

Sensitivity of fusion performance to classifier model variations

Kai Goebel*
GE Global Research

ABSTRACT

During design of classifier fusion tools, it is important to evaluate the performance of the fuser. In many cases, the output of the classifiers needs to be simulated to provide the range of fusion input that allows an evaluation throughout the design space. One fundamental question is how the output should be distributed, in particular for multi-class continuous output classifiers. Using the wrong distribution may lead to fusion tools that are either overly optimistic or otherwise distort the outcome. Either case may lead to a fuser that performs sub-optimal in practice. It is therefore imperative to establish the bounds of different classifier output distributions. In addition, one must take into account the design space that may be of considerable complexity. Exhaustively simulating the entire design space may be a lengthy undertaking. Therefore, the simulation has to be guided to populate the relevant areas of the design space. Finally, it is crucial to quantify the performance throughout the design of the fuser. This paper addresses these issues by introducing a simulator that allows the evaluation of different classifier distributions in combination with a design of experiment setup, and a built-in performance evaluation. We show results from an application of diagnostic decision fusion on aircraft engines.

Keywords: Classification; Diagnostics; Information Fusion; Decision Fusion; Simulation;

1. INTRODUCTION

Classifier design is in practice most of the times guided by particular performance requirements. That is, there are particular specifications about the false positive and true positive rates (or false negative or true negative rates). The actual specification is driven by the application and the domain where the classifier is executed. For example, it might be extremely undesirable to issue false positives in certain domains while (even for the same application) it is more important to avoid false negatives in a different domain (such as in some military vs. civilian applications). At any rate, the classifier designer would follow a path of data pre-processing, feature extraction and selection, classifier design and optimization, and classifier evaluation¹. If the classifier performance is not satisfactory, the steps of classifier selection and optimization as well as feature selection and extraction may be revisited to improve on the results (see Figure 1).

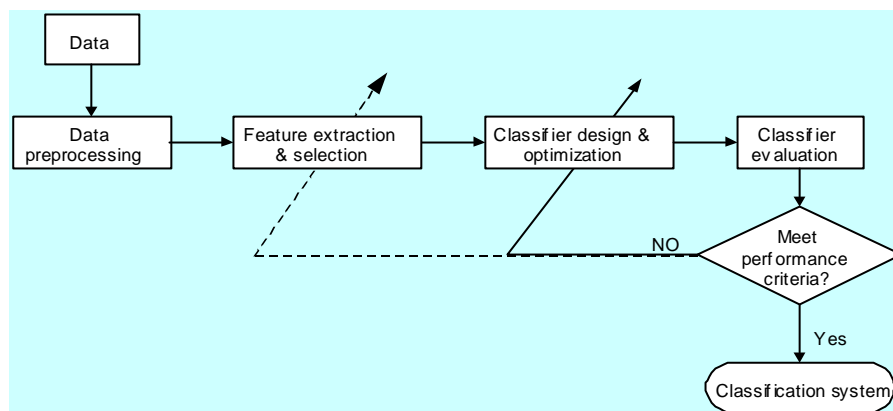


Figure 1: Typical classification system design process¹

* goebelk@research.ge.com; phone 1 518 387-4194; fax 1 418 387-6104; <http://best.me.berkeley.edu/~goebel/kai.html>; GE Global Research, K1-5C4A, 1 Research Circle, Niskayuna, NY 12309, USA

Although there are a number of classifiers that might be considered for a given problem, each classifier will exhibit a performance ceiling. This ceiling might prevent the overall goals of the problem to be achieved. A convenient way to display the classifier performance over the range of performance criteria is the receiver operating characteristic (ROC) curve. ROC analysis is an established method of measuring classification performance in various domains such as medical imaging. Originated from the field of signal detection to depict tradeoffs between hit rate and false alarm rate², ROC analysis and its associated indices have been extended for use in evaluating performance of 2-class classifiers^{3,4}. The ROC space is a coordinate system that is used for visualizing classifier performance. In ROC space, the true positive rate, TPR, is plotted on the Y-axis and the false positive rate, FPR, is plotted on the X-axis, where the TPR is sometimes referred to as “sensitivity” while (1-FPR) is called “specificity”. A point in ROC space corresponds to a (FPR, TPR) pair of a classifier. By varying the decision threshold of the classifier, a series of points (FPR, TPR) can be obtained and a ROC curve can be generated in ROC space by connecting these points. Typical ROC curves are shown in Figure 2 where the three ROC curves represent three different classifiers. Classifiers with ROC curves located in the upper-left corner in ROC space are better because they represent classifiers that have lower false positive rate and higher true positive rate than the classifiers below them.

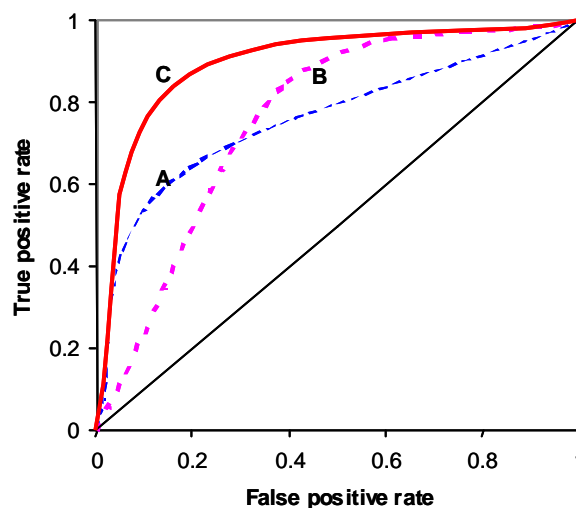


Figure 2: ROC curves for classifiers A, B, and C.

ROC curves are a valuable technique for visualizing classifier behavior over a range of decision rules, therefore ROC curves are frequently used for selecting a suitable operating point, or decision threshold, for the task at hand. However, when used for comparing or ranking classifiers based on their performance, evaluation based on ROC curves becomes more involved when the curves overlap. In Figure 2 the three ROC curves correspond to three different classifiers A, B, and C. One can easily conclude that classifier C is better than or at least as good as the other two classifiers for all possible cost and class distributions since curve C dominates others in all range. Determining which of the two classifiers (A and B) is better, on the other hand, would not be so straightforward unless a specific performance requirement is given. For example, given the maximum number of false positive rate, one would simply draw a vertical line at the specified maximum FPR, and rank the classifiers based on TPR at the intersection of ROC curves with this vertical line. Similarly, one would use a horizontal line to rank the classifiers if the maximum true positive rate is given. ROC curves are a valuable tool in visualizing the classifier performance and can aid in finding the classifier performance ceiling

Overcoming the performance ceiling is a strong motivator to use classifier fusion systems. Such a fusion scheme gathers and combines the results of different classification tools to maximize the advantages of each one while minimizing the disadvantages. The fusion system holds the promise to deliver a result that is better than the best result possible by any one tool employed. In part this can be accomplished because redundant information is available, which when combined correctly improves the estimate of the better tool and compensates for the shortcomings of the less capable tool. In addition, there is a gamut of secondary information that can potentially be folded into the fusion scheme to boost the overall classification performance. However, there is no substitute for a good classification tool and, ordinarily, multiple, marginal-performance tools do not necessarily combine to produce an improved result and in

fact may worsen the outcome⁵. Depending on the choice of fusion tool, it may also be important to ensure that the methods implemented contribute to an improvement of the result. This is in particular true for systems that employ elaborate preprocessing procedures that are not part of the core fuser⁶.

During design of classifier fusion tools, it is vital to evaluate the performance of the fuser. Where the output of the classifiers is unknown (perhaps because the classifier itself is still under development by another party), its output needs to be simulated to in turn act as input to the fusion tool. One fundamental question is how the output should be distributed, in particular for multi-class continuous output classifiers. Using the wrong distribution may lead to fusion tools that are either overly optimistic or otherwise distort the outcome. Either case may lead to a fuser that performs sub-optimal in practice. It is therefore imperative to establish the bounds of different classifier output distributions. In addition, one must take into account the design space that may be of considerable complexity. Exhaustively simulating the entire design space may be a lengthy undertaking. Therefore, the simulation has to be guided to populate the relevant areas of the design space. Finally, it is important to quantify the performance throughout the design of the fuser. The following sections address these issues by introducing a simulator that allows the evaluation of different classifier distributions in combination with a design of experiment setup, and a built-in performance evaluation.

2. CLASSIFIER OUTPUT SIMULATION

2.1 Simulation Set-up

To facilitate the performance evaluation during the fusion design process, each new routine needs to be subjected to an iterative cycle of conception, implementation, and testing. This is done to reduce the re-design necessary after completion of the whole fusion architecture which at that stage would be more difficult because the influence of individual routines are then masked in the overall architecture. At the onset, it is not clear what effect the addition of new modules of the algorithm will have on the overall performance and whether the chosen implementation of that heuristic will move the error into the desired direction. Similarly, the parameters are not known beforehand and at least coarse tuning needs to be carried out for each routine. To appraise the performance of the algorithm during development, a metric to assess incremental enhancements to the code is needed. Where possible, it is advisable to establish an error metric using expert judgments. In cases where the design space is of rather large size (a common issue) it will be difficult to elicit expert opinions for the entire space. To cut down on the number of case permutations to look at and to at the same time cover the entire design space as exhaustively as possible, we propose to employ the Design-of-Experiment (DoE) approach⁷ for a conceptual set of cases. Because the domain expert will be exposed only to the tool output without knowledge of the root causes for the cases, this step is called the “bottom-up DoE”. The purpose is to capture expert’s reasoning and to get a training set that can be used to design the reasoning tool. Using the experts’ heuristics, the DoE set, and the associated ratings, one can then implement successively the heuristics (or routines) for the fusion tool with constant evaluation of performance. Implementations of heuristics not leading to an error reduction can be redesigned and can also be rapidly re-evaluated. This cuts down significantly on development time and ensures a successful architecture. In a second step, one may run a full Monte Carlo simulations^{8,9} with known inputs (“top-down Monte Carlo”). The Monte Carlo runs can be carried out in batch mode for all cases, i.e., a desired (typically large) number of runs can be employed. Results are returned in cumulative fashion and automatically converted into false positive (FP), false negatives (FN), and falsely classified (FC). This is particular useful for simulations that operate on the far ends of the probability tails, i.e., for events that do not occur very often. The Monte Carlo simulations are no longer constrained to the conceptual case set but are used on the actual cases and tools under consideration for the application. This step helps to understand the complex behavior of the various classifiers, establishes that the fusion tool works properly and aids in fine-tuning of the algorithm and parameters.

2.2 Performance Evaluation

During the conceptual phase of tool development, the sum-squared-error (SSE) may be used as a metric for the DoE stage. To confirm that the fusion tool leads to actual performance improvement in the second phase, we propose an overall performance index by weighing the individual components false positives (FP), false negatives (FN), and false classified (FC). The weightings of the individual components are driven by the application at hand. One choice for this index is shown in equation (1).

$$0.6*(1-FP)+0.3*(1-FN)+0.1*(1-FC) \quad (1)$$

The benchmark performance index can set to zero for a known benchmark performance (such as the best single classifier). An increase in performance is measured as the fraction of improvement from that baseline to perfect performance, expressed in percent.

2.2 Classifier Output Simulation

There are different ways to simulate the classifier output. Among the many, we considered in this study 1.) A one-sided Gaussian distribution; 2.) A bi-variate uniform distribution; 3.) A bi-variate Gaussian distribution..

2.2.1 One-sided Gaussian distribution

We simulated diagnostic tool output by assuming a Gaussian distribution of fault recognition. That is, if the reliability of the tool for a specific fault was listed as 0.8 in the diagonal entry of the confusion matrix, then 80% of the faults were simulated between confidence 0.5 and 1. The other 20% were simulated between 0 and 0.5. Specifically, for a given case, the associated z-score is computed. The z-score is a statistical measure relating to how many standard deviations away from the mean the particular value is located. It is computed by interpreting the associated entries in the confusion matrix as the area under the curve of a Gaussian distribution from which the z-score is obtained through iterative integration. Next, random values based on the z-score are assigned. Fig. 3 illustrates the example.

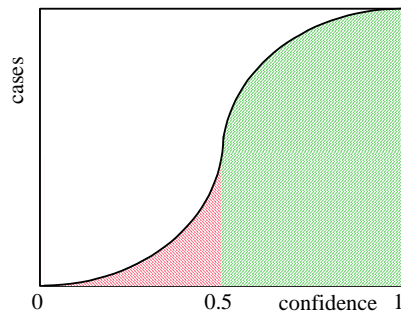


Figure 3: Simulation of diagnostic tool output

This distribution is somewhat idealized in the sense that it assumes that most of the cases have confidence values that lie closer to the extremes than at the decision intersection. On the other hand, there is a continuous transition from no fault decision to fault decision. In the results section below we will illustrate the performance of this simulation model in comparison with other models.

2.2.2 Bi-variate uniform distributions

This distribution assumes that the output of the classifiers is uniform across the confidence space. This distribution is depicted in Figure 4 below. While this model would in most cases not represent a real world situation, there are certain advantages such as the ease of modeling. In addition, this distribution might still be useful in providing a bound to fusion performance.

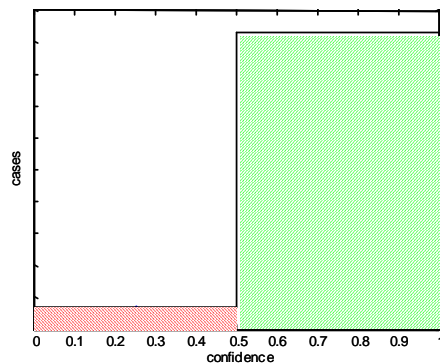


Figure 4: Bi-variate uniform confidence distribution

2.2.3 Bi-variate Gaussian distribution

This distribution does more closely represent typical classifier output. For example, the histogram in Figure 5 shows the output of a 2-class classifier that was trained to detect whether or not features represented corrosion in pipelines. Figure 5a shows the distribution for corrosion, Figure 5b represents no-corrosion, and Figure 5c shows the overlap of the two classes over the confidence space.

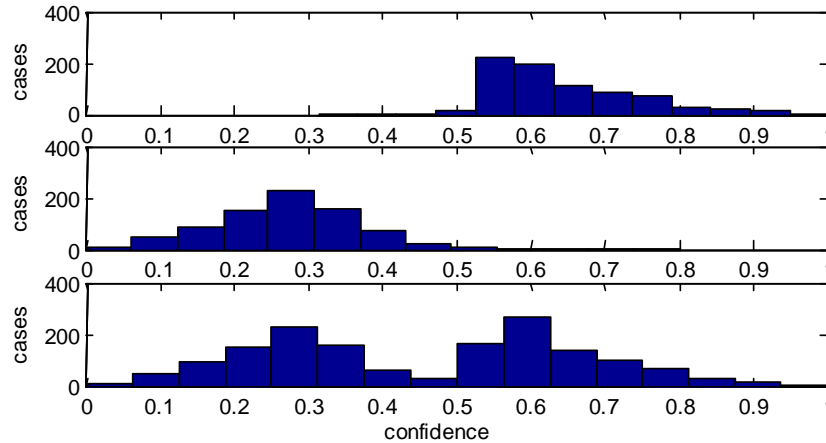


Figure 5: Histogram of 2-class classifier

Figure 6 shows the corresponding modeled 2-class Gaussian model for the classifier output. Although this model seems to reflect the output above to some degree, caution must be advised against generalizing from this observation. In the case below, the mean and standard deviation for both classes was known. Generally, this is not the case. Results may differ when the distributions are assumed to have a larger or smaller standard deviation (and different means) than in reality. Generally, we postulate that the larger the difference of the mean of the two classes, the easier it is for the classifier (and the fuser) to make a correct decision. The same holds true to some degree for a tighter standard deviations. As a bounding agent at one end of the spectrum, the bi-variate uniform distribution represents the case where the standard deviation is infinitely large. Figure 7 shows the output of the simulation which closely resembles the model.

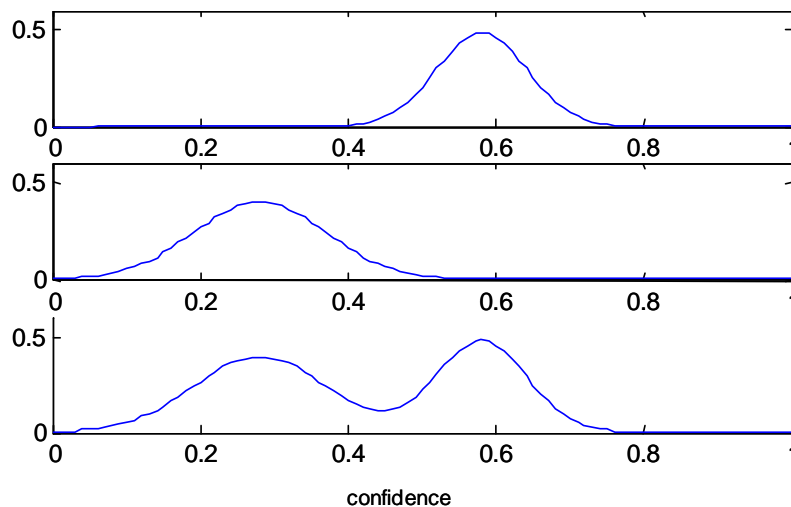


Figure 6: Model for 2-class classifier: 6(a): class 1 (corrosion);
6(b): class 0 (no corrosion);
6(c): overlap of both classes

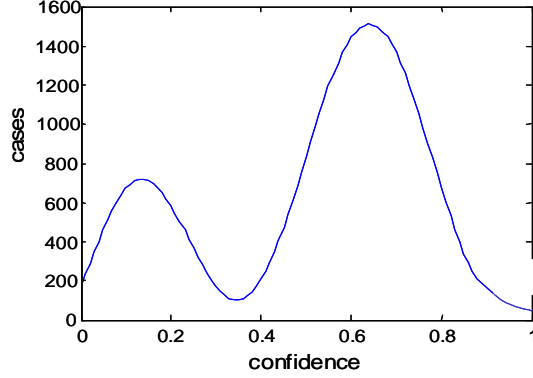


Figure 7: Simulation output: bi-variate Gaussian model

3. APPLICATION AND RESULTS

We simulated the distributions for a 7-class classifier suite that dealt with gas path faults for an aircraft engine¹⁰ and used a fusion engine that was developed specifically for that domain. We postulate that the observations for the classifier output model is applicable also to other fusion engines. The following three tables show the results for the three models compiled in confusion matrix format where $C_0 - C_6$ represent the actual classes and $\hat{C}_0 - \hat{C}_6$ represent the estimated classes.

– **Table 1: Confusion matrix for fusion engine using Gaussian distribution for classifier output**

	\hat{C}_0	\hat{C}_1	\hat{C}_2	\hat{C}_3	\hat{C}_4	\hat{C}_5	\hat{C}_6
C_0	0.9999	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001
C_1	0.0007	0.9991	0.0000	0.0000	0.0000	0.0001	0.0001
C_2	0.0001	0.0000	0.9999	0.0000	0.0000	0.0000	0.0000
C_3	0.0011	0.0000	0.0000	0.9985	0.0000	0.0004	0.0000
C_4	0.0056	0.0001	0.0001	0.0001	0.9922	0.0015	0.0004
C_5	0.0057	0.0000	0.0007	0.0002	0.0001	0.9922	0.0011
C_6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

– **Table 2: Confusion matrix for fusion engine using Uniform distribution for classifier output**

	\hat{C}_0	\hat{C}_1	\hat{C}_2	\hat{C}_3	\hat{C}_4	\hat{C}_5	\hat{C}_6
C_0	0.9906	0.0006	0.0002	0.0003	0.0001	0.0056	0.0026
C_1	0.0087	0.9813	0.0010	0.0006	0.0002	0.0065	0.0017
C_2	0.0062	0.0002	0.9891	0.0001	0.0000	0.0036	0.0008
C_3	0.0157	0.0001	0.0006	0.9655	0.0009	0.0140	0.0032
C_4	0.0242	0.0019	0.0021	0.0043	0.9110	0.0435	0.0130
C_5	0.0235	0.0032	0.0069	0.0038	0.0013	0.9435	0.0178
C_6	0.0009	0.0000	0.0000	0.0000	0.0000	0.0008	0.9983

– **Table 3: Confusion matrix for fusion engine using Bi-variate Gaussian distribution for classifier output**

	\hat{C}_0	\hat{C}_1	\hat{C}_2	\hat{C}_3	\hat{C}_4	\hat{C}_5	\hat{C}_6
C_0	0.9975	0.0000	0.0000	0.0000	0.0000	0.0023	0.0002
C_1	0.0000	0.9952	0.0000	0.0000	0.0000	0.0045	0.0003
C_2	0.0000	0.0000	0.9993	0.0000	0.0000	0.0007	0.0000
C_3	0.0002	0.0001	0.0000	0.9879	0.0000	0.0100	0.0018
C_4	0.0005	0.0011	0.0014	0.0005	0.9432	0.0430	0.0103
C_5	0.0028	0.0000	0.0027	0.0010	0.0003	0.9863	0.0069
C_6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.9999

Table 4 summarizes the results displaying the false positives (FP), false negatives (FN), false classified (FC), and the results of the performance index introduced in equation (1).

Table 4: Summary of Performance metrics for different classifier output models

	FP	FN	FC	index
One-sided Gaussian	0.0001	0.0019	0.0026	99.91%
Bi-variate uniform	0.0094	0.0113	0.0315	98.78%
Bi-variate Gaussian	0.0025	0.0035	0.0141	99.60%

As can be seen, the one-sided Gaussian distribution performs the best, while the bi-variate uniform distribution performs worst. The bi-variate Gaussian performs quite well but not as well as the one-sided Gaussian. The results of the performance index were bounded on average by $\pm 0.03\%$ within the 95% confidence interval.

4. CONCLUSIONS

While the overall fusion output achieves substantial gains with all models used, the resulting fused performance may differ to some degree depending on the classifier output model used for simulation. Whereas the variation shown here does not seem to be very large, it must be noted that such differentiation in the higher accuracy ranges can make a substantial difference with respect to either meeting or not meeting specific performance goals, in particular since any gain at this level is typically exponentially harder to achieve than gains at the lower performance level.

As expected, the bi-variate uniform classifier model performs worst while the one-sided Gaussian performs best. If possible, one should attempt to get an understanding of the output distribution (which is likely bi-variate Gaussian) and model it accordingly. Where this is not possible, the bi-variate uniform distribution gives a reasonable lower bound for the performance while the one-sided Gaussian tends to be somewhat too optimistic. One caveat with using the bi-variate model revolves around the possibly unknown spread and mean. Larger variations here may distort true performance to some degree.

Other classifier output models can of course be used such as the uni-variate Gaussian distribution. Such a distribution is expected to produce results that are even more conservative than the bi-variate uniform distribution since the decision threshold is in an area with poor partition properties.

Exhaustively simulating the entire design space may be a lengthy undertaking. Therefore, the simulation has to be guided to populate the relevant areas of the design space. Ideally, this guided simulation is methodically conducted. We have briefly touched on using a DoE driven approach by first establishing validated performance goals which are used for the design phase. Later, a larger scale simulation can explore the entire classifier output space.

ACKNOWLEDGMENTS

This research was in part supported by DARPA project MDA 972-98-3-0002.

REFERENCES

1. Yan, W. Goebel, K., and Li, J. "Classifier Performance Measures in Multi-Fault Diagnosis for Aircraft Engines", *Proceedings of SPIE, Component and Systems Diagnostics, Prognostics, and Health Management II*, vol. 4733, pp. 88-97, 2002.
2. Egan, J.P., *Signal detection theory and ROC analysis*, Series in Cognition and Perception, Academic Press, New York, 1975.
3. Downey, T.J., Meyer, D.J., Price, R.K., and Spitznagel, E.L., "Using the receiver operating characteristic to assess the performance of neural classifiers", *IJCNN '99 – International Joint Conference on Neural Networks*, Vol.5, pp3642-3646, .., 1999.
4. Bradley, A.P., "The use of the area under the ROC curve in the evaluation of machine learning algorithms", *Pattern Recognition*, **Vol.30**, No.7, pp1145-1159, 1997.
5. D. Hall and A. Garga, "Pitfalls in Data Fusion (and How to Avoid Them)", *Proceedings of the Second International Conference on Information Fusion (Fusion '99)*, pp. 429-436, 1999.
6. Goebel, K. and Yan, W., "Fusing Binary and Continuous Output of Multiple Classifiers", *Proceedings of the Fifth International Conference on Information Fusion, Fusion 2002*, vol. 1, pp. 380-387, 2002.
7. Fowlkes, W. and Creveling, C., *Engineering Methods for Robust Product Design*, Addison-Wesley Publishing Company, Reading, MA, 1995.
8. Fishman, G., *Monte Carlo : concepts, algorithms, and applications*. Springer-Verlag, New York, 1996.
9. Sobol, I., *A primer for the Monte Carlo method*. CRC Press, Boca Raton, FL, 1994.
10. Goebel, K. Krok, M., and Sutherland, H., "Diagnostic Information Fusion: Requirements Flowdown and Interface Issues", *Proceedings of IEEE Aerospace Conference*, p. 11.0303, 2000.
11. Goebel, K. "Architecture and Design of a Diagnostic Information Fusion Tool", *AIEDAM: Special Issue on AI in Equipment Service*, **Vol. 15 (4)**, pp. 335-348, Sept. 2001.