8 & C0 & O6
T6	(V11 | !P8) & V13

Notes:
1. The "&" is the logical AND operator, "|" is the logical OR operator, and "!" is the logical NOT operator.

Stateflow Autocoder

A high-fidelity simulation environment is crucial for G&C algorithm development, validation, and verification. Such an environment must provide a seamless interface between the time-driven subsystems representing vehicle dynamics and conventional G&C subsystems and the event-driven processes such as the diagnostic function of the IVHM subsystem and the decision-making logic of the autocommander. The Matlab/Simulink environment, in which simulation models of time-driven subsystems are developed, together with the additional capabilities afforded by Stateflow to handle event-driven dynamics has been identified for this purpose (for an working example, see the MATLAB Stateflow Demo [11]). To more efficiently support autocommander algorithm development and implementation, an automated database-to-Stateflow conversion tool called the Stateflow Autocoder is also being developed [17]. The Stateflow Autocoder allows vehicle specifications, decision-making logic rules, optimization criteria and simulation scenario information to be captured in a database format and automatically generates the Stateflow components (called charts) that govern the event-driven aspects of the simulation and inserts theses components into the overall Simulink simulation.

As alluded to earlier, possibly thousands of states and transitions are required to characterize flight regimes and failures modes. Currently, Stateflow only provides a graphical user interface to adequately create and maintain charts, and implementing systems on a very large scale would be a cumbersome and tedious process. The capability to generate a Stateflow chart from information captured in a relational database is a highly desirable tool for the development of an autocommander.

The autocoder uses the Application Programming Interface (API) of Stateflow 4.2 (or later) to automate the construction of a complete working chart. API only provides a way to insert Stateflow objects into a chart; and by no means, a way to automatically define a Stateflow objects' parameters, such as position, size, or state label. Beyond the typical states, junctions and transitions that characterize an automaton, the autocoder creates, or will create, data variables, events, functions (graphical or text) and the remaining Stateflow objects to allow users utilize the full features of Stateflow in the course of autocommander development.

Stateflow provides a capability of concurrent automata by setting the decomposition parameter of the chart to *parallel*. The states that are children of this chart are referred to as *AND* (*parallel*) states, which are distinguished by dashed borders and numbered in the top right hand corner of each state as shown in Figure 6. This is a particularly important feature of Stateflow that allows the autocommander to be separated into different automata. As shown in Figure 6, for example, representing the flight mode requires one *AND* state that contains an *OR* (*exclusive*) state for each mission phase. Additionally, for every RLV component upon which vehicle status and health is based, separate *AND* states can be generated with each containing a nominal and a failure *OR* state.

The different abort modes could be implemented by one of two methods. The first method is via a single automaton with children OR states that represent each of the different abort modes. The second method is to have a separate automaton for each abort mode with feasible and not feasible OR sub-states and, in addition, to have an additional automaton within the chart to supervise the most desirable feasible abort scenario.

In Stateflow, two different types of operators can trigger a transition from one state to another, an event broadcast or a valid logical expression. A transition's label may contain either one or both types of operators. The scope of an event dictates which Stateflow objects receives its broadcast, which triggers a transition. It can be Input from Simulink, local to a group of states, or Output to Simulink. The logical expression is formed in the same way as an "if" statement in Matlab syntax. The data variables that are used within a transition's logical expression must be defined in the data dictionary of the chart. Data variables can be local, Input from Simulink, Output to Simulink, temporary, or constant. With Input from Simulink and Output to Simulink for both events and data variables, ports are generated on the Simulink Stateflow chart block for direct connection to a Simulink Model. Figure 7 shows a generated Stateflow chart and the input and output ports that are generated by the autocoder within a simulation model. In addition to generating the Stateflow chart, wireless connection blocks (goto and from blocks) are generated to provide a plug and play capability for the generated chart.

Another key aspect of Stateflow is that when a transition occurs and/or while in a state (entry, during, exit and on event) actions can be implemented. Actions may consist of the setting of flags, function calls, variable definitions, and broadcast events, etc. Not only do these actions provide the means for Stateflow to interact with Simulink and Matlab, but also provide the underlying structure and power of Stateflow.

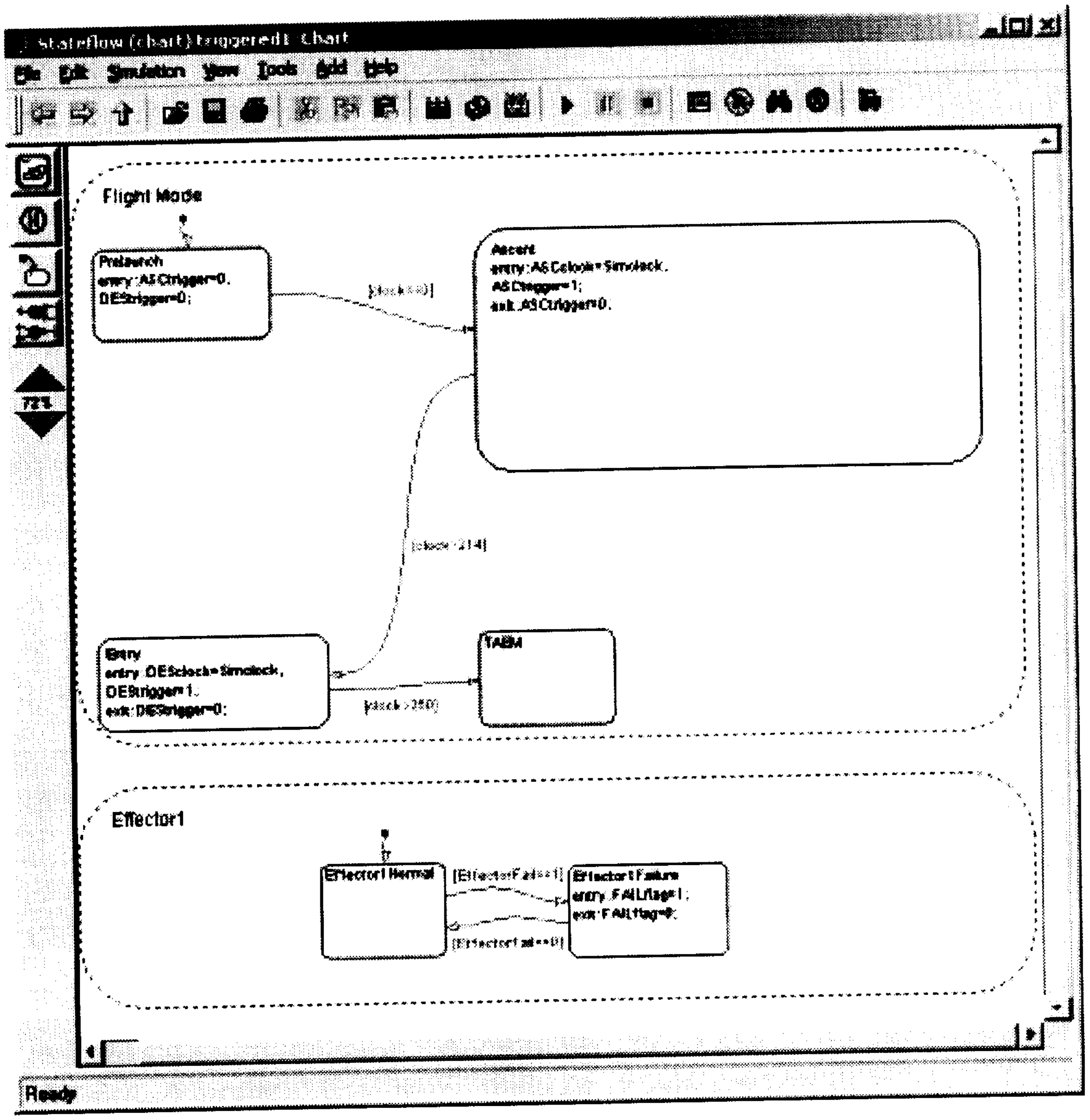

Figure 6. Autocommander Stateflow Diagram.

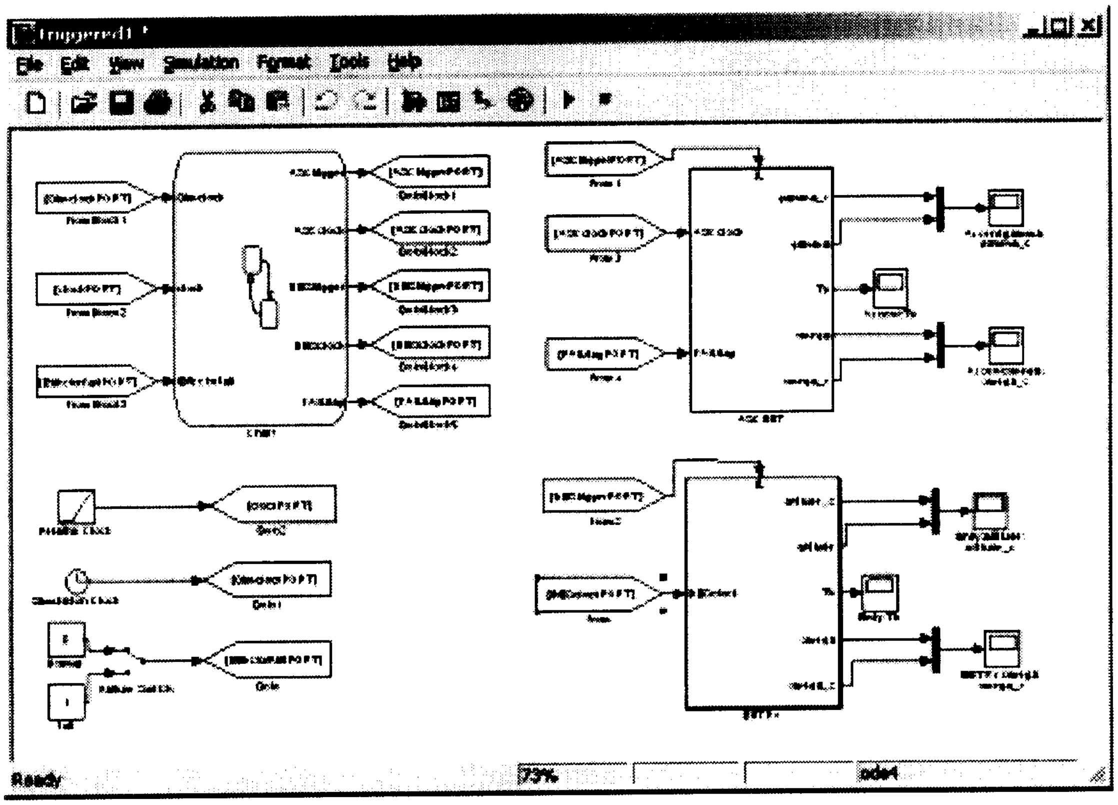

Figure 7. Simulink Simulation Containing a Generated Stateflow Chart.

Conclusions

With the goal of developing a crosscutting technology for autonomous mission management, G&C reconfiguration, and abort management, our approach is to formulate a generic discrete event driven hybrid control system architecture that will capture the broad knowledge of the IVHM, Flight Mechanics, Flight Control and Flight Operation engineers and designers pertaining to nominal flight regime transition events, G&C reconfiguration events, abort triggering events, and optimization criteria for abort strategy selection based on confidence levels and consequences of the events. The autocommander architecture will include a database with Graphical User Interface (GUI) for the domain engineers to specify the discrete event states and events along with their attributes for constructing the automata together with built in optimization algorithms for abort decision-making. Although the autocommander architecture is mainly for onboard operation, vehicle status display, OPP simulation, and the DFTC, it can also be used during the developmental phase and mission preparation to verify and optimize the abort logic and decision algorithms, and during flight phase by the GMC with additional computational power to perform more accurate and extensive contingency analysis.

References

[1] J. Hanson, "Advanced Guidance and Control Project for Reusable Launch Vehicles", Proceedings of the AIAA Guidance, Navigation, and Control Conference, August 14-17, 2000, Denver, CO.

[2] J. Hanson, D. Krupp, G. Dukeman, C. Hall, M. Gallaher, M. Phillips, J. McCarter, R. Jones, G. Kirkham, *Advanced Guidance and Control Project,* Test Report (draft), NASA/MSFC, September 2000.

[3] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999.

[4] Bienveniste, "Compositional and uniform modeling of hybrid systems," *IEEE Transactions on Automatic Control*, Vol. 43, No. 4, pp. 579-584, 1998.

[5] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: model and optimal control theory," *IEEE Transactions on Automatic Control*, Vol. 43, No. 1, pp. 31-45, 1998.

[6] C. G. Cassandras, Q. Liu, and K. Gokbayrak, "Optimal control of a two-stage hybrid manufacturing system model," *Proc. of the 38th Conf. on Decision and Control*, pp. 450-455, Phoenix, Arizona, December 1999.

[7] E. Frazzoli, "Robust Hybrid Control for Autonomous Vehicle Motion Planning," Ph.D. Dissertation, Massachusetts Institute of Technology, 2001.

[8] Gius and E. Usai, "High-level hybrid Petri nets: a definition," *Proc. of the 35th Conf. on Decision and Control*, pp. 148-150, Kobe, Japan, December 1996.

[9] G. J. Pappas, G. Lafferriere, and S. Sastry, "Hierarchically Consistent Control Systems," *IEEE Transactions on Automatic Control*, Vol. 45, No. 6, pp. 1144-1160, 2000.

[10] H. Ye, A. N. Michel, and L. Hou, "Stability theory for hybrid dynamical systems," *IEEE Transactions on Automatic Control*, Vol. 43, No. 4, pp. 461-474, 1998.

[11] "Fault Tolerant Fuel Control System," Stateflow Demo, MATLAB Version 5.3, R11, 1999.

[12] P. Lu and H. Sun, "Closed-Loop Endoatmospheric Ascent Guidance," submitted to this Invited Session ("Safety Gains for 2nd Gen RLVs Using Advanced G&C II" for 2002 AIAA GNC Conference).

[13] P. Lu and Z. Shen, "On-Board Generation of Three-Dimensional Constrained Entry Trajectories," submitted to the Invited Session "Safety Gains for 2nd Gen RLVs Using Advanced G&C I" for this conference (2002 AIAA GNC Conference).

[14] Yuri. B. Shtessel, J. Jim Zhu and Dan Daniels "Reusable Launch Vehicle Attitude Control Using A Time-Varying Sliding Mode Control Technique," submitted to the Invited Session "Safety Gains for 2nd Gen RLVs Using Advanced G&C III" for this conference (2002 AIAA GNC Conference).

[15] J. Jim Zhu, D. A. Lawrence, J. Fisher, Y. B. Shtessel, A. S. Hodel and P. Lu, "Direct Fault Tolerant RLV Attitude Control – A Singular Perturbation

Approach," submitted to the Invited Session "Safety Gains for 2nd Gen RLVs Using Advanced G&C III" for this conference (2002 AIAA GNC Conference).

[16] T. Adami, E. A. Best and J. Jim Zhu, "Stability Assessment Based on Lyapunov's First Method," to appear in Proceedings, 34th IEEE Southeastern Symposium on Systems Theory, Huntsville, AL, March, 2002.

[17] J. E. Fisher, D. A. Lawrence and J. Jim Zhu, "Stateflow Autocoder," to appear in Proceedings, 34th IEEE Southeastern Symposium on Systems Theory, Huntsville, AL, March, 2002.

[18] A. S. Hodel and R. Callahan, "Autonomous Reconfigurable Control Allocation (ARCA) for Reusable Launch Vehicles," submitted to the Invited Session "Safety Gains for 2nd Gen RLVs Using Advanced G&C III" for this conference (2002 AIAA GNC Conference).

[19] A. S. Hodel and Y. B. Shtessel, "On-line Computation of a Local Attainable Moment Set for Reusable Launch Vehicles," submitted to the Invited Session "Safety Gains for 2nd Gen RLVs Using Advanced G&C III" for this conference (2002 AIAA GNC Conference).

[20] A. S. Hodel and R. Callahan, "Dynamic Control Allocation for Reusable Launch Vehicles," to appear in Proceedings, 34th IEEE Southeastern Symposium on Systems Theory, Huntsville, AL, March, 2002.