An Alternative Flight Software Trigger Paradigm: Applying Multivariate Logistic Regression to Sense Trigger Conditions using Inaccurate or Scarce Information

Kelly M. Smith[1], Robert S. Gay[2], and Susan J. Stachowiak[3]
NASA Johnson Space Center, Houston, Texas, 77058

In late 2014, NASA will fly the Orion capsule on a Delta IV-Heavy rocket for the Exploration Flight Test-1 (EFT-1) mission. For EFT-1, the Orion capsule will be flying with a new GPS receiver and new navigation software. Given the experimental nature of the flight, the flight software must be robust to the loss of GPS measurements. Once the high-speed entry is complete, the drogue parachutes must be deployed within the proper conditions to stabilize the vehicle prior to deploying the main parachutes. When GPS is available in nominal operations, the vehicle will deploy the drogue parachutes based on an altitude trigger. However, when GPS is unavailable, the navigated altitude errors become excessively large, driving the need for a backup barometric altimeter to improve altitude knowledge. In order to increase overall robustness, the vehicle also has an alternate method of triggering the parachute deployment sequence based on planet-relative velocity if both the GPS and the barometric altimeter fail. However, this backup trigger results in large altitude errors relative to the targeted altitude. Motivated by this challenge, this paper demonstrates how logistic regression may be employed to semi-automatically generate robust triggers based on statistical analysis. Logistic regression is used as a ground processor pre-flight to develop a statistical classifier. The classifier would then be implemented in flight software and executed in real-time. This technique offers improved performance even in the face of highly inaccurate measurements. Although the logistic regression-based trigger approach will not be implemented within EFT-1 flight software, the methodology can be carried forward for future missions and vehicles.

Nomenclature

EFT-1	=	Exploration Flight Test-1
MPCV	=	Multi-Purpose Crew Vehicle
GPS	=	Global Positioning System
FBC	=	Forward bay cover
EI	=	Entry interface (400,000 feet)
PDS	=	Parachute deployment sequence
GN&C	=	Guidance, Navigation, & Control
θ	=	Parameter vector
x	=	Feature vector
h_θ	=	Hypothesis function, parameterized by θ
ANTARES	=	Advanced NASA Technology Architecture for Exploration Studies
IMU	=	Inertial measurement unit
RCS	=	Reaction control system

[1] Aerospace Engineer, Flight Dynamics Division, Mission Operations Directorate/DM42
[2] Senior Aerospace Engineer, GN&C Autonomous Flight Systems Branch, Engineering Directorate/EG6
[3] Senior Aerospace Engineer, Flight Mechanics & Trajectory Design Branch, Engineering Directorate/EG5

I. Introduction & Motivation

THE Orion Multi-Purpose Crew Vehicle (MPCV) will be launched aboard a Delta IV-Heavy rocket for Exploration Flight Test-1 (EFT-1). Once the vehicle completes a single revolution in the initial parking orbit, the upper stage will re-ignite to inject the vehicle into a highly eccentric orbit. The orbit has a very high apogee, but the perigee has dropped below the surface of the Earth. After injecting into this elliptical orbit, the vehicle will coast to apogee and accelerate back to Earth. The vehicle will encounter Earth's atmosphere at approximately 29,000 feet per second and will fly a high-speed entry trajectory to deliver the vehicle to a targeted water landing in the Pacific Ocean, west of Baja California. During entry, the vehicle will deplete its energy through atmospheric drag and slow itself until its becomes subsonic. During the final descent, the vehicle will first jettison the Forward Bay Cover (FBC), a protective covering on the apex end of the vehicle. Immediately after FBC jettison, two drogue parachutes will be deployed to dampen the vehicle attitude rates and further slow the vehicle's descent. After a given amount of time and/or altitude conditions are met, the vehicle will release the drogue parachutes and immediately deploy its three main parachutes. While the main parachutes are deployed, the vehicle will continue to slow its descent so that the vertical velocity at water impact is survivable. Once water impact has been detected, the vehicle will cut away the main parachutes.

This sequence of jettisoning the forward bay cover (FBC), deploying the drogue parachutes, releasing the drogue parachutes, and deploying the main parachutes shall be referred to as the parachute deployment sequence, or PDS. The PDS is designed to begin once the vehicle has fallen through an altitude threshold, currently targeted at 24,000 feet.

During hypersonic entry, the vehicle will become engulfed in ionized flow, and this flow will prevent the GPS receiver from receiving GPS information. During this period, the vehicle will rely on sensed accelerations from its inertial measurement units (IMUs) to update its navigation solution. Once the vehicle has decelerated sufficiently so that there is no longer ionized flow, the GPS receiver is expected to reacquire GPS satellites and begin updating its navigation solution based on the high-precision measurements. As planned, the GPS receiver is expected to reacquire prior to the critical parachute deployment sequence.

However, this is the first spaceflight for this particular GPS receiver and the navigation software. In the event that there is an anomaly preventing the GPS receiver from reacquiring, the vehicle is equipped with three redundant barometric altimeters that can provide altitude information in the atmosphere. However, the barometric altimeters may produce unreliable altitude estimates due to local variations in the barometric pressure.

In the event that the GPS receiver fails to reacquire and the barometric altimeters have failed, there exists an additional backup trigger (tertiary) which serves to initiate the parachute deployment sequence. This backup trigger compares the vehicle velocity relative to a pre-stored velocity vector that is representative of the nominal drogue parachute deployment condition. Once the vehicle velocity vector becomes close enough to the representative velocity vector (within some tolerance), the vehicle will activate the parachute deployment sequence. This backup velocity-based trigger has been evaluated in Monte Carlo flight simulations, and its performance has been documented in [1] and [2]. The best altitude performance achieved with this technique produced altitude spreads in excess of 25,000 feet when the minimum altitude was no less than 24,000 feet.

As a tertiary trigger (two failures deep), this performance was deemed acceptable by the Orion MPCV Program for the first flight test for Orion. However, it is desirable to develop a more accurate trigger for future missions to provide more robustness.

This paper focuses on bridging the large chasm that typically exists between the mathematical formulation of an algorithm and its practical implementation in actual flight software. It introduces a statistical technique, known as logistic regression, which may be employed for certain problems to reduce the difficulty of identifying and developing robust flight software triggers.

II. Flight Software Triggers

In this section, an overview of the need for and typical development of flight software triggers is discussed. A space vehicle's flight software must be capable of switching from one mode to another in order to achieve mission objectives. For example, once the sensible atmosphere is detected, the spacecraft should begin flying entry guidance. Once hypersonic entry flight is completed, then the vehicle should initiate its final descent sequence by deploying parachutes to stabilize and slow itself in order to reduce its terminal descent rate for splashing down in the ocean. Finally, once the vehicle splashes down in the ocean, the reaction control system (RCS) jets should be disabled so that the local area is not contaminated with hydrazine, a hazard to recovery forces.

In each of these cases, the vehicle must have some criteria for activating a particular logic path or switching logic paths in flight software. While conceptually simple, GN&C designers must design triggers that work reliably

in a wide range of potential scenarios using only the information that the on-board vehicle has at the time. Given navigation system failures, the difficulty of developing triggers insensitive to knowledge errors is compounded. For actual flight implementations, GN&C engineers must contend with noisy sensor data, as opposed to smooth signals. As a result, a simple trigger involving a threshold can be inadvertently activated by a noisy signal spiking briefly over the threshold. To mitigate these spurious signals, auxiliary variables known as persistence counters are frequently employed. Persistence counters simply count the number of consecutive instances that a given condition is true. Once the persistence counter reaches a certain value, then the algorithm trusts that the trigger is not being fooled by spurious sensor noise. However, these persistence counters must also be tuned, and by their very nature, they add latency to the system in detecting events, adversely impacting system performance.

For these reasons, the triggers designed for operational flight software tend to use multiple conditions to mitigate noisy sensor readings and navigation system errors. Traditionally, GN&C algorithm developers have embraced the use of simple comparisons that can be easily encoded into software through the use of if-then statements. As a result, triggers have been primarily conceived along the lines of simple comparisons against thresholds or table values.

However, for many problems, it is very difficult to accurately isolate a condition with a single variable. Instead, the conditions must be based on several variables to account for the high-dimensionality of the conditions. If the designer chooses to isolate the condition by using if-then statements to account for all possible scenarios, they may be forced to exhaustively bound the multidimensional region of acceptable solutions. This approach is time-consuming and prone to logical errors. Additionally, the threshold values typically need to be tuned manually. Inevitably in the design cycle, there will be vehicle change or performance change which may invalidate the original trigger threshold values. As a result, the designer may be required to re-tune the threshold values and revisit the logic to ensure its design is still valid. Finally, once all the conditions have been defined for the trigger, the software complexity has grown substantially, requiring a lengthy software testing and validation effort.

III. Logistic Regression Overview

Logistic regression is a statistical technique used to classify data into categories. It has been employed by statisticians and the machine learning community to classify data based on its characteristics. The simplest form of logistic regression simply classifies data into two categories, and this simplified version is known as binary logistic regression. The remainder of this paper will focus on applying binary logistic regression to the problem of creating triggers that activate for some targeted condition.

Whereas linear regression attempts to fit a straight line (or surface) to a dataset to predict a continuously-valued output, logistic regression attempts to fit a logistic function, also known as a sigmoid function, to a dataset. The logistic function is an S-shaped curve which has continuous output bounded between 0 and 1 for all input values. The logistic function can be written as

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

where h_θ is known as the hypothesis function, θ is known as the parameter vector, and x is the input vector. The parameter vector θ holds constant values that help determine the shape of the multivariate logistic fit. For a given input vector x, and parameter vector θ, the value of h_θ is determined. This model is analogous to the linear regression fit which can be expressed as:

$$f(x) = x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \bar{\theta}^T \bar{x}$$

A notional logistic function is visualized in the figure below:

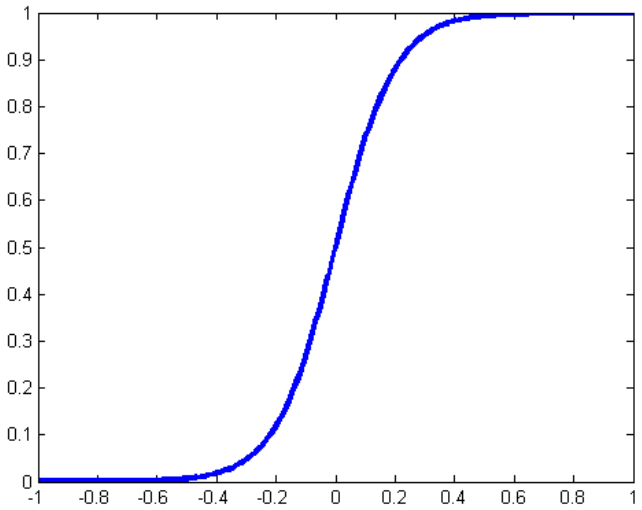

Figure 1: Notional Logistic Function

While linear regression fits can be achieved in closed-form by solving the normal equation, a logistic function is typically fit using the least-squares gradient descent algorithm. The output of the function, h_θ, can be interpreted as the likelihood that the input vector x belongs to the target class $y = 1$.

In order to use this approach for classifying data, one must provide the algorithm with many training examples with their respective labels specifying to which class the example belongs. The algorithm then iterates on θ to minimize the error between the predicted value of h_θ and y, the truth labels specifying the correct class. Once the gradient descent algorithm has converged, the parameter vector θ has been been determined, and the logistic function is ready to classify new data.

The combination of the the logistic function and the parameter vector θ is typically known as the classifier. To classify data, a new input vector x is provided to the classifier and the resulting value indicates the likelihood that the hypothesis is true (that $y = 1$ for this input vector x).

IV. Application of Logistic Regression to Flight Software Triggers

Logistic regression was identified as an alternative approach to the problem of developing flight software triggers because it tolerates noisy data and does not require perfect separation of classes.

Logistic regression can be applied to this problem by providing training examples to the algorithm which show examples of vehicle states where the vehicle should and should not have initiated its parachute deploy sequence. Next, the logistic regression algorithm will determine the characteristics which tend to distinguish between the two classes of data. Once the parameter vector θ is determined through gradient descent optimization, the trigger is ready to be used.

Logistic regression is very well suited to the problem of detecting a complex, multi-dimensional vehicle state with noisy data. Whereas the IF-statement style logic is easy to conceptualize, it cannot easily account for noisiness in the data. Logistic regression acknowledges that the data is noisy and that perfect separation between classes is generally not possible. Even if there is no strong single indication, logistic regression may use several weak signals to determine whether or not the target condition is true.

Furthermore, logistic regression can be semi-automated so that triggers can be more easily generated without requiring substantial tuning analysis. The trigger is automatically developed on the ground using the logistic regression algorithm, and the resulting parameter vector is stored as a pre-specified vector of numbers to be used in flight software. The flight software simply evaluates the logistic function with the pre-loaded parameter vector and the measured values from the feature vector.

V. Implementation

Using the 6-degree-of-freedom Advanced NASA Technology Architecture for Exploration Studies (ANTARES) trajectory simulation system and Orion flight software, 3000 Monte Carlo trajectories were flown from entry

interface (EI) to splashdown using the initial conditions for EFT-1. During the simulated flights, the GPS receiver and barometric altimeters were disabled so that the navigation solution was informed solely by the inertial measurement unit's (IMU's) measurements during entry. Additionally, the parachutes were disabled so that the trajectory dynamics were not influenced by parachute deployment events.

For each trajectory, the time-series values of various trajectory parameters was recorded. These parameters include:

- Navigated altitude
- Atmospheric relative velocity magnitude
- Sensed aerodynamic acceleration magnitude
- Time elapsed since 0.2Gs (1.960 m/s^2) of acceleration was first detected
- Navigation estimate of Mach number
- Navigation estimate of dynamic pressure

These parameters, among others, were included in the set of available features for the logistic regression algorithm for this analysis. At each timestamp in the trajectory, the value of each variable is concatenated into a single vector. Additionally, truth altitude from the simulation was recorded so that the training examples could be labeled with the correct command.

At each time step (1Hz) in each trajectory, each feature vector $\bar{x}_i$was associated with its respective label y_i. For all feature vectors above a truth altitude of 25,000 feet, the associated label was $y = 0$, indicating that the parachute deployment sequence should not be initiated. This altitude of 25,000 feet was selected so that the spread of altitude at PDS initiation was biased slightly above 24,000 feet, the lowest acceptable altitude for PDS initiation. It is much more acceptable to deploy slightly too early at a higher altitude than to deploy too late at a lower altitude. All feature vectors below a truth altitude of 25,000 feet were labeled as $y = 1$, indicating that the parachute deployment sequence should be initiated. Using gradient descent, a statistical fit was produced for the provided training data and labels. The final output of the algorithm was the parameter vector θ.

Within flight software, the trigger can be evaluated (tersely) in a single line of code. However, for clarity, the logic shall be expanded into multiple lines for the purposes of this paper. In the MATLAB programming language, the code can be expressed as:

```
>> h = 1 / (1 + exp(transpose(-theta)*x));
>> command = h > threshold;
```

This small snippet of code evaluates the logistic function and compares the value against a pre-specified threshold. If the current value of the logistic function exceeds the threshold, then the trigger is activated.

This trigger does not require many lines of code for on-board implementation and is easy to verify. Additionally, the evaluation of the trigger is computationally inexpensive, requiring only a single exponentation of a vector. This approach mitigates the massive complexity associated with the traditional IF-statement approach. Because the equation is easily vectorized, one can add or subtract new features from the feature vector and continue to use the exact same logistic function code. This flexibility allows engineers to continue to improve performance even after flight software code has been frozen for development. Changes to the trigger can be made completely through configuration data.

Designers are able to specify the activation threshold that the logistic function output must exceed to activate the command. A higher activation threshold effectively requires the trigger to be more confident in its input signals before activating, whereas a lower activation threshold is more prone to triggering with less reliable information. The model fitting process attempts to balance the false positives and false negatives in its classification performance. Consequently, for a well-fit model, the mean PDS initiation altitude tends to be very near the targeted altitude condition. In this application, the vehicle tends to deploy at higher altitudes when the activation threshold is low; conversely, for higher activation threshold, the deployment altitudes tend to be lower. It is desirable to ensure that the lowest deployment altitude should not fall below the targeted altitude. As a result, one may select an activation threshold slightly less than 0.5 such that the altitude spread is shifted slightly higher so that no deployments occur below the target altitude.

VI. Results

A large set of trajectory parameters were evaluated for their fitness with respect to logistic regression. A subset of those trajectory parameters, their associated classifiers, and their classifier performance are presented in this section, as well as a selected vector of the best trajectory features used to perform multivariate logistic regression.

Each trigger presented was trained on 200 trajectories and tested against 600 trajectories. For each trajectory, the altitude at which the trigger first commanded the parachute deployment sequence (PDS) was recorded, and the PDS altitudes were plotted in a histogram to visualize the distribution. For each altitude distribution, statistics were computed to characterize performance, allowing for easy comparison between features and classifier models.

For numerical reasons, all input features were normalized such that all values of the variable are mapped into the [-1, 1] domain. This technique increases the speed of convergence when fitting the parameter vector. As a result, all values of the features shown in the following plots will be normalized between -1 and 1.

A. Truth Altitude

As a test case, it is instructive to see how this approach subsumes the standard approach of comparing the altitude against some threshold, such as:

```
IF (altitude < ALTITUDE_THRESHOLD) THEN Deploy
```

For this reason, the truth altitude was used as the only feature available in the training data. All data points in which the truth altitude is below the targeted deployment altitude of 25,000 feet are labeled with a "1" and all cases above that altitude are labeled as a "0." As one should expect, there is perfect separation between these two classes, as shown in Figure 2. Given the perfect separation between classes, the fitting a logistic function to this dataset should result in an excellent fit. The training dataset and the resulting model are shown below.

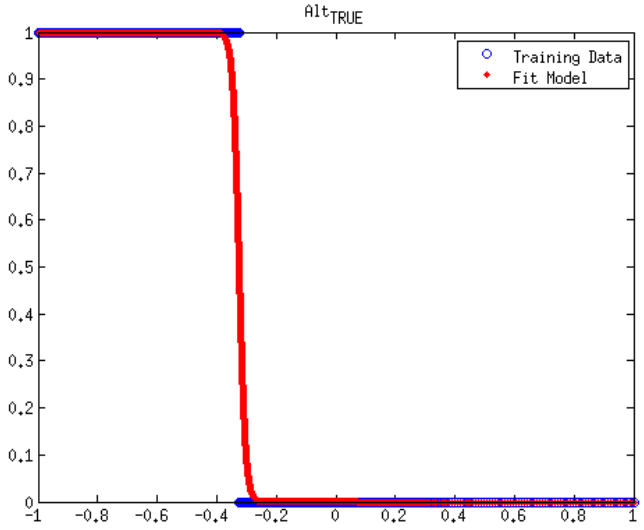

Figure 2: Distribution of altitude classes and the logistic model fit

The model is fit such that errors are balanced about the target condition, resulting in the ideal model activation threshold of 0.5. However, it is instructive to demonstrate how the altitude performance varies when the activation threshold is varied between values of 0.1 to 0.9. For each activation threshold, the spread in deployment altitude is shown for 600 dispersed trajectories.

Each red dot represents the altitude at PDS initiation for a single case. The dashed red line represents the desired target altitude for PDS initiation (25,000 feet). The dashed black lines connect the $\pm 3\sigma$ altitude cases, shown as black triangles, and the black squares represent the mean PDS initiation altitude for each distribution.

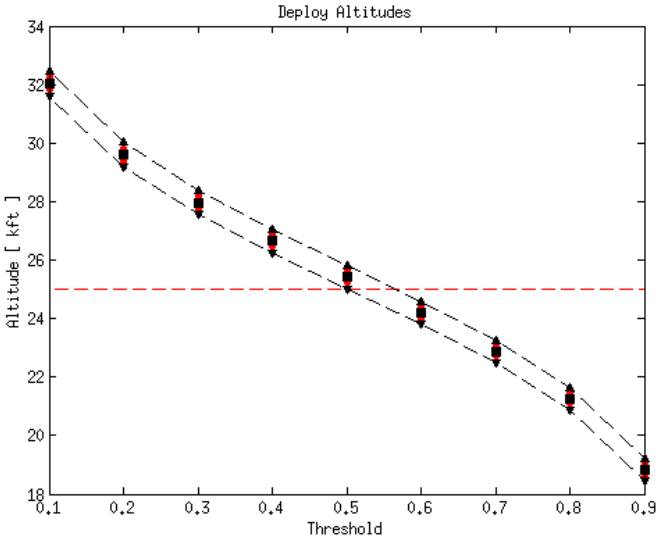

Figure 3: Altitude spread at PDS using a truth altitude classifier at varying activation thresholds

For the activation threshold of 0.5, the spread in altitude for the 600 dispersed trajectories tested is 550 feet with a minimum altitude of approximately 25,200 feet. This indicates that the trigger is very well-tuned.

This example demonstrates the viability of this approach for idealized, perfect information, and it demonstrates the best altitude spread achievable using the logistic regression approach. Unfortunately, the spacecraft does not have knowledge of the true altitude. The spacecraft's knowledge is limited to the accuracy of the navigation solution, which may grow in error when measurements are scarce.

B. Navigated Altitude

For actual implementation in real flight software, the truth altitude is unavailable. Instead, the navigation solution must be used as the source of altitude information. For this scenario when the vehicle does not have GPS or barometric altimeter measurements, the navigation accuracy degrades throughout hypersonic entry due to navigation attitude errors at entry interface. Specifically, the navigation altitude errors at low altitudes are sensitive to navigation attitude errors at entry interface. Due to small errors in the navigation attitude solution will lead to aerodynamic accelerations being applied to the slightly "wrong" direction during entry. Consequently, the altitude errors at low altitude tend to be relatively large after substantial aerodynamic acceleration has been experienced during entry flight.

For this case, the navigated altitude was used as the singular feature in the training set. After associating each navigation altitude with the truth label (above or below the targeted deployment altitude), the two groups are visualized in Figure 4.

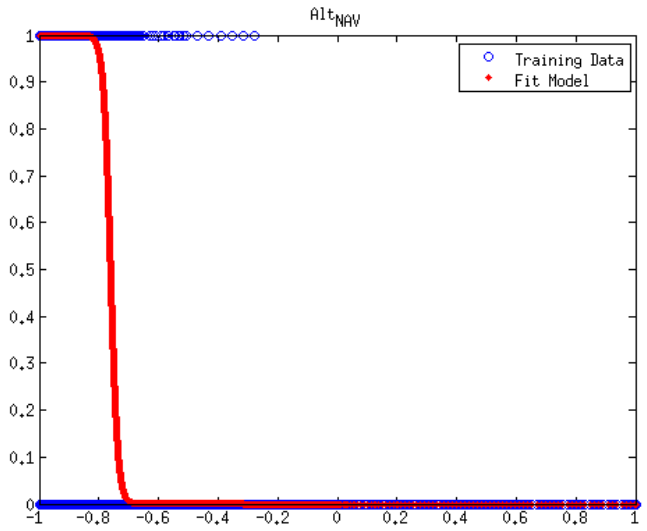

Figure 4: Class distribution for navigated altitude

Inspection reveals significant overlap between the two classes. The overlap between the classes serves to reduce the tightness of the fit model, which will tend to increase the classification errors.

After fitting the model, the new trigger is evaluated against the test data over a range of activation thresholds. The altitude spread for each activation threshold is shown in Figure 5. Each red dot represents the altitude at PDS initiation. The dashed red line represents the desired target altitude for PDS initiation. The dashed black lines connect the ±3σ altitude cases, shown as black triangles, and the black squares represent the mean PDS initiation altitude for each distribution.

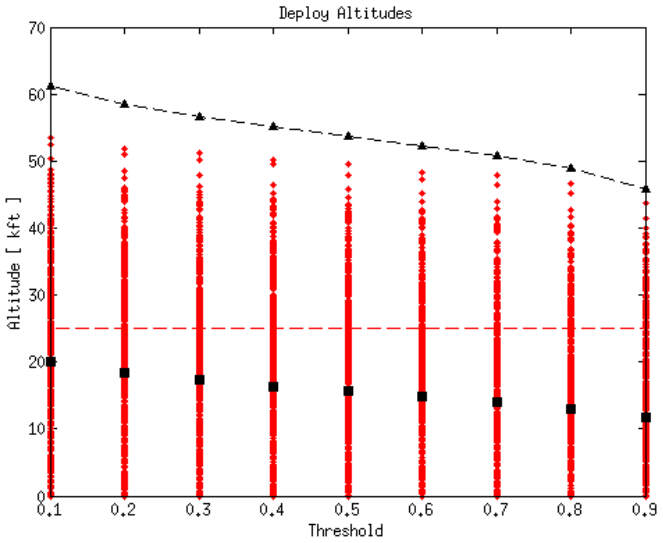

Figure 5: Altitude spread at PDS using a navigated altitude classifier at varying activation thresholds

Given the noisy altitude data, it is clear that the altitude deployment spread has grown substantially. As shown in the figure above, the spread of altitude at PDS initiation is unacceptable; many cases crash before initiating the parachute deployment sequence. For this reason, it is clearly unacceptable to attempt to initiate the PDS solely based on navigated altitude in the no-GPS, no-barometric altimeter failure scenario.

C. Truth Relative Velocity Magnitude

In this example, the training set uses only truth atmospheric relative velocity magnitude as the single feature. The intent is to determine how suitable relative velocity magnitude is as a training parameter. Visualizing the distribution of the values of truth velocity magnitude for the different classes reveals some minor overlap between classes. The fit model output is shown in Figure 6 for the spread of normalized velocity values. The fit logistic function reaches its peak value in this interval of [-1 1] just above 0.6. This implies that setting activation thresholds above 0.7 will result in cases never initiating the PDS.

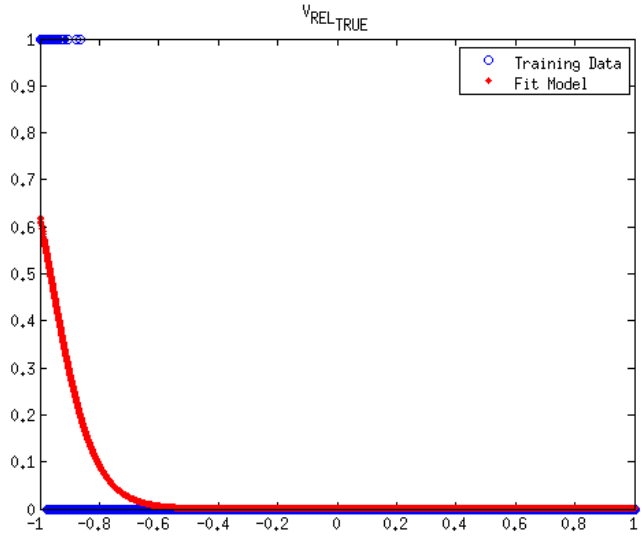

Figure 6: Class distribution for truth relative velocity magnitude

This model has been evaluated against the testing data to assess its altitude performance for a range of activation thresholds, and the altitude performance is shown in Figure 7. Each red dot represents the altitude at PDS initiation. The dashed red line represents the desired target altitude for PDS initiation. The dashed black lines connect the ±3σ altitude cases, shown as black triangles, and the black squares represent the mean PDS initiation altitude for each distribution.

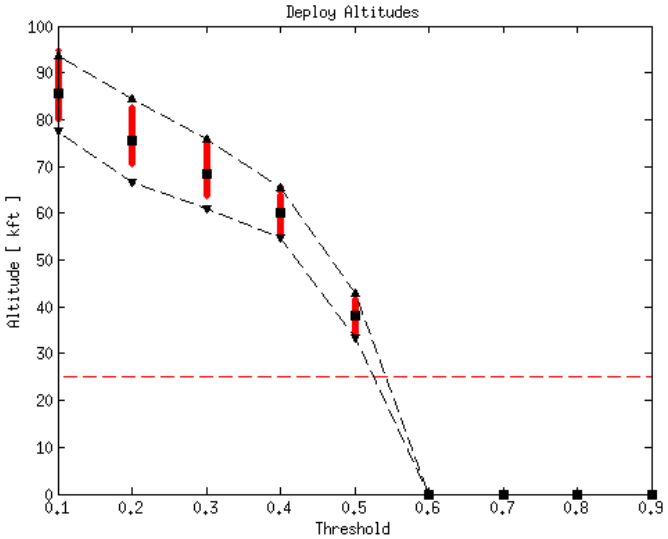

Figure 7: Altitude spread at PDS using a truth relative velocity magnitude classifier at varying activation thresholds

For the activation threshold of 0.5, the minimum altitude was approximately 33,000 feet, and the altitude spread was approximately 9,000 feet for the 600 trajectories evaluated. For all activation thresholds tested above 0.5, no cases deploy because of the statistical fit. Regardless of how low of altitude the vehicle thinks it is at, the output of the logistic function is failing to deploy at all thresholds of 0.6 and higher, as evident in Figure 7. All of the

initiation altitudes for activation thresholds of 0.6 and higher are all zero, implying that the vehicle never initiated the PDS prior to impacting the Earth. However, for a properly selected activation threshold, the relative velocity magnitude has a relatively tight spread in altitude performance.

D. Navigated Relative Velocity Magnitude

Although truth velocity was a suitable parameter for triggering the parachute deployment sequence, the on-board vehicle does not know the true velocity; instead, it must use its navigated value, as was the case with altitude. For the navigated velocity magnitude, the distribution of values per class is shown as well as the model fit.

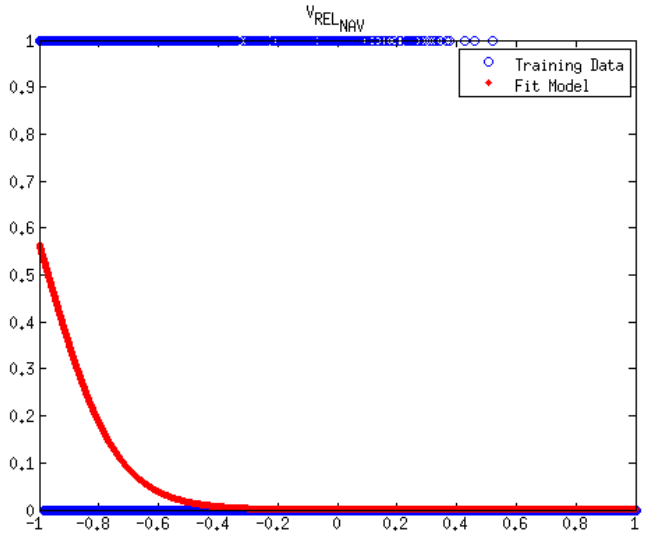

Figure 8: Class distribution for navigated relative velocity magnitude

As visible in Figure 8, there is substantial overlap between the classes for the navigated relative velocity magnitude. However, there are far more data points in the $y = 0$ class than the $y = 1$ class over the majority of the interval, resulting in the model fit shown in red. Only at very low velocities does the fraction of $y = 1$ cases exceed the number of $y = 0$ cases, and this change in relative frequency is what causes the fit logistic function to rise at low velocities. As was the case in the truth relative velocity magnitude, the peak value of the logistic function does not reach 1.0 within this interval. Instead, it peaks at approximately 0.57. For this reason, all activation thresholds above this peak value will result in no trajectories initiating the PDS.

This model, only using navigated relative velocity magnitude, was evaluated at a variety of activation thresholds, and the resulting altitude performance was plotted in Figure 9. Each red dot represents the altitude at PDS initiation. The dashed red line represents the desired target altitude for PDS initiation. The dashed black lines connect the ±3σ altitude cases, shown as black triangles, and the black squares represent the mean PDS initiation altitude for each distribution.

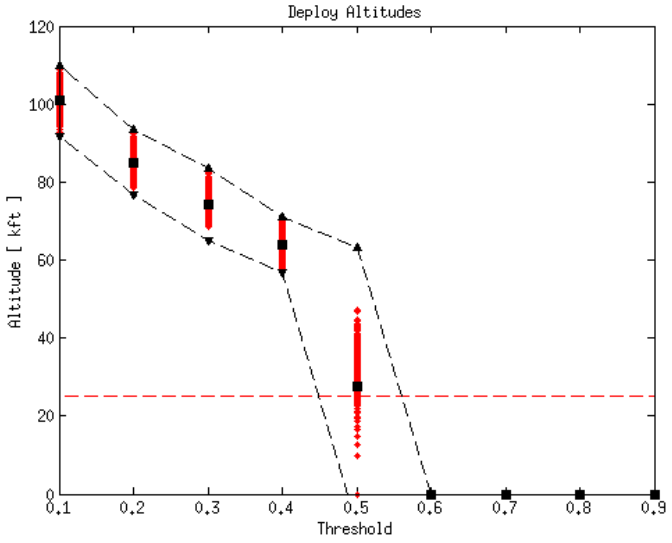

Figure 9: Altitude spread at PDS using a navigated relative velocity magnitude classifier at varying activation thresholds

At the activation threshold of 0.5, there are many cases which impact the Earth without initiating the PDS. For this reason, the distribution of altitude at 0.5 is non-Gaussian due to so many cases impacting the Earth; therefore, the ±3σ curves lay significantly outside the dataset due to the non-Gaussian distribution. However, at the activation threshold of 0.4, the PDS altitude spread is relatively tight, approximately 14,000 feet; the minimum altitude of approximately 57,000 feet. While this is high above the targeted condition, the spread is relatively tight in comparison with the navigated altitude performance documented in section B.

Just as with the true navigated relative velocity, the activation thresholds larger than 0.5 fail to initiate the PDS because the value of the logistic function is bound by a maximum value of approximately 0.57 at minimum altitude. For this reason, activation thresholds of larger than this value will not initiate the PDS sequence.

Clearly, using a classifier soley based on relative velocity magnitude is not sufficient to provide adequate performance. However, relative velocity magnitude offers value as a member of a multivariate training set, described later.

E. Elapsed Time since 0.2Gs

While slight errors in an IMU alignment may result in large altitude errors after entry flight, the accelerometers in an IMU capture the magnitude of aerodynamic accelerations very well. For this reason, aerodynamic acceleration magnitude is reliable signal with little error. Given the reliability of this signal, it would be simple to measure the elapsed time from the first time that the total acceleration magnitude sensed by the IMU exceeds some small threshold indicating atmospheric entry. For this study, the threshold was arbitrarily set at 1.960 m/s^2, or 0.2 Gs. For all vehicle the elapsed time was computed since 0.2Gs of aerodynamic acceleration was detected.

As shown in Figure 10, it is apparent that elapsed time is a powerful feature for applying logistic regression; there is very small ovelap between the classes, and the logistic fit is very tight. This type of visualization is helpful to the designer, indicating that a very information-rich feature variable has been identified.

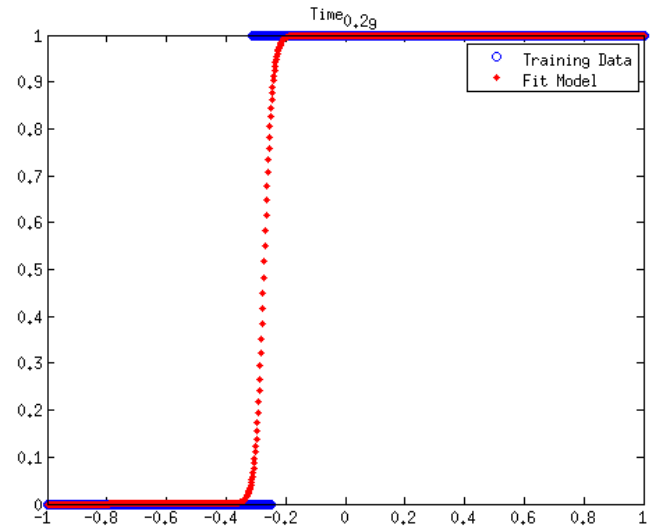

Figure 10: Class distribution for elapsed time since 0.2G (~1.96 m/s^2)

After training a classifier using only elapsed time since 0.2G (1.96 m/s^2), the classifier performance was evaluated on 600 trajectories for a range of activation thresholds. For each activation threshold, the altitude spread is shown in Figure 11. Each red dot represents the altitude at PDS initiation. The dashed red line represents the desired target altitude for PDS initiation. The dashed black lines connect the ±3σ altitude cases, shown as black triangles, and the black squares represent the mean PDS initiation altitude for each distribution.

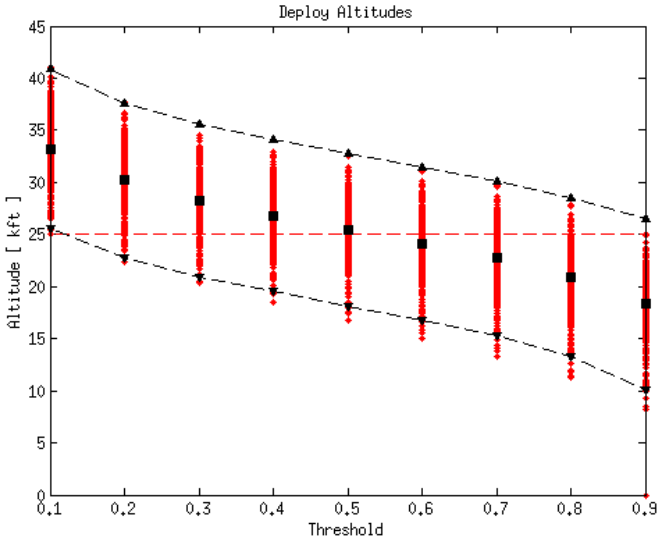

Figure 11: Altitude spread at PDS using a elapsed time classifier at varying activation thresholds

It is very apparent that using elapsed time since 0.2G (1.96 m/s^2) is a very powerful metric. The spread in altitude is very consistent across activation thresholds, and so one may simply choose a suitable activation threshold to ensure that all PDS initiations occur above 25,000 feet, the targeted altitude floor.

The mean value of PDS initiation altitude for the 0.5 activation threshold is nicely centered about the targeted altitude condition, indicating that the model fit has properly balanced the false positives and false negatives. However, it is desired to have no cases initiating PDS at altitudes lower than the altitude targeted floor of 25,000 feet. Across all activation thresholds considered, the altitude spread changes very little. This suggests that no significant loss of altitude spread (accuracy) is required to shift the distribution of PDS initiation altitudes such that the distribution is shifted above 25,000 feet.

F. Multiple Features

While the previous subsections described the altitude performance while using single features in a logistic regression model, this section will use multiple features to create a multivariate logistic fit in order to improve performance. For each considered feature, its standalone performance was used as the criterion to determine whether or not it should be included in the set of features in the multivariate logistic fit. The selected set of features include:

- Navigated altitude
- Navigated relative velocity magnitude
- Elapsed time since sensing 0.2Gs of aerodynamic acceleration
- Sensed aerodynamic acceleration
- Navigated Mach number
- Navigated dynamic pressure

After fitting the parameter vector to the training data, one may inspect the elements of the parameter vector to gain insight into which parameters are powerful. In this case, the parameter vector contained the following values:

Constant	Altitude	Velocity	Time	Aerodynamic Acceleration	Mach	Dynamic Pressure
1.37	-1.45	-2.27	21.43	1.38	-2.16	1.43

The first element of the parameter vector, referred to as "Constant" in the above table, is a non-zero constant used in fitting the filter. From inspecting the elemenets, we can see that the element for elapsed time in the parameter vector is the largest magnitude. This corresponds to a very steep slope in the logistic regression curve which serves to effectively split the classes between $y = 1$ and $y = 0$. After fitting the model with logistic regression, one may inspect the relative magnitudes of the parameter vector elements and learn more about the problem space. This is a useful analysis technique for identifying the "driving" features.

The trigger was evaluated using multiple values of the activation threshold, and the resulting altitude spread at PDS initiation is shown in Figure 12. When trained on 600 trajectories, and tested against 3000 trajectories, the resulting altitude spreads at PDS initiation are visualized below. Each red dot represents the altitude at PDS initiation. The dashed red line represents the desired target altitude for PDS initiation. The dashed black lines connect the ±3σ altitude cases, shown as black triangles, and the black squares represent the mean PDS initiation altitude for each distribution.

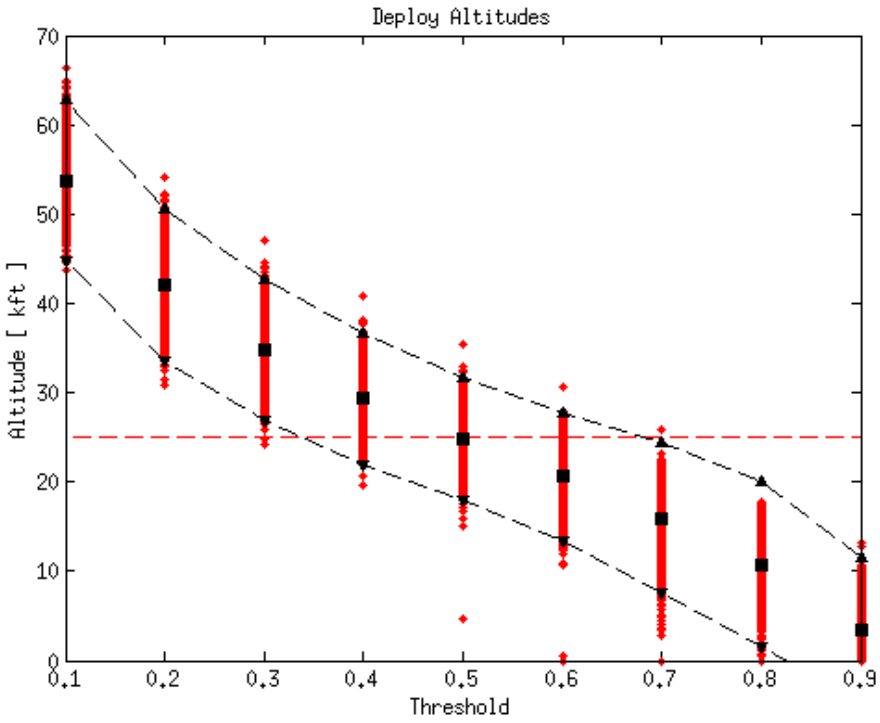

Figure 12: Altitude spread at PDS using a multivariate classifier at varying activation thresholds

For the activation threshold of 0.5, the minimum altitude is approximately 4,700 feet (single outlier case), and the maximum altitude is approximately 35,500 feet, resulting in a total spread of approximately 30,800 feet. The mean deployment altitude is approximately 24,800 feet, only 200 feet below our targeted altitude of 25,000 feet. This suggests that our statistical model balances false positives and false negatives such that the mean performance is approximately centered on the targeted altitude condition for the 0.5 activation threshold.

If instead the 3-sigma altitude spread is considered instead of the total spread, then the altitude spread contracts to approximately 13,800 feet. Only 0.43% of cases fall outside this altitude spread.

However, if the activation threshold is selected such that no cases fall below the targeted altitude, then the activation threshold of 0.2 must be selected where the minimum altitude is approximately 30,900 feet. For this case, the total altitude spread is approximately 23,200 feet.

However, for the activation threshold of 0.3, only two cases deploy below the 25,000 altitude target (still above the 24,000 altitude floor), resulting in slightly lower distribution of altitude at PDS initiation. For this activation threshold, the total altitude spread (22,700 feet) is reduced by 500 feet as compared to activating the trigger at a 0.2 threshold, and it lowers the mean PDS initiation altitude by approximately 7,200 feet to 34,800 feet.

Using the multivariate distribution resulted in slighted improved performance as compared to the adjusted velocity-trigger approach [1] currently employed by Orion for EFT-1. This approach demonstrates improved the PDS altitude precision by reducing altitude spread by approximately 2,000 ft. Given the slight increase in total performance, it suggests that the different approaches are asympoting towards some lower limit of precision achievable when navigation knowledge is so degraded.

VII. Lessons Learned

Although logistic regression-based triggers offer several advantages, there were new challenges that were encountered in the development & implementation of the trigger. For instance, one must carefully monitor the gradient descent optimization of the model fit. Depending on the problem space, one may need to reduce or increase the step size used for gradient descent to aid in model convergence.

Additionally, trigger performance was shown to be sensitive to the statistical model fit as produced by numerical gradient descent optimization. For a particular feature set, gradient descent optimization may struggle to converge, and it would be relatively easy to overlook a poor model fit for a particular variable when fitting multiple

dimensions. For this reason, the information in a particular feature may not be exploited well by the statistical model.

Logistic regression works well for features that are monotonic, but may struggle with variables that are not monotonically distributed. For instance, altitude rate was a troublesome feature. Intuitively, the values of altitude rate at low altitudes are relatively tightly clustered. However, the same altitude rate may be observed during hypersonic flight at shallow negative flight path angles while at high speeds. For this reason, it was difficult to achieve a tight model fit for altitude rate using logistic regression. In these cases like these, variables typically are transformed into new features, requiring additional analysis and insight from the algorithm designer.

Another difficulty encountered with logistic regression was the difficulty in determining why the trigger was activated for a given scenario, given multiple input signals. It is easy to compute the individual contributions, but understanding why a particular variable contributed more than another variable can be difficult to explain without first understanding the underlying statistical model.

A final lesson learned is that logistic regression is a powerful tool for identifying strong features which easily separate data classes. This technique may be employed as a valuable analysis technique for designers seeking to identify the key features which separate different classes of data.

VIII. Future Work

The applicability of multivariate logistic regression to the problem of creating robust flight software triggers was explored in this paper, but other techniques remain to be explored for easing the burden of "tuning" a GN&C system's triggers. Another approach under consideration is to estimate the likelihood of the vehicle being at or below the target altitude condition given the current navigation solution and prior Monte Carlo information. Similar to logistic regression, this is a probabilistic approach which performs inference to estimate the true state of the vehicle given the inaccurate navigation information in double-failure scenarios. This concept has demonstrated excellent performance in preliminary assessments, and it is much more "automatic" than using logistic regression, requiring no numerical optimization. This approach exploits the probability of observing any value of an continuous input feature given that the vehicle should or should not initiate the PDS. Armed with this information for a set of trajectory parameters, Bayes' Theorem may be employed to estimate the likelihood that the vehicle should initiate the PDS given the output of the navigation system. Unlike logistic regression, this approach does not assume a certain distribution of the probability. This approach will be explored further as an alternative strategy to logistic regression.

IX. Conclusion

The algorithm described in this paper, binary logistic regression, has been applied to help the Orion vehicle determine whether or not the conditions are right to initiate the parachute sequence during descent through the atmosphere. The current implementation in Orion flight software compares adjusted velocity against a threshold value to determine when to initiate the parachute deployment sequence when GPS and barometric altitmeters have failed (two failures deep). This proposed approach using logistic regression has helped to develop a new trigger condition that reduces the total altitude spread to approximately 22,700 feet as compared with the adjusted velocity trigger approach described in [1]. This approach is less susceptible to the noise endemic to actual flight hardware signals, and it enables the semi-automated generation of robust flight software triggers for a new trajectory design. The approach is very generic, and the algorithm can be re-applied to solve several other problems for flight software event detection.

Acknowledgments

K. S. thanks Dr. Andrew Ng of Stanford University for making available his excellent, free machine learning course materials through the Stanford University website and Coursera.org.

References

[1] Gay, Robert, Stachowiak Susan, and Smith, Kelly. "Orion Exploration Flight Test-1 Contingency Drogue Deploy Velocity Trigger," 36th Annual Guidance and Control Conference; Breckenridge, CO; 1-6 Feb. 2013; United States

[2] Stachowiak, S., "Assessment of Velocity Trigger as a Backup Drogue Chute Deploy Option for Orion Exploration Flight Test #1," FltDyn-CEV-12-4, NASA Johnson Space Center, February 14, 2012, Houston, TX.