AAS 18-134

OPTIMIZATION OF SECOND FAULT DETECTION THRESHOLDS TO MAXIMIZE MISSION PROBABILITY OF SUCCESS

Evan J. Anzalone*

In order to support manned spaceflight safety requirements, the Space Launch System (SLS) has defined program-level requirements for key systems to ensure successful operation under single fault conditions. The SLS program has also levied requirements relating to the capability of the Inertial Navigation System to detect a second fault. This detection functionality is required in order to feed abort analysis and ensure crew safety. Increases in navigation state error due to sensor faults in a purely inertial system can drive the vehicle outside of its operational as-designed environmental and performance envelope. As this performance outside of first fault detections is defined and controlled at the vehicle level, it allows for the use of system level margins to increase probability of mission success on the operational edges of the design. A top-down approach is utilized to assess vehicle sensitivity to second sensor faults. A wide range of failure scenarios in terms of both fault magnitude and time is used for assessment. The approach also utilizes a schedule to change fault detection thresholds autonomously. These individual values are optimized along a nominal trajectory in order to maximize probability of mission success in terms of system-level insertion requirements while minimizing the probability of false positives. This paper will describe an approach integrating Genetic Algorithms and Monte Carlo analysis to tune the threshold parameters to maximize vehicle resilience to second fault events over an ascent mission profile. The analysis approach and performance assessment and verification will be presented to demonstrate the applicability of this approach to second fault detection optimization to maximize mission probability of success through taking advantage of existing margin.

INTRODUCTION

To accommodate manned spaceflight safety requirements with regards to Navigation, the Space Launch System (SLS) utilizes an internally redundant Inertial Navigation System (INS) with built-in capability to detect, isolate, and recover from first failure conditions and still maintain adherence to performance requirements. The unit utilizes multiple hardware- and software-level techniques to enable detection, isolation, and recovery from these events in terms of its built-in Fault Detection, Isolation, and Recovery (FDIR) algorithms, which are outside of the scope of this work and are well documented in literature[1,2,3,4]. Successful operation is defined in terms of sufficient navigation accuracy at insertion while operating under worst case single sensor outages (gyroscope and accelerometer faults at launch).

*PhD, Aerospace Engineer, EV42 Guidance Navigation and Mission Analysis Branch, NASA/MSFC, Huntsville, AL 35812.

As described[5], the SLS program has taken a model-based approach to requirements development and negotiation in an effort to reduce system-level requirements and streamline systems engineering and management. At the vehicle level, the primary GNC-specific performance requirements on the INS are in terms of vehicle performance to a fixed trajectory. For this work, only faults that would affect sensor performance are considered. Faults that would manifest from other hardware failures, such as power supply or external communications, are captured through other methods. These requirements are then levied onto the prime contractor whose responsibility is to select or contract out the design and building of individual components that meet these high-level requirements. For the SLS effort, the INS has been implemented as the Redundant Inertial Navigation Unit (RINU) being developed and designed by Honeywell International[6].

For the SLS program, the INS primary performance requirement is to meet its insertion requirements over a reference trajectory under single fault conditions. This is proven through dynamic analysis provided by the vendor that demonstrates execution of flight algorithms with as-built sensor-level specifications. Sensor fault events are modeled as discrete events resulting in a change in sensor specification. Through the use of flight fault detection, isolation, and recovery software, these changes in sensor output are detected and removed from the integrated vehicle solution, minimizing their impact on its capability to meet system requirements. Statistical analysis then demonstrates that the INS meets its requirements under faulted conditions.

In addition to first fault detection and recovery, the SLS program has also levied requirements relating to the capability of the INS to detect a second fault. This detection functionality is required in order to feed abort analysis and ensure crew safety. The criteria for operation under second faults allows for a larger set of achievable missions in terms of potential fault conditions over time due to the significant consequences of an untrustworthy navigation system. As this performance is defined and controlled at the vehicle level, it allows for the use of system level margins to increase probability of mission success on the operational edges of the design space. A performance requirement on second fault detection was not levied through requirement flow-down to the vendor. As the second fault detection has much larger effects on vehicle performance, aborts, and operations, the responsibility for tuning the threshold parameters for this scenario lies within the SLS integrated GNC Team. For this analysis, the performance is defined in terms of meeting orbit insertion requirement with an integrated vehicle, as opposed to accuracy at the end of reference trajectory running open-loop simulation.

To support vehicle performance analysis and design, the SLS program utilizes Design Math Models (DMMs) that can be integrated into existing high fidelity dynamic analysis tools, such as the primary simulation tool MAVERIC[7] for 6 degree of freedom (DOF) Monte Carlo-based analysis. This model is intended to be verified against the vendor documentation and software models, and validated against flight hardware. As such, this software model is considered to be an analytically equivalent representation of the hardware, allowing for standalone and integrated analysis to assess performance, sensitivities, and fault detection capability. This allows for increased confidence in the vehicle's integrated performance prior to flight, and reduces the need for expensive flight testing. This allows for optimization of any input parameters, of which fault detection thresholds are considered. For the analysis that follows, this model is considered to be a black box with a defined generic input schedule.

FAULT ASSESSMENT APPROACH

With the design models in hand, it is possible to assess the capability in terms of detection times and navigation accuracy at the end of the mission through the use of open and closed analysis. Due to having a verified model, the results have a high confidence of being accurate. The two

analysis scenarios allow for both standalone and vehicle-integrated analysis. While the full closed loop simulation provides insight into coupling between Guidance, Navigation, and Controls functions, open-loop navigation analysis allows for focused insights that can guide closed-loop analysis, with a dramatically improved execution time. For scenarios requiring closed loop analysis, a local cluster asset is utilized to improve run-time.

First-Fault Analysis Results

The initial thresholds were optimized and assessed at the vendor- and VM-level using a reference trajectory coincident with SLS Design Analysis Cycle 2 assessments in support of vehicle Critical Design Review. These thresholds were assessed against this mission as part of the RINU Software Verification efforts, as well as in completed RINU Model DMM verification tasks. In order to assess their performance against the current missions as planned, all missions assessed have included one accelerometer and one gyroscope failure at launch. This was simulated as a large magnitude bias shift (1G and 10 deg/hr respectively) applied at Mission Elapsed Time (MET) = 0. This simulated error provides a "worst-case" sensor out scenario (defined as flying the entire mission with a reduced number of accelerometers and gyroscopes, resulting in reduced sensor accuracy). Implementation of this fault scenario also matches assumptions used in verification of hardware requirements.

To expand the scope of these results, the scope of failure scenarios was expanded to include varying magnitude faults and times. For these scenarios, two primary metrics of interest are orbit insertion requirements, Semi-Major Axis and Radius of Apogee. Two sets of 4000 runs were generated using the full 6DOF MAVERIC vehicle simulation, with faults of random magnitude occurring across the trajectory. The plots in Figure 1 and Figure 2 present scatterplots of the resulting orbital insertion capability in terms of 3-sigma bounds (the error in insertion for each case was divided by the requirement; i.e. values outside of +/- 1 do not meet requirements). From these results it is clear to see the first FDIR detection thresholds are working adequately to detect and isolate any first failures causing deviation from insertion requirements. It also provides insight into the remaining margin of the vehicle in terms of navigation performance.

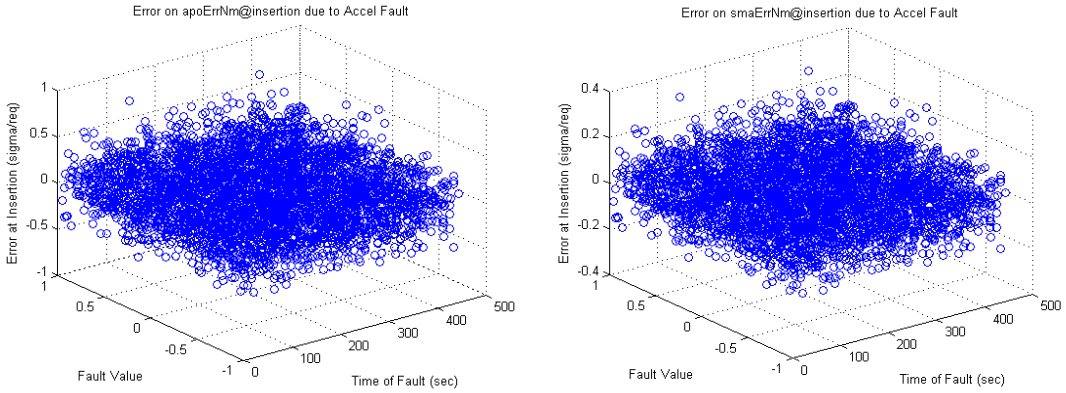

Figure 1. Insertion Errors due to Randomized Accelerometer Bias Fault (fault in g's)

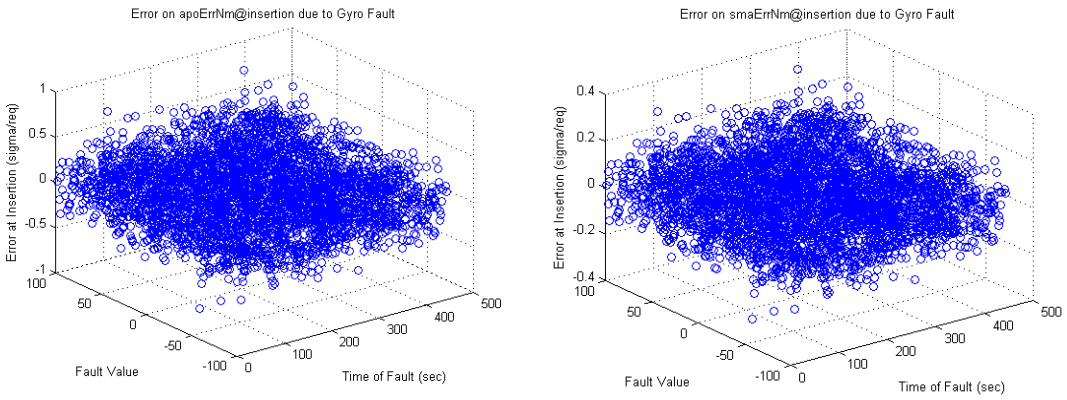

Figure 2. Insertion Errors due to Randomized Gyroscope Fault (fault in deg/hr)

Second-Fault Analysis Results

A requirement does not exist to specifically define FDIR performance under second fault scenarios. This is implicitly tied to mission insertion requirements and probability of mission success through the coupling of second faults to RINU overall trustworthiness. Upon detection of a second failure of a similar sensor type, the RINU is defined to be failed as a whole, and its output inertial state can no longer be trusted. At this point, the vehicle has no knowledge as to the severity of the failure event or the specific sensor acting out of specifications, and this error is integrated into the navigation state. This can cause immediate problems in term of flight control interaction (if the error occurs on a gyroscope), and limits the ability of the vehicle to accurately be guided to the desired insertion orbit in the case of accelerometer errors (as well as gyroscope errors). For second failure determination, the intent of the FDIR thresholds is to detect any failure which would cause the vehicle to not achieve its orbit insertion requirements. As such, the primary goal is to open the values to reduce sensitivity and remove margin above the RINU's insertion requirements to improve the probability of mission success.

The first step in opening up the thresholds is understanding the margin available in terms of vehicle insertion requirements. For this assessment, the standalone implementation of the RINU-DMM model was used to simulate second failure scenarios of a gyroscope and accelerometer independently and capture insertion capability against the reference trajectory. Figure 3 demonstrates the capability of the vehicle to meet insertion requirements under 2nd accelerometer failures. These were simulated as sensor error bias shifts on a particular channel. Throughout this analysis, the focus will be on bias shift errors, with further results demonstrating robustness against other potential sensor fault error manifestations. The horizontal axis provide the log(base 10) magnitude in g's (left) and time of fault (right). The vertical axis represents the ability of the vehicle to meet insertion requirements as the fraction of the insertion error over the requirement and plotted on a log scale. As shown, increases in accelerometer errors directly integrated into state errors over time, with the induced error scaled approximately to a product of the time remaining in flight squared times the additional accelerometer bias fault (representing the integration of the faulted accelerometer into the RINU's equations of motion). All of these failure magnitudes were detected by the RINU. Although it is limited, there do exist a group of small failure magnitudes across simulation time that do adequately meet RINU insertion performance. This shows that a slight tweaking of the accelerometer detection thresholds for second fault has only a small bit of room for improving vehicle probability of success.

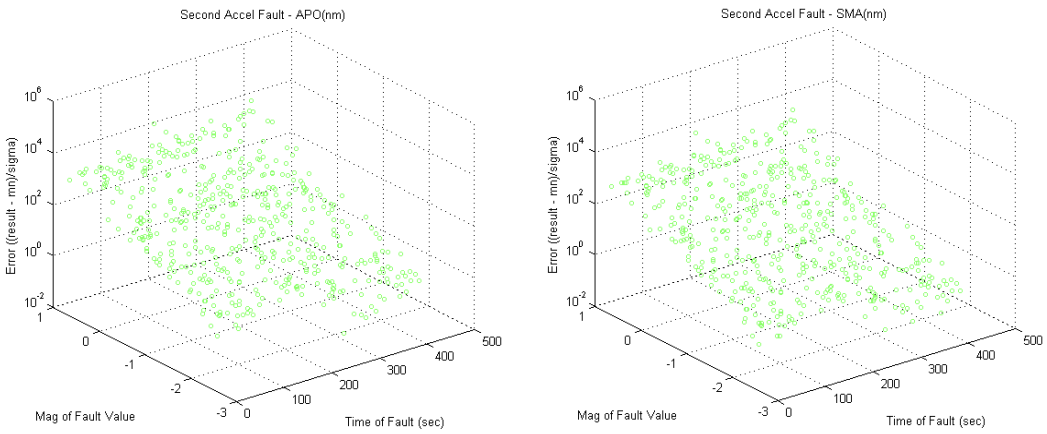

Figure 3: Insertion Errors with Second Accelerometer Fault (fault in log(g)'s)

Similar behavior can be observed in terms of the ability to detect second gyroscope fault events. The capability for this is demonstrated in Figure 4 below. For example, failure magnitudes less than 3.6 deg/hour were able to meet mission requirements, but were detected by the baseline thresholds. The performance with respect to gyroscope errors exhibits a much stronger correlation to the time of the fault. For example, as opposed to the fairly steady (and large) effect of accelerometer bias shifts across flight, large bias shifts on a gyroscope have decreasing effect on orbital insertion parameters. This time-based behavior strongly demonstrates the need for using a schedule to change performance over flight as a function of time, to match the changing response of the fault scenarios coupled to magnitude and time of fault.

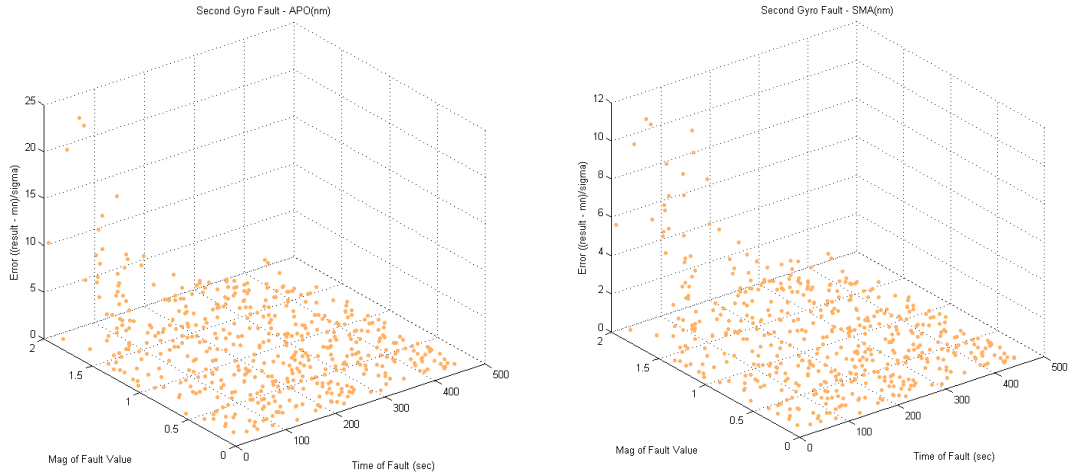

Figure 4: Insertion Errors with Second Gyroscope Fault (fault in log(degs/hr))

Threshold Scheduling

The SLS INS's threshold values are delivered as-built from the hardware vendor to optimize performance under 1st fault scenarios. Conservative detection of small failure events (which if undetected, would not cause large mission violations), is desirable in order to ensure meeting insertion requirements. Under nominal flight conditions, these are sized in order to ensure insertion

into the desired orbit for a reference trajectory, with margin captured at the VM level relative to vehicle requirements as identified above. The INS includes the capability to store a large number of unique sets of these parameters (each called a phase) that can be externally commanded (either from ground or onboard Flight Software (FSW) interfaces). The implementation of this functionality allows the potential to optimize FDIR performance across the design trajectory. The current FSW design includes the ability to load a different set of phases prior to launch and modify the current phase on the ground or in flight based on flight conditions. The design of these phases will be tuned to each individual flight, to optimize detection capability over the planned trajectory. This allows the sensitivity of the fault detection algorithms to change due to external drivers, such as the dynamics of the trajectory or detected failures.

The vehicle's FSW algorithms hold responsibility for tracking any detected failures and reporting this information to the crew and ground through collection, recording, and processing of hardware telemetry. Thus, the system can change FDIR parameters from FSW based upon observed failure conditions as well as MET via predefined schedule. This enables fine control over the detection capability to allow for tuning of the INS's FDIR capability (and the probability of missing a fault event or false detection of an event as a function of flight conditions). The primary operational FDIR modes are defined in Table 1.

Table 1. Defined FDIR Modes of Operation

Mode	Description
Nom	No detected sensor channel failures
AF	One accelerometer channel has faulted, but all gyroscopes are still good
GF	One Gyroscope channel has faulted, but all accelerometers are still good
AFGF	One Gyroscope and one Accelerometer have each faulted

The GN&C analysis will define the FDIR parameters to operate in each scenario. This allows for capturing all operational modes of the RINU in terms of per-sensor failure detection capability. The schedule table format is defined in Table 2, with notional phase inputs. The FDIR parameters and phase schedule will be updated as the mission trajectory and vehicle dynamics data are matured. An additional table is used when the vehicle is operating under off-nominal conditions. These flight scenarios are defined by the loss of an engine (either through software-commanded, or hardware-fault) during flight. Loss of an engine during boost and core flight has a large effect on vehicle dynamics and FDIR detection capability. In order to maintain capability, the phase will need to be chosen from an alternate table that is independent of MET. This data is meant to be a more robust, guarded set of algorithms, to account for the potential of an engine-out at any point along the trajectory. This logic can be enabled by tracking individual engine operational status.

Table 2. FDIR Phase Schedule

MET Range (ms) (from, to)		NOM	AF	GF	AFGF
0	10000	2	9	16	23
10020	50000	3	10	17	24
50020	120000	4	11	18	25
120020	150000	5	12	19	26
150020	200000	6	13	20	27

200020	300000	7	14	21	28
300020	600000	8	15	22	29

THRESHOLD SCHEDULE OPTIMIZATION

Further sensitivity analysis of the potential parameters included in each phase informed the difficulty of tweaking of and coupling between the individual inputs. In order to maintain conservatism, the primary focus is to open up the thresholds to more cases that would meet insertion requirements, while not allowing an undetected faults that would not meet mission. This analysis allowed for focus on a reduced set of design variables. In order to provide more detailed tuning of the FDIR algorithms, a batch of optimizations on these terms were performed. These took advantage of a nonlinear optimization algorithm to minimize the following fitness function:

FITNESS = -1(number of cases that meet requirement and not detected) + (number of cases that did not meet requirements and were not detected)*

This functions encompasses the tradeoff between opening the thresholds to allow more faults through detection (that increase the probability of mission success) while also considering additional faults that are not detected. These cases may be minimally acceptable as these faults would only cause a fault of the core mission with the potential for the upper stage to close the mission. Premature identification of second faults would potentially result in the core stage calling for an abort, causing an end to the mission.

A Genetic Algorithm[8] was utilized to perform the optimization, due to the stochastic (random failure events, and input variables) and nonlinear (coupled threshold parameters and integrated navigation system) nature of the problem. This is contrast to other methods[9,10] which focus on the statistical performance of the INS system. The algorithm operates by generating a population of random points within the potential design space, and then assessing each against the fitness function. After this data is collected, random pairs are selected out of the population and put through a competitive process. The better performing individuals are then put through a process called crossover (effectively breeding the characteristics of the best individuals) and mutation (allowing for random changes in population to enable robust exploration of the design space). These are typically performed by capturing each set of design variables as a representative binary string upon which crossover and mutation is applied. This process is repeated for several generations until the population has effectively converged to a global optimum solution. This assessment used the MatLAB Genetic Algorithm within its Optimization Toolkit[11]. Utilization of this framework allowed for rapid generation of unique input files to the INS model as well as processing and integration of simulation results. This toolkit is operated through a visual interface and manages all of the backend processes required in Genetic Algorithm-based optimizations. The MatLAB environment allowed use of the Parallel Processing Toolkit[12] enabling assessment of multiple individuals simultaneously. This helped dramatically to improve run-time.

Optimization Results

For this assessment, a population of 20 was used. Each individual in the population is generated from a Monte Carlo assessment of 100 dispersed runs in terms of sensor errors and fault events. The populations typically converged to a global solution in 20-30 generations. This analysis assumed 4 variables per type of sensor, encapsulating both settings for accelerometers and gyroscopes, with the noise terms being optimized on a logarithmic scale to provide a wide range. A graphic representation of the optimization process is provided in Figure 5. This graphic shows the convergence of the population to the optimal solution in the upper left line chart. The steps

down in fitness, represent where cross-over and mutation processes generated an individual with performance better than any of the previous population. The bar chart in the top right depicts the best solution. The bottom histograms show the score distribution across the latest generation and the individual scores of each individual. This plot is unique to the optimization of gyroscope terms over the entire mission.

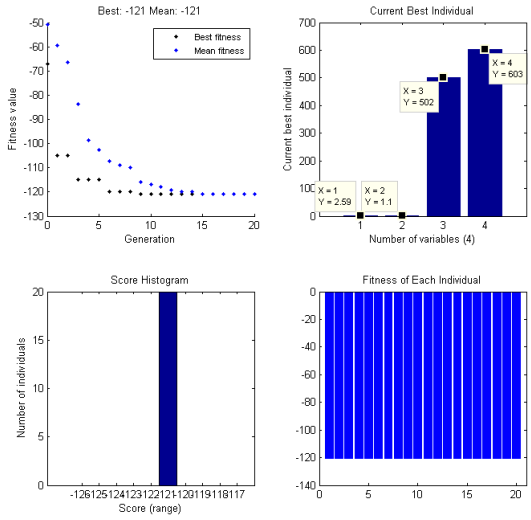

Figure 5: Genetic Algorithm Optimization Process

Due to the coupling between variables, one set of thresholds across the entire mission is not considered optimal. Re-optimizing across individual sub-periods across the mission (as proposed in the FDIR operational architecture laid out above) allows for improved performance and tuned threshold capability across the entire mission window. The thresholds were optimized across time periods matching the proposed schedule. This schedule was developed to allow for quick thresholds changes during early flight dynamics (including high dynamic environments) and then stabilizes out for longer periods with relatively smooth and stable flight. The optimization process was repeated for each focused time period with faults limited to each for gyroscope and accelerometer failure events independently.

Upon integrated analysis, the shifting thresholds caused disruptions in fault detection due to periods where the thresholds would become tighter, inducing detection of smaller faults not detected during an earlier time period. To address this, the thresholds were optimized all-at-once to second sensor faults dispersed across the entire flight trajectory. This caused an increase in number of design variables per optimization from 4 to 24 (for gyroscope and accelerometer faults independently). In order to accommodate the larger design space, the number of variables was reduced to 2 for each sensor type per time.

The performance results of each threshold provides insight into the capability of the simulation. For this assessment, 1000 cases (up from the original 100) were assessed with the RINU standalone model, injecting faults of random time, axis, and magnitude of a second fault event, assuming an initial large bias shift at launch to simulate a first sensor channel failure. A summary of the thresholds for the accelerometer channels is given in Table 3, which compares results for the baseline thresholds, an optimization assuming one threshold for failures across the entire flight, and an optimized schedule. The results show there is minimal additional benefit to tuning the accelerometer errors, as even with 1000 runs, with typically only 10% of the runs successfully

met the mission requirements with the modeled fault events. For optimizing one set of thresholds, a large increase in successful cases not having a fault detected is increased, but at the cost of also increasing the number of unsuccessful missions. Optimizing the thresholds across time did serve to minimize the number of unsuccessful missions not detected in flight, but also reduced the number of successful missions completed. The baseline thresholds provide a balance between the two, and program-level mission risk perspectives inform the switching to other sets of thresholds.

Table 3 Accelerometer Threshold Performance

Case	Number of Cases	Min. Time of Fault	Max. Time of Fault	Min. Fault Magnitude (log g Error! Bookmark not defined.)	Max. Fault Magnitude (log g Error! Bookmark not defined.)	Number Met Orbital Insertion	Number Met and Not Detected	Number Not Met and Not Detected
Baseline Threshold	1000.00	10.00	450.00	-3.00	1.00	115	34	12
Single Optimized Threshold	1000.00	10.00	450.00	-3.00	1.00	143	102	90
Optimized Threshold Schedule	1000.00	10.00	450.00	-3.00	1.00	123	14	4

In terms of gyroscope capability, there is a much larger capability enabled, as seen in Table 4. By tuning the input thresholds, a large number of additional passing cases are not declared faulted due to FDIR. Additionally, this benefit is much larger than the additional number of cases not detected that do not meet mission requirements. This is seen as a reduced sensitivity to gyroscope fault events over the course of the flight.

Table 4 Gyroscope Threshold Performance

Case	Number of Cases	Min. Time of Fault	Max. Time of Fault	Min. Fault Magnitude (log deg/hr)	Max. Fault Magnitude (log deg/hr)	Number Met Orbital Insertion	Number Met and Not Detected	Number Not Met and Not Detected
Baseline Threshold	1000.00	10.00	450.00	-1.00	2.00	800	240	0
Single Optimized Threshold	1000.00	10.00	450.00	-1.00	2.00	800	757	114
Optimized Threshold Schedule	1000.00	10.00	450.00	-1.00	2.00	806	767	75

THRESHOLD SCHEDULE VERIFICATION APPROACH

To provide verification of this capability, these thresholds were implemented into MAVERIC to capture performance using a large number of runs, 4000. This is used to demonstrate the performance of the algorithms with a complete schedule over the flight. In addition to capturing the performance of the optimized thresholds, baseline scenarios were also assessed, which used a

fixed thresholds set. For these cases, fault magnitudes were dispersed uniformly across [-1,1] g for accelerometer bias shifts, and [-100,100] deg/hr for gyroscope fault events. Table 5 shows the capability of the optimized thresholds in terms of meeting the orbital insertion targets.

Table 5 Performance of Thresholds in MAVERIC

		Baseline/Fixed Threshold			Optimized Single Thresholds			Optimized Threshold Schedule		
Type of Fault	Time of 2nd Fault	# Met	# Met + Not Detect	# Not Met + Not Detect	# Met	# Met + Not Detect	# Not Met + Not Detect	# Met	# Met + Not De-tect	# Not Met + Not Detect
Accel.	[10, 500]	194	153	32	194	142	32	194	145	32
Gyro.	[10, 500]	2554	183	0	2529	1803	354	2564	1930	216

This verification assessment in MAVERIC demonstrates the same performance of the thresholds as compared to the standalone model results. Again, the optimized accelerometer thresholds perform very similar to the baseline values with the gyroscope channels vastly reducing the probability of false detections. This provides evidence to support the de-coupling of FDIR analysis from the 6DOF models, demonstrating minimum effect due to GNC algorithm and coupling sensitivities.

To provide insight into algorithm robustness, additional Monte Carlos were performed focusing on additional failure mechanism: Scale Factor shifts (correlating the sensor error to the input dynamics), and null errors (where the output defaults to 0.00). For this analysis, the baseline and optimized (one threshold across entire mission) threshold performance is demonstrated against random magnitude (for scale factor, in units of PPM), axis, and time (across the entire trajectory). Table 6 and Table 7 document the results for the accelerometer and gyroscope channels, respectively.

Table 6: Threshold Performance against Accelerometer Null and SF Faults

Fault Scenario	Magnitude (log ppm)	Thresholds Used	Num. Met	Num. Met and Not Detected	Num. Not Met and Not Detected
2nd-null	n/a	Baseline	33	32	5
2nd-SF	[-3,5]	Baseline	1428	1322	78
2nd-null	n/a	Optimized	33	25	0
2nd-SF	[-3,5]	Optimized	1428	1211	18

Table 7 Thresholds Performance against Gyro Null and SF Faults

Fault Scenario	Magnitude (log ppm)	Thresholds Used	Num. Met	Num. Met and Not Detected	Num. Not Met and Not Detected
2nd-null	n/a	Baseline	671	14	0
2nd-SF	[-3,5]	Baseline	1898	1694	12
2nd-null	n/a	Optimized	587	360	3

2nd-SF	[-3,5]	Optimized	1895	1863	44

The results presented mirror those demonstrated for bias shift-type fault behavior. Again, the baseline accelerometer thresholds are shown to be well-tuned for the fault conditions simulated. In comparison, the optimized gyro parameters vastly increased the number of successful mission scenarios, at a very small cost of missed detections, also matching behavior over the nominal trajectory. These results demonstrate the robustness of the algorithms to additional failure scenarios, both in their capabilities and limited effects on missed detections.

CONCLUSION

In order to maximize probability of success against the vehicle's insertion requirements, the FDIR thresholds have been optimized to minimize the probability of false detection. A Genetic Algorithm was utilized where each member of the population represented the statistical results of a Monte Carlo open-loop simulation utilizing a verified INS software model. This approach optimized the threshold values across the trajectory in an all-at-once manner to find a global optimized schedule. This has improved the probability of meeting the desired performance under second fault conditions on the gyroscope and accelerometer channels under nominal flight. Due to the limited evidence for improved performance under optimized accelerometer thresholds, the baseline parameters have been kept in use. Additionally, this limits SLS testing and software verification requirements. The optimized gyroscope parameters are being used, though, due to their clear improvement in performance under the modeled scenarios.

These thresholds were optimized against bias shifts, verified against scale factor and null fault conditions, and assessed in both standalone and integrated vehicle models. The chosen values provide an improvement of flight capability for reducing false detection rates of second failure faults. This is enabled by the inherent margin between SLS insertion requirements and RINU performance requirements. The results presented are optimized against the current design trajectory, and will be re-verified as part of further verification analysis and flight readiness assessments to demonstrate their capability. Additionally these thresholds will be tested as part of software and hardware-in-the-loop testing to demonstrate and verify the FSW to RINU FDIR interface, usage of FDIR schedule, and operational data. Additionally, these thresholds will be further validated against RINU sensor data collected during Green Run to both validate the underlying FDIR standalone model and its performance under flight-like conditions.

This optimization approach was only possible due to the large amount of design into the INS design, both hardware and software, with thorough understanding of the underlying fault detection algorithms. With this knowledge, the design space was able to be reduced to limit the need for a large population. This was necessary due to each member in the population being the statistical results of a stochastic open-loop analysis. Additionally, this approach was enabled due to the systems engineering processes focused on managing system- and sub-system level margins. For a design scenario with limited performance margin, this approach would provide a minimal benefit. This scenario focused on an ascent-only scenario. As these systems are integrated onto future vehicles[13,14,15] with longer lifetimes, more insight is needed in definition of the mission schedule, specifically over long coast. Also, the computational runtime per member increases dramatically as the analysis expands from 100's of seconds to several hours, requiring a more nuanced approach. This analysis utilized MatLAB'S built-in Genetic Algorithm Toolkit, further tweaking and modification of the input parameters could provide improved performance as well increasing the number of individuals in the population. A Mutli-objective Genetic Algorithm[16,,17] could also

be utilized to provide insight into the trade between optimizing mission success and fault detection.

ACKNOWLEDGMENTS

The author would like to acknowledge the support of other SLS Navigation Team members in supporting review of this work, the SLS Navigation Team Lead Emerson Oliver for his support of this approach to threshold optimization, as well as branch and division management, Heather Koehler and Don Krupp, for their support in publishing this work. This work would not have been possible without the tight integration between NASA and contractor team members, and much gratitude is due to David Lick of The Boeing Company and Roger Bacon, Joe Protola, and Kerry Sutherland while at Honeywell International, for their support and insight into the RINU design and architecture.

REFERENCES

[1]J. V. Harrison, et al. "Performance of an aided redundant navigator in normal and failure modes." *Journal of Guidance, Control, and Dynamics*, Vol. 3, No. 6, 1980, pp. 481-486.

[2]Hwang Inseok, Kim Sungwan, Kim Youdan, and C.E. Seah, "A Survey of Fault Detection, Isolation, and Reconfiguration Methods," *IEEE Transactions on Control Systems Technology*, Vol.18, No.3, May 2010, pp.636-653.

[3]S. Osder, "Practical View of Redundancy Management Application and Theory", *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 1, 1999, pp. 12-21.

[4]J.E. Potter and M.C. Sumen, "Thresholdless Redundancy Management with Arrays of Skewed Sensors". *Integrity in Electronic Flight Control Systems*. AGARD-AG-224. Jan. 1977.

[5]E. Oliver et al. "SLS Model-Based Design Approach: A Navigation Perspective", *Proceedings of the AAS Guidance and Control Conference*, Breckenridge, CO., 2018.

[6]B. Hawkins. "Space Launch System: Exploration Class Capability for Deep Space Exploration." *2015 Workshop on Spacecraft Flight Software (FSW-15)*, 27-19 Oct. 2015.

[7]J. Hanson and C. Hall, "Learning about Ares I from Monte Carlo Simulation", *Proceedings of the AAS Guidance, Navigation, and Control Conference*, August 2008.

[8]D.L. Carroll, "Chemical laser modeling with genetic algorithms." *AIAA Journal*, Vol. 34, No. 2, 1996, pp. 338-346.

[9]E. Gai, M.B. Adams, and B.K. Walker, "Determination of Failure Thresholds in Hybrid Navigation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 12, No.6, Nov. 1976, pp. 744-755.

[10]B.K. .Walker and E. Gai, "Fault Detection Threshold Determination Technique Using Markov Theory", *Journal of Guidance, Control, and Dynamics*, Vol. 2, No. 4, 1979, pp. 313-319.

[11]A.J. Chipperfield, P. J. Fleming, and C. M. Fonseca. "Genetic algorithm tools for control systems engineering." *Proceedings of Adaptive Computing in Engineering Design and Control*, 1994.

[12]G. Sharma, and J. Martin. "MATLAB®: a language for parallel computing." *International Journal of Parallel Programming*, Vol. 37, No. 1, 2009, pp. 3-36.

[13]B. Donahue et al. "Space Launch System: Development Status," *AIAA SPACE 2016 Forum*, 2016.

[14]B. Donahue and S. Sigmon, "Space Launch System: Block 1B Configuration: Development and Mission Opportunities." *AIAA Propulsion and Energy Forum, 53rd AIAA/SAE/ASEE Joint Propulsion Conference*, 2017.

[15]G. Williams. "Human Exploration & Operations Update for NAC HEO Committee", November 14, 2016.

[16]A. Konak, D.W. Coit, and A. E. Smith. "Multi-objective optimization using genetic algorithms: A tutorial." *Reliability Engineering & System Safety*, Vol. 91, No. 9, 2006, pp. 992-1007.

[17]F.A. Fortin, et al. "DEAP: Evolutionary algorithms made easy." *Journal of Machine Learning Research*, Vol. 13, July 2012, pp. 2171-2175.