Negative Selection Algorithm for Aircraft Fault Detection

D. Dasgupta[1], K. KrishnaKumar[2], D. Wong[1], M. Berry[3]

[1] Division of Computer Science, University of Memphis
Memphis, TN
[2] Computational Sciences Division, NASA Ames Research Center
Moffett Field, CA
[3] QSS Group Inc, NASA Ames Research Center, Moffett Field, CA

Abstract. We investigated a real-valued Negative Selection Algorithm (NSA) for fault detection in man-in-the-loop aircraft operation. The detection algorithm uses body-axes angular rate sensory data exhibiting the normal flight behavior patterns, to generate probabilistically a set of fault detectors that can detect any abnormalities (including faults and damages) in the behavior pattern of the aircraft flight. We performed experiments with datasets (collected under normal and various simulated failure conditions) using the NASA Ames man-in-the-loop high-fidelity C-17 flight simulator. The paper provides results of experiments with different datasets representing various failure conditions.

1 Introduction

Early detection of a fault or damage of aircraft subsystems is very crucial for its control and maneuver during the flight [5, 15]. 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 are necessary to achieve acceptable flight performance and higher flight survivability under abnormal conditions. There are several techniques available for aircraft fault accommodation problems [2-3, 11, 13] and for fault detection and isolation issues [6, 16]. This work investigated an immunity-based approach that can detect a broad spectrum of known and unforeseen faults. The goal is to apply the immunity-based fault detection algorithm to improve the fault tolerance capabilities of the existing Intelligent Flight Controller (IFC) architecture [11, 13].

Prior studies have established the benefits of intelligent flight control [2]. However, one area of weakness that needed 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 reduced over time due to learning, was present and caused degradation in handling qualities. This also makes it challenging for outer loop control design. If the failure can be identified, this dead band can fur-

ther be minimized to ensure rapid fault accommodation and better handling qualities [12, 13, 15].

2 Real-Valued Negative Selection (RNS) Algorithm

The negative selection algorithm [9] is based on the principles of self-nonself discrimination in the immune system (Fig. 1 shows the concept of self and nonself space). This negative selection algorithm can be summarized as follows (adopted from [7]):

- Define self as a collection S of elements in a feature space U, a collection that needs to be monitored. For instance, if U corresponds to the space of states of a system represented by a list of features, S can represent the subset of states that are considered as normal for the system.
- Generate a set F of *detectors*, each of which fails to match any string in S. An approach that mimics the immune system generates random detectors and discards those that match any element in the self set. However, a more efficient approach [8] tries to minimize the number of generated detectors while maximizing the covering of the nonself space.
- Monitor S for changes by continually matching the detectors in F against S. If any detector ever matches, then a change is known to have occurred, as the detectors are designed not to match any representative samples of S.

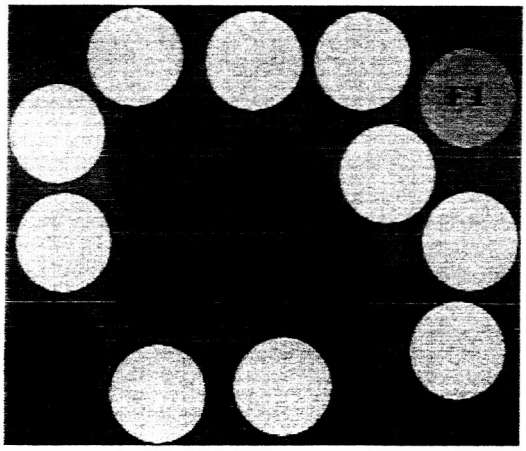


Figure 1. The figure illustrates the concept of self and nonself in a feature space. Here F1, F2, etc. indicate different fault conditions represented by detectors.

The above description is very general and does not say anything about the representation of the problem space and the type of *matching* rule is used. It is, however, clear that the algorithmic complexity of generating good detectors can vary significantly, which depends on the type of problem space (continuous, discrete, mixed, etc.), detector representation scheme, and the rule that determines if a detector *matches* an element or not. Most of the research works on the NS algorithm have been restricted to the binary matching rules like *r*-contiguous [9]. The primary reason for this choice is ease of use, and there exist efficient algorithms to generate detectors, exploiting the simplicity of the binary representation and its *matching* rules [8]. However, the scalability issue has prevented it from being applied more extensively.

We adopted a real-valued NS (RNS) algorithm, which tries to alleviate the limitations previously mentioned, while using the structure of the higher-level-representation to speed up the detector generation process. The real-valued NS algorithm applies a heuristic process that changes iteratively the position of the detectors driven by two goals: to maximize the coverage of the nonself subspace and to minimize the coverage of the self samples.

2.1 Details of the RNS Algorithm – Detector Generation phase:

The RNS detector generation starts with a population of candidate detectors, which are then matured through an iterative process. In particular, the center of each detector is chosen at random and the radius is a variable parameter which determines the size (in *m*-dimensional space) of the detector. The basic algorithmic steps of the RNS detector generation algorithm are given in Fig. 2, and some computational details of are illustrated in Figs. 3(a)-(d).

At each iteration, first, the radius of each candidate detector is calculated, and the ones that fall inside self region are moved (i.e. its center is successively adjusted by moving it away from training data and existing detectors). The set of nonself detectors are then stored and ranked according to their size (radius). The detectors with larger radii (and smaller overlap with other detectors) are considered as better-fit and selected to go to the next generation. Detectors with very small radii, however, are replaced by the clones of better-fit detectors. The clones of a selected detector are moved at a fixed distance in order to produce new detectors in its close proximity. Moreover, new areas of the nonself space are explored by introducing some random detectors. The whole detector generation process terminates when a set of mature (minimum overlapping) detectors are evolved which can provide significant coverage of the nonself space.

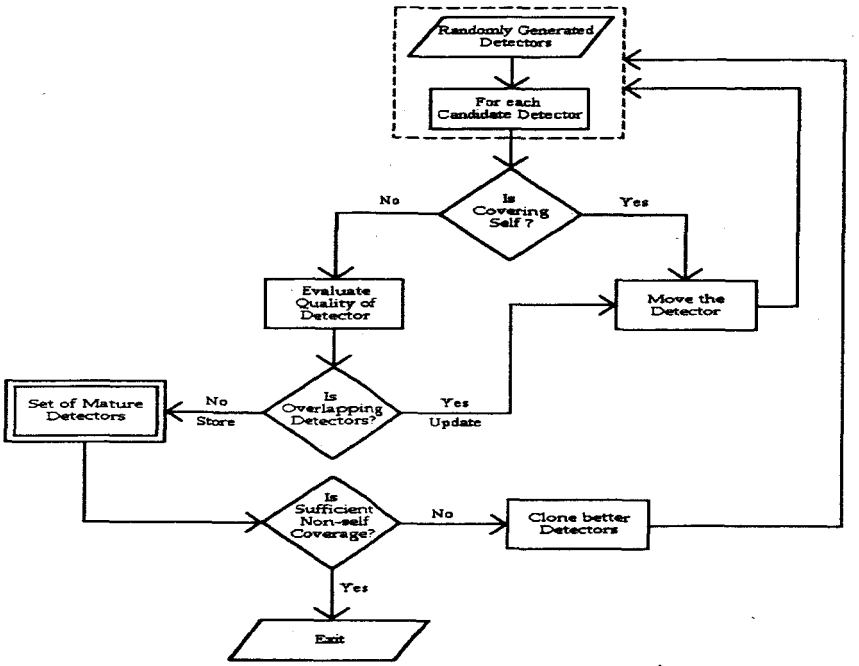


Figure 2: Flow diagram showing the algorithmic steps for the real-valued negative selection algorithm.

The purpose of the fault detection is to identify which states of a system are normal (self) and which are faulty. The states of a system are represented by a set of control variables which can exhibit the current and past system behavior. The actual values of these variables are scaled or normalized in the range [0.0, 1.0] in order to define the self-nonself space with a unit hypercube.

A detector is defined as $d = (c, r_d)$, where $c = (c_1, c_2, ..., c_m)$ is an m-dimensional point that corresponds to the center of a unit hypersphere with r_d as its radius. The following parameters are used for the detector generation process:

r_s: threshold value (allowable variation) of a self point; in other words, a point at a distance greater than or equal to r_s from a self sample is considered to be abnormal.

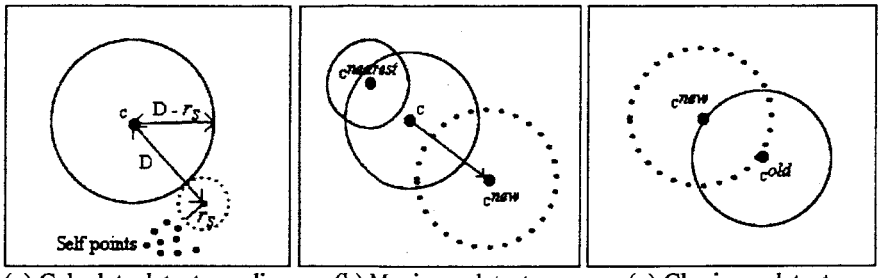
α: variable parameter to specify the movement of a detector away from a self sample or existing detectors.

ξ: maximum allowable overlap among the detectors, which implicates that allowing some overlap among detectors can reduce holes in the nonself coverage.

2.1.1 Calculating the detector radius: We used the *Minkowski distance* to measure the distance *(D)* between two points x and y, which is defined as

$$D(x, y) = (\sum | x_i - y_i |^{\lambda})^{1/\lambda}$$

where $x = \{ x_0, x_1, \ldots x_{N-1} \}$ and $y = \{ y_0, y_1, \ldots y_{N-1} \}$. The *Minkowski* distance with $\lambda = 2$ is equivalent to *Euclidean Distance*.



(a) Calculate detector radius (b) Moving a detector (c) Cloning a detector

Figure 3: Illustrate different computational steps used during the detector maturation process. (a) Shows a way to calculate and update the radius of a detector (b) If a candidate detector overlaps with an existing detector (or self points), then the candidate detector (i.e. its center, c) is moved in the opposite direction to its nearest neighbor detector; (c) Given a mature detector, a clone is created at a distance equal to its radius, and the direction where it is created is selected at random.

This approach allows having variable size detectors to cover the nonself space. As shown in Fig. 3(a), if the distance between a candidate detector, $d = (c, r_d)$ and its nearest self point in the training dataset is D, then the detector radius is considered as $r_d = (D - r_s)$. However, if a detector is close to any edge of the hypercube and has a large radius, then only a portion of it is inside the space, and such a detector is referred to as an *edge* detector. If the value $r_d = (D - r_s)$ is *negative* then it falls inside the self (radius); and this detector is expected to be discarded or moved in subsequent iteration.

2.1.2 Moving detectors: Let $d = (c, r_d)$ represents a candidate detector and $d^{nearest} = (c^{nearest}, r_d{}^{nearest})$ is its nearest detector (or a self point), then the center of d is moved such that

$$c^{new} = c + \alpha \frac{dir}{\|dir\|},$$

where $dir = c - c^{nearest}$, and $\|\cdot\|$ denotes the norm of a m-dimensional vector (Fig. 3(b)). Accordingly, if a detector overlaps significantly with any other existing detec-

tors, then it is also moved away from its nearest neighbor detector. In order to guarantee the convergence of this process, an exponential decay function is used for the moving parameter (α).

2.1.3 Detector Cloning and Random Exploration: At every generation, a few better-fitted detectors are chosen to be cloned. Specifically, let $d = (c^{old}, r_d{}^{old})$ be a detector to be cloned and, say $d^{clon} = (c^{clon}, r_d{}^{clon})$, is a cloned detector whose center is located at a distance $r_d{}^{old}$ from d and whose radius is the same as that of the detector, d. Accordingly, the center of d^{clon} is computed as

$$c^{clon} = c^{old} + r_d^{old} \frac{dir}{\|dir\|}$$

$dir = c^{old} - c^{nearest}$ (see Fig. 3(c)), and where $c^{nearest}$ is the center of d's nearest detector.

In addition, a few less-fit detectors are replaced by random detectors at each generation. This process allows the exploration of new regions of the nonself space, which may need to be covered.

2.1.4 Evaluation of nonself detectors: Detectors which do not fall in the self region are sorted according to their size. A detector with large radius gets selected for the next generation population, if it has small overlap with existing detectors i.e. less than an overlapping threshold (ξ).

Accordingly, the overlapping measure W of a detector is computed as the sum of its overlap with all other detectors as follows,

$$W(d) = \sum_{d \neq d'} w(d, d'),$$

where $w(d, d')$ is the measured overlap between two detectors $d = (c, r_d)$ and $d' = (c, r_{d'})$; and is defined by $w(d,d') = (\exp(\delta) - 1)^m$, m is the dimension of the feature space,

$$\text{and } \delta = \left(\frac{r_d + r'_d - D}{2r_d} \right)$$

The value of δ is considered to be bounded between 0 and 1; and D is the distance between two detector centers c and c' as shown in Fig. 3(d). This overlapping measure seems to favor the detectors with bigger radii, i.e. detectors having larger coverage of the nonself space with minimum overlap among them. Accordingly, the closer the center of two detectors is, the higher the value of the overlapping measure $w(d, d')$.

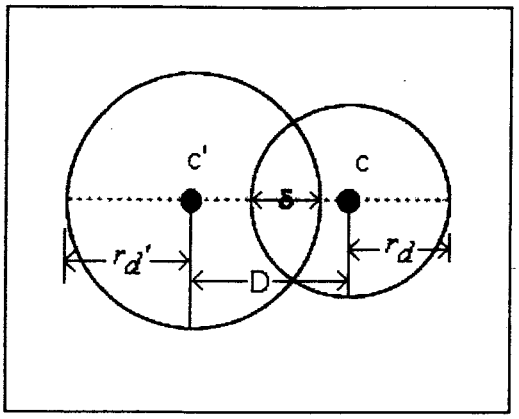


Figure 3(d): The overlap between two detectors d and d' is computed in terms of the distance (D) between their centres (c, c') and radii $(r_d, r_{d'})$.

2.2 Testing phase: Detection process

The detection process is straightforward -- the matured detectors are continually examined with new samples in test datasets. For example, the distance between a sample pattern, $p = (c_p, r_s)$ and a detector, $d = (c, r_d)$ is computed as $D(c_p, c)$, where $D(c_p, c)$ is the distance between the sample pattern and the detector calculated in the same way as in the detector generation phase. If the distance, $D < (r_s + r_d)$ then the detector d gets activated indicating possible fault.

3. Fault Detection in Aircraft operation

There are many applications where the aircraft behavior monitoring for indicating flight subsystem fault/damage, detection of changes (in trends) in the operating condition, etc. appears to be very useful [12]. We used a C-17 man-in-the-loop simulation data for this study.

An aircraft typically has many dedicated primary and secondary control surfaces to maneuver the aircraft through space. In modern fly-by-wire aircraft, the pilot commands angular rates of the aircraft using a side stick device. Speed and altitude control is mostly achieved using auto pilot settings chosen by the pilot. The C-17 aircraft, in addition to four engines, has a stabilizer, four elevators, two ailerons, eight spoiler panels, four flap sections, and two rudders for control. In the figure shown below, the pilot stick commands are shaped by a reference model to generate body-axes roll,

pitch, and yaw acceleration commands to the NASA Ames developed Intelligent Flight Controller [11, 13]. The sensory feedback to the controller includes the body-axes angular rates, airspeed, altitude, and angle-of-attack. An error vector, computed using the difference between commanded and sensed body-axes angular rates, is used to drive the intelligent flight controller. For this study, data generated through piloted simulation studies using the C-17 simulator are used.

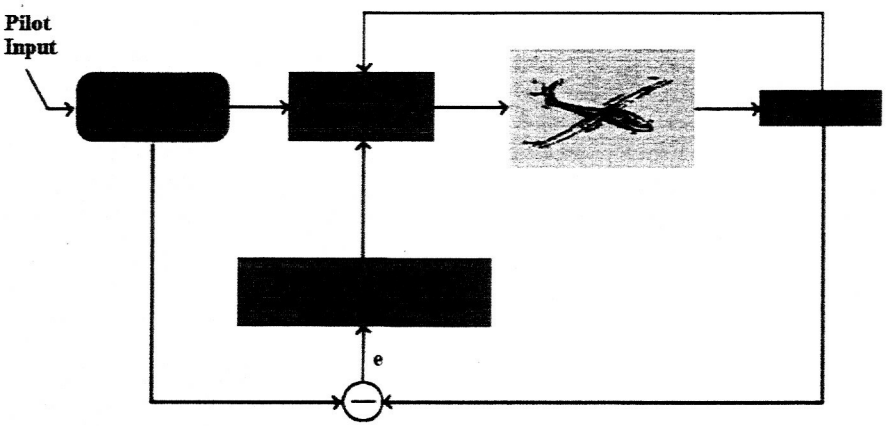


Figure 4: The block diagram showing the integration of immunity-based Fault detection system and intelligent flight control (IFC) system.

We experimented with different detector generator schemes for real-valued negative selection algorithm for better detection of different faults. This fault detection system takes real-valued data set as input; extracts the important semantic information by applying 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 type of faults (known and unknown faults). The aim is to find a small number of specialized detectors (as signature of known failure conditions) and a bigger set of generalized detectors for unknown (or possible) fault conditions. The output of the fault detection system will be provided to IFC for isolating the faulty control surface.

4. Experiments and Results

The data from the simulator is collected at the rate of 30 Hz per second. Since the data is from man-in-the–loop simulation, there is an extra control (the human pilot) that simply cannot be removed from the data. We considered three sets of in-flight sensory information—namely, body-axes roll rate, pitch rate and yaw rate—to detect five

different simulated faults. The error rates are measured from the desired output of the reference model and the actual sensed output (as shown in Fig. 4).

The training data set was created by taking all of the data until the moment of the failure. For both cases, this is a different amount of time which varies between 2-3 seconds. 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 maneuver by then, and the aircraft would still be flying straight and level.

4.1 Preprocessing Data:

First, data are normalized where the actual values of the variables are scaled or normalized to fit a defined range [0.0, 1.0] using the maximum and minimum (+/- 20% to normal data) value of each dimension in the data set. Any value above and below the defined *max* and *min* is considered as *1* and *0*, respectively. Data window shift—shifts the data based on the window shifting and data overlapping parameters. In these experiments, K-mean clustering is used to reduce the training dataset to improve the time complexity of the detector generation process.

4.2 Detector generation phase:

The RNS algorithm takes as input a set of hyper-spherical detectors randomly distributed in the self/nonself space. The algorithm applies a heuristic process that changes iteratively the position of the detectors driven by the objective to maximize the coverage of the nonself subspace and to minimize their overlap, while not covering the self samples. The flow diagram for the proposed detector generation algorithm (RNS) is given in previous section (Fig. 2).

The aim is to find a small number of specialized detectors (as signature of known fault conditions) and other generalized detectors for unknown (or possible) fault conditions.

4.3 Testing phase:

After generating the detectors, the next step is to examine the effectiveness of detectors in identifying various faults. We used the same preprocessing steps and distance measure during the testing phase as was used in generating detectors. If a detector gets activated with current pattern, a change in the behavior pattern is known to have occurred and an alarm signal is generated regarding the fault. In particular,

- For each fault condition: Combine all (or as many as possible) activated detectors to form a small set of *specialized* fault detectors that can be *labelled* with specific fault type.

- For rest of the detectors: Use clustering method to form different clusters with cluster centre and radius of the cluster to be labelled as *probable (or possible)* fault detectors

Some of the generalized detectors can be used for detecting simulated faults during the testing phase, however, others may be considered as being useless and discarded. It is expected that a limited number of detector generated by NSA (initially) may not be sufficient to cover the entire space, so it may be necessary to change the number, their distribution and resolution in nonself space to provide system and condition specific fault detection capabilities.

4.4 Results:

The sensor parameters considered for these experiments include body-axes roll angle 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. The following graphs illustrate results of experiments with different failure conditions.

Fig. 5 shows the error rate(e) for three sensory inputs that are considered as the normal operating conditions, and used to generate the detector set. Fig. 6 illustrates the performance of the detection system when tested with wing fault data, where this type of fault is manifested in roll error rate (starting at 300 time step). The graph also shows the number of detectors activated (upper bar chart) as significant deviations in data patterns appear. Fig. 7 displays similar results for the tail failure. It is to be noted that simulation data that have a full tail failure, show up first in the pitch axis as the pitch axis has very little coupling with the roll and yaw axes.

As this detector generation algorithm generates variably sized detectors to cover spaces within and outside self regions, Fig. 8 gives the number of detectors with different sizes in a typical run. In this case, more number of detectors is generated with radii ranging between 0.25-0.3. However, some of the detectors with larger radii are edge detectors, partially covering the nonself space.

Table 1 gives the statistical results of 10 runs for two different faults—tail and wing damage. Here the number of detector generated is 368; when tested on two faulty data sets, the detection rates of 89% and 92% are observed with relatively small false alarm rates.

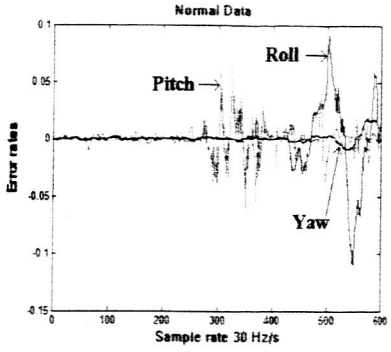


Figure 5. Show error rates for three sensory inputs (roll, pitch and yaw) that are considered as the normal data pattern.

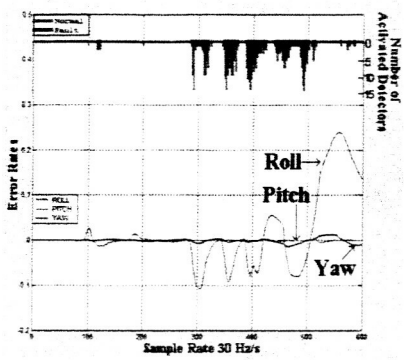


Figure 7: Test results indicating the detection of Tail 1 failure.

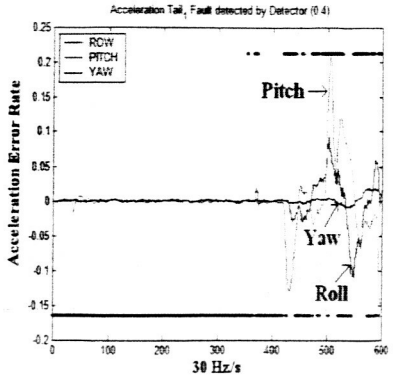


Figure 6. Show the detection of a wing failure with number of activated detectors.

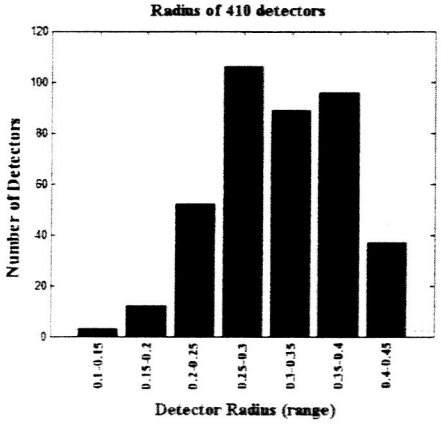


Figure 8: Indicates different sized detectors that are generated in a typical run.

Table 1. Table shows some statistics of testing two different faults (tail and wing failure).

Type of Faults	Performance			
	Detection date		False alarm rate	
	Mean	*S.D.*	*Mean*	*S.D*
Tail	89%	1.43	0.87%	0.45
Wing	92%	1.67	0.98%	0.32

Fig. 9 shows the number of detectors activated for five different faults. We observed that a good number of detectors get activated in each fault case. There appears to be three possible reasons: first, because of allowing some overlap among detectors; second, faulty data may be clustered at several locations in the nonself space, and third, the same detectors may be activated for more than one fault cases. Fig. 10 shows different false alarm rates with the change in number of detectors. It indicates that the false positive rate reduces as number of detectors increases; moreover, the increase in false negative is insignificantly small (up to 1.4%) at the same time.

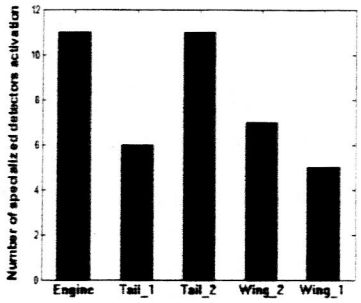


Figure 9: Average number of detectors activated for each fault tested.

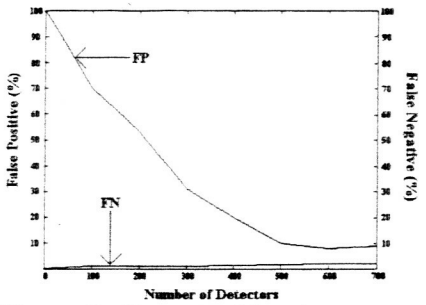


Figure 10. Shows the false alarm rates (both false positive and false negative) with change in number of detectors.

5. Conclusions:

There exist many techniques for aircraft failure detection [2-3,5-6,13,15]. One of the drawbacks of existing fault detection and isolation (FDI) based approaches is that they cannot detect unexpected and unknown fault types. The immunity-based approaches have been used in many applications including fault detection [1, 4, 17, 19, 20], anomaly detection [14, 18], etc. We investigated a real-valued negative selection algorithm that could detect a broad spectrum of known and unforeseen faults. In this work, 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) 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.

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 nonself space.

- A 'good' detector:
  - It must not cover self space.
  - It 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 nonself 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, and merge some and generating new ones for better coverage.

Some faulty conditions can be simulated (by changing some 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 NSA 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. M. Araujo, J. Aguilar and H. Aponte. *Fault detection system in gas lift well based on Artificial Immune System.* In the proceedings of the International Joint Conference on AI, pp. 1673 -1677, No. 3, July 20 - 24, 2003.
2. Jovan D. Boskovic and 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), Volume: 23 Number: 5 Pages: 876-884, 2000
3. Jovan D. Boskovic and 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), Volume: 25 Number: 4 Pages: 712-724, 2002.
4. 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.

5. 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), Volume: 24, Number: 5 Pages: 903-909, 2001.
6. Y.M. Chen and M.L. Lee. *Neural networks-based scheme for system failure detection and diagnosis.* In Mathematics and Computers in Simulation (Elsevier Science), Volume: 58 Number: 2 Pages: 101-109, 2002.
7. D. Dasgupta, S. Forrest. *An anomaly detection algorithm inspired by the immune system.* In: Dasgupta D (eds) Artificial Immune Systems and Their Applications, Springer-Verlag, pp.262–277, 1999.
8. P. D'haeseleer, S. Forrest, and P. Helman. *An immunological approach to change detection: algorithms, analysis, and implications.* In Proceedings of the IEEE Symposium on Computer Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, pp. 110–119, 1996.
9. S Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. *Self-nonself discrimination in a computer.* In Proc. of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, pp. 202–212, 1994.
10. F. Gonzales and D. Dasgupta. *Anomaly Detection Using Real-Valued Negative Selection.* In Genetic Programming and Evolvable Machines, 4, pp.383-403, 2003.
11. 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.
12. 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.
13. K. KrishnaKumar, G. Limes, K. Gundy-Burlet, D. Bryant. *An Adaptive Critic Approach to Reference Model Adaptation.* In AIAA GN&C Conf. 2003.
14. 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, LNCS 2723, p. 243, July 12-16, 2003.
15. Meir Pachter and Yih-Shiun Huang. *Fault Tolerant Flight Control.* In Journal of Guidance, Control, and Dynamics (American Institute of Aeronautics and Astronautics), Volume: 26 Number: 1 Pages: 151-160, 2003.
16. Rolf T. Rysdyk and Anthony J. Calise, Fault *Tolerant Flight Control via Adaptive Neural Network Augmentation*, AIAA 98-4483, August 1998.
17. 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.
18. 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 9-11, 2002.
19. Dan W Taylor and David W Corne. *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), Napier University, Edinburgh, UK, September 1-3, 2003.
20. 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.