

Regressive Model Approach to the Generation of Test Trajectories

Brian J. Taylor
Institute for Software Research
1000 Technology Drive
Fairmont, WV 26554
Email: btaylor@softwareresearch.org

Bojan Cukic
Dept. of Computer Science and Electrical Engineering
West Virginia University
PO Box 6109
Morgantown, WV 26506-6109
Email: cukic@csee.wvu.edu

Abstract

One aspect of system safety assurance is the application of large test sets. To aid in the development of test cases, researchers have investigated automated test generation systems which decrease the time and cost of acquiring new tests. Many automated systems currently exist, but few address the generation of trajectories of data. A trajectory is defined as a series of data points with each point relying upon the state of the system and previous point values. Examples of trajectories include an airplane's flight path or data describing a nuclear reaction.

This paper describes an original approach to test trajectory generation using statistical regression models. An existing small set of trajectories is utilized to build regressive models. Then, the independent variables of the model can be perturbed, providing the basis for the use of the regressive model in the generation of new trajectories. A case study has been conducted within the scope of testing the sensor failure detection, identification, and accommodation (SFDIA) flight control scheme. We report and evaluate the results of this case study.

1. Introduction

Safety assessment requires a large number of test cases to ensure meaningful coverage of the critical sections of the system's input domain. In the aerospace engineering domain, for example, safety assessment is performed to validate system's conformance with safety related rules, called envelopes. The envelopes are usually derived from previous experience (*experience envelopes*), and known system constraints (*system envelopes*). If any of the system constraint rules is violated, the violation must be reported and investigated, potentially leading to dismissal in the flight qualification process and redesign [9, 10]. Manual development of test cases needed for this type of system assessment is tedious, time consuming and expensive. In these situations, automated test data

generators represent an attractive alternative, provided that building them is feasible.

The types of programs that we consider are process-control programs. The purpose of a control system is to maintain specified properties of the outputs of the process, provided that reference values are given. Properties such as temperature, pressure, flow rates or altitudes are monitored and their values are used to control the process by changing the settings of the active components, such as valves, heaters, thrusters, etc. The architectural solution frequently used for these systems is the control loop paradigm. The controller executes a series of cycles or frames. At the beginning of each frame, it reads inputs from the sensors, then it performs some computations, and at the end of the frame it sends commands to the mechanisms. Computations in each frame depend on the inputs read at the beginning of the frame as well as the values of some internal variables called history or state variables. A complication in the automated test generation stems from the fact that inputs are not independent snapshots of variables from the input domain at the beginning of a frame. Meaningful inputs consist of the *sequences* of snapshots. Values of different variables across the frame boundaries depend on the history of the computation. We call these sequences *input trajectories*. Trajectories define system inputs as a function of time and history. Common examples are real-time control systems, including robot controllers, flight control software and nuclear monitoring systems [8].

Whereas automated test data generation has addressed the creation of individual test points, test trajectory generation has attracted limited attention in the research community. Related work on test trajectory generation includes data specification systems, pathwise test data generators, random test data generators, and specification based test generation [1, 3]. Data specification systems generate test data from a language that describes the input domain. The apparent weakness of this approach is the ability of a system designer to adequately describe the input domain, because this may be as difficult as the

system requirements definition. Pathwise test data generators generate test data that follow execution paths of the program. While this type of testing is suitable for achieving high path coverage, it is not useful for safety assessment based on the extensive coverage of the critical sections of the input domain. Random test data generators are growing in importance because of the popularity of the Cleanroom process, but their randomness is usually confined to the variations of "important" use cases or scenarios [11]. Specification based test data generation uses grammatical rules on the system specifications to generate test cases.

We propose a trajectory generation algorithm as an important tool for the safety assessment of control systems. Our approach is based on the idea of expanding an existing, possibly manually developed, set of test trajectories into a larger set. It makes use of regressive models to create new trajectories, which are statistically similar to the trajectories in the original set, but sufficiently different to represent additional test cases.

The rest of the paper is organized as follows. Section 2 gives a general description of our approach to trajectory generation. Section 3 looks in detail at the individual modules making up the algorithm. Section 4 presents our results from the application of the approach on the SFDIA case study. A brief conclusion with a description of future work is provided in Section 5.

2. Description of the approach

In our approach, regressive models are developed to

determine relationships between independent variables and dependent variables. The dependent variables, representing complete descriptions of generated test trajectories, are predicted by the regressive models. The independent variables are controllable variables, highly correlated to the independent variables.

The existing trajectories are collected prior to the development of the regressive model. These trajectories are clustered into regions based upon the similarities of the independent variables. Because of the correlation the independent variables have with the dependent variables, the dependent variables are correspondingly clustered into regions as well. For example, when considering flight trajectories of an aircraft, if pilot commands are used as independent variables, the corresponding angular rates for the given aircraft would be the dependent variables. In this case, clusters of trajectories would represent different flight maneuvers.

Based on the availability of the previously collected trajectories, regions representing different operational regimes of the system can be represented by clusters. For safety assessment, the regimes on the borders of experience and system envelopes are the best candidates for automated test case generation. Regressive models are then developed to describe each of these clusters, based upon the relationships between the independent and dependent variables. The models are then applied to perturbations of the independent variables, thus producing new test trajectories. A visual representation of the trajectory generation is shown in Figure 1.

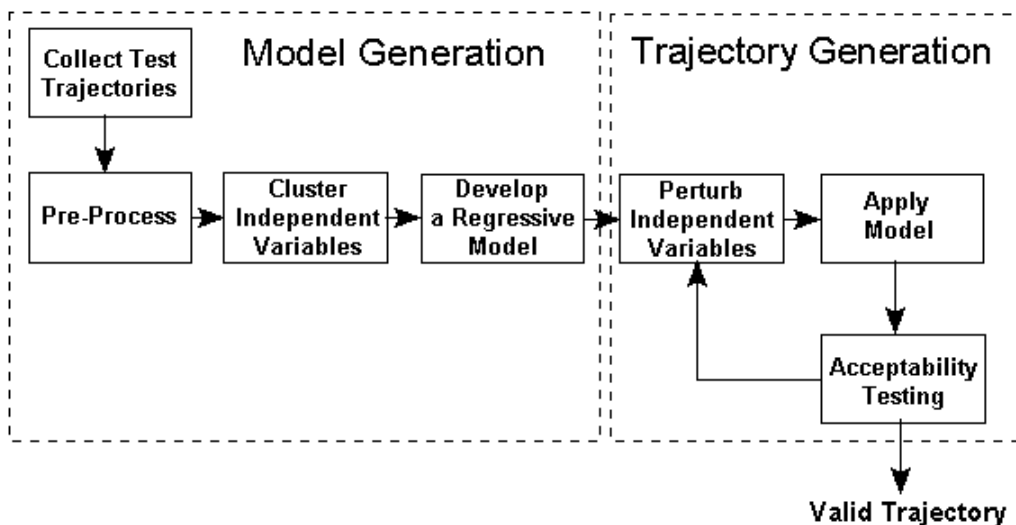


Figure 1. Mapping of Controllable Variables to Input Domain Trajectories

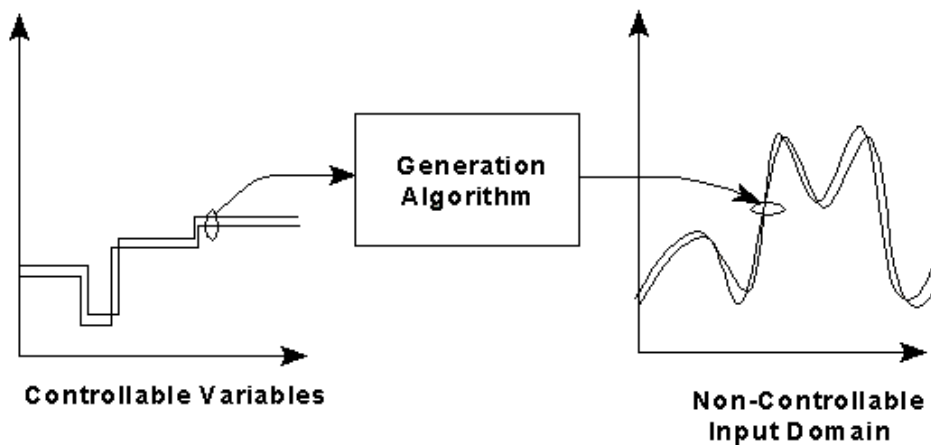


Figure 2. Generation Algorithm Layout

The algorithm can be thought of as a function that transforms one set of inputs, which can be controlled, into a set of trajectories described by the corresponding dependent variables. Sequences of dependent variables will in the safety assessment testing represent inputs to the control program. The program structure of the automated trajectory generator can be seen in Figure 2.

The algorithm is separated into two sections: the model generation and the trajectory generation. The model generation section consists of collecting a set of existing trajectories, preprocessing the data for use by later modules, clustering the existing trajectories and developing a regressive model which can best fit each clustered group. Different regressive models can be used in the developing module, including simple linear, multiple linear, autoregressive, and non-linear regressive models.

The trajectory generation module perturbs the independent variables from the collected set of trajectories to generate new data for the regressive models. The new sets of dependent data generated by the regressive models are checked against a set of acceptability rules. These rules provide reasonableness checks, i.e., they decide if a created trajectory is valid as the input of the system undergoing test. The generation section continues automatically generating new trajectories, as desired.

Each of the two sections is composed of individual modules that are designed to work independently of each other, allowing for a 'refine and replace' approach. As the generation approach is improved, especially the acceptability rules and the model development, the approach can be refined to aid for better regressive model fit or for better acceptance rate of the outputs. When the system changes, the individual modules can be replaced by those refined to work better with the new application.

3. Automated test trajectory generation

3.1 Collection and preprocessing of existing trajectories

Collection of the existing trajectories can be done from various sources, such as those collected from actual system usage, retrieved via a system simulator, or from test cases applied to similar systems in the past. Because regression predicts the relationship between independent and dependent variables, the collected data must consist of the set of trajectories to be expanded (containing dependent variables), and some additional controllable (independent) variables that are correlated with the trajectories. The requirement that the independent variables be controllable allows for their perturbation in the later phases of the algorithm.

Depending upon the data collected, preprocessing of the data may be required before it can be used by the model generation routine. For the clustering algorithm to work correctly, the data sets should be of the same length in terms of the number of frames. This can be accomplished, for example, by truncating data sets to the size of the shortest trajectory. If such a truncation eliminates too much of useful information, other possibilities include the elimination of shorter data sets or interpolation of the data to increase the size of shorter sequences. Data conversion may also be required if test trajectories have been collected from more than one type of source with each using different units of measurement. Noise removal may also be applied at this stage.

3.2 Clustering

By clustering test trajectories into regions representing different operational conditions, regressive models can be defined for each of these regions. Note that such models

define relatively coarse regions of system operation. Each test trajectory generated by the model serves as a system test [5].

Clustering techniques fall into one of the following two categories: hierarchical and non-hierarchical [4]. In non-hierarchical techniques, trajectories are assigned into k arbitrary clusters until the intragroup variances of each cluster reach a minimum. The value of k depends upon a given threshold used to decide the minimum variance allowed within a cluster. When the addition of another trajectory to a cluster increases the group variance, a new cluster is created. The threshold is chosen based upon the desired relation of the members of the clusters. Higher thresholds will certainly allow more trajectories per cluster as lower thresholds increase the number of clusters.

In hierarchical techniques, the set of trajectories is divided into n desired groups. Hierarchical techniques may be either agglomerative or divisive. With agglomerative techniques each trajectory is separated into its own cluster. Neighboring clusters are merged together based upon distance metrics until the desired n groups are attained. Divisive techniques start with all trajectories in one cluster. The cluster is then divided until it reaches the desired number of clusters.

While it can happen in our framework that each cluster contains only one trajectory, having more than one per cluster will allow for better model fitting. Basic steps for the clustering module, shown in Figure 2, consist of the following:

1. Select independent variable to act as clustering parameters.
2. Transform these parameters, if necessary.
3. Remove parameter outliers, if necessary.
4. Select a distance measure.
5. Perform clustering.
6. Interpret results.
7. Change parameters or modify the clustering technique, if necessary.
8. Repeat steps 1 to 7 until the clustering process is satisfactory.
9. Select a representative component for each cluster.
10. Select a representative trajectory for each cluster.

The independent variables that correlate to a trajectory will be the parameters that guide the clustering process. Clustering does not need to be performed upon all of the independent variables, if there are several. Characteristics of important parameters are their significance to the trajectory, and their amount of variance. Low variance, for example, will not provide a distinguishing metric between trajectories and have little impact on clustering.

Clustering allows the elimination of outliers and brings more uniformity to the trajectories within a cluster. An outlier is defined as a trajectory that lies on the outer

bounds of the similar trajectories. Outlier elimination improves the regressive model fit. The regressive model, then, generates dependent variables that lie closer to the centroid of the original cluster. Removal or inclusion of outliers will give different clusters and, consequently, result in different regressive models for each cluster. This choice has important consequences for the rest of the trajectory generation approach, and should be made in accordance with the assessment goals of the specific system.

A standard way of expressing the relationship between trajectories within a cluster is through a distance metric [2, 4]. The distance metric is calculated over an n -dimensional space where n represents the number of parameters used to describe a trajectory. This metric is used to identify which group a trajectory belongs to by determining how close it is to the group centroid. The distance metric selection is just as important as the selection of variables used to perform the clustering. One distance metric may perform well at distinguishing between trajectories in the cluster, while another distance metric may include all trajectories in a single cluster.

The most commonly used distance metric is Euclidean Distance. Assuming that a trajectory X is defined by $\{x_1, x_2, \dots, x_i, \dots, x_k\}$, where x_i 's are the values of variable x at time i , then the distance between two trajectories x and y is given by:

$$d = \left\{ \sum_{i=1}^k (x_i - y_i)^2 \right\}^{\frac{1}{2}}$$

If the trajectories are n dimensional, the distance metric changes to:

$$d = \left\{ \sum_{i=1}^k \left((x_{1i} - y_{1i})^2 + (x_{2i} - y_{2i})^2 + \dots + (x_{ni} - y_{ni})^2 \right) \right\}^{\frac{1}{2}}$$

Other acceptable distance metrics include the Weighted Euclidean Distance and the Chi-Square Distance.

Once the clusters are populated, their centroids must be computed. The centroid is defined as the median value of all independent variables contained within the cluster. If multiple parameters have been used in the clustering technique, the centroid will be multidimensional. Centroids are needed for selecting the representative component of each cluster. The representative component lies closest to the centroid of the cluster, as determined by the distance measure. The representative trajectory consists of the dependent variables that correspond to the representative component. The representative trajectory becomes the input into the regressive model development. Because the regressive model will describe the cluster, any other trajectory of dependent variables chosen will not produce as good a fit across the entire cluster.

3.3 Developing the regressive model

Noting that the representative component and representative trajectory are composed of multiple variables each, different regressive models can be developed with different combinations of those independent-dependent variables. In this paper, our attention is limited to linear regression, but nonlinear models can be utilized as well. For example, let us consider simple linear model. This model predicts a dependent variable's behavior based upon a single independent variable. For each representative trajectory, a simple linear model can be developed for *each pair* of independent-dependent variables. By exhaustively trying all the combinations of independent-dependent variables, the algorithm can select the model that works the best for a given dependent variable. This prevents the algorithm from being 'locked' into any a specific type of regressive models across all of the dependent variables.

After a regressive model is constructed from the representative trajectory, it can be applied to the remaining trajectories in the cluster and analyzed. Cross-correlation analysis looks at the relationship between two sequences of data through a correlation coefficient, r . The stronger the relationship, the higher the value of the coefficient. The values of r can range from 1.00 down to -1.00 with a perfect match occurring at 1.00 and a perfect inverse relationship occurring at -1.00. The equation for correlation is given by:

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{n}}{\sqrt{\left[\sum X^2 - \frac{(\sum X)^2}{n} \right] \left[\sum Y^2 - \frac{(\sum Y)^2}{n} \right]}}$$

where $\sum X$ is the summation of all data points in $X \in \{x_1, x_2, \dots, x_i, \dots, x_k\}$, $\sum Y$ is the summation of all data points in $Y \in \{y_1, y_2, \dots, y_i, \dots, y_k\}$, and $\sum XY$ is the summation of the product of all data points in X and Y , and n is the total number of data points.

Improvement of the correlation between the predicted and actual trajectories can come from applying a smoothing function to the output of the regressive models. Sometimes the output exhibits sharp peaks and transitions due to the linear process. A filter, such as local averaging, can smooth regression output and, generally, yield stronger correlations.

The selection of the best model is based on a cost function. The cost function looks, for example at the computational time needed to generate a new trajectory using the model and the accuracy of the model. The accuracy is determined from cross-correlations between the predicted output of the model and the dependent variables in the set of existing trajectories, as explained

above. An example of a cost function is given by the following expression:

$$P = \bar{T}_{model} + e^{(1.0 - \overline{cor}_{model}) \cdot 10}$$

where T_{model} represents the average time to generate a trajectory using the particular regressive model, and cor_{model} represents the average correlation of that model. P is then the cost of applying the particular regressive model for prediction within the cluster. P is smaller for regressive models that can generate trajectories faster and for models that have higher correlations between the predicted and recorded trajectories. The model is selected by choosing the candidate with the smallest value of P .

3.4 Generating new trajectories

The generation of new test trajectories comes from the perturbation of the independent variables within the cluster. Controllable variables of any trajectory within a cluster are available to undergo perturbation. Methods of perturbation vary and should be chosen based upon the characteristics of the independent variables. Once perturbed, the new data is used as input into the selected regressive model producing a new set of dependent variables. If a smoothing function was applied to the regressive model earlier during the selection module, it should be applied to the new trajectories too.

One of the most important aspects of the entire approach is to determine if the newly created trajectory actually qualifies as a valid test case. All generated trajectories should be compared against the set of rules describing acceptable trajectories, to determine valid tests. These rules must check the dependent variables predicted by the model, as well as the perturbed independent variables used in the generation.

One of the guides in developing acceptability rules can be the distance metric used in the clustering process. Clustering is based on values of the independent variables, so the distance between the perturbed values and the centroid of the cluster can decide if the new perturbed independent data still resides within the cluster. Perturbations that produce independent variables falling outside the cluster can, for example, be discarded.

Acceptability rules defined to analyze the outputs of regressive model can be based on the correlation between generated trajectories to the trajectory defined as the representative trajectory for the cluster undergoing regression. Predictive trajectories that fall outside of an acceptable correlation threshold could be rejected.

A very important set of additional acceptability rules is system specific. These rules would identify any trajectories that violate the definition of the input domain, perhaps by exceeding maximum stress expected to be tolerated by the system. These rules can look at slope

analysis of the trajectories, compare global maximums and minimums, etc.

4. Case study

A case study chosen for this work is the Sensor Failure Detection, Identification, and Accommodation (SFDIA) flight control scheme [6, 7]. The SFDIA scheme is part of an advanced flight control system that uses analytical instead of a physical redundancy to achieve fault-tolerance. Traditional flight control systems cope with sensor failures by duplicating, triplicating, or in some instances, quadruplicating sensor packages. Problems with these approaches include extra hardware costs, power consumption, weight and space considerations. The SFDIA scheme replaces redundant hardware by implementing a neural network based analytical scheme that learns the correlation between values provided by different sensors. In case of a sensor failure, the neural network provides the values that mimic the expected (learned) value of the failed sensor.

Partial assessment of the SFDIA scheme's safety is performed by applying a large number of flight trajectories as system inputs. Each of these test trajectories is composed of an aircraft's rolling, pitching, and yawing moments, where a moment is the angular rate of change that an aircraft experiences during flight.

SFDIA test trajectories were generated by the above described approach to automated test trajectory generation. Pre-existing flight paths were collected from a flight simulation program which allowed for recording of multiple variables during flight maneuvers. This subset of data was then used in the regression modeling. Because of the potential problem with linear regressive models applied to a typical nonlinear domain, we only considered short flight path segments lasting between 20 and 30 seconds. By limiting the duration of each trajectory, we hoped to maintain as much linearity as possible. The short duration of flight segments does not seriously diminish the value of safety assessment, because many critical flight maneuvers can be represented by short trajectories. The loss of nonlinearity can be coped with by introducing nonlinear regressive models, but these are out of the scope of this paper.

Two different linear regressive models were considered for the SFDIA: the simple linear and 2-variable multiple linear regressive models. In the given problem domain, there exists a close relationship between the pilot inputs of aileron deflection, elevator deflection, and rudder deflection to the angular rates we are interested in. This prompted us to choose the pilot inputs as independent variables for the models. One other variable recorded during the simulations was the aircraft's forward velocity,

measured in mach. Forward velocity was used as an independent variable because of its relationship to the pitching moment of the aircraft. The dependent variables for the regressive models of test trajectories for the SFDIA system were the rolling, pitching, and yawing moments of the aircraft.

A total of 17 different flight maneuvers were flown for the data collection step. The flight maneuvers were diving, climbing, banking, and their combinations. In a simulated environment, each maneuver was repeated 10 to 15 times, creating multiple runs. Even though we knew that each maneuver would have similar independent variables, we still applied clustering to each maneuver to eliminate extreme values of the independent variables and enforce certain level of intercluster uniformity required by the customer.

Examples of the performance of regressive models built in our experiments are shown in Figures 3-7.

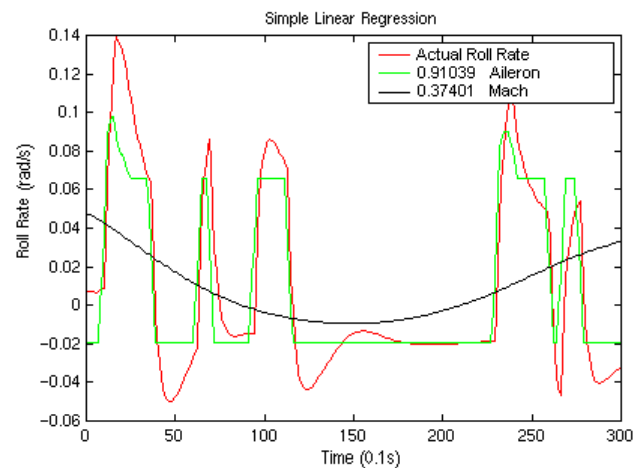


Figure 3. Roll Rate Simple Linear Models for Maneuver 8

Figure 3 shows a simple linear regressive model developed for roll rate prediction from flight maneuver 8. By using aileron deflection as an input to the model, the model reached the correlation of 91.039%, whereas using mach as an input to the model only achieved the correlation of 37.4%.

Figure 4 shows how the correlation of the regressive model can change by adding a level of complexity to the simple linear model and using two independent variables instead of one. The 2-variable multiple linear regression model, using the aileron and elevator deflections as inputs, has a correlation to the actual dependent variables of 91.61%. If aileron and rudder deflections are used, the correlation is 90.57%. Aileron and mach, used as independent variables, produce a 90.54% correlation.

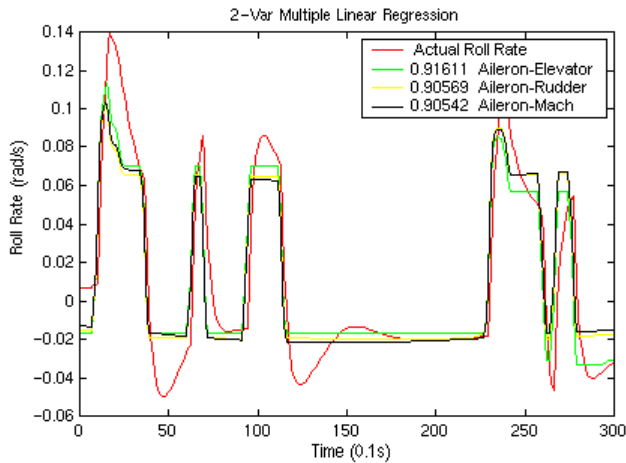


Figure 4. Roll Rate Multiple Linear Models for Maneuver 8

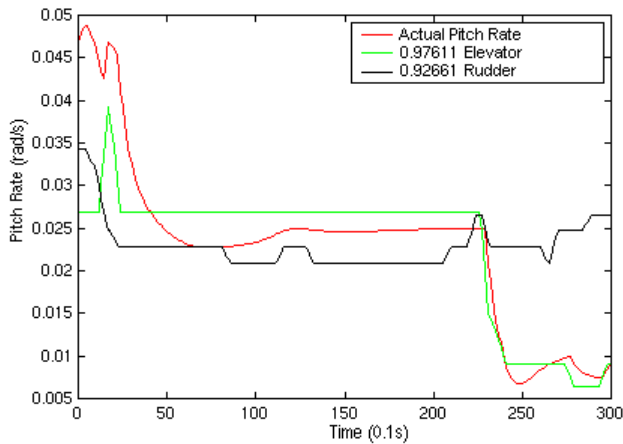


Figure 5. Pitch Rate Simple Linear Models for Maneuver 8

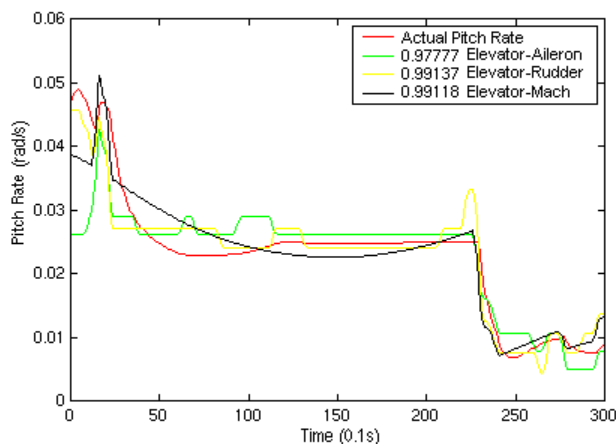


Figure 6. Pitch Rate Multiple Linear Models for Maneuver 8

Sample regressive models from the pitch rate model development are shown in Figures 5 and 6. Elevator deflection as an independent variable for a simple linear

model was able to correlate to 97.6% of the actual value of pitch rate, while rudder deflection had a 92.66% correlation. The addition of the second independent variable to the model apparently improves the prediction. In Figure 6, when both elevator and rudder deflections are used, the model correlates 99.14% to the actual pitch rate with other 2-variable combinations performing just as well.

Yaw rate prediction is shown only for a 2-variable model in Figure 7. The linear models had a more difficult time fitting to the yawing moment trajectories. The best correlation came from the rudder and elevator deflection combination of independent variables, achieving the correlation of 94.19%.

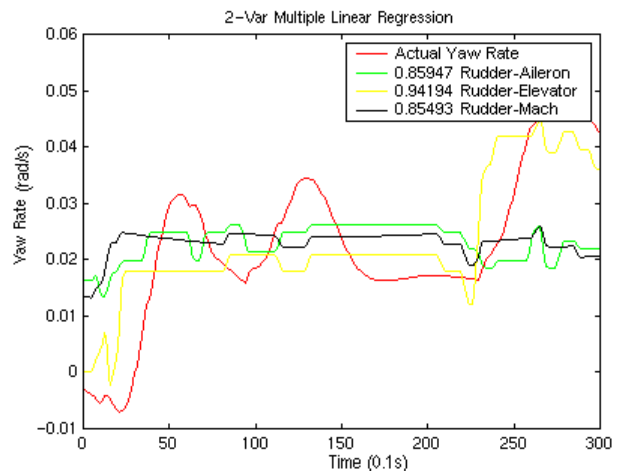


Figure 7. Yaw Rate Multiple Linear Models for Maneuver 8

Two sets of acceptability rules were developed for the SFDIA trajectory generation. The first set tested the acceptability of the pilot inputs. The second set was developed to test the acceptability of the generated roll, pitch, and yaw rates. The acceptability tests applied to the perturbed pilot inputs checked the deflections to determine if they exceeded the acceptable range which an aircraft might experience during normal flight. The test was defined by an upper and lower limit bound on the pilot's deflection, as shown below:

$$-0.053 \text{ rad} \leq \text{pilot deflection} \leq 0.053 \text{ rad}$$

If a perturbed input exceeds the threshold, it is discarded from the set of test cases. The choice to use 0.053 radians is based upon typical deflections of surface areas in commercial jetliners.

The second acceptability test checked if the perturbed deflections remained within the clusters determined earlier. By applying the Euclidean distance metric, the distance of each of the perturbed deflections from the cluster centroid was calculated. If a distance exceeded the intragroup variance threshold, then that particular perturbed deflection was eliminated.

Only one set of acceptability rules was created for application to the generated trajectories. A limit was placed upon the degree in which a generated trajectory differed from its corresponding actual angular rate. Essentially, if a generated trajectory fell below an 85% correlation with a trajectory from the original collected dependent variables, that trajectory was rejected.

Maneuver	Roll %	Pitch %	Yaw %
1	70.0	90.0	60.71
2	10.0	94.0	66.0
3	91.11	91.11	11.11
4	93.75	93.75	0.0
5	93.75	93.75	62.5
6	80.0	92.5	0.0
7	0.0	85.0	2.5
8	93.75	93.75	93.75
9	90.0	90.0	84.0
10	92.5	92.5	0.0
11	86.25	86.25	17.5
12	61.25	86.25	1.25
13	71.25	93.75	93.75
14	0.0	91.25	60.0
15	78.75	78.75	72.5
16	68.75	75.0	80.0
17	31.25	93.75	12.5

Table 1. Percentage of Acceptable Trajectories for Maneuvers 1-17

Table 1 shows the percentages of acceptable generated trajectories for roll, pitch, and yaw rate predictions with each maneuver having its own regressive model developed as either a simple linear or 2-variable multiple linear regressive model.

Because each of the generated angular rates is related to a specific maneuver, the lowest acceptability rate was the limiting factor for some maneuvers. For example, in cases 4 and 6 roll and pitch were highly acceptable, but not the yaw rate, thus creating no new tests for that maneuver. Only a few maneuvers experienced poor results, while most others had very high acceptance rates, as defined by the specific acceptability rules.

5. Conclusions

Early results show that the trajectory generation algorithm is able to produce new test trajectories faster than they could be simulated or collected from actual usage.

Areas of further study include looking at the cost function used to select the best regressive model and perhaps modifying it to include additional calculations such as least square errors between the actual and

predicted trajectories or perhaps replacing the correlation computation with a simple Euclidean distance metric. There also needs to be further refinement and addition to the acceptability rules as applied to the SFDIA scheme. Flying requirements were not used in the development of acceptability rules and these may have a considerable impact on the choice of valid generated trajectories. More sophisticated regressive models can also be constructed to attempt better yaw rate prediction such as 3-variable multiple linear regression, autoregressive moving-average models and non-linear models.

Acknowledgements

This work was partially supported by NASA through cooperative agreement #NCC2-979 and through the research grant No. NAG4-163.

Bibliography

- [1] Demillo, R. A. W. Michael McCracken, R. J. Martin, John F. Passafiume. *Software Testing and Evaluation*. Benjamin Cummings. Menlo Park, Ca. 1987.
- [2] Hartigan, J. A., *Clustering Algorithms*. Wiley. 1975.
- [3] Ince, D. C., "The Automatic Generation of Test Data," *Computer*, pp. 63-69, Vol. 30, No. 1, 1987.
- [4] Jain, R. *The Art of Computer Systems Performance Analysis, Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, Inc. New York, 1991.
- [5] Lyu, M. R. *Handbook of Software Reliability Engineering*. IEEE Computer Society Press. McGraw-Hill. New York, 1996.
- [6] Napolitano, M.R., Molinaro, G., Innocenti, M., Martinelli, D. A Complete Hardware Package for a Fault Tolerant Flight Control System Using On-Line Learning Neural Networks. *IEEE Control Systems Technology*, Jan 1998.
- [7] Napolitano, M.R., Neppach, C.D., Casdorph, V., Naylor, S., Innocenti, M., and Silvestri, G. "A Neural Network-Based Scheme for Sensor Failure Detection, Identification and Accomodation", *AIAA Journal of Control and Dynamics*, Vol 18, No 6, pp. 1280-1286, 1995.
- [8] Parnas, D., Schouwen, A. J., and Kwan, S. P. Evaluation of Safety-Critical Software. *Communications of the ACM*, vol. 33, no. 6, June 1990, pp.636-648.
- [9] Rockwell International, "Software Requirements Specifications, Flight design and Dynamics ascent discipline Ascent Subsystem Day of launch function (DOLILU-II) DIVDT program function: Ver: 4.2 FD", March 1993.
- [10] Cukic, B., Chakravarthy, D., McCaugherty, D., "Reliability Prediction of a Trajectory Verification System", *IEEE Workshop on Application Specific Software Engineering and Technology (ASSET'98)*, Dallas, TX, March 1998.
- [11] J. A. Whittaker, J. H. Poore, "Statistical Testing for Cleanroom Software Engineering," *Proc. 25th Hawaii International Conference on System Sciences*, Kanai, HI, Jan. 1992, pp. 428-436.