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. In order to increase overall robustness, the vehicle also has an alternate method of triggering the drogue parachute deployment based on planet-relative velocity if both the GPS and the barometric altimeter fail. However, this velocity-based 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 automatically generate robust triggers based on statistical analysis. Logistic regression is used as a ground processor pre-flight to develop a classifier. The classifier would then be implemented in flight software and executed in real-time. This technique offers excellent 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
EI	=	entry interface (400,000 feet)
θ	=	parameter vector
x	=	feature vector
α	=	gradient descent step size
tol	=	gradient descent change tolerance
h_θ	=	hypothesis function, parameterized by θ
M_∞	=	estimated Mach number
$\dot{h}$	=	altitude rate
q	=	estimated dynamic pressure
t_{EI}	=	elapsed time since entry interface
v_∞	=	free stream velocity magnitude
a_{AERO}	=	sensed aerodynamic acceleration magnitude
γ	=	flight path angle
ANTARES	=	Advanced NASA Technology Architecture for Exploration Studies

[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 30,000 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 damp the vehicle attitude rates and further slow the vehicle's descent. After a given amount of time, the drogue parachutes will be released from the vehicle, and the vehicle will 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.

The sequence of parachute deployments begins once the vehicle has fallen through an altitude threshold, currently targeted at 24,000 feet. Once the FBC is jettisoned, the remaining sequence is automatically executed based on timers and other triggers.

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 should be able to reacquire GPS satellites and begin updating its navigation system based on the high-precision measurements. As planned, the GPS receiver is expected to reacquire prior to the critical parachute deployment events.

However, this is the first spaceflight for this 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 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 a backup tertiary trigger 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 [Gay 2013, AAS]. The best performance achieved with this technique produced large altitude spreads in excess of 25,000 feet at drogue deployment. These errors are primarily driven by the attitude error at entry interface (EI).

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

II. Logistic Regression Overview

Logistic regression was identified as a technique that could determine whether the vehicle should initiate the parachute deployment sequence. Logistic regression is a statistical technique used to classify data. It has been employed by statisticians and the machine learning community to classify data based on its characteristics.

Whereas linear regression attempts to fit a straight line (or surface) to a dataset to predict a continuous-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. To use logistic regression, one must provide the algorithm with many examples with their respective labels specifying to which class the example belongs. The simplest version, binary logistic regression, simply distinguishes between two classes of data. This paper shall focus on the binary logistic regression algorithm for the remainder of this paper. The algorithm fits a statistical model to the dataset by minimizing the error between the model and the truth labels. Once the model has converged, the statistical model is ready to classify new data.

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.

Traditionally, analysts have embraced the use of simple comparisons that can be easily encoded into software through the use of IF 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 condtions. If the analyst chooses to capture the logic paths by using IF statements to account for all scenarios, they are forced to exhaustively list out all potential logic paths. This approach is time-consuming and prone to logical errors. Additionally, the threshold values frequently must be tuned manually, a very time-consuming process. Inevitably in the design cycle, there will be vehicle or model changes which may nullify the trigger threshold values. As a result, the analyst 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.

For actual flight implementations, GN&C engineers must contend with noisy data, as opposed to smooth signals. As a result, a simple trigger involving a threshold can easily be inadvertently activated by a noisy signal spiking briefly over the threshold. To mitigate these spurious signals, engineers frequently use auxiliary variables called persistence counters which simply count the number of iterations that a given condition is true. Once the persistence counter reaches a certain value, then the algorithm can trust that the trigger is not being fooled. However, these persistence counters must also be tuned. By their very nature, they add latency to the system in detecting events, adversely impacting system performance.

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 can use many weak signals to determine whether or not the target condition is true.

Furthermore, logistic regression can easily be automated so that triggers can be generated without requiring substantial analysis to determine trigger thresholds. Additionally, these triggers account for the amount of reliability and noise in each signal in determining their weight.

The trigger is automatically developed on the ground using the logistic regression algorithm, and the resulting parameter vector is loaded as a 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.

III. Implementation

Using the 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 IMU measurements during entry. From each trajectory, the relative velocity, flight path angle, sensed aerodynamic acceleration magnitude, time elapsed since 0.2g's, altitude rate, estimated Mach number, and estimated dynamic pressure were recorded. These parameters defined the set of features for the logistic regression algorithm. Each variable is combined into a single vector, known as the feature vector, at each timestamp. 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, the feature vector was associated with its respective label. For all feature vectors above a truth altitude of 25,000 feet (targeted drogue deployment altitude), the associated label was y=0 (do not initiate parachute sequence). All feature vectors below a truth altitude of 25,000 feet were labeled as y=1 (initiate parachute sequence).

The logistic regression algorithm ingested the training examples and fit a multivariate logistic function to the training data. The final output of the algorithm was an 8x1 element 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 MATLAB, the code can be expressed as:

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

This approach is appealing from a flight software perspective because the logic is very inexpensive to compute. Additionally, this approach mitigates the massive complexity associated with the traditional IF-statement approach.

IV. Results

The trigger was trained on 300 trajectories and cross validated against 2,700 trajectories. For each trajectory, the altitude at which the trigger first commanded the parachute deployment sequence was recorded. After recording each altitude, the altitudes were plotted in a histogram to visualize the distribution. The total altitude spread was approximately 5,500 feet. This is a approximately an 80% reduction in altitude error as compared to the previous velocity-based approach.

V. 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. Compared to the previous approach of using a velocity-based trigger, this approach reduces the altitude errors by a factor of five to approximately ±2,750 feet (5,500 feet total). This approach embraces the noise endemic to actual flight hardware, and it allows for the automated generation of robust flight software triggers. The approach is very generic, and the algorithm can be re-applied to solve several other problems for flight software event detection, such as splashdown detection and drogue parachute release.

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