Immunity-Based Aircraft Fault Detection System

D. Dasgupta[*], K. KrishnaKumar[†], D. Wong[‡], M. Berry[§]

Abstract

In the study reported in this paper, we have developed and applied an Artificial Immune System (AIS) algorithm for aircraft fault detection, as an extension to a previous work on intelligent flight control (IFC). Though the prior studies had established the benefits of IFC, one area of weakness that needed to be strengthened was the control dead band induced by commanding a failed surface. Since the IFC approach uses fault accommodation with no detection, the dead band, although it reduces over time due to learning, is present and causes degradation in handling qualities. If the failure can be identified, this dead band can be further minimized to ensure rapid fault accommodation and better handling qualities. The paper describes the application of an immunity-based approach that can detect a broad spectrum of known and unforeseen failures. The approach incorporates the knowledge of the normal operational behavior of the aircraft from sensory data, and probabilistically generates a set of pattern detectors that can detect any abnormalities (including faults) in the behavior pattern indicating unsafe in-flight operation. We developed a tool called MILD (Multi-level Immune Learning Detection) based on a real-valued negative selection algorithm that can generate a small number of specialized detectors (as signatures of known failure conditions) and a larger set of generalized detectors for unknown (or possible) fault conditions. Once the fault is detected and identified, an adaptive control system would use this detection information to stabilize the aircraft by utilizing available resources (control surfaces). We experimented with data sets collected under normal and various simulated failure conditions using a piloted motion-base simulation facility. The reported results are from a collection of test cases that reflect the performance of the proposed immunity-based fault detection algorithm.

[*] Professor, Division of Computer Science, The University of Memphis, Memphis, TN, dasgupta@memphis.edu
[†] Scientist, NASA Ames Research Center, MS 269-1, Moffett Field, CA 94035-1000, Associate Fellow
[‡] Graduate Student, Division of Computer Science, The University of Memphis, Memphis, TN
[§] Engineer, QSS Inc, NASA Ames Research Center, MS 269-1, Moffett Field, CA 94035-1000

1. Introduction

In the last 30 years, at least 10 aircraft have experienced major flight control system failures claiming more than 1100 lives. Early detection of a fault or damage of aircraft subsystems is very crucial for its control and maneuver during the flight. These events include sudden loss of control surfaces, engine failure, and other components that may result in abnormal flight operating conditions. Monitoring and detection of such events is necessary in order to achieve acceptable flight performance and higher flight survivability under abnormal conditions. The Intelligent Flight Control (IFC) research program at NASA Ames began in 1992 to address the need to examine alternate sources of control power to accommodate in-flight control system failures. These events can include sudden loss of control surfaces, engine thrust, and other causes that may result in the departure of the aircraft from safe flight conditions. The major feature of IFC technology is its ability to adapt to unforeseen events through the use of an adaptive neural flight control architecture.

In this study, an extension to prior work is considered, which involves an innovation for fault detection: an Artificial Immune System (AIS) algorithm. The goal is to apply the immunity-based fault detection algorithm to improve the fault tolerance capabilities of the Intelligent Flight Controller (IFC) architecture. Prior studies had established the benefits of intelligent flight control (IFC)[12-14]. One area of weakness that needed to be strengthened was the control dead band induced by commanding a failed surface. Since the IFC approach uses fault accommodation with no detection, the dead band, although reducing over time due to learning, is present and causes degradation in handling qualities[3, 6]. This also makes outer loop control design challenging . The dead band problem could be eliminated with reliable fault identification.

This paper describes an immunity-based approach that can detect a broad spectrum of known and unforeseen faults. Once the fault is detected and identified, a direct adaptive control system would use this detection information to stabilize the aircraft by utilizing available resources (control surfaces). This fault detection algorithm is a probabilistic approach (motivated by the negative selection mechanism of the immune system) in order to detect deviations in aircraft flight behavior patterns. In particular, the detection system learns the knowledge of the normal flight patterns from sensory data, to generate probabilistically a set of (novel) pattern detectors that can detect any abnormalities (including faults) in the behavior pattern of the aircraft flight.

2. Real-Valued Negative Selection (RNS) Algorithm

The negative selection algorithm is based on the principles of self-non-self discrimination in the immune system. The negative-selection algorithm can be summarized as follows[19,20]:

- Define self as a collection S of strings of length l over a finite alphabet, a collection that some way models the normal operation (or system behavior). For example, S may represent the normal operating conditions of an aircraft.
- Generate a set R of *detectors*, each of which fails to match any string in S. Instead of exact or perfect matching, the method uses an approximate matching rule, in which two strings match if and only if they are within a certain distance r, where r is a suitably chosen parameter.
- Monitor S for changes by continually matching the detectors in R against new observations of S. If any detector ever matches, then an abnormality is known to have occurred, because the detectors are designed not to match any similar sample strings in S.

In this study, we applied a real-valued Negative Selection Algorithm (NSA) to prior work on intelligent flight control for fault detection and isolation. The RNS algorithm uses the self/non-self space that corresponds to a subset of R^n, specifically $[0, 1]^n$. A detector is defined by an n-dimensional vector that indicates the center with varying radius; therefore, a detector can be seen as a hypersphere in R^n. Accordingly, the RNS algorithm tries to evolve a set of variable-size detectors that cover the non-self space. This is accomplished by an iterative process that updates the position of the detector driven by two goals:

- Move the detector away from self points.
- Keep the detectors separated in order to maximize the covering of non-self space.

The RNS detector generation starts with a population of initial (random) detectors, which are then matured through an iterative process. The centers of these detectors are chosen at random. The radius of a particular detector is defined in terms of Euclidian distance to its nearest neighbor in the training dataset (self sample). The logical steps of the algorithm are shown in Figure 2, which are described in more detail elsewhere.

In each of the iterations, first, the center of each detector is successively adjusted by moving it away from training data, and away from other detectors (in order to reduce the amount of overlap among detectors). This process is continued until the amount of coverage and overlapping stabilizes. The set of detectors are ranked according to the size of their radii. Then a greedy algorithm is used to select a new population of detectors, i.e., detectors with large radii are selected to go to the next generation, but they will only go to the next generation if their overlapping rate with other detectors is small. Detectors with very small radii are discarded and replaced by clones of better-fit detectors. The clones of a detector are moved at a fixed distance to produce new detectors in its close proximity. Moreover, new areas of the non-self space are explored by introducing some random detectors.

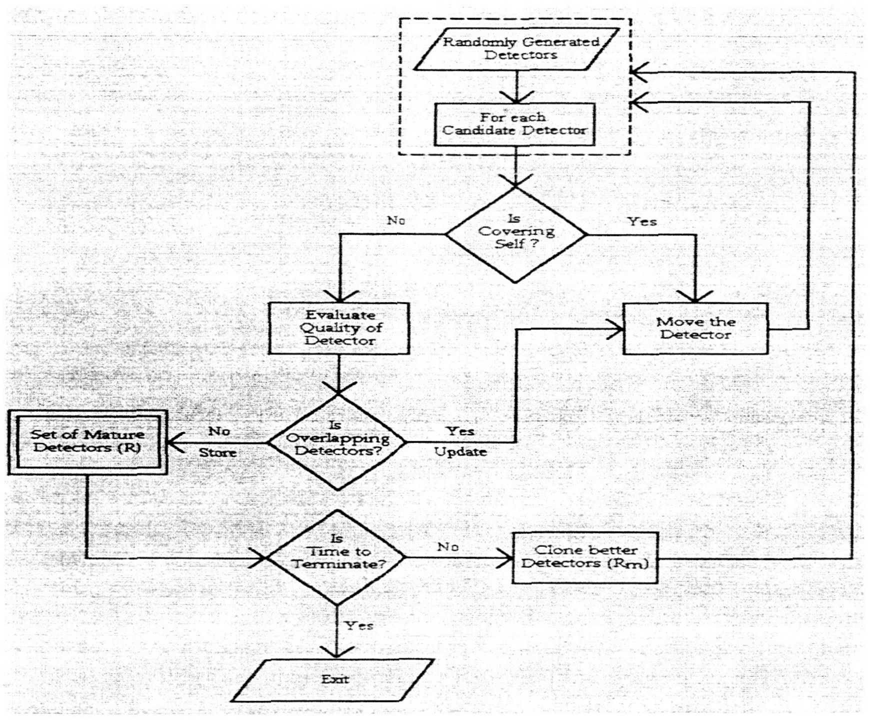

Randomly Generated Detectors
For each Candidate Detector
Is Covering Self ?
No
Yes
Evaluate Quality of Detector
Move the Detector
Set of Mature Detectors (R)
No
Store
Is Overlapping Detectors?
Yes
Update
Is Time to Terminate?
No
Clone better Detectors (Rm)
Yes
Exit

Figure 1: Flow diagram shows the steps of the RNS algorithm.

3. Failure Detection in the IFC Architecture

To provide a real-time system capable of compensating for a broad spectrum of failures, researchers at NASA Ames have investigated a neural flight control architecture shown in Figure 2 for both flight and propulsion control[12-14]. The concept was to develop a system capable of utilizing all remaining sources of control power after damage or failures. The Intelligent Flight Control (IFC) system uses an optimal allocation technique to ensure that conventional flight control surfaces will be utilized under normal operating conditions. A linear programming theory was used in conjunction with a cost function approach to provide generalized control reallocation over an arbitrary number of aircraft surfaces for the real-time control allocation problem. Under damage or failure conditions, the system may allocate flight control surfaces, and incorporate propulsion control, when additional control power is necessary for achieving desired flight control performance.

Another innovation in the work cited is the use of adaptive critics for reference model adaptation. In the case of extreme damage to the plane, the original reference model may over-drive the remaining control surfaces and cause instability in the plane. Adaptive critic technology was utilized to modify the reference model in these cases.

The IFC architecture was evaluated utilizing a full-motion simulator available in the Crew Vehicle Systems Research Facility (CVSRF) at NASA-Ames Research Center. The Advanced Concepts Flight Simulator (ACFS) was modified to accommodate a model of a Boeing C-17 aircraft[10, 11]. The simulator is equipped with a six degree-of-freedom motion system, programmable flight displays, digital sound and aural cueing system, and a 180-degree field of view visual system. The Boeing C-17 is a fly-by-wire transport aircraft with a stabilizer, four elevators, two ailerons, eight spoiler panels and two rudders, for a total of 17 surfaces used for active control. Slats, flaps and engines were not used by the IFC for control of the aircraft configuration[12-14].

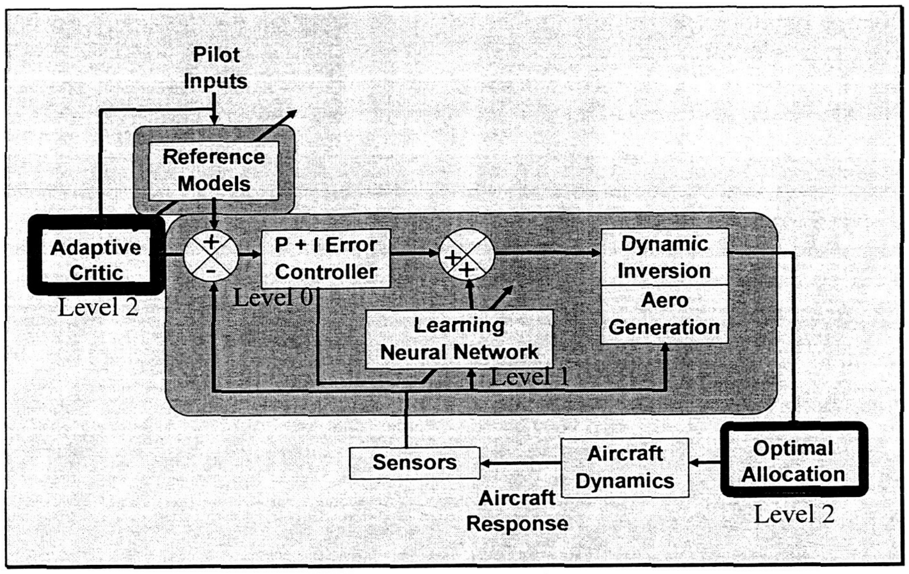


Figure 2: Intelligent flight control architecture

In the study reported here, an extension to prior work is considered. This involves using the AIS algorithm for fault detection. As stated earlier, one area of weakness that needed to be strengthened is the control "dead band" induced by commanding a failed surface. If the failure can be identified, the dead band could be minimized to ensure rapid fault accommodation and better handling qualities. The modification to the existing IFC architecture is shown in Figure 3, where an immunity-based fault detection system is included. The purpose is to use RNS to detect control surface area loss caused by damage (or failure) and other causes that may result in the departure of the aircraft from safe flight conditions. Once the failure is detected and identified, the Intelligent Flight Controller (IFC) then utilizes all remaining source of control power necessary to achieve the desired flight performance.

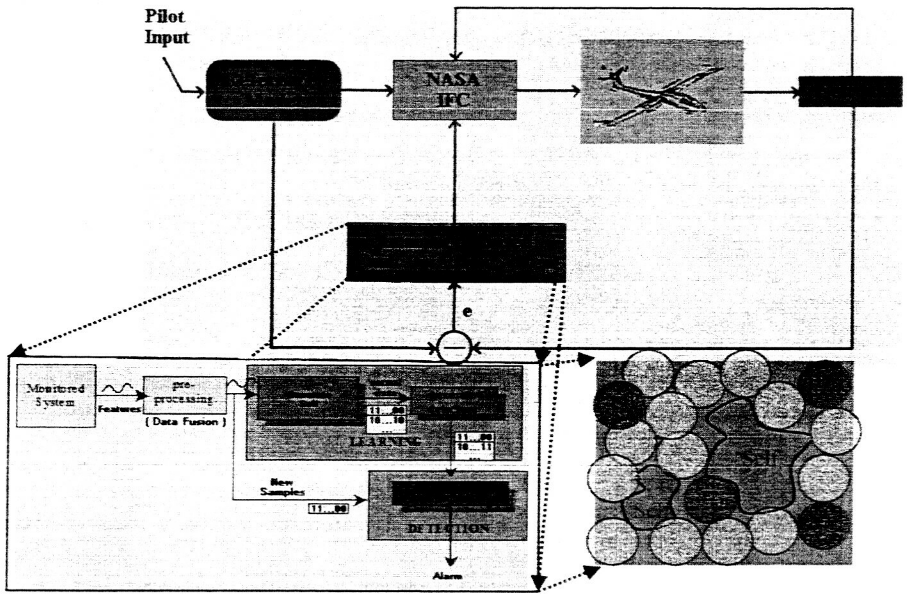


Figure 3: IFC architecture with AIS fault detection modules. The concept of self and non-self space and the generation of detectors in non-self space are illustrated. Here F1, F2, etc. represent different failure conditions.

4. MILD: Implementation Details

We developed the MILD (Multi-level Immune Learning Detection) software based on the real-valued negative selection algorithm described in section 2. The following flow diagram (Figure 4) illustrates the MATLAB implementation of different functional components.

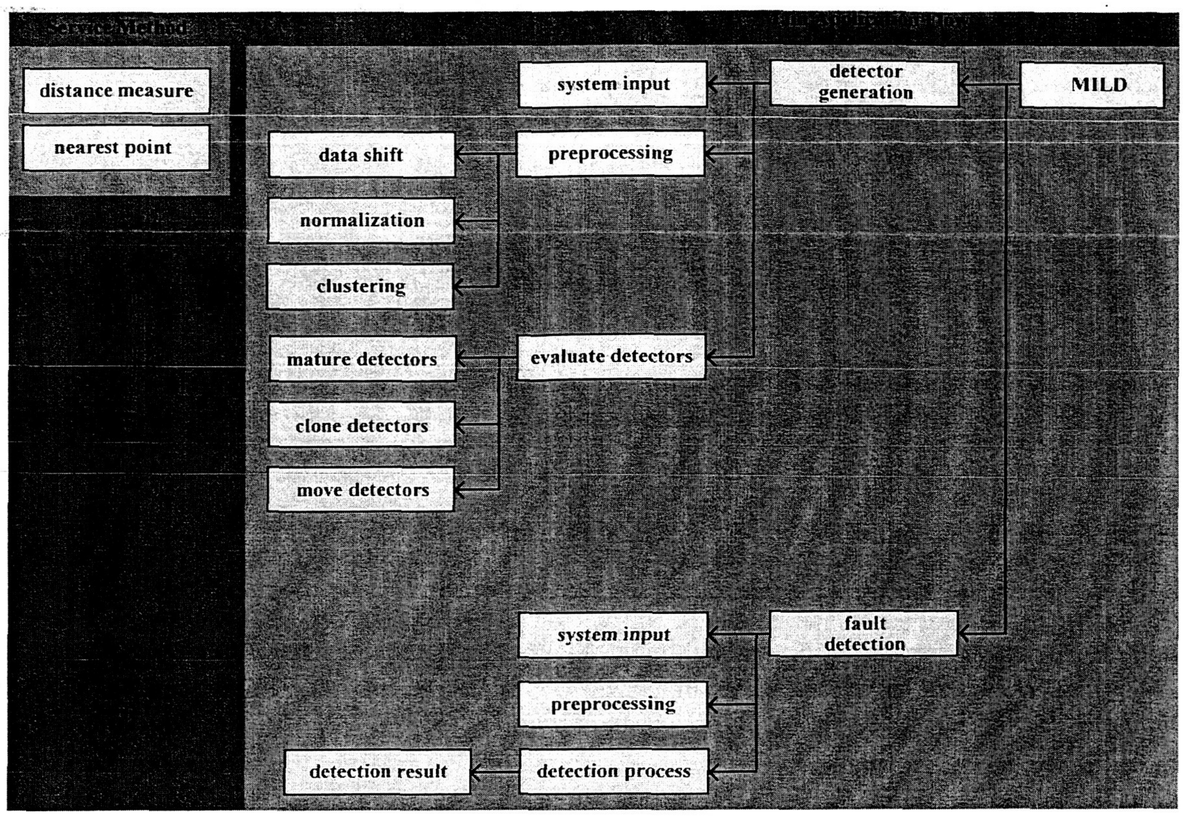


Figure 4: Matlab implementation details and illustrating the sequence of function calls.

4.1 The main functions of the MILD tool

There is a main GUI from which the user can select either the detector generation GUI or the fault detection GUI in order to go to the corresponding phase.

detector generation GUI: implements the detector generation scheme based on the real-valued negative selection algorithm, this function triggers 3 sub-functions:

- detector generation inputs: includes the training file name, window size, overlapping value, percentage increase, number of initial detectors, stopping criterion (maximum number of iterations)
- preprocessing: the training data is preprocessed based on the value of the input parameters and 3 routines are called in order to accomplish the preprocessing task:
    1. data shift: shifts the training data based on the window size and overlapping value specified by the user
    2. normalization: normalizes the training (and test) data between 0 and 1
    3. clustering: clusters the training data
- evaluate detectors: evaluates the detectors based on the Minkowski distance measure. This function has 3 sub-functions that handle the evaluated detectors accordingly:
    1. mature detectors: collects mature detectors
    2. clone detectors: clones the good detectors based on each good detector's radius
    3. move detectors: moves the infeasible detectors in order to cover the non-self space

fault detection GUI: detects the anomaly of the testing data by using the mature detectors that were generated by the detector generation function. The detection function uses 3 sub-functions:

- fault detection inputs: includes the testing file name, mature detectors, buffering, detection score
- preprocessing: shifts and normalizes the testing data
- detection process: detects anomalies by the mature detectors, shows the detection result in another function.

- detection result: displays the alert signal when a fault happens and also indicates whether the fault is detected by the regular detectors and/or the edge detectors

Utility routines

- distance measure: measures the Minkowski distance of multi-dimension data points
- nearest point: finds the nearest data point for one local data point to all other data points

We first examined the detector coverage using some simple 2-D data, Figure 5 exhibits the coverage of the non-self space with bigger and smaller detectors. This snapshot is taken after 20 iterations, and there are 7 bigger size detectors generated which appear to cover most of the non-self space, while smaller size detectors generated slowly as the run progresses. A big portion of larger detectors fall outside the unit hypercube, but they are necessary in covering the edges.

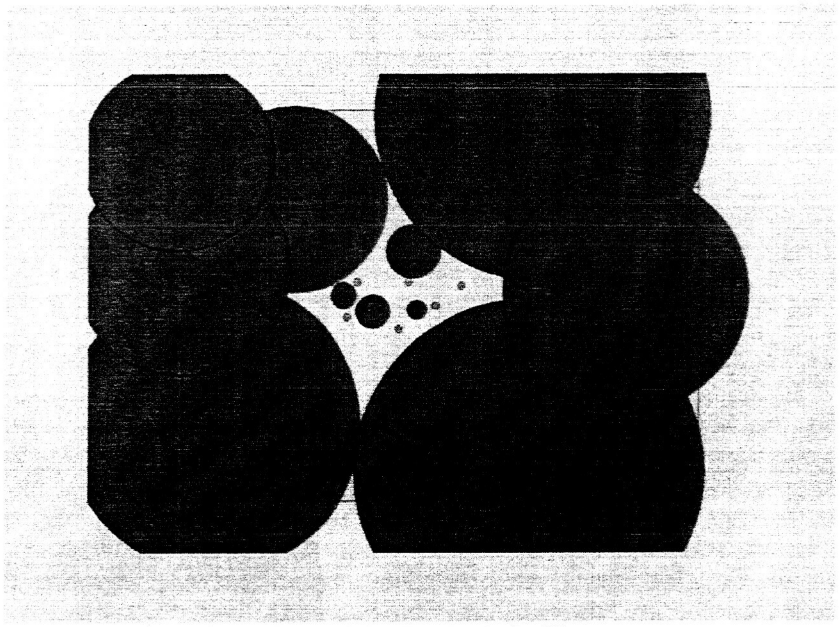


Figure 5: Coverage of the detectors in 2D space

5. Experiments and Results

The fault detection system takes a real-valued data set as input; and extracts the important semantic information by applying a data fusion and normalization techniques. The reduced information is then represented as a collection of strings, which forms the self set (normal patterns) that can be used to generate a diverse set of detectors. The set of detectors are subsequently used for detection of different types of faults (known and unknown faults). The aim is to find a small number of specialized detectors (as signatures of known failure conditions) and a bigger set of generalized detectors for unknown (or possible) fault conditions.

5.1 Aircraft Simulated Data

The goal of this work was to evaluate the fault detection capabilities of the RNS algorithm outlined earlier. Five C-17 pilots from the Air Force, and from NASA were used to test the IFC architecture shown in Figure 2. Detailed flight data was collected for post-flight analysis. These same data sets were also utilized for studying the efficacy of the fault detection algorithm. The data from the simulator is collected at the rate of 30 Hz (frames per second). Since the data is from a man-in-the–loop simulation, there is an extra control (the human pilot) that simply cannot be removed from the data.

The failure scenarios for the C-17 test are outlined in Table 1. The scenarios were designed to test performance of the controllers relative to primary failures in all 3 axes as well as a failure sequence which would couple over all the axes. The full tail failure and wing failure scenarios were utilized during normal landing operations, and the engine failure on a takeoff sequence.

Table 1. Failure scenarios for the C-17 Experiment considered for this study.

C17 Scenario	Scenario Characteristics: Winds 190 @ 10, light turbulence
Full Tail Failure	Full tail failure. Stabilizer failed at trim. 2 rudders, 4 elevators failed at 0 deg.
Wing Failure	2 ailerons and 8 spoiler panels failed at 0 deg.
Engine Failure	Two engines out on one side on takeoff, minimum climb speed + 10Kts.

Before starting the sequence, the aircraft has been flying in level flight with the autopilot, proceeding at 250 kts on heading 330 magnetic at 8000 feet MSL altitude. The IFC is active. At approximately 160 seconds into the data run, all ailerons and spoilers are failed at their idle position for the wing failure case and both rudders and all elevators and stabilizer are failed at their idle position for the tail failure case. The flight control software must achieve its roll and pitch control, for the respective failures, using nontraditional deployment of the remaining surfaces [14].

The parameters considered for the fault detection study included body-axes commanded rates, actual aircraft body-axes rates, and corresponding neural network outputs. The training data were collected by combining data from non-failed axes using several single axis failure cases. For the single axis failure cases, the failures happened 2-3 seconds after the beginning of data collection. No experimental data were collected for cases without a failure. The test data were generated by windowing the data 1.5 seconds before and 1.5 seconds after the failure. As a result, the "normal" part of the test data looks similar to the training data for each case. It is to be noted that some of the entries are zero at the beginning of the data set, because the pilot has likely not entered a command to move by then, and the aircraft would still be flying straight and level.

5.2 Experiments

The sensor parameters considered for these experiments included body-axes roll rate, pitch rate and yaw rate, where both expected and observed values are monitored and error rate (e) is calculated. If these error rates are abnormal, the NS fault detection algorithm should detect them indicating possible failures.

For all experiments, the number of self detectors generated was 103. After generating the detectors, we examined the effectiveness of the detectors in identifying various faults. Figure 6 shows the error rates (e) for the body-axes rates that are considered as the normal data (self), and used to generate the detector set.

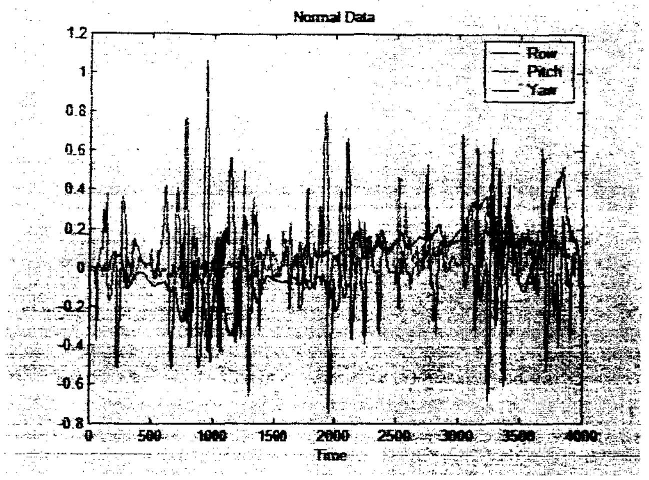
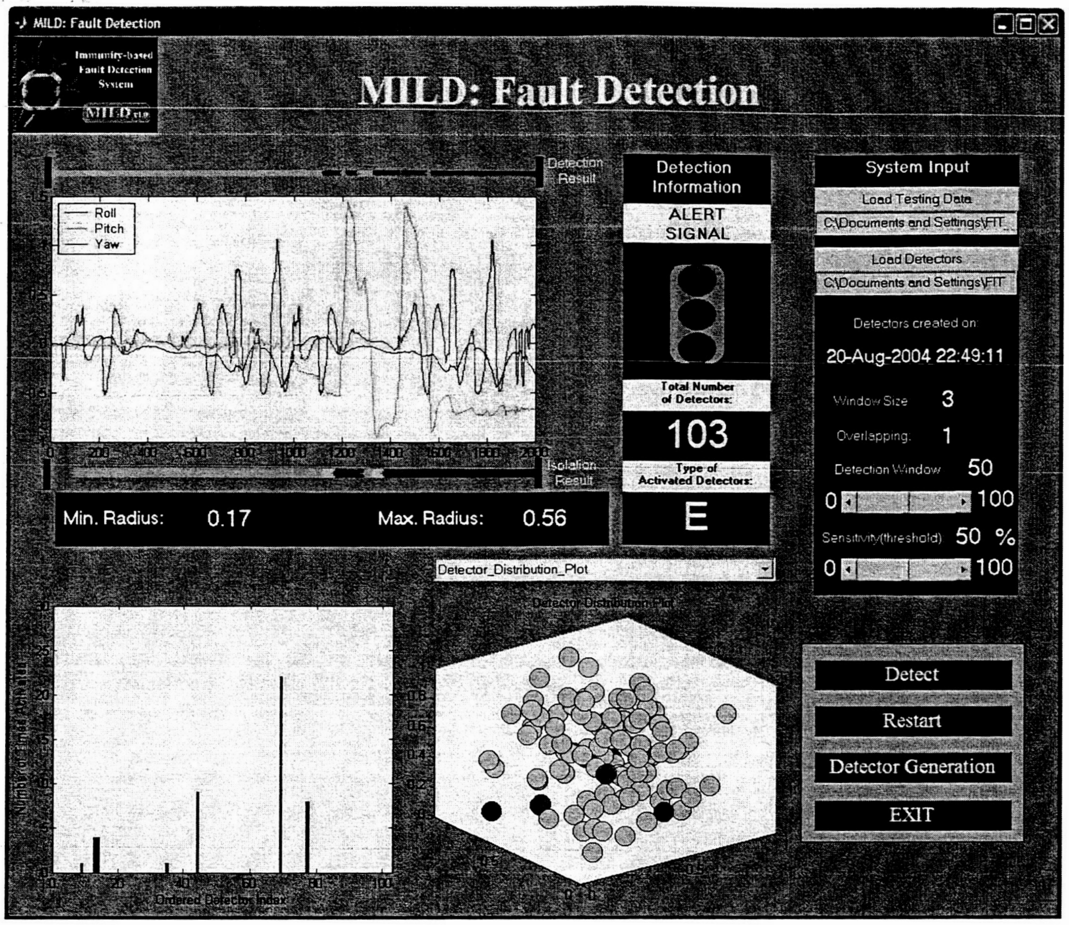


Figure 6. Shows the normal data pattern for three sensory data (roll, pitch and yaw) observed at NN output.

Figure 7 illustrates the performance of the detection system when tested with "full tail failure" data, where this type of fault is manifested in pitch error rate (starting at the 1200th time step). The graph also shows the number of detectors activated (lower bar chart) as significant deviations in data patterns appear. The bar chart shows the arrangement of the detectors with increased radius.

Figures 8-10 display results for other failures. There are two types of results reported for different fault detection scenarios. One is the detection result (horizontal bar on the top), which displays raw detection result for each data point, and the other one is isolation result (horizontal bar at the bottom), which displays detection signal based on the buffering and sensitivity parameters defined by the user. All types of faults occur right after the 1000th time stamp, but based on the buffering and sensitivity, the anomaly alarm is triggered at various times as reflected in each of the detection result figures. For example, for the tail failure detection result, the actual fault occurs at the 1000th time stamp, the detection occurs at the 1000th stamp, the isolation result gives warning at the 1000th time stamp as well. However, the actual fault alarm given by the isolation result is at the 1200th time stamp.

The most obvious fault and the easiest to detect is the left engine fault, because the fault involves all three axes (roll, pitch and yaw). A relatively challenging fault type is the full tail failure marked "tail_3 fault," because it is very similar to the pattern of the normal data. However, our system is capable of detecting all kinds of possible faults, regardless of the change of the pattern, the instability of the amplitude or the frequency of the oscillation.



ALERT SIGNAL

Total Number of Detectors: 103

Type of Activated Detectors: E

Min. Radius: 0.17 Max. Radius: 0.56

Detectors created on: 20-Aug-2004 22:49:11

Window Size 3

Overlapping: 1

Detection Window 50

Sensitivity(threshold) 50 %

Figure 7: Shows the detection of Tail_3 failure with number of activated detectors

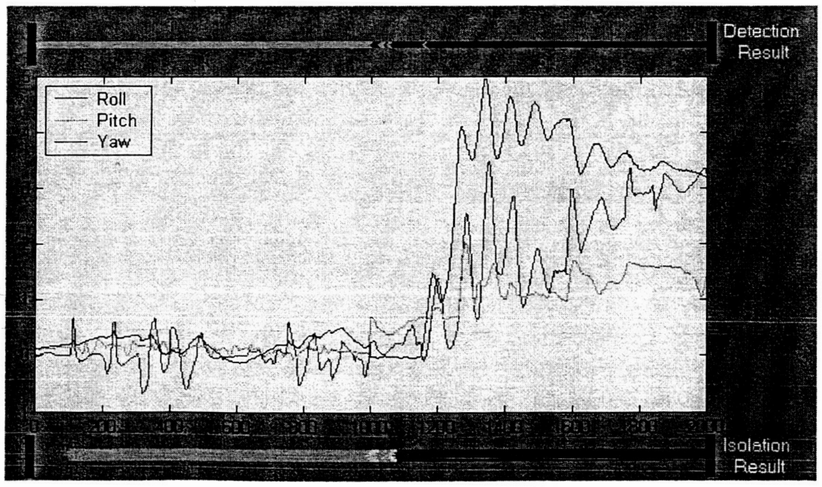


Figure 8: Left Engine failure case

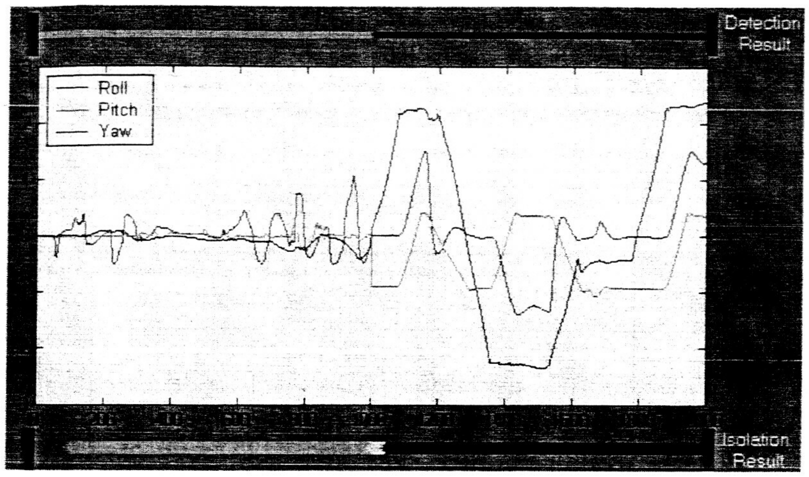


Figure 9: A tail failure case

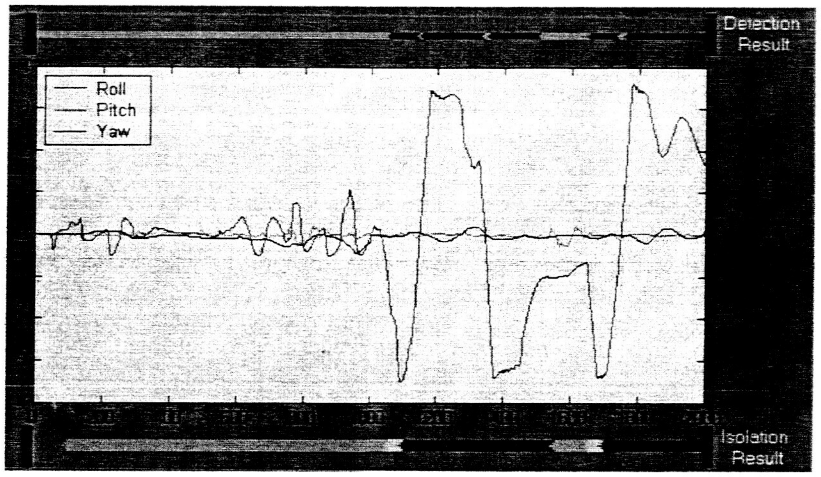


Figure 10: A wing failure case

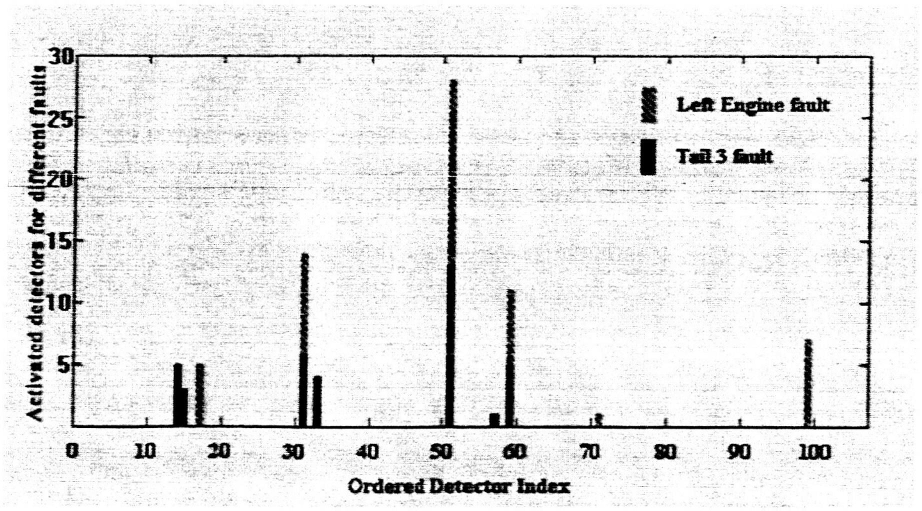


Figure 11: Activated detectors for different types of faults

Figure 11 shows the number of detectors activated for two different faults—engine and tail faults. We observed that a number of detectors get activated in each case. There appears to be three possible reasons: some overlap was allowed among detectors; faulty data may be clustered at several locations in the non-self space, and the same detectors may be activated for more than one fault case. Table 2 gives statistical results of several test runs of MILD with different faulty datasets demonstrating that the false alarm rates are very small in all test cases.

Table 2: Statistical results of 10 runs

Fault Type	Activated Detectors	Detection Rate (mean)	Detection Rate (std)	False Alarm (mean)	False Alarm (std)
Left Engine	10	97.8 %	1.24	0.15 %	0.33%
Tail 3	9	94.7 %	1.03	0.76 %	0.26%
Tail 1	7	91.8 %	1.43	1.04%	0.47%
Wing 3	3	95.6 %	1.29	0.43%	0.36%

Conclusions

We investigated an immunity-based approach that can detect a broad spectrum of known and unforeseen failures. Once the fault is detected and identified, a direct adaptive control system would use this detection information to stabilize the aircraft by utilizing available resources (control surfaces). The proposed intelligent fault detection algorithm is inspired by the principles of the biological immune system. This fault detection algorithm is a probabilistic approach motivated by the negative selection mechanism of the immune system. The detection system learns the knowledge of the normal flight patterns from sensory data, and generates probabilistically a set of (novel) pattern detectors that can detect any abnormalities (including faults) in the behavior pattern of the aircraft flight. The long-term goal is to use RNS to detect control surface area loss caused by damage (or failure) and other causes that may result in the departure of the aircraft from safe flight conditions. Once the failure is detected and identified, the Intelligent Flight Controller (IFC) then utilizes all remaining source of control power necessary to achieve the desired flight performance.

There are many techniques for aircraft failure detection[1]. One of the drawbacks of existing fault detection and isolation (FDI) based approaches is that they cannot detect unexpected and unknown fault types. The proposed immuno-inspired fault detection algorithm can detect a broad spectrum of known and unforeseen failures.

In summary, the proposed method works as follows:

Based on the dataset (given) of normal operating conditions, generate a set of fault detectors; the goal is, however, to evolve 'good' detectors that cover the non-self space.

- A 'good' detector:
  - Must not cover self space.
  - Has to be as general as possible: the larger the volume, the better.
- One detector may not be enough; instead, a set of detectors is required that can collectively cover the non-self space with minimum overlap.

During the testing phase, we used the data collected during different fault conditions.

- Detectors that get activated (match) for each fault are labeled as specific fault detectors. These constitute a set of specialized detectors for identifying different class of faults.
- It may be necessary to go through the detector optimization process: filter out some overlapping detectors, merge some and generate new ones for better coverage.

Some faulty conditions can be simulated (by changing the crucial monitored parameters) in order to check which generalized detectors get activated. These detectors can also provide the knowledge of possible (unknown) faults. The goal is to achieve a certain level of damage control under any known fault or unknown abnormalities.

The long-term goal is to use RNS to detect control surface area loss caused by damage (or failure) and other causes that may result in the departure of the aircraft from safe flight conditions. Once the failure is detected and identified, the Intelligent Flight Controller (IFC) then utilizes all remaining source of control power necessary to achieve the desired flight performance.

References

[1] Y.M. Chen; M.L. Lee. **Neural networks-based scheme for system failure detection and diagnosis**. In Mathematics and Computers in Simulation (Elsevier Science), **Year:** 2002 **Volume:** 58 **Number:** 2 **Pages:** 101-109

[2] Rube B. Williams Jr; Alexander G. Parlos. **Adaptive State Filtering for Space Shuttle Main Engine Turbine Health Monitoring**. In Journal of Spacecraft and Rockets (American Institute of Aeronautics and Astronautics), **Year:** 2003 **Volume:** 40 **Number:** 1 **Pages:** 101-109.

[3] Jovan D. Boskovic; Raman K. Mehra. **Intelligent Adaptive Control of a Tailless Advanced Fighter Aircraft Under Wing Damage**. In Journal of Guidance, Control, and Dynamics (American Institute of Aeronautics and Astronautics), **Year:** 2000 **Volume:** 23 **Number:** 5 **Pages:** 876-884.

[4] Bodson, M., and Groszkiewicz, J., **Multivariable Adaptive Algorithms for Reconfigurable Flight Control**. IEEE Transactions on Control Systems Technology, Vol. 5, No. 2, 1997, pp. 217–229.

[5] Jovan D. Boskovic; Raman K. Mehra. **Multiple-Model Adaptive Flight Control Scheme for Accommodation of Actuator Failures**. In Journal of Guidance, Control, and Dynamics (American Institute of Aeronautics and Astronautics), 2002 **Volume:** 25 **Number:** 4 **Pages:** 712-724.

[6] P. Melin; O. Castillo. **Intelligent control of aircraft dynamic systems with a new hybrid neuro-fuzzy-fractal approach**. In Information Sciences (Elsevier Science), 2002 **Volume:** 142 **Number:** 1 **Pages:** 161-175.

[7] Marc L. Steinberg. **Comparison of Intelligent, Adaptive, and Nonlinear Flight Control Laws**. In Journal of Guidance, Control, and Dynamics (American Institute of Aeronautics and Astronautics) **Year:** 2001 **Volume:** 24 **Number:** 4 **Pages:** 693-699.

[8] Giampiero Campa; Marcello Napolitano; Brad Seanor; Mario G. Perhinschi. **Online Parameter Estimation Techniques Comparison Within a Fault Tolerant Flight Control System**. In Journal of Guidance, Control, and Dynamics (American Institute of Aeronautics and Astronautics) **Year:** 2002 **Volume:** 25 **Number:** 3 **Pages:** 528-537.

[9] Joseph S. Brinker and Kevin A. Wise. **Flight Testing of Reconfigurable Control Law on the X-36 Tailless Aircraft**. In Journal of Guidance, Control, and Dynamics (American Institute of Aeronautics and Astronautics) Year: 2001 Volume: 24 Number: 5 Pages: 903-909.

[10] William P. Grogan. **Aggregate Surface Requirements for C-17 Aircraft Operations.**

[11] William B. Blake. **Development of the C-17 Formation Airdrop Element Geometry. In Journal of Aircraft**, Year: 1998 Volume: 35 Number: 2 Pages: 175-182.

[12] Karen Gundy-Burlet, K. Krishnakumar, Greg Limes and Don Bryant. **Control Reallocation Strategies for Damage Adaptation in Transport Class Aircraft**. In AIAA 2003-5642, August, 2003.

[13] Kaneshige, J. and Gundy-Burlet, K. **Integrated Neural Flight and Propulsion Control System**, AIAA 2001-4386, August 2001.

[14] KrishnaKumar, K., Limes, G., Gundy-Burlet, K., Bryant, D., **An Adaptive Critic Approach to Reference Model Adaptation**. In AIAA GN&C Conf. 2003

[15] Rysdyk, Rolf T., and Anthony J. Calise, **Fault Tolerant Flight Control via Adaptive Neural Network Augmentation,** AIAA 98-4483, August 1998.

[16] Meir Pachter and Yih-Shiun Huang. **Fault Tolerant Flight Control**. In Journal of Guidance, Control, and Dynamics (American Institute of Aeronautics and Astronautics) Year: 2003 Volume: 26 Number: 1 Pages: 151-160.

[17] Ki-Seok Kim; Keum-Jin Lee; Youdan Kim. **Reconfigurable Flight Control System Design Using Direct Adaptive Method**. In Journal of Guidance, Control, and Dynamics (American Institute of Aeronautics and Astronautics) **Year:** 2003 **Volume:** 26 **Number:** 4 **Pages:** 543-550.

[18] Gonzales, F., Dasgupta, D.: Anomaly Detection Using Real-Valued Negative Selection. In Genetic Programming and Evolvable Machines, 4, (2003) 383-403

[19] Dasgupta D, Forrest S (1999). An anomaly detection algorithm inspired by the immune system. In: Dasgupta D (eds) Artificial Immune Systems and Their Applications, Springer-Verlag, pp.262–277

[20] D'haeseleer P, Forrest S, Helman P (1996) An immunological approach to change detection: algorithms, analysis, and implications, Proc of the 1996 IEEE Symposium on Computer Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, pp. 110–119.

[21] Forrest S, Perelson AS, Allen L, Cherukuri R (1994) Self-nonself discrimination in a computer, Proc. of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, pp. 202–212.

[22] D. Bradley and A. Tyrrell. Hardware Fault Tolerance: An Immunological Solution. In the proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMC), Nashville, October 8-11, 2000.

[23] M. Araujo, J. Aguilar, H. Aponte. Fault detection system in gas lift well based on Artificial Immune System. In the proceedings of the International Joint Conference. pp. 1673 -1677, No. 3, July 20 - 24, 2003.

[24] Canham and Tyrrell. A Multilayered Immune System for Hardware Fault Tolerance within an Embyronic Array. In the proceedings of 1st International Conference on Artificial Immune Systems (ICARIS-2002), University of Kent at Canterbury, UK, September 9th-11th, 2002.

[25] Liu Shulin, Zhang Jiazhong, Shi Wengang, Huang Wenhu. Negative-selection algorithm based approach for fault diagnosis of rotary machinery. In the Proceedings of American Control Conference, 2002, Vol. 5, pp. 3955 -3960. 8-10 May 8-10, 2002.

[26] Dan W Taylor and David W Corree. An Investigation of the Negative Selection Algorithm for Fault Detection in Refrigeration Systems. In the Proceeding of Second International Conference on Artificial Immune Systems (ICARIS), September 1-3, 2003, Napier University, Edinburgh, UK.

[27] S. Xanthakis, S. Karapoulios, R. Pajot and A. Rozz. Immune System and Fault Tolerant Computing. In J.M. Alliot, editor, Artificial Evolution, volume 1063 of Lecture Notes in Computer Science, pages 181-197. Springer-Verlag, 1996.

[28] F. Niño, D. Gómez, and R. Vejar. A Novel Immune Anomaly Detection Technique Based on Negative Selection. In the proceedings of the Genetic and Evolutionary Computation Conference (GECCO) [Poster], Chicago, IL, USA, July 12-16, 2003. LNCS 2723, p. 243.

[29] S. Singh. Anomaly detection using negative selection based on the r-contiguous matching rule. In 1st International Conference on Artificial Immune Systems (ICARIS), University of Kent at Canterbury, UK, September 9th-11th, 2002.

[30] K. KrishnaKumar. Artificial Immune System Approaches for Aerospace Applications. American Institute of Aeronautics and Astronautics 41st Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 6-9 January 2003.